Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою

Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРПЗс.230134.01.06.ПЗ

Виконав студент ІІІ курсу, група ІПЗс-23-1 _____ Підпис Миколай ДРОЗДОВ Ім'я, ПРІЗВИЩЕ

Керівник асистент Науковий ступінь, вчене звання _____ Підпис В'ячеслав БОЙКО Ім'я, ПРІЗВИЩЕ

Нормоконтролер старший викладач Посада _____ Підпис Ганна БЕДРАТЮК Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення

_____ Підпис Леонід БЕДРАТЮК Ім'я, ПРІЗВИЩЕ

1 червня 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
_____ Л. П. Бедратюк
20 01 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дроздова Миколая Олексійовича
Прізвище, ім'я, по батькові студента

1. Тема роботи Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою

Керівник роботи Бойко В'ячеслав Олександрович, асистент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7
2. Строк подання студентом роботи на кафедру 01.06.2026 р.
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проєктування програмного забезпечення, програмна реалізація та тестування
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
1. Діаграма варіантів використання
2. Діаграма зв'язків модулів
3. Діаграма класів

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтр	Бедратюк Г. І., старший викл.	15.05.26	25.05.26
Антиплагіат	Форкун Ю. В., доцент	15.05.26	25.05.26

7. Дата видачі завдання « 20 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проектування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12–31.12.2025	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01–20.02.2026	
3 Проектування програмного забезпечення	21.02–20.03.2026	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03–30.04.2026	
5 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05–25.05.2026	
6 Попередній захист КвР	Травень	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05–30.05.2026	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2024	

Студент _____ Підпис _____ Миколай ДРОЗДОВ Ім'я, ПРІЗВИЩЕ

Керівник роботи _____ Підпис _____ В'ячеслав БОЙКО Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи «Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою».

Автор роботи: Дроздов Миколай Олексійович.

Керівник роботи: Бойко В'ячеслав Олександрович.

Пояснювальна записка: 77 с., 43 рис., 2 табл., 5 дод., 43 джерела.

Графічна частина: 3 креслення ф. А3.

ДОПОВНЕНА РЕАЛЬНІСТЬ, ІНЖЕНЕРНІ МЕРЕЖІ, ПЛАН ПОВЕРХУ, ANDROID, JAVA.

Мета кваліфікаційної роботи: розроблення застосунку для автоматизованого створення планів поверхів будівель з використанням сенсорних даних Android-пристрою.

У кваліфікаційній роботі проведено аналіз предметної області та її інформаційного забезпечення, визначено функціональні та нефункціональні вимоги до застосунку, розроблено загальну архітектуру застосунку, спроєктовано сховище даних планів та структура застосунку.

Для реалізації програмного продукту використано мову програмування Java, мову розмітки XML, фреймворк доповненої реальності Google ARCore, та середовище розробки Android Studio. За допомогою цих засобів розроблено застосунок для автоматизованого створення планів поверхів будівель з використанням сенсорних даних Android-пристрою.

Практичне значення результатів роботи полягає в тому, що впровадження розробленого застосунку дозволяє автоматизувати створення планів поверхів будівель, включно з документуванням існуючих або місць встановлення запланованих інженерних мереж, що значно полегшить процес збирання даних про будівлю до початку планування та виконання робіт спеціалістам з встановлення та обслуговування різноманітних інженерних мереж.

25.05.26
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.230134.01.06.ПЗ	Пояснювальна записка	77		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А3	КвРІПЗ.230134.01.06.Е8	UML-діаграма варіантів	1		
			використання			
5	А3	КвРІПЗ.230134.01.06.Е8	Діаграма зв'язків	1		
			модулів			
6	А3	КвРІПЗ.230134.01.06.Е8	UML-діаграма класів	1		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗс.230134.01.06.ВД		
Розроб.		Дроздов М. О.		27.05.2	Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою Відомість документів	Літ.	Арк. 1 / Аркушів 1
Перевір.		Бойко В. О.		27.05.2			
Реценз.						ХНУ. ІПЗс-23-1	
Н. Контр.		Бедратюк Г.І.		27.05.2			
Затверд.		Бедратюк Л.П.		27.05.2			

ЗМІСТ

					КВРІПЗс.230134.01.06.ПЗ	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Дроздов М. О.		24.05.2	Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою Пояснювальна записка	Літ. / Арк. 4 / Аркушів 77
Перевір.		Бойко В. О.		27.05.2		
Реценз.						ХНУ, ІПЗс-23-1
Н. Контр.		Бедратюк Г.І.		27.05.2		
Затверд.		Бедратюк Л.П.		27.05.2		

ПЕРЕЛІК СКОРОЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability
API – Application Programming Interface
APK – Android Package Kit
AR – Augmented Reality
CAD – Computer-Aided Design
CI/CD – Continuous Integration / Continuous Deployment
CRUD – Create, Read, Update, Delete
DAP – Debug Adapter Protocol
DXF – Drawing Exchange Format
FPS – Frames Per Second
GPU – Graphics Processing Unit
HUD – Head-Up Display
IDE – Integrated Development Environment
IFC – Industry Foundation Classes
JSON – JavaScript Object Notation
KMP – Kotlin Multiplatform
LiDAR – Light Detection and Ranging
LSP – Language Server Protocol
MAU – Monthly Active Users
MVC – Model-View-Controller
MVI – Model-View-Intent
MVP – Model-View-Presenter
MVVM – Model-View-ViewModel
OBJ – Object File
ORM – Object-Relational Mapping
PDF – Portable Document Format
PNG – Portable Network Graphics
SDK – Software Development Kit
SOC – System on a Chip
SQL – Structured Query Language

UI – User Interface
UX – User Experience
VIO – Visual-Inertial Odometry
VPS – Visual Positioning System
XML – Extensible Markup Language
СКБД – Система керування базами даних
SLAM – Simultaneous Localization And Mapping

ВСТУП

Створення засобів автоматизованого створення планів будівель є однією з найбільш динамічних та перспективних галузей сучасного розроблення програмного забезпечення. З кожним роком цей напрям лише посилюється, що зумовлено дедалі більшими потребами у точному моделюванні довколишнього середовища для будівництва, архітектури та керування експлуатацією споруд. Тенденції розвитку галузі включають постійне вдосконалення алгоритмів комп’ютерного зору, впровадження технологій доповненої реальності, а також перехід від спеціалізованого дорогого устаткування до портативних рішень. З урахуванням цих тенденцій, розроблення мобільних засобів сканування простору стає не лише вагомим інженерним завданням, але й ключовим напрямком, який відкриває безліч можливостей для оптимізації процесів проєктування та обслуговування будівель.

Автоматизоване створення планів будівель полягає у обробці сенсорних даних пристрою для визначення його положення та орієнтації в простоті, виявлення структурних елементів будівлі, таких як стіни, перекриття, дверні та віконні отвори та/або встановлення користувачем точок за допомогою систем доповненої реальності, що вказують на положення цих елементів. Наступним етапом є обробка цих даних, для перетворення координат та розмірів цих елементів у план будівлі. Системи, що виконують ці завдання, зазвичай є однокористувацькими.

Особливо актуальним є створення мобільного застосунку для побудови планів поверхів будівель у контексті потреби точного документування інженерних мереж. Цей процес привертає увагу фахівців своєю здатністю створювати цифрові копії нерухомості, обов'язкові для проведення робіт із модернізації систем зв'язку, електропостачання, водопроводу та опалення. Останні досягнення в галузі обробки сенсорних даних відкривають нові можливості для створення надійних засобів, де камера мобільного пристрою виступає основним сенсором для збору просторової інформації. Такі програмні рішення стають дієвим інструментом для швидкого розрахунку витрат матеріалів та часу, вимагаючи від розробників поєднання

глибоких технічних знань алгоритмів просторового відстеження із забезпеченням високої зручності користувацького інтерфейсу та суворого захисту конфіденційності даних без передачі їх через мережу Інтернет.

Важливим чинником розробки цього проєкту є відсутність у сучасних рішеннях цієї галузі засобів, необхідних для документування та планування розміщення інженерних мереж.

Важливим чинником є зменшення порогу входу у використання засобів просторового сканування для індивідуальних спеціалістів та невеликих підприємств. Сучасні технології роблять процес обмірювання приміщень більш доступним та швидким. Застосування пристроїв на базі операційної системи Android та платформ просторового обчислення, таких як Google ARCore, дозволяє відмовитися від лазерних засобів вимірювання. Водночас використання перевірених підходів до проєктування, зокрема шаблону відокремлення даних від представлення (MVC) та закритої тришарової архітектури, забезпечує високу швидкодію та надійність системи під час роботи з тривимірною графікою.

Саме тому розроблення мобільного застосунку для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою є надзвичайно доцільним. Якісне проєктування та програмна реалізація подібного проєкту дозволяє поглибити знання з об'єктно-орієнтованого програмування, роботи з просторовими даними, проєктування інтерфейсів та локального збереження інформації. У підсумку це свідчить про високий рівень підготовки фахівця та може слугувати основою для створення повноцінного програмного продукту, який надаватиме інженерам і будівельникам зручний та автономний інструмент для автоматизації їхньої повсякденної діяльності.

Метою кваліфікаційної роботи є розроблення мобільного застосунку для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою, що функціонує за допомогою встановлення користувачем точок через доповнену реальність та відповідає сучасним вимогам до продуктивності й інформаційної безпеки, а також містить функціонал, необхідний для документування інженерних мереж на цих планах.

Для успішного досягнення мети роботи необхідно розв'язати такі завдання проєктування:

– провести змістовий аналіз предметної області автоматизації формування плану приміщень та наявного програмно-технічного забезпечення;

– сформувати перелік функціональних та нефункціональних вимог до розроблюваної системи;

– здійснити вибір архітектурних підходів, шаблонів проєктування та методів зберігання даних;

– виконати загальну та модульну декомпозицію системи з детальним описом залежностей та інтерфейсів модулів;

– спроєктувати структуру бази даних та макети інтерфейсу користувача;

– проаналізувати та обрати оптимальні технології і засоби реалізації застосунку;

– здійснити програмну реалізацію підсистем просторового відстеження, керування записами та двовимірної візуалізації;

– виконати тестування розробленого застосунку;

– проаналізувати отримані результати та сформулювати висновки.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовий аналіз предметної області, її структурних та функціональних особливостей

Дослідження предметної області спрямоване на виявлення наявних суперечностей та технологічних прогалин у процесах автоматизації збору, опрацювання та передачі просторових даних. На основі проведеного аналізу формується обґрунтування для розробки програмного забезпечення, що розв'язує окреслені проблеми.

Сучасний рівень розвитку обчислювальної техніки перетворює мобільні пристрої на дієві інструменти для вимірювання та аналізу довколишнього середовища. Побудова планів споруд за допомогою камери смартфона є комплексним завданням, що поєднує методи комп’ютерного зору та архітектурного моделювання. Цей процес полягає у перетворенні двовимірних зображень у векторизовані схеми або об’ємні моделі.

Предметна область ґрунтується на розгляді будівлі як цілісної системи взаємопов’язаних конструкцій. Ключовими об’єктами дослідження є огороджувальні конструкції: стіни, перекриття підлоги та стелі, а також функціональні прорізи – вікна та двері.

План поверху – це схематичне зображення горизонтального розрізу будівлі, умовно проведеного на рівні віконних та дверних отворів (зазвичай на висоті одного метра від підлоги). На такій схемі відображають елементи, що потрапили у площину розрізу, а також об’єкти, розташовані нижче цього рівня.

Автоматизоване створення планів будівель полягає у обробці сенсорних даних пристрою для визначення його положення та орієнтації в простоті, виявлення структурних елементів будівлі, таких як стіни, перекриття, дверні та віконні отвори та/або встановлення користувачем точок за допомогою систем

доповненої реальності, що вказують на положення цих елементів. Наступним етапом є обробка цих даних, для перетворення координат та розмірів цих елементів у план будівлі. Системи, що виконують ці завдання, зазвичай є однокористувацькими.

Типовий процес взаємодії користувача з системами цієї галузі полягає у, переміщенні користувачем пристрою по будівлі, використанні інтерфейсу для визначення положення та параметрів елементів будівлі, подання команди про завершення сканування, та формування системою дво- чи тривимірного плану будівлі.

Потреба в автоматизації цього процесу зумовлена необхідністю створення цифрових копій нерухомості для ринку житла, розробки внутрішнього оздоблення приміщень та керування експлуатацією споруд [7]. Окрім первинного будівництва, плани необхідні для подальшого обслуговування об'єктів.

Зокрема, наявність точних планів є обов’язковою умовою для проведення робіт із модернізації таких інженерних мереж:

– мереж передачі даних та зв’язку;
– електричних мереж та систем освітлення;
– мереж водопостачання та водовідведення;
– газорозподільних систем;
– обладнання для опалення та кондиціонування повітря.

Відсутність актуальної документації унеможливлює якісне проектування, точне оцінювання витрат матеріалів та розрахунок часу на виконання робіт. Тому створення доступних засобів для швидкої побудови планів є важливою науково-технічною задачею.

Прикладом практичного використання такої системи може бути встановлення комп’ютерних мереж у call-центрі, адже для цього необхідно визначити розміри приміщення, де розташовується мережа, конкретні довжини кабелів, необхідні для створення з’єднань, а також точки розташування маршрутизаторів, що розгалужують мережу.

Класичним підходом до цього завдання буде проведення ручних вимірів приміщень, створення плану, та використання його для планування розташування елементів мережі, та розрахунку їх характеристик.

Сучасні засоби автоматизованого створення планів дозволять швидко створити план будівлі, проте, при їх застосуванні, все ще необхідні великі об’єми ручної роботи для розташування на цьому плані елементів мережі, та розрахунку їх характеристик.

Розроблювана система забезпечить автоматизацію як створення плану будівлі, так і планування розташування мережі, та автоматично розрахує деякі з її характеристик, що зробить розроблювану систему найкращим доступним рішенням для цієї проблеми.

Застосування пристроїв на базі операційної системи Android дозволяє відмовитися від використання дорогого спеціалізованого устаткування [8], наприклад лазерних засобів вимірювання відстані. Це забезпечує можливість масового впровадження технології та спрощує процес отримання точних даних про архітектурне середовище.

Сучасні рішення в цій галузі використовують різноманітні сенсори сучасних мобільних пристроїв та алгоритми обробки даних. Найпростіші рішення в цій галузі вдаються до методів SLAM з використанням даних камери що дозволяють користувачу створювати віртуальні точки з прив’язкою до фізичного простору, та потім перетворювати ці точки у план будівлі. Складніші рішення вдаються до аналізу даних time-of-flight камер та LiDAR систем та використання засобів машинного навчання для автоматичного виявлення елементів будівлі.

Вхідними даними в предметній області є дані з сенсорів пристрою, такі як зображення чи відео з камери, а також команди користувача, наприклад про встановлення точки, що визначає кут кімнати.

Вихідними даними є готовий дво- чи тривимірний план будівлі.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Сучасний ринок програмного забезпечення для побудови планів поверхів класифікується за технологічним базисом та цільовим призначенням. Виділяють три основні групи засобів:

– системи на основі доповненої реальності, що фіксують геометрію приміщення через камеру;

– застосунки для точного креслення з підтримкою зовнішніх вимірювальних пристроїв;

– рішення на основі штучного інтелекту, що обробляють відеопотік або панорамні зображення у хмарних сховищах.

Для розуміння необхідних функцій розроблюваної системи, їх якості, та деталей їх роботи, необхідно провести аналіз наявного ПЗ предметної області, з фокусом на ті застосунки, що є популярними.

Застосунок «ARRuler», деякі екрани якого зображені на рисунку 1.1. Даний програмний продукт функціонує як цифровий вимірювальний інструмент [9]. Він використовує технологію візуального виміру відстаней для визначення лінійних розмірів та об’ємів об’єктів.

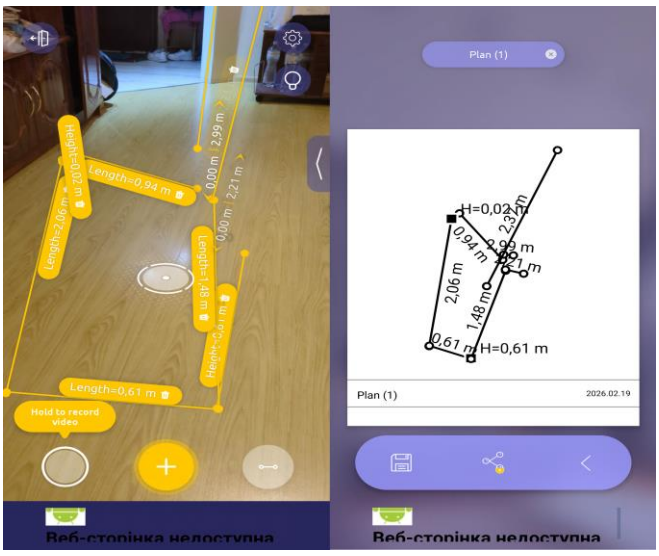


Рисунок 1.1 – Екрани «ARuler»

Переваги:

– наявність системи керування проєктами та ієрархічної структури папок;

– можливість побудови схем через відеокамеру та на основі статичних світлин;

– функція визначення висоти приміщення та кутів нахилу поверхонь;

– функція запису фото та відео в доповненій реальності.

Недоліки:

– застосунок часто здійснює автоматичне перемикання між режимами вимірювання вертикальних та горизонтальних величин залежно від кута огляду, а через відсутність можливості примусової фіксації обраної площини це призводить до незручностей та помилок вимірювання;

– алгоритм створення відрізків вимагає повторного визначення координат кожної точки, що суперечить логіці обходу приміщення, де початкова координата нового відрізка зазвичай відповідає кінцевій точці попереднього;

– не забезпечує належної перевірки взаємного розташування елементів фізичного та віртуального просторів, що дозволяє створювати відрізки, що проходять крізь стіни або інші перешкоди;

– орієнтований лише на створення двовимірних планів;

– відсутність різних типів відрізків не дозволяє документувати різні структурні одиниці будівлі та елементи інженерних мереж;

– містить рекламні матеріали, що відволікають від виконання завдань.

Застосунок «ARPlan» деякі екрани якого зображені на рисунку 1.2. Орієнтований на створення цілісних моделей будівель.

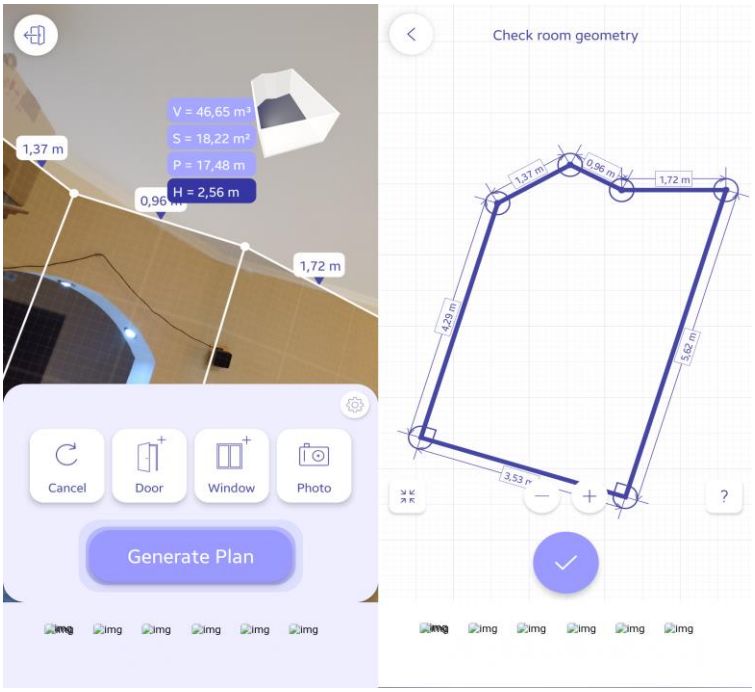


Рисунок 1.2 – Екрани «ARPlan»

Переваги:

– наявність системи керування проєктами та ієрархічної структури папок;

– можливість побудови схем через відеокамеру;

– можливість ручного створення та редагування планів;

– функція визначення висоти приміщення;

– генерація тривимірних моделей приміщень.

Недоліки:

– негнучка система видалення елементів (лише у зворотній послідовності);

– не забезпечує належної перевірки взаємного розташування елементів фізичного та віртуального просторів, що дозволяє створювати відрізки, що проходять крізь стіни або інші перешкоди;

– низька стійкість відстеження положення у просторі, що призводить до зміщення вже виміряних координат, виправлення яких є неможливим;

– складність об’єднання окремих кімнат у загальний план поверху;

– обмежений набір вбудованих типів відрізків не дозволяє документувати елементи інженерних мереж;

– містить рекламні матеріали, що відволікають від виконання завдань.

Застосунок «ARMeter», головний екран якого зображений на рисунку 1.3. Розроблений для швидких обмірів без складних обчислювальних витрат. Основна увага приділена автономній роботі без залучення зовнішніх серверів.

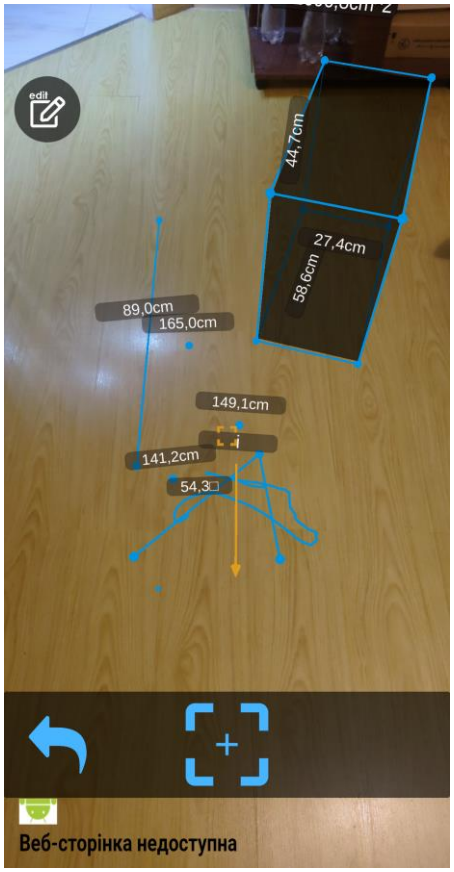


Рисунок 1.3 – Екран «ARMeter»

Переваги:

– мінімалістичний інтерфейс та базовий набір геометричних вимірів;

– можливість вимірювання через відеокамеру;

– можливість додавання текстових коментарів до об’єктів;

– функція знімання фото в доповненій реальності.

Недоліки:

– алгоритм створення відрізків вимагає повторного визначення координат кожної точки, що суперечить логіці обходу приміщення, де початкова координата нового відрізка зазвичай відповідає кінцевій точці попереднього;

– неможливість проведення вимірювань по площині стелі;

– не забезпечує належної перевірки взаємного розташування елементів фізичного та віртуального просторів, що дозволяє створювати відрізки, що проходять крізь стіни або інші перешкоди;

– відсутність функцій для повноцінної побудови планів (лише окремі виміри);

– відсутність різних типів відрізків не дозволяє документувати різні структурні одиниці будівлі та елементи інженерних мереж;

– містить рекламні матеріали, що відволікають від виконання завдань.

Застосунок «Floor Plan Creator».

Переваги:

– інтеграція з галузевими стандартами, зокрема з «Xactimate» та «CoreLogic»;

– широкий набір форматів експорту, що включає дво- та тривимірні плани: PDF, DXF, OBJ та IFC;

– підтримка багатоповерхових конструкцій;

– широкий набір інструментів для створення приміщень, їх з’єднань, та інженерного обладнання;

– функцію друку плану.

Недоліки:

– відсутність різних типів відрізків не дозволяє документувати елементи інженерних мереж;

– висока вартість ліцензії для повного доступу до функцій експорту.

Застосунок «AR Measure Plan», екран вимірів якого зображено на рисунку 1.4. Продукт базується на поєднанні методів одночасної локалізації та картографування з алгоритмами глибокого навчання для аналізу об’єктів [15].

Рисунок 1.4 – Екран «AR Measure Plan»

Переваги:

– можливість побудови схем через відеокамеру;

– функція зняття фото в доповненій реальності;

– забезпечує високу точність прив’язки віртуальних точок до фізичних перешкод;

– автоматизація розпізнавання типових елементів інтер’єру.

Недоліки:

– критичні помилки у програмному коді, що призводять до втрати даних при збереженні;

– відсутність різних типів відрізків не дозволяє документувати різні структурні одиниці будівлі та елементи інженерних мереж;

– найвищий рівень вартості передплати серед аналогів.

Застосунок «Planner 5D», деякі екрани якого зображені на рисунку 1.5. Є комплексним середовище для моделювання, що охоплює не лише внутрішні приміщення, а й прилеглі території [16].

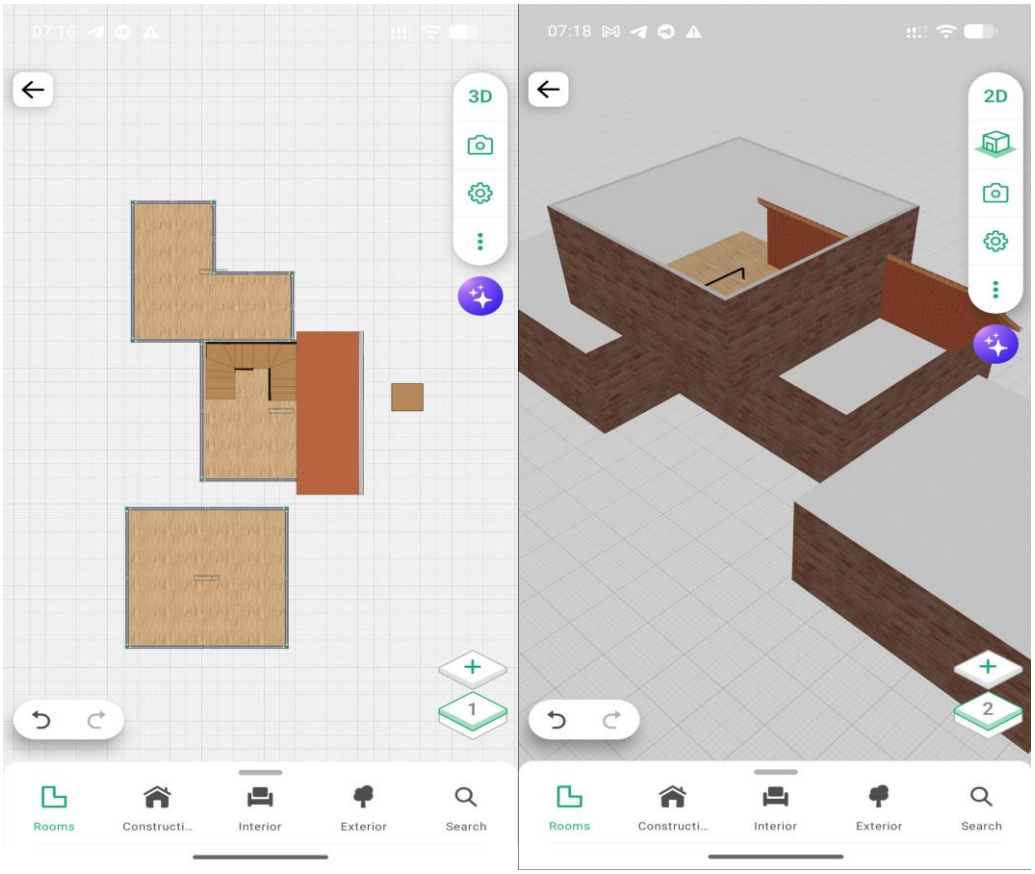


Рисунок 1.5 – Екрани «Planner 5D»

Переваги:

– набір форматів експорту включає дво- та тривимірні плани;

– підтримка багатоповерхових конструкцій;

– можливість детального планування ландшафту та дахів;

– велика база моделей меблів.

Недоліки:

– відсутність різних типів відрізків не дозволяє документувати елементи інженерних мереж;

– необхідність постійної авторизації та створення облікового запису;

– висока вартість ліцензії для повного доступу до розташування меблів.

Застосунок «Measure Tools», деякі екрани якого зображено на рисунку 1.6.

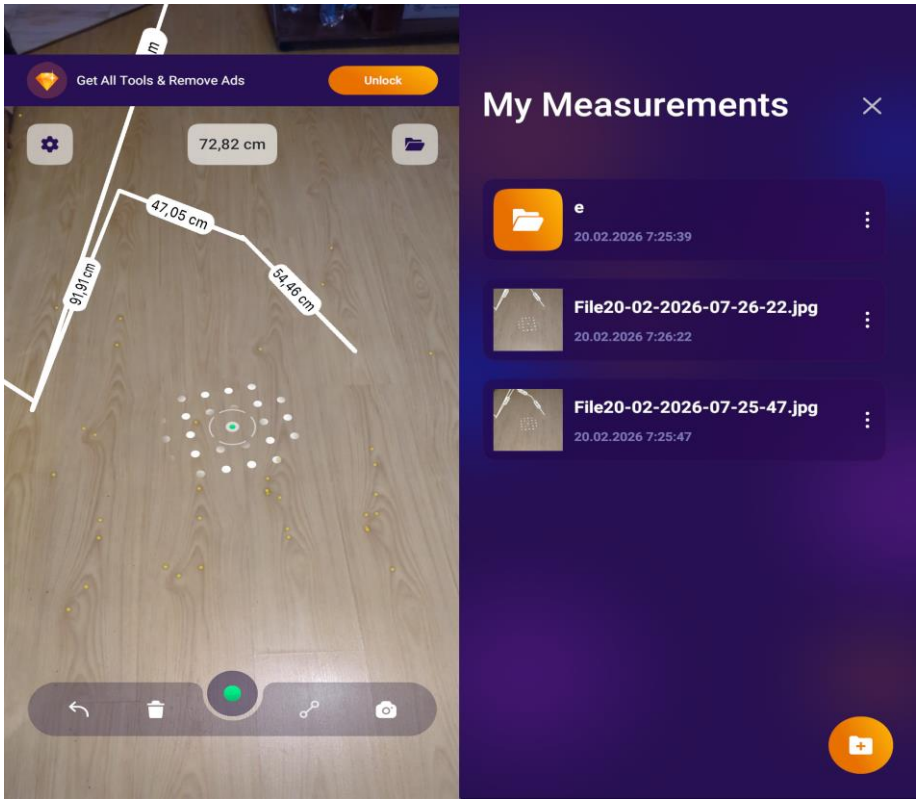


Рисунок 1.6 – Екрани «Measure Tools»

Переваги:

– можливість вимірів через відеокамеру;

– функція знімання фото в доповненій реальності;

– забезпечує високу точність прив’язки віртуальних точок до фізичних перешкод.

Недоліки:

– відсутність різних типів відрізків не дозволяє документувати різні структурні одиниці будівлі та елементи інженерних мереж;

– відсутність функцій для повноцінної побудови планів (лише окремі виміри);

– негнучка система видалення елементів (лише у зворотній послідовності).

Застосунок «CubiCasa». Реалізує методику керованого відеосканування. Користувач здійснює безперервну зйомку, після чого дані опрацьовуються на віддалених серверах.

Переваги:

– мінімальні трудовитрати під час польових робіт;

– висока якість фінальної візуалізації.

Недоліки:

– відсутність миттєвого результату (обробка триває до 24 годин);

– повна залежність від мережі Інтернет та платна модель використання кожного сканування.

Порівняння всіх застосунків подано у таблиці 1.

Таблиця 1 – Порівняння застосунків

Назва застосунку	Формат планів	AR / Камера	Наявність реклами / Платних послуг	Ключові переваги	Головні недоліки
ARRuler	Тільки 2D	Так	Реклама	Вимірювання висоти/кутів, схеми за фото, відео/фото в AR, керування проєктами.	Автоперемикання площин (помилки), перетин стін, незручна логіка відрізків.
ARPlan	2D та 3D	Так	Реклама	Генерація 3D-моделей, ручне редагування, керування проєктами.	Зміщення координат у просторі, перетин стін, видалення лише з кінця, важко об'єднувати кімнати.
ARMeter	Лише окремі виміри	Так	Реклама	Повністю автономний, мінімалістичний, підтримка текстових коментарів.	Немає виміру стелі, немає повноцінних планів, перетин стін, незручна логіка відрізків.
Floor Plan Creator	2D та 3D	Ні	Платний експорт	Галузеві стандарти, багатоповерховість, експорт (PDF, DXF, OBJ, IFC), друк.	Висока вартість ліцензії для повного доступу до експорту.
AR Measure Plan	2D	Так	Необхідна дорога платна підписка для абсолютно всіх функцій	Висока точність прив’язки, розпізнавання типових елементів інтер'єру (AI).	Критичні помилки в коді (втрата даних при збереженні).

Продовження таблиці 1.

Назва застосунку	**Формат планів**	**AR / Камера**	**Наявність реклами / Платних послуг**	**Ключові переваги**	**Головні недоліки**
Planner 5D	2D та 3D	Ні	Платні меблі	Багатоповерховість, детальне планування дахів/ландшафту, велика база меблів.	Обов’язкова авторизація, висока вартість доступу до меблів.
Measure Tools	Лише окремі виміри	Так	Ні	Висока точність прив’язки до перешкод, фото в AR.	негнучке видалення елементів (лише з кінця).
CubiCasa	Фінальна 2D/3D візуалізація	Відеосканування	Плата за кожне сканування	Мінімальні зусилля під час зйомки об'єкта, висока якість фінального результату.	Залежність від Інтернету, очікування результату до 24 годин.

Поточні рішення предметної області не містять інструментів для документування інженерних мереж, чи планування їх встановлення, отже необхідне створення нового застосунку, що реалізує цей функціонал.

1.3 Визначення функціональних та нефункціональних вимог до програмного забезпечення

Наявність на ринку численних програмних рішень із різним рівнем якості та обсягом можливостей зумовлює недоцільність повного копіювання їхніх функцій. З огляду на обмеженість часових та технічних ресурсів, розробка зосереджена на заповненні вільної ніші, яка має стабільний попит.

Аналіз предметної області та наявних засобів підтвердив необхідність створення інструментарію для документації інженерних мереж та планування робіт на їхній основі. На відміну від типових рішень для архітектурного моделювання, у цій роботі висота приміщення та параметри віконних отворів не є визначальними, тому їх вимірювання не передбачене.

Функціональні вимоги:

– керування проєктами, що включає :створення плану поверху, відкриття плану поверху, видалення плану поверху, та зміна назви плану поверху;

– керування користувацькими типами відрізків, що включає їх створення, перейменування, та видалення;

– керування користувацькими типами відрізків у плані, що складається з їх включення в та виключення з плану;

– сканування плану поверху за допомогою камери, що складається з: вибору типу відрізка, створення точки обраного типу відрізка, обрання точки як «пов’язаної» (з якою автоматично створиться лінія при створенні нової точки), скидання вибору «пов’язаної» точки, створення точки з’єднання відрізків користувацьких типів, перетворення точки з’єднання відрізків користувацьких типів у точку відрізку користувацького типу, перетворення точки відрізків користувацьких типів у точку з’єднання відрізків користувацьких типів, видалення точки, з’єднання точки з «пов’язаною», та роз’єднання точки з «пов’язаною»;

– формування звіту про відрізки користувацьких типів;

– експорт плану поверху у вигляді PNG зображення.

Детальний опис варіантів використання подано у додатку Б.

Вимоги до захисту даних: застосунок не повинен автоматично передавати будь-які дані через мережу інтернет.

Вимоги до портативності: застосунок повинен бути здатен виконувати всі функції без доступу до мережі інтернет.

Вимоги до зручності використання: інтерфейс має бути пристосований до роботи в умовах низького освітлення та не містити рекламних вставок. Процес створення точок базується на визначенні місця перетину променя, що виходить із центру об’єктива, з поверхнями об’єктів. На екрані має відображатися

графічний покажчик цієї точки та відстань від неї до «пов’язаної» точки, якщо вона обрана.

Вимоги но інтерфейсу користувача: інтерфейс має бути адаптований до використання в основному правою рукою, що, одночасно, тримає пристрій, для забезпечення комфортної роботи в темних приміщеннях основні кольори інтерфейсу мають бути темними.

Вимоги до використання ресурсів: обсяг оперативної пам’яті, яку споживає програма під час сканування або відображення планів обсягом до 50 точок, не має перевищувати 1.5 ГБ.

Вимоги до відображення даних: Основними одиницями вимірювання довжини є метри. Під час візуалізації схеми встановлюються такі параметри:

– точність вимірів: до двох знаків після коми;

– характеристики тексту: розмір – 20 точок, розташування – паралельно до лінії на відстані 10 точок від її центру;

– орієнтація напису: від -90 до 90 градусів відносно горизонталі.

Масштаб плану обирається автоматично з урахуванням технічних можливостей платформи Android та обмежень у межах від 100 до 3000 точок на один метр. Пріоритет надається чіткості відображення тексту без накладання на інші графічні елементи.

Вимоги до продуктивності: система повинна забезпечувати в середньому не менше 10 кадрів за секунду при скануванні плану, що містить не більше 50 точок.

Вимоги до швидкодії: система повинна відобразити план, що містить не більше 50 точок не більше ніж за 1 секунду. Система повинна виконати експорт плану, що містить не більше 50 точок не більш ніж за 1 секунду.

Опис відображення елементів плану на екрані сканування:

– точки відображаються у вигляді сфер;

– лінії між точками відображаються у вигляді циліндрів

– колір точки стіни – білий;

– колір звичайної точки інженерної мережі, якщо поточний тип відрізків не збігається – синій;

– колір звичайної точки інженерної мережі, якщо поточний тип відрізків збігається – зелений;

– колір лінії стіни – білий;

– колір лінії інженерної мережі, якщо поточний тип відрізків не збігається – синій;

– колір лінії інженерної мережі, якщо поточний тип відрізків збігається – зелений;

– колір точки з'єднання інженерних мереж – жовтий;

– відображення «пов'язаної» точки – блимання пурпуровим кольором;

– відображення точки, з якою виконуватимуться взаємодії – блимання червоним кольором;

Опис відображення елементів плану на екрані плану:

– точки не відображаються;

– лінії між точками відображаються у вигляді прямих ліній

– колір лінії стіни – білий;

– колір лінії інженерної мережі – синій;

– над центром кожного відрізка відображається його довжина з точністю до 1 см.

Система є однокористувацькою.

Діаграму варіантів використання, що демонструє їх зв'язки між собою, та з акторами представлено у графічних матеріалах до цієї роботи на аркуші 1.

1.4 Висновки. Постановка задачі

У цьому розділі розглянуто галузь створення креслень поверхів споруд за допомогою переносних пристроїв, а також вивчено шляхи автоматизованого створення таких креслень на основі сенсорних даних Android-пристроїв.

Внаслідок вивчення зазначеної галузі встановлено, що утворення креслень споруд є критично важливим етапом не лише під час задуму, а й у ході робіт по обслуговуванню та модернізації будівель. Особливо важливим є завдання точного фіксування розміщення інженерних мереж. Наявні традиційні методи вимірювань є трудомісткими та часто призводять до помилок, що обґрунтовує необхідність впровадження мобільних систем на основі комп'ютерного зору.

Було проаналізовано основні процеси і етапи створення мобільних застосунків для створення планів поверхів будівель та будівель в цілому, яку задачу вони вирішують і яка мета в створені ще одного застосунку. Для визначення вимог до застосунку було проаналізовано сфери застосування подібних застосунків. З метою виокремити необхідні функції застосунку, а також найти застосунку в насиченому ринку було розглянуто найпопулярніших представників на платформі Google Play Market.

Під час роботи було вивчено ключові процеси та стадії розроблення мобільних застосунків для створення планів поверхів, визначено їхні основні завдання та, фактом того, що наявне ПЗ предметної області не реалізує функцій, необхідних для документування чи планування встановлення інженерних мереж, обґрунтовано доцільність створення нового програмного продукту. Для формування вимог до системи досліджено потенційні сфери застосування таких рішень. З метою визначення необхідних функцій застосунку та пошуку вільної ніші на насиченому ринку проведено огляд найбільш поширених програмних засобів, представлених на платформі Google Play Market.

Аналіз предметної області дозволив сформувати перелік функціональних та нефункціональних вимог до застосунку, зокрема до методів сканування будівель та способів візуалізації планів. На основі вичерпної технічної

специфікації, сформованої під час підготовки розділу, необхідно розробити застосунок для автоматизованого створення планів поверхів будівель з використанням сенсорних даних Android-пристрою. Для досягнення цієї цілі було поставлено набір задач.

Перелік поставлених задач:

– проведення аналізу наявних архітектурних підходів і шаблонів проєктування, які зазвичай використовуються у подібних системах, та визначення на їх основі структури майбутнього застосунку;

– проведення проєктування системи, включаючи її декомпозицію, та встановлення зв’язків між компонентами та опис інтерфейсів модулів;

– розроблення макетного представлення інтерфейсу користувача;

– проведення аналізу та вибір технологій для реалізації розробленої архітектури;

– програмна розробка основних модулів системи;

– підготовка візуальних ресурсів для режиму сканування;

– створення інтерфейсу користувача;

– визначення методології та виконання тестування;

– формулювання висновків на основі результатів виконаної кваліфікаційної роботи.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір типу архітектури та шаблонів проєктування

В першу чергу необхідно розглянути, які шаблони проєктування використовуються для мобільних застосунків платформи Android загалом, адже значна частина розроблюваної системи відповідальна просто за обробку та відображення даних, що є функціоналом звичайних Android-застосунків. Нижче наведено опис декількох поширених шаблонів проєктування, які використовуються в Android-застосунках.

Класичним підходом до Android-застосунків є шаблонів проєктування Model-View-Controller, що пропонує зрозуміле розділення на модель, що оперує даними, представлення, що відображає інтерфейс, та контролер, який керує взаємодією [19].

Основними перевагами шаблону MVC є простота його реалізації та мінімальний об’єм шаблонного коду.

Основними недоліками MVC є висока зв’язаність, обмежена тестованість, та ручне управління станом системи.

Шаблон MVP виник як еволюційне рішення для подолання недоліків MVC, пропонуючи ізоляцію логіки представлення від Android-фреймворку [18]. У цій моделі Presenter виступає посередником, який отримує вхідні дані від View (Activity або Fragment), взаємодіє з Model і повертає вже оброблені дані назад у View через чітко визначений інтерфейс.

Основною перевагою MVP є висока тестованість.

Основним недоліком MVP є висока кількість шаблонного коду.

Впровадження архітектурних компонентів Android Jetpack ознаменувало перехід до MVVM як до рекомендованого галузевого стандарту. На відміну від MVP, ViewModel не має прямого посилання на View. Натомість вона надає потоки даних, на які підписується представлення [20]. Такий підхід дозволяє реалізувати відношення "один до багатьох", де одна ViewModel може обслуговувати декілька

View, що особливо корисно при створенні складних UI-потоків на планшетах або складних пристроях.

Переваги MVVM включають покращення життєвого циклу об’єктів. ViewModel переживає зміну конфігурації пристрою, що автоматично вирішує проблему втрати даних при повороті екрана без додаткових зусиль з боку розробника. Впровадження таких інструментів, як LiveData та пізніше StateFlow у Kotlin Coroutines, зробило оновлення інтерфейсу безшовним та реактивним. Проте розробники часто стикаються з проблемою фрагментації стану: коли у ViewModel існує десять різних потоків для одного екрана, стає важко передбачити, в якому саме стані перебуває інтерфейс користувача в конкретний момент часу.

Основними перевагами шаблону MVVM є висока тестованість, низька зв’язність, та реактивне керування станом.

Основними недоліками MVVM є складність реалізації та кількість шаблонного коду.

MVI є найбільш сучасною ітерацією шаблонів проєктування, що базується на принципах циклічного односпрямованого потоку даних та незмінності стану. Натхненний архітектурою Redux, MVI пропонує розглядати будь-яку дію користувача як намір, який обробляється централізованим обробником для створення абсолютно нового об’єкта стану [20].

Основними перевагами шаблону MVI є висока тестованість, низька зв’язність, та атомарне керування станом.

Основними недоліками MVI є значна складність реалізації та висока кількість шаблонного коду.

Враховуючи необхідність використання доповненої реальності для виконання деяких функцій застосунку, необхідно розглянути придатність шаблонів проектування до реалізації доповненої реальності.

Доповнена реальність створює додаткові вимоги до архітектури застосунку, а саме: глибока інтеграція рендерингових циклів та передачі

сенсорних даних. Доповнена реальність також значно збільшує вимоги до швидкодії та пропускної здатності системи.

Іншим важливим аспектом доповненої реальності є значне зростання складності автоматизованого тестування порівняно з звичайними застосунками.

В контексті доповненої реальності, модель MVI перестає бути привабливою, адже використовувана в ній модель незмінного стану значно збільшує час відгуку та значно зменшує пропускну здатність системи.

Зважаючи на обмежені ресурси в розробленні цього проєкту, короткі терміни його розробки, та досвід команди з цим шаблоном, для цього проєкту обрано шаблон проектування MVC.

Важливим для цього проєкту є також вибір архітектури, адже вона є визначальним фактором складності розробки та тестування системи, та навіть теоретичної можливості реалізації деяких вимог.

Монолітна архітектура представляє собою традиційний підхід до побудови програмного забезпечення, де всі функціональні компоненти застосунку – від інтерфейсу користувача та бізнес-логіки до методів доступу до даних та інтеграційних модулів – інтегровані в єдину, неподільну кодову базу та виконуються як один цілісний процес у єдиному середовищі [21].

Існує два варіанти монолітної архітектури:

– тісно пов’язана монолітна архітектура – у такій системі компоненти мають глибокі залежності один від одного, що значно ускладнює фазу підтримки життєвого циклу ПЗ [22];

– модульна монолітна архітектура – застосунок розділяється на логічні модулі за бізнес-доменами, проте ці модулі розгортаються разом, та виконуються як частини одного процесу [23].

Перевагами монолітної архітектури є спрощена розробка, легкість розгортання, та простота інтеграційного тестування

Основним недоліком монолітної архітектури є значна складність підтримки та модернізації системи.

Багатошарова архітектура, організовує програмну систему у вигляді горизонтальних шарів, де кожен шар має чітко визначену відповідальність та роль у загальному процесі обробки даних [24]. Ця модель виникла як перший підхід до реалізації принципу розподілу обов'язків у ПЗ. У багатошаровій системі компоненти групуються за технічною спорідненістю.

Існує два підходи до багатошарової архітектури:

– закрита архітектура – шар може викликати методи лише з наступного шару нижче за нього, це полегшує тестування але ускладнює оптимізацію;

– відкрита архітектура – шари можуть звертатися до всіх шарів під ними, що може дозволити оптимізацію простих операцій, де проміжні шари не змінюють дані, але збільшує зв'язність системи та ускладнює підтримку [25].

Зважаючи на попереднє рішення про використання шаблону проектування MVC, прийнято рішення використовувати поєднання модульної монолітної архітектури, та закритої тришарової архітектури. Рівнями цієї архітектури будуть:

– тонкий рівень представлень – формує та подає фінальне представлення даних користувачу, передає команди користувача наступному рівню;

– товстий рівень контролерів – перетворює дані між форматами придатними до відображення та до зберігання, обробляє команди користувача;

– рівень даних – містить логіку зберігання та модифікації даних.

2.2 Проектування бази даних

Для проектування системи необхідне розуміння структури даних, що вона обробляє та зберігає, а також методів їх зберігання. Для досягнення цього розуміння необхідно спроектувати базу даних системи.

Об'єктами предметної області, необхідними для реалізації проєкту є:

– план поверху будівлі;

– користувацький тип відрізку;

– точка;

– відрізок.

Єдиною властивістю плану поверху будівлі є його назва.

Єдиною властивістю користувацького типу відрізка є його назва.

Властивостями точки є:

– координати x, y, та z;

– тип точки (користувацький тип відрізка або стіна або точка з’єднання відрізків користувацьких типів).

Єдиною властивістю відрізка є його тип, що окрім користувацького може бути стіною.

