Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200124.01.07.ПЗ

Виконав студент III курсу, група ІПЗс-20-1 _____ Савич Н. В.
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ О.Г. Онишко
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. тех. наук, доцент _____ І.В. Гурман
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

2 червня 2023 р.

Хмельницький 2023

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
02 01 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Савичу Ніколі Властимировичу
Прізвище, ім'я, по батькові студента

1. Тема кваліфікаційної роботи Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях

Керівник кваліфікаційної роботи Онишко Оксана Григоріївна, канд. пед. наук., доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2023 р. № 5
2. Строк подання студентом роботи на кафедру 01.06.2023 р.
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі,
проектування програмного забезпечення
програмна реалізація
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення: UML - діаграма варіантів використання, Схема авторизації через JWT токен, схема передачі мультимедійних файлів через мережу

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Гурман І. В., доцент кафедри ІПЗ		
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ		

7. Дата видачі завдання « 05 » лютого 2023р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи (КР), визначення та узгодження індивідуальних тем КР	01.12– 31.12.2022	
2 Збір матеріалу за темою КР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2023	
3 Проектування програмного забезпечення	01.02 – 28.02 2023	
4 Програмна реалізація з використанням відповідних засобів розробки	01.03 – 10.04.2023	
5 Тестування програмного забезпечення	11.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КР згідно вимог	01.05 – 25.05.2023	
7 Попередній захист КР	Травень 2023 (згідно графіка)	
8 Перевірка КР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2023	
9 Здача КР на кафедру; підготовка КР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КР	з 01.06.2023	

Студент _____ Н.В. Савич
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О.Г. Онишко
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях.

Автор роботи: Савич Нікола Властімірович.

Керівник роботи: Онишко Оксана Григорівна.

Пояснювальна записка: 63 с.,24 рис.,2 табл., 4дод., 40 джерел.

Графічна частина: 3 креслення.

REACT, DETOX, MONGODB, NODEJS, REACT-NATIVE, SOCKET.IO

Метою кваліфікаційної роботи розробити програмний застосунок для синхронізації відтворення мультимедійних файлів на різних мобільних платформах, що має мати зрозумілий та простий інтерфейс.

Актуальність теми кваліфікаційної роботи є у необхідності наявності системи синхронізації відтворення медіаматеріалів на мобільних платформах, яка буде доступною для користувачів та легкою у використанні.

У кваліфікаційній роботі було проаналізовано предметну область задачі, з'ясовані характерні проблеми, було проведено аналіз існуючих рішень та альтернативних програмних застосунків, були визначені та спроектовані модулі для системи та здійснена їх програмна реалізація.

Для розробки мобільного застосунку була використана мова програмування JavaScript, фреймворки React Native та Node.js та база даних MongoDB.

В результаті було розроблено мобільний застосунок для синхронізації відтворення мультмедійних матеріалів на різних пристроях.

01.06.2023
Дата


Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200124.01.07.ПЗ	Пояснювальна записка			
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А3	КвРІПЗ.200124.01.07.ПЗ	UML - Діаграма варіантів	1		
			використання			
5	А3	КвРІПЗ.200124.01.07.ПЗ	Схема авторизації через	1		
			JWT токен			
6	А3	КвРІПЗ.200124.01.07.ПЗ	Схема передачі мульти -	1		
			медійних файлів			
			через мережу			

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200124.01.07.ВД		
Виконав.		Савич Н.В		1.06	Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях Відомість документів	Літ.	Арк. 1 / Акрушів 1
Керівник.		Онишко О.Г.		1.06			
Рецензент						ХНУ, ІПЗс-20-1	
Н. Контр.		Гурман І.В.		2.06			
Затверд.		Бедратюк Л.П.		2.06			

ЗМІСТ

					КвРІПЗ.200124.01.07.ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Виконав.		Савич Н.В		1.06	Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях	Літ.	Арк. 5 / Акрушів 104
Керівник.		Онишко О.Г.		1.06			
Рецензент						ХНУ, ІПЗс-20-1	
Н. Контр.		Гурман І.В.		2.06			
Затверд.		Бедратюк Л.П.		2.06			

ВСТУП

Ринок програмного забезпечення сьогодення значною мірою орієнтований на мобільні застосунки. З поступовим розвитком великих мобільних платформ, збільшувалася необхідність адаптацій різних можливостей персональних комп'ютерів для використання з мобільного пристрою. Якщо раніше спроби цих адаптацій могли представляти собою лише імітацію основного функціоналу, або обрізані версії звичного програмного забезпечення, то з виходом на ринок принципового нового виду мобільної техніки з'явилася можливість не тільки повноціно віддтворювати функціональність абсолютної більшості програмних застосунків загальної направленості.

Цим новим типом мобільної техніки виявився смартфон. З появою смартфона, вперше вдалося помістити достатньо потужні апаратні складові, такі як швидкісні процесори, обсяг оперативної пам'яті, який відповідає деяким стаціонарним машинам та якісні дисплеї. Це дозволило значно полегшити розробку програмного забезпечення для смартфонів. Використання більших обчислювальних можливостей, відкрило шлях до реалізації широкого спектру програмних рішень, що раніше впиралось в апаратні обмеження мобільних платформ.

Іншою важливою інновацією в рамках появи смартфонів, стала сенсорна модель управління. Хоча можливість керувати пристроями додитоком не була новою на той час, саме з появою смартфонів вона стала довершеною. Можливість керування жестами та додитиком, доповнювалась гармонійним дизайном користувацьких інтерфейсів, що орієнтувалися на зручність інтеракції. Це призвело до створення нової ергономічної парадигми керування, яка дозволяла використувати екран одначасно як засіб виведення, так і введення інформації, а можливість використання жестів гармонійно доповнювала її через можливість використання замість звичного покажчика.

Таким чином поява смартфона відкрила шлях до реплікації більшості застосунків стаціонарних машин на мобільному пристрої в тому чи іншому виді, чим і займалися різні компанії останніх десять років.

Не зважаючи на це досі існують програмні продукти, які не адаптовані під мобільні пристрої. Одною із предметних областей, які не мають альтернативної реалізації для мобільних платформ є застосунки для синхронізації мультимедійних матеріалів. В сучасному світі відчувається необхідність для синхронізованого відтворення презентацій, аудіо, або відеофайлів саме на мобільних платформах. Це може бути застосовано на сам перед в сферах освіти та науки, для проведення дистанційних семінарів, або лекцій. Крім того подібна технологія може бути використана для дозвілля.

Актуальність теми дипломного проекту полягає у необхідності наявності системи синхронізації відтворення медіаматеріалів на мобільних платформах, яка буде доступною для користувачів та легкою у використанні.

Мета проекту розробити програмний застосунок для синхронізації відтворення мультимедійних файлів на різних мобільних платформах, що має мати зрозумілий та простий інтерфейс.

Розробка відповідного програмного продукту може бути корисна в різних сферах включаючи освіту та дозвілля. Програмне забезпечення буде відповідати технічним вимогам.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Наведемо деякі структурні особливості предметної області.

Мультимедійні файли: застосунок буде працювати з різними типами мультимедійних файлів, такими як відео, аудіо, зображення тощо. Він повинен забезпечувати зручну навігацію і відтворення цих файлів на різних платформах.

Мобільні платформи: застосунок буде розроблятися для різних мобільних платформ, таких як Android і iOS. Кожна платформа має свої особливості і вимоги щодо розробки, тому додаток повинен бути адаптований до цих платформ для забезпечення максимальної сумісності та продуктивності.

Синхронізація відтворення: Однією з ключових функціональних особливостей застосунку є можливість синхронізувати відтворення мультимедійних файлів на різних пристроях. Це означає, що користувачі зможуть почати перегляд або прослуховування файлу на одному пристрої і продовжити його на іншому, зберігаючи поточну позицію відтворення.

Зрозумілий та простий інтерфейс: застосунок повинен мати інтуїтивно зрозумілий і простий інтерфейс, що дозволить користувачам легко навігувати по функціях застосунку, вибирати та відтворювати мультимедійні файли. Простота використання є важливим аспектом для забезпечення задоволення користувачів та залучення більш широкої аудиторії.

До функціональних особливостей предметної області можна віднести наступні:

– Відтворення мультимедійних файлів: застосунок повинен забезпечувати функцію відтворення різних типів мультимедійних файлів на мобільних пристроях. Це включає у себе контроль програвання, паузу, перемотування, регулювання гучності тощо.

– Синхронізація відтворення: застосунок повинен мати можливість синхронізувати відтворення мультимедійних файлів між різними пристроями. Це означає, що користувачі зможуть почати перегляд або прослуховування файлу на одному пристрої і продовжити його на іншому без втрати поточного стану.

– Управління мультимедійними матеріалами: застосунок може надавати функціональність управління мультимедіа матеріалами, дозволяючи користувачам створювати, редагувати та видаляти свої власні мультимедійні файли.

– Підтримка різних форматів файлів: застосунок повинен підтримувати широкий спектр мультимедійних форматів файлів, таких як MP3, MP4, AVI, JPEG тощо, для забезпечення сумісності з різними типами вмісту.

– Навігація та пошук: застосунок може містити функції навігації та пошуку, що дозволяють користувачам швидко знаходити потрібні мультимедійні файли або створювати власні категорії та мітки для зручного управління вмістом.

Цей змістовний аналіз предметної області допоможе розуміти основні аспекти та вимоги до розробки програмного застосунку для синхронізації відтворення мультимедійних файлів на різних мобільних платформах. Враховуючи ці особливості, можна зосередитися на ключових функціях та розробити інтуїтивно зрозумілий і простий у використанні інтерфейс для задоволення потреб користувачів.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Watch2Gather – це платформа або сервіс, який надає можливість групового перегляду відео або стрімів у режимі реального часу. Цей сервіс дозволяє користувачам віддалено дивитися відео разом з друзями, родичами або колегами, незалежно від їх географічного розташування.

Watch2Gather забезпечує синхронізацію відтворення відео між усіма учасниками групи, тобто всі учасники дивляться одну й ту ж саму точку відео одночасно. Це створює враження спільного перегляду навіть на віддаленій відстані.

Основні особливості Watch2Gather можуть включати можливість створення приватних або публічних груп для спільного перегляду, чату або обміну повідомленнями під час перегляду, можливість вибору вмісту з різних джерел (стріми, відеофайли, платформи для спільного перегляду відео тощо) і інші функції, які роблять процес спільного перегляду більш інтерактивним і зручним для користувачів.

Watch2Gather може бути використаний для різних цілей, включаючи спільний перегляд фільмів, серіалів, спортивних подій, відеоігор, навчальних матеріалів та багато іншого. Цей сервіс створює віртуальне спільноту переглядачів, які можуть насолоджуватися вмістом разом і взаємодіяти одне з одним під час перегляду. З інтерфейсом під час синхронного перегляду можна ознайомитись нижче, на рисунку 1.1

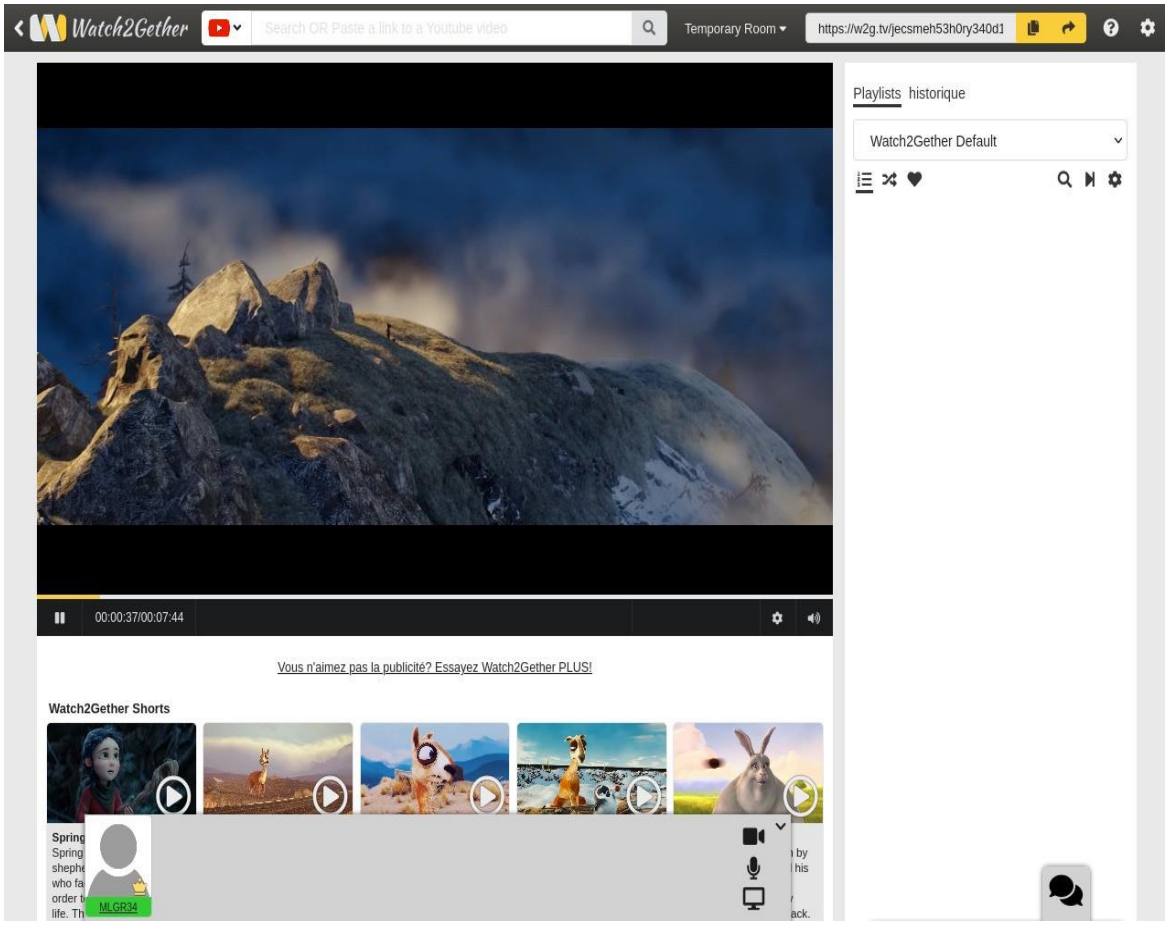


Рисунок 1.1 – Інтерфейс Watch2Gather – https://w2g.tv/en/

Платформа Watch2Gather має декілька основних переваг:

- Груповий перегляд: Watch2Gather дозволяє користувачам віддалено дивитися відео або стріми разом зі своїми друзями, родичами або колегами. Це створює відчуття спільного перегляду навіть на відстані, надаючи можливість обговорювати вміст та ділитися емоціями в реальному часі.
- Синхронізоване відтворення: Watch2Gather забезпечує синхронізацію відтворення відео між усіма учасниками групи. Всі учасники дивляться одну й ту саму точку відео одночасно, що створює спільне досвід перегляду і спілкування.
- Варіативність контенту: Платформа дозволяє вибирати вміст з різних джерел, включаючи стріми, відеофайли та платформи для спільного перегляду відео. Це дозволяє користувачам вибирати свої улюблені відео або спільно дивитися актуальні події.
- Інтерактивність: Watch2Gather може надавати функції чату або обміну повідомленнями під час перегляду відео. Це дозволяє учасникам спілкуватися, обговорювати вміст та виражати свої думки та реакції під час перегляду.
- Гнучкість і зручність: Watch2Gather надає можливість вибору приватних або публічних груп для спільного перегляду, а також зручний інтерфейс для навігації та управління вмістом. Користувачі можуть насолоджуватися вмістом разом з іншими, вибираючи зручний спосіб перегляду.
- Ці переваги роблять платформу Watch2Gather привабливою для тих, хто шукає спосіб спільного перегляду відео і спілкування з віддаленими друзями та рідними, або для організації віртуальних переглядів спортивних подій, фільмів, серіалів та інших мультимедійних вмісту.

Незважаючи на свою корисність і популярність, платформа Watch2Gather має деякі недоліки:

- Залежність від стабільного і швидкого Інтернет-з'єднання: Для успішного використання Watch2Gather необхідне надійне та швидке Інтернет-з'єднання для всіх учасників. Відсутність стабільного з'єднання може призвести до переривання або затримки синхронізації відтворення, що негативно впливає на користувацький досвід.

- Обмеження на вибір контенту: Watch2Gather може обмежувати доступний вміст, особливо якщо він працює з певними платформами чи провайдерами контенту. Це може обмежити різноманітність доступних стрімів або відео, які користувачі можуть спільно переглядати, так як не дозволяє користувачам завантажити власний матеріал для перегляду. Крім того, доступні платформи росповсюджують лише відео, тому відсутня можливість перегляду аудіо та презентаційних матеріали.
- Проблеми з приватністю і безпекою: Використання платформи Watch2Gather може викликати питання щодо приватності та безпеки користувачів. Збирання та обробка особистої інформації, яка може здійснюватись сервісом, може створювати ризики для конфіденційності даних користувачів.
- Відсутність розширеної функціональності: Хоча Watch2Gather забезпечує основні функції спільного перегляду відео, він може бути обмежений в розширених можливостях або інтерактивності. Наприклад, відсутність можливості одночасної відеоконференції або спільного контролю відтворення для всіх учасників.
- Обмежена підтримка платформ: Watch2Gather може мати обмежену підтримку для певних мобільних платформ або пристроїв. Це може призвести до обмеження доступу для деяких користувачів, які використовують інші пристрої або операційні системи. Сайт не є оптимізованим під стандартні екрани телефонів та планшетів

SyncPlay - це безкоштовний програмний застосунок для синхронізації відтворення відеофайлів на різних пристроях.

Він дозволяє користувачам віддалено дивитися відео разом з іншими учасниками, забезпечуючи синхронізацію відтворення і забезпечуючи однакову позицію відтворення для всіх учасників.

Основна ідея SyncPlay полягає в тому, що усі учасники вводять однаковий код або IP-адресу сервера, на якому розміщений відеофайл. Після цього програма автоматично синхронізує відтворення відео, що дозволяє всім учасникам

переглядати одну й ту ж саму точку відео одночасно. З інтерфейсом SyncPlay можна ознайомитись нижче, на рисунку 1.2

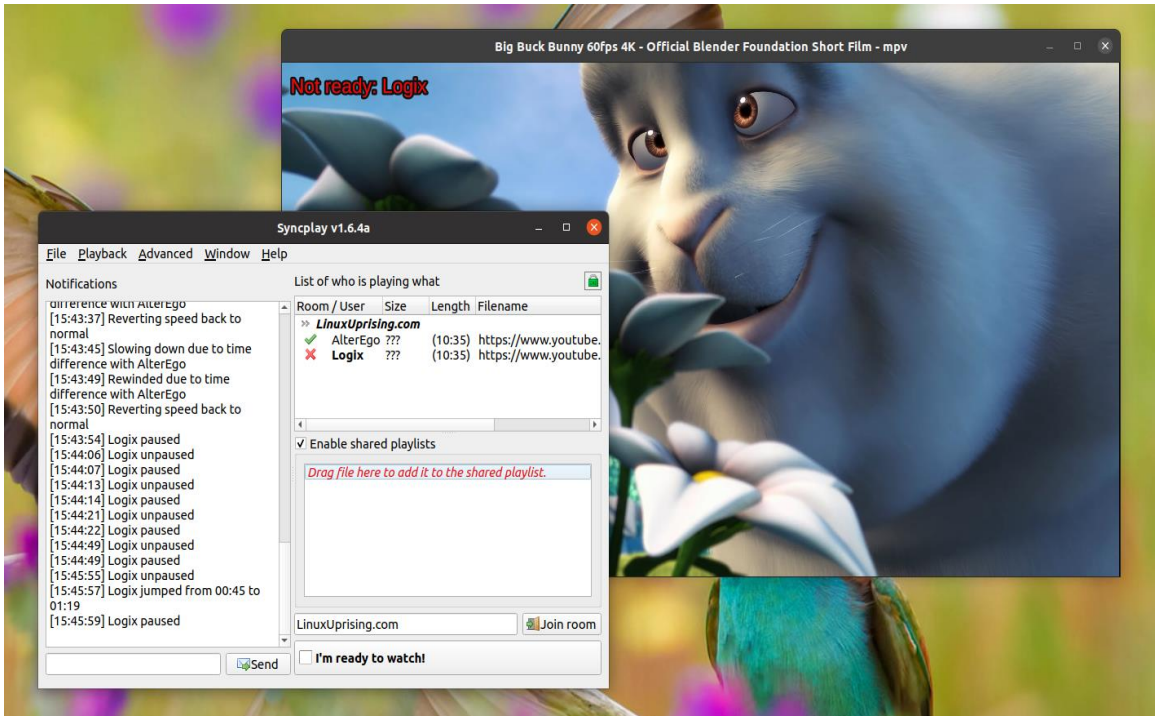


Рисунок 1.2 – Інтерфейс SyncPlay – https://syncplay.pl

Основні переваги SyncPlay включають:

– Груповий перегляд: SyncPlay дозволяє декільком користувачам віддалено дивитися відео разом, незалежно від їх географічного розташування. Це створює відчуття спільного перегляду, під час якого користувачі можуть обговорювати вміст і спілкуватися один з одним.

– Синхронізація відтворення: Програма автоматично синхронізує відтворення відео між усіма учасниками, щоб вони всі дивилися одну й ту саму точку відео одночасно. Це дозволяє уникнути незручностей, пов'язаних зі затримками або невідповідністю відтворення.

– Різні платформи: SyncPlay підтримує різні операційні системи, включаючи Windows, macOS і Linux. Це дозволяє користувачам використовувати програму на своїх улюблених пристроях та платформах.

– Безкоштовність та відкритий код: SyncPlay є безкоштовним програмним забезпеченням з відкритим вихідним кодом, що означає, що ви можете використовувати його безкоштовно та налаштовувати його за своїми потребами.

SyncPlay є корисним інструментом для тих, хто хоче дивитися відео разом з віддаленими друзями, організовувати віртуальні перегляди фільмів або спільно вивчати відеоуроки.

Незважаючи на свою корисність, SyncPlay також має деякі недоліки, серед яких можна виділити наступні:

- Залежність від стабільного з'єднання з Інтернетом: SyncPlay вимагає стабільного та достатньо швидкого з'єднання з Інтернетом для синхронізації відтворення відео. При поганому з'єднанні можуть виникати проблеми зі синхронізацією, затримками або недостатньою якістю відтворення;
- Обмеженість підтримуваних медіаплеєрів: SyncPlay працює з певними медіаплеєрами, які підтримують протокол синхронізації SyncPlay. Це означає, що ви повинні використовувати сумісний медіаплеєр для коректної роботи програми. Це може бути обмеженням для користувачів, які вже мають свої улюблені медіаплеєри та не хочуть переключатися на новий;
- Обмежена функціональність: SyncPlay зосереджений на синхронізації відтворення відео та чаті між учасниками, тому він може бути обмежений в інших функціях, які можуть бути доступні в інших програмах для спільного перегляду відео. Наприклад, він може не мати розширеного набору інструментів для керування відео, підтримки підписів, ефектів тощо;
- Необхідність встановлення на кожному пристрої: Для синхронного перегляду відео всі учасники повинні мати встановлений SyncPlay на своїх пристроях. Це може вимагати часу та зусиль для організації групового перегляду, особливо якщо деякі учасники не знаходяться в одному місці або не мають встановленого застосунку;
- Потенційні проблеми з безпекою: З огляду на використання Інтернету для синхронізації, існує ризик потенційних проблем з безпекою, таких як можливість перехоплення даних або вразливості програми. Тому важливо дотримуватись заходів безпеки при використанні SyncPlay та забезпечити безпечне з'єднання з Інтернетом;

– Недоступність для мобільних платформ: Незважаючи на велику кількість підримуванних операційних систем, до їх складу не входять найпопулярніші мобільні платформи Android та IOS, що в свою чергу унеможливлює використання застосунку абсолютною більшістю користувачів;

1.3 Аналіз вимог до програмного забезпечення та розробка технічного завдання

Технічне завдання для розробки мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях може включати наступні складові. Поставлена проблема являє собою потребу у зручному способі синхронізації відтворення мультимедійних матеріалів на різних пристроях.
Розробка мобільного застосунку, який забезпечить синхронізацію відтворення мультимедійних матеріалів на різних пристроях. Провівши аналіз предметної області та постановки завдання можна виділити наступні вимоги:

– Підтримка платформ: Застосунок повинен працювати на основних мобільних платформах, таких як iOS і Android;

– Сумісність: Застосунок повинен бути сумісним з різними пристроями, включаючи смартфони, планшети, тощо;

– Мультимедійні формати: Застосунок повинен підтримувати різні мультимедійні формати, такі як відео (MP4, AVI, MKV), аудіо (MP3, WAV, AAC) та зображення (JPEG, PNG, GIF);

– Синхронізація: Застосунок повинен забезпечувати точну синхронізацію відтворення мультимедійних матеріалів на різних пристроях, зокрема можливість паузи, перемотування і відтворення з певного часового моменту;

– Мережева підтримка: Застосунок повинен підтримувати безперервний доступ до Інтернету для синхронізації відтворення через мережу;

– Інтерфейс: Застосунок повинен мати зрозумілий та інтуїтивно зрозумілий інтерфейс користувача для навігації та керування відтворенням мультимедійних матеріалів.

Функціональні вимоги можуть включати наступні:

– Реєстрація користувача: Застосунок повинен мати можливість реєстрації користувачів з використанням електронної пошти або облікових записів соціальних мереж.

– Аутентифікація користувача: Застосунок повинен забезпечувати аутентифікацію користувача перед доступом до його особистого облікового запису.

– Вибір мультимедійних матеріалів: Користувач повинен мати можливість вибрати мультимедійні матеріали для синхронізації відтворення.

– Створення груп: Користувач повинен мати можливість створювати групи і запрошувати інших користувачів для синхронного відтворення мультимедійних матеріалів.

– Синхронізація відтворення: Застосунок повинен забезпечувати синхронізацію відтворення мультимедійних матеріалів між різними пристроями в групі.

– Керування відтворенням: Користувач повинен мати можливість керувати відтворенням мультимедійних матеріалів, включаючи паузу, перемотування та відтворення з певного часового моменту.

– Спільний доступ: Користувач повинен мати можливість надавати спільний доступ до своїх мультимедійних матеріалів іншим користувачам.

Нефункціональні вимоги полягують в наступному:

– Безпека: Застосунок повинен забезпечувати безпеку персональних даних користувачів, використовуючи шифрування та заходи для запобігання несанкціонованому доступу.

– Продуктивність: Застосунок повинен працювати швидко та ефективно, забезпечуючи мінімальні затримки в синхронізації відтворення.

– Надійність: Застосунок повинен бути надійним і стабільним, мінімізуючи відмови та збої в роботі.

– Сумісність з різними пристроями: Застосунок повинен бути розроблений з урахуванням різних розмірів екранів, роздільних здатностей та можливостей різних пристроїв.

Провівши аналіз наведених функціональних вимог можна скласти Діаграму варіантів використання:

Таблиця 1.1 – Опис акторів розроблюваного ПЗ

Актор	Опис
Користувач	Цей актор представляє собою користувача який не є адміністратором групи і кімнати перегляду медіа матеріалів тому йому доступна мінімальна кількість функцій, такі як: Підняти руку, проголусувати за паузу, та редагувати свої персональні дані в профілі
Адміністратор	Відповідає за управління відтворенням мультимедіа матеріалів, може керувати файлами та адмініструвати користувачів в своїй кімнаті або групі для перегляду

Діаграма варіантів використання представляє звязки між сутностями прецедентів та акторів. Актори - це зовнішні сутності (користувачі, системи, пристрої), що взаємодіють з системою. Прецеденти – це деякі конкретні операції, які можуть бути виконані системою від імені акторів.

Діаграма варіантів використання дозволяє наглядно відобразити функціональність системи та її взаємодію з різними групами користувачами, зображає потреби та вимоги користувачів і часто використовується для подальшого аналізу та проектування. Розроблена діаграма варіантів використання представлена нижче на рисунку 1.3.

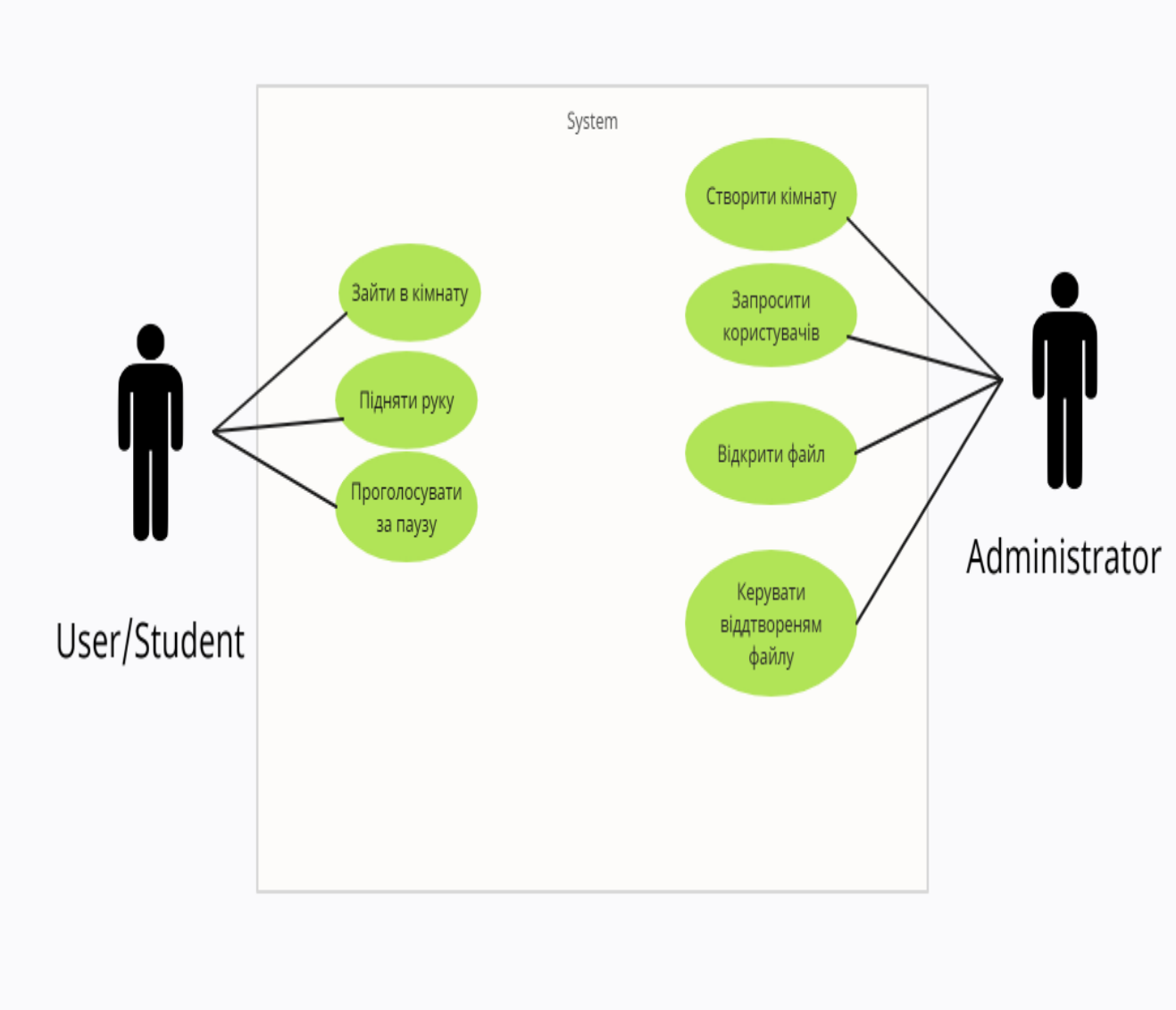


Рис 1.3 – Діаграма варіантів використання.

Після аналізу вимог до застосунку, було розроблено технічне завдання, яке можна знайти у додатку А. Це технічне завдання являє собою основою для подальшої реалізації застосунку.

1.4 Висновки. Постановка задачі

Висновки:

Отже були розглянуті особливості і невирешені завдання характерні для предметної області. Крім того були проаналізовані наявні програмні рішення, виявлені їх переваги та недоліки. Був складений список функціональних, нефункціональних та загальних вимог до розроблюваного програмного продукту.

Проаналізувавши вимоги та предметну область кваліфікаційної роботи було створено список необхідних для вирішення завдань. Було визначено, що дана область має велику перспективу розвитку, так як усі наявні програмні рішення містять значні недоліки. Крім наявних недоліків, дані рішення є недоступними для великої кількості користувачів, так як не є доступними для найпопулярніших мобільних платформ, як Android та IOS, або не мають достатньої технічної та інтерфейсної оптимізації для їх застосування в рамках сучасних мобільних операційних систем. Це в свою чергу свідчість про сегмент ринку програмної продукції в якому відсутні повноцінна конкурентність.

Виходячи з наведених чинників можна зробити висновок що реалізація застосунку для синхроного відтворення мультимедійних матеріалів є затребуваною широким колом користувачів і тому можна прогнозувати успішність цього проекту.

Постановка задачі:

На основі аналізу предметної області можна вивести наступну постановку завдання: Розробити мобільний застосунок для синхронізованого відтворення аудіо, відео, або презентаційних файлів. Забезпечити підтримкою основних форматів змереженя відповідних файлів. Забезпечити можливість відтворення та синхронізації файлів з внутрішньої пам’яті пристрою та за посиланням з мережі. Розробити компоненти керування переглядом мультимедійного файлу. Застосунок повинен бути реалізований з підтримкою систем на базі iOS та Android, мати зручний інтерфейс, оптимізований під екрани мобільних пристроїв.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Архітектура та функціональна структура застосунка

Проект програмного застунку відноситься до розподільних систем, так як синхронізація відтворення файлів на декількох пристроях передбачує декілька архітектурних компонентів.

Наведемо найпопулярніші моделі розродільних систем

P2P (Peer-to-Peer): У P2P-архітектурі кожен вузол (пір) виконує як клієнтські, так і серверні функції. Вузли співпрацюють між собою без централізованого сервера. Це може бути використано для обміну файлами (Наприклад BitTorrent), відеозв'язку (наприклад, Skype) або блокчейн-технологій (наприклад, Bitcoin).

SOA (Service-Oriented Architecture): SOA використовує поняття сервісів, які надають функціональні можливості, які можуть бути доступні для використання іншими компонентами системи. У SOA сервіси надаються через мережу і доступні для використання клієнтами за допомогою стандартизованих протоколів.

Мікросервісна архітектура: В мікросервісній архітектурі система розбивається на набір невеликих, незалежних сервісів, які можуть бути розгорнуті, масштабовані і керовані окремо. Кожен сервіс може виконувати певну функцію або операцію, і вони можуть взаємодіяти між собою за допомогою різних інтерфейсів або API.

Клієнт-серверна архітектура є широко використовуваною моделлю у багатьох застосунках і системах. У цій архітектурі клієнтські програми взаємодіють з серверами, які надають послуги або ресурси, які клієнти запитують. Клієнти і сервери можуть знаходитись на різних пристроях та комунікувати між собою за допомогою протоколів мережі, таких як TCP/IP або HTTP.

Враховуючи суть програмного продукту, яка полягає у надані доступу до мультимедіа матеріалів та синхронізації їх відтворення, потрібно зважати на необхідність компенента в системі, що міг би відігравати роль централізованого медіатора.

Цим медіатором міг би стати мобільний пристрій адміністратору, у випадку з P2P архітектурою, проте він не зможе надати можливості збереження даних користувача. Крім того використання мобільного пристрою користувача у якості сервера, погано вплине на продуктивність застосунку, адже буде використовувати процесорні потужності та оперативну пам'ять, для того щоб уникнути наведених проблем, можна використати клієнт-серверну архітектуру, так як роль централізованого медіатору візьме на себе сервер на окремій фізичній машині, яка буде оптимізована виключно під обробку клієнтських запитів, а персональна інформація користувачів буде захищеною і не доступною для інших мобільних пристроїв.

У клієнт-серверній архітектурі сервери відповідають на запити клієнтів, обробляють їх і надають відповідні результати або ресурси. Це може бути надання даних з бази даних, виконання обчислень або надання доступу до певних послуг. Клієнти, у свою чергу, виконують запити до серверів, отримують результати і відображають їх для користувачів.

Одна з основних переваг клієнт-серверної архітектури полягає в тому, що вона розділяє відповідальність між клієнтами і серверами. Це дозволяє забезпечити більшу масштабованість, ефективність та керованість системи. Крім того, ця архітектура дозволяє зручно реалізовувати розподілені системи, де клієнти можуть бути розташовані на різних пристроях або мережах.

Однак, клієнт-серверна архітектура також має свої обмеження. Вона може бути централізованою, що означає, що сервер є однією точкою входу для всіх клієнтів, що може створювати проблеми з масштабованістю і витривалістю. Крім того, вона вимагає постійного з'єднання між клієнтами і серверами, що може бути проблемою в умовах нестабільного мережевого зв'язку.

Вищенаведені недоліки ж не є перешкодами для реалізації основних функціональних і технічних вимог застосунку, тому ними можна знехтувати на даному етапі.

Враховуючи ж вищенаведені переваги можна обрати клієнт-серверну архітектуру, як основу розподільної системи майбутнього мобільного застосунку.

2.2 Опис структури даних та моделі бази даних

Структура даних майбутнього застосунку представляє собою інформацію про користувача, групи і мультимедійні матеріали. Моделі даних можуть бути побудована наступним чином:

Медіа-файли: Це основні об'єкти даних, які включають в себе відео, аудіо, фотографії тощо. Медіа-файли можуть бути збережені як файли на сервері або в хмарному сховищі. У базі даних можуть бути збережені метадані про медіа-файли, такі як назва, опис, тривалість, розмір тощо.

Користувачі: Користувачі - це особисті акаунти, які реєструються в застосунку. Для кожного користувача в базі даних можуть бути збережені дані, такі як ім'я, електронна пошта, пароль, налаштування відтворення та збережені вибори.

Кімнати: Кімнати представляють з'єднання між користувачами і медіа матеріалом, який відтворюється Для кожної кімнати можуть бути збережені дані, такі як ідентифікатор користувача, ідентифікатори пристрою, стан синхронізації, позиція відтворення тощо.

Пристрої: Це пристрої, на яких користувачі відтворюють мультимедійні матеріали. Кожен пристрій може мати унікальний ідентифікатор, модель, назву та інші характеристики.

Групи: Це представлення згрупованих користувачів, які вже запрошувались в кімнати адміністратором в минулому.

У моделі бази даних можуть бути використані різні підходи, такі як реляційна модель, документ-орієнтована модель, графова модель або ключ-значення модель, в залежності від потреб застосунку.

Розглянемо можливі моделі баз даних для реалізації застосунку:

Реляційна модель баз даних: Реляційна модель баз даних є однією з найпоширеніших моделей і використовує таблиці для організації даних зі зв'язками між ними за допомогою ключів. Вона підходить для структурованих даних зі статичною схемою, наприклад, для збереження даних про користувачів та їх аутентифікацію. Однак, для медіа-файлів, які можуть мати різні метадані та

властивості, реляційна модель може бути обмеженою, оскільки змінність структури даних не є її сильною стороною.

Документна модель баз даних: Документна модель баз даних використовує документи (наприклад, у форматі JSON або XML) для збереження даних. Кожен документ може мати свою власну структуру, що дозволяє гнучко зберігати різні типи даних. У випадку мобільного застосунку для синхронізації мультимедійних матеріалів, документна модель може бути вибрана через її простоту і гнучкість. Кожен медіа-файл може бути представлений як окремий документ з власними метаданими, такими як назва, опис, тривалість, розмір, теги тощо. Це дає змогу зберігати і оновлювати метадані медіа-файлів без необхідності змінювати структуру бази даних.

Модель ключ-значення баз даних: У цій моделі бази даних дані зберігаються у вигляді пар ключ-значення. Це проста та швидка модель, яка може бути ефективною для збереження та отримання медіа-файлів на основі їх унікальних ідентифікаторів. Однак, вона може бути обмеженою в розширеному управлінні структурованими даними, які можуть виникати при збереженні складних метаданих або взаємозв'язків між різними об'єктами.

Зважаючи на вищенаведені характеристки різних моделей баз даних можна прийти до висновку, що документна модель бази даних є відмінним вибором для мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях з наступними перевагами:

– Гнучкість: Документна модель дозволяє зберігати медіа-файли та їх метадані у форматі документів, що дозволяє гнучко враховувати різні типи даних та їх структуру. Вона дає змогу додавати, оновлювати та видаляти атрибути медіа-файлів без зміни загальної структури бази даних.

– Простота розширення: Документна модель дозволяє додавати нові типи медіа-файлів або метадані без необхідності змінювати схему бази даних. Це особливо важливо в динамічному середовищі, де можуть виникати нові типи медіа-файлів або вимоги до метаданих.

– Швидкий доступ: Документна модель бази даних дозволяє ефективно отримувати медіа-файли та їх метадані за допомогою унікальних ідентифікаторів документів. Це дозволяє швидко знаходити та синхронізувати дані між різними пристроями без зайвого завантаження сервера.

– Сумісність з JSON: Документна модель бази даних добре сумісна з JSON (JavaScript Object Notation), який є популярним форматом обміну даними. JSON легко обробляти та інтерпретувати мобільними пристроями, що спрощує взаємодію між мобільним додатком та базою даних.

Загалом, вибір документної моделі бази даних для мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях дозволяє забезпечити гнучкість, простоту розширення та ефективний доступ до даних.

Крім того, для забезпечення ефективності та швидкості синхронізації можуть використовуватись технології кешування, реплікації даних або використання CDN (Content Delivery Network) для розподілу контенту на різні сервери.

Важливо враховувати вимоги безпеки, які можуть включати захист даних користувачів, автентифікацію, авторизацію та аудит доступу до даних.

2.3 Проектування серверної частини застосунка

Серверну частину майбутнього застосунку можна представити наступними компонентами

REST API: Розробка API для обміну даними між мобільним застосунком та сервером. Це може включати ендпоінти для отримання медіа-файлів, відтворення, збереження та оновлення даних.

Уніфікованість інтерфейсу, кешування, незалежність стану між клієнт-серверними запитами та ресурсо-орієнтованість є основними вимогами до API

REST архітектури. Дані вимоги забезпечують масштбування системи та відповідають завданням проектування серверної частини застосунку.

Аутентифікація та авторизація: Реалізація механізмів аутентифікації та авторизації користувачів для забезпечення безпеки даних та обмеження доступу до функцій застосунку, являє собою важливу частину логіки програмного продукту. Існує декілька варіантів проектування логіки аутентифікації в клієнт-серверній архітектурі в рамках REST API, але в найлогічнішим варіантом з найкращим співвідношенням безпеки і простоти реалізації для застосунку для відтворення мультимедійних матеріалів є JWT.

JWT (JSON Web Token) - це стандарт відкритих токенів, який використовується для передачі безпечної та автономної інформації між двома сторонами в форматі JSON. JWT забезпечує механізм аутентифікації та авторизації в розподілених системах, таких як веб-застосунки або мікросервіси.

JWT складається з трьох частин, які представлені у вигляді строкових значень і розділені крапками: заголовок (header), твердження (claims) та підпис (signature). Кожна з цих частин представлена у форматі JSON і закодована у Base64 URL-safe.

Заголовок (header): Вказує тип токена і алгоритм, який використовується для підпису. Наприклад, {"alg": "HS256", "typ": "JWT"} вказує на використання алгоритму HMAC-SHA256 для підпису токена.

Твердження (claims): Включає корисну інформацію про підтвердження, таку як ідентифікатор користувача, строк дії токена, ролі користувача тощо. Існують три типи тверджень: зарезервовані (registered claims), публічні (public claims) і приватні (private claims). Зарезервовані твердження включають такі поля, як "iss" (видавець), "exp" (строк дії), "sub" (суб'єкт), "aud" (аудиторія) тощо.

Підпис (signature): Створюється шляхом підпису заголовка та тверджень за допомогою секретного ключа або приватного ключа (якщо використовується асиметричний алгоритм). Підпис перевіряється при отриманні токена для підтвердження його цілісності.

JWT є компактним та самодостатнім форматом, що дозволяє передавати інформацію без необхідності звертатися до сервера для перевірки токена. Використовуючи JWT, можна реалізувати безпеку, аутентифікацію та авторизацію в розподілених системах, дозволяючи сторонам обмінюватися надійними токенами. Загальна схема використання JWT зображена нижче на рисунку 2.1

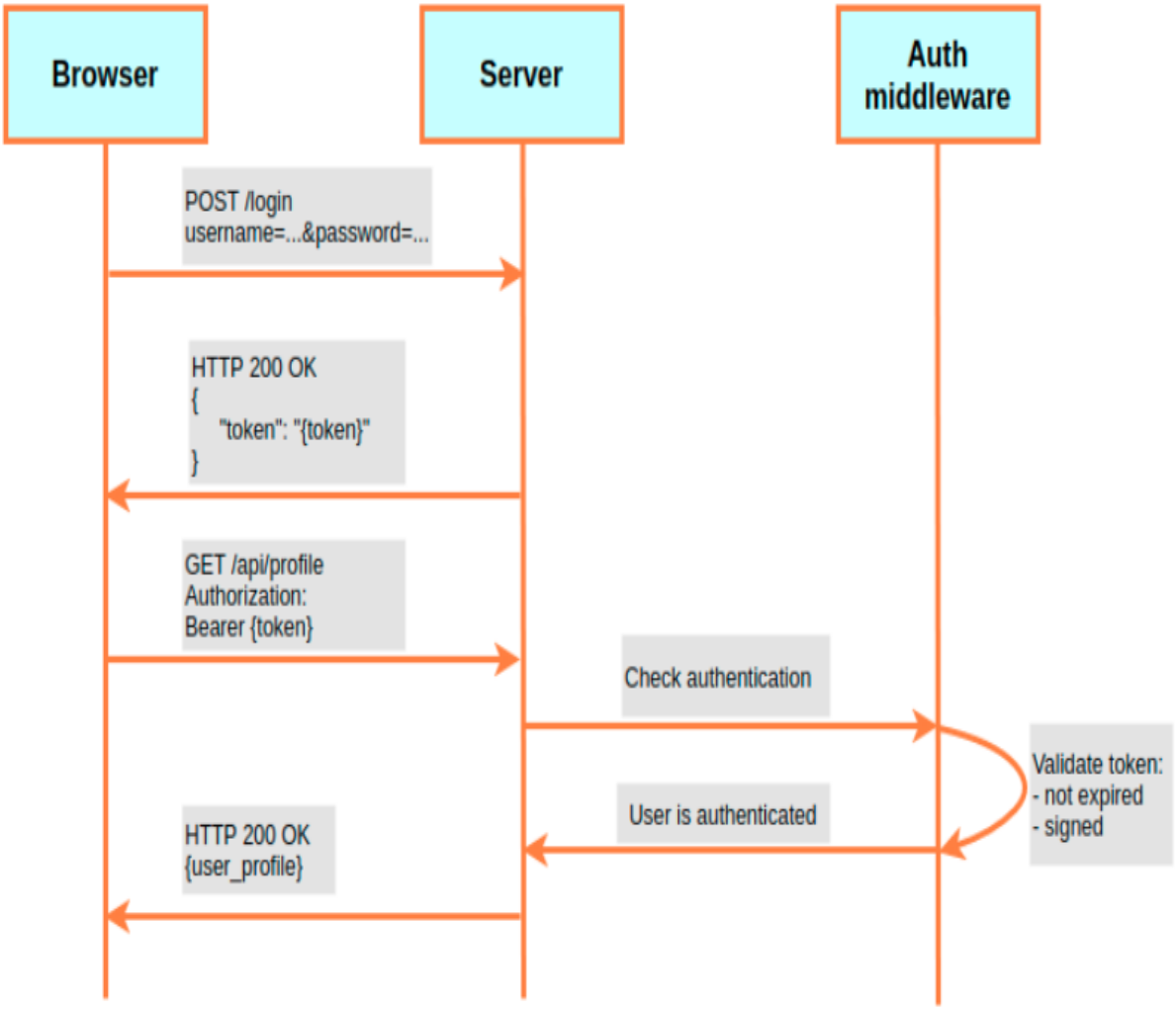


Рисунок 2.1 – Загальна схема роботи JWT – https://nixstech.com

Синхронізація даних: Розробка логіки синхронізації відтворення мультимедійних матеріалів між різними пристроями користувача. Це може включати відстеження поточного стану відтворення, синхронізацію позицій відтворення та оновлення статусу відтворення на всіх підключених пристроях.

Для забезпечення синхроного відтворення легше всього скористатись технологією веб сокетів. Альтернативні варіанти такі як push нотифікації хоч і

можуть бути використані для цієї мети, але не будуть правильно застосованими інструментами, так як не були призначені для налаштування лінії комунікації між двома клієнтськими застосунками.

Інтеграція з зовнішніми сервісами: Забезпечення інтеграції з сервісами для зберігання медіа-файлів та для зберігання даних, може включати взаємодію зі збереженням, вилученням та оновленням медіа-файлів, а також зберігання та отримання даних користувачів. Використання зовнішніх сервісів є доцільним і зручним в даному випадку, так як дозволяє зменшити витрати на інфраструктуру серверної частини до мінімуму, зосередившись на логікі роботи застосунку. Сучасні хмарні сервіси для зберігання даних дають широкий спектр можливостей, зручний інтерфейс доступу, гнучкість використання та не велику ціну, а в деяких випадках навіть можливість безкоштовного використання на певному етапі масштабування системи.

2.4 Проектування інтерфейсу користувача

Проектування інтерфейсу користувача для мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях включає детальне планування та створення зручного, привабливого та інтуїтивно зрозумілого інтерфейсу, який забезпечує зручність взаємодії користувача з застосунком. Вивчивши потреби та очікування цільової аудиторії, аналіз конкурентних продуктів і трендів у дизайні мобільних застосунків. Результатом цього етапу буде створення образу мети та контексту використання застосунку.

Далі логічно буде розробити інформаційну архітектуру застосунку. Розподіл функцій та контенту програмного продукту на логічні блоки і категорії, створення ієрархії сторінок та навігаційної структури.

Інформаційна архітектура визначає структуру та організацію контенту в застосунку. Основною метою є забезпечення зручності користування та легкості знаходження потрібної інформації. Для мобільного застосунку можна розглянути таку структуру:

Головний екран: На головному екрані можуть розміщуватись основні функції застосунку, такі як пошук, перегляд останніх відтворених матеріалів, вибір категорій або плейлистів.

Кімнати: Користувачі можуть переглядати список публічних кімнат, додавати до них відео, аудіо або інші мультимедійні матеріали для зручного відтворення у випадку якщо вони є адміністраторами. Кімнати можуть бути організовані за темами, настроєм або іншими параметрами.

Користувачі: Список користувачів яких можна запросити до свої кімнат, або користувачі які вже були запрошені раніше.

Профіль: Профіль відображає інформацію про користувача, надає можливість редагуавання цієї інформації.

Наступним етапом буде визначення сценаріїв використання. Визначення типових сценаріїв використання застосунку та вивчення послідовностей дій користувача допоможе врахувати реальні потреби користувачів при проектуванні інтерфейсу.

Сценарії використання описують типові послідовності дій, які користувачі можуть виконувати у застосунку. Нижче приведені деякі сценарії для мобільного застосунку для синхронізації відтворення мультимедійних матеріалів:

- Створення кімнат – має забезпечувати сторінку створення кімнат, що буде включати в себе вибір медіаматеріалу для відтворення, розсилка запрошень для користувачів, налаштування додаткових параметрів кімнати;
- Сторінка відтворення медіматеріалу має забезпечувати синхронізацію відтворення, а також ряд різних функцій для адміну та користувача, адмін має мати можливість керувати відтворенням і адмініструвати список користувачів, а користувач має право підняти руку і проголосувати за паузу;
- Редагування інформації користувача користувач має мати можливість відредагувати свою персональну інформацію, та керувати своїми власними кімнатами;

Варто враховувати потреби користувачів та забезпечити зручність навігації та зручність використання.

Виходячи з інформаційної структури інтерфейсу та наведених сценаріїй використання можна розробити схему основних екранів інтерфесу.

Почати можна з головного екрану, основну частину екрану доцільно використати для відображення доступних користувачу на даний момент кімнат, це можуть бути як приватні кімнати до яких в користувача є доступ, так і публічні кімнати, до яких будуть мати доступ усі користувачі.

В нижній частині головного екрана буде розсташована навігація організована у вигляді нижніх табів, це дозволить зробити основні функції застосунку доступними і видимими з перших хвилин використання.

Наступним екраном є екран створення кімнати основними компонентами будуть список запрошених користувачів, компоненти керування запрошеннями та компоненти керування мультимедіа внизу.

Список запрошених користувачів дозволить переглядати користувачів в яких є доступ до даної кімнати, прибирати їх доступ, показувати статус цих користувачів.

Компоненти керування запрошенями дозволять запрошувати користувачів, реалізація запрошення буде організована через діалогове вікно в якому можна ввести ідентифікатор користувача, якого потрібно запросити.

Також через модальне вікно буде реалізована форма додавання нового мультмимедійного файлу, що буде відтворюватись для всіх запрошених користувачів та адміністраторів в кімнаті, модальне вікно буде містити форму для введення інформації необхідної для первинного створення кімнати, такими як: тип мультимедійного файлу, джерело файлу (інтернет чи локальне сховище даних пристрою) та саме поле для файлу.

Компоненти керування мультимедіа дозволять обирати матеріал переглядяємий в кімнаті, а також надаватимуть користувачу адміністратору кімнати доступ до управління переглядом або прослуховуванням матеріалу, це може включати в себе: прокрутку відеоролику, призупинення перегляду, перемикання слайду, регулювання гучності, або вибір іншого типу медіаматеріалу для подальшого перегляду.

Схематичне зображення інтерфейсу двох описаних екранів можна побачити нижче на рисунку 2.2

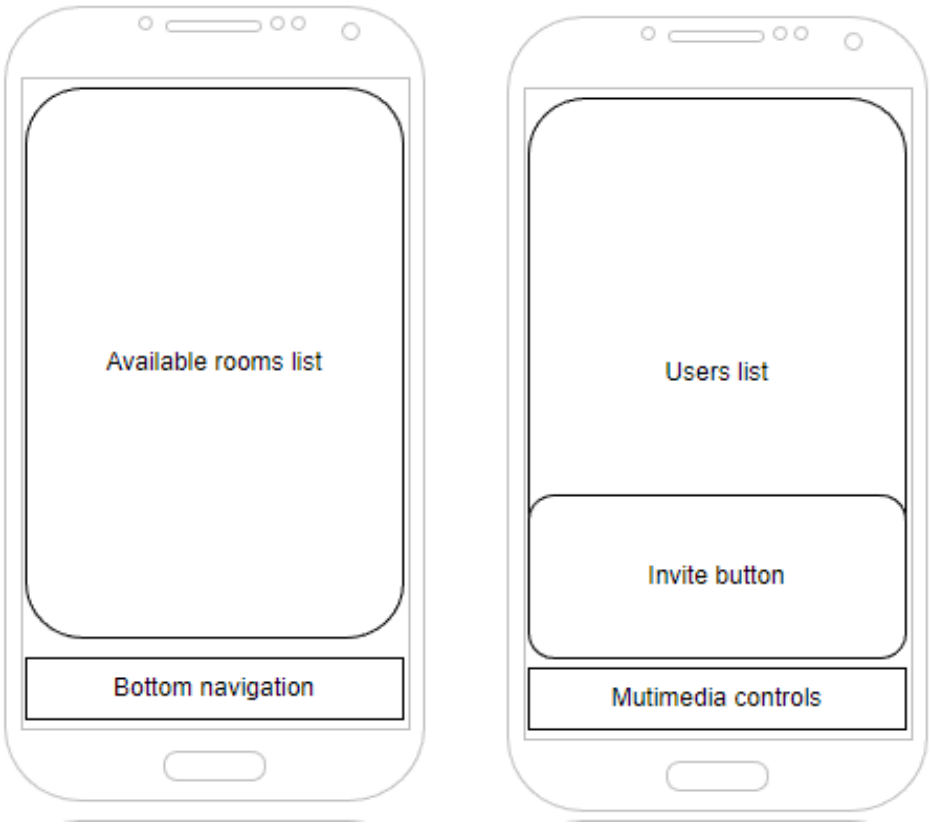


Рисунок 2.2 – Схема головного екрану і екрану кімнати

Далі можна описати структуру екрану кімнати під час синхронізованого перегляду, для користувача. Основну частину екрана буде займати компонент відображення медіаматеріалу, внизу екрану будуть також розсташовані кнопки які відповідають за дії “підняти руку” та проголосувати за паузу, ці дії мають бути розсташовані таким чином, щоб не заважати перегляду медіа матеріалу. З схематичною структурою екрану кімнати під час перегляду можна ознайомитись нижче на рисунку 2.3.

Наступним можна описати екран профілю користувача. Схема даного екрану буде складатися з трьох основних компонентів, компонент профілю, компонент редагування імені для відображення і компонент власних кімнат користувача.

Компонент профілю буде відображати інформацію про користувача яку не можна редагувати, це можуть бути ідентифікатори основані на електроній пошті, логін та інша інформація.

Компонент редагування імені для відображення буде містити в собі ім’я яким користувач буде позначений в кімнатах і видимий для інших користувачів, також компонент відповідає за редагування цього поля.

Список власних кімнат користувача відповідає за відображення кімнат, які були створенні користувачем, або кімнати в яких користувач має права адміністратора, цей компонент дозволить адмініструвати та видаляти кімнати або переходити до них. Структура особистої сторінки користувача такого виду буде сприйати швідкій та простій навігації в межах застосунку, дозволить користувачу мати доступ до всіх функцій, які відносяться до його сутності в межах одного екрану. Структурну схему екрану профіля користувача можна побачити нижче на рисунку 2.3

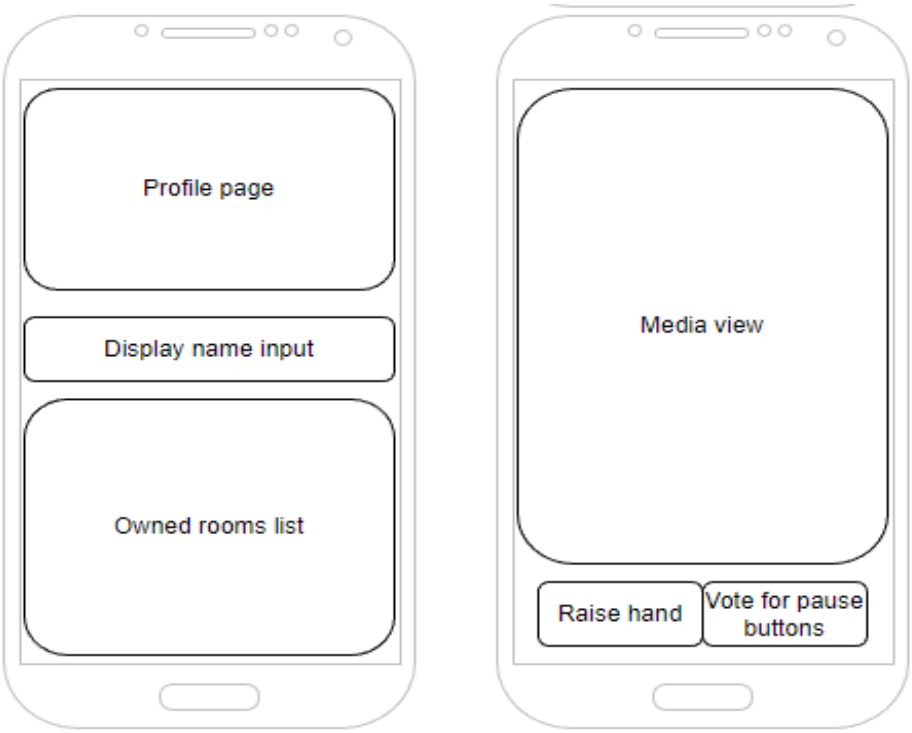


Рисунок 2.3 – Схема екрану профілю і екрану кімнати під час перегляду

Слід також памятити про специфіку розробки мобільних застосунків, більшість сучасних пристроїв мають дві можливі орієнтації: портретну, або

альбомну. Для багатьох застосунків стало стандартом відключати можливість зміни орієнтації на альбомну, адже дуже важко зробити інтерфейс адаптивним під горизонтально розсташований екран, але у випадку з розроблюваним застосунком підтримка такої орієнтації є необхідною, адже абсолютна більшість мультимедіа матеріалів мають співвідношення сторін форма 16 на 9, або схоже. Тому можна дозволити зміну орієнтації на екрані перегляду. Для цього потрібно створити схематичну структуру албомної орієнтації екрану перегляду кімнати користувача.

Альбомна орієнтація повинна містити всі ті самі компоненти керування та функції що і портретна, тому є сенс просто перенести нижні компоненти управління на ліву сторону екрана, таким чином вони не будуть забирати велику частину екрану, а при відсутності дій користувача будуть зникати з екрану. Основний мульти медіа програвач буде займати основну частину екрану і буде мати пропорції відповідні до більшості відео та презентаційних файлів. Ознайомитися з альбомним варіантом орієнтації кімнати перегляду мультимедіа матеріалів можна нижче на рисунку 2.4.

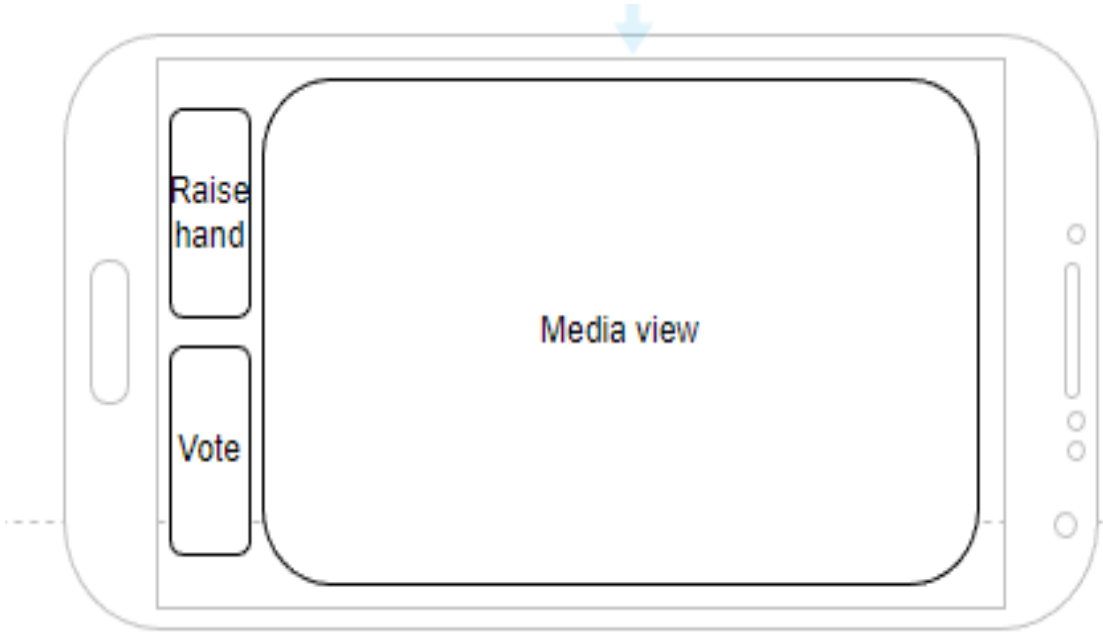


Рисунок 2.4 – Альбомна схема екрану кімнати під час перегляду

2.5 Аналіз та вибір технологій і методів реалізації застосунка

React Native - технологія, що дозволяє розробляти мобільні застосунки для iOS та Android, використовуючи знайомий для багатьох розробників JavaScript.

Одними з головних переваг React Native є те, що вона дозволяє створювати мобільні застосунки швидше та з меншими витратами, оскільки дозволяє використовувати код для обох платформ, а не писати його окремо для кожної з них.

Крім того, використання React Native зменшує кількість помилок, оскільки дозволяє відслідковувати їх на ранніх етапах розробки.

Схема роботи технології "моста", що забезпечує сумісність бібліотек мобільних платформ з кодом на основі javascript наведена нижче на рисунку 2.5. Код Javascript виконується на застосунку паралельно з викликами до бібліотек платформ таким чином відбувається комунікація між Node.js та компонентами системи.

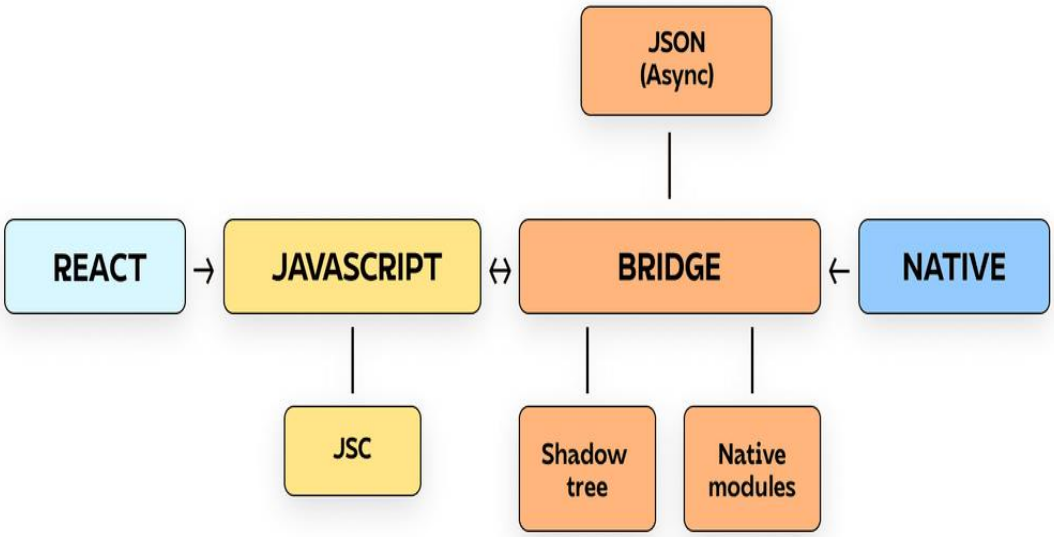


Рисунок 2.5 – Схема роботи "моста" – https://collectivemind.dev

Ще однією перевагою React Native є те, що вона дозволяє розробникам створювати мобільні застосунки з високою якістю та швидкістю, яка набагато

перевищує інші технології розробки мобільних застосунків. Це особливо важливо для бізнес проектів та ігор, де швидкість та продуктивність є ключовими факторами.

Крім того, React Native є дуже гнучкою технологією, яка дозволяє розробникам забезпечувати просту навігацію для користувачів, а також забезпечує високу стійкість до збоїв. Завдяки використанню React Native, розробники можуть швидко створювати застосунки, які працюють стабільно та без збоїв, що дуже важливо для забезпечення задоволення користувачів.

Отже, React Native - дозволяє розробникам створювати мобільні застосунки з високою продуктивністю, стійкістю та гнучкістю. Використання React Native дозволяє зменшити витрати на розробку, тому для проекту було обрано саме цю технологію.

Для реалізації подібної системи можливий широкий вибір СКБД та баз даних. PostgreSQL є однією з найпопулярніших та найбільш надійних відкритих систем керування базами даних (СКБД), що використовуються в розробці програмного забезпечення. Використання PostgreSQL для розробки бази даних для застосунку має декілька переваг.

По-перше, PostgreSQL є дуже стабільним та надійним СКБД. Він має довгу історію розробки та випуску оновлень, які регулярно виправляють помилки та уразливості. Крім того, PostgreSQL має високий рівень безпеки, що робить його ідеальним вибором для бази даних, яку будуть використовувати багато користувачів.

По-друге, PostgreSQL має потужний набір функцій та можливостей. Він підтримує багато типів даних, включаючи географічні та геодезичні дані, а також великі об'єми даних. PostgreSQL також має підтримку стандарту SQL, що робить його легким для використання та інтеграції з іншими системами.

По-третє, PostgreSQL є відкритою системою, що означає, що він має велику спільноту розробників, які активно працюють над вдосконаленням та підтримкою СКБД. Це означає, що ви можете легко знайти документацію, підтримку та допомогу в розвитку вашої бази даних.

Загалом, використання PostgreSQL для розробки бази даних для застосунку є хорошим вибором з багатьох причин, включаючи його стабільність, надійність, потужність та відкритість. Він дозволяє розробникам ефективно працювати з даними та забезпечує користувачам безпечний та зручний доступ до даних.

MongoDB є документною системою керування базами даних (СКБД), яка знаходить широке використання в розробці програмного забезпечення. Використання MongoDB для розробки бази даних для застосунку має такі переваги.

По-перше, MongoDB має дуже легкий підхід до організації даних. У MongoDB дані зберігаються у вигляді документів, що робить його ідеальним вибором для розробки застосунків, які використовують динамічні дані не стійкої структури. MongoDB також дозволяє легко масштабувати базу даних, що дозволяє розробникам легко зберігати великі об'єми даних та працювати з ними ефективно. По-друге, MongoDB має потужну систему запитів та індексів. MongoDB підтримує багато типів запитів, включаючи запити за критеріями, запити на підсумки, групування та багато іншого. Це дозволяє розробникам отримувати необхідні дані швидко і ефективно. Крім того, MongoDB має потужний механізм індексації, що прискорює пошукові запити. По-третє, MongoDB має відкритий код та велику спільноту розробників. MongoDB має широку підтримку і документацію, що дозволяє розробникам швидко засвоїти та використовувати цей СКБД. Крім того, велика спільнота розробників забезпечує активний розвиток та підтримку MongoDB.

Хмарна панель керування має широкий спектр інструментів та дозволяє адмініструвати документну базу в браузері без інстальованих систем управління базами даних. В ній доступні всі операції створення, видалення, редагування фільтрації та пошуку даних, крім того можливість використовувати стандартні запити MongoDB для більш складних агрегатних операцій або фільтрів. Існує можливість керування доступами до баз даних, створення груп користувачів та можливість створення транзакції, прямо через панель керування, що полегшує розробки системи безпеки для бази. Нижче, на рисунку 2.5 наведений скріншот хмарної панелі керування для бази даних

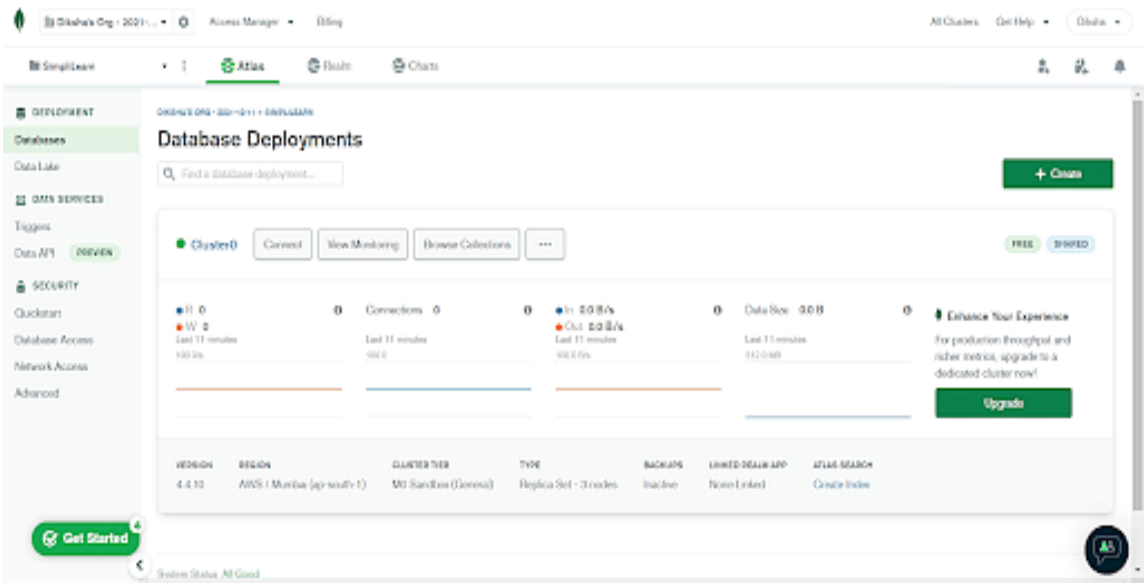


Рисунок 2.6 – Хмарна панель керування MongoDB

Загалом, використання MongoDB для розробки бази даних є хорошим вибором.

DynamoDB є хмарною NoSQL системою керування базами даних, яка забезпечує швидкий та масштабований доступ до даних. Використання DynamoDB для розробки бази даних для застосунку має наступні переваги.

По-перше, DynamoDB є хмарною системою, це означає, що вона забезпечує ефективний доступ до даних з будь-якої точки світу. Це дозволяє розробникам застосунків зберігати дані на серверах, забезпечуючи масштабований доступ до даних.

По-друге, DynamoDB є системою NoSQL, що дозволяє зберігати дані у вигляді ключ-значення. Це дозволяє розробникам зберігати дані зі змінною структурою, що робить DynamoDB ідеальним вибором для розробки застосунків, які використовують динамічні дані.

По-третє, DynamoDB має потужну систему масштабування, що дозволяє розробникам зберігати великі об'єми даних та працювати з ними ефективно. DynamoDB може автоматично масштабувати обсяги даних, що забезпечує високу доступність та швидкий доступ до даних незалежно від їх обсягу.

Схема загального принципу використання DynamoDB наведена нижче на рисунку 2.7

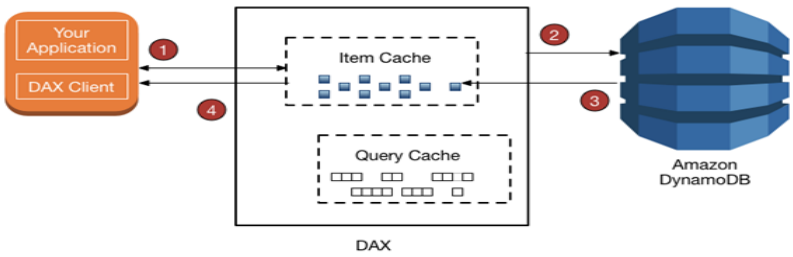


Рисунок 2.7 –Схема DynamoDB – https://aws.amazon.com/database

Загалом, використання DynamoDB для розробки бази даних для застосунку є хорошим вибором для хмарної системи з гнучкою структурою.

Так як основні технічні складності завдання представляють собою взаємодію мобільних застосунків через мережу, база даних не є основним фокусом проекту. Прогнозована структура бази даних не є складною так як має містити в собі лише базову інформацію про користувачів.

Виходячи з вищесказаного використання важких реляційних систем не є доцільним, так як збільшить термінрозробку, але не принесе жодних переваг крім, полегшення масштабування проекту. Таким чином є сенс обрати докумнтноорієнтовану базу, так як структура інших нереляційних баз не відповідає складності системи.

Остаточний вибір було зроблено на користь MongoDB. Сучасна документна база даних, підтримує транзакції, складні запити, масштабування через репліки, яке відбувається автоматично. Хостинг бази можна безпечно та просто налаштувати через сервіс Atlas.

JavaScript – в якості мови програмування JavaScript випливає з вибору технології для розробки мобільних застосунків. React Native можна використовувати лише на базі JavaScript. Це динамічно типізована та сучасна мова програмування. JavaScript дозволяє користуватися сучасним синтаксисом, великою

кількістю зручних вбудованих функції, механізм для обробки асинхронних операцій. Загалом ця мова цілком підійде для використання в розробці проекту. В випадку React Native JavaScript буде використовуватись на базі NodeJS (середи виконання javascript), що в свою чергу дозволить використання модулів з реєстра npm. Це дозволить зменшети витрати та значно пришвидшити розробку. Як видно зі схеми Npm має також деякі вразливості, однією з них є неможливість вичерпної перевірки модулів. Тому модулі можуть бути використані для атаки на застосунок, як це вже бувало в минулому. Проте існує велика кількість інструментів що допомагають вирішити наведені проблеми, такі як менеджмент залежностей через боти, відсутність автоматичного обновління мажорних версій та інші. Схема роботи модулів регістра NPM наведена нижче на рисунку 2.8.

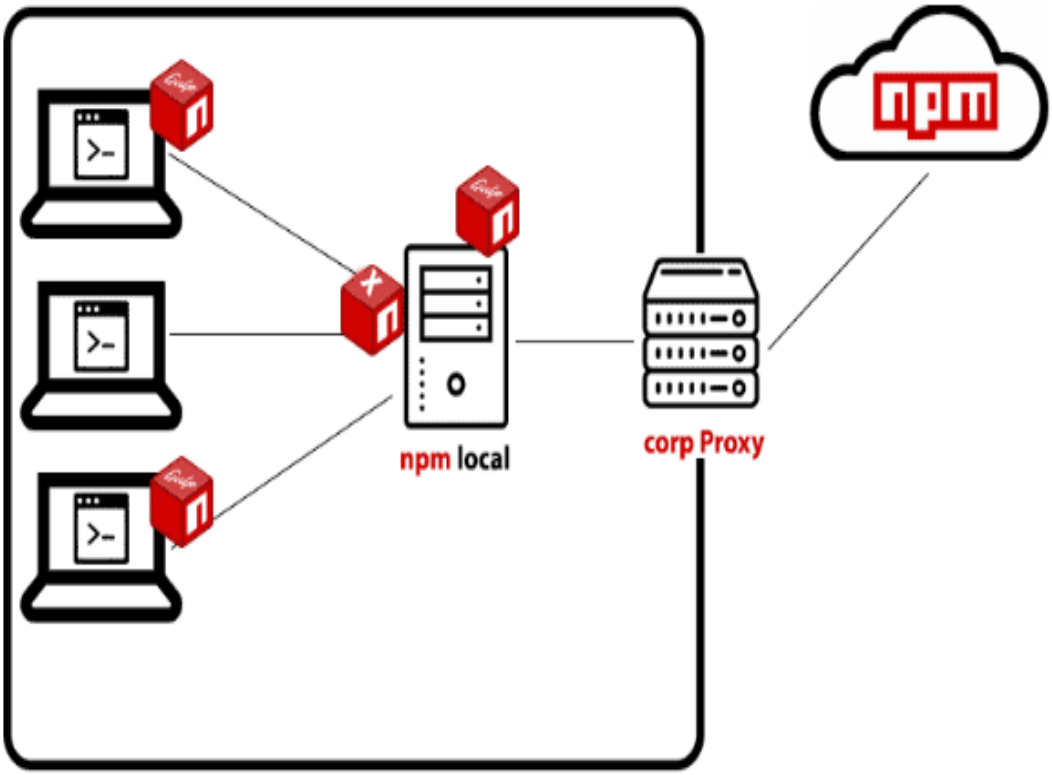


Рисунок 2.8 – Схема регістру npm – https://www.researchgate.net

Крім того вибір на користь JavaScript та NodeJS означає можливість використання цієї самої зв’язки при розробці сервера, що спростить реалізацію BackEnd функціоналу.

Висновки:

Отже JavaScript повністю відповідає вимогам, для використання в якості основної мови програмування, при роботі над індивідуальним завданням.

Можна підвести підсумки проекутвання.

Після завершення проектування застосунку можна зробити ряд висновків стосовно проектування. По-перше, було прийнято рішення використовувати клієнт-серверну структуру для створення застосунку, проаналізовані її переваги та недоліки. Був проведений аналіз та розроблена структура даних. Розроблена клієнтська частина застосунку, яка повністю задовольнятиме потреби користувачів. Розроблена клієнтська частина має простий і зрозумілий інтерфейс, який дозволить користувачам гарно орієнтуватися в застосунку. Після проведення аналізу існуючих рішень був проведений аналіз технологій та обраний найкращий набір технологій для бази даних, клієнтської та серверної частин. Отже, цей набір технологій та обрана структура дозволять вирішити всі проблеми та завдання, які стоять перед розробкою програмного забезпечення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

За результатами проектування було визначено, що для зберігання інформації застосунку будуть використані хмарні сервіси, які значне зменшують затрати на інфраструктуру. Такими хмарними сервісами було обрано Amazon S3 – для збереження файлів та MongoDB для збереження даних.

Отже можна розглянути процес створення бази даних в хмарному сервісі MongoDB. Спочатку треба створити кластер для майбутньої бази даних, вибрати тип кластеру, провайдер на регіон, вказати додаткові параметри. Сторінку створення кластеру можна побачити на рисунку 3.1

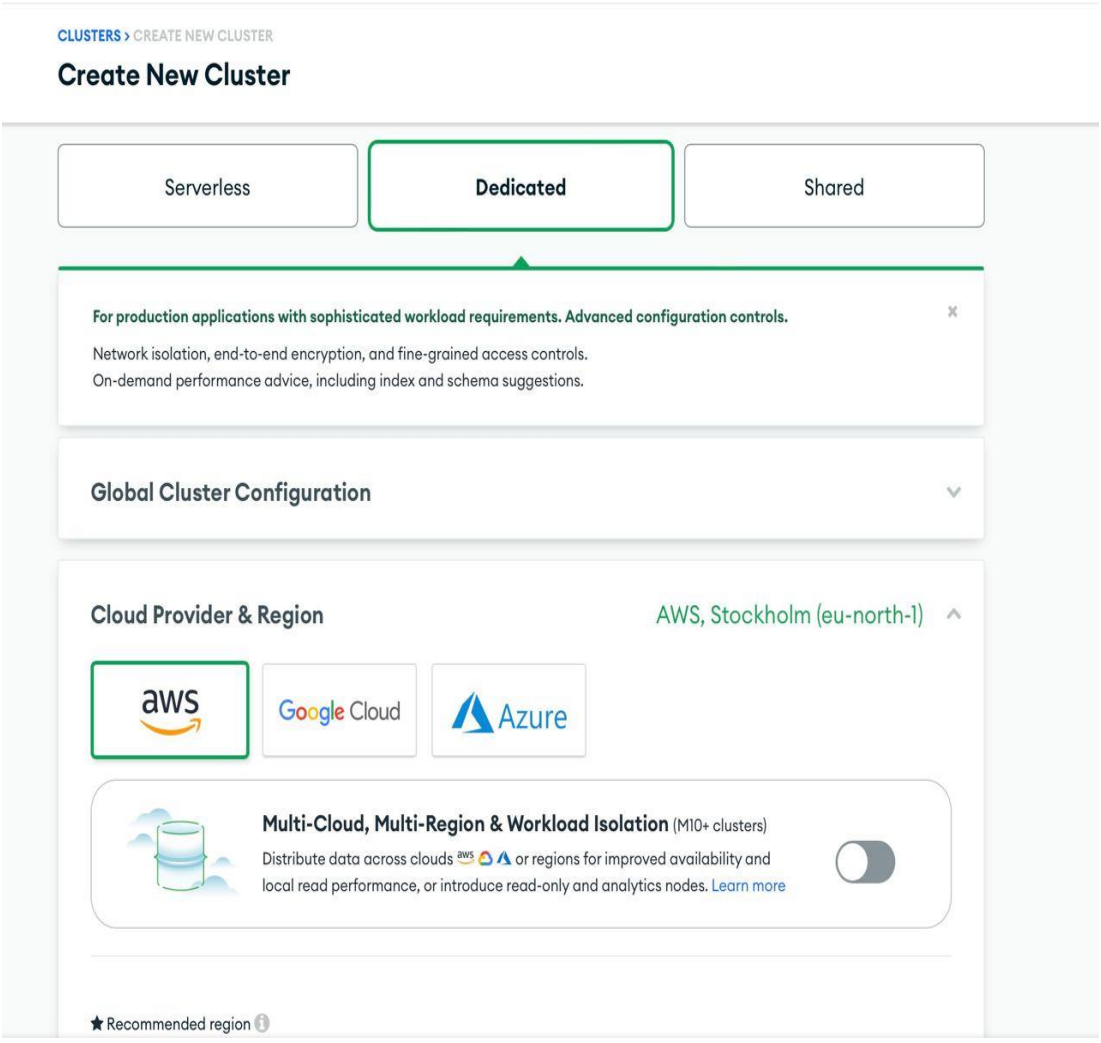


Рисунок 3.1 – Сторінка створення кластеру

Після успішного створення кластеру, потрапляємо на сторінку кластеру. Тут бачимо назву новоствореного кластеру, його тип та таби навігації по колекціям. Також на сторінці відображені шарди даних, які система автоматично створила. Шарди це спосіб реалізації горизонтального масштабування впроваджений mongoDb. За допомогою шардів систему автоматично оптимізується під розмір створеного кластеру. Створені кластери можна побачити на нижче рисунку 3.2

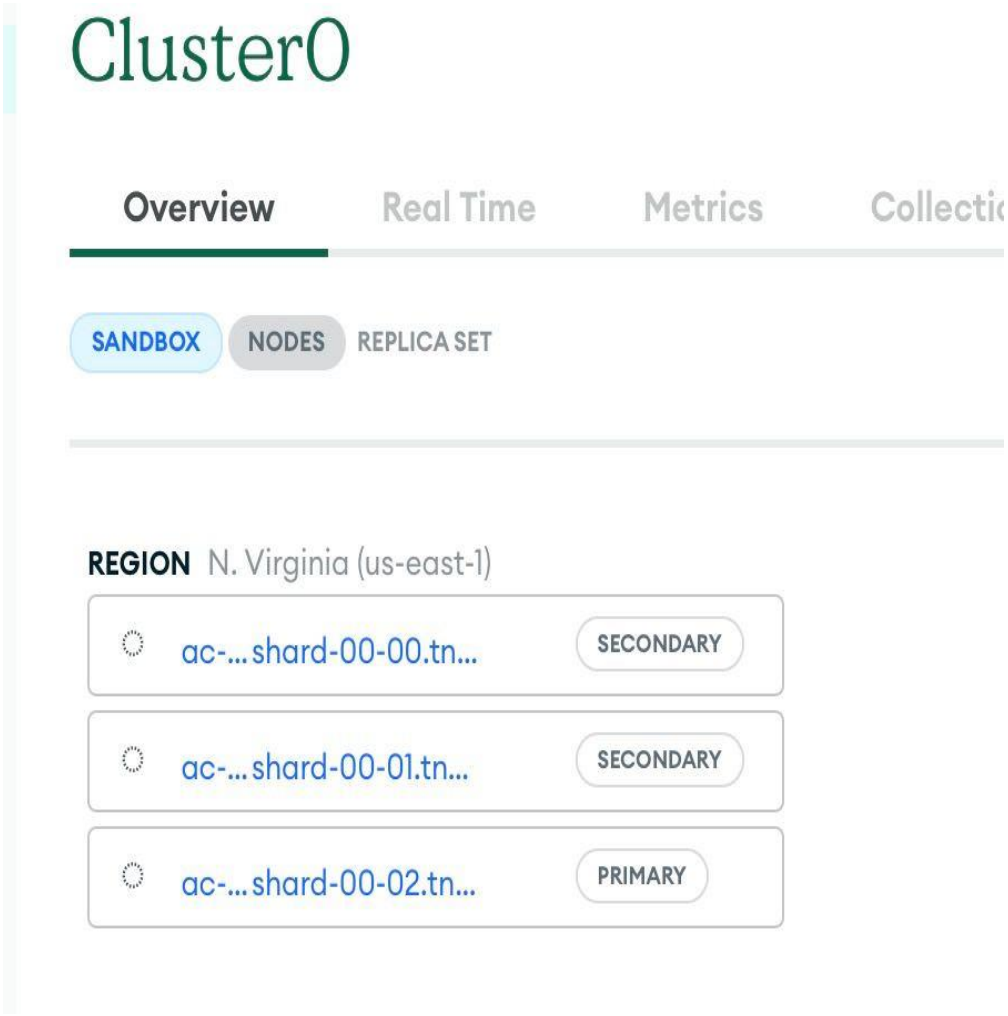


Рисунок 3.2 – Автоматично створені шарди

Далі так як кластер лежить на вищому рівні абстракції ніж база даних в mongodb, нам пропонують створити саму базу. Після натискання на відповідну кнопку, можна побачити діалогове вікно з полями для введення даних Потрібно ввести ім’я бази, додаткові параметри та назву першої колекції. Колекція на етапі створення бази задається через специфіку внутрішньої роботи бази. Навідміну від

реляційних баз даних на основі мови SQL, mongoDB не створює допоміжні таблиці автоматично. Через це при створені бази потрібно також задавати назву і для її першої колекції. Вигляд вікна створення бази даних можна побачити на рисунку 3.3

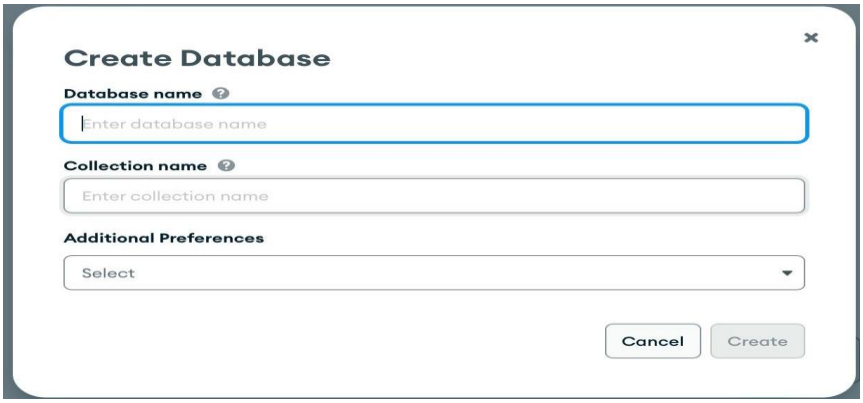


Рисунок 3.3 – Вікно створення бази даних

Після успішного створення бази даних, та її першої колекції, система запропонує заповнити їх даними. Після згоди можна вибрати заповнення вручну, або згенерувати дані на основі заданих параметрів. Після заповнення даних відредагувати їх можна прямо в вікні колекції. При створенні кожного рядка в колекції буде генеруватися унікальний ID схожий на автоіндексацію в реляційниї базах даних, але він не відповідає нумерації і складається як з цифр так і букв. Вигляд даних і згенерованого ідинтифікатора можна побачити на рисунку 3.4 нижче

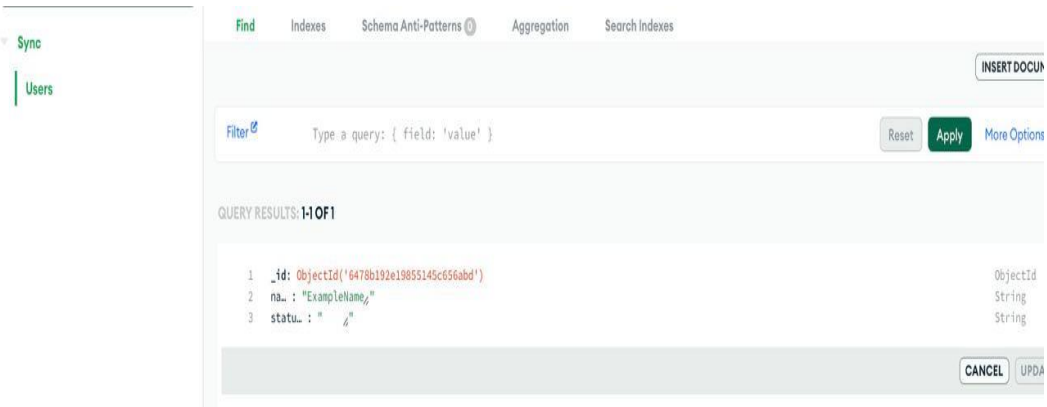


Рисунок 3.4 – Згенерований ідентифікатор та дані

3.2 Розробка програмних модулів

Основний етап розробки в даному випадку передбачає ітеративний процес, втілення різноманітного функціоналу. Ініціюється розробка шляхом запуску npm скриптів, що дозволяють створити початковий проект з підготовленою файловою структурою. В отриманій файловій структурі вже є автоматично згенеровані файли необхідні для роботи з обома основним платформами. Ці файли хгруповані по назвам, в директорії android можна знайти різноманітні файли конфіграції gradle, який менеджить залежності для застосунків на базі андроід систем. В директорії IOS, бачимо файли подів та файл проекту для xcode. За допомогою цих файлів можна запускати емулятори операційних систем для подальшого тестування застосунку. Файлова структура застосунку наведена нижче на рисунку 3.5

Рисунок 3.5 – Файлова структура проекту

Далі розробляється користувацький інтерфейс та реалізується створення екранів застосунку. Крім екранів реєстрації і входу, присутній також головний екран, на якому відображаються користувачі, яких можна додати до кімнати, екран очікування кімнати та екран кімнати з мультимедійним вмістом.

Екрани авторизації включають в себе поля для введення даних користувача для авторизації та аутентифікації. Також присутні посилання на екран з умовами використання застосунку та на екран реєстрації. На екрані реєстрації доступні поля для введення даних для реєстрації користувача.

Далі переходимо на головний екран, на якому відображаються доступні кімнати (кімнати, до яких користувач отримав запрошення), кнопка створення власної кімнати та кнопка переходу на екран контактів.

На екрані контактів користувач може додавати або видаляти контакти. Також є можливість створювати кімнату з запрошенням для обраного контакту та повертатися назад до головного екрану.

Після створення власної кімнати, користувач переходить на екран кімнати, де може вибрати медіа для перегляду в кімнаті та запросити інших користувачів зі своїх контактів. Після підготовки медіафайлу можна натиснути кнопку "Почати" і розпочати перегляд медіа.

На екрані кімнати для гостя відображається медіа для перегляду, а власнику кімнати доступний екран керування, де можна видалити учасників з кімнати, поставити медіа на паузу, зупинити перегляд або перейти до наступного відрізку медіафайлу.

Після створення необхідних екранів для користувацького інтерфейсу потрібно створити логіку взаємодії користувачів. Для цього використовуються модулі з реєстру npm. За допомогою цих модулів реалізується передача мультимедійних файлів з мобільного пристрою власника кімнати на пристрої гостей.

Далі потрібно налагодити контроль стану застосунку за допомогою шаблону Redux, який є стандартом для React та React Native застосунків. Це дозволить оживити створені інтерфейси та додати логіку взаємодії з даними. Для

спрощення реалізації патерна використовується допоміжний модуль react-redux. Схема організаційї даних в redux зображена нижче на рисунку 3.6.

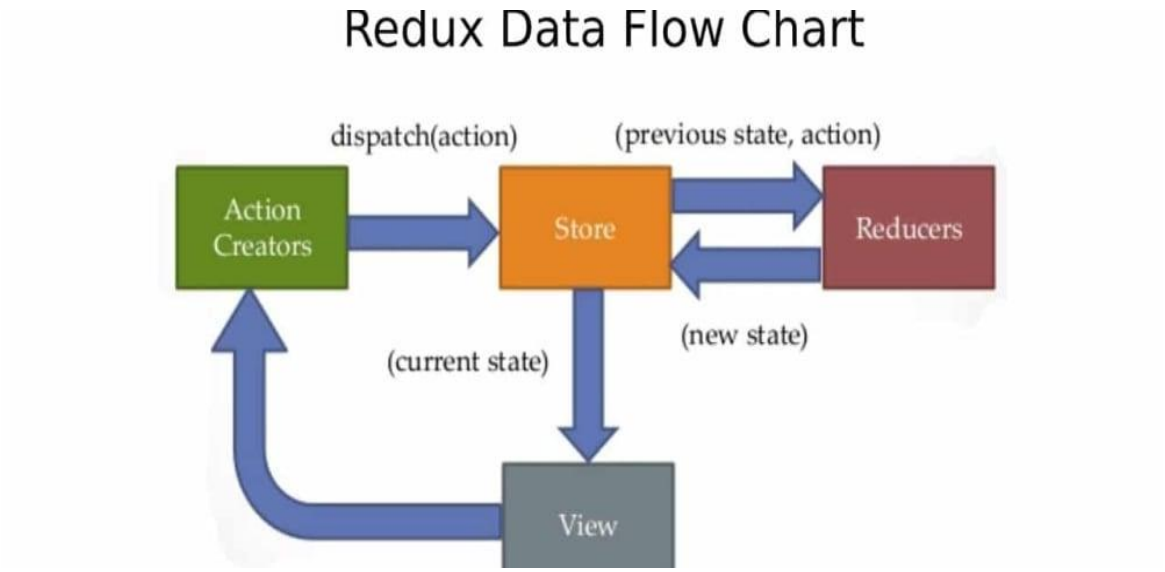


Рисунок 3.6 – Схема організації даних з Redux – https://studiosoftware.com

Далі ми зосередимось на реалізації серверної частини. Для цього ми використовуємо REST API на основі фреймворку Express.js. Процес розробки включає наступні кроки.

Створення маршрутів: Ми створюємо шляхи для ендпоінтів, які будуть використовуватися для звернень до серверу. Ці шляхи повинні відповідати стандартам REST, щоб забезпечити якісну роботу.

REST-архітектура: REST є стилем архітектури для розподілених систем. Вона ґрунтується на кількох стандартах, які допомагають створювати ефективні та масштабовані API. До основних стандартів REST входять:

Клієнт-серверна архітектура: система розділяється на клієнтську та серверну частини, що дозволяє розробникам створювати різні клієнти та сервери, які взаємодіють між собою.

Безстійність: кожен запит клієнта містить усю необхідну інформацію для обробки на сервері, а сервер не зберігає стан клієнта між запитами.

Кешування: система дозволяє кешувати запити клієнта для зменшення кількості запитів до сервера та зменшення часу відповіді.

Єдність ідентифікатора ресурсу: кожен ресурс має унікальний ідентифікатор, що дозволяє однозначно визначати його.

Маніпулювання ресурсами через представлення: клієнт може маніпулювати ресурсами через їх представлення, такі як HTML-форми або JSON-об'єкти.

Система гіперпосилань: клієнт може отримувати інформацію про ресурси за допомогою гіперпосилань, які пов'язують різні ресурси між собою.

Стандартизовані методи: REST використовує стандартні HTTP-методи (GET, POST, PUT, DELETE) для маніпулювання ресурсами.

Створення контролерів: Наступним кроком є створення контролерів, які обробляють маршрути відповідно до логіки проекту. Також необхідно підключити клієнтський проект до сервера.

Тестування і налагодження: Останнім етапом є тестування та налагодження проекту. Для тестування можна використовувати різні бібліотеки, але рекомендується використовувати Jest. Jest забезпечує зручний механізм асертів та можливість створення заглушок різних рівнів абстракції. Варто використовувати Jest для написання юніт-тестів, тестування функціональності на рівні сервісів та тестування самого сервера. Для мобільного застосунку рекомендується використовувати фреймворк Detox, який забезпечує автоматизоване тестування на різних платформах, включаючи iOS та Android.

Загалом фреймворк Detox є потужним інструментом для автоматизованого тестування мобільних застосунків, який може допомогти підтримувати стабільність та надійність мобільного застосунку. Detox є незмінним при роботі з проектами де немає виділеного тестувальника, адже допомагає розробнику не хвилюватись про те, що його зміни спричинять баги в якійсь іншій частині застосунку.

Наступним кроком є розробка серверної частини застосунку. Її суть полягає в розробці роутів, що будуть використовуватись клієнтським мобільним застосунком для отримання чи виставлення даних.

Почати розробку роутів доцільно з авторизації. Спираючись на раніше спроектовану JWT модель, можна виділити наступну схему авторизації. Реєстрація починається з формування даних для запиту в мобільному застосунку. Після запиту на сервері його обробка починається з запиту до хмарного сховища mongoDB, для отримання користувача з електроною поштою, яку було отримано в параметрах запиту. У разі якщо користувача знайдено сервер верне помилку, адже користувач з такою електронною поштою вже зареєстрований. В іншому випадку можна продовжити, реєстрацію, перейшовши до процесу створення нового користувача. На основі параметрів отриманих з запиту створюється запит на додавання нового запису в таблицю користувачів в MongoDB. Після успішного додавання користувача на основі його даних та секретного ключа генерується токен JWT. Токен вертається в відповіді до запиту клієнта разом з даними створеного користувача згідно з стандартами REST. Після отримання токену клієнтський застосунок може відправляти його разом з іншими запитами і сервер зможе аутинтифікувати користовуча без зайвих запитів до бази даних і без частої передачі пароля через вразливий мережевий протокол HTTPs .

Наступним буде маршрут логіну, він буде працювати схожим чином тільки замість створення користувача, після отримання паролю та логіну від клієнта, серверна частина буде хешувати отриманий пароль, та звіряти його з збереженим хешем пароля в базі даних. На основі результатів цієї перевірки буде вертатись токен JWT, або помилка авторизації.

Після завершення роботи над маршрутами авторизації, можна перейти до роботи з даними. Сутності даних представляють собою Кімнати, Користувачі які до них належать і файли.

Кімнати можна створювати за допомогою запиту методу POST, який в свому обробнику буде просто викликати запит до mongoDB, використовуючи інтерфейс класу MongoClient. Це класичний singleton, який надає уніфікований доступ до стандартних запитів бази. Оскільки створення кімнати доступне всім групам користувачів, ніяких додаткових перевірок крім стандартної валідації додавати не потрібно. Результатом запиту є інформація про створену кімнату.

Видалення та редагування кімнат зводиться до перевірки чи є користувач, що робить запит власником кімнати, якщо його право не редагування чи видалення кімнати не підтверджується запит буде вертати помилку. В іншому випадку буде викликаний один з методів класу MongoClient. Результатами видалення буде логічне значення, яке буде вказувати на те чи успішно було видалено запис. Результатом запиту на редагування буде об'єкт який буде представляти запис в базі даних після редагування.

Робота з користувачами які належать до тих чи інших кімнат є в своїй суті окремим випадком редагування інформації про кімнату, так як в документній структурі користувачі, що відносяться до кімнат є просто полем типу масив в кожному окремому записі колекції кімнат. Виходячи з цього робота над реалізацією цієї функціональної вимоги зводиться до формування правильно запиту на клієнтській частині додатку, який буде містити лише зміни списку користувачів, що відносяться до окремої кімнати.

Наступним кроком в реалізації логіки застосунку є розробка навігаційних екранів, згідно до спроектованих схем. Для цього можна використати одну з бібліотек спільноти, яка вирішує більшість проблем навігації в застосунках react-native.

React Navigation є популярною бібліотекою навігації для React-проектів, яка дозволяє реалізувати навігацію між екранами та стеками у мобільних застосунках. Вона спрощує управління станами навігації та надає потужний API для створення різних типів навігаційних патернів.

React Navigation підтримує такі типи навігації:

Stack Navigator: Це найпоширеніший тип навігації, який використовує стекову модель. Він дозволяє навігувати між екранами, додаючи їх в стек. Коли ви переходите до нового екрана, він додається до верху стеку, а коли ви повертаєтесь назад, останній екран видаляється зі стеку. Це дозволяє реалізувати поведінку "назад" на основі стеку екранів.

Bottom Tab Navigator: Цей тип навігації дозволяє розмістити нижню панель з вкладками, які переключаються між різними екранами. Користувачі можуть

торкнутися вкладки, щоб перейти до відповідного екрана. Bottom Tab Navigator зручний для застосунків з багатьма головними розділами або навігацією між основними функціональними областями застосунка.

Drawer Navigator: Цей тип навігації відкриває бічне меню, що з'являється збоку екрана. Він надає можливість навігації до різних екранів за допомогою висувного меню. Відкриття та закриття бічного меню може бути зроблено жестом чи натисканням відповідної кнопки.

В реальних застосунках майже ніколи не використовується лише один з цих навігаторів, як правило це ті чі інші комбінації декількох. В випадку застосунку для віддтворення мультимедійних матеріалів є сенс скористатись Stack Navigator для навігації глибокої вложеної навігації, а також Bottom Navigator для швидкого доступу користувачів до основних функції застосунку.

Згідно з схематичною структурою інтерфейсу Bottom Navigator буде містити три вкладки Profile, Home, Room. Для вирізнення цих екранів можна додати написи та іконки що їх ідентифікують. Логіка виділення окремим кольором активного екрана є вбудованою в Bottom Navigator та є частиною бібліотеки. Після додавання кольорової схеми візуальну частину навігації можна вважати готовою. Результуючий компонент наведений нижче на рисунку 3.7

Рисунок 3.7 – Навігація Bottom Navigation

Далі потрібно створити екрани та логіку синхронізованого перегляду мультмедіа матеріалів. Profile (профіль): Цей екран призначений для відображення інформації про користувача. Він може містити дані, такі як ім'я, аватар, електрону

пошту, статус тощо. Користувачі можуть переглядати свій профіль, редагувати інформацію або виконувати інші дії, пов'язані з профілем.

Home (головна сторінка): Цей екран представляє собою головний розділ додатка, де користувачі переглянути список кімнат які доступні для відвідування. Користувачі можуть взаємодіяти з кімнатами, переглядати деталі, переходити до кімнат.

Room (кімната): Цей екран може бути призначений для навігації до розділу конкретної кімнати, тут міститься інформація про кімнату та можливі дії повязані з нею.

Наступним кроком є налаштування основної логіки відтворення мультимедійних матеріалів за допомогою сокетів. Для налаштування синхронізації відтворення мультимедійних матеріалів за допомогою сокетів у React Native застосунку, знадобиться виконати наступні кроки:

Встановити та налаштувати бібліотеку Socket.IO. Встановити бібліотеку Socket.IO можна використовуючи менеджер пакетів npm чи yarn. Далі потрібно Імпортувати необхідні класи та функції з бібліотеки Socket.IO у React Native проекті. Наступний крок передбачає створення з'єднання з сервером. Для цього потрібно визначити адресу та порт сервера, з яким потрібно встановити з'єднання, використати функцію socket.connect() для підключення до сервера.

Для відправлення та отримання повідомлень, можна використати функцію socket.emit() для відправлення повідомлення з застосунку на сервер. Наприклад, можна відправити повідомлення з командою відтворення мультимедійного матеріалу. Використовуючи функцію socket.on() для слухання подій (повідомлень) від сервера. Наприклад, можна очікувати повідомлення з командами відтворення, що приходять з сервера, і виконувати їх у React Native застосунку.

При отриманні команди від сервера, слід виконати відповідні дії у React Native застосунку для синхронізації відтворення мультимедійних матеріалів. Наприклад, можете запустити відтворення аудіо або відео, перемотати до певного моменту чи призупинити відтворення, базуючись на інформації отриманій в повідомлені.

3.3 Керівництво користувача

Для встановлення застосунку на пристрій потрібно відкрити сторінку застосунку в AppStore, або Google Play.

Після завершення інсталяції можна відкрити застосунок натиснувши на його графічну піктограму на екрані пристрою. Відкривши застосунок користувач потрапляє на головний екран. Вигляд головного екрану можна побачити на рисунку 3.8 нижче.

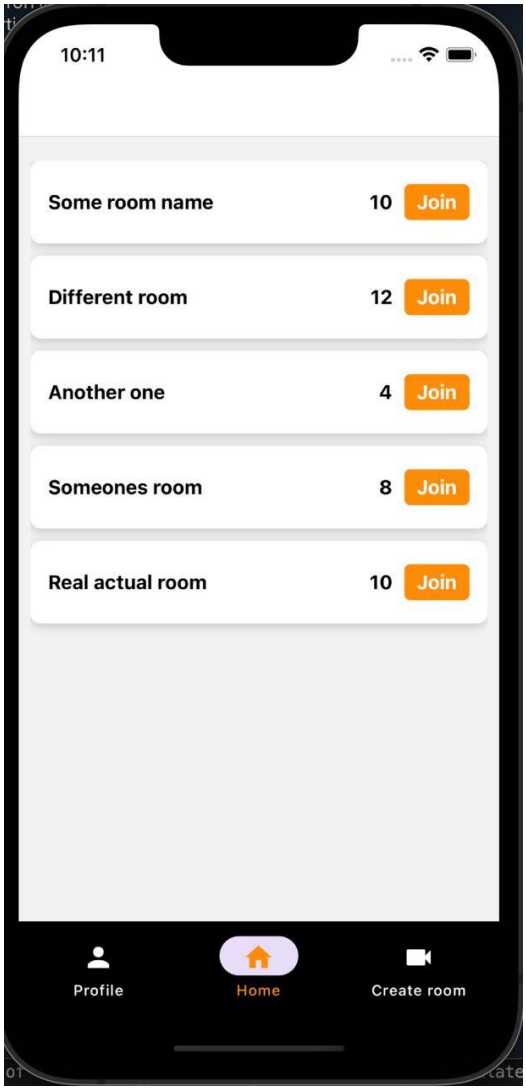


Рисунок 3.8 – Вигляд головного екрану

На головному екрані застосунку можна переглянути список доступних кімнат та приєднатись до них.

Натиснувши на кнопку профілю в нижній навігації користувач може перейти на екран профілю користувача. Тут він може побачити свою інформацію змінити аватар, або ім’я для відображення, а також адмініструвати кімнати які йому належать в списку нижче. Вигляд екрану профілю користувача можна побачити нижче на рисунку 3.9.

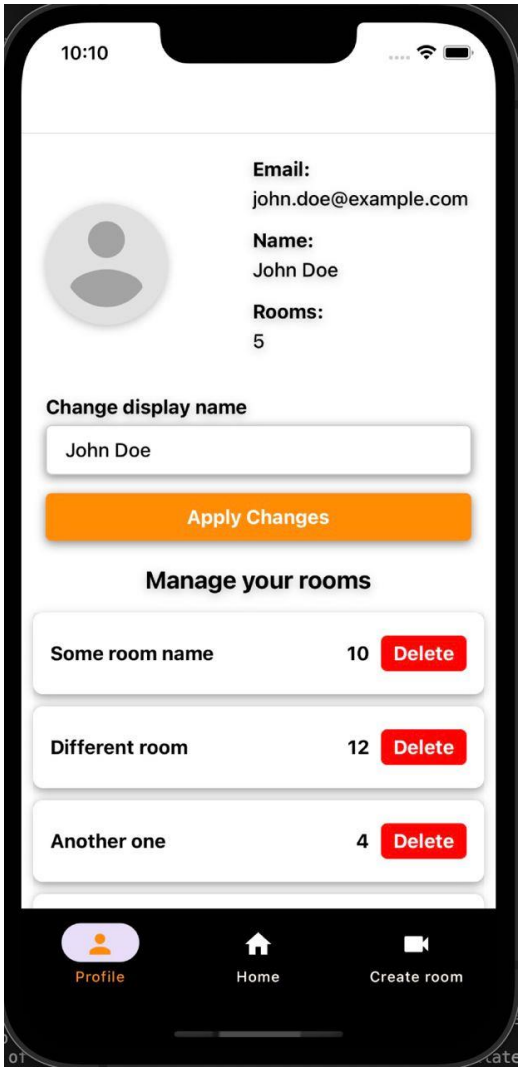


Рисунок 3.9 – Вигляд головного екрану

З головного меню нижньої навігації користувачу також доступний екран створення кімнати. На цьому екрані можна побачити інтерфейс для вибору мультимедіа матеріалу та список запрошених користувачів. Є кнопка для запрошення додаткових користувачів. В рядку кожного користувача є піктограми

кнопок вилучення користувача з кімнати та редагування ролі користувача для адміністратору. Вигляд екрану можна побачити нижче, на рисунку 3.10.

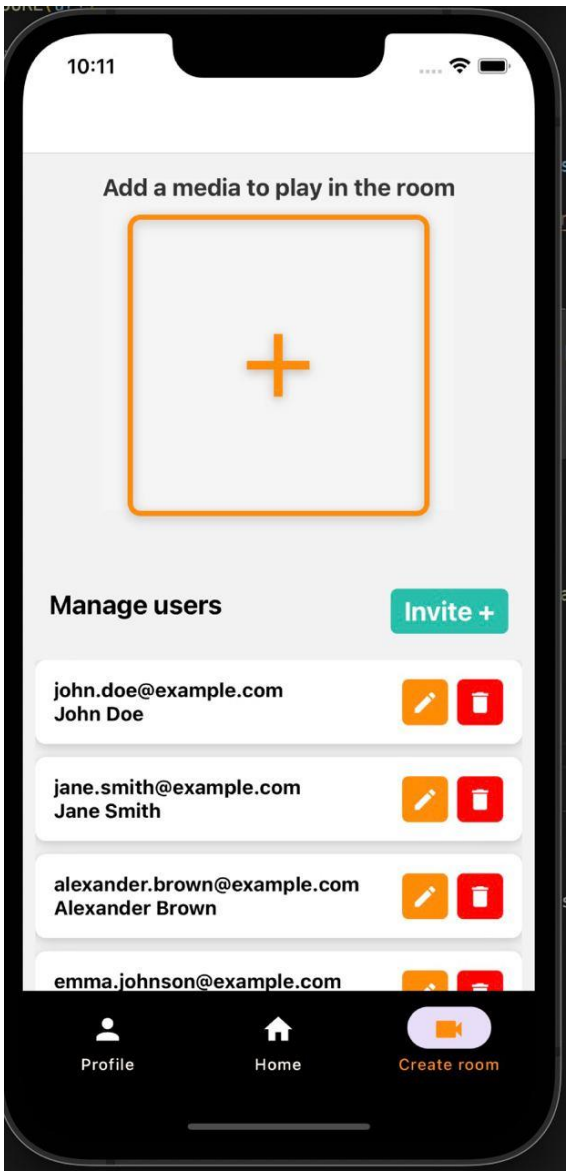


Рисунок 3.10 – Вигляд екрану створення кімнати

По натисканю на кнопку “+” користувач може відкрити модальне вікно для додавання мультимедіа матеріалу. На формі, що представлена в модальному вікні можна вибрати тип мультимедійниого матеріалу з одного з трьох представлених: відео, аудіо, або презентаційний матеріал. Після вибору типу потрібно обрати посилання на сам матеріал та натиснути зберегти. З виглядом

модального вікна для вбору мультимедійного матеріалу можна ознайомитися нижче на рисунку 3.11.

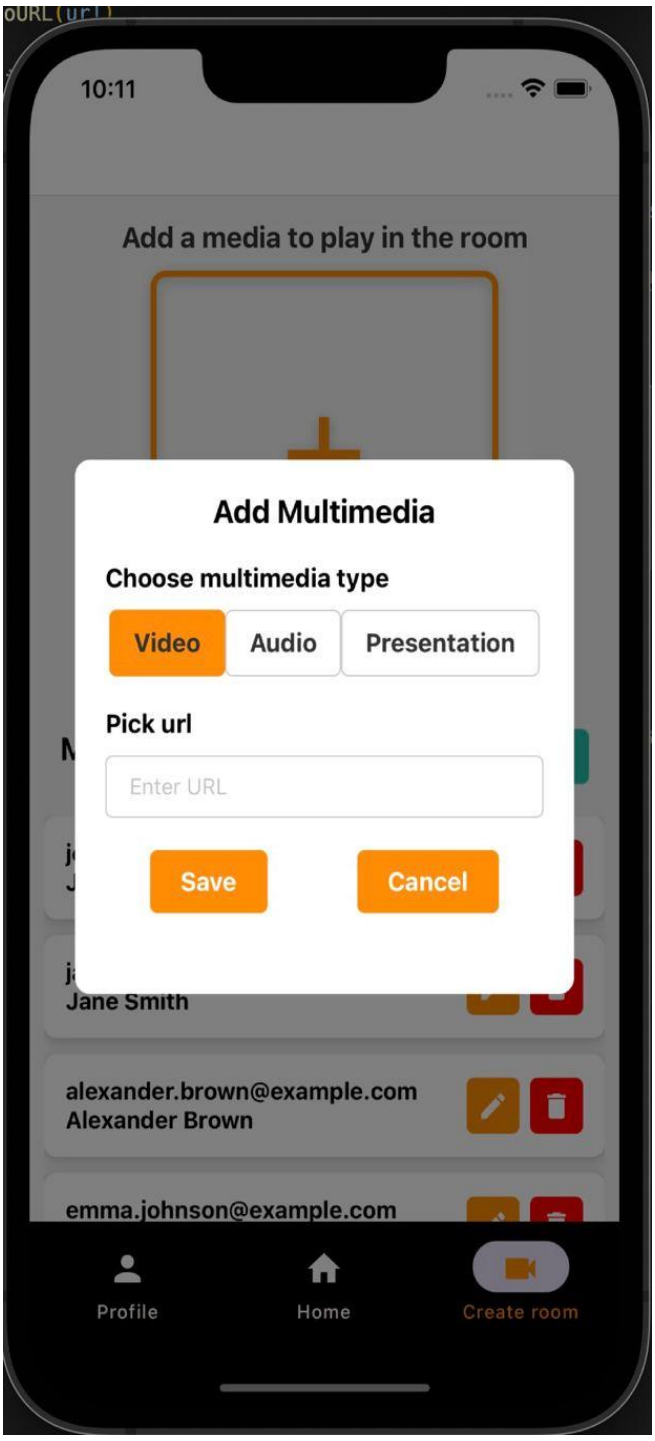


Рисунок 3.11 – Вигляд модального вікна

Після вибору мультимедійного матеріалу модальне вікно закривається і можна побачити компонент відтворення мультимедійних матеріалів. Компоненти

керування відтворенням знаходяться всередині компонента відтворення. Наприклад натиснувши кнопку для перематування відео матеріалу на 10 секунд вперед, відеоматеріал продовжить відтворення з позначки на 10 секунд попереду для кожного користувача в цій кімнаті, аналогічно компонент працює з призупиненям відтворення, та пермотуванням. З виглядом компонента відтворення в портретному режимі можна ознайомитись на рисунку 3.12. нижче

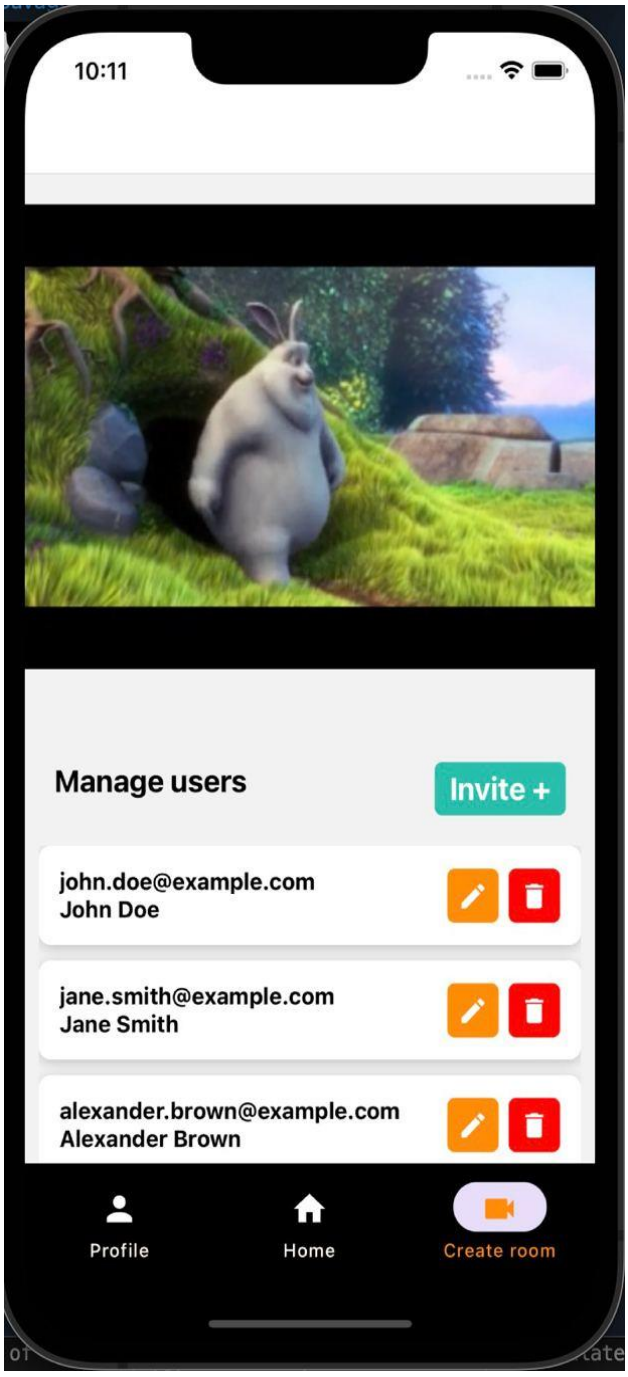


Рисунок 3.12 – Вигляд компонента відтворення

3.4 Технічні характеристики застосунку

Мінімальні технічні характеристики для пристрою, на якому можна запускати застосунок для синхронізованого відтворення мультимедійних матеріалів, можуть варіюватись залежно від конкретних дій користувача в застосунку. Нижче наведені загальні рекомендації для запуску такого застосунку:

Операційна система:

- Android: Android 5.0 (Lollipop) або новіше;
- iOS: iOS 10 або новіше;

Процесор:

- Android: ARM або x86 процесор з 1,8 ГГц чи вище;
- iOS: 64-бітовий процесор;

Мінімум 2 ГБ оперативної пам'яті. Однак, для кращої продуктивності рекомендується мати 4 ГБ або більше.

Для встановлення застосунку потрібно мати деякий обсяг внутрішньої пам'яті. Залежно від розміру даних, з якими буду працювати застосунок, рекомендується мати принаймні 16 ГБ внутрішньої пам'яті.
Рекомендована мінімальна роздільна здатність екрану становить 720x1280 пікселів. Наявність Wi-Fi або мобільного зв'язку для здійснення мережевих запитів. Батарея з достатньою ємністю для тривалого використання застосунку.

3.5 Розгортання та встановлення системи

Для успішного впровадження цього проекту потрібно мати конкретні компоненти встановлені на вашому комп'ютері, зокрема Node.JS та налаштувати підключення до хмарних сервісів MongoDB. Розпочати роботу можна з відкриття папки проекту і запуску двох терміналів.

Для запуску серверної частини виконайте такі дії:

- Перейдіть до теки в якій знаходиться код API, тобто "server" у терміналі за допомогою команди "cd server".

- Виконайте команду "npm install" для встановлення залежностей проекту.
- Запустіть команду "node index.js" для запуску серверної частини.

Для запуску середи розробника та розгортання для клієнтської частини мобільного застосунку потрібно виконати декілька кроків, починаючи встановленням необхідних бібліотек і закінчуючи запуском емуляторів. Ці кроки описані нижче

Встановіть пакет Expo CLI, відкрийте термінал або командний рядок і виконайте команду npm install -g expo-cli для глобального встановлення Expo CLI. Це інструментарій, який допоможе вам створити та управляти React Native проектами. Перейдіть до папки проекту: В терміналі або командному рядку перейдіть до каталогу вашого нового проекту, виконавши команду cd project-name, де project-name - це назва вашого проекту.

Запустіть проект, виконайте команду expo start для запуску проекту. Це запустить Expo DevTools в браузері і покаже QR-код.

Запустіть застосунок на пристрої: Встановіть Expo Client на свій мобільний пристрій з App Store або Google Play. Відскануйте QR-код, що з'явився в браузері після запуску Expo DevTools, за допомогою Expo Client. Це запустить ваш застосунок на мобільному пристрої.

Перевірте результат, тепер ви повинні бачити ваш застосунок, який запускається на мобільному пристрої. Ви можете редагувати файлів проекту, і зміни будуть автоматично оновлюватись у вашому застосунку.

Для запуску проекту на емуляторі необхідно встановити Android Studio або XCode в залежності від необхідної операційної системи емулятору. Далі потрібно виконати інструкції відповідних IDE по запуску емуляторів.

У цьому описі було пояснено процес створення бази даних та її підключення до Застосунку. Також була розглянута декомпозиція системи на модулі й надано детальний опис кожного модуля.

Визначені технічні характеристики інтернет-платформи. Надано детальний алгоритм для встановлення, розгортання та використання розробленого програмного продукту.

3.6 Тестування застосунку

Тестування застосунку для синхронізованого відтворення мультимедійних матеріалів за допомогою Detox може включати наступні кроки:

Для встановлення detox потрібно переконатися, що на комп'ютері встановлено Node.js та npm. Далі слід відкрити термінал або командний рядок і виконати команду npm install -g detox-cli для глобального встановлення Detox CLI.

Для налаштування проекту потрібно додати Detox до React Native проекту. Зазвичай це вимагає налаштування в файлі package.json та створення Detox конфігураційного файлу для вашої платформи (Android або iOS).

Створення тестів для перевірки синхронізованого відтворення мультимедійних матеріалів.Слід використовувати Detox API для взаємодії з застосунком, включаючи запуск відтворення, перевірку стану відтворення, паузу, перемотку і т.д.

Конфігурація тестового середовища полягає в налаштуванні тестового середовища для Detox, включаючи зазначення конкретних пристроїв або емуляторів, на яких будуть виконуватись тести. Detox може працювати з різними Android емуляторами або пристроями iOS, які будуть налаштовані.

Для запуску тестів можна виконати команду detox test в терміналі або командному рядку, щоб запустити тести. Detox автоматично взаємодіє з застосунком на налаштованому пристрої або емуляторі, виконуючи послідовні дії, які були вказані в тестах.

Аналіз результатів: Після завершення тестів Detox надасть звіт про результати, який можна проаналізувати. На основі чого можна перевірити, чи пройшли тести успішно або чи виникли помилки під час відтворення мультимедійних матеріалів.

Якщо в тестах виникають проблеми або помилки, їх слід виправити , редагуючи ваш застосунок або тести. Повторювання циклу написання тестів, запуску і аналізу результатів, має сенс доки застосунок для синхронізованого

відтворення мультимедійних матеріалів не пройде всі необхідні перевірки. З кодом написаних тестів Detox можна ознайомитися нижче на рисунку 3.13.

```
describe('Мультимедійний відтворювач', () => {
  beforeEach(async () => {
    await device.reloadReactNative();
  });

  it('Перевірка початкового стану відтворення', async () => {
    await expect(element(by.id('відтворення_екран'))).toBeVisible();
    await expect(element(by.id('кнопка_відтворення'))).toBeVisible();
    await expect(element(by.id('кнопка_паузи'))).toBeVisible();
    await expect(element(by.id('кнопка_перемотки'))).toBeVisible();
  });

  it('Тест відтворення мультимедійного матеріалу', async () => {
    await element(by.id('кнопка_відтворення')).tap();
    await expect(element(by.id('статус_відтворення'))).toHaveText('Відтворюєть
    // Додаткові перевірки стану відтворення можна додати тут
  });

  it('Тест паузи відтворення мультимедійного матеріалу', async () => {
    await element(by.id('кнопка_відтворення')).tap();
    await element(by.id('кнопка_паузи')).tap();
    await expect(element(by.id('статус_відтворення'))).toHaveText('Пауза');
  });

  it('Тест перемотки відтворення мультимедійного матеріалу', async () => {
    await element(by.id('кнопка_відтворення')).tap();
    // Виконати дії перемотки (наприклад, пересунути повзунок до певного часового
    await element(by.id('повзунок_відтворення')).setSliderValue(0.5);
    await expect(element(by.id('поточний_час'))).toHaveText('00:30');
  });
});
```

Рисунок 3.13 – Вигляд тестів Detox

Висновки:

Отже було заверешено програмну реалізацію застосунку для відтворення мультимедійних матеріалів. Результатами став код серверної частини застосунку та клієнтський код написаний за допомогою технології react native. Використання даної технології дозволило використовувати одну і ту саму базу коду для розробки застосунку на платформах IOS та Android. За допомогою бібліотеки спільности з реєстру npm React Navigation, були розроблені основні екрани застосунку та методи навігації між ними. Крім того було створено документну базу даних за допомогою хмарних сервісів MongoDB. Було також написано ряд тестів використовуючи технології jest та detox для серверних та клієнтських тестів відповідно.

Основним результатом етапу програмування став застосунок для синхронізації відтворення мультимедійних матеріалів. Програмний продукт відповідає всім функціональни і технічним вимогами.

