Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
«Хмарно-стрімінговий медіа плеєр для лекцій та навчальних матеріалів .Частина 2. Архітектура додатку та медіа плеєру»

Виконав: студент 2 курсу, групи 2КІ-20м
напряму підготовки (спеціальності)
123 — «Комп'ютерна інженерія»
_____Козубський В. В_____

Керівник: к.т.н., доц. каф ОТ

_____Ткаченко О.М._____

«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«____» _____ 2021 р.

Вінниця ВНТУ - 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Освітньо—кваліфікаційний рівень магістр
Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О.Д. Азаров

«___»_____2021 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Козубського Владислава Володимировича
(прізвище, ім'я, по батькові)

1 Тема роботи «Хмарно-стрімінговий медіа плеєр для лекцій та навчальних матеріалів. Частина 2. Архітектура додатку та медіа плеєру» керівник роботи Ткаченко Олександр Миколайович, кандидат технічних наук, доцент, доцент кафедри обчислювальної техніки затверджені наказом Вінницького національного технічного університету від 06.03.2021 року №_75.

2 Строк подання студентом роботи 05.12.2021.

3 Вихідні дані до роботи: проаналізувати методи і засоби розробки додатків на основі операційної системи Android, розробити способи побудови сучасного користувацького макету .

4 Зміст розрахунково—пояснювальної записки (перелік питань, які потрібно розробити) _розглянути та проаналізувати методи та засоби розробки додатків на основі операційної системи Android; розробити макет користувацького інтерфейсу ; реалізувати програмно всі складові.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) — лістинги скетчів та фото розроблювальних пристроїв .

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3	Ткаченко О.М к.т.н., доцент каф. ОТ		
4	Кавецький В.В к.е.н.,доцент каф ЕПВМ		

7. Дата видачі завдання 06.09.2021 р.

Календарний план наведено в таблиці 2.

Таблиця 2 –– Календарний план

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Огляд і аналіз джерел інформації	6.09.2021	
	Розробка технічного завдання	29.09.2021	
	Огляд та аналіз методів та засобів побудови сучасних макетів у системах Android	21.10.2021	
	Розробка кресленого та програмного графу макету	27.10.2021	
	Реалізація програмного додатку системи Android	24.11.2021	
	Підготовка до презентації і захист роботи	25.11.2021	

Студент _____ Козубський В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ткаченко О.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Козубський В.В. Хмарно-стрімінговий медіа плеєр для лекцій та навчальних матеріалів. Частина 2. Архітектура додатку та медіа плеєру. Магістерська кваліфікаційна робота зі спеціальності 123 – комп'ютерна інженерія, освітня програма – комп'ютерна інженерія. Вінниця: ВНТУ, 2021. 130с.

На укр.мові. Бібліогр.: 26 назв; рис.: 20 ; табл. 11.

Магістерська робота присв'ячена системі на основі Modern Compose. Підбирається оптимальна система і програмне забезпечення.

У ході роботи було проаналізовано існуючі програми для демонстрації навчальних матеріалів на ОС Android, різноманітні варіації меда плеєрів,розібрано взаємодію HTTP протоколів та відтворення медіа файлів в потоковому форматі ,системи побудови сучасного та інтуїтивно зрозумілого інтерфейсу користувача та зручного медіа плеєру.

В магістерській роботі було виконано розширеня можливостей медалплеєр в звязці з графічним UI та UX додатку, з тестуванням на AVD і APD , наведено переваги та недоліки альтернативних додатків

Ключові слова: Медіа плеєр , стрімінг , UI та UX, Compose, android ОС

ABSTRACT

Kozubsky VV Cloud streaming media player for lectures and study materials. Part 2. Application and media player architecture. Master's thesis in specialty 123 - computer engineering, educational program - computer engineering. Vinnytsia: VNTU, 2021. 130p.

In Ukrainian. Bibliogr .: 26 titles; fig .: 20; table 11.

The master's thesis is devoted to the system based on Modern Compose. The optimal system and software are selected.

During the work we analyzed the existing applications for Android, various variations of honey players, the interaction of HTTP protocols and playback of media files in streaming format, a system for building a modern and intuitive user interface and user-friendly media player.

In the master's thesis, the capabilities of the medal player were performed in connection with the graphical UI and UX applications, with testing for AVD and APD, the advantages and disadvantages of alternative applications

Keywords: Media player, streaming, UI and UX, Compose, android OC

ЗМІСТ

					08—23.КМКР.028.00.000 ПЗ		
Змн.	*Лист*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Козубський В.В.*			*Хмарно-стрімінговий медіа плеєр для лекцій та навчальних матеріалів.Частина 2.Архітектура додатку та медіа плеєру*	*Літ.*	*Арк.* 6 / *Аркушів* 130
Керівник		*Ткаченко О.М.*					
Рецензент		*КондратенкоН.Р*				*ВНТУ, гр. 1КІ-20м*	
Н. Контроль		*Швець С.І.*					
Затверджую		*Азаров О.Д.*			*Пояснювальна записка*		

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

OS (ОС) — операційна система

ПЗ — програмне забезпечення

API — Application Programming Interfaces

APK — Application Package Kit

GUI — Graphical User Interface

JSON — JavaScript Object Notation

SDK — Software Development Kit

XML — Extensible Markup Language — розширювана мова розмітки

AVD — Android Virtual Device

APD — Android Physical Device

ВСТУП

Актуальність теми. З появою сучасних операційних систем, таких як Android та iOS, актуальними стали завдання, які раніше вирішувалися лише для настільних операційних систем. Вибір правильного тестового макету має вирішальне значення для передбачуваного користувача та інтерфейсу Android.

Він дозволяє розміщувати і розміщувати компоненти. Хороше уявлення про те, що використовується в кожному з доступних шаблонів, може зробити різницю між простим і складним UI та UX як для розробника, так і для користувача. Це особливо важливо, якщо потрібно керувати кількома пристроями.

Для кожної програми потрібно кілька макетів, що допомагає створити простий у використанні, сучасний і удобний інтерфейс. Тому важливо дослідити існуючі технології для створення інтерфейсу та вибрати найбільш оптимальне рішення. Операційна система Android підтримує такі мови програмування: Kotlin і Java. Для дослідження буде використана мова програмування Kotlin.

Об’єктом дослідження є процес створення макету інтерфейсу користувача та відповідного програмного забезпечення.

Предмет дослідження — методи та інструменти для створення гнучких макетів інтерфейсу користувача.

Мета роботи є розширення функціональних можливостей мобільних пристроїв шляхом створення мобільного додатка для Android, що забезпечує сучасне розташування інтерфейсу користувача. Для досягнення вищевказаних цілей виконайте такі завдання:

— проаналізувати програми для мобільних пристроїв;

— вмконати класифікацію існуючих систем для розробки;

— розглянути існуючі інструменти для створення власних макетів;

— розробити мобільний додаток для Android;

— розробка UI та UX для мобільного додатка;

— діагностувати роботу з розробленим додатком;

— розробити рекомендації та пропозиції щодо використання програмина практиці.

Наукова новизна роботи полягає у тому, що в ній знайшла подальшого розвитку Compose технологія відтворення мультимедіа в ОС Андроїд, в якій, на відміну від існуючих, було введено підтримку протоколів HTTP та передачу файлі по мережі в потоковому форматі, що дозволило зменщити часові затримки під час виведення меда контенту та вдосконалення моделі Compose та медіаплеєру для написання додатків, в якій дані, на відміну від існуючих, зберігаються в хмарній базі даних Firebase та в локальній базі даних Room, що збільшило швидкість відображення даних за рахунок відсутність затримок у передачі даних.

Практична цінність роботи полягає в тому, що на основі теоретичних досліджень створено мобільний додаток, що дозволяє отримати доступ до дидактичних матеріалів на Android.

Апробація Основний результат робота здійненай на Всеукраїнській науково-практичній онлайн-конференції «Молодь у науці: дослідження, проблеми, перспективи» (МН—2021) (Вінниця, 05.01.2021 — 14.05.2021)

Публікації. За результатами дослідження опубліковано 1 дисертацію: ДОСЛІДЖЕННЯ ЛОКАЛЬНИХ БАЗ ДАНИХ ДЛЯ ANDROID ПРИСТРОЇВ. РОЗРОБКА МЕТОДУ ОБРОБКИ ДАНИХ.. [Текст]В.В.Козубський // Молодь у науці: дослідження, проблеми, перспективи (МН—2021) Тез. доп.—Виноградник, 2021.—1. Режим https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/13157/11054

1 ОГЛЯД ТЕХНОЛОГІЙ, МЕТОДІВ ТА ПРОГРАМНІ СИСТЕМИ РОЗРОБКИ ANDROID

1.1 Операційна система Android

Сьогодні з Android OS стикається практично кожен, передаючи за проїзд у трамваях, дивлячись фільми на смарт тв, слідкуючи за сердечним ритмом за допомогою годинників.

Але спочатку операційна система Android була розроблена під пристрої із сенсорним екраном: планшети та смартфони. Вона створена на основі Linux kernel в Google і є однією з найпопулярніших мобільних операційних систем на сьогоднішній день. Хоча зараз Android — це не тільки операційна система для смартфонів і планшетів. Існують також:

— Android Wear для пристроїв, що носяться (переважно розумного годинника);

— Android Auto для автомобілів;

— Android TV для розумних телевізорів[1].

На базі цієї операційної системи працює величезна кількість девайсів, що перевищує кількість пристроїв на основних конкуруючих — iOS та Windows Mobile.

Android дозволяє запускати Java — програми, що керують пристроєм через розроблені Google бібліотеки. Під систему розробляються програми, які можна завантажити через Google Play або сторонні ресурси. На її основі пишуться інші операційні системи під кастомні та краудфандингові девайси завдяки наявності відкритого коду для розробників. І перекладено її вже більш ніж на 100 світових мов.

1.2 Огляд можливих архітектур додатку

Вибір між різними патернами розробки завжди супроводжується рядом суперечок і дискусій, а різні погляди розробників на це ще більше ускладнюють завдання. Існує досить багато MV*—патернів: Model-View-

Controller, Model-View-Presenter, Presentation Model, Passive View, Supervising Controller, Model-View-ViewModel та багато інших.

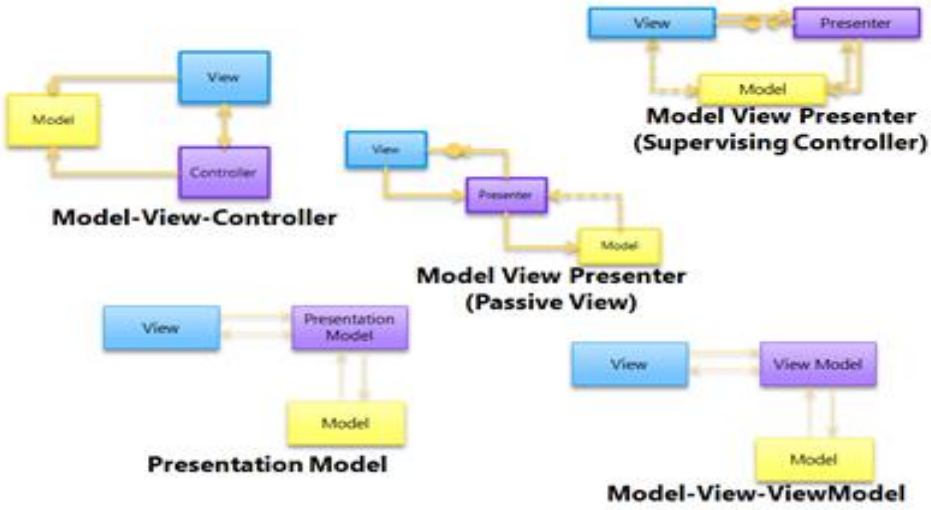

Рисунок 1.1 — Схема взаємодії MV* патернів

В android використовуються : MVC, MVP, MVVM і MVI. Дивлячись на схеми ,на рисунку 1.1. стрілками показано відносини між компонентами патернів. Далі буде розглянуто яка між ними різниця чи є Controller тим же, що і Presenter або PresentationModel, порівняємо між собою Model-View-Presenter та Model-View-ViewModel щоб описати подібності та відмінності між найбільш поширеними MV*—паттернами.

1.2.1 Побудова UI без використання MV*—патернів

Було розглянуто питання «Як би можна побудувати інтерфейс (UI), не використовуючи вищезгадані патерни?» Взяли б форму, додали на неї віджети, а логіку написали в коді. Такий код, що описує логіку View, жорстко пов'язаний з інтерфейсом користувача, так як він безпосередньо взаємодіє з елементами на екрані. Це добрий, але прямолінійний підхід. Він застосовується лише для дуже простих інтерфейсів. Коли логіка стає складнішою, підтримка такого UI може перетворитися на кошмар!

Корінь проблеми полягає в тому, що побудова UI у такий спосіб порушує принцип єдиної відповідальності (single responsibility principle), який

свідчить: «У класу має бути лише одна причина зміни». Якщо UI—компонент містить код для відображення, логіки та даних, він має кілька причин для зміни. Наприклад, якщо ви хочете змінити тип користувача елемента, який використовується для відображення даних, то зміни не повинні вплинути на логіку. Однак оскільки логіка так тісно пов'язана з елементами управління, її також доведеться міняти. Це так званий "код із душком" (code smell), який сигналізує, що принцип єдиної відповідальності порушено[1].

Таким чином, якщо форма містить код для відображення елементів керування, логіку інтерфейсу (що відбувається при натисканні кнопки) та дані для відображення на екрані, ви зіткнетеся з такими проблемами:

Зміни в UI, логіці або даних, швидше за все, спричинять зміни в інших частинах. Тому вносити виправлення набагато складніше, що ускладнює підтримку.

Логіка та дані програми можуть бути написані таким чином, щоб кожен компонент міг бути протестований окремо. Однак код, пов'язаний з інтерфейсом користувача, погано піддається модульному тестуванню, тому що для цього часто потрібна участь користувача для запуску логіки в UI. Крім того, будь—яка візуалізація часто потребує оцінки з боку людини, що все «виглядає правильно». Зазначимо, що існують рішення для автоматизації тестування інтерфейсу користувача. Однак вони лише імітують взаємодію з користувачем. Як правило, вони складніші в налаштуванні та обслуговуванні, ніж unit—тести, і найчастіше використовуються при інтеграційному тестуванні, тому що для цього потрібний запуск усієї програми.

Якщо UI—код змішаний з кодом логіки та даних, його стає набагато складніше перевикористовувати[2].

Хоча кожен із патернів має досить багато відмінностей, їх цілі схожі: відокремити UI—код (View) від коду логіки (Presenter, Controller, ViewModel тощо) та коду обробки даних (Model). Це дозволяє кожному їх розвиватися самостійно. Наприклад, ви зможете змінити зовнішній вигляд і стиль програми, не торкаючись логіки та даних.

Крім того, оскільки логіка та дані відокремлені від відображення, то вони можуть бути протестовані окремо. Для простих програм це може бути не так важливо. Наприклад, якщо ваша програма є простим редактором даних. Однак, якщо у вас складніша логіка інтерфейсу, то можливість автоматично перевірити, що вона працює правильно, буде дуже цінною.

1.2.2 Опис MV* патернів

Одним із найперших патернів для відокремлення уявлення від логіки та моделі став Model-View-Controller (MVC). Ця концепція була описана Трюгве Реєнскауг.

1.2.2.1 Model-View-Controller

Цей патерн був розроблений в далекому 1979 році для написання додатків на Smalltalk. Але в ті дні програмування було не таким, як сьогодні. Не було Windows. Немає графічного інтерфейсу користувача. Не було бібліотек віджетів. Якщо ви хочете інтерфейс користувача, його потрібно відмалювати самостійно. Або якщо ви хочете взаємодіяти з пристроями введення, такими як клавіатура.

Але те, що зробив Трюгве, було дуже революційним. Там, де всі змішували код відображення, логіки та даних, він застосував патерн, щоб розділити ці обов'язки між окремими класами.

Проблема патерну MVC полягає в тому, що це, ймовірно, один із найнеправильніших патернів у світі. І я думаю, це через назву. Трюгве спочатку назвав патерн Model—View—Editor, але пізніше зупинився на Model-View-Controller. Зрозуміло, що таке Model (дані) та що таке View (те, що я бачу на екрані). Але що таке Controller? Чи Application Controller є таким же, як у патерні MVC.[3]

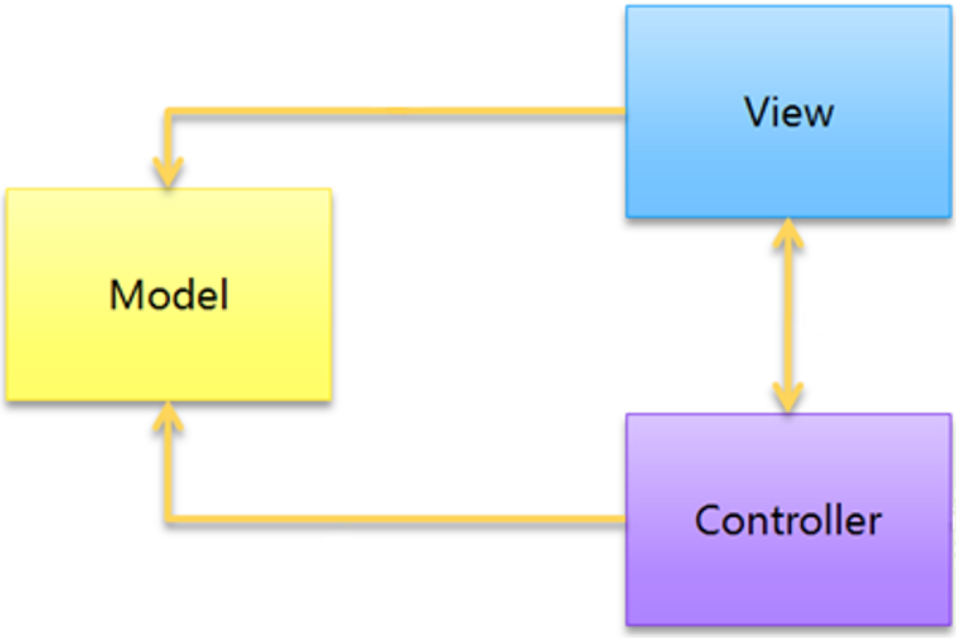

Рисунок 1.2 — Схема взаємодії MVC патерна

Model—це дані вашої програми, логіка їх отримання та збереження. Найчастіше це модель предметної області (domain model), заснована на базі даних чи результатах від веб—сервісів. У деяких випадках domain model добре проектується на те, що ви бачите на екрані. Але іноді перед використанням необхідно адаптувати, змінити або розширити.

View відповідала за відображення UI на екрані. Без бібліотек віджетів, це означало самостійне відображення блоків, кнопок, полів введення тощо. View також може спостерігати за моделлю та відображати дані з неї.

Controller обробляє дії користувача та оновлює Model або View. Якщо користувач взаємодіє з програмою (натискає кнопки на клавіатурі, пересуває курсор миші), контролер отримує повідомлення про ці дії та вирішує, що з ними робити.

Слід зазначити, що Controller отримує події введення безпосередньо, а чи не через View. Контролер інтерпретує введення користувача від клавіатури або миші, і посилає команди моделі та/або уявленню внести відповідні зміни.

було написав приклад нашвидкуруч для ілюстрації того, як виглядатиме контролер у «чистій» реалізації MVC. Я його реалізував на звичайному asp.net

(не asp.net MVC), але без застосування будь—яких елементів управління. Так що це більш традиційний asp стиль. (Так, це не дуже вдалий приклад, але я сподіваюся, що він стане відправною точкою для розуміння справжньої ролі контролера)[3].

```
{
    private readonly IView _view;
    public Controller(IView view){
        _view = view;
        HttpRequest request = HttpContext.Current.Request;
        if (request.Form["ShowPerson"] == "1"){
            if (string.IsNullOrEmpty(request.Form["Id"]))  {
                ShowError("The ID was missing");
                return;
            }
            ShowPerson(Convert.ToInt32(request.Form["Id"]));
        }
    }
    private void ShowError(string s){
        _view.ShowError(s);
    }
    private void ShowPerson(int Id){
        var model = new Repository().GetModel(Id);
        _view.ShowPerson(model);
    }
 }
```

Лістінг 1.1 — Приклад використання Controller

Після багатьох років парадигма програмування дещо змінилася — з'явилися елементи управління (віджети). Віджети як малюють самих себе, так і інтерпретують введення користувача. Кнопка знає, що робити, якщо ви

клацніть по ній. Поле введення знає, що робити, якщо ви вводите текст. Це зменшує потребу в контролері, і патерн MVC став менш актуальним. Однак оскільки існує необхідність відділення логіки додатка від представлення і від даних, набрав популярність інший патерн під назвою Model-View-Presenter (MVP).

Більшість прикладів MVC фокусуються на дуже невеликих компонентах, таких як реалізація текстового вікна або реалізація кнопки. При використанні більш сучасних технологій інтерфейсу користувача (Visual Basic 3 є сучасним в порівнянні зі Smalltalk 1979), як правило, немає необхідності в цьому патерні. Але він може допомогти, якщо ви розробляєте віджет, використовуючи дуже низький рівень API (наприклад, Direct X).

Останні кілька років патерн MVC став знову актуальним, але вже з іншої причини у зв'язку з появою ASP.NET MVC. Фреймворк ASP.NET MVC не використовує концепцію віджетів на відміну від ASP.NET. В ASP.NET MVC View є елементом управління ASPX, який малює HTML. І контролер знову обробляє дії користувача, оскільки він приймає запити HTTP. На підставі http—запиту він визначає, що робити (оновити Model або відобразити конкретну View)[4].

1.2.2.2 Model-View-Presenter

З розвитком середовища візуального програмування та впровадження віджетів, які інкапсулює малювання та обробку введення користувача, відпала необхідність у створенні окремого класу контролера. Але розробники все ще потребують відокремлення логіки від уявлення, тільки тепер на вищому рівні абстракції. Тому що виявилося, що якщо ви створюєте форму з декількох елементів користувача, вона також містить і логіку інтерфейсу і даних. Паттерн MVP описує, як відокремити UI від логіки інтерфейсу (що відбувається при взаємодії з віджетами) та даних (які дані відображати на екрані).

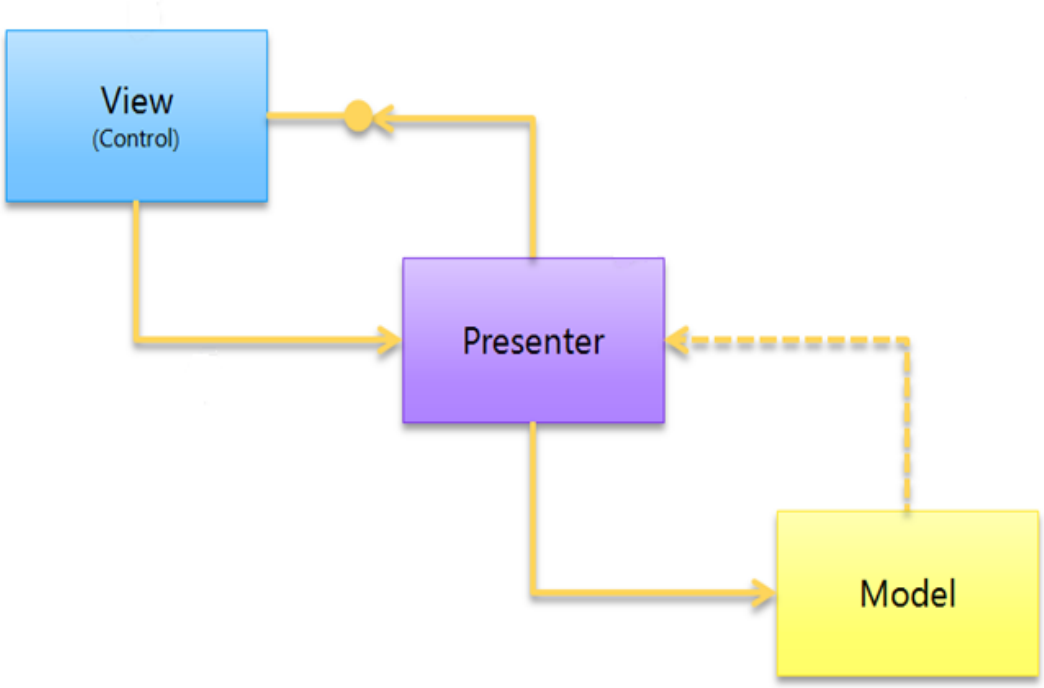

Рисунок 1.3 — Схема взаємодії MVP патерна

Model — це дані вашої програми, логіка їх отримання та збереження. Найчастіше вона базується на базі даних або результатах від веб—сервісів. У деяких випадках потрібно перед адаптацією, переглядати або змінити або розширити її.

View—зазвичай є формою з віджетами. Користувач може взаємодіяти з її елементами, але коли якась подія віджету зачіпатиме логіку інтерфейсу, View направлятиме його презентеру.

Presenter містить всю логіку інтерфейсу користувача і відповідає за синхронізацію моделі та уявлення. Коли презентація повідомляє презентатору, що користувач щось зробив (наприклад, натиснув кнопку), презентатор приймає рішення про оновлення моделі та синхронізує всі зміни між моделлю та представленням.[4]

Варто відзначити одну важливу річ, що презентатор не спілкується безпосередньо з поданням. Натомість він спілкується через інтерфейс. Завдяки цьому презентатор та модель можуть бути протестовані окремо.

Існує два варіанти цього патерну: Passive View та Supervising Controller.

У цьому варіанті MVP уявлення нічого не знає про модель, але натомість надає прості властивості для всієї інформації, яку необхідно відобразити на екрані. Презентер зчитуватиме інформацію з моделі та оновлюватиме властивості у View.

```
public PersonalDataView : UserControl, IPersonalDataView{
    TextBox _firstNameTextBox;
    public string FirstName{
        get{
            return _firstNameTextBox.Value;
        }
        set
        {
            _firstNameTextBox.Value = value;
        }
    }
}
```

Лістінг 1.2 — Приклад використання PassiveView

можна побачити, що потрібно писати багато коду як у View, так і в презентері. Тим не менш, це зробить взаємодію між ними тестованішим.

У цьому варіанті MVP уявлення знає про модель та відповідає за зв'язування даних з відображенням. Це робить спілкування між презентером та View більш лаконічним, але на шкоду тестованості взаємодії View—Presenter. Особисто я ненавиджу той факт, що цей патерн містить назву «Controller». Тому що контролер знову не той, що в MVC і не такий, як Application Controller. Цей інтерфейс менш детальним і покладає більше відповідальності на View.[5]

```
public class PersonalDataView : UserControl, IPersonalDataView {
    protected TextBox _firstNameTextBox;
    public void SetPersonalData(PersonalData data){
        _firstNameTextBox.Value = data.FirstName;
    }
    public void UpdatePersonalData(PersonalData data){
        data.FirstName = _firstNameTextBox.Value;
    }
}
```

Лістінг 1.2 — Приклад використання Supervising Controller

1.2.2.3 Presentation Model

Мартін Фаулер описує на своєму сайті інший підхід для досягнення розподілу відповідальності, який називається Presentation Model. PresentationModel є логічним уявленням інтерфейсу користувача, не спираючись на будь — які візуальні елементи.

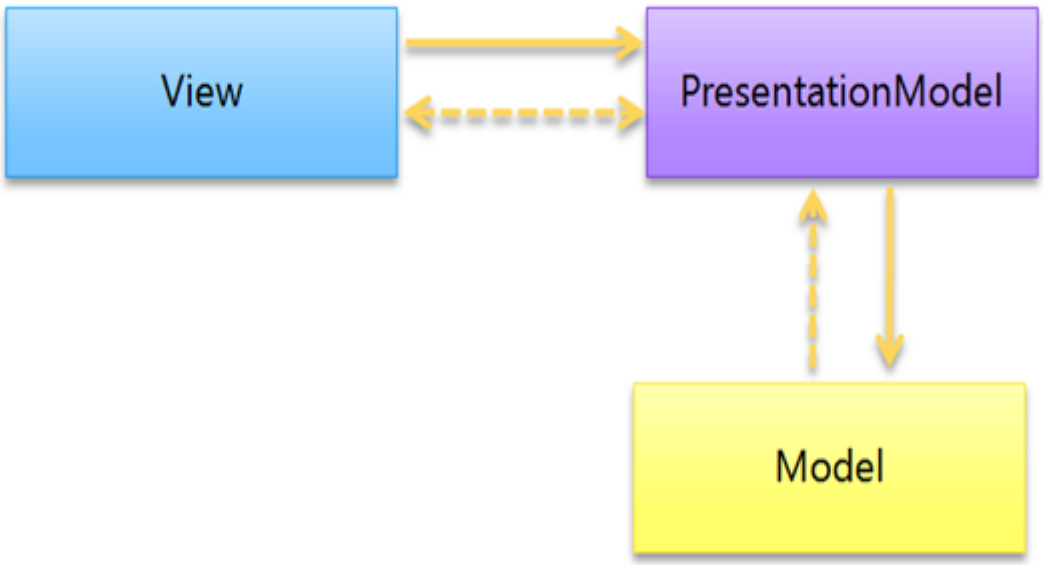

Рисунок 1.4 — Схема взаємодії Presentation Model патерна

PresentationModel має кілька обов'язків які містять логіку інтерфейсу користувача: Так само, як і презентер PresentationModel містить логіку інтерфейсу користувача. Коли ви натискаєте кнопку, ця подія направляється в PresentationModel, яка потім вирішує, що з нею робити.[5]

Також надає дані з моделі для відображення на екрані PresentationModel може конвертувати дані з моделі так, щоб вони були легко відображені на екрані. Часто інформація, що міститься в моделі, не може використовуватися безпосередньо на екрані. Вам, можливо, потрібно буде перетворити дані, їх доповнити або зібрати з декількох джерел. Це найімовірніше, коли у вас немає повного контролю за моделлю. Наприклад, якщо ви отримуєте дані від сторонніх веб—сервісів або бази даних існуючої програми.

Зберігає стан інтерфейсу користувачаЧасто інтерфейс користувача повинен зберігати додаткову інформацію, яка не має нічого спільного з моделлю. Наприклад, який елемент вибрано на екрані? Які помилки відбулися? PresentationModel може зберігати цю інформацію у властивостях.

View може легко витягувати дані з PresentationModel та отримувати всю необхідну інформацію для відображення на екрані. Одна з переваг такого підходу полягає в тому, що ви можете створити логічне та повністю тестоване уявлення вашого UI, не покладаючись на тестування візуальних елементів.

Паттерн Presentation Model не визначає, як View використовує дані з моделі (PresentationModel).

1.2.2.4 Model-View-ViewModel

MVVM або просто шаблон ViewModel. Він дуже схожий на патерн Presentation Model.

Насправді чи не єдиною відмінністю є явне використання можливостей зв'язування даних (databinding) у WPF та Silverlight. Не дивно, тому що Джон Госсман був одним із перших, хто згадав про цей патерн у своєму блозі.

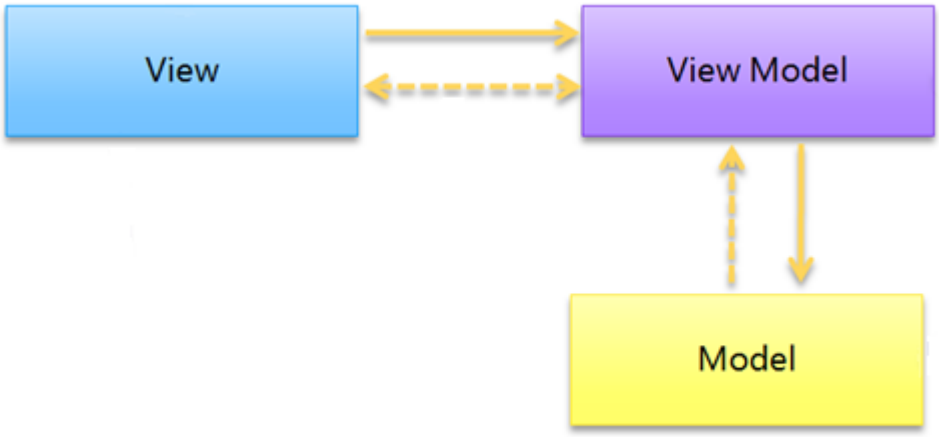

Рисунок 1.5 — Схема взаємодії MVVM патерна

ViewModel не може спілкуватися з View безпосередньо. Натомість вона представляє легко зв'язувані властивості та методи у вигляді команд. View може прив'язуватися до цих властивостей, щоб отримувати інформацію з ViewModel та викликати на ній команди (методи). Це не вимагає того, щоб View знала про ViewModel. XAML Databinding використовує рефлексію, щоб зв'язати View та ViewModel. Таким чином, ви можете використовувати будь—який ViewModel для View, який надає потрібні властивості.

Деякі з речей, які мені дійсно подобається в цьому патерні, коли він застосовується до Silverlight або WPF.

Отримується повністю логічну модель вашого додатка.Оскільки ViewModel надає View всю необхідну інформацію у зручному вигляді, саме уявлення може бути досить простим. А дизайнер може експериментувати із зовнішнім виглядом і стилем у редакторі Expression Blend і змінювати його, не впливаючи на інтерфейс користувача.

Уникнути написання коду для View (code behind). Тепер це привід для суперечок серед шанувальників патерну MVVM. Я особисто вважаю, що зазвичай вам не потрібно писати додатковий код для View, і знайдеться рішення краще. Так, іноді потрібно зробити деякі трюки (такі як створення attached behaviors), але вони забезпечують хороші і перевикористовувані

рішення. Проте я також визнаю, що не всі люблять XAML розмітку та зв'язування даних у XAML. Паттерн ViewModel не змушує вас використовувати чи уникати code behind. Робіть те, що здається правильним.[5]

1.2.2.5 Model View Intent

Ще один елемент, який можна запровадити в архітектурі, зазвичай називається MVI.

Якщо взяти якийсь елемент розмітки, наприклад кнопку, то можна сказати, що кнопка нічого не робить, крім того, що робить будь—які дані, зокрема посилає відомості про те, що вона натиснута чи ні.

У бібліотеці RxJava те, що створює події — називається observable, тобто кнопка буде observable в парадигмі реактивного програмування.
А ось TextView тільки відображає якийсь текст і жодних даних не створює. У RxJava такі елементи, які приймають дані, називаються consumer. Також існують елементи, які роблять і те, і приймають і відправляють інформацію, наприклад TextEdit. Такий елемент одночасно є і творцем (producer) і приймачем (receiver), а RxJava він називається subject.

При такому підході все є потік, і кожен потік починається з того моменту, як якийсь producer починає випускати інформацію, а закінчується на якому—небудь receiver, який, у свою чергу, інформацію приймає. Як результат, програму можна розглядати як потоки даних. Потоки даних— головна ідея RxJava.

1.3 Огляд технології хмарних обчислень

Хмарні технології—це технології розподіленої обробки цифрових даних, за допомогою яких комп'ютерні ресурси надаються інтернет—користувачеві як онлайн—сервіс. Програми запускаються та видають результати роботи у вікні web—браузера на локальному ПК. При цьому всі

необхідні для роботи програми та їх дані знаходяться на віддаленому інтернет—сервері та тимчасово кешуються на стороні клієнта: на ПК та ін.[6]

Перевага технології в тому, що користувач має доступ до власних даних, але не повинен піклуватися про інфраструктуру, операційну систему та програмне забезпечення, з яким він працює. Слово «хмара» — це метафора, що уособлює складну інфраструктуру, приховує всі технічні деталі.

Публічна хмара—одночасний доступ багатьох користувачів до IT — інфраструктури. Але можливості керувати та обслуговувати цю хмару у користувачів немає, вся відповідальність покладена на її власника. Абонентом запропонованих сервісів може стати будь—яка компанія чи приватна особа.

Приватна хмара—IT—інфраструктура, яку контролює та експлуатує лише один абонент у власних інтересах. Інфраструктура для керування приватною хмарою може розміщуватися або у приміщеннях користувача, або у зовнішнього оператора або частково у користувача та оператора.[6]

Гібридна хмара — це IT—інфраструктура, в якій об'єднані найкращі якості публічної та приватної хмари. Така композиція є унікальними об'єктами, пов'язаними між собою стандартизованими або власними технологіями, які дозволяють переносити дані або програми між компонентами.

Існує кілька рівнів хмарних обчислень:

Низький рівень "Інфраструктура як послуга" (IaaS, infrastructure as a service). Користувачі отримують базові обчислювальні ресурси: процесори та пристрої для зберігання інформації — і використовують їх для створення власних операційних систем та додатків. Споживач не управляє базовою інфраструктурою хмари, але має контроль над операційними системами, системами зберігання, розгорнутими програмами. Можливий обмежений контроль вибору компонентів мережі (наприклад, хост з мережевими екранами).[7]

Наступний рівень "Платформа як послуга" (PaaS, platform as a service). Користувачі мають можливість встановлювати власні програми на платформі,

що надається провайдером послуги. Користувач не керує базовою інфраструктурою хмари: мережами, серверами, операційними системами та системами зберігання даних, але має контроль над розгорнутими програмами та деякими параметрами конфігурації середовища хостингу.[8]

Вищий рівень хмарних обчислень "Програмне забезпечення як послуга" (SaaS, software as a service). У «хмарі» зберігаються як дані, а й пов'язані із нею програми, а користувачеві до роботи потрібен лише веб—браузер. Споживач користується програмами провайдера, який працює в хмарній інфраструктурі. При цьому користувач не управляє базовою інфраструктурою хмари — мережами, серверами, операційними системами, системами зберігання, а також індивідуальними налаштуваннями програм за винятком деяких налаштувань конфігурації програми.

На даний момент у світі правлять три гіганти — AWS, Azure, Google Cloud. Ці компанії займають левову частку ринку по всьому світу (крім Китаю, там є ще Alibaba Cloud), є технологічними лідерами та задають тренди у розвитку хмарних IaaS сервісів. Наприклад, зараз AWS має у своєму портфоліо понад 100 сервісів (IaaS, SaaS, PaaS).

Завдяки хмарним обчисленням дані організації можна аналізувати для пошуку шаблонів та відомостей, робити прогнози, покращувати їх та приймати інші бізнес—рішення. Хмарні служби можуть надати вашій організації більш високу обчислювальну потужність і просунуті засоби для отримання величезної кількості даних, а також можливість швидкого масштабування середовища з збільшенням їх обсягу.[8]

Якщо підійти з технічного боку, хмарні технології — спосіб організації фізичних і програмних засобів, а також набір інструментів, за допомогою яких користувач отримує обчислювальні потужності, щоб виконувати завдання, що стоять перед ним.

Хмарні обчислення—це ресурс, який користувач отримує у вигляді сервісу, та працює з ним віддалено. Це означає, що щоб робити обчислення та

обробляти інформацію, ви використовуєте не потужності свого комп'ютера, а сторонні. Наприклад, хмарні послуги—це:

— пошта: gmail, hotmail;

— дистанційна робота з документами: Google—документи, Office Web Apps;

— зберігання даних: Google Drive, OneDrive, Dropbox;

— редагування зображень у режимі реального часу: Figma;

— сервіси для створення нотаток, спільної роботи над завданнями: Trello, Jira, Evernote;

— онлайн—магазини програм: Google Play, App Store та Microsoft Store;

— хмарний хостинг — розміщення свого сайту в "хмарі".

У перерахованих сервісів є набір послуг для пересічних користувачів та хмарні рішення для бізнесу. У першому випадку ви отримуєте мінімальний набір функцій, якого вистачить для повсякденних завдань. Для роботи підприємства потрібен хмарний сервіс для бізнесу, бо функціонал там ширший.

Хоча хмара представляють як щось абстрактне, за ним стоїть цілком конкретний набір «заліза», програмного забезпечення та своя архітектура.

Хмарні обчислення будують на основі серверного та мережевого обладнання. Обладнання об'єднане програмним рішенням і в ньому є інтерфейс користувача для управління послугою.

Щоб зрозуміти, як працюють хмарні технології, уявіть прохолодне приміщення, де в спеціальних шафах розміщені сервери — потужні комп'ютери з великим запасом пам'яті та дисків для зберігання та обробки інформації. Щоб користувач отримав доступ до цих комп'ютерів, у них встановлено мережеве обладнання — свитчі, роутери, комутатори.

Кожна одиниця обладнання може працювати як така. Хмарні системи це коли всі елементи працюють як єдине ціле, як добре налагоджений механізм.

Щоб хмарний сервіс працював саме так, йому потрібен набір спеціального програмного забезпечення, яке диригент керуватиме всіма процесами.

Кінцевий користувач бачить готовий продукт — можливість відкрити сайт і скористатися сервісом: перевірити пошту, встановити програму на телефон, керувати проектом або отримати доступ до віддаленої бази даних.

Можливості хмари: види послуг

Компанії надають хмарні послуги для бізнесу та окремих користувачів як сервіс. Для зручності види послуг позначають абревіатурою. Найпоширеніші з них:

— SaaS — Software as a Service, або ПЗ як сервіс;

— PaaS — Platform as a Service, або платформа як сервіс;

— IaaS — Infrastructure as a Service, або інфраструктура як сервіс;

— FaaS — Function as a Service, або функція як сервіс.[9]

1.3.1 SaaS сервіс

Тут ховається Software as a Service, буквально програмне забезпечення як сервіс. Клієнт використовує програмне забезпечення провайдера, яке працює в хмарній інфраструктурі. При цьому підході створюються облікові записи клієнта: в пошті, курсах, інструментах для дизайнерів, в календарі. Завдяки цьому програми доступні з будь—яких пристроїв.

Хоча під SaaS за умовчанням мають на увазі саме програмний продукт, за цією абревіатурою може стояти Storage—as—a—Service або зберігання як сервіс. Хмарні ресурси також використовують для зберігання даних, наприклад, Google Drive, Dropbox.

1.3.2 PaaS сервіс

Platform as a Service — ви отримуєте комп'ютерну платформу, аналог комп'ютера з операційною системою, яку використовуєте для розгортання своїх додатків.[9]

PaaS як Process as a Service, або процес як сервіс, все частіше йде з приставкою Business — використовує хмарні ресурси для управління та автоматизації складних бізнес—процесів.

1.3.3 IaaS сервіс

Infrastructure as a Service — інфраструктура як сервіс — передбачає, що ви отримуєте буквально шматочок хмарної інфраструктури, в якому самі встановлюєте потрібні програми.

Information as a Service, або інформація як сервіс, відкриває доступ до масиву інформації, що швидко змінюється. Сюди можна зарахувати біржові котирування, курси валют.[9]

1.3.4 FaaS сервіс

Function as a service—функція як сервіс—дозволяє розробляти, запускати програмні продукти та керувати ними. Основна особливість—запускає певні функції у момент, коли виконується задана умова.

Ще одна відмінність у тому, що з вас знімають не абонплату за місяць, а гроші за обсяг дискового простору, що використовується, і кількість операцій на місяць, тобто за активний час користування.[9]

1.3.5 Переваги та недоліки хмарних обчислень

Хмарний сервіс—це технологія, яка покликана спростити життя та зробити сервіси доступнішими.

Головні переваги хмарних технологій:

— можливість працювати з особистими акаунтами та даними з будь—якого пристрою;

— не потрібно зберігати інформацію на флешку або на інший накопичувач;

— декілька користувачів можуть одночасно редагувати документи та файли;

— хмарні сервіси працюють у браузері, тому немає значення, яка операційна система стоїть на вашому телефоні, планшеті або комп'ютері;

— інформація зберігається на хмарному сервері — навіть якщо ПК або телефон зламається, ви не втратите дані;

— ви користуєтеся найсвіжішою версією програми: постачальник послуги сам стежить за її оновленням;

— можете ділитися інформацією віддалено, не надсилаючи великий обсяг даних, наприклад, надати доступ до папки з документами або фотографіями;

— не потрібно купувати потужний комп'ютер для розробки та розгортання програм — користуєтеся можливостями хмари та заощаджуйте;

— не потрібно бути гуру програмування та адміністрування — хмарні обчислення доступні як людям з досвідом, так і для чайників.

Основні мінуси:

— потрібний стабільний інтернет — без нього не скористаєтеся послугою;

— не будь—який продукт можна налаштувати під свої цілі та завдання;

— хоча провайдери надійно захищають хмару, завжди є ризик злому;

— створювати свою хмару дорого, тому малим підприємствам вигідніше користуватися приватною хмарою або навіть публічною, якщо йдеться про приватних підприємців;

— багато сервісів доступні безкоштовно, але не факт, що вони будуть такими завжди. Подумайте, чи готові потенційно платити за послугу і скільки.

Поки що переваги хмарних технологій переважують їхні недоліки та несуть більше вигоди, ніж ризиків.

Переважна більшість хмарних послуг доступна в розширеному варіанті — для бізнесу. Наприклад, ви користуєтеся безкоштовним особистим Google Drive. Аналогічний Google Drive, але з ширшим набором функцій, використовують дрібні та середні підприємства. Так вони отримують готове рішення для роботи з документацією, зручний доступ до неї працівників та можливість налаштувати права доступу.[9]

Так само використовують Trello та Evernote для особистих записів та корпоративний варіант для спільної роботи над завданнями.

Але крім організації адміністративної роботи та зберігання даних, бізнес має інші потреби. Розглянемо докладніше, що таке хмарні послуги для бізнесу, якими вони бувають і які компанії можуть використовувати хмарні технології.

Хмарні рішення використовують малі та великі організації у різних сферах. Цілі у них теж різні:

— резервне копіювання даних із подальшим їх відновленням;

— розробка та тестування програм;

— аналіз великих масивів інформації;

— робота з електронною поштою та налаштування віддалених робочих столів;

— зберігання програм для кінцевого користувача.

Компанії по—різному використовують переваги хмари. Розробники відеоігор відкрили для своїх користувачів можливість грати по мережі та спілкуватися між собою. Фінансові компанії відстежують шахрайські схеми як реального часу. Охоронні організації та власники магазинів бачать, що відбувається у торговому залі та оперативно реагують на ситуацію. Але хмара використовує не лише з цією метою.

Якщо ви бізнес — користувач і купуєте програмне забезпечення як послугу, SaaS, то можливостей вплинути на її роботу мінімум. Можна налаштувати її під свої потреби або звернутися до постачальника з проханням додати якийсь функціонал. Коли таких запитів набереться достатня кількість, її впровадять чи ні. Якість роботи програмного продукту та доступність даних також залежать від постачальника.

Великий плюс SaaS як послуги для бізнесу — не потрібно розумітися на технічних деталях або наймати спеціаліста, який підтримуватиме працездатність програми.

Платформи надають великі постачальники послуг, завдяки чому ваш віртуальний комп'ютер буде швидко та стабільно працювати. Плюс у тому, що не обов'язково визначати обсяг ресурсів, які знадобляться в майбутньому. Якщо потрібно розширити масштаби, це легко зробити у хмарі.

Багато провайдерів пропонують тарифні плани, де ви оплачуєте лише обсяг використаних ресурсів: пам'яті, місця на диску та кількість операцій.

Такий тип послуги підходить найчастіше середньому та великому бізнесу: щоб створити не один віртуальний сервер, а наприклад, групу серверів і запустити серйозну програму, потрібні великі ресурси, за які доведеться платити.[10]

Оскільки компанія сама налаштовує всі складові, потрібно наймати штат фахівців, які цим займатимуться.

Приклади послуг для бізнесу:

— обчислювальні потужності;

— хостинг програм;

— бази даних;

— сховища даних.

Найбільш популярні постачальники послуг у цьому напрямку: Amazon Web Services, Windows Azure, Google Cloud Platform.

IaaS—інфраструктура як послуга—означає, що ви орендуєте сервер: виділений фізичний, віртуальний або навіть віртуальний датацентр. Варіант підходить для досвідчених IT—фахівців чи компаній, у штаті яких такі є.

Постачальник послуги забезпечує стабільну роботу заліза та програм віртуалізації. Ви отримуєте налаштований сервер та доступ до управління, а також право встановити будь—яку операційну систему, програми та самостійно ними керувати.

Якщо сервер перестав підходити, можна змінити його на інший, не переживаючи, що витратилися на обладнання. Послуги IaaS надають хостинг — провайдери, у тому числі HOSTiQ.ua.

Складно передбачити, як розвиватиметься технологія, тому розглянемо тенденції, що намітилися.

Підприємства уникають покупки обладнання і воліють розміщувати обчислювальні ресурси у хмарі. У майбутньому ця тенденція збережеться і посилиться.

На ринок IT загалом та хмари зокрема виходить дедалі більше провайдерів, тому зростає конкуренція, що на користь кінцевому користувачеві, адже послуги дешевшають.

З приходом ідеї BYOD —Bring Your Own Device—співробітники пересіли зі стаціонарних комп'ютерів на ноутбуки та планшети, стали менше залежати від фізичного місця роботи. Тенденція така, що все більше людей воліють працювати з дому або коворкінгу, що додає мобільності. Сама потреба утримувати офіс знижується, а такий тренд як цифрове кочівництво набирає популярності.[10]

Позитивний момент для локальних провайдерів—довіра та інтерес до них зростають. Років п'ять тому багато хто орієнтувався на світових гігантів у хмарній індустрії, зараз «більше» не означає вигідніше та зручніше. Місцевому хостеру простіше адаптувати послуги під потреби ринку, а отже, простіше завоювати аудиторію, надавши зручний продукт.

Вже поширене таке поняття як інтернет речей, коли домашні пристрої взаємодіють між собою через інтернет без участі людини або з мінімальним втручанням. Можливість увімкнути мультиварку, закип'ятити чайник або нагріти бойлер до вашого приходу додому виглядає все привабливіше. На це є попит, а отже напрямок розвиватиметься.

Інтерес до хмарних обчислень продовжуватиме зростати, а технологія продовжить розвиватися. Водночас у клієнтів зростуть вимоги до якості послуг, а постачальників — відповідальність перед клієнтами. Чим більша частина роботи та повсякденного життя пов'язана на хмарі, тим дорожче ціна помилки та вартість хвилини простою.[11]

Хмарний хостинг стає більш затребуваним. Компанії, у тому числі vps.ua, нарощують потужності та розробляють більш досконалі системи для безперебійної роботи сервісів.