Зв’язки в системі наступні:

– план поверху будівлі має від 0 до безлічі користувацький типів відрізків;

– користувацький тип відрізку має від 0 до безлічі планів поверхів будівель;

– план поверху будівлі має від 0 до безлічі точок;

– точка має 1 план поверху будівлі;

– точка має від 0 до 1 користувацького типу відрізка;

– користувацький тип відрізка має від 0 до безлічі точок;

– план поверху будівлі має від 0 до безлічі відрізків;

– відрізок має 1 план поверху будівлі;

– відрізок має 2 точки;

– точка має від 1 до безлічі відрізків;

– відрізок має від 0 до 1 користувацького типу відрізка;

– користувацький тип відрізка має від 0 до безлічі відрізків.

Результати аналізу між цими об’єктами подано на діаграмі сутність-зв’язок, зображеної на рисунку А.1.

Враховуючи те, що дві найчастіші операції з цими даними будуть введення/виведення плану, включаючи всі точки та відрізки, а також виведення всіх користувацьких типів відрізків, включених у план, за умови застосування нереляційної бази даних, доцільними будуть наступні оптимізації:

– включення всіх точок та відрізків у об’єкт плану, якому вони належать;

– включення списку ідентифікаторів користувацьких типів у план поверху будівлі.

2.3 Вибір системи керування базами даних

Для будь-якого проєкту, що зберігає дані, вибір СКБД є надзвичайно важливим. Враховуючи те, що дані цього проєкту можна зберігати як у реляційних так і нереляційних базах даних, необхідно розглянути обидва варіанти, та робити вибір враховуючи переваги як підходу так і конкретної СКБД.

Room – це бібліотека абстракції над SQLite, розроблена Google як частина екосистеми Android Jetpack. Вона покликана спростити роботу розробника, надаючи високорівневий API, де взаємодія з базою даних відбувається через анотації та об’єкти доступу до даних. Room дозволяє використовувати всю потужність SQL, але при цьому забезпечує зручність об’єктно-орієнтованого підходу, автоматично перетворюючи результати запитів у об'єкти Kotlin або Java.

Основними превагами Room є перевірка схем та запитів під час компіляції, безперешкодна робота з LiveData, ViewModel та Paging Library, значне зменшення кількості шаблонного коду, підтримка KmP, та автоматизація змін структури БД.

Основними недоліками Room є затримка компіляції проєкту через використання генераторів коду, обмеження в роботі зі складними БД, та уповільнення в записі великих масивів даних в порівнянні з прямим доступом до SQLite.

Realm – це об'єктно-орієнтована NoSQL база даних, яка використовує власне ядро зберігання на C++ замість SQLite. Її ключовою особливістю є архітектура «нульового копіювання», де дані не десеріалізуються в пам’ять, а відображаються безпосередньо з диска за допомогою вказівників. Це робить Realm надзвичайно швидким при читанні великих масивів об'єктів.

Основними перевагами Realm є значне збільшення швидкості читання, робота з об’єктами, кросплатформність, автоматичні оновлення об’єктів у БД.

Основними недоліками Realm є неможливість передавати об’єкти між потоками, обмеження на структуру об’єктів що зберігаються, складність забезпечення цілісності даних через автоматичні оновлення БД, а також великий розмір файлу БД.

Realm значно ускладнює забезпечення цілісності даних, тому не є правильним вибором для цього проєкту.

З розглянутих баз даних та бібліотек доступу до них єдиним прийнятним кандидатом є Room, проте реляційна архітектура SQLite, на якій побудований Room, та ідентифіковані раніше неефективності при реалізації цього проєкту з реляційною БД, вимагають пошуку додаткових альтернатив.

Враховуючи вбудовані в Java засоби серіалізації та десеріалізації об’єктів, побудова власного сховища об’єктів не є складним завданням.

Враховуючи попередні рішення про включення точок та відрізків до планів поверхів, зберігати необхідно лише два типи об’єктів.

З метою забезпечення ефективності операцій над сховищем даних, для кожного об’єкту буде створено окремий файл. Це допустимо, оскільки очікувана кількість об’єктів в системі не перевищує кількох сотень.

Для забезпечення кожному з цих об’єктів унікального ідентифікатора, та зберігання списків всіх об’єктів, необхідне також сховище ідентифікаторів. Це сховище буде реалізоване як ще один об’єкт, що серіалізується та десеріалізується у окремий файл. Це рішення є допустимим через рідкісність

операцій створення та видалення об’єктів у створюваній системі, та незначну кількість об’єктів.

2.4 Опис декомпозиції

Модульна декомпозиція полягає у поділі програмного комплексу на автономні функціональні блоки з чітко окресленими зонами відповідальності. Такий підхід дозволяє мінімізувати взаємозалежність компонентів, забезпечуючи високу масштабованість та надійність мобільного застосунку для побудови планів поверхів та інженерних комунікацій.

Процес безпосереднього збору геометричної інформації середовища покладено на модуль просторового сканування та реєстрації даних. Цей компонент тісно взаємодіє з підсистемою доповненої реальності, здійснює захоплення просторових точок, обчислює позиції якорів.

Фундаментом для збереження цілісності бази даних застосунку виступає модуль сховища ідентифікаторів. Він діє як централізований реєстр, що відстежує унікальні ключі всіх створених об'єктів, забезпечуючи коректне зберігання даних у файловій системі та унеможливлюючи виникнення колізій при генерації нових ідентифікаторів. На базі цього механізму функціонує модуль адміністрування глобального реєстру інженерних мереж, який дозволяє створювати, редагувати та видаляти універсальні типи комунікацій, що будуть доступні системі на наскрізному рівні для всіх проєктів.

Для організації робочого процесу впроваджено модуль управління проєктами планів поверхів. Він містить логіку життєвого циклу планів та маршрутизації користувача до інтерфейсів конкретного об'єкта. У межах кожного окремого плану простору діє модуль управління інженерними мережами конкретного плану, що відповідає за локальну конфігурацію,а також агрегує дані про довжину сполучень інженерних мереж та виконує оптимізацію і об'єднання суміжних сегментів мережі для точного підрахунку матеріалів.

Центральним агрегатором зібраної просторової топології є модуль планів поверхів будівель. Він об'єднує всі геометричні параметри та метадані об'єкта, керує процесами серіалізації структури у пам'ять пристрою. Деталізація цієї топології забезпечується через модуль просторових точок плану, який фіксує точні координати вузлів та їхній просторовий стан, та модуль відрізків плану поверху, що встановлює логічні зв'язки між точками, формуючи контури стін та траєкторії прокладання мереж.

Гнучкість та розширюваність опису різноманітних ліній зв'язку досягається завдяки модулю типів інженерних мереж. Цей блок чітко розмежовує властивості стін, звичайних точок прокладання інженерних мереж та вузлів з'єднання, визначаючи суворі правила взаємодії між ними.

У результаті проведеного аналізу було визначено наступний перелік автономних модулів проєкту:

– модуль просторового сканування та реєстрації даних;
– модуль адміністрування глобального реєстру інженерних мереж;
– модуль управління проєктами планів поверхів;
– модуль управління інженерними мережами конкретного плану;
– модуль типів інженерних мереж;
– модуль планів поверхів будівель;
– модуль просторових точок плану;
– модуль відрізків плану поверху;
– модуль сховища ідентифікаторів.

За результатами аналізу було створено діаграму зв’язків модулів, яку подано у графічних матеріалах роботи на аркуші 2.

2.5 Опис залежностей

Проєктування архітектури мобільного застосунку для створення планів поверхів будівель вимагає детального аналізу ієрархії залежностей між його

функціональними компонентами. Систематизація цих зв’язків дозволяє визначити вектори передачі інформації, забезпечити цілісність даних при серіалізації та оптимізувати взаємодію між модулями доповненої реальності та внутрішньою логікою представлення об'єктів. Опис структури залежностей виступає фундаментом для забезпечення високої надійності системи, особливо в контексті обробки сенсорних даних у реальному часі та їх трансформації у топологічні моделі інженерних мереж.

Початкова стадія декомпозиції системи дозволяє ідентифікувати складну мережу взаємодій, де центральним вузлом виступає об’єкт плану поверху. Модуль контейнеризації плану безпосередньо взаємодіє з базовими сутностями точок та ліній, координуючи їхній стан та забезпечуючи логічну цілісність при зміні специфікацій комунікацій. Водночас модуль адміністрування ідентифікаторів функціонує як незалежний сервіс реєстрації, що забезпечує глобальний доступ до реєстрів проєктів для модулів навігації та керування списками.

Особливе місце в ієрархії посідає модуль просторового сканування та реєстрації даних, який використовує камеру пристрою як основний сенсор для збору просторової інформації та побудови карти оточення. Він оперує отриманими координатами для формування екземплярів точок і ліній та ініціює запити до модуля типів інженерних мереж для динамічного призначення класифікаційних властивостей створюваним об’єктам. Процеси конфігурації логіки розмежовані: модуль адміністрування глобального реєстру інженерних мереж керує загальним переліком доступних комунікацій, тоді як модуль управління інженерними мережами конкретного плану здійснює селективне прив'язування цих глобальних типів до локального екземпляра плану.

Систематизований опис структурних зв’язків та ієрархії компонентів системи представлений у формі переліку ключових міжмодульних залежностей:

– Модуль сканування звертається до модуля типів інженерних мереж для ідентифікації об'єктів та передає згенеровані топологічні дані до модуля планів поверхів будівель для їх фіксації;

– модуль керування інженерними мережами взаємодіє з модулем типів інженерних мереж для виконання операцій модифікації, модулем сховища ідентифікаторів для реєстрації нових сутностей, а також модулем планів поверхів будівель для каскадного оновлення або видалення типів ліній з існуючих креслень;

– модуль управління планами поверхів ініціює виклики до модуля планів поверхів будівель для створення або завантаження об'єктів та синхронізується з модулем сховища ідентифікаторів для валідації індексів;

– модуль управління інженерними мережами конкретного плану утворює зв'язок між модулем планів поверхів будівель та глобальним модулем типів інженерних мереж;

– модуль типів інженерних мереж виступає базовою інформаційною сутністю, що надає дані для серіалізації та класифікації з’єднань інженерних мереж у системі;

– модуль планів поверхів будівель виступає центральним ядром, що включає залежності від модуля просторових точок плану та модуля відрізків плану поверху, а також зберігає масив ідентифікаторів підключених інженерних мереж;

– модуль просторових точок плану та модуль відрізків плану поверху формують основу для розрахунку метричних характеристик;

– модуль сховища ідентифікаторів діє як рівень персистентності, до якого звертаються інші керуючі модулі для уникнення колізій при створенні нових планів та типів комунікацій.

На основі результатів аналізу була створена діаграма міжмодульних зв’язків зображена на рисунку А.2, що відображає зв’язки цих модулів.

Сформована структура залежностей демонструє чітке розділення між рівнем отримання сирих просторових даних та рівнем їхньої логічної обробки. Це створює необхідні передумови для подальшого розширення функціональності застосунку, зокрема впровадження нових типів інженерних об’єктів без порушення стабільності основної архітектури.

2.6 Опис інтерфейсу модулів

Програмні інтерфейси компонентів мобільного застосунку для побудови планів поверхів мають забезпечувати стандартизований доступ до геопросторових даних та параметрів інженерних мереж, суворо дотримуючись принципів інкапсуляції та модульності.

Модулі сканування, адміністрування глобального реєстру інженерних мереж, адміністрування планів, інженерних мереж планів, а також модуль екрана конкретного плану, не викликається напряму іншими модулями.

Предметна область описується моделями даних із чітко визначеними інтерфейсами доступу. Модуль типів інженерних мереж (CableType) та модуль планів поверхів будівель (FloorPlan) приховують свої внутрішні стани, надаючи безпечний доступ через гетери та сетери, а також забезпечують персистентність через методи Save(), Load() та Delete(). Інтерфейси модуля просторових точок плану (FloorPlanPoint, IPointType) та модуля відрізків (FloorPlanLine, ILineType) гарантують поліморфну обробку топології: сімейство методів isWallPoint(), isCableConnectionPoint(), isCable() та їх аналоги дозволяють класифікувати геометричні примітиви під час рендерингу графа приміщення. Нарешті, модуль сховища ідентифікаторів (IdStorage) виконує роль глобального індексу, керуючи розподілом унікальних ідентифікаторів бази даних через методи addFloorPlanId(), addCableTypeId() та функції їх вилучення.

Узагальнені специфікації визначених інтерфейсів модулів проєкту наведено в таблиці 2.

Таблиця 2 – Модулі та їх інтерфейси

Модуль	Інтерфейс
CameraActivity	
IdStorage	getFloorPlanIds(), getCableTypeIds(), addFloorPlanId(), removeFloorPlanId(), addCableTypeId(), removeCableTypeId(), Save(), Load()
CableTypeListActivity, CreateCableTypeActivity, EditCableTypeActivity	
FloorPlanListActivity, FloorPlanActivity, CreateActivity	
FloorPlanCableTypeListActivity	
CableType	getId(), getName(), setName(), Save(), Load(), Delete()
FloorPlan	captureSet(), getName(), setName(), setCapture(), addCableTypeId(), removeCableTypeId(), getPoints(), getLines(), Save(), Load(), Delete()
FloorPlanPoint, IPointType	getX(), getY(), getZ(), setX(), setY(), setZ(), getPointType(), isWallPoint(), isCableConnectionPoint(), isCablePoint()
FloorPlanLine, ILineType	getPoint1Index(), getPoint2Index(), getLineType(), isWall(), isCable(), getCableTypeId()

Систематизація зазначених зв’язків створює необхідні передумови для реалізації алгоритмів автоматизованої обробки геометрії та переходу до етапу детального проектування логіки візуалізації планів. Глибоке розуміння міжмодульної комунікації дозволяє забезпечити стабільну роботу застосунку в умовах динамічної зміни стану сесії доповненої реальності.

2.7 Проектування інтерфейсу користувача

На цьому етапі необхідно розглянути, які функції повинен підтримувати інтерфейс, визначитись з екранами, які потрібно розробити та створити відповідні макети, на яких буде зображено загальну структуру інтерфейсу.

Аналізуючи функціональні вимоги до застосунку можна виділити дві основні групи екранів, які будуть потрібні:

– звичайні екрани, що дадуть можливість створювати типи інженерних мереж та плани, перейменовувати та видаляти їх, керувати інженерними мережами у планах та отримувати звіт про них, переглядати самі плани та експортувати їх у PNG файли;

– HUD (Head-Up Display) екран, що буде виводитись поверх зображень з камери, доповнених об’єктами доповненої реальності на екрані доповненої реальності, та буде надавати інформацію про поточний стан системи, та дозволяти встановленням точок та ліній для побудови плану.

Звичайні екрани:

– головний екран (кнопка створення плану, кнопка переходу до списку типів інженерних мереж, список планів, що містить назви планів, що використовується для їх відкриття, та кнопки видалення);

– екран списку типів інженерних мереж (кнопка створення інженерної мережі, список типів інженерних мереж, що містить назви інженерних мереж, кнопки редагування, та кнопки видалення);

– екран створення плану (кнопка завершення створення, поле введення назви);

– екран створення типу інженерної мережі (кнопка завершення створення, поле введення назви);

– екран перейменування типу інженерної мережі (поле введення назви);

– екран плану (поле введення назви, ділянка відображення плану, кнопка переходу на екран сканування, кнопка переходу на екран типів інженерних мереж плану, кнопка експорту плану);

– екран типів інженерних мереж плану (кнопка створення типу інженерної мережі, список типів, що складається з check-box-ів вибору типу, назв типів, поля виведення довжини сегментів, та поля виведення загальної довжини).

HUD екрану сканування буде складатись з таких елементів:

– кнопка створення звичайної точки;

– кнопка прив’язки до точки;

– кнопка створення зв’язку;

– кнопка видалення зв’язку;

– кнопка створення точки з’єднання;

– кнопка перетворення точки;

– кнопка видалення точки;

– меню вибору типу відрізка;

– поле виведення відстані;

– приціл, що вказує точний центр екрану.

Після визначення структуру елементів екранів інтерфейсу користувача, необхідно створити макети екранів, щоб розуміти і мати можливість проаналізувати, як ці елементи будуть розміщуватись відносно екрану і один одного, що дасть змогу швидко реалізувати їх в подальшому.

Аналізуючи те, як користувачі зазвичай взаємодіють з мобільними пристроями, було визначено, яке розміщення елементів основного ігрового екрану ефективне.

Макет головного екрану зображено на рисунку А.2. Список планів займає увесь екран, назва плану розташовується з ліва, кнопка видалення – з права, статичні кнопки розташовані у правій нижній частині екрану.

Макет екрану списку типів інженерних мереж зображено на рисунку А.3. Список типів інженерних мереж займає увесь екран, з ліва на право розташовуються назва типу, кнопки редагування та видалення, кнопка створення типу розташована у правій нижній частині екрану.

Макет екрану створення плану зображено на рисунку А.4. Екран містить поле введення назви у верхній частині, та кнопку завершення у правій нижній частині.

Макет екрану створення типу інженерної мережі зображено на рисунку А.5. Екран містить поле введення назви у верхній частині, та кнопку завершення у правій нижній частині.

Макет екрану перейменування типу інженерної мережі зображено на рисунку А.6. Екран містить поле введення назви у верхній частині.

Макет екрану плану зображено на рисунку А.7. Екран складається з поля введення назви вгорі, кнопок, що розташовані у нижній частині стовпця з права, решту екрану займає площа виведення плану.

Макет екрану типів інженерних мереж плану зображено на рисунку А.8. Екран містить кнопку створення типу у правій нижній частині, та список типів, який складається з рядка що містить з ліва на право: check-box, назву, загальну довжина комунікацій, нижня частина елемента виділена під виведення довжин окремих сегментів.

Макет HUD екрану сканування зображено на рисунку А.9. Цей екран містить приціл рівно в центрі, та поле виведення відстані одразу під ним, всі кнопки крім кнопки видалення розташовані у правій нижній частині екрану, а кнопка видалення – у лівій нижній; меню вибору типу відрізка розташоване в нижній частині екрану між кнопками.

Після реалізації всіх макетів інтерфейсу, стає можливим створення відповідних елементи інтерфейсу під час реалізації застосунку.

2.8 Детальне проєктування модулів

Визначення основних модулів було проведено в попередньому розділі, було визначено самі модулі, їх функціонал та розглянуто в загальному їх зв’язки. Першим етапом детально проєктування модулів стане поділ модулів на окремі класи та інтерфейси, повний опис їх зв’язків та властивостей.

Модуль просторового сканування та реєстрації даних є ключовим компонентом підсистеми взаємодії з доповненою реальністю. Він реалізований насамперед за допомогою класу CameraActivity, який відповідає за обробку даних із сенсорів, просторове відстеження та рендеринг віртуальних об'єктів. До цього ж модуля належать допоміжні структури даних WrappedAnchor та AnchorLink, які використовуються для збереження просторових прив'язок та топологічних зв'язків між ними безпосередньо під час активної сесії сканування приміщення.

Модуль адміністрування глобального реєстру інженерних мереж відповідає за загальне управління доступними типами комунікацій на рівні всього застосунку. Цей модуль включає класи CableTypeListActivity, CreateCableTypeActivity та EditCableTypeActivity, які реалізують інтерфейс користувача та бізнес-логіку для перегляду, створення, редагування та видалення записів із глобального довідника типів інженерних мереж.

Модуль управління проєктами планів поверхів призначений для керування життєвим циклом проєктів користувача та координації відповідних візуальних інтерфейсів. Цей блок об'єднує клас FloorPlanListActivity, що забезпечує відображення переліку наявних планів у пам'яті пристрою, клас CreateActivity, який відповідає за ініціалізацію нових проєктів, та клас FloorPlanActivity, що слугує основним середовищем для перегляду, двовимірного масштабування та графічного представлення результатів сканування конкретного поверху.

Модуль управління інженерними мережами конкретного плану інкапсульований у класі FloorPlanCableTypeListActivity. Основним завданням

цього компонента є забезпечення логіки прив'язки визначених типів інженерних мереж із глобального реєстру до поточного активного плану поверху, а також автоматизоване обчислення сумарної довжини використаних комунікацій (відрізків сполучень інженерних мереж) певного типу за допомогою допоміжної внутрішньої структури CableSegment.

Модуль типів інженерних мереж представляє частину предметної області, що описує семантичні характеристики лінійних об'єктів. До складу цього модуля входять базова сутність CableType, що зберігає ідентифікатор та іменування типу мережі. Для розмежування фізичної суті ліній застосовується поліморфізм через інтерфейс ILineType, який має дві конкретні реалізації: CableLineType для представлення інженерних мереж та WallLineType для опису базової геометрії стін.

Модуль планів поверхів будівель представлений ядром предметної області системи – класом FloorPlan. Цей об'єкт виконує роль основного агрегатора просторових даних, пов'язаних із конкретним приміщенням. Він містить колекції точок, ліній та вибраних ідентифікаторів типів мереж, керує станом завершеності сканування, а також реалізує методи серіалізації для персистентного зберігання даних у файловій системі Android.

Модуль просторових точок плану охоплює сутності, необхідні для представлення вузлів збереженої геометрії. Він реалізований через клас FloorPlanPoint, який зберігає локальні тривимірні координати, та ієрархію маркерів типу точки, що базується на інтерфейсі IPointType. До цієї ієрархії входять класи WallPointType, CablePointType і CableConnectionPointType, які визначають логічну роль кожного просторового вузла у топології приміщення.

Модуль відрізків плану поверху структурно представлений класом FloorPlanLine. Цей клас забезпечує встановлення логічних та геометричних зв'язків між просторовими точками шляхом збереження індексів початкового та кінцевого вузлів у масиві точок плану, а також агрегує посилання на тип відповідної лінії (стіна чи конкретний тип інженерної мережі).

Модуль сховища ідентифікаторів реалізує системний механізм глобального керування унікальними ключами та представлений класом IdStorage. Цей компонент діє як реєстр виділених ідентифікаторів, забезпечуючи збереження, генерацію та коректний розподіл числових ключів для нових планів поверхів і типів інженерних мереж, що гарантує відсутність колізій даних між різними сесіями використання програмного продукту.

На основі виконаної деталізації модулів було створено діаграму класів системи, яку подано у графічних матеріалах до цієї роботи на аркуші 3.

2.9 Аналіз та вибір технологій та методів реалізації застосунку

Створення застосунку з елементами доповненої реальності є комплексним процесом, що вимагає детального планування та обґрунтованого вибору технологій і методів. У цьому розділі буде проведено аналіз доступних технологій та методів, які можна використовувати для створення застосунку для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою. Основна мета цього розділу – забезпечити технічну основу, яка дозволить створити високоякісний застосунок, відповідний усім вимогам та очікуванням. Аналіз врахує сучасні тенденції в розробці мобільних застосунків, досвід успішних проєктів, а також визначені вимоги і цільову аудиторію.

Першим необхідно обрати фреймворк для реалізації інтерфейсу користувача.

Jetpack Compose є сучасним декларативним інструментарієм для розробки нативних інтерфейсів Android, що базується на мові Kotlin. Архітектура фреймворку реалізує концепцію інтелектуальної рекомпозиції, при якій оновлюються виключно ті сегменти дерева інтерфейсу, стан яких зазнав змін.1 На відміну від імперативних методів, Compose інтегрує інтерфейсну логіку безпосередньо в програмний код, усуваючи необхідність десеріалізації XML-файлів під час виконання.

Перевагами Jetpack Compose є скорочення вихідного коду інтерфейсу на 50%, підвищення швидкості виконання коду інтерфейсу на 30%, та відсутність витрат на створення екранів з XML-документів.

Основними недоліками Jetpack Compose є довгий процес компіляції та великий розмір APK файлу.

Традиційна імперативна модель розробки інтерфейсів Android, яка базується на дескриптивному описі структури екрана у XML-файлах та програмній реалізації логіки на мовах Kotlin або Java. Архітектура системи побудована на ієрархії об'єктів View та ViewGroup, де кожен елемент інтерфейсу є важковаговим об'єктом, що вимагає ручного управління станом.

Основними перевагам цього підходу є висока стабільність, наявність глибокої та обширної бази знань, та доступ до тисяч сторонніх бібліотек.

Недоліками цього підходу є значний обсяг шаблонного коду, та складність синхронізації стану системи з інтерфейсом.

Зважаючи на попередній досвід розробника з цим підходом, його надійність, та доступну базу знань, обрано традиційну імперативну модель розробки інтерфейсів Android.

Необхідним для цього проєкту також є фреймворк для доповненої реальності.

Google ARCore – це фундаментальна платформа від Google для створення AR-досвіду на Android, яка забезпечує сприйняття фізичного світу через камеру та датчики. Технологія базується на візуально-інерційній одометрії (VIO), що поєднує дані з камери з показниками акселерометра та гіроскопа для відстеження пристрою в шістьох ступенях свободи. Даний підхід дозволяє мобільним пристроям розуміти свою позицію в просторі та стабільно закріплювати віртуальні об'єкти з високою точністю.

Основними перевагами ARCore є найширше серед усіх альтернатив охоплення ринку Android пристроїв, підтримка визначення глибини для

реалістичної оклюзії та фізики на 88% активних пристроїв, безкоштовний доступ до SDK, та великий об’єм документації.

Недоліками ARCore є дрейф трекінгу на на пристроях середнього та бюджетного сегментів, та залежність від Google Play Services for AR, що унеможливлює його використання на пристроях без сервісів Google

Unity – це провідний кросплатформний двигун, який використовує інструментарій AR Foundation для абстрагування нативних можливостей ARCore та ARKit. Технологія функціонує на базі архітектури XR Plug-in Management, що дозволяє розробникам використовувати єдиний C# API для розробки на Android та iOS. Такий підхід радикально прискорює ітерацію та дозволяє використовувати потужні системи рендерингу, такі як Universal Render Pipeline (URP).

Перевагами Unity є найбільша в світі бібліотека готових AR-ресурсів та плагінів, можливість тестувати AR сцени прямо в редакторі з використанням Unity MARS або AR Foundation Remote, велика спільнота розробників, та великий об’єм доступних навчальних матеріалів.

Недоліками Unity є складність налаштування рендерингу для специфічних мобільних пристроїв, та залежність від пакетів AR Foundation, зо можуть відставати від ARCore

Використання Unity не бажане через складність інтеграції з нативним інтерфейсом, тож прийнято рішення використати Google ARCore.

Важливим для розробки будь-якого складного ПЗ є інтегроване середовище розробки, тож необхідно провести аналіз доступних IDE, та обрати один з них.

Android Studio – це офіційне середовище розробки від Google, побудоване на базі IntelliJ IDEA Community Edition та спеціально оптимізоване для екосистеми Android.

Перевагами Android Studio є глибока інтеграція з Android розробкою, та повністю безкоштовна ліцензія для всіх розробників.

Недоліками Android Studio є тривалий час підготовки до роботи з великими проєктами, довгий час компіляції проєктів, складність налаштування через велику кількість специфічних параметрів, та обмежена підтримка технологій, що не стосуються Android або JVM-мов.

Visual Studio Code (VS Code) – це легковаговий та розширюваний редактор коду від Microsoft, що базується на архітектурі Electron та використовує протоколи LSP (Language Server Protocol) та DAP (Debug Adapter Protocol).

Основною перевагою VS Code є можливість працювати з будь-якими мовами та платформаvи завдяки протоколам LSP та DAP.

Недоліками VS Code є відсутність вбудований інструментів для роботи з інтерфейсами користувача Android, відсутність інструментів для профілювання роботи Android-застосунку, складна конфігурація для роботи з Android-проєктами, необхідність ручного керування залежностями Android SDK та налаштуваннями емуляторів.

Зважаючи на глибоку інтеграцію з найновішими Android API та офіційними інструментами Google а також досвід розробника з цим IDE, обрано Android Stuio.

Необхідно також обрати мову програмування, оскільки, навіть з врахуванням обраних інструментів та фреймворків, вибір обмежений двома мовами: Kotlin та Java.

Java – це об'єктно-орієнтована мова програмування з довгою історією, яка десятиліттями слугувала фундаментом для Android та всього JVM-стеку. Вона базується на принципах надійності та стабільності, забезпечуючи розробникам передбачуване середовище завдяки зрілій віртуальній машині та великій екосистемі інструментів. Завдяки своїй поширеності Java залишається стандартом для підтримки великих корпоративних систем, де консервативний підхід є пріоритетними.

Основними перевагами Java є стабільність та надійність, велика кількість бібліотек, фреймворків, навчальних матеріалів, простота вивчення мови.

Основні недоліки Java – це висока кількість шаблонного коду, відсутність вбудованої Null безпеки, відсутність підтримки сучасних декларативних UI-інструментів, та складна модель асинхронності.

Kotlin – це сучасна, статично типізована мова програмування, офіційно рекомендована Google як основний інструмент для розробки Android-додатків з 2019 року. Мова була створена компанією JetBrains з метою усунути обмеження Java, зробивши код безпечнішим, лаконічнішим та продуктивнішим для інженера. Kotlin пропонує повну сумісність із Java-кодом, дозволяючи проектам плавно еволюціонувати та використовувати найновіші технології мобільної розробки, такі як Jetpack Compose та Multiplatform.

Основними перевагами Kotlin є значне скорочення шаблонного коду, вбудована null безпека, краща модель асинхронності, ніж в Java, та можливість кросплатформної розробки.

Недоліками Kotlin є довгий час компіляції, складність вивчення, та менша кількість навчальних ресурсів.

Враховуючи складність вивчення Kotlin, та значно вищий досвід розробника з Java, її обрано як мову програмування для цього проєкту.

Останнім аспектом розробки ПЗ, що необхідно розглянути, є модель життєвого циклу ПЗ.

Для більшості сучасних Android застосунків використовується методологія Agile, що є ітеративним та інкрементальним підходом до управління проєктами, який базується на розбитті розробки на короткі цикли, відомі як спринти, тривалістю від 1 до 4 тижнів. Методологія фокусується на гнучкості, безперервному постачанню робочого ПЗ та тісній взаємодії між крос-функціональними командами й замовником. У контексті Android цей підхід дозволяє командам оперативно адаптувати архітектуру застосунку до щорічних оновлень Google Play Store та специфічної фрагментації пристроїв.

Перевагами Agile є можливість швидкого випуску мінімального проєкту для перевірки ринкових гіпотез та збору відгуків користувачів, здатність адаптуватися до змін вимог навість на пізніх етапах розробки, та раннє виявлення технічних ризиків та дефектів завдяки безперервному тестуванню.

Основними недоліками Agile є складність прогнозування фінального бюджету та термінів розробки, а також потреба в зрілій та самоорганізованй команді.

Waterfall – це лінійна, послідовна модель розробки, де кожна наступна фаза розпочинається лише після повного завершення попередньої. Методологія робить акцент на ретельному попередньому плануванні та створенні вичерпної документації, такої як специфікація вимог SRS. Такий підхід забезпечує жорстку структуру та чітку простежуваність, що робить його стандартом для проектів із суворим регулюванням, де безпека та відповідність стандартам є пріоритетними.

Перевагами Waterfall є прогнозованість фінансових та часових витрат, фіксований обсяг робіт, та зменшення архітектурних ризиків завдяки комплексному проектуванню до початку реалізації.

Основними недоліками Waterfall є відсутність результатів придатних до використання до повного завершення розробки, та надзвичайна складність адаптації до змін у вимогах.

Зважаючи на обмежені ресурси, та той факт, що вимоги до проєкту вже визначені в повному обсязі, обрано каскадну модель життєвого циклу.

2.10 Висновки

Пройшовши всі етапи проєктування, необхідно оцінити результати проведених робіт та зробити підсумковий висновок. Під час проєктування було проведено ґрунтовний аналіз наявних архітектурних рішень та шаблонів. Зважаючи на вимоги щодо швидкодії під час роботи з доповненою реальністю та

обмежені терміни, для розроблюваного застосунку обрано шаблон проєктування MVC у поєднанні з модульною монолітною та закритою тришаровою архітектурою.

Здійснено проєктування структури збереження інформації. Проаналізовано сучасні системи керування базами даних та, з огляду на неефективність реляційних і сторонніх нереляційних підходів для збереження просторової топології, прийнято рішення про розроблення власного файлового сховища об’єктів на основі механізмів серіалізації.

На наступному етапі було проведено декомпозицію системи на різних рівнях. Спочатку визначено головні сутності предметної області, на основі яких архітектуру розділено на дев'ять автономних функціональних модулів. Для кожного з них детально описано зони відповідальності, внутрішні залежності та специфіковано програмні інтерфейси взаємодії. Опираючись на ці дані, було проведено детальне проєктування із визначенням структури класів, що відобразило повну внутрішню організацію програмного рішення.

Паралельно було спроєктовано інтерфейс користувача: визначено необхідний перелік вікон, структуру інтерфейсу для режиму просторового сканування та розроблено їхні макети для подальшої реалізації.

Заключним етапом став аналіз та вибір інструментальних засобів і методології. Зважаючи на надійність та стабільність, для розроблення обрано традиційну імперативну систему побудови інтерфейсів, платформу Google ARCore для обробки просторових даних, інтегроване середовище Android Studio, а також каскадну модель життєвого циклу розроблення через повну визначеність початкових вимог.

Результатом цього розділу стала вичерпна технічна документація. Вона забезпечує чітке розуміння внутрішньої структури розроблюваної системи і створює надійне підґрунтя для переходу до етапу безпосереднього написання коду застосунку для побудови планів поверхів будівель.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

3.1 Особливості програмної реалізації Android застосунків з використанням ARCore, Java, та інтерфейсу на основі XML файлів

Особливості програмної реалізації Android застосунків з використанням ARCore, Java, та інтерфейсу на основі XML файлів

Програмна реалізація застосунку доповненої реальності базується на використанні платформи просторового обчислення Google ARCore та класичного стека розробки під Android мовою Java. Базовою вимогою для функціонування системи є наявність апаратної підтримки алгоритмів ARCore. Для інтеграції бібліотек у середовищі розробки налаштовано систему збірки Gradle із підключенням репозиторію Maven від Google та ввімкненням сумісності з мовними конструкціями Java 8, що є обов'язковою вимогою для роботи сучасних AR-фреймворків.

Важливим етапом реалізації є конфігурація маніфесту застосунку AndroidManifest.xml. Програма потребує обов'язкового дозволу на використання камери. Застосунок налаштовано як такий, що критично залежить від доповненої реальності, для чого в маніфесті прописано метадані com.google.ar.core зі значенням required, а також додано вимогу до апаратного забезпечення android.hardware.camera.ar. Це гарантує, що програма буде доступна для встановлення лише на сертифікованих пристроях.

Реалізація користувацького інтерфейсу виконана за допомогою класичної XML-розмітки і розділена на два рівні: екранний та просторовий. В основі екранного інтерфейсу лежить компонент GLSurfaceView, який слугує контейнером для рендерингу графіки за допомогою апаратного прискорення. Користувацький інтерфейс реалізується за допомогою патерну накладання: GLSurfaceView розміщується на задньому плані, а поверх нього у контейнері розташовуються стандартні двовимірні XML-віджети Android для управління програмою.

Основна логіка взаємодії з простором написана мовою Java. Після ініціалізації сесії алгоритми ARCore безперервно аналізують відеопотік для виявлення площин (Plane). Коли користувач взаємодіє з інтерфейсом, програма виконує операцію Hit Test – проєктує промінь від точки на екрані у віртуальний 3D-простір для знаходження точок перетину з фізичними поверхнями. Для того, щоб розміщені тривимірні об'єкти візуально не зміщувалися відносно реального світу під час руху камери, створюються спеціальні об'єкти прив'язки (Anchor), до яких програмно прикріплюються вузли графа сцени (TransformableNode).

Критичним аспектом програмної реалізації на Java є управління пам'яттю. Оскільки об'єкт Session в ARCore керує значним обсягом пам'яті у Native Heap для обробки буферів камери та сенсорів, збирач сміття Java не здатен автоматично та вчасно звільняти ці ресурси. Тому в архітектурі застосунку реалізовано примусове звільнення нативних ресурсів шляхом явного виклику методу session.close() під час знищення активності (у методі onDestroy()), що запобігає витокам пам'яті та критичним збоям.

3.2 Програмна реалізація модулів

Типовим планом реалізації модулів для застосунку побудови планів поверхів будівель є розроблення в такому порядку:

– розроблення структур даних предметної області та підсистеми локального збереження, таких як моделі планів, просторових точок, відрізків та типів комунікацій;

– реалізація модуля просторового відстеження з використанням камери для відображення та фіксації вузлових точок у реальному часі;

– розроблення модуля двовимірної візуалізації зібраних просторових даних, обчислення відстаней та їх експорту у графічне подання;

– розроблення підсистем керування записами, а саме створення, редагування та видалення збережених планів і класифікаторів комунікацій;

– реалізація загальних засобів взаємодії з користувачем та навігації між екранами.

Розроблення структур даних предметної області та підсистеми локального збереження є базовим етапом створення застосунку, оскільки вони визначають спосіб подання, взаємодії та довготривалого зберігання зібраної просторової інформації. Архітектура даних будується на основі об’єктно-орієнтованого підходу з використанням механізмів абстракції для розділення фізичних стін та інженерних мереж.

Модель плану поверху (FloorPlan) є центральною сутністю, що агрегує всі дані конкретного приміщення. Вона містить унікальний ідентифікатор, назву плану, масив просторових точок, масив відрізків, а також перелік ідентифікаторів типів комунікацій, що присутні на цьому плані. Додатково модель зберігає стан готовності плану (прапорець завершеності сканування).

Модель просторової точки (FloorPlanPoint) описує координати вузла у тривимірному просторі відносно початкової точки відліку за осями X, Y, та Z. Кожна точка обов’язково містить посилання на свій тип (IPointType), що дозволяє системі розрізняти точки стін, точки прокладання комунікацій та вузли з'єднання комунікацій.

Модель відрізка (FloorPlanLine) визначає зв’язок між двома просторовими точками. Замість дублювання координат, модель зберігає лише індекси початкової та кінцевої точок з масиву плану. Відрізок також містить класифікатор типу (ILineType), який визначає, чи є ця лінія частиною стіни, чи відрізком конкретного типу комунікації.

Абстракції типів (IPointType та ILineType) використовуються для типізації точок і ліній забезпечує гнучкість системи. Реалізації цих інтерфейсів (WallPointType, CablePointType, CableConnectionPointType, WallLineType,

CableLineType) інкапсулюють логіку перевірки належності об'єкта до стіни або конкретної інженерної мережі.

Тип комунікації (CableType) є сутністю, що містить ідентифікатор та користувацьку назву типу інженерної мережі.

Сховище ідентифікаторів (IdStorage) відповідає за облік використаних ідентифікаторів для планів та типів комунікацій, запобігаючи конфліктам при створенні нових записів, а також дозволяє швидко отримувати списки ідентифікаторів існуючих об’єктів для їх відображення.

Для забезпечення вимог щодо конфіденційності даних та автономності роботи застосунку, підсистема збереження реалізована без використання мережевих баз даних. Збереження здійснюється виключно в ізольованій внутрішній пам'яті пристрою.

Основою підсистеми є механізм побайтового перетворення об'єктів (стандартна серіалізація java.io.Serializable). Усі моделі предметної області реалізують цей інтерфейс, що дозволяє зберігати весь граф об'єктів плану, включаючи всі точки, лінії, та їх типи єдиним потоком даних.

Для уникнення колізій та забезпечення швидкого пошуку використовується система префіксів у назвах файлів:

– файли планів поверхів зберігаються з назвами у форматі fp_<ідентифікатор> (наприклад, fp_1, fp_2);

– файли користувацьких типів комунікацій іменуються як ct_<ідентифікатор>;

– реєстр ідентифікаторів зберігається у статичному файлі id_storage.

Такий підхід дозволяє швидко розгортати підсистему збереження, гарантує цілісність складних просторових структур при записі та повністю відповідає вимогам щодо відсутності неконтрольованої передачі даних через інтернет.

Реалізація модуля просторового відстеження базується на використанні технологій доповненої реальності ARCore для розпізнавання оточення. Цей модуль виконує роль мосту між фізичним середовищем та цифровою моделлю плану.

Архітектурно процес відстеження та фіксації вузлових точок поділяється на кілька взаємопов'язаних етапів, що безперервно виконуються у циклі відтворення.

Модуль веде безперервний аналіз відеопотоку і датчика руху для виявлення горизонтальних площин, позначаючи їх напівпрозорими сітками для зручності користувача. Фіксація конкретних вузлів відбувається за допомогою трасування: система випускає уявний промінь із центру екрана та обчислює точку його перетину з реальною поверхнею, миттєво перетворюючи двовимірні координати прицілу на тривимірні дані.

Щоб об'єкти залишалися на своїх місцях під час руху пристрою, кожна точка отримує стабільну просторову прив'язку – Anchor. Завдяки спеціальному класу WrappedAnchor до цих точок додається інформація про їх тип та логічні зв’язки з іншими вузлами. Модуль також виконує складні векторні розрахунки для позиціювання об’єктів у просторі екрану, а також повороті ліній у напрямку від точки до точки, та їх масштабування до точної відстані між точками.

Розроблення модуля двовимірної візуалізації та експорту є завершальним етапом обробки зібраних просторових даних. Цей модуль відповідає за перетворення тривимірного графа вузлових точок та зв'язків у плоске ортогональне креслення, зручне для сприйняття, аналізу та подальшого використання у зовнішніх системах.

Архітектура модуля базується на послідовній трансформації даних від тривимірного сканування до готового графічного плану. Спочатку система здійснює проєкцію точок на площину, відкидаючи вертикальну вісь та визначаючи межі обмежувального прямокутника. На основі отриманих координат розраховується динамічний коефіцієнт масштабування, який адаптує фізичні метри приміщення до розмірів екранного полотна.

На фінальному етапі модуль відображає зображення в інтерфейсі або формує растрове PNG-зображення високої роздільної здатності. Готовий файл передається через потік виведення до системного засобу вибору документів, що

дозволяє користувачеві безпечно зберегти план у пам'яті пристрою без надання застосунку повного доступу до файлової системи.

Розроблення підсистем керування записами забезпечує життєвий цикл основних сутностей застосунку: планів приміщень та користувацьких типів комунікацій (кабелів). Ця підсистема відповідає за консистентність даних, синхронізацію між оперативною пам'яттю та файловим сховищем, а також за підтримку цілісності зв'язків між сутностями.

Підсистема базується на реєстрі IdStorage, який замінює механізм автоінкременту традиційних баз даних, динамічно виділяючи унікальні ідентифікатори для планів та типів комунікацій шляхом заповнення числових прогалин. Створення та редагування об'єктів реалізовано через спеціалізовані Activity, де зміни метаданих фіксуються миттєво за допомогою TextWatcher, а просторові дані інтегруються з AR-модуля. При видаленні плану система одночасно очищує відповідний запис у реєстрі та фізично стирає файл із пам'яті пристрою. Найскладнішим процесом є каскадне видалення типів комунікацій, яке вимагає автоматичного аудиту всіх наявних планів для вилучення пов'язаних ліній та очищення «осиротілих» точок, що не належать до стін. Така архітектура забезпечує повну автономність і цілісність даних.

Фрагмент коду каскадного видалення типів комунікацій:

```
public void removeCableTypeId(int id) {
    if(_cableTypeIds == null)
        return;

    _cableTypeIds = Arrays.stream(_cableTypeIds).filter(x -> x != id).toArray();
    if(isCaptured == false || _points.length == 0)
        return;

    ArrayList<ArrayList<ILineType>> lineMatrix = new ArrayList<>(_points.length);
    for(int i = 0; i < _points.length; i++){
        ArrayList<ILineType> column = new ArrayList<>(_points.length);
        lineMatrix.add(column);
        for(int j = 0; j < _points.length; j++){
            column.add(null);
        }
    }

    for(FloorPlanLine line : _lines){
        if(line.getLineType().isWall() || line.getLineType().getCableTypeId() != id) {
            lineMatrix.get(line.getPoint1Index()).set(line.getPoint2Index(),
line.getLineType());
            lineMatrix.get(line.getPoint2Index()).set(line.getPoint1Index(),
line.getLineType());
        }
```

```
    }

    List<FloorPlanPoint> pointsToRemove1 = Arrays.stream(_points).filter(x ->
x.getPointType().isCablePoint() && x.getPointType().getCableTypeId() ==
id).collect(Collectors.toList());
    List<FloorPlanPoint> pointsList = new ArrayList<>(_points.length);
    for(FloorPlanPoint point : _points){
        pointsList.add(point);
    }
    for(FloorPlanPoint pointToRemove: pointsToRemove1){
        int index = pointsList.indexOf(pointToRemove);
        lineMatrix.remove(index);
        for(ArrayList<ILineType> sublist : lineMatrix){
            sublist.remove(index);
        }
        pointsList.remove(index);
    }
    List<FloorPlanPoint> pointsToRemove2 = new ArrayList<>();
    for(int i = 0; i < lineMatrix.size(); i++){
        if(lineMatrix.get(i).stream().allMatch(x -> x == null)){
            pointsToRemove2.add(pointsList.get(i));
        }
    }
    for(FloorPlanPoint pointToRemove: pointsToRemove2){
        int index = pointsList.indexOf(pointToRemove);
        lineMatrix.remove(index);
        for(ArrayList<ILineType> sublist : lineMatrix){
            sublist.remove(index);
        }
        pointsList.remove(index);
    }
    _points = pointsList.stream().toArray(FloorPlanPoint[]::new);
    ArrayList<FloorPlanLine> linesList = new ArrayList<>();
    for(int i = 0; i < lineMatrix.size(); i++){
        for(int j = 0; j < i; j++){
            ILineType lineType = lineMatrix.get(i).get(j);
            if(lineType != null){
                linesList.add(new FloorPlanLine(i, j, lineType));
            }
        }
    }
    _lines = linesList.stream().toArray(FloorPlanLine[]::new);
}
```

Повний код програми наведено у додатку В.

3.3 Реалізація інтерфейсу користувача

В Android Studio є вбудований редактор для створення елементів інтерфейсу користувача UMG UI Designer. Цей інструмент надає змогу швидко створювати та програмувати весь потрібний інтерфейс, надаючи набір готових загальних елементів. При потребі можна створити потрібні елементи самостійно. Головним об’єктом інтерфейсу є View. Цей об’єкт може представляти собою, як весь екран так і його елемент.

Реалізація екрана списку планів базується на поєднанні XML-розмітки з вертикальним ScrollView та динамічної генерації елементів у FloorPlanListActivity, у межах якої активність програмно наповнює контейнер LinearLayout рядками у вигляді RelativeLayout, кожен з яких містить інтерактивну назву плану та кнопку видалення.

Реалізація екрана списку типів кабелів базується на поєднанні XML-розмітки з вертикальним ScrollView та динамічної генерації елементів у CableTypeListActivity, у межах якої активність програмно наповнює контейнер LinearLayout рядками у вигляді RelativeLayout, кожен з яких містить TextView для відображення назви типу та окремі кнопки для редагування параметрів і каскадного видалення об'єкта з усіх пов'язаних планів поверхів.

Реалізація екрана вибору типів кабелів для конкретного плану базується на поєднанні XML-розмітки з вертикальним ScrollView та динамічної генерації елементів у FloorPlanCableTypeListActivity, у межах якої активність програмно наповнює контейнер LinearLayout рядками у вигляді RelativeLayout, кожен з яких містить CheckBox для прив'язки типу до поточного плану, TextView із назвою та спеціалізовані текстові поля для відображення розрахованої сумарної довжини ліній і структурованого списку довжин окремих безперервних сегментів обраного типу комунікацій.

Реалізація екрана створення нового плану базується на XML-розмітці з кореневим RelativeLayout, що включає текстове поле EditText для задання назви та функціональну кнопку ImageButton для підтвердження дій користувача в інтерфейсі застосунку.

Реалізація екрана створення нового типу кабелю базується на XML-розмітці з кореневим RelativeLayout, що включає текстове поле EditText для введення найменування та функціональну кнопку ImageButton для підтвердження додавання нового типу комунікації до загального переліку застосунку.

