

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

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка методу та програмних засобів реалізації мобільної системи
підбору відеоконтенту "Topical Films"»

Виконав: студент IV курсу
групи ЗПІ-206
спеціальності

121 – Інженерія програмного забезпечення

(шифр і назва напрямку підготовки, спеціальності)

Бабійчук І.С.

(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.

(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ Колесник І. С.

(прізвище та ініціали)

Допущено до захисту
Зав. кафедри

« 10 » червня 2024 р.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

ЗАВДАННЯ НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Бабійчуку Івану Сергійовичу

1. Тема роботи – «Розробка методу та програмних засобів реалізації мобільної системи підбору відеоконтенту "Tropical Films"».

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Термін подання студентом роботи: 3 червня 2024.

3. Вихідні дані до роботи: середовище розробки – Android Studio; мови розробки – Kotlin, XML, Java; операційна система – Android.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз розвитку мобільних систем для підбору відеоконтенту та постановка задач дослідження; розробка методу, моделі та алгоритмів програмного продукту; розробка програмних модулів реалізації мобільної системи для підбору відеоконтенту; тестування програми; висновки; список використаних джерел; додатки.

5. титульний слайд; актуальність розробки; мета, предмет та об'єкт дослідження; постановка задач; новизна; практична цінність отриманих результатів; існуючі аналоги; аналіз аналогів; Метод персоналізованого підбору

відеоконтенту (ч.1); Метод персоналізованого підбору відеоконтенту (ч.2); Блок-схема алгоритму роботи методу персоналізованого підбору відеоконтенту; Модель системи у вигляді діаграми діяльності; Блок-схема алгоритму роботи мобільної системи підбору відеоконтенту; розробка графічного інтерфейсу; використані технології; функціонал системи; демонстрація роботи; апробація публікація результатів роботи; висновки; фінальний слайд.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	Войтко В. В., к.т.н., доцент кафедри ПЗ	13.03.24 <i>Войтко</i>	03.06.24 <i>Войтко</i>

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку мобільних систем для підбору відеоконтенту та постановка задач дослідження	14.03.24 – 24.03.24	<i>Виконано</i>
2	Розробка методу, моделі та алгоритмів програмного продукту	25.03.24 – 15.04.24	<i>Виконано</i>
3	Розробка програмних модулів реалізації мобільної системи для підбору відеоконтенту	15.04.24 – 30.04.24	<i>Виконано</i>
4	Тестування програми	01.05.24 – 10.05.24	<i>Виконано</i>
5	Оформлення матеріалів до захисту БДР	11.05.24 – 30.05.24	<i>Виконано</i>

Студент

Бабійчук І.С.
(підпис)

Бабійчук І.С.
(прізвище та ініціали)

Керівник бакалаврської дипломної роботи

Войтко В.В.
(підпис)

Войтко В.В.
(прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 62 сторінки формату А4, на яких є 28 рисунка, 3 таблиці, список використаних джерел містить 27 найменувань.

У бакалаврській дипломній роботі було проведено аналіз сервісів для підбору відеоконтенту. Проведено аналіз аналогів, визначено їх переваги і недоліки та доведено актуальність розробки власної системи.

Мобільна система "Topical Films" є інноваційним рішенням у сфері підбору відеоконтенту, спрямованим на задоволення індивідуальних потреб користувачів. Цей проєкт передбачає розробку ефективних методів та засобів для автоматизованого відбору та пропозиції відеоматеріалів на основі інтересів користувачів та актуальних тем. Також надає можливість завантажити відеоматеріали собі на пристрій для подальшого перегляду в оф-лайн режимі.

Основні завдання включають аналіз інтересів аудиторії, розробку алгоритмів рекомендацій, створення зручного інтерфейсу, інтеграцію з відео платформами та забезпечення безпеки даних.

Підходи, що використовуються у розробці, враховують сучасні тенденції у сфері аналітики даних. Очікується, що мобільна система "Topical Films" стане популярним інструментом для споживачів відео розваг, забезпечуючи їм доступ до різноманітного та персоналізованого контенту просто через їхні мобільні пристрої.

Додаток розроблено з використанням мови Kotlin та середовища AndroidStudio.

Ключові слова: мобільна система, підбір відеоконтенту.

ABSTRACT

The bachelor thesis consists of 62 pages of A4 format, on which there are 28 figures, 3 tables, the list of used sources contains 27 items.

In the bachelor thesis, an analysis of services for the selection of video content was carried out. An analysis of analogues was carried out, their advantages and disadvantages were determined, and the relevance of developing one's own system was proven.

The "Topical Films" mobile system is an innovative solution in the field of video content curation, aimed at satisfying individual user needs. This project involves the development of efficient methods and tools for automated selection and recommendation of video materials based on users' interests and current topics. It also provides an opportunity to download video materials to your device for further viewing in offline mode.

Key tasks include audience interest analysis, recommendation algorithm development, user-friendly interface creation, integration with video platforms, and data security provision.

The approaches used in the development take into account modern trends in the field of data analytics. It is expected that the "Topical Films" mobile system will become a popular tool for entertainment consumers, providing them with access to diverse and personalized content directly through their mobile devices.

The application is developed using the Kotlin language and AndroidStudio environment.

Keywords: mobile system, selection of video content.

ЗМІСТ

ВСТУП.....	3
1 АНАЛІЗ РОЗВИТКУ МОБІЛЬНИХ СИСТЕМ ДЛЯ ПІДБОРУ ВІДЕОКОНТЕНТУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ.....	8
1.1 Аналіз стану мобільних систем для підбору відеоконтенту	8
1.2 Порівняльний аналіз аналогів	9
1.3 Аналіз методів розв’язання задачі	14
1.4 Постановка задач для розробки мобільної системи для підбору відеоконтенту	17
1.5 Висновки.....	17
2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ	19
2.1 Аналіз вхідних даних системи	19
2.2 Розробка моделі системи	21
2.3 Розробка методу персоналізованого підбору відеоконтенту	22
2.4 Розробка алгоритмів роботи системи	27
2.5 Аналіз функціоналу системи для завантаження відеоконтенту	29
2.6 Висновки.....	31
3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ МОБІЛЬНОЇ СИСТЕМИ ДЛЯ ПІДБОРУ ВІДЕОКОНТЕНТУ	33
3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення.....	33
3.2 Розробка модуля відображення контенту у мобільній системі підбору відеоконтенту	36
3.3 Розробка модуля для кешування відеоконтенту в мобільній системі підбору відеоконтенту	38
3.4 Розробка модуля для завантаження відеоконтенту.....	39
3.5 Розробка інтерфейсу програмного додатку	45
3.6 Висновки.....	52
4 ТЕСТУВАННЯ ПРОГРАМИ.....	53
4.1 Тестування системи.....	53
4.2 Розробка інструкції користувача.....	56
4.3 Висновки.....	57
ВИСНОВКИ	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	60
ДОДАТКИ	63
Додаток А – Технічне завдання.....	Ошибка! Закладка не определена.
Додаток Б – Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень	Ошибка! Закладка не определена.
Додаток В – Лістинг програми	68
Додаток Г – Ілюстративний матеріал	87

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному інформаційному суспільстві, де величезна кількість відеоконтенту доступна в онлайн-просторі, проблема відбору цікавого та відповідного смакам користувача матеріалу стає все більш актуальною та складною. За даними статистики, кількість відеоматеріалів, доступних в інтернеті, зростає експоненційно, що призводить до збільшення складнощів у пошуку контенту, який відповідає індивідуальним уподобанням кожного користувача та враховує його потреби та інтереси [1].

У такому контексті, розробка методу та засобів реалізації мобільної системи підбору відеоконтенту "Topical Films" набуває особливої ваги. Ця система не лише відповідає актуальним потребам сучасного споживача відеоконтенту, але і вирішує проблему перенасичення інформацією шляхом забезпечення персоналізованого підходу до підбору контенту. Розробка інноваційних методів та засобів дозволить ефективно відбирати відеоматеріали, які максимально відповідають індивідуальним уподобанням кожного користувача, що в свою чергу сприятиме покращенню якості користування відеоконтентом та задоволенню його потреб.

Завдяки мобільній системі "Topical Films" користувачі матимуть можливість швидко та зручно знаходити відеоматеріали, які відповідають їхнім індивідуальним уподобанням та інтересам. Реалізація такого проєкту є кроком вперед у вдосконаленні інтернет-середовища та розширенні можливостей користувачів у споживанні відеоконтенту, що відповідає сучасним вимогам і тенденціям розвитку інформаційних технологій, тому актуальною є розробка власної мобільної системи підбору відеоконтенту.

Потрібно створити інноваційну та функціональну мобільну систему, яка враховуватиме індивідуальні потреби кожного користувача та забезпечуватиме персоналізований підбір відеоконтенту. В сучасному інформаційному суспільстві, де індивідуальність та персоналізація стають ключовими аспектами, мобільна система підбору відеоконтенту "Topical Films" вирішує ці проблеми, надаючи кожному користувачеві унікальний досвід споживання медіа.

Підходящі рекомендації є ключовим елементом у поліпшенні взаємодії користувача з мобільною системою. Шляхом аналізу даних про перегляди, вподобання та інші дії користувачів, система може зрозуміти їхні індивідуальні потреби та надати рекомендації, які максимально відповідають їхнім очікуванням. Такий підхід допомагає зробити процес пошуку та споживання відеоконтенту більш ефективним та задовільним для кожного користувача.

Крім того, у мобільному додатку "Topical Films" користувачі матимуть можливість завантажувати підібраний для них відеоконтент на пристрій для подальшого перегляду в офлайн режимі. Це дозволяє забезпечити доступ до улюблених відеоматеріалів навіть без підключення до інтернету, що є особливо зручним для тих, хто часто подорожує або має обмежений доступ до мережі [2]. Таким чином, користувачі можуть насолоджуватися персоналізованим відеоконтентом у будь-який час та в будь-якому місці, що робить використання додатку ще більш зручним та привабливим.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є покращення можливостей підбору відеоконтенту шляхом розробки та використання спеціалізованого програмного продукту, який дозволяє проводити аналіз переглядів користувача, визначати їх вподобання та надавати рекомендації й здійснювати пошук потрібного відеоконтенту, що дозволяє персоналізувати процес підбору відеоконтенту для потреб конкретного користувача.

Задачі дослідження:

- розробити метод і модель системи;
- розробити алгоритми, що базуються на аналізі великих обсягів даних;
- урахувати історію переглядів та уподобань користувача;
- використати сучасні технології штучного інтелекту для підвищення точності рекомендацій;

- забезпечити виробництво індивідуалізованих рекомендацій для кожного користувача;
- надати можливість користувачам завантажувати підібраний для них відеоконтент на пристрій для подальшого перегляду в офлайн режимі;
- підвищити якість обслуговування та забезпечити задоволення від використання мобільної системи;
- забезпечити взаємодію користувача з мобільною системою за допомогою підходящих рекомендацій;
- аналізувати дані про перегляди, вподобання та інші дії користувачів для надання індивідуальних рекомендацій.

Об’єкт дослідження – процес розробки та впровадження мобільної системи підбору відеоконтенту "Topical Films".

Предмет дослідження – методи та засоби, які дозволять ефективно відбирати та рекомендувати відеоконтент користувачам з урахