Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Коберник Дар'ї Сергіївни
Прізвище, ім'я, по батькові студента(ки)

на здобуття ступеня вищої освіти Бакалавра

Мобільний застосунок для читання книг з Google Books
Назва теми

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ. 200165.01.10.ПЗ

Виконав студент IV курсу, група ІПЗ-20-1 _____ Дар'я КОБЕРНИК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник старший викладач _____ Ганна БЕДРАТЮК
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. техн. наук, доцент _____ Юрій ФОРКУН
Посада Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

7 червня 2024 р.

Хмельницький 2024

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
Л. П. Бедратюк
2.01 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коберник Дар'ї Сергіївні

Прізвище, ім'я, по батькові студента

1. Тема роботи Мобільний застосунок для читання книг з Google Books

Керівник проєкту (роботи) Бедратюк Ганна Іванівна, старший викладач

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 02.01.2024 р. № 6

2. Строк подання студентом роботи на кафедру 01.06.2024 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 програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди, 16 шт.)
Три креслення:
1. Розкрита діаграма класів
2. Загальна діаграма потоків даних в застосунку
3. Блок-схема відправки запиту до Google Book API

6. Консультанти розділів дипломного проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Форкун Ю. В., доцент	31.05.2024	06.06.2024
Антиплагіат	Форкун Ю. В., доцент		

7. Дата видачі завдання « 02 » січня 2024р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проєктування (ДП), визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 – 31.12.2023	
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 – 20.02.2024	
3 Проєктування програмного забезпечення	21.02 – 20.03.2024	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2024	
5 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог стандартів	01.05 – 25.05.2024	
6 Попередній захист КвР	Травень 2024	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2024	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2024	

Студент _____ Дар'я КОБЕРНИК
Підпис Ініціали, ПРІЗВИЩЕ

Керівник роботи _____ Ганна БЕДРАТЮК
Підпис Ініціали, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: Мобільний застосунок для читання книг з Google Books.

Автор проєкту: Коберник Дар'я Сергіївна.

Керівник проєкту: Бедратюк Ганна Іванівна.

Пояснювальна записка: 79 с., 14 рис., 11 табл., 6 дод., 40 джерел.

Графічна частина: 16 слайдів, три плакати ф. А3

МОБІЛЬНИЙ ЗАСТОСУНОК, ЧИТАННЯ КНИГ, GOOGLE BOOKS API, JAVA, ANDROID STUDIO, БІБЛІОТЕКА, ІНТЕРФЕЙС КОРИСТУВАЧА.

Метою проєкту є розробка високоефективного мобільного застосунку для читання книг, який інтегрується з Google Books API і надає користувачам зручний та зрозумілий інтерфейс для доступу до великої бібліотеки літератури.

У кваліфікаційній роботі було проведено дослідження предметної області та постановку задач, спроєктовано програмне забезпечення, розроблено і протестовано застосунок та підведено підсумки. Для розробки були використані мова програмування Java та середовище програмування Android Studio.

В результаті було розроблено застосунок для читання книг, який має дружній інтерфейс і підтримує різноманітні функціональні можливості, такі як: пошук книг, читання онлайн. Застосунок забезпечує безпечний доступ до особистої бібліотеки користувачів та інтегрується з Google Books API для отримання актуальної інформації про книги.

02.01.24
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200165.01.10.ПЗ	Пояснювальна записка	79		
2	А4	КвРІПЗ.200165.01.10.ПЗ	Завдання на кваліфікаційну роботу	1		
3	А4	КвРІПЗ.200165.01.10.ПЗ	Анотація	1		
			Графічні документи			
			Презентаційні матеріали	19		
4	А3	КвРІПЗ.200165.01.10.ПЗ	Розкрита діаграма класів			
5	А3	КвРІПЗ.200165.01.10.ПЗ	Загальна діаграма потоків даних в застосунку			
6	А3	КвРІПЗ.200165.01.10.ПЗ	Блок-схема відправки запиту до Google Book API			

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200165.01.10.ПЗ		
Виконав		Коберник Д.С.			Мобільний застосунок для читання книг з Google Books	Літ.	Арк. 5 / Аркушів 79
Керівник		Бедратюк Г.І.					
Н. Контр.		Форкун Ю.В.			Відомість документів	ХНУ, ІПЗ-20-1	
Зав. Каф.		Бедратюк Л.П.					

ЗМІСТ

ВСТУП

У світлі постійного технологічного розвитку та зростаючого інтересу до мобільних рішень, мобільні застосунки для читання книг стають ефективним засобом розвитку читацької культури та підтримки інноваційних форм споживання контенту. Вони забезпечують можливість не тільки зануритися в світ літератури, але і активно взаємодіяти з текстами, створюючи нові форми читацького досвіду і розвиваючи цифрову грамотність серед користувачів.

Застосунки для читання книг на сьогоднішній день мають безліч переваг, які сприяють їх популярності та зручності використання. Перш за все, ці застосунки роблять доступ до літератури миттєвим і зручним, оскільки користувачі можуть мати доступ до тисяч книг безпосередньо на своїх мобільних пристроях. Це особливо важливо в сучасному швидкому темпі життя, коли час на відвідування бібліотек або книжкових магазинів може бути обмежений.

Розробка мобільного застосунка для читання книг з Google Books окреслює важливу проблему доступності літературного контенту для користувачів у сучасному цифровому світі. Однією з основних проблем традиційних книг є обмежена доступність і складність утримання великої бібліотеки книг для багатьох людей. Мобільні застосунки, зокрема ті, які працюють з Google Books, пропонують рішення, дозволяючи зберігати велику кількість книг у цифровому форматі на мобільних пристроях, що робить доступ до літературних творів більш зручним і ефективним, що є великою плюсом для користувачів.

Ще однією важливою перевагою цих застосунків є їх мобільність. Користувачі можуть мати доступ до своєї віртуальної бібліотеки з будь-якого місця і в будь-який час через свої смартфони або планшети. Це дозволяє задовольняти різні вподобання читачів і розширює аудиторію, яка може скористатися послугами цих застосунків. Все це робить такі застосунки для читання книг надзвичайно корисними та популярними серед сучасних читачів.

Актуальність мобільних застосунків для читання книг особливо відчутна в контексті зростання популярності цифрових технологій і впливу інтернету на

наш спосіб споживання інформації. Вони відповідають потребам сучасного читача, який шукає швидкий і зручний спосіб отримання доступу до літературного світу. Такі застосунки забезпечують не лише можливість читати книги на мобільних пристроях, але й створюють сприятливі умови для інтерактивного досвіду читання, де користувачі можуть обмінюватися враженнями та думками про прочитані твори. На сьогоднішній день багато користувачів віддають перевагу електронному формату читання книг через зручність і доступність. За допомогою мобільних застосунків, таких як ті, що працюють з Google Books, читання стає більш гнучким і адаптивним до сучасного способу життя, де мобільність і швидкість доступу є ключовими та важливими факторами.

Метою розробки мобільного застосунка для читання книг з Google Books є створення зручного і функціонального інструменту, який надає користувачам доступ до широкого асортименту електронних книг з бібліотеки Google Books через їх мобільні пристрої.

Об’єктом дослідження є процес читання книги користувачем.

Предмет дослідження – інформація про названий вище застосунок, його варіанти реалізації та використання.

Досягнення поставленої мети здійснюється шляхом її деталізації за допомогою систематизованого плану цілеспрямованих дій – задач (завдань) проєктування. Для досягнення поставленої мети будуть виконані наступні визначені завдання:

– вивчення сучасних тенденцій та розгляд існуючих аналогів;

– вибір архітектурної концепції з урахуванням вимог до безпеки, швидкодії та масштабованості;

– вибір необхідних технологій для реалізації архітектури;

– розробка дизайну та кодової частини застосунку;

– тестування отриманого програмного продукуту;

– оцінка ефективності розробленого рішення та визначення можливостей для подальшого розвитку.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз ПО, її структурних та функціональних особливостей

Предметна область, пов'язана з розробкою мобільного застосунку для читання книг, охоплює різні аспекти взаємодії користувачів з електронними книгами. Сучасні мобільні застосунки для читання книг повинні забезпечувати зручний доступ до великої бібліотеки літератури, надавати функції пошуку, збереження закладок, налаштування інтерфейсу для комфорту читання та синхронізації між різними пристроями. Ці застосунки інтегруються з онлайн бібліотеками, такими як Google Books, та забезпечують користувачів інструментами для управління їхніми колекціями книг, з можливістю зберігати книги для офлайн доступу.

Мобільні застосунки для читання книг стають все більш популярними у сучасному світі, у якому користувачі шукають зручність та доступність у використанні цифрових ресурсів [1]. Google Books надає обширну бібліотеку книг у цифровому форматі, що робить його ідеальним джерелом для інтеграції з мобільним застосунком. Розробка такого застосунку вимагає глибокого розуміння як технічних аспектів так і вподобань користувачів.

Перший крок у змістовному аналізі полягає у вивченні поточних тенденцій у сфері електронного читання. Значне зростання числа користувачів смартфонів сприяє збільшенню попиту на мобільні читацькі додатки. Користувачі цінують застосунки, що дозволяють легко отримати широкий спектр текстів, вміють синхронізувати закладки та нотатки між пристроями, а також пропонують різноманітні налаштування читацького досвіду. Одним із ключових аспектів є вивчення інтерфейсу та функціональності, які вже доступні в існуючих застосунках для читання книг. Це включає можливість персоналізації читацького досвіду, наприклад, регулювання шрифтів, тем, яскравості та інших візуальних налаштувань, що роблять читання менш втомливим та приємнішим. Важливо

також забезпечити функції, які допомагають управлінню колекцією книг, такі як створення списків для читання, маркування прочитаних книг та синхронізація цієї інформації між різними пристроями.

Другий етап полягає у зрозумінні особливостей інтеграції з Google Books. Цей сервіс надає API, який дозволяє доступ до мільйонів книг, включаючи публічні домени та ті, що продаються через Google Play. Використання API вимагає знань про OAuth 2.0 для авторизації користувачів, а також знань про обробку JSON-форматів для отримання інформації про книги. Також потрібно враховувати політики конфіденційності та умови користування сервісу, щоб забезпечити законність доступу до контенту.

Ще один важливий [2] аспект – це те, для якої операційної системи буде створено застосунок. З моменту свого запуску в кінці 2007 року платформа Android зазнала експоненціального зростання. Починаючи з однієї трубки та оператора, платформа виросла до десятків пристроїв усіх основних операторів по всьому світу [3]. Для розробки обрано платформу Android, оскільки ця платформа є найпопулярнішою мобільною платформою у світі. За останніми підрахунками, у світі було понад два мільярди активних пристроїв Android, і це число швидко зростає. Android – це комплексна платформа з відкритим вихідним кодом на базі Linux, яку підтримує Google [4]. Це потужний фреймворк для розробки, який містить усе необхідне для створення чудових застосунків за допомогою суміші Java та XML. Це також дозволяє розгортати програми на різноманітних пристроях – телефонах, планшетах тощо.

Типова програма для Android складається з одного або кількох екранів. Розробник визначає, як виглядає кожен екран, створюючи макет для його зовнішнього вигляду [5]. Макети зазвичай визначаються в XML і можуть містити такі компоненти графічного інтерфейсу, як кнопки, текстові поля та мітки. Одним з основних недоліків Android є фрагментація. Оскільки Android використовується на різних пристроях від різних виробників, існує багато різних версій операційної системи, які можуть відрізнятися за функціями та сумісністю. Це може призводити до проблем з оновленням програмного забезпечення та

підтримкою нових функцій. Тому вирішено розробляти застосунок для новіших версій операційної системи Android [6]. З безперервним зростанням кількості пристроїв, постійним збільшенням потужності ЦП і ГП, а також безупинною еволюцією самої операційної системи Android, потреба в професійних розробниках додатків буде тільки зростати [7].

Структурні особливості мобільного застосунку для читання книг з Google Books вимагають ретельного планування та розробки, щоб забезпечити гладке та приємне використання застосунку. Основними компонентами структури такого застосунку є користувацький інтерфейс, система управління контентом, інтеграція з зовнішніми API та система управління даними.

Інтерфейс користувача – це візуальні та інтерактивні елементи цифрового продукту, такого як вебсайт або застосунок, з якими користувачі взаємодіють безпосередньо [8]. Користувацький інтерфейс є однією з найважливіших частин мобільного застосунку для читання книг. Інтерфейс має бути інтуїтивно зрозумілим, легким для навігації та візуально привабливим. Важливо забезпечити, щоб інтерфейс дозволяв користувачам легко переглядати каталог книг, шукати книги за ключовими словами, авторами або жанрами, та читати обрані книги з комфортом. Включення функцій налаштування тексту, таких як зміна розміру шрифту та колір тла, допоможе користувачам адаптувати досвід читання до своїх персональних вподобань.

Система управління контентом є ключовою для ефективної роботи застосунку. Вона має управляти бібліотекою контенту, забезпечуючи оновлення книжкових колекцій і забезпечуючи, що нові видання книг автоматично інтегруються в застосунок. Ця система повинна взаємодіяти з Google Books API для отримання та синхронізації даних про книги, включаючи описи, обкладинки та інформацію про доступність.

Наостанок, система управління даними має вирішувати завдання зберігання, доступу та захисту користувацької інформації. Це включає дані про читацькі звички, закладки, нотатки та історію читання. Оптимальне управління даними дозволяє користувачам продовжувати читання з точки, де вони

зупинилися, незалежно від пристрою, на якому вони користуються застосунком. Захист особистих даних має бути на високому рівні, особливо у випадках, коли користувачі здійснюють покупки книг або вводять персональну інформацію.

Мобільний застосунок для читання книг з Google Books повинен включати ряд функціональних особливостей, що забезпечують зручність та ефективність користування для читачів. Ці особливості повинні охоплювати широкий спектр вимог користувачів, від базового перегляду та читання книг до розширених функцій управління бібліотекою та персоналізації.

Однією з основних функцій застосунку є можливість пошуку та вибору книг. Користувачі повинні мати змогу легко знаходити книги за назвою, автором, жанром або ключовими словами.

Організація книг за категоріями, які обере користувач, є ще однією ключовою функцією, яку варто включити у мобільний застосунок для читання книг з Google Books. Це дозволить користувачам персоналізувати свій досвід шляхом створення власних категорій чи колекцій, таких як «Прочитані», «Улюблені», «В планах» або «Читаю зараз». Такий підхід не тільки полегшує навігацію та доступ до бажаних книг, але й дозволяє користувачам краще організувати свої читацькі зацікавлення та плани.

Додатково, для забезпечення ширшого доступу і зручності користування, інтерфейс мобільного застосунку буде повністю англійською мовою. Це допоможе залучити міжнародну аудиторію та забезпечити, що користувачі з різних країн матимуть можливість ефективно використовувати застосунок.

Конкретною проблемою, яку вирішує мобільний застосунок для читання книг з Google Books, є складність доступу до великої кількості літературних ресурсів у зручному та інтегрованому форматі. Багато користувачів стикаються з проблемами організації своїх бібліотек, доступу до нових книг та збереження своїх нотаток та закладок при зміні пристроїв. Ця проблема значно впливає на кінцевих користувачів, оскільки вони витрачають більше часу на пошук і організацію книг, що може знижувати їхнє задоволення від читання. Вирішення цієї проблеми через розробку інтегрованого мобільного застосунку дозволить

користувачам ефективніше управляти своїми книгами, економити час та підвищити якість читання, що є важливим для забезпечення позитивного користувацького досвіду.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Аналіз наявного програмно-технічного забезпечення для мобільного застосунку читання книг з Google Books включає вивчення існуючих технологій, платформ, бібліотек і інтерфейсів, які можуть бути використані або адаптовані для створення функціонального і зручного застосунку. Цей процес забезпечує основу для вибору найбільш ефективних інструментів та методів розробки.

В нинішній час існує багато схожих розробок у даній предметній області, тож для кращого порівняння буде описано кілька з них, а саме Librarius, Goodreads, Bookmate, Chapter1. Вказуватиметься: призначення програмного продукту; фірма-розробник; основні інтерфейсні вікна; переваги та недоліки даних застосунків тощо.

Librarius – це перша в Україні система з захищеним форматом файлу електронної книжки та аудіокниги (рисунок 1.1) [9]. Застосунок дозволяє імпортувати колекцію книг і розташовувати їх на віртуальних полицях, моделюючи реальні полиці користувача. Програма розроблена Amitai Blickstein і пропонує такі інтерфейсні вікна, як головна сторінка з віртуальними полицями та менеджер книжкових кейсів.

Переваги включають інтуїтивно зрозумілий спосіб організації книг та можливість візуального моделювання книжкового простору.

Недоліки полягають у тому, що програма все ще знаходиться у Бета-версії, що може призвести до технічних помилок і обмежень у функціоналі. Також програма доступна англійською мовою.

Рисунок 1.1 – Каталог Librarius [10]

Goodreads – це популярна онлайн-платформа, яка дозволяє людям оцінювати книги та ділитися своїми враженнями, а також взаємодіяти з іншими читачами [11]. Платформа розроблена компанією Goodreads Inc., яка наразі є частиною Amazon.

Основні інтерфейсні вікна включають:

– домашня сторінка з рекомендаціями книг;

– сторінки книг з деталями, рецензіями та рейтингами;

– сторінки користувачів з їхніми полицями книг та наявними читацькими статистиками.

Goodreads (рисунок 1.2) володіє значними перевагами: великою базою даних книг із рецензіями, активною спільнотою читачів для обговорень, а також інструментами для відстеження та планування читання. Це робить платформу особливо цінною для ентузіастів літератури.

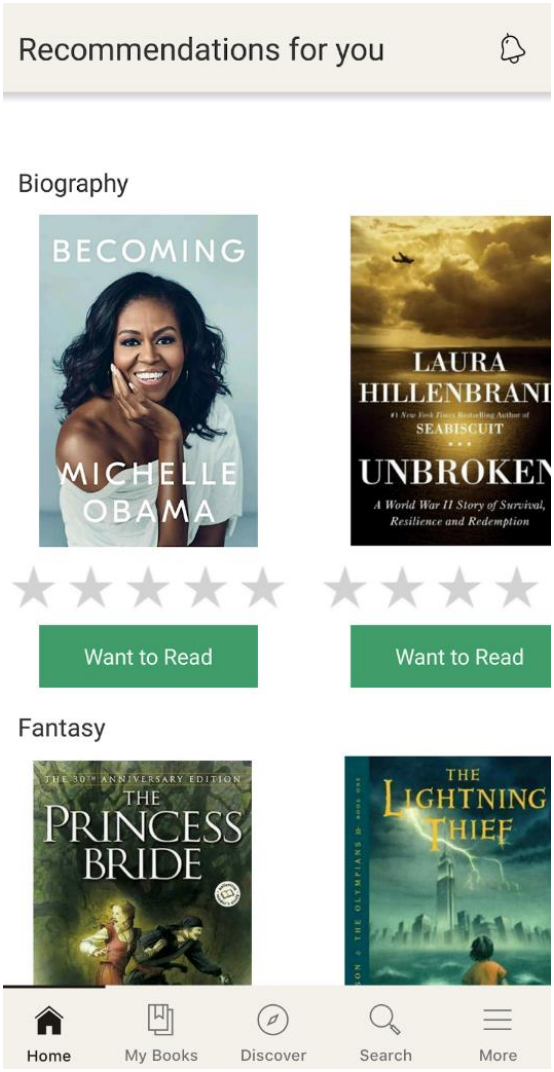


Рисунок 1.2 – Головна сторінка Goodreads [12]

Проте існують і певні недоліки: інтерфейс може здатися перевантаженим, приватність даних може викликати занепокоєння через тісний зв'язок з Amazon, а також функціональність сервісу є обмеженою без доступу до інтернету. Ці аспекти можуть вплинути на досвід користувача.

Bookmate – це багатоплатформенний сервіс, що надає доступ до великої бібліотеки електронних книг і аудіокниг (рисунок 1.3). Він призначений для широкого кола читачів, які бажають зручно читати та слухати книги в дорозі, вдома або в інших місцях. Користувачі можуть вибирати книги з різних жанрів і авторів, створювати власні колекції та ділитися враженнями з іншими. Bookmate був розроблений однойменною компанією. З часом продукт набув популярності і тепер використовується читачами по всьому світу.

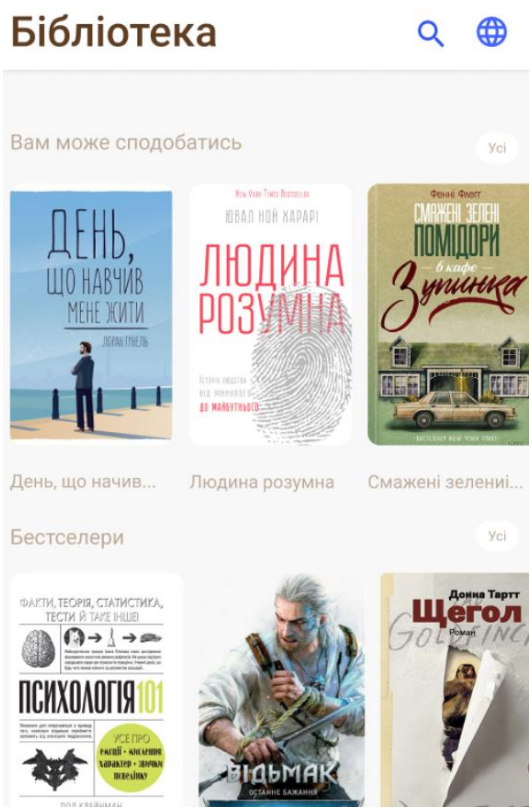


Рисунок 1.3 – Бібліотека в Bookmate [13]

Основні інтерфейсні вікна:

– Головна сторінка з рекомендаціями книг, базуючись на інтересах і читацькій історії користувача.

– Бібліотека, де користувачі можуть переглядати свої книги, аудіокниги та колекції.

– Пошукова сторінка для знаходження книг за назвами, авторами або ключовими словами.

– Вікно налаштувань для кастомізації досвіду читання та управління аккаунтом користувача.

Застосунок надає широку бібліотеку з різноманіттям жанрів, дозволяючи користувачам читати і слухати книги на будь-якому пристрої з доступом до інтернету. Персоналізовані рекомендації, що адаптуються до індивідуальних вподобань користувача, разом із функціями соціальної взаємодії, які дозволяють обмінюватися думками та колекціями з іншими, роблять цей застосунок зручним інструментом для любителів книг. Але для повного доступу до всіх книг та

функцій необхідна підписка, що може бути обтяжливим для деяких користувачів. Крім того, вибір з усіх книг може бути обмеженим у певних мовах або жанрах.

Chapter1 – це мобільний застосунок для читання книг онлайн та офлайн, який пропонує персоналізовані рекомендації користувачам. Метою цього застосунку є забезпечення зручного доступу до великої бібліотеки книг та надання індивідуальних рекомендацій на основі уподобань користувачів.

Застосунок розроблений компанією Chapter1 Ltd., яка спеціалізується на створенні програмного забезпечення для читання та управління цифровими бібліотеками. Головні інтерфейсні вікна включають Головну сторінку, де відображаються рекомендовані книги, Пошук для швидкого знаходження книг, Моя бібліотека, яка містить особисту колекцію книг користувача, Читання книги для зручного перегляду контенту, та Рекомендації, що надають персоналізовані пропозиції книг.

Застосунок Chapter1 пропонує широку бібліотеку з різноманітними жанрами, що задовольняє потреби широкого кола читачів. Важливою перевагою є можливість отримання персоналізованих рекомендацій, що допомагає користувачам відкривати нові книги, які можуть відповідати їхнім інтересам. Крім того, можливість читання офлайн дозволяє користувачам продовжувати читання навіть без підключення до інтернету, що додає зручності у використанні.

Основна функціональність включає пошук і читання книг, збереження закладок для зручного повернення до потрібних сторінок, отримання персоналізованих рекомендацій на основі прочитаних книг та уподобань користувача, а також можливість читання книг в офлайн режимі.

Підсумковий аналіз представлений у таблиці А1 у додатку А.

Проаналізувавши деякі наявні програмні рішення для обраної предметної області було вирішено зробити застосунок, що має подібний функціонал з деякими відмінностями, створити новий цікавий та інтуїтивний дизайн інтерфейсу для майбутнього користувача.

1.3 Визначення вимог до програмного забезпечення та технічне завдання

Основним інструментом інженерії вимог до програмного забезпечення в рамках об'єктно-орієнтованої парадигми є мова Unified Modeling Language (UML). У першу чергу розробляються моделі варіантів використання (ВВ). Діаграма варіантів використання відображає виконання конкретних обов'язків користувачами з використанням програмного забезпечення. Правильне використання нотацій UML може значно підвищити ясність і комунікативну силу програмного забезпечення, спрощуючи фіксацію складних взаємозв’язків і взаємодій, що є в програмі [14].

При створенні такої моделі доцільно заздалегідь скласти короткі описи користувачів програмного забезпечення (акторів) та необхідних їм сервісів (ВВ). Приклад складеного опису користувачів наведено у таблиці 1.1.

Таблиця 1.1 – Опис користувачів (акторів)

Актор	Короткий опис
Користувач (Читач)	Основний користувач застосунку, який читає книги. Взаємодіє з системою через графічний інтерфейс.
Система Google Books	Обробляє запити користувачів. Інтегрується з Google Books API для отримання даних про книги.

Варіанти використання допомагають визначити функціональності, які система повинна забезпечити, та взаємодію між користувачами та системою. Кожен варіант використання описує сценарій, в якому система взаємодіє з кінцевими користувачами чи іншими системами, з метою досягнення конкретної цілі користувача [15]. Цей підхід дозволяє розробникам та стейкхолдерам краще розуміти функціональні вимоги до системи та сприяє розробці користувацьки-орієнтованого програмного забезпечення. Приклад відповідний до предметної області зображено в таблиці 1.2.

Таблиця 1.2 – Опис варіантів використання

Актор	Найменування ВВ	Опис ВВ
Система Google Books	Перегляд книжкового каталогу	Користувач може переглядати список доступних книг, що надаються через Google Books.
Користувач (Читач)	Пошук книг	Користувач може використовувати функціонал пошуку для знаходження книг у каталозі.
Користувач (Читач)	Читання книг	Користувач може відкривати і читати книги в застосунку.
Користувач (Читач)	Зміна налаштувань читання	Користувач може змінювати налаштування читання (наприклад, розмір шрифту, колір тла).

UML діаграми варіантів використання моделюють поведінку системи та допомагають охопити вимоги системи. Ці діаграми також ідентифікують взаємодію між системою та її акторами. Варіанти використання та актори на діаграмах варіантів використання описують, що робить система та як актори її використовують, але не описують те, як система працює всередині [16].

Діаграми варіантів використання ілюструють і визначають контекст і вимоги всієї системи або важливих частин системи. Можна змоделювати складну систему за допомогою однієї діаграми варіантів використання або створити багато діаграм варіантів використання для моделювання компонентів системи. Зазвичай діаграми варіантів використання розробляються на ранніх етапах розробки проєкту. UML і шаблони разом забезпечують потужний інструментарій для моделювання та вирішення проблем у розробці програмного забезпечення, підтримуючи ітераційні процеси та забезпечуючи кращі дизайнерські рішення [17].

У даному проєкті буде показано одну UML-діаграму варіантів використання тому, що акторів як і варіантів використання не багато та сама структура не є складною для розуміння. Нижче подано діаграму на рисунку 1.4.

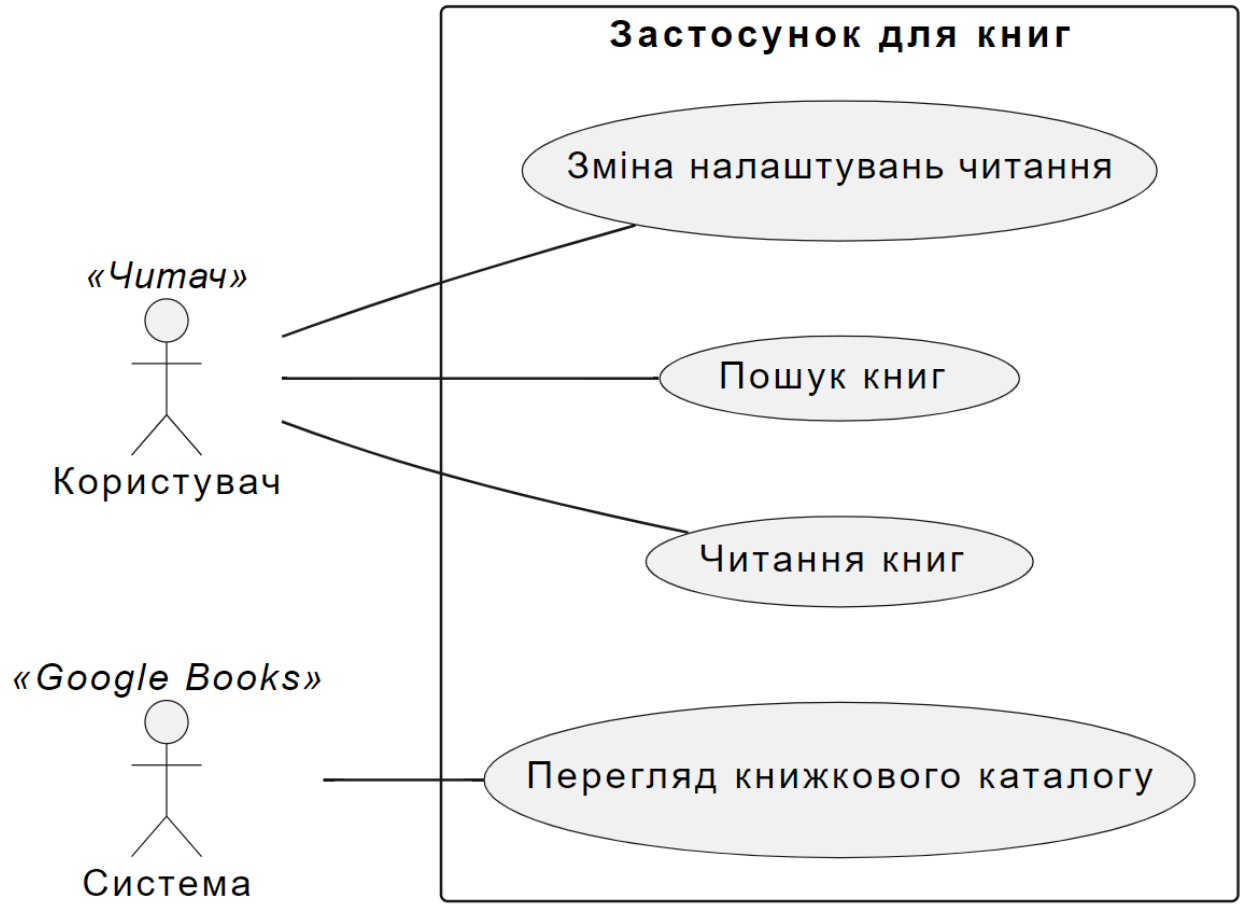


Рисунок 1.4 – Загальна UML-діаграма варіантів використання

Функціональні вимоги до мобільного застосунку для читання книг з Google Books визначають, які можливості та сервіси повинні бути реалізовані для задоволення потреб користувачів. Функціональна вимога - це формулювання того, як система повинна поводитися [18]. Ефективна розробка вимог має вирішальне значення для успішних програмних проєктів, оскільки вона закладає основу для розуміння того, що потрібно користувачам і як повинна функціонувати система [19].

Перш за все, застосунок має забезпечити зручний доступ до обширної бібліотеки книг, яка включає як безкоштовні, так і платні видання. Користувачі повинні мати можливість легко шукати книги за назвою, автором, жанром або ключовими словами. Окрім того, важливою є можливість перегляду детальної інформації про книгу.

Далі, застосунок має надавати функції персоналізації, що дозволяють користувачам створювати власні списки для читання, додавати книги до обраного, відстежувати свій прогрес у читанні. Інтерфейс має бути інтуїтивно зрозумілим та адаптивним до різних розмірів екранів, забезпечуючи високу якість читання як на смартфонах, так і на планшетах.

Особливу увагу потрібно приділити захисту особистої інформації та даних користувачів, особливо при синхронізації та передачі даних через мережу [20].

Функціональні вимоги повинні враховувати потреби всіх користувачів, включаючи тих, хто має спеціальні потреби. Застосунок має включати функції налаштування зовнішнього вигляду (контрастність, розмір шрифту), щоб забезпечити комфортне читання для всіх користувачів.

Для повного розуміння функціональних вимог можна додати конкретні сценарії використання, наприклад вхід до системи (рисунок 1.5):

Рисунок 1.5 – UML-діаграми варіантів використання входу користувача до системи

Передумови: Користувач увійшов до системи.

Основний сценарій:

– Користувач вводить ключові слова у поле пошуку.
– Користувач натискає кнопку "Пошук".
– Система відправляє запит до Google Books API.
– Система отримує результати пошуку.
– Система відображає результати пошуку користувачу.

Альтернативні сценарії: Якщо запит не дає результатів, система повідомляє про відсутність відповідних книг.

Умови завершення: Система відображає результати пошуку або повідомлення про відсутність результатів.

Післяумови: Користувач переглядає список знайдених книг і може вибрати потрібну книгу.

Нефункціональні вимоги – це набір специфікацій, які описують робочі можливості та обмеження системи [21]. Нефункціональні вимоги до мобільного застосунку для читання книг з Google Books визначають якість та стандарти, які застосунок має відповідати для забезпечення належного рівня сервісу. Ці вимоги включають аспекти, такі як продуктивність, безпека, надійність, та здатність до масштабування майбутнього застосунку.

Однією з ключових нефункціональних вимог є продуктивність [22]. Застосунок повинен забезпечувати швидке завантаження та відображення книг, незалежно від розміру файлів або якості мережевого з'єднання. Це особливо важливо для користувачів, які можуть використовувати застосунок в умовах зі слабким інтернетом. Оптимізація відклику на дії користувача та мінімізація часу очікування на завантаження також є критичними для покращення загального досвіду користування.

Безпека є ще однією критичною нефункціональною вимогою. Застосунок має включати сучасні механізми шифрування для захисту особистих даних користувачів та їхніх платіжних інформацій [23]. Реалізація строгих політик авторизації та аутентифікації допоможе убезпечити користувачів від несанкціонованого доступу до їхніх акаунтів.

Надійність та доступність також є важливими нефункціональними вимогами. Застосунок має бути спроєктований таким чином, щоб забезпечити стабільну роботу навіть у пікові навантаження або під час технічних збоїв.

Функціональні вимоги: інтеграція з Google акаунтом для входу в застосунок; можливість шукати книги за назвою, автором, жанром; вибір тем оформлення, шрифтів та інших налаштувань читання; можливість створювати списки для читання, додавати книги до обраного.

Нефункціональні вимоги: застосунок повинен швидко завантажувати контент і відповідати на дії користувача; захист особистих даних користувачів і їх платіжної інформації; висока доступність сервісу та стійкість до помилок; підтримка останніх версій операційної системи Android; здатність застосунку ефективно обробляти збільшення кількості користувачів.

Для мобільного додатку для читання книг з Google Books, дизайн інтерфейсу має бути інтуїтивно зрозумілим і зручним для користувача. Нижче описано основні інтерфейсні сторінки, які повинні бути реалізовані у застосунку.

– Сторінка входу – сторінка дозволяє користувачам ввійти в застосунок за допомогою свого Google акаунта. Сама сторінка має назву додатку, логотип та одну кнопку при натисканні якої застосунок відкриє головну сторінку.

– Після входу в програму, користувачі потрапляють на головну сторінку, де відображаються всі книги, які користувач додав до своєї бібліотеки. Книги можуть бути організовані за категоріями або списками, що дозволяє користувачам легко управляти своїми читацькими планами. На даному екрані є поле пошуку [24] – це пошукове поле, яке дозволяє користувачу здійснити пошук за допомогою введення ключового слова. Після його заповнення відбувається відображення результатів. Зазвичай біля такого поля завжди знаходиться кнопка пошуку із символом лупи в центрі.

– Також з даного екрану, за допомогою кнопки в верхньому правому куті, можна перейти на додаткові сторінки, що мають назви «Читаю зараз», «Прочитано», «Улюблені» та «В планах».

– Додаткові сторінки мають майже ідентичний вигляд окрім підпису самої сторінки зверху та її наповнення. Тут мають опинятися книги обрані одну з категорій «Читаю зараз», «Прочитано», «Улюблені» та «В планах».

– Коли користувач вибирає книгу, він переходить на сторінку з детальною інформацією про книгу. Тут відображається обкладинка книги, опис, а також кнопки для читання або додавання книги до списку.

– Наявно 5 кнопок, перша це випадаюче меню для вибору категорії в яку поміститься книга, наступна кнопка відповідає за відкриття самої книги для

читання, третя кнопка відчиняє попередній перегляд книги, тобто кілька перших сторінок. Четверта кнопка дає змогу купити книгу, якщо вона не є безкоштовною, та остання, п’ята – видаляє книгу з обраної раніше категорії. Також є кнопка повернення на попередній екран застосунку для читання.

– Читання книги. На цій сторінці відображається текст книги. Користувачі можуть налаштувати параметри читання, такі як розмір шрифту, колір тла та інші візуальні налаштування для забезпечення комфортного читання. Для цього є маленьке меню, що викликається кнопкою у вигляді шестерні. На ньому можна обрати одну із кольорових схем та змінити колір шрифту на один з запропонованих застосунком.

– Важливим елементом є меню, яке є майже на усіх сторінках застосунку. Це дозволяє легко проводити навігацію по програмному продукту. На меню є три кнопки. Перша у вигляді будинку яка веде на головну сторінку, друга у вигляді сердечка, натиснувши яке можна потрапити на сторінку з улюбленими книгами, та третя є кнопкою загальної інформації про програму.

1.4 Висновки. Постановка задачі

Розробка мобільного застосунку для читання книг вимагає дотримання певних технічних вимог та обмежень, щоб забезпечити зручність, продуктивність та безпеку. Застосунок повинен працювати на операційній системі Android версії 4.1 і вище, підтримувати різні розміри екранів, включаючи смартфони та планшети, а також забезпечувати швидке і стабільне з'єднання з Google Books API для мінімізації затримок при пошуку та завантаженні книг. Інтерфейс користувача повинен бути інтуїтивно зрозумілим, з підтримкою темного і світлого режимів. Функціональність застосунку включає вхід користувача через обліковий запис Google, пошук книг, читання книг в онлайн та офлайн режимах, збереження закладок і приміток, а також персоналізовані

рекомендації на основі історії читання. Застосунок повинен забезпечувати захист персональних даних користувачів застосунку.

У створенні мобільного застосунку для читання книг з Google Books основним завданням є розробка інтуїтивно зрозумілого, зручного та функціонального інтерфейсу, який інтегрується з великою базою даних книг Google. Застосунок має надавати користувачам можливості не тільки читання, але й управління персоналізованими бібліотеками книг.

Важливо забезпечити високу продуктивність застосунку при роботі з великими об'ємами даних та забезпечити надійність і безпеку користувацьких даних. Завдання проєктування включає в себе створення докладного технічного завдання, вибір технологічного стеку для розробки, планування архітектури додатку та розробку стратегії тестування і впровадження.

Постановка задач проєктування для мобільного додатку для читання книг з Google Books вимагає інтеграції декількох ключових компонентів. Перш за все, необхідно забезпечити ефективну інтеграцію з Google Books API для доступу до величезної бази книг. Дизайн інтерфейсу повинен бути зорієнтований на користувацький досвід, пропонуючи простий та інтуїтивно зрозумілий навігаційний потік. Дизайн інтерфейсу користувача створює візуальні елементи та інтерактивні властивості, які дозволяють користувачам отримувати доступ, розуміти та використовувати інтерфейс для виконання бажаних дій. Ця галузь дизайну має на меті забезпечити гарний вигляд інтерфейсу [25].

Забезпечуючи інтерактивний досвід, прототипи з можливістю натискання полегшують тестування користувачами та відгуки, що зрештою призводить до більш досконалого та успішного кінцевого продукту [26].

Застосунок має підтримувати основні функції читання, включаючи зміну шрифтів, яскравості екрану та інші налаштування для комфорту читача. Забезпечення безпеки особистих даних користувачів є критичним, особливо при обробці авторизації.

Також необхідно розробити систему управління контентом, яка дозволить користувачам вести управління власними колекціями книг. Кожен аспект

розробки повинен бути підтверджений через ретельне тестування на різних пристроях, щоб забезпечити стабільність та високу продуктивність застосунку.

Серія ітерацій використовується для агрегування більшої загальносистемної функціональності для випуску (або потенційного випуску) для зовнішніх користувачів [27].

Очікувані результати проєкту включають безпечний вхід користувачів у систему за допомогою своїх облікових записів Google, швидкий і точний пошук книг за ключовими словами, авторами або назвами, а також зручне читання книг в онлайн та офлайн режимах. Показники успіху включають високий відсоток успішних входів без помилок автентифікації, швидкий час відповіді на пошукові запити, плавне перегортання сторінок без затримок, та надійне збереження останньої прочитаної сторінки і закладок. Важливо також, щоб користувачі отримували персоналізовані рекомендації, що відповідають їхнім інтересам.

Аналіз предметної області показав необхідність створення зручного та функціонального мобільного застосунку для читання книг, який би вирішував проблеми організації бібліотек, доступу до нових книг та збереження прогресу читання. Існуючі програмні рішення мають різні недоліки, такі як складність інтерфейсу або обмежений вибір книг, що підкреслює важливість створення нового застосунку.

Вимоги до майбутнього ПЗ включають забезпечення високої продуктивності, безпеки та сумісності з різними пристроями, а також надання користувачам інтуїтивно зрозумілого інтерфейсу та широкого функціоналу. Важливо врахувати, що застосунок має бути гнучким, щоб адаптуватися до швидко змінюваних технологічних умов і користувацьких вимог.

Стабільне оновлення функціоналу, безпечне зберігання даних і здатність до масштабування будуть ключовими для довгострокового успіху у розробці цього проєкту.

2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та вибір технологій і методів реалізації

У даному розділі буде оцінено можливі платформи для розробки, мови програмування, фреймворки та бібліотеки, які допоможуть нам забезпечити функціональність, зручність і швидкодію застосунку.

Оскільки застосунок розроблятиметься на операційну систему Android логічно обрати ті середовища розробки, що найкраще підходять для цього та мають великий функціонал та інструментарій в своєму арсеналі. Дуже важливо щоб засоби, що допоможуть в розробці не були застарілими та відповідали сучасним вимогам.

Щоб створити привабливий дизайн для застосунку, потрібно обрати програму, в якій можна створити динамічний прототип інтерфейсу для повної наглядності того, яким буде майбутній застосунок. Тому для цієї задачі було обрано дуже популярний сервіс під назвою Figma.

Figma – це графічний інструмент для дизайну та прототипування. Figma дає змогу користувачам взаємодіяти в реальному часі для створення та прототипування користувацьких інтерфейсів і вебзастосунків. Використовуючи інструменти векторного графічного дизайну, команди можуть створювати складні каркасні макети вебсайтів, масштаби яких можна змінювати для оптимізації під екрани будь-якого заданого розміру.

Крім того, користувачі можуть додавати у свої проєкти інтерактивні елементи, як-от функції прокрутки та наведення, що дає змогу їм переконатися, що їхні вебсторінки мають сучасний дизайн. Це дає змогу користувачам створювати робочі прототипи своїх вебсторінок, які можна легко протестувати. Figma навіть дає змогу експортувати деякий обсяг коду з проєктів прототипів для передання розробникам. Головна відмінність даного сервісу у тому, що можна працювати безпосередньо у браузері [28].

Але порівняно з іншими програмами подібного типу Figma має менші можливості для детального редагування векторних об'єктів. Також, не має

багатьох функцій типографіки. Відсутній повноцінний механізм для створення та збереження кастомних шаблонів, що однозначно позначиться на швидкості роботи під час повторного створення елементів дизайну. Інструменти редагування тексту в Figma можуть бути не зручними для користувачів, які працюють з більш складним текстом, наприклад, з довгими списками [29].

Для розробки кодової частини засотсунку було обрано середовище розробки Android Studio. Це офіційне інтегроване середовище розробки Google для операційної системи Android, створене на основі програмного забезпечення JetBrains IntelliJ IDEA та розроблене спеціально для розробки Android. Система збірки Android – це набір інструментів, які використовуються для створення, тестування, запуску та упаковки ваших програм [30].

Найбільш очевидною перевагою використання Android Studio є наявність розширених інструментів кодування. Розширені набори інструментів дозволяють розробникам створювати програми преміум-класу з відмінною структурою кодування за набагато менший час.

Android Studio поставляється з віртуальним пристроєм Android, його емулятором, який дозволяє розробникам запускати та налагоджувати програми в ньому. Крім того, він також має інструменти Lint, за допомогою яких розробники можуть перевіряти продуктивність додатка до його розробки, що призводить до розгортання бездоганних застосунків.

Коли справа доходить до налаштування, Android Studio пропонує розробникам більше можливостей для цього завдяки передовим інструментам розробки, таким як дуже гнучка система на основі Gradle та багатофункціональні API попереднього перегляду. Завдяки редактору макета розробники також можуть створювати розкішний інтерфейс користувача. Він підтримує C++ і Java. Останні версії також підтримують Kotin. Його підтримка різних мов також полегшує інтеграцію з інструментами сторонніх розробників [31].

Але також є недоліки, наприклад встановлення Android Studio займає якийсь час. Незалежно від того, наскільки гарною є конфігурація системи, потрібно посидіти та спостерігати за тривалим процесом встановлення.

Ще одним недоліком Android Studio є системні вимоги. Програма не працюватиме на системі низького рівня. Перш ніж працювати з даним середовищем розробки, необхідно перевірити процесор, оперативну пам'ять і інші характеристики комп'ютера.

Емулятор Android Studio хороший, коли мова йде про функції, але недостатній, коли справа стосується швидкого тестування. Порівняно з емуляторами сторонніх розробників, власний емулятор працює повільніше та може обробляти тестування однієї програми за раз. Системні провисання є звичайним явищем для Android Studio і часто можуть заважати розробнику.

Java – це широко використовувана об'єктно-орієнтована мова програмування та програмна платформа, яка працює на мільярдах пристроїв, включаючи ноутбуки, мобільні пристрої, ігрові консолі, медичні пристрої та багато інших. Правила і синтаксис Java засновані на мовах C і C++ [32].

Однією з основних переваг розробки програмного забезпечення за допомогою Java є його переносимість. Після того як код буде написано для програми Java на ноутбуці, його можна легко перемістити на мобільний пристрій чи інший портативний гаджет схожий до смартфона.

Ця мова програмування була обрана для створення застосунку в основному тому, що це одна з мов які підтримує Android Studio.

XML (розширювана мова розмітки) – це спосіб перепризначення даних у файлі або автоматизації процесу заміни даних в одному файлі даними з іншого файлу. XML використовує теги щоб описувати частини файлу – наприклад, заголовок чи матеріал. Ці теги відзначають дані, так що їх можна зберігати у файлі XML та належним чином обробляти, коли їх експортують до інших файлів. Думайте про XML як про механізм перетворення даних. Теги XML маркують текст та інший вміст у файлі так, що програми можуть розпізнавати та представляти дані [33].

Також буде використовуватися багато різних бібліотек, що доступні для Android Studio, щоб легше було розробляти кодову частину та вдосконалювати сам застосунок. Наприклад бібліотека AndroidX є сучасною бібліотекою для

розробки Android застосунків. AndroidX пропонує модульну структуру, складну з кількох компонентів (модулів), кожен з яких спрямований на певну функціональність або компонент Android. AndroidX також містить сучасні компоненти і інструменти для розробки, такі як підтримка Material Design, покращена підтримка фрагментів, тощо.

Або дуже корисна бібліотека Picasso, що дозволяє завантаження та відображення зображень в додатках Android. Також бібліотека Volley, яка використовується для мережевих запитів в додатках Android. Додавши цю залежність до проєкту, можна взаємодіяти з мережею, виконувати HTTP-запити та обробляти відповіді сервера.

Усі обрані програми та застосунки будуть використані для розробки

2.2 Виділення особливостей інформаційної системи Google API

Мобільний застосунок для читання книг з Google, побудований на основі Book API, яка надає функціонал для прив’язки профілів до системи обробки книг в акаунті. Перед проєктуванням архітектури майбутнього застосунку, потрібно вивчити структуру даної системи.

Перше, що було опрацьовано, це те що дана технологія надає можливості звертатись до її компонентів за допомогою системи RESTFul API. Даний підхід до запитів – це спосіб побудови вебсервісів у відповідності до принципів архітектури REST. REST використовує HTTP протокол для передачі даних між клієнтом і сервером. Основні принципи REST включають в себе використання уніфікованих URL.

Це дозволяє звертатись до елементів структури обробки запитів за допомогою коротких посилань, які надаватимуть або відповідь, у форматі JSON, або надаватимуть статус сервера по обробці даних. Перша частина запиту завжди починається з розділу, до якого звертається користувач (Таблиця 2.1).

Таблиця 2.1 – Список розділів Google API

Ключове слово	Опис
Volume	Представляє дані про книгу чи журнал, які містяться в Книгах Google. Це основний ресурс в Books API. Усі інші ресурси в цьому API містять або анотують том.
Bookshelf	Колекція томів. Google Books надає набір попередньо визначених книжкових полиць для кожного користувача. Користувачі можуть створювати, змінювати або видаляти інші книжкові полиці, які завжди заповнюються томами вручну. Книжкові полиці користувач може зробити приватними або загальнодоступними.
Review	Комбінація рейтингу та тексту. Користувач може подати одну рецензію на том. Відгуки доступні із зовнішніх джерел і мають атрибути.
Reading Position	Позиція читання вказує на останню позицію читання в тому для користувача. Користувач може мати лише одну позицію читання на том. Якщо користувач раніше не відкривав цей том, позиція не існує.

Для кожного розділу надається список функціоналу, який визначається, як друге ключове слово в запиті. Список функцій обмежена, і залежить від ключових слів, які представлені нижче (Таблиця 2.2).

Таблиця 2.2 – Список доступних функцій Google API

Ключове слово	Тип запиту	Опис
List	GET	Повертає колекцію певних даних.
Insert	POST	Дозволяє вставляти нові дані в Google Book.
Get	GET	Повертає конкретний екземпляр зі списку.
Update	PUT	Дозволяє оновити інформацію про дані.
Delete	DELETE	Видаляє інформацію з Google Book в межах доступності користувача.

Потрібно звернути увагу на те, що Google Book API надає функціонал не для кожного розділу. Повний залежності функціоналу знаходиться на наступному рисунку 2.1.

Тип ресурсу	Підтримувані операції				
	list	insert	get	update	delete
Volumes	так		так		
Bookshelf	так	так, АВТЕНТИФІКАЦІЯ	так	так, АВТЕНТИФІКАЦІЯ	так, АВТЕНТИФІКАЦІЯ
Position		так, АВТЕНТИФІКАЦІЯ	так, АВТЕНТИФІКАЦІЯ	так, АВТЕНТИФІКАЦІЯ	так, АВТЕНТИФІКАЦІЯ

Рисунок 2.1 – Залежність функціоналу від розділів

Для даного застосунку планується використовувати лише доступ до книг та книжкових полиць. Для того, щоб використовувати Google Books API, розробники мають використовувати ключ API, який вимагає реєстрації в Google Cloud Platform. Це забезпечує контроль доступу та дозволяє моніторити використання API.

Окрім того, використання API може бути обмежене в залежності від кількості запитів, що може вимагати планування квоти запитів, яке має бути розглянуте під час впровадження системи на основі цього API [34].

Для реалізації системи управління книгами, можуть бути використані чотири основні полички гугл, які не можуть бути видалені користувачем: «В процесі читання», «Прочитано», «Улюблені», «Відкладено на потім». Ідентифікатори даних поличок починаються від нуля до трьох відповідно. Усі інші ідентифікатори можуть надаватись поличкам створеними самими користувачами у особистому кабінеті, але в контексті даного застосунку, вони використовуватись не будуть.

Якщо створювати запити, які потребують автентифікацію, то в тілі запиту потрібно передавати ключ користувача, який отримується шляхом передачі даних до OAuth 2.0 API [35].

Дії з автентифікацію передбачають саме керування запитами пов’язаними з особистою інформацією. Наприклад, якщо це стосується переміщенням книг по поликах. Повний процес створення запиту і отримання відповіді представлений на наступній діаграмі (рисунок 2.2).

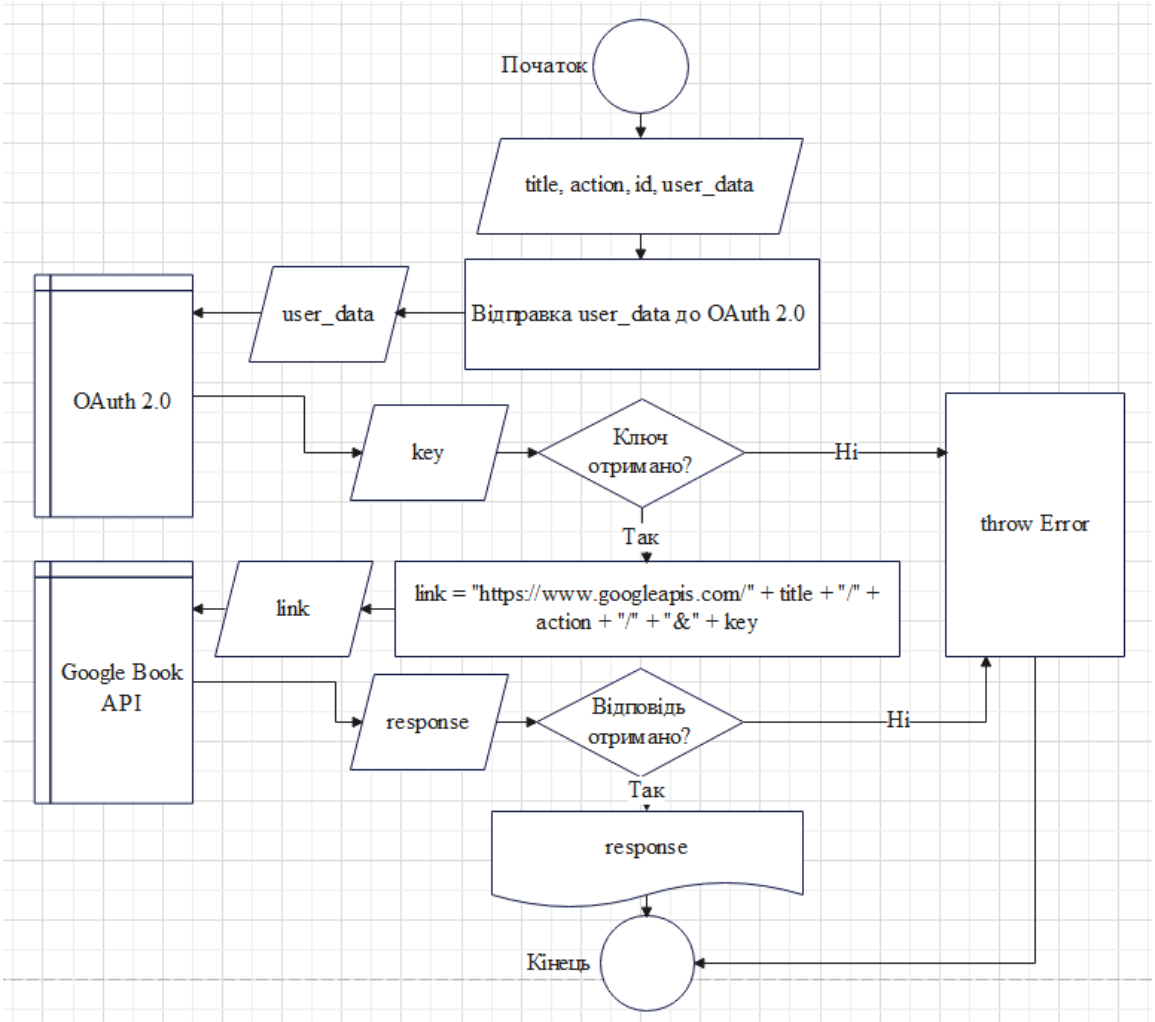


Рисунок 2.2 – Блок-схема відправки запиту до Google Book API

Відповідь, яку надає Google Book API може різниться в залежності від типу запиту. Якщо запит відбувається на вставку, видалення, або додавання то сервер відправляє код статусу успішності операції. Передбачається, що на отримання інформації, відправлятимуться лише запити на отримання книг, або списків книг. В такому разі, відповіді будуть надаватись у форматі JSON і міститимуть усю інформацію про книгу, яку може надати Google [36]. Для даного проєкту будуть застосовані лише деякі компоненти від відповіді, представлені в наступній таблиці (Таблиця 2.3).

Варто також відзначити, що Google не завжди надає коректну інформацію у своїх відповідях, тому, наприклад, посилання на купівлю книги, або на пердперегляд, можуть бути або недійсними, або вести на різні ресурси. Обробку

даної проблеми потрібно занести у алгоритми майбутнього застосунку і повідомляти користувача про несправності.

Таблиця 2.3 – Список використаних властивостей від Google Book API

Властивість	Тип даних	Опис
id	Int	Ідентифікатор книги
shelfID	Int	Ідентифікатор полички, на якій знаходиться книга
title	String	Назва книги
subtitle	String	Підназва книги
authors	String[]	Автори книги. Передається у вигляді масиву.
publisher	String	Видавець
publishedDate	Date	Дата публікації
description	String	Опис книги
pageCount	Int	Число сторінок
thumbnail	String	Посилання на обкладинку книги
previewLink	String	Посилання на передперегляд книги
infoLink	String	Посилання на читання книги
buyLink	String	Посилання на покупку книги
accessViewStatus	Boolean	Перевірка, чи книга уже куплена користувачем

Як видно з таблиці (Таблиця 2.3), посилання на важливі сторінки передаються, як рядок, що вимушує створювати додатковий запит для відображення деякої інформації. Також, це стосується і перегляду обкладинки.

Також, потрібно перевіряти, чи книга уже куплена користувачем. У випадку, коли книга не придбана, потрібно створити вікно з пропозицією перейти на сторінку покупки.

Отже, Google Book API має хорошу структуру запитів, яку зручно можна використати. Але потрібно створювати додаткові запити для відправки інформації про користувача, щоб отримувати код доступу. Це свідчить про те, що застосунок має мати також і систему авторизації у акаунті Google. Також, з даним API потрібно бути обережним, тому що не завжди відповідь може бути подана вірно.

2.3 Проєктування архітектури та структури системи

Перед визначенням компонентів архітектури, на основі функціоналу, який має бути розроблений, було визначено дані, та процеси, які відбуватимуться в середині системи. Загальну діаграму потоків даних, зображено на рисунку 2.3.

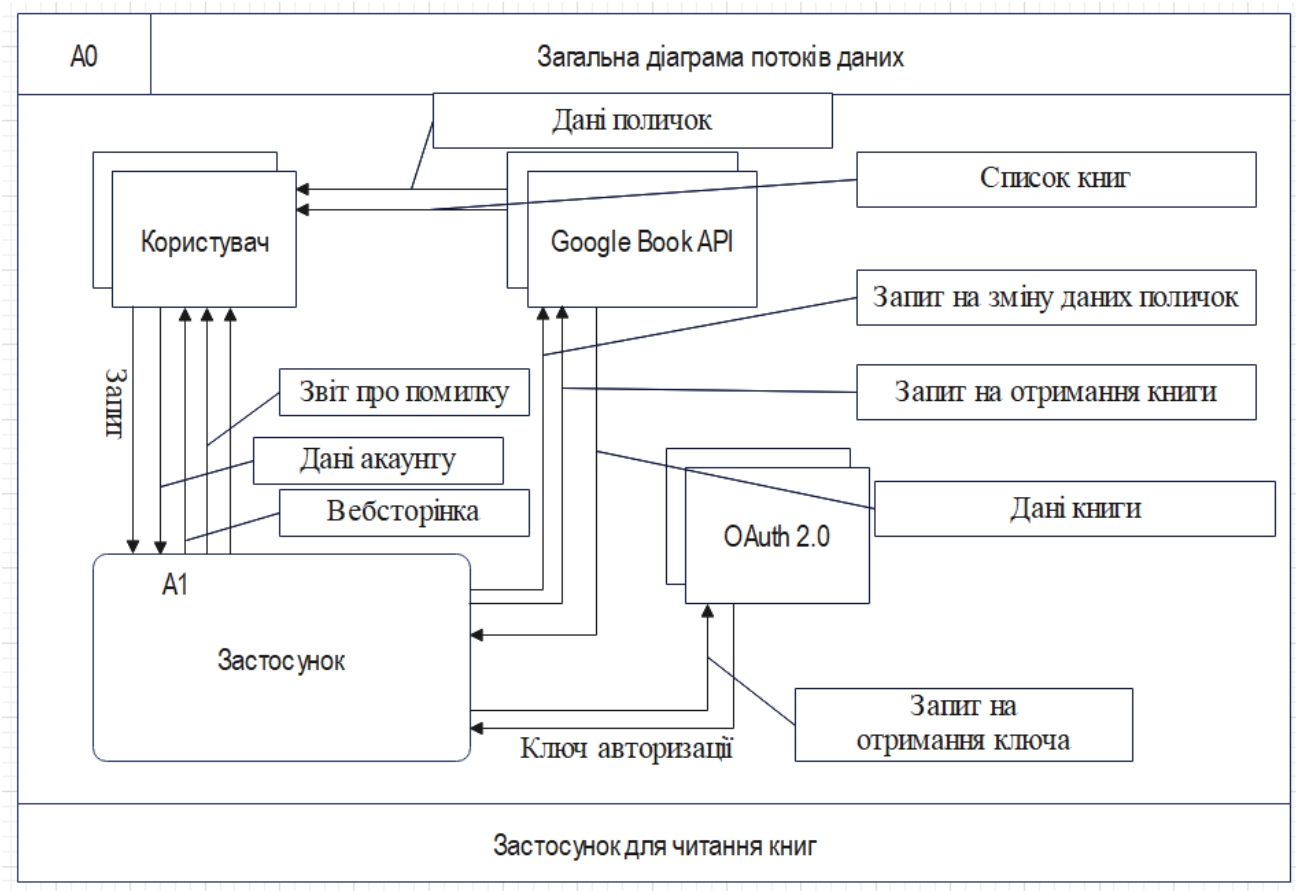


Рисунок 2.3 – Загальна діаграма потоків даних в застосунку

У застосунку, головною сутністю виступає «Користувач». Він може передавати дані власного акаунту, для підтвердження авторизації, що супроводжується і передачею запитів. Велика частина інформації залежить від зовнішніх систем, таких як OAuth 2.0 та Google Book API. В даному випадку, OAuth 2.0 слугує для створення унікального ключа доступу для функціоналу Google API. В свою чергу, інша система слугує як основний механізм, навколо якого побудована система. Якщо користувач робитиме запити на зміни у

структурі власних даних, то система повідомлятиме його сама про успішність виконання операції. В випадку, якщо запити стосуються отримання інформації з віддаленої бази даних, то вона передається прямо з серверу користувачу.

Велика частина шляхів передачі даних в застосунку, як і процеси, виділені в окремій діаграмі. Процеси, які відбуваються під час роботи застосунку, можна побачити на рисунку 2.4.

Рисунок 2.4 – Діаграма потоків даних в функція користувача

Дані акаунту завжди проходять процес автентифікації, що передбачає перевірку доступності виконання запиту. Запит, в свою чергу, може набувати одну з трьох форм: «Доступ до поличок», «Посилання», «Доступ до книг».

Робота з поличками завжди передбачає наявність ключа автентифікації, тому що зміни відбуватимуться саме в акаунті користувача. Функціонал даного типу запитів може включати як додавання книг до поличок, так і перенесення

між ними. Перенесення передбачає видалення книги з поточної полички і додавання в нову, так як Google API не передбачає функцій одноетапної зміни.

Робота з книгами передбачає отримання або списків книг за певними параметрами, так і отримання даних конкретного елементу. Дана інформація передбачається, як відкрита, тому не потребує проходження автентифікації.

Разом з даними про книгу, також користувачу будуть надходити посилання на купівлю книги, перегляд прев'ю і на читання книги. Дані процеси будуть передбачати відкриття посилань в браузері. Але, у випадках, коли отримання посилання не дійсне, користувача потрібно повідомити про це.

На основі визначених потоків даних, застосунок можна поділити на функціональні частини. Повний список розподілених функцій та зв'язків між модулями, зображено на рисунку нижче (рисунок 2.5).

Рисунок 2.5 – Функціональні компоненти застосунку

Для роботи з книгами, застосунок має мати інформаційні додатки. Коли з Google Book API буде надходити набір даних серіалізованих у формат JSON, перш за все, потрібно визначити потрібні для роботи властивості. Ці властивості будуть десеріалізовані і розміщені в екземпляри певних класів. Інформація про статус сторінок має повідомляти систему про те, з якою книжковою поличкою відбувається робота. Це дозволяє згрупувати і спростити процес звернення до різних частин API.

Відповідно до зв’язків моделі, найбільше до даного блоку буде звертатись система управління книгами. Уся обробка даних книг, відбуватиметься саме у цій системі. Це включатиме створення списків елементів, отримання конкретних екземплярів, формування описів та робота з дійсністю посилань.

Інформація отримується за допомогою модуля «Управління запитами». Будь-яка дія, що вимагає втручання Google Book API, супроводжується функціями даного модуля. Також, даний модуль перевірятиме, чи можливо виконати переадресацію по посиланню до Google Book, що стосується покупок, перегляду та передперегляду. Також, з системою запитів працюватиме і модуль «Управління поличками».

Для того, щоб отримувати доступ до функціоналу з особистими даними акаунту, передбачається окрема система авторизації. По-перше, коли користувач відвідуватиме застосунок, йому буде пропонуватись обрати Google-акаунт, з книгами і полицями якого відбуватиметься робота. При першому вході з кожного акаунту, користувачу буде пропонуватись підтвердити обробку особистих даних для безпеки.

По-друге, модуль авторизації відповідає за створення ключів, для підтвердження дій, які ще також називаються токенами. Дані ключі мають кожного разу передаватись до запитів роботи з поличками, навіть, якщо запити стосуються лише отримання списку книг з поличок. Це зумовлено тим, що в власних налаштуваннях, користувач може зробити полиці приватними, через що вільний доступ буде присікатись Google API, якщо відсутні дані підтвердження особистої інформації.

Для взаємодії користувача з функціоналом, буде застосована система управління відображеннями. Ключова особливість полягає в тому, що дані і запити прив’язуються саме до відображень. Зміна будь-якого відображення, змінить статус сторінки, який надаватиме інформацію про дані, які потрібно отримати. Також, застосунок матиме поділ на динамічні частини – фрагменти, які динамічно змінюватимуть контент сторінок і також залежатимуть від стану активності, в якій розміщені.

Архітектурний паттерн Model-View-View-Presenter буде вдалим вибором для розробки даного мобільного застосунку, через його спроможність спрощувати управління та відображення даних. Основною перевагою MVVP є розділення бізнес-логіки та логіки представлення, що дозволяє відділити логіку від сторінок. Структура даного патерну, добре лягає на структуру визначених компонентів і часто використовується саме в розробці мобільних застосунків.

У контексті мобільного застосунку для читання книг, де користувач може бачити списки книг, їх деталі, і читати обрані тексти, MVVP дозволяє розробникам незалежно оновлювати вигляд додатка без впливу на основну логіку або управління. Хоча при цьому, саме представлення викликає функціонал застосунку. Наприклад, якщо користувач спробує знайти якусь книгу через Google Book API, саме подія натискання на кнопку зможе викликати функції відправик запитів на сервер.

2.4 Проєктування модулів застосунку

У процесі проєктування мобільного застосунку для читання книг з Google Books була розроблена модульна система, що дозволяє ефективно розподіляти функціональні обов'язки і спрощує управління кодом. Архітектура системи була поділена на декілька ключових підсистем, що відповідають за різні аспекти функціональності додатку. Клас [37] – логічний опис чогось, шаблон, за допомогою якого можна створювати реальні екземпляри цього самого чогось. Іншими словами, це просто опис того, якими мають бути створені сутності: які властивості та методи повинні мати.

Перший модуль – це модуль формування ключів, який включає механізми аутентифікації та авторизації з використанням OAuth 2.0. Цей модуль гарантує безпеку користувацьких даних та взаємодію з вебсервісами Google, забезпечуючи стійкість системи до несанкціонованого доступу.

Другим важливим компонентом є формування активностей. Модуль відповідає за управління різними активностями в застосунку, які уможливлюють навігацію користувача та взаємодію з інтерфейсом. Кожна активність розроблена таким чином, щоб забезпечувати оптимальну відповідь на запити користувача, а також забезпечити логічний і інтуїтивно зрозумілий перехід між різними частинами застосунку.

Третій модуль, модуль формування фрагментів, включає компоненти, що управляють частинами користувацького інтерфейсу. Фрагменти служать для відображення інформації, такої як списки книг, деталі про книги, пошукові результати та інші. Вони спрощують розробку та дозволяють перевикористовувати інтерфейсні елементи в різних частинах застосунку, підвищуючи ефективність взаємодії з користувачем.

На зображенні 2.6 можна побачити декілька класів, які використовуються в мобільному додатку для читання книг з Google Books, такі як «BookDetails», «TokenCreator», «SearchedBook», і так далі.

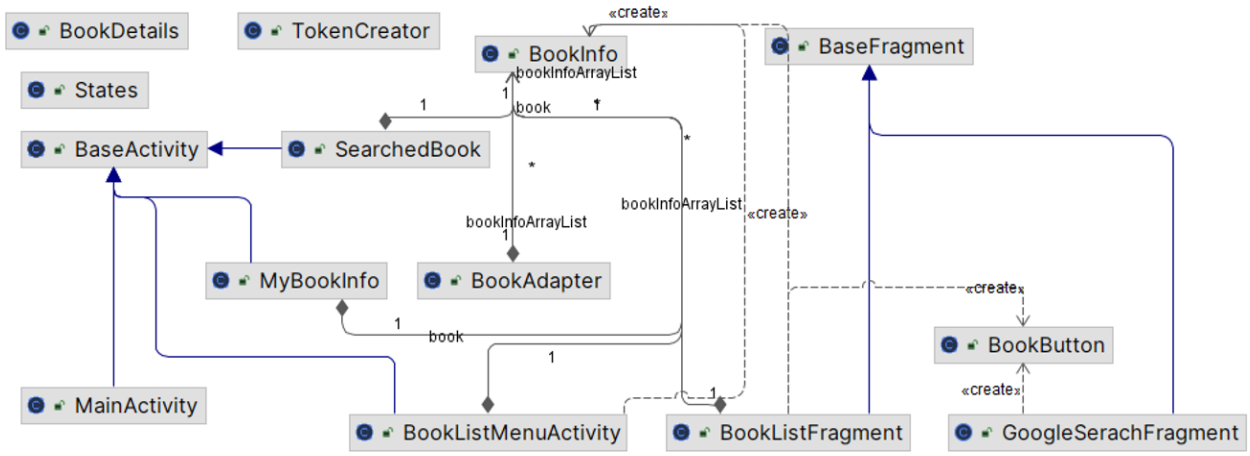


Рисунок 2.6 – Список класів інформаційної системи

На представленій діаграмі класів (рисунок 2.7) видно реалізацію системи для управління токенами OAuth 2.0 в мобільному застосунку для читання книг. Клас «TokenCreator» відповідає за створення та керування доступними токенами авторизації. Він містить методи, такі як «generateServerToken(Context)» для

генерації серверних токенів, «useAccessToken(Context, GoogleSignInAccount, DoSmthCallback)» для використання токенів доступу, а також «getExpressTime(int)» та «isTimeOver()», які використовуються для контролю часу дії токена.

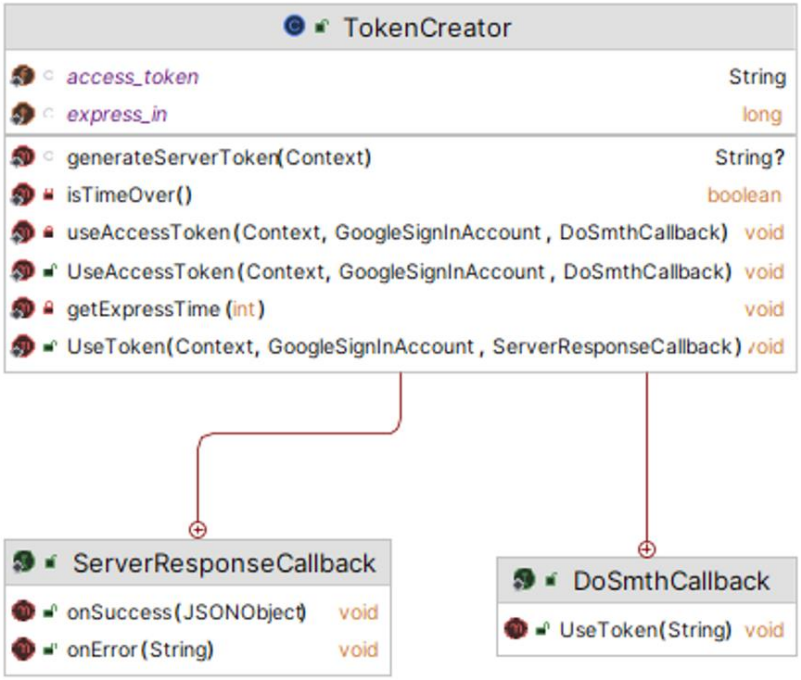


Рисунок 2.7 – Розкрита діаграма класів формування ключів для OAuth 2.0

Клас «TokenCreator» взаємодіє з «ServerResponseCallback» і «DoSmthCallback» для обробки відповідей сервера та виконання зворотних викликів після завершення аутентифікації. «ServerResponseCallback» має методи «onSuccess(JSONObject)» та «onError(String)», які обробляють успішні та невдалі відповіді відповідно. Ця взаємодія допомагає управляти авторизаційними даними ефективно, гарантуючи безпеку та належне керування сесіями користувачів у даному застосунку.

Діаграма класів активностей відображає організацію основних компонентів мобільного застосунку для читання книг (рисунок 2.8). Клас «BaseActivity» служить базою для інших активностей, надаючи загальні методи для зміни фрагментів (changeFragment) та управління меню

(changeOptionsMenu). Це дозволяє уникнути дублювання коду і забезпечує консистентність поведінки між різними частинами застосунку.

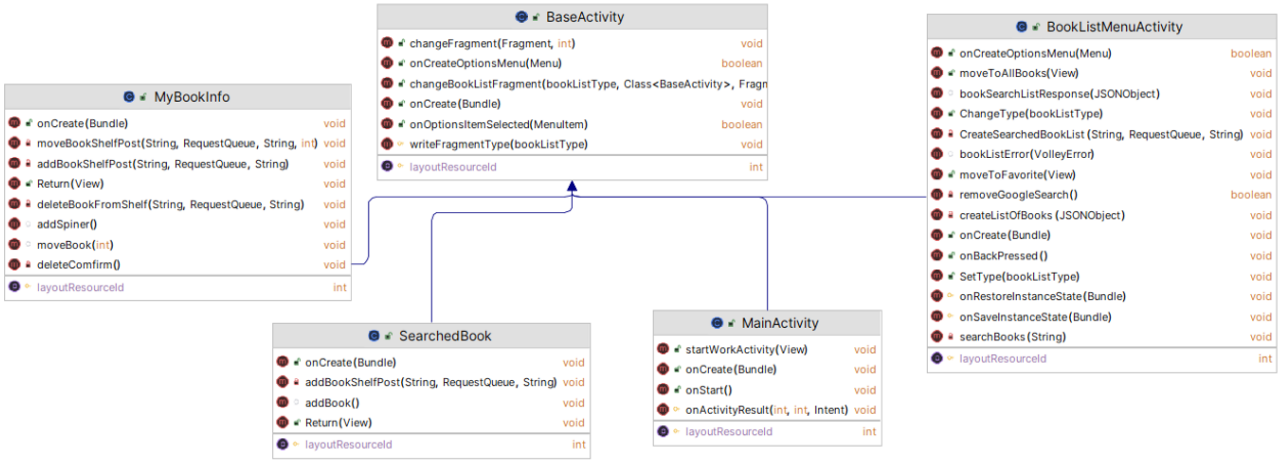


Рисунок 2.8 – Розкрита діаграма класів активностей

Клас «MainActivity» є точкою входу в застосунок, де відбувається ініціалізація та управління основними взаємодіями користувача. У «MainActivity», метод «startWorkActivity()» викликає іншу активність для виконання певних задач, наприклад, відкриття нової книги чи пошуку. Також є метод «onCreate()», який ініціює активність, встановлюючи всі необхідні налаштування при її старті. Клас «BookListMenuActivity» управляє списками книг, включаючи функціонал для перегляду всіх книг, улюблених, і книг, які користувач планує прочитати. Містить методи, які дозволяють керувати списками книг, такі як «moveToAllBooks()», що переносить користувача до перегляду всіх доступних книг, а «createListOfBooks()» створює список книг на основі отриманих даних.

Класи «MyBookInfo» та «SearchedBook» відповідають за відображення інформації про книги, включаючи деталі книги, які користувач вже додав у свою колекцію, та результати пошуку книг відповідно. Клас «MyBookInfo» включає метод «moveBookShelfPost()», який використовується для переміщення книги на іншу полицю у віртуальній бібліотеці. Також є метод «deleteBookFromShelf()», який дозволяє користувачам видалити книгу зі своєї колекції.

Ці класи забезпечують взаємодію з сервером для отримання та відправлення даних, що включає додавання книг до різних списків або видалення їх з колекції.

На зображенні діаграми класів фрагментів представлено структуру управління різними частинами інтерфейсу користувача у мобільному застосунку для читання книг (рисунок 2.9).

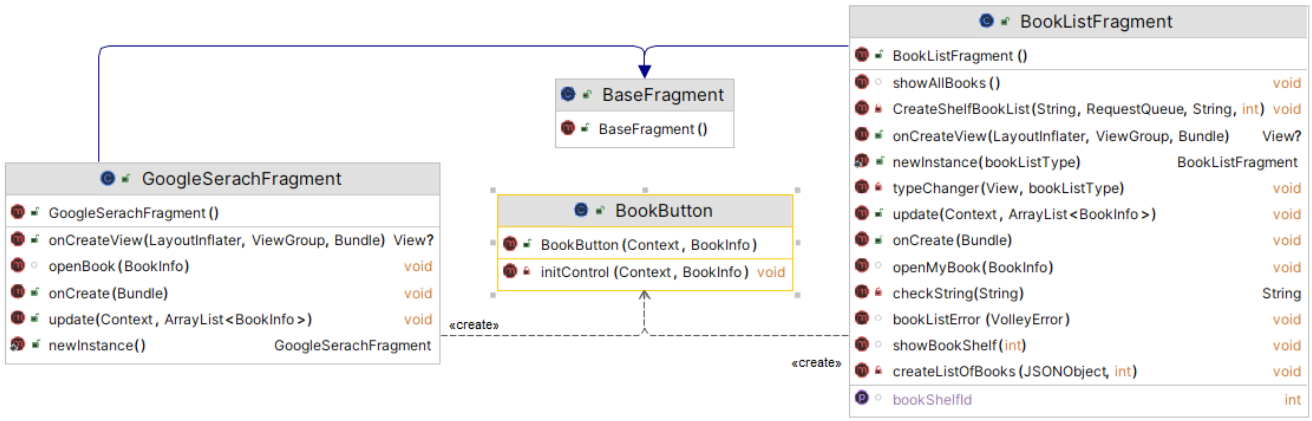


Рисунок 2.9 – Розкрита діаграма класів фрагментів

«BaseFragment» служить як базовий клас для інших фрагментів, забезпечуючи спільну функціональність, яку можуть використовувати інші фрагменти. «GoogleSearchFragment» відповідає за реалізацію пошукових запитів та відображення результатів пошуку книг, використовуючи Google API. Він містить методи для обробки пошукових запитів та оновлення відображення списку книг за допомогою відповідей, які отримуються з сервера. Метод «onCreateView» відповідає за створення вигляду фрагменту, тоді як «update» оновлює список книг, базуючись на пошукових запитах користувачів. Функція «openBook» дозволяє відкрити детальну інформацію про вибрану книгу.

«BookButton» - це клас, який управляє кнопками у фрагментах, дозволяючи ініціалізацію та обробку кліків з відповідною інформацією про книги, які передаються як параметри.

«BookListFragment» відповідає за відображення списку книг. Цей фрагмент має функції для управління різними видами списків книг, такими як всі книги, улюблені книги, а також книги, що плануються до читання. Даний

фрагмент обробляє динамічну зміну типів списків і відображає відповідні книги залежно від вибраного типу. Методи, такі як «showAllBooks» та «CreateShelfBookList», організовують показ усіх доступних книг або створення спеціалізованих списків книг. Інші функції, такі як «bookListError», обробляють помилки, які можуть виникнути під час запиту до сервера.

Повна діаграма класів представлена на рисунку Б.1 в додатку Б. Ця структура дозволяє забезпечити модульний та гнучкий підхід до розробки інтерфейсу користувача, де кожен фрагмент може бути оптимізований для виконання специфічних задач у контексті загальної системи навігації та взаємодії в застосунку.

2.5 Проєктування інтерфейсу користувача

Дизайн інтерфейсу користувача (UI) – це процес, який дизайнери використовують для створення інтерфейсів у програмному забезпеченні чи комп'ютеризованих пристроях, зосереджуючись на зовнішньому вигляді чи стилі. Дизайнери прагнуть створювати інтерфейси, які користувачі вважають простими у використанні та приємними. Дизайн інтерфейсу користувача відноситься до графічних інтерфейсів користувача та інших форм.

Інтерфейс користувача дозволяє кінцевому користувачеві взаємодіяти з апаратним чи програмним забезпеченням пристрою або керувати ним. Наприклад, екрани, миші, клавіатури, кнопки, піктограми програм і текстові поля складають інтерфейс користувача.

Хороший інтерфейс користувача повинен гарантувати, що користувач докладає мінімум зусиль для досягнення максимальних результатів. Він має бути інтуїтивно зрозумілим і привабливим для людських почуттів і працювати ефективно. Все це відбувається через користувальницький інтерфейс і дизайн взаємодії з користувачем.

Інтерфейс повинен мати наступні атрибути, щоб бути зручним, ефективним та інтуїтивно зрозумілим.

Чіткість. Інтерфейс користувача повинен дозволяти користувачам взаємодіяти з програмою. Потрібно уникати всього, що може заплутати або відволікти взаємодію між користувачем та інтерфейсом. Метафори інтерфейсу (візуальні підказки) мають бути безпомилковими, наприклад, папка чи закладка.

Конструкція системи має бути простою у використанні. Хороший інтерфейс повинен дозволяти користувачам легко виконувати ті завдання, які вони хочуть виконувати.

Візуальне задоволення. Інтерфейс має бути естетично привабливим і привабливим настільки, щоб привернути увагу користувачів. Інтерфейс має бути простим у навігації та дозволяти користувачам знаходити потрібну інформацію.

Надійні відомості про компанію та відповідні відомості про безпеку мають бути видимими для користувачів, щоб підвищити та зміцнити довіру.

Постійне використання інтерфейсу, ймовірно, призведе до того, що користувач виробить звички. Саме тому потрібно створювати інтерфейси, які не дозволяють звичкам створювати проблеми для користувача. Можна реалізувати інтерфейс, розуміючи звички користувачів, щоб заохочувати застосовувати кращі функції породжуючі звички. Варто остерігатися несвідомих припущень щодо того, як користувач взаємодіятиме з інтерфейсом.

Вибір кольорової схеми для створення інтерфейсу відіграє критичну роль в забезпеченні ефективності та естетичності користувацького досвіду. Кольори мають здатність впливати на сприйняття та емоції користувачів, тому правильний вибір кольорової палітри може значно підвищити залученість та задоволення користувачів. Наприклад, теплі кольори можуть викликати відчуття комфорту, тоді як холодні тони сприяють концентрації та зосередженості.

Крім того, кольори допомагають організувати інформацію на екрані, підкреслюючи важливі елементи та спрямовуючи користувача до ключових дій. Яскраві та контрастні кольори часто використовуються для кнопок дій або

сповіщень, щоб привернути увагу користувача, тоді як більш нейтральні тони можуть бути використані для фону або менш важливих компонентів.

Для проєкту було обрано просту кольорову гаму, яка складається з фіолетового та білого кольорів. Цей вибір не випадковий, адже фіолетовий колір часто асоціюється з креативністю, мудрістю та спокоєм, що може сприяти концентрації та зануренню в читання. Використання білого кольору додає відчуття простору та чистоти, роблячи інтерфейс свіжим і легким на вигляд.

Така кольорова схема дозволяє візуально виділити важливі елементи інтерфейсу, такі як кнопки дії чи посилання, не перевантажуючи користувача зайвими візуальними стимулами. Це робить навігацію інтуїтивно зрозумілою та приємною, що є ключовим фактором для забезпечення високої залученості користувачів. Вибір такої кольорової палітри також підкреслює сучасний дизайн застосунку та може сприяти його впізнаваності серед конкурентів.

Для того, щоб створити робочий інтерфейс для програми спершу потрібно розробити динамічний прототип його дизайну. Динамічний прототип – це представлення цифрового продукту, яке імітує його основні функції та взаємодію з користувачем. Це візуальна та інтерактивна демонстрація користувальницького інтерфейсу (UI) продукту та дизайну взаємодії з користувачем (UX).

Інтерактивні прототипи служать цінним інструментом для перевірки дизайнерських рішень і збору відгуків користувачів. Тестуючи прототип із реальними користувачами, дизайнери можуть спостерігати, як користувачі переміщуються продуктом, визначати проблемні точки та збирати уявлення про те, що працює, а що потребує вдосконалення.

До розгляду розробки інтерфейсу буде братися саме такий прототип.

Зображення демонструє стартову сторінку мобільного застосунку для читання книг, яка використовує простий та ефективний інтерфейс (Додаток В.1). Фіолетовий фон з білою іконкою книги в центрі створює яскравий та виразний візуальний ефект, сприятливий для користувацького сприйняття. Назва застосунку «SHELF» розташована над іконою, що допомагає користувачам

швидко ідентифікувати призначення додатку. Фраза «YOUR PERSONAL BOOK APP» подальше уточнює функціональність додатку як персональної бібліотеки.

Кнопка «TAP TO START» яскраво виділена на тлі, що спрямовує користувача до наступної дії. Використання контрастного кольору для кнопки входу підвищує її видимість і таким чином забезпечує кращий користувацький досвід. В цілому, дизайн чистий і не перевантажений зайвими елементами, що робить інтерфейс дружнім та легким для використання.

Кольорова палітра та стиль дизайну відображають сучасні тенденції в оформленні мобільних застосунків, спрямовані на забезпечення приємного візуального сприйняття та ефективності взаємодії.

Наступний та дуже важливий елемент інтерфейсу, що потребує окремого розгляду, є головним та постійним меню на екрані.

На зображенні показано навігаційний бар, який є частиною інтерфейсу мобільного застосунку для читання книг з Google Books (Додаток В.2). Цей бар є постійним елементом на більшості сторінок застосунку, що забезпечує легкий доступ до основних функцій. Бар має три кнопки: домашня сторінка, улюблені книги та інформація про програму, кожна з яких має іконку, що інтуїтивно відповідає її функції.

Дизайн меню використовує фіолетовий колір, що відповідає загальному стилю застосунку, сприяючи візуальній уніфікації. Використання іконок допомагає користувачам швидко розпізнавати функції без потреби читати текст, що покращує їхній досвід користування, особливо коли вони переглядають різні розділи додатку. Простота і інтуїтивна зрозумілість цього навігаційного бару забезпечують легкість переходів між основними секціями звстосунку, сприяючи плавній та ефективній взаємодії з програмою.

Наступна сторінка мобільного застосунку для читання книг демонструє частину, де користувачі можуть переглядати список книг, які вони зараз читають. Інтерфейс організований таким чином, що книги відображаються у формі сітки з чітко виділеними обкладинками та назвами книг під кожною з них, що дозволяє користувачам легко ідентифікувати та вибирати потрібні твори.

Обкладинки книг будуть замінені після вводу програми в роботу, на місце картинки в прототипі будуть підставлятися справжні обкладинки книг, що дозволить ще легше зрозуміти користувачу, що той бачить (Додаток В.3).

Ніжний фіолетовий колір фону та елементів інтерфейсу підтримує єдину кольорову схему, яка відповідає загальному дизайну застосунку. Це не тільки забезпечує візуальну привабливість, але й сприяє легшому сприйняттю візуальної інформації користувачем. Верхня частина екрану містить заголовок «I am reading», що одразу інформує користувача про контекст відображеної інформації, допомагаючи зорієнтуватися в навігації застосунку. В цілому, такий дизайн сприяє кращій організації користувацького простору та зручності взаємодії з додатком.

Це лише одна з трьох ключових категорій, які допомагають користувачам організувати свої читацькі переваги. Дві інші категорії, «In the plans» та «It has been read», також представлені в застосунку та доступні через випадаюче меню, що забезпечує користувачам легкий доступ до книг, які вони планують прочитати, або тих, які вже прочитані. Кожна з цих категорій має схожий дизайн відображення книг, що забезпечує консистентність і зручність користування застосунком. Такий підхід допомагає користувачам ефективно керувати своєю бібліотекою і сприяє кращій організації читацького процесу.

На зображенні можна бачити випадаюче меню мобільного дзастосунку (Додаток В.4) для читання книг, яке пропонує користувачам три основні категорії для організації їхнього читацького досвіду: «I am reading» (Я читаю), «In the plans» (В планах), та «It has been read» (Прочитано). Це дозволяє користувачам легко управляти своєю колекцією книг, розділяючи їх на книги, які вони зараз читають, планують прочитати, або вже завершили читати. Такий підхід не тільки спрощує навігацію по застосунку, але й допомагає користувачам підтримувати порядок у їхній особистій бібліотеці та відслідковувати свій прогрес у читанні. Це меню легко доступне і забезпечує швидкий перехід між різними категоріями, роблячи досвід користування застосунком більш приємним та ефективним. Для комфорту користувача дане випадаюче меню було

розміщено п верхній правий кут екрану, так як різні дослідження показують, що від 70 % до 90 % людей на нашій планеті є прашами і тільки 9-10 % лівші [38].

Це зображення демонструє сторінку деталей книги в мобільному застосунку для читання. Вгорі сторінки розташований заголовок «ABOUT BOOK», який чітко вказує на контент сторінки. Нижче заголовка знаходиться біле тло, у якому відображатиметься обкладинка книги, текстовий блок, який міститиме загальний опис книги (Додаток В.5).

Під текстовим блоком розташований інтерактивний елемент «STATUS», який, ймовірно, дозволяє користувачам змінювати статус прочитаної книги. Внизу сторінки знаходиться кнопка «READ», яка дозволяє відкрити книгу для читання. Дизайн сторінки чистий і мінімалістичний, з використанням світлих тонів, що допомагає уникнути візуального перевантаження і забезпечує користувачу комфортне сприйняття інформації. Використання жирних та великих шрифтів для важливих елементів навігації та інтерфейсу також сприяє кращій читабельності.

На зображенні представлені чотири варіації одного і того ж тексту в інтерфейсі мобільного застосунку для читання книг (Додаток В.6). Ці варіації демонструють, як користувач може налаштовувати кольорову схему інтерфейсу для забезпечення комфорту читання. Інтерфейс дозволяє змінювати колір тексту та фону, що важливо для адаптації до різних умов освітлення або персональних переваг. Зміна кольору фону та тексту не лише підвищує естетичну привабливість додатку, але й сприяє зниженню втоми очей під час тривалого читання через застосунок.

На зображенні показано інтерфейс налаштувань мобільного додатку для читання книг, що дозволяє користувачам кастомізувати кольорову схему та колір шрифту для свого читацького інтерфейсу (Додаток В.7). Це меню активується натисканням на іконку у вигляді шестерні, розташовану в лівому верхньому кутку екрану, що забезпечує легкий доступ до налаштувань. Користувачам пропонуються три варіанти кольору тла: білий, чорний та сепія, а також кілька варіантів кольорів шрифту, включно з синім, жовтим, зеленим, червоним,

чорним та сірим. Такі опції дозволяють користувачам налаштувати читацький простір на свій смак та в залежності від зовнішніх умов освітлення, сприяючи комфортнішому читанню та зменшенню втоми очей. Також присутня сторінка з можливістю обирати улюблені кники, яка дозволяє користувачам зберігати обрані твори для швидкого доступу. Ця сторінка виконана в тому ж стилі, що і решта елементів інтерфейсу, забезпечуючи єдність дизайну та сприяючи зручнішому взаємодії. Користувачі можуть додавати книги до списку улюблених, обравши відповідну опцію у випливаючому меню на сторінці з інформацією про обрану книгу.

У мобільному застосунку для читання книг з Google Books також існує пошукова система, яка дозволяє користувачам з легкістю знаходити книги за ключовими словами, авторами або назвами. Ця система інтегрована таким чином, що вона вписується у загальний дизайн інтерфейсу, зберігаючи єдність та стиль застосунку. Окрім того, існує окрема сторінка з інформацією про застосунок, де користувачі можуть отримати детальніші відомості про функції та можливості застосунку, а також про розробника програми та дизайну користувацького інтерфейсу.

2.6 Короткі висновки до розділу

У процесі розробки мобільного застосунку для читання книг важливим було вибрати технології, що забезпечують оптимальне співвідношення продуктивності, сумісності та зручності в розробці. Використання Android Studio та інтеграція з Google API виявилися найбільш вдалими рішеннями. Інтеграція з Google API відкрила доступ до величезної бібліотеки книг і забезпечила зручні інструменти для управління користувацькими даними та авторизації, підвищуючи функціональність застосунку.

Розробка чіткої та модульної архітектури дозволила забезпечити гнучкість системи та її масштабованість. Завдяки цьому застосунок може легко

адаптуватися до мінливих вимог ринку і користувацьких вподобань. Модульний підхід у проєктуванні допоміг в оптимізації процесу розробки та сприяв легшому тестуванню та управлінню кодом. Кожен модуль був спроєктований з можливістю незалежного використання та оновлення.

Основна увага була приділена розробці інтуїтивно зрозумілого та зручного інтерфейсу, який підтримує ефективне взаємодію з користувачем. Дизайн інтерфейсу зосереджений на забезпеченні комфорту читання і мінімалістичному візуальному сприйнятті.

Майбутній розвиток проєкту включає додавання нових функцій та покращення існуючих на основі зворотного зв'язку від користувачів. Планується розширення функціональності, включаючи підтримку нових мов, інтеграцію з іншими книжковими сервісами та платформами, додавання функцій соціальної взаємодії, таких як обмін рекомендаціями та рецензіями між користувачами.

Практична значимість розробленого програмного забезпечення полягає в тому, що воно надає користувачам зручний доступ до великої бібліотеки літератури, дозволяючи зберігати, читати та організовувати свої книги в одному місці. Застосунок значно спрощує процес пошуку і вибору книг, забезпечуючи персоналізовані рекомендації на основі індивідуальних уподобань. Це сприяє підвищенню задоволеності користувачів і ефективності використання часу. Крім того, можливість читання офлайн дозволяє користувачам мати доступ до своїх улюблених книг у будь-який час і в будь-якому місці, незалежно від наявності інтернет-з'єднання.

Застосунок також сприятиме популяризації читання серед широкої аудиторії, надаючи легкий доступ до різноманітної літератури. Це особливо важливо в контексті сучасних тенденцій цифрової трансформації, де зручність і доступність є ключовими факторами успіху.

У підсумку, розроблений мобільний застосунок для читання книг з Google Books має великий потенціал для подальшого розвитку і вдосконалення, забезпечуючи високу якість користувацького досвіду та задоволення потреб сучасних читачів.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Програмна реалізація модулів

У цьому розділі буде зосереджено увагу на деталях втілення основних компонентів мобільного застосунку для читання книг з Google Books. Розділ охоплює процес технічної реалізації різних модулів програмного забезпечення, включаючи управління користувацьким інтерфейсом, обробку даних та інтеграцію з зовнішніми API.

Застосунок не завжди складається з одного екрану. Потрібно реалізувати перехід від одної до іншої. Спочатку потрібно використати елементи та інші різні ресурси для створення сторінок. Для кожного проєкту використовуються ресурси, структура ресурсів у проєкті Android включає наступні папки:

- Anim – зберігає анімації.
- Drawable – містить зображення та інші графічні ресурси.
- Font – включає шрифти.
- Layout – папка, що зберігає файли розмітки, дані файли визначають візуальний інтерфейс.
- Menu – містить XML файли, що описують меню.
- Mipmap – зберігає іконки застосунку.
- Values – включає ресурси, такі як рядки, кольори, стилі.
- Xml – містить різноманітні XML налаштування.

У цих папках містяться усі наявні ресурси для хорошої візуалізації проєкту. Кожна папка містить свої фрагменти та картинки. Всі папки підписані згідно того, що у них міститься тому можливо інтуїтивно зрозуміти призначення даних папок.

Код програми міститься у Java-файлах. Ця частина містить в собі практичний код для вступної сторінки застосунку, для головної сторінки та кількох додаткових. Там використовується передача даних при переході між

сторінками, запускаються анімації, виконуються переходи на інші сторінки, використання ресурсів з попередньо описаних папок та інше.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void startWorkActivity(View v) {
        Intent intent = new Intent(this, BookListMenuActivity.class);
        intent.putExtra("type", "all");
        startActivity(intent);
        overridePendingTransition(R.anim.swipe_to_up, R.anim.no_animation);
    }}
```

Цей невеликий фрагмент коду представляє собою основу для Android-застосунку, розробленого на мові програмування Java з використанням Android SDK. У цьому коді визначений клас «MainActivity», який є основною активністю програми. «MainActivity» розширює клас «AppCompatActivity», що дозволяє використовувати сучасні можливості бібліотеки підтримки для створення інтерфейсу користувача.

Метод «onCreate()» є одним з життєвих циклів активності і викликається при створенні цієї активності. У ньому встановлюється вміст головного екрану за допомогою «setContentView(R.layout.activity_main)», що означає завантаження макету з файлу «activity_main.xml» для відображення на екрані мобільного пристрою.

Метод «startWorkActivity(View v)» викликається при натисканні на певний елемент інтерфейсу (наприклад, кнопку), оскільки він має атрибут «android:onClick="startWorkActivity"» в XML-файлі макету. У цьому методі створюється новий «Intent», який вказує на перехід до «BookListMenuActivity».

За допомогою методу «putExtra()» в «Intent» додається додаткова інформація про тип (type) книги, яка буде передана в «BookListMenuActivity». Після цього запускається нова активність за допомогою «startActivity(intent)», а також встановлюється анімація переходу між активностями за допомогою

«overridePendingTransition(R.anim.swipe_to_up, R.anim.no_animation)», що покращує візуальний ефект переходу між екранами програми.

Клас «BaseActivity» є абстрактним базовим класом для активностей Android, який розширює функціональність класу «AppCompatActivity». Основна мета цього класу полягає в абстрагуванні загальної логіки відображення макету екрану від конкретної реалізації. Цей клас містить абстрактний метод «getLayoutResourceId()», який необхідно реалізувати в підкласах.

Метод «onCreate()» встановлює зміст активності, використовуючи макет, що повертається з методу «getLayoutResourceId()». Використання «BaseActivity» дозволяє зменшити дублювання коду та забезпечити однаковий підхід до відображення UI для різних активностей у застосунку, що сприяє підтримці, розширенню і зручності управління кодом.

Управління меню реалізується через методи «onCreateOptionsMenu» і «onOptionsItemSelected». «onCreateOptionsMenu» ініціалізує меню активності, надуваючи його з XML-файла. Метод «onOptionsItemSelected» обробляє вибір пунктів меню, перенаправляючи користувача до активності «BookListMenuActivity» з додатковими параметрами, що визначають тип книжкового списку, який слід відобразити, що забезпечує функціональність переходу між різними частинами програмного забезпечення.

«BookListFragment», який є підкласом «Fragment» і відповідає за відображення списку книг в залежності від переданого типу. Даний фрагмент містить методи для створення нового екземпляру фрагмента з переданим параметром і для відображення відповідного тексту заголовку відповідно до цього параметра. У методі «newInstance(String param1)» створюється новий екземпляр «BookListFragment» з переданим параметром типу книги.

Метод «onCreate(Bundle savedInstanceState)» викликається при створенні фрагмента. В цьому методі перевіряється наявність переданих аргументів, і якщо вони є, параметр «mParam1» ініціалізується значенням параметра «type».

Метод «onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)» викликається для створення і відображення інтерфейсу

фрагмента. У цьому методі здійснюється інфляція макету фрагмента з використанням «LayoutInflater», і потім метод «typeChanger(view, mParam1)» викликається для встановлення відповідного тексту заголовку залежно від значення «mParam1» (типу книги), переданого в аргументах фрагмента. Метод «typeChanger(View view, String param)» встановлює текст заголовку (TextView з ідентифікатором headerText) на основі переданого параметра «param». Залежно від значення «param», текст заголовку повинен змінюватися на відповідну назву типу книги.

Тут демонструються основні принципи створення фрагментів в Android, включаючи передачу даних у фрагмент, ініціалізацію інтерфейсу у методі «onCreateView», а також використання методів життєвого циклу фрагментів для обробки переданих даних та відображення відповідного інтерфейсу.

«BookListMenuActivity», який є підкласом «BaseActivity», відповідає за відображення списку книг у відповідності до вибраного типу. Дія класу зосереджена на обробці подій вибору пунктів меню і перемиканні між різними типами списку книг за допомогою фрагментів.

У методі «onCreate(Bundle savedInstanceState)» спочатку викликається метод батьківського класу для налаштування вмісту активності з використанням відповідного макету. Потім перевіряється наявність переданих даних через інтент. Якщо дані присутні, з параметрів витягується тип книги (type) і викликається метод «ChangeType(type)» для встановлення відповідного типу списку книг. Якщо дані відсутні, за замовчуванням встановлюється тип «all».

Метод «onCreateOptionsMenu(Menu menu)» викликається для створення меню у тулбарі (або панелі дій) активності, але не має власної реалізації і просто повертає результат батьківського методу.

Метод «onOptionsItemSelected(MenuItem item)» відповідає за обробку вибору пунктів меню. Залежно від ідентифікатора вибраного пункту меню встановлюється відповідний тип книги (type), і потім викликається метод «ChangeType(type)» для зміни типу списку книг.

Методи «moveToFavorite(View v)» і «moveToAllBooks(View v)» викликаються при натисканні відповідних кнопок у графічному інтерфейсі активності. Вони викликають метод «ChangeType(type)» з відповідним типом книги («favorite» або «all»). Метод «ChangeType(String type)» відповідає за зміну типу списку книг. Він створює новий об'єкт «FragmentTransaction» для взаємодії з фрагментами, задає анімацію переходу між фрагментами, створює новий екземпляр «BookListFragment» з переданим типом книги, замінює поточний фрагмент на новий і додає транзакцію до стеку, щоб забезпечити можливість повернення назад до попереднього стану.

Всього існує два режими – портретний і альбомний. На більшості телефонів використовується за умовчанням портретний режим. Альбомний режим відомий нам по звичайним широкоформатним моніторам.

В коді, під час запуску програми, перевіряється, чи були внесені зміни в збереження стану програми. При першому запуску, стан збережений не буває, саме тому, там де список книг буде відкриватись фрагмент «All books». Якщо користувач переходить на інший фрагмент, наприклад «Favorite», після чого змінює орієнтацію екрану, то в стан зберігається інформація, що це вже не перший перезапуск цієї активності.

Якщо, програма бачить, що стан збережено, то при зміні орієнтації, вона буде брати фрагмент, який був до перезапуску активності, наприклад «Favorite» і виводити уже його.

Для реалізації випадаючого меню, що з’являється при натисканні на вертикальні точки потрібно зробити кілька дій. Для початку створюється ресурс з відповідною назвою, щоб меню відображалось у активності, потрібно вказати ресурс конкретного меню у відповідній функції. В даному застосунку, відображення меню керується за допомогою прапорців налаштування навігаційних елементів. Саме тому код меню винесений в «BasicActivity», що є батьківською для всіх елементів програми.

XML-файл містить кілька елементів меню, кожен з яких представляє пункт, який користувач може вибрати. Кожен елемент <item> вказує на окрему

категорію книг, яку можна переглядати в застосунку. Атрибути кожного елемента включають «android:id», який використовується для ідентифікації пункту в коді Java, «android:orderInCategory», що визначає порядок пунктів у меню, та «android:title», який вказує текст, що буде відображатися на кнопці меню. Наприклад, пункт меню «IN THE PLANS» допомагає користувачам перейти до перегляду книг, які вони планують прочитати.

Кожного разу, коли натискається будь-який пункт меню, потрібно знайти його ідентифікатор і на основі цього виконати певні дії. Для початку програма встановлює тип фрагменту, на який потрібно перейти користувачеві. Усі типи, для коректності, були записані, як значення класу типу «enum» – «States», що містить перерахування усіх доступних станів. Після отримання стану, виконується один з декількох варіантів коду. У першому випадку, якщо користувач знаходиться на сторінці з фрагментами, що відповідають спискам книг, один фрагмент просто заміниться на інший. В іншому випадку, фрагмент заміниться разом з активністю.

У даній роботі буде реалізоване та використане випадаюче меню типу Spinner. Даний тип меню дозволяє зручно, за допомогою XML-коду, створювати функціональні випадаючі меню.

```
public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
    int bookshelfId = -1;
    switch (position) {
        case 0:
            bookshelfId = -1;
            break;
        case 1:
            bookshelfId = 2;
            break;
        case 2:
            bookshelfId = 0;
            break;
        case 3:
            bookshelfId = 3;
            break;
        case 4:
            bookshelfId = 4;
            break;}
    if (bookshelfId != -1) {
        moveBook(bookshelfId);
        Toast.makeText(getApplicationContext(), "BOOK MOVED",
Toast.LENGTH_SHORT).show();}}
```

Для того, щоб меню отримувало значення, для цього в коді потрібно створити адаптер, який при підключенні до Spinner-меню буде встановлювати можливі значення. Також, за допомогою Java-коду, для цього меню потрібно підключити прослуховувач подій, який на основі обраних дій буде виконувати певну функцію. В даному випадку, до Google Book API надсилається запит, який видаляє книгу з старої полиці та переміщує її на нову.

Перед надсиланням запиту, генерується ключ доступу до акаунту, який разом з запитами надсилається до бібліотеки. надсиланян відбувається за допомогою фреймворку Volley.

Цей код є обробником подій для вибору елемента в «AdapterView» (наприклад, в списку або спадному меню). Він присвоює ідентифікатор полиці для книги залежно від вибору користувача та викликає метод «moveBook()», якщо вибрано дійсний ідентифікатор. Якщо книгу переміщено, відображається відповідне сповіщення (toast).

Для реалізації Toast-запитів потрібно створити об'єкт Toast та передати у нього інформацію про повідомлення, яке мусить бути виведено на екран. У цьому застосунку, було вирішено використовувати Toast-повідомлення одразу у двох місцях. По-перше, при аутентифікації користувача, застосунок буде повідомляти користувача про те, чи вдалося програмі підключитись до його Google-акаунту. По-друге, повідомлення про переміщення полиці буде виводитись під час переміщення книги, що дозволить зробити інтерфейс інтуїтивно зрозуміліший майбутнім користувачам.

При розробці програмного продукту, виникла ситуація, при якій деякі функції потребують додаткового підтвердження дії, перед її виконанням. Це стосується пункту видалення книги з списку усіх обраних раніше. Для цього, потрібно вивести вікно, в якому користувачу буде задаватись питання про підтвердження дії видалення.

Гарним варіантом реалізації даної функції є використання класу «AlertDialog». Екземпляри даного класу менше піддаються змінам, що пов’язані з декоруванням, але надають різні варіанти власного використання, включаючи

потрібний для даного застосунку – з двома кнопками. Враховуючи низьку складність даного компонента, а також його лаконічність вигляду, було вирішено обрати саме цей спосіб.

Перед відображення вікна, створюється екземпляр класу «Builder», який є вкладеним класом у потрібний для проєкту тип діалогу. Наступним кроком, є внесення змін у створений об’єкт. Використовуючи послідовні методи, відповідно було внесено заголовок, тіло повідомлення, подія підтвердеження дії та відмови від продовження.

Приватний метод «deleteComfirm()» створює діалогове вікно підтвердження з питанням про видалення книги з полиці. У діалоговому вікні користувач може обрати "Так" або "Ні" щодо видалення книги.

Даний компонент містить заголовок з назвою дії, текст, що запитує про коректність дій, а також дві кнопки, що позначають відповідь на відображене питання. У першому рядку методу створюється новий екземпляр «AlertDialog.Builder», який пов'язаний з поточним контекстом (this). За допомогою цього будівельника встановлюються заголовок діалогу («Remove book») та текст повідомлення («Do you want to remove the book from the shelf?»).

Після заповнення усіх необхідних елементів, використовуючи Builder створюється екземпляр класу «AlertDialog», який виводиться користувачеві.

При натисканні кнопки відміни дії, не виконується ніяких дій, а вікно з повідомленням закривається.

При натисканні кнопки підтвердження, виконується функція видалення книги з полиці. Для цього створюється запит то Google Book API у вигляді рядка. Використовуючи бібліотеку Volley, створюється асинхронний запит, який розміщується в чергу. На основі раніше введеного акаунту користувача, створюється ключ доступу до функцій API. Створений ключ, за допомогою колбеків, надсилається разом з запитом. При отриманні відповіді, формується результат, який прибирає видалену книгу з відображення та перенаправляє користувача до сторінки перегляду всіх книг.

У коді, окрім ініціалізації вибраного акаунта, також відбувається надання прав доступу до потрібних функцій Google Books. При першому вході з конкретного акаунту, користувачу надається запит на дозвіл використання відповідних даних. Це забезпечується наступним лістингом коду:

```
public void onStart() {
    GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken("410211102598-
ug99uocplb3m1f45q5na5p4ia1eaig02.apps.googleusercontent.com")
            .requestScopes(new Scope("https://www.googleapis.com/auth/books"))
            .requestServerAuthCode("410211102598-
ug99uocplb3m1f45q5na5p4ia1eaig02.apps.googleusercontent.com")
            .requestEmail().build();

    GoogleSignInClient gsc = GoogleSignIn.getClient(this, gso);
    gsc.signOut();
    gsc = GoogleSignIn.getClient(this, gso);
    Intent signInIntetent = gsc.getSignInIntent();
    startActivityForResult(signInIntetent, 1000);

    super.onStart();}
```

Як можна побачити, визначається доступ до Google Book API, запитується токен авторизації, які потрібний для використання функціоналу реалізованого в наступних модулях, а також авторизація відбувається на основі поштової скриньки користувача.

При переході до інших модулів, програма використовує функцію автоматичної авторизації на основі останнього обраного акаунту, що дозволило зменшити кількість запитів з пристрою користувачів. Дана функція надається бібліотекою Google API, підключеною через офіційне посилання.

Існує можливість вибору кожного разу нового акаунта, що дозволяє зручно користуватись на одному пристрої функціями декількома користувачами.

Статичний метод «UseToken» використовується для взаємодії з Google OAuth 2.0 для отримання токена доступу до API. Метод виконує POST-запит на URL-адресу https://www.googleapis.com/oauth2/v4/token для обміну авторизаційного коду на токен доступу.

У методі створюється черга запитів «RequestQueue» за допомогою бібліотеки Volley для відправлення HTTP-запитів. Визначається URL-адреса url

для POST-запиту до Google API. Створюється об'єкт «StringRequest» для виконання POST-запиту та у конструкторі передається набір параметрів.

У перевизначеному методі «getParams()» об'єкту «StringRequest» встановлюються параметри запиту. Цей метод використовується для обміну авторизаційного коду та ідентифікаційного токена на токен доступу за допомогою Google OAuth 2.0, дозволяючи застосунку отримувати доступ до захищених ресурсів API Google.

3.2 Керівництво користувача

У цьому розділі буде розглянуто ключові функції та інтерфейс мобільного додатку для читання книг з Google Books. Буде пояснено як ефективно навігувати в застосунку, шукати книги, зберігати їх у власних колекціях, а також налаштовувати вигляд тексту для комфортного читання. Це керівництво допоможе максимально використовувати всі можливості застосунку та насолоджуватися читанням у будь-який час і в будь-якому місці.

Запустивши програму першим, що зустрічає користувача є вхідний екран. Якщо користувач вперше має справу з даним застосунком, потрібно буде увійти в акаунт Google для подальшого користування. Після підтвердження акаунта, можна зайти в сам застосунок.

Першим, що побачить користувач буде головна сторінка з усіма книгами, які доступні на акаунті. Звідси можливо зразу почати читати книгу, якщо на цій сторінці знайдеться та, що підходить під вподобання користувача. У випадку того, що бажаної книги не знайдено, та користувач хоче знайти іншу, є можливість скористатися пошуковою системою в самому застосунку.

Пошукова система дозволяє найти будь-яку книгу, що доступна в Google Books API. Пошук проводиться за назвою книги, ключовим словом чи автором. Після того, як книги відповідні до запиту знайдено, їх буде відображено на екрані пристрою, що використовується.

Для того щоб почати читати книгу потрібно натиснути на її обкладинку чи назву. Відчиниться нове вікно з відповідною інформацією про цю книгу. Спершу це обкладинка та назва книги, далі кількість сторінок, видання та автор.

Також доступні деякі дії над книгою. Перше, що можна зробити, це занести книгу до полиці. Таким чином можна відсортувати її для подальшого читання. Наявні кілька полиць: «Улюблені», «В планах», «Прочитано», «Читаю». Для того щоб отримати доступ до цих полиць, потрібно на головному екрані натиснути на кнопку з трьох крапок у верхньому правому куті екрану. З'явиться випадаюче меню через яке можна потрапити на відповідні полиці.

Також на карті книги є інші кнопки, що дозволяють початок читання книги, попередній перегляд, покупку, якщо це потрібно, та видалення книги з полиці. Для повного видалення книги з полиці потрібно це підтвердити у спливаючому вікні, натиснувши «Так» або «Ні» при зміні думки. Попередній перегляд переносить користувача на сторінку з коротким початковим текстом книги. Також нижче наявний опис того, про що ця книга чи що в ній розглядається. Натиснувши кнопку повернення, в нижній частині екрану, програма поверне користувача назад на сторінку.

На головному екрані присутнє меню, воно прикріплене майже до всіх сторінок застосунку, так як є головною навігацією на сайті. Перша кнопка у вигляді будиночка повертає користувача на головну сторінку. Друга кнопка у вигляді серця переносить користувача на сторінку з обраними та улюбленими книгами, які можна почати читати прямо звідти. Та третя кнопка символізує інформацію, та дає змогу користувачеві отримати довідку про застосунок.

У середовищі для читання буде відображено текст книги, який можна редагувати деяким чином. Є змога змінити фоновий колір на чорний чи сепію, також є можливість зміни кольору тексту на представлені в застосунку. Ці дії робляться у випадаючому меню, що викликається натисненням на кнопку шестерні у верхньому лівому куті екрану.

Цей розділ дав огляд основних функцій мобільного застосунку для читання книг з Google Books. Тепер можна зрозуміти, як ефективно користуватися

додатком, починаючи з головного екрану, до використання пошуку для знаходження конкретних творів або авторів. Обширна діаграма послідовності дій користувача показана на рисунку Г.1 в додатку Г.

Завдяки цим функціям дана програма стає зручним і приємним інструментом для читання, дозволяючи насолоджуватися улюбленими книгами у будь-який час. Наступний крок – відкрити застосунок і почати досліджувати світ літератури, маючи під рукою величезну бібліотеку Google Books!

3.3 Вимоги до технічних та програмних засобів

Для забезпечення належної функціональності та користувацького досвіду мобільного застосунку для читання книг з Google Books на Android, необхідно дотримуватися певних технічних вимог. Застосунок розроблено для роботи на операційній системі Android, починаючи з версії 4.1 (API рівень 16) і вище. Це забезпечує сумісність з більшістю сучасних пристроїв та доступ до оновлень Android, які покращують безпеку та продуктивність.

Рекомендується, щоб пристрої мали принаймні 1 ГБ оперативної пам'яті та достатньо внутрішньої пам'яті для зберігання додаткових ресурсів та книг. Щодо середньої пам'яті для мобільних пристроїв, рекомендується мати як мінімум 16 ГБ внутрішньої пам'яті, щоб забезпечити достатній простір для зберігання книг та інших додаткових ресурсів. Це особливо важливо для книжкових застосунків, де контент часто включає великі обсяги даних та мультимедійні елементи. Вища кількість пам'яті також дозволяє покращити загальну швидкість та продуктивність застосунку, оскільки операційній системі не доведеться часто використовувати кешування на зовнішніх носіях, що зазвичай може сповільнити роботу пристрою.

Також важлива наявність стабільного з'єднання з Інтернетом для завантаження книг та оновлення бібліотеки. Оптимальна працездатність

застосунку залежить від потужності процесора техніки чи девайсу, що впливає на швидкість обробки даних та інтерфейсу користувача.

Для оптимальної роботи мобільного застосунку для читання книг з Google Books, середній процесор рекомендований для Android пристроїв повинен мати хорошу багатозадачність та енергоефективність. В ідеалі, процесори середнього класу, такі як Qualcomm Snapdragon серії 600 або вище, забезпечують достатню продуктивність для забезпечення плавної анімації та швидкого відгуку інтерфейсу застосунку.

Забезпечення цих технічних вимог допоможе користувачам безперебійно користуватися застосунком, насолоджуючись читанням та взаємодією з внутрішнім контентом.

3.4 Вибір та обґрунтування методів тестування застосунку

У процесі розробки мобільних застосунків, вибір та обґрунтування методів тестування є критичними для забезпечення якості та надійності програмного продукту. Тестування застосунків дозволяє виявити та виправити помилки до того, як продукт потрапить до кінцевого користувача, забезпечуючи таким чином кращий користувацький досвід.

Одним з основних методів тестування є ручне тестування – це тип тестування програмного забезпечення, при якому тестовий кейс виконується тестувальником вручну без допомоги будь-яких автоматизованих інструментів [39]. Цей метод дозволяє імітувати дії реальних користувачів та перевірити не лише функціональність, але й зручність інтерфейсу, відповідність дизайну та інтуїтивність використання застосунку. Тестувальники виконують різноманітні сценарії користування, перевіряють реакцію застосунку. Ручне тестування є особливо важливим, коли йдеться про перевірку застосунків із складними інтерфейсами та функціоналом.Завдяки ручному тестуванню можна отримати детальний зворотній зв'язок про користувацький досвід, що є незамінним для

вдосконалення програмного продукту перед його випуском на ринок. Ручне тестування застосунків має ряд переваг, особливо коли йдеться про забезпечення якості користувацького досвіду.

Однією з основних переваг є можливість точно відтворити користувацькі сценарії, що допомагає знайти помилки, які можуть не бути очевидними під час автоматизованого тестування.

З іншого боку, ручне тестування вимагає значних витрат часу та ресурсів, особливо у великих проєктах, де потрібно перевірити велику кількість функціоналу. Крім того, воно може виявитися менш повторюваним і залежним від суб'єктивних оцінок тестувальників, що потенційно може призвести до пропуску деяких дефектів.

Застосування системи документування через Test Case дозволяє структурувати процес тестування, підвищуючи його ефективність та забезпечуючи детальне відстеження вимог до застосунку [40]. Test Case формулює конкретні умови тестування та вимоги до результатів для кожного сценарію, що допомагає тестувальникам чітко оцінити, як застосунок повинен реагувати на різні дії. Це включає опис стартових умов, кроків для виконання, очікуваних результатів та фактичних результатів, забезпечуючи повторюваність та легкість верифікації виправлень помилок. Іноді строга структура Test Case може обмежувати швидкість тестування і адаптацію до нових умов без втрати актуальності документації. Отже, використання цієї системи потребує зваженого підходу та чіткого розуміння поточних потреб проєкту та ресурсів.

3.5 Тестування застосунку

Для початку потрібно визначити, що саме буде тестуватися в застосунку. Першим, що робить користувач, це входить в сам застосунок де потрібно пройти реєстрацію або обрати свій Google акаунт. Результати тестування наведено нижче в таблиці 3.1.

Таблиця 3.1 – Тестування процесу входу користувача в застосунок

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Натиснення на піктограму застосунку	Відчиниться застосунок та з'явиться вхідна сторінка з вибором акаунту	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути кнопку з вибором відповідного акаунту	Відбудеться підтвердження та відображення Toast-повідомлення	Passed
Натиснути кнопку «Tap to start»	Відбувається перехід на головну сторінку застосунку	Passed
Натиснення кнопки «Головна сторінка» в застосунку	Показ усіх доступних книг для акаунта нинішнього користувача	Passed

Також потрібно протестувати інші варіанти виборів дій користувачів. Спершу пройдеться сценарій, що випробовує головне меню у нижній частині екрану на сторінках. Відтворюватиметься натискання трьох кнопок наявних на даному меню та описуватимуться подальші події та наслідки натискання таких кнопок на панелі. Результат даного тестування поданий у таблиці 3.2.

Таблиця 3.2 – Тестування головного меню в застосунку

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Натиснення на піктограму кнопки	Відчиниться певна відповідна сторінка	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути кнопку з піктограмою хатинки тільки запустивши застосунок	Не відбувається нічого	Passed
Натиснути кнопку з піктограмою серця	Відбувається перехід на сторінку з улюбленими та обраними книгами користувача	Passed
Натиснення кнопки з піктограмою букви «І»	Перехід на сторінку з інформацією про застосунок	Passed

Наступне тестування проведено на випадаючому меню з «полицями» книг, що знаходиться зверху з правого боку. Виглядає меню як три верикальні крапки, що відкриває меню менших розмрів з кнопками для вибору полиці. Результати показані в таблиці 3.3.

Таблиця 3.3 – Тестування головного меню в застосунку

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Натиснення на піктограму трьох вертикальних крапок	Відчиниться випадаюче меню з кількома кнопками	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути кнопку з написом «It has been read»	Відбувається відкриття сторінки з книгами, що були обрані під дану категорію	Passed
Натиснути кнопку з з написом «Favorite»	Відбувається відкриття сторінки з книгами, що були обрані під дану категорію	Passed
Натиснення кнопки з з написом «In the plans»	Відбувається відкриття сторінки з книгами, що були обрані під дану категорію	Passed

Додатково було перевірено функції повернення на інші сторінки за допомогою головного меню для навігації та кнопки для повернення у вигляді стрілки, що знаходиться у верху сторінки, на яку зайшов користувач.

Важливим тестуванням є перевірка роботи пошукової системи в застосунку для читання книг.

Першочергово потрібно перевірити те, чи працює дана система в цілому, після чого потрібно вести якесь ключове слово чи автора книги для того щоб програма знайшла книги в бібліотеці. Після знаходження потрібно перевірити чи працює перехід на сторінку книги. Результати тестування представлені у таблиці 3.4, що наведена нижче.

Таблиця 3.4 – Тестування пошукової системи

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Натиснення на піктограму книги	Відчиниться рядок для вводу	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Ввести слово «Minecraft»	Відбувається пошук книг з даною назвою чи ключовим словом	Passed
Ввести ім’я автора – Stephen King	Відбувається знаходження доступних книг, що написав даний автор	Passed
Натиснення на обкладинку книги	Перехід на сторінку з інформацією про книгу	Passed

Було перевірено відображення інформації про книгу та роботу усіх кнопок на даній сторінці. Основними кнопками на сторінці є кнопки про перехід до читання, попередній перегляд книги, покупка книги, та видалення з полиці. Так як на сторінці з картою книги є багато кнопок тестування є доволі довгим та представлене у таблиці 3.5.

Таблиця 3.5 – Тестування кнопок на сторінці інформації про книгу

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Натиснення на обкладинку книги	Відчиниться сторінка з інформацією про книгу	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути на кнопку вибору «полиці»	Відчиняється випадаюче меню з відповідними кнопками	Passed
Натиснути кнопку на випадаючому меню	Відображення повідомлення про успішне додавання книги на «полицю»	Passed
Натиснення на кнопку «Читати»	Перехід на сторінку з середовищем для читання	Passed
Натиснення на кнопку «Попередній перегляд»	Перехід на сторінку з середовищем для читання скороченої версії книги	Passed

Продовження таблиці 3.5.

Натиснення на кнопку «Купити»	Нічого не відбувається, книгу не потрібно купляти	Passed
Натиснення на кнопку «Видалити з полиці»	Відчиняє спливаюче вікно з уточненням про видалення книги з «полиці»	Passed
Натиснення на кнопку «Так»	Повідомлення про видалення	Passed

Останньою областю тестування стане саме середовище читання. Воно має функції кастомізації тексту та фонової частини екрану.

Робиться це за допомогою маленького меню, що можна викликати натиснувши на шестерню в лівій верхній частині екрану. Результати тестування наведені нижче у таблиці 3.6. У таблиці описані передумови та вказані деталі даного випадку сценарію.

Таблиця 3.6 – Тестування середовища для читання

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Натиснення на кнопку «Читати»	Відчиниться сторінка з текстом	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути на кнопку шестерні	Відчиняється випадаюче меню	Passed
Натиснути кнопку на випадаючому меню	Зміна вигляду середовища читання	Passed

3.6 Аналіз результатів тестування

Аналізуючи результати тестування за допомогою таблиць Test Case для мобільного застосунку для читання книг з Google Books, можна зробити висновок про високу стабільність та надійність застосунку. Кожен тестовий випадок детально описує конкретну дію, очікуваний результат та кінцевий

статус, що дозволяє чітко визначити, чи відповідає застосунок встановленим вимогам до застосунку.

Детальніше можна розглянути деякі з таблиць тестування для мобільного застосунку для читання книг з Google Books. Зокрема, увагу привертає таблиця, яка описує тестування процесу входу користувача в застосунок. В цьому тесті перевіряється відкриття застосунку, натискання кнопки для входу, вибір аккаунта, а також реакція системи на ці дії.

Результати вказують на успішне виконання завдань, що свідчить про надійність процесів авторизації та входу.

Інша таблиця зосереджена на тестуванні навігаційних функцій в застосунку, таких як перехід до різних розділів через головне меню. Оцінюється відповідь застосунку на вибір розділів «Прочитано», «У планах», та «Улюблені», де перевіряється, як застосунок відображає відповідний контент. Успішне проходження цих тестів підтверджує ефективність інтерфейсу користувача та його взаємодію.

Ці таблиці важливі для забезпечення, що застосунок не тільки функціональний, але й зручний і інтуїтивно зрозумілий для користувачів. Таке детальне тестування відіграє ключову роль у розробці надійних і високоякісних мобільних застосунків.

Всі тестові сценарії, які включають навігацію між сторінками, взаємодію з елементами інтерфейсу та пошукові функції, показали позитивні результати, що свідчить про правильну реалізацію основних функцій застосунку. Це важливо для забезпечення хорошого користувацького досвіду і підвищує загальне задоволення користувачів. Зокрема, здатність застосунку коректно реагувати на введення і виводити точні результати при пошуку специфічних книг або авторів є вирішальною для користувацької ефективності.

Перевірка реакції застосунку на натискання різних кнопок інтерфейсу також підтвердила стабільність роботи програми. Успішне проходження всіх цих тестів вказує на те, що застосунок добре спроєктований та готовий до використання в реальних умовах.

3.7 Короткі висновки

У цьому розділі було розглянуто кілька ключових аспектів розробки та впровадження мобільного застосунку для читання книг з Google Books. Програмна реалізація модулів детально описала структуру та функціонування основних компонентів застосунку, забезпечуючи стабільність та високу продуктивність програми. Керівництво користувача надало інструкції з ефективного використання застосунку, що зробило його доступнішим для ширшої аудиторії. Вимоги до технічних та програмних засобів встановили необхідні умови для оптимальної роботи застосунку.

У підрозділі про вибір та обґрунтування методів тестування було забезпечено ефективне виявлення та виправлення помилок, підвищивши якість кінцевого продукту. Процес тестування застосунку та аналіз результатів підтвердив надійність та відповідність встановленим вимогам застосунку, що забезпечить зростання довіри користувачів та підвищення загальної задоволеності досвідом використання застосунку.

Зокрема, детальний аналіз результатів тестування показав, що застосунок відповідає всім критеріям якості, включаючи швидкість роботи, стабільність, безпеку та зручність використання. Це підтверджує, що застосунок готовий до випуску на ринок та здатний забезпечити користувачам надійний інструмент для цифрового читання.

Застосунок має потенціал значно покращити досвід користувачів у сфері цифрового читання, пропонуючи зручний доступ до великої кількості літератури та інструментів для її організації. Високий рівень безпеки даних та швидкість роботи забезпечують комфортне використання, а можливість читання офлайн робить застосунок незамінним для читачів у будь-яких умовах.

Таким чином, всі ці етапи виконані з метою створення надійного, ефективного і користувацьки приємного продукту.

ВИСНОВКИ

Під час розробки Інтернет-магазину з продажу продуктів харчування було описано та використано багато різних технологій та опрацьовано чимало інформації, що була вкладена у структуру трьох розділів.

Перший розділ описує фундаментальну підготовку для розробки інтернет-магазину, включаючи змістовний аналіз програмного забезпечення, його структурні та функціональні особливості. Інтернет-магазин харчових продуктів має інтегрувати функції, які відповідають сучасним вимогам користувачів до зручності, швидкості та інформативності. Важливою характеристикою такого застосунку є можливість швидко знаходити і купувати продукти з великого асортименту. Також було розглянуто наявне програмно-технічне забезпечення предметної області. «Смак життя», «Сільпо», «Producto», «Prime Food» – це обрані аналоги для розгляду та аналізу функціоналу, що потрібен у програмах обраного типу.

Визначення вимог до програмного забезпечення та формування технічного завдання були ключовими моментами, що дозволили чітко окреслити межі та цілі проєкту. В результаті, було створено міцну основу для подальшого розроблення та дизайну застосунку, забезпечивши повне розуміння потреб ринку та кінцевих користувачів.

Другий розділ зосереджений на виборі технологій і методів реалізації, проєктуванні бази даних, архітектурі системи, модулях застосунку та інтерфейсі користувача. В рамках розробки інтернет-магазину з продажу продуктів харчування було застосовано ряд сучасних інформаційних технологій, що дозволили реалізувати поставлені цілі та задачі. Зокрема, використання фреймворка Laravel для бекенду, забезпечило надійність та високу швидкість роботи застосунку. Система управління базами даних MySQL була обрана через її продуктивність та масштабованість. За середовища написання програми було обрано Visual Studio Code та Open Server, що підтримують потрібні мови програмування такі як PHP та Java Script.

Особливу увагу було приділено модульності та масштабованості системи, що забезпечує легке додавання нових функціональностей та обробку зростаючих обсягів даних. Проєктування інтерфейсу користувача було спрямоване на забезпечення інтуїтивної взаємодії та високої зручності для користувачів.

Третій розділ включав безпосередню програмну реалізацію модулів, підготовку керівництва для користувачів, а також визначення технічних та програмних вимог для оптимальної експлуатації застосунку. Були розроблені різні модулі, що дозволили підтримувати усі ключові процеси застосунку, від управління товарами до обробки замовлень.

Важливу роль зіграло тестування, яке охоплювало всі рівні застосунку, від одиничних модулів до системи в цілому. Аналіз результатів тестування показав високу надійність та продуктивність застосунку, підтверджуючи його готовність до запуску та експлуатації.

Після опрацювання усіх завдань, що були поставленіна початку розробки проєкту в результаті, було створено повнофункціональний інтернет-магазин, який не тільки відповідає всім сучасним вимогам до електронної комерції, але й забезпечує високий рівень задоволеності користувачів завдяки швидкості, зручності навігації та простоті використання. Застосунок демонструє високу продуктивність та стабільність при обробці замовлень та управлінні продуктами.

Впровадження розробленого застосунку принесе значні переваги для користувачів, включаючи можливість швидкого та легкого доступу до широкого асортименту продуктів, зручні опції для замовлення та отримання продукції, а також високий рівень індивідуалізації та персоналізації покупок. Користувачі також виграють від системи лояльності та персональних рекомендацій, що робить процес покупки більш ефективним і приємним.

Загальні висновки по трьох розділах підкреслюють успішну реалізацію комплексного підходу до розробки інтернет-магазину продуктів харчування. Від початкового аналізу та проєктування до фінальної реалізації та тестування, кожен етап роботи був спрямований на створення ефективного застосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Мобільні застосунки: їх види та особливості [Електронний ресурс]. Дата розміщення: 26.01.2023. Режим доступу: https://highload.today/uk/mobilni-zastosunki-yih-vidi-ta-osoblivosti/ (дата звернення: 06.03.2024)

2. Dawn Griffiths "Head First Android Development" (2nd Edition). O'Reilly. 2021. 928p.

3. Phillips, B., Stewart, C. "Android Programming: The Big Nerd Ranch Guide" (4th Edition). Addison-Wesley Professional, 2020. 784 pages.

4. Android (operating system) [Електронний ресурс]. Дата розміщення: 07.05.2024. Режим доступу: https://en.wikipedia.org/wiki/Android_(operating_system) (дата звернення: 08.03.2024)

5. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс]. Дата розміщення: 01.01.2021. Режим доступу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/ (дата звернення: 09.03.2024)

6. Android: переваги та недоліки [Електронний ресурс]. Дата розміщення: 05.03.2024. Режим доступу: https://ua5.org/opersys/2347-android-perevagy-ta-nedoliky.html (дата звернення: 11.03.2024)

7. Horton, J., Felker, R. "Android Programming for Beginners: Build in-depth, full-featured Android apps starting from zero programming experience, 3rd Edition". Packt Publishing, 2020. 766 pages.

8. Користувацький інтерфейс [Електронний ресурс]. Дата розміщення: 25.03.2024 Режим доступу: https://voll.com.ua/uk/glossary/koristuvackij-interfejs-ui (дата звернення: 13.03.2024)

9. Український додаток з книгами Librarius [Електронний ресурс]. Дата розміщення: 14.04.2021. Режим доступу: https://24tv.ua/tech/ukrayini-zapratsyuvav-dodatok-knigami-librarius-novini-tehnologiy_n1598074 (дата звернення: 15.03.2024)

10. Librarius. Тисячі книжок завжди з тобою. [Електронний ресурс]. Дата розміщення: 07.04.2020. Режим доступу: https://librarius.pro (дата звернення: 16.03.2024)

11. Що таке Goodreads і як використовувати цей сервіс [Електронний ресурс]. Дата розміщення: 17.01.2024. Режим доступу: https://contentguide.com.ua/shho-take-goodreads-i-yak-vykorystovuvaty-czej-servis/ (дата звернення: 19.03.2024)

12. Goodreads [Електронний ресурс]. Дата розміщення: 06.05.2024. Режим доступу: https://play.google.com/store/apps/details?id=com.goodreads&hl=en&pli=1 (дата звернення: 19.03.2024)

13. Bookmate: книги й аудіокниги [Електронний ресурс]. Дата розміщення: 14.05.2024. Режим доступу: https://play.google.com/store/apps/details?id=com.bookmate&hl=uk&gl=US (дата звернення: 19.03.2024)

14. Wiegers, K. E., Beatty, J. "Software Requirements" (4th Edition). Microsoft Press, 2020. 816 pages.

15. Варіанти використання та сценарії (Use Cases and Scenarios) [Електронний ресурс]. Дата розміщення: 05.03.2023. Режим доступу: https://www.maxzosim.com/use-cases-and-scenarios/ (дата звернення: 24.03.2024)

16. Use-case diagrams [Електронний ресурс]. Дата розміщення: 04.03.2021. Режим доступу: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case (дата звернення: 27.03.2024)

17. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 4th ed. Boston: Pearson, 2020. 736 p.

18. Що таке функціональні вимоги: приклади, визначення, повний посібник [Електронний ресурс]. Дата розміщення: 10.03.20. Режим доступу: https://visuresolutions.com/uk/blog/functional-requirements/ (дата звернення: 30.03.2024)

19. Wiegers, K. E., Beatty, J. "Software Requirements" (4th Edition). Microsoft Press, 2020. 816 pages.

20. Захист даних та інформаційна безпека [Електронний ресурс]. Дата розміщення: 25.01.2022. Режим доступу: https://www.dqsglobal.com/uk-ua/navchajtesya/blog/zahist-danih-ta-informacijna-bezpeka-za-standartami-iso-27001-ta-iso-27701 (дата звернення: 02.04.2024)

21. Нефункціональні вимоги: приклади, типи, підходи [Електронний ресурс]. Дата розміщення: 23.08.2023. Режим доступу: https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/ (дата звернення: 03.04.2024)

22. Функціональні та нефункціональні вимоги [Електронний ресурс]. Дата розміщення: 30.12.2023. Режим доступу: https://www.guru99.com/uk/functional-vs-non-functional- requirements.html?gpp &gpp_sid (дата звернення: 04.04.2024)

23. Безпека в мобільних додатках: На що звернути увагу? [Електронний ресурс]. Дата розміщення: 07.05.2024. Режим доступу: https://mindscope.biz.ua/bezpeka-v-mobilnyh-dodatkah-na-shho-zvernuty-uvagu/ (дата звернення: 04.04.2024)

24. СЛОВНИК ОСНОВНИХ UI/UX ЕЛЕМЕНТІВ [Електронний ресурс]. Дата розміщення: 10.06.2021. Режим доступу: https://training.qatestlab.com/blog/technical-articles/dictionary-of-basic-ui-ux-elements/ (дата звернення: 06.04.2024)

25. What is user interface (UI): meaning, principles, and examples [Електронний ресурс]. Дата розміщення: 12.02.2021. Режим доступу: https://dovetail.com/ux/ui-meaning/ (дата звернення: 07.04.2024)

26. Інтерактивний прототип: покращення взаємодії з користувачем за допомогою інтерактивного дизайну [Електронний ресурс]. Дата розміщення: 17.06.2022. Режим доступу: https://dizz.in.ua/uk/klikabelnyj-prototip-uluchshenie-polzovatelskogo-opyta-posredstvom-interaktivnogo-dizajna/ (дата звернення: 08.04.2024)

27. Leffingwell, D., Widrig, D. "Managing Software Requirements: A Unified Approach". Addison-Wesley Professional, 2020. 528 pages.

28. Що таке Figma і навіщо вона потрібна? [Електронний ресурс]. Дата розміщення: 28.09.23. Режим доступу: https://dan-it.com.ua/uk/blog/chto-takoe-figma-i-zachem-ona-nuzhna/ (дата звернення: 13.04.2024)

29. Основні недоліки Figma: про що варто знати [Електронний ресурс]. Дата розміщення: 21.07.21. Режим доступу: https://designtalk.club/osnovni-nedoliky-figma-pro-shho-varto-znaty/ (дата звернення: 16.04.2024)

30. Android Studio: що це таке і для чого потрібна [Електронний ресурс]. Дата розміщення: 17.05.2024. Режим доступу: https://androidayuda.com/uk/андроїд-студія/ (дата звернення: 17.04.2024)

31. Android Studio — A Comparison Guide with Pros and Cons [Електронний ресурс]. Дата розміщення: 17.01.2023. Режим доступу: https://medium.com/nerd-for-tech/xamarin-vs-android-studio-a-comparison-guide-with-pros-and-cons-dab30cb11008 (дата звернення: 19.04.2024)

32. What is Java? [Електронний ресурс]. Дата розміщення: 11.04.23. Режим доступу: https://www.ibm.com/topics/java (дата звернення: 20.04.2024)

33. Робота із XML [Електронний ресурс]. Дата розміщення: 16.08.22. Режим доступу: https://helpx.adobe.com/ua/incopy/using/xml.html (дата звернення: 21.04.2024)

34. Books APIs Overview [Електронний ресурс]. Дата розміщення: 16.04.2022. Режим доступу: https://developers.google.com/books/docs/overview (дата звернення: 22.04.2024)

35. OAuth 2.0 Authorization Framework [Електронний ресурс]. Дата розміщення: 19.09.2023. Режим доступу: https://auth0.com/docs/authenticate/protocols/oauth (дата звернення: 22.04.2024)

36. Using OAuth 2.0 to Access Google APIs [Електронний ресурс]. Дата розміщення: 08.01.2024. Режим доступу: https://developers.google.com/identity/protocols/oauth2 (дата звернення: 22.04.2024)

37. Класи, види вкладених класів із прикладами [Електронний ресурс]. Дата розміщення: 04.09.23. Режим доступу: https://javarush.com/ua/groups/posts/uk.588.klasi-vidi-vkladenikh-klasv-z-prikladami (дата звернення: 24.04.2024)

38. Міжнародний день шульги : сумські психологи про особливості лівшів [Електронний ресурс]. Дата розміщення: 13.08.2019. Режим доступу: https://sumypost.com/sumynews/suspilstvo/mizhnarodnyj-den-shulgy-sumski-psyhology-pro-osoblyvosti-livshiv/ (дата звернення: 25.04.2024)

39. Що таке ручне тестування? [Електронний ресурс]. Дата розміщення: 02.06.2021 Режим доступу: https://www.zaptest.com/uk/ручне-тестування-що-це-таке-типи-проц (дата звернення: 25.04.2024)

40. How to write Test Cases – Software Testing [Електронний ресурс]. Дата розміщення: 29.04.2024 Режим доступу: https://www.geeksforgeeks.org/test-case/ (дата звернення: 26.04.2024)

ДОДАТОК А
ПОРІВНЯЛЬНА ТАБЛИЦЯ

Таблиця А.1 – Порівняльна таблиця існуючих програмних рішень

Застосунок	Функціональність	Фірма-розробник	Інтерфейсні вікна	Переваги	Недоліки
Bookmate	Пошук і читання книг, прослуховування аудіокниг, збереження закладок, соціальні функції	Bookmate Ltd.	Головна сторінка, Пошук, Моя бібліотека, Читання книги, Слухання аудіокниг	Широка бібліотека з різноманіттям жанрів, можливість читати і слухати книги, рекомендації, функції соціальної взаємодії	Потребує підписки для повного доступу до всіх книг і функцій, обмежений вибір книг у певних мовах або жанрах
Goodreads	Відстеження прочитаних книг, читання рецензій, планування читання, соціальні функції	Goodreas Inc. (належить Amazon)	Головна сторінка, Пошук, Моя бібліотека, Рецензії, Соціальні функції	Велика база даних книг із рецензіями, інструменти для планування читання відстеження та	Інтерфейс може бути перевантаженим, приватність даних може викликати занепокоєння через зв'язок з Amazon
Librarius	Пошук і читання книг, збереження закладок, управління бібліотекою, офлайн режим	Librarius Ltd.	Головна сторінка, Пошук, Моя бібліотека, Читання книги	Простий і зручний інтерфейс, доступ до великої кількості книг, можливість читання офлайн	Обмежена функціональніь безпідключення до інтернету, потребує підписки для доступу до всіх функцій
Chapter1	Пошук і читання книг, збереження закладок, персоналізовані рекомендації, офлайн режим	Chapter1 Ltd.	Головна сторінка, Пошук, Моя бібліотека, Читання книги, Рекомендації	Широка бібліотека з різноманітними жанрами, персоналізовані рекомендації	Потребує підписки для доступу до повної бібліотеки, обмежений вибір книг у деяких жанрах

ДОДАТОК Б

ДІАГРАМА КЛАСІВ

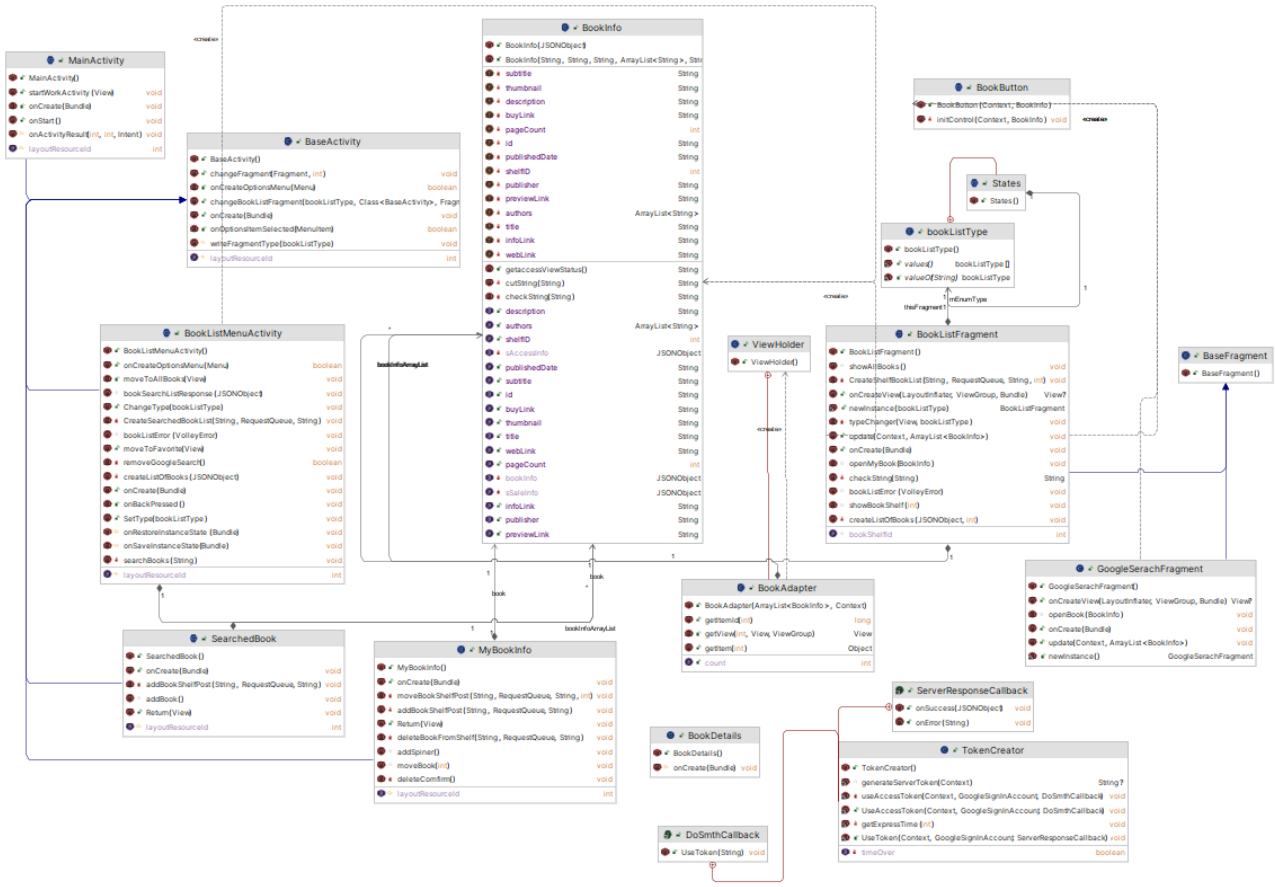


Рисунок Б.2 – Розкрита діаграма класів

ДОДАТОК В

КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС

Рисунок В.1 – Екран входу в застосунок

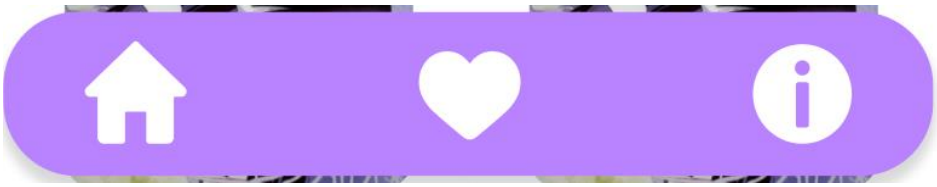


Рисунок В.2 – Головне меню екрану

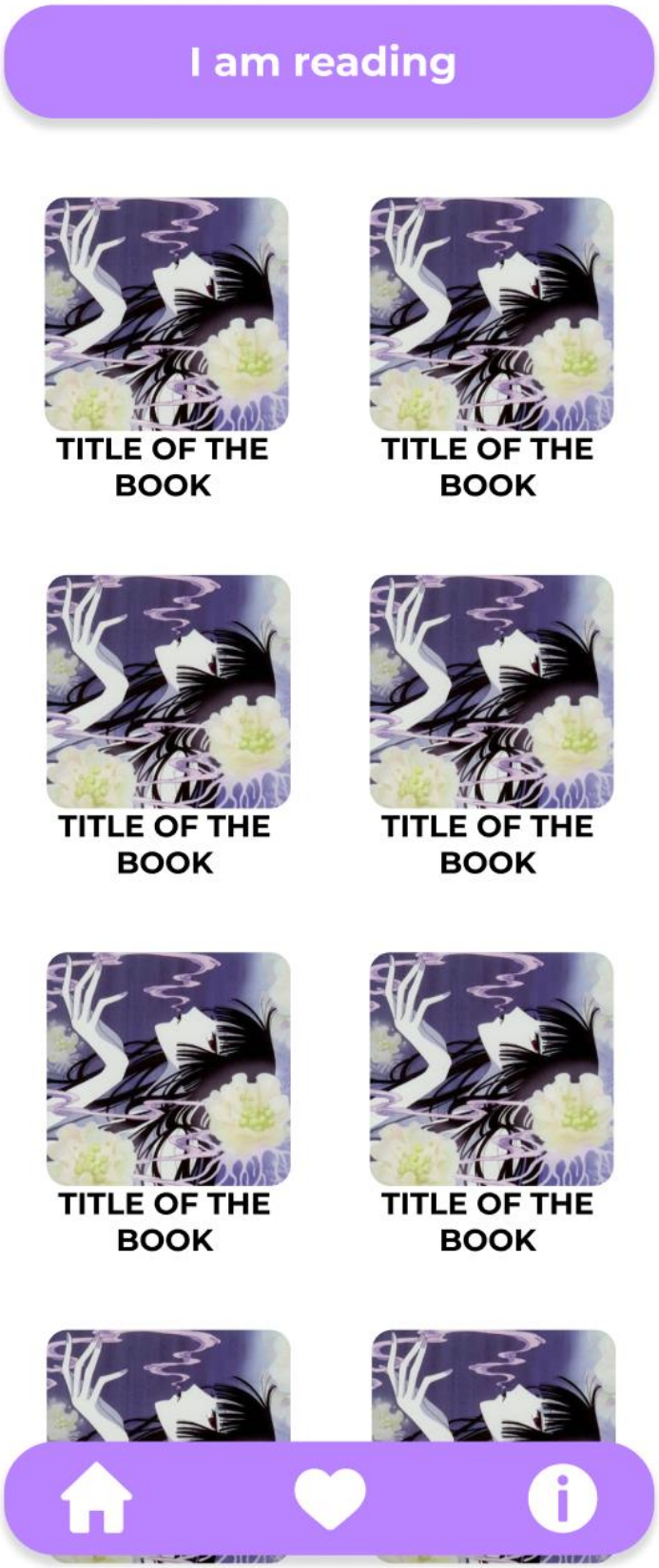


Рисунок В.3 – Екран з назвою «I am reading»

Рисунок В.4 – Випадаюче меню з категоріями книг

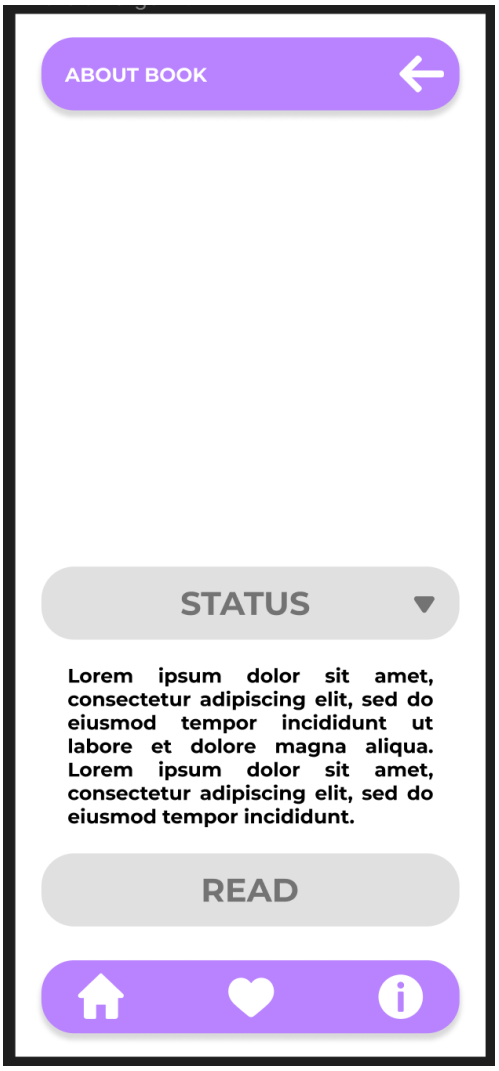


Рисунок В.5 – Випадаюче меню з категоріями книг

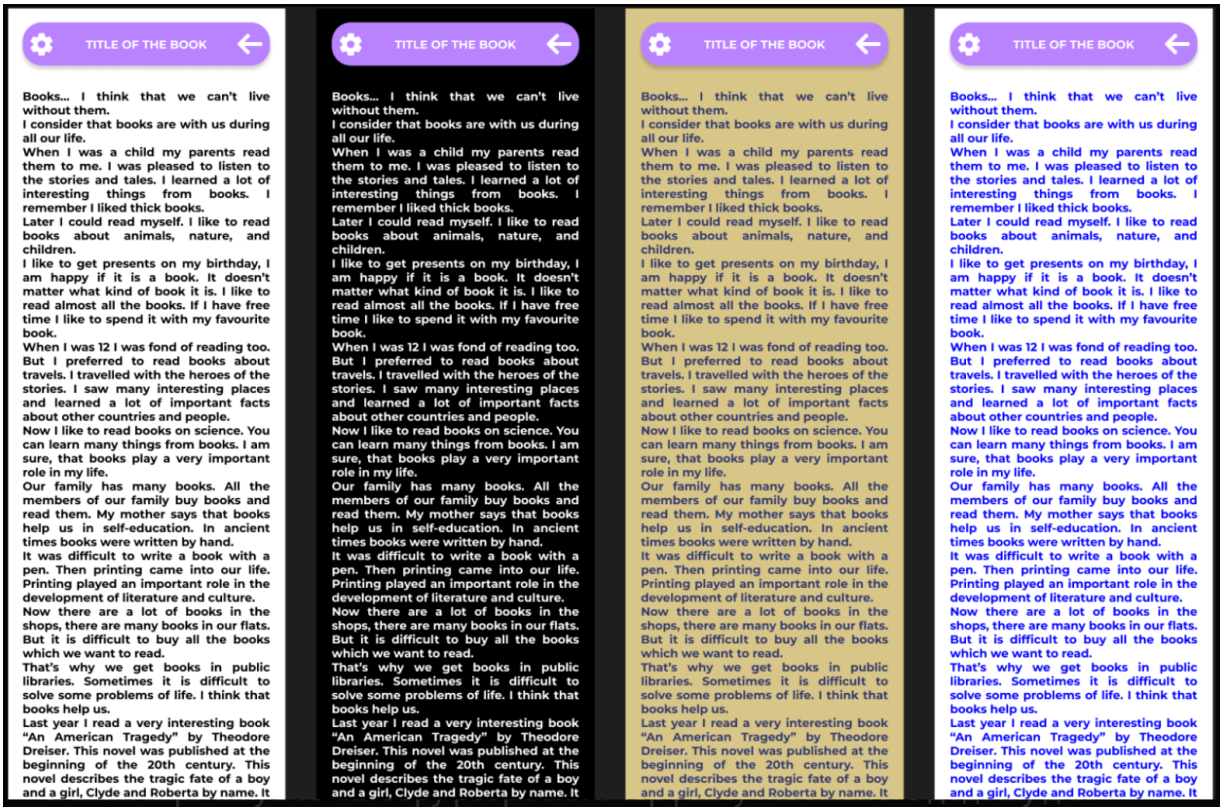


Рисунок В.6 – Варіації поля для читання

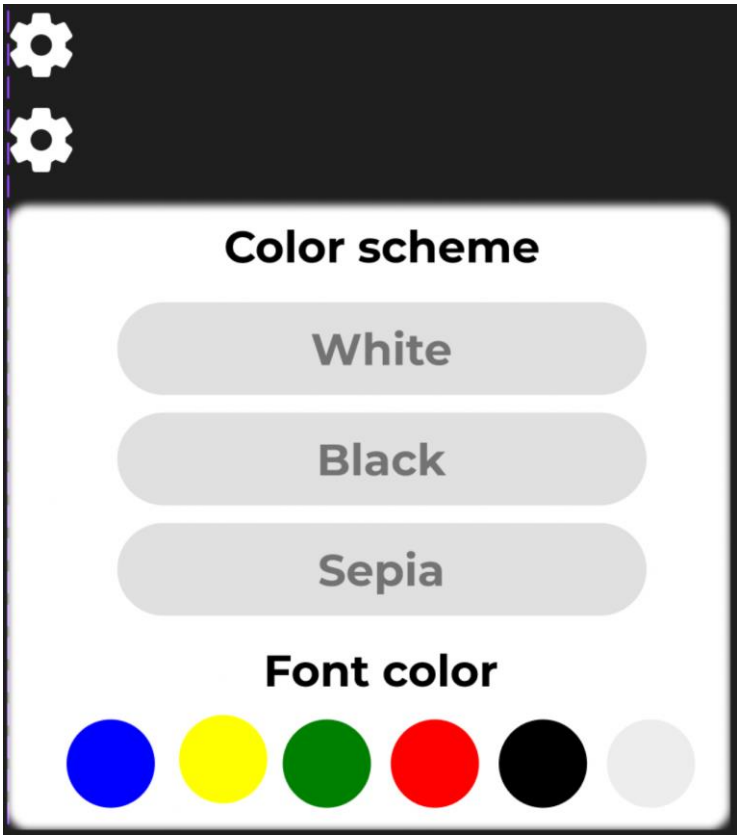


Рисунок В.7 – Випадаюче меню для налаштувань поля для читання

ДОДАТОК Г

ДІАГРАМА ДІЙ КОРИСТУВАЧА

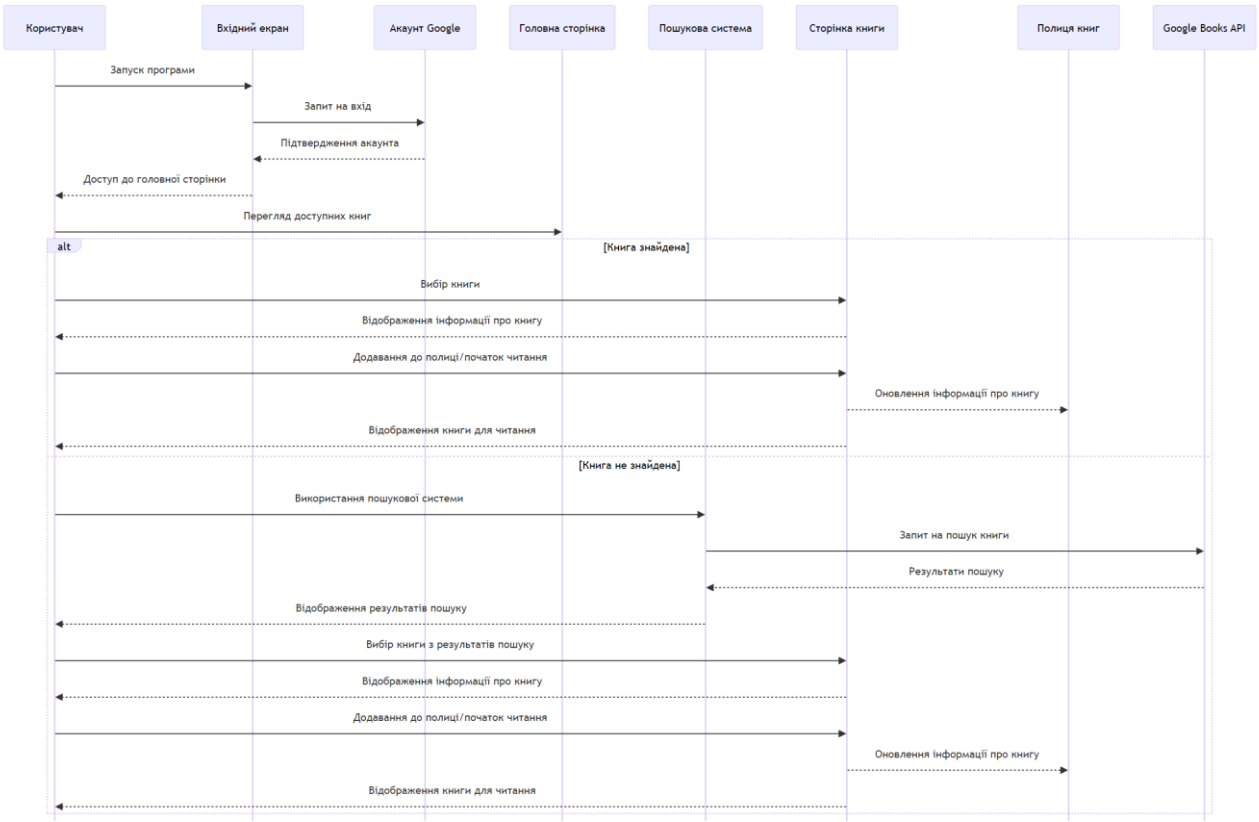


Рисунок Г.1 — Діаграма послідовності дій для користувача

ДОДАТОК Ґ
ПРОГРАМНИЙ КОД

Ґ.1 Програмний код модуля BaseAactivity.java

```
package com.example.myapplication;

import static androidx.constraintlayout.helper.widget.MotionEffect.TAG;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;

import com.example.myapplication.SupportClasses.States;

public abstract class BaseActivity extends AppCompatActivity {

    protected abstract int getLayoutResourceId();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(getLayoutResourceId());


this.getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_HIDE_N
AVIGATION
                | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY |
                View.SYSTEM_UI_FLAG_FULLSCREEN );
    }

    @SuppressLint("ClickableViewAccessibility")
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        return super.onCreateOptionsMenu(menu);
    }

    @SuppressLint("NonConstantResourceId")
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        boolean isSerach = false;

        States.bookListType type = null;

        switch(id){
```

```
            case R.id.reading :
                type = States.bookListType.read;
                break;
            case R.id.in_the_plans:
                type = States.bookListType.plans;
                break;
            case R.id.has_been_read:
                type = States.bookListType.readed;
                break;
            case R.id.favorite:
                type = States.bookListType.favorite;
                break;
            case R.id.menu_search:
                isSerach = true;
                break;
            default:
                type = States.bookListType.all;
                break;
        }


        if(type == null) return super.onOptionsItemSelected(item);

        if(this.getClass() == BookListMenuActivity.class) {
            BookListFragment fragment = BookListFragment.newInstance(type);
            changeBookListFragment(type, BookListMenuActivity.class, fragment,
R.id.book_list_fragment);
        } else {
            Intent intent = new Intent(this, BookListMenuActivity.class);
            intent.putExtra("type", type);
            startActivity(intent);
            overridePendingTransition(R.anim.swipe_to_up, R.anim.no_animation);
        }

        return super.onOptionsItemSelected(item);
    }

    public void changeFragment(Fragment fragment, int place) {
        FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
        ft.setCustomAnimations(R.anim.slide_in_left, R.anim.slide_in_right);
        ft.replace(place, fragment);
        ft.commit();
    }

    public void changeBookListFragment(States.bookListType type, Class<? extends
BaseActivity> activity,
                                       Fragment fragment, int place) {

        if(States.thisFragment == type && this.getClass() == activity) return;
        writeFragmentType(type);
        changeFragment(fragment, place);
    }

    protected void writeFragmentType(States.bookListType type) {
        States.thisFragment = type;
    }
}
```

Ґ.2 Програмний код модуля BookAdapter.java

```
package com.example.myapplication.SupportClasses;

import android.annotation.SuppressLint;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.myapplication.R;
import com.example.myapplication.SupportClasses.BookInfo;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;

public class BookAdapter extends BaseAdapter {
    private ArrayList<BookInfo> bookInfoArrayList;
    private Context mContext;

    public BookAdapter(ArrayList<BookInfo> bookInfoArrayList, Context context) {
        this.bookInfoArrayList = bookInfoArrayList;
        this.mContext = context;
    }

    @Override
    public int getCount() {
        return bookInfoArrayList.size();
    }

    @Override
    public Object getItem(int position) {
        return bookInfoArrayList.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }


    @SuppressLint("ViewHolder")
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(mContext);
        convertView = inflater.inflate(R.layout.book_rv_item, parent, false);

        ViewHolder viewHolder = new ViewHolder();
        viewHolder.nameTV = convertView.findViewById(R.id.idTVBookTitle);
        viewHolder.bookIV = convertView.findViewById(R.id.idIVbook);

        convertView.setTag(viewHolder);

        BookInfo bookInfo = bookInfoArrayList.get(position);
        viewHolder.nameTV.setText(bookInfo.getTitle());
        Picasso.get().load(bookInfo.getThumbnail()).into(viewHolder.bookIV);

        return convertView;
    }

    public class ViewHolder {
        TextView nameTV;
        ImageView bookIV;}}
```

Ґ.3 Програмний код модуля BookInfo.java

```
package com.example.myapplication.SupportClasses;
import static androidx.constraintlayout.helper.widget.MotionEffect.TAG;

import android.util.Log;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.Serializable;
import java.util.ArrayList;

public class BookInfo implements Serializable {

    private String id;
    private int shelfID;
    private String title;
    private String subtitle;
    private ArrayList<String> authors;
    private String publisher;
    private String publishedDate;
    private String description;
    private int pageCount;
    private String thumbnail;
    private String previewLink;
    private String infoLink;
    private String buyLink;
    private String webLink;
    private String accessViewStatus;

    // creating getter and setter methods
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getSubtitle() {
        return subtitle;
    }

    public void setSubtitle(String subtitle) {
        this.subtitle = subtitle;
    }

    public ArrayList<String> getAuthors() {
        return authors;
    }

    public void setAuthors(ArrayList<String> authors) {
        this.authors = authors;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
```

```
    this.publisher = publisher;
}

public String getPublishedDate() {
    return publishedDate;
}

public void setPublishedDate(String publishedDate) {
    this.publishedDate = publishedDate;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public int getPageCount() {
    return pageCount;
}

public void setPageCount(int pageCount) {
    this.pageCount = pageCount;
}

public String getThumbnail() {
    return thumbnail;
}

public void setThumbnail(String thumbnail) {
    this.thumbnail = thumbnail;
}

public String getPreviewLink() {
    return previewLink;
}

public void setPreviewLink(String previewLink) {
    this.previewLink = previewLink;
}

public String getInfoLink() {
    return infoLink;
}

public void setInfoLink(String infoLink) {
    this.infoLink = infoLink;
}

public String getBuyLink() {
    return buyLink;
}

public String getId() {
    return id;
}
public String getWebLink() {
    return webLink;
}
public String getaccessViewStatus() {
    return accessViewStatus;
}
```

```
    public void setBuyLink(String buyLink) {
        this.buyLink = buyLink;
    }
    public int getShelfID() {
        return shelfID;
    }
    public void setShelfID(int id) {
        shelfID = id;
    }

    public BookInfo(String id, String title, String subtitle, ArrayList<String>
authors, String publisher,
                    String publishedDate, String description, int pageCount,
String thumbnail,
                    String previewLink, String infoLink, String buyLink) {
        this.id = id;
        this.title = title;
        this.subtitle = subtitle;
        this.authors = authors;
        this.publisher = publisher;
        this.publishedDate = publishedDate;
        this.description = description;
        this.pageCount = pageCount;
        this.thumbnail = thumbnail;
        this.previewLink = previewLink;
        this.infoLink = infoLink;
        this.buyLink = buyLink;
        this.shelfID = -1;
    }

    public BookInfo(JSONObject itemsObj) throws JSONException {

            this.id = itemsObj.optString("id");
            this.shelfID = -1;

            JSONObject volumeObj = itemsObj.getJSONObject("volumeInfo");
            setBookInfo(volumeObj);

            JSONObject saleInfoObj = itemsObj.optJSONObject("saleInfo");
            setsSaleInfo(saleInfoObj);

            JSONObject accessInfo = itemsObj.optJSONObject("accessInfo");
            setsAccessInfo(accessInfo);

            ArrayList<String> authorsArrayList = new ArrayList<>();
            if (volumeObj.has("authors")) {
                JSONArray authorsArray = volumeObj.getJSONArray("authors");
                authorsArrayList = new ArrayList<>();
                if (authorsArray.length() != 0) {
                    for (int j = 0; j < authorsArray.length(); j++) {
                        authorsArrayList.add(authorsArray.optString(j));
                    }
                }
            }
            this.authors = authorsArrayList;
        }

    private String checkString(String str) {
        return (str != null) ? str : "";
    }

    private String cutString(String text) {
        int startIndex = text.indexOf("hl=");
        return text.substring(0, startIndex);
```

```
    }

    private void setBookInfo(JSONObject volumeObj){
        this.title = checkString(volumeObj.optString("title"));
        this.subtitle = checkString(volumeObj.optString("subtitle"));
        this.publisher = checkString(volumeObj.optString("publisher"));
        this.publishedDate = checkString(volumeObj.optString("publishedDate"));
        this.description = checkString(volumeObj.optString("description"));
        this.pageCount = volumeObj.optInt("pageCount");
        JSONObject imageLinks = volumeObj.optJSONObject("imageLinks");
        this.previewLink = checkString(volumeObj.optString("previewLink"));
        this.infoLink = checkString(volumeObj.optString("infoLink"));

        if (imageLinks != null) {
            this.thumbnail = checkString(imageLinks.optString("thumbnail"));
        } else {
            this.thumbnail = null;
        }
    }

    private void setsSaleInfo(JSONObject saleInfoObj) {
        if (saleInfoObj != null)
            this.buyLink = checkString(saleInfoObj.optString("buyLink"));
        else
            this.buyLink = null;
    }

    private void setsAccessInfo(JSONObject accessInfo) {
        if (accessInfo != null) {
            this.webLink =
cutString(checkString(accessInfo.optString("webReaderLink")));
            this.accessViewStatus =
checkString(accessInfo.optString("accessViewStatus"));
        } else {
            this.webLink = null;
            this.accessViewStatus = null;
        }
    }
}
```

Ґ.4 Програмний код модуля TokenCreator.java

```
package com.example.myapplication.SupportClasses;
//410211102598-ug99uocplb3m1f45q5na5p4ia1eaig02.apps.googleusercontent.com

import static androidx.constraintlayout.helper.widget.MotionEffect.TAG;

import android.content.Context;
import android.util.Log;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;

import org.json.JSONException;
import org.json.JSONObject;
```

```
import java.time.LocalDateTime;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class TokenCreator{

    static String access_token;
    static long express_in = 0;

    public interface ServerResponseCallback {
        void onSuccess(JSONObject jsonObject) throws JSONException;
        void onError(String error);
    }

    public interface DoSmthCallback {
        void UseToken(String token);
    }

    static String generateServerToken(Context context) {
        GoogleSignInAccount acc = GoogleSignIn.getLastSignedInAccount(context);
        if(acc != null) {
            return acc.getServerAuthCode();
        }
        return null;
    }

    public static void UseToken(Context context, GoogleSignInAccount acc,
ServerResponseCallback callback){
        RequestQueue queue = Volley.newRequestQueue(context);
        String url = "https://www.googleapis.com/oauth2/v4/token";

        StringRequest stringRequest = new StringRequest(Request.Method.POST,
url,
                response -> {
                    try {
                        JSONObject jsonObject = new JSONObject(response);
                        callback.onSuccess(jsonObject);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                },
                error -> {
                    Log.e(TAG, error.toString());
                    error.printStackTrace();
                }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> params = new HashMap<>();
                params.put("grant_type", "authorization_code");
                params.put("client_id", "410211102598-
ug99uocplb3m1f45q5na5p4ia1eaig02.apps.googleusercontent.com");
                params.put("client_secret", "GOCSPX-
IHqYwKIlh_1OJLuTM0x0n4gkOekW");
                params.put("redirect_uri", "");
                params.put("code", generateServerToken(context));
                params.put("id_token", acc.getIdToken());
                return params;
            }
        };

        queue.add(stringRequest);
```

```
    }

    private static void useAccessToken(Context context, GoogleSignInAccount acc,
DoSmthCallback callback) {
        UseToken(context, acc, new ServerResponseCallback() {
                    @Override
                    public void onSuccess(JSONObject jsonObject) {
                        access_token = jsonObject.optString("access_token");
                        getExpressTime(jsonObject.optInt("expires_in"));
                        callback.UseToken(access_token);
                    }

                    @Override
                    public void onError(String error) {

                    }
                }
        );
    }

    public static void UseAccessToken(Context context, GoogleSignInAccount acc,
DoSmthCallback callback){
        if(isTimeOver() || access_token == null) {
            useAccessToken(context, acc, callback);
            return;
        }
        callback.UseToken(access_token);
    }

    private static void getExpressTime(int timeInSecond) {
        long currentTime = Calendar.getInstance().getTimeInMillis();
        express_in = currentTime + (timeInSecond * 1000L);
    }

    private static boolean isTimeOver() {
        return express_in <= Calendar.getInstance().getTimeInMillis();
    }
}
```

Ґ.5 Програмний код модуля BookListMenuActivity.java

```
package com.example.myapplication;

import static androidx.constraintlayout.helper.widget.MotionEffect.TAG;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.nfc.Tag;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ProgressBar;
import android.widget.Toast;

import androidx.appcompat.widget.SearchView;
```

```
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.recyclerview.widget.LinearLayoutManager;

import com.android.volley.toolbox.StringRequest;
import com.example.myapplication.SupportClasses.TokenCreator;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import com.example.myapplication.SupportClasses.BookInfo;
import com.example.myapplication.SupportClasses.States;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.tasks.OnCompleteListener;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class BookListMenuActivity extends BaseActivity {

    private RequestQueue mRequestQueue;
    private ArrayList<BookInfo> bookInfoArrayList;
    private ProgressBar progressBar;
    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_book_list_menu;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        States.bookListType arguments = (States.bookListType)
getIntent().getSerializableExtra("type");

        boolean isFirstStart = true;

        if (savedInstanceState != null) {
            isFirstStart = savedInstanceState.getBoolean("isFirstStart");
        }

        if (!isFirstStart) return;

        if (arguments != null ) {
            SetType(arguments);
        } else {
            if(States.thisFragment == null)
                SetType(States.bookListType.all);
            else SetType(States.thisFragment);
        }
    }

    @Override
```

```
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        outState.putBoolean("isFirstStart", false);
    }

    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater findMenuItems = getMenuInflater();
        findMenuItems.inflate(R.menu.main_menu, menu);

        MenuItem menuItem = menu.findItem(R.id.menu_search);
        SearchView searchView = (SearchView) menuItem.getActionView();

        menuItem.setOnActionExpandListener(new MenuItem.OnActionExpandListener()
{

            @Override
            public boolean onMenuItemActionExpand(MenuItem menuItem) {
                GoogleSerachFragment fragment =
GoogleSerachFragment.newInstance();
                changeFragment(fragment, R.id.google_books_list);
                return true;
            }

            @Override
            public boolean onMenuItemActionCollapse(MenuItem menuItem) {
                removeGoogleSearch();
                return true;
            }
        });
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                progressBar = findViewById(R.id.idLoadingPB);
                progressBar.setVisibility(View.VISIBLE);

                if (query.isEmpty()) {
                    return false;
                }

                try {
                    searchBooks(query);
                } catch (JSONException e) {
                    throw new RuntimeException(e);
                }
                return true;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                return false;
            }
        });

        return super.onCreateOptionsMenu(menu);
    }
```

```
    private boolean removeGoogleSearch() {
        if (this.getClass() == BookListMenuActivity.class) {
            FragmentManager fragmentManager = getSupportFragmentManager();
            Fragment fragment =
fragmentManager.findFragmentById(R.id.google_books_list);
            if (fragment != null) {
                fragmentManager.beginTransaction().remove(fragment).commit();
            }
            return true;
        }
        return false;
    }

    @Override
    public void onBackPressed() {
        ChangeType(States.bookListType.all);
    }

    public void moveToFavorite(View v) {
        ChangeType(States.bookListType.favorite);
    }

    public void moveToAllBooks(View v) {
        ChangeType(States.bookListType.all);
    }

    public void ChangeType(States.bookListType type) {
        BookListFragment fragment = BookListFragment.newInstance(type);
        changeBookListFragment(type, this.getClass(), fragment,
R.id.book_list_fragment);
    }

    public void SetType(States.bookListType type) {
        BookListFragment fragment = BookListFragment.newInstance(type);
        writeFragmentType(type);
        changeFragment(fragment, R.id.book_list_fragment);
    }

    private void searchBooks(String query) throws JSONException {
        bookInfoArrayList = new ArrayList<>();
        mRequestQueue = Volley.newRequestQueue(this);
        mRequestQueue.getCache().clear();

        GoogleSignInAccount acc = GoogleSignIn.getLastSignedInAccount(this);
        RequestQueue queue = Volley.newRequestQueue(this);

        String url = "https://www.googleapis.com/books/v1/volumes?q=" + query +
"&maxResults=40";
        TokenCreator.UseAccessToken(this, acc,
                 (String key)-> {
                     CreateSearchedBookList(url, queue, key);
         });
    }

    private void CreateSearchedBookList(String url, RequestQueue queue, String
ACCESS_TOKEN) {
        JsonObjectRequest booksObjrequest =
                new JsonObjectRequest(Request.Method.GET, url,null,
this::bookSearchListResponse, this::bookListError)
        {
            @Override
            public Map<String, String> getHeaders() {

                    Map<String, String> headers = new HashMap<>();
```

```
                    headers.put("Authorization", "Bearer " + ACCESS_TOKEN);
                    return headers;
                }
            };

            queue.add(booksObjrequest);
        }
        void bookSearchListResponse(JSONObject response) {
            progressBar.setVisibility(View.GONE);
            try {
                createListOfBooks(response);

                FragmentManager fragmentManager = getSupportFragmentManager();
                GoogleSerachFragment fragment = (GoogleSerachFragment)
fragmentManager.findFragmentById(R.id.google_books_list);
                if (fragment != null) {
                    fragment.update(this, bookInfoArrayList);
                }

            } catch (JSONException e) {
                Log.d(TAG, "" + e);
                Toast.makeText(this, ("No Book Found"), Toast.LENGTH_SHORT).show();
                e.printStackTrace();
            }
        }

        void bookListError(VolleyError error) {
            Toast.makeText(this, "Error found is " + error,
Toast.LENGTH_SHORT).show();
            Log.d(TAG, "" + error);
            error.printStackTrace();
        }

        private void createListOfBooks(JSONObject response) throws JSONException {
            JSONArray itemsArray = response.getJSONArray("items");
            for (int i = 0; i < itemsArray.length(); i++) {
                JSONObject itemsObj = itemsArray.getJSONObject(i);

                BookInfo bookInfo = new BookInfo(itemsObj);

                bookInfoArrayList.add(bookInfo);}}}
```

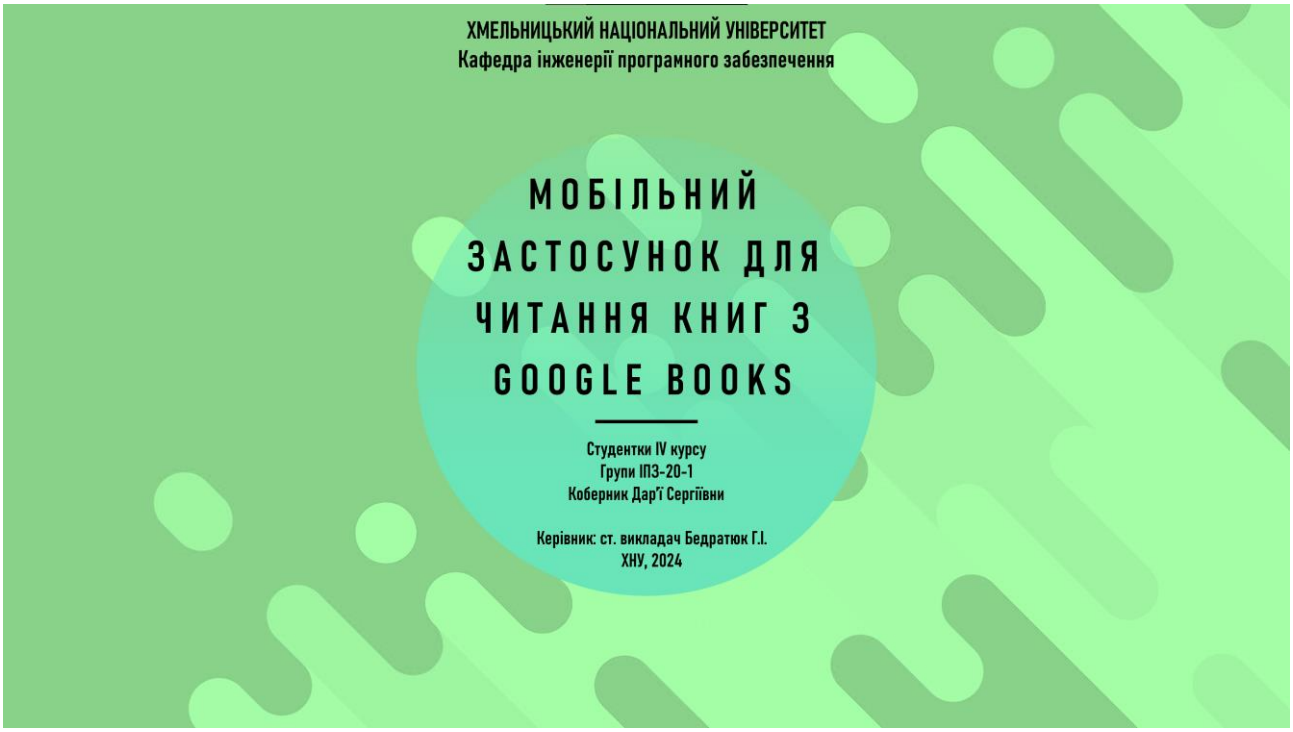
Ґ.5 Програмний код модуля GoogleSearchFragment.java

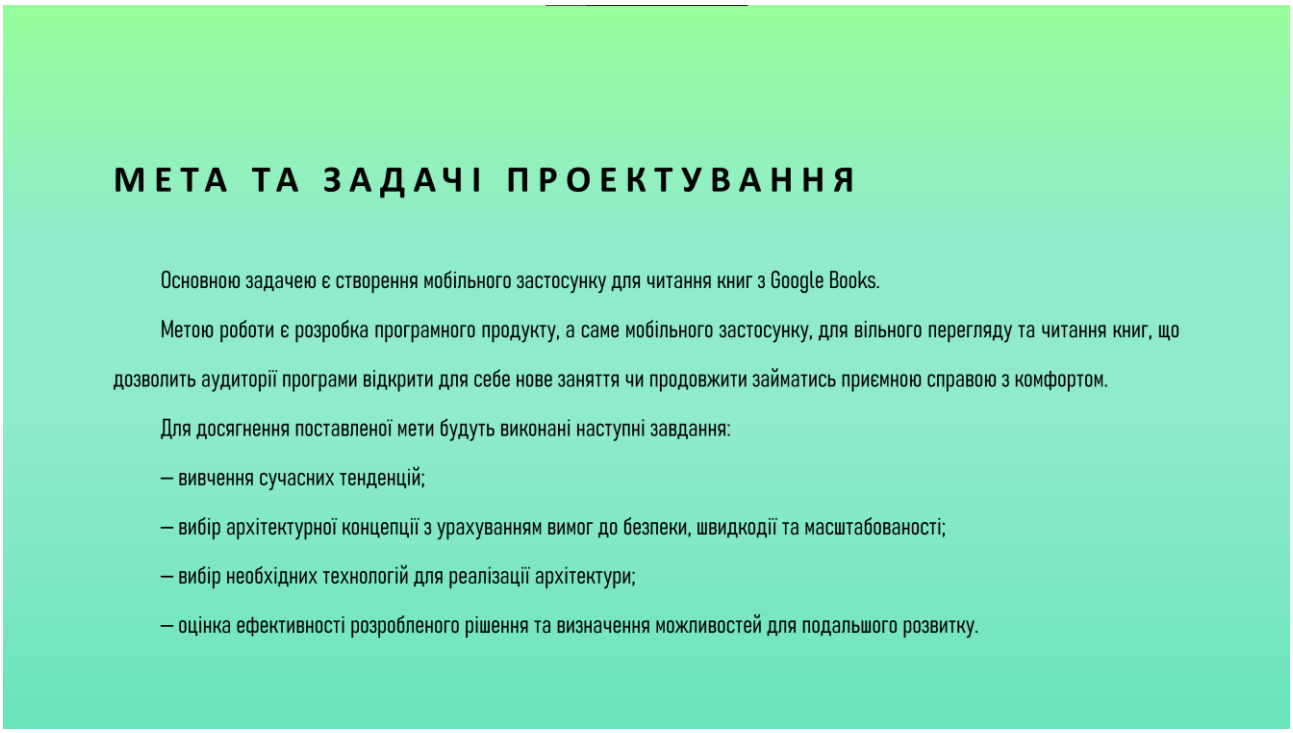
```
package com.example.myapplication;

import static androidx.constraintlayout.helper.widget.MotionEffect.TAG;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.GridLayout;
import android.widget.TextView;
```

```
import com.example.myapplication.SupportClasses.BookAdapter;
import com.example.myapplication.SupportClasses.BookInfo;
import com.example.myapplication.SupportClasses.States;

import java.util.ArrayList;

public class GoogleSerachFragment extends BaseFragment {

    private ViewGroup gridbox;

    public GoogleSerachFragment() {
    }

    public static GoogleSerachFragment newInstance() {
        GoogleSerachFragment fragment = new GoogleSerachFragment();
        return fragment;
    }


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        android.view.View view =
inflater.inflate(R.layout.fragment_google_serach, container, false);
        gridbox = view.findViewById(R.id.google_book_grid);
        return view;
    }


    public void update(Context context, ArrayList<BookInfo> list) {
        gridbox.removeAllViews();
        for (int i = 0; i < list.size(); i++) {
            BookInfo book = list.get(i);
            BookButton btn = new BookButton(context, book);
            GridLayout.LayoutParams parem = new
GridLayout.LayoutParams(GridLayout.spec(GridLayout.UNDEFINED, 1f),
GridLayout.spec(GridLayout.UNDEFINED, 1f));;
            btn.setLayoutParams(parem);

            btn.setOnClickListener(v -> {
                openBook(book);
            });
            gridbox.addView(btn);
        }
    }

    void openBook(BookInfo bookInfo) {
        Intent i = new Intent(getContext(), SearchedBook.class);

        i.putExtra("book", bookInfo);

        startActivity(i);
        getActivity().overridePendingTransition(R.anim.swipe_to_up,
R.anim.no_animation);
    }
}
```

ДОДАТОК Д

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ





АКТУАЛЬНІСТЬ

Створення мобільного застосунку для читання книг з Google Books є актуальним завданням у сучасному світі, оскільки зростає популярність книг та відповідно і електронних книг, також, зручність доступу до літератури через мобільні пристрої. За допомогою такого програмного забезпечення користувачі можуть легко знаходити, читати та зберігати книги з великого асортименту доступних ресурсів Google Books, що дозволяє забезпечити гнучкий і доступний спосіб споживання контенту для читачів у будь-який час і в будь-якому місці.

АНАЛОГИ

Bookmate: книги й аудіокниги.

Велика кількість книг та аудіокниг. Хороший функціонал редагування тексту під потреби користувача. Приємний дизайн. На Букмейті є товсті романи, детективи, наук-поп та інші книжки й аудіокниги кількома мовами, серед яких багато української літератури.

Bookmory - Книжковий трекер

Цікавий застосунок, що має додаткові функції пов'язані з трекінгом часу читання. Bookmory допомагає відстежувати, читання, керувати книгами, виробити тривалу звичку до читання та краще запам'ятовувати прочитане.

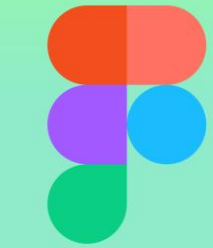
Goodreads та **Chapter1** – іноземні застосунки для читання книг. Це програми для читання книг, які містять тисячі електронних книг. У них є величезна колекція відомих і загальнодоступних книг, де можна читати необмежену кількість книг безкоштовно.




ВИБІР ЗАСОБІВ РОЗРОБКИ

Для реалізації прототипу було обрано онлайн-сервіс Figma, який дозволяє створювати динамічніта статичні прототипи додатків. У Фігмі можна відобразити елементи інтерфейсу, створити інтерактивний прототип сайту та програми, ілюстрації, векторну графіку.

Так як застосунок створюється на платформу Anroid, для розробки кодової частини було обрано програму Android Studio - це інтегроване середовище розробки (IDE), спеціально створене для розробки програмного забезпечення під платформу Android. Воно надає розширений набір інструментів і функцій, що допомагають розробникам створювати якісні, продуктивні та ефективні додатки для мобільних пристроїв.



ШАБЛОН ПРОЄКТУВАННЯ

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі).

Цей шаблон проектування зручно використовувати замість класичного MVC та йому подібних у тих випадках, коли на платформі, де ведеться розробка, присутнє «зв'язування даних». MVVM була створена з метою поділу праці дизайнера і програміста.

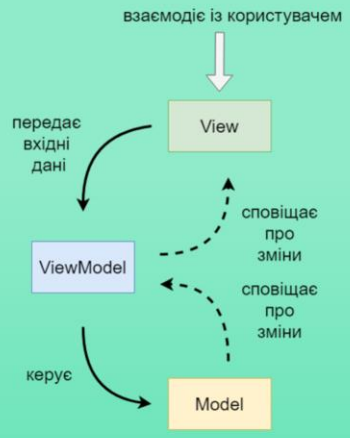
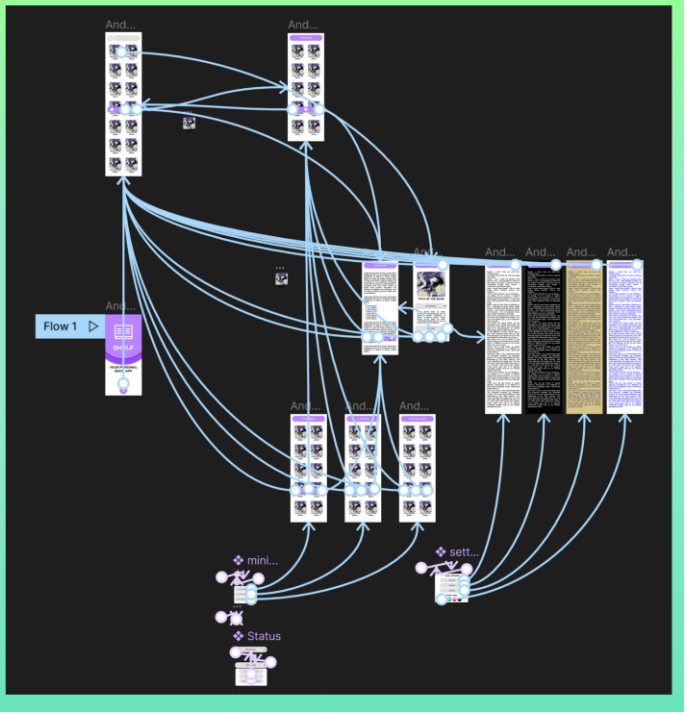


СХЕМА ПРОТОТИПУ

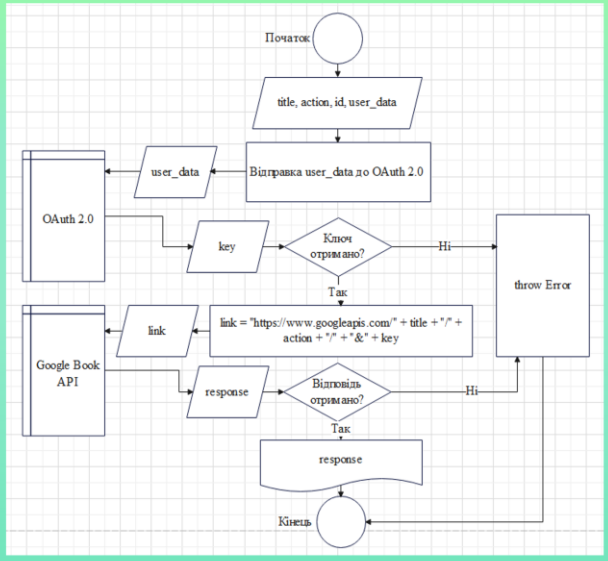
Для створення прототипу знадобилось розробити дизайн застосунку, продумати основні сторінки та їх складові. Додано різні текстові поля, кнопки та варіації виду тексту.



GOOGLE BOOK API

Мобільний застосунок для читання книг з Google, побудований на основі Book API, яка надає функціонал для прив'язки профілів до системи обробки книг в акаунті.

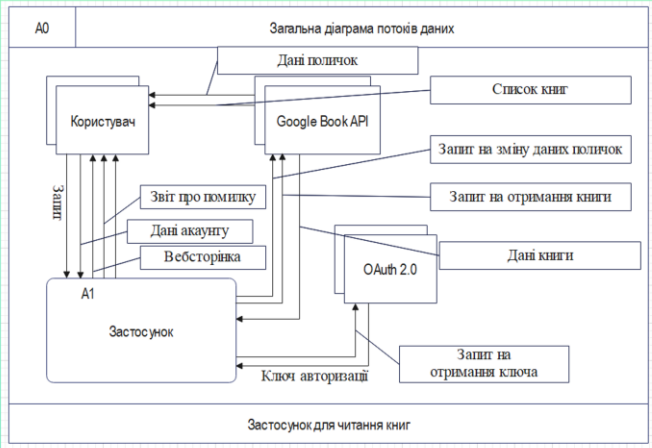
Передбачається, що на отримання інформації, відправлятимуться лише запити на отримання книг, або списків книг. В такому разі, відповіді будуть надаватись у форматі JSON і міститимуть усю інформацію про книгу, яку може надати Google.



ПОТОКИ ДАНИХ В ЗАСТОСУНКУ

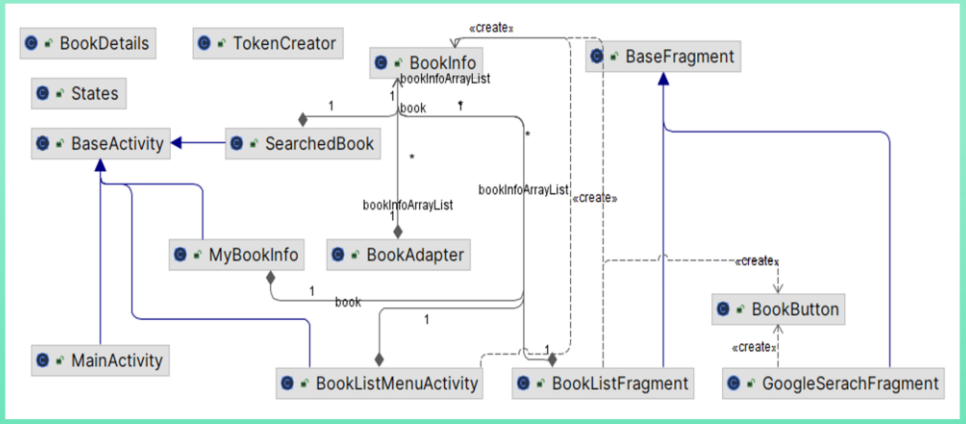
У застосунку, головною сутністю виступає «Користувач». Він може передавати дані власного акаунту, для підтвердження авторизації, що супроводжується і передачею запитів. Велика частина інформації залежить від зовнішніх систем, таких як OAuth 2.0 та Google Book API.

В даному випадку, OAuth 2.0 слугує для створення унікального ключа доступу для функціоналу Google API. В свою чергу, інша система слугує як основний механізм, навколо якого побудована система.



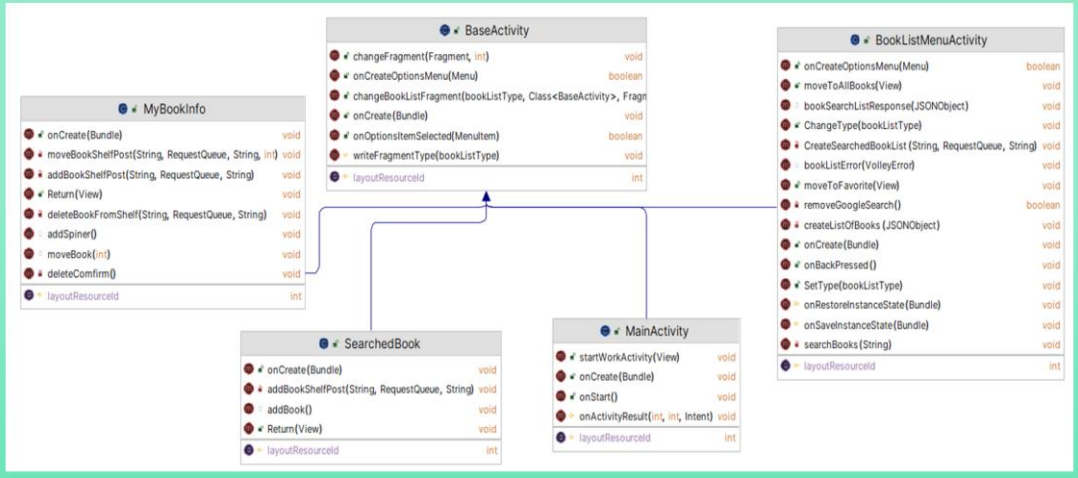
ДІАГРАМА КЛАСІВ

На зображенні можна побачити декілька класів, які використовуються в мобільному додатку для читання книг з Google Books, такі як «BookDetails», «TokenCreator», «SearchedBook», і так далі.



ДІАГРАМА КЛАСІВ АКТИВНОСТЕЙ

Діаграма класів активностей відображає організацію основних компонентів мобільного застосунку для читання книг.



SHELF

Створений програмний продукт має назву “SHELF”. Його призначення полягає у наданні користувачам зручного інтерактивного інтерфейсу, для менеджменту, покупки та читання книг, які знаходяться у акаунті Google Books.

Головна сторінка створена для початкової авторизації до додатку. На ній користувач може обрати власний Google акаунт, для якого планується виконувати менеджмент книжок.

СКРИНШОТИ ДОДАТКУ

ПУБЛІКАЦІЇ

Коберник Д.С. Мобільний застосунок для читання книг з google books: методології програмної інженерії та архітектурні рішення// Актуальні проблеми комп'ютерних наук АПКН-2023: Збірник наукових праць за матеріалами XV Всеукраїнської наук.-практ. конф., м. Хмельницький, 17-18 листопада 2023 р. Хмельницький, 2023. С. 133–136.


Міністерство освіти і науки України
Хмельницький національний університет

СЕРТИФІКАТ

АПКН 2023
АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК

Коберник Дар'я Сергіївна

учасник XV Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2023»
24 години участі (0,8 ECTS credits)

Голова оргкомітету АПКН-2023

СИНЮК Олег Миколайович
проректор Хмельницького національного університету з наукової роботи,
доктор технічних наук, професор

м. Хмельницький
17-18 листопада 2023

E-mail: apkt.khnu@gmail.com

ВИСНОВКИ

В результаті було отримано функціонуючий мобільний застосунок на платформу Android.

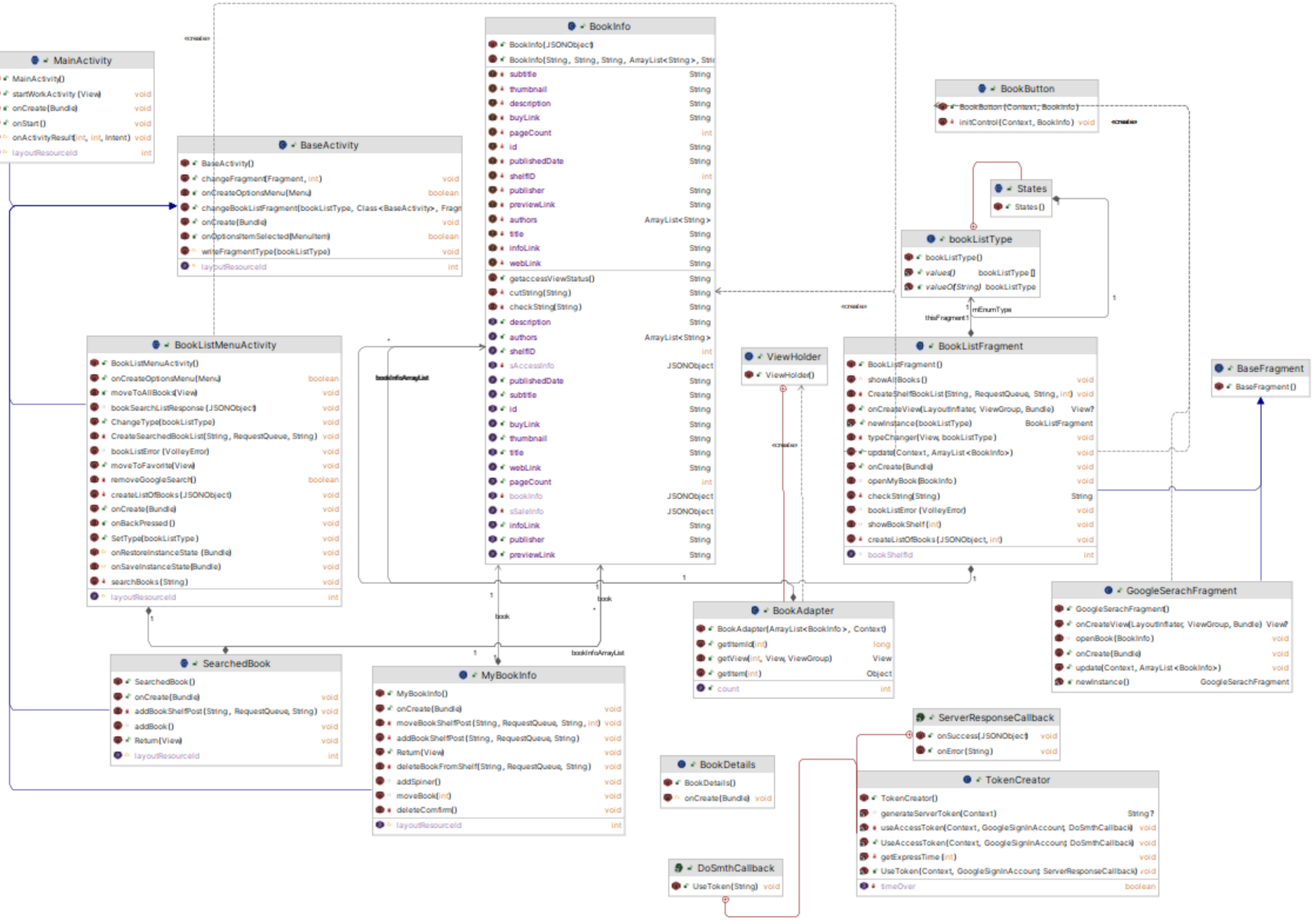
Для створення даного застосунку було виконано різні поставлені завдання. В першу чергу розглянуто сучасні тенденції та взято до уваги існуючі аналоги.

Обрано шаблон проектування, а саме MVVM, що є одним з найкращих рішень для розробки мобільного застосунку такого типу. Для реалізації шаблону проектування обрано програми Figma та Android Studio, що мають великий функціонал та дуже підходять для створення прототипу та його кодингу.

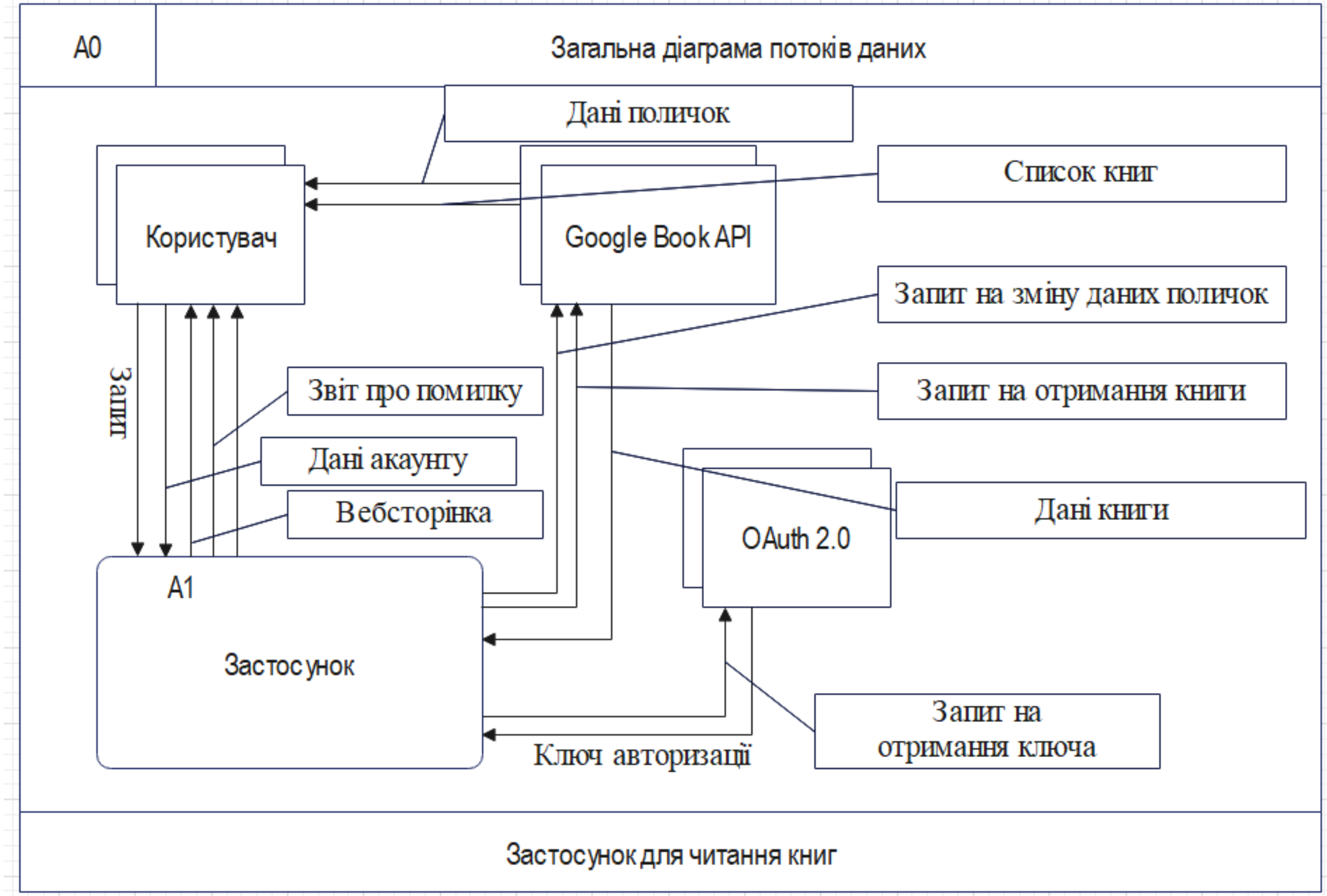
Після створення дизайну, прототипу та розробки було проведено тестування та перевірено весь наявний функціонал у застосунку. В узагальненому вигляді, цей застосунок є результатом поєднання новітніх підходів у програмній інженерії з глибоким розумінням потреб користувачів. Цей застосунок надає зручний доступ до широкого асортименту книг, що дозволяє користувачам з легкістю знаходити, читати та зберігати книги на своїх пристроях.

ДЯКУЮ ЗА УВАГУ!

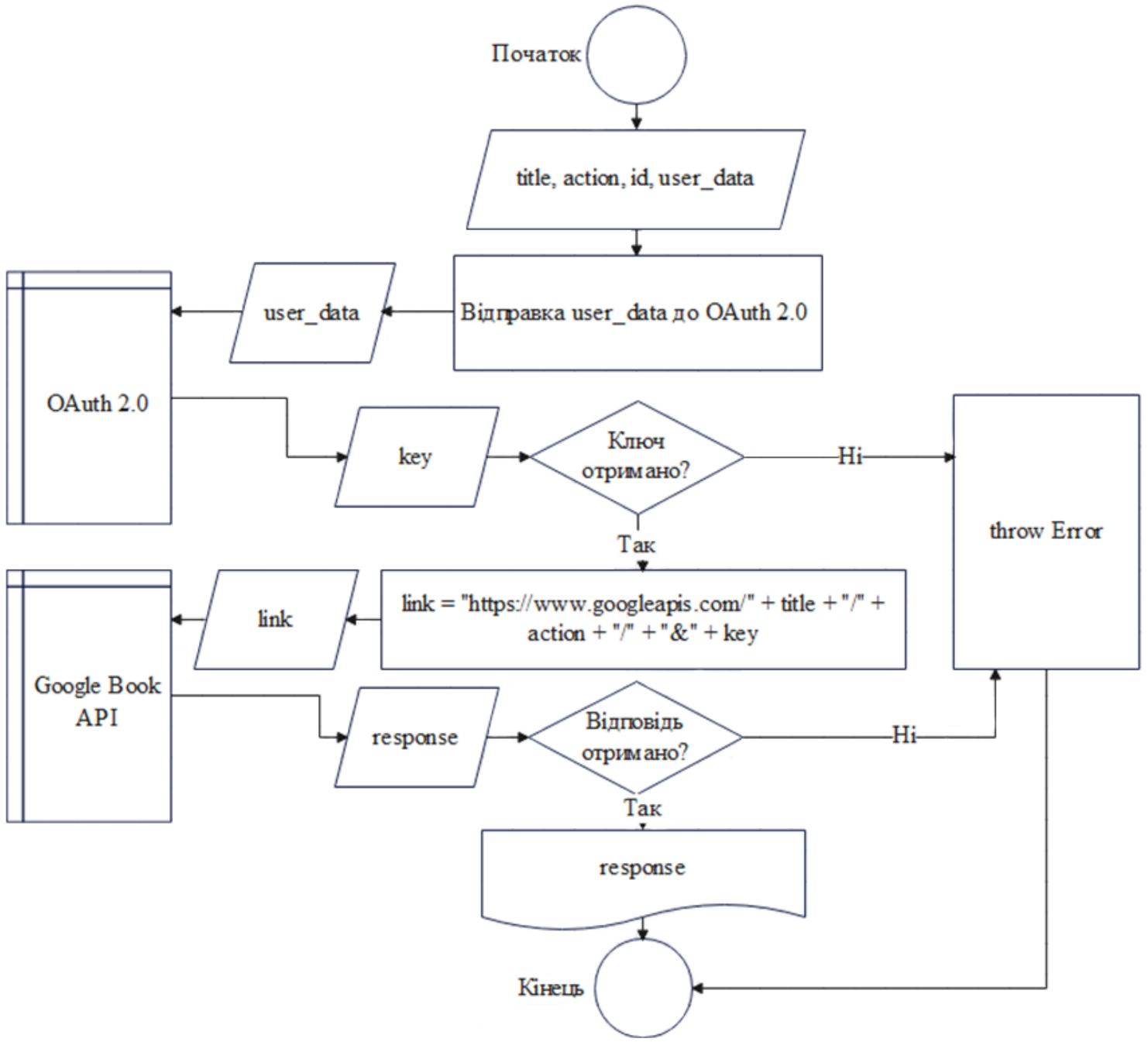
ГРАФІЧНА ЧАСТИНА



					КПІПЗ.200165.01.10.ПЗ	
Зм.	Арк.	№ докум.	Підпис	Дата	Мобільний застосунок для читання книг з Google Books Відомість документів	Літера / Маса / Масштаб
Розробив		Коберник Д.С.				
Керівник		Бедратюк Г.І.				Аркуш / Аркушів
Консульт.						
Н. Контр.					Розкрита діаграма класів	
Зав. каф.		Бедратюк Л.П.				



					КПІПЗ.200165.01.10.ПЗ	
Зм.	Арк.	№ докум.	Підпис	Дата	Мобільний застосунок для читання книг з Google Books Відомість документів	Літера / Маса / Масштаб
Розробив		Коберник Д.С.				
Керівник		Бедратюк Г.І.				Аркуш / Аркушів
Консульт.						
Н. Контр.					Загальна діаграма потоків даних в застосунку	
Зав. каф.		Бедратюк Л.П.				



					КПІПЗ.200165.01.10.ПЗ			
					Мобільний застосунок для читання книг з Google Books Відомість документів	Літера	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Коберник Д.С.						
Керівник		Бедратюк Г.І.				Аркуш	Аркушів	
Консульт.								
Н. Контр.					Блок-схема відправки запиту до Google Book API			
Зав. каф.		Бедратюк Л.П.						

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Коберник Д.С.
Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-20-1

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-технічних засобів (Unicheck та/або Anti-Plagiarism) 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.

02.01.2024
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Виявлено в документах: 11%**

ID: 128669 Назва: БКР Мобільний застосунок для читання книг з Google Books Додано в БД: 2024-06-03 Автори: Кобзєнов Д.С. Керівники: Бедратов Г.І. ст. викладач Консультанти: Оновити:	Документ		Сумарний збіг по базі Даних	
	Слова/цифри	Лексеми	Слова/цифри	Лексеми
	115757	969	2378 (2%)	31 (3%)

Джерело плагіату

ID	Опис	Найбільше плагіату в документі	
		Слова/цифри	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
ІПЗ

ID перевірки:
1016313966

Дата перевірки:
03.06.2024 11:48:57 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2024 22:21:02 EEST

ID користувача:
100012953

Назва документа: БКР_Мобільний застосунок для читання книг з Google Books_Коберник Д.С., Бедратюк Г.І

Кількість сторінок: 78 Кількість слів: 16905 Кількість символів: 134014 Розмір файлу: 2.62 MB ID файлу: 1016111099

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.54%
Схожість

Найбільша схожість: 0.5% з джерелом з Бібліотеки (ID файлу: 1008265198)

4.96% Джерела з Інтернету | 765 ..... Сторінка 80

1.83% Джерела з Бібліотеки | 150 ..... Сторінка 84

0.07% Цитат

Цитати | 3 ..... Сторінка 85

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Заміжні символи | 2

Підозріле форматування | 12 сторінок

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «Бакалавр»

Дипломник Коберник Дар'я Сергіївна

Тема Мобільний застосунок для читання книг з Google Books

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень _____; кількість сторінок записки _____

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі проведено ретельний аналіз предметної області, визначено всі функціональні та нефункціональні вимоги. Було здійснено порівняльний аналіз існуючих програмних рішень на ринку, зокрема їх переваг та недоліків, що підтвердило необхідність розробки нового програмного забезпечення. Вибір інструментів для реалізації програмного рішення був здійснений на основі їхньої відповідності поставленим завданням. Після розробки програмне забезпечення було протестовано, що підтвердило його коректну роботу та готовність до впровадження.

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 обґрунтовано актуальність теми, визначено мету та завдання дипломного проектування. Перший розділ присвячено аналізу предметної області, розгляду існуючих рішень та визначенню функціональних і нефункціональних вимог до розроблюваного програмного забезпечення. У другому розділі розглянуто сучасні підходи до покращення користувацького досвіду та архітектурні моделі, зокрема патерн MVVM. Спроектовано всі компоненти застосунку та їхні зв'язки. Третій розділ містить практичну розробку програмних модулів та їхній опис, включаючи тестування системи, яке підтвердило її коректну роботу.

4. Позитивні сторони роботи Тема кваліфікаційної роботи є актуальною, оскільки забезпечує користувачів зручним доступом до цифрових книг. Застосовано новітні технології та сучасні архітектурні рішення, що забезпечують високу функціональність та зручність використання програми.

5. Негативні сторони роботи Робота обмежується лише основними функціями, такими як пошук та читання книг. Більш розширений функціонал, наприклад, інтеграція з іншими сервісами або підтримка різних форматів книг, може бути доданий у майбутньому.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та представлено у вигляді діаграм і рисунків. Пояснювальна записка підготовлена згідно з вимогами чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує на високу оцінку. Матеріал, що поданий у пояснювальній записці є структурованим, логічним та цілком зрозумілим. Це сприяє чіткому розумінню викладеного матеріалу в контексті теми проектування. Графічний матеріал наочно демонструє деталі проектування системи та ширше показує аспекти розробки.

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) Говорущенко Т.О., к.т.н., професор, завідувачка кафедри КІС ХНЕУ

«07» 06 2024 р. _____
(підпис)

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 читання книг з Google Books»

Автор: Коберник Дар'я Сергіївна
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Бедратюк Ганна Іванівна
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота 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. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є академічним плагіатом, але 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 Unicheck виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів, у рамках основних написів, у назвах публікацій переліку джерел посилання;

2) запозичення, виявлені в тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 1.0%. Обсяг запозичень, визначений системою Unicheck виявлення збігів ідентичності/схожості, складає 5.54% і адресується до 765 джерел з Інтернету і 150 джерел з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 6.06.2024р.

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Ганна БЕДРАТЮК