Реалізація екрана редагування типу кабелю базується на XML-розмітці з кореневим RelativeLayout, що містить текстове поле EditText для корегування назви існуючого типу комунікації в інтерфейсі застосунку.

Реалізація екрана перегляду та редагування плану базується на XML-розмітці з кореневим RelativeLayout, що включає інтерактивну область ImageView для відображення згенерованого креслення, текстове поле EditText для налаштування назви та вертикальну панель функціональних кнопок ImageButton для переходу до AR-сканування, експорту плану у графічний файл та вибору типів комунікаційних мереж.

Реалізація екрана доповненої реальності базується на XML-розмітці з кореневим RelativeLayout, що включає компонент GLSurfaceView для рендерингу AR-сцени, центральний приціл ImageView та інформаційне текстове поле TextView для відображення поточної відстані. Інтерфейс доповнений випадаючим списком Spinner для вибору типу лінії та розгалуженою системою функціональних кнопок ImageButton, призначених для встановлення опорних точок, створення та конвертації зв'язків, приєднання або від'єднання вузлів, видалення елементів і фіксації результатів сканування.

3.4 Вимоги до технічних та програмних засобів

Визначення системних вимог є важливим етапом. Вони допомагають гарантувати, що гра працюватиме належним чином на обраному обладнанні, надають гравцям чітке уявлення про те, чи зможе їх комп'ютер запустити гру і сприяють кращому плануванню та оптимізації під час розроблення.

Головними характеристиками системи, до яких вставляються вимоги, це тип операційної системи, продуктивність SOC та кількість оперативної пам’яті, вільний дисковий простір, а також наявність підтримки Google ARCore пристроєм.

Аналізуючи розроблений застосунок було визначено мінімальні та рекомендовані системні вимоги 3.

Таблиця 3 – системі вимоги застосунку

	Мінімальні	Рекомендовані
ОС	Android 10	Android 15
SOC	Qualcomm Snapdragon 855, Exynos 9820, Mediatek MT6762 Helio P22	Qualcomm SM8550-AC Snapdragon 8 Gen 2 Exynos 1680 Mediatek Dimensity 9500s
Оперативна пам’ять	4 ГіБ	8 ГіБ
Місце на диску	256 МіБ	1 ГіБ
Підтримка ARCore	Обов’язкова	
Сервіси Google Play для AR	Наявність обов’язкова	

3.5 Тестування застосунку

Враховуючи те, що контролери цього проєкту мають сильне зчеплення з фреймворком інтерфейсу Android, їх модульне тестування неможливе.

Інтеграційне тестування, що не має таких обмежень, дозволить провести перевірку цих контролерів, а також решти функцій системи.

Автоматизоване тестування набагато більш повторюване, за ручне, повторення автоматизованих тестів займає менше часу та потребує значно менше уваги людей, проте деякі функції застосунку покладаються на доповнену реальність, або результати роботи функцій, що покладаються на неї, їх автоматизоване тестування є значно ускладненим, та не буде виконуватися.

Буде виконано ручне інтеграційне тестування та автоматизоване інтеграційне тестування за допомогою UIAutomator та JUnit.

Автоматизованому тестуванню підлягають наступні сценарії:

– створення плану поверху;

– відкриття плану поверху;

– видалення плану поверху;

– зміна назви плану поверху;

– створення користувацького типу відрізка;

– зміна назви користувацького типу відрізка;

– видалення користувацького типу відрізка;

– включення користувацького типу відрізку до плану;

– виключення користувацького типу відрізку з плану.

Ручному тестуванню підлягають наступні сценарії:

– вибір типу відрізка;

– створення точки обраного типу відрізка;

– обрання точки як «пов’язаної» (з якою автоматично створиться лінія при створенні нової точки);

– скидання вибору «пов’язаної» точки;

– створення точки з’єднання відрізків користувацьких типів;

– перетворення точки з’єднання відрізків користувацьких типів у точку відрізку користувацького типу;

– перетворення точки відрізків користувацьких типів у точку з’єднання відрізків користувацьких типів;

– видалення точки;

– з’єднання точки з «пов’язаною»;

– роз’єднання точки з «пов’язаною»;

– формування звіту про відрізки користувацьких типів;

– експорт плану поверху у вигляді PNG зображення.

Тестування сценаріїв створення полягає у навігації до екрану створення, введення назви, натискання кнопки завершення створення, та перевірки факту створення об’єкту.

Тестування сценаріїв перейменування полягає у створенні об’єкту, навігації до екрану перейменування, введення назви, повернення до списку, та перевірки факту перейменування об’єкту.

Тестування сценаріїв видалення полягає у створенні об’єкту, натисненні кнопки видалення, та перевірки факту видалення об’єкту.

Тестування сценаріїв відкриття полягає у створенні об’єкту, поверненні до списку, відкритті об’єкту, та перевірці факту переходу на екран цього об’єкту.

Тестування сценаріїв долучення/вилучення типів мереж до планів полягає у створенні відповідних об’єктів, відкритті екрану мереж плану, взаємодії з checkbox-ом вибору типу мережі, та перевірці його стану.

Для проведення автоматизованого тестування створено автоматизовані тести для кожного сценарію, код яких подано у додатку Б.1, а результати яких зображено на рисунку 3.1.

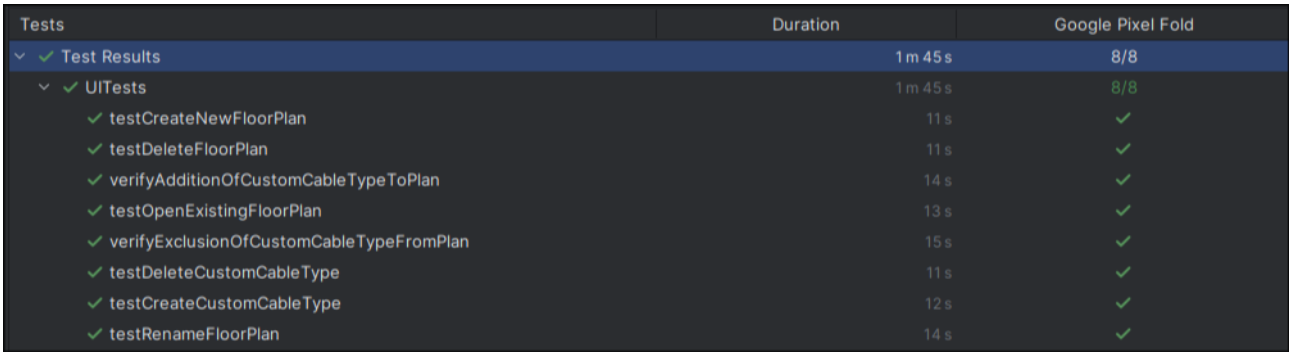


Рисунок 3.1 – результати автоматизованих тестів

Тестування сценарію вибору типу відрізка полягає у виборі типу відрізка на екрані сканування, перевірці збереження цього вибору, та перевірці відповідної зміни у відображенні об’єктів інженерних мереж.

Тестування сценарію встановлення точки полягає у наведенні центру екрану на виявлену площину на екрані сканування, натисканні відповідної кнопки, перевірці створення точки, та відрізка, що з’єднує її з «пов’язаною». Цей сценарій має кілька варіантів, для кожного можливого поєднання типів точок що створюються, та типів «пов’язаних» точок, оскільки від цього залежить факт створення відрізка.

Тестування сценарію обрання пов’язаної точки полягає в наведенні центру екрану близько до цієї точки, натисканні відповідної кнопки, та перевірці факту вибору.

Тестування сценарію скидання вибору «пов’язаної» точки полягає в наведенні центру екрану на площину далеко від існуючих точок, натисканні відповідної кнопки, та перевірці факту скидання вибору.

Тестування сценарію створення та видалення зв’язків полягає у виборі «пов’язаної» точки, наведенні екрану на іншу точку, та натисканні відповідної кнопки. Перевірка сценарію утворення зв’язку має варіанти для всіх комбінацій типів точок.

Тестування сценарію перетворення точки з точки з’єднання мереж у точку мережі полягає в наведенні центру екрану на таку точку, натисканні відповідної кнопки, та перевірці факту перетворення. Варіанти тесту: точка не має зв’язків, обрано тип відрізку «стіна»; точка не має зв’язків, та обрано користувацький тип відрізків, точка має від одного до двох зв’язків з одним типом відрізка, точка має більше двох зв’язків, точка має зв’язки з різними типами відрізків.

Тестування сценарію перетворення точки з точки інженерних мереж у точку з’єднання мереж полягає в наведенні центру екрану на таку точку, натисканні відповідної кнопки, та перевірці факту перетворення.

Тестування сценарію формування звіту про інженерні мережі проводиться шляхом відкриття екрану інженерних мереж плану, після сканування плану, що їх містить, та перевірці відповідності розрахованих значень.

Тестування сценарію експорту плану проводиться шляхом натискання відповідної кнопки на екрані плану, після сканування плану, обранні шляху файлу, та перевірці його вмісту.

Тестування сценаріїв екрану доповненої реальності зображено на рисунку 3.2.

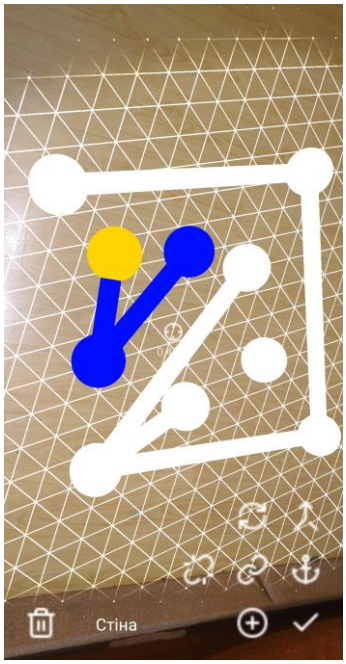


Рисунок 3.2 – тестування екрану доповненої

Виведення плану на екрані плану зображено на рисунку 3.3.

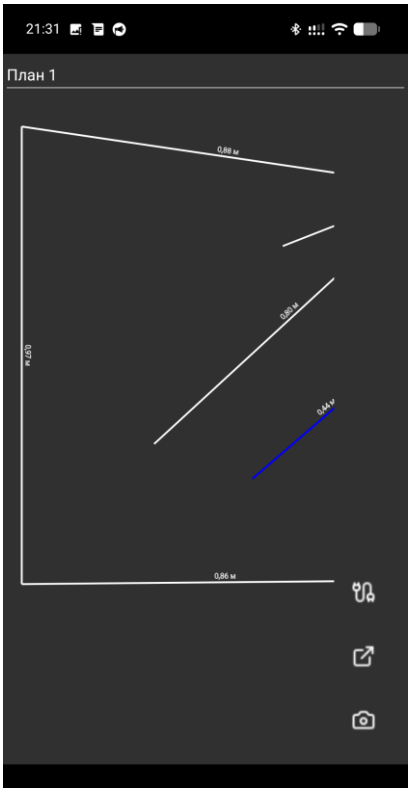


Рисунок 3.3 – виведення плану

Результати експорту плану зображено на рисунку 3.4.

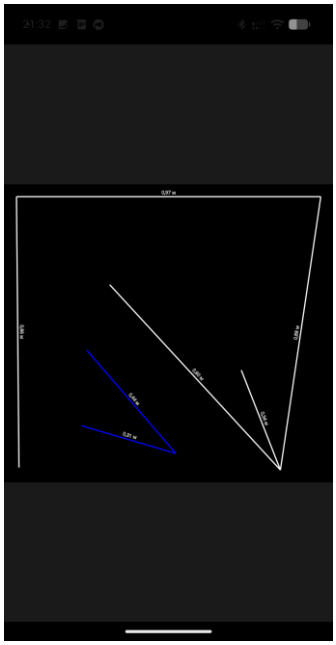


Рисунок 3.4 – експорт плану

Звіт про інженерні мережі плану зображено на рисунку 3.5.

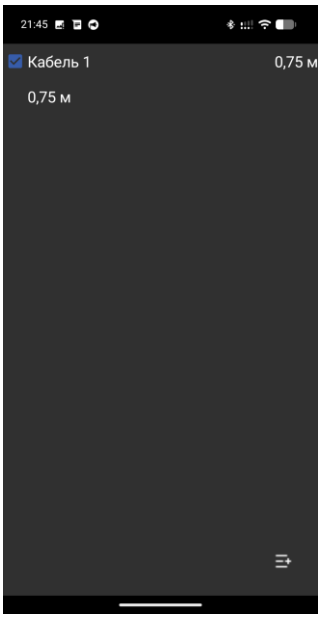


Рисунок 3.5 – звіт про інженерні мережі

Тестування показало повну відповідність системи функціональним вимогам, дефектів не виявлено.

На основі цього етапу було створено керівництво користувача, яке знаходиться в додатку Г.

3.6 Висновки

Для реалізації Android-застосунку побудови планів поверхів було проведено комплекс етапів роботи, які базувалися на результатах попереднього проєктування системи.

Першим важливим етапом початку реалізації застосунку була підготовка середовища розробки та конфігурація проєкту, що включало інтеграцію платформи просторового обчислення Google ARCore, управління нативною пам'яттю та чітке визначення мінімальних і рекомендованих системних вимог.

Після налаштування базової архітектури було реалізовано спроєктовані раніше модулі, використовуючи класичний стек розробки під Android мовою Java. Було розроблено об'єктно-орієнтовані структури даних предметної області та підсистему безпечного локального збереження на основі побайтової серіалізації, що забезпечило повну автономність роботи програми. Також було створено підсистему двовимірної візуалізації для обробки зібраних 3D-даних та їх експорту в ортогональні графічні креслення.

Окремим і найскладнішим етапом роботи стала програмна реалізація модуля просторового відстеження. За допомогою алгоритмів ARCore було реалізовано логіку виявлення площин, виконання операцій Hit Test та створення стабільних просторових прив’язок (Anchor) для коректного позиціювання об'єктів у реальному просторі під час руху камери.

Створення елементів користувацького інтерфейсу за допомогою класичної XML-розмітки та програмування їх функціоналу було наступним кроком розробки. Інтерфейс успішно поєднав стандарту двовимірну навігацію (екрани списків, створення та редагування записів) із просторовим відображенням через компонент GLSurfaceView.

Разом зі створенням модулів відбувалося тестування застосунку. Зважаючи на специфіку технологій доповненої реальності, було застосовано комбінований підхід: автоматизоване інтеграційне тестування (за допомогою UIAutomator та JUnit) для перевірки логіки роботи з даними (CRUD операції) та ретельне ручне тестування для верифікації просторових функцій, візуалізації і коректності генерації звітів.

Результатом цього розділу став повністю завершений та протестований Android-застосунок доповненої реальності, призначений для сканування приміщень і побудови планів інженерних комунікацій.

ВИСНОВКИ

Узагальнення результатів розробки мобільного застосунку для побудови планів поверхів будівель із використанням сенсорних даних Android-пристрою дало змогу сформулювати висновки.

Виконання розпочалось з дослідження предметної області та постановки задачі. Спочатку було визначено рівень актуальності технологій автоматизації збору просторових даних, проаналізовано існуюче програмно-технічне забезпечення та виявлено технологічні прогалини у засобах для документування інженерних мереж. Це підтвердило високу актуальність теми, дозволило обґрунтувати доцільність створення нового програмного продукту, сформувати вичерпні функціональні та нефункціональні вимоги до нього, розробити діаграму варіантів використання та поставити чіткі задачі для подальшого проєктування і реалізації.

Після визначення вимог і задач було проведено проєктування програмного забезпечення. Першим важливим етапом був вибір архітектурних підходів. На основі глибокого аналізу було обрано шаблон проєктування MVC у поєднанні з модульною монолітною та закритою тришаровою архітектурою. Враховуючи специфіку обробки просторової топології, було спроєктовано власну підсистему локального збереження даних на основі механізмів серіалізації, що забезпечило автономність та ефективність роботи застосунку.

На основі обраного архітектурного підходу було здійснено декомпозицію системи на дев'ять автономних функціональних модулів та побудовано діаграму класів системи. Було розроблено детальну структуру класів, визначено їхні властивості, а також описано міжмодульні залежності та програмні інтерфейси. Після цього було визначено вимоги до візуальної складової та створено макети всіх необхідних екранів, включно зі спеціалізованим інтерфейсом сканування доповненої реальності.

Наявність цих макетів та детального опису структурних зв'язків надала можливість глибоко зрозуміти композицію системи та ефективно підготуватися до етапу безпосередньої розробки програмного продукту.

Наприкінці стадії проєктування було обрано технології та методи реалізації. Для роботи з просторовими даними було обрано платформу Google ARCore, базовою мовою програмування визначено Java, а середовищем розроблення – Android Studio. Як метод керування життєвим циклом було обрано каскадну модель розробки через повну визначеність початкових технічних вимог.

Після завершення проєктування перед початком реалізації було розглянуто особливості програмування Android-застосунків з використанням ARCore та нативного інтерфейсу на основі XML. Спочатку було реалізовано структури даних предметної області та розроблено підсистеми управління збереженням. Найважливішим етапом стала програмна реалізація модуля просторового відстеження, що відповідає за розпізнавання поверхонь, обчислення векторів та встановлення просторових прив'язок. Згодом було створено модуль двовимірної візуалізації планів та інтегровано всі розроблені підсистеми з користувацьким інтерфейсом.

Останнім етапом було проведення тестування системи. Враховуючи специфіку взаємодії з фізичним простором, було визначено комбінований підхід до перевірки якості: автоматизоване інтеграційне тестування за допомогою UIAutomator та JUnit для перевірки логіки управління даними, а також інтенсивне ручне тестування для верифікації просторових функцій і алгоритмів креслення. Результати тестування підтвердили стабільну швидкодію, точність вимірювань та загальну надійність розробленого застосунку.

Отже, можна стверджувати, що мета кваліфікаційної роботи, яка полягала у створенні мобільного застосунку для побудови планів поверхів будівель, була повністю досягнута. Програмний продукт успішно розроблений, пройшов етап тестування і довів свою відповідність усім встановленим технічним та функціональним вимогам. Застосунок забезпечує точне просторове сканування, зручне керування типами інженерних мереж та безпечне збереження інформації на пристрої. Результати виконання роботи підтверджують її успішне завершення і готовність створеної системи до практичного використання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Будівельне креслення та особливості його оформлення [Електронний ресурс] – Режим доступу: https://naurok.com.ua/budivelne-kreslennya-287226.html (дата звернення: 20.02.2026).

2. Augmented Reality Measurement App Development Guide for Product Owners [Електронний ресурс] – Режим доступу: https://mobidev.biz/blog/ar-measurement-application-development (дата звернення: 19.02.2026).

3. Building an AR Land Measurement App with Android and ARCore(kotlin) [Електронний ресурс] – Режим доступу: https://medium.com/@akilalochana7/building-an-ar-land-measurement-app-with-android-and-arcore-795c26388baa (дата звернення: 22.02.2026).

4. 9–Step Augmented Reality App Development Guide [Електронний ресурс] – Режим доступу: https://codeit.us/blog/how-to-create-an-augmented-reality-app (дата звернення: 30.02.2026).

5. A Guide to Creating an Augmented Reality Application in 2025 [Електронний ресурс] – Режим доступу: https://www.appmaisters.com/a-guide-to-creating-an-augmented-reality-application/ (дата звернення: 25.02.2026).

6. Документація ARCore [Електронний ресурс] – Режим доступу: https://developers.google.com/ar/develop (дата звернення: 15.03.2026).

7. NadirFloorNet: reconstructing multi-room floorplans from a small set of registered panoramic images [Електронний ресурс] – Режим доступу: https://openaccess.thecvf.com/content/CVPR2025W/USM3D/papers/Pintore_NadirFloorNet_reconstructing_multi-room_floorplans_from_a_small_set_of_registered_CVPRW_2025_paper.pdf (дата звернення: 01.05.2026).

8. Indoor Localization using Computer Vision and Visual-Inertial Odometry [Електронний ресурс] – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC6497170/ (дата звернення: 03.05.2026).

9. AR Ruler App: Tape Measure Cam - Google Play [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details/AR_Ruler_App_Tape_Measure_Cam?id=com.grymala.aruler&hl=en_us&pli=1 (дата звернення: 03.05.2026).

10. AR Plan 3D Tape Measure, Ruler - Apps on Google Play [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.grymala.arplan&listing=ar-plan-3d-tape-measure&hl=en (дата звернення: 27.02.2026).

11. ARPLAN3D: The path of inspiration [Електронний ресурс] – Режим доступу: https://medium.com/@dzmitry_prybytak/arplan3d-the-story-of-creation-abcd4a6b5bdd (дата звернення: 27.02.2026).

12. AR Measure: Tape Measure Tool - Apps on Google Play [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.istack.ar.meter&hl=en (дата звернення: 27.02.2026).

13. AR Meter: Tape Measure Camera - Apps on Google Play [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.ignatiusDevStudio.ARMeter&hl=en-US (дата звернення: 27.02.2026).

14. AR Measure Plan: 3D Tape Ruler - Apps on Google Play [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.q4u.ruler.measurement.measure.ar.tape.camera.ai&hl=en-US (дата звернення: 27.02.2026).

15. AR Measure Plan: 3D Tape Ruler - Apps on Google Play [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.q4u.ruler.measurement.measure.ar.tape.camera.ai&hl=en-US (дата звернення: 27.02.2026).

16. Planner 5D: AI Home Design - Apps on Google Play [Електронний ресурс] – Режим доступу:

https://play.google.com/store/apps/details?id=com.planner5d.planner5d&hl=en-US (дата звернення: 27.02.2026).

17. Measure Tools: AR Ruler Camera - Apps on Google Play [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.craftars.measuretools&hl=en-US (дата звернення: 27.02.2026).

18. Android Architecture Patterns: Comparing MVC, MVP, MVVM, and MVI [Електронний ресурс] – Режим доступу: https://medium.com/@YodgorbekKomilo/android-architecture-patterns-comparing-mvc-mvp-mvvm-and-mvi-b282712cfb46 (дата звернення: 05.05.2026).

19. android - MVC vs MVP vs MVVM use cases - Stack Overflow [Електронний ресурс] – Режим доступу: https://stackoverflow.com/questions/54106509/mvc-vs-mvp-vs-mvvm-use-cases (дата звернення: 05.05.2026).

20. MVVM vs MVI: Which Architecture to Choose in 2026? [Електронний ресурс] – Режим доступу: https://sharpskill.dev/en/blog/android/mvvm-vs-mvi-architecture (дата звернення: 06.05.2026).

21. Types of Software Architecture in Product Development [Електронний ресурс] – Режим доступу: https://automios.com/types-of-software-architecture/ (дата звернення: 06.05.2026).

22. The Evolution of Software Architecture: From Mainframes to Microservices[Електронний ресурс] – Режим доступу: https://hosseinnejati.medium.com/the-evolution-of-software-architecture-from-mainframes-to-microservices-66ecc4052c7b (дата звернення: 06.05.2026).

23. The Evolution of Software Architecture: From Mainframes to Microservices [Електронний ресурс] – Режим доступу: https://hosseinnejati.medium.com/the-evolution-of-software-architecture-from-mainframes-to-microservices-66ecc4052c7b (дата звернення: 09.05.2026).

24. Understanding Software Architecture Types [Електронний ресурс] – Режим доступу: https://systemdesignschool.io/blog/software-architecture-types

25. Layered Architecture: The Traditional N-Tier Pattern [Електронний ресурс] – Режим доступу: https://bitloops.com/resources/software-architecture/layered-architecture (дата звернення: 09.05.2026).

26. Local-first software [Електронний ресурс] – Режим доступу: https://www.inkandswitch.com/essay/local-first/ (дата звернення: 05.05.2026).

27. Relational vs. Non-Relational Databases [Електронний ресурс] – Режим доступу: https://www.couchbase.com/blog/relational-vs-non-relational-database/ (дата звернення: 07.05.2026).

28. Recommendations for Android architecture | App architecture Databases [Електронний ресурс] – Режим доступу: https://developer.android.com/topic/architecture/recommendations (дата звернення: 05.05.2026).

29. Difference Between MVC, MVP and MVVM Architecture Pattern in Android – GeeksforGeeks [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/android/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android/ (дата звернення: 07.05.2026).

30. Architectural Patterns in Android Development: Comparing MVP, MVVM, and MVI [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/380208963_Architectural_Patterns_in_Android_Development_Comparing_MVP_MVVM_and_MVI (дата звернення: 07.05.2026).

31. Modern Mobile Architecture in Native Android and iOS | by Aleksey Lichtman – Medium [Електронний ресурс] – Режим доступу: https://medium.com/@aleksey.lichtman/modern-mobile-architecture-in-native-android-and-ios-cadd061cc42b (дата звернення: 07.05.2026).

32. How to Build a Simple Augmented Reality Android App? – GeeksforGeeks [Електронний ресурс] – Режим доступу:

https://www.geeksforgeeks.org/android/how-to-build-a-simple-augmented-reality-android-app/ (дата звернення: 05.05.2026).

33. Understanding Lifecycles in Jetpack Compose: Best Practices and Strategies for Effective Memory Management | by Rohit Neel | Medium[Електронний ресурс] – Режим доступу: https://medium.com/@rohitneel007/understanding-lifecycles-in-jetpack-compose-best-practices-and-strategies-for-effective-memory-9ffbec13b475 (дата звернення: 06.05.2026).

34. Android Jetpack Compose State: What I Wish Someone Had Explained Earlier [Електронний ресурс] – Режим доступу: https://levelup.gitconnected.com/android-jetpack-compose-state-what-i-wish-someone-had-explained-earlier-75350ecf907c (дата звернення: 07.05.2026).

35. Handling lifecycles with lifecycle-aware components (Views) - Android Developers [Електронний ресурс] – Режим доступу: https://developer.android.com/topic/architecture/views/lifecycle-views (дата звернення: 05.05.2026).

36. How to Build an Augmented Reality App Like IKEA Place - Next Big Technology [Електронний ресурс] – Режим доступу: https://nextbigtechnology.com/how-to-build-an-augmented-reality-app-like-ikea-place/ (дата звернення: 07.05.2026).

37. Adopting Local-First Architecture for Your Mobile App: A Game-Changer for User Experience and Performance - DEV Community [Електронний ресурс] – Режим доступу: https://dev.to/gervaisamoah/adopting-local-first-architecture-for-your-mobile-app-a-game-changer-for-user-experience-and-309g (дата звернення: 07.05.2026).

38. Relational vs. Non-Relational Databases: Key Differences [Електронний ресурс] – Режим доступу: https://www.couchbase.com/blog/relational-vs-non-relational-database/ (дата звернення: 05.05.2026).

39. Android Databases Performance Tests — CRUD | by Maciej Kozłowski – ProAndroidDev [Електронний ресурс] – Режим доступу: https://proandroiddev.com/android-databases-performance-crud-a963dd7bb0eb (дата звернення: 07.05.2026).

40. What is the difference between a Relational and Non-Relational Database? - Stack Overflow [Електронний ресурс] – Режим доступу: https://stackoverflow.com/questions/4811744/what-is-the-difference-between-a-relational-and-non-relational-database (дата звернення: 07.05.2026).

41. Realm vs SQLite: Which database to choose – Orangesoft [Електронний ресурс] – Режим доступу: https://orangesoft.co/blog/realm-vs-sqlite (дата звернення: 05.05.2026).

42. What to choose Realm or SQLite with Room? | by Mukesh Solanki | ITNEXT [Електронний ресурс] – Режим доступу: https://itnext.io/what-to-choose-realm-or-sqlite-with-room-e55c34b1675c (дата звернення: 07.05.2026).

43. Offline-First Android System Design: A Complete Guide — 1 | by Akshay Nandwana [Електронний ресурс] – Режим доступу: https://medium.com/@anandwana/offline-first-android-system-design-a-complete-guide-1-dae47eac680c (дата звернення: 07.05.2026).

ДОДАТОК А
(обов’язковий)

ГРАФІЧНІ МАТЕРІАЛИ

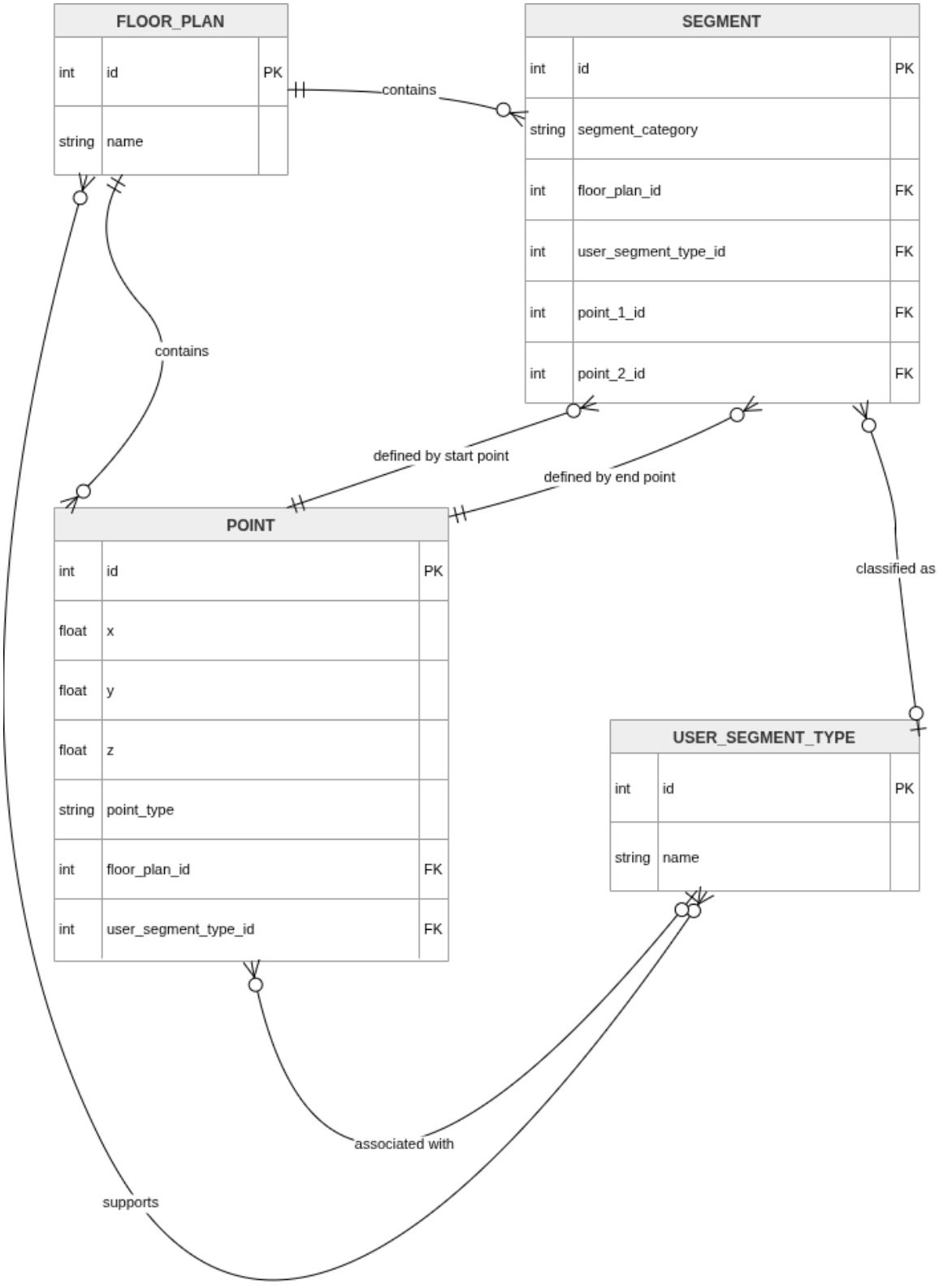


Рисунок А.1 – діаграма сутність-зв'язок

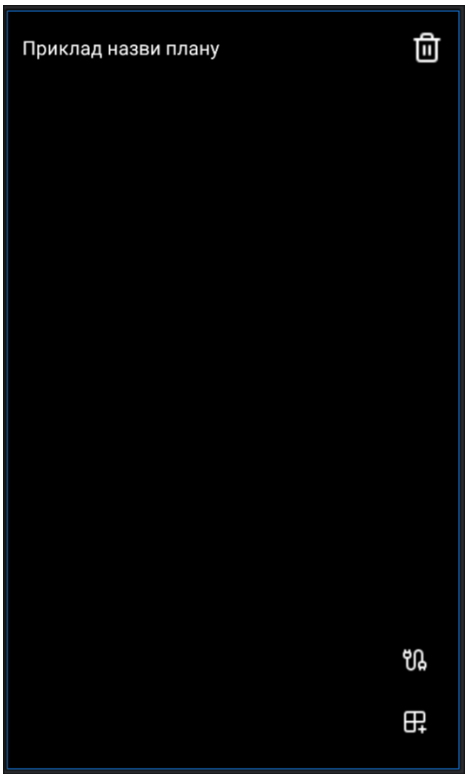


Рисунок А.2– макет головного екрану

Рисунок А.3 – макет екрану списку інженерних мереж

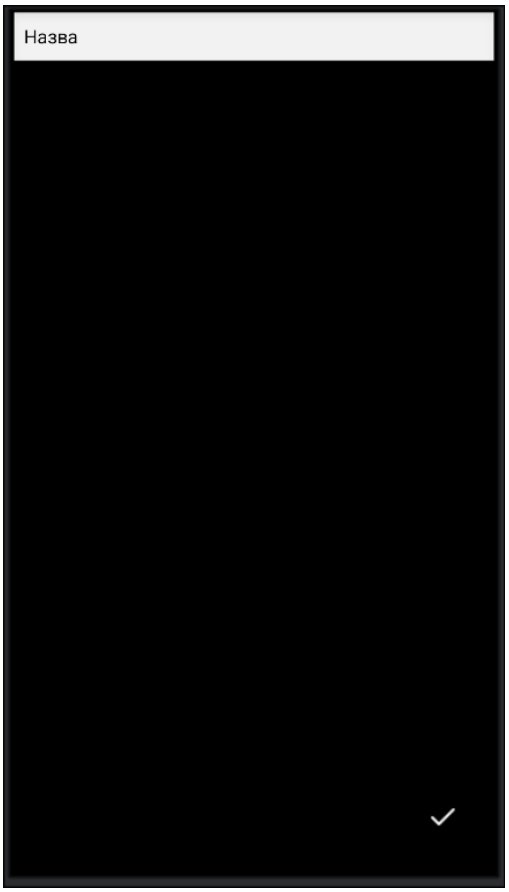


Рисунок А.4 – макет екрану створення плану

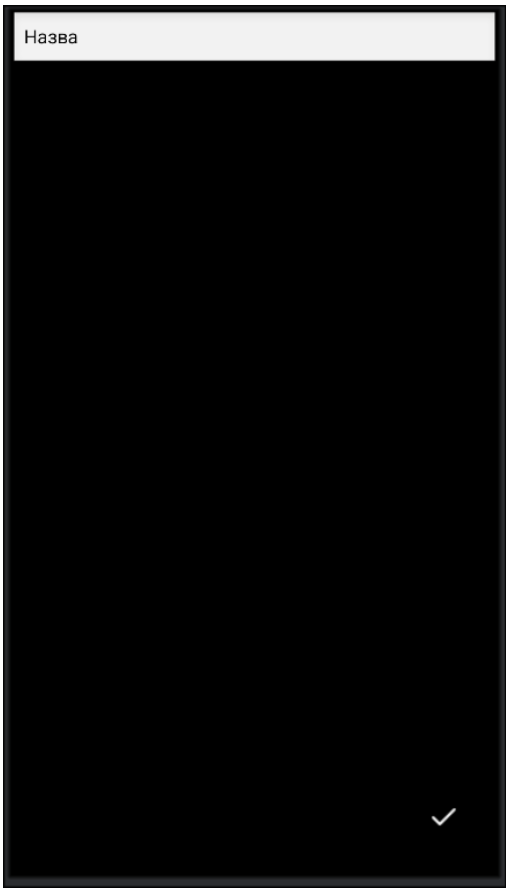


Рисунок А.5 – макет екрану створення інженерної мережі

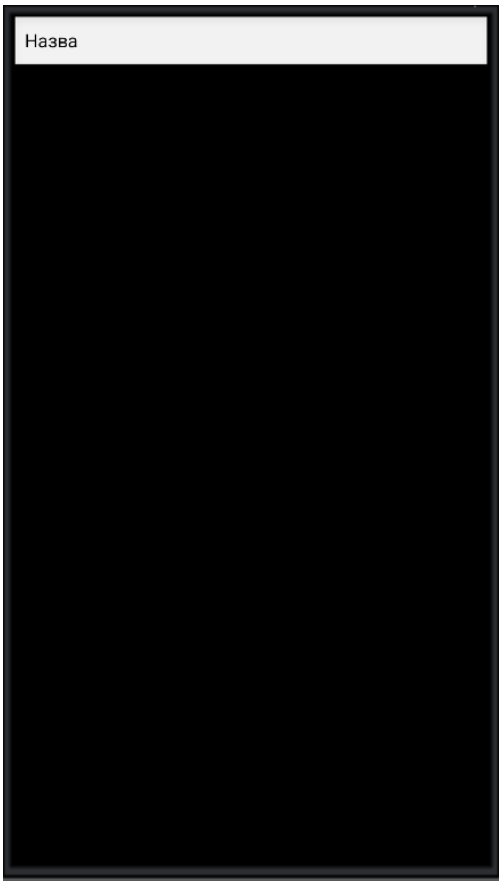


Рисунок А.6 – макет екрану перейменування типу інженерної мережі

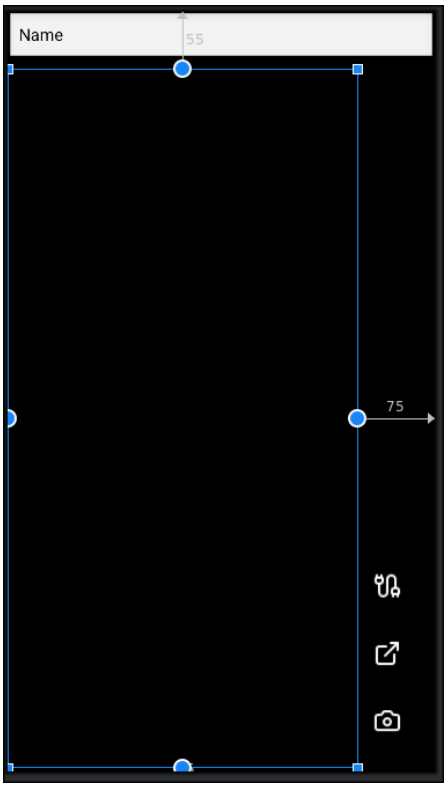


Рисунок А.7 – макет екрану плану

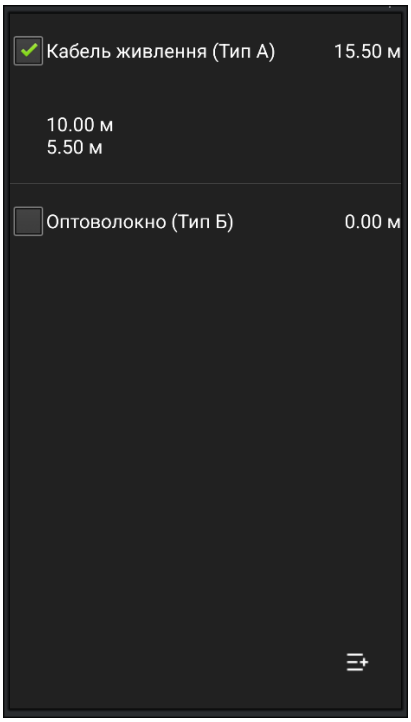


Рисунок А.8 – Макет екрану інженерних мереж плану

Рисунок А.9 – макет HUD екрану сканування

ДОДАТОК Б
(обов’язковий)

ДЕТАЛЬНИЙ ОПИС ВАРІАНТІВ ВИКОРИСТАННЯ

Варіант використання: створення плану поверху.

Передумови:

– користувач відкрив головний екран застосунку, на якому відображається перелік раніше створених об'єктів;

– пристрій має достатньо вільного місця у внутрішній пам'яті для збереження нових даних.

Тригер: користувач натискає кнопку додавання нового об'єкта на екрані переліку планів.

Основний потік:

1. після активації тригера система відображає екран для введення початкових даних нового об'єкта;

2. користувач вводить назву плану поверху у відповідне текстове поле;

3. користувач підтверджує створення, натиснувши кнопку підтвердження створення;

4. система ініціалізує новий об'єкт «План поверху» із заданою назвою;

5. система зберігає об’єкт у внутрішній пам’яті пристрою;

6. система автоматично перенаправляє користувача на екран управління планом.

Альтернативні потоки:

– скасування створення: користувач може повернутися до попереднього екрана за допомогою стандартних засобів навігації пристрою. У такому разі новий об'єкт не створюється, і дані не зберігаються.

Заборонні потоки:

– помилка доступу до сховища: Якщо система не може зберегти план, процес створення переривається. Користувач залишається на екрані введення даних.

Гарантії:

– мінімальна гарантія: У разі помилки або скасування жодних змін у внутрішній пам'яті пристрою не відбувається; цілісність існуючих планів зберігається;

– гарантія успіху: Створено новий об'єкт плану, дані збережено у внутрішній пам'яті, а користувачу надано інтерфейс для подальшої роботи з цим планом.

Діаграму станів для варіанту використання «створення плану поверху» зображено на рисунку Б.1.

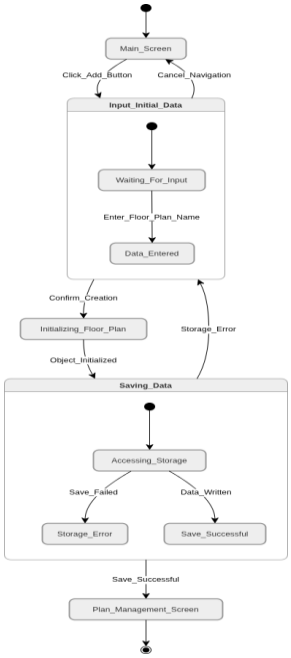


Рисунок Б.1 – діаграма станів створення плану поверху

Варіант використання: відкриття плану поверху.

Передумови:

– користувач перебуває на екрані зі списком збережених планів поверхів;

– у списку наявний принаймні один раніше створений план.

Тригер: натискання користувачем на текстове поле з назвою конкретного плану у списку.

Основний потік:

1. система зчитує збережені дані про геометричну структуру приміщення та розміщення інженерних мереж;

2. система відображає екран перегляду та редагування обраного плану;

3. у полі введення у верхній частині екрана відображається поточна назва плану;

4. система автоматично формує графічне зображення плану;

5. на графічному зображенні відображаються лінії стін та лінії сполучень інженерних мереж;

6. над кожним сегментом стіни або сполучення система виводить числове значення його довжини у метрах;

7. система надає доступ до кнопок переходу в режим вимірювання за допомогою камери, керування типами відрізків та збереження плану як окремого зображення.

Альтернативні потоки:

– відкриття порожнього плану: якщо для обраного плану ще не було проведено замірів у режимі доповненої реальності, система відображає порожню область замість графічного плану.

Заборонні потоки:

– помилка доступу до сховища: у разі неможливості отримати дані про обраний об'єкт, система припиняє спробу відкриття та повертає користувача до загального списку планів.

Гарантії:

– дані обраного плану завантажені та доступні для перегляду;

– користувач бачить актуальну візуалізацію виміряних об'єктів та їх параметрів.

Діаграму станів для варіанту використання «відкриття плану поверху» зображено на рисунку Б.2.

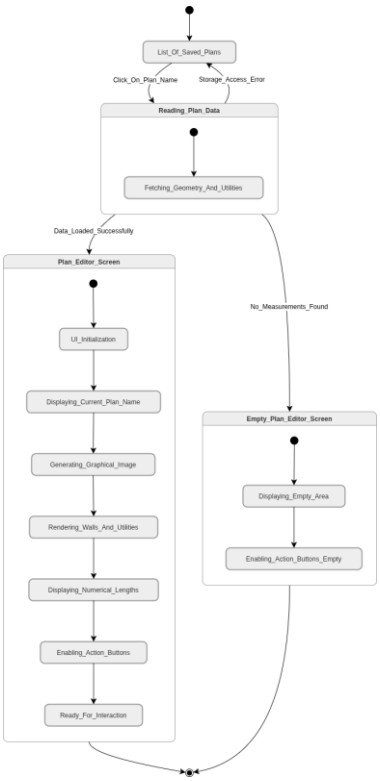


Рисунок Б.2 – діаграма станів відкриття плану поверху

Варіант використання: видалення плану поверху.

Передумови:

– користувач перебуває на екрані переліку планів поверхів;

– у списку відображається принаймні один раніше створений план;

Тригер: натискання кнопки видалення, що розташована поруч із назвою конкретного плану поверху в загальному списку.

Основний потік:

1. користувач обирає план, який необхідно вилучити, та натискає кнопку видалення;

2. система ініціює процес очищення збережених даних, що стосуються цього плану;

3. система оновлює відображення списку на екрані, видаляючи відповідний рядок;

4. користувач бачить актуальний перелік планів без видаленого об'єкта;

Альтернативні потоки відсутні.

Заборонні потоки:

– помилка доступу до сховища: у разі неможливості отримати дані про обраний об'єкт, система припиняє спробу видалення, відображений список залишається без змін.

Гарантії:

– дані про план поверху та пов'язані з ним типи інженерних мереж остаточно вилучаються зі збережених об'єктів;

– на екрані відображається лише актуальний перелік планів, доступних для подальшої роботи.

Діаграму станів для варіанту використання «видалення плану поверху» зображено на рисунку Б.3.

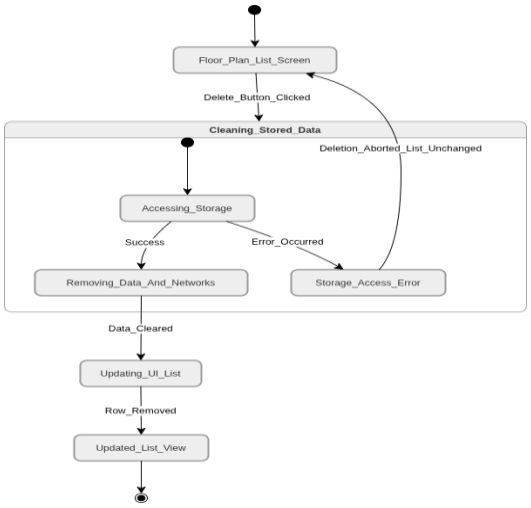


Рисунок Б.3 – діаграма станів видалення плану поверху

Варіант використання: зміна назви плану поверху.

Передумови:

– користувач обрав існуючий план поверху зі списку доступних об'єктів;

– система відкрила екран перегляду та редагування плану поверху.

Тригер: взаємодія користувача з текстовим полем, що містить назву плану.

Основний потік:

1. користувач активує текстове поле для назви плану на екрані редагування;

2. користувач вносить зміни в текст або вводить абсолютно нову назву за допомогою клавіатури пристрою;

3. система автоматично відстежує кожну зміну введених символів у реальному часі;

4. після введення кожного нового символу система оновлює назву об'єкта в пам’яті;

5. система ініціює автоматичне збереження оновлених даних плану до сховища без потреби у додатковому натисканні кнопки підтвердження.

Альтернативні потоки відсутні.

Заборонні потоки:

– під час спроби автоматичного запису оновлених даних виникає помилка доступу до пам'яті пристрою: система припиняє спробу оновлення та залишає попередню успішно збережену назву.

Гарантії:

– назва плану поверху успішно оновлена та зафіксована у системі;

– при наступному відкритті списку всіх об'єктів або повторному відкритті цього плану користувач побачить нову назву.

Діаграму станів для варіанту використання «зміна назви плану поверху» зображено на рисунку Б.4.

