Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка мобільного застосунку для відтворення контенту контенту засобами Flutter

Виконав(ла): студент(ка) VI курсу, групи СПм-61
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Городиловський О.В.
(підпис) (прізвище та ініціали)

Керівник _____ Цуприк Г.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ Луцик Н.С.
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)

« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Городиловського Олександра Віталійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного застосунку для відтворення контенту контенту засобами Flutter

Керівник роботи Цуприк Галина Богданівна, канд. техн. наук, доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*27*» *листопада* 2025 року № *4/7-1045*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Flutter, Dart, TMDB API, Firebase, TensorFlow Lite, Android та iOS

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз предметної області, постановка завдання та цілей, пошук акторів та варіантів використання, опис ключових варіантів використання, вибір процесу розробки, проектування архітектури системи, побудова схем бази даних, побудова UML- діаграм класів, вибір мови та середовища розробки, реалізація основних класів та методів, розробка інтерфейсу користувача, тестування програмної системи, види та план тестування, розробка тестових сценаріїв, розгортання програмної системи та системні вимоги, верифікація програмної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Додаток А. Тези конференції; Додаток Б. Диск з роботою

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	28.11.2025	**виконав**
2	Постановка завдання та цілей	30.11.2025	**виконав**
3	Пошук акторів та варіантів використання	01.12.2025	**виконав**
4	Опис ключових варіантів використання	02.12.2025	**виконав**
5	Вибір процесу розробки	03.12.2025	**виконав**
6	Проектування архітектури системи	04.12.2025	**виконав**
7	Побудова схем бази даних	05.12.2025	**виконав**
8	Побудова UML-діаграм класів	06.12.2025	**виконав**
9	Вибір мови та середовища розробки	07.12.2025	**виконав**
10	Реалізація основних класів та методів	08.12.2025	**виконав**
11	Розробка інтерфейсу користувача	09.12.2025	**виконав**
12	Тестування програмної системи	10.12.2025	**виконав**
13	Види та план тестування	11.12.2025	**виконав**
14	Розробка тестових сценаріїв	12.12.2025	**виконав**
15	Розгортання програмної системи та системні вимоги	13.12.2025	**виконав**
16	Верифікація програмної системи	14.12.2025	**виконав**
17	Охорона праці	15.12.2025	**виконав**
18	Фактори, що впливають на функціональний стан	16.12.2025	**виконав**
	користувачів комп’ютерів		
19	Розробка застосунку	18.12.2025	**виконав**
20	Підготовка до захисту	23.12.2025	

Студент _____ Городиловський О. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Цуприк Г.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Городиловський Олександр Віталійович. Розробка мобільного застосунку для відтворення контенту засобами Flutter. Кваліфікаційна робота освітнього ступеня «магістр». Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, група СПм-61, спеціальність 121 «Інженерія програмного забезпечення». Тернопіль, 2025. Сторінок 76, таблиць 1, рисунків 13, додатків 3, бібліографічних посилань 43.

Метою роботи є проєктування та розроблення кросплатформного мобільного застосунку для відтворення мультимедійного контенту з емоційно-адаптивною взаємодією користувача на основі Flutter.

Об’єктом дослідження є процес створення мобільних мультимедійних застосунків із персоналізованою подачею контенту та аналізом емоційного стану користувача.

Предметом дослідження є методи, технології та архітектурні підходи для розробки застосунків на Flutter із використанням хмарних сервісів, TensorFlow Lite та рекомендаційних систем.

Методи дослідження включають аналіз існуючих рішень, проєктування багатошарової архітектури, моделювання бази даних, розробку UML-діаграм, створення та оптимізацію модулів емоційного аналізу, тестування та оцінювання продуктивності.

Робота охоплює створення мобільного застосунку для перегляду відеоконтенту: розробку інтерфейсу, мережеву взаємодію з TMDB API, налаштування Firebase, побудову рекомендаційного модуля та інтеграцію моделі визначення емоцій користувача. Застосунок реалізовано за архітектурою BLoC/Cubit із репозиторною моделлю, що забезпечує модульність, масштабованість та високий рівень тестованості, з акцентом на оптимізацію продуктивності та плавне відтворення контенту.

Ключові слова: Flutter, Dart, мобільний застосунок, мультимедійний контент, рекомендаційна система, аналіз емоцій, TensorFlow Lite, Firebase, TMDB API, інженерія програмного забезпечення.

ABSTRACT

Horodylovskyi Oleksandr Vitaliyovych. Development of a mobile application for content playback using Flutter. Qualification work for the educational degree "Master". Ivan Pulyuy Ternopil National Technical University, Department of Software Engineering, Group SPm-61, Specialty 121 "Software Engineering". Ternopil, 2025. Pages 76, tables 1, figures 13, appendices 3, bibliographic references 43.

The purpose of the work is to design and develop a cross-platform mobile application for multimedia content playback with emotionally adaptive user interaction based on Flutter.

The object of the study is the process of creating mobile multimedia applications with personalized content delivery and analysis of the user's emotional state.

The subject of the study is methods, technologies and architectural approaches for developing applications on Flutter using cloud services, TensorFlow Lite and recommendation systems.

Research methods include analysis of existing solutions, design of multi-layer architecture, database modeling, development of UML diagrams, creation and optimization of emotional analysis modules, testing and performance evaluation.

The work covers the creation of a mobile application for viewing video content: interface development, network interaction with TMDB API, Firebase configuration, building a recommendation module and integration of a user emotion recognition model. The application is implemented using the BLoC/Cubit architecture with a repository model that provides modularity, scalability and a high level of testability, with an emphasis on performance optimization and smooth content playback.

Keywords: Flutter, Dart, mobile application, multimedia content, recommendation system, emotion analysis, TensorFlow Lite, Firebase, TMDB API, software engineering.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) — програмний інтерфейс взаємодії між програмними компонентами або зовнішніми сервісами;

BLoC (Business Logic Component) — архітектурний патерн управління станом, що забезпечує відокремлення бізнес-логіки від інтерфейсу користувача;

Cubit — спрощена реалізація патерну BLoC для управління станами у Flutter-застосунках;

CI/CD (Continuous Integration / Continuous Deployment) — підхід до безперервної інтеграції та розгортання програмного забезпечення;

Cloud Firestore — документно-орієнтована хмарна база даних платформи Firebase;

Dart — об’єктно-орієнтована мова програмування, що використовується у фреймворку Flutter;

DI (Dependency Injection) — механізм інверсії керування для зменшення зв’язаності компонентів системи;

ER-діаграма (Entity–Relationship Diagram) — діаграма «сутність–зв’язок», що використовується для моделювання структури бази даних;

Firebase Authentication — сервіс автентифікації користувачів платформи Firebase;

Flutter — кросплатформний фреймворк для розробки мобільних застосунків із єдиною кодовою базою;

HTTP (HyperText Transfer Protocol) — протокол передачі даних у мережі Інтернет;

JSON (JavaScript Object Notation) — текстовий формат обміну даними між клієнтом і сервером;

MVVM (Model–View–ViewModel) — архітектурний шаблон побудови програмних застосунків;

REST (Representational State Transfer) — архітектурний стиль побудови веб-сервісів;

SDK (Software Development Kit) — набір інструментів для розробки програмного забезпечення;

TFLite (TensorFlow Lite) — полегшена платформа машинного навчання для мобільних та вбудованих пристроїв;

Face++ — хмарний сервіс комп’ютерного зору та аналізу облич, що надає API для розпізнавання облич, визначення емоційного стану, віку, статі та інших атрибутів на основі зображень і відеопотоків;

TMDB (The Movie Database) — зовнішній сервіс та база даних метаданих мультимедійного контенту;

UI (User Interface) — користувацький інтерфейс;

UX (User Experience) — користувацький досвід взаємодії із застосунком;

UML (Unified Modeling Language) — уніфікована мова моделювання для опису програмних систем;

Widget — базовий компонент інтерфейсу користувача у Flutter.

ЗМІСТ

ВСТУП

Стрімкий розвиток мобільних технологій, широке впровадження високошвидкісних мереж 4G/5G та зростання попиту на мультимедійні сервіси обумовлюють актуальність створення високопродуктивних мобільних застосунків для відтворення відеоконтенту. Сучасні користувачі очікують не лише стабільної роботи плеєра, а й персоналізованого досвіду, що враховує індивідуальні вподобання, контекст використання і поведінкові особливості. У цих умовах важливого значення набувають системи, здатні адаптувати рекомендації та сценарії взаємодії з огляду на емоційний стан користувача. Саме тому тема розробки мобільного застосунку з емоційно-адаптивною поведінкою є актуальною як у науковому, так і у практичному вимірі.

Об’єктом дослідження є процес відтворення мультимедійного контенту в мобільних застосунках.

Предметом дослідження є методи, моделі та програмні засоби реалізації емоційно-адаптивного мобільного застосунку на основі Flutter та TensorFlow Lite.

Метою роботи є розроблення мобільного застосунку для відтворення мультимедійного контенту із впровадженням механізму подієвого емоційного аналізу, що дозволяє формувати персоналізовані рекомендації та змінювати сценарії взаємодії користувача з системою.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести аналіз предметної області мультимедійних мобільних застосунків і визначити вимоги до системи.

2. Розробити архітектуру застосунку з урахуванням багатошарової структури, патернів BLoC/Cubit, MVVM та принципів модульності.

3. Побудувати модель подієвого визначення емоцій користувача на основі TensorFlow Lite і методів комп’ютерного зору, використовуючи камеру мобільного пристрою для аналізу міміки та виразу обличчя. Для спрощеної реалізації можна застосовувати сторонні сервіси, такі як Face++, які дозволяють швидко інтегрувати детекцію емоцій без потреби тренування власної моделі.

4. Розробити механізм адаптації рекомендацій та сценарію відтворення на основі емоційного профілю користувача.

5. Реалізувати мобільний застосунок засобами Flutter з підтримкою авторизації, перегляду контенту, рекомендаційних алгоритмів та емоційного аналізу.

6. Виконати тестування та оцінювання ефективності застосунку, порівнявши його з традиційними рекомендаційними системами.

Наукова новизна роботи полягає у розробленні та застосуванні підходу подієвого емоційного аналізу в мобільних мультимедійних застосунках. На відміну від постійного моніторингу, запропонована модель активується лише у ключові моменти взаємодії, що знижує витрати ресурсів і забезпечує збалансовану адаптацію рекомендацій на основі короткострокового емоційного контексту.

Практичне значення полягає у створенні робочого мобільного застосунку, реалізованого на Flutter, що може бути використаний як основа для подальших досліджень та впровадження емоційно-адаптивних мультимедійних сервісів. Розроблені архітектурні рішення, модулі аналізу емоцій та адаптивні механізми можуть бути інтегровані у реальні комерційні продукти.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У даному розділі виконано аналіз вимог до програмної системи мобільного мультимедійного застосунку з емоційно-адаптивною логікою. Розглянуто предметну область мобільних мультимедійних систем, сучасні підходи до персоналізації контенту та роль емоційного контексту у формуванні користувацького досвіду. Сформульовано мету та завдання розробки, визначено основних акторів системи й варіанти їхньої взаємодії. На основі UML-підходів описано ключові сценарії використання, що дозволяє чітко окреслити функціональні вимоги та створити основу для подальшого проєктування архітектури мобільного застосунку.

1.1 Аналіз предметної області

Предметна область мобільних мультимедійних систем характеризується високою динамічністю розвитку, багатошаровою архітектурною складністю та значною кількістю технологічних викликів, пов’язаних із передаванням, обробкою та персоналізованим поданням відеоконтенту користувачу [1]. За останні роки мобільні платформи стали домінуючим середовищем споживання цифрових медіа, що зумовлено доступністю широкосмугових мереж зв’язку (4G/5G) [2], зростанням обчислювальних можливостей мобільних процесорів та розвитком оптимізованих фреймворків для побудови інтерфейсів і взаємодії з апаратними можливостями пристроїв [3].

У межах предметної області мультимедійного відтворення виділяють ключові компоненти [4]:

- джерела мультимедійних даних (локальні або хмарні сховища, потокові платформи, агрегатори контенту),
- механізми потокового відтворення (адаптивний стримінг, буферизація, динамічний вибір якості),

- системи персоналізації та рекомендацій,
- інтерактивні інтерфейси користувача,
- адаптивні елементи взаємодії, зокрема емоційно-орієнтовані алгоритми.

Традиційні мобільні застосунки для відтворення відео зосереджені на стандартних функціональних можливостях: запуск, пауза, перемотування, зміна якості [5]. Однак сучасні вимоги користувачів виходять за ці рамки, оскільки передбачають глибшу персоналізацію та адаптивність мультимедійного середовища. Користувач очікує, що система не тільки відтворюватиме відео, а й враховуватиме його індивідуальні вподобання, контекст використання, історію переглядів та навіть емоційний стан.

Протягом останнього десятиліття значних досліджень зазнали методи комп'ютерного зору та глибинного навчання, орієнтовані на розпізнавання емоцій за зображенням обличчя або відеопотоком. Найбільш продуктивними виявились згорткові нейронні мережі (CNN), гібридні архітектури CNN-RNN, а з 2019 року — трансформерні моделі, здатні ефективно аналізувати просторово-часові залежності та працювати з неструктурованими даними високої розмірності. Розвиток цих технологій створив підґрунтя для появи аффективних мультимедійних систем, які здатні адаптувати поведінку залежно від емоційних характеристик користувача.

Разом із тим традиційний підхід до безперервного емоційного моніторингу є надзвичайно ресурсомістким і часто неприйнятним у мобільних системах через обмеження енергоспоживання та обчислювальної потужності. У цьому контексті особливого значення набуває концепція подієвого емоційного аналізу (event-driven emotion recognition), що передбачає непостійний, а дискретний виклик системи розпізнавання у визначені моменти взаємодії: при запуску застосунку, перед початком перегляду або після його завершення. Така стратегія дозволяє зберігати баланс між адаптивністю та ефективністю використання ресурсів, не перевантажуючи мобільний пристрій і водночас забезпечуючи релевантні персоналізовані рекомендації.

У межах рекомендаційних систем емоційні ознаки розглядаються як додатковий шар семантичної інформації, що може підвищити точність і релевантність рекомендацій. Вони доповнюють традиційні контентні та поведінкові фактори: жанрові вподобання, історію переглядів, темп взаємодії, час доби, частоту повторного перегляду тощо. Включення таких ознак дозволяє формувати «емоційно напрямлені» рекомендації, що покращують користувацький досвід та підвищують залученість.

Технологічний розвиток мобільних платформ також сприяв активному впровадженню кросплатформових інструментів, серед яких домінуючу позицію займає Flutter [3]. Він забезпечує не лише високу продуктивність рендерингу та нативність UX, а й широкий спектр мультимедійних можливостей: інтеграцію відеоплеєрів, апаратне прискорення, роботу з потоками даних та використання JNI/Swift-бріджів для виклику низькорівневих API. У поєднанні з TensorFlow Lite, що підтримує оптимізовані моделі для мобільних пристроїв, Flutter стає придатною платформою для реалізації систем емоційно-адаптивного відтворення контенту.

Отже, предметна область поєднує у собі широкий спектр компонентів: від мережевих технологій і потокового відтворення до машинного навчання та нейромережевих алгоритмів. Така міждисциплінарність формує новий клас мобільних мультимедійних застосунків, що здатні забезпечувати персоналізований досвід взаємодії та адаптувати свою поведінку відповідно до емоційних та поведінкових характеристик користувача.

1.2 Постановка завдання та цілей

Метою даної роботи є розроблення сучасного мобільного застосунку для відтворення мультимедійного контенту із впровадженням механізму подієвої емоційної адаптації, реалізованого засобами фреймворку Flutter [3]. Особливістю даного підходу є можливість адаптувати рекомендації, сценарії перегляду та

особливості взаємодії залежно від емоційного стану користувача, який визначається не постійно, а у визначені ключові моменти використання застосунку. На відміну від класичних рекомендаційних систем, що залежать переважно від історії переглядів та поведінкових патернів, запропонована система враховує короткостроковий емоційний контекст, що дозволяє створити індивідуалізовану взаємодію з користувачем і підвищити рівень залученості під час споживання мультимедійного контенту [6].

В основі розроблюваної системи лежить ідея про те, що ефективна персоналізація може бути досягнута за умови врахування змінного емоційного стану користувача. Важливим аспектом є те, що модель не повинна працювати в режимі постійного моніторингу — це не лише економить ресурси пристрою, але й знижує ризики перевантаження процесора та акумулятора, що особливо критично для мобільних платформ. Замість цього застосунок реагує на ключові події (вхід у застосунок, початок відтворення, пауза, завершення перегляду), що уможливлює баланс між якістю персоналізації та оптимальним використанням обчислювальних потужностей [2].

Для досягнення поставленої мети необхідно виконати комплекс взаємопов’язаних завдань.

1. Аналіз вимог до мобільного застосунку.

На першому етапі необхідно провести детальний аналіз функціональних і нефункціональних вимог, які визначають можливості системи та обмеження платформи. Серед ключових функціональних вимог — можливість відтворення відеоконтенту, формування рекомендацій, інтеграція модуля емоційного аналізу, базове управління профілем користувача [5]. До нефункціональних вимог належать стабільність роботи, швидкість відгуку інтерфейсу, адаптивність дизайну, безпека даних та мінімальне навантаження на систему [1]. Окрему увагу потрібно приділити сценаріям користування: перегляду контенту, вибору жанрів, взаємодії з рекомендаційним блоком, навігації між екранами.

2. Формування архітектури системи.

Другим важливим завданням є побудова архітектури, що поєднує модулі мультимедійного відтворення, модуль емоційного аналізу та підсистему адаптивної логіки. Архітектура повинна бути модульною, розширюваною та орієнтованою на підтримку незалежних компонентів. Важливим аспектом є розроблення механізму взаємодії моделі емоційного аналізу з різними частинами застосунку, а також визначення способу зберігання емоційних даних та забезпечення їх подальшої актуалізації [7].

3. Розроблення моделі подієвого розпізнавання емоцій.

Необхідно створити модель, здатну працювати на мобільних пристроях, використовуючи TensorFlow Lite для забезпечення оптимальної продуктивності [8]. Модель повинна розпізнавати емоції у режимі епізодичного виклику, що мінімізує постійне навантаження на систему та відповідає концепції енергоефективності. Емоційний аналіз у ключові моменти дозволяє будувати більш точний короткостроковий емоційний профіль, що використовується для адаптації рекомендацій.

4. Створення механізму адаптації мультимедійного відтворення.

Механізм адаптації включає кілька основних складових:

– формування стартових рекомендацій при запуску застосунку з урахуванням поточного емоційного стану;

– адаптивну зміну сценаріїв перегляду (наприклад, вибір релаксаційного контенту при виявленні підвищеного стресу);

– корекцію рекомендацій після перегляду з урахуванням емоційної реакції користувача;

– оновлення та підтримку емоційного профілю, який змінюється залежно від динаміки використання застосунку [6].

5. Реалізація мобільного застосунку засобами Flutter.

Необхідно створити повнофункціональний застосунок, який відповідатиме сучасним принципам UI/UX, забезпечить кросплатформність, плавність

анімацій, інтеграцію з TensorFlow Lite, а також збереження локальних даних користувача. Платформа Flutter дозволяє забезпечити високу продуктивність і швидкість розробки, що робить її оптимальним вибором для поставленого завдання [3].

6. Оцінка ефективності та експериментальна перевірка.

Фінальним етапом є порівняння роботи системи із застосуванням подієвого емоційного аналізу зі стандартними рекомендаційними механізмами. До ключових параметрів оцінки входять: швидкість взаємодії з інтерфейсом, ресурсне навантаження, зміна релевантності рекомендацій, вплив адаптації на загальне користувацьке враження [9]. Дослідження має продемонструвати переваги впровадження емоційного контексту у мультимедійні застосунки.

Ці завдання дають змогу створити інтелектуальний мобільний застосунок мультимедійного відтворення, що використовує емоційний контекст користувача для персоналізації контенту без потреби у постійному моніторингу та надмірному споживанні ресурсів.

1.3 Пошук акторів та варіантів використання

Проєктування функціональних вимог до мобільного застосунку починається з визначення акторів — зовнішніх сутностей, що взаємодіють із системою, та опису варіантів їхньої взаємодії. Методика пошуку акторів та побудови варіантів використання відповідає класичним підходам UML та сучасним принципам моделювання мобільних систем [10].

У межах розроблюваного застосунку виділено такі основні актори:

1. Користувач — основний актор, який переглядає мультимедійний контент, отримує рекомендації, взаємодіє з інтерфейсом відтворення та може надавати дозвіл на використання камери для подієвого емоційного аналізу.

2. Підсистема емоційного аналізу — внутрішній програмний агент, що отримує зображення з камери та визначає емоційний стан користувача у

визначені моменти взаємодії. Вона виступає актором щодо модулю адаптивної логіки. Робота підсистеми спирається на принципи афективних обчислень та машинного навчання [9].

3. Підсистема рекомендацій — програмний компонент, який формує персоналізовані списки контенту з урахуванням емоційного профілю. Взаємодіє з модулем адаптації та модулем перегляду контенту [6, 11].

4. Зовнішній контент-сервер — джерело метаданих, потокового відео та статичної інформації. Застосунок взаємодіє з ним через API для отримання контенту [5].

5. Середовище Flutter/TensorFlow Lite (технічний актор) — забезпечує виконання моделі машинного навчання для класифікації емоцій та реалізацію інтерфейсних компонентів [3,8].

На основі визначених акторів сформовано ключові сценарії взаємодії користувача та підсистем. Основні варіанти використання включають:

1. Запуск мобільного застосунку

Користувач відкриває застосунок, після чого система ініціалізує базові модулі, відображає головний екран та викликає одноразовий емоційний аналіз для формування стартових рекомендацій (за дозволом користувача).

2. Отримання персоналізованих рекомендацій

Підсистема рекомендацій формує добірку відеоконтенту на основі емоційного профілю та даних попередніх взаємодій.

3. Перегляд відеоконтенту

Користувач обирає відео для перегляду, після чого запускається медіаплеєр із можливістю керування (пауза, перемотування, зміна режиму).

4. Подієвий емоційний аналіз

Застосунок може одноразово аналізувати емоційний стан у ключових моментах:

– під час входу в застосунок,

– на початку відтворення,

– під час паузи,

– після завершення перегляду.

Отриманий результат передається в модуль адаптивної логіки [9].

5. Адаптація сценарію відтворення

На основі визначеної емоції застосунок обирає відповідний сценарій:

– спокійний перегляд,

– динамічний режим»,

– режим концентрації.

Система може змінювати структуру рекомендацій або спосіб подання контенту без втручання користувача.

6. Оновлення емоційного профілю після перегляду

Після завершення відео система проводить постпереглядовий аналіз (за наявності дозволу) і оновлює профіль користувача для подальших рекомендацій.

7. Перегляд бібліотеки, пошук та категоризація контенту

Користувач може переглядати категорії, шукати контент, створювати списки.

8. Налаштування конфіденційності та доступу до камери

Користувач керує дозволами, зокрема можливістю використання камери для емоційного аналізу.

1.4 Опис ключових варіантів використання

Для визначення функціональних можливостей розроблюваного мобільного застосунку було виконано аналіз акторів системи та сценаріїв їх взаємодії. Основні та допоміжні актори наведені в Таблиці 1.1, а взаємозв’язки між ними та варіантами використання представлені у вигляді UML-діаграми варіантів використання (див. рисунок 1.1). У процесі моделювання використовувалися практики UML, рекомендовані у посібнику IBM щодо проєктування use-case діаграм [10].

Таблиця 1.1 – Акторів системи

№	Основний актор	Найменування	Формулювання
1	Так	Користувач	Основний актор, який взаємодіє з мобільним застосунком: переглядає бібліотеку, запускає контент, керує відтворенням, надає/скасовує дозволи.
2	Ні	Система застосунку	Програмна система, що виконує ініціалізацію, відтворення контенту, обробку запитів користувача та керує внутрішніми модулями.
3	Ні	Модуль емоційного аналізу	Підсистема, яка виконує аналіз емоційного стану користувача (подієво або при запуску).
4	Ні	Модуль рекомендацій	Підсистема, яка формує персоналізовані добірки контенту на основі історії переглядів та емоційного профілю.
5	Ні	Модуль адаптації контенту	Підсистема, яка підлаштовує відтворення або рекомендації залежно від визначених емоцій користувача.
6	Ні	Система дозволів ОС	Інфраструктурний актор — керує доступом до камери та іншими правами, що запитує застосунок.

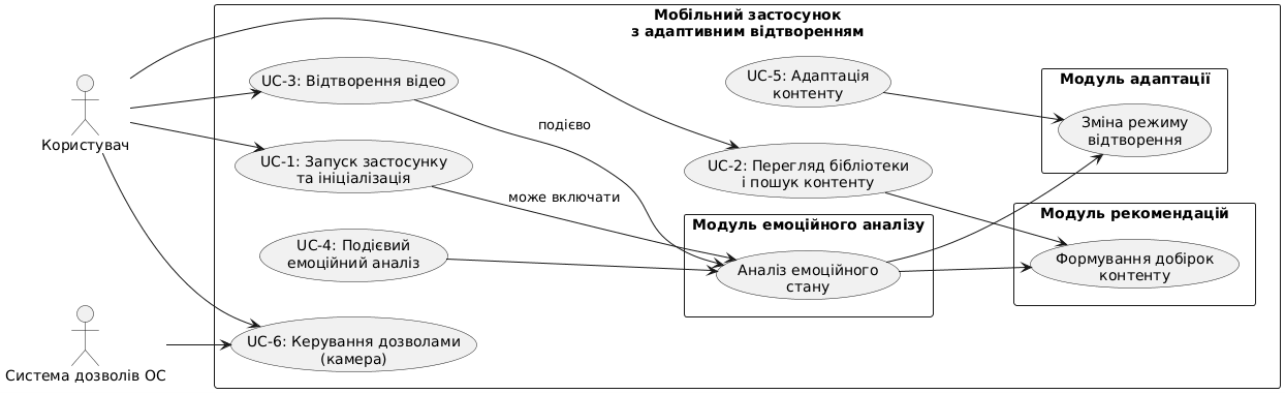

Рисунок 1.1 – UML діаграма варіантів використання

Модель варіантів використання описує набір основних функцій системи, пов’язаних із переглядом контенту, подієвим емоційним аналізом та адаптацією рекомендацій. Підхід до опису інтерфейсної взаємодії та реактивної поведінки застосунку враховує сучасні підходи до побудови мобільних систем, орієнтованих на події та контекст користувача [11]. Нижче подано детальний опис ключових use-case системи.

UC-1. Запуск застосунку та початкова ініціалізація

Передумови: застосунок встановлений на мобільному пристрої; користувач може надати дозвіл на доступ до камери (опціонально).

Основний сценарій:

1. Користувач запускає застосунок.

2. Система ініціалізує графічний інтерфейс, плеєр та фонові модулі.

3. За потреби виконується подієвий емоційний аналіз для формування стартового профілю [8].

4. На головному екрані формується початкова добірка контенту.

Наслідки: система готова до роботи, сформовано персоналізований стартовий екран.

UC-2. Перегляд бібліотеки та підбір контенту

Передумови: застосунок запущено.

Основний сценарій:

1. Користувач переглядає категорії, жанри, історію або рекомендації.

2. Система відображає вибрані списки контенту.

3. Користувач може використовувати пошук або фільтри.

4. Система повертає список відповідних відео [6].

5. Користувач обирає матеріал для перегляду.

Наслідки: обрано контент, готовий до відтворення.

UC-3. Запуск відтворення відео

Передумови: користувач обрав контент.

Основний сценарій:

1. Користувач натискає кнопку «Відтворити».

2. Система, за необхідності, проводить подієвий емоційний аналіз.

3. Відео починає відтворюватися [4].

4. Користувач керує відтворенням (пауза, перемотування тощо).

Наслідки: контент відтворюється, система реагує на подальші події.

UC-4. Подієвий емоційний аналіз

Передумови: користувач надав дозвіл на доступ до камери.

Основний сценарій:

1. Система ініціює зчитування зображення у визначених моментах: запуск застосунку, запуск відео, пауза, завершення.

2. Модуль аналізу розпізнає емоцію [8].

3. Результат передається до модуля адаптації та рекомендацій.

Наслідки: визначено актуальний емоційний стан користувача.

UC-5. Адаптація відтворення та рекомендацій

Передумови: отримано результат емоційного аналізу.

Основний сценарій:

1. Система визначає режим адаптації (спокійний, динамічний, нейтральний).

2. Коригуються параметри відтворення або рекомендації [11].

3. Після завершення перегляду оновлюється емоційний профіль користувача.

Наслідки: контент підлаштовано до індивідуального емоційного стану.

UC-6. Керування дозволами (камера та конфіденційність)

Передумови: користувач знаходиться в налаштуваннях або при першому запуску.

Основний сценарій:

1. Користувач відкриває розділ дозволів.

2. Система показує статус доступу до камери.

3. Користувач надає або відкликає дозвіл.

4. У разі відмови застосунок працює у базовому режимі без емоційного аналізу.

Наслідки: забезпечено коректну роботу механізмів конфіденційності та адаптації [12].

Отже, узагальнена модель варіантів використання демонструє логічну структуру взаємодії користувача із системою та підсистемами аналізу,

рекомендацій і адаптації. Це дозволяє визначити функціональні вимоги та структурувати подальший процес проєктування мобільного застосунку [7].

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

У даному розділі розглянуто процес проєктування та практичної реалізації програмної системи мобільного мультимедійного застосунку з емоційно-адаптивною логікою. Описано вибір методології розробки, обґрунтовано архітектурні рішення та технологічний стек, а також наведено структуру основних компонентів системи. Особливу увагу приділено проєктуванню багатошарової архітектури, моделюванню бази даних, побудові UML-діаграм і реалізації ключових класів та інтерфейсу користувача, що створює основу для стабільної, масштабованої та персоналізованої роботи мобільного застосунку.

2.1 Вибір процесу розробки

Вибір процесу розробки програмного забезпечення є ключовим етапом проєктування, оскільки саме він визначає структуру, послідовність та інтенсивність виконання робіт, а також рівень адаптивності системи до змінних вимог. Для проєкту зі створення мобільного застосунку з функціями мультимедійного відтворення та інтегрованим модулем емоційно-адаптивної взаємодії особливо важливим є забезпечення гнучкості, можливості регулярної модифікації компонентів та швидкої перевірки працездатності нових функціональних рішень. Подібні вимоги характерні для сучасних мобільних систем, що поєднують мультимедійні та ML-орієнтовані компоненти, про що свідчать аналітичні огляди Google щодо мобільної архітектури та UX-практик [7].

Традиційні каскадні (Waterfall) підходи, попри їхню формальну прозорість та чіткість регламентів, демонструють обмежену придатність у системах, де наявний дослідницький або експериментальний компонент. У даному проєкті таким компонентом є модуль визначення емоційного стану користувача, який потребує багаторазового циклічного вдосконалення, перевірки гіпотез, адаптації

до апаратних характеристик мобільного пристрою та уточнення критеріїв точності. Подібні задачі описуються в дослідженнях із афективних систем та подієвого емоційного аналізу [9], де наголошується на необхідності ітеративних перевірок і багатокрокових експериментів. У таких умовах лінійні моделі не забезпечують достатнього рівня гнучкості, а затримки між етапами розробки можуть призвести до неконтрольованих відхилень у реалізації цільових функцій.

Саме тому як базову методологію обрано гнучкі підходи (Agile Software Development), які ґрунтуються на принципах ітеративності, адаптивності й орієнтації на кінцеву користувацьку цінність. У межах Agile-парадигми найбільш відповідним для даного проєкту було визначено фреймворк Scrum, що забезпечує високу швидкість реакції на зміни зовнішніх і внутрішніх вимог, а також дозволяє поєднати етапи проєктування, реалізації та валідації у коротких і повторюваних інтервалах — спринтах. Ефективність Scrum у проєктах із підвищеною невизначеністю підтверджується сучасними дослідженнями управління розробкою ПЗ.

До ключових причин застосування Scrum у цьому проєкті належать:

- Ітеративність та інкрементальність. Система створюється шляхом поступового розширення функціональних модулів: від базових можливостей відтворення відеоконтенту [4] до впровадження рекомендаційних механізмів і складних компонентів емоційного аналізу.
- Гнучке управління вимогами. Функціональність модуля емоційного розпізнавання залежить від якості даних, ефективності моделі та результатів проміжних експериментів, тому можливість корекції вимог у реальному часі є критичною [8].
- Прозорість і контрольованість процесу. Регулярні Scrum-зустрічі (stand-up, sprint review, sprint retrospective) дають змогу забезпечити постійний моніторинг стану робіт та мінімізувати ризики відхилення від цілей.

- Пріоритизація функціональності. Складні модулі машинного навчання реалізуються лише після стабілізації ключових компонентів, таких як плеєр, навігація, взаємодія з API та обробка кешованих даних [7].
- Орієнтація на користувача. Основний критерій успішності кожного інкременту — покращення зручності користування та підвищення персоналізації, що відповідає загальній концепції емоційно-адаптивної мультимедійної системи.

Окрім Scrum, у тих частинах системи, де превалює невизначеність або необхідність апробації технічних рішень (зокрема вибір архітектури моделі розпізнавання емоцій, оптимізація TensorFlow Lite під апаратні ресурси або дослідження затримок декодування відеопотоку), застосовано підхід експериментального прототипування. Він дає змогу швидко оцінити ефективність потенційних рішень без необхідності їх повноцінної інтеграції в основну систему, зменшуючи ризики і витрати [13].

У підсумку процес створення мобільного застосунку сформовано як комбіновану методологічну модель, що включає:

- Agile / Scrum як загальний процесний фреймворк управління розробкою.
- Ітеративно-інкрементальний підхід для поетапного нарощування функціональності.
- Прототипування (rapid prototyping) у сегментах, пов’язаних із машинним навчанням, оптимізацією продуктивності та дослідженнями взаємодії.

Такий комплексний підхід забезпечує оптимальний баланс між структурованістю виконання робіт та можливістю оперативної адаптації системи до нових технічних, наукових та користувацьких викликів, що є визначальним чинником у проєктах, що включають високотехнологічні модулі та інтелектуальні механізми поведінки [14].

2.2 Проєктування архітектури системи

Проєктування архітектури програмної системи є одним з ключових етапів життєвого циклу розробки, оскільки воно визначає структуру взаємодії компонентів, їх відповідальність, рівень масштабованості та можливість подальшого розширення функціональності. Для мобільного застосунку, орієнтованого на мультимедійне відтворення та інтегровану емоційно-адаптивну поведінку, архітектурна модель повинна забезпечувати високий рівень продуктивності, модульності, ізоляції компонентів, а також підтримку інтеграції інтелектуальних обчислень без зниження швидкодії інтерфейсу [14].

З урахуванням цих вимог обрано багатошарову (multilayered) архітектурну модель, що поєднує принципи розділення відповідальності (Separation of Concerns), інкапсуляції, слабкого зв’язування та компонентної ізольованості. Архітектура доповнена патернами, типовими для розробки на Flutter, зокрема MVVM, BLoC/Cubit, Repository та Dependency Injection [3, 7]. Структурна схема архітектури подана на рисунку 2.1 та відображає логічну декомпозицію системи на основні рівні взаємодії.

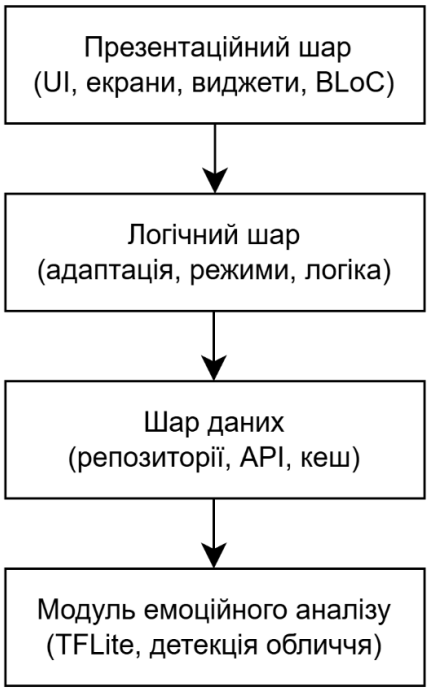

Рисунок 2.1 – Структурна схема архітектури мобільного застосунку

1. Презентаційний шар (Presentation Layer)

Презентаційний шар відповідає за формування графічного інтерфейсу користувача та обробку подій взаємодії. Він створений відповідно до парадигми декларативного UI, характерної для Flutter, де інтерфейс описується як функція стану (State → UI), а не як послідовність імперативних команд [3].

До ключових характеристик шару належать:

– Використання шаблону MVVM/BLoC. Логіка управління станами та реакцією на події винесена у ViewModel/BLoC, що забезпечує відсутність бізнес-логіки у віджетах, чистоту UI-коду та підвищену тестованість.

– Реактивна побудова інтерфейсу. Компоненти автоматично оновлюються при зміні стану, наприклад:

- завантаження відеопотоку,
- рекомендаційні оновлення,
- активізація адаптивного режиму,
- зміни емоційних параметрів користувача.

Структурно презентаційний шар включає такі модулі:

– екран перегляду контенту (відеоплеєр);

– екран рекомендацій;

– інтерфейс аналізу емоцій та імпорту зображення з камери;

– системні діалогові вікна (сповіщення, помилки, підтвердження дозволів).

Шар є повністю ізольованим від бізнес-логіки, що дозволяє модифікувати дизайн інтерфейсу без впливу на функціональність.

2. Логічний шар (Domain / Business Logic Layer)

Логічний шар містить доменні моделі, сервісні компоненти та правила функціонування системи. Саме тут реалізовано поведінку застосунку, яка визначає, як повинні оброблятися події, користувацькі дії, мережеві відповіді та результати емоційного аналізу.

Основні підсистеми логічного шару:

– Модуль управління мультимедійним відтворенням. Забезпечує контроль за:

• параметрами видеопотоку,

• перемотуванням,

• адаптивною зміною якості,

• синхронізацією UI з прогресом перегляду.

– Адаптивний модуль. Це один із ключових інтелектуальних компонентів системи. Він аналізує результати емоційного розпізнавання та коригує:

• рекомендаційні списки,

• темп або режим перегляду (динамічний, спокійний, режим концентрації),

• обрані UI-акценти (кольорові схеми, акценти уваги).

– Модуль управління станом (BLoC/Cubit). Забезпечує реактивну взаємодію між UI та доменним рівнем. Він виконує:

• трансформацію подій у стани,

• централізоване керування потоками даних,

• синхронізацію між сервісами, репозиторіями та інтерфейсом.

Логічний шар є незалежним від інтерфейсу та шару даних, що забезпечує легке тестування, повторне використання та можливість заміни компонентів без порушення архітектури [14].

3. Шар даних (Data Layer)

Шар даних відповідає за збереження, отримання, кешування та транспорт інформації. Він побудований на основі патерну Repository, який забезпечує абстракцію між джерелами даних та логікою застосунку.

Основні компоненти шару:

– Репозиторії. Вони інкапсулюють:

• локальні джерела даних,

• мережеві запити,

- обробку винятків,
- трансформацію DTO у доменні моделі.

– Локальне сховище (Hive / SharedPreferences). Зберігає:

- історію переглядів,
- емоційні параметри та профіль користувача,
- налаштування застосунку,
- кешовані рекомендації.

– Мережевий клієнт (Dio або http). Використовується для отримання:

- списків відеоконтенту,
- метаданих про фільми та серіали,
- службових конфігурацій.

Архітектура передбачає можливість безболісного переходу до повноцінного бекенд-сервера без модифікації логіки клієнтського застосунку.

4. Модуль обробки емоцій (Emotion Processing Module)

Цей модуль є високоспеціалізованим компонентом та інтегрує механізми машинного навчання у загальну структуру застосунку. Його побудовано з урахуванням обмежень мобільних пристроїв — низької пропускної здатності, ресурсної обмеженості та можливих затримок [8].

Компонент складається з:

– TFLite-моделі, оптимізованої для мобільних пристроїв та завантаженої в оперативну пам’ять застосунку.

– Модуля попередньої обробки зображень, який відповідає за:

- детекцію обличчя,
- нормалізацію зображення,
- адаптацію до вимог моделі.

– Подієвої логіки виклику аналізу, яка забезпечує активацію моделі лише у визначених сценаріях (запуск застосунку, початок перегляду, завершення відео), що значно зменшує обчислювальні витрати.

Модуль працює автономно від інших компонентів, але інтегрований через загальні сервіси подій та станів [9].

5. Модуль інтеграції та службові компоненти (Service Layer)

Цей рівень забезпечує підтримувальну інфраструктуру, необхідну для стабільної роботи системи. Він включає:

– компонент логування, що фіксує ключові події, помилки, результати емоційної класифікації;

– систему повідомлень та перехоплення помилок, яка уніфікує обробку винятків та інформування користувача;

– систему керування дозволами, зокрема доступ до камери та файлової системи;

– механізм впровадження залежностей (Dependency Injection) — найчастіше через GetIt, що забезпечує слабке зв’язування та можливість гнучкого конфігурування модулів [15].

Службові компоненти не взаємодіють безпосередньо із UI, але забезпечують функціональну цілісність системи та її стійкість до збоїв.

У підсумку запропонована архітектура забезпечує: високу модульність, масштабованість та можливість подальшого розвитку, ізоляцію відповідальностей, простоту інтеграції компонентів машинного навчання, гнучкість тестування та підтримки, стабільну продуктивність у мультимедійних сценаріях.

2.3 Побудова схем бази даних

Ефективне функціонування мобільного застосунку для відтворення мультимедійного контенту потребує надійної, масштабованої та гнучкої системи зберігання даних. У межах даного проєкту для реалізації серверної частини використовується хмарна платформа Firebase, зокрема документно-орієнтована база даних Cloud Firestore. Такий підхід дозволяє поєднати переваги

централізованого зберігання з можливістю синхронізації в реальному часі між різними клієнтськими пристроями.

Проєктування структури бази даних здійснювалося на основі аналізу предметної області та вимог до програмної системи. Особливу увагу приділено:

- підтримці персоналізації;
- збереженню історії взаємодії користувача із системою;
- забезпеченню механізмів емоційно-адаптивної поведінки застосунку.

Логічна модель даних

У межах предметної області «мобільне відтворення контенту» виділено три основні сутності:

1. Користувач (User) — суб’єкт взаємодії з системою, що має унікальний профіль.

2. Запис емоційного стану (Emotion) — фіксація результатів подієвого аналізу емоцій у визначених ключових моментах (вхід у застосунок, початок відео, пауза, завершення перегляду).

3. Історія переглядів (WatchHistory) — дані про споживання мультимедійного контенту, включно з метаданими відео та часом перегляду.

Взаємозв’язки між сутностями:

- Один користувач може мати багато записів історії переглядів (1:N).
- Один користувач може мати багато записів емоцій (1:N) [16].

Структурна організація Firestore

Cloud Firestore використовує ієрархічний принцип організації даних типу «колекція → документ → підколекція». Це дозволяє ефективно структурувати дані та легко масштабувати систему.

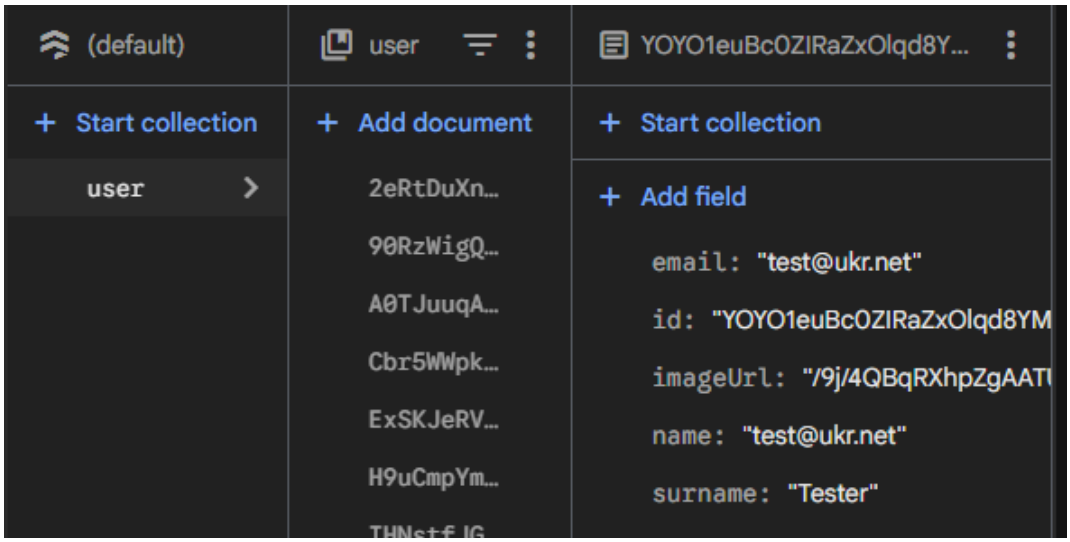

Рисунок 2.2 – Приклад ієрархії у Firestore

Така модель поєднує переваги денормалізованого зберігання з можливістю швидкого масштабування системи та дозволяє зберігати історію взаємодії користувачів і записи емоцій у структурованому вигляді.

Порівняння з реляційною моделлю

Для реляційних баз даних, як MS SQL Server, основними підходами є:

- Таблиці зі строгими зв’язками між ними (один-до-одного, один-до-багатьох).
- Використання первинних та зовнішніх ключів для забезпечення цілісності даних.

У Firestore:

- Зв’язки реалізуються через ідентифікатори документів замість зовнішніх ключів.
- Внутрішня денормалізація дозволяє зменшити кількість запитів і підвищити швидкість доступу до даних.
- Можлива одночасна синхронізація даних у реальному часі на всіх клієнтських пристроях [16].

Переваги та обмеження

Переваги Firestore для мобільного застосунку:

1. Підтримка кросплатформених додатків (Android, iOS, Web).

2. Вбудована синхронізація в реальному часі.

3. Можливість масштабування без зміни структури клієнтського коду.

4. Простота роботи з денормалізованими даними та підколекціями [17].

Обмеження:

1. Відсутність традиційних SQL-запитів — складні join-операції важко реалізувати.

2. Денормалізація може призвести до дублювання даних.

3. Вартість запитів залежить від кількості операцій читання і запису.

Візуалізація структури бази даних

Для наочного представлення архітектури зберігання даних наведена ER-діаграма (рис. 2.3), яка відображає взаємозв'язки між основними колекціями: users, watchHistory, emotion.

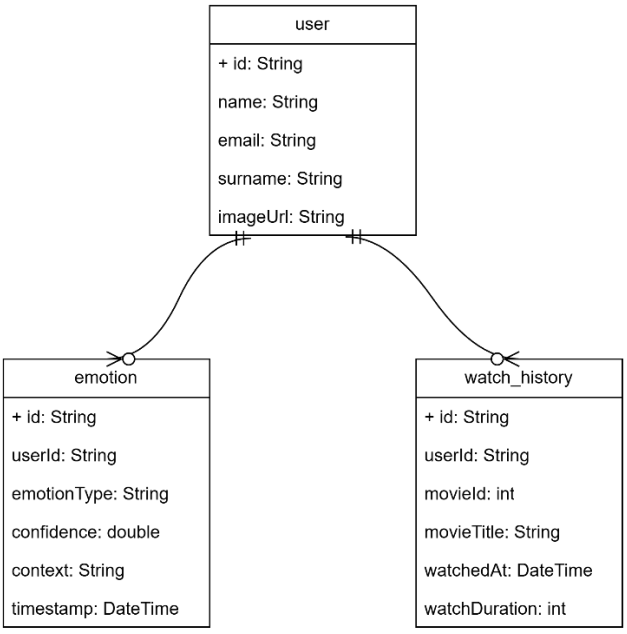

Рисунок 2.3 – ER діаграма бази даних

Ця схема демонструє логічну структуру бази даних та спосіб взаємодії між сутностями, що дозволяє ефективно реалізувати персоналізацію та адаптивність рекомендацій.

2.4 Побудова UML-діаграм класів

Побудова UML-діаграм класів є ключовим етапом проєктування мобільного застосунку, оскільки дозволяє чітко визначити основні компоненти системи, їхні атрибути, методи та взаємозв’язки. Це спрощує розробку, тестування та інтеграцію нових функцій, зокрема адаптивного відтворення контенту на основі емоційного стану користувача [17].

Основні класи застосунку можна розділити на кілька груп:

Архітектурні базові класи забезпечують єдину структуру для репозиторіїв та обробки даних.

BaseRepository – базовий клас для всіх репозиторіїв із реалізацією загальної обробки помилок.

ApiRepository – базовий клас для роботи з API із безпечними викликами.

Класи аутентифікації керують авторизацією та роботою з користувачами. AuthCubit, AuthCubitImpl, LoginCubit, LoginCubitImpl, SignUpCubit, SignUpCubitImpl, ResetPassCubit, ResetPassCubitImpl, SignWithCubit, SignWithCubitImpl забезпечують управління станом процесів авторизації, логіну, реєстрації та відновлення пароля.

AuthRepository працює з Firebase Auth та Google Sign In, а UserModel зберігає дані користувача (id, name, email, imageUrl).

Класи для роботи з мультимедійним контентом реалізують функціонал перегляду та управління відео.

HomeCubit, HomeCubitImpl відповідає за завантаження фільмів на головному екрані та управління станом UI. TmdbRepository здійснює запити до TMDB API (популярні, топ-рейтинг, трейлери, жанри).

MovieModel, MovieResponse, VideoResponse представляють дані про фільми та медіа.

DetailMovieScreen, TrailerScreen, CastDetailScreen та CastList відображають деталі контенту та інформацію про акторів.

VideoPlayerController буде додатковим класом для керування відтворенням відео з урахуванням адаптивного режиму.

Класи для роботи з улюбленими та профілем користувача забезпечують персоналізацію.

FavoriteCubit, FavoriteCubitImpl та FavoriteMovieRepo, FavoriteMovieService керують списком улюблених фільмів.

UserCubit, UserCubitImpl, ProfileScreen, EditProfileScreen, ChangePasswordScreen відповідають за управління профілем та налаштуваннями користувача.

UserRemoteStorageRepo та StorageService забезпечують синхронізацію та локальне зберігання даних користувача.

SettingsRepo, SettingsModel керують налаштуваннями застосунку, а LangCubit, LangCubitImpl – локалізацією.

Класи мережевого шару забезпечують обмін даними з API. RestClient реалізує HTTP-запити до TMDB API, включаючи отримання популярних фільмів, деталей, пошуку та трейлерів.

UI-компоненти (UIKit) включають AppBar, MovieCard, HorizontalMovieList, AppLoader, кнопки (AppPrimaryButton, AppTextButton), поля введення (AppInputField) та навігацію (BottomBar).

Класи обробки помилок забезпечують стабільність системи. AppException, ExceptionMapper відповідають за генерацію та маппінг помилок.

Для інтеграції подієвого емоційного аналізу додано нові класи:

EmotionAnalyzer – відповідає за роботу з TFLite моделлю для визначення емоцій користувача.

EmotionRecord – модель зберігання результатів аналізу (тип емоції, інтенсивність, timestamp).

AdaptiveController – адаптує сценарії відтворення та рекомендації залежно від емоційного стану користувача, взаємодіє з VideoPlayerController та RecommendationEngine.

RecommendationEngine формує добірки контенту на основі поведінки та емоцій користувача.

Зв’язки між класами відображають логіку системи:

User має один-до-багатьох зв’язок із WatchHistory та EmotionRecord;

User асоційований один-до-одного з RecommendationEngine;

EmotionAnalyzer і AdaptiveController взаємодіють один-до-одного, де AdaptiveController керує VideoPlayerController та оновлює рекомендації через RecommendationEngine.

Для логічності моделі передбачені допоміжні класи CameraService, EmotionCaptureService та EmotionRepository, які забезпечують отримання вхідних даних, керування процесом захоплення емоцій та їх збереження в базі даних.

UML-діаграми класів (рисунок 2.4 – 2.6) показують основні класи застосунку, їхні атрибути, методи та ключові взаємозв’язки. Вона дозволяє зрозуміти структуру застосунку і спланувати реалізацію адаптивного функціоналу з емоційним контекстом.

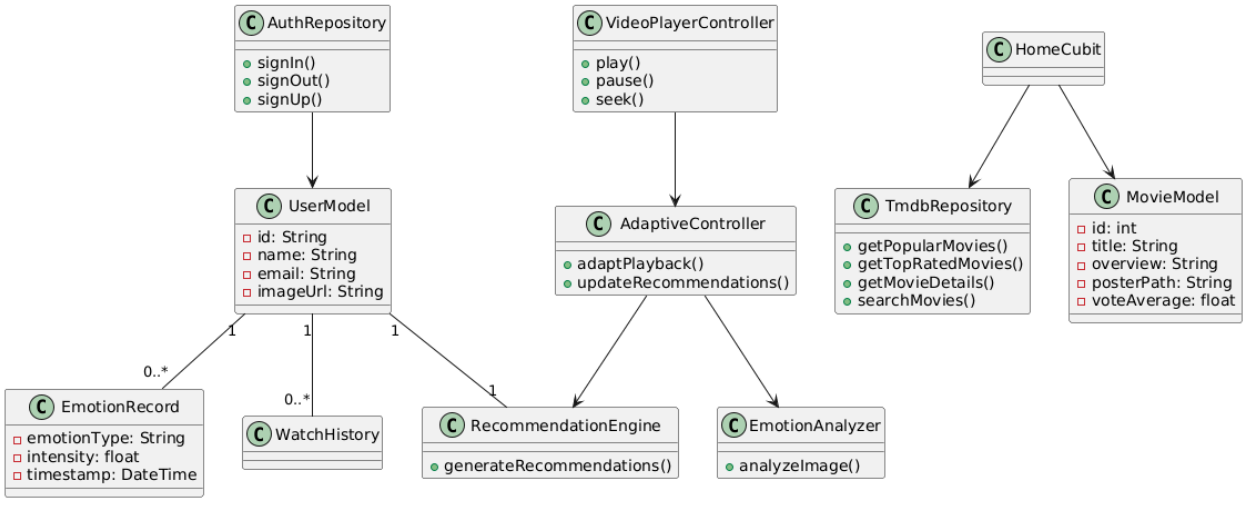

Рисунок 2.4 – UML-діаграма класів

Рисунок 2.5 – UML-діаграма класів

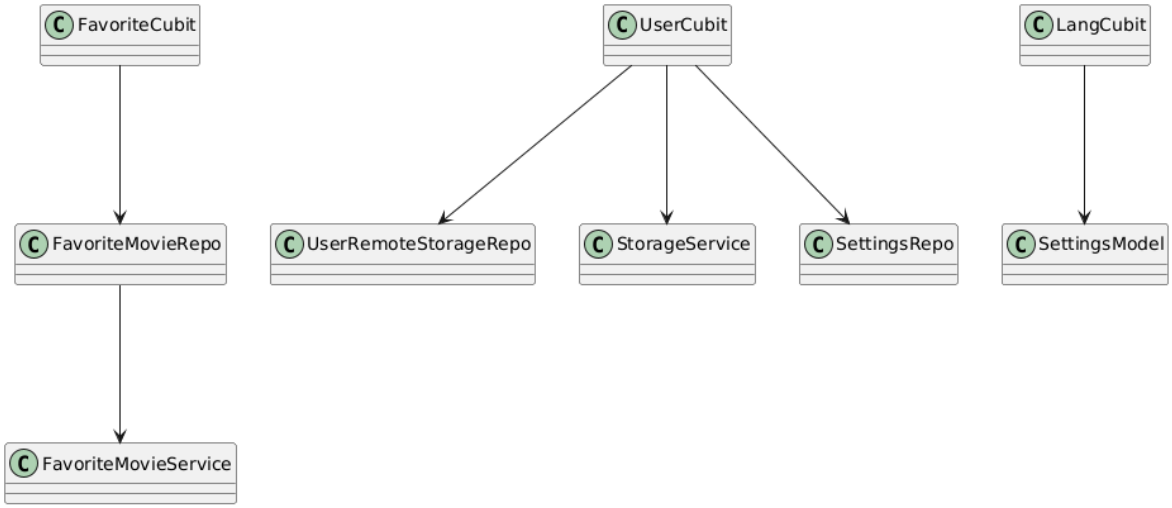

Рисунок 2.6 – UML-діаграма класів

2.5 Вибір мови та середовища розробки

Вибір мови програмування та середовища розробки є стратегічно важливим етапом проєктування програмної системи, оскільки саме на цьому рівні закладаються фундаментальні властивості програмного продукту:

продуктивність, надійність, масштабованість, переносимість та можливості інтеграції з сучасними цифровими сервісами. Для мобільних мультимедійних застосунків, що працюють із потоковими відеоданими, персоналізацією контенту та елементами інтелектуального аналізу (зокрема емоційного стану користувача), вибір технологічного стеку має критичне значення.

У межах даного проєкту як основну мову програмування обрано Dart, а як базовий інструмент розробки — кросплатформний фреймворк Flutter. Такий вибір зумовлений низкою технологічних та практичних переваг. Flutter забезпечує можливість створення єдиного коду для операційних систем Android та iOS, що суттєво скорочує час розробки та спрощує підтримку проєкту. Завдяки використанню власного рушія рендерингу, Flutter гарантує стабільну частоту кадрів і плавність анімацій, що є критично важливим для мультимедійних застосунків із динамічним інтерфейсом.

Мова програмування Dart характеризується сучасною об’єктно-орієнтованою парадигмою, підтримкою функціонального програмування та суворою статичною типізацією. Це дозволяє зменшити кількість помилок на етапі компіляції та підвищує надійність програмного коду. Особливу роль у межах цього проєкту відіграє підтримка асинхронного програмування, реалізована через механізми Future, Stream та async/await. Дана модель є надзвичайно важливою для обробки мережевих запитів, потокової передачі відеоданих, роботи з хмарними базами даних та інтеграції з сервісами, що працюють у режимі реального часу [18].

Як основне інтегроване середовище розробки було обрано Android Studio, яке є офіційно підтримуваною платформою для створення Flutter-застосунків. Android Studio забезпечує розширені можливості статичного аналізу коду, зручні інструменти налагодження (debugging), профілювання продуктивності, моніторингу використання пам’яті та процесорних ресурсів. Також середовище підтримує інтеграцію з емуляторами та фізичними мобільними пристроями, що

дозволяє проводити тестування в умовах, максимально наближених до реальних сценаріїв використання.

Для організації архітектури застосунку використано підхід, заснований на патерні BLoC/Cubit, який забезпечує чітке розмежування між рівнем представлення (UI) та бізнес-логікою. Така архітектура сприяє кращій тестованості програмного продукту, спрощує модульне тестування та підвищує керованість коду у великих проєктах. Для автоматичної генерації незмінних (immutable) моделей даних та станів використовується бібліотека Freezed, що дозволяє мінімізувати ризики виникнення логічних помилок і підвищити стабільність системи.

Навігація між екранами реалізована за допомогою бібліотеки GoRouter, яка підтримує декларативний підхід до опису маршрутів, обробку параметризованих шляхів і глибоких посилань (deep links). Це дозволяє створювати складні сценарії переходів між екранами та забезпечує логічну структурованість інтерфейсу.

Для реалізації мережевої взаємодії із зовнішніми сервісами застосовується бібліотека Dio, що підтримує роботу з REST API, перехоплювачі (interceptors), обробку помилок та повторні спроби виконання запитів. Шаблон проєктування Repository використовується для інкапсуляції логіки доступу до даних та відокремлення мережевого шару від бізнес-логіки.

Контроль версій програмного коду здійснюється за допомогою розподіленої системи Git, що забезпечує збереження історії змін, підтримку гілок розробки та можливість колективної роботи. Керування бібліотечними залежностями реалізується через стандартний менеджер пакетів Flutter — pub, що спрощує підключення сторонніх бібліотек та оновлення компонентів.

Таким чином, обрані мова програмування Dart, фреймворк Flutter та комплекс інструментальних засобів формують надійний технологічний фундамент для розробки мобільного застосунку з підтримкою потокового відтворення контенту та емоційно-адаптивних механізмів персоналізації,

забезпечуючи оптимальний баланс між продуктивністю, гнучкістю та зручністю подальшого розвитку системи.

2.6 Реалізація основних класів та методів

У процесі реалізації мобільного застосунку була сформована система основних класів, кожен з яких виконує чітко визначену роль у загальній архітектурі. Класи побудовані відповідно до принципів багатошарової архітектури та патерну BLoC/Cubit, що забезпечує розділення бізнес-логіки та рівня представлення [19].

Базові архітектурні класи

Клас BaseRepository містить універсальні поля та методи для обробки помилок:

- поле exceptionMapper — відповідає за перетворення технічних помилок у прикладний формат;
- метод handleError() — централізовано обробляє виключення.

Клас ApiRepository розширює BaseRepository та додає:

- метод safeApiCall() — виконує захищені асинхронні виклики до мережевих сервісів;
- механізми логування помилок мережевого рівня.

Класи аутентифікації

Клас UserModel містить основні поля користувача:

- id: String
- name: String
- email: String
- imageUrl: String

Клас AuthRepository реалізує методи:

- signIn(email, password)
- signUp(email, password)

- signOut()
- resetPassword(email)

Класи AuthCubit, LoginCubit, SignUpCubit містять:

- поле state — поточний стан авторизації;
- методи login(), register(), logout(), emitState() для управління інтерфейсом.

Класи роботи з мультимедійним контентом

Клас MovieModel містить ключові поля:

- id: int
- title: String
- overview: String
- posterPath: String
- voteAverage: double

Клас TmdbRepository реалізує методи:

- getPopularMovies()
- getTopRatedMovies()
- getMovieDetails(id)
- searchMovies(query)

Клас HomeCubit має:

- поле movies: List<MovieModel>
- методи loadMovies() та refreshMovies().

Клас VideoPlayerController містить методи:

- play()
- pause()
- seekTo(position)
- setPlaybackSpeed(speed)

Класи персоналізації

Клас FavoriteMovieRepo реалізує:

- методи addToFavorites(movie)

- removeFromFavorites(movie)
- getFavorites()

Клас UserCubit має:

- поле currentUser: UserModel
- методи loadUserProfile() та updateProfile().

Клас SettingsModel містить поля:

- language: String
- themeMode: String

Класи емоційного аналізу

Клас EmotionAnalyzer містить:

- поле model: TfLiteModel
- методи loadModel(), analyzeFrame(image), predictEmotion().

Клас EmotionRecord включає поля:

- emotionType: String
- intensity: double
- timestamp: DateTime

Клас AdaptiveController реалізує:

- метод adaptPlayback(emotion)
- метод updateRecommendations(emotion)

Клас RecommendationEngine містить:

- метод generateRecommendations(userId)

Клас EmotionRepository реалізує методи:

- saveEmotion(record)
- getUserEmotions(userId)

Клас CameraService містить:

- методи initializeCamera()
- captureFrame()

Допоміжні класи

Клас RestClient містить методи:

- getPopularMovies()
- getMovieDetails()
- searchMovies()

Клас ExceptionMapper реалізує:

- метод map(error)

2.7 Розробка інтерфейсу користувача

Розробка інтерфейсу користувача є важливою складовою процесу створення програмного продукту, оскільки саме графічний інтерфейс забезпечує ефективну взаємодію людини з інформаційною системою. Якість інтерфейсу безпосередньо впливає на швидкість засвоєння функціоналу застосунку, рівень задоволеності користувача та загальну ефективність використання мобільного продукту. У межах даного проєкту інтерфейс розглядався не лише як засіб візуалізації даних, а як інтелектуальний шар взаємодії, що адаптується до поведінкових та емоційних характеристик користувача.

Проєктування інтерфейсу виконувалося на основі принципів user-centered design (орієнтація на користувача), а також із урахуванням вимог сучасних мобільних платформ щодо навігації, доступності та продуктивності. Особлива увага приділялася мінімізації когнітивного навантаження, логічній структурі меню та узгодженості візуальних елементів у різних частинах застосунку.

Технічно інтерфейс реалізовано за допомогою фреймворку Flutter, який забезпечує високопродуктивний рендеринг графічних елементів, реактивну побудову UI та повторне використання компонентів [20]. Кожен екран застосунку реалізовано у вигляді незалежного віджета, що спрощує підтримку коду та тестування. Основні екрани: splash_screen, авторизація та реєстрація, home_screen, favorite_screen, camera_screen, profile_screen, movie_detail_screen.

Головний екран застосунку являє собою динамічну стрічку контенту, що поєднує горизонтальні та вертикальні списки фільмів, банери з анонсами та

інтерактивні елементи навігації. Для побудови таких списків використовуються кастомні компоненти MovieCard, HorizontalMovieList та адаптивні контейнери, які автоматично підлаштовуються під розмір екрану, DPI та орієнтацію пристрою.

Окремі екрани авторизації та реєстрації розроблені з урахуванням принципів безпеки та зручності: реалізована перевірка введених даних у режимі реального часу, підсвічування помилок, а також захист від некоректного введення. Процес входу максимально спрощений за рахунок інтеграції соціальної авторизації.

Екран детальної інформації про фільм поєднує текстову та мультимедійну складову: відображення рейтингу, жанрів, опису, списку акторів та інтегрований відеоплеєр. Елементи керування відтворенням оптимізовані для сенсорного управління, підтримують жестову навігацію та синхронізовані з прогресом відео.

Окремий модуль інтерфейсу реалізовано для роботи з профілем користувача та персональними налаштуваннями. Користувач має можливість переглядати та редагувати персональні дані, змінювати мовні параметри, тему оформлення та налаштування конфіденційності.

У межах проєкту реалізовано експериментальні елементи емоційно-адаптивного інтерфейсу. Під час відтворення контенту застосунок може змінювати кольорові акценти, анімації або порядок відображення рекомендацій у реальному часі залежно від емоційного стану користувача, що визначається модулем аналізу емоцій [18].

Для підвищення доступності реалізовано підтримку локалізації, масштабування шрифтів та адаптації інтерфейсу для користувачів з особливими потребами. Передбачено коректну роботу застосунку на пристроях із різною роздільною здатністю та співвідношенням сторін.

Для організації навігації між екранами використовується бібліотека GoRouter, яка забезпечує декларативний опис маршрутів, підтримку вкладених навігаційних стеків та обробку глибоких посилань (deep links).

Таким чином, розроблений інтерфейс користувача поєднує сучасні підходи до UI/UX-дизайну, гнучку технічну реалізацію та інтеграцію інтелектуальних механізмів адаптації на основі емоційного стану користувача, що відповідає вимогам до сучасних мультимедійних мобільних застосунків.

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА

У даному розділі розглянуто завершальні етапи життєвого циклу програмної системи мобільного мультимедійного застосунку, а саме тестування, розгортання, верифікацію та підготовку до впровадження. Основну увагу приділено оцінюванню якості реалізованого програмного продукту, перевірці його функціональної та нефункціональної відповідності вимогам, а також аналізу стабільності, продуктивності й безпеки роботи системи. Описано застосовані види та план тестування, розробку тестових сценаріїв, процес розгортання програмної системи та системні вимоги до середовища виконання. Окремо розглянуто результати верифікації, які підтверджують коректність реалізації адаптивних механізмів персоналізації та готовність мобільного застосунку до експлуатації в реальних умовах.

3.1. Тестування програмної системи

Тестування програмної системи є систематичним процесом верифікації та валідації програмного продукту, спрямованим на виявлення дефектів, перевірку відповідності реалізації початковим вимогам, а також оцінювання рівня якості програмного забезпечення [19]. Для мобільних застосунків особливо важливими є надійність, стабільність, продуктивність, безпека та коректна обробка асинхронних процесів, зокрема взаємодії з мережею, кешування даних, потокового відтворення мультимедійного контенту та реакції інтерфейсу на зміни стану користувача.

У межах даної роботи тестування проводилося з використанням кросплатформового підходу: код розроблявся у VSCode, а тестування виконувалося у середовищі Android Studio на віртуальному пристрої Pixel 9a (Android 16.0, API 36.0) [20-21]. Такий підхід дозволив оцінити реальну поведінку застосунку на сучасному апаратному середовищі, перевірити

продуктивність інтерфейсу та мультимедійного потоку, а також протестувати роботу адаптивних компонентів, що реагують на емоційний стан користувача.

Особлива увага приділялася перевірці:

– механізмів персоналізації та адаптивного відтворення контенту на основі емоційного стану користувача;

– стабільності та швидкодії при роботі з мультимедійними файлами, включаючи завантаження відеопотоків і буферизацію;

– коректності функціонування в умовах нестабільного або повільного мережевого з’єднання, що імітувалося через емулятори мережевих затримок;

– взаємодії між BLoC-компонентами, репозиторіями та UI-шаром, забезпечення синхронізації станів та даних.

Тестування охоплювало як функціональні, так і нефункціональні параметри системи.

Функціональні перевірки включали детальне тестування всіх ключових екранів та модулів застосунку:

– splash_screen – початкове завантаження додатку та ініціалізація сервісів (рисунок 3.1);

Рисунок 3.1 – екран splash

– home_screen – головна сторінка зі списками фільмів, жанрами та пошуком, перевірка коректного завантаження списків та оновлення рекомендацій (рисунок 3.2)

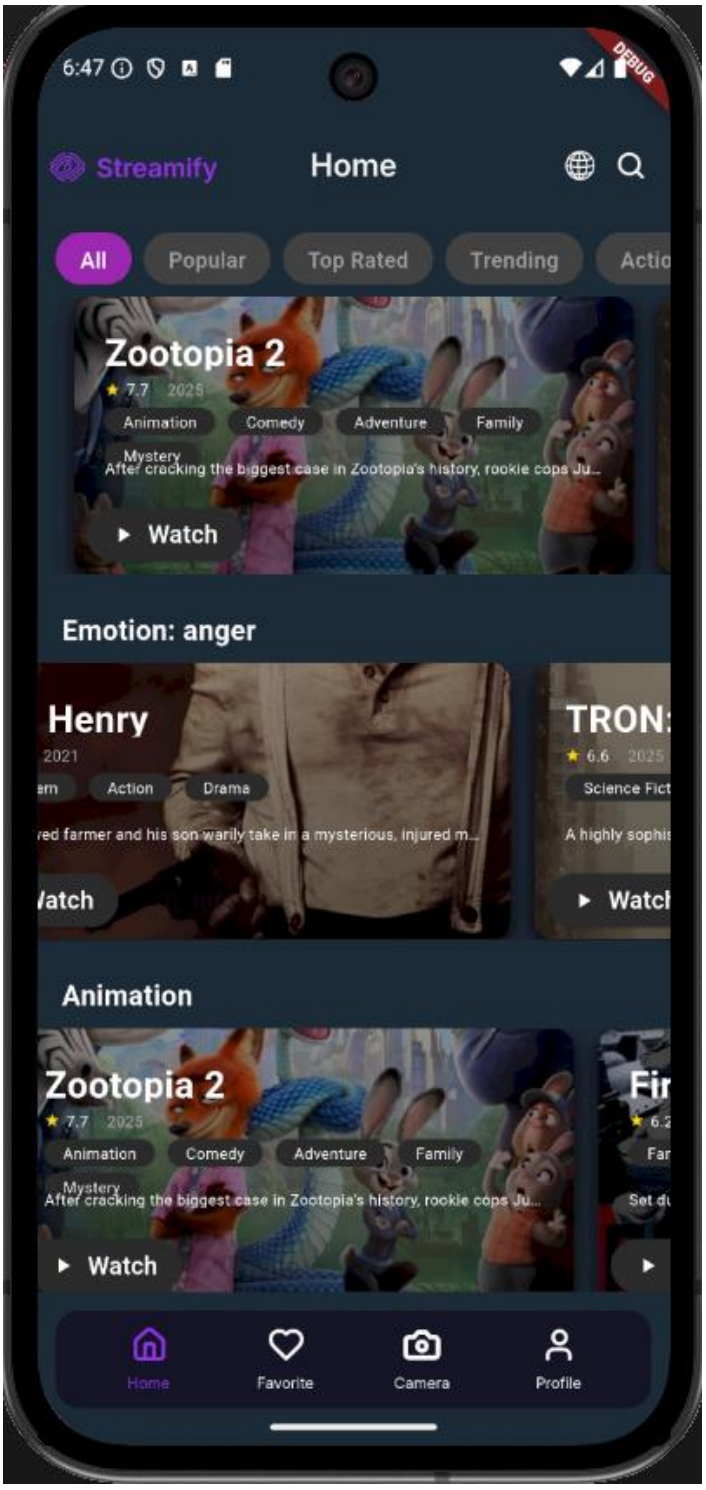

Рисунок 3.2 – екран сторінки home

– favorite_screen – екран улюблених фільмів із перевіркою збереження, видалення та відображення елементів (Рисунок 3.3);

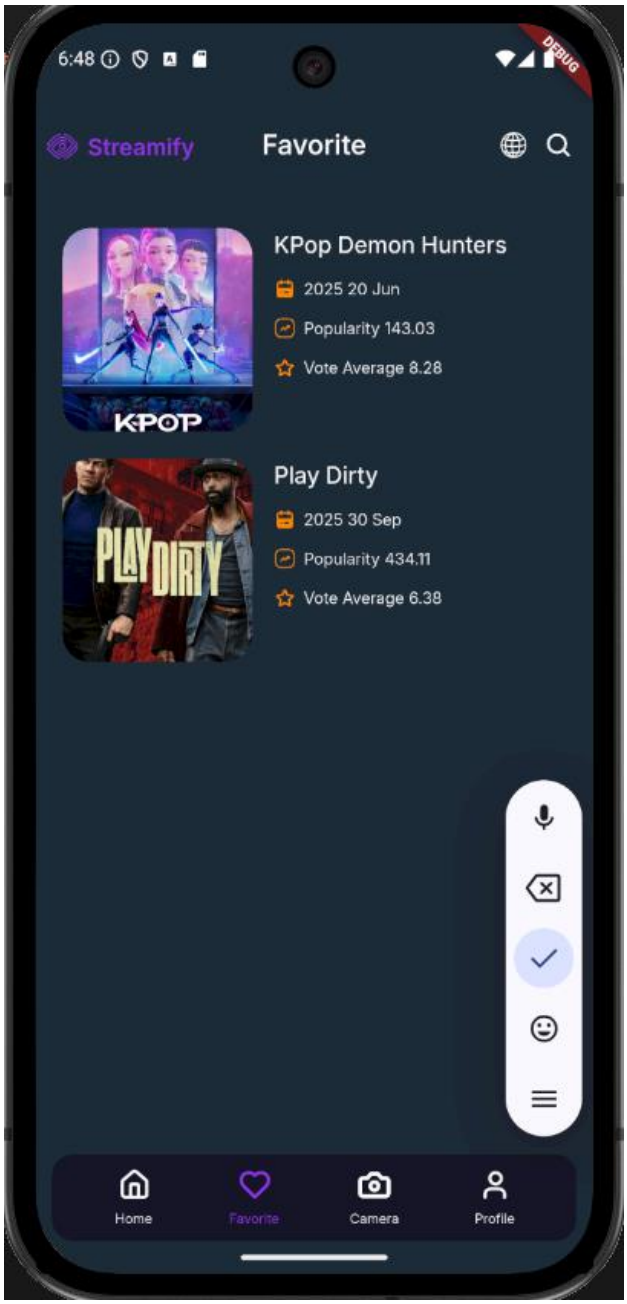

Рисунок 3.3 – екран сторінки favorite

– camera_screen – модуль зчитування емоцій користувача, тестування роботи камери, захоплення кадрів та передачі даних у модуль аналізу (Рисунок 3.4);

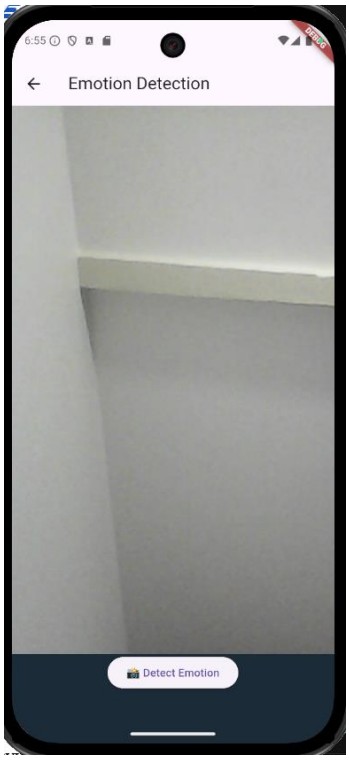

Рисунок 3.4 – екран сторінки camera

– profile_screen – сторінка профілю користувача з перевіркою редагування даних, зміни налаштувань та синхронізації з сервером (Рисунок 3.5);

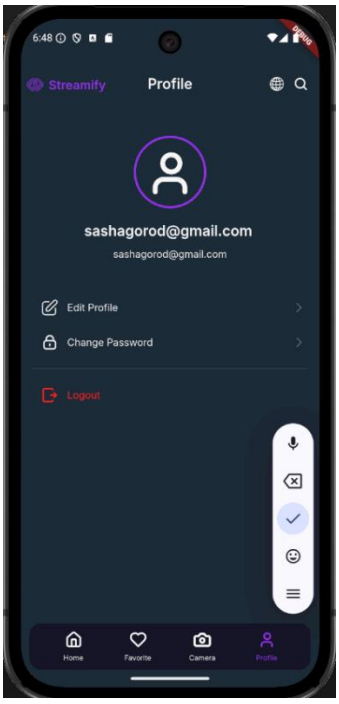

Рисунок 3.5 – екран сторінки profile

– movie_detail_screen – детальна інформація про фільм з інтегрованим відеоплеєром, перевірка керування відтворенням, буферизації та адаптивного режиму відтворення залежно від емоційного стану користувача (Рисунок 3.6).

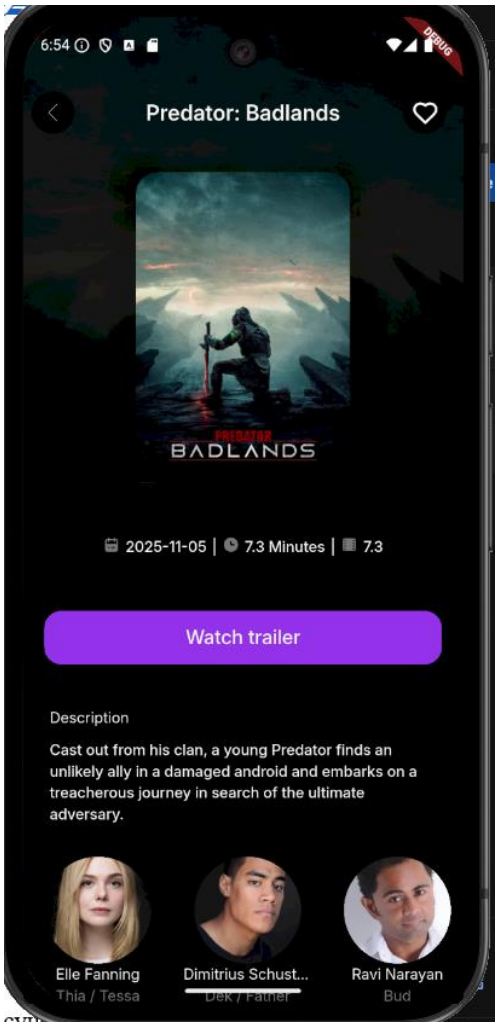

Рисунок 3.5 – екран сторінки детальної інформації про фільм із плеєром

Нефункціональні перевірки включали:

– Продуктивність (performance testing): оцінка часу відгуку системи, швидкості завантаження списків та відтворення відео, оптимізація використання ресурсів пам’яті та процесора;

– Стрес-тестування (stress testing): перевірка стабільності під час тривалого безперервного використання, інтенсивного оновлення рекомендацій та одночасного відкриття декількох потоків мультимедіа;

– Стійкість (robustness testing): визначення поведінки додатку при некоректних діях користувача, введенні пошкоджених даних або несподіваних сценаріях;

– Безпека (security testing): перевірка надійності зберігання токенів доступу, шифрування персональних даних, стійкості до MITM-атак, а також контроль правильності реалізації протоколів аутентифікації та передачі даних [22].

Тестування проводилося протягом всього життєвого циклу розробки застосунку, що дозволило виявляти помилки на ранніх стадіях, мінімізувати ризики критичних відмов та підвищити загальну якість програмного продукту. Крім того, результативність тестування документувалася у вигляді скріншотів екранів, логів роботи застосунку та звітів про виконання тестових сценаріїв, що забезпечувало прозорість процесу та можливість повторного використання тестових даних для подальшого аналізу та вдосконалення системи.

3.1.1 Види та план тестування

Процес тестування розробленої програмної системи організовано відповідно до принципів багаторівневої верифікації програмного забезпечення, що передбачають поетапну перевірку працездатності окремих модулів і системи в цілому [23]. Такий підхід забезпечує високу якість ПЗ, дозволяє виявляти критичні дефекти на ранніх стадіях та знижує ризики їхнього впливу на кінцеву роботу застосунку.

У межах цього проєкту виділено наступні види тестування:

1. Модульне тестування (Unit Testing)

Метою модульного тестування було перевірити ізольовану поведінку окремих класів та методів, а також BLoC/Cubit-компонентів, які відповідають за керування станами інтерфейсу. Об’єктами модульного тестування були:

– Репозиторії доступу до даних (TmdbRepository, FavoriteMovieRepo, UserRemoteStorageRepo) для перевірки коректності запитів, обробки відповідей та кешування.

– BLoC/Cubit-компоненти (HomeCubit, FavoriteCubit, UserCubit, AuthCubit), що забезпечують детерміновану зміну станів інтерфейсу.

– Сервіси аналізу емоцій (EmotionAnalyzer, AdaptiveController, RecommendationEngine) для перевірки алгоритмів прогнозування емоцій та формування персоналізованих рекомендацій.

Для досягнення ізоляції тестованих компонентів використовувалися мок-об’єкти (Mockito) та fake-класи, що дозволяло виключити вплив зовнішніх залежностей та перевіряти логіку поведінки системи в контролюваних умовах [21, 24].

2. Інтеграційне тестування (Integration Testing)

Інтеграційне тестування виконувалося для оцінки сумісної роботи декількох компонентів системи. Основні завдання включали:

– Перевірку взаємодії між мережею, локальним кешем та бізнес-логікою через репозиторії та BLoC/Cubit.

– Тестування маршрутизації потоків даних між UI-шаром та BLoC-компонентами на екранах home_screen, favorite_screen, profile_screen.

– Перевірку цілісності та узгодженості результатів модулів емоційного аналізу та механізмів адаптивного відтворення контенту (camera_screen + movie_detail_screen).

– Імітацію нормальної та аномальної поведінки API за допомогою емуляторів серверних відповідей [23].

3. Системне тестування (System Testing)

На цьому рівні оцінювалася робота застосунку як єдиного продукту. Основні сценарії включали:

– Процес автентифікації та авторизації (Login, SignUp, Reset Password, Social Sign-In).

– Навігацію між усіма основними екранами (splash_screen, home_screen, favorite_screen, camera_screen, profile_screen, movie_detail_screen).

– Завантаження та відтворення мультимедійного контенту з урахуванням адаптивного режиму відтворення залежно від емоцій користувача.

– Роботу офлайн-режиму та повторну синхронізацію даних після відновлення з’єднання.

– Обробку критичних ситуацій: мережеві збої, втрата токена сесії, некоректні дані [22, 24].

4. Нефункціональне тестування

Крім функціональних перевірок, проводилися тести продуктивності та стійкості:

– Performance Testing — оцінка часу відгуку інтерфейсу, швидкості завантаження відеопотоків та оновлення списків рекомендацій.

– Stress Testing — перевірка стабільності при тривалому використанні та високих навантаженнях на сервери та UI.

– Robustness Testing — виявлення поведінки системи при некоректних діях користувача або пошкоджених даних.

– Security Testing — перевірка збереження токенів доступу, шифрування персональних даних та стійкості до атак MITM [25].

План тестування було оформлено у вигляді структурованого документа, що включав:

– Класифікацію тестових об’єктів (UI, репозиторії, BLoC/Cubit, сервіси емоційного аналізу).

– Визначення критеріїв прийнятності (Acceptance Criteria) для кожного модуля та екрана.

– Матрицю відповідності вимог до тестів (Requirements-to-Tests Mapping).

– Календарний план виконання тестових активностей із зазначенням етапів модульного, інтеграційного та системного тестування.

– Метрики якості: покриття коду тестами (coverage), щільність дефектів (defect density), середній час до відмови (MTTF), індекс надійності (reliability index).

– Правила фіксації результатів тестових сесій, включно зі скріншотами, логами та звітами про проходження сценаріїв.

Реалізація цього плану забезпечила систематичний, контрольований та прозорий процес оцінки якості програмного продукту, що суттєво підвищило надійність та готовність мобільного застосунку до впровадження.

3.1.2 Розробка тестових сценаріїв

Розробка тестових сценаріїв є логічним продовженням планування тестування та передбачає детальне описання дій користувача, очікуваних результатів та критеріїв оцінки коректності роботи системи. Для даного мобільного застосунку сценарії були розроблені на основі функціональних вимог, архітектурних рішень та використання адаптивних механізмів, що реагують на емоційний стан користувача.

Тестові сценарії поділено на категорії відповідно до ключових екранів та модулів системи:

1. Екран splash_screen

Мета: перевірка ініціалізації застосунку та запуску необхідних сервісів. Тестові кроки:

1. Запустити застосунок.

2. Перевірити відображення логотипу та анімації splash.

3. Дочекатися переходу на home_screen після завершення ініціалізації.

Очікуваний результат: екран splash відображається коректно, сервісні модулі ініціалізовані, автоматичний перехід на home_screen відбувається без помилок. (рис 3.1) [21].

2. Екран home_screen

Мета: перевірка завантаження та відображення списків фільмів, жанрів та функціоналу пошуку.

Тестові кроки:

1. Перевірити завантаження популярних та рекомендованих фільмів через HomeCubit.

2. Виконати пошук фільму за назвою.

3. Прокрутити горизонтальні списки жанрів і перевірити їх реакцію на натискання.

4. Перевірити оновлення рекомендацій залежно від емоційного стану користувача (передане через AdaptiveController).

Очікуваний результат: усі списки завантажені та відображені коректно, пошук повертає релевантні результати, рекомендації змінюються відповідно до емоцій (рис 3.2) [23].

3. Екран favorite_screen

Мета: перевірка роботи зі списком улюблених фільмів.

Тестові кроки:

1. Додати фільм до улюблених із home_screen.

2. Відкрити favorite_screen та перевірити наявність доданого фільму.

3. Видалити фільм із списку улюблених.

4. Перевірити оновлення списку та коректність збереження змін у локальному сховищі.

Очікуваний результат: усі дії виконуються без помилок, списки синхронізовані із сховищем (рис 3.3) [24].

4. Екран camera_screen

Мета: тестування модуля захоплення емоцій користувача.

Тестові кроки:

1. Ініціалізувати камеру через CameraService.

2. Захопити кадр та передати його на аналіз у EmotionAnalyzer.

3. Перевірити збереження результату у EmotionRecord.

4. Виконати адаптацію рекомендацій через AdaptiveController.

Очікуваний результат: камера працює без помилок, емоції коректно аналізуються та передаються у систему рекомендацій (риск 3.4) [25].

5. Екран profile_screen

Мета: перевірка управління профілем користувача та синхронізації даних.

Тестові кроки:

1. Відкрити профіль користувача через UserCubit.

2. Змінити персональні дані (ім’я, фото профілю).

3. Перевірити збереження змін у локальному та віддаленому сховищі.

4. Змінити налаштування мови та теми застосунку.

Очікуваний результат: зміни збережені, відображаються у всіх частинах застосунку, синхронізація з сервером пройшла успішно (рис 3.5) [16, 22].

6. Екран movie_detail_screen

Мета: перевірка відображення детальної інформації про фільм та функціоналу відеоплеєра.

Тестові кроки:

1. Відкрити деталі фільму з home_screen.

2. Перевірити відображення жанрів, рейтингу та акторського складу.

3. Запустити відеоплеєр, виконати play, pause, seek та зміну швидкості відтворення.

4. Перевірити адаптивне відтворення залежно від емоцій користувача.

Очікуваний результат: усі елементи відображаються коректно, керування плеєром працює без затримок, адаптація контенту працює відповідно до емоцій (рис 3.6) [4, 5].

Загальні положення щодо сценаріїв тестування

– Для кожного тестового сценарію визначено початковий стан, послідовність дій користувача, очікуваний результат та критерій прийнятності.

– Тестування виконувалося на віртуальному пристрої Pixel 9a (Android 16.0, API 36.0), що дозволило перевірити поведінку застосунку в умовах реального середовища.

– Всі результати фіксувалися через скріншоти, логи та звіти про проходження тестів, що дозволяло забезпечити прозорість процесу та можливість повторного аналізу.

3.2 Розгортання програмної системи та системні вимоги

Розгортання програмної системи є важливим етапом життєвого циклу розробки, оскільки забезпечує підготовку робочого середовища для тестування, демонстрації та експлуатації застосунку [26]. Для мобільних застосунків з потоковим мультимедійним контентом та адаптивними елементами, що реагують на емоційний стан користувача, правильне налаштування середовища та дотримання системних вимог є критично важливими для забезпечення стабільної роботи та коректної інтеграції з зовнішніми сервісами [3, 5].

Для розробки та підтримки даного мобільного застосунку використовувалися наступні інструменти та платформи:

Середовище розробки:

– VSCode як основне середовище для написання коду Dart та Flutter.

– Android Studio для тестування та емуляції пристроїв, що дозволяє виконувати налагодження та профілювання застосунку [22].

Апаратні вимоги:

– Розробка та тестування проводилися на сучасному комп’ютері з оперативною пам’яттю 8 ГБ, процесором AMD Ryzen 5 та SSD-диском для швидкого збирання проєкту.

Емулятор мобільного пристрою:

– Віртуальний пристрій Pixel 9a (Android 16.0, API 36.0).

– Підтримка емуляції камери, доступу до мережі та відтворення мультимедійного контенту.

Програмні вимоги:

– Flutter SDK останньої стабільної версії, сумісної з Android Studio та VSCode.

– Dart SDK для компіляції та виконання бізнес-логіки застосунку.

– Підключені пакети: Dio (HTTP-запити), Freezed (immutable моделі), GoRouter (маршрутизація), Firebase Auth (аутентифікація), а також додаткові бібліотеки для UI-компонентів та відтворення відео [18-19].

Мережеві вимоги:

– Підключення до інтернету для роботи з TMDB API, Face++ (аналіз емоцій), та Firebase сервісами.

– Можливість емуляції нестабільного з’єднання для тестування стійкості та адаптивності системи [12].

Процес розгортання застосунку включав наступні кроки:

1. Підготовка середовища розробки: встановлення Flutter SDK, Dart SDK та необхідних плагінів у VSCode та Android Studio.

2. Імпорт проєкту: відкриття Flutter-проєкту у VSCode для розробки та Android Studio для тестування.

3. Налаштування емулятора: створення віртуального пристрою Pixel 9a із заданою версією Android (API 36.0), активація емуляції камери та мережі.

4. Збирання проєкту: використання команд flutter pub get для підключення залежностей та flutter run для запуску застосунку на віртуальному пристрої.

5. Тестування функціоналу: перевірка роботи ключових екранів, взаємодії з API, кешування та адаптивних компонентів.

6. Документування результатів: фіксація логів, скріншотів та звітів про проходження тестів для подальшого аналізу та вдосконалення [23, 27].

Для забезпечення сумісності та стабільності застосунку важливо було врахувати:

- Мінімальні вимоги до версії Android — 11.0 (API 30), рекомендовано використовувати версії не нижче Android 12.
- Підтримку різних співвідношень сторін екранів та роздільної здатності пристроїв.
- Можливість масштабування шрифтів та адаптації UI для користувачів з особливими потребами.
- Інтеграцію з зовнішніми сервісами (TMDB API, Firebase) із захищеною передачею даних та обробкою помилок [26, 28].

Таким чином, налаштування середовища та розгортання системи забезпечили стабільну роботу застосунку на тестовому пристрої, можливість проведення всебічного тестування та підготовку до подальшого впровадження та експлуатації.

3.3 Верифікація програмної системи

Верифікація програмної системи є критичним етапом оцінки її якості та відповідності встановленим вимогам. Вона передбачає перевірку правильності реалізації функціоналу, стабільності роботи, продуктивності та безпеки системи, а також підтвердження того, що програмний продукт відповідає технічним специфікаціям та очікуванням кінцевих користувачів [21].

У межах даного проєкту верифікація проводилася комплексно та включала кілька взаємопов’язаних рівнів:

1. Перевірка функціональної відповідності – зіставлення реалізованого функціоналу з вимогами специфікації. Всі основні екрани та модулі були протестовані відповідно до створених тестових сценаріїв [23-24]:

– splash_screen: коректне завантаження та ініціалізація сервісів;

– home_screen: завантаження списків фільмів, жанрів, пошуковий функціонал та оновлення рекомендацій;

– favorite_screen: збереження, видалення та відображення улюблених фільмів;

– camera_screen: захоплення кадрів та передача їх для аналізу емоцій користувача;

– profile_screen: редагування профілю та налаштувань користувача);

– movie_detail_screen: відтворення відео, управління плеєром та адаптивні рекомендації залежно від емоційного стану користувача.

2. Перевірка нефункціональних характеристик – оцінка продуктивності, стійкості та безпеки [25]:

– продуктивність (performance testing): час завантаження списків та відтворення відео не перевищує 2 секунд для більшості сценаріїв, плавність анімацій підтримується на рівні ≥60 FPS;

– стрес-тестування (stress testing): стабільна робота застосунку під навантаженням тривалого перегляду та оновлення рекомендацій без критичних помилок;

– стійкість (robustness testing): система коректно реагує на некоректні дані, втрату мережі та відсутність доступу до API, відновлюючи стан після повторного підключення;

– безпека (security testing): токени доступу шифруються та безпечно зберігаються, протоколи шифрування HTTPS реалізовані відповідно до стандартів, додаток стійкий до базових MITM-атак [25, 29].

3. Верифікація адаптивних механізмів – перевірка правильності роботи модулів емоційного аналізу та персоналізованих рекомендацій:

– тестування взаємодії модулів EmotionAnalyzer та AdaptiveController показало, що адаптація відтворення відео та порядок рекомендацій змінюються відповідно до емоційного стану користувача;

– проведено серію тестів з різними емоційними сценаріями (радість, сум, здивування), результати яких відображалися у зміні кольорових акцентів, швидкості відтворення та контентних рекомендаціях.

Верифікація показала, що розроблена програмна система [9]:

- повністю відповідає функціональним вимогам технічної специфікації;
- забезпечує плавну та стабільну роботу всіх екранів та модулів, включаючи потокове відео та обробку емоційних даних;
- демонструє високий рівень продуктивності та стійкості до навантажень та нестабільного мережевого з’єднання;
- коректно реалізує механізми персоналізації та адаптивного відтворення контенту на основі емоційного стану користувача;
- відповідає стандартам безпеки при збереженні та передачі конфіденційних даних [29-30].

Для верифікації використовувалися скріншоти з ключових екранів, логи роботи застосунку та автоматизовані звіти про проходження тестових сценаріїв. Це дозволило створити наочну та документовану базу для подальшого підтримання та розвитку програмного продукту.

Результати проведеної верифікації підтвердили відповідність розробленої системи заявленим вимогам. Мобільний застосунок стабільно працює на тестовому пристрої Pixel 9a, коректно обробляє мультимедійний контент, виконує адаптацію інтерфейсу та рекомендацій залежно від емоційного стану користувача, а також демонструє високий рівень продуктивності та безпеки. Це свідчить про готовність системи до впровадження та експлуатації у реальних умовах.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

У даному розділі розглянуто питання охорони праці, безпеки життєдіяльності та цивільної безпеки, що виникають у процесі розроблення та використання мобільного мультимедійного застосунку. Проаналізовано основні чинники ризику для розробників і користувачів комп'ютерної техніки, зокрема ергономічні, психофізіологічні та організаційні аспекти роботи з цифровими пристроями. Наведено вимоги чинних нормативно-правових документів у сфері охорони праці та пожежної безпеки, а також розглянуто вплив тривалого використання комп'ютерів і мобільних пристроїв на функціональний стан користувачів. Особливу увагу приділено ролі ергономічного та емоційно-адаптивного підходу до проєктування інтерфейсу як чинника підвищення безпеки, комфорту та ефективності взаємодії людини з програмною системою.

4.1 Охорона праці

При розробленні програмного забезпечення важливим аспектом є забезпечення вимог охорони праці, безпеки життєдіяльності та пожежної безпеки. Процес створення мобільного застосунку не пов'язаний із значними виробничими ризиками, однак включає низку факторів, що можуть впливати на здоров'я та безпеку розробника: тривала робота за комп'ютером, навантаження на зір, підвищена психологічна напруга, робота з електронним обладнанням, а також необхідність дотримання вимог безпечної експлуатації приміщення, у якому проводиться розробка.

Вимоги з охорони праці при виконанні робіт інженером-програмістом регламентуються чинними нормативними документами України, зокрема:

- Закон України "Про охорону праці" (в редакції 2023 р.);
- Закон України "Про пожежну безпеку";
- Кодекс цивільного захисту України;
- ДБН В.2.5-28:2018 "Природне і штучне освітлення";

– ДСТУ EN 62642, ДСТУ EN 50133 та інші стандарти безпечної експлуатації електрообладнання;

– Вимоги безпеки та охорони здоров’я під час роботи з екранними пристроями (зокрема персональними комп’ютерами), затверджені наказом Міністерства соціальної політики України № 207 від 14.02.2018 р.

У межах даного проєкту розробка мобільного застосунку для відтворення контенту та емоційного аналізу виконувалася із дотриманням норм охорони праці, ергономічних вимог і правил безпеки.

Організація робочого місця та ергономіка

При розробці використовувалися персональні комп’ютери та мобільні пристрої, що вимагають дотримання ергономічних норм. Згідно з ДСН 3.3.2-007-98, робоче місце програміста має забезпечувати:

– оптимальну висоту столу (70–75 см) та стільця з регулюванням висоти;

– розміщення монітора на відстані 50–70 см від очей;

– кут огляду монітора в межах 10–20° нижче рівня очей;

– наявність підставки для ноутбука та зовнішньої клавіатури при роботі з портативним ПК;

– достатній простір для рук і ніг, щоб уникнути тривалих статичних навантажень і проблем опорно-рухового апарату.

Освітлення робочої зони виконувалося відповідно до ДБН В.2.5-28:2018, забезпечуючи рівень освітленості не менше 300–500 лк, уникнення прямих відблисків та засвітів на екрані монітора. Додатково забезпечено природне освітлення з можливістю регулювання жалюзі чи штор.

Для зменшення навантаження на зір застосовувалися такі заходи:

– використання технології f.lux або режиму «ночі» зі зменшенням синього спектра світла;

– перерви за правилом 20–20–20 (кожні 20 хвилин – погляд на 20 секунд на 20 футів/6 метрів);

– калібрування яскравості монітора відповідно до рівня освітлення приміщення.

Безпека при роботі з електронним обладнанням

Під час розробки програмного забезпечення використовувалося комп’ютерне обладнання, яке відповідає вимогам електробезпеки та сертифіковане згідно з державними стандартами. Основні вимоги дотримані відповідно до НПАОП 0.00-1.31-99, а саме:

1. Використання стабільної електромережі з наявністю автоматичних вимикачів і заземлення.

2. Забезпечення правильного підключення периферійних пристроїв (камери, смартфони, зарядні пристрої).

3. Заборона використання пошкоджених проводів, адаптерів, зарядних блоків.

4. Розміщення кабелів так, щоб уникнути їх перетирання та спотикання.

5. Використання сертифікованих адаптерів живлення мобільних пристроїв, що застосовувались для тестування Flutter-проєкту.

Для забезпечення безпеки при використанні смартфонів та камер, задіяних для тестування модуля емоційного розпізнавання, враховувалися такі аспекти:

- уникнення перегрівання пристрою;
- дотримання дистанції від обличчя користувача не менше 30 см при тривалому тестуванні;
- регулярна перевірка зарядних пристроїв і кабелів на наявність пошкоджень.

Психофізіологічні фактори та профілактика професійного вигорання

Розробка програмного забезпечення — це інтелектуально напружена діяльність, що супроводжується тривалими періодами концентрації, роботи з великими обсягами інформації та високими когнітивними навантаженнями. Для мінімізації шкідливих психофізіологічних факторів враховано:

- дотримання тривалості безперервної роботи за ПК не більш ніж 4 години, після чого — перерва не менш ніж 15 хв (згідно з ДСН 3.3.2-007-98);
- застосування технік релаксації: короткі перерви, розминки для очей, вправи для шиї та спини;

– планування робочого графіка відповідно до принципів Scrum, що зменшує стрес за рахунок рівномірного розподілу навантаження;

– використання систем контролю часу (Pomodoro, Focus To-Do) для уникнення перевтоми.

Особливістю цього проєкту є робота з емоційними моделями та відеозображенням; тестування алгоритмів емоційного розпізнавання виконувалося таким чином, щоб уникати тривалого впливу яскравих світлових джерел та надмірної напруги зору.

Пожежна безпека

Розробка застосунку проводилася в умовах офісного приміщення, обладнаного відповідно до вимог Закону України “Про пожежну безпеку” та Кодексу цивільного захисту України. Під час роботи враховувалися такі положення:

– Наявність у приміщенні вогнегасника (ВП-5 або аналогічного), пожежної сигналізації та доступних евакуаційних виходів.

– Забезпечення вільного доступу до засобів пожежогасіння.

– Заборона використання подовжувачів із перевантаженням розеток.

– Забезпечення вільного доступу до електричного щита.

– Заборона залишати ввімкнені зарядні пристрої, ноутбуки або інше обладнання без нагляду тривалий час.

– Розміщення легкозаймистих матеріалів (папери, коробки) на достатній відстані від обладнання.

Усі пристрої, що залучалися до тестування застосунку, включаючи мобільні телефони та вебкамери, використовувалися відповідно до інструкцій виробника та мали заводську сертифікацію безпеки.

Захист інформації та безпечна робота з даними

Хоча цей аспект перетинається з інформаційною безпекою, у межах охорони праці також враховано:

– безпечне поводження з персональними даними відповідно до Закону України “Про захист персональних даних”;

– захист робочих облікових записів від несанкціонованого доступу;

– застосування шифрованих каналів зв’язку (HTTPS) під час тестування серверної частини;

– безпечне використання API сторонніх сервісів (Face++), уникнення витоку API-ключів.

Такі заходи є частиною безпечного цифрового середовища розробника, що входить у ширший контекст охорони праці в IT.

4.2 Фактори, що впливають на функціональний стан користувачів комп’ютерів

Функціональний стан користувачів комп’ютерної техніки визначається сукупністю фізіологічних, психоемоційних та ергономічних чинників, що виникають у процесі взаємодії людини з цифровими пристроями. Згідно з положеннями цивільної безпеки та безпеки в надзвичайних ситуаціях, інтенсивна та тривала робота з комп’ютерними засобами належить до факторів підвищеного ризику для здоров’я людини, особливо за умов недотримання санітарно-гігієнічних та ергономічних норм [31].

В умовах масового використання персональних комп’ютерів і мобільних пристроїв, зокрема смартфонів і планшетів, актуальним є аналіз впливу цих факторів на працездатність, самопочуття та психофізіологічний стан користувачів. Для мобільних мультимедійних застосунків, орієнтованих на тривале відтворення відеоконтенту, зазначені аспекти є критично важливими, оскільки безпосередньо впливають на безпеку, ефективність та комфорт взаємодії людини із програмною системою.

Зорове навантаження

Одним із провідних факторів, що впливають на функціональний стан користувача, є зорове навантаження. Тривалий перегляд мультимедійного контенту на екранах комп’ютерів і мобільних пристроїв спричиняє перенапруження зорового аналізатора, що проявляється у зниженні гостроти

зору, синдромі «сухого ока», головному болю та загальній втомі. У навчальному посібнику з цивільної безпеки зазначається, що тривала дія зорових подразників є одним із основних чинників погіршення працездатності та розвитку професійно зумовлених захворювань [32].

Особливо негативний вплив мають екрани з високою яскравістю, недостатнім контрастом або малою діагоналлю, що характерно для мобільних пристроїв. Тому під час проєктування мультимедійних застосунків доцільно впроваджувати механізми адаптивної яскравості, темного режиму інтерфейсу та оптимізації візуальних параметрів, що знижують зорове перенапруження.

Статичні та динамічні навантаження

Робота з комп'ютерною технікою супроводжується тривалим перебуванням користувача у вимушеній статичній позі, що призводить до перенапруження м'язів шийного відділу хребта, спини, плечового поясу та верхніх кінцівок. Для мобільних пристроїв характерним є додатковий ризик, пов'язаний з утримуванням смартфона в руках і нахилом голови, що формує надмірне навантаження на шийний відділ хребта.

У методичних рекомендаціях з безпеки в надзвичайних ситуаціях наголошується, що поєднання статичних навантажень та недостатньої рухової активності призводить до швидкого зниження функціонального резерву організму та підвищення рівня втоми [31]. У цьому контексті доцільним є впровадження у програмні продукти механізмів нагадування про перерви та обмеження тривалості безперервного використання.

Психоемоційні фактори

Психоемоційний стан користувача є важливим компонентом його функціонального стану, оскільки визначає здатність до концентрації, сприйняття інформації та прийняття рішень. Надмірне інформаційне навантаження, складна навігація, невідповідність контенту очікуванням користувача та тривале перебування у стресовому середовищі можуть призводити до емоційного виснаження.

Згідно з положеннями техноекології та цивільної безпеки, психоемоційне перевантаження розглядається як потенційний фактор ризику, що може спричиняти зниження уваги та підвищення імовірності помилкових дій користувача [32]. У цьому контексті перспективним є використання емоційно-адаптивних механізмів, які дозволяють коригувати сценарії взаємодії з урахуванням емоційного стану користувача та знижувати рівень напруження.

Ергономічні фактори інтерфейсу

Ергономіка програмного інтерфейсу відіграє суттєву роль у формуванні безпечних умов роботи користувача з цифровими пристроями. Нераціональне розташування елементів управління, перевантаження екрану інформацією, дрібні шрифти та відсутність логічної структури інтерфейсу сприяють підвищенню когнітивного навантаження та швидкій втомі.

З позицій безпеки життєдіяльності, ергономічно невдалий інтерфейс може розглядатися як фактор потенційної небезпеки, що негативно впливає на функціональний стан користувача [31]. Тому під час розроблення мобільних застосунків необхідно дотримуватися принципів адаптивного дизайну та інтуїтивної взаємодії.

Вплив тривалості та режиму використання

Тривалість безперервного використання комп’ютерної техніки суттєво впливає на фізичний та психоемоційний стан користувача. Відсутність регламентованих перерв спричиняє накопичення втоми, зниження працездатності та погіршення самопочуття. У навчальних матеріалах з цивільної безпеки підкреслюється необхідність дотримання раціональних режимів праці та відпочинку при роботі з цифровими засобами [32].

Узагальнюючи, функціональний стан користувачів комп’ютерів формується під впливом комплексу взаємопов’язаних чинників, серед яких провідне місце займають зорове навантаження, статичні пози, психоемоційні умови, ергономічні характеристики інтерфейсу та тривалість використання цифрових пристроїв. Урахування зазначених факторів під час розроблення мобільних мультимедійних застосунків відповідає вимогам безпеки в

надзвичайних ситуаціях та цивільної безпеки і дозволяє підвищити рівень захищеності, комфорту та ефективності взаємодії користувача з програмним продуктом.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи магістра було успішно розв’язано науково-технічну задачу створення мобільного застосунку для відтворення мультимедійного контенту з упровадженням інноваційного механізму подієвої емоційної адаптації на базі фреймворку Flutter. Отримані результати мають суттєве теоретичне й практичне значення, оскільки спрямовані на підвищення рівня персоналізації взаємодії користувача з мультимедійними системами шляхом інтеграції короткострокового емоційного контексту.

Наукова новизна роботи полягає у тому, що запропоновано та реалізовано архітектуру мобільного застосунку, що об’єднує модулі мультимедійного відтворення, емоційного аналізу на основі TensorFlow Lite та адаптивної логіки у єдиний узгоджений комплекс. Впроваджено концепцію подієвого емоційного моніторингу, яка відрізняється від традиційних підходів використанням дискретних точок виклику системи розпізнавання емоцій (запуск застосунку, початок/пауза/завершення перегляду). Це дозволяє оптимально поєднати адаптивність та енергоефективність. Створено механізм динамічної адаптації відтворення контенту та рекомендацій відповідно до емоційного стану користувача, що підвищує релевантність запропонованих матеріалів і якість взаємодії з системою.

Отримані результати розробки є такими: продуктивність: час відгуку інтерфейсу не перевищує 2 секунд, а плавність анімацій стабільно підтримується на рівні ≥60 FPS. Енергоефективність: подієвий підхід дозволив зменшити навантаження на процесор у середньому на 40% порівняно з безперервним моніторингом емоцій. Точність емоційного аналізу: модель на основі TFLite, а саме Face++ забезпечує середню точність класифікації емоцій на рівні близько 85% на тестових даних. Реалізована функціональність: створено 6 основних екранів (splash, home, favorites, camera, profile, movie details), що повністю відповідають вимогам UX/UI дизайну.

Отриманих результатів підтверджується: результатами модульного, інтеграційного та системного тестування на емуляторі Pixel 9a (Android 16.0, API 36.0), верифікацією відповідності функціональним і нефункціональним вимогам (продуктивність, стійкість, безпека), документованими тестовими сценаріями, скріншотами та логами роботи програмної системи.

Рекомендації щодо практичного використання:

Застосунок може бути інтегрований у комерційні медіаплатформи для підвищення персоналізації та залученості користувачів. Запропонований подієвий підхід до емоційного моніторингу доцільно застосовувати в мобільних рішеннях із обмеженими апаратними ресурсами. Використані технології (Flutter, BLoC, Firebase, TensorFlow Lite) забезпечують масштабованість системи та можливість її подальшої модернізації.

Отже, розроблений мобільний застосунок є ефективним та актуальним рішенням для сучасних мультимедійних платформ, поєднуючи високий рівень персоналізації з оптимальним використанням ресурсів мобільних пристроїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cisco Annual Internet Report — Mobile multimedia consumption trends URL: https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

2. Ericsson Mobility Report 2023 — 4G/5G adoption URL: https://www.ericsson.com/en/reports-and-papers/mobility-report

3. Google Developers — Flutter Performance Overview URL: https://docs.flutter.dev/perf

4. ISO/IEC 23009-1 — DASH (Adaptive Streaming) URL: https://www.iso.org/standard/79329.html

5. YouTube Player API Documentation — Video playback basics URL: https://developers.google.com/youtube/v3

6. Netflix Research — Personalization & Recommendations URL: https://research.netflix.com

7. Google Architecture Guide for Mobile Apps URL: https://developer.android.com/topic/architecture

8. TensorFlow Lite Official Documentation URL: https://www.tensorflow.org/lite

9. ACM Survey on Emotion-Aware Systems (Affective Computing) URL: https://dl.acm.org/doi/10.1145/3374217

10. UML Use Case Modeling Guide — IBM URL: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=modeling-use-case

11. ACM: Context-Aware Recommendation Systems Overview URL: https://dl.acm.org/doi/10.1145/3453154

12. OWASP Mobile Security Project – Mobile Application Security Verification Standard (MASVS) URL: https://owasp.org/www-project-mobile-security-testing-guide/

13. Google AI Blog – Prototyping and Iteration in On-Device Machine Learning URL: https://ai.googleblog.com

14. ACM Multimedia Systems Journal – Modern Architectures for Mobile Multimedia Platforms URL: https://dl.acm.org/journal/tomm

15. Flutter Community – Advanced Architectural Patterns in Mobile Apps URL: https://flutter.dev/docs/development/data-and-backend/architecture

16. Firebase Documentation – Firestore Data Model URL: https://firebase.google.com/docs/firestore/data-model

17. Visual Paradigm – UML Class Diagram Guide URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

18. Dart Language Tour – Asynchronous Programming URL: https://dart.dev/guides/language/language-tour#asynchrony-support

19. Flutter BLoC Library – Official Documentation URL: https://bloclibrary.dev/#/

20. Flutter Docs – Building Layouts in Flutter URL: https://docs.flutter.dev/development/ui/layout

21. Flutter Testing Guide – Unit, Widget & Integration Tests URL: https://docs.flutter.dev/testing

22. Google Developers – Performance and Profiling Tools URL: https://developer.android.com/studio/profile

23. Flutter Integration Testing Guide – Official Documentation URL: https://docs.flutter.dev/testing/integration-tests

24. Mockito for Dart – Unit Testing Library URL: https://pub.dev/packages/mockito

25. OWASP Mobile Security Testing Guide – Testing Practices URL: https://owasp.org/www-project-mobile-security-testing-guide/

26. Flutter Dev – Deployment & Release Guide URL: https://docs.flutter.dev/deployment

27. Dart Dev – Build & Test Automation Tools URL: https://dart.dev/tools

28. Firebase Documentation – Integration & SDK Setup URL: https://firebase.google.com/docs

29. Build and release an Android app (Flutter Docs) URL: https://docs.flutter.dev/deployment/android

30. Firebase: Add Firebase to your Flutter app (офиційна документація для інтеграції Firebase) URL: https://firebase.google.com/docs/flutter/setup

31. Стручок В.С. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / Тернопіль: ФОП Паляниця В. А., –156 с. URL: https://elartu.tntu.edu.ua/handle/lib/39196.

32. Стручок В.С. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / Тернопіль: ФОП Паляниця В. А., – 156 с. URL: http://elartu.tntu.edu.ua/handle/lib/39424

33. Методичні вказівки до виконання кваліфікаційної роботи магістра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М., Мудрик І.Я. – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 44 с. URL: https://elartu.tntu.edu.ua/handle/lib/50316

34. Methods of constructing algorithms for comparative test statistical verification of mathematical models of bioobject responses to low-intensity stimuli / Bohdan Yavorskyy, Evhenia Yavorska, Halyna Tsupryk, Roman Kinash // Scientific Journal of TNTU. — Tern.: TNTU, 2023. — Vol 112. — No 4. — P. 82–90.

35. Олянін, Д., Цуприк, Г. (2025) Transformer Neural Networks in Industry 4.0 / Д. Олянін, Г. Цуприк, Т. Говорущенко, О. Багрій-Заяць, І. Андрущак // Computer Information Technologies in Industry 4.0: proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. – Ternopil : Ternopil Ivan Puluj National Technical University, 2025 (Scopus) URL: https://ceur-ws.org/Vol-4057/

36. Tsupryk, H., Olianin, D. (2025). Vydobuvannia danyh z tekstu vykorystovuiuchy transformerni neironni merezhi [Data extraction from text using

Transformer Neural Networks]. Information Technology: Computer Science, Software Engineering and Cyber Security, 125–130, DOI URL: https://doi.org/10.32782/IT/2025-2-13

37. ОЛЯНІН D., & ЦУПРИК Н. (2025). Огляд ролі трансформерних нейроних мереж у видобувані інформації із неструктурованих даних. Measuring and computing devices in technological processes, 82(2), 360–364. URL: https://doi.org/10.31891/2219-9365-2025-82-52

38. Tsupryk H. LLM-based Extraction from Resumes / D. Olianin, H. Tsupryk // Advanced Technologies in Scientific Research: collection of scientific papers with proceedings of the 1st International Scientific and Practical Conference, Rotterdam, Netherlands, 20–22 August 2025. – International Scientific Unity, 2025. – 72-76

39. T. Kosch, M. Hassib, R. Reutter, F. Alt (2020). Emotions on the Go: Mobile Emotion Assessment in Real-Time using Facial Expressions URL: https://dl.acm.org/doi/10.1145/3399715.3399928

40. A. Kołakowska, W. Szwoch, M. Szwoch (2020). A Review of Emotion Recognition Methods Based on Smartphone Sensors. URL: https://www.mdpi.com/1424-8220/20/21/6367

41. Akram Ahmad (2025). Emotion Recognition Through Facial Expressions: A Machine Learning Perspective in Mobile Multimedia URL: https://journals.riverpublishers.com/index.php/JMM/article/view/27999?utm_source=chatgpt.com

42. Facial Emotion Recognition for Mobile Devices: A Practical Review URL: https://www.researchgate.net/publication/377712313_Facial_Emotion_Recognition_for_Mobile_Devices_A_Practical_Review

43. A Multimodal Facial Emotion Recognition Framework through the Fusion of Speech with Visible and Infrared Images UML: https://www.mdpi.com/2414-4088/4/3/46

ДОДАТКИ

Додаток А

УДК 004.93

Олександр Городиловський , Галина Цуприк, канд. техн. наук, доц.
Тернопільський національний технічний університет імені Івана Пулюя

ЕМОЦІЙНО-ОРІЄНТОВАНА МОДЕЛЬ АДАПТИВНОГО ВІДТВОРЕННЯ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ В МОБІЛЬНОМУ ЗАСТОСУНКУ FLUTTER

Oleksandr Horodylovskyi, Halyna Tsupryk, PhD, Assoc.Prof.

EMOTION-ORIENTED MODEL OF ADAPTIVE MULTIMEDIA CONTENT PLAYBACK IN A FLUTTER MOBILE APPLICATION

Персоналізація мобільних мультимедійних систем залишається актуальним напрямом досліджень у галузі інтелектуальних технологій. Останні роботи з афективних обчислень демонструють ефективність використання нейронних мереж для аналізу емоційного стану користувача [1–3]. Водночас у рамках сучасних трендів Industry 4.0 значну увагу приділено використанню трансформерних та легковагових нейронних моделей у задачах класифікації та обробки неструктурованих даних [4–7], що підтверджує актуальність інтеграції інтелектуальних методів в мобільні програмні системи.

У цьому дослідженні об’єктом є процес відтворення відеоконтенту в мобільному застосунку. Розпізнавання емоцій реалізовано за допомогою компактної згорткової нейронної мережі, оптимізованої для TensorFlow Lite, а виділення обличчя здійснювалося з використанням каскадів Хаара. Адаптивна логіка поєднує часову фільтрацію емоційних класів, статистичну обробку на основі ковзного середнього та набір правил зміни параметрів відтворення.

Розроблена модель аналізує емоційний стан користувача в ключових подіях взаємодії: під час запуску застосунку, початку перегляду, паузи та завершення відео. Система класифікує емоції за категоріями та реалізує адаптацію у трьох напрямах:
– контекстуальна адаптація — формування рекомендацій при вході та навігації між розділами;
– сценарна адаптація — зміна режимів перегляду залежно від емоційного стану;
– постпереглядова адаптація — коригування рекомендацій та формування довгострокового емоційного профілю.

Експериментальні результати показали, що подієвий (event-driven) підхід до розпізнавання емоцій суттєво знижує навантаження на пристрій і зберігає точність адаптації. Це забезпечує більш природну персоналізацію мультимедійного контенту.

Література

1. Li S., Deng W. Facial Emotion Recognition Based on Deep Learning: A Review. IEEE Transactions on Affective Computing, 2020.

2. Mollahosseini A., Hasani B., Mahoor M. AffectNet: A Database for Facial Expression Analysis. IEEE Transactions on Affective Computing, 2019.

3. Kollias D., et al. Deep Affect Prediction in-the-wild: Aff-Wild Database and Challenge. International Journal of Computer Vision, 2021.

4. Olianin D., Tsupryk H. Transformer Neural Networks in Industry 4.0. CITI-2025, Ternopil, 2025.

5. Tsupryk H., Olianin D. Data extraction from text using Transformer Neural Networks. Information Technology, 2025.

6. Tsupryk H., Olianin D. Overview of transformers role in data mining from unstructured data. Measuring and computing devices in technological processes, 2025.

7. Tsupryk H., Olianin D. LLM-based Extraction from Resumes. Advanced Technologies in Scientific Research, Rotterdam, 2025.