У даному розділі було розглянуто властивості операційної системи Android та різноманітні MV* патерни для зручного проектування додатку з новою технологією Compose та взаэмодією з медіа плеєром

2 ПОЄДНАННЯ ТЕХНОЛОГІЇ COMPOSE ТА МОДЕРНІЗАЦІЯ МЕДІА ПЛЕЄРУ ЛЕКЦІЙНИХ МАТЕРІАЛІВ

2.1 Огляд основних взаэмодіючих компонентів додатку

2.1.1 Архітектура мультимедійного плеєра на базі Android

Багато користувачів люблять дивитися відео за допомогою мобільного телефону, але медіаплеєр має багато обмежень. З швидким розвитком комунікацій та мереж, мультимедійні технології використовуються у медіаплеєрі. Різні підходи, показані в у даній роботі: технологія розширення плагінів, мультимедіа на основі ієрархії, медіаплеєр на основі браузера файлів, медіаплеєр заснований на FFmpeg (Fast Forward Moving Picture Expert Group), медіаплеєр з урахуванням файлового сервера.[12]

З безперервним розвитком науки і техніки мобільний телефон перестав бути просто засобом зв'язку, а став мультимедійною платформою, яка надає мультимедійні можливості.

Відтворення відео/аудіо на мобільному телефоні стало основною функцією мобільного телефону, на мобільному телефоні стало основною функцією, але медіаплеєр у платформі Android підтримує лише обмежену кількість форматів.

У цьому розділі показані різні підходи до проектування медіаплеєра. Перший—технологія розширення плагінів на програмній платформі мультимедійного плеєра android, другий—медіаплеєр, заснований на ієрархії, третій—медіаплеєр, заснований на ієрархії, третій—розробка програмного забезпечення для android—медіаплеєра, четвертий—android media framework та п'ятий—безперервний медіаплеєр.

У розділі наведено огляд різних підходів, які будуть
які використовуватимуться. Далі описується порівняння різних підходів.

Android — це інтегрована відкрита платформа для мобільних пристроїв.

Вона включає в себе операційну систему, проміжне програмне забезпечення, інтерфейс користувача та основні додатки.

FFmpeg є відкритим вихідним кодом, який виробляє бібліотеки та програми, що використовуються в аудіо та відео областях. Він підтримує більше 90 видів декодерів, а також підтримує такі протоколи, як H.261/H.263/H.264 тощо. Він надає повне рішення для перекодування, запису та потокової передачі відео.

Комплексне рішення для перекодування, запису та потокової передачі аудіо/відео. Він включає бібліотеку аудіо/відео кодеків, аудіо/відео контейнер бібліотека mux та demux, пересадка та якість кодеків.

SDL — це бібліотека, яка використовується для відображення аудіо та відеоінформації. Модуль відображення більшості медіаплеєрів заснований на:

— технологія розширення плагінів у мультимедійному програвачі android програмна платформа;

— програмна платформа мультимедійного плеєра на Android.

Jin та Jiaming описують програмну платформу мультимедійного плеєра на Android з використанням ядра OpenCORE. Упаковка ядра та надання у вигляді SDK використовується для розробки програми мультимедійного плеєра в мобільному терміналі, таких як відеоплеєр, потоковий медіаплеєр.

Практика показує, що для розвитку додатків потрібні потужне декодування та контейнер, який підтримує більше відео форматів кодування. Він описує верхній рівень — Java, який забезпечує розробку додатків інтерфейс розробки додатків (Android_FFmpeg API).Нижній шар—це шар C/C++/Java/Kotlin, який є основним шаром, що використовується дляобробки аудіо/відео даних за допомогою FFmpeg. Реалізація FFmpeg складається з аудіо/відео який упакований у нижній шар, включає джерело даних аналіз аудіо/відео, відтворення аудіо/відео, призупинення та помилки, пов'язані зі зворотним викликом, і т.д. нижнього рівня буде надана APIAndroid_FFmpeg через Java Native Interface (JNI).[13]

Нижній рівень включає ядро обробки аудіо/відео даних ядро (libffmpeg.so). У ньому зібрані пов'язані аудіо/відео функції FFmpeg. Воно включає декодування всіх основних аудіо/відео, синхронізацію між

декількома медіапотоками , завершення функцій відображення та відтворення аудіо/відео , після декодування низькорівневих бібліотек аудіо/відео обладнання(libvideo.so; libaudio.so).

Декодування вибраних компонентів досягає формату декодування із заголовка рідного файлу або потокового медіафайлу. Він також вибирає відповідний декодер для декодування стислих медіапотоків в. медіаплеєр на основі ієрархії[13]

2.1.2 Проектування медіаплеєра:

Для відтворення медіафайлів медіаплеєр спочатку збирає медіадані, декодує аудіо/відеопотоки а потім відображає дані після декодування . Під час цих трьох етапів медіаплеєру необхідно розібрати формат кодування медіафайлу, декодувати вихідні дані за допомогою відповідних програм декодування, помістити вихідні дані в буферні черги, а потім відобразити вихідні дані після синхронізації. Дизайн зменшує зв'язок між програмами .

Рівні структури системи медіаплеєра наступні інтерфейс користувача, шар декодування, шар попередньої обробки та шар вилучення даних.

Інтерфейс користувача використовується для відображення вихідних даних на медіаплеєрі для користувачів, таких як функції відтворення, паузи, зменшення та збільшення сторінки тощо.

Декодуючий шар використовується для збору інформації про медіа форматів файлів, потім декодує аудіо/відео потоки відповідним декодером, а потім синхронізує аудіо/відео потоки. Він включає всілякі декодери, модуль вибору декодера та модуль синхронізації .

Перед декодуванням необхідно зареєструвати всі формати, які можуть бути декодовані модулем. Потім необхідно забезпечити зв'язок для з'єднання відповідного блоку декодування та медіаформату.[13]

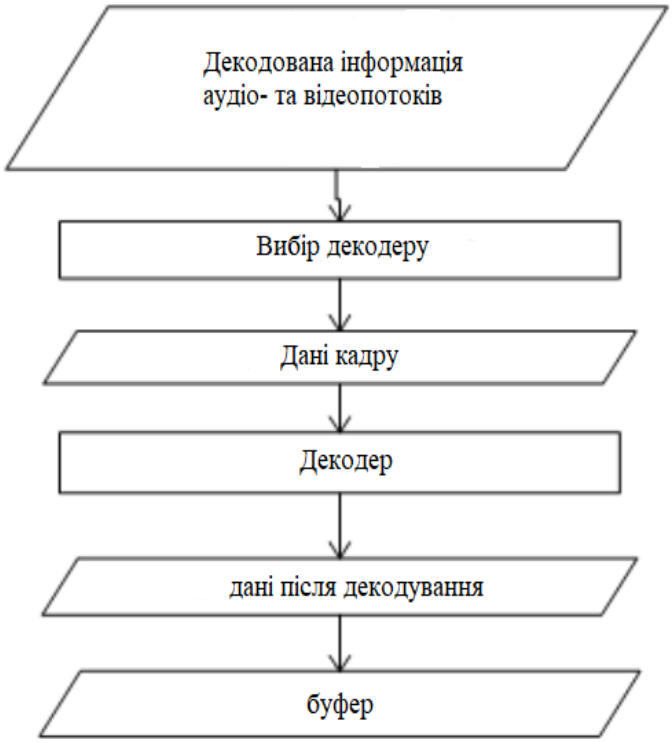

Рисунок 2.1 — Блок-схема рівня декодування

Рівень попередньої обробки використовується для демуксування медіафайлу відповідно до наявного формату та збереження інформації про медіафайл у буфері.медіафайл в буфер.

Шар отримання даних використовується для читання медіафайлу.

1.4.3 Медіаплеєр для Android

Медіаплеєр для Android це загальний механізм медіа кодека, тому він легко інтегрується в мультимедійні файли, такі як аудіо, відео та фотографії. Android Media Player відтворює аудіо файли, такі як локальні файли, файли ресурсів та мережеві потоки файлів з багатьох джерел. До них відноситься медіаплеєр, що відтворює аудіофайли з SD карти та синхронно відображає тексти пісень .

Модуль головного меню входить до початкового інтерфейсу медіаплеєра, складається із трьох опцій: Вся музика, список нещодавно

відтворених та список найчастіше відтворюваних . При запуску системи Android медіаплеєр автоматичносканує мультимедійні файли на картці SD.

2.1.2.1 Модуль головного меню

Водночас, зберігає отриману інформацію у системну базу даних. Для того, щоб дані в базу даних мультимедіа, використовується механізм широкомовлення. Потім надсилання широкомовного розсилки в програмі для оновлення бази даних мультимедіа шляхом сканування SD—карти. Потім зареєструйте scanSd Receiver, Broadcast Receiver.[15]

2.1.2.2 Модуль Play List:

Play List є основною частиною для відображення назви аудіофайлів, їх виконавців та тривалості. Play List використовується для створення списку, коли при клацанні мишею на одному з елементів вListView, він буде спрацьовує монітор setOnItemClickListener.Нещодавно відтворений модуль та найчастіше відтворений модуль:

Функція onCreate()і onUpgrade() використовується для створення та оновлення бази даних. Потім викликається функціяqueryRecently()для реалізації список нещодавно прослуханих пісень шляхом запиту пісень за часом відтворення за спаданням та викликати запитByClicks()для реалізації списку Most often список найчастіше відтворюваних пісень шляхом запиту пісень зі спадання хітів.

2.1.2.3 Модуль відтворення:

Основна функція полягає у відображенні інформації про назву, текст та час звучання пісні, а також деяких функціональних клавіш медіаплеєра, таких як відтворення, пауза та ін. функціональні клавіші, такі як відтворення, пауза, стоп, остання, наступна, назад і вперед, а також відображення тексту .D. Медіа—фреймворк Android Архітектура мультимедійної платформи Android:

Song та інші. al., визначає мета Android media framework — забезпечити узгоджений інтерфейс всім сервісів, наданих бібліотеками. Основна частина медіа—фреймворку складається з libmediaplayerservice, libmedia та libmedia_jni .

ibmediaplayerservice реалізує плеєри та медіа сервіс, які керують екземплярами плеєрів. Libmedia визначає базові інтерфейси та ієрархію спадкування. libmedia_jni є центром між java—додатком та рідною бібліотекою.

По—перше, вона реалізує специфікацію JNI так, що вона може бути використана java—додатком. По—друге, вона реалізує патерн для зручності сторони .[15]

2.2 Модернізація Android медіаплеєра

Медіаплеєр є важливою частиною мультимедійної системи Android фреймворку. Він використовується для управління відтворенням аудіо та відеофайлів.та відео .Методи Media Player реалізовані мовою C/C++ та потім компілюються у файл .so. JavaTM Native Interface (JNI)—це стандартний інтерфейс програмування для написання нативних методів Java та вбудовування віртуальної машини JavaTM у нативні програми. програми. Після того, як компоненти декодують вихідний аудіо файлу, декодований потік відправляється на аудіоапаратуру для перетворення на звуки.

Плеєр на платформі Android підтримує лише обмежений формат, тому необхідний потужний медіаплеєр на базі платформи Android форматом, що підтримується.Бібліотека декодування є найскладнішою частиною медіаплеєра. плеєра. В даний час модуль декодування більшості медіаплеєрів.

Нажаль стандартний плеєр не зможе виконати поставлене завдання , а саме відображати відео з Google Drive API , змінювати швидкість програвання файлів , підтримку відображженя лекцій вживу за допомогою потокового відображення та динамічну потокову передачу через HTTP

Тому було вирішено для модернізації взяти за основу Exo плеєр який уже має підтримку HTTP для подальшого використовування .

2.2. Добавлення нового функционалу в Медіаплеєр

Перше що було потрібно від плеєру щоб він відображав навчальні матеріали які знаходяться на Google Drive . Для цього ми використали Google Drive API щоб безпечно з допомогою декількох запросів отримати URI посилання для подальшоъ обробки та отримання медіа файлу

```
private fun buildMediaItem(source: String): MediaItem {

    return when (PublicFunctions.getMimeType(source)) {

        PublicValues.KEY_MP4 -> buildMediaItemMP4(source)
        PublicValues.KEY_M3U8 -> buildMediaHLS(source)
        PublicValues.KEY_MP3 -> buildMediaItemMP4(source)

        else -> buildMediaGlobal(source)
    }
}
```

Рисунок 2.2 Функція визначення типу Uri посилання

Перед створення медіа файлу проходить перевірка його типу з подальшим перенаправленням в функцію створення файлу MP4, HLS, MP3 типу.

HLS (HTTP Live Streaming) — комунікаційний протокол для потокової передачі медіа на основі HTTP, Архітектура. HTTP Live Streaming використовує стандартний веб—сервер для поширення аудіовізуальних матеріалів за запитом, при цьому потрібно спеціальне програмне забезпечення для того, щоб забезпечити передачу контенту в режим реальної години.

MP4 — формат медіаконтейнера, що є частиною стандарту MPEG—4. Використовується для упаковки цифрових відео— та аудіопотоків, субтитрів, афіш та метаданих, визначених групою спеціалістів MPEG. Як і більшість сучасних медіаконтейнерів, MPEG—4, частина 14, передбачає можливість показати відео через Інтернет, додатково до файлу передаються методанні, що містять потрібну для інформації інформацію. Контейнер дозволяє упаковувати кілька відеоаудіопотоків, а також субтитрів.

MP3 — це розроблений командою MPEG формат файлу для зберігання аудіоінформації. MP3 є одним із найпоширеніших і найпопулярніших форматів цифрового кодування звукової інформації. Він широко використовується у файлообмінних мережах для оцінного завантаження музичних творів. Формат може відтворюватися практично у всіх популярних операційних системах, на більшості портативних аудіоплеєрів, а також підтримується всіма сучасними моделями музичних центрів та DVD—плеєрів.

Після програвання файлу він буде зявлятись в списку де його можна буде завантажити локально в базу даних Room/Dao[14]

```
private fun buildMediaItemMP4(source: String): MediaItem {
    return MediaItem.Builder()
            .setUri(source)
            .setMimeType(MimeTypes.APPLICATION_MP4)
            .build()
}
```

Рисунок 2.3 Функція створення медіа файлу

Далі потрібно було повязати медіа плеэр з Android ОС та Compose.Щоб не виникало багів та андроід міг определити що це View для подальшого відображення медіа файлів , для цього було створено життевий цикл плеєру

```
interface AndExoPlayerListener {

    fun onExoPlayerStart() {}

    fun onExoPlayerFinished() {}

    fun onExoPlayerLoading() {}

    fun onExoPlayerError(errorMessage: String?) {}

    fun onExoBuffering() {}

    fun onExoEnded() {}

    fun onExoIdle() {}

    fun onExoReady() {}
```

Рисунок 2.4 Життевий цикл медіа плеєру

Також для управління цим плеєром було вдосконалено контрольна панель медіаплеєру, були добавлені кнопки змінни гучності та вимкнення звуку, змінювання швидкострі програвання локальних файлів

Далі було додано подвійний натиск на екран для перемотування медіа файлу на 10 секунд назад і 10 секунд вперід .

```
override var customClickListener: DoubleClick
    get() = DoubleClick(object : DoubleClickListener {
        override fun onSingleClickEvent(view: View) {
            when (view.id) {
                retryButton.id -> {
                    hideRetryView()
                    restartPlayer()
                }
                mute.id -> {
                    unMutePlayer()
                }
                unMute.id -> {
                    mutePlayer()
                }
            }
        }

        override fun onDoubleClickEvent(view: View) {
            when (view.id) {
                backwardView.id -> {
                    seekBackward()
                }
                forwardView.id -> {
                    seekForward()
                }
            }
        }
    })
    set(value) {}
```

Рисунок 2.5 Реалізація подвійного кліка на екрані

2.3 Функція складного пристрою

2.3.1 Сумісні методи

Тут функція отримує дані як параметри для класу appData. В ідеалі це незмінні дані, які функція Composable не змінює: Composable функція повинна бути функцією перетворення для цих даних. Тож ми можемо використовувати будь—який код Kotlin, щоб взяти ці дані та використати їх для опису нашої ієрархії, наприклад, викликавши функції Header () і Body ().

Це означає, що ми викликаємо інші функції Composable, і ці виклики відображають структуру нашого інтерфейсу користувача. ми можемо використовувати будь—який з примітивів Kotlin. ми можемо включити оператори if та for, які керують структурою інтерфейсу користувача для обробки більш складної логіки інтерфейсу користувача.

Сумісні функції часто використовують кінцевий лямбда—синтаксис Kotlin, тому Body () є функцією композитора, яка приймає компонуючу лямбду як параметр. Це пов’язано з ієрархією або структурою, тому Body () оточує тут набір елементів. У розробці програмного забезпечення композиція складається з об’єднання кількох бітів простішого коду в більш складний блок коду. В моделі об’єктно—орієнтованого програмування однією з найпоширеніших форм композиції є спадкування на основі класів. У світі Jetpack Compose, оскільки ми працюємо лише з функціями, а не з класами, метод складання дуже відрізняється, але він має багато переваг перед успадкуванням. Давайте розглянемо приклад.

Припустимо, у нас є попередній перегляд і ми хочемо створити поле введення. У випадку успадкування наш код може виглядати так

View є базовим класом. ValidatedInput є підкласом Input. ValidatedInput успадковується, щоб контролювати дату DateInput. Але тоді виникає проблема: ми хочемо створити компонент з діапазоном дат, тому нам потрібно перевірити дві дати—дату початку та дату завершення. ви можете підклас DateInput, але ви повинні зробити це двічі, і ви не можете цього зробити. Це обмеження спадковості: ми повинні мати одного батька, від якого ми успадковуємо.

Складати не так вже й складно. Припустимо, ви починаєте з базового введення компонента для компонування

Коли ми створюємо наш ValidatedInput, ми просто викликаємо Input у тілі нашої функції. Потім ми можемо завершити його чимось для перевірки, тоді ми можемо викликати ValidatedInput для DataInput.

Тепер, коли ми маємо справу зі введенням діапазону дат, це більше не проблема: це просто два виклики замість одного. При створенні компонентів інтерфейсу користувача за допомогою Compose вони не мають єдиного батька, що вирішує проблему, що виникла зі спадкуванням. Інший тип проблеми композиції—абстрагування від типу декоратора.

FancyBox — це подання, яке прикрашає інші уявлення у випадку Story та EditForm. ми хочемо створити FancyStory і FancyEditForm, але як? ми успадковуємо від FancyBox чи ми успадковуємо від Story? Це незрозуміло, оскільки ми знову можемо мати лише одного з батьків у ланцюжку спадкування. Compose справляється з цим дуже добре. У функції Composamble є лямбда, в якій ми описуємо дочірнє представлення, тобто. ми визначаємо View, який обертає другий View. Отже, тепер, коли ми хочемо створити FancyStory, ми називаємо історію всередині FancyBox і можемо зробити те ж саме за допомогою FancyEditForm. Це спосіб складання..

2.3.2 Компоненти перемалювання

Таким чином, можна сказати, що будь—яку функцію компонування можна викликати в будь—який час. Якщо у вас є дуже велика ієрархія для створення, вам не потрібно вказувати всю ієрархію, коли частина ієрархії змінюється. Оскільки функції компонента можна викликати знову, ви можете використовувати цю функцію для деяких корисних речей. Наприклад, ось функція гучності, яка сьогодні є в програмуванні Android, показана на рисунку 2.6.

```
@Composable
fun Messages(liveMsgs: LiveData<MessageData>) {
  val msgs by liveMsgs.observeAsState()
  for (msg in msgs) {
    Message(msg)
  }
}
```

Рисунок 2.6—Composable—компонент Messages

LiveData, для якого ми хочемо підписати оновлення View. Для цього викличте метод Observe для класу, який має життєвий цикл (LifecycleOwner — Діяльність або Фрагмент), а потім передайте лямбда. Лямбда викликається щоразу, коли LiveData оновлюється, і коли це станеться, ми будемо оновлювати та дивитися. Завдяки Compose ми можемо змінити спосіб взаємодії з LiveData

Це схоже на повідомлення композиційного компонента, які отримують LiveData і викликають метод складання integrationAsState. Метод ObserveAsState перетворює LiveData <T> у стан <T>. Це означає, що ви можете використовувати значення, отримане в тілі функції. Примірник стану підписаний екземпляром LiveData, що означає, що він оновлюватиметься щоразу, коли LiveData оновлюється. Це також означає, що незалежно від того, де зчитується екземпляр State, функція компонування навколишнього середовища, в якій він читається, автоматично підписується на ці зміни. В результаті вам більше не потрібно вказувати власника життєвого циклу або зворотний виклик, щоб оновити, оскільки Composable може неявно виконувати обидві функції.

Інша річ, яка добре справляється з компонуванням, — це інкапсуляція. Ось що слід пам’ятати під час створення композиційних API: Загальнодоступний композиційний API—це набір налаштувань, які він отримує, щоб не міг ними керувати. В якості альтернативи компонент компонента може керувати та створювати режим, а потім передавати цей режим разом з усіма отриманими даними іншим компонованим компонентам як параметри.

Тепер, коли ви керуєте цим режимом, якщо ви хочете змінити режим, ви можете дозволити дочірнім компонентам сигналізувати про цю зміну за допомогою відкликання

2.4 Побудуйте модель для написання прикладного додатку для навчального матеріалу на основі Compose

Запропоновано нову модель роботи з даними навчального матеріалу та відображення інтерфейсу користувача..

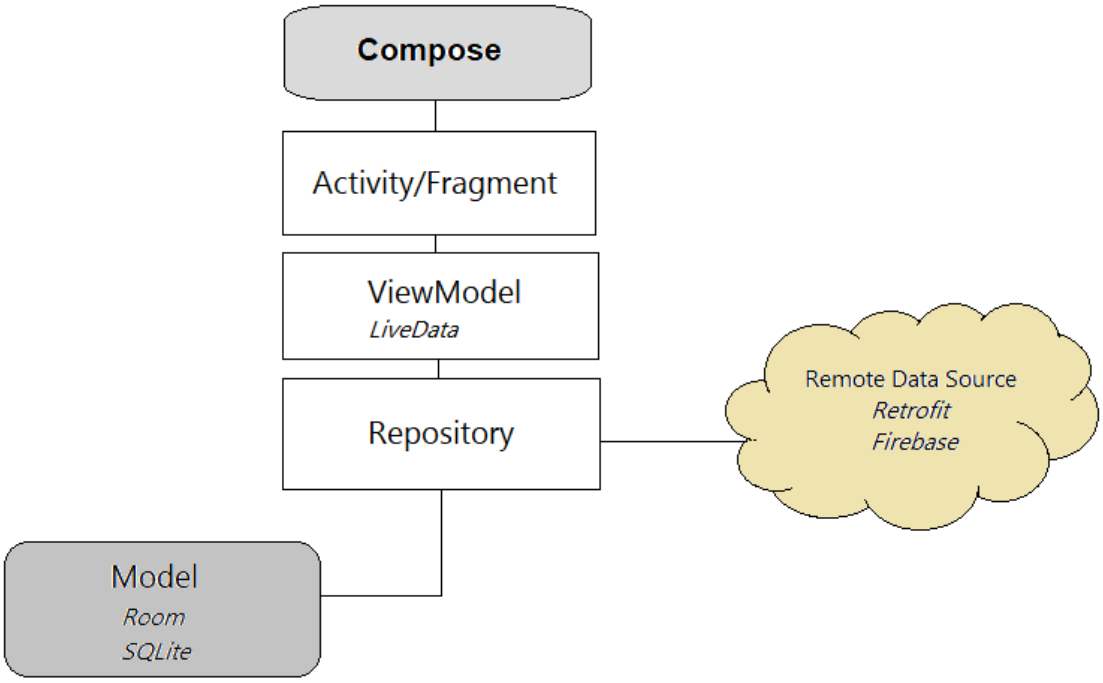

Рисунок 2.7 –додаток з комбінованим зберіганням данних та медіа файлів з використанням Compose

Це завдання реалізує роботу з компонуванням і хмарним потоковим медіаплеєром. Новим у цій роботі є те, що дані зберігаються в хмарній базі даних Firebase і в кімнаті локальної бази даних.

Ця розробка перша на ринку і не має аналогів. Цей макет даних для якнайшвидшого навчального матеріалу та розробки також забезпечує максимальну цілісність даних, оскільки він використовує хмарну базу даних Firebase, а також базу даних локальної кімнати, щоб допомогти вам зберігати дані локально та локально редагувати дані. База даних Firebase від другої сторони допомагає зберігати ваші дані в хмара також допомагає редагувати дані в хмарі.

Оскільки вам не потрібно зберігати великі мікросхеми, програма займе набагато менше пам’яті, а також менше оперативної пам’яті. Оскільки за допомогою цієї технології частина UI та UX програмно записується та відображається як у звичайному додатку. Ця технологія також дуже добре працює з точки зору швидкості програми.

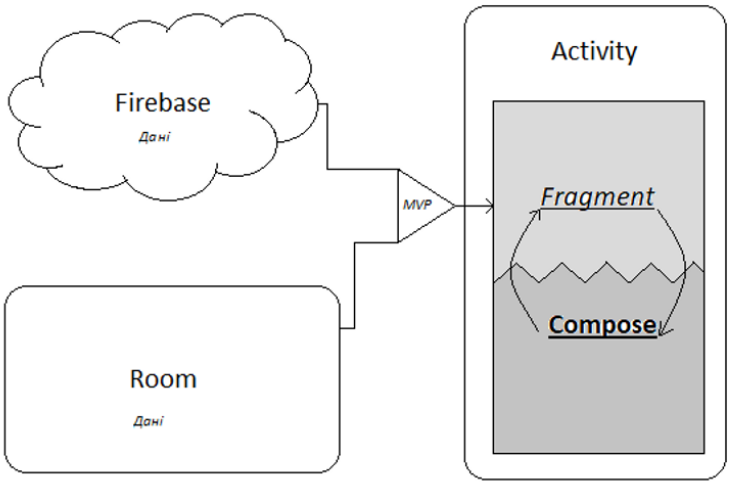

Рисунок 2.7—Модель передачі даних

Унікальною особливістю цієї роботи є додавання нової локальної технології до локальних хмарних баз даних, яка допомагає відображати дані якомога швидше, а також заощаджувати якомога більше пам’яті для зберігання великої кількості навчальних даних. Швидкість цієї програми також збільшується, оскільки не використовується система компонування XML, як показано на рисунку 2.7.