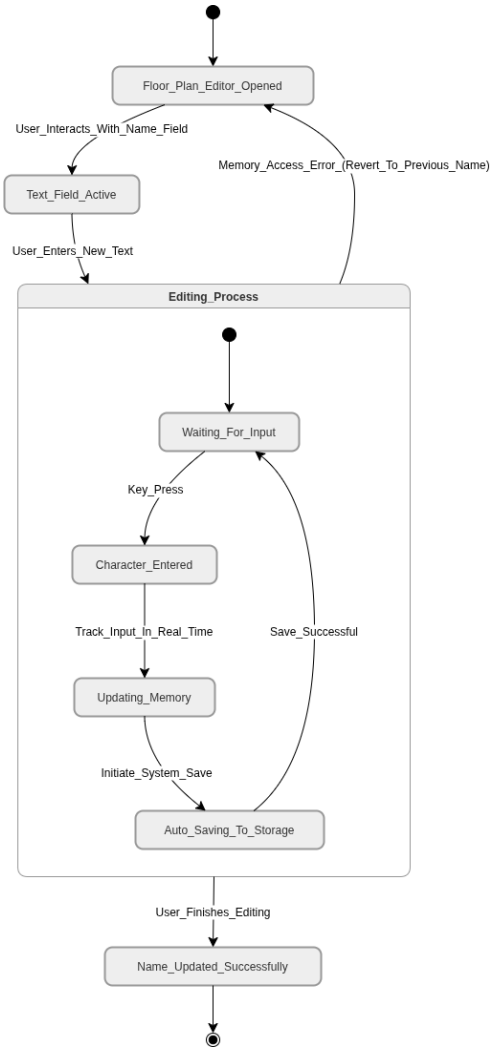


Рисунок Б.4 – діаграма станів зміни назви плану поверху

Варіант використання: створення користувацького типу відрізка.

Передумови:

– користувач перебуває на екрані перегляду списку доступних типів елементів інженерних мереж;

– пристрій має достатньо вільної пам'яті для збереження нових даних.

Тригер: натискання кнопки додавання нового типу об'єкта на екрані переліку типів.

Основний потік:

1. система відкриває екран із полем для введення текстової назви;
2. користувач вводить назву, що ідентифікує конкретну інженерну мережу;
3. користувач підтверджує дію, натиснувши кнопку створення;

4. система фіксує введені дані та зберігає новий об'єкт;

5. система автоматично повертає користувача до оновленого списку типів.

Альтернативні потоки:

– користувач може відмовитися від створення, скориставшись функцією повернення назад до моменту підтвердження.

Заборонні потоки:

– помилка доступу до сховища: у разі неможливості зберегти об’єкт система залишає користувача на екрані введення даних.

Гарантії:

– новий тип інженерної мережі успішно внесено до переліку;

– створений тип стає доступним для подальшого використання та призначення відрізкам під час сканування приміщень;

– дані про тип інженерної мережі залишаються доступними після перезапуску застосунку.

Діаграму станів для варіанту використання «створення користувацького типу відрізка» зображено на рисунку Б.5.

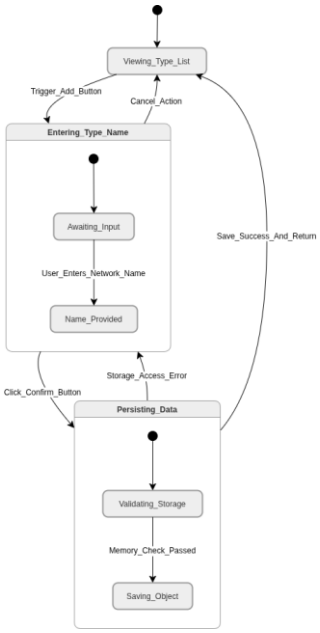


Рисунок Б.5 – діаграма станів створення користувацького типу

Варіант використання: зміна назви користувацького типу відрізка.

Передумови:

– користувач відкрив екран зі списком доступних типів інженерних мереж;

– у переліку відображається принаймні один раніше створений тип мережі.

Тригер: натискання кнопки редагування поруч із назвою конкретного типу мережі у списку;

Основний потік:

1. система відкриває екран редагування обраного типу мережі;

2. система відображає текстове поле, автоматично заповнене поточною назвою цього типу;

3. користувач встановлює курсор у текстове поле та вносить зміни в назву;

4. система автоматично фіксує та зберігає оновлену назву під час введення кожного нового символу;

5. користувач завершує редагування та повертається до екрана зі списком типів мереж.

Альтернативні потоки відсутні.

Заборонні потоки:

– помилка доступу до сховища: у разі неможливості зберегти об’єкт система залишає користувача на екрані введення даних.

Гарантії:

– оновлена назва типу мережі зафіксована у системі;

– нова назва коректно відображається у загальному списку типів;

– на всіх планах приміщень, де було використано цей тип інженерної мережі, назву оновлено;

Діаграму станів для варіанту використання «зміна назви користувацького типу відрізка» зображено на рисунку Б.6.

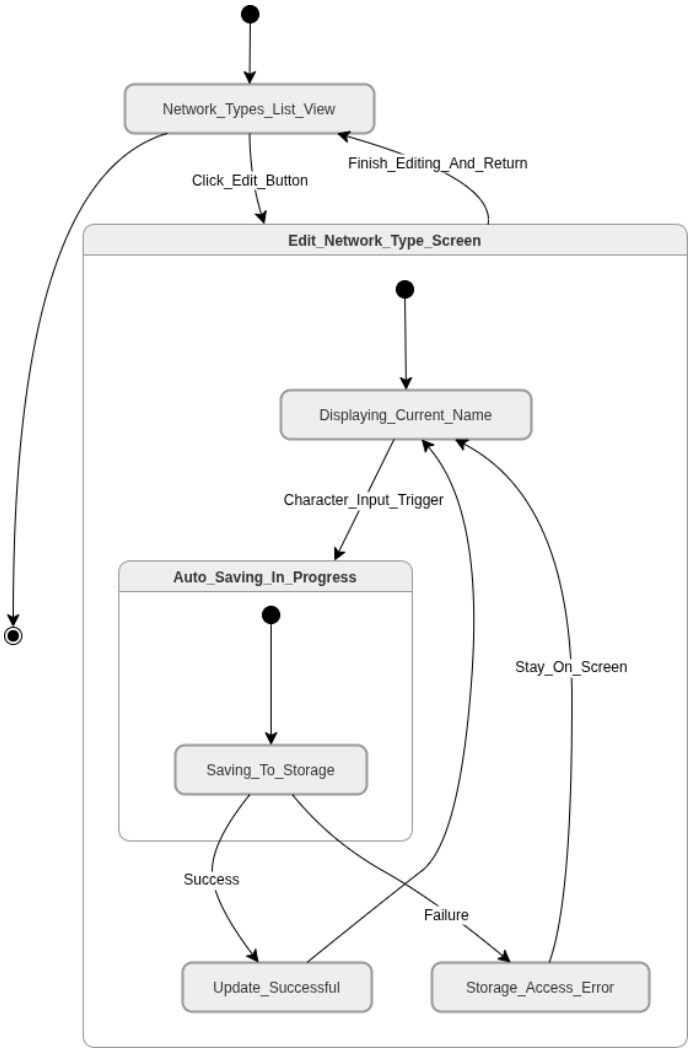


Рисунок Б.6 – діаграма станів зміни назви користувацького типу

Варіант використання: видалення користувацького типу відрізка.

Передумови:

– користувач перебуває на екрані перегляду списку доступних типів інженерних мереж;

– у переліку наявний принаймні один створений раніше тип мережі.

Тригер: натискання користувачем кнопки видалення (іконка кошика) поруч із назвою конкретного типу мережі.

Основний потік:

1. система ініціює перевірку всіх збережених планів поверхів на предмет використання цього типу;

2. система знаходить кожен план, де було задіяно обраний тип мережі, та видаляє згадку про нього з налаштувань плану;

3. система автоматично вилучає з планів усі відрізки (лінії), що були позначені цим типом;

4. система проводить аналіз точок (вузлів) на планах та видаляє ті вузли, що належали до цього типу мережі або стали ізольованими після видалення ліній;

5. система зберігає оновлені плани поверхів;

6. система остаточно вилучає тип інженерної мережі із загального переліку;

7. система оновлює екран списку, демонструючи актуальний перелік типів без видаленого об'єкта.

Альтернативні потоки відсутні.

Заборонні потоки:

– якщо під час операції стається критична помилка доступу до збережених даних, процедура переривається;

– якщо цільовий тип мережі не знайдено у системі під час ініціалізації видалення, система оновлює список без виконання додаткових дій.

Гарантії:

– тип інженерної мережі повністю вилучено із системи;

– усі плани поверхів очищені від специфічних елементів (ліній та вузлів), що належали до видаленого типу.

Діаграму станів для варіанту використання «видалення користувацького типу відрізка» зображено на рисунку Б.7.

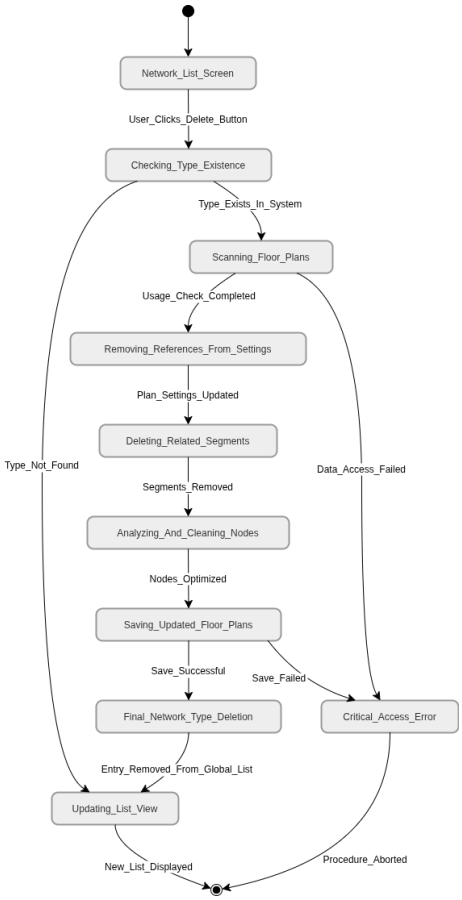


Рисунок Б.7 – діаграма станів видалення користувацького типу відрізка

Варіант використання: включення користувацького типу відрізку до плану.

Передумови:

– користувач створив або відкрив для редагування існуючий план будівлі;

– користувач перейшов до спеціалізованого екрана керування переліком типів інженерних мереж для цього плану.

Тригер: активація користувачем індикатора вибору (позначки) поруч із назвою відповідної мережі;

Основний потік:

1. збереження оновленого складу плану з урахуванням обраного типу комунікацій;

2. відображення зміненого стану обраного типу комунікацій.

Альтернативні потоки відсутні

Заборонні потоки:

– помилка доступу до сховища: у разі неможливості зберегти об’єкт система залишає користувача на екрані списку керування переліком типів інженерних мереж для цього плану.

Гарантії:

– обраний тип інженерної мережі успішно закріплений за планом;

– обраний тип стає доступним для використання у режимі проведення вимірювань за допомогою камери пристрою;

– всі внесені зміни зафіксовані у пам'яті пристрою.

Діаграму станів для варіанту використання «включення користувацького типу відрізку до плану» зображено на рисунку Б.8.

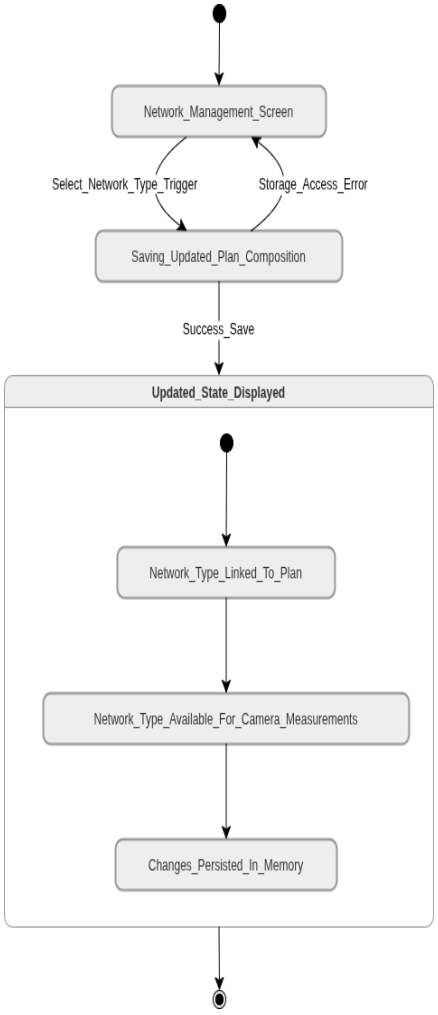


Рисунок Б.8 – діаграма станів включення користувацького типу відрізку

Варіант використання: виключення користувацького типу відрізку з плану.

Передумови:

– користувач відкрив екран переліку типів інженерних мереж для конкретного плану поверху;

– у списку відображаються раніше створені типи мереж;

– для обраного плану встановлено належність хоча б одного типу інженерної мережі;

Тригер: користувач знімає позначку з прапорця (вимикає вибір) навпроти назви конкретного типу інженерної мережі.

Основний потік:

1. система видаляє цей тип мережі зі списку активних типів для поточного плану поверху;

2. система автоматично перевіряє наявність на графічному плані відрізків, що належать до виключеного типу;

3. система видаляє з плану всі відрізки, які відповідають виключеному типу мережі;

4. система аналізує точки з’єднань та автоматично видаляє ті з них, які належали виключно до цієї мережі або стали ізольованими (не з’єднані з жодним іншим об’єктом) після видалення відрізків;

5. система оновлює показники загальної довжини для цього типу мережі на екрані, встановлюючи значення на нуль;

6. система зберігає оновлений стан плану поверху.

Альтернативні потоки:

– якщо на плані ще не було побудовано жодних відрізків виключеного типу, система лише видаляє зв’язок між типом мережі та планом;

Заборонні потоки:

– помилка доступу до сховища: у разі неможливості зчитати або зберегти об’єкт система залишає користувача на екрані списку керування переліком типів інженерних мереж для цього плану, збережені дані не змінено.

Гарантії:

– обраний тип інженерної мережі більше не вважається частиною поточного плану поверху;

– усі графічні елементи мережі (лінії та специфічні точки), що належали до цього типу, остаточно видалені з плану;

– оновлені дані плану успішно зафіксовані;

– інтерфейс користувача відображає актуальний стан мереж.

Діаграму станів для варіанту використання «виключення користувацького типу відрізку з плану» зображено на рисунку Б.9.

Рисунок Б.9 – діаграма станів виключення користувацького типу відрізку

Варіант використання: вибір типу відрізка.

Передумови:

– користувач перейшов до екрана вимірювання та побудови плану приміщення;

– система ініціалізувала сесію доповненої реальності та відображає відеопотік з камери;

– для поточного плану визначено хоча б один тип інженерної мережі (необов'язково, за замовчуванням доступний тип для побудови стін).

Тригер: користувач натискає на випадаючий список вибору типу лінії на екрані вимірювання.

Основний потік:

1. система розгортає список доступних типів ліній;
2. користувач переглядає назви доступних інженерних мереж та архітектурних елементів;
3. користувач натискає на назву конкретної інженерної мережі;
4. система фіксує обрану мережу як активну;
5. система згортає список та відображає назву обраного типу в інтерфейсі;
6. система оновлює візуальні параметри елементів плану.

Альтернативні потоки:

– користувач обирає у списку пункт «Стіна» – система встановлює режим побудови стандартних архітектурних перегородок та оновлює візуальні параметри елементів плану.

Заборонні потоки:

– користувач відкриває список, у якому доступний лише варіант «Стіна»;

– користувач закриває список без зміни вибору.

Гарантії:

– обраний тип лінії залишається активним до наступної зміни або до завершення сеансу вимірювання;

– всі нові точки та з'єднання, створені після вибору, автоматично класифікуються згідно з обраною інженерною мережею;

– система автоматично розраховує довжину нових відрізків з урахуванням специфіки обраного типу (враховуючи вертикальні зміщення для мереж але не для стін).

Діаграму станів для варіанту використання «вибір типу відрізка» зображена на рисунку Б.10.

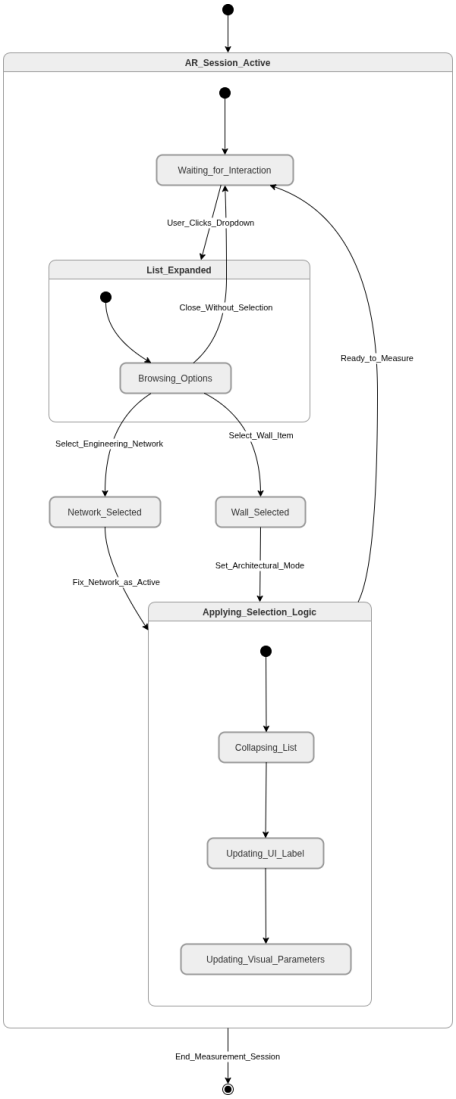


Рисунок Б.10 – діаграма станів вибору типу відрізка

Варіант використання: створення точки обраного типу відрізка.

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

Тригер: натискання кнопки створення точки;

Основний потік:

1. система виконує аналіз середовища у центральній точці екрана камери;

2. система визначає просторові координати на виявленій поверхні;

3. система створює нову точку відповідно до обраного користувачем типу об’єкта (стіна або конкретний тип інженерної мережі);

4. система автоматично формує лінію зв’язку між новоствореною точкою та «пов’язаною»;

5. візуальне відображення плану оновлюється з урахуванням нового сегмента та точки;

6. система призначає новостворену точку як «пов’язану».

Альтернативні потоки:

– «пов’язана» точка не обрана – система не створює лінію зв’язку;

– з’єднання точки з «пов’язаною» неприпустиме (з’єднання стіни з інженерною мережею, різних інженерних мереж не через спеціальну точку з’єднання або розгалуження інженерної мережі не через спеціальну точку з’єднання) – система не створює лінію зв’язку;

Заборонні потоки:

– система не знаходить розпізнаної площини у центрі екрана – команда створення точки ігнорується;

– втрата орієнтації у просторі – команда створення точки ігнорується;

Гарантії:

– координати нової точки та інформація про її приналежність до типу мережі додаються до плану.

Діаграму станів для варіанту використання «створення точки обраного типу відрізка» зображено на рисунку Б.11.

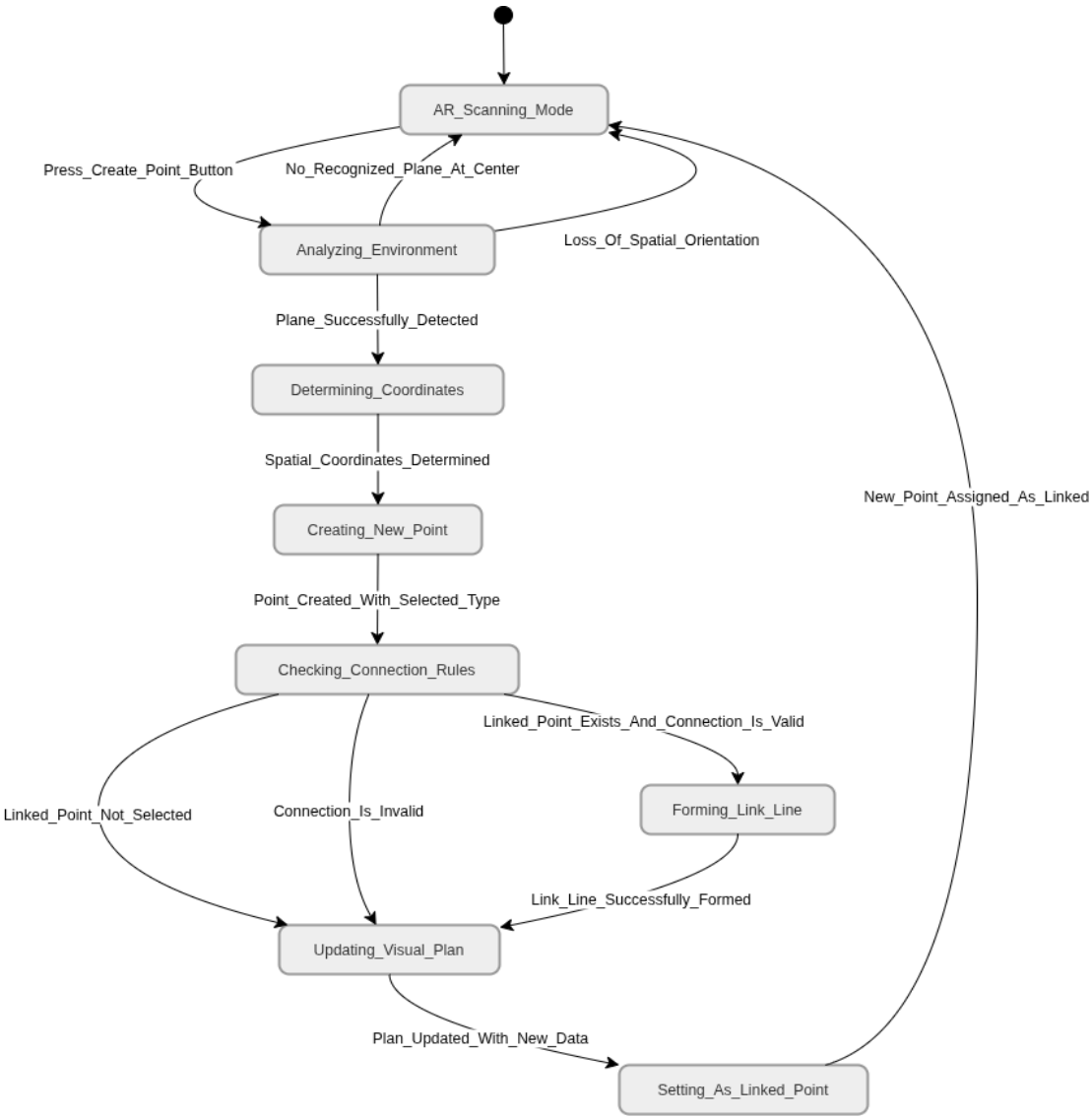


Рисунок Б.11 – діаграма станів створення точки обраного типу

Варіанти використання: обрання точки як «пов’язаної» та скидання вибору.

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

– на поточному плані вже створено принаймні один об’єкт (вузол інженерної мережі або кут стіни).

Тригер: натискання користувачем кнопки вибору існуючої точки на екрані камери.

Основний потік:

1. користувач спрямовує камеру пристрою таким чином, щоб існуюча на плані точка опинилася в центрі екрана;

2. користувач натискає кнопку вибору існуючої точки;

3. система виконує віртуальне сканування області в центрі екрана для пошуку встановлених об’єктів;

4. система визначає найближчу до центру екрана точку в межах допустимої відстані;

5. система призначає знайдений об’єкт як поточну «пов’язану» точку;

6. система змінює візуальний стан об’єкта, додаючи анімацію пульсуючого кольору для підтвердження вибору;

7. система активує динамічне текстове поле, що відображає поточну відстань у метрах від «пов’язаної» точки до центрального маркера на екрані.

Альтернативні потоки:

– точок на допустимій відстані не виявлено – система скидає вибір «пов’язаної» точки;

Заборонні потоки:

– втрата орієнтації у просторі – команда зміни “пов’язаної” точки ігнорується;

Гарантії:

– обрана точка фіксується у пам’яті як активна база для подальшого створення з’єднань інженерних мереж;

– візуальна індикація та вимірювання відстані залишаються активними до моменту видалення точки, вибору іншої точки, скасування вибору, або завершення сеансу роботи з камерою.

Діаграму станів для варіантів використання «обрання точки як «пов’язаної» та скидання вибору» зображено на рисунку Б.12.

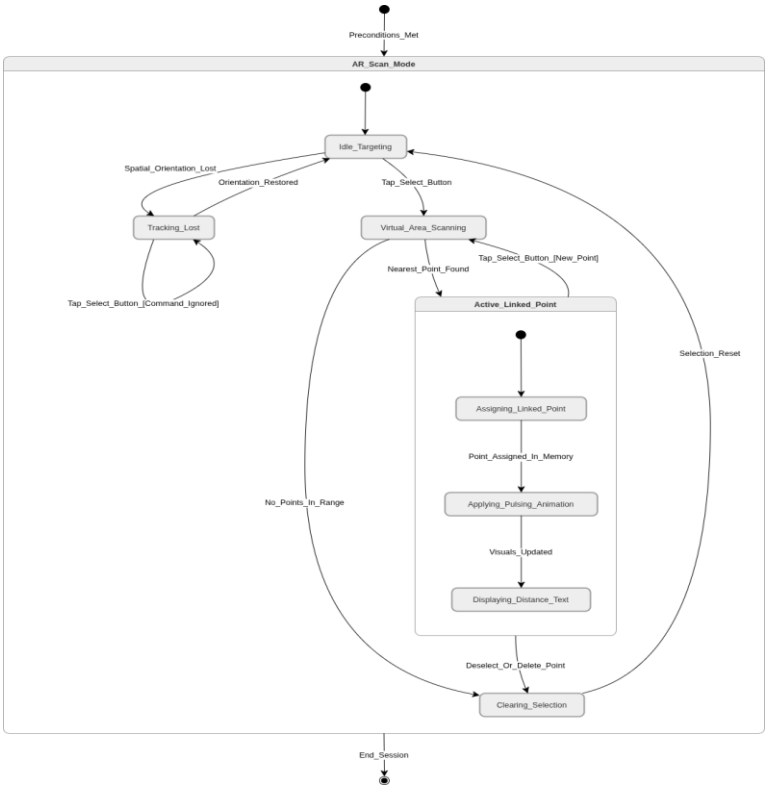


Рисунок Б.12 – діаграма станів обрання точки як «пов’язаної»

Варіант використання: створення точки з’єднання відрізків користувацьких типів.

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

Тригер: натискання користувачем кнопки створення точки з’єднання мереж на екрані камери.

Основний потік:

1. користувач активує режим створення точки з’єднання;

2. система виконує аналіз простору за напрямком погляду камери для пошуку точок перетину з розпізнаною поверхнею;

3. система визначає найближчу існуючу точку в зоні фокусування камери;

4. система автоматично створює новий віртуальний об’єкт у формі жовтої сфери на поверхні;

5. якщо до цього була обрана або створена «пов’язана» точка, система автоматично будує лінію зв’язку між новою точкою та попередньою;

6. користувач бачить візуальне підтвердження створення вузла та його зв’язку з іншими елементами плану;

7. система оновлює відображення поточної відстані від створеної точки до центру екрана;

8. дані про нову точку та її зв’язки зберігаються в проекті плану.

Альтернативні потоки:

– «пов’язана» точка не обрана – система не створює лінію зв’язку;

– з’єднання точки з «пов’язаною» неприпустиме (з’єднання стіни з інженерною мережею, різних інженерних мереж не через спеціальну точку з’єднання або розгалуження інженерної мережі не через спеціальну точку з’єднання, або обрано стіну як тип відрізку і йде спроба пов’язати дві точки з’єднання) – система не створює лінію зв’язку;

Заборонні потоки:

– система не знаходить розпізнаної площини у центрі екрана – команда створення точки ігнорується;

– втрата орієнтації у просторі – команда створення точки ігнорується;

Гарантії:

– координати нової точки додаються до плану.

Діаграму станів для варіанту використання «створення точки з’єднання відрізків користувацьких типів» зображено на рисунку Б.13.

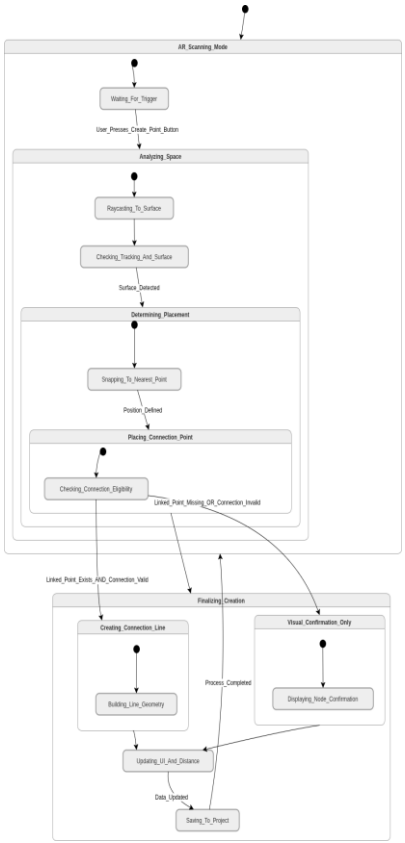


Рисунок Б.13 – діаграма станів створення точки з’єднання

Варіант використання: перетворення точки з’єднання відрізків користувацьких типів у точку відрізку користувацького типу.

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

– на плані вже розміщено принаймні одну точку, що є точкою з’єднання відрізків користувацьких типів.

Тригер: натискання користувачем кнопки перетворення типу точки в інтерфейсі сканування.

Основний потік:

1. користувач спрямовує камеру пристрою на існуючу на плані точку з’єднання інженерних мереж;

2. система візуально виділяє найближчу точку підтверджуючи готовність до взаємодії;

3. користувач натискає кнопку перетворення типу точки;

4. система визначає тип мережі, у який необхідно перетворити точку;

5. система перетворює точку у точку відповідної мережі;

Альтернативні потоки:

– знайдена точка – точка стіни – команда ігнорується;

– знайдена точка – точка інженерної мережі – натомість виконується ВВ «перетворення точки відрізків користувацьких типів у точку з’єднання відрізків користувацьких типів»;

– точку не знайдено – команда ігнорується.

Заборонні потоки:

– перетворення точки неприпустиме(з’єднує різні інженерні мережі, або є точкою розгалуження, або не пов’язана з жодним відрізком, а обраний тип відрізків – стіна) – команда ігнорується.

– втрата орієнтації у просторі – команда перетворення точки ігнорується.

Гарантії:

– тип точки на плані змінено у точку відповідного типу інженерної мережі;

– візуальна модель точки у просторі доповненої реальності відповідає її новому типу.

Діаграму станів для варіанту використання «перетворення точки з’єднання відрізків користувацьких типів у точку відрізку користувацького типу» зображено на рисунку Б.14.

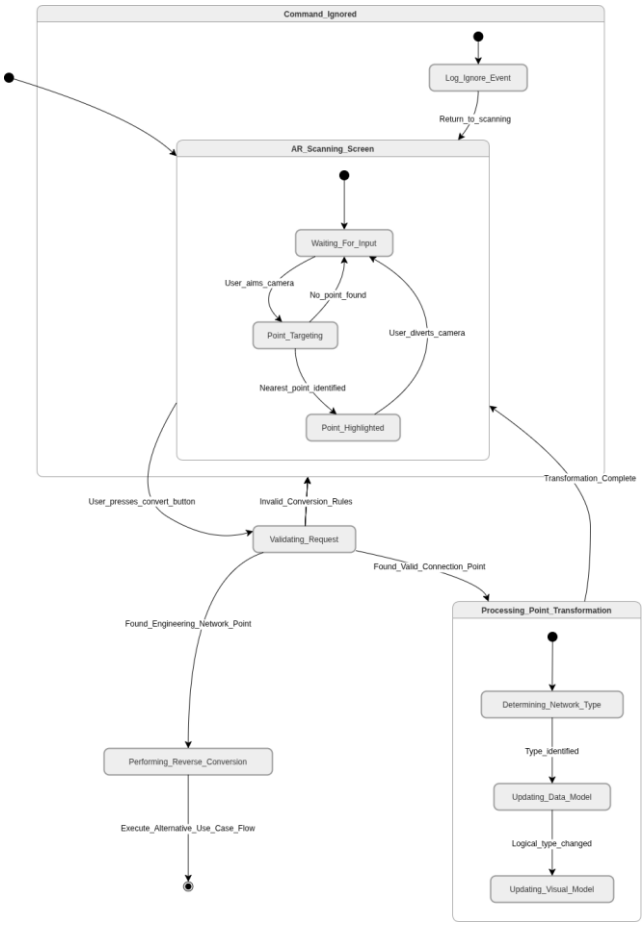


Рисунок Б.14 – діаграма станів перетворення точки з’єднання

Варіант використання: перетворення точки відрізків користувацьких типів у точку з’єднання відрізків користувацьких типів.

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

– на плані вже розміщено принаймні одну точку, що є точкою відрізків користувацького типу.

Тригер: натискання користувачем кнопки перетворення типу точки в інтерфейсі сканування.

Основний потік:

1. користувач спрямовує камеру пристрою на існуючу на плані точку інженерних мереж;

2. система візуально виділяє найближчу точку підтверджуючи готовність до взаємодії;

3. користувач натискає кнопку перетворення типу точки;

4. система перетворює точку у точку з’єднання відрізків користувацьких типів;

Альтернативні потоки:

– знайдена точка – точка стіни – команда ігнорується;

– знайдена точка – точка з’єднання інженерної мережі – натомість виконується ВВ «перетворення точки з’єднання відрізків користувацьких типів у точку відрізку користувацького типу.»;

– точку не знайдено – команда ігнорується.

Заборонні потоки:

– втрата орієнтації у просторі – команда перетворення точки ігнорується.

Гарантії:

– тип точки на плані змінено у точку з’єднання відрізків користувацьких типів;

– візуальна модель точки у просторі доповненої реальності відповідає її новому типу.

Діаграму станів для варіанту використання «перетворення точки відрізків користувацьких типів у точку з’єднання відрізків користувацьких типів» зображено на рисунку Б.15.

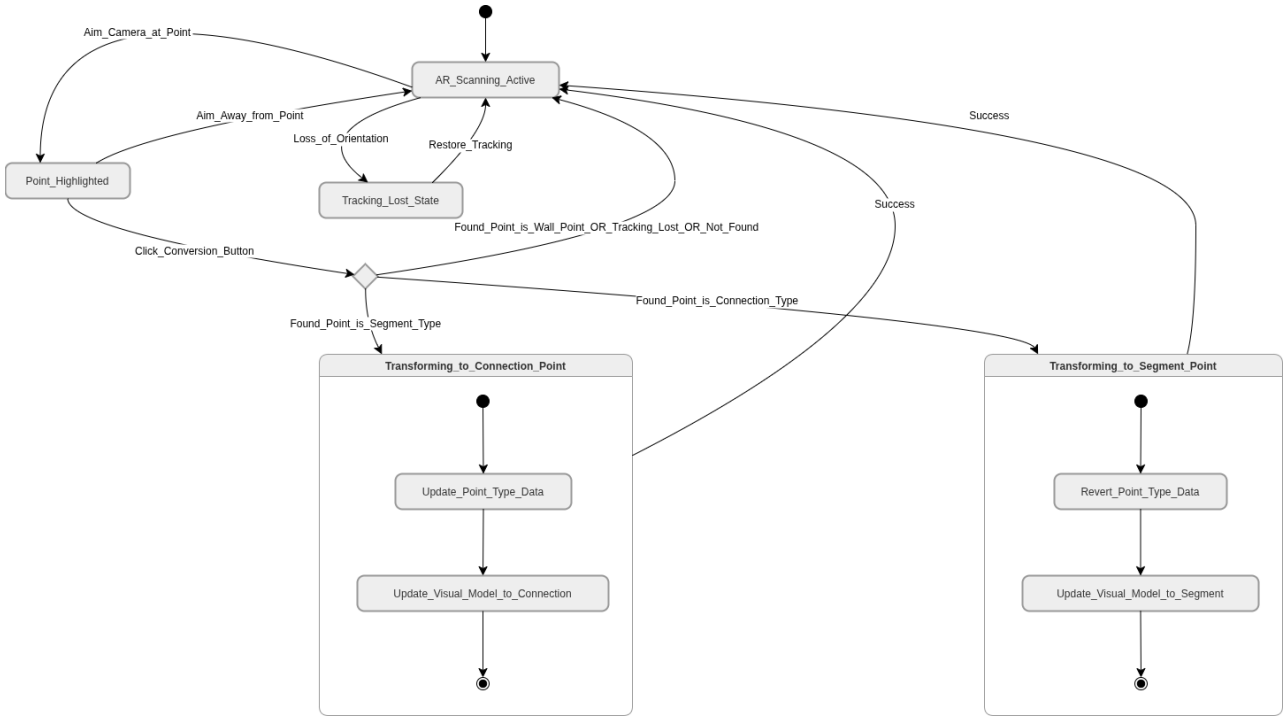


Рисунок Б.15 – діаграма станів перетворення точки відрізків інженерних мереж

Варіант використання: видалення точки.

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

– на плані створено принаймні одну точку.

Тригер: натискання користувачем кнопки видалення точки в інтерфейсі сканування.

Основний потік:

1. користувач спрямовує камеру пристрою так, щоб центральна частина екрана була наведена на раніше створену точку;

2. система візуально виділяє найближчу до центру екрана точку блиманням червоним кольором для підтвердження вибору;

3. користувач натискає кнопку видалення точки;

4. система автоматично розриває та видаляє всі лінії стін або інженерних мереж, що були з’єднані з цією точкою;

5. система видаляє обрану точку з віртуального простору;

6. система оновлює візуалізацію об’єктів у доповненій реальності на екрані пристрою.

Альтернативні потоки:

– видалення точки, що є «пов’язаною» – система також скидає вибір пов’язаної точки;

– знайдена точка – точка з’єднання інженерної мережі – натомість виконується ВВ «перетворення точки з’єднання відрізків користувацьких типів у точку відрізку користувацького типу.»;

– точку не знайдено – команда ігнорується.

Заборонні потоки:

– відсутність об'єктів для видалення – команда ігнорується;

– втрата орієнтації у просторі – команда ігнорується.

Гарантії:

– обрана точка та всі лінії, що входять до неї або виходять з неї, повністю вилучаються з плану.

Діаграму станів для варіанту використання «видалення точки» зображено на рисунку Б.16.

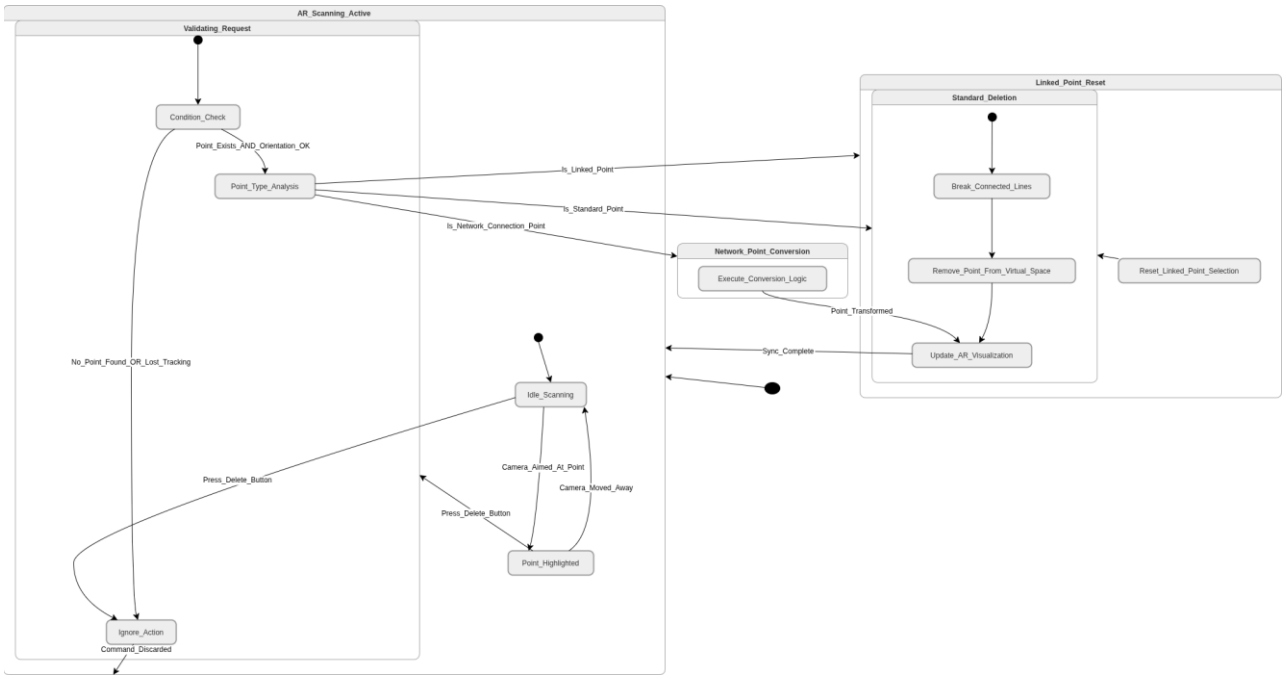


Рисунок Б.16 – діаграма станів видалення точки

Варіант використання: з’єднання точки з «пов’язаною».

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

– у просторі вже розміщено принаймні дві точки;

– одна з існуючих точок активована як «пов’язана» (візуально виділена блиманням пурпуровим кольором).

Тригер: натискання кнопки встановлення зв’язку на екрані вимірювання.

Основний потік:

1. система ідентифікує існуючу точку, яка знаходиться найближче до центру екрана;

2. система перевіряє наявність раніше встановленої або обраної «пов’язаної» точки;

3. система автоматично створює лінію між «пов’язаною» точкою та точкою в центрі екрана;

4. система оновлює статус «пов’язаної» точки, передаючи цей статус точці, що була в центрі екрана;

5. система візуалізує створену лінію у просторі доповненої реальності;

6. система розраховує та відображає на екрані актуальну відстань між новою «пов’язаною» точкою та центром екрана.

Альтернативні потоки:

– спроба з’єднати точки, між якими вже існує прямий зв’язок: система ігнорує створення лінії, але оновлює статус «пов’язаної» точки для поточної обраної точки;

– з’єднання точки з «пов’язаною» неприпустиме (з’єднання стіни з інженерною мережею, різних інженерних мереж не через спеціальну точку з’єднання або розгалуження інженерної мережі не через спеціальну точку з’єднання) – з’єднання не створюється, але «прив’язана» точка змінюється.

Заборонні потоки:

– точку не знайдено – команда ігнорується;

– не обрано «пов’язану» точку – команда ігнорується;

– втрата орієнтації у просторі – команда ігнорується.

Гарантії:

– нове з’єднання між точками додано до структури плану;

– статус «пов’язаної» точки успішно перенесено на останню обрану точку;

– створену лінію відображено в інтерфейсі.

Діаграму станів для варіанту використання «з’єднання точки з «пов’язаною» зображено на рисунку Б.17.

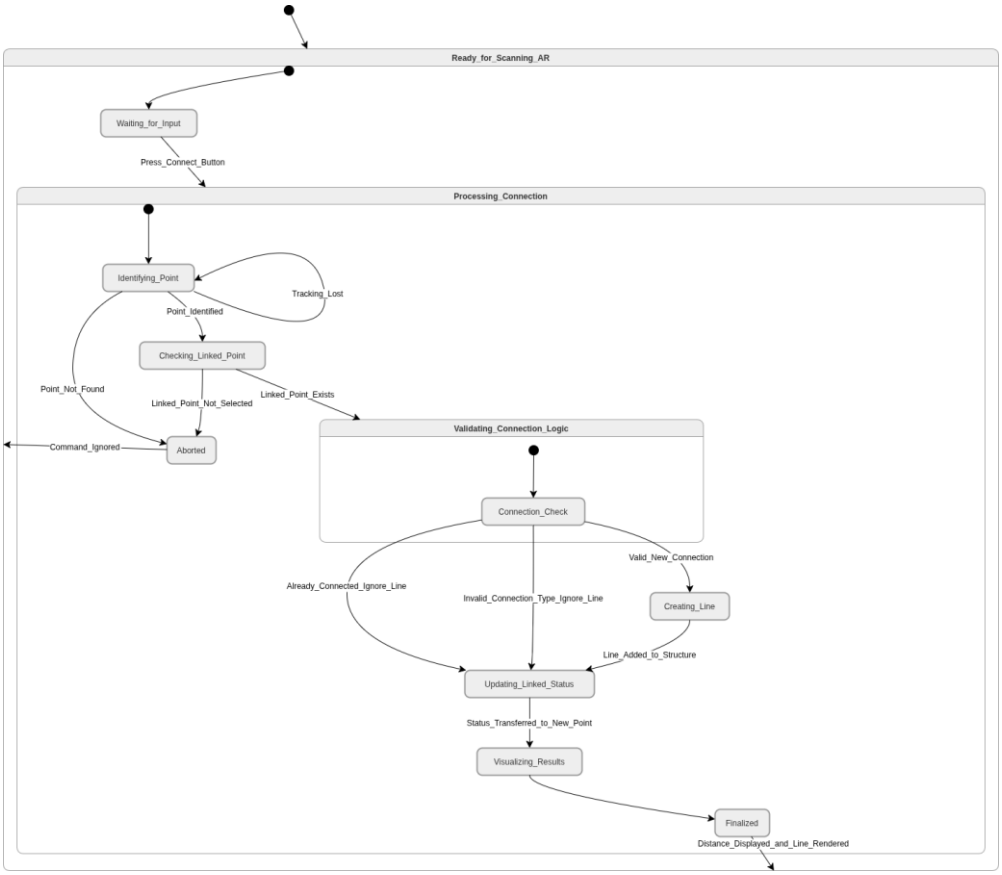


Рисунок Б.17 – діаграма станів з’єднання точки з

Варіант використання: роз’єднання точки з «пов’язаною».

Передумови:

– застосунок перебуває у режимі доповненої реальності на екрані сканування;

– система успішно виявила поверхні у навколишньому просторі;

– у просторі вже розміщено принаймні дві точки;

– одна з існуючих точок активована як «пов’язана» (візуально виділена блиманням пурпуровим кольором);

– у просторі існує побудована лінія (зв’язок), що з’єднує «пов’язану» точку з іншою точкою.

Тригер: натискання кнопки роз’єднання точок на екрані вимірювання.

Основний потік:

1. система ідентифікує існуючу точку, яка знаходиться найближче до центру екрана;

2. система перевіряє наявність раніше встановленої або обраної «пов’язаної» точки;

3. система виконує пошук прямого зв'язку між активною точкою в центрі екрана та поточною «пов’язаною» точкою;

4. система видаляє лінію зв'язку між цими двома об'єктами;

5. система оновлює графічне відображення AR-простору, прибираючи модель лінії;

6. система оновлює статус точок, призначаючи точку, що була в центрі екрана, новою «пов’язаною» точкою.

Альтернативні потоки:

– спроба роз’єднати точки, між якими не існує прямий зв’язок: система ігнорує видалення лінії, але оновлює статус «пов’язаної» точки для поточної обраної точки;

Заборонні потоки:

– точку не знайдено – команда ігнорується;

– не обрано «пов’язану» точку – команда ігнорується;

– втрата орієнтації у просторі – команда ігнорується.

Гарантії:

– з структури плану прибрано з’єднання між точками;

– статус «пов’язаної» точки успішно перенесено на останню обрану точку;

– цілісність інших елементів плану зберігається.

Діаграму станів для варіанту використання «роз’єднання точки з «пов’язаною» зображено на рисунку Б.18.