ВИСНОВКИ

Отже під час виконання кваліфікаційної роботи на тему "Реалізація мобільного застосунку для синхронізації відтворення мультимедйіних матеріалів на різних пристроях" було розглянуто предметну область, було виявилено характерні для неї проблеми та невирішені завдання. На основі чого була виведена актуальність розробки застосунку. На підставі проведення аналізу аналогічних застосунків, було виявлено їх переваги і недоліки.

Були також визначені функціональні, нефункціональні та технічні вимоги до програмного продукту. На основі даних вимог було сформовано технічне завдання, а також були проаналізовані можливі варіанти використання застосунку.

В рамках даної роботи був розроблений мобільний застосунок для синхронізації відтворення мультимедійних матеріалів. Застосунок надає можливість синхронізувати відтвоерення відео, аудіо, або презентаційних матеріалів.

Для реалізації проекту було використано фреймворк React Native для реалізації клієнтської частини та Node.js з використанням Express.js для створення REST API. MongoDB була обрана у якості бази даних для зберігання інформації про користувачів та кімнат відтворення мультимедіа. Для довгострокового зберігання мультимедійних файлів було використано Amazon S3

Застосунок має простий та зрозумілий інтерфейс, що дозволяє користувачам легко орієнтуватись в навігації застосунку та отримати доступ до більшості функцій програмного продукту..

Для забезпечення контролю якості та надійності мобільного застосунку, було проведено тестування різних його частин та функціональних модулів. Окремо було проведено тестування серверної частини через створення модульних тестів на основі JEST та тестування клієнтських інтеракцій з інтерфейсом за допомогою тестувального фреймворку Detox.

В результаті роботи всі поставлені завдання були вирішені, а саме створено функціональний мобільний застосунок для синхронізації відтворення

мультимедійних матеріалів. Всі функціональні, не функціональні та технічні вимоги були виконані.

Проект має потенціал для подальшого розширення, наприклад інтеграція з сторонніми стрімінговими сервісами, такими як: YouTube, Spotify, Vimeo, Netflix, Soundcloud та іншими.

Крім того існує потреба збільшення можливостей інтеракцій користувачів під час відтворення матеріалів, а також збільшення можливостекй адміністрування кімнат під час перегляду та покращення протоколів передачі даниз для досягнення вщого рівня безпеки збережених мультимедіа матеріалів.

В цілому, мобільний застосунок було реалізовано успішно. Зважаючи на стан предметної області та якість отриманого програмного продукту можна розраховувати на комерційний успіх застосунку та широку популярність серед користувачів мобільних пристроїв на базі iOS та Android.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Л. П. Бедратюк. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун. О. М. Яшина. Хмельницький: ХНУ, 2020. – 77с . (дата звернення – 07.05.2023).
2. React.dev документація: веб-сайт. URL: https://react.dev. (дата звернення – 14.04.2023).
3. Документація React Native: веб-сайт. URL: https://reactnative.dev/. (дата звернення – 01.04.2023).
4. Leonard Richardson, Mike Amundsen, Sam Ruby. RESTful Web APIs – 72 с. (дата звернення – 28.03.2023).
5. Lindstorn, Steve. CSS Refactoring – New York: O'Relly Media, 2016. – 159с. . (дата звернення – 02.04.2023).
6. Legacy документація React: веб-сайт. URL: https://uk.reactjs.org/. (дата звернення – 12.04.2023).
7. Документація фреймворку тестування Detox: веб-сайт. URL: https://wix.github.io/Detox/docs/introduction. (дата звернення – 12.04.2023).
8. Кантелон Майк, Хартер Марк, Головайчук TJ, Райлих Натан. Node.js в дії, друге видання: "Издательский дом ""Питер""". – 123с.
9. Боні Ейсенман . Вивчення React Native. – 18с.
10. Девід Гріфітс, Доун Гріфітс. Кулінарна книга React: Рецепти для опанування React Framework, 1 видання – 89с.
11. Документація реляційної бази даних PostgresSQL: веб-сайт. URL: https://www.postgresql.org/docs/. (дата звернення – 16.04.2023).
12. Сучасний посібник з вивчення javascrip: веб-сайт. URL: https://uk.javascript.info/. (дата звернення – 19. 04.2023).
13. Метт Фрісбі. JavaScript для професійних веброзробників – 53с.
14. Девід Херон. Node.js. Розробка серверних застосунків на мові JavaScript – 123с

15. Бред Дейлі, Брендан Дейді, Калєб Дейбі. Розробка веб-застосунків за допомогою Node.js, MongoDB: – 123с.

16. Документація express: веб-сайт. URL: https://expressjs.com/. (дата звернення – 18.04.2023).

17. Документація документної бази даних MongoDb: веб-сайт. URL: https://www.mongodb.com/docs//. (дата звернення – 18.03.2023).

18. Watch2Gather: веб-сайт. URL: https://w2g.tv/en/. (дата звернення – 19.04.2023).

19. SyncPlay: веб-сайт. URL: https://syncplay.pl/ . (дата звернення – 22.04.2023).

20. Салім Сіддікі. Навчання розробки на основі тестування: Посібник для поліглотів із написання лаконічного коду, 1-е видання – 115с

21. Джунтао Цю. Test-Driven Development with React. 1st Ed. (english) – 12с

22. Герард Месарош. xUnit Test Patterns: Refactoring Test Code – 221с

23. Введення в Redux [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/. (дата звернення – 08.05.2023).

24. Уніфікована мова програмування UML: портал знань. URL: http://www.znannya.org/?view=uml. (дата звернення – 11.05.2023).

25. Документація Redux : веб-сайт. URL: https://redux.js.org (дата звернення – 11.05.2023).

26. NOSQL – Основні принципи нереляційних баз: веб-сайт. URL: https://aws.amazon.com/. (дата звернення – 04.05.2023).

27. S3 – Як це працює: веб-сайт. URL: https://aws.amazon.com/. (дата звернення – 04.05.2023).

28. Amazon DynamoDB: веб-сайт. URL: https://aws.amazon.com/. (дата звернення – 01.05.2023).

29. ScyllaDB vs DynamoDB заміри продуктивності: веб-сайт. URL: https://www.scylladb.com/. (дата звернення – 01.05.2023).

30. Документація для бібліотеки Socket.io: веб-сайт. URL: https://socket.io/. (дата звернення – 04.05.2023).

31. Бібліотека Socket.IO: що це таке і як з нею працювати: веб-сайт. URL: https://hexlet.io/. (дата звернення – 04.05.2023).

32. Різниця між веб-сокетами та Socket.IO: веб-сайт. URL: https:// habr.com /. (дата звернення – 22.04.2023).

33. Керівництво по Socket.IO: веб-сайт. URL https:// my-js.org /. (дата звернення – 18.04.2023).

34. Документація Jest: веб-сайт. URL: https://jestjs.io/ . (дата звернення – 02.05.2023).

35. Документація React Navigation: веб-сайт. URL: https://reactnavigation. org/. (дата звернення – 02.04.2023).

36. Навігація React Native посібник з прикладами: веб-сайт. URL: https://logrocket.com/. (дата звернення – 03.04.2023).

37. Налаштування стеку навігації в React: веб-сайт. URL: https://jscamp.app. (дата звернення – 05.04.2023).

38. Міст до нативних модулів React Native: веб-сайт. URL: https://medium.com. (дата звернення – 05.04.2023).

39. Що таке “Міст” в React: веб-сайт. URL: https://geeksforgeeks.org. (дата звернення – 02.04.2023).

40. Створення нативного Мосту в React: веб-сайт. URL: https:// mobilityquotient.com (дата звернення – 018.04.2023).

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях. Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Мобільний застосунок для синхронізації відтворення мультимедійних матеріалів на різних пристроях.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням додатку є клієнтська частина застосунку створенна для перегляду відео, аудіо, або презентаційних матеріалів синхроніховано з іншими пристроями.

2.2 Експлуатаційне призначення

Програма повинна експлуатуватися на мобільних пристроях на базі систем iOS та Android. Кінцевим користувачем додатку може виступати будь-яка особа.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Для користувача:

- відтворення різних типів мультимедійних файлів;
- синхронізація відттворення мультмедійних матеріалів;
- можливість увправління наявними мальтимедійними матеріалами;
- можливість видалення та завантаження різних медійних файлів
- можливість авторизації користувача;
- можливість перегляду та зміни особистих даних користувача;
- можливість голосувати за пауза під час перегляду мультмедійного матеріалу;

Для адміністратора:

- можливість створення кімнати для перегляду медіа матеріалу;
- можливість керувати переглядом відеоматеріалу;
- можливість керувати прослуховуванням аудіоматеріалу;
- можливість видаляти кімнати перегляду;
- можливість вилучати користувачів з кімнати перегляду;
- можливість надавати права адміністратора, користувачу в кімнаті;
- можливість керувати доступністю кімнати;
- можливість запрошувати користувачів в кімнату;

3.2 Вимоги до надійності

Застосунок повинен забезпечувати такі вимоги до надійності:

- обробляти невірні дії користувача і попереджати його про можливі наслідки;
- можливість самостійно відновлюватись у разі збою;
- можливість резервного копіювання даних.

3.3 Умови експлуатації та вимоги до технічних засобів

Застосунок повинен працювати на всіх пристроях, які мають в своїй основі систему Android, або iOS, стабільний доступ до мережі «Інтернет».

Для роботи платформи без збоїв, потрібно, щоб пристрій, на якому вона запускається задовольняв наступні мінімальні вимоги:

- Операційна система Android 5.0 (Lollipop) або новіше. IOS 10 або новіше;
- Процесор ARM або x86 процесор з 1,8 ГГц чи вище, або 64-бітовий процесор;
- Мінімум 2 ГБ оперативної пам'яті;
- Залежно від розміру даних, з якими буду працювати застосунок, рекомендується мати принаймні 16 ГБ внутрішньої пам'яті;
- Рекомендована мінімальна роздільна здатність екрану становить 720x1280 пікселів;

– Наявність Wi-Fi або стійкого мобільного зв'язку для здійснення мережевих запитів;

- Батарея з достатньою ємністю для тривалого використання застосунку;

3.4 Вимоги до інформаційної та програмної сумісності

Для розробки застосунку був використаний фронтенд JavaScript фреймворк React native, бекенд фреймворк express, та хмарна база даних Mongodb.

3.5 Спеціальні вимоги

Програма повинна мати зручний та зрозумілий інтерфейс користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

– текст програми;

– опис програми;

– технічне завдання;

– керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки веб-застосунку для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.23 – 31.01.23	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.23 – 26.02.23	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури

Кінець таблиці А.1

1	2	3
Технічний проект 29.02.23 – 19.03.23	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми
Робочий проект 20.03.23 – 15.04.23	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 23.04.23 – 30.04.23	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування застосунку.

ДОДАТОК Б
(обов’язковий)

ДІАГРАМИ

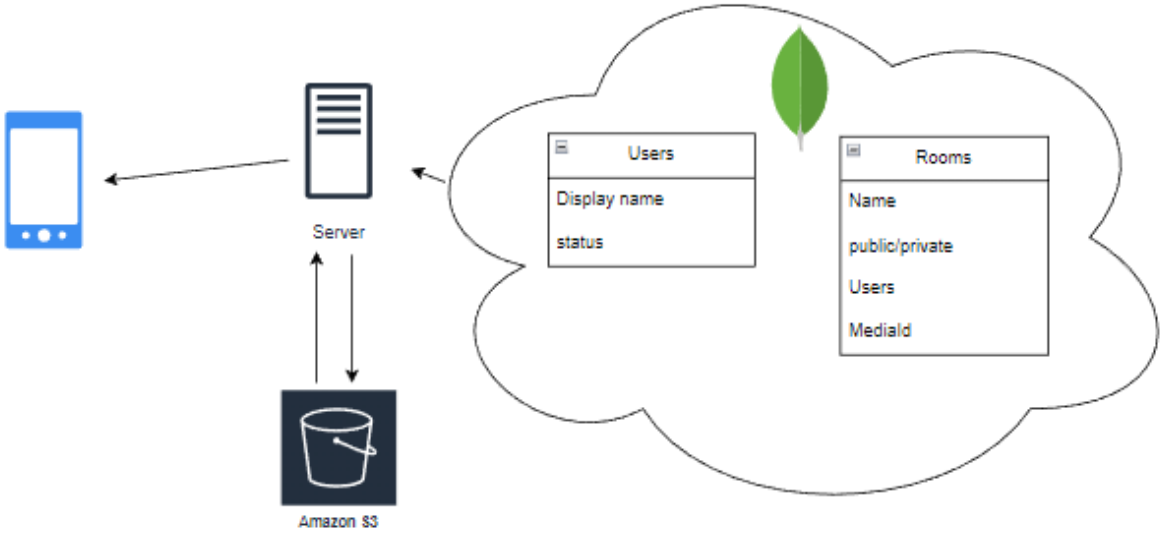


Рисунок Б.1 – Схема бази даних

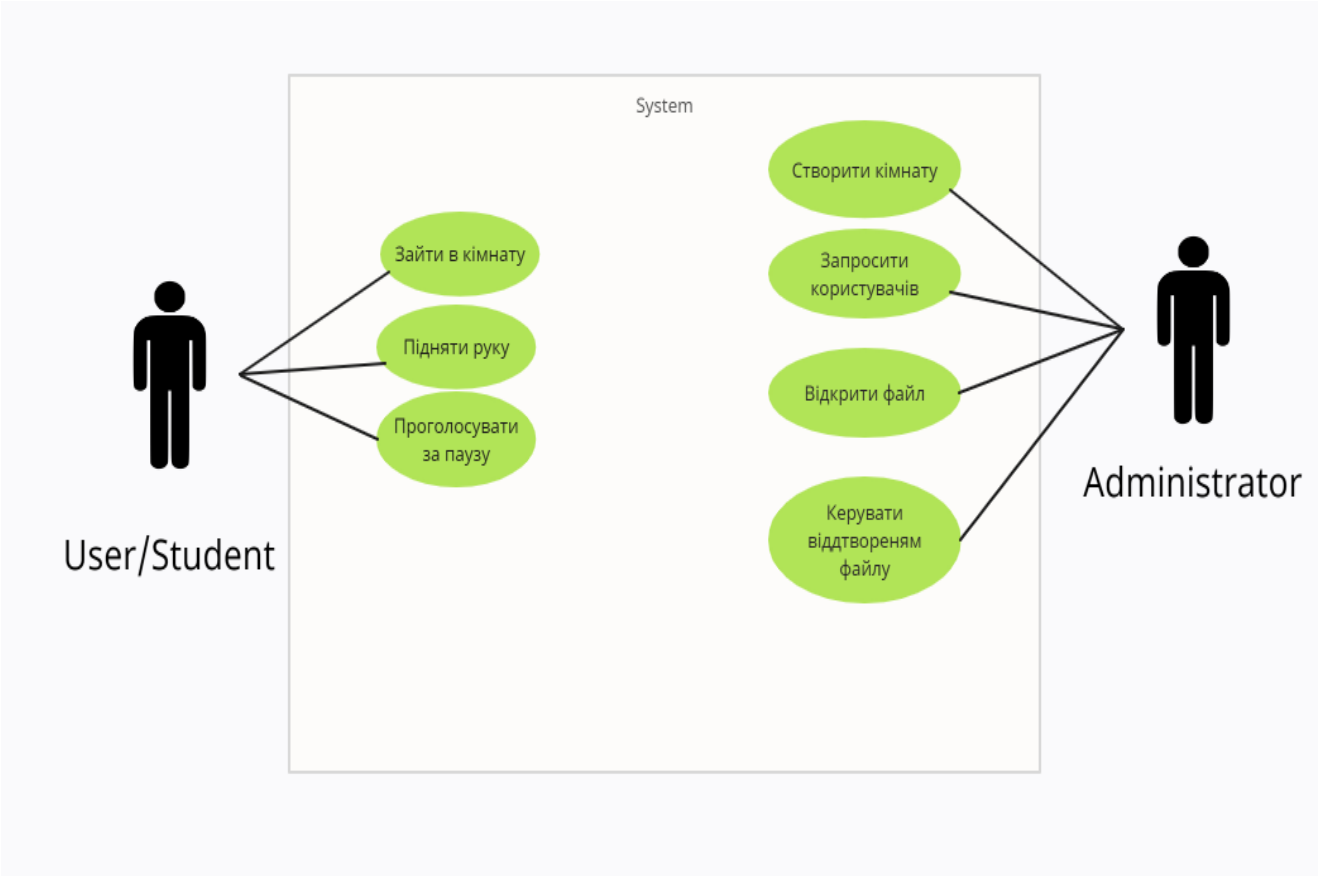


Рисунок Б.2 – Діаграма варіантів використання

ДОДАТОК В
(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

MultiMediaPicker.js

```
import React, { useState } from 'react';
import { View, Text, StyleSheet, Modal, TextInput, TouchableOpacity } from 'react-native';
import MaterialCommunityIcons from 'react-native-vector-icons/MaterialCommunityIcons';

const MultimediaModal = ({ visible, onClose }) => {
  const [mediaType, setMediaType] = useState('video');
  const [url, setURL] = useState('');

  const handleSave = () => {
    // Implement logic to save the media type and URL
    // Close the modal
    onClose(mediaType, url);
  };

  return (
    <Modal visible={visible} animationType="slide" transparent>
      <View style={styles.modalContainer}>
        <View style={styles.modalContent}>
          <Text style={styles.title}>Add Multimedia</Text>

          <Text style={styles.textLabel}>Choose multimedia type</Text>
          <View style={styles.radioGroup}>
            <TouchableOpacity
              style={[styles.radioButton, mediaType === 'video' && styles.radioButtonSelected]}
              onPress={() => setMediaType('video')}
            >
              <Text style={styles.radioLabel}>Video</Text>
            </TouchableOpacity>
            <TouchableOpacity
              style={[styles.radioButton, mediaType === 'audio' && styles.radioButtonSelected]}
              onPress={() => setMediaType('audio')}
            >
              <Text style={styles.radioLabel}>Audio</Text>
            </TouchableOpacity>
            <TouchableOpacity
              style={[styles.radioButton, mediaType === 'presentation' && styles.radioButtonSelected]}
              onPress={() => setMediaType('presentation')}
            >
              <Text style={styles.radioLabel}>Presentation</Text>
            </TouchableOpacity>
          </View>

          <Text style={styles.textLabel}>Pick url</Text>
          <TextInput
            style={styles.textInput}
            value={url}
            onChangeText={setURL}
            placeholder="Enter URL"
          />

        <View style={styles.ButtonGroup}>
          <TouchableOpacity style={styles.button} onPress={handleSave}>
            <Text style={styles.buttonText}>Save</Text>
          </TouchableOpacity>
          <TouchableOpacity style={styles.button} onPress={onClose}>
            <Text style={styles.buttonText}>Cancel</Text>
          </TouchableOpacity>
        </View>
        </View>
      </View>
    </Modal>
  );
};

const styles = StyleSheet.create({
  modalContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
```

```
    backgroundColor: 'rgba(0, 0, 0, 0.5)',
  },
  modalContent: {
    backgroundColor: '#FFFFFF',
    borderRadius: 10,
    padding: 20,
    marginHorizontal: 20,
    alignItems: 'center',
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 20,
  },
  radioGroup: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    marginBottom: 20,
  },
  ButtonGroup: {
    width: 290,
    flexDirection: 'row',
    justifyContent: 'space-around',
    marginBottom: 20,
  },
  radioButton: {
    borderWidth: 1,
    borderColor: '#CCCCCC',
    borderRadius: 5,
    paddingVertical: 10,
    paddingHorizontal: 15,
  },
  radioButtonSelected: {
    backgroundColor: '#FF8C00',
    borderColor: '#FF8C00',
  },
  radioLabel: {
    fontSize: 16,
    fontWeight: 'bold',
    color: '#333333',
  },
  textLabel: {
    fontSize: 16,
    fontWeight: 'bold',
    marginBottom: 10,
    alignSelf: 'flex-start',
  },
  textInput: {
    width: 290,
    borderWidth: 1,
    borderColor: '#CCCCCC',
    borderRadius: 5,
    marginBottom: 20,
  },
  button: {
    backgroundColor: '#FF8C00',

  },
  buttonText: {
    color: '#FFFFFF',
    fontSize: 16,
    fontWeight: 'bold',
  },
});

export default MultimediaModal;
```

Navigation/BottomTabs.js

```
import { createMaterialBottomTabNavigator } from '@react-navigation/material-bottom-tabs';
import MaterialCommunityIcons from 'react-native-vector-icons/MaterialCommunityIcons';
import HomeScreen from '../../Screens/Home';
```

```
import ProfileScreen from '../../Screens/Profile';
import RoomManagemenScreen from '../../Screens/RoomManagement';

const Tab = createMaterialBottomTabNavigator();

export function BottomTabs() {
  return (
    <Tab.Navigator
      initialRouteName="Feed"
      activeColor="#FF8C00"
      inactiveColor='white'
      barStyle={{ backgroundColor: 'black' }}
    >
      <Tab.Screen
        name="Profile"
        component={ProfileScreen}
        options={{
          tabBarLabel: 'Profile',
          tabBarIcon: ({ color }) => (
            <MaterialCommunityIcons name="account" color={color} size={26} />
          ),
        }}
      />
      <Tab.Screen
        name="Feed"
        component={HomeScreen}
        options={{
          tabBarLabel: 'Home',
          tabBarIcon: ({ color }) => (
            <MaterialCommunityIcons name="home" color={color} size={26} />
          ),
        }}
      />
      <Tab.Screen
        name="Room Creating"
        component={RoomManagemenScreen}
        options={{
          tabBarLabel: 'Create room',
          tabBarIcon: ({ color }) => (
            <MaterialCommunityIcons name="video" color={color} size={26} />
          ),
        }}
      />
    </Tab.Navigator>
  );
}
```

RoomList.js

```
import { FlatList, View, Text, StyleSheet, TouchableOpacity, Dimensions } from "react-native";

const renderItem = ({ item }) => (
    <View style={styles.row}>
    <Text style={styles.name}>{item.name}</Text>
    <View style={{ flexDirection: 'row', alignItems: 'center'}}>
    <Text style={styles.number}>{item.number}</Text>
    <TouchableOpacity style={styles.button}>
      <Text style={styles.buttonText}>Join</Text>
    </TouchableOpacity>
    </View>
  </View>
);

const adminRenderItem = ({ item }) => (
    <View style={styles.row}>
    <Text style={styles.name}>{item.name}</Text>
    <View style={{ flexDirection: 'row', alignItems: 'center'}}>
    <Text style={styles.number}>{item.number}</Text>
    <TouchableOpacity style={[styles.button, { backgroundColor: '#FF0000' }]}>
      <Text style={styles.buttonText}>Delete</Text>
    </TouchableOpacity>
    </View>
```

```
  </View>
);

export const RoomList = ({ data, adminRoomList = false}) => {
 return (<FlatList
    style={{ flex: 1 }}
    data={data}
    renderItem={adminRoomList ? adminRenderItem : renderItem}
    keyExtractor={(item) => item.id}
  />)
}


const windowWidth = Dimensions.get('window').width;
const rowWidth = windowWidth * 0.95;

const styles = StyleSheet.create({
  row: {
    width: rowWidth,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    padding: 14,
    backgroundColor: '#FFFFFF',
    borderRadius: 8,
    shadowColor: '#000000',
    shadowOpacity: 0.2,
    shadowOffset: {
      width: 0,
      height: 2,
    },
    shadowRadius: 4,
    elevation: 3,
    marginBottom: 10,
  },
  name: {
    fontSize: 16,
    fontWeight: 'bold',
    textAlign: 'left',
  },
  number: {
    fontSize: 16,
    fontWeight: 'bold',
    textAlign: 'right',
    margin: 10,
  },
  button: {
    backgroundColor: '#FF8C00',
    borderRadius: 5,
    paddingVertical: 5,
    paddingHorizontal: 10,
  },
  buttonText: {
    color: '#FFFFFF',
    fontSize: 16,
    fontWeight: 'bold',
  },
});
```

HomeScreen.js

```
import { View } from 'react-native'
import { RoomList } from "../RoomList";
const AVAILABLE_ROOMS = [
  {
    id: '33213213451',
    name: 'Some room name',
    number: '10'
  },
  {
    id: '332132132122',
```

```
    name: 'Different room',
    number: '12',
  },
  {
    id: '3321321328',
    name: 'Another one',
    number: '4',
  },
  {
    id: '3321321322',
    name: 'Someones room',
    number: '8',
  },
  {
    id: '3321321326',
    name: 'Real actual room',
    number: '10',
  }
]


function HomeScreen() {
  return (
    <View style={{ flex: 1, alignItems: 'center', marginTop: 20, justifyContent: 'center' }}>
      <RoomList data={AVAILABLE_ROOMS}/>
    </View>
  );
}


export default HomeScreen
```

ProfileScreen.js

```
import React, { useState } from 'react';
import { View, Text, StyleSheet, Image, TextInput, TouchableOpacity } from 'react-native';
import { RoomList } from '../RoomList';

const AVAILABLE_ROOMS = [
    {
        id: '33213213451',
        name: 'Some room name',
        number: '10'
    },
    {
        id: '332132132122',
        name: 'Different room',
        number: '12',
    },
    {
        id: '3321321328',
        name: 'Another one',
        number: '4',
    },
    {
        id: '3321321322',
        name: 'Someones room',
        number: '8',
    },
    {
        id: '3321321326',
        name: 'Real actual room',
        number: '10',
    }
]

const Profile = () => {
    const [displayName, setDisplayName] = useState('John Doe');

    const handleChangeDisplayName = () => {
```

```
        // Implement logic to apply changes to display name
        // For example, you can make an API call to update the display name on the server
        // Once the changes are applied, you can show a success message or perform any other necessary actions
        console.log('New display name:', displayName);
    };

    return (
        <View style={styles.container}>
            <View style={styles.userDataContainer}>
                <View style={styles.userInfoContainer}>
                    <Image
                        source={{ uri: 'https://upload.wikimedia.org/wikipedia/commons/7/7c/Profile_avatar_placeholder_large.png'
}}
                        style={styles.avatar}
                    />
                    <View style={styles.userData}>
                        <Text style={styles.label}>Email:</Text>
                        <Text style={styles.value}>john.doe@example.com</Text>

                        <Text style={styles.label}>Name:</Text>
                        <Text style={styles.value}>{displayName}</Text>

                        <Text style={styles.label}>Rooms:</Text>
                        <Text style={styles.value}>5</Text>
                    </View>
                </View>
                <Text style={styles.label}>Change display name</Text>
                <TextInput
                    style={styles.input}
                    value={displayName}
                    onChangeText={setDisplayName}
                />
                <TouchableOpacity style={styles.button} onPress={handleChangeDisplayName}>
                    <Text style={styles.buttonText}>Apply Changes</Text>
                </TouchableOpacity>
            </View>
            <View style={styles.innerContainer}>
                <Text style={[styles.label, { fontSize: 20, marginBottom: 15 } ]}>Manage your rooms</Text>
                <RoomList data={AVAILABLE_ROOMS} adminRoomList={true} />
            </View>
        </View>
    );
};

const styles = StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: '#FFFFFF',
    },
    userDataContainer: {
        padding: 20,
        shadowColor: '#000',
        shadowOffset: {
            width: 0,
            height: 2,
        },
        shadowOpacity: 0.25,
        shadowRadius: 3.84,
        elevation: 5,
    },
    innerContainer: {
        flex: 1, alignItems: 'center', justifyContent: 'center',
        shadowColor: '#000',
        shadowOffset: {
          width: 0,
          height: 2,
        },
        shadowOpacity: 0.25,
        shadowRadius: 3.84,
        elevation: 3,
    },
    userInfoContainer: {
```

```
        flexDirection: 'row',
        justifyContent: 'space-between',
        alignItems: 'center',
        marginBottom: 20,
    },
    avatar: {
        width: 100,
        height: 100,
        borderRadius: 50,
        marginRight: 20,
    },
    userData: {
        paddingLeft: 50,
        flex: 1,
    },
    label: {
        fontSize: 16,
        fontWeight: 'bold',
        marginBottom: 5,
    },
    value: {
        fontSize: 16,
        marginBottom: 15,
        fontWeight: '500',
    },
    input: {
        fontWeight: '500',
        fontSize: 16,
        marginBottom: 15,
        paddingVertical: 10,
        paddingHorizontal: 15,
        borderWidth: 1,
        borderColor: '#CCCCCC',
        borderRadius: 5,
        shadowColor: '#000',
        shadowOffset: {
            width: 0,
            height: 2,
        },
        shadowOpacity: 0.25,
        shadowRadius: 3.84,
        elevation: 5,
        backgroundColor: '#FFFFFF',
    },
    button: {
        backgroundColor: '#FF8C00',
        borderRadius: 5,
        paddingVertical: 10,
        alignItems: 'center',
        shadowColor: '#000',
        shadowOffset: {
            width: 0,
            height: 2,
        },
        shadowOpacity: 0.25,
        shadowRadius: 3.84,
        elevation: 5,
    },
    buttonText: {
        color: '#FFFFFF',
        fontSize: 16,
        fontWeight: 'bold',
    },
});

export default Profile;
```

RoomManagement.js

```
import { StyleSheet, View, Text, TouchableOpacity } from 'react-native'
```

```
import { UserList } from '../UserList';
import Video from 'react-native-video';
import { useState } from 'react';
import VideoBox from '../Video';
import MultimediaModal from '../Modals/MultiMediaPicker';

const placeholderData = [
    { id: 1, email: 'john.doe@example.com', name: 'John Doe' },
    { id: 2, email: 'jane.smith@example.com', name: 'Jane Smith' },
    { id: 3, email: 'alexander.brown@example.com', name: 'Alexander Brown' },
    { id: 4, email: 'emma.johnson@example.com', name: 'Emma Johnson' },
    { id: 5, email: 'michael.wilson@example.com', name: 'Michael Wilson' },
    { id: 6, email: 'olivia.thompson@example.com', name: 'Olivia Thompson' },
    { id: 7, email: 'william.davis@example.com', name: 'William Davis' },
    { id: 8, email: 'sophia.miller@example.com', name: 'Sophia Miller' },
    { id: 9, email: 'jackson.anderson@example.com', name: 'Jackson Anderson' },
    { id: 10, email: 'ava.white@example.com', name: 'Ava White' }
];


function RoomManagemenScreen() {
    const [videoURL, setVideoURL] = useState(null)
    const [modalVisible, setModalVisible] = useState(false)

    const handleCloseModal = (type, url) => {
        if(type === 'video') {
            setVideoURL(url)
        }
        setModalVisible(false)
    }
    return (
        <>
        <View style={{ flex: 1, alignItems: 'center', marginTop: 20, justifyContent: 'center' }}>
            {videoURL ? <Video
                source={{ uri: 'https://www.learningcontainer.com/wp-content/uploads/2020/05/sample-mp4-file.mp4?_=1' }}
                style={{ width: '100%', height: 300 }}
                controls={true}
                ref={(ref) => {
                    this.player = ref
                }} /> : <VideoBox onPress={()=>setModalVisible(true)}/>}
            <View style={styles.userHeader}>
                <Text style={styles.label}>Manage users</Text>
                <TouchableOpacity style={styles.button}>
                    <Text style={styles.buttonText}>Invite +</Text>
                </TouchableOpacity>
            </View>
            <UserList data={placeholderData} />
        </View>
        <MultimediaModal visible={modalVisible} onClose={handleCloseModal}/>
        </>
    );
}

const styles = StyleSheet.create({
    label: {
        fontSize: 20,
        fontWeight: 'bold',
        marginBottom: 5,
    },
    userHeader: {
        marginTop: 40,
        width: '100%',
        paddingHorizontal: 20,
        paddingVertical: 20,
        flexDirection: 'row',
        justifyContent: 'space-between'
    },
    button: {
        backgroundColor: '#26bfab',
        borderRadius: 5,
        paddingVertical: 5,
        paddingHorizontal: 10,
```

```
    },
    buttonText: {
        color: '#FFFFFF',
        fontSize: 20,
        fontWeight: 'bold',
    },
})

export default RoomManagemenScreen
```

UserList.js

```
import { FlatList, View, Text, StyleSheet, TouchableOpacity, Dimensions } from "react-native";
import MaterialCommunityIcons from 'react-native-vector-icons/MaterialCommunityIcons';

const renderItem = ({ item }) => (
    <View style={styles.row}>
        <View>
            <Text style={styles.name}>{item.email}</Text>
            <Text style={styles.name}>{item.name}</Text>
        </View>
        <View style={{ flexDirection: 'row', alignItems: 'center' }}>
            <TouchableOpacity style={[styles.button]}>
                <MaterialCommunityIcons name="pencil" color='white' size={21} />
            </TouchableOpacity>
            <TouchableOpacity style={[styles.button, { backgroundColor: 'red', marginLeft: 7 }]}>
                <MaterialCommunityIcons name="delete" color='white' size={21} />
            </TouchableOpacity>
        </View>
    </View>
);

export const UserList = ({ data }) => {
    return (<FlatList
        style={{ flex: 1 }}
        data={data}
        renderItem={renderItem}
        keyExtractor={(item) => item.id}
    />)
}


const windowWidth = Dimensions.get('window').width;
const rowWidth = windowWidth * 0.95;

const styles = StyleSheet.create({
    row: {
        width: rowWidth,
        flexDirection: 'row',
        justifyContent: 'space-between',
        alignItems: 'center',
        padding: 14,
        backgroundColor: '#FFFFFF',
        borderRadius: 8,
        shadowColor: '#000000',
        shadowOpacity: 0.2,
        shadowOffset: {
            width: 0,
            height: 2,
        },
        shadowRadius: 4,
        elevation: 3,
        marginBottom: 10,
    },
    name: {
        fontSize: 15,
        fontWeight: 'bold',
        textAlign: 'left',
    },
```

```
    button: {
        backgroundColor: '#FF8C00',
        borderRadius: 5,
        paddingVertical: 7,
        paddingHorizontal: 7,
    },
    buttonText: {
        color: '#FFFFFF',
        fontSize: 15,
        fontWeight: 'bold',
    },
});
```

VideoBox.js

```
import React from 'react';
import { View, Text, StyleSheet, TouchableOpacity } from 'react-native';
import MaterialCommunityIcons from 'react-native-vector-icons/MaterialCommunityIcons';

const VideoBox = ({ onPress }) => {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Add a media to play in the room</Text>
      <TouchableOpacity onPress={onPress} style={styles.box}>
        <MaterialCommunityIcons name="plus" color={"#FF8C00"} size={82} />
      </TouchableOpacity>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5F5F5',
  },
  box: {
    width: 230,
    height: 230,
    borderRadius: 10,
    borderColor: '#FF8C00',
    borderWidth: 4,
    justifyContent: 'center',
    alignItems: 'center',
    shadowColor: '#000',
    shadowOffset: {
      width: 0,
      height: 2,
    },
    shadowOpacity: 0.25,
    shadowRadius: 3.84,
    elevation: 5,
  },
  text: {
    marginBottom: 10,
    fontSize: 18,
    fontWeight: 'bold',
    color: '#333333',
  },
});

export default VideoBox;
```

App.jsx

```
import { NavigationContainer } from '@react-navigation/native';
```

```
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import React from 'react';
import { BottomTabs } from './Components/Navigation/BottomTabs';
import HomeScreen from './Components/Screens/Home';

const Stack = createNativeStackNavigator()

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen options={{ headerTitle: '' }}  name={"BottomTabs"} component={BottomTabs} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}


export default App;
/**
 * Sample React Native App
 * https://github.com/facebook/react-native
 *
 * @format
 */

import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import React from 'react';
import { BottomTabs } from './Components/Navigation/BottomTabs';
import HomeScreen from './Components/Screens/Home';

const Stack = createNativeStackNavigator()

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen options={{ headerTitle: '' }}  name={"BottomTabs"} component={BottomTabs} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}


export default App;
```

Package.json

```
{
  "name": "Mediator",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "android": "react-native run-android",
    "ios": "react-native run-ios",
    "lint": "eslint .",
    "start": "react-native start",
    "test": "jest"
  },
  "dependencies": {
    "@react-navigation/bottom-tabs": "^6.5.7",
    "@react-navigation/material-bottom-tabs": "^6.2.15",
    "@react-navigation/native": "^6.1.6",
    "@react-navigation/native-stack": "^6.9.12",
    "react": "18.2.0",
    "react-native": "0.71.8",
    "react-native-paper": "^5.8.0",
    "react-native-safe-area-context": "^4.5.2",
    "react-native-screens": "^3.20.0",
```

```
    "react-native-vector-icons": "^9.2.0",
    "react-native-video": "^5.2.1"
  },
  "devDependencies": {
    "@babel/core": "^7.20.0",
    "@babel/preset-env": "^7.20.0",
    "@babel/runtime": "^7.20.0",
    "@react-native-community/eslint-config": "^3.2.0",
    "@tsconfig/react-native": "^2.0.2",
    "@types/jest": "^29.2.1",
    "@types/react": "^18.0.24",
    "@types/react-test-renderer": "^18.0.0",
    "babel-jest": "^29.2.1",
    "eslint": "^8.19.0",
    "jest": "^29.2.1",
    "metro-react-native-babel-preset": "0.73.9",
    "prettier": "^2.4.1",
    "react-test-renderer": "18.2.0",
    "typescript": "4.8.4"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

MongoClient.js

```
const { MongoClient } = require('mongodb')


class MongoDBClient {
  constructor () {
    this.client = new MongoClient(process.env.MONGO_URI)
  }

  async init () {
    if (this.db) {
      console.log('Mongo connection already open')
      return
    }
    await this.client.connect()
    console.log('Opened mongo connection')

    // README: Create a unique index on the token key in the pushTokens collection
    this.db = this.client.db()
  }

  async close () {
    await this.client.close()
    this.db = undefined
    logMongo('Closed mongo connection')
  }

  getDb () {
    if (!this.db) {
      throw new Error('Mongo client is not connected')
    }
    return this.db
  }
}

module.exports = new MongoDBClient()
```

ErrorMiddleware.js

```
const ErrorResponse = require('../utils/errorResponse');
```

```
const errorHandler = (err, req, res, next) => {
  let error = { ...err };

  error.message = err.message;
  console.log(err);


  if (err.name === 'CastError') {
    const message = Resource not found;
    error = new ErrorResponse(message, 404);
  }

  if (err.code === 11000) {
    const message = 'Duplicate field value entered';
    error = new ErrorResponse(message, 400);
  }

  if (err.name === 'ValidationError') {
    const message = Object.values(err.errors).map(val => val.message);
    error = new ErrorResponse(message, 400);
  }

  res.status(error.statusCode || 500).json({
    success: false,
    error: error.message || 'Server Error'
  });
};

module.exports = errorHandler;
```

Server.js

```
const express = require('express')
const dotenv = require('dotenv')

dotenv.config({ path: './.env' })

const errorHandler = require('./src/middleware/error')
const articleRouter = require('./src/routers/articles')
const MongoClient = require('./src/database/MongoClient')


MongoClient.init()
const app = express()

app.use(express.json())
app.use(errorHandler)

const PORT = process.env.PORT || 8000

app.use('/api/article', articleRouter)

const server = app.listen(
  PORT,
  console.log(
    Server running in ${process.env.NODE_ENV} mode on port ${PORT}
  )
);
```

UserController.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncWrap = require('express-async-wrap');
const { getArtcileRepository } = require('../repositories');

const { getuser, getusers, createuser, deleteuser, updateuser } = getArtcileRepository()

exports.getusers = asyncWrap(async (req, res, next) => {
    const users = await getusers(req.params.offset, req.params.limit);
```

```
    res.status(200).json({
        success: true,
        data: users
    });
});


exports.getuser = asyncWrap(async (req, res, next) => {
    const user = await getuser(req.params.id);
    if (!user) {
        return next(
            new ErrorResponse(`No user with the id of ${req.params.id}`),
            404
        );
    }

    res.status(200).json({
        success: true,
        data: user
    });
});


exports.adduser = asyncWrap(async (req, res, next) => {

    const user = await createuser(req.body);

    res.status(200).json({
        success: true,
        data: user
    });
});


exports.updateuser = asyncWrap(async (req, res, next) => {
    let user = await getuser(req.params.id);

    if (!user) {
        return next(
            new ErrorResponse(`No user with the id of ${req.params.id}`),
            404
        );
    }

    user = await updateuser(req.params.id, req.body);

    res.status(200).json({
        success: true,
        data: user
    });
});


exports.deleteuser = asyncWrap(async (req, res, next) => {
    const user = await getuser(req.params.id);

    if (!user) {
        return next(
            new ErrorResponse(`No user with the id of ${req.params.id}`),
            404
        );
    }

    await deleteuser(req.params.id)

    res.status(200).json({
        success: true,
        data: {}
    });
});
```

Package.json

```
{
  "name": "crud",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "start": "nodemon server.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "express-async-wrap": "^1.0.0",
    "knex": "^2.4.0",
    "mongodb": "^4.13.0",
    "mongoose": "^6.8.4",
    "nodemon": "^2.0.20",
    "pg": "^8.8.0"
  }
```

RoomController.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncWrap = require('express-async-wrap');
const { getArtcileRepository } = require('../repositories');

const { getroom, getrooms, createroom, deleteroom, updateroom } = getArtcileRepository()

exports.getrooms = asyncWrap(async (req, res, next) => {
    const rooms = await getrooms(req.params.offset, req.params.limit);

    res.status(200).json({
        success: true,
        data: rooms
    });
});


exports.getroom = asyncWrap(async (req, res, next) => {
    const room = await getroom(req.params.id);
    if (!room) {
        return next(
            new ErrorResponse(`No room with the id of ${req.params.id}`),
            404
        );
    }

    res.status(200).json({
        success: true,
        data: room
    });
});


exports.addroom = asyncWrap(async (req, res, next) => {

    const room = await createroom(req.body);

    res.status(200).json({
        success: true,
        data: room
    });
});
```

```
exports.updateroom = asyncWrap(async (req, res, next) => {
    let room = await getroom(req.params.id);

    if (!room) {
        return next(
            new ErrorResponse(`No room with the id of ${req.params.id}`),
            404
        );
    }

    room = await updateroom(req.params.id, req.body);

    res.status(200).json({
        success: true,
        data: room
    });
});


exports.deleteroom = asyncWrap(async (req, res, next) => {
    const room = await getroom(req.params.id);

    if (!room) {
        return next(
            new ErrorResponse(`No room with the id of ${req.params.id}`),
            404
        );
    }

    await deleteroom(req.params.id)

    res.status(200).json({
        success: true,
        data: {}
    });
});

exports.updateroom = asyncWrap(async (req, res, next) => {
    let room = await getroom(req.params.id);

    if (!room) {
        return next(
            new ErrorResponse(`No room with the id of ${req.params.id}`),
            404
        );
    }

    room = await updateroom(req.params.id, req.body);

    res.status(200).json({
        success: true,
        data: room
    });
});
```

MultimediaController.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncWrap = require('express-async-wrap');
const { getArtcileRepository } = require('../repositories');

const { getMultiMedia, getMultiMedias, createMultiMedia, deleteMultiMedia, updateMultiMedia } = getArtcileRepository()

exports.getMultiMedias = asyncWrap(async (req, res, next) => {
    const MultiMedias = await getMultiMedias(req.params.offset, req.params.limit);

    res.status(200).json({
        success: true,
        data: MultiMedias
    });
});
```

```
exports.getMultiMedia = asyncWrap(async (req, res, next) => {
    const MultiMedia = await getMultiMedia(req.params.id);
    if (!MultiMedia) {
        return next(
            new ErrorResponse(`No MultiMedia with the id of ${req.params.id}`),
            404
        );
    }

    res.status(200).json({
        success: true,
        data: MultiMedia
    });
});


exports.addMultiMedia = asyncWrap(async (req, res, next) => {

    const MultiMedia = await createMultiMedia(req.body);

    res.status(200).json({
        success: true,
        data: MultiMedia
    });
});


exports.updateMultiMedia = asyncWrap(async (req, res, next) => {
    let MultiMedia = await getMultiMedia(req.params.id);

    if (!MultiMedia) {
        return next(
            new ErrorResponse(`No MultiMedia with the id of ${req.params.id}`),
            404
        );
    }

    MultiMedia = await updateMultiMedia(req.params.id, req.body);

    res.status(200).json({
        success: true,
        data: MultiMedia
    });
});


exports.deleteMultiMedia = asyncWrap(async (req, res, next) => {
    const MultiMedia = await getMultiMedia(req.params.id);

    if (!MultiMedia) {
        return next(
            new ErrorResponse(`No MultiMedia with the id of ${req.params.id}`),
            404
        );
    }

    await deleteMultiMedia(req.params.id)

    res.status(200).json({
        success: true,
        data: {}
    });
});
```

UserRouter.js

```
const express = require('express');
const {
  getUsers,
  getUser,
  addUser,
  updateUser,
  deleteUser
} = require('../controllers/Users');


const router = express.Router({ mergeParams: true });


router
  .route('/')
  .get(getUsers)
  .post(addUser);

router
  .route('/:id')
  .get(getUser)
  .put(updateUser)
  .delete(deleteUser);

module.exports = router;
```

RoomRouter.js

```
const express = require('express');
const {
  getRooms,
  getRoom,
  addRoom,
  updateRoom,
  deleteRoom
} = require('../controllers/Rooms');


const router = express.Router({ mergeParams: true });


router
  .route('/')
  .get(getRooms)
  .post(addRoom);

router
  .route('/:id')
  .get(getRoom)
  .put(updateRoom)
  .delete(deleteRoom);

module.exports = router;
```

ДОДАТОК Г
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Кваліфікаційна робота

Факультет Програмування та комп'ютерних і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях

Керівник: канд. пед. наук, доцент_Онишко О. Г
Студент: Савич Н. В. ІПЗс-20-1

Мета та постановка задачі

Мета дипломного проекту: Реалізувати мобільний застосунок для синхронізації відтворення мультимедійних матеріалів на різних пристроях.

Задачі проектування:

1. Дослідити ринок та існуючі рішення у сфері синхронізації відтворення мультимедійних матеріалів на різних пристроях.
2. Розробити архітектуру та інтерфейс мобільного застосунку для синхронізації відтворення мультимедійних матеріалів.
3. Реалізувати функціональність для відтворення мультимедійних матеріалів на основному пристрої та синхронізації зі зазначеними пристроями.
4. Розробити механізм для керування відтворенням мультимедійних матеріалів на різних пристроях.
6. Забезпечити безпеку передачі даних між пристроями.
7. Провести тестування та валідацію розробленого мобільного застосунку.

Актуальність задачі

В сучасному світі відчувається необхідність для синхронізованого відтворення презентацій, аудіо, або відео файлів саме на мобільних платформах. Програмний продукт такого роду може бути застосовано на сам перед в сферах освіти та науки, для проведення дистанційних семінарів, або лекцій. Крім того подібна технологія може бути використана для дозвілля.

Існуючі програмні рішення

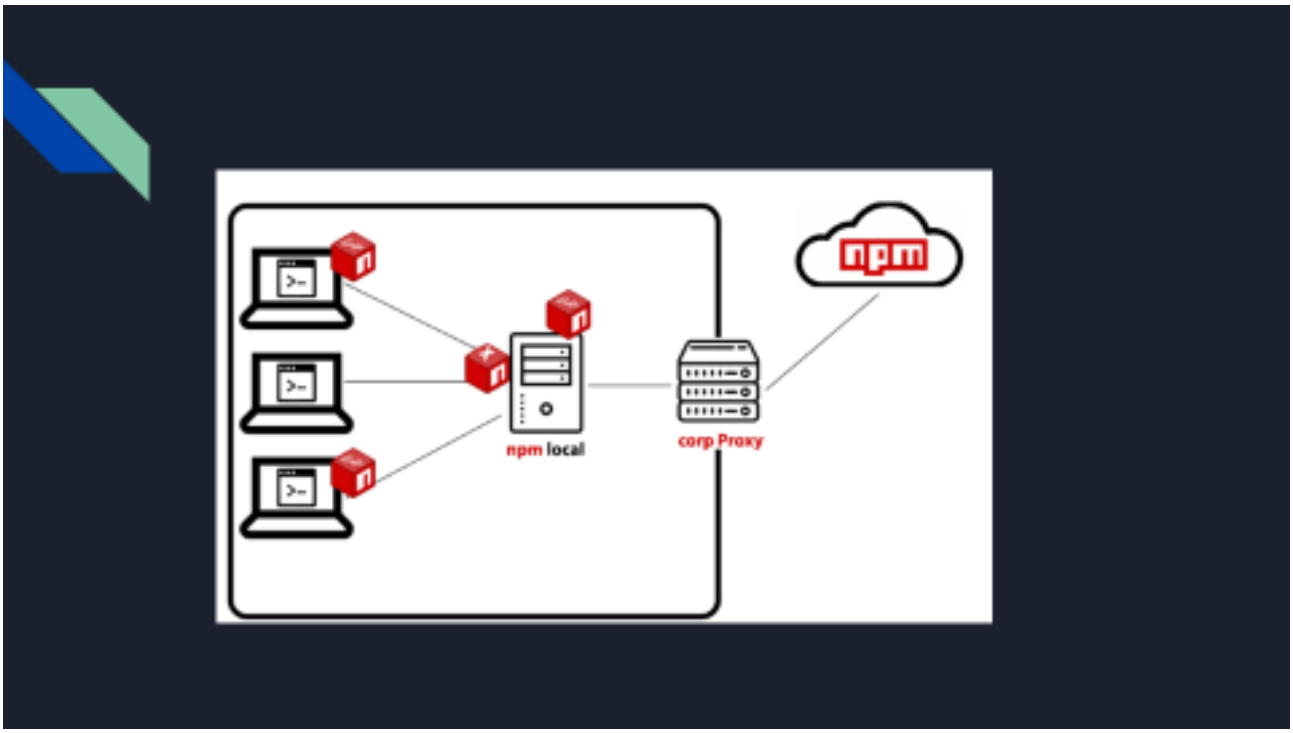
Watch2Gather – Онлайн сервіс синхронного перегляду медіа матеріалів. Дозволяє створити кімнату з запрошувальним посиланням і переглядати та послуховувати матеріали разом

Не дивлячись на те, що Watch2gether є онлайн свервісом, він дає доступ до широкого вибору матеріалів з різних платформ, таких як: YouTube, Vimeo, Dailymotion, SoundCloud.
Основним недіоліками є:
- Відсутність можливісті синхроного перегляду, або прослуховування матеріалів з не піддтримуємих платформ
- Відсутність можливості синхроного відтворення власних медіа матеріалів
- Не досконала мобільна версія сайту.

SyncPlay – застосунок синхронного перегляду медіа матеріалів. Дозволяє разом переглядати выдео і аудіоатеріали на платформах windows I MacOs

SyncPlay – представляє собою програмне забезпечення для платформ windows та MacOS. Не має можливості стрімінгу з популярних онлайн сервісів, але має можливість відтворення власних матеріалів. Основний недолік недоступність для мобільних платформ

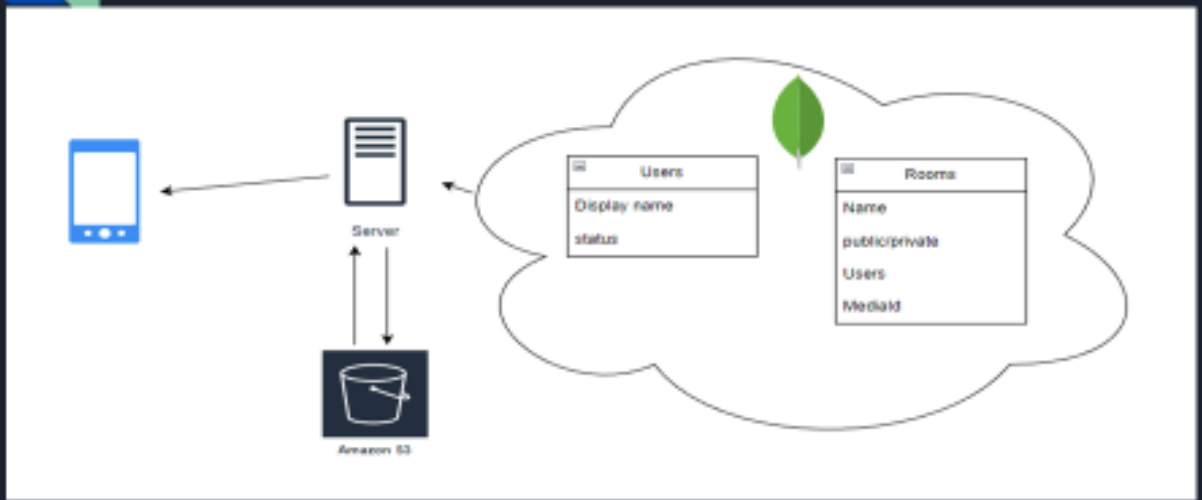
Технології



Проектування

Use Case Diagram

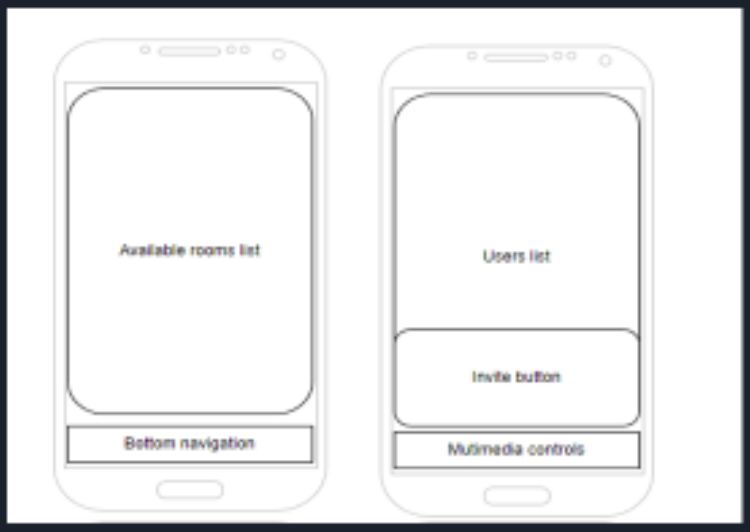
Структура даних



Розробка екранів додатку на основі React Navigation

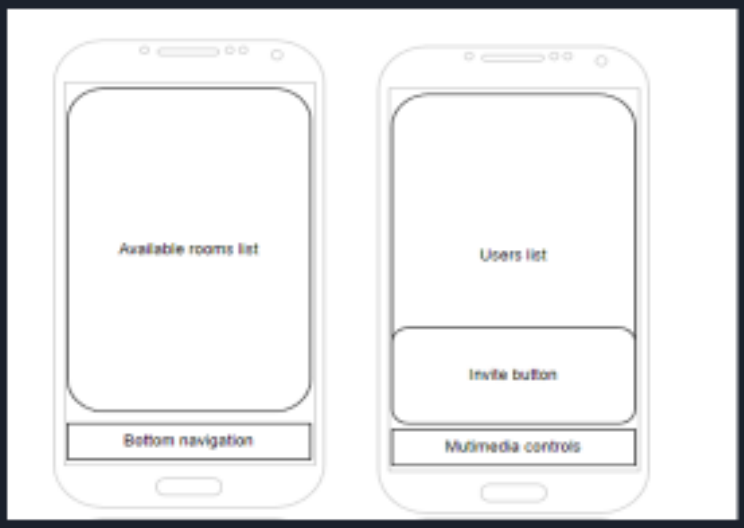
Головний екран

Екран кімнати адміна



Головний екран

Екран кімнати адміна



Екрани профілю та кімнати користувача

кімнати користувача в горизонтальній орієнтації

Ініціалізація проекту

Основні компонети

```
export default class VideoComponent extends React.Component {

  renderVideo () {
      return(
        <Video
          source={require('../assets/Plans_Playing_Close.mp4')}
          style={{ width: 800, height: 600 }}
          muted={true}
          repeat={true}
          resizeMode={"cover"}
          volume={1.0}
          rate={1.0}
          ignoreSilentSwitch={"obey"}

        />
      )
  }

  render () {
    return (
      <View>
        {this.renderVideo()}
      </View>
    )
  }
}

// Later on in your styles...
var styles = StyleSheet.create({
  backgroundVideo: {
    position: 'absolute',
    top: 0,
    left: 0,
    bottom: 0,
    right: 0,
  },
});
```

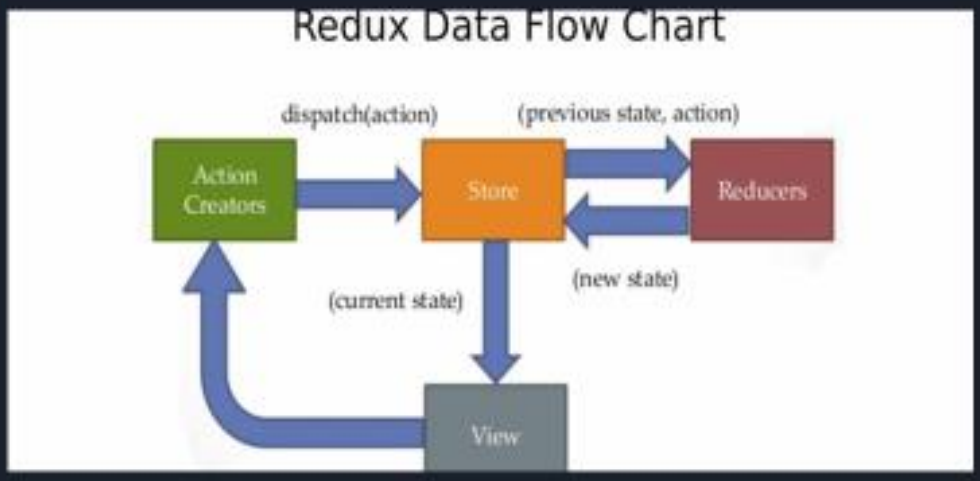
```
/**
 * Sample React Native App
 * https://github.com/facebook/react-native
 *
 * @format
 * @flow
 */

import React, {Component} from 'react';
import {Platform, StyleSheet, Text, View} from 'react-native';
import VideoComponent from './VideoComponent'

export default class App extends Component<Props> {
  render() {
    return (
        <View>
        <VideoComponent />
        </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  }
});
```

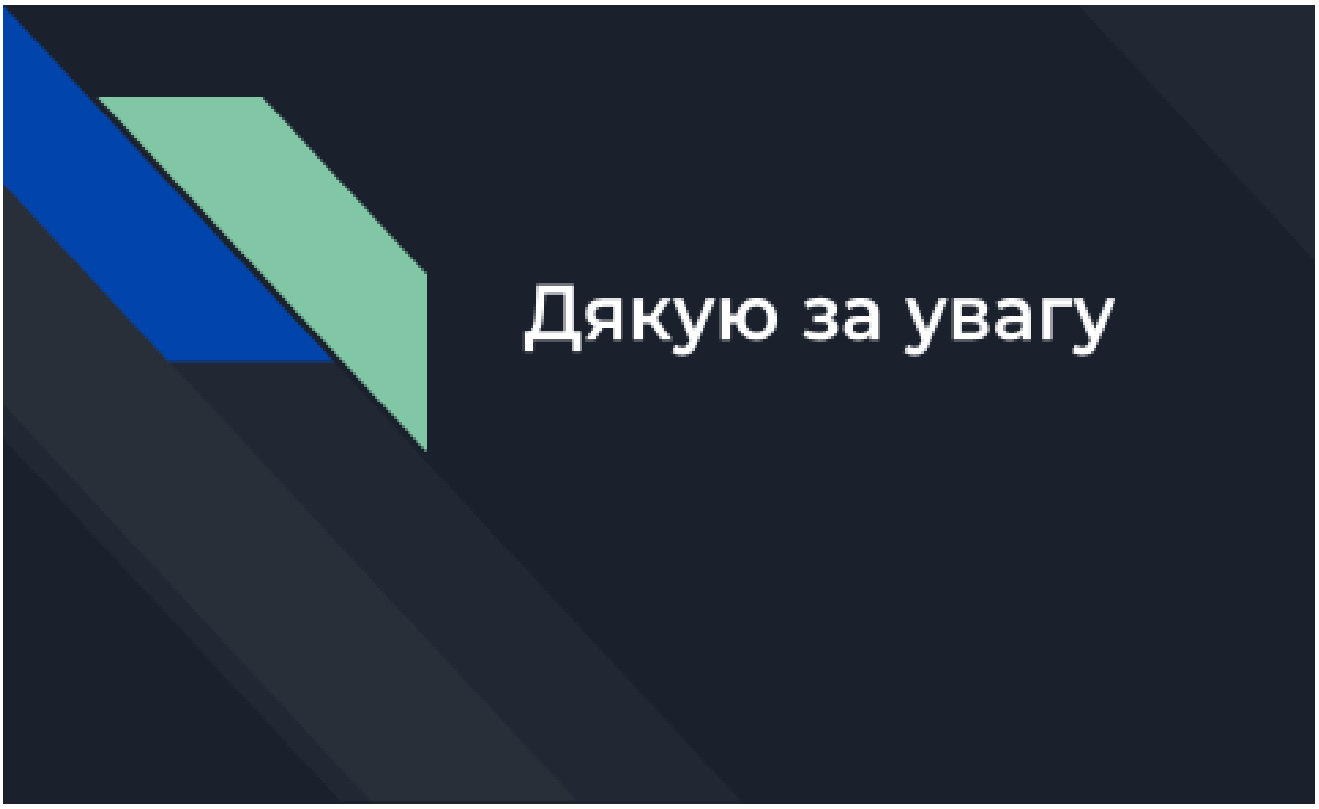
Контроль стану додатку за допомогою Redux



Тестування проекту за допомогою Detox

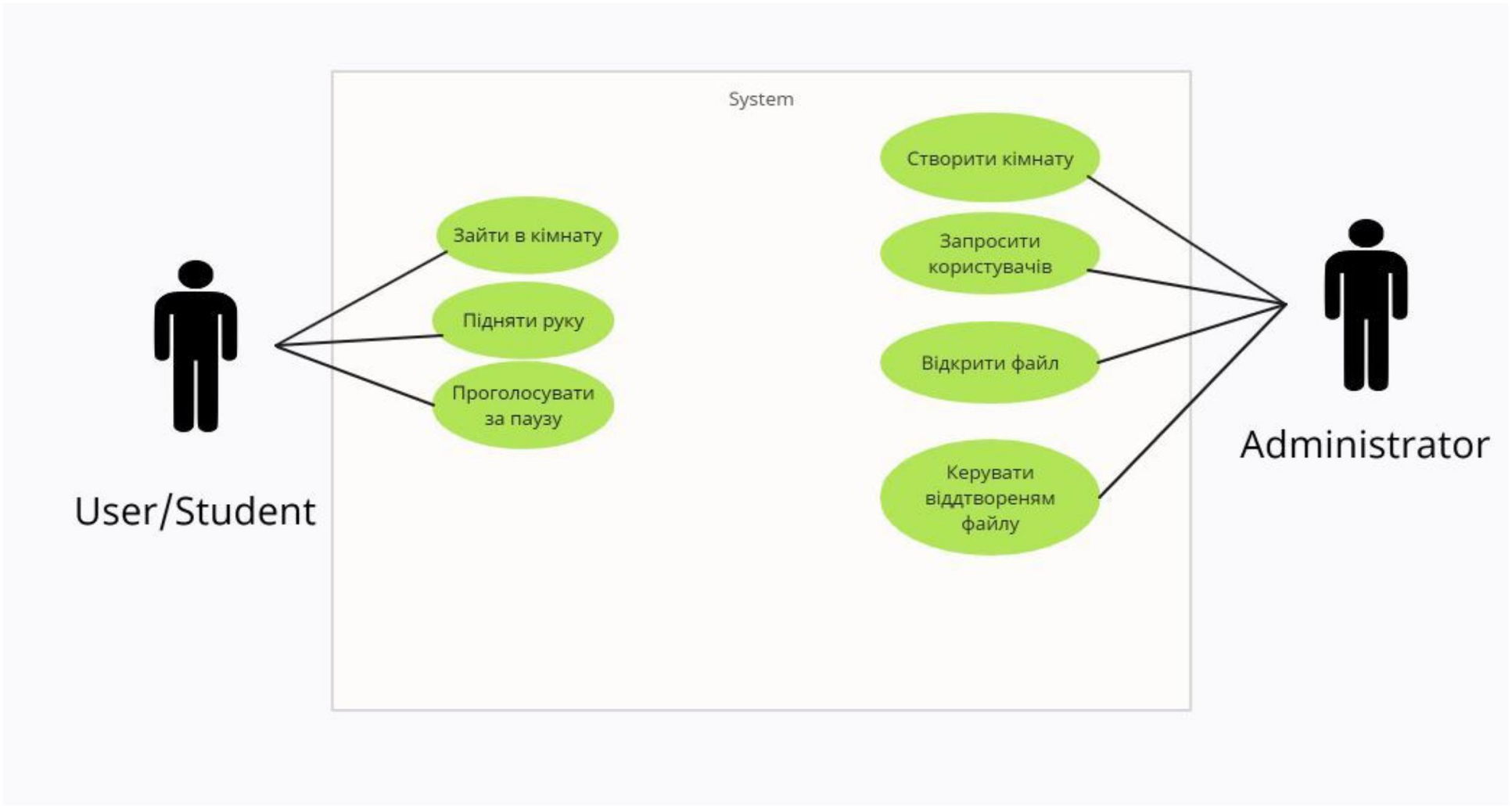
Висновки

Отже програмний продукт може задовольнити існуючі потреби на ринку мобільних додатків. Було використано провідні технології. Програмне забезпечення відповідає вимогам. Можливості покращення додатку полягають в додавані підтримки стрімінгових сервісів, відеохостингів та інших онлайн платформ, імплементації deep linking, для можливості створення кімнат з посиланням


Дякую за увагу

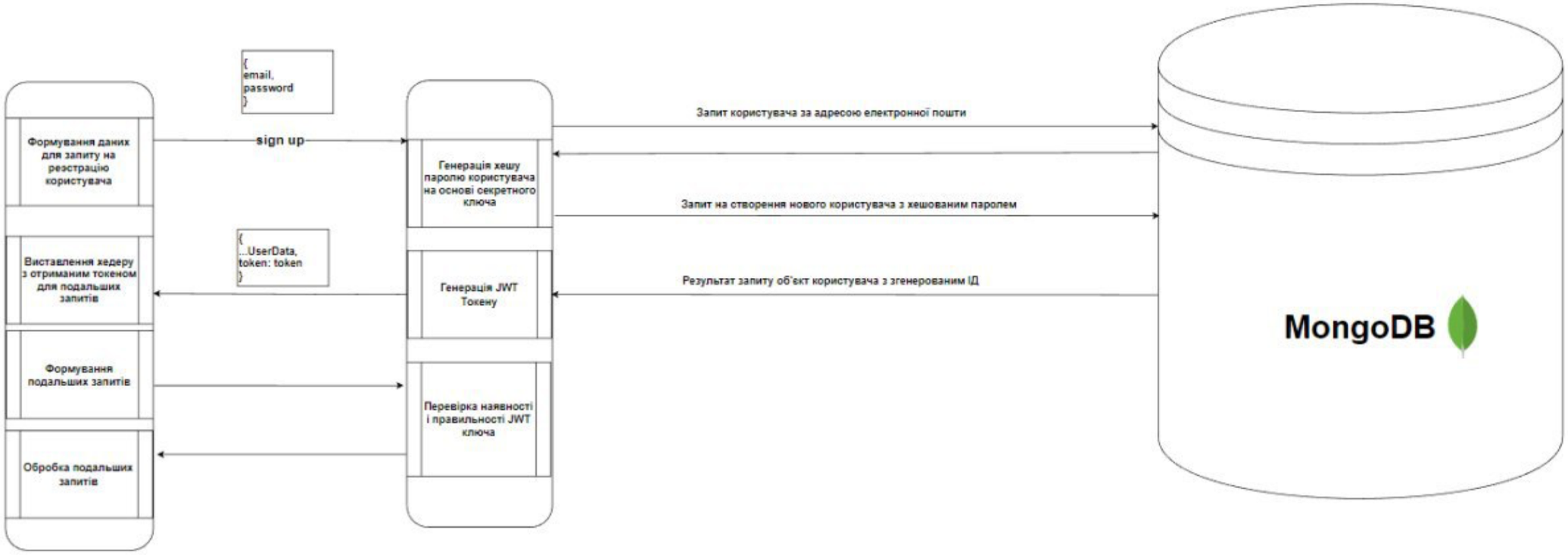
ГРАФІЧНА ЧАСТИНА

Рисунок 1 - Діаграма варіантів використання



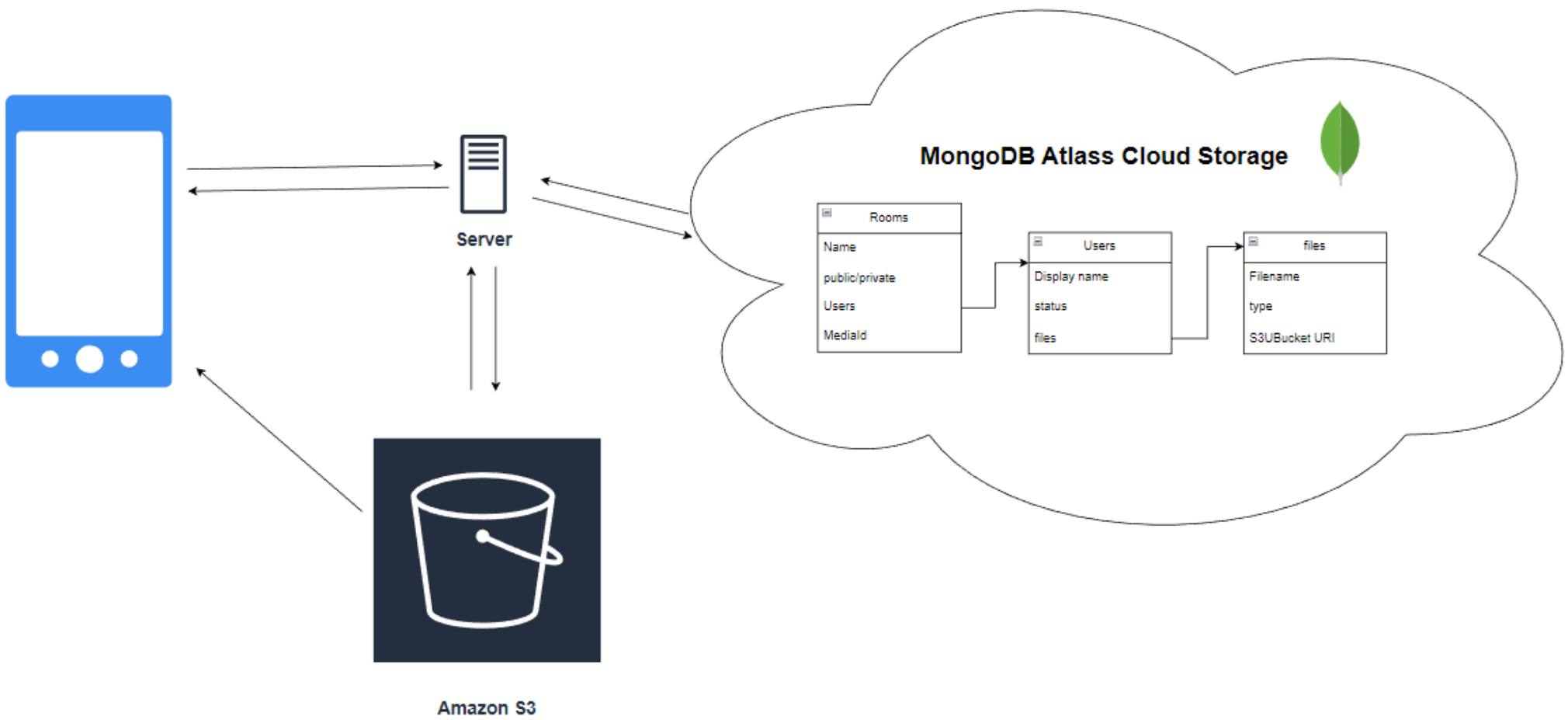
					КвРІП3.200124.01.07.Е8			
Зм.	Арк.	№докум.	Підпис	Дата	USE-CASE Діаграма застосунку для синхронізованого перегляду мультимедіа матеріалів	Літера	Маса	Масштаб
Розробив		Савич Н. В		22.05.23				
Керівник		Онишко О. Г.		22.05.23		Аркуш 1	Аркушів 1	
Консульт.								
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІП3с-20-1		
Зав.каф.		Бедратюк Л.П.		22.05.23				

Рисунок 2 - Схема авторизації через JWT токен



					КвРІП3.200124.01.07.Е8		
Зм.	Арк.	№докум.	Підпис	Дата	Схема авторизації через JWT токен застосунку для синхронізованого перегляду мультимедіа матеріалів	Літера	Маса / Масштаб
Розробив		Савич Н. В		22.05.23			
Керівник		Онишко О. Г.		22.05.23		Аркуш 1	Аркушів 1
Консульт.							
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІП3с-20-1	
Зав.каф.		Бедратюк Л.П.		22.05.23			

Рисунок 3 - Схема передачі мультмедійних файлів мережею



					КвРІП3.200124.01.07.Е8			
						Літера	Маса	Масштаб
Зм.	Арк.	№докум.	Підпис	Дата	Схема передачі мультимедійних файлів мережею застосунку			
Розробив		Савич Н. В		22.05.23		Аркуш 1	Аркушів 1	
Керівник		Онишко О. Г.		22.05.23				
Консульт.								
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІП3с-20-1		
Зав.каф.		Бедратюк Л.П.		22.05.23				

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-20-1
Сович Н В
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях.

(керівник роботи – Онищко Оксана Григорівна)
Прізвище, ім'я, по батькові

05.02.23
Дата

Підпис студента

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

здобувача вищої освіти
Савич Н. В.
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення 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 плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії 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 плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

30.05.2023
дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Совин Ніколо Вячеславович
Прізвище, ім'я, по батькові

студент III курсу факультету інформаційних
здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)/ науково-педагогічний працівник (назва кафедри)

технологій, кафедри інженерії програмного забезпечення
назва факультету

підтверджую, що ознайомився (- лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

«05» лютого 20 23 р.

Підпис

Fri Jun 02 12:31:51 EEST 2023, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 8.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114574 Назва: БКР Реалізація мобільного застосунку для синхронізації відтворення Мультимедійних матеріалів на різних пристроях Додано в БД: 2023-06-02 Автора: Савич Н. В. Керівники: Онишко. О. Г. к.п.н. доц. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	79708	654	8452 (11%)	82 (13%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

Дата перевірки:
02.06.2023 13:30:56 EEST

Дата звіту:
02.06.2023 13:33:25 EEST

ID перевірки:
1015392922

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005589

Назва документа: **Савич_ІПЗс20-1_КВР_Антиплагіат**

Кількість сторінок: **65** Кількість слів: **11463** Кількість символів: **92160** Розмір файлу: **3.47 MB** ID файлу: **1015057367**

6.39%
Схожість

Найбільша схожість: **3.05%** з джерелом з Бібліотеки (ID файлу: **1015017362**)

4.2% Джерела з Інтернету	436	Сторінка 67
4.67% Джерела з Бібліотеки	90	Сторінка 71

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Реалізація мобільного застосунку для синхронізації відтворення мультимедійних матеріалів на різних пристроях»
Автор: Савич Нікола Властимірович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник Онишко Оксана Григоріївна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності 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 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 та електронної версії роботи	
3	Виявлені запозичення не є 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 досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов`язкових словосполучень у стандартних бланках (титулка, бланк завдання, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 6,39% і адресується до 526 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Керівник _____ О. Г. Онишко

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Савич Нікола Властимірович

Тема Реалізація мобільного застосунку для синхронізації відтворення Мультимедійних матеріалів на різних пристроях

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 0; кількість сторінок записки 104

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі було досліджено і проаналізовано предметну область, визначено усі функціональні та нефункціональні вимоги. Був проведений аналіз існуючих застосунків на ринку, розглянуто їх переваги і недоліки, та виведено актуальність розробки нового програмного забезпечення. Розглянуто інструменти для реалізації спроектованих рішень, в результаті чого створено програмне забезпечення. Також було проведено тестування програми, за результатами якого було виявлено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. Крім того було виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу застосунку.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки на сьогодні на ринку немає достатньо зручних та функціональних застосунків аналогів. Також було застосовано новітні технології для побудови програмного продукту та актуальні архітектурні рішеня.

5. Негативні сторони роботи У роботі було використано достатньо складні в супроводжені технології. Безпеці передачі файлів медіа матеріалів, в межах системи, не було приділено належної уваги.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ професор кафедри комп'ютерної інженерії та інформаційних систем, д.т.н., професор Лисенко Сергій Миколаєвич

"2" червня 2023 р. _____
(підпис)