Після того як учень або викладач введе деякі дані, вони спочатку зберігаються в локальній базі даних, а потім передаються через архітектуру шаблону MVVM або MVP. Потім він розбивається на дві гілки, перша гілка надсилає дані на зовнішню firebase, друга гілка надсилає дані до моделі представлення, де дані спеціально обробляються, якщо необхідно, і передаються до фрагмента чи активності.

Використання технології, що відображається на екрані. Особливістю цього проекту є те, що програма після отримання даних не витрачатиме занадто багато часу на перенесення їх у макет, а відразу збереже фрагмент, оскільки композитна технологія знаходиться трохи нижче фрагмента. У кінці самого класу фрагмента його частини UI та UX розширюються за допомогою компонування в одному файлі.

Розроблений макет ідеально підходить для навчальних матеріалів та лекцій, оскільки в цьому випадку доводиться працювати з великою кількістю даних, а також показувати широкий спектр різноманітних елементів, що дуже трудомістко та пам’яті.

Таке поєднання технічних елементів і вже відомих технологій дозволяє набагато краще працювати з такими додатками і нейтралізувати ці проблеми.

Тож для телефону чи іншого пристрою не потрібно завантажувати графіку, потрібно лише реалізувати кілька рядків коду, що набагато швидше, ніж графічні мікросхеми.

У цьому розділі було оглянуто можливі архітектури медіа плеєру та описано модифікацію програвача Android exo, щоб було виконано поставлені перед додатком задачі а саме: безпечно завантажувати та відтворювати мультимедійні файли локально та за допомогою потокового мультимедіа на основі HTTP. Також була створена функція створення файлів HLS для подальшого відтворення. Для цього мобільного додатка було б доцільно використовувати Compose для зберігання пам’яті пристрою. Compose забезпечує сучасний підхід до розробки додатків, що дозволяє ефективно розподіляти обов’язки в коді. Оскільки функції розробки дуже схожі на стандартні функції Kotlin, і є можливість доведеться використовувати ті ж інструменти рефакторингу, що й для звичайного коду Kotlin.

3 РОЗРОБКА ANDROID ДОДАТКУ

3.1 Основні характеристики системи, що досліджується

Програмним середовищем для розробки мобільного програмного забезпечення було вибрано інтегроване середовище розробки Android Studio. Підчас створення проекту, потрібно вказати його властивості : ім’я , мінімальний SDK тамову програмування Kotlin.

Далі потрібно обрати дизайн, та визначити, як елементи дизайну будуть взаємодіяти між собою, для цього створюємо Navigation menu або navigation graph, які дозволяють спростити реалізацію навігації між екранами призначення (destinations) у вашому додатку. За промовчанням, Navigation підтримує фрагменти (Fragments) та активності (Activities) як екрани призначення, але ви також можете додати підтримку нових типів екранів призначення. Набір екранів призначення називається навігаційним графом (navigation graph) програми.

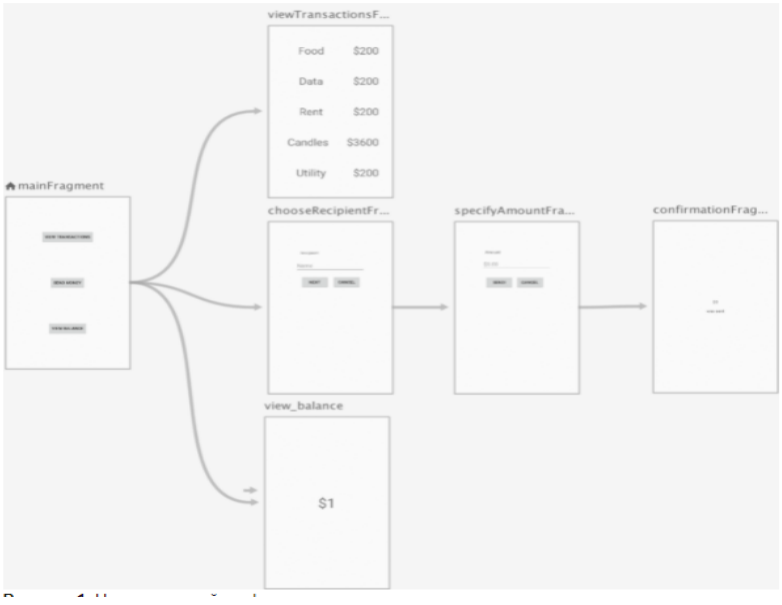

Рисунок 3.1 Навігаційний граф

Крім екранів призначення на навігаційному графі є з'єднання з—поміж них, звані діями (actions). Рисунок 3.1 демонструє візуальне представлення

навігаційного графа для простого застосування шести екранів призначення, з'єднаних п'ятьма діями.

Також з допомогою navigation graph можливо передавати дані в виді Bundl , але нами було вирішено для передачі даних саме між фрагментами і між усіма Activity була обрана спеціальна змінна дати, яка дозволяє оновлювати і оновлювати дані в режимі реального часу, а саме Livedata/Flow.

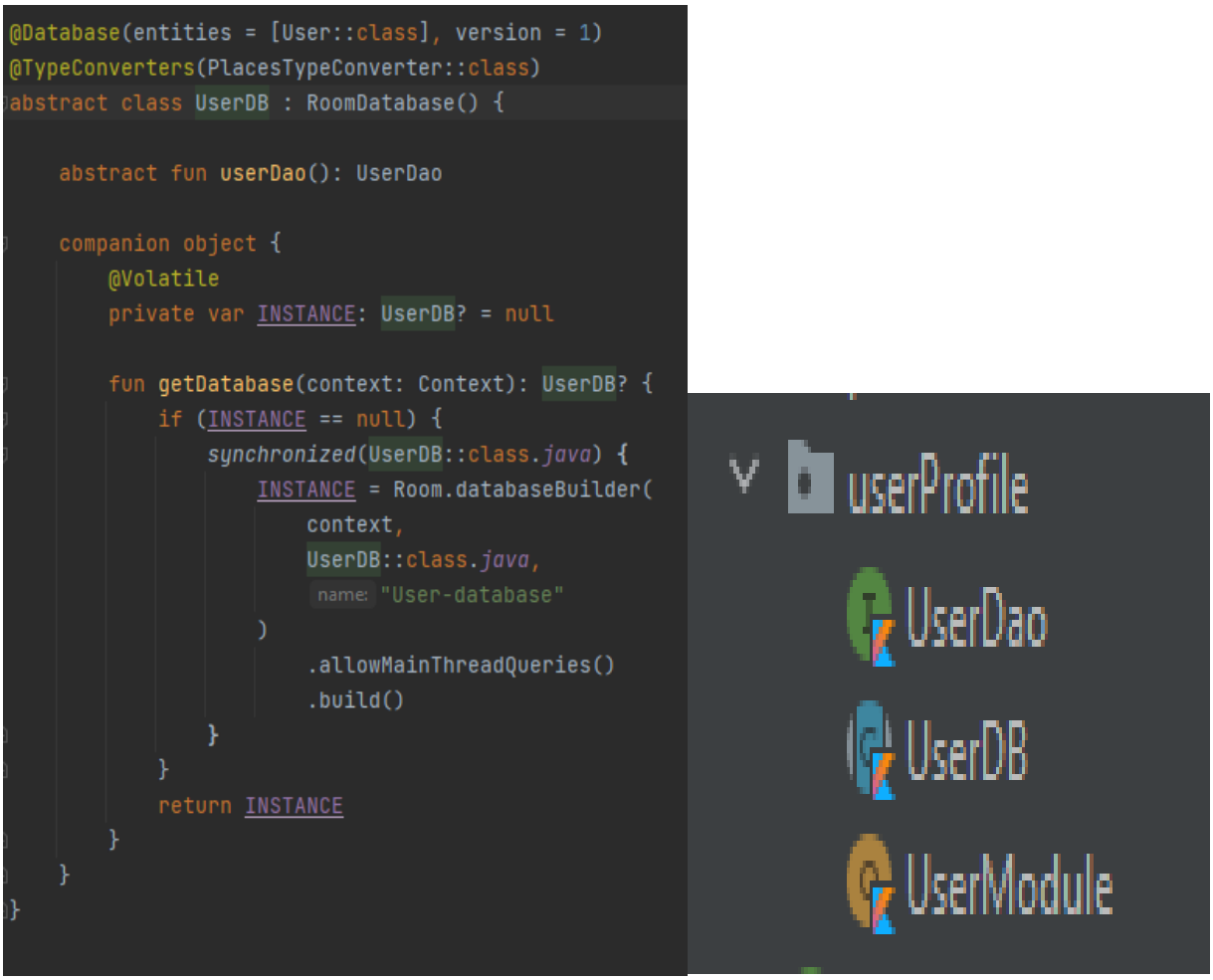

Рисунок 3.2. Локальна баха даних рум

Після чого створити локальну базу даних Room з таблицями для юзерів рисунок 3.2.Також була розглянута можливість виводу повыдомлень на екран (Notification) для подальшого використання. Далі потрібно обрати дизайн, та визначити, як елементи дизайну будуть взаємодіяти між собою, для цього створюємо Navigation

3.2 Розробка екранів

3.2.1 Розробка реєстрації та логіну в додатку

Реєстрація та Логін є невід'ємною частиною більшості цифрових продуктів. Навіть FAANG компанії не можуть створити зручну та безпечну систему входу та відновлення, здається, що всі дизайнери продуктів настільки звикли до того, що реєстрація проста, що навіть не докладають особливих зусиль до її розробки.

Реалізація схеми реєстрації, аутентифікації та відновлення доступу через електронну пошту та соціальні мережі. У зв’язку з універсальністю інструкцій не допускається посилання на конкретні фрейми, бази даних або програмні рішення, розроблені частини UI та UX на рисунку 3.3.

Рисунок 3.3—Головна сторінка та вхід у додаток

Крім того, у схемі не реалізована подвійна автентифікація, оскільки вона передбачає певні технічні рішення (наприклад, TOTP або SMS), які можуть бути продиктовані особливостями продукту або бізнес—логіки.

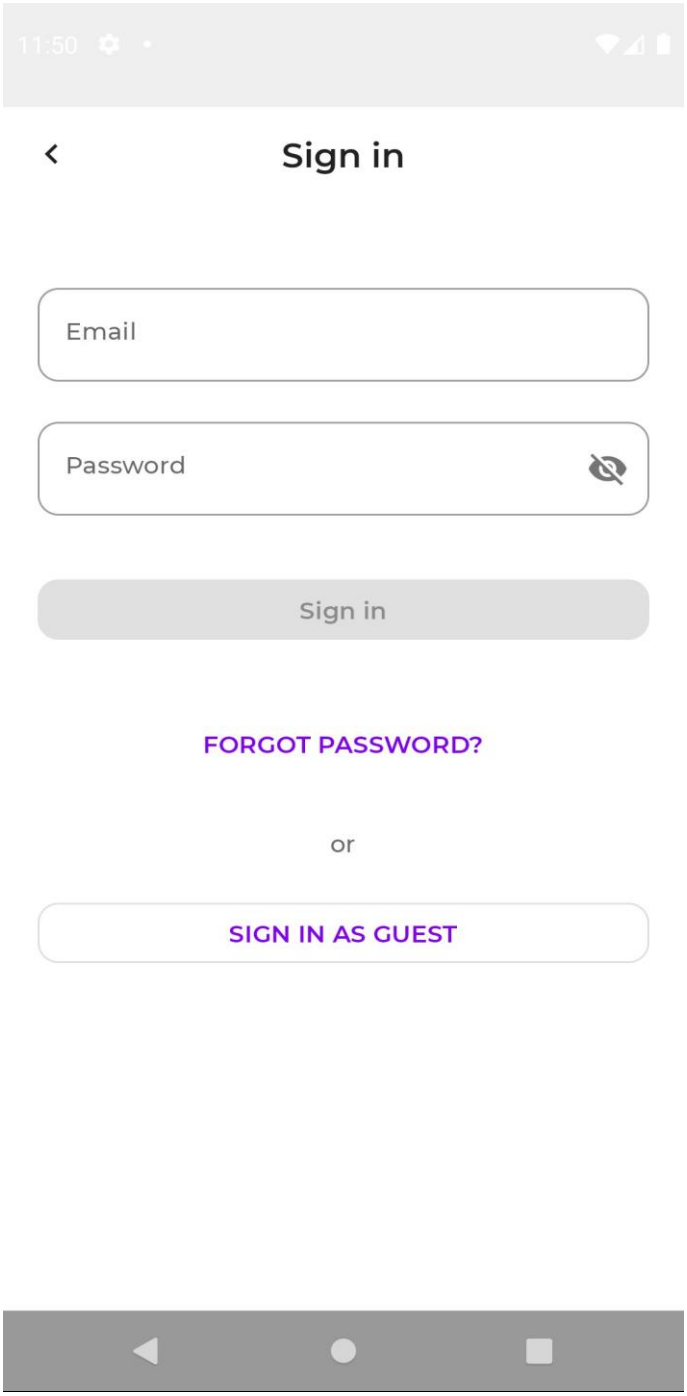

Рисунок 3.4 — Реєстрація за допомогою Copmose технології

При реєстрації через email, зображено на рисунку 3.4, проводиться перевірка на наявність пошти в базі даних з допомогою алгоритму на

рисунку 3.5. Якщо користувач з таким email вже зареєстрований, відбувається передача управління процесом загальної функції перевірка на прив'язані соцмереж».

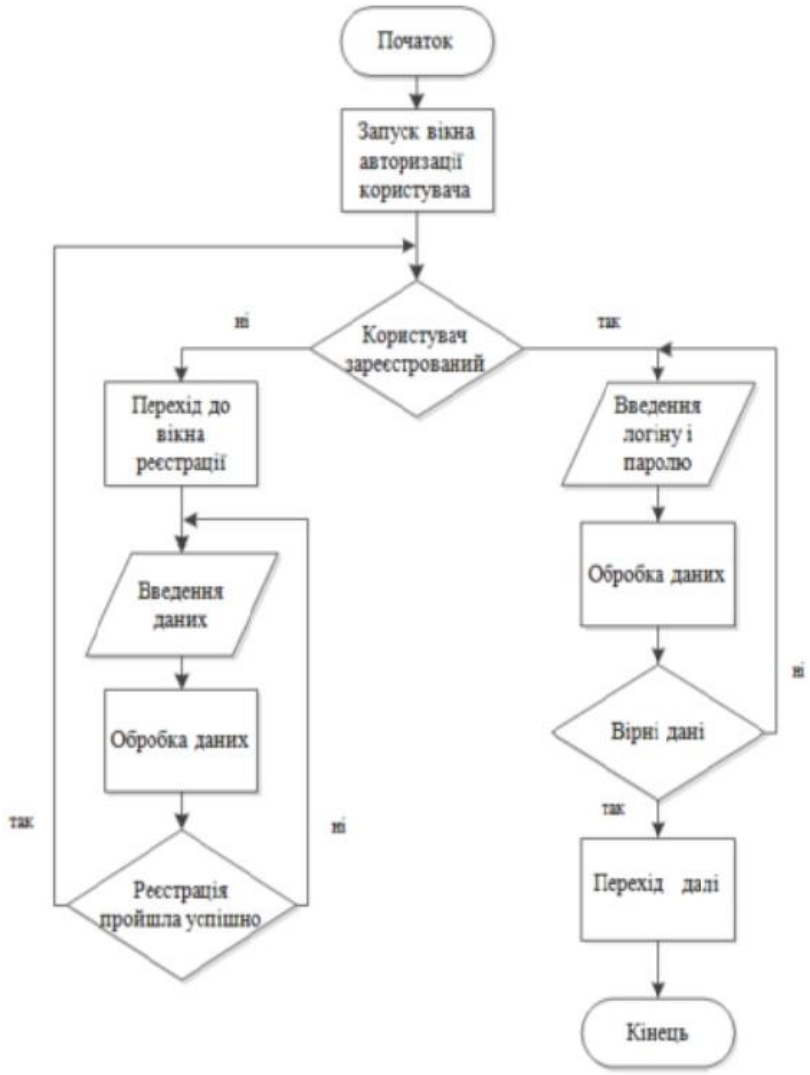

Рисунок 3.5 Алгоритм авторизації з допомогою Email

Тимчасовий додатковий обліковий запис з деякими функціями та унікальним логіном. Поки користувач не підтвердить свою адресу електронної пошти, його обліковий запис є тимчасовим. Це забезпечує більш гнучкий контроль доступу, дозволяючи користувачам обмежувати кількість дозволених дій, автоматично очищаючи базу даних від неактивних облікових записів через деякий час.

У базі даних тимчасовий обліковий запис може відхилятися від звичайного простого логічного прапорця — або такі облікові записи можуть зберігатися в окремій таблиці. Різниця в цій програмі полягає в простому прапорці.

3.2.2 Створення системи чату та каналів для навчальних матеріалів

Далі потрбіно створити розвиток системи чату та каналів

Під час розробки цього розділу було розглянуто кілька різних макетів і конструкцій компонентів. Ви вибрали шаблон оформлення за замовчуванням для цього чату. Функції, що створюють додаткові можливості для перегляду лекції.

Цей чат включає:

— назва каналу;

— кількість учасників;

— можливість пошуку в чаті;

— повідомлення;

— міні—профіль користувача з ім'ям та прізвищем;

— час повідомлення;

— кнопка для переходу до відеотеки;

— поле для написання повідомлення;

— логотип;

— панель швидкого доступу під час написання повідомлення;

— додаткова динамічна кнопка;

— кнопка надсилання повідомлення з порожнім прапорцем.

Коли ви клацнете на різницю людини із символом собаки, ви перейдете до профілю цієї особи, що допоможе вам детальніше працювати з повідомленнями.

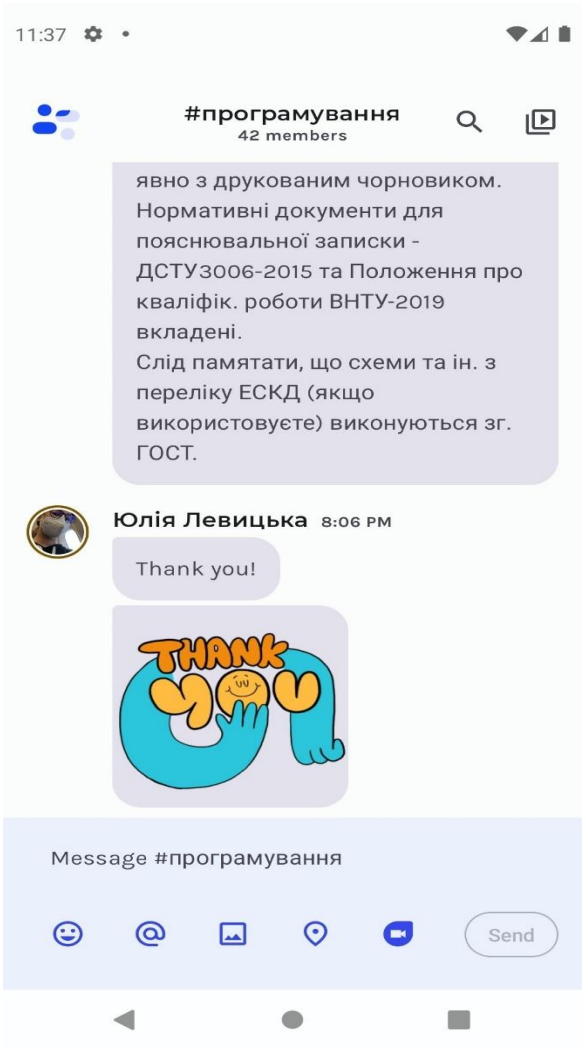

Рисунок 3.6— UI реалізація чату

Щоб зробити цей чат простішим у використанні, була використана стандартна технологія переходу до реальних даних, див. Рисунок 3.6. Додана динамічна кнопка, яка допомагає належним чином слідувати навігації, більш детально показано на рисунку 3.7

Крім того, кнопка Надіслати не буде активною, поки ви не введете саме повідомлення, що допомагає запобігти надсиланню порожніх повідомлень.
На даний момент доступно кілька функцій створення повідомлень, перша — використання емоцій та стікерів, наступна — відхилення конкретних людей за допомогою значка собаки. Існує також пошук символів, який знаходить і виділяє ключові слова в Інтернеті, як показано на рисунку 3.7.

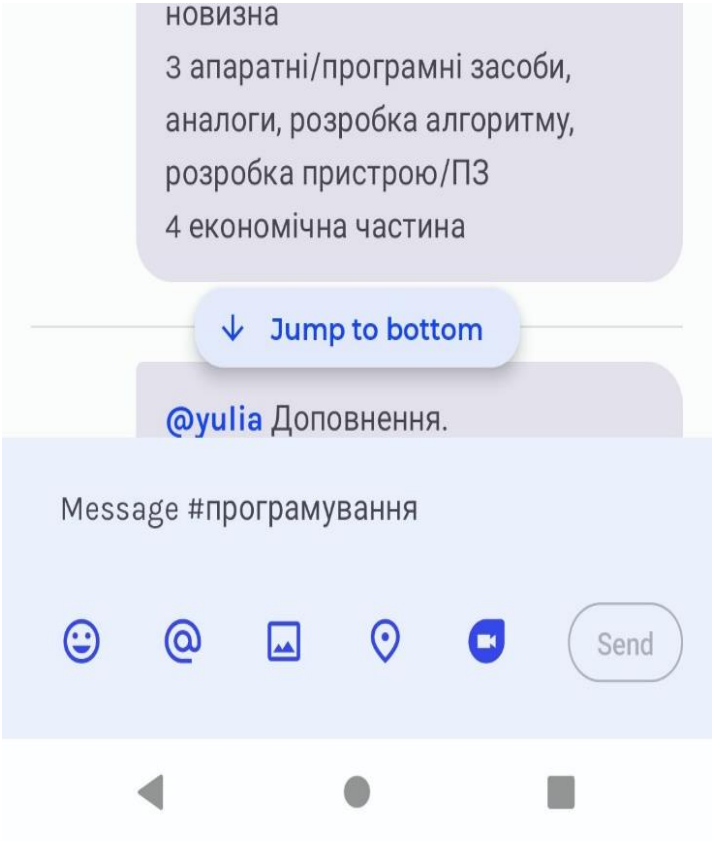

Рисунок 3.7— Реалізація кнопки навігації чату

Наступне, що реалізовано в цьому чаті—це можливість позначати власні повідомлення, а також виділяти час, час і дату відправлення. Також додається можливість стежити за чатом.

Перед відправкою повідомлення у вікні повідомлень ви можете побачити назву чату, яка допомагає вшанувати певний текстовий запис у цьому чаті

Також у цьому чаті будуть представлені деякі додаткові функції, такі як локальна передача фото та відео даних, а також онлайн—відео та фотозйомка, показані на рисунку 3.9.

Якщо користувач спробує скористатися ще не доданою функцією, він отримає спеціальне повідомлення..

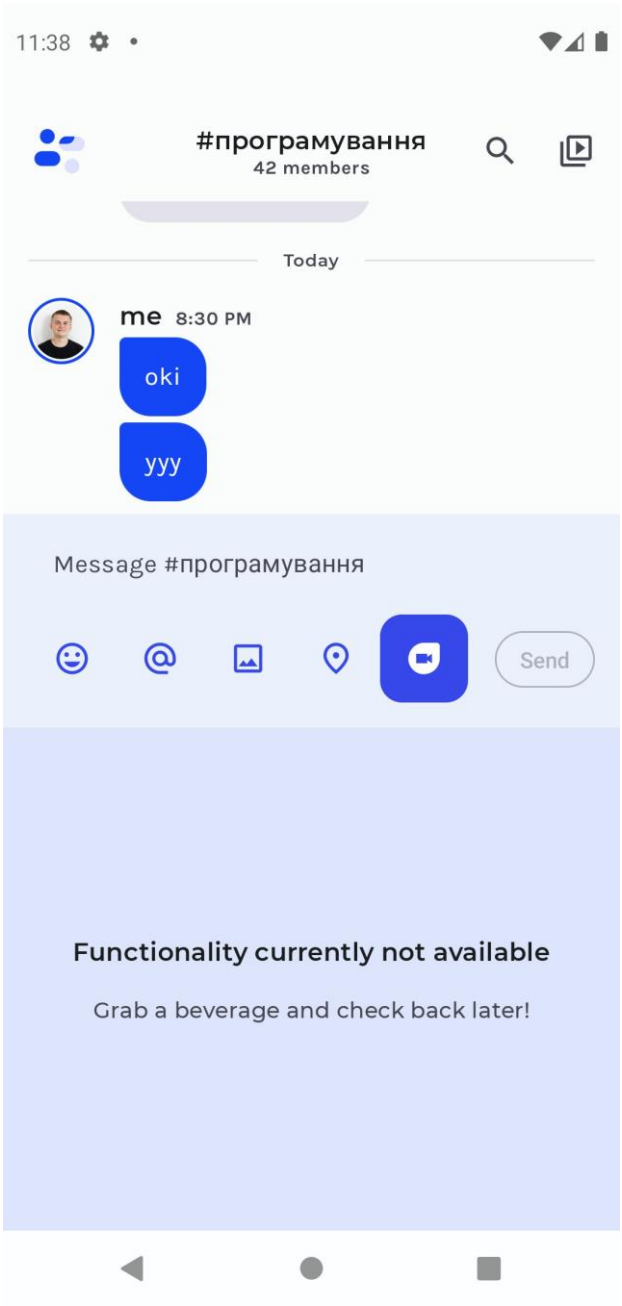

Рисунок 3.9— Використання недоступного функціоналу

Після натискання на логотип цього додатка користувач відразу переходить до реалізації фрагмента навігації сторінкою, який допоможе краще орієнтуватися в чатах і отримати доступ до всіх учасників цієї групи. При повторному натисканні користувач зможе увімкнути навігацію на бічній панелі, яка повернеться до чату та дозволить продовжити обмін повідомленнями.

3.2.3 Розробка витягу з навігації сторінкою

У цьому розділі розроблено панель навігації, яка дозволяє правильно переміщатися між чатами яка досі використовується з новими технологіями.

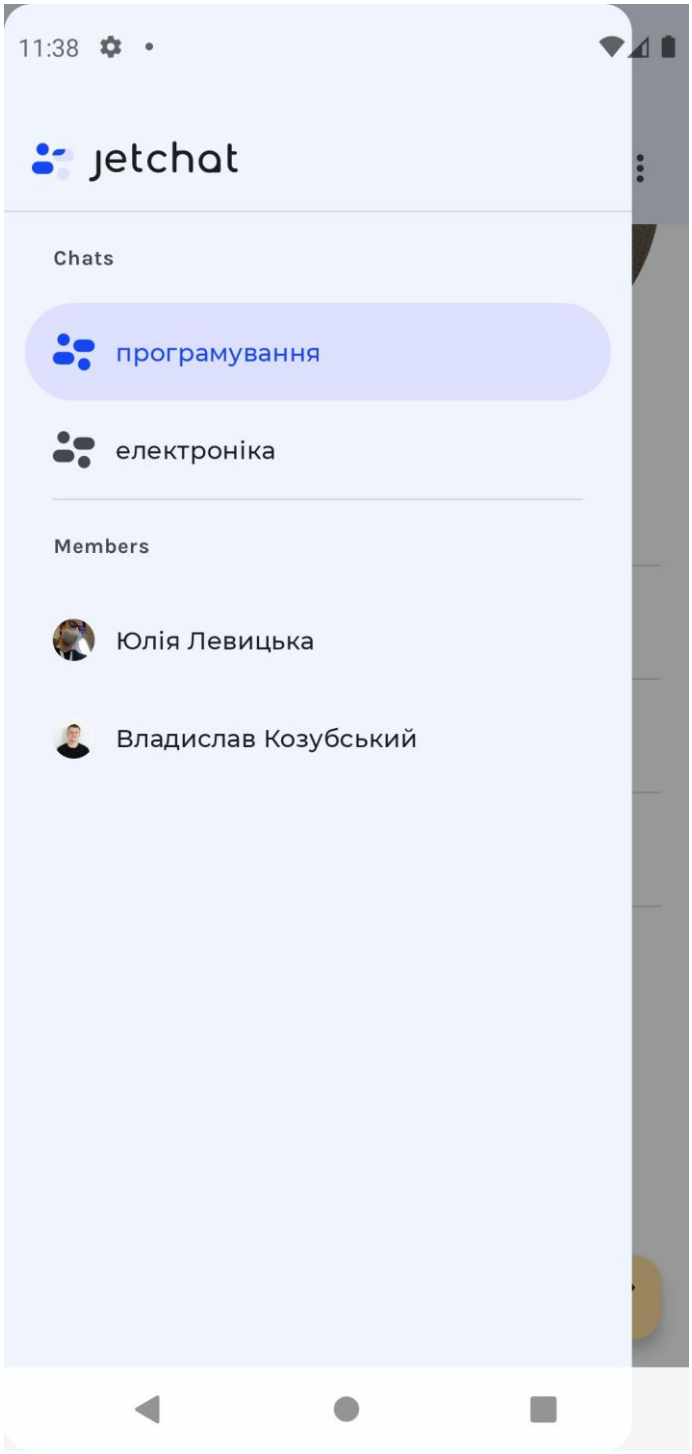

Рисунок 3.10— Панель навігації фрагменту

3.2.4 Розробка макету медіа плеєра

Стаття охоплює як ручне, так і автоматичне відтворення відео в ефективний спосіб, збереження / відновлення позиції останнього відтвореного відео, призупинення відтворення, якщо відеокарта не видима для користувача, а також обробку життєвого циклу програми. ExoPlayer буде використовуватися для відтворення фільмів і цих фільмів як тестовий набір даних. Котушка буде використовуватися для відображення мініатюр відео на рисунку 3.11.

Почнемо впроваджувати всі ці функції крок за кроком. Спочатку ми створюємо складений файл, який містить екземпляр exoPlayer, список відео та playItemIndex. Дуже важливою частиною тут є використання клавіш lazyColumn. Коротше кажучи: ми надаємо ключ, який дозволяє точно налаштувати стан елемента у разі будь—яких змін у наборі даних.

Відтворення ми завжди можемо знати, відтворюється відео чи ні. Якщо відновлено, відновлення ItemIndex відображатиме позицію елемента в списку, якщо нічого не відновить нуль. Нам потрібне це поле, щоб знати, чи хочемо ми відображати піктограму відтворення/паузи, ескіз або перегляд програвача.

Він складається з відеокарти, яка показує подію кліку на піктограмі відтворення/паузи. Клац буде оброблено всередині viewModel. Зауважте, що ми передаємо поточну позицію відтворення з exoPlayer, а також індекс елемента, який друкується. Ми будемо використовувати його для збереження та відновлення позиції відтворення кожного фільму. Кожен раз, коли ви натискаєте піктограму відтворення/паузи, ми перевіряємо три сценарії.

CurrentPlayingIndex має значення null — відео наразі не відтворюється, тому ми призначаємо videoIndex для currentPlayingIndex. CurrentPlayingIndex працює так само, як і натискання videoIndex — це означає, що той самий фільм уже відтворюється, і ми хочемо його призупинити, детальніше дивіться на рисунку 3.10.

Тому ми призначаємо значення null для currentPlayingIndex. Щоб зберегти позицію останнього відтвореного відео, ми змінюємо список і зберігаємо позицію, яку ми отримали від exoPlayer. Якщо відтворюється поточний елемент, ми створюємо VideoPlayer і надаємо йому екземпляр exoPlayer, щоб останній міг бути пов’язаний з playerView.

VideoPlayer Composable відкриває лямбда—програму, яка повідомляє нам, чи є інтерфейс програвача видимим чи ні (поточний час відтворення, тривалість відео та панель пошуку). Це дозволяє нам синхронізувати поведінку між піктограмою відтворення/паузи та попереднім інтерфейсом дисплея, який постачався з playerView.

PlayingButtonVisible—визначає, показувати чи приховувати піктограму відтворення/паузи. Ескіз відео показує зображення з URL—адреси.

Для цього нам потрібно зробити деякі налаштування в snapshotFlow. Раніше він відповідав за те, щоб ми знали, чи був предмет у грі, чи ні. Тепер він хоче розповісти нам, який предмет у центрі уваги. Точніше—якщо ми опинимося у верхній частині списку—буде зіграно перше очко. Якщо ми докрутимо список до кінця — буде відтворена остання точка. Все, що між ними, буде грати за стратегією, найближчою до середини

Рисунок 3.11— Медіа—плеєр exoPlayer

Як тільки елемент не буде видно, викличте onPlayVideoClick, який виконує описану раніше логіку збереження позицій відтворення та паузи програвача, показану на рисунку 3.11

Цей підхід базується на індексах, але ви завжди можете посилатися на об’єкти або ідентифікатори, якщо це найкраще відповідає вашим потребам. LazyListState.layoutInfo.visibleItemsInfo повідомляє нам, які індекси є видимими, ми конвертуємо їх у відеоелементи та перевіряємо, чи відповідають вони відео, яке зараз відтворюється.

3.2.5 Розробка профілю юзера

Профіль є невід’ємною частиною будь—якої програми з більшою чи більшою кількістю користувачів, тому особливий акцент був зроблений на цьому розділі.

Щоб краще зрозуміти цей розділ, були використані різні анімації, і користувач може легко переміщатися по цьому розділу завдяки продуманому дизайну ux.

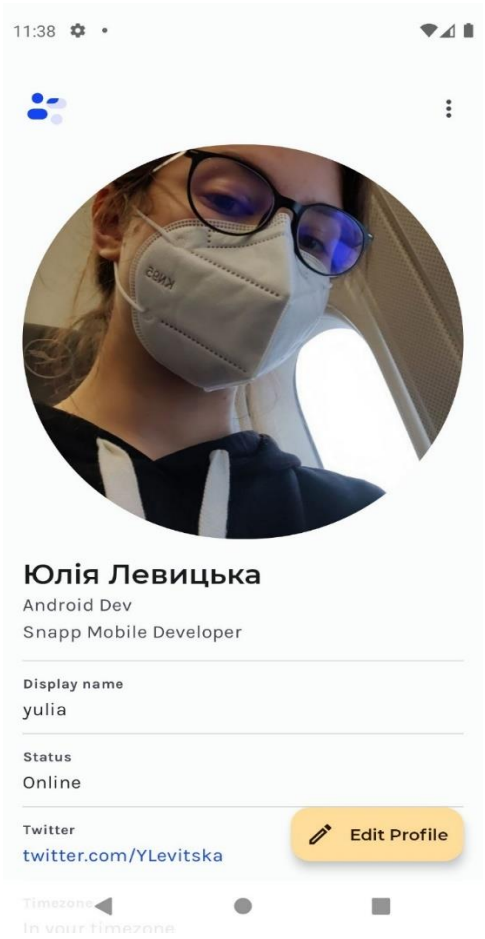

Рисунок 3.12— UI та UX профілю

Було вирішено додати до цього розділу деяку особисту інформацію, як — от фотографію, посаду чи посаду в коледжі, онлайн — та офлайн — статус, а також канал Twitter, як показано на рисунку 3.12.

Реалізовано анімацію переходу повної фотографії в стан верхнього рядка стану, а також деформацію форми заданого зображення. У розробці профілю були відомі різні власні профілі, і його функції є додатковими клавішами, які допоможуть вам або написати це безсоння, або продиктувати свій профіль..

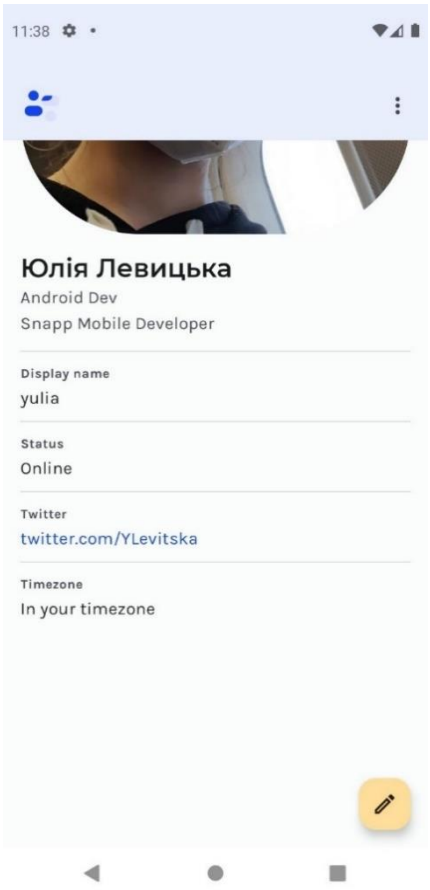

Рисунок 3.13— Динамічна зміна профілю

Незалежно від профілю користувача, його чи іншого користувача. Динамічна кнопка, яка в обох випадках відрізняється, зміниться далі. Який би профіль ми не спостерігали, він буде відкритий і доступний, як показано на рисунку 3.13.

Після відкриття кнопки ми бачимо піктограму, а також текстове пояснення. У згорнутому ключі ми бачимо лише піктограму цієї кнопки, що робить її більш контактною та дозволяє переглядати більше інформації профілю. Ця кнопка не перевантажує будь—яку інформацію користувача, роблячи її більш функціональною.

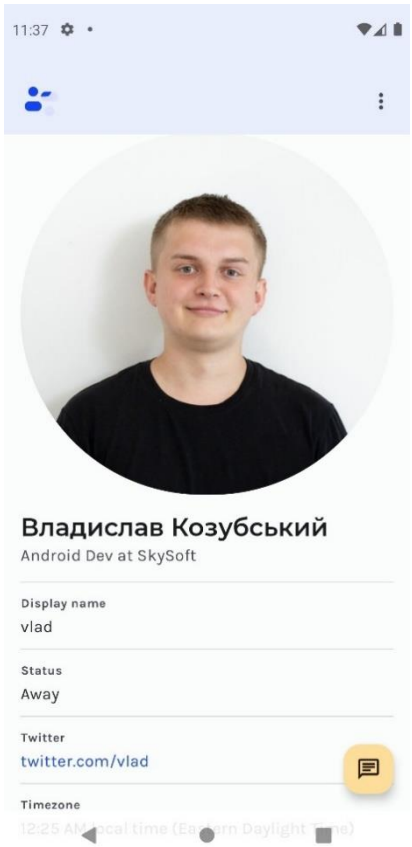

Рисунок 3.14 — Профіль іншого користувача

Коли ми переходимо на інший профіль користувача, ми можемо отримати таку функціональність, як написання повідомлення в особистих повідомленнях. Кнопка також динамічна і дозволяє відкривати приватні повідомлення з тією ж функціональністю, що й у чаті.

Деякі інші повідомлення є приватними, коли ми закриваємо приватний чат, ми отримуємо доступ до головної сторінки, що показано на рисунку 3.14.

3.2.6 Розробка розділу з навчально—методичними матеріалами та лекціями

Основним розділом цього додатка є розділ з навчальними матеріалами та лекціями. Саме тому було розроблено цей додаток. Було реалізовано декілька методів та можливостей роботи з лекціями як для студентів, так і для викладачів. Реалізовано медіаплеєр, за допомогою якого можна переглядати навчальний матеріал та лекції. Також був розроблений список, щоб правильно зберігати всі навчальні дані.

На рисунку 3.15 наведено приклад збережених навчальних матеріалів та лекцій. У цьому прикладі можна переглянути більше відео, а також посилання на міф. Також було вирішено дотримуватися стилю цього додатка, і при натисканні на логотип, який розташований у верхньому лівому кутку, буде перехід до навігації сторінкою. Також є можливість додаткового функціоналу, за допомогою значка з трьома точками, який розташований у верхньому правому куті, можна редагувати лекції та додавати нові.

Щодо студентів, то вони можуть позначати лекції та зберігати їх на локальному пристрої, в такому випадку студентам було надано такий функціонал, а також додалася локальна база даних аудиторій, щоб допомогти студентам читати лекції без доступу до Інтернету, тобто. офлайн. Для цього достатньо заздалегідь зберегти цю лекцію, яка буде доступна до тих пір, поки студент не видалить її вручну.

Використане діалогове вікно запрограмовано таким чином, що його не можна закрити, просто клацнувши на полі, крім цього вікна. Це зроблено для того, щоб користувач міг прочитати все повністю і безпечно закрити повідомлення вручну.

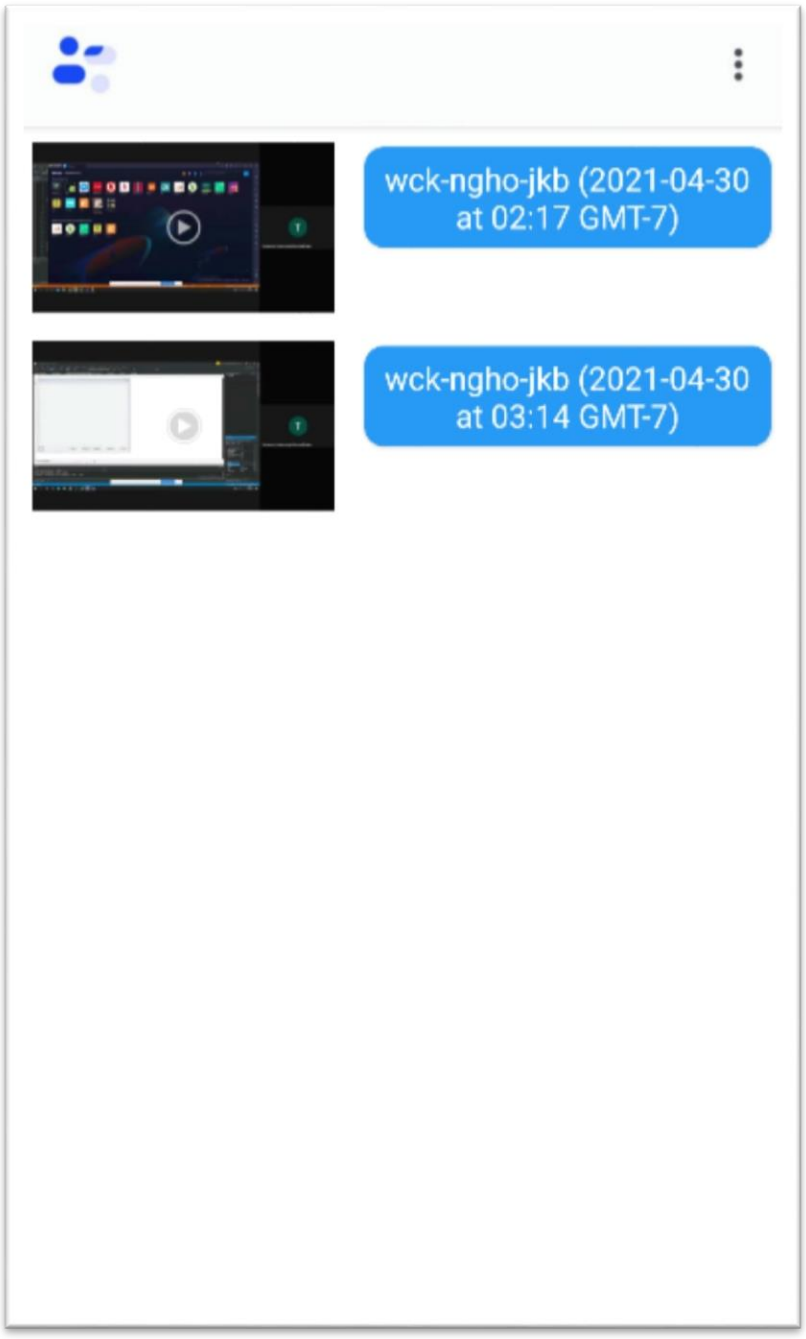

Рисунок 3.15—Список навчальних матеріалів та лекцій розробленого додатку

Юзер може переглянути лекцію, натиснувши на цей фільм або назву, після чого програвач відкриється за допомогою технології exoplauer, яка допоможе вам переглянути цю лекцію. Для більш зручного використання цього додатка можна було поділитися лекціями, а також виділити цікаву інформацію.

Використовувати чат, щоб надсилати та відзначати веселі моменти з усією групою студентів або з такою ж кількістю викладачів.

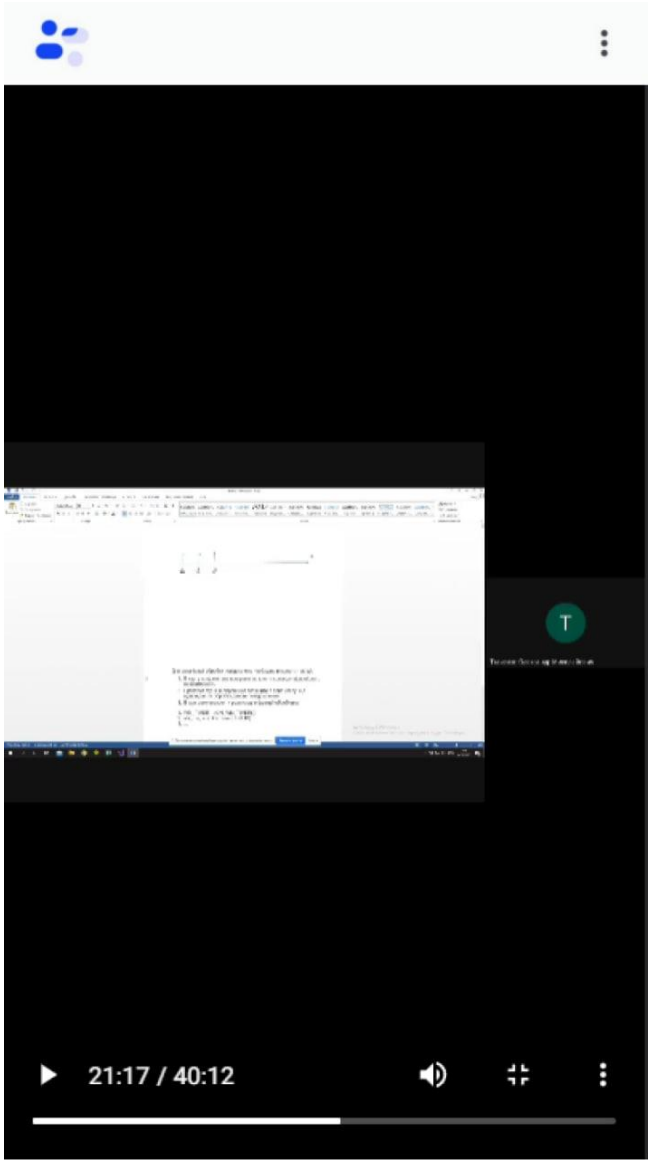

Рисунок 3.16—Відтворення лекції за допомогою exoPlayer

Відтворення лекції показано на рисунку 3.16. За допомогою технології програвача відображалося відео, а також можливість налаштовувати основні елементи, такі як час, звук, розмір тощо. Щоб відкрити цю лекцію в повному розмірі, просто переверніть екран і активуйте опцію перегляду ландшафтна орієнтація. Це допоможе учню краще побачити матеріал, надісланий вчителем.

3.2.7 Розробка додоткового функціоналу

Для цієї програми надано додаткові функції. Оскільки додаток знаходиться на стадії розробки і містить не всі реалізовані функції та ідеї. Коли ви відкриваєте таку ідею чи функцію, з’являється повідомлення. Використання діалогового вікна, яке сповіщає користувача про недоступні функції. Це вікно також можна закрити за допомогою спеціальної кнопки.

Щоб уникнути деяких моментів, коли користувач випадково проігнорував це діалогове вікно і не встиг прочитати про недоступний функціонал або ідея не була реалізована. Такий функціональний діалог показано нижче, а саме на рисунку 3.17..

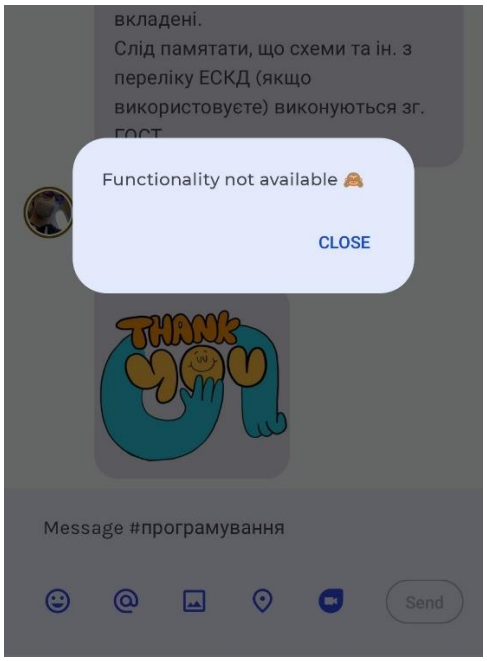

Рисунок 3.17—Функціональне діалоговек вікно.

Такі діалогові повідомлення було використано у всіх нералізованих функціях та ідеях .у самому чаті та на головній сторінці даного додатку чи то при відкриття навігації, діалогові повідомлення не відрізняються один від одного та викликається й закриваються однаково.

3.3 Порівняння з існуючими аналогами

Сучасний світ і сучасна реальність потребують спеціальних пристроїв для функціонування в мережі. Це найголовніше сьогодні. Необхідно проаналізувати переваги та недоліки альтернативних застосувань. Хоча їх дуже мало, ми можемо навести декілька з них. Розроблений додаток порівняли з таким гігантом, як Google Classroom, а також з Jetiq.

Звичайно, кожен із цих прикладів відрізняється від подібних додатків, в чомусь виграє, а в чомусь ні. Щоб краще зрозуміти, як нове програмне забезпечення обходить аналогові програми на ринку, необхідно проаналізувати недоліки цих додатків і впровадити їх виправлення в їх розробку, а також додати готові переваги.

Було проведено ряд досліджень, щоб допомогти ідентифікувати та проаналізувати існуючі аналоги. Найважливішими вимогами будуть швидкість і обсяг пам'яті, а також додаткові можливості програми. З цією метою буде проведено два дослідження. Технічно це допоможе визначити швидкість кожної програми, а також вартість пам’яті. Нижче наведено функціональне дослідження, яке допоможе вам проаналізувати низку функціональних можливостей, а також простоту використання.

Порівняння функціональної частини та характеристики переваг/недоліків конкуруючих мобільних додатків представлено нижче, а саме в таблиці 3.2. На основі даних, що містяться в цій таблиці, можна зробити висновки та виправити дії для реалізації окремих елементів розробленого додатка.

Після перегляду конкуруючих аналогових додатків для Android ви можете проаналізувати їх плюси та мінуси та на основі цієї інформації зробити корисні висновки щодо концепцій, які слід розглянути, або уникнути тих самих помилок під час виконання цієї вичерпної дипломної роботи..

Технічні переваги та недоліки альтернативних додатків зображенні в таблиці 3.1.

Таблиця 3.1 — Технічні переваги та недоліки альтернативних додатків

Назва програми	Швидкість відкриття додатку та лекції	Кількість затраченої пам яті
Google Classroom	—1890 мс	40 Мб
JetIq	—3120 мс	32 Мб
Jetchat for VNTU	—200 мс	9 Мб

Таблиця 3.2 — Функціональні переваги та недоліки альтернативних додатків

Назва програми	Переваги	Недоліки
Google Classroom	—можливість менеджити студентів у спеціальних групах; —можливість перевіряти завдання; —захищеність; —можливість залишати коментарі;	—не користувацько дружельбний інтерфейс; —немає можлиіості створювати онлайн чат; —не розвинена система додаткової інформації про студента (профіль); —не має можливості одразу додавати кілька зображень у завдання;
JetIq	—присутній розклад для студента; —інтуєтивно зрозумілий інтерфейс; — розвинена система додаткової інформації про студента; —можливість менеджити студентів у групи;	—немає можлиіості створювати онлайн чат; —немає можливості залишати коментарі; —не має можливості завантажити матеріали; —не можливість працювати офлайн; —часті проблеми при вході у додаток;
Jetchat for VNTU	—інтуєтивно зрозумілий інтерфейс; —можливість менеджити студентів у групи; —захищеність; — можливість додавати кілька зображень та відео у завдання; — можливість працювати офлайн;	—відсутній розклад для студента; —немає можливості залишати коментарі; —не має конкретної інформації про підгрупи студента.

З технічної точки зору цей додаток не має аналогів, оскільки є найшвидшим і найдешевшим в пам'яті. Це завдяки новітній технології складання, яка допомагає відображати все це з феноменальною швидкістю, а також заощаджує мінімальний обсяг пам’яті. Як показано в таблиці 3.2, інші програми не мають шансів конкурувати з цією розробкою.

У цьому розділі розроблено мобільний додаток на базі Android і створено правильну навігаційну карту для програми. Я розробив інтерфейс користувача та додав модифікований плеєр як модуль. Для вирішення цієї проблеми були узгоджені та застосовані такі технології верстки, як Compose, Room, Exo player.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ СТВОРЕННЯ ANDROID ДОДАТКУ ДЛЯ НАВЧАЛЬНИХМАТЕРІАЛІВ

Дослідження та впровадження в роботу нових технологій є витратними. Тому витрати на виробництво та реалізацію товарів необхідно постійно зменшувати, оскільки в перспективі кошти можуть принести прогрес для будь-якого виробництва. На основі економічних розрахунків можна продемонструвати рентабельність та ефективність впровадження результатів досліджень у виробництво, тобто комерціалізація наукових досліджень. Дана магістерська кваліфікаційна робота відноситься до розряду прикладної науково-технічної роботи. Прогнозується виведення науково-технічної розробки на ринок із залученням потенційним інвестора. Дану послідовність приведено на рисунку 4.1.

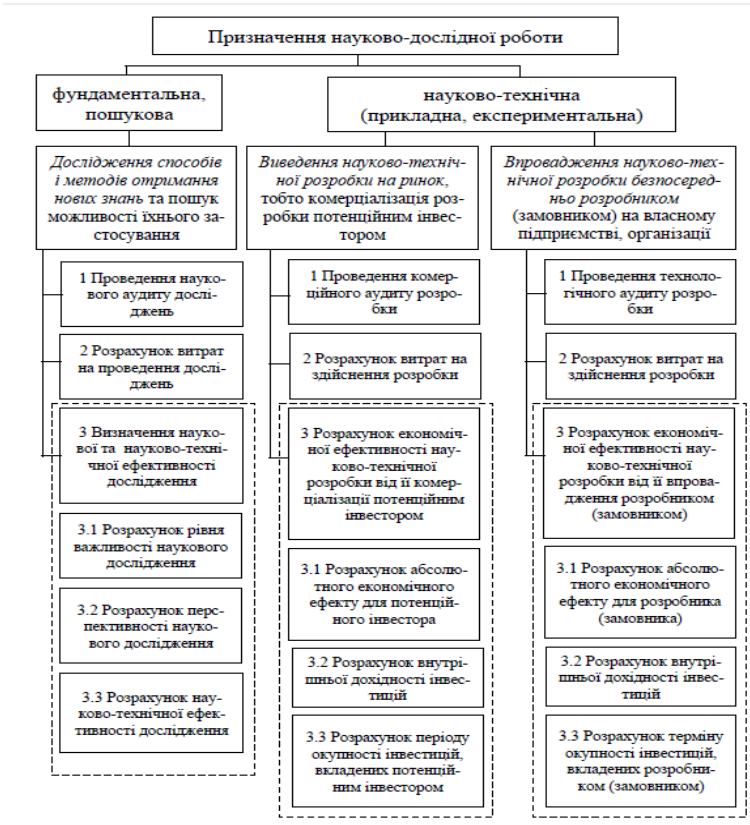

Рисунок 4.1 — Складові економічної частини магістерської кваліфікаційної роботи

Економічна частина цієї магістерської роботи буде поділена на такі елементи. Усі подальші економічні розрахунки будуть розглянуті у згаданих розділах економічної частини.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою оцінки потенціалу комерційного розвитку є оцінка потенціалу комерційного розвитку, що випливає з науково-технічних досліджень. За результатами оцінки робляться висновки про напрямки (особливості) організації в майбутньому її впровадження з урахуванням встановленої оцінки. Комерційний потенціал інвестицій буде оцінюватись відповідно до дванадцяти критеріїв, наведених у таблиці 4.1.

Таблиця 4.1 — Оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Багато аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
Практична здійсненність					
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси.Джерела фінансування є	Потрібні незначніфінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років.Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років.Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності інвестицій менше 3-х років
12	Необхідно регламентні документи та велика кількість дозвільних документів на виробництво продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

На основі таблиці різні експерти, у нашому випадку викладачі кафедри ОТ визначають різні результати. Результати цієї оцінки комерційного потенціалу узагальнено у таблиці 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	1 Черняк О.І., к.т.н., доц. кафедри ОТ	2 Семеренко В. П., к.т.н., доц. кафедри ОТ	3 Крупельницький Л.В., к.т.н., доц. кафедри ОТ
	Бали:		
1.Технічна здійсненність концепції	3	3	3

2.Ринкові переваги (наявність аналогів)	4	4	2
3. Ринкові переваги (ціна продукту)	3	3	3
4.Ринкові переваги (технічні властивості)	3	4	3
5. Ринкові переваги (експлуатаційні витрати)	3	2	3
6. Ринкові перспективи (розмір ринку)	4	4	4
7. Ринкові перспективи (конкуренція)	3	3	2
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	2	3	3
10.Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	$СБ_1 = 37$	$СБ_2 = 38$	$СБ_3 = 36$
Середньо-арифметична сума балів $СБ_С$	$СБ_С = \frac{\sum_1^3 СБ_i}{3} = \frac{37 + 38 + 36}{3} = 37$		

Відповідно до таблиці 4.2, а також відповідно до рекомендацій, наведених у таблиці 4.3, можна зробити висновок про рівень потенціалу комерційного розвитку .

Таблиця 4.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 — 10	Низький
11 — 20	Нижче середнього
21 — 30	Середній
31 — 40	Вище середнього
41 — 48	Високий

З урахуванням середніх арифметичних балів $СБ_С$ = 37, які були визначені експертами, можна зробити висновок, що рівень комерційного потенціалу цієї розробки буде вище середнього.

Також, ще один аналог розробленого програмного забезпечення — JetIQ, який є повністю розробкою ВНТУ. Дане програмне забезпечення є максимально простим для освоєння та використання, та тримає в собі інформацію щодо розкладу студента , його тестів , та успіхів

Основний недолік – повна відсутність інтеграції з сторонніми сервісами та вивантаження даних. Також для корпоративних клієнтів може стати недоліком неможливість встановлення серверної частини android додатку для навчальних матеріалів в локальній мережі підприємства, лише в хмарі. Порівння розробки з її аналогами приведено в таблиці 4.4.

Таблиця 4.4 — Порівняння характеристик розробки із аналогом

Показники	Розробка	Аналог1	Аналог2
Функціонал	9	9	9
Швидкодія	8	7	8
Надійність	8	8	7
Метод розповсюдження	8	7	8
Інтерфейс, простота використання	8	6	8

Продукт буде просуватись за допомогою його презентації потенційним клієнтам. Додатково потенційного клієнта можна знайти за допомогою реклами в соціальних мережах, пошукових системах та багатьох інших джерелах Інтернету. Використовуючи аналітику цих сервісів, можна буде націлити рекламу на цільову групу захисників інформації.

Продукт може бути використаний як для підприємств великого, так і середнього та малого бізнесу. Продукт є універсальним в плані сфери університету.

Новизна дослідження полягає у використанні android додатку для навчальних матеріалів, яка дозволить викладачам контролювати студентів в вивченні навчальних матеріалів

Виходячи з результатів цього порівняння, можна з упевненістю сказати, що новий дизайн є конкурентоспроможним, оскільки в деяких аспектах він переважає одного з найкращих аналогів на ринку.

4.3 Розрахунок витрат на здійснення науково-дослідної роботи

У магістерській роботі розглядається android додатку для навчальних матеріалів, тому значну частину витрат складають витрати на розробку, а не на виробництво та відтворення. Відповідно, є певна специфіка розрахунків.

4.3.1 Витрати на оплату праці

Основна заробітна плата розробників, що працюють над проектом, визначена у формулі:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де k — кількість посад дослідників, залучених до процесу досліджень;

$М_{ni}$ — місячний посадовий оклад конкретного дослідника, грн;

$Т_р$ — середня кількість робочих днів в місяці, Тр = 21…23 дні; обрано 22 дні;

t_i— кількість днів роботи конкретного дослідника, дн.

Над створенням розробки працював менеджер проекту та android інженер програмного забезпечення, тому ми виконаємо для данх працівників усі необхідні розрахунки, та після чого вносимо їх до таблиці 4.5:

$$Зо.к. = \frac{16500 \cdot 8}{22} = 6000(грн),$$

$$Зо.в. = \frac{15000 \cdot 40}{22} = 27272\ (грн).$$

Таблиця 4.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	16500	750	8	6000
android інженер	15000	600	40	27272
Всього				33272

Додаткова заробітна плата ($З_{дод.}$) усіх розробників та працівників, які брали участь у цьому етапі роботи, обчислюється як 10 ... 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод.} = (З_о + З_р) \cdot \frac{Н_{дод.}}{100\%}, \quad (4.2)$$

де $Н_{дод.}$ — норма нарахування додаткової заробітної плати.

$$З_{дод.к.} = \frac{10 \cdot 6000}{100} = 600 \text{ (грн)},$$

$$З_{дод.в.} = \frac{10 \cdot 27272}{100} = 2727 \text{ (грн)},$$

$$З_{дод} = З_{дод.к.} + З_{дод.в.} = 3327 \text{ (грн)}.$$

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та нарахування на заробітну плату працівників, які брали участь у цьому етапі роботи, розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за наступною формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (4.3)$$

де $Н_{зп}$ — норма нарахування на заробітну плату.

$$З_н = (33272 + 0 + 3327) \cdot \frac{22\%}{100\%} = 10647 \text{ (грн.)}$$

4.3.3 Сировина та матеріали

Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за наступною формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.4)$$

де $Н_j$ — кількість матеріалу j-го виду, шт.;

n — кількість видів матеріалу.

$Ц_j$ — ціна матеріалу j-го виду, грн;

$К_j$ — коефіцієнт транспортних витрат, $К_j$ = (1,1…1,15), обираємо $К_j$ 1,15;

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.;

Результати розрахунків занесено до таблиці 4.6.

Таблиця 4.6 — Витрати на матеріали

Найменування комплектуючих	Ціна за 1 штуку, грн	Кількість матеріалу, штук	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Ручка	30,00	1	0,06	1,80	21,53
Пачка офісного паперу	176,00	1	0,5	2,50	45,25
Всього (з урахуванням транспортних витрат)					66,78

4.3.4 Розрахунок витрат на комплектуючі

Оскільки кінцевий продукт, який ми створюємо — це android додаток , це не спричиняє жодних витрат на компоненти та $К_в = 0$.

4.3.5 Спецустаткування для наукових (експериментальних) робіт

Спецустаткування для проведення експериментальних робіт по створеню android додатку для навчальних матеріалів

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Програмне забезпечення для створення android додатку для навчальних матеріалів використовується таке, що є у вільному розповсюдженні, тому витрати на придбання такого забезпечення відсутні. Це мова програмування Java/Kotlin та програмні продукти із бібліотек із відкритим кодом Compose.

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12,} \quad (4.5)$$

Таблиця 4.7 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців.	Амортизаційні відрахування, грн
ЕОМ	11000	2	2	916,67
Приміщення	100000	20	2	833,33
Всього				1750

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) розраховують за формулою:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.6)$$

де W_{yi} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), $Ц_е = 4{,}62$ [25];

Квпі — коефіцієнт, що враховує використання потужності, Квпі <1; обираємо

η

$$В_е = \sum_{i=1}^{1} \frac{0{,}1 \cdot 352 \cdot 4{,}62 \cdot 0{,}7}{0{,}8} = 142{,}30 \text{ (грн.)}$$

Проведені розрахунки необхідно звести до таблиці 4.8.

Таблиця 4.8 — Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ЕОМ	0,1	352	142,30
Всього			142,30

4.3.9 Службові відрядження

Під час розробки android додатку для навчальних матеріалів відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, магістрів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та

приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень, не плануються.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» не плануються, так як у цьому не має потреби.

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$I_{в.к.} = 6000 \cdot \frac{50\%}{100\%} = 3000 \text{ (грн.)},$$

$$I_{в.в.} = 27272 \cdot \frac{50\%}{100\%} = 13636 \text{ (грн.)},$$

$$I_в = I_{в.к.} + I_{в.в.} = 16636 \text{ (грн.)}.$$

4.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Беремо норму нарахування 100%.

$$В_{нзв.в.} = 6000 \cdot \frac{100\%}{100\%} = 6000\ (грн.),$$

$$В_{нзв.в.} = 27272 \cdot \frac{100\%}{100\%} = 27272\ (грн.),$$

$$В_{нзв} = В_{нзв.к.} + В_{нзв.в.} = 33272\ (грн.).$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.9)$$

У нашому випадку:

$$К_в = 0, В_{спец} = 0, В_{прг} = 0, В_{св} = 0, В_{сп} = 0, \text{тому отримаємо:}$$

$$В_{заг} = 33272 + 3327 + 10647 + 66{,}78 + 1750 + 142{,}30 + 16636 + 332 = 66173{,}08\ (грн).$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.10)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи; обираємо η = 0,5.

$$ЗВ = \frac{66173{,}08}{0{,}5} = 132346{,}16\ (грн).$$

4.4 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Розробка чи суттєве вдосконалення android додатку для навчальних матеріалів для використання масовим спо-живачем (студентами).

Для всіх наведених випадків можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комер-ціалізації науково-технічної розробки [30], розраховується за формулою:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N \cdot Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \qquad (4.11)$$

де $\pm \Delta Ц_o$— зміна основного якісного показника від впровадження результа-тів науково-технічної розробки в аналізованому році;

N — основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний якісний показник, який визначає ціну реалізації нової на-уково-технічної розробки в аналізованому році, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Ц_б$ — основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

ΔN — зміна основного кількісного показника від впровадження результа-тів науково-технічної розробки в аналізованому році;

λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість становить 20%, а коефіцієнт λ=0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту (

п ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ=18%.

с В результаті впровадження результатів наукових розробок

л

у

г

поліпшується якість програмного забезпечення, що дозволяє подорожчати за його впровадження, а кількість потенційних користувачів ресурсу збільшиться — у перший рік — на 210 одиниць, на другий рік — ще на 350 одиниць, на третій рік — ще 450 штук.

Ми прогнозуємо щорічний приріст чистого прибутку компанії від впровадження результатів наукових розробок щодо вихідного стану. Збільшення чистого прибутку підприємства $\Delta П_i$ за перший рік складе:

$$\Delta П_1 = [1100 \cdot 0 + (3500 + 1100) \cdot 210] \cdot 0,8333 \cdot 0,3 \cdot \left(1 - \frac{18\%}{100\%}\right) = 198\,022,08 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ на другий рік складе:

$$\Delta П_2 = [1100 \cdot 0 + (3500 + 1100) \cdot (210 + 350)] \cdot 0,8333 \cdot 0,3 \cdot \left(1 - \frac{18\%}{100\%}\right) = 528\,058,88 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ на третій рік складе:

$$\Delta П_3 = [1100 \cdot 0 + (3500 + 1100) \cdot (210 + 350 + 450)] \cdot 0,8333 \cdot 0,3 \cdot \left(1 - \frac{18\%}{100\%}\right) = 952\,391,90 \text{ (грн)}.$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.12)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15. Обираємо τ = 0,1;

t— період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{198\,022{,}08}{(1+0{,}1)^1} + \frac{528\,058{,}88}{(1+0{,}1)^2} + \frac{952\,391{,}90}{(1+0{,}1)^3}$$

$$= 180\,020{,}07 + 436\,412{,}30 + 715\,546{,}13 = 1\,331\,978{,}5\ (\text{грн.})$$

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ, \qquad (4.13)$$

де $k_{інв}$— коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим. Обираємо $k_{інв}$ = 2;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 132346{,}16 = 264\,692{,}32(\text{грн.})$$

Тоді абсолютний економічний ефект $E_{абс}$або чистий приведений дохіддля потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \qquad (4.14)$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV — теперішня вартість початкових інвестицій, грн.

$$E_{абс} = 1\,331\,978{,}5 - 264\,692{,}32 = 1\,067\,286{,}18\ (грн.)$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.15)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$Т_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[3]{1 + \frac{1\,067\,286{,}18}{264\,692{,}32}} - 1 = 0{,}71$$

Далі визначають бар'єрну ставку дисконтування τ_{min}, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть .

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$визначається за формулою:

$$\tau_{мін} = d + f, \qquad (4.16)$$

де d— середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = 0,9...0,12. Обираємо $d = 0,11$;

f — показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f=0,05...0,5, але може бути і значно вищою. Обираємо $f = 0,2$;

$$\tau_{мін} = d + f = 0,11 + 0,2 = 0,3\%$$

Величина $E_в > \tau_{мін}$, отже інвестор може бути зацікавлений у фінансуванні цього дослідження.

Далі розраховуємо період окупності інвестицій $T_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \qquad (4.17)$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0,71} = 1,41 \text{ року}$$

Оскільки $T_{ок} = 1,41$ року тоді розвиток доречний.

ВИСНОВКИ

На основі проведеного аналізу можливих архітектур медіа плеєру було проведено модифікацію програвача Android exo, з метою виконання поставлених перед додатком задачі а саме: безпечно завантажувати та відтворювати мультимедійні файли локально та за допомогою потокового мультимедіа на основі HTTP. Також була створена функція створення файлів HLS для подальшого відтворення. Показано, що для цього мобільного додатка було б доцільно використовувати Compose для зберігання пам’яті пристрою. Compose забезпечує сучасний підхід до розробки додатків, що дозволяє ефективно розподіляти обов’язки в коді.

Створено мобільний додаток на базі Android і створено правильну навігаційну карту для програми. Розроблено інтерфейс користувача та додано модифікований плеєр як модуль. Для вирішення цієї проблеми були узгоджені та застосовані такі технології верстки, як Compose, Room, Exo player.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Google Play Hits 1 Million Apps [Електронний ресурс] — Режим доступу до ресурсу: http://mashable.com/2013/07/24/googleplay—1—million/

2. Android App Stats [Електронний ресурс] — Режим доступу до ресурсу: http://www.androlib.com/appstats.aspx

3. Google: 3 Billion Android Apps Installed; Downloads Up 50 Percent From Last Quarter [Електронний ресурс] — Режим доступу до ресурсу: http://techcrunch. com/2011/04/14/google—3—billion—android—appsinstalled—up—50—percent—from—last—quarter

4. Smartphone OS Market Share, Q1 2015 [Електронний ресурс] — Режим доступу до ресурсу: http://www.idc.com/prodserv/smartphone—os—market—share.jsp 16.06.2015.

5. Tools Overview [Електронний ресурс] — Режим доступу до ресурсу: http://developer.android.com/tools/help/index.html

6. AIDE— IDE for Android Java C++ [Електронний ресурс] — Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.aide.ui

7. Java Editor [Електронний ресурс] — Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=air.JavaEditor

8. JavaIDEdroid [Електронний ресурс] — Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=ch.tanapro.JavaIDEdroid

9. The Professional Android IDE [Електронний ресурс] — Режим доступу до ресурсу: http://www.jetbrains.com/idea/features/android.html

10 NBAndroid [Електронний ресурс] — Режим доступу до ресурсу: http://plugins.netbeans.org/plugin/19545/nbandroid

11. Android Studio [Електронний ресурс] — Режим доступу до ресурсу: http://developer.android.com/sdk/index.html

12. Backup & restore Android apps using adb [Електронний ресурс] — Режим доступу до ресурсу: http://jonwestfall.com/2009/08/backup—restore—android—apps—using—adb/

13. SDK Tools [Електронний ресурс] — Режим доступу до ресурсу: http://developer.android.com/tools/sdk/tools—notes.html

14. ДОСЛІДЖЕННЯ ЛОКАЛЬНИХ БАЗ ДАНИХ ДЛЯ ANDROID ПРИСТРОЇВ. РОЗРОБКА МЕТОДУ ОБРОБКИ ДАНИХ.. [Текст]В.В.Козубський // Молодь у науці: дослідження, проблеми, перспективи (МН—2021) Тез. доп.—Виноградник, 2021.—1.[Електронний ресурс] Режим доступу https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/13157/11054

15. Android — Invoke JNI Based Methods (Bridging C/C++ And Java) [Електронний ресурс] — Режим доступу до ресурсу: https://davanum.wordpress.com/2007/12/09/android—invoke—jni—based—methodsbridging—cc—and—java

16. Native C applications for Android [Електронний ресурс] — Режим доступу до ресурсу: http://benno.id.au/blog/2007/11/13/android—native—apps

17. Android NDK [Електронний ресурс] — Режим доступу до ресурсу: https://developer.android.com/tools/sdk/ndk/index.html

18. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ РОЗПІЗНАВАННЯ ОБРАЗІВ НА ОСНОВІ НЕЙРОМЕРЕЖІ. РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ. [Текст] Ю.Р.Левицька // XLVII Науково—технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2019): Тез. доп. — Вінниця, 2019. — 1. Режим доступу https:// conferences.vntu.edu.ua/index.php/all—fitki/all—fitki—2020/paper/view/9470/7736

19. Edwards G. J. Face recognition using active appearance models/ G. J. Edwards, T. F. Cootes, C. J. Taylor // Computer Vision . Volume 1407 of the series Lecture Notes in Computer Science, 2006 — pp 595 — 599.

20. Визильтер Ю. В. Обработка и анализ изображения в задачах машинного зрения. / Ю.В.Визильтер, С. Ю. Желтов — М.: Физматкнига, 2010. — 672 с.

21. Розробка мобільних додатків. [Електронний ресурс]. — Режим доступу: http://kiev.itstep.org/razrabotka-mobilnyh-prilozhenij-pod-android

22. Методологія розробки мультимедійних додатків. [Електронний ресурс]. —

Режим доступу: http://android.mobile-review.com/articles/22580

23. Класифікація мобільних додатків. [Електронний ресурс]. — Режим доступу:
http://voroninstudio.eu/uk/service/razrabotka-mobilnih-prilozheniy.

24. Технології розробки мобільних додатків [Електронний ресурс]. — Режим

доступу: http://lektsii.net/2-50017.html

25. Android Studio. [Електронний ресурс]. — Режим доступу:
https://uk.wikipedia.org/wiki/Android_Studio

26. Шмидт Г.А. Практическое введение в Android Studio — Оренбург, 2014. 458с.

27. Попов Е.Г. Теоретический курс по Java — Москва, Триумф, 2016. — 523с.

28. Ашок К.С. Mastering Firebase for Android Development — Лондон, 2018.

29. Лаура Томсон, Люк Веллинг. Архітектура Android додатків, 2003. — 872 с.

30.Activity. [Електронний ресурс]. — Режим доступу:
http://metanit.com/java/android/2.1.php

31. Мельник Д., Петренко К. Алгоритми роботи додатків, 2014 г.- 378 с.

32. Layout. [Електронний ресурс]. — Режим доступу:
http://developer.alexanderklimov.ru/android/theory/layout.php

33. Гизберт Д. UX/UI дизайн - Санкт-Петербург, НТ Пресс, 2015 г. - 320 с.

34.Чедвік Діксон, Роберт Ройс. Тестування мобільних додатків, 2017. — 127 с.

35.Чаффер Д., Шведберг К. Тестирование мобильных приложений - Москва, Символ-Плюс, 2010 г. - 448 с.

36.Емулятори Android. [Електронний ресурс]. — Режим доступу:
http://appstips.ru/articles/emulyatory-android.html

37.Гарнага В. А. Модель операції порівняння при порозрядному аналогоцифровому перетворенні з прогресуючим набором тривалостей тактів 70 урівноваження / О. Д. Азаров, О. О. Решетнік, В. А. Гарнага // Інформаційні технології та комп’ютерна інженерія. — 2008. — № 3 (13). — С. 5—12.

38. Гарнага В. А. Модель операції порівняння при порозрядному аналогоцифровому перетворенні з прогресуючим набором тривалостей тактів

урівноваження / О. Д. Азаров, О. О. Решетнік, В. А. Гарнага // Інформаційні технології та комп’ютерна інженерія. — 2008. — № 3 (13). — С. 5—12.

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____проф., д.т.н. О. Д. Азаров

«___» _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Хмарно-стрімінговий медіа плеєр для лекцій та навчальних матеріалів

Частина 2. Архітектура додатку та медіа плеєру»

08-23.КМКР.028.00.000 ПЗ

Науковий керівник: д.т.н., професор

_____ Ткаченко О.М.

Магістрант групи 1КІ-20м

_____ Козубський В.В.

Вінниця 2021 р.

1 Підстава для виконання магістерської кваліфікаційної роботи (КМКР)

1.1 Актуальність даного дослідження визначається необхідністю створення прогнозування попиту на основі попередніх даних. За допомогою даного дослідження ми зможемо ефективніше відвідувати лекцію в наслідок чого будемо отримувати більше знань.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення КМКР

2.1 Мета магістерської роботи полягає у розробці android до.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані

Вихідні дані для виконання КМКР — ключі для доступу до бази даних, дані доступних товарів, секретні ключі для доступу до Heroku.

4 Вимоги до виконання КМКР

— оглянути сучасний стан розвитку мобільних додатків на ОС Android;

— проаналізувати галузі застосування;

— виявити галузі застосування мобільних додатків;

— розробити макет навігаційного графу додатку;

— організувати коректну взаємодію між макетом та програмним забезпеченням;

— Розробити сучасний макет користувацького інтерфейсу.

торгівлі.

5 Етапи КМКР та очікувані результати

Етапи МКР та очікуванні результати наведено в таблиці А.1

6 Матеріали

Матеріали, що подаються до захисту КМКР — пояснювальна записка КМКР, ілюстративні та графічні матеріали, протокол попереднього захисту КМКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи

складання державних екзаменів, анотації до КМКР українською та іноземною мовами, довідка про відповідність оформлення КМКР діючим вимогам.

Таблиця А.1 — Етапи КМКР та очікуванні результати

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)		Примітка
		початок	кінець	
	Огляд і аналіз джерел інформації	6.09.2021	11.09.21	
	Розробка технічного завдання	29.09.2021	3.10.21	
	Огляд та аналіз методів та засобів побудови сучасних макетів у системах Android	21.10.2021	28.10.21	
	Розробка кресленого та програмного графу макету	27.10.2021	10.11.21	
	Реалізація програмного додатку системи Android	24.11.2021	29.11.21	
	Підготовка до презентації і захист роботи	25.11.2021	11.12.21	

7 Порядок контролю виконання та захисту КМКР

Виконання етапів розрахункової та графічної документації КМКР контролюється науковим керівником згідно зі встановленими термінами. Захист КМКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення КМКР

Вимоги викладені в ДСТУ 3008:2015 та положення ВНТУ про КМКР - 2021.

Технічне завдання до виконання отримав _____ Козубський В.В.

ДОДАТОК Б

Лістинг програми

```
private const val EMAIL_VALIDATION_REGEX = "^(.+)@(.+)\$"
class EmailState :
   TextFieldState(validator = ::isEmailValid, errorFor = ::emailValidationError)
private fun emailValidationError(email: String): String {
   return "Invalid email: $email"}
private fun isEmailValid(email: String): Boolean {
   return Pattern.matches(EMAIL_VALIDATION_REGEX, email)}
class PasswordState :
   TextFieldState(validator = ::isPasswordValid, errorFor =
::passwordValidationError)
class ConfirmPasswordState(private val passwordState: PasswordState) :
TextFieldState() {
   override val isValid
      get() = passwordAndConfirmationValid(passwordState.text, text)
   override fun getError(): String? {
      return if (showErrors()) {
         passwordConfirmationError()
      } else {
         null}}}
private fun passwordAndConfirmationValid(password: String, confirmedPassword:
String): Boolean {
   return isPasswordValid(password) && password == confirmedPassword
}
private fun isPasswordValid(password: String): Boolean {
   return password.length > 3}
@Suppress("UNUSED_PARAMETER")
private fun passwordValidationError(password: String): String {
   return "Invalid password"}
private fun passwordConfirmationError(): String {
   return "Passwords don't match"}
class SignInFragment : Fragment() {
   private val viewModel: SignInViewModel by viewModels {
SignInViewModelFactory() }
   override fun onCreateView(
      inflater: LayoutInflater,
      container: ViewGroup?,
      savedInstanceState: Bundle?
   ): View? {
      viewModel.navigateTo.observe(viewLifecycleOwner) { navigateToEvent —>
         navigateToEvent.getContentIfNotHandled()?.let { navigateTo —>
```

```
            navigate(navigateTo, Screen.SignIn)
        }}
      return ComposeView(requireContext()).apply {
        // In order for savedState to work, the same ID needs to be used for all
instances.
        id = R.id.sign_in_fragment
        layoutParams = ViewGroup.LayoutParams(
          ViewGroup.LayoutParams.MATCH_PARENT,
          ViewGroup.LayoutParams.MATCH_PARENT
        )
        setContent {
          JetsurveyTheme {
            SignIn(
              onNavigationEvent = { event —>
                when (event) {
                  is SignInEvent.SignIn —> {
                    viewModel.signIn(event.email, event.password)}
                  SignInEvent.SignUp —> {
                    viewModel.signUp()}
                  SignInEvent.SignInAsGuest —> {
                    viewModel.signInAsGuest()}
                  SignInEvent.NavigateBack —> {
                    activity?.onBackPressedDispatcher?.onBackPressed()
                  }
                }
              }
            )
          }
        }
      }
    }}
sealed class SignInEvent {
  data class SignIn(val email: String, val password: String) : SignInEvent()
  object SignUp : SignInEvent()
  object SignInAsGuest : SignInEvent()
  object NavigateBack : SignInEvent()
@OptIn(ExperimentalMaterialApi::class)
@Composable
fun SignIn(onNavigationEvent: (SignInEvent) —> Unit) {
  val snackbarHostState = remember { SnackbarHostState() }
  val scope = rememberCoroutineScope()
  val snackbarErrorText = stringResource(id = R.string.feature_not_available)
  val snackbarActionLabel = stringResource(id = R.string.dismiss)
  Scaffold(
```

```
        topBar = {
            SignInSignUpTopAppBar(
                topAppBarText = stringResource(id = R.string.sign_in),
                onBackPressed = { onNavigationEvent(SignInEvent.NavigateBack) }
            )
        },
        content = {
            SignInSignUpScreen(
                modifier = Modifier.supportWideScreen(),
                onSignedInAsGuest = { onNavigationEvent(SignInEvent.SignInAsGuest) }
            ) {
                Column(modifier = Modifier.fillMaxWidth()) {
                    SignInContent(
                        onSignInSubmitted = { email, password —>
                            onNavigationEvent(SignInEvent.SignIn(email, password))
                        }
                    )
                    Spacer(modifier = Modifier.height(16.dp))
                    TextButton(
                        onClick = {
                            scope.launch {
                                snackbarHostState.showSnackbar(
                                    message = snackbarErrorText,
                                    actionLabel = snackbarActionLabel
                                )
                            }
                        },
                        modifier = Modifier.fillMaxWidth()
                    ) {
                        Text(text = stringResource(id = R.string.forgot_password))
                    }
                }
            }
        }
    )
    Box(modifier = Modifier.fillMaxSize()) {
        ErrorSnackbar(
            snackbarHostState = snackbarHostState,
            onDismiss = { snackbarHostState.currentSnackbarData?.dismiss() },
            modifier = Modifier.align(Alignment.BottomCenter)
        )
    }
}
```

```
@Composable
fun SignInContent(
    onSignInSubmitted: (email: String, password: String) —> Unit,
) {
    Column(modifier = Modifier.fillMaxWidth()) {
        val focusRequester = remember { FocusRequester() }
        val emailState = remember { EmailState() }
        Email(emailState, onImeAction = { focusRequester.requestFocus() })
        Spacer(modifier = Modifier.height(16.dp))
        val passwordState = remember { PasswordState() }
        Password(
            label = stringResource(id = R.string.password),
            passwordState = passwordState,
            modifier = Modifier.focusRequester(focusRequester),
            onImeAction = { onSignInSubmitted(emailState.text, passwordState.text) }
        )
        Spacer(modifier = Modifier.height(16.dp))
        Button(
            onClick = { onSignInSubmitted(emailState.text, passwordState.text)},
            modifier = Modifier
                .fillMaxWidth()
                .padding(vertical = 16.dp),
            enabled = emailState.isValid && passwordState.isValid
        ) {
            Text(
                text = stringResource(id = R.string.sign_in)
            )
        }
    }
}
@OptIn(ExperimentalMaterialApi::class)
@Composable
fun ErrorSnackbar(
    snackbarHostState: SnackbarHostState,
    modifier: Modifier = Modifier,
    onDismiss: () —> Unit = { }
) {
    SnackbarHost(
        hostState = snackbarHostState,
        snackbar = { data —>
            Snackbar(
                modifier = Modifier.padding(16.dp),
                content = {
                    Text(
```

```
                    text = data.message,
                    style = MaterialTheme.typography.body2
                )
            },
            action = {
                data.actionLabel?.let {
                    TextButton(onClick = onDismiss) {
                        Text(
                            text = stringResource(id = R.string.dismiss),
                            color = MaterialTheme.colors.snackbarAction
                        )
                    }
                }
            }
        )
    },
    modifier = modifier
        .fillMaxWidth()
        .wrapContentHeight(Alignment.Bottom)
)}
@Preview(name = "Sign in light theme")
@Composable
fun SignInPreview() {
    JetsurveyTheme {
        SignIn {}
    }}
@Preview(name = "Sign in dark theme")
@Composable
fun SignInPreviewDark() {
    JetsurveyTheme(darkTheme = true) {
        SignIn {}
    }}
@Composable
fun SignInSignUpScreen(
    onSignedInAsGuest: () —> Unit,
    modifier: Modifier = Modifier,
    content: @Composable() () —> Unit
) {
    LazyColumn(modifier = modifier) {
        item {
            Spacer(modifier = Modifier.height(44.dp))
            Box(
                modifier = Modifier
                    .fillMaxWidth()
```

```
                .padding(horizontal = 20.dp)
            ) {
                content()
            }
            Spacer(modifier = Modifier.height(16.dp))
            OrSignInAsGuest(
                onSignedInAsGuest = onSignedInAsGuest,
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(horizontal = 20.dp)
            )
        }
    }}
@Composable
fun SignInSignUpTopAppBar(topAppBarText: String, onBackPressed: () —> Unit)
{
    TopAppBar(
        title = {
            Text(
                text = topAppBarText,
                textAlign = TextAlign.Center,
                modifier = Modifier
                    .fillMaxSize()
                    .wrapContentSize(Alignment.Center)
            )
        },
        navigationIcon = {
            IconButton(onClick = onBackPressed) {
                Icon(
                    imageVector = Icons.Filled.ChevronLeft,
                    contentDescription = stringResource(id = R.string.back)
                )
            }
        },
        // We need to balance the navigation icon, so we add a spacer.
        actions = {
            Spacer(modifier = Modifier.width(68.dp))
        },
        backgroundColor = MaterialTheme.colors.surface,
        elevation = 0.dp
    )}
@Composable
fun Email(
    emailState: TextFieldState = remember { EmailState() },
```

```
    imeAction: ImeAction = ImeAction.Next,
    onImeAction: () —> Unit = {}
) {
    OutlinedTextField(
        value = emailState.text,
        onValueChange = {
            emailState.text = it
        },
        label = {
            CompositionLocalProvider(LocalContentAlpha                provides
ContentAlpha.medium) {
                Text(
                    text = stringResource(id = R.string.email),
                    style = MaterialTheme.typography.body2
                )
            }
        },
        modifier = Modifier
            .fillMaxWidth()
            .onFocusChanged { focusState —>
                emailState.onFocusChange(focusState.isFocused)
                if (!focusState.isFocused) {
                    emailState.enableShowErrors()
                }
            },
        textStyle = MaterialTheme.typography.body2,
        isError = emailState.showErrors(),
        keyboardOptions = KeyboardOptions.Default.copy(imeAction = imeAction),
        keyboardActions = KeyboardActions(
            onDone = {
                onImeAction()
            }
        )
    )

    emailState.getError()?.let { error —> TextFieldError(textError = error) }
}
@Composable
fun Password(
    label: String,
    passwordState: TextFieldState,
    modifier: Modifier = Modifier,
    imeAction: ImeAction = ImeAction.Done,
    onImeAction: () —> Unit = {}
```

```
) {
    val showPassword = remember { mutableStateOf(false) }
    OutlinedTextField(
        value = passwordState.text,
        onValueChange = {
            passwordState.text = it
            passwordState.enableShowErrors()
        },
        modifier = modifier
            .fillMaxWidth()
            .onFocusChanged { focusState —>
                passwordState.onFocusChange(focusState.isFocused)
                if (!focusState.isFocused) {
                    passwordState.enableShowErrors()
                }
            },
        textStyle = MaterialTheme.typography.body2,
        label = {
            CompositionLocalProvider(LocalContentAlpha                provides
ContentAlpha.medium) {
                Text(
                    text = label,
                    style = MaterialTheme.typography.body2
                )
            }
        },
        trailingIcon = {
            if (showPassword.value) {
                IconButton(onClick = { showPassword.value = false }) {
                    Icon(
                        imageVector = Icons.Filled.Visibility,
                        contentDescription = stringResource(id = R.string.hide_password)
                    )
                }
            } else {
                IconButton(onClick = { showPassword.value = true }) {
                    Icon(
                        imageVector = Icons.Filled.VisibilityOff,
                        contentDescription = stringResource(id = R.string.show_password)
                    )
                }
            }
        },
        visualTransformation = if (showPassword.value) {
```

```
            VisualTransformation.None
        } else {
            PasswordVisualTransformation()
        },
        isError = passwordState.showErrors(),
        keyboardOptions = KeyboardOptions.Default.copy(imeAction = imeAction),
        keyboardActions = KeyboardActions(
            onDone = {
                onImeAction()
            }
        )
    )

    passwordState.getError()?.let { error —> TextFieldError(textError = error) }
}
@Composable
fun TextFieldError(textError: String) {
    Row(modifier = Modifier.fillMaxWidth()) {
        Spacer(modifier = Modifier.width(16.dp))
        Text(
            text = textError,
            modifier = Modifier.fillMaxWidth(),
            style = LocalTextStyle.current.copy(color = MaterialTheme.colors.error)
        )
    }
}
@Composable
fun OrSignInAsGuest(
    onSignedInAsGuest: () —> Unit,
    modifier: Modifier = Modifier
) {
    Column(
        modifier = modifier,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Surface {
            CompositionLocalProvider(LocalContentAlpha                provides
ContentAlpha.medium) {
                Text(
                    text = stringResource(id = R.string.or),
                    style = MaterialTheme.typography.subtitle2,
                    modifier = Modifier.paddingFromBaseline(top = 25.dp)
                )
            }
```

```
        }
        OutlinedButton(
            onClick = onSignedInAsGuest,
            modifier = Modifier
                .fillMaxWidth()
                .padding(top = 20.dp, bottom = 24.dp)
        ) {
            Text(text = stringResource(id = R.string.sign_in_guest))
        }
    }
}
@Preview
@Composable
fun SignInSignUpScreenPreview() {
    SignInSignUpScreen(
        onSignedInAsGuest = {},
        content = {}
    )
}
class SignInViewModel(private val userRepository: UserRepository) :
ViewModel() {
    private val _navigateTo = MutableLiveData<Event<Screen>>()
    val navigateTo: LiveData<Event<Screen>>
        get() = _navigateTo
    fun signIn(email: String, password: String) {
        userRepository.signIn(email, password)
        _navigateTo.value = Event(Survey)
    }
    fun signInAsGuest() {
        userRepository.signInAsGuest()
        _navigateTo.value = Event(Survey)
    }
    fun signUp() {
        _navigateTo.value = Event(SignUp)
    }
}
class SignInViewModelFactory : ViewModelProvider.Factory {
    @Suppress("UNCHECKED_CAST")
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(SignInViewModel::class.java)) {
            return SignInViewModel(UserRepository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
```

```
}
class SignUpFragment : Fragment() {
  private val viewModel: SignUpViewModel by viewModels { SignUpViewModelFactory() }

  override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
  ): View {
    viewModel.navigateTo.observe(viewLifecycleOwner) { navigateToEvent —>
      navigateToEvent.getContentIfNotHandled()?.let { navigateTo —>
        navigate(navigateTo, Screen.SignUp)
      }
    }

    return ComposeView(requireContext()).apply {
      // In order for savedState to work, the same ID needs to be used for all instances.
      id = R.id.sign_up_fragment

      layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
      )
      setContent {
        JetsurveyTheme {
          SignUp(
            onNavigationEvent = { event —>
              when (event) {
                is SignUpEvent.SignUp —> {
                  viewModel.signUp(event.email, event.password)
                }
                SignUpEvent.SignIn —> {
                  viewModel.signIn()
                }
                SignUpEvent.SignInAsGuest —> {
                  viewModel.signInAsGuest()
                }
                SignUpEvent.NavigateBack —> {
                  activity?.onBackPressedDispatcher?.onBackPressed()
                }
              }
            }
```

```
                )
            }
        }
    }
}
sealed class SignUpEvent {
    object SignIn : SignUpEvent()
    data class SignUp(val email: String, val password: String) : SignUpEvent()
    object SignInAsGuest : SignUpEvent()
    object NavigateBack : SignUpEvent()
}
@Composable
fun SignUp(onNavigationEvent: (SignUpEvent) —> Unit) {
    Scaffold(
        topBar = {
            SignInSignUpTopAppBar(
                topAppBarText = stringResource(id = R.string.create_account),
                onBackPressed = { onNavigationEvent(SignUpEvent.NavigateBack) }
            )
        },
        content = {
            SignInSignUpScreen(
                onSignedInAsGuest                    =                    {
onNavigationEvent(SignUpEvent.SignInAsGuest) },
                modifier = Modifier.supportWideScreen()
            ) {
                Column {
                    SignUpContent(
                        onSignUpSubmitted = { email, password —>
                            onNavigationEvent(SignUpEvent.SignUp(email, password))
                        }
                    )
                }
            }
        }
    )
}
@Composable
fun SignUpContent(
    onSignUpSubmitted: (email: String, password: String) —> Unit,
) {
    Column(modifier = Modifier.fillMaxWidth()) {
        val passwordFocusRequest = remember { FocusRequester() }
```

```
        val confirmationPasswordFocusRequest = remember { FocusRequester() }
        val emailState = remember { EmailState() }
        Email(emailState, onImeAction = { passwordFocusRequest.requestFocus() })
        Spacer(modifier = Modifier.height(16.dp))
        val passwordState = remember { PasswordState() }
        Password(
            label = stringResource(id = R.string.password),
            passwordState = passwordState,
            imeAction = ImeAction.Next,
            onImeAction = { confirmationPasswordFocusRequest.requestFocus() },
            modifier = Modifier.focusRequester(passwordFocusRequest)
        )
        Spacer(modifier = Modifier.height(16.dp))
        val confirmPasswordState = remember { ConfirmPasswordState(passwordState = passwordState) }
        Password(
            label = stringResource(id = R.string.confirm_password),
            passwordState = confirmPasswordState,
            onImeAction = { onSignUpSubmitted(emailState.text, passwordState.text) },
            modifier = Modifier.focusRequester(confirmationPasswordFocusRequest)
        )
        Spacer(modifier = Modifier.height(16.dp))
        CompositionLocalProvider(LocalContentAlpha provides ContentAlpha.medium) {
            Text(
                text = stringResource(id = R.string.terms_and_conditions),
                style = MaterialTheme.typography.caption
            )
        }
        Spacer(modifier = Modifier.height(16.dp))
        Button(
            onClick = { onSignUpSubmitted(emailState.text, passwordState.text) },
            modifier = Modifier.fillMaxWidth(),
            enabled = emailState.isValid &&
                passwordState.isValid && confirmPasswordState.isValid
        ) {
            Text(text = stringResource(id = R.string.create_account))
        }
    }
}
@Preview(widthDp = 1024)
@Composable
fun SignUpPreview() {
```

```
    JetsurveyTheme {
        SignUp {}
    }
}
class SignUpViewModel(private val userRepository: UserRepository) :
ViewModel() {
    private val _navigateTo = MutableLiveData<Event<Screen>>()
    val navigateTo: LiveData<Event<Screen>>
        get() = _navigateTo
    fun signUp(email: String, password: String) {
        userRepository.signUp(email, password)
        _navigateTo.value = Event(Survey)
    }
    fun signInAsGuest() {
        userRepository.signInAsGuest()
        _navigateTo.value = Event(Survey)
    }
    fun signIn() {
        _navigateTo.value = Event(SignIn)
    }
}
class SignUpViewModelFactory : ViewModelProvider.Factory {
    @Suppress("UNCHECKED_CAST")
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(SignUpViewModel::class.java)) {
            return SignUpViewModel(UserRepository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
======================
open class TextFieldState(
    private val validator: (String) —> Boolean = { true },
    private val errorFor: (String) —> String = { "" }
) {
    var text: String by mutableStateOf("")
    // was the TextField ever focused
    var isFocusedDirty: Boolean by mutableStateOf(false)
    var isFocused: Boolean by mutableStateOf(false)
    private var displayErrors: Boolean by mutableStateOf(false)

    open val isValid: Boolean
        get() = validator(text)
    fun onFocusChange(focused: Boolean) {
```

```
        isFocused = focused
        if (focused) isFocusedDirty = true
    }
    fun enableShowErrors() {
        // only show errors if the text was at least once focused
        if (isFocusedDirty) {
            displayErrors = true
        }
    }
    fun showErrors() = !isValid && displayErrors
    open fun getError(): String? {
        return if (showErrors()) {
            errorFor(text)
        } else {
            null
        }
    }
}
sealed class User {
    @Immutable
    data class LoggedInUser(val email: String) : User()
    object GuestUser : User()
    object NoUserLoggedIn : User()
}
object UserRepository {
    private var _user: User = User.NoUserLoggedIn
    val user: User
        get() = _user
    @Suppress("UNUSED_PARAMETER")
    fun signIn(email: String, password: String) {
        _user = User.LoggedInUser(email)
    }
    @Suppress("UNUSED_PARAMETER")
    fun signUp(email: String, password: String) {
        _user = User.LoggedInUser(email)
    }
    fun signInAsGuest() {
        _user = User.GuestUser
    }
    fun isKnownUserEmail(email: String): Boolean {
        // if the email contains "sign up" we consider it unknown
        return !email.contains("signup")
    }
}
```

```
enum class SurveyActionType { PICK_DATE, TAKE_PHOTO, SELECT_CONTACT }
sealed class SurveyActionResult {
   data class Date(val date: String) : SurveyActionResult()
   data class Photo(val uri: Uri) : SurveyActionResult()
   data class Contact(val contact: String) : SurveyActionResult()
}
sealed class PossibleAnswer {
   data class SingleChoice(val optionsStringRes: List<Int>) : PossibleAnswer()
   data class SingleChoiceIcon(val optionsStringIconRes: List<Pair<Int, Int>>) : PossibleAnswer()
   data class MultipleChoice(val optionsStringRes: List<Int>) : PossibleAnswer()
   data class MultipleChoiceIcon(val optionsStringIconRes: List<Pair<Int, Int>>) : PossibleAnswer()
   data class Action(
      @StringRes val label: Int,
      val actionType: SurveyActionType
   ) : PossibleAnswer()

   data class Slider(
      val range: ClosedFloatingPointRange<Float>,
      val steps: Int,
      @StringRes val startText: Int,
      @StringRes val endText: Int,
      @StringRes val neutralText: Int,
      val defaultValue: Float = 5.5f
   ) : PossibleAnswer()
}
sealed class Answer<T : PossibleAnswer> {
   object PermissionsDenied : Answer<Nothing>()
   data class SingleChoice(@StringRes val answer: Int) : Answer<PossibleAnswer.SingleChoice>()
   data class MultipleChoice(val answersStringRes: Set<Int>) :
      Answer<PossibleAnswer.MultipleChoice>()

   data class Action(val result: SurveyActionResult) : Answer<PossibleAnswer.Action>()
   data class Slider(val answerValue: Float) : Answer<PossibleAnswer.Slider>()
}
fun Answer.MultipleChoice.withAnswerSelected(
   @StringRes answer: Int,
   selected: Boolean
): Answer.MultipleChoice {
   val newStringRes = answersStringRes.toMutableSet()
```

```
    if (!selected) {
        newStringRes.remove(answer)
    } else {
        newStringRes.add(answer)
    }
    return Answer.MultipleChoice(newStringRes)
}
class SurveyFragment : Fragment() {
    private val viewModel: SurveyViewModel by viewModels {

SurveyViewModelFactory(PhotoUriManager(requireContext().applicationContext
))
    }
    private val takePicture = registerForActivityResult(TakePicture()) { photoSaved
—>
        if (photoSaved) {
            viewModel.onImageSaved()
        }
    }
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        return ComposeView(requireContext()).apply {
            // In order for savedState to work, the same ID needs to be used for all
instances.
            id = R.id.sign_in_fragment
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )
            setContent {
                JetsurveyTheme {
                    viewModel.uiState.observeAsState().value?.let { surveyState —>
                        when (surveyState) {
                            is SurveyState.Questions —> SurveyQuestionsScreen(
                                questions = surveyState,
                                shouldAskPermissions = viewModel.askForPermissions,
                                onAction = { id, action —> handleSurveyAction(id, action) },
                                onDoNotAskForPermissions = {
viewModel.doNotAskForPermissions() },
                                onDonePressed = { viewModel.computeResult(surveyState) },
                                onBackPressed = {
```

```
                        activity?.onBackPressedDispatcher?.onBackPressed()
                    },
                    openSettings = {
                        activity?.startActivity(
                            Intent(
Settings.ACTION_APPLICATION_DETAILS_SETTINGS,
                                Uri.fromParts("package", context.packageName, null)
                            )
                        )
                    }
                )
                is SurveyState.Result —> SurveyResultScreen(
                    result = surveyState,
                    onDonePressed = {
                        activity?.onBackPressedDispatcher?.onBackPressed()
                    }
                )
            }
        }
    }
}
}

    private fun handleSurveyAction(questionId: Int, actionType: SurveyActionType)
{
        when (actionType) {
            SurveyActionType.PICK_DATE —> showDatePicker(questionId)
            SurveyActionType.TAKE_PHOTO —> takeAPhoto()
            SurveyActionType.SELECT_CONTACT —> selectContact(questionId)
        }
    }

    private fun showDatePicker(questionId: Int) {
        val date = viewModel.getCurrentDate(questionId = questionId)
        val picker = MaterialDatePicker.Builder.datePicker()
            .setSelection(date)
            .build()
        activity?.let {
            picker.show(it.supportFragmentManager, picker.toString())
            picker.addOnPositiveButtonClickListener {
                viewModel.onDatePicked(questionId, picker.selection)
            }
```

```
        }
    }

    private fun takeAPhoto() {
        takePicture.launch(viewModel.getUriToSaveImage())
    }

    @Suppress("UNUSED_PARAMETER")
    private fun selectContact(questionId: Int) {
        // TODO: unsupported for now
    }
}
@Composable
fun SurveyQuestionsScreen(
    questions: SurveyState.Questions,
    shouldAskPermissions: Boolean,
    onDoNotAskForPermissions: () —> Unit,
    onAction: (Int, SurveyActionType) —> Unit,
    onDonePressed: () —> Unit,
    onBackPressed: () —> Unit,
    openSettings: () —> Unit
) {
    val questionState = remember(questions.currentQuestionIndex) {
        questions.questionsState[questions.currentQuestionIndex]
    }

    Surface(modifier = Modifier.supportWideScreen()) {
        Scaffold(
            topBar = {
                SurveyTopAppBar(
                    questionIndex = questionState.questionIndex,
                    totalQuestionsCount = questionState.totalQuestionsCount,
                    onBackPressed = onBackPressed
                )
            },
            content = { innerPadding —>
                Question(
                    question = questionState.question,
                    answer = questionState.answer,
                    shouldAskPermissions = shouldAskPermissions,
                    onAnswer = {
                        if (it !is Answer.PermissionsDenied) {
                            questionState.answer = it
                        }
```

```
                    questionState.enableNext = true
                },
                onAction = onAction,
                openSettings = openSettings,
                onDoNotAskForPermissions = onDoNotAskForPermissions,
                modifier = Modifier
                    .fillMaxSize()
                    .padding(innerPadding)
            )
        },
        bottomBar = {
            SurveyBottomBar(
                questionState = questionState,
                onPreviousPressed = { questions.currentQuestionIndex—— },
                onNextPressed = { questions.currentQuestionIndex++ },
                onDonePressed = onDonePressed
            )
        }
    )
  }
}
@Composable
fun SurveyResultScreen(
  result: SurveyState.Result,
  onDonePressed: () —> Unit
) {
  Surface(modifier = Modifier.supportWideScreen()) {
    Scaffold(
      content = { innerPadding —>
        val modifier = Modifier.padding(innerPadding)
        SurveyResult(result = result, modifier = modifier)
      },
      bottomBar = {
        OutlinedButton(
          onClick = { onDonePressed() },
          modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 20.dp, vertical = 24.dp)
        ) {
          Text(text = stringResource(id = R.string.done))
        }
      }
    )
  }
```

```
}
@Composable
private fun SurveyResult(result: SurveyState.Result, modifier: Modifier = Modifier)
{
    LazyColumn(modifier = modifier.fillMaxSize()) {
        item {
            Spacer(modifier = Modifier.height(44.dp))
            Text(
                text = result.surveyResult.library,
                style = MaterialTheme.typography.h3,
                modifier = Modifier.padding(horizontal = 20.dp)
            )
            Text(
                text = stringResource(
                    result.surveyResult.result,
                    result.surveyResult.library
                ),
                style = MaterialTheme.typography.subtitle1,
                modifier = Modifier.padding(20.dp)
            )
            Text(
                text = stringResource(result.surveyResult.description),
                style = MaterialTheme.typography.body1,
                modifier = Modifier.padding(horizontal = 20.dp)
            )
        }
    }
}
@Composable
private fun TopAppBarTitle(
    questionIndex: Int,
    totalQuestionsCount: Int,
    modifier: Modifier = Modifier
) {
    val indexStyle = MaterialTheme.typography.caption.toSpanStyle().copy(
        fontWeight = FontWeight.Bold
    )
    val totalStyle = MaterialTheme.typography.caption.toSpanStyle()
    val text = buildAnnotatedString {
        withStyle(style = indexStyle) {
            append("${questionIndex + 1}")
        }
        withStyle(style = totalStyle) {
            append(stringResource(R.string.question_count, totalQuestionsCount))
```

```
        }
    }
    Text(
        text = text,
        style = MaterialTheme.typography.caption,
        modifier = modifier
    )
}

@Composable
private fun SurveyTopAppBar(
    questionIndex: Int,
    totalQuestionsCount: Int,
    onBackPressed: () —> Unit
) {
    Column(modifier = Modifier.fillMaxWidth()) {
        Box(modifier = Modifier.fillMaxWidth()) {
            TopAppBarTitle(
                questionIndex = questionIndex,
                totalQuestionsCount = totalQuestionsCount,
                modifier = Modifier
                    .padding(vertical = 20.dp)
                    .align(Alignment.Center)
            )

            CompositionLocalProvider(LocalContentAlpha                provides
ContentAlpha.medium) {
                IconButton(
                    onClick = onBackPressed,
                    modifier = Modifier
                        .padding(horizontal = 20.dp, vertical = 20.dp)
                        .fillMaxWidth()
                ) {
                    Icon(
                        Icons.Filled.Close,
                        contentDescription = stringResource(id = R.string.close),
                        modifier = Modifier.align(Alignment.CenterEnd)
                    )
                }
            }
        }
        val animatedProgress by animateFloatAsState(
            targetValue = (questionIndex + 1) / totalQuestionsCount.toFloat(),
            animationSpec = ProgressIndicatorDefaults.ProgressAnimationSpec
```

```
        )
        LinearProgressIndicator(
            progress = animatedProgress,
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 20.dp),
            backgroundColor = MaterialTheme.colors.progressIndicatorBackground
        )
    }
}

@Composable
private fun SurveyBottomBar(
    questionState: QuestionState,
    onPreviousPressed: () —> Unit,
    onNextPressed: () —> Unit,
    onDonePressed: () —> Unit
) {
    Surface(
        elevation = 7.dp,
        modifier = Modifier.fillMaxWidth() // .border(1.dp, MaterialTheme.colors.primary)
    ) {

        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp, vertical = 20.dp)
        ) {
            if (questionState.showPrevious) {
                OutlinedButton(
                    modifier = Modifier
                        .weight(1f)
                        .height(48.dp),
                    onClick = onPreviousPressed
                ) {
                    Text(text = stringResource(id = R.string.previous))
                }
                Spacer(modifier = Modifier.width(16.dp))
            }
            if (questionState.showDone) {
                Button(
                    modifier = Modifier
                        .weight(1f)
```

```
                    .height(48.dp),
                onClick = onDonePressed,
                enabled = questionState.enableNext
            ) {
                Text(text = stringResource(id = R.string.done))
            }
        } else {
            Button(
                modifier = Modifier
                    .weight(1f)
                    .height(48.dp),
                onClick = onNextPressed,
                enabled = questionState.enableNext
            ) {
                Text(text = stringResource(id = R.string.next))
            }
        }
    }
}
@Stable
class QuestionState(
    val question: Question,
    val questionIndex: Int,
    val totalQuestionsCount: Int,
    val showPrevious: Boolean,
    val showDone: Boolean
) {
    var enableNext by mutableStateOf(false)
    var answer by mutableStateOf<Answer<*>?>(null)
}

sealed class SurveyState {
    data class Questions(
        @StringRes val surveyTitle: Int,
        val questionsState: List<QuestionState>
    ) : SurveyState() {
        var currentQuestionIndex by mutableStateOf(0)
    }

    data class Result(
        @StringRes val surveyTitle: Int,
        val surveyResult: SurveyResult
    ) : SurveyState()
```

```
}
@Composable
fun JetchatAppBar(
    modifier: Modifier = Modifier,
    scrollBehavior: TopAppBarScrollBehavior? = null,
    onNavIconPressed: () —> Unit = { },
    title: @Composable () —> Unit,
    actions: @Composable RowScope.() —> Unit = {}
) {
    val backgroundColors = TopAppBarDefaults.centerAlignedTopAppBarColors()
    val backgroundColor = backgroundColors.containerColor(
        scrollFraction = scrollBehavior?.scrollFraction ?: 0f
    ).value
    val foregroundColors = TopAppBarDefaults.centerAlignedTopAppBarColors(
        containerColor = Color.Transparent,
        scrolledContainerColor = Color.Transparent
    )
    Box(modifier = Modifier.background(backgroundColor)) {
        CenterAlignedTopAppBar(
            modifier = modifier,
            actions = actions,
            title = title,
            scrollBehavior = scrollBehavior,
            colors = foregroundColors,
            navigationIcon = {
                JetchatIcon(
                    contentDescription = stringResource(id =
R.string.navigation_drawer_open),
                    modifier = Modifier
                        .size(64.dp)
                        .clickable(onClick = onNavIconPressed)
                        .padding(16.dp)
                )
            }
        )
    }
}
@Preview
@Composable
fun JetchatAppBarPreview() {
    JetchatTheme {
        JetchatAppBar(title = { Text("Preview!") })
    }
}
```

```
@Preview
@Composable
fun JetchatAppBarPreviewDark() {
    JetchatTheme(isDarkTheme = true) {
        JetchatAppBar(title = { Text("Preview!") })
    }
}
@Composable
private fun ProfileItem(text: String, @DrawableRes profilePic: Int?,
onProfileClicked: () —> Unit) {
    Row(
        modifier = Modifier
            .height(56.dp)
            .fillMaxWidth()
            .padding(horizontal = 12.dp)
            .clip(CircleShape)
            .clickable(onClick = onProfileClicked),
        verticalAlignment = CenterVertically
    ) {
        val paddingSizeModifier = Modifier
            .padding(start = 16.dp, top = 16.dp, bottom = 16.dp)
            .size(24.dp)
        if (profilePic != null) {
            Image(
                painter = painterResource(id = profilePic),
                modifier = paddingSizeModifier.then(Modifier.clip(CircleShape)),
                contentScale = ContentScale.Crop,
                contentDescription = null
            )
        } else {
            Spacer(modifier = paddingSizeModifier)
        }
        Text(
            text,
            style = MaterialTheme.typography.bodyMedium,
            color = MaterialTheme.colorScheme.onSurface,
            modifier = Modifier.padding(start = 12.dp)
        )
    }
}
@Composable
fun DividerItem(modifier: Modifier = Modifier) {
    // TODO (M3): No Divider, replace when available
```

```
    Divider(
        modifier = modifier,
        color = MaterialTheme.colorScheme.onSurface.copy(alpha = 0.12f)
    )
}
@Composable
@Preview
fun DrawerPreview() {
    JetchatTheme {
        Surface {
            Column {
                JetchatDrawer({}, {})
            }
        }
    }
}
@Composable
@Preview
fun DrawerPreviewDark() {
    JetchatTheme(isDarkTheme = true) {
        Surface {
            Column {
                JetchatDrawer({}, {})
            }
        }
    }
}
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun JetchatScaffold(
    scaffoldState: ScaffoldState = rememberScaffoldState(),
    onProfileClicked: (String) —> Unit,
    onChatClicked: (String) —> Unit,
    content: @Composable (PaddingValues) —> Unit
) {
    JetchatTheme {
        Scaffold(
            scaffoldState = scaffoldState,
            drawerContent = {
                JetchatDrawer(
                    onProfileClicked = onProfileClicked,
                    onChatClicked = onChatClicked
                )
            },
```

```
            content = content
        )
    }
}
```

ДОДАТОК В

Compose додаток з комбінованим зберіганням данних

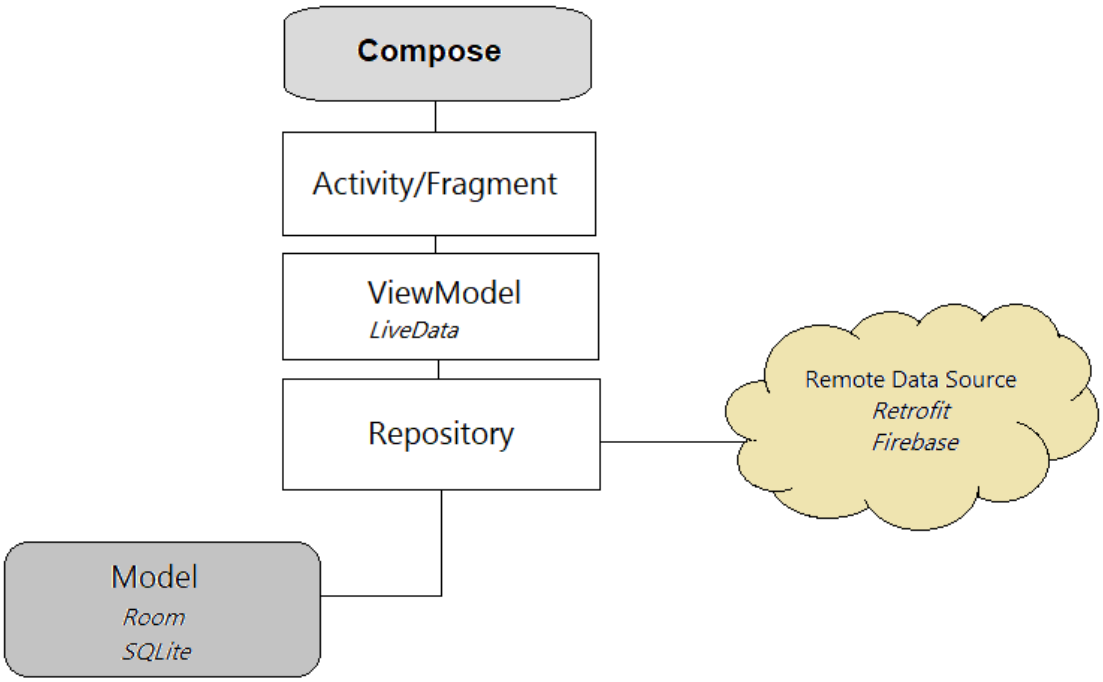

Рисунок В.1 — Compose додаток з комбінованим зберіганням данних

ДОДАТОК Г

Передача даних за допомогою розробленої моделі

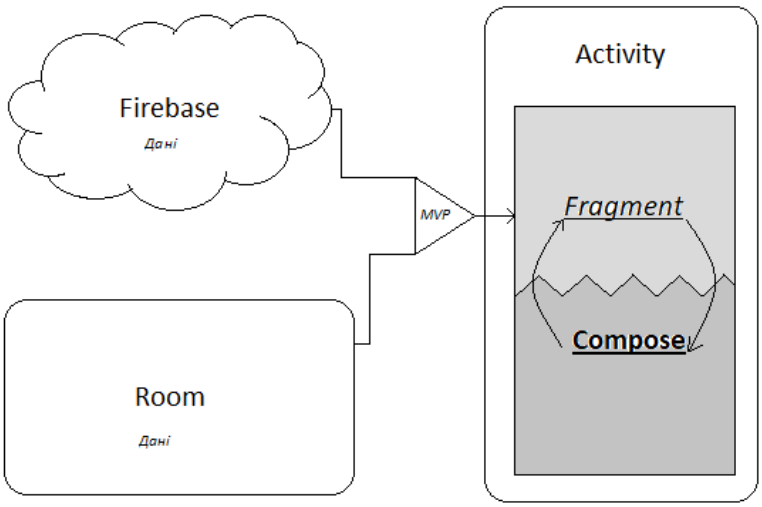

Рисунок Г.1 — Передача даних за допомогою розробленої моделі

ДОДАТОК Д

Блок—схема рівня декодування відео

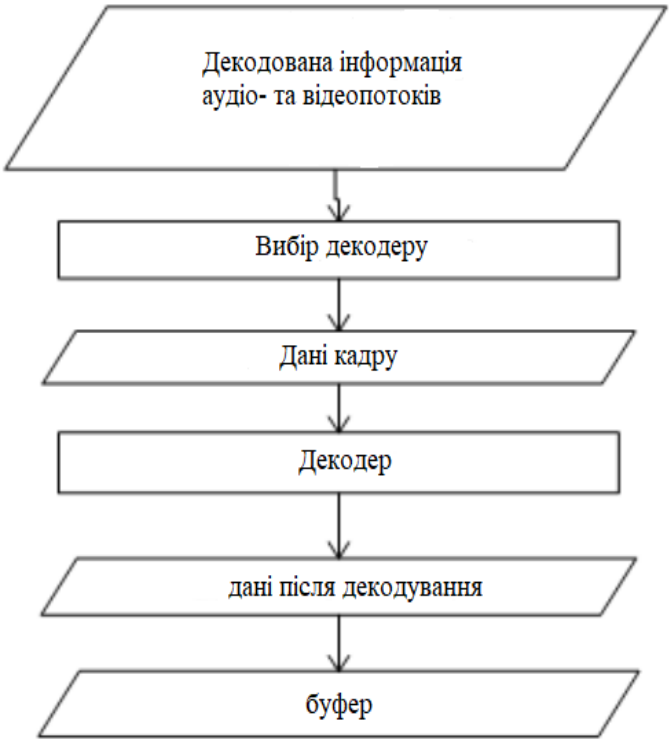

Рисунок Д.1 Блок—схема рівня декодування відео

ДОДАТОК Е

Рузультат роботи додатку

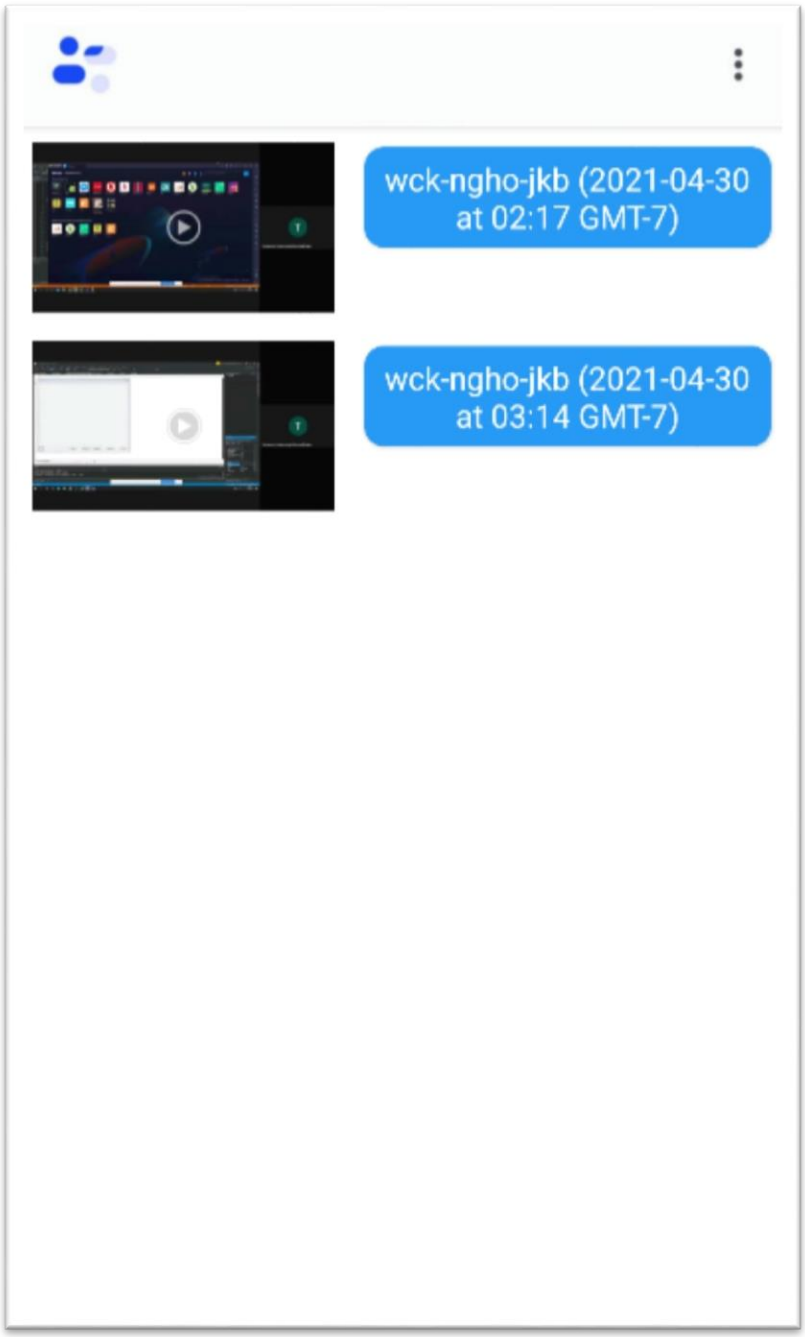

Рисунок Е.1 — Список навчальних матеріалів та лекцій розробленого додатку

Відтворення лекції в портативному режимі за допомогою exoPlayer

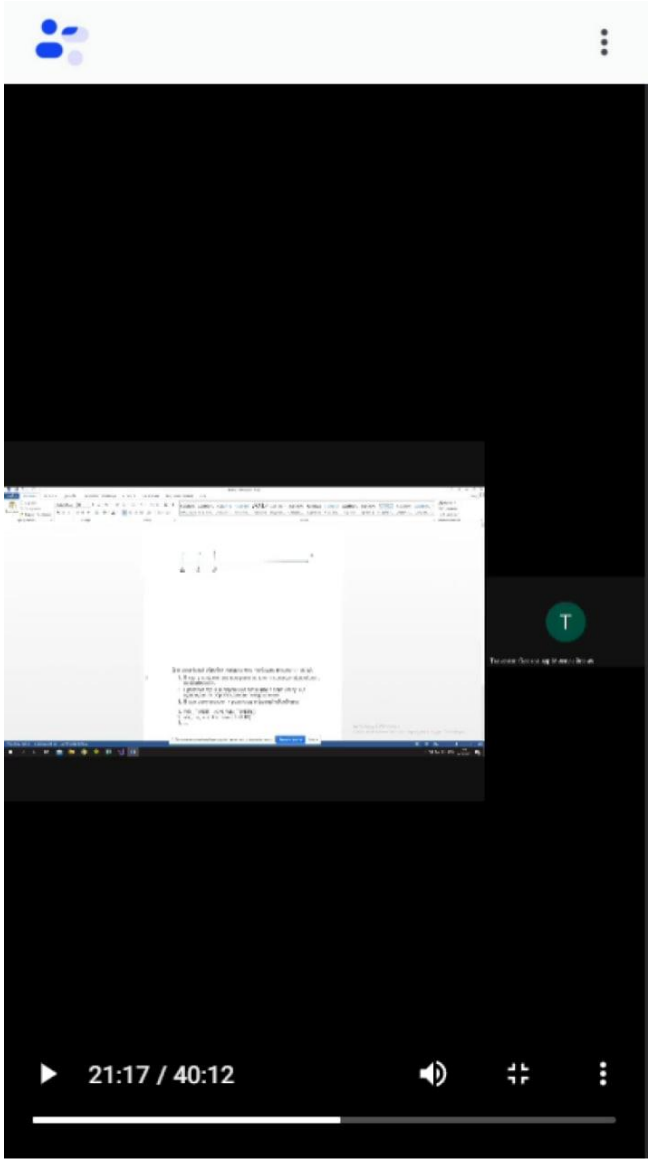

Рисунок Е.2 — Відтворення лекції за допомогою exoPlayer

ДОДАТОК Ж

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Хмарно—стрімінговий медіа плеєр для навчальних матеріалів та лекцій . Частина 2. Архітектура додатку та медіа плеєру

Тип роботи: магістерська кваліфікаційна робота

(кваліфікаційна роботи, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет (інститут), навчальна група)

Науковий керівник к.т.н., доц. Ткаченко О.М.

(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	88%
Тривога/Білі знаки	/	Схожість	12%

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(—на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Козубський В. В.

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

Особа, відповідальна за перевірку_____ Захарченко С.М.

(підпис) (прізвище, ініціали)

Експерт _____

(за потреби) (підпис) (прізвище, ініціали)