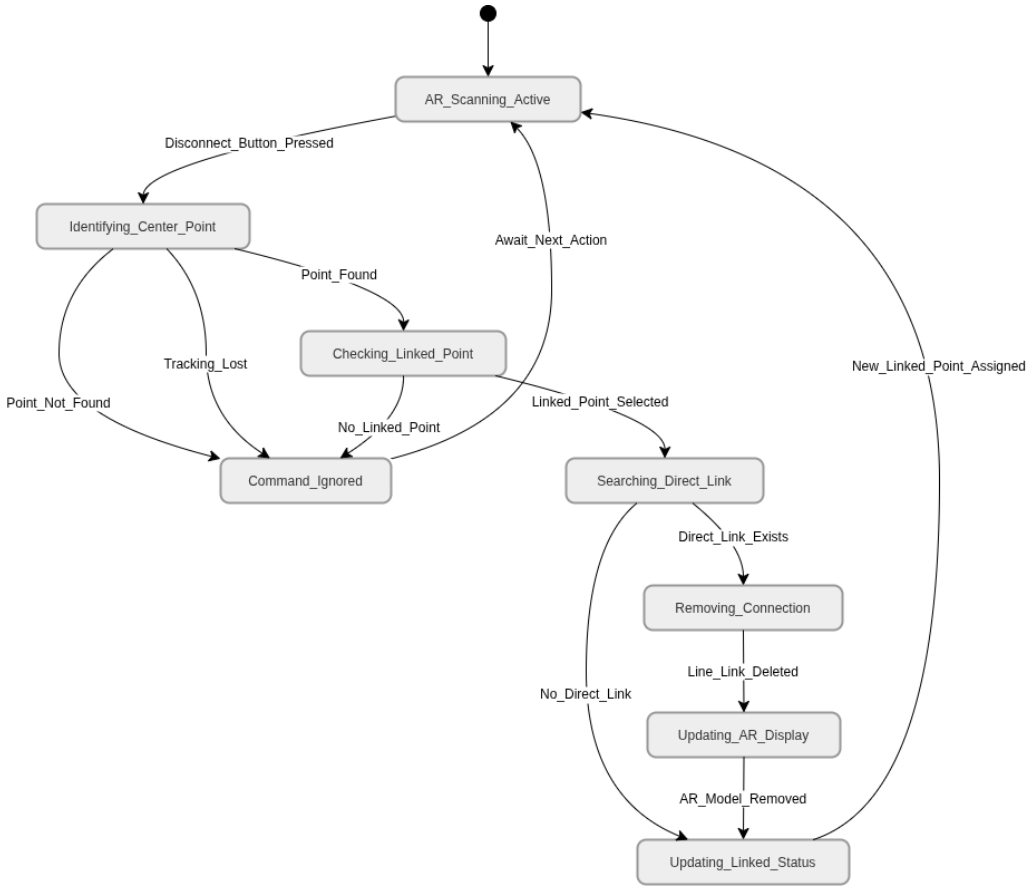


Рисунок Б.18 – діаграма станів роз’єднання точки з «пов’язаною»

Варіант використання: формування звіту про відрізки користувацьких типів.

Передумови:

– користувач відкрив екран перегляду існуючого плану приміщення;

– у системі визначено принаймні один тип інженерних мереж;

Тригер: натискання користувачем кнопки керування типами інженерних мереж на екрані плану приміщення.

Основний потік:

1. система відкриває екран переліку типів інженерних мереж та відображає всі доступні типи;

2. для кожного типу мережі система автоматично аналізує наявні на плані лінії;

3. система розраховує фізичну довжину кожного відрізка на основі тривимірних координат зафіксованих точок;

4. система виконує автоматичне групування відрізків: суміжні відрізки одного типу, що з’єднуються через рядові точки (які не є вузлами з’єднання), об’єднуються у цілісні сегменти;

5. система обчислює загальну сумарну довжину всіх відрізків для кожного типу інженерної мережі;

6. система сортує сегменти за довжиною у порядку спадання.

7. система формує візуальний список, де для кожного типу відображається його назва, загальна довжина та деталізований перелік довжин усіх сформованих сегментів.

Альтернативні потоки:

– план не містить даних – розрахунки не проводяться, список сегментів відображається порожнім, загальна довжина відображається 0.

Заборонні потоки:

– помилка доступу до сховища: у разі неможливості зчитати план система повертає користувача на екран плану.

Гарантії:

– користувач бачить актуальні розрахунки протяжності інженерних комунікацій, згруповані за їхніми типами.

Діаграму станів для варіанту використання «формування звіту про відрізки користувацьких типів» зображено на рисунку Б.19.

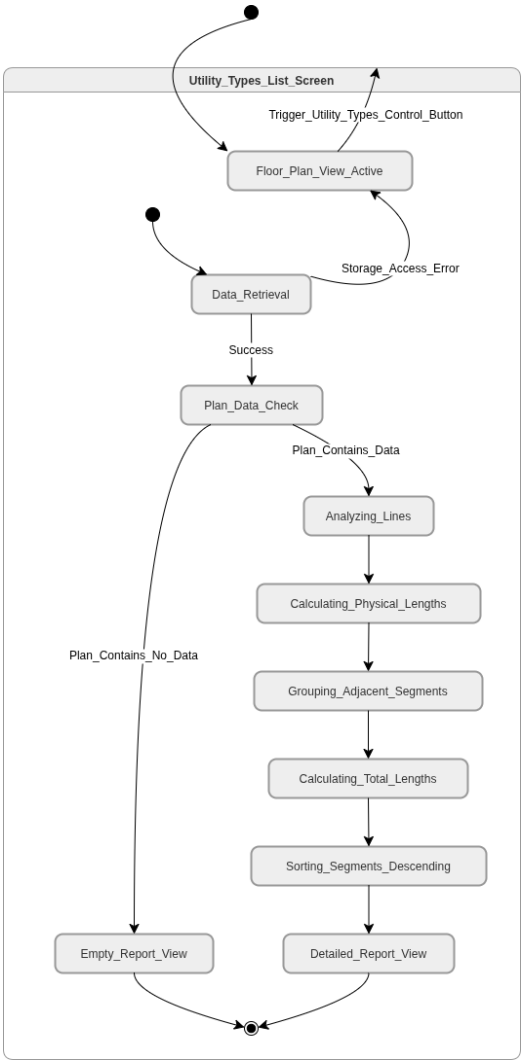


Рисунок Б.19 – діаграма станів формування звіту про інженерні

Варіант використання: експорт плану поверху у вигляді PNG зображення.

Передумови:

– користувач знаходиться на екрані перегляду конкретного плану поверху;

– план поверху містить зафіксовані дані вимірювань стін або об'єктів інженерних мереж.

Тригер: натискання користувачем кнопки експорту плану на екрані перегляду.

Основний потік:

1. користувач активує функцію збереження за допомогою кнопки експорту;
2. система відкриває системне діалогове вікно для створення документа;
3. користувач вводить назву файлу та обирає директорію для збереження;

4. система автоматично обчислює масштаб плану для забезпечення чіткості всіх елементів на зображенні;

5. система формує графічне представлення плану, що включає лінії стін, лінії інженерних мереж різних типів та текстові позначення відстаней між точками;

6. система генерує файл у форматі PNG;

7. система виконує запис даних у обране користувачем місце.

Альтернативні потоки:

– користувач скасовує операцію у системному вікні вибору місця збереження – система закриває вікно вибору та повертає користувача до екрана перегляду плану без створення файлу.

Заборонні потоки:

– якщо план не містить зафіксованих точок або ліній, система відображає повідомлення про відсутність даних для експорту та припиняє процес.

Гарантії:

– план поверху залишається незмінним та доступним для подальшого редагування;

– створене зображення точно відображає геометрію та числові параметри виміряних точок та ліній.

Діаграму станів для варіанту використання «експорт плану поверху у вигляді PNG зображення» зображено на рисунку Б.20.

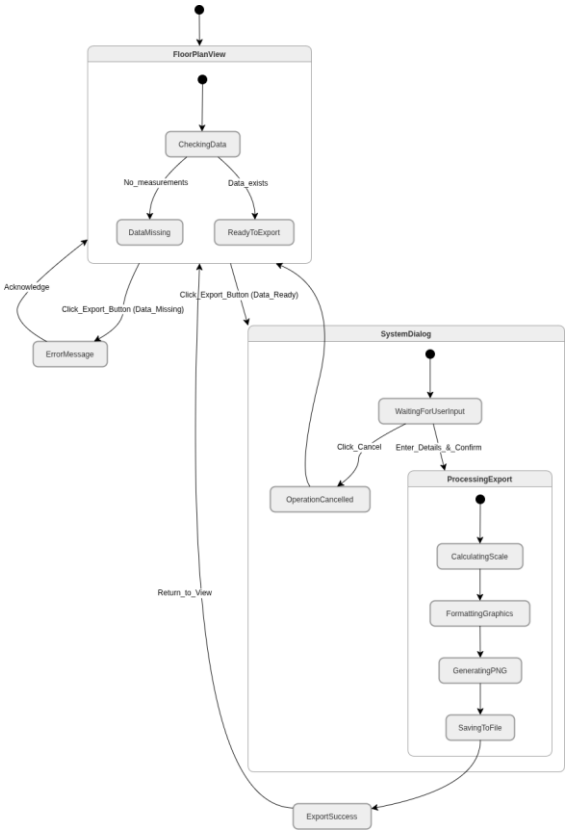


Рисунок Б.20 – діаграма станів експорту у PNG

ДОДАТОК В
(обов'язковий)

ЛІСТИНГ КОДУ ПРОГРАМИ

В1. Код автоматизованих тестів.

```
package com.google.ar.core.examples.java.helloar;

import android.content.Context;
import android.content.Intent;

import androidx.test.core.app.ApplicationProvider;
import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.platform.app.InstrumentationRegistry;
import androidx.test.uiautomator.By;
import androidx.test.uiautomator.UiDevice;
import androidx.test.uiautomator.UiObject2;
import androidx.test.uiautomator.Until;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;

@RunWith(AndroidJUnit4.class)
public class UITests {

    private static final String APP_PACKAGE = "com.google.ar.core.examples.java.helloar";
    private static final int TIMEOUT_MS = 5000;

    private UiDevice device;

    @Before
    public void setUp() throws Exception {
        // Initialize UiDevice instance
        device = UiDevice.getInstance(InstrumentationRegistry.getInstrumentation());

        // Start from the home screen
        device.pressHome();

        // Clear app data to ensure a clean state for each test
        device.executeShellCommand("pm clear " + APP_PACKAGE);

        // Wait for the launcher
        final String launcherPackage = device.getLauncherPackageName();
        assertNotNull(launcherPackage);
        device.wait(Until.hasObject(By.pkg(launcherPackage).depth(0)), TIMEOUT_MS);

        // Launch the app
        Context context = ApplicationProvider.getApplicationContext();
        Intent intent =
context.getPackageManager().getLaunchIntentForPackage(APP_PACKAGE);
        assertNotNull("Failed to find Intent to launch the application", intent);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
        context.startActivity(intent);

        // Wait for the app to appear
        device.wait(Until.hasObject(By.pkg(APP_PACKAGE).depth(0)), TIMEOUT_MS);
```

```
    }

    @Test
    public void testCreateNewFloorPlan() {
        final String expectedPlanName = "Plan_123";

        // Click the create plan button
        UiObject2 createActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Plan creation button not found", createActivityButton);
        createActivityButton.click();

        // Enter the plan name
        UiObject2 nameInputField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")),
                TIMEOUT_MS
        );
        assertNotNull("Plan name input field not found", nameInputField);
        nameInputField.setText(expectedPlanName);

        // Confirm creation
        UiObject2 confirmCreateButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")),
                TIMEOUT_MS
        );
        assertNotNull("Create confirmation button not found", confirmCreateButton);
        confirmCreateButton.click();

        // Verify the plan details screen is loaded
        UiObject2 planNameTitle = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextText")),
                TIMEOUT_MS
        );
        assertNotNull("Plan details screen failed to load", planNameTitle);
        assertEquals("Plan name is displayed incorrectly", expectedPlanName,
planNameTitle.getText());

        // Return to the main list
        device.pressBack();

        // Verify the plan is in the list
        UiObject2 savedPlanItem = device.wait(
                Until.findObject(By.desc("plan_name_" + expectedPlanName)),
                TIMEOUT_MS
        );
        assertNotNull("Created plan is missing from the main list", savedPlanItem);
    }

    @Test
    public void testOpenExistingFloorPlan() {
        final String expectedPlanName = "Test_Plan_To_Open";

        // Create a plan first to open it later
        UiObject2 createActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Plan creation button not found", createActivityButton);
        createActivityButton.click();

        UiObject2 nameInputField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")),
                TIMEOUT_MS
        );
        assertNotNull("Plan name input field not found", nameInputField);
        nameInputField.setText(expectedPlanName);

        UiObject2 confirmCreateButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")),
```

```
                TIMEOUT_MS
        );
        assertNotNull("Create confirmation button not found", confirmCreateButton);
        confirmCreateButton.click();

        device.wait(Until.findObject(By.res(APP_PACKAGE, "editTextText")), TIMEOUT_MS);
        device.pressBack();

        // Find the created plan in the list
        UiObject2 planListItem = device.wait(
                Until.findObject(By.desc("plan_name_" + expectedPlanName)),
                TIMEOUT_MS
        );
        assertNotNull("Created plan is missing from the main list", planListItem);

        // Click on the plan to open it
        planListItem.click();

        // Verify the screen (FloorPlanActivity) has loaded
        UiObject2 arActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "ar_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Opened plan screen (FloorPlanActivity) failed to load",
arActivityButton);

        // Check the title matches
        UiObject2 planNameTitle = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextText")),
                TIMEOUT_MS
        );
        assertNotNull("Plan name field not found on details screen", planNameTitle);
        assertEquals("Plan with incorrect name was opened", expectedPlanName,
planNameTitle.getText());
    }

    @Test
    public void testDeleteFloorPlan() {
        final String expectedPlanName = "Plan_To_Delete";

        // Create a plan to delete
        UiObject2 createActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Plan creation button not found", createActivityButton);
        createActivityButton.click();

        UiObject2 nameInputField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")),
                TIMEOUT_MS
        );
        assertNotNull("Plan name input field not found", nameInputField);
        nameInputField.setText(expectedPlanName);

        UiObject2 confirmCreateButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")),
                TIMEOUT_MS
        );
        assertNotNull("Create confirmation button not found", confirmCreateButton);
        confirmCreateButton.click();

        device.wait(Until.findObject(By.res(APP_PACKAGE, "editTextText")), TIMEOUT_MS);
        device.pressBack();

        UiObject2 planListItem = device.wait(
                Until.findObject(By.desc("plan_name_" + expectedPlanName)),
                TIMEOUT_MS
        );
        assertNotNull("Created plan is missing from the main list, cannot proceed with
test", planListItem);
```

```
    // Find and click the delete button for the plan
    UiObject2 deleteButton = device.wait(
            Until.findObject(By.desc("plan_delete_" + expectedPlanName)),
            TIMEOUT_MS
    );
    assertNotNull("Delete button for the created plan not found", deleteButton);

    deleteButton.click();

    // Verify the plan is gone
    boolean isDeleted = device.wait(
            Until.gone(By.desc("plan_name_" + expectedPlanName)),
            TIMEOUT_MS
    );

    assertTrue("Plan item is still displayed in the list after deletion", isDeleted);
}

@Test
public void testRenameFloorPlan() {
    final String initialPlanName = "Initial_Plan_Name";
    final String updatedPlanName = "Updated_Plan_Name";

    // Create a plan
    UiObject2 createActivityButton = device.wait(
            Until.findObject(By.res(APP_PACKAGE, "create_activity_button")),
            TIMEOUT_MS
    );
    assertNotNull("Plan creation button not found", createActivityButton);
    createActivityButton.click();

    UiObject2 createNameField = device.wait(
            Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")),
            TIMEOUT_MS
    );
    assertNotNull("Plan name input field not found", createNameField);
    createNameField.setText(initialPlanName);

    UiObject2 confirmCreateButton = device.wait(
            Until.findObject(By.res(APP_PACKAGE, "create_button")),
            TIMEOUT_MS
    );
    assertNotNull("Create confirmation button not found", confirmCreateButton);
    confirmCreateButton.click();

    device.wait(Until.findObject(By.res(APP_PACKAGE, "editTextText")), TIMEOUT_MS);
    device.pressBack();

    UiObject2 planListItem = device.wait(
            Until.findObject(By.desc("plan_name_" + initialPlanName)),
            TIMEOUT_MS
    );
    assertNotNull("Initial plan is missing from the main list", planListItem);

    // Rename the plan
    planListItem.click();

    UiObject2 editNameField = device.wait(
            Until.findObject(By.res(APP_PACKAGE, "editTextText")),
            TIMEOUT_MS
    );
    assertNotNull("Plan name field not found on edit screen", editNameField);

    editNameField.clear();
    editNameField.setText(updatedPlanName);
    device.pressBack();

    // Verify the old name is gone and the new one exists
    boolean isOldNameGone = device.wait(
            Until.gone(By.desc("plan_name_" + initialPlanName)),
```

```
                TIMEOUT_MS
        );
        assertTrue("Item with the old name is still displayed", isOldNameGone);

        UiObject2 renamedPlanListItem = device.wait(
                Until.findObject(By.desc("plan_name_" + updatedPlanName)),
                TIMEOUT_MS
        );
        assertNotNull("Plan with the new (updated) name not found in the list",
renamedPlanListItem);
    }

    @Test
    public void testCreateCustomCableType() {
        final String expectedCableTypeName = "Fiber_optic_cable";

        // Go to cable types list
        UiObject2 cableTypeListActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "cabletypelist_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Button to navigate to cable types list not found",
cableTypeListActivityButton);
        cableTypeListActivityButton.click();

        // Click create new cable type
        UiObject2 createActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE,
"create_cabletype_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Add new cable type button not found", createActivityButton);
        createActivityButton.click();

        // Enter name
        UiObject2 nameInputField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")),
                TIMEOUT_MS
        );
        assertNotNull("Cable type name input field not found", nameInputField);
        nameInputField.setText(expectedCableTypeName);

        // Confirm creation
        UiObject2 confirmCreateButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")),
                TIMEOUT_MS
        );
        assertNotNull("Create confirmation button not found", confirmCreateButton);
        confirmCreateButton.click();

        // Verify elements are present
        UiObject2 deleteButtonForNewType = device.wait(
                Until.findObject(By.desc("delete_button_" + expectedCableTypeName)),
                TIMEOUT_MS
        );

        UiObject2 typeNameTextView = device.wait(
                Until.findObject(By.text(expectedCableTypeName)),
                TIMEOUT_MS
        );

        assertNotNull("Delete button for the new cable type not generated (type not
saved)", deleteButtonForNewType);
        assertNotNull("Text identifier for the new cable type is missing from the
screen", typeNameTextView);
    }

    @Test
    public void testDeleteCustomCableType() {
        final String expectedCableTypeName = "Cable_For_Deletion";
```

```
        // Go to cable types list
        UiObject2 cableTypeListActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "cabletypelist_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Button to navigate to cable types list not found",
cableTypeListActivityButton);
        cableTypeListActivityButton.click();

        // Create a type to delete
        UiObject2 createActivityButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE,
"create_cabletype_activity_button")),
                TIMEOUT_MS
        );
        assertNotNull("Add new cable type button not found", createActivityButton);
        createActivityButton.click();

        UiObject2 nameInputField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")),
                TIMEOUT_MS
        );
        assertNotNull("Cable type name input field not found", nameInputField);
        nameInputField.setText(expectedCableTypeName);

        UiObject2 confirmCreateButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")),
                TIMEOUT_MS
        );
        assertNotNull("Create confirmation button not found", confirmCreateButton);
        confirmCreateButton.click();

        // Find and click delete button
        String deleteButtonDesc = "delete_button_" + expectedCableTypeName;
        UiObject2 deleteButton = device.wait(
                Until.findObject(By.desc(deleteButtonDesc)),
                TIMEOUT_MS
        );
        assertNotNull("Delete button for the created cable type not found (possibly not
saved)", deleteButton);

        deleteButton.click();

        // Verify it's gone
        boolean isDeleted = device.wait(
                Until.gone(By.desc(deleteButtonDesc)),
                TIMEOUT_MS
        );

        assertTrue("Cable type item is still displayed in the list after clicking the
delete button", isDeleted);
    }

    @Test
    public void verifyAdditionOfCustomCableTypeToPlan() {
        // Create plan
        UiObject2 createPlanButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_activity_button")),
TIMEOUT_MS);
        assertNotNull("Plan creation button not found", createPlanButton);
        createPlanButton.click();

        UiObject2 planNameField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")), TIMEOUT_MS);
        assertNotNull("Plan name input field not found", planNameField);
        planNameField.setText("Research Plan");

        UiObject2 confirmPlanCreationButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")), TIMEOUT_MS);
        confirmPlanCreationButton.click();
```

```
        // Open cable types
        UiObject2 cableTypesButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "cableTypes_button")), TIMEOUT_MS);
        assertNotNull("Cable types button not found", cableTypesButton);
        cableTypesButton.click();

        // Create new type
        UiObject2 createCableTypeButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE,
"create_cabletype_activity_button")), TIMEOUT_MS);
        assertNotNull("Add new cable type button not found", createCableTypeButton);
        createCableTypeButton.click();

        String customCableTypeName = "Optical conductor";
        UiObject2 cableTypeNameField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")), TIMEOUT_MS);
        assertNotNull("Cable type name input field not found", cableTypeNameField);
        cableTypeNameField.setText(customCableTypeName);

        UiObject2 confirmCableCreationButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")), TIMEOUT_MS);
        confirmCableCreationButton.click();

        // Find and check the checkbox
        UiObject2 cableTypeCheckbox = device.wait(
                Until.findObject(By.desc("type_checkbox_" + customCableTypeName)),
TIMEOUT_MS);
        assertNotNull("Created cable type did not appear in the list",
cableTypeCheckbox);

        if (!cableTypeCheckbox.isChecked()) {
            cableTypeCheckbox.click();
            device.waitForIdle();
        }

        assertTrue("Custom segment type must be activated for the plan",
                cableTypeCheckbox.isChecked());
    }

    @Test
    public void verifyExclusionOfCustomCableTypeFromPlan() {
        // Create plan
        UiObject2 createPlanButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_activity_button")),
TIMEOUT_MS);
        assertNotNull("Plan creation button not found", createPlanButton);
        createPlanButton.click();

        UiObject2 planNameField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")), TIMEOUT_MS);
        assertNotNull("Plan name input field not found", planNameField);
        planNameField.setText("Research Plan");

        UiObject2 confirmPlanCreationButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")), TIMEOUT_MS);
        confirmPlanCreationButton.click();

        // Open cable types
        UiObject2 cableTypesButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "cableTypes_button")), TIMEOUT_MS);
        assertNotNull("Cable types button not found", cableTypesButton);
        cableTypesButton.click();

        // Create new type
        UiObject2 createCableTypeButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE,
"create_cabletype_activity_button")), TIMEOUT_MS);
        assertNotNull("Add new cable type button not found", createCableTypeButton);
        createCableTypeButton.click();

        String customCableTypeName = "Copper cable";
```

```
        UiObject2 cableTypeNameField = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "editTextCreateName")), TIMEOUT_MS);
        assertNotNull("Cable type name input field not found", cableTypeNameField);
        cableTypeNameField.setText(customCableTypeName);

        UiObject2 confirmCableCreationButton = device.wait(
                Until.findObject(By.res(APP_PACKAGE, "create_button")), TIMEOUT_MS);
        confirmCableCreationButton.click();

        UiObject2 cableTypeCheckbox = device.wait(
                Until.findObject(By.desc("type_checkbox_" + customCableTypeName)),
TIMEOUT_MS);
        assertNotNull("Created cable type did not appear in the list",
cableTypeCheckbox);

        // Enable it first
        if (!cableTypeCheckbox.isChecked()) {
            cableTypeCheckbox.click();
            device.waitForIdle();
        }
        assertTrue("Setup error: cable type must be enabled before disabling",
                cableTypeCheckbox.isChecked());

        // Now disable it
        cableTypeCheckbox.click();
        device.waitForIdle();
        assertFalse("Custom segment type should be successfully excluded from the plan",
                cableTypeCheckbox.isChecked());
    }
}
```

В2. Код сутності типу точки з’єднання мереж.

```
package com.google.ar.core.examples.java.helloar;

public class CableConnectionPointType implements IPointType{
    public boolean isWallPoint(){
        return false;
    }
    public boolean isCableConnectionPoint(){
        return true;
    }
    public boolean isCablePoint(){
        return false;
    }
    public Integer getCableTypeId(){
        return null;
    }
}
```

В3. Код сутності типу відрізку прокладання мереж.

```
package com.google.ar.core.examples.java.helloar;

public class CableLineType implements ILineType{
    int _cableTypeId;

    public CableLineType(int cableTypeId){
        _cableTypeId = cableTypeId;
    }

    public boolean isWall(){
        return false;
    }
    public boolean isCable(){
        return true;
    }
    public Integer getCableTypeId(){
        return _cableTypeId;
    }
```

```
}
```

В4. Код сутності типу точки прокладання мереж.

```
package com.google.ar.core.examples.java.helloar;

public class CablePointType implements IPointType{
    int _cableTypeId;

    public CablePointType(int cableTypeId){
        _cableTypeId = cableTypeId;
    }

    public boolean isWallPoint(){
        return false;
    }
    public boolean isCableConnectionPoint(){
        return false;
    }
    public boolean isCablePoint(){
        return true;
    }
    public Integer getCableTypeId(){
        return _cableTypeId;
    }
}
```

В5. Код сутності типу інженерної мережі.

```
package com.google.ar.core.examples.java.helloar;

import android.content.Context;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class CableType implements Serializable {
    private int _id;
    private String _name = "";

    public CableType(int id){
        _id = id;
    }

    public CableType(int id, String name){
        _id = id;
        _name = name;
    }

    public int getId(){
        return _id;
    }

    public String getName(){
        return _name;
    }

    public void setName(String newName){
        _name = newName;
    }

    public boolean Save(Context context){
        File file = new File(context.getFilesDir(), "ct_"+_id);
        if (file.exists()) {
            file.delete();
```

```
        }
        try {
            file.createNewFile();
            FileOutputStream fos = new FileOutputStream(file);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(this);
            fos.close();
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    public static CableType Load(Context context, int id){
        File file = new File(context.getFilesDir(), "ct_"+id);
        if(file.exists()) {
            try {
                FileInputStream fis = new FileInputStream(file);
                ObjectInputStream ois = new ObjectInputStream(fis);
                Object result = ois.readObject();
                if(result.getClass() == CableType.class) {
                    return (CableType) result;
                }
            } catch (Exception e) {
            }
        }
        return null;
    }

    public static void Delete(Context context, int id){
        File file = new File(context.getFilesDir(), "ct_"+id);
        if(file.exists()) {
            file.delete();
        }
    }
}
```

В6. Код контролера списку типів інженерних мереж.

```
package com.google.ar.core.examples.java.helloar;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.fonts.Font;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.WindowManager;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.io.File;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Arrays;

public class CableTypeListActivity extends AppCompatActivity {
    private static final int CREATE = 2;
    private static final int EDIT = 123;
    private CableType[] cableTypes;

    private Font nameFont;
    final Handler handler = new Handler();
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cabletypelist);
        getWindow().setAttributes(new
WindowManager.LayoutParams(WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER
));

        ImageButton createActivityButton =
findViewById(R.id.create_cabletype_activity_button);
        createActivityButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        callCreateView();
                    }
                });

        loadCableTypelist();
    }

    @Override
    protected void onResume() {
        super.onResume();
        loadCableTypelist();
    }

    private void loadCableTypelist(){
        LinearLayout floorPlanList = findViewById(R.id.floor_plan_list);
        floorPlanList.removeAllViews();
        Context context = getApplicationContext();

        IdStorage idStorage = IdStorage.Load(context);
        if(idStorage == null)
            idStorage = new IdStorage();

        int[] cableTypeIds = idStorage.getCableTypeIds();
        cableTypes = new CableType[cableTypeIds.length];
        for(int i = 0; i < cableTypeIds.length; i++){
            cableTypes[i] = CableType.Load(context, cableTypeIds[i]);
        }
        cableTypes = Arrays.stream(cableTypes).filter(fp -> fp != null).toArray(size ->
new CableType[size]);
        int i = 0;
        for(CableType cableType : cableTypes) {
            View tv = createViewForType(context, cableType.getName(), cableType.getId());
            floorPlanList.addView(tv);
            i++;
        }
    }

    private View createViewForType(Context context, String name, int id){
        RelativeLayout layout = new RelativeLayout(context);

        TextView tv = new TextView(context);
        tv.setId(View.generateViewId());
        tv.setText(name);
        tv.setTextColor(Color.WHITE);
        tv.setTextSize(20);
        tv.setPadding(0, 25, 0, 25);

        RelativeLayout.LayoutParams tvRelativeParams = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.MATCH_PARENT);
        tvRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_LEFT);
        layout.addView(tv, tvRelativeParams);

        ImageButton editButton = new ImageButton(context);
        editButton.setId(View.generateViewId());
```

```
        ImageButton deleteButton = new ImageButton(context);
        deleteButton.setId(View.generateViewId());
        deleteButton.setImageResource(R.drawable.trash2);
        deleteButton.setScaleType(ImageView.ScaleType.FIT_CENTER);
        deleteButton.setBaselineAlignBottom(false);
        deleteButton.setBackgroundColor(0x00ffffff);
        deleteButton.setContentDescription("delete_button_"+name);
        deleteButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                IdStorage idStorage = IdStorage.Load(context);
                if(idStorage == null)
                    idStorage = new IdStorage();

                for(int floorPlanId : idStorage.getFloorPlanIds()){
                    FloorPlan fp = FloorPlan.Load(context, floorPlanId);
                    if(fp == null)
                        continue;

                    fp.removeCableTypeId(id);
                    fp.Save(context);
                }

                idStorage.removeCableTypeId(id);
                CableType.Delete(getApplicationContext(), id);
                loadCableTypelist();
            }
        });
        RelativeLayout.LayoutParams dbRelativeParams = new
RelativeLayout.LayoutParams(125, RelativeLayout.LayoutParams.MATCH_PARENT);
        dbRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
        dbRelativeParams.addRule(RelativeLayout.ALIGN_TOP, tv.getId());
        dbRelativeParams.addRule(RelativeLayout.ALIGN_BOTTOM, tv.getId());
        layout.addView(deleteButton, dbRelativeParams);

        editButton.setImageResource(R.drawable.pencilline);
        editButton.setBaselineAlignBottom(false);
        editButton.setBackgroundColor(0x00ffffff);
        editButton.setScaleType(ImageView.ScaleType.FIT_CENTER);
        editButton.setContentDescription("edit_button_"+name);
        editButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                callEditView(id);
            }
        });
        RelativeLayout.LayoutParams ebRelativeParams = new
RelativeLayout.LayoutParams(125, RelativeLayout.LayoutParams.MATCH_PARENT);
        ebRelativeParams.addRule(RelativeLayout.LEFT_OF, deleteButton.getId());
        ebRelativeParams.addRule(RelativeLayout.ALIGN_TOP, tv.getId());
        ebRelativeParams.addRule(RelativeLayout.ALIGN_BOTTOM, tv.getId());
        layout.addView(editButton, ebRelativeParams);
        return layout;
    }

    private void callCreateView(){
        Intent intent = new Intent(CableTypeListActivity.this,
CreateCableTypeActivity.class);
        startActivityForResult(intent, CREATE);
    }

    private void callEditView(int id){
        Intent intent = new Intent(CableTypeListActivity.this,
EditCableTypeActivity.class);
        intent.putExtra("ID", id);
        startActivityForResult(intent, EDIT);
    }
}
```

В7. Код контролера екрану сканування.

```
/*
 * Copyright 2017 Google LLC, 2025-2026 Mykolai Drozdov
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.google.ar.core.examples.java.helloar;

import android.content.Context;
import android.content.Intent;
import android.media.Image;
import android.opengl.GLSurfaceView;
import android.opengl.Matrix;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.util.DisplayMetrics;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.google.ar.core.Anchor;
import com.google.ar.core.ArCoreApk;
import com.google.ar.core.ArCoreApk.Availability;
import com.google.ar.core.Camera;
import com.google.ar.core.Config;
import com.google.ar.core.Config.InstantPlacementMode;
import com.google.ar.core.Frame;
import com.google.ar.core.HitResult;
import com.google.ar.core.InstantPlacementPoint;
import com.google.ar.core.LightEstimate;
import com.google.ar.core.Plane;
import com.google.ar.core.PointCloud;
import com.google.ar.core.Pose;
import com.google.ar.core.Session;
import com.google.ar.core.Trackable;
import com.google.ar.core.TrackingFailureReason;
import com.google.ar.core.TrackingState;
import com.google.ar.core.examples.java.common.helpers.CameraPermissionHelper;
import com.google.ar.core.examples.java.common.helpers.DepthSettings;
import com.google.ar.core.examples.java.common.helpers.DisplayRotationHelper;
import com.google.ar.core.examples.java.common.helpers.FullScreenHelper;
import com.google.ar.core.examples.java.common.helpers.SnackbarHelper;
import com.google.ar.core.examples.java.common.helpers.TrackingStateHelper;
import com.google.ar.core.examples.java.common.samplerender.Framebuffer;
import com.google.ar.core.examples.java.common.samplerender.Mesh;
import com.google.ar.core.examples.java.common.samplerender.SampleRender;
import com.google.ar.core.examples.java.common.samplerender.Shader;
import com.google.ar.core.examples.java.common.samplerender.Texture;
import com.google.ar.core.examples.java.common.samplerender.VertexBuffer;
import com.google.ar.core.examples.java.common.samplerender.arcore.BackgroundRenderer;
import com.google.ar.core.examples.java.common.samplerender.arcore.PlaneRenderer;
```

```
import com.google.ar.core.examples.java.common.samplerender.arcore.SpecularCubemapFilter;
import com.google.ar.core.exceptions.CameraNotAvailableException;
import com.google.ar.core.exceptions.NotYetAvailableException;
import com.google.ar.core.exceptions.UnavailableApkTooOldException;
import com.google.ar.core.exceptions.UnavailableArcoreNotInstalledException;
import com.google.ar.core.exceptions.UnavailableDeviceNotCompatibleException;
import com.google.ar.core.exceptions.UnavailableSdkTooOldException;
import com.google.ar.core.exceptions.UnavailableUserDeclinedInstallationException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;

public class CameraActivity extends AppCompatActivity implements SampleRender.Renderer {

  private static final String TAG = CameraActivity.class.getSimpleName();

  private static final String SEARCHING_PLANE_MESSAGE = "Searching for surfaces...";
  private static final String WAITING_FOR_TAP_MESSAGE = "Tap on a surface to place an
object.";

  // See the definition of updateSphericalHarmonicsCoefficients for an explanation of
these
  // constants.
  private static final float[] sphericalHarmonicFactors = {
          0.282095f,
          -0.325735f,
          0.325735f,
          -0.325735f,
          0.273137f,
          -0.273137f,
          0.078848f,
          -0.273137f,
          0.136569f,
  };

  private static final float Z_NEAR = 0.1f;
  private static final float Z_FAR = 100f;

  private static final int CUBEMAP_RESOLUTION = 16;
  private static final int CUBEMAP_NUMBER_OF_IMPORTANCE_SAMPLES = 32;

  // Rendering. The Renderers are created here, and initialized when the GL surface is
created.
  private GLSurfaceView surfaceView;

  private boolean installRequested;

  private Session session;
  private final SnackbarHelper messageSnackbarHelper = new SnackbarHelper();
  private DisplayRotationHelper displayRotationHelper;
  private final TrackingStateHelper trackingStateHelper = new TrackingStateHelper(this);
  private SampleRender render;

  private PlaneRenderer planeRenderer;
  private BackgroundRenderer backgroundRenderer;
  private Framebuffer virtualSceneFramebuffer;
  private boolean hasSetTextureNames = false;

  private final DepthSettings depthSettings = new DepthSettings();
  // Assumed distance from the device camera to the surface on which user will try to
place objects.
  // This value affects the apparent scale of objects while the tracking method of the
  // Instant Placement point is SCREENSPACE_WITH_APPROXIMATE_DISTANCE.
  // Values in the [0.2, 2.0] meter range are a good choice for most AR experiences. Use
lower
  // values for AR experiences where users are expected to place objects on surfaces
close to the
  // camera. Use larger values for experiences where the user will likely be standing and
trying to
```

```
  // place an object on the ground or floor in front of them.
  private static final float APPROXIMATE_DISTANCE_METERS = 2.0f;

  // Point Cloud
  private VertexBuffer pointCloudVertexBuffer;
  private Mesh pointCloudMesh;
  private Shader pointCloudShader;
  // Keep track of the last point cloud rendered to avoid updating the VBO if point cloud
  // was not changed.  Do this using the timestamp since we can't compare PointCloud
objects.
  private long lastPointCloudTimestamp = 0;

  // Virtual object (ARCore pawn)
  private Mesh pointMesh;
  private Shader pointShader;
  private Texture wallPointAlbedoTexture;
  private Texture otherCablePointAlbedoTexture;
  private Texture matchingCablePointAlbedoTexture;
  private Texture cableConnectionPointAlbedoTexture;
  private Texture linkedPointAlbedoTexture;
  private Texture nearestPointAlbedoTexture;

  private Mesh lineMesh;
  private Shader lineShader;
  private Texture wallLineAlbedoTexture;
  private Texture otherCableLineAlbedoTexture;
  private Texture matchingCableLineAlbedoTexture;

  private final List<WrappedAnchor> wrappedAnchors = new ArrayList<>();
  private WrappedAnchor linkedAnchor = null;

  private boolean runAnchoring = false;
  private boolean runRegularCreation = false;
  private boolean runCableConnectionCreation = false;
  private boolean runConversion = false;
  private boolean runAttachment = false;
  private boolean runDetachment = false;
  private boolean runDeletion = false;


  private SpecularCubemapFilter cubemapFilter;

  // Temporary matrix allocated here to reduce number of allocations for each frame.
  private final float[] modelMatrix = new float[16];
  private final float[] viewMatrix = new float[16];
  private final float[] projectionMatrix = new float[16];
  private final float[] modelViewMatrix = new float[16]; // view x model
  private final float[] modelViewProjectionMatrix = new float[16]; // projection x view x
model
  private final float[] sphericalHarmonicsCoefficients = new float[9 * 3];

  private TextView distanceDisplay;

  private FloorPlan floorPlan;
  private int id;

  private CableType[] planCableTypes;

  ILineType currentLineType;

  int frameNumber = 0;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera);
    surfaceView = findViewById(R.id.surfaceview);
    displayRotationHelper = new DisplayRotationHelper(/* context= */ this);

    // Set up touch listener.
    //tapHelper = new TapHelper(/* context= */ this);
```

```
    //surfaceView.setOnTouchListener(tapHelper);

    // Set up renderer.
    render = new SampleRender(surfaceView, this, getAssets());

    installRequested = false;

    Intent intent = getIntent();
    id = intent.getIntExtra("ID", -1);

    Context context = getApplicationContext();
    floorPlan = FloorPlan.Load(context, id);
    if(floorPlan == null){
      Intent returnIntent = new Intent();
      setResult(RESULT_FIRST_USER, returnIntent);
      finish();
    }

    int[] cableTypeIds = floorPlan.getCableTypeIds();
    if(cableTypeIds == null)
      cableTypeIds = new int[0];

    ArrayList<CableType> cableTypeList = new ArrayList<CableType>(cableTypeIds.length);
    for(int cableTypeId : cableTypeIds){
      cableTypeList.add(CableType.Load(context, cableTypeId));
    }

    planCableTypes = cableTypeList.stream().filter(x -> x != null).toArray(size -> new
CableType[size]);
    currentLineType = new WallLineType();
    Spinner lineTypeSelector = findViewById(R.id.line_type_selector);

    LineTypeDropdownItem[] lineTypeItems = new LineTypeDropdownItem[planCableTypes.length
+ 1];
    lineTypeItems[0] = new LineTypeDropdownItem(null);
    int i = 1;
    for(CableType cableType : planCableTypes){
      lineTypeItems[i] = new LineTypeDropdownItem(cableType);
      i++;
    }

    lineTypeSelector.setAdapter(new ArrayAdapter<LineTypeDropdownItem>(context,
R.layout.spinner_item, lineTypeItems));

    lineTypeSelector.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
      @Override
      public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{
        LineTypeDropdownItem item = (LineTypeDropdownItem)parent.getSelectedItem();
        if(item.getCableType() == null){
          currentLineType = new WallLineType();
        }else{
          currentLineType = new CableLineType(item.getCableType().getId());
        }
      }

      @Override
      public void onNothingSelected(AdapterView<?> parent) {
        currentLineType = new WallLineType();
      }
    });

    depthSettings.onCreate(this);
    ImageButton doneButton = findViewById(R.id.done_button);
    doneButton.setOnClickListener(
            new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                ArrayList<FloorPlanPoint> points = new
ArrayList<FloorPlanPoint>(wrappedAnchors.size());
                ArrayList<AnchorLink> links = new ArrayList<AnchorLink>();
```

```
                List<WrappedAnchor> processableWrappedAnchors =
wrappedAnchors.stream().filter(x -> x.getAnchorLinks().length > 0 &&
x.getLastKnownPose() != null).collect(Collectors.toList());
                for(WrappedAnchor wa : processableWrappedAnchors){
                  AnchorLink[] als = wa.getAnchorLinks();
                  if(als.length == 0)
                    continue;

                  Pose lastKnownPose = wa.getLastKnownPose();
                  if(lastKnownPose == null)
                    continue;

                  float[] translation = lastKnownPose.getTranslation();
                  points.add(new FloorPlanPoint(translation[0], translation[2],
translation[1], wa.getPointType()));
                  for(AnchorLink al : als){
                    if(!links.contains(al)){
                      links.add(al);
                    }
                  }
                }
                FloorPlanPoint[] pointsArray = new FloorPlanPoint[points.size()];
                int i = 0;
                for(FloorPlanPoint f : points){
                  pointsArray[i] = f;
                  i++;
                }
                Intent intent = new Intent();
                FloorPlanLine[] pointLinesArray = new FloorPlanLine[links.size()];
                for (i = 0; i < pointLinesArray.length; i+=1){
                  pointLinesArray[i] = new
FloorPlanLine(processableWrappedAnchors.indexOf(links.get(i).getAnchor1()),
processableWrappedAnchors.indexOf(links.get(i).getAnchor2()),
links.get(i).getLineType());
                }
                preprocessArData(pointsArray, pointLinesArray);
                setResult(RESULT_OK, intent);
                finish();
              }
            });

    ImageButton anchorButton = findViewById(R.id.anchor_button);
    anchorButton.setOnClickListener(
            new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                runAnchoring = true;
              }
            });

    ImageButton creationButton = findViewById(R.id.creation_button);
    creationButton.setOnClickListener(
            new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                runRegularCreation = true;
              }
            });

    ImageButton connectionCreationButton = findViewById(R.id.cable_link_creation_button);
    connectionCreationButton.setOnClickListener(
            new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                runCableConnectionCreation = true;
              }
            });

    ImageButton conversionButton = findViewById(R.id.cable_link_convert_button);
    conversionButton.setOnClickListener(
            new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
              runConversion = true;
            }
          });

    ImageButton attachmentButton = findViewById(R.id.attach_button);
    attachmentButton.setOnClickListener(
            new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                runAttachment = true;
              }
            });

    ImageButton detachmentButton = findViewById(R.id.detach_button);
    detachmentButton.setOnClickListener(
            new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                runDetachment = true;
              }
            });

    ImageButton deleteButton = findViewById(R.id.remove_button);
    deleteButton.setOnClickListener(
            new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                runDeletion = true;
              }
            });

    distanceDisplay = findViewById(R.id.distanceDisplay);
    distanceDisplay.setText("-.-- м");
  }

  private class LineTypeDropdownItem{
    @Nullable
    private CableType _cableType;

    public LineTypeDropdownItem(@Nullable CableType cableType){
      _cableType = cableType;
    }

    @Nullable
    public CableType getCableType(){
      return _cableType;
    }

    @Override
    public String toString(){
      if(_cableType == null)
        return "Стіна";

      return _cableType.getName();
    }
  }

  @Override
  protected void onDestroy() {
    if (session != null) {
      // Explicitly close ARCore Session to release native resources.
      // Review the API reference for important considerations before calling close() in
apps with
      // more complicated lifecycle requirements:
      //
https://developers.google.com/ar/reference/java/arcore/reference/com/google/ar/core/Sessi
on#close()
      session.close();
      session = null;
```

```
    }

    super.onDestroy();
  }

  @Override
  protected void onResume() {
    super.onResume();

    if (session == null) {
      Exception exception = null;
      String message = null;
      try {
        // Always check the latest availability.
        Availability availability = ArCoreApk.getInstance().checkAvailability(this);

        // In all other cases, try to install ARCore and handle installation failures.
        if (availability != Availability.SUPPORTED_INSTALLED) {
          switch (ArCoreApk.getInstance().requestInstall(this, !installRequested)) {
            case INSTALL_REQUESTED:
              installRequested = true;
              return;
            case INSTALLED:
              break;
          }
        }

        // ARCore requires camera permissions to operate. If we did not yet obtain
runtime
        // permission on Android M and above, now is a good time to ask the user for it.
        if (!CameraPermissionHelper.hasCameraPermission(this)) {
          CameraPermissionHelper.requestCameraPermission(this);
          return;
        }

        // Create the session.
        session = new Session(/* context= */ this);
      } catch (UnavailableArcoreNotInstalledException
               | UnavailableUserDeclinedInstallationException e) {
        message = "Please install ARCore";
        exception = e;
      } catch (UnavailableApkTooOldException e) {
        message = "Please update ARCore";
        exception = e;
      } catch (UnavailableSdkTooOldException e) {
        message = "Please update this app";
        exception = e;
      } catch (UnavailableDeviceNotCompatibleException e) {
        message = "This device does not support AR";
        exception = e;
      } catch (Exception e) {
        message = "Failed to create AR session";
        exception = e;
      }

      if (message != null) {
        messageSnackbarHelper.showError(this, message);
        Log.e(TAG, "Exception creating session", exception);
        return;
      }
    }

    // Note that order matters - see the note in onPause(), the reverse applies here.
    try {
      configureSession();
      // To record a live camera session for later playback, call
      // `session.startRecording(recordingConfig)` at anytime. To playback a previously
recorded AR
      // session instead of using the live camera feed, call
      // `session.setPlaybackDatasetUri(Uri)` before calling `session.resume()`. To
      // learn more about recording and playback, see:
```

```
      // https://developers.google.com/ar/develop/java/recording-and-playback
      session.resume();
    } catch (CameraNotAvailableException e) {
      messageSnackbarHelper.showError(this, "Camera not available. Try restarting the
app.");
      session = null;
      return;
    }

    surfaceView.onResume();
    displayRotationHelper.onResume();
  }

  @Override
  public void onPause() {
    super.onPause();
    if (session != null) {
      // Note that the order matters - GLSurfaceView is paused first so that it does not
try
      // to query the session. If Session is paused before GLSurfaceView, GLSurfaceView
may
      // still call session.update() and get a SessionPausedException.
      displayRotationHelper.onPause();
      surfaceView.onPause();
      session.pause();
    }
  }

  @Override
  public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
results) {
    super.onRequestPermissionsResult(requestCode, permissions, results);
    if (!CameraPermissionHelper.hasCameraPermission(this)) {
      // Use toast instead of snackbar here since the activity will exit.
      Toast.makeText(this, "Camera permission is needed to run this application",
Toast.LENGTH_LONG)
              .show();
      if (!CameraPermissionHelper.shouldShowRequestPermissionRationale(this)) {
        // Permission denied with checking "Do not ask again".
        CameraPermissionHelper.launchPermissionSettings(this);
      }
      finish();
    }
  }

  @Override
  public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);
    FullScreenHelper.setFullScreenOnWindowFocusChanged(this, hasFocus);
  }

  @Override
  public void onSurfaceCreated(SampleRender render) {
    // Prepare the rendering objects. This involves reading shaders and 3D model files,
so may throw
    // an IOException.
    try {
      planeRenderer = new PlaneRenderer(render);
      backgroundRenderer = new BackgroundRenderer(render);
      virtualSceneFramebuffer = new Framebuffer(render, /* width= */ 1, /* height= */ 1);

      cubemapFilter =
              new SpecularCubemapFilter(
                      render, CUBEMAP_RESOLUTION, CUBEMAP_NUMBER_OF_IMPORTANCE_SAMPLES);


      // Point cloud
      pointCloudShader =
              Shader.createFromAssets(
                              render,
                              "shaders/point_cloud.vert",
```

```
                                "shaders/point_cloud.frag",
                                /* defines= */ null)
                        .setVec4(
                                "u_Color", new float[] {31.0f / 255.0f, 188.0f / 255.0f,
210.0f / 255.0f, 1.0f})
                        .setFloat("u_PointSize", 5.0f);
        // four entries per vertex: X, Y, Z, confidence
        pointCloudVertexBuffer =
                new VertexBuffer(render, /* numberOfEntriesPerVertex= */ 4, /* entries= */
null);
        final VertexBuffer[] pointCloudVertexBuffers = {pointCloudVertexBuffer};
        pointCloudMesh =
                new Mesh(
                        render, Mesh.PrimitiveMode.POINTS, /* indexBuffer= */ null,
pointCloudVertexBuffers);

        // Virtual object to render (ARCore pawn)
        wallPointAlbedoTexture =
                Texture.createFromAsset(
                        render,
                        "models/point_wall_albedo.png",
                        Texture.WrapMode.CLAMP_TO_EDGE,
                        Texture.ColorFormat.SRGB);

        otherCablePointAlbedoTexture =
                Texture.createFromAsset(
                        render,
                        "models/point_other_cable_albedo.png",
                        Texture.WrapMode.CLAMP_TO_EDGE,
                        Texture.ColorFormat.SRGB);

        matchingCablePointAlbedoTexture =
                Texture.createFromAsset(
                        render,
                        "models/point_matching_cable_albedo.png",
                        Texture.WrapMode.CLAMP_TO_EDGE,
                        Texture.ColorFormat.SRGB);

        cableConnectionPointAlbedoTexture =
                Texture.createFromAsset(
                        render,
                        "models/point_cable_connection_albedo.png",
                        Texture.WrapMode.CLAMP_TO_EDGE,
                        Texture.ColorFormat.SRGB);

        linkedPointAlbedoTexture =
                Texture.createFromAsset(
                        render,
                        "models/point_linked_albedo.png",
                        Texture.WrapMode.CLAMP_TO_EDGE,
                        Texture.ColorFormat.SRGB);

        nearestPointAlbedoTexture =
                Texture.createFromAsset(
                        render,
                        "models/point_nearest_albedo.png",
                        Texture.WrapMode.CLAMP_TO_EDGE,
                        Texture.ColorFormat.SRGB);

        Texture pointPbrTexture =
                Texture.createFromAsset(
                        render,
                        "models/point_roughness_metallic_ao.png",
                        Texture.WrapMode.CLAMP_TO_EDGE,
                        Texture.ColorFormat.LINEAR);

        pointMesh = Mesh.createFromAsset(render, "models/point.obj");
        pointShader =
                Shader.createFromAssets(
                                render,
                                "shaders/environmental_hdr.vert",
```

```
                                "shaders/environmental_hdr.frag",
                                /* defines= */ new HashMap<String, String>() {
                                  {
                                    put(
                                            "NUMBER_OF_MIPMAP_LEVELS",

Integer.toString(cubemapFilter.getNumberOfMipmapLevels()));
                                  }
                                })
                        .setTexture("u_AlbedoTexture", wallPointAlbedoTexture)
                        .setTexture("u_RoughnessMetallicAmbientOcclusionTexture",
pointPbrTexture)
                        .setTexture("u_Cubemap",
cubemapFilter.getFilteredCubemapTexture());

      wallLineAlbedoTexture =
              Texture.createFromAsset(
                      render,
                      "models/line_wall_albedo.png",
                      Texture.WrapMode.CLAMP_TO_EDGE,
                      Texture.ColorFormat.SRGB);

      otherCableLineAlbedoTexture =
              Texture.createFromAsset(
                      render,
                      "models/line_other_cable_albedo.png",
                      Texture.WrapMode.CLAMP_TO_EDGE,
                      Texture.ColorFormat.SRGB);

      matchingCableLineAlbedoTexture =
              Texture.createFromAsset(
                      render,
                      "models/line_matching_cable_albedo.png",
                      Texture.WrapMode.CLAMP_TO_EDGE,
                      Texture.ColorFormat.SRGB);

      Texture linePbrTexture =
              Texture.createFromAsset(
                      render,
                      "models/line_roughness_metallic_ao.png",
                      Texture.WrapMode.CLAMP_TO_EDGE,
                      Texture.ColorFormat.LINEAR);

      lineMesh = Mesh.createFromAsset(render, "models/line.obj");
      lineShader =
              Shader.createFromAssets(
                                render,
                                "shaders/environmental_hdr.vert",
                                "shaders/environmental_hdr.frag",
                                /* defines= */ new HashMap<String, String>() {
                                  {
                                    put(
                                            "NUMBER_OF_MIPMAP_LEVELS",

Integer.toString(cubemapFilter.getNumberOfMipmapLevels()));
                                  }
                                })
                        .setTexture("u_AlbedoTexture", wallLineAlbedoTexture)
                        .setTexture("u_RoughnessMetallicAmbientOcclusionTexture",
linePbrTexture)
                        .setTexture("u_Cubemap",
cubemapFilter.getFilteredCubemapTexture());
    } catch (IOException e) {
      Log.e(TAG, "Failed to read a required asset file", e);
      messageSnackbarHelper.showError(this, "Failed to read a required asset file: " +
e);
    }
  }

  @Override
  public void onSurfaceChanged(SampleRender render, int width, int height) {
```

```
    displayRotationHelper.onSurfaceChanged(width, height);
    virtualSceneFramebuffer.resize(width, height);
  }

  @Override
  public void onDrawFrame(SampleRender render) {
    if (session == null) {
      return;
    }

    // Texture names should only be set once on a GL thread unless they change. This is
done during
    // onDrawFrame rather than onSurfaceCreated since the session is not guaranteed to
have been
    // initialized during the execution of onSurfaceCreated.
    if (!hasSetTextureNames) {
      session.setCameraTextureNames(
              new int[] {backgroundRenderer.getCameraColorTexture().getTextureId()});
      hasSetTextureNames = true;
    }

    // -- Update per-frame state

    // Notify ARCore session that the view size changed so that the perspective matrix
and
    // the video background can be properly adjusted.
    displayRotationHelper.updateSessionIfNeeded(session);

    // Obtain the current frame from the AR Session. When the configuration is set to
    // UpdateMode.BLOCKING (it is by default), this will throttle the rendering to the
    // camera framerate.
    Frame frame;
    try {
      frame = session.update();
    } catch (CameraNotAvailableException e) {
      Log.e(TAG, "Camera not available during onDrawFrame", e);
      messageSnackbarHelper.showError(this, "Camera not available. Try restarting the
app.");
      return;
    }
    Camera camera = frame.getCamera();

    // Update BackgroundRenderer state to match the depth settings.
    try {
      backgroundRenderer.setUseDepthVisualization(
              render, depthSettings.depthColorVisualizationEnabled());
      backgroundRenderer.setUseOcclusion(render, false);
    } catch (IOException e) {
      Log.e(TAG, "Failed to read a required asset file", e);
      messageSnackbarHelper.showError(this, "Failed to read a required asset file: " +
e);
      return;
    }
    // BackgroundRenderer.updateDisplayGeometry must be called every frame to update the
coordinates
    // used to draw the background camera image.
    backgroundRenderer.updateDisplayGeometry(frame);

    if (camera.getTrackingState() == TrackingState.TRACKING
            && (depthSettings.useDepthForOcclusion()
            || depthSettings.depthColorVisualizationEnabled())) {
      try (Image depthImage = frame.acquireDepthImage16Bits()) {
        backgroundRenderer.updateCameraDepthTexture(depthImage);
      } catch (NotYetAvailableException e) {
        // This normally means that depth data is not available yet. This is normal so we
will not
        // spam the logcat with this.
      }
    }

    // Handle one tap per frame.
```

```
    handleTap(frame, camera);

    // Keep the screen unlocked while tracking, but allow it to lock when tracking stops.
    trackingStateHelper.updateKeepScreenOnFlag(camera.getTrackingState());

    // Show a message based on whether tracking has failed, if planes are detected, and
if the user
    // has placed any objects.
    String message = null;
    if (camera.getTrackingState() == TrackingState.PAUSED) {
      if (camera.getTrackingFailureReason() == TrackingFailureReason.NONE) {
        message = SEARCHING_PLANE_MESSAGE;
      } else {
        message = TrackingStateHelper.getTrackingFailureReasonString(camera);
      }
    } else if (!hasTrackingPlane()) {
      message = SEARCHING_PLANE_MESSAGE;
    }
    if (message != null) {
      messageSnackbarHelper.hide(this);
    }

    // -- Draw background

    if (frame.getTimestamp() != 0) {
      // Suppress rendering if the camera did not produce the first frame yet. This is to
avoid
      // drawing possible leftover data from previous sessions if the texture is reused.
      backgroundRenderer.drawBackground(render);
    }

    // If not tracking, don't draw 3D objects.
    if (camera.getTrackingState() == TrackingState.PAUSED) {
      return;
    }

    // -- Draw non-occluded virtual objects (planes, point cloud)

    // Get projection matrix.
    camera.getProjectionMatrix(projectionMatrix, 0, Z_NEAR, Z_FAR);

    // Get camera matrix and draw.
    camera.getViewMatrix(viewMatrix, 0);

    // Visualize tracked points.
    // Use try-with-resources to automatically release the point cloud.
    try (PointCloud pointCloud = frame.acquirePointCloud()) {
      if (pointCloud.getTimestamp() > lastPointCloudTimestamp) {
        pointCloudVertexBuffer.set(pointCloud.getPoints());
        lastPointCloudTimestamp = pointCloud.getTimestamp();
      }
      Matrix.multiplyMM(modelViewProjectionMatrix, 0, projectionMatrix, 0, viewMatrix,
0);
      pointCloudShader.setMat4("u_ModelViewProjection", modelViewProjectionMatrix);
      render.draw(pointCloudMesh, pointCloudShader);
    }

    // Visualize planes.
    planeRenderer.drawPlanes(
            render,
            session.getAllTrackables(Plane.class).stream().filter((Plane x) ->
x.getType() != Plane.Type.VERTICAL).collect(Collectors.toList()),
            camera.getDisplayOrientedPose(),
            projectionMatrix);

    // -- Draw occluded virtual objects

    for (int i = 0; i < wrappedAnchors.size(); i++) {
      WrappedAnchor wrappedAnchor = wrappedAnchors.get(i);
      Anchor anchor = wrappedAnchor.getAnchor();
      if (anchor.getTrackingState() != TrackingState.TRACKING) {
```

```
        if(anchor.getTrackingState() == TrackingState.STOPPED){
          for(AnchorLink al : wrappedAnchor.getAnchorLinks()){
            al.getAnchor1().removeAnchorLink(al);
            al.getAnchor2().removeAnchorLink(al);
          }
          wrappedAnchor.getAnchor().detach();
          wrappedAnchors.remove(wrappedAnchor);
          i--;
        }
        continue;
      }

      wrappedAnchor.setLastKnownPose(anchor.getPose());
    }

    // Visualize lines
    render.clear(virtualSceneFramebuffer, 0f, 0f, 0f, 0f);
    ArrayList<AnchorLink> processedLinks = new ArrayList<>();
    for (int i = 0; i < wrappedAnchors.size(); i++) {
      WrappedAnchor wrappedAnchor = wrappedAnchors.get(i);
      AnchorLink[] anchorLinks = wrappedAnchor.getAnchorLinks();
      for(AnchorLink al : anchorLinks){
        if(processedLinks.contains(al)){
          continue;
        }
        processedLinks.add(al);
        Anchor anchor1 = al.getAnchor1().getAnchor();
        Anchor anchor2 = al.getAnchor2().getAnchor();
        if(anchor1.getTrackingState() != TrackingState.TRACKING ||
anchor2.getTrackingState() != TrackingState.TRACKING){
          continue;
        }

        float[] position1 = anchor1.getPose().getTranslation();
        float[] position2 = anchor2.getPose().getTranslation();
        float[] position = new float[3];
        for(int j = 0; j < 3; j++){
          position[j] = (position1[j] + position2[j]) / 2;
        }
        float length = (float)Math.pow(Math.pow(position1[0] - position2[0], 2) +
Math.pow(position1[1] - position2[1], 2) + Math.pow(position1[2] - position2[2], 2), 1 /
2.0);

        float[] modelDirectionVector = new float[] {0, 1, 0};
        float[] lineVector = new float[] {position2[0] - position1[0], position2[1] -
position1[1], position2[2] - position1[2]};
        float[] rotationAxisVector = normalizeVector(crossProduct(lineVector,
modelDirectionVector));
        float rotationAngle = vectorAngle(lineVector, modelDirectionVector);
        Matrix.setIdentityM(modelMatrix, 0);
        Matrix.translateM(modelMatrix, 0, position[0], position[1], position[2]);
        Matrix.rotateM(modelMatrix, 0, -(float)(rotationAngle * 180 / Math.PI),
rotationAxisVector[0], rotationAxisVector[1], rotationAxisVector[2]);
        Matrix.scaleM(modelMatrix, 0, 1, length, 1);

        float[] scalingVector = normalizeVector(lineVector);

        //scale the line to match length
        //Matrix.scaleM(modelMatrix, 0,lineVector[0], lineVector[1], lineVector[2]);

        // Calculate model/view/projection matrices
        Matrix.multiplyMM(modelViewMatrix, 0, viewMatrix, 0, modelMatrix, 0);
        Matrix.multiplyMM(modelViewProjectionMatrix, 0, projectionMatrix, 0,
modelViewMatrix, 0);

        // Update shader properties and draw
        lineShader.setMat4("u_ModelView", modelViewMatrix);
        lineShader.setMat4("u_ModelViewProjection", modelViewProjectionMatrix);

        if(al.getLineType().isWall()){
          lineShader.setTexture("u_AlbedoTexture", wallLineAlbedoTexture);
```

```
        }else if(al.getLineType().isCable()){
          if(currentLineType.isCable() && al.getLineType().getCableTypeId() ==
currentLineType.getCableTypeId()){
            lineShader.setTexture("u_AlbedoTexture", matchingCableLineAlbedoTexture);
          }else{
            lineShader.setTexture("u_AlbedoTexture", otherCableLineAlbedoTexture);
          }
        }else{
          lineShader.setTexture("u_AlbedoTexture", wallLineAlbedoTexture);
        }



        render.draw(lineMesh, lineShader, virtualSceneFramebuffer);
      }
    }
    // Visualize points
    for (int i = 0; i < wrappedAnchors.size(); i++) {
      WrappedAnchor wrappedAnchor = wrappedAnchors.get(i);
      Anchor anchor = wrappedAnchor.getAnchor();

      // Get the current pose of an Anchor in world space. The Anchor pose is updated
      // during calls to session.update() as ARCore refines its estimate of the world.
      anchor.getPose().toMatrix(modelMatrix, 0);

      // Calculate model/view/projection matrices
      Matrix.multiplyMM(modelViewMatrix, 0, viewMatrix, 0, modelMatrix, 0);
      Matrix.multiplyMM(modelViewProjectionMatrix, 0, projectionMatrix, 0,
modelViewMatrix, 0);

      // Update shader properties and draw
      pointShader.setMat4("u_ModelView", modelViewMatrix);
      pointShader.setMat4("u_ModelViewProjection", modelViewProjectionMatrix);

      WrappedAnchor relatedAnchor = null;
      List<HitResult> hitResultList;
      DisplayMetrics displayMetrics = new DisplayMetrics();

      getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
      hitResultList = frame.hitTest(displayMetrics.widthPixels / 2f,
displayMetrics.heightPixels / 2f);
      for (HitResult hit : hitResultList) {
        Trackable trackable = hit.getTrackable();
        if ((trackable instanceof Plane
                && ((Plane) trackable).isPoseInPolygon(hit.getHitPose())
                && (PlaneRenderer.calculateDistanceToPlane(hit.getHitPose(),
camera.getPose()) > 0
                && ((Plane) trackable).getType() != Plane.Type.VERTICAL))) {
          Pose anchorPoseInverted = hit.getHitPose().extractTranslation().inverse();
          double lowestDistance = 0.1;
          for(WrappedAnchor wa : wrappedAnchors){
            float[] distanceArray =
anchorPoseInverted.compose(wa.getLastKnownPose()).getTranslation();
            double distance = Math.sqrt(Math.pow(distanceArray[0], 2) +
Math.pow(distanceArray[1], 2) + Math.pow(distanceArray[2], 2));
            if(distance < lowestDistance){
              relatedAnchor = wa;
              lowestDistance = distance;
            }
          }
        }
      }

      if(wrappedAnchor == relatedAnchor && relatedAnchor == linkedAnchor && frameNumber %
30 < 5){
        pointShader.setTexture("u_AlbedoTexture", nearestPointAlbedoTexture);
      }else if(wrappedAnchor == relatedAnchor && relatedAnchor == linkedAnchor &&
frameNumber % 30 < 10){
        pointShader.setTexture("u_AlbedoTexture", linkedPointAlbedoTexture);
      }else if(wrappedAnchor == relatedAnchor && relatedAnchor != linkedAnchor &&
frameNumber % 30 < 10){
```

```
        pointShader.setTexture("u_AlbedoTexture", nearestPointAlbedoTexture);
      }else if(wrappedAnchor == linkedAnchor && frameNumber % 30 < 10){
        pointShader.setTexture("u_AlbedoTexture", linkedPointAlbedoTexture);
      }else if(wrappedAnchor.getPointType().isWallPoint()){
        pointShader.setTexture("u_AlbedoTexture", wallPointAlbedoTexture);
      }else if(wrappedAnchor.getPointType().isCableConnectionPoint()){
        pointShader.setTexture("u_AlbedoTexture", cableConnectionPointAlbedoTexture);
      }else if(wrappedAnchor.getPointType().isCablePoint()){
        if(currentLineType.isCable() && wrappedAnchor.getPointType().getCableTypeId() ==
currentLineType.getCableTypeId()){
          pointShader.setTexture("u_AlbedoTexture", matchingCablePointAlbedoTexture);
        }else{
          pointShader.setTexture("u_AlbedoTexture", otherCablePointAlbedoTexture);
        }
      }else{
        pointShader.setTexture("u_AlbedoTexture", wallPointAlbedoTexture);
      }

      render.draw(pointMesh, pointShader, virtualSceneFramebuffer);
    }

    if(linkedAnchor != null && camera.getTrackingState() == TrackingState.TRACKING) {
        List<HitResult> hitResultList;
        DisplayMetrics displayMetrics = new DisplayMetrics();

        getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
        hitResultList = frame.hitTest(displayMetrics.widthPixels / 2f,
displayMetrics.heightPixels / 2f);

      String str = "-.-- м";
        for (HitResult hit : hitResultList) {
          Trackable trackable = hit.getTrackable();
          if ((trackable instanceof Plane
                  && ((Plane) trackable).isPoseInPolygon(hit.getHitPose())
                  && (PlaneRenderer.calculateDistanceToPlane(hit.getHitPose(),
camera.getPose()) > 0
                  && ((Plane) trackable).getType() != Plane.Type.VERTICAL))) {

            Pose hitPoseInverted = hit.getHitPose().extractTranslation().inverse();
            WrappedAnchor wa = linkedAnchor;
            float[] distanceArray =
hitPoseInverted.compose(wa.getLastKnownPose()).getTranslation();
            double distance = Math.sqrt(Math.pow(distanceArray[0], 2) +
Math.pow(distanceArray[2], 2));
            str = String.format("%.2f м", distance);
            break;
          }
        }
      final String str2 = str;
      runOnUiThread(() -> {
        distanceDisplay.setText(str2);
      });
    }else{
      runOnUiThread(() -> {
        distanceDisplay.setText("-.-- м");
      });
    }

    // Compose the virtual scene with the background.
    backgroundRenderer.drawVirtualScene(render, virtualSceneFramebuffer, Z_NEAR, Z_FAR);
    frameNumber++;
  }

  private static float vectorAngle(float[] vector1, float[] vector2){
    float length1 = (float)Math.sqrt(Math.pow(vector1[0], 2) + Math.pow(vector1[1], 2) +
Math.pow(vector1[2], 2));
    float length2 = (float)Math.sqrt(Math.pow(vector2[0], 2) + Math.pow(vector2[1], 2) +
Math.pow(vector2[2], 2));
    float angleCos = (vector1[0]*vector2[0] + vector1[1]*vector2[1] +
vector1[2]*vector2[2]) / (length1 * length2);
    return (float)Math.acos(angleCos);
```

```
  }

  private static float[] normalizeVector(float[] vector){
    float length = (float)Math.sqrt(Math.pow(vector[0], 2) + Math.pow(vector[1], 2) +
Math.pow(vector[2], 2));
    return new float[] {vector[0] / length, vector[1] / length, vector[2] / length};
  }

  private static float[] crossProduct(float[] vector1, float[] vector2){
    float[] result = new float[3];
    result[0] = vector1[1] * vector2[2] - vector1[2] * vector2[1];
    result[1] = vector1[2] * vector2[0] - vector1[0] * vector2[2];
    result[2] = vector1[0] * vector2[1] - vector1[1] * vector2[0];
    return result;
  }

  // Handle only one tap per frame, as taps are usually low frequency compared to frame
rate.
  private void handleTap(Frame frame, Camera camera) {
    boolean creatingRegular = runRegularCreation;
    boolean creatingCableConnection = runCableConnectionCreation;
    boolean converting = runConversion;
    boolean attaching = runAttachment;
    boolean detaching = runDetachment;
    boolean anchoring = runAnchoring;
    boolean deleting = runDeletion;
    runRegularCreation = false;
    runCableConnectionCreation = false;
    runConversion = false;
    runAttachment = false;
    runDetachment = false;
    runAnchoring = false;
    runDeletion = false;
    if ((creatingRegular ^ creatingCableConnection ^ converting ^ attaching ^ detaching ^
anchoring ^ deleting) && camera.getTrackingState() == TrackingState.TRACKING) {
      List<HitResult> hitResultList;
      DisplayMetrics displayMetrics = new DisplayMetrics();

      getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
      hitResultList = frame.hitTest(displayMetrics.widthPixels / 2f,
displayMetrics.heightPixels / 2f);

      for (HitResult hit : hitResultList) {
        Trackable trackable = hit.getTrackable();
        if ((trackable instanceof Plane
                && ((Plane) trackable).isPoseInPolygon(hit.getHitPose())
                && (PlaneRenderer.calculateDistanceToPlane(hit.getHitPose(),
camera.getPose()) > 0
                && ((Plane) trackable).getType() != Plane.Type.VERTICAL))) {
          Pose anchorPoseInverted = hit.getHitPose().extractTranslation().inverse();
          WrappedAnchor relatedAnchor = null;
          double lowestDistance = 0.1;
          for(WrappedAnchor wa : wrappedAnchors){
            float[] distanceArray =
anchorPoseInverted.compose(wa.getLastKnownPose()).getTranslation();
            double distance = Math.sqrt(Math.pow(distanceArray[0], 2) +
Math.pow(distanceArray[1], 2) + Math.pow(distanceArray[2], 2));
            if(distance < lowestDistance){
              relatedAnchor = wa;
              lowestDistance = distance;
            }
          }
          if(creatingRegular || creatingCableConnection){
            IPointType pointType;
            if(creatingCableConnection){
              pointType = new CableConnectionPointType();
            }else if(currentLineType.isWall()){
              pointType = new WallPointType();
            }else{
              pointType = new CablePointType(currentLineType.getCableTypeId());
            }
```

```
            WrappedAnchor newAnchor = new WrappedAnchor(hit.createAnchor(), trackable,
pointType);
            wrappedAnchors.add(newAnchor);

            if (linkedAnchor != null) {
              tryLinkAnchors(newAnchor, linkedAnchor);
            }
            linkedAnchor = newAnchor;
            break;
          }
          if(converting && relatedAnchor != null
&& !relatedAnchor.getPointType().isWallPoint()){
            if(relatedAnchor.getPointType().isCablePoint()){
              relatedAnchor.setPointType(new CableConnectionPointType());
            }else if(relatedAnchor.getPointType().isCableConnectionPoint() &&
relatedAnchor.getAnchorLinks().length <= 2){
              if(relatedAnchor.getAnchorLinks().length == 0 &&
currentLineType.isCable()){
                relatedAnchor.setPointType(new
CablePointType(currentLineType.getCableTypeId()));
              } else if (relatedAnchor.getAnchorLinks().length == 1){
                relatedAnchor.setPointType(new
CablePointType(relatedAnchor.getAnchorLinks()[0].getLineType().getCableTypeId()));
              } else if (relatedAnchor.getAnchorLinks().length == 2 &&
relatedAnchor.getAnchorLinks()[0].getLineType().getCableTypeId() ==
relatedAnchor.getAnchorLinks()[1].getLineType().getCableTypeId()){
                relatedAnchor.setPointType(new
CablePointType(relatedAnchor.getAnchorLinks()[0].getLineType().getCableTypeId()));
              }
            }
            linkedAnchor = relatedAnchor;
          }
          if(anchoring){
            linkedAnchor = relatedAnchor;
            break;
          }
          if(attaching && relatedAnchor != null && linkedAnchor != null){
            tryLinkAnchors(relatedAnchor, linkedAnchor);
            linkedAnchor = relatedAnchor;
            break;
          }
          if(detaching && relatedAnchor != null && linkedAnchor != null){
            tryLinkAnchors(relatedAnchor, linkedAnchor);
            for(AnchorLink al : relatedAnchor.getAnchorLinks()){
              if((al.getAnchor1() == relatedAnchor && al.getAnchor2() == linkedAnchor) ||
(al.getAnchor2() == relatedAnchor && al.getAnchor1() == linkedAnchor)){
                relatedAnchor.removeAnchorLink(al);
                linkedAnchor.removeAnchorLink(al);
              }
            }
            linkedAnchor = relatedAnchor;
            break;
          }
          if(deleting && relatedAnchor != null){
            if(linkedAnchor == relatedAnchor){
              linkedAnchor = null;
            }
            for(AnchorLink al : relatedAnchor.getAnchorLinks()){
              al.getAnchor1().removeAnchorLink(al);
              al.getAnchor2().removeAnchorLink(al);
            }
            relatedAnchor.getAnchor().detach();
            wrappedAnchors.remove(relatedAnchor);
          }
        }
      }
    }
  }

  private void tryLinkAnchors(WrappedAnchor anchor1, WrappedAnchor anchor2){
    for(AnchorLink al : anchor1.getAnchorLinks()){
```

```
      if(al.getAnchor1() == anchor2 || al.getAnchor2() == anchor2){
        return;
      }
    }
    if(anchor1.getPointType().isWallPoint() && anchor2.getPointType().isWallPoint()){
      AnchorLink al = new AnchorLink(anchor1, anchor2, new WallLineType());
      anchor1.addAnchorLink(al);
      anchor2.addAnchorLink(al);
      return;
    }

    if(anchor1.getPointType().isCableConnectionPoint() ||
anchor2.getPointType().isCableConnectionPoint()){
      WrappedAnchor ccAnchor, otherAnchor;
      if(anchor1.getPointType().isCableConnectionPoint()){
        ccAnchor = anchor1;
        otherAnchor = anchor2;
      }else{
        ccAnchor = anchor2;
        otherAnchor = anchor1;
      }
      if(otherAnchor.getPointType().isCablePoint()){
        AnchorLink al = new AnchorLink(ccAnchor, otherAnchor, new
CableLineType(otherAnchor.getPointType().getCableTypeId()));
        ccAnchor.addAnchorLink(al);
        otherAnchor.addAnchorLink(al);
        return;
      }
      if(otherAnchor.getPointType().isCableConnectionPoint() &&
currentLineType.isCable()){
        AnchorLink al = new AnchorLink(ccAnchor, otherAnchor, currentLineType);
        ccAnchor.addAnchorLink(al);
        otherAnchor.addAnchorLink(al);
        return;
      }
      return;
    }

    if(anchor1.getPointType().isCablePoint() && anchor2.getPointType().isCablePoint()){
      if(anchor1.getPointType().getCableTypeId() !=
anchor2.getPointType().getCableTypeId()){
        return;
      }
      if(anchor1.getAnchorLinks().length >= 2 || anchor2.getAnchorLinks().length >= 2)
        return;

      AnchorLink al = new AnchorLink(anchor1, anchor2, new
CableLineType(anchor1.getPointType().getCableTypeId()));
      anchor1.addAnchorLink(al);
      anchor2.addAnchorLink(al);
    }
  }

  private void preprocessArData(FloorPlanPoint[] points, FloorPlanLine[] lines){
    if(points.length == 0)
      return;

    float maxX = points[0].getX(), minX = points[0].getX(), maxY = points[0].getY(), minY
= points[0].getY();
    for(int i = 1; i < points.length; i += 1){
      float x = points[i].getX();
      if(x > maxX){
        maxX = x;
      }
      if(x < minX){
        minX = x;
      }

      float y = points[i].getY();
      if(y > maxY){
        maxY = y;
```

```
      }
      if(y < minY){
        minY = y;
      }
    }
    float xCenter = (maxX + minX) / 2, yCenter = (maxY + minY) / 2;
    for(int i = 0; i < points.length; i += 1){
      points[i].setX(points[i].getX() - xCenter);
      points[i].setY(points[i].getY() - yCenter);
    }

    double longestLineAngle = 0;
    double longestLineLength = 0;
    for(int pointConnectionIterator = 0; pointConnectionIterator < lines.length;
pointConnectionIterator += 1){
      FloorPlanPoint start = points[lines[pointConnectionIterator].getPoint1Index()];
      FloorPlanPoint end = points[lines[pointConnectionIterator].getPoint2Index()];

      double leftY, rightY;
      if(start.getX() > end.getX()){
        leftY = start.getY();
        rightY = end.getY();
      }else{
        leftY = end.getY();
        rightY = start.getY();
      }
      double absXDiff = Math.abs(start.getX() - end.getX());
      double yDiff = leftY - rightY;
      double angle = Math.atan2(yDiff, absXDiff);
      double lineLength = Math.pow(Math.pow(start.getX() - end.getX(), 2) +
Math.pow(start.getY() - end.getY(), 2), 1/2f);
      if(lineLength > longestLineLength){
        longestLineAngle = angle;
        longestLineLength = lineLength;
      }
    }
    double correctionAngle = -longestLineAngle;
    for (FloorPlanPoint point : points) {
        float x = point.getX();
        float y = point.getY();
        float newX = (float) (x * Math.cos(correctionAngle) - y *
Math.sin(correctionAngle));
        float newY = (float) (x * Math.sin(correctionAngle) + y *
Math.cos(correctionAngle));
        point.setX(newX);
        point.setY(newY);
    }

    floorPlan.setCapture(points, lines);
    saveFloorPlan();
  }

  private void saveFloorPlan(){
    Context context = getApplicationContext();
    floorPlan.Save(context);
  }

  /** Checks if we detected at least one plane. */
  private boolean hasTrackingPlane() {
    for (Plane plane : session.getAllTrackables(Plane.class)) {
      if (plane.getTrackingState() == TrackingState.TRACKING) {
        return true;
      }
    }
    return false;
  }

  private void updateSphericalHarmonicsCoefficients(float[] coefficients) {
    // Pre-multiply the spherical harmonics coefficients before passing them to the
shader. The
    // constants in sphericalHarmonicFactors were derived from three terms:
```

```
    //
    // 1. The normalized spherical harmonics basis functions (y_lm)
    //
    // 2. The lambertian diffuse BRDF factor (1/pi)
    //
    // 3. A <cos> convolution. This is done to so that the resulting function outputs the
irradiance
    // of all incoming light over a hemisphere for a given surface normal, which is what
the shader
    // (environmental_hdr.frag) expects.
    //
    // You can read more details about the math here:
    // https://google.github.io/filament/Filament.html#annex/sphericalharmonics

    if (coefficients.length != 9 * 3) {
      throw new IllegalArgumentException(
              "The given coefficients array must be of length 27 (3 components per 9
coefficients");
    }

    // Apply each factor to every component of each coefficient
    for (int i = 0; i < 9 * 3; ++i) {
      sphericalHarmonicsCoefficients[i] = coefficients[i] * sphericalHarmonicFactors[i /
3];
    }
    pointShader.setVec3Array(
            "u_SphericalHarmonicsCoefficients", sphericalHarmonicsCoefficients);
  }

  /** Configures the session with feature settings. */
  private void configureSession() {
    Config config = session.getConfig();
    config.setLightEstimationMode(Config.LightEstimationMode.ENVIRONMENTAL_HDR);
    if (session.isDepthModeSupported(Config.DepthMode.AUTOMATIC)) {
      config.setDepthMode(Config.DepthMode.AUTOMATIC);
    } else {
      config.setDepthMode(Config.DepthMode.DISABLED);
    }
    //if (instantPlacementSettings.isInstantPlacementEnabled()) {
    //  config.setInstantPlacementMode(InstantPlacementMode.LOCAL_Y_UP);
    //} else {
    config.setInstantPlacementMode(InstantPlacementMode.DISABLED);
    //}
    session.configure(config);
  }
}

/**
 * Associates an Anchor with the trackable it was attached to. This is used to be able to
check
 * whether or not an Anchor originally was attached to an {@link InstantPlacementPoint}.
 */
class WrappedAnchor {
  private Anchor anchor;
  private Trackable trackable;
  private Pose lastKnownPose;

  private IPointType _pointType;
  private List<AnchorLink> _anchorLinks;

  public WrappedAnchor(Anchor anchor, Trackable trackable, IPointType pointType) {
    this.anchor = anchor;
    this.trackable = trackable;
    lastKnownPose = null;
    _pointType = pointType;
    _anchorLinks = new ArrayList<>();
  }

  public Anchor getAnchor() {
    return anchor;
  }
```

```
  public Trackable getTrackable() {
    return trackable;
  }

  public Pose getLastKnownPose(){
    return lastKnownPose;
  }

  public void setLastKnownPose(Pose newLastPose){
    lastKnownPose = newLastPose;
  }

  public IPointType getPointType(){
    return _pointType;
  }

  public void setPointType(IPointType pointType){
    _pointType = pointType;
  }

  public AnchorLink[] getAnchorLinks(){
    return _anchorLinks.stream().toArray(AnchorLink[]::new);
  }

  public void addAnchorLink(AnchorLink link){
    _anchorLinks.add(link);
  }

  public void removeAnchorLink(AnchorLink link){
    _anchorLinks.remove(link);
  }
}

class AnchorLink{
  private WrappedAnchor _anchor1;
  private WrappedAnchor _anchor2;
  private ILineType _lineType;

  public AnchorLink(WrappedAnchor anchor1, WrappedAnchor anchor2, ILineType lineType){
    _anchor1 = anchor1;
    _anchor2 = anchor2;
    _lineType = lineType;
  }

  public WrappedAnchor getAnchor1(){
    return _anchor1;
  }

  public WrappedAnchor getAnchor2(){
    return _anchor2;
  }

  public ILineType getLineType(){
    return _lineType;
  }
}
```

В8. Код контролера екрану створення плану.

```
package com.google.ar.core.examples.java.helloar;

import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.WindowManager;
import android.widget.EditText;
```

```
import android.widget.ImageButton;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Arrays;

public class CreateActivity extends AppCompatActivity {
    private static final int AR_MEASURE = 0;
    private static final int SAVE = 1;
    private static final int RETURN = 3;
    private static final int GO_TO_PLAN = 4;
    final Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create);
        getWindow().setAttributes(new
WindowManager.LayoutParams(WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER
));

        ImageButton createButton = findViewById(R.id.create_button);
        createButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        goToFloorPlanView();
                    }
                });
    }

    private void goToFloorPlanView(){
        EditText nameEdit = findViewById(R.id.editTextCreateName);
        String name = nameEdit.getText().toString();
        Context context = getApplicationContext();
        IdStorage idStorage = IdStorage.Load(context);
        if(idStorage == null)
            idStorage = new IdStorage();

        int[] floorPlanIds = idStorage.getFloorPlanIds();
        int freeId = 0;
        for(;freeId <= floorPlanIds.length; freeId++){
            final int tempFreeId = freeId;
            if(Arrays.stream(floorPlanIds).noneMatch(x -> x == tempFreeId))
                break;
        }
        idStorage.addFloorPlanId(freeId);
        FloorPlan newFloorPlan = new FloorPlan(freeId, name);
        if(newFloorPlan.Save(context)){
            if(idStorage.Save(context)) {
                Intent intent = new Intent(CreateActivity.this, FloorPlanActivity.class);
                intent.putExtra("ID", freeId);
                setResult(GO_TO_PLAN, intent);
                finish();
            }
        }
    }
}
```

В9. Код контролера екрану створення типу мережі.

```
package com.google.ar.core.examples.java.helloar;

import android.content.Context;
import android.content.DialogInterface;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.WindowManager;
import android.widget.EditText;
import android.widget.ImageButton;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Arrays;

public class CreateCableTypeActivity extends AppCompatActivity {
    final Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_createcabletype);
        getWindow().setAttributes(new
WindowManager.LayoutParams(WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER
));

        ImageButton createButton = findViewById(R.id.create_button);
        createButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        createAndReturnToListView();
                    }
                });
    }

    private void createAndReturnToListView(){
        EditText nameEdit = findViewById(R.id.editTextCreateName);
        String name = nameEdit.getText().toString();
        Context context = getApplicationContext();
        IdStorage idStorage = IdStorage.Load(context);
        if(idStorage == null)
            idStorage = new IdStorage();

        int[] cableTypeIds = idStorage.getCableTypeIds();
        int freeId = 0;
        for(;freeId <= cableTypeIds.length; freeId++){
            final int tempFreeId = freeId;
            if(Arrays.stream(cableTypeIds).noneMatch(x -> x == tempFreeId))
                break;
        }
        idStorage.addCableTypeId(freeId);
        CableType newCableType = new CableType(freeId, name);
        if(newCableType.Save(context)){
            if(idStorage.Save(context)) {
                Intent intent = new Intent(CreateCableTypeActivity.this,
CableTypeListActivity.class);
                setResult(RESULT_OK, intent);
                finish();
            }
        }
    }
}
```

В10. Код контролера екрану редагування типу мережі.

```
package com.google.ar.core.examples.java.helloar;
```

```
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.view.WindowManager;
import android.widget.EditText;
import android.widget.ImageButton;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import java.util.Arrays;

public class EditCableTypeActivity extends AppCompatActivity {
    private static final int RETURN = 3;
    private CableType cabletype;
    int id;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_editcabletype);
        getWindow().setAttributes(new
WindowManager.LayoutParams(WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER
));

        Intent intent = getIntent();
        id = intent.getIntExtra("ID", -1);

        EditText nameEdit = findViewById(R.id.editTextCreateName);
        Context context = getApplicationContext();

        cabletype = CableType.Load(context, id);
        if(cabletype == null){
            Intent intent2 = new Intent(EditCableTypeActivity.this,
CableTypeListActivity.class);
            setResult(RETURN, intent2);
            finish();
        }
        nameEdit.setText(cabletype.getName());
        nameEdit.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int
after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                cabletype.setName(s.toString());
                saveCableType();
            }

            @Override
            public void afterTextChanged(Editable s) {
            }
        });
    }

    private void saveCableType(){
        Context context = getApplicationContext();
        cabletype.Save(context);
    }
}
```

В11. Код сутності плану.

```
package com.google.ar.core.examples.java.helloar;

import android.content.Context;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class FloorPlan implements Serializable {
    private int _id;
    private boolean isCaptured = false;
    private String _name = "";
    private FloorPlanPoint[] _points = null;
    private FloorPlanLine[] _lines = null;

    private int[] _cableTypeIds = null;

    public FloorPlan(int id){
        _id = id;
    }

    public FloorPlan(int id, String name){
        _name = name;
        _id = id;
    }

    public int getId() {
        return _id;
    }

    public boolean captureSet(){
        return isCaptured;
    }

    public String getName(){
        return _name;
    }

    public void setName(String newName){
        _name = newName;
    }

    public void setCapture(FloorPlanPoint[] points, FloorPlanLine[] lines){
        _points = points;
        _lines = lines;
        isCaptured = _points != null;
    }

    public void setCableTypeIds(int[] cableTypeIds){
        _cableTypeIds = cableTypeIds;
    }

    public void addCableTypeId(int id) {
        if(_cableTypeIds != null && Arrays.stream(_cableTypeIds).anyMatch(x -> x == id))
            return;

        if(_cableTypeIds == null)
            _cableTypeIds = new int[0];

        int[] newCableTypeIds = new int[_cableTypeIds.length + 1];
        for(int i = 0; i < _cableTypeIds.length; i++){
            newCableTypeIds[i]= _cableTypeIds[i];
        }
        newCableTypeIds[newCableTypeIds.length - 1] = id;
```

```
        _cableTypeIds = newCableTypeIds;
    }

    public void removeCableTypeId(int id) {
        if(_cableTypeIds == null)
            return;

        _cableTypeIds = Arrays.stream(_cableTypeIds).filter(x -> x != id).toArray();
        if(isCaptured == false || _points.length == 0)
            return;

        ArrayList<ArrayList<ILineType>> lineMatrix = new ArrayList<>(_points.length);
        for(int i = 0; i < _points.length; i++){
            ArrayList<ILineType> column = new ArrayList<>(_points.length);
            lineMatrix.add(column);
            for(int j = 0; j < _points.length; j++){
                column.add(null);
            }
        }

        for(FloorPlanLine line : _lines){
            if(line.getLineType().isWall() || line.getLineType().getCableTypeId() != id)
{
                lineMatrix.get(line.getPoint1Index()).set(line.getPoint2Index(),
line.getLineType());
                lineMatrix.get(line.getPoint2Index()).set(line.getPoint1Index(),
line.getLineType());
            }
        }

        List<FloorPlanPoint> pointsToRemove1 = Arrays.stream(_points).filter(x ->
x.getPointType().isCablePoint() && x.getPointType().getCableTypeId() ==
id).collect(Collectors.toList());
        List<FloorPlanPoint> pointsList = new ArrayList<>(_points.length);
        for(FloorPlanPoint point : _points){
            pointsList.add(point);
        }
        for(FloorPlanPoint pointToRemove: pointsToRemove1){
            int index = pointsList.indexOf(pointToRemove);
            lineMatrix.remove(index);
            for(ArrayList<ILineType> sublist : lineMatrix){
                sublist.remove(index);
            }
            pointsList.remove(index);
        }
        List<FloorPlanPoint> pointsToRemove2 = new ArrayList<>();
        for(int i = 0; i < lineMatrix.size(); i++){
            if(lineMatrix.get(i).stream().allMatch(x -> x == null)){
                pointsToRemove2.add(pointsList.get(i));
            }
        }
        for(FloorPlanPoint pointToRemove: pointsToRemove2){
            int index = pointsList.indexOf(pointToRemove);
            lineMatrix.remove(index);
            for(ArrayList<ILineType> sublist : lineMatrix){
                sublist.remove(index);
            }
            pointsList.remove(index);
        }
        _points = pointsList.stream().toArray(FloorPlanPoint[]::new);
        ArrayList<FloorPlanLine> linesList = new ArrayList<>();
        for(int i = 0; i < lineMatrix.size(); i++){
            for(int j = 0; j < i; j++){
                ILineType lineType = lineMatrix.get(i).get(j);
                if(lineType != null){
                    linesList.add(new FloorPlanLine(i, j, lineType));
                }
            }
        }
        _lines = linesList.stream().toArray(FloorPlanLine[]::new);
    }
```

```
    public FloorPlanPoint[] getPoints(){
        return _points;
    }

    public FloorPlanLine[] getLines(){
        return _lines;
    }

    public int[] getCableTypeIds(){
        return _cableTypeIds;
    }

    public boolean Save(Context context){
        File file = new File(context.getFilesDir(), "fp_"+_id);
        if (file.exists()) {
            file.delete();
        }
        try {
            file.createNewFile();
            FileOutputStream fos = new FileOutputStream(file);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(this);
            fos.close();
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    public static FloorPlan Load(Context context, int id){
        File file = new File(context.getFilesDir(), "fp_"+id);
        if(file.exists()) {
            try {
                FileInputStream fis = new FileInputStream(file);
                ObjectInputStream ois = new ObjectInputStream(fis);
                Object result = ois.readObject();
                if(result.getClass() == FloorPlan.class) {
                    return (FloorPlan)result;
                }
            } catch (Exception e) {
            }
        }
        return null;
    }

    public static void Delete(Context context, int id){
        File file = new File(context.getFilesDir(), "fp_"+id);
        if(file.exists()) {
            file.delete();
        }
    }
}
```

В12. Код контролера екрану плану.

```
package com.google.ar.core.examples.java.helloar;

import static
android.view.WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_DEFAULT;
import static
android.view.WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER;

import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.res.Configuration;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
```

```
import android.graphics.Paint;
import android.net.Uri;
import android.opengl.EGL14;
import android.opengl.EGLConfig;
import android.opengl.EGLContext;
import android.opengl.EGLDisplay;
import android.opengl.EGLSurface;
import android.opengl.GLES20;
import android.opengl.GLES30;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.WindowManager;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.PopupMenu;
import android.widget.RelativeLayout;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.nio.IntBuffer;

public class FloorPlanActivity extends AppCompatActivity {
    private static final int AR_MEASURE = 0;
    private static final int SAVE = 1;
    private static final int RETURN = 3;
    private static final int CHANGE_CABLE_TYPES = 7;
    private FloorPlan floorPlan;
    private int id;
    final Handler handler = new Handler();

    int bitmapWidth = 0;
    int bitmapHeight = 0;

    ScrollableViewOnTouchListener imgViewTouchListener;

    float minimumPixelsPerMeterNoSwap = 0;
    float minimumPixelsPerMeterSwapXY = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_floorplan);
        getWindow().setAttributes(new
WindowManager.LayoutParams(WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER
));

        ImageButton arActivityButton = findViewById(R.id.ar_activity_button);
        arActivityButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        callArView();
                    }
                });
        ImageButton saveButton = findViewById(R.id.save_button);
        saveButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        callSaveDialog();
```

```
                }
            });
        ImageButton cableTypesButton = findViewById(R.id.cableTypes_button);
        cableTypesButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        callCableTypeList();
                    }
                });
        ImageView imgView = findViewById(R.id.imageView);
        imgView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showPopupMenu(imgView);
            }
        });

        Intent intent = getIntent();
        id = intent.getIntExtra("ID", -1);

        EditText nameEdit = findViewById(R.id.editTextText);
        Context context = getApplicationContext();

        floorPlan = FloorPlan.Load(context, id);
        if(floorPlan == null){
            returnToList();
        }
        nameEdit.setText(floorPlan.getName());
        nameEdit.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int
after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                floorPlan.setName(s.toString());
                saveFloorPlan();
            }

            @Override
            public void afterTextChanged(Editable s) {
            }
        });

        imgViewTouchListener = new ScrollableViewOnTouchListener(imgView);

        imgView.setOnTouchListener(imgViewTouchListener);

        clearImage();
        if(floorPlan.captureSet()){
            calculateMinimumPPMIfZero();
            handler.postDelayed(this::drawImage, 200);
        }
    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        getWindow().getAttributes().layoutInDisplayCutoutMode =
LAYOUT_IN_DISPLAY_CUTOUT_MODE_DEFAULT;
    }

    private class ScrollableViewOnTouchListener implements View.OnTouchListener{
        public ScrollableViewOnTouchListener(ImageView imgView){
            imageView = imgView;
        }
        ImageView imageView;
        float downX, downY;
        int totalX, totalY;
```

```
        private int getRestrictScrollDistance(int desiredScroll, int currentScroll, int
maxScroll){
            if(desiredScroll < 0){
                if(desiredScroll + currentScroll < -maxScroll)
                    return -maxScroll - currentScroll;

                return desiredScroll;
            }else {
                if(desiredScroll + currentScroll > maxScroll)
                    return maxScroll - currentScroll;

                return desiredScroll;
            }
        }

        private int getXScrollRestriction(){
            int imageViewWidth = imageView.getWidth();
            int scrollRestriction = (FloorPlanActivity.this.bitmapWidth - imageViewWidth)
/ 2;
            if(scrollRestriction < 0)
                return 0;

            return scrollRestriction;
        }

        private int getYScrollRestriction(){
            int imageViewHeight = imageView.getHeight();
            int scrollRestriction = (FloorPlanActivity.this.bitmapHeight -
imageViewHeight) / 2;
            if(scrollRestriction < 0)
                return 0;

            return scrollRestriction;
        }

        public void resetScroll(){
            imageView.scrollBy(-totalX, -totalY);
            totalX = 0;
            totalY = 0;
        }

        public boolean onTouch(View view, MotionEvent event)
        {
            switch (event.getAction())
            {
                case MotionEvent.ACTION_DOWN:
                    downX = event.getX();
                    downY = event.getY();
                    break;

                case MotionEvent.ACTION_MOVE:
                    float currentX, currentY;
                    currentX = event.getX();
                    currentY = event.getY();
                    int scrollByX = getRestrictScrollDistance(Math.round(downX -
currentX), totalX, getXScrollRestriction());
                    int scrollByY = getRestrictScrollDistance(Math.round(downY -
currentY), totalY, getYScrollRestriction());

                    imageView.scrollBy(scrollByX, scrollByY);
                    totalX += scrollByX;
                    totalY += scrollByY;
                    downX = currentX;
                    downY = currentY;
                    break;

            }
            return true;
        }
    }
```

```
@Override
public void onConfigurationChanged(Configuration newConfig){
    super.onConfigurationChanged(newConfig);
    clearImage();
    if(floorPlan.captureSet()){
        calculateMinimumPPMIfZero();
        handler.postDelayed(this::drawImage, 200);
    }
}

private void showPopupMenu(View v) {
    PopupMenu popupMenu = new PopupMenu(FloorPlanActivity.this, v);
    popupMenu.inflate(R.menu.main_menu);
    popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener()
    {
        @Override
        public boolean onMenuItemClick (MenuItem item) {
            int itemId = item.getItemId();
            if (itemId == R.id.ar_activity_item){
                callArView();
            }else if(itemId == R.id.save_item){
                callSaveDialog();
            }else{
                return false;
            }
            return true;
        }
    });
    popupMenu.show();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if(id == R.id.ar_activity_item){
        callArView();
        return true;
    }
    if(id == R.id.save_item){
        callSaveDialog();
        return true;
    }
    return super.onOptionsItemSelected(item);
}

@Override
protected void onResume() {
    super.onResume();
    Context context = getApplicationContext();
    floorPlan = FloorPlan.Load(context, id);
    clearImage();
    resetMinimumPPM();
    if(floorPlan.captureSet()) {
        calculateMinimumPPMIfZero();
        handler.postDelayed(this::drawImage, 200);
    }
}

protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
   if(requestCode == SAVE && resultCode == RESULT_OK && floorPlan.captureSet()) {
        Uri uri = data.getData();
        try {
            Bitmap bmp = renderWithAutoscaleToAtLeast(500, 500, 50, 50);
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            bmp.compress(Bitmap.CompressFormat.PNG, 0, bos);
            byte[] bitmapdata = bos.toByteArray();
```

```
                OutputStream output =
getApplicationContext().getContentResolver().openOutputStream(uri);
                output.write(bitmapdata);
                output.flush();
                output.close();
            }
            catch(IOException e) {
                Toast.makeText(getApplicationContext(), "Error",
Toast.LENGTH_SHORT).show();
            }
        }
    }

    private void callArView(){
        AlertDialog.Builder builder = new AlertDialog.Builder(FloorPlanActivity.this);
        if(!floorPlan.captureSet()){
            Intent intent = new Intent(FloorPlanActivity.this, CameraActivity.class);
            intent.putExtra("ID", id);
            startActivityForResult(intent, AR_MEASURE);
        }else{
            builder.setTitle("Попередження")
                    .setMessage("Повторне захоплення призведе до втрати існуючого плану")
                    .setCancelable(true)
                    .setNegativeButton("Ok",
                            new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int id) {
                                    dialog.cancel();
                                    Intent intent = new Intent(FloorPlanActivity.this,
CameraActivity.class);
                                    intent.putExtra("ID", FloorPlanActivity.this.id);
                                    startActivityForResult(intent, AR_MEASURE);
                                }
                            });
            AlertDialog alert = builder.create();
            alert.show();
        }
    }

    private void callCableTypeList(){
        Intent intent = new Intent(FloorPlanActivity.this,
FloorPlanCableTypeListActivity.class);
        intent.putExtra("ID", id);
        startActivityForResult(intent, CHANGE_CABLE_TYPES);
    }

    private void callSaveDialog(){
        if(!floorPlan.captureSet()){
            Toast.makeText(this, "Даних в плані немає", Toast.LENGTH_SHORT).show();
            return;
        }

        Intent intent = new Intent(Intent.ACTION_CREATE_DOCUMENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        intent.setType("image/*");
        String name = ((EditText)findViewById(R.id.editTextText)).getText() + ".png";
        intent.putExtra(Intent.EXTRA_TITLE, name);
        startActivityForResult(intent, SAVE);
    }

    private void returnToList(){
        Intent intent = new Intent(FloorPlanActivity.this, FloorPlanListActivity.class);
        setResult(RETURN, intent);
        finish();
    }

    private void saveFloorPlan(){
        Context context = getApplicationContext();
        floorPlan.Save(context);
    }

    private void clearImage(){
```

```
        imgViewTouchListener.resetScroll();
        bitmapWidth = 0;
        bitmapHeight = 0;
    }

    private void resetMinimumPPM(){
        minimumPixelsPerMeterNoSwap = 0;
        minimumPixelsPerMeterSwapXY = 0;
    }

    private void calculateMinimumPPMIfZero(){
        if(minimumPixelsPerMeterNoSwap != 0 && minimumPixelsPerMeterSwapXY != 0)
            return;

        minimumPixelsPerMeterNoSwap = (float)getMinimumPixelsPerMeterToFitText(false);
        minimumPixelsPerMeterSwapXY = (float)getMinimumPixelsPerMeterToFitText(true);
    }

    private void drawImage(){
        imgViewTouchListener.resetScroll();
        ImageView iv = findViewById(R.id.imageView);
        int w = iv.getWidth();
        int h = iv.getHeight();
        if (w < 1 || h < 1)
            return;

        Bitmap bmp = renderWithAutoscaleToAtLeast(w, h, 50, 50);
        bitmapWidth = bmp.getWidth();
        bitmapHeight = bmp.getHeight();
        iv.setImageBitmap(bmp);
    }

    private static double quaterTurn = Math.PI / 2;

    private double getMinimumPixelsPerMeterToFitText(boolean swapXY){
        float textWidthMultiplier = 1.2f;
        float textHeightMultiplier = 2.2f;
        float textHeight = 20;
        textHeight *= textHeightMultiplier;
        Paint textPaint = new Paint();
        textPaint.setColor(Color.WHITE);
        textPaint.setStyle(Paint.Style.STROKE);
        textPaint.setStrokeWidth(1F);
        textPaint.setAntiAlias(true);
        textPaint.setTextSize(textHeight);

        FloorPlanPoint[] pointsArray = floorPlan.getPoints();
        float[][] points = new float[pointsArray.length][];
        for(int i = 0; i < pointsArray.length; i += 1){
            float xFloat;
            float yFloat;
            if(swapXY){
                xFloat = pointsArray[i].getY();
                yFloat = -pointsArray[i].getX();
            }else{
                xFloat = pointsArray[i].getX();
                yFloat = pointsArray[i].getY();
            }
            float zFloat = pointsArray[i].getPointType().isWallPoint() ? 0 :
pointsArray[i].getZ(); //take actual value only for cables
            points[i] = new float[] {xFloat, yFloat, zFloat};
        }

        FloorPlanLine[] lines = floorPlan.getLines();

        float minimumPPM = 1;

        for(int i = 0; i <  lines.length; i++){
            float[] start = points[lines[i].getPoint1Index()];
            float[] end = points[lines[i].getPoint2Index()];
            float[] center = {(start[0] + end[0]) / 2, (start[1] + end[1]) / 2};
```

```
            double leftY, rightY;
            if(start[0] > end[0]){
                leftY = start[1];
                rightY = end[1];
            }else{
                leftY = end[1];
                rightY = start[1];
            }
            double absXDiff = Math.abs(start[0] - end[0]);
            double yDiff = leftY - rightY;
            double angle = Math.atan2(yDiff, absXDiff);

            double[] length1TextUpVector = new double[] {Math.cos(angle - quaterTurn),
Math.sin(angle - quaterTurn)};
            double[] length1TextRightVector = new double[] {Math.cos(angle),
Math.sin(angle)};

            double lineLength = Math.pow(Math.pow(start[0] - end[0], 2) +
Math.pow(start[1] - end[1], 2), 1/2f);
            double textLineLength = Math.pow(Math.pow(start[0] - end[0], 2) +
Math.pow(start[1] - end[1], 2) + Math.pow(start[2] - end[2], 2), 1/2f);

            String str = String.format("%.2f м", textLineLength);
            float textWidth = textPaint.measureText(str);
            textWidth *= textWidthMultiplier;

            if(textWidth/lineLength > minimumPPM){
                minimumPPM = (float)(textWidth / lineLength);
            }


            float[] rectangleLeftBottom = new float[] {(float)(center[0] -
length1TextRightVector[0] * (textWidth / 2)), (float)(center[1] -
length1TextRightVector[1] * (textWidth / 2))};
            float[] rectangleLeftTop = new float[] {(float)(rectangleLeftBottom[0] +
length1TextUpVector[0] * (textHeight + 10)), (float)(rectangleLeftBottom[1] +
length1TextUpVector[1] * (textHeight + 10))};
            float[] rectangleRightTop = new float[] {(float)(rectangleLeftTop[0] +
length1TextRightVector[0] * (textWidth)), (float)(rectangleLeftTop[1] +
length1TextRightVector[1] * (textWidth))};
            float[] rectangleRightBottom = new float[] {(float)(rectangleRightTop[0] -
length1TextUpVector[0] * (textHeight + 10)), (float)(rectangleRightTop[1] -
length1TextUpVector[1] * (textHeight + 10))};

            float[][][] rectangleLines = new float[][][] {new float[][]
{rectangleLeftBottom, rectangleLeftTop}, new float[][] {rectangleLeftTop,
rectangleRightTop}, new float[][] {rectangleRightTop, rectangleRightBottom}, new
float[][] {rectangleRightBottom, rectangleLeftBottom}};
            float[][] endsToCheck = new float[][] {rectangleLeftBottom, rectangleLeftTop,
rectangleRightTop, rectangleRightBottom};
            for(int j = 0; j < lines.length; j++){
                if(j == i)
                    continue;

                float[] partnerStart = points[lines[j].getPoint1Index()];
                float[] partnerEnd = points[lines[j].getPoint2Index()];
                float[][] lineToCheck = new float[][] {new float[] {partnerStart[0],
partnerStart[1]}, new float[] {partnerEnd[0], partnerEnd[1]}};
                for (float[] endToCheck: endsToCheck) {
                    float[][] checkingLine = new float[][] {center, endToCheck};
                    float[] intersection = getIntersectionOfLines(checkingLine,
lineToCheck);
                    if(intersection == null)
                        continue;

                    if(!IsPointIsOnCorrectSideOfRay(checkingLine, intersection))
                        continue;

                    if(!IsPointInSegmentOfLine(lineToCheck, intersection))
                        continue;
```

```
                    float checkingLineLength = getDistanceBetweenPoints(checkingLine[0],
checkingLine[1]);
                    float distanceToIntersection =
getDistanceBetweenPoints(checkingLine[0], intersection);
                    if(distanceToIntersection <= Math.pow(10, -3))
                        continue;

                    float requiredPPM = checkingLineLength / distanceToIntersection;
                    if(requiredPPM > minimumPPM)
                        minimumPPM = requiredPPM;
                }

                for(float[] lineToCheckEnd: lineToCheck){
                    float[][] checkingLine = new float[][] {center, lineToCheckEnd};
                    for(float[][] rectangleLine: rectangleLines){
                        float[] intersection = getIntersectionOfLines(checkingLine,
rectangleLine);
                        if(intersection == null)
                            continue;

                        if(!IsPointIsOnCorrectSideOfRay(checkingLine, intersection))
                            continue;

                        float distanceToIntersection =
getDistanceBetweenPoints(checkingLine[0], intersection);
                        float checkingLineLength =
getDistanceBetweenPoints(checkingLine[0], checkingLine[1]);
                        if(distanceToIntersection <= Math.pow(10, -3))
                            continue;

                        float requiredPPM = checkingLineLength / distanceToIntersection;
                        if(requiredPPM > minimumPPM)
                            minimumPPM = requiredPPM;
                    }
                }
            }
        }
        return minimumPPM;
    }

    static float[] getIntersectionOfLines(float[][] line1, float[][] line2){
        float intersectionDenominator = (line1[0][0] - line1[1][0]) * (line2[0][1] -
line2[1][1]) - (line1[0][1] - line1[1][1]) * (line2[0][0] - line2[1][0]);
        if(intersectionDenominator == 0)
            return null;

        float coordinateMultiplier1 = (line1[0][0]*line1[1][1] -
line1[0][1]*line1[1][0]);
        float coordinateMultiplier2 = (line2[0][0] * line2[1][1] - line2[0][1] *
line2[1][0]);
        float intersectionX = (coordinateMultiplier1 * (line2[0][0] - line2[1][0]) -
(line1[0][0] - line1[1][0]) * coordinateMultiplier2) / intersectionDenominator;
        float intersectionY = (coordinateMultiplier1 * (line2[0][1] - line2[1][1]) -
(line1[0][1] - line1[1][1]) * coordinateMultiplier2) / intersectionDenominator;
        return new float[] {intersectionX, intersectionY};
    }

    static float getDistanceBetweenPoints(float[] point1, float[] point2){
        return (float)Math.pow(Math.pow(point1[0] - point2[0], 2) + Math.pow(point1[1] -
point2[1], 2), 1/2f);
    }

    static boolean IsPointIsOnCorrectSideOfRay(float[][] ray, float[] point){
        float marginForError = 1.01f; //because floating point math
        float segmentLength = getDistanceBetweenPoints(ray[0], ray[1]);
        float distanceToStart = getDistanceBetweenPoints(ray[0], point);
        float distanceToDirectionPoint = getDistanceBetweenPoints(ray[1], point);
        if(distanceToDirectionPoint <= distanceToStart * marginForError){
            return true;
        }
```

```
        return distanceToStart + distanceToDirectionPoint <= segmentLength *
marginForError;
    }

    static boolean IsPointInSegmentOfLine(float[][] line, float[] point){
        float marginForError = 1.01f; //because floating point math
        float segmentLength = getDistanceBetweenPoints(line[0], line[1]);
        float sumOfDistancesToPoint = getDistanceBetweenPoints(line[0], point) +
getDistanceBetweenPoints(line[1], point);
        return sumOfDistancesToPoint <= segmentLength * marginForError;
    }

    private Bitmap renderWithAutoscaleToAtLeast(int minW, int minH, int hPadding, int
wPadding){
        int maxTextureSize = getMaxTextureSize();

        float minAllowedPPM = 50;
        float maxAllowedPPM = 3000;
        float textHeight = 20;

        float xMax = Float.NEGATIVE_INFINITY;
        float xMin = Float.POSITIVE_INFINITY;
        float yMax = Float.NEGATIVE_INFINITY;
        float yMin = Float.POSITIVE_INFINITY;
        FloorPlanPoint[] pointsArray = floorPlan.getPoints();
        for(FloorPlanPoint f : pointsArray){
            if(f.getX() > xMax){
                xMax = f.getX();
            }
            if(f.getX() < xMin){
                xMin = f.getX();
            }
            if(f.getY() > yMax){
                yMax = f.getY();
            }
            if(f.getY() < yMin){
                yMin = f.getY();
            }
        }

        float planWidth = xMax - xMin;
        float planHeight = yMax - yMin;

        boolean swapXAndY;
        if(planWidth > planHeight){
            swapXAndY = minH > minW;
        }else{
            swapXAndY = minH < minW;
        }

        if(swapXAndY){
            float oldXMin = xMin;
            float oldXMax = xMax;
            xMin = yMin;
            xMax = yMax;
            yMin = -oldXMax;
            yMax = -oldXMin;

            float oldWidth = planWidth;
            planWidth = planHeight;
            planHeight = oldWidth;
        }

        float a = planWidth * planHeight;
        float b = planWidth * 2 * hPadding + planHeight * 2 * wPadding;
        float c = 4 * hPadding * wPadding - maxTextureSize;

        float d = b*b - 4 * a * c;
        float sqrtd = (float)Math.sqrt(d);

        float maxPPM1 = (-b + sqrtd) / (2 * a);
```

```
        float maxPPM2 = (-b + sqrtd) / (2 * a);
        float maxPPM = Math.max(maxPPM1, maxPPM2);

        float minPPMByW = planWidth == 0 ? 0 : minW / planWidth;
        float minPPMByH = planHeight == 0 ? 0 : minH / planHeight;

        float textMinPPM = swapXAndY ? minimumPixelsPerMeterSwapXY :
minimumPixelsPerMeterNoSwap;

        float pixelsPerMeter = Math.max(textMinPPM, Math.min(minPPMByW, minPPMByH));

        if(pixelsPerMeter < minAllowedPPM){
            pixelsPerMeter = minAllowedPPM;
        }else if (pixelsPerMeter > maxAllowedPPM){
            pixelsPerMeter = maxAllowedPPM;
        }

        pixelsPerMeter = Math.min(pixelsPerMeter, maxPPM);

        int w = (int)Math.floor(planWidth * pixelsPerMeter);
        int h = (int)Math.floor(planHeight * pixelsPerMeter);

        w += wPadding * 2;
        h += hPadding * 2;

        if(w < minW)
            w = minW;

        if(h < minH)
            h = minH;

        Bitmap bmp = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);

        Coordinates[] coordinates = new Coordinates[pointsArray.length];
        int xAdjustment = Math.round(-xMin * pixelsPerMeter) + wPadding, yAdjustment =
Math.round(-yMin * pixelsPerMeter) + hPadding;
        for(int i = 0; i < pointsArray.length; i += 1){
            float xFloat;
            float yFloat;
            if(swapXAndY){
                xFloat = pointsArray[i].getY();
                yFloat = -pointsArray[i].getX();
            }else{
                xFloat = pointsArray[i].getX();
                yFloat = pointsArray[i].getY();
            }
            float zFloat = pointsArray[i].getPointType().isWallPoint() ? 0 :
pointsArray[i].getZ();
            int x = Math.round(xFloat * pixelsPerMeter) + xAdjustment;
            int y = Math.round(yFloat * pixelsPerMeter) + yAdjustment;
            coordinates[i] = new Coordinates(x, y, xFloat, yFloat, zFloat);
        }
        Canvas canvas = new Canvas(bmp);

        Paint wallPaint = new Paint();
        wallPaint.setColor(Color.WHITE);
        wallPaint.setStyle(Paint.Style.STROKE);
        wallPaint.setStrokeWidth(5F);
        wallPaint.setAntiAlias(true);

        Paint cablePaint = new Paint();
        cablePaint.setColor(Color.BLUE);
        cablePaint.setStyle(Paint.Style.STROKE);
        cablePaint.setStrokeWidth(5F);
        cablePaint.setAntiAlias(true);

        Paint textPaint = new Paint();
        textPaint.setColor(Color.WHITE);
        textPaint.setStyle(Paint.Style.STROKE);
        textPaint.setStrokeWidth(1F);
        textPaint.setAntiAlias(true);
```

```
        textPaint.setTextSize(textHeight);

        Paint blackPaint = new Paint();
        blackPaint.setColor(Color.BLACK);
        blackPaint.setStyle(Paint.Style.FILL);
        blackPaint.setStrokeWidth(10F);
        blackPaint.setAntiAlias(true);

        FloorPlanLine[] lines = floorPlan.getLines();

        for(int i = 0; i <  lines.length; i++){
            Coordinates start = coordinates[lines[i].getPoint1Index()];
            Coordinates end = coordinates[lines[i].getPoint2Index()];
            canvas.drawLine(start.x, start.y, end.x, end.y,
lines[i].getLineType().isWall() ? wallPaint : cablePaint);

            int[] center = {(start.x + end.x) / 2, (start.y + end.y) / 2};
            double leftY, rightY;
            if(start.x > end.x){
                leftY = start.y;
                rightY = end.y;
            }else{
                leftY = end.y;
                rightY = start.y;
            }
            double absXDiff = Math.abs(start.x - end.x);
            double yDiff = leftY - rightY;
            double angle = Math.atan2(yDiff, absXDiff);
            double angleDegrees = angle / Math.PI * 180;

            double textLineLength = Math.pow(Math.pow(start.xMeters - end.xMeters, 2) +
Math.pow(start.yMeters - end.yMeters, 2) + Math.pow(start.zMeters - end.zMeters, 2),
1/2f);

            String str = String.format("%.2f м", textLineLength);
            float textWidth = textPaint.measureText(str);

            canvas.save();
            canvas.rotate((float)angleDegrees, center[0], center[1]);
            canvas.drawText(str,0, str.length(), center[0] - textWidth / 2, center[1] -
(10), textPaint);
            canvas.restore();
        }
        return bmp;
    }

    private static int getMaxTextureSize(){
        return (int)Math.floor(150 * 1024 * 1024 / 4.0);
    }

    private static class Coordinates{
        public int x;
        public int y;
        public float xMeters;
        public float yMeters;
        public float zMeters;

        public Coordinates(int x, int y, float xMeters, float yMeters, float zMeters){
            this.x = x;
            this.y = y;
            this.xMeters = xMeters;
            this.yMeters = yMeters;
            this.zMeters = zMeters;
        }
    }
}
```

В13. Код контролера екрану інженерних мереж плану.

```
package com.google.ar.core.examples.java.helloar;
```

```
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.fonts.Font;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.WindowManager;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;
import java.util.Arrays;

public class FloorPlanCableTypeListActivity extends AppCompatActivity {
    private static final int CREATE = 2;
    private static final int RETURN = 3;
    private CableType[] cableTypes;
    private FloorPlan floorPlan;
    private int id;

    private Font nameFont;
    final Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_floorplancabletypelist);
        getWindow().setAttributes(new
WindowManager.LayoutParams(WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER
));

        ImageButton createActivityButton =
findViewById(R.id.create_cabletype_activity_button);
        createActivityButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        callCreateView();
                    }
                });

        LinearLayout floorPlanList = findViewById(R.id.floor_plan_list);
        Context context = getApplicationContext();

        Intent intent = getIntent();
        id = intent.getIntExtra("ID", -1);

        floorPlan = FloorPlan.Load(context, id);
        if(floorPlan == null){
            Intent returnIntent = new Intent(FloorPlanCableTypeListActivity.this,
CreateCableTypeActivity.class);
            startActivityForResult(returnIntent, RETURN);
        }

        IdStorage idStorage = IdStorage.Load(context);
        if(idStorage == null)
            idStorage = new IdStorage();

        int[] cableTypeIds = idStorage.getCableTypeIds();
        cableTypes = new CableType[cableTypeIds.length];
        for(int i = 0; i < cableTypeIds.length; i++){
            cableTypes[i] = CableType.Load(context, cableTypeIds[i]);
        }
```

```
        cableTypes = Arrays.stream(cableTypes).filter(fp -> fp != null).toArray(size ->
new CableType[size]);
        int i = 0;
        for(CableType cableType : cableTypes) {
            View tv = createViewForType(context, cableType, i);
            floorPlanList.addView(tv);
            i++;
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        LinearLayout floorPlanList = findViewById(R.id.floor_plan_list);
        floorPlanList.removeAllViews();
        Context context = getApplicationContext();

        IdStorage idStorage = IdStorage.Load(context);
        if(idStorage == null)
            idStorage = new IdStorage();

        int[] cableTypeIds = idStorage.getCableTypeIds();
        cableTypes = new CableType[cableTypeIds.length];
        for(int i = 0; i < cableTypeIds.length; i++){
            cableTypes[i] = CableType.Load(context, cableTypeIds[i]);
        }
        cableTypes = Arrays.stream(cableTypes).filter(fp -> fp != null).toArray(size ->
new CableType[size]);
        int i = 0;
        for(CableType cableType : cableTypes) {
            View tv = createViewForType(context, cableType, i);
            floorPlanList.addView(tv, new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.WRAP_CONTENT));
            i++;
        }
    }

    private View createViewForType(Context context, CableType cableType, int index){
        RelativeLayout layout = new RelativeLayout(context);

        CheckBox cb = new CheckBox(context);
        cb.setId(View.generateViewId());
        cb.setContentDescription("type_checkbox_" + cableType.getName());
        TextView tv = new TextView(context);
        tv.setId(View.generateViewId());
        TextView lv = new TextView(context);
        lv.setId(View.generateViewId());
        TextView tlv = new TextView(context);
        tlv.setId(View.generateViewId());

        if(floorPlan.getCableTypeIds() != null){
            for(int id : floorPlan.getCableTypeIds()){
                if(id == cableType.getId()){
                    cb.setChecked(true);
                    break;
                }
            }
        }
        RelativeLayout.LayoutParams cbRelativeParams = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        cbRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_LEFT);
        cbRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
        cbRelativeParams.addRule(RelativeLayout.ALIGN_TOP, tv.getId());
        cbRelativeParams.addRule(RelativeLayout.ALIGN_BOTTOM, tv.getId());
        layout.addView(cb, cbRelativeParams);

        tv.setText(cableType.getName());
        tv.setTextColor(Color.WHITE);
        tv.setTextSize(20);
```

```
        tv.setPadding(0, 25, 0, 25);

        RelativeLayout.LayoutParams tvRelativeParams = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        tvRelativeParams.addRule(RelativeLayout.RIGHT_OF, cb.getId());
        tvRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
        layout.addView(tv, tvRelativeParams);

        lv.setText(cableType.getName());
        lv.setTextColor(Color.WHITE);
        lv.setTextSize(20);
        lv.setPadding(0, 25, 0, 25);
        lv.setMaxLines(1000);

        RelativeLayout.LayoutParams lvRelativeParams = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        lvRelativeParams.addRule(RelativeLayout.ALIGN_LEFT, tv.getId());
        lvRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_BOTTOM);
        lvRelativeParams.addRule(RelativeLayout.BELOW, tv.getId());
        lvRelativeParams.addRule(RelativeLayout.BELOW, cb.getId());
        lvRelativeParams.addRule(RelativeLayout.BELOW, tlv.getId());
        layout.addView(lv, lvRelativeParams);

        float totalLength = 0;
        ArrayList<CableSegment> segments = new ArrayList<>();
        if(floorPlan.captureSet()) {
            FloorPlanLine[] lines = floorPlan.getLines();
            FloorPlanPoint[] points = floorPlan.getPoints();
            for (FloorPlanLine line : lines){
                if(line.getLineType().isCable() && line.getLineType().getCableTypeId() ==
cableType.getId()){
                    float length =
(float)Math.pow(Math.pow(points[line.getPoint1Index()].getX() -
points[line.getPoint2Index()].getX(), 2) + Math.pow(points[line.getPoint1Index()].getY()
- points[line.getPoint2Index()].getY(), 2) +
Math.pow(points[line.getPoint1Index()].getZ() - points[line.getPoint2Index()].getZ(), 2),
1/2f);

                    totalLength += length;
                    CableSegment newSegment = new CableSegment(line.getPoint1Index(),
points[line.getPoint1Index()].getPointType().isCableConnectionPoint(),
line.getPoint2Index(),
points[line.getPoint2Index()].getPointType().isCableConnectionPoint(), length);
                    boolean combined = false;
                    for(CableSegment segment : segments){
                        if(segment.isCombineable(newSegment)){
                            segment.combineWith(newSegment);
                            combined = true;
                            break;
                        }
                    }
                    if(!combined){
                        segments.add(newSegment);
                    }
                    boolean combinedExisting;
                    do{
                        combinedExisting = false;
                        for(int i = 0; i < segments.size() - 1; i++){
                            for(int j = i + 1; j < segments.size();){
                                if(segments.get(i).isCombineable(segments.get(j))){
                                    segments.get(i).combineWith(segments.get(j));
                                    segments.remove(j);
                                    combinedExisting = true;
                                }else{
                                    j++;
                                }
                            }
                        }
                    }while(combinedExisting);
```

```
                }
            }
        }
        Float[] lengths = segments.stream().map(x -> x.getLength()).sorted((a, b) -> -
Float.compare(a, b)).toArray(size -> new Float[size]);
        StringBuilder lvText = new StringBuilder();
        for(float l : lengths){
            String str = String.format("%.2f м", l);
            lvText.append(str).append("
");
        }
        lv.setText(lvText.toString());

        tlv.setTextColor(Color.WHITE);
        tlv.setTextSize(20);lvRelativeParams.addRule(RelativeLayout.BELOW, cb.getId());
        tlv.setPadding(0, 25, 0, 25);

        RelativeLayout.LayoutParams tlvRelativeParams = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        tlvRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
        tlvRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
        layout.addView(tlv, tlvRelativeParams);

        String str = String.format("%.2f м", totalLength);
        tlv.setText(str);

        cb.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                Context context = getApplicationContext();
                if(isChecked){
                    floorPlan.addCableTypeId(cableType.getId());
                    floorPlan.Save(context);
                }else{
                    floorPlan.removeCableTypeId(cableType.getId());
                    String str2 = String.format("%.2f м", 0f);
                    tlv.setText(str2);
                    lv.setText("");
                    floorPlan.Save(context);
                }
            }
        });
        return layout;
    }

    private void callCreateView(){
        Intent intent = new Intent(FloorPlanCableTypeListActivity.this,
CreateCableTypeActivity.class);
        startActivityForResult(intent, CREATE);
    }

    private static class CableSegment{
        private float _length;
        private int _startIndex;
        private boolean _startIsCC;
        private int _endIndex;
        private boolean _endIsCC;

        public CableSegment(int startindex,  boolean startIsCableConnection, int
endIndex, boolean endIsCableConnection, float length){
            _startIndex = startindex;
            _startIsCC = startIsCableConnection;
            _endIndex = endIndex;
            _endIsCC = endIsCableConnection;
            _length = length;
        }

        public float getLength(){
            return _length;
        }
```

```
        public boolean isCombineable(CableSegment other){
            if(!_startIsCC){
                if(_startIndex == other._startIndex || _startIndex == other._endIndex)
                    return true;
            }
            if(!_endIsCC){
                if(_endIndex == other._startIndex || _endIndex == other._endIndex)
                    return true;
            }
            return false;
        }

        public void combineWith(CableSegment other){
            if(_startIndex == other._startIndex){
                _startIndex = other._endIndex;
                _startIsCC = other._endIsCC;
            }else if(_startIndex == other._endIndex){
                _startIndex = other._startIndex;
                _startIsCC = other._startIsCC;
            }else if(_endIndex == other._startIndex){
                _endIndex = other._endIndex;
                _endIsCC = other._endIsCC;;
            }else if(_endIndex == other._endIndex){
                _endIndex = other._startIndex;
                _endIsCC = other._startIsCC;
            }
            _length += other._length;
        }
    }
}
```

В14. Код сутності відрізка.

```
package com.google.ar.core.examples.java.helloar;

import java.io.Serializable;

public class FloorPlanLine implements Serializable {
    private int _point1Index;
    private int _point2Index;
    private ILineType _lineType;

    public FloorPlanLine(int point1Index, int point2Index, ILineType lineType){
        _point1Index = point1Index;
        _point2Index = point2Index;
        _lineType = lineType;
    }

    public int getPoint1Index(){
        return _point1Index;
    }

    public int getPoint2Index(){
        return _point2Index;
    }

    public ILineType getLineType(){
        return _lineType;
    }
}
```

В15. Код контролера головного екрану.

```
package com.google.ar.core.examples.java.helloar;

import android.app.ComponentCaller;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
```

```
import android.graphics.fonts.Font;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.WindowManager;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import java.util.Arrays;

public class FloorPlanListActivity extends AppCompatActivity {
    private static final int AR_MEASURE = 0;
    private static final int SAVE = 1;
    private static final int CREATE = 2;
    private static final int OPEN_FLOORPLAN = 5;

    private static final int GO_TO_PLAN = 4;
    private FloorPlan[] floorPlans;

    private Font nameFont;
    final Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_floorplanlist);
        getWindow().setAttributes(new
WindowManager.LayoutParams(WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER
));

        ImageButton createActivityButton = findViewById(R.id.create_activity_button);
        createActivityButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        callCreateView();
                    }
                });

        ImageButton cableTypeListActivityButton =
findViewById(R.id.cabletypelist_activity_button);
        cableTypeListActivityButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        callCableTypeListView();
                    }
                });

        loadPlanList();
    }

    @Override
    public void onResume() {
        super.onResume();
        loadPlanList();
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data,
@NonNull ComponentCaller caller) {
        super.onActivityResult(requestCode, resultCode, data, caller);
        if(resultCode == GO_TO_PLAN && data != null){
            int id = data.getIntExtra("ID", -1);
```

```
            if(id == -1)
                return;

            Intent intent = new Intent(FloorPlanListActivity.this,
FloorPlanActivity.class);
            intent.putExtra("ID", id);
            startActivityForResult(intent, OPEN_FLOORPLAN);
        }
    }

    private void loadPlanList(){
        LinearLayout floorPlanList = findViewById(R.id.floor_plan_list);
        floorPlanList.removeAllViews();
        Context context = getApplicationContext();

        IdStorage idStorage = IdStorage.Load(context);
        if(idStorage == null)
            idStorage = new IdStorage();

        int[] floorPlanIds = idStorage.getFloorPlanIds();
        floorPlans = new FloorPlan[floorPlanIds.length];
        for(int i = 0; i < floorPlanIds.length; i++){
            floorPlans[i] = FloorPlan.Load(context, floorPlanIds[i]);
        }
        floorPlans = Arrays.stream(floorPlans).filter(fp -> fp != null).toArray(size ->
new FloorPlan[size]);
        int i = 0;
        for(FloorPlan floorPlan : floorPlans) {
            View tv = createViewForPlan(context, floorPlan.getName(), floorPlan.getId());
            floorPlanList.addView(tv);
            i++;
        }
    }

    private View createViewForPlan(Context context, String name, int id){
        RelativeLayout layout = new RelativeLayout(context);

        TextView tv = new TextView(context);
        tv.setId(View.generateViewId());
        tv.setText(name);
        tv.setTextColor(Color.WHITE);
        tv.setTextSize(20);
        tv.setPadding(0, 25, 0, 25);
        tv.setOnClickListener(new FloorPlanCLickListener(id));
        tv.setContentDescription("plan_name_" + name);

        RelativeLayout.LayoutParams tvRelativeParams = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.MATCH_PARENT);
        tvRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_LEFT);
        layout.addView(tv, tvRelativeParams);

        ImageButton deleteButton = new ImageButton(context);
        deleteButton.setId(View.generateViewId());
        deleteButton.setImageResource(R.drawable.trash2);
        deleteButton.setScaleType(ImageView.ScaleType.FIT_CENTER);
        deleteButton.setScaleX(3.5f);
        deleteButton.setScaleY(3.5f);
        deleteButton.setBaselineAlignBottom(false);
        deleteButton.setBackgroundColor(0x00ffffff);
        deleteButton.setContentDescription("plan_delete_" + name);
        deleteButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                IdStorage idStorage = IdStorage.Load(context);
                if(idStorage == null)
                    idStorage = new IdStorage();

                idStorage.removeFloorPlanId(id);
                FloorPlan.Delete(getApplicationContext(), id);
                loadPlanList();
```

```
            }
        });
        RelativeLayout.LayoutParams dbRelativeParams = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.MATCH_PARENT);
        dbRelativeParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
        dbRelativeParams.addRule(RelativeLayout.ALIGN_TOP, tv.getId());
        dbRelativeParams.addRule(RelativeLayout.ALIGN_BOTTOM, tv.getId());
        layout.addView(deleteButton, dbRelativeParams);
        return layout;
    }

    private void callCableTypeListView(){
        Intent intent = new Intent(FloorPlanListActivity.this,
CableTypeListActivity.class);
        startActivity(intent);
    }

    private void callCreateView(){
        Intent intent = new Intent(FloorPlanListActivity.this, CreateActivity.class);
        startActivityForResult(intent, CREATE);
    }

    private void callFloorPlanView(int id){
        Intent intent = new Intent(FloorPlanListActivity.this, FloorPlanActivity.class);
        intent.putExtra("ID", id);
        startActivityForResult(intent, OPEN_FLOORPLAN);
    }

    private class FloorPlanCLickListener implements View.OnClickListener {
        private int _id;

        public FloorPlanCLickListener(int id){
            _id = id;
        }

        @Override
        public void onClick(View v) {
            callFloorPlanView(_id);
        }
    }
}
```

В16. Код сутності точки.

```
package com.google.ar.core.examples.java.helloar;

import java.io.Serializable;

public class FloorPlanPoint implements Serializable {
    private float _x;
    private float _y;
    private float _z;
    private IPointType _pointType;

    public FloorPlanPoint(float x, float y, float z, IPointType pointType){
        _x = x;
        _y = y;
        _z = z;
        _pointType = pointType;
    }

    float getX(){
        return _x;
    }
    float getY(){
        return _y;
    }
    float getZ(){
        return _z;
```

```
    }

    public IPointType getPointType(){
        return _pointType;
    }

    void setX(float x){
        _x = x;
    }

    void setY(float y){
        _y = y;
    }

    void setZ(float z){
        _z = z;
    }
}
```

В17. Код сутності сховища ідентифікаторів.

```
package com.google.ar.core.examples.java.helloar;

import android.content.Context;
import android.widget.TextView;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.Arrays;

public class IdStorage implements Serializable {
    private int[] _floorPlanIds;
    private int[] _cableTypeIds;

    public IdStorage() {
        _floorPlanIds = new int[0];
        _cableTypeIds = new int[0];
    }

    public IdStorage(int[] floorPlanIds, int[] cableTypeIds) {
        _floorPlanIds = floorPlanIds;
        _cableTypeIds = cableTypeIds;
    }

    public int[] getFloorPlanIds() {
        return _floorPlanIds;
    }

    public int[] getCableTypeIds() {
        return _cableTypeIds;
    }

    public void addFloorPlanId(int id) {
        if(Arrays.stream(_floorPlanIds).anyMatch(x -> x == id))
            return;

        int[] newFloorPlanIds = new int[_floorPlanIds.length + 1];
        for(int i = 0; i < _floorPlanIds.length; i++){
            newFloorPlanIds[i]= _floorPlanIds[i];
        }
        newFloorPlanIds[newFloorPlanIds.length - 1] = id;
        _floorPlanIds = newFloorPlanIds;
    }

    public void removeFloorPlanId(int id) {
```

```
        _floorPlanIds = Arrays.stream(_floorPlanIds).filter(x -> x != id).toArray();
    }

    public void addCableTypeId(int id) {
        if(Arrays.stream(_cableTypeIds).anyMatch(x -> x == id))
            return;

        int[] newCableTypeIds = new int[_cableTypeIds.length + 1];
        for(int i = 0; i < _cableTypeIds.length; i++){
            newCableTypeIds[i]= _cableTypeIds[i];
        }
        newCableTypeIds[newCableTypeIds.length - 1] = id;
        _cableTypeIds = newCableTypeIds;
    }

    public void removeCableTypeId(int id) {
        _cableTypeIds = Arrays.stream(_cableTypeIds).filter(x -> x != id).toArray();
    }

    public boolean Save(Context context){
        File file = new File(context.getFilesDir(), "id_storage");
        if (file.exists()) {
            file.delete();
        }
        try {
            file.createNewFile();
            FileOutputStream fos = new FileOutputStream(file);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(this);
            fos.close();
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    public static IdStorage Load(Context context){
        File file = new File(context.getFilesDir(), "id_storage");
        if(file.exists()) {
            try {
                FileInputStream fis = new FileInputStream(file);
                ObjectInputStream ois = new ObjectInputStream(fis);
                Object result = ois.readObject();
                if(result.getClass() == IdStorage.class) {
                    return (IdStorage)result;
                }
            } catch (Exception e) {
            }
        }
        return null;
    }
}
```

В18. Код інтерфейсу типу лінії.

```
package com.google.ar.core.examples.java.helloar;

import java.io.Serializable;

public interface ILineType extends Serializable {
    public boolean isWall();
    public boolean isCable();
    public Integer getCableTypeId();
}
```

В19. Код інтерфейсу типу точки.

```
package com.google.ar.core.examples.java.helloar;
```

```
import java.io.Serializable;

public interface IPointType extends Serializable {
    public boolean isWallPoint();
    public boolean isCableConnectionPoint();
    public boolean isCablePoint();
    public Integer getCableTypeId();
}
```

В20. Код типу лінії стіни.

```
package com.google.ar.core.examples.java.helloar;

public class WallLineType implements ILineType{
    public boolean isWall(){
        return true;
    }
    public boolean isCable(){
        return false;
    }
    public Integer getCableTypeId(){
        return null;
    }
}
```

В21. Код типу точки стіни.

```
package com.google.ar.core.examples.java.helloar;

public class WallPointType implements IPointType{
    public boolean isWallPoint(){
        return true;
    }
    public boolean isCableConnectionPoint(){
        return false;
    }
    public boolean isCablePoint(){
        return false;
    }
    public Integer getCableTypeId(){
        return null;
    }
}
```

В22. Код manifest-у застосунку.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.google.ar.core.examples.java.helloar" >

    <uses-permission android:name="android.permission.CAMERA" /> <!-- Required to post
notifications for Android T+ devices -->
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <!--
 Google Play assumes that certain hardware related permissions indicate that the
underlying
       hardware features are required by default.
       (https://developer.android.com/topic/arc/manifest.html#implied-features).
    -->
    <uses-feature
        android:name="android.hardware.camera"
        android:required="true" />
    <!--
 Limits app visibility in the Google Play Store to ARCore supported devices
       (https://developers.google.com/ar/devices).
    -->
    <uses-feature
        android:name="android.hardware.camera.ar"
```

```
        android:required="true" />
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:allowBackup="false"
        android:icon="@drawable/building2"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="false"
        android:windowOptOutEdgeToEdgeEnforcement="true"
        tools:ignore="GoogleAppIndexingWarning" >
        <activity
            android:name=".FloorPlanListActivity"
            android:exported="true"
            android:screenOrientation="locked"
            android:theme="@style/Theme.AppCompat.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".CameraActivity"
            android:configChanges="orientation|screenSize"
            android:exported="true"
            android:screenOrientation="fullSensor"
            android:theme="@style/Theme.AppCompat.NoActionBar" >
        </activity>
        <activity
            android:name=".FloorPlanActivity"
            android:configChanges="orientation|screenSize"
            android:exported="true"
            android:theme="@style/Theme.AppCompat.NoActionBar">
        </activity>
        <activity
            android:name=".FloorPlanCableTypeListActivity"
            android:configChanges="orientation|screenSize"
            android:exported="true"
            android:theme="@style/Theme.AppCompat.NoActionBar">
        </activity>
        <activity
            android:name=".CreateActivity"
            android:configChanges="orientation|screenSize"
            android:exported="true"
            android:theme="@style/Theme.AppCompat.NoActionBar" >
        </activity>
        <activity
            android:name=".CableTypeListActivity"
            android:exported="true"
            android:screenOrientation="locked"
            android:theme="@style/Theme.AppCompat.NoActionBar" >
        </activity>
        <activity
            android:name=".CreateCableTypeActivity"
            android:configChanges="orientation|screenSize"
            android:exported="true"
            android:theme="@style/Theme.AppCompat.NoActionBar" >
        </activity>
        <activity
            android:name=".EditCableTypeActivity"
            android:configChanges="orientation|screenSize"
            android:exported="true"
            android:theme="@style/Theme.AppCompat.NoActionBar" >
        </activity>
        <!-- Indicates whether "Google Play Services for AR" (ARCore) is "required" or
"optional". -->
        <meta-data
            android:name="com.google.ar.core"
            android:value="required" />
```

```
    </application>

</manifest>
```

В23. Код контролера екрану списку типів мереж.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="0dp"
        android:layout_marginBottom="0dp">

        <LinearLayout android:id="@+id/floor_plan_list_with_spacer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

        <LinearLayout
            android:id="@+id/floor_plan_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" />

            <Space
                android:layout_width="match_parent"
                android:layout_height="90dp"/>
        </LinearLayout>

    </ScrollView>

    <ImageButton
        android:id="@+id/create_cabletype_activity_button"
        android:contentDescription="list_plus_cabletype"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="25dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/listplus" />

</RelativeLayout>
```

В24. Код інтерфейсу екрану сканування.

```
<!--
   Copyright 2016 Google LLC, 2025-2026 Mykolai Drozdov

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.google.ar.core.examples.java.helloar.CameraActivity">

<android.opengl.GLSurfaceView
    android:id="@+id/surfaceview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="top"/>

<ImageButton
    android:id="@+id/done_button"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_alignParentTop="false"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="25dp"
    android:layout_marginLeft="25dp"
    android:layout_marginTop="25dp"
    android:layout_marginEnd="25dp"
    android:layout_marginRight="25dp"
    android:layout_marginBottom="25dp"
    android:background="@android:color/transparent"
    android:scaleType="fitCenter"
    android:src="@drawable/check" />

<ImageButton
    android:id="@+id/anchor_button"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_alignParentTop="false"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="25dp"
    android:layout_marginLeft="25dp"
    android:layout_marginTop="25dp"
    android:layout_marginEnd="25dp"
    android:layout_marginRight="25dp"
    android:layout_marginBottom="90dp"
    android:background="@android:color/transparent"
    android:scaleType="fitCenter"
    android:src="@drawable/anchor" />

<ImageButton
    android:id="@+id/creation_button"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_alignParentTop="false"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="25dp"
    android:layout_marginLeft="25dp"
    android:layout_marginTop="25dp"
    android:layout_marginEnd="25dp"
    android:layout_marginRight="90dp"
    android:layout_marginBottom="25dp"
    android:background="@android:color/transparent"
    android:scaleType="fitCenter"
    android:src="@drawable/circleplus" />

<ImageButton
    android:id="@+id/cable_link_creation_button"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_alignParentTop="false"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="25dp"
    android:layout_marginLeft="25dp"
```

```
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginRight="25dp"
        android:layout_marginBottom="155dp"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:src="@drawable/merge" />

    <ImageButton
        android:id="@+id/cable_link_convert_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentTop="false"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginLeft="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginRight="90dp"
        android:layout_marginBottom="155dp"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:src="@drawable/refreshcw" />

    <ImageButton
        android:id="@+id/attach_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentTop="false"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginLeft="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginRight="90dp"
        android:layout_marginBottom="90dp"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:src="@drawable/link" />

    <ImageButton
        android:id="@+id/detach_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentTop="false"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginLeft="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginRight="155dp"
        android:layout_marginBottom="90dp"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:src="@drawable/unlink" />

    <ImageButton
        android:id="@+id/remove_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="false"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginLeft="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginRight="25dp"
```

```
        android:layout_marginBottom="25dp"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:src="@drawable/trash2" />

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:scaleX="3"
        android:scaleY="3"
        app:srcCompat="@drawable/crosshair" />

    <TextView
        android:id="@+id/distanceDisplay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView3"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:text="TextView" />

    <Spinner
        android:id="@+id/line_type_selector"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/done_button"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="155dp" />

</RelativeLayout>
```

В25. Код інтерфейсу екрану створення плану.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/i"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageButton
        android:id="@+id/create_button"
        android:contentDescription="check"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/check" />

    <EditText
        android:id="@+id/editTextCreateName"
        android:contentDescription="name_field"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="0dp"
```

```
        android:ems="10"
        android:fontFamily="sans-serif"
        android:inputType="text"
        android:singleLine="true"
        android:text="Назва"
        android:textSize="16sp" />

</RelativeLayout>
```

В26. Код інтерфейсу екрану створення типу мережі.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/i"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageButton
        android:id="@+id/create_button"
        android:contentDescription="check"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/check" />

    <EditText
        android:id="@+id/editTextCreateName"
        android:contentDescription="name_field"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="0dp"
        android:ems="10"
        android:fontFamily="sans-serif"
        android:inputType="text"
        android:singleLine="true"
        android:text="Назва"
        android:textSize="16sp" />

</RelativeLayout>
```

В27. Код інтерфейсу екрану перейменування типу мережі.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/i"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/editTextCreateName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
```

```
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="0dp"
        android:ems="10"
        android:fontFamily="sans-serif"
        android:inputType="text"
        android:singleLine="true"
        android:text="Назва"
        android:textSize="16sp" />

</RelativeLayout>
```

В28. Код інтерфейсу екрану плану.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/floorplan_relative_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageButton
        android:id="@+id/ar_activity_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="25dp"
        android:layout_marginBottom="25dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/camera" />

    <ImageButton
        android:id="@+id/save_button"
        android:contentDescription="save"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="90dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/externallink" />

    <ImageButton
        android:id="@+id/cableTypes_button"
        android:contentDescription="save"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="155dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/cable" />
```

```
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="0dp"
        android:layout_marginTop="55dp"
        android:layout_marginEnd="75dp"
        android:layout_marginBottom="5dp"
        android:scaleType="center" />

    <EditText
        android:id="@+id/editTextText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="0dp"
        android:ems="10"
        android:fontFamily="sans-serif"
        android:inputType="text"
        android:singleLine="true"
        android:text="Name"
        android:textSize="16sp" />

    <Space
        android:layout_width="wrap_content"
        android:layout_height="40dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        />

</RelativeLayout>
```

В29. Код інтерфейсу екрану інженерних мереж плану.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="0dp"
        android:layout_marginBottom="0dp">

        <LinearLayout android:id="@+id/floor_plan_list_with_spacer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
        <LinearLayout
            android:id="@+id/floor_plan_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" >

        </LinearLayout>
            <Space
                android:layout_width="match_parent"
                android:layout_height="90dp"/>
        </LinearLayout>
    </ScrollView>

    <ImageButton
```

```
        android:id="@+id/create_cabletype_activity_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="25dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:contentDescription="list_plus_cabletype"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/listplus" />

</RelativeLayout>
```

В30. Код інтерфейсу головного екрану.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="0dp"
        android:layout_marginBottom="0dp">

        <LinearLayout android:id="@+id/floor_plan_list_with_spacer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

        <LinearLayout
            android:id="@+id/floor_plan_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" >
        </LinearLayout>

            <Space
                android:layout_width="match_parent"
                android:layout_height="155dp"/>
        </LinearLayout>
    </ScrollView>

    <ImageButton
        android:id="@+id/cabletypelist_activity_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="25dp"
        android:layout_marginBottom="90dp"
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:contentDescription="list_plus"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/cable" />

    <ImageButton
        android:id="@+id/create_activity_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="25dp"
```

```
        android:backgroundTint="#00FFFFFF"
        android:baselineAlignBottom="false"
        android:contentDescription="list_plus"
        android:scaleType="fitCenter"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@drawable/gridplus" />

</RelativeLayout>
```

В31. Код елемента інтерфейсу типу мережі в меню екрану сканування.

```
<?xml version="1.0" encoding="utf-8"?>

<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:gravity="left"
    android:textColor="#FFFFFF"
    android:padding="5dip"
    />
```

ДОДАТОК Г
(обов’язковий)

КЕРІВНИЦТВО КОРИСТУВАЧА

Дане керівництво користувача описує порядок роботи з мобільним застосунком, призначеним для побудови планів поверхів будівель та прокладання інженерних мереж із використанням технології доповненої реальності. Основним сенсором у цьому проєкті виступає камера мобільного пристрою, яка забезпечує просторове відстеження, розпізнавання поверхонь та фіксацію координат об'єктів у реальному часі. Програмний засіб дозволяє формувати двовимірні креслення, вести облік різних типів комунікаційних ліній та експортувати отримані результати для подальшого використання.

Головний екран та управління проєктами

Після ініціалізації застосунку користувач потрапляє на головний екран, де відображається перелік раніше збережених планів поверхів. Для ініціалізації нового проєкту необхідно скористатися кнопкою зі знаком додавання, розміщеною у нижній правій частині робочої області. Система запропонує ввести ідентифікаційну назву нового плану у відповідне текстове поле, після чого відбудеться автоматичний перехід до робочого простору проєкту. Видалення неактуальних планів здійснюється за допомогою кнопки із зображенням кошика, що розташована поруч із назвою відповідного запису. З головного екрана також передбачено перехід до загального класифікатора типів комунікацій за допомогою кнопки із зображенням кабелю.

Класифікатор типів комунікацій

Підсистема керування типами комунікацій призначена для ведення довідника кабельних ліній, які будуть задіяні під час проєктування. У цьому розділі користувач має змогу додавати нові категорії, вказуючи їхні найменування. Редагування існуючих записів виконується шляхом натискання на кнопку із зображенням олівця, що відкриває форму для зміни назви. Для вилучення типу комунікації з бази даних використовується відповідна кнопка

видалення. Усі створені категорії автоматично стають доступними для вибору під час роботи у режимі просторового сканування.

Двовимірна візуалізація та розрахунки

Робочий простір конкретного плану являє собою середовище двовимірної візуалізації зібраних просторових даних. Користувач може візуально аналізувати побудовану схему, виконувати масштабування та панорамування креслення за допомогою стандартних жестів керування сенсорним екраном. Для перегляду детальної інформації щодо прокладених комунікацій та обчислення їхньої сумарної довжини необхідно відкрити меню кабелів (кнопка із зображенням кабелю). У цьому вікні здійснюється вибір необхідних типів мереж шляхом відмічання відповідних позицій, після чого програмний засіб автоматично розраховує та відображає загальну довжину обраних сегментів. Збереження готового креслення у форматі графічного зображення ініціюється кнопкою експорту, яка дозволяє зберегти файл у файлову систему пристрою.

Режим просторового сканування

Формування геометрії приміщення та прокладання мереж відбувається у режимі просторового сканування, який активується з робочого простору плану натисканням кнопки із зображенням камери. Функціонування цього модуля базується на аналізі відеопотоку, тому користувачеві необхідно плавно переміщувати пристрій для забезпечення коректного розпізнавання просторових площин. Після ідентифікації поверхні на екрані з'являється візуальний маркер-курсор.

Панель інструментів просторового сканування містить елементи для повного циклу моделювання. Вибір поточного типу об'єкта (стіна або певний тип кабелю) здійснюється за допомогою випадного меню у нижній частині екрана. Встановлення базових точок креслення виконується інструментом створення (кнопка з плюсом). Для формування складних комунікаційних вузлів застосовується інструмент об'єднання ліній.

У разі необхідності зміни властивостей існуючого вузла використовується інструмент перетворення. Зв'язування нових просторових точок із вже

побудованою мережею забезпечується інструментом приєднання, а скасування таких зв'язків – інструментом від'єднання. Інструмент видалення дозволяє вилучати помилково встановлені точки простору. Для фіксації якоря віртуального об'єкта використовується відповідна кнопка фіксації. Після завершення процесу збору координатних даних необхідно натиснути кнопку підтвердження (зображення галочки). Система виконає алгоритми оптимізації та вирівнювання геометрії, запише дані у підсистему локального збереження та поверне користувача до режиму двовимірної візуалізації.

ДОДАТОК Д
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

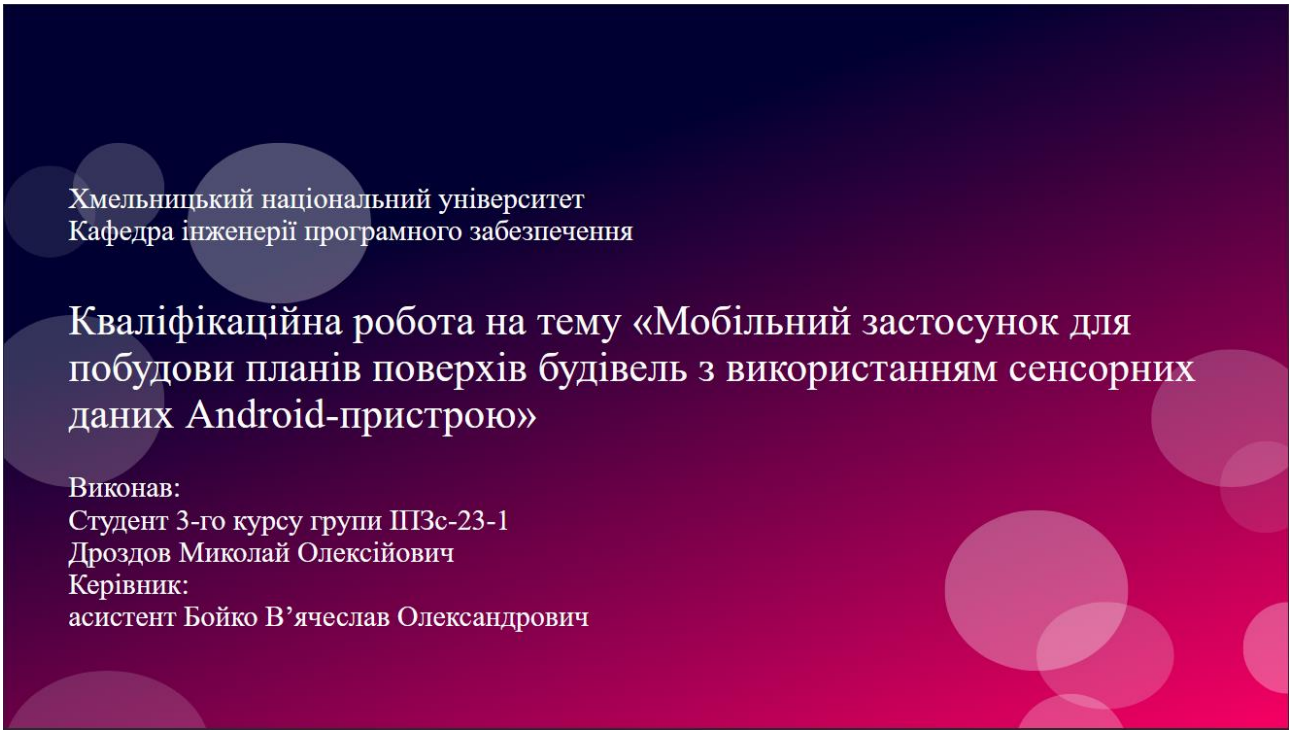


Рисунок Д.1 – слайд 1

Рисунок Д.2 – слайд 2

Мета та завдання роботи

Метою роботи є створення мобільного застосунку для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою, що відповідатиме актуальним технічним вимогам індустрії.

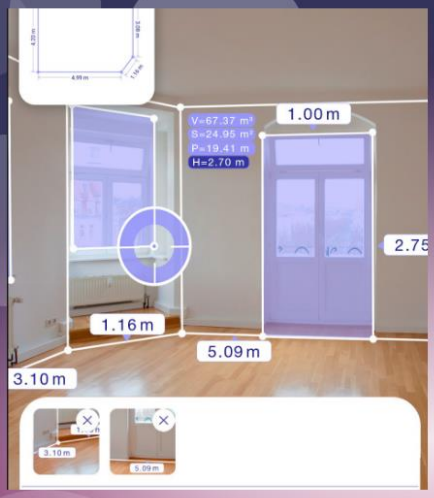
Завдання:

- аналіз архітектурних підходів, шаблонів проектування
- проєктування системи
- створення макету інтерфейсу
- аналіз та вибір технологій
- програмування системи
- створення візуальних ресурсів для режиму сканування
- створення інтерфейсу користувача
- тестування
- формування висновків

3

Рисунок Д.3 – слайд 3

Змістовний аналіз предметної області, її структурних та функціональних особливостей



- Елементи: стіни, підлога, стеля, дверні та віконні отвори.
- Основні функції — автоматичне виявлення та/або позначення користувачем об’єктів, формування з цієї інформації плану.
- Технології: доповнена реальність, LiDAR, ToF камери, машинне навчання.

4

Рисунок Д.4 – слайд 4

Рисунок Д.5 – слайд 5

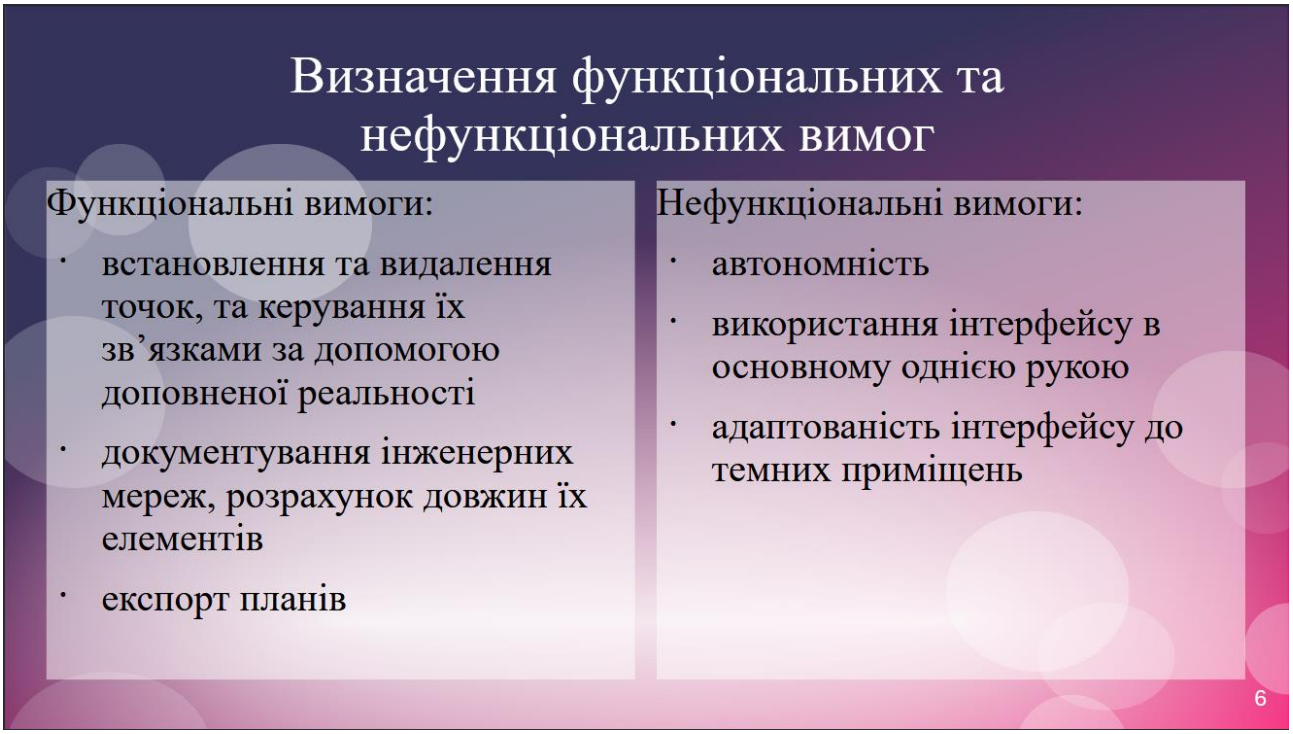


Рисунок Д.6 – слайд 6

Рисунок Д.7 – слайд 7

Рисунок Д.8 – слайд 8

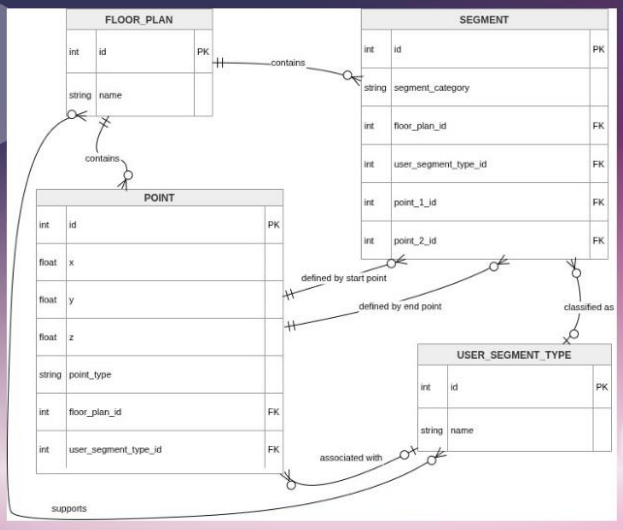
Опис інтерфейсів

Модуль	Інтерфейс
CameraActivity	
IdStorage	getFloorPlanIds(), getCableTypeIds(), addFloorPlanId(), removeFloorPlanId(), addCableTypeId(), removeCableTypeId(), Save(), Load()
CableTypeListActivity, CreateCableTypeActivity, EditCableTypeActivity	
FloorPlanListActivity, FloorPlanActivity, CreateActivity	
FloorPlanCableTypeListActivity	
CableType	getId(), getName(), setName(), Save(), Load(), Delete()
FloorPlan	captureSet(), getName(), setName(), setCapture(), addCableTypeId(), removeCableTypeId(), getPoints(), getLines(), Save(), Load(), Delete()
FloorPlanPoint, IPointType	getX(), getY(), getZ(), setX(), setY(), setZ(), getPointType(), isWallPoint(), isCableConnectionPoint(), isCablePoint()
FloorPlanLine, ILineType	getPoint1Index(), getPoint2Index(), getLineType(), isWall(), isCable(), getCableTypeId()

9

Рисунок Д.9 – слайд 9

Проєктування даних



10

Рисунок Д.10 – слайд 10

Рисунок Д.11 – слайд 11

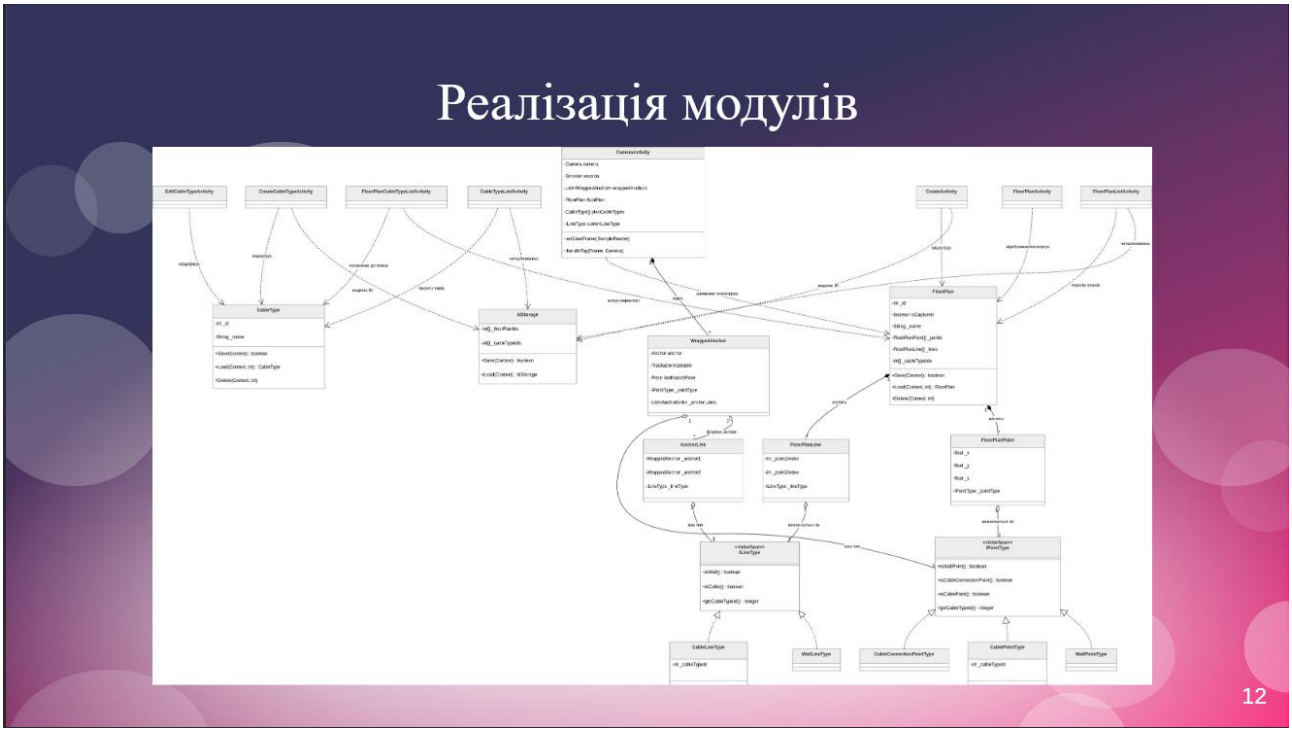


Рисунок Д.12 – слайд 12

Вимоги до технічного та програмного забезпечення

	Мінімальні	Рекомендовані
ОС	Android 10	Android 15
SOC	Qualcomm Snapdragon 855, Exynos 9820, Mediatek MT6762 Helio P22	Qualcomm SM8550-AC Snapdragon 8 Gen 2 Exynos 1680 Mediatek Dimensity 9500s
Оперативна пам'ять	4 ГіБ	8 ГіБ
Місце на диску	256 МіБ	1 ГіБ
Підтримка ARCore	Обов'язкова	
Сервіси Google Play для AR	Наявність обов'язкова	

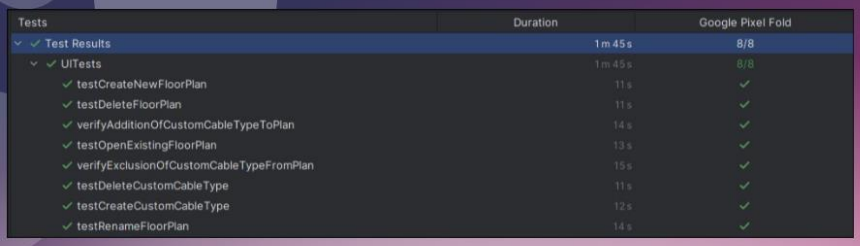
13

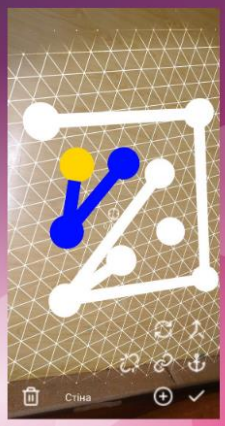
Рисунок Д.13 – слайд 13

Тестування ПЗ

Автоматизоване

Ручне





Тестування показало відповідність системи всім функціональним вимогам, а дефектів не було виявлено

14

Рисунок Д.14 – слайд 14

Рисунок Д.15 – слайд 15

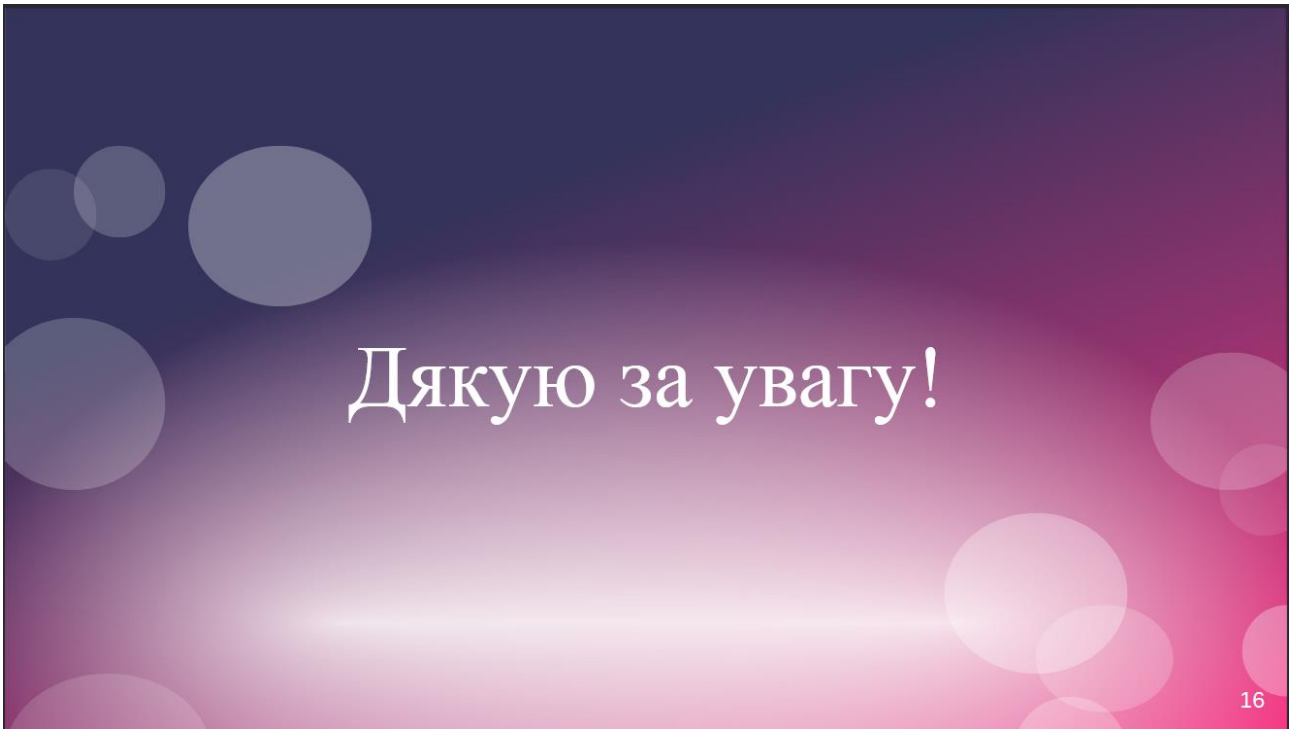
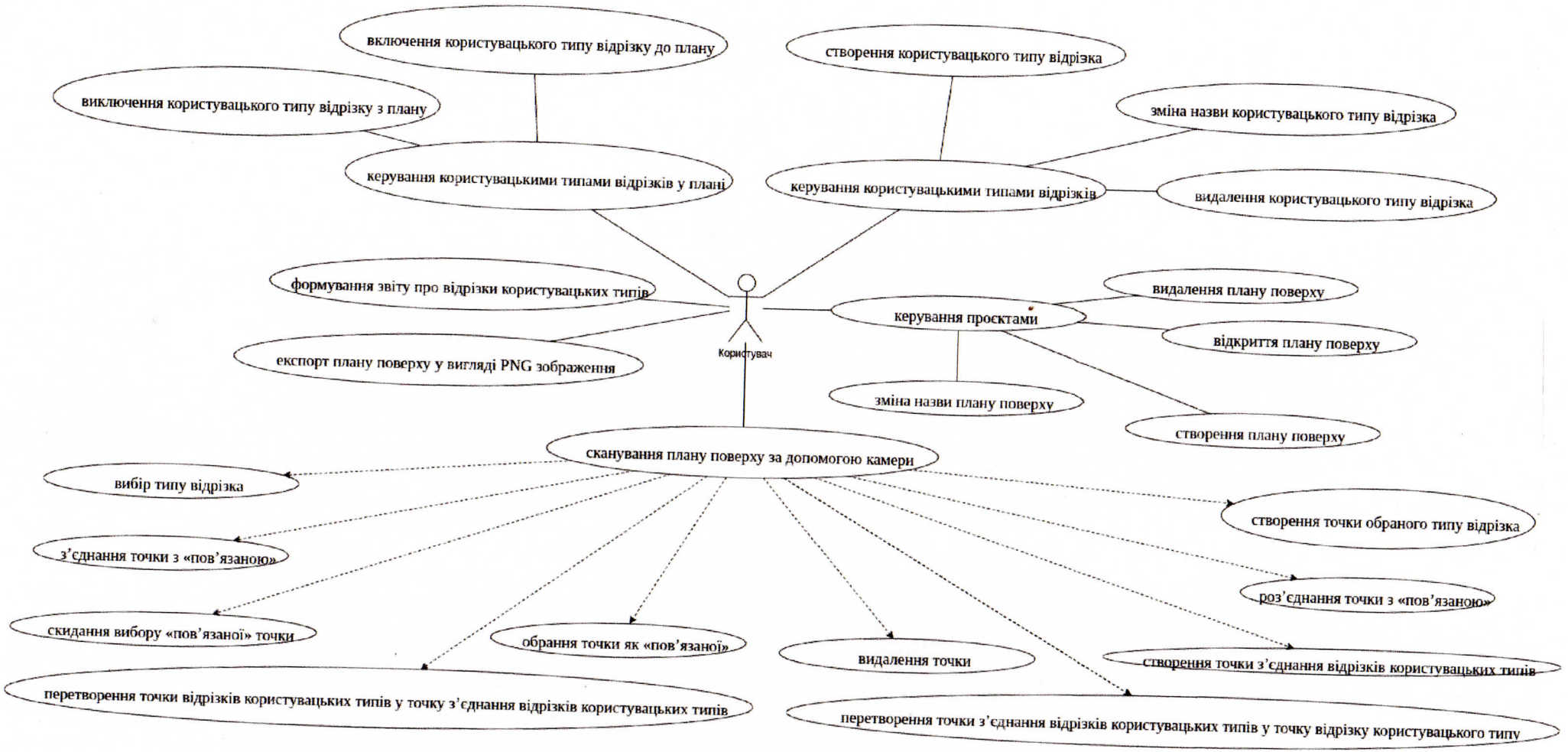


Рисунок Д.16 – слайд 16

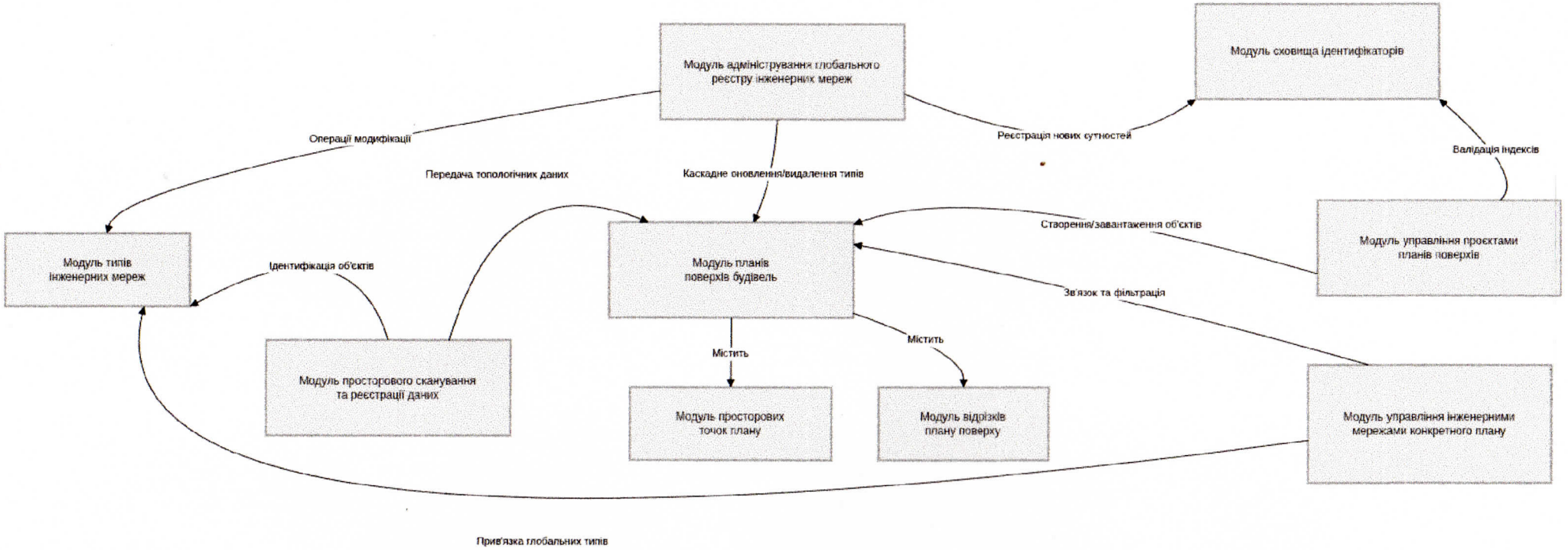
ГРАФІЧНІ МАТЕРІАЛИ



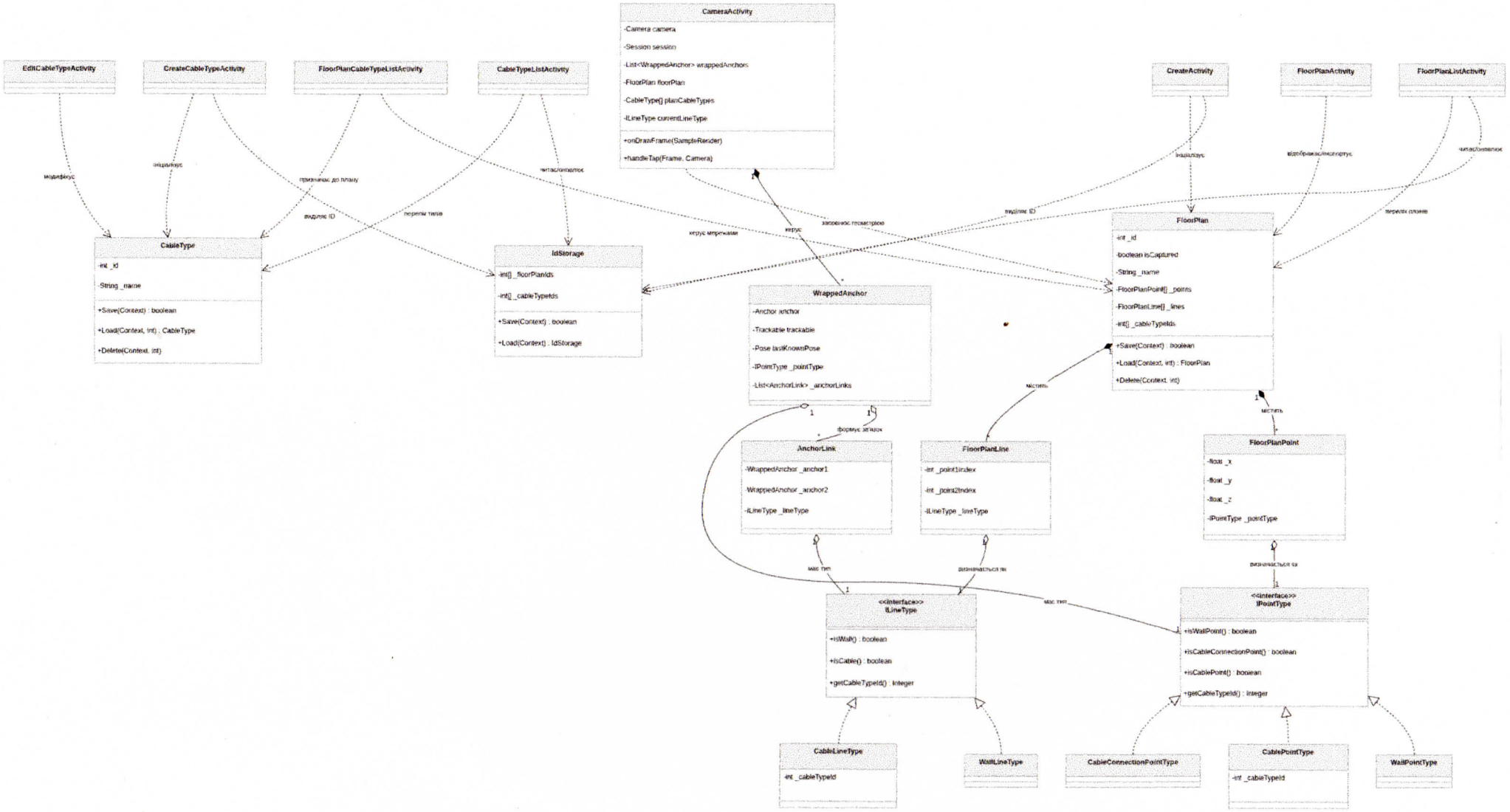
					КВРІПЗс.230134.01.06.Е8
Змн.	Лист	№ докум.	Підпис	Дата	Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою UML-діаграма варіантів використання
Розробив		Дроздов М. О.		27.05.26	
Керівник		Байко В. О.		27.05.26	
Консульт.					
Н. Контр.		Бедратюк Г. І.		27.05.26	
Затверд.		Бедратюк Л. П.		27.05.26	

Літ.	Маса	Масштаб
Арк. 1	Аркушів 3	

ХНУ. ІПЗс-23-1



					КВРІПЗс.230134.01.06.Е8	
Змн.	Лист	№ докум.	Підпис	Дата		
Розробив		Дроздов М. О.			Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою Діаграма зв'язків модулів	Літ. / Маса / Масштаб
Керівник		Бойко В. О.				Арк. 2 / Аркушів 3
Консульт.						
Н. Контр.		Бедратюк Г. І.				ХНУ. ІПЗс-23-1
Затверд.		Бедратюк Л. П.				



					КВРІПЗс.230134.01.06.Е8		
Змн.	Лист	№ докум.	Підпис	Дата	Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою UML-діаграма класів	Літ.	Маса	Масштаб
Розробив		Дроздов М. О.		27.05.25				
Керівник		Бойко В. О.		27.05.25		Арк. 3	Аркушів 3	
Консульт.								
Н. Контр.		Бедратюк Г. І.		27.05.25		ХНУ. ІПЗс-23-1		
Затверд.		Бедратюк Л. П.		27.05.25				

СУПРОВІДНІ ДОКУМЕНТИ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

першого освітнього рівня «Бакалавр»

Студента: Дроздова Миколая
Група: ІПЗс-23-1

Тема: *«Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою»*

Спеціальність: 121 – Інженерія програмного забезпечення

Короткий зміст пояснювальної записки

У вступі кваліфікаційної роботи окреслено актуальність створення мобільного застосунку для автоматизованого формування планів будівель, що зумовлено зростаючими потребами у точному моделюванні середовища. Метою роботи є розроблення такого застосунку, який використовує сенсорні дані Android-пристрою та відповідає сучасним вимогам продуктивності та інформаційної безпеки. Завданнями проєктування є проведення аналізу предметної області, формування вимог, вибір архітектурних підходів, декомпозиція системи, проєктування бази даних та інтерфейсу, а також реалізація та тестування застосунку.

Відповідність отриманих результатів роботи поставленим завданням

Завдання, сформульовані у вступі, включають проведення змістовного

аналізу предметної області, формування вимог, вибір архітектурних підходів, декомпозицію системи, проєктування бази даних, реалізацію та тестування. Висновки роботи підтверджують, що результати в основному відповідають поставленим завданням, оскільки всі ключові етапи проєктування та реалізації були виконані. Оцінка відповідності: **в основному відповідають**.

Оцінка розділів

Розділ 1

У розділі 1 представлено змістовний аналіз предметної області автоматизації збору та обробки просторових даних, що включає опис ключових об'єктів дослідження та важливість точних планів для обслуговування будівель. Однак, опис предметної області може бути визнаний недостатньо структурованим, оскільки не всі ключові аспекти розкриті в повному обсязі. Аналіз наявного програмного забезпечення охоплює класифікацію засобів, але не всі можливі рішення на ринку були розглянуті. Визначення функціональних та нефункціональних вимог до програмного забезпечення виконано детально, проте деякі вимоги потребують уточнення. Висновки підсумовують результати аналізу, але можуть бути недостатньо узагальненими.

Розділ 2

У розділі 2 розглянуто вибір архітектури та шаблонів проєктування, зокрема, описано переваги та недоліки різних архітектурних підходів. Однак, не було проведено порівняння між усіма згаданими архітектурами. Декомпозиція застосунку описана на автономні функціональні блоки, але деталі навігації між модулями не були надані. Залежності між екранами та сервісами описані, але не було надано конкретних прикладів викликів. Проєкт інтерфейсу користувача охоплює основні екрани, але не було надано графічних схем переходів. Вибір технологій реалізації аргументовано, але не було проведено порівняння нативних та кросплатформенних рішень.

Розділ 3

У розділі 3 детально описано реалізацію основних компонентів застосунку, зокрема, використання ARCore та Java. Однак, не згадано про репозиторії, що може свідчити про їх відсутність. Реалізація узгоджена з архітектурними рішеннями, але не було чіткої інформації про те, як конкретні класи відповідають архітектурним шаблонам. Описано локальне зберігання даних, але не було надано конкретних технологій, таких як Room. Тестування проведено, але відсутня таблиця з тест-кейсами та їх статусами.

Позитивні сторони

Кваліфікаційна робота демонструє оригінальність у виборі теми, що є актуальною та перспективною. Якісно виконано опис ключових об'єктів дослідження, а також підкреслено важливість точних планів для обслуговування будівель. Добре структуровано інформацію про різні групи програмного забезпечення, що дозволяє швидко оцінити їх переваги та недоліки. Реалізація основних компонентів застосунку описана детально, що забезпечує зрозуміле уявлення про структуру.

Недоліки

Деякі розділи потребують уточнення та доповнення, зокрема, недостатня структурованість опису предметної області, відсутність конкретних прикладів у аналізі програмного забезпечення, а також недостатня деталізація вимог до продуктивності. Відсутність чіткої інформації про репозиторії та конкретні технології локального зберігання також є недоліком. У розділі тестування не надано таблиці з тест-кейсами, що ускладнює оцінку результатів тестування.

Відгук в цілому

Кваліфікаційна робота є актуальною та має практичну значущість, оскільки відповідає сучасним вимогам до автоматизації процесів у будівництві. Зміст роботи відповідає темі та завданню, проте деякі розділи потребують доопрацювання для покращення якості. Новизна ідей та рішень є очевидною, але обґрунтованість деяких аспектів потребує уточнення. Загалом, робота демонструє високий рівень підготовки здобувача, але потребує доопрацювання в окремих моментах.

Оцінка кваліфікаційної роботи

Кваліфікаційна робота заслуговує оцінки **добре**. Вона виконана в повному обсязі з дотриманням основних вимог, але має деякі недоліки, які потребують виправлення. Здобувач володіє матеріалом, грамотно викладає суть роботи, хоча може припускатися дрібних неточностей.

Рекомендації

Рекомендується доопрацювати роботу, зокрема, уточнити вимоги до продуктивності, додати більше прикладів у аналізі програмного забезпечення, а також надати таблиці з тест-кейсами. Це підвищить якість роботи та забезпечить більш чітке розуміння реалізації проекту.

OpenAI API-асистент
Session ID: 9070ddf7-6ad3-4391-8d58-e9bd3c31c30e
Підписано автоматично, модель gpt-4o-mini
Дата: 15.05.2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Дроздов Миколай Олексійович

Тема Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 77

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі досліджено сферу автоматизації побудови планів приміщень та розроблено застосунок для документування інженерних мереж. На основі аналізу конкурентів доведено актуальність такого рішення. Програмне забезпечення реалізовано з використанням Google ARCore, Java та Android Studio. Успішне ручне й автоматизоване тестування підтвердило коректну роботу системи та її готовність до практичного використання.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання кваліфікаційної роботи. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення для побудови планів поверхів та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур і шаблонів проєктування, розглянуто їх переваги і недоліки та визначено, що система буде відповідати шаблону MVC у поєднанні з модульною монолітною та закритою тришаровою архітектурою. У третьому розділі розглянуто особливості реалізації з використанням ARCore, Java та інтерфейсу на основі XML файлів, виконано практичну розробку програмних модулів просторового відстеження й візуалізації та описано їх особливості, в результаті чого створено програмний продукт. Також у цьому розділі виконано автоматизоване та ручне інтеграційне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу програми та відсутність дефектів.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки на сьогодні у сучасних рішеннях цієї галузі відсутні засоби та функціональні можливості, необхідні для документування та планування розміщення інженерних мереж. Також було застосовано новітні технології для побудови програмного продукту, зокрема платформу просторового обчислення Google ARCore, та обґрунтовані актуальні архітектурні рішення, а саме шаблон MVC у поєднанні з модульною монолітною та закритою тришаровою архітектурою.

5. Негативні сторони роботи У роботі експорт плану реалізований лише у форматі растрового зображення PNG – було б доцільно додати можливість експорту у векторні (наприклад, PDF або DXF) чи тривимірні формати для зручнішої подальшої роботи у професійних системах проєктування. Окрім цього, оскільки застосунок є виключно однокористувацьким, було б корисно додати функцію опціональної хмарної синхронізації для спільної роботи інженерів над одним проєктом.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому заслуговує високої позитивної оцінки. Кваліфікаційна робота заслуговує на позитивну оцінку. Матеріал пояснювальної записки структурований, послідовний та чіткий, що дозволяє повною мірою зрозуміти викладений матеріал у рамках тематики проєктування застосунків доповненої реальності. Автор вдало обґрунтував вибір архітектурних рішень та методів збереження даних. Графічний матеріал, до якого увійшли діаграми варіантів використання, зв'язків модулів та класів, виконаний на належному рівні та дає можливість наочно побачити деталі проєктування розробленої програмної системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «відмінно»

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Капустян Марія Вікторівна, доцент каф. КПСС ХНУ

"25" травня 2026 р. _____
(підпис)

Wed May 20 00:40:10 EEST 2026, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (http://ap.km.ua) v-16.718

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: UA, US, RU. **Помилок в документах: 13%**

	Документ		Сумарний збіг по Базі Даних	
ID: 271745 Назва: БКР_Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою Додано в БД: 2026-05-20 Автора: Миколай ДРОЗДОВ Керівники: В’ячеслав БОЙКО Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	108996	850	3981 (4%)	52 (6%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Завідувачу кафедри інженерії програмного забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Дроздова Миколая Олексійовича
факультет ІТ, ІІкурс, група ІПЗс-23-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

07.05.2026
дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Дроздов Миколай Олексійович,

студент III курсу спеціальності 121 – Інженерія програмного забезпечення, групи ІПЗс-23-1

здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)

підтверджую, що ознайомився з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

07 Вересня 2026 р. Дб

Підпис

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-23-1
Дроздова М. О.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою

(керівник роботи – Бойко В'ячеслав Олександрович)
Прізвище, ім'я, по батькові

02.07.2026
Дата

Підпис студента

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.

Назва кваліфікаційної роботи: «Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою»

Автор: Дроздов Микола Олексійович

Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»

Спеціальність: 121 – Інженерія програмного забезпечення

Науковий керівник: Бойко В'ячеслав Олександрович, асистент

Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Anti-Plagiarism виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках, у структурі змісту, назвах розділів/підрозділів, рамках форм, у назвах та URL-адресах публікацій переліку джерел посилання;

2) запозичення, виявлені у тексті роботи, є фрагментарними.

Загальна сумарна подібність у базі даних складає 4% за символами та 6% за лексемами. Крім того, за результатами додаткового аналізу коефіцієнт подібності 1 становить 3,91%, коефіцієнт подібності 2 – 1,58%. Не виявлено мікропробілів, зайвих білих знаків або маніпуляцій з інтервалами. З урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 25.05.2026

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ В'ячеслав БОЙКО

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Миколай ДРОЗДОВ

Співавтор:

Назва: Мобільний застосунок для побудови планів поверхів будівель з використанням сенсорних даних Android-пристрою

Науковий керівник: В'ячеслав БОЙКО

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:3.91%

Коефіцієнт подібності 2:1.58%

Мікропробіли: 0

Заміна букв: 78

Інтервали: 0

Білі знаки: 5

Дата створення звіту: 2026-05-20 00:37:24.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата
25.05.26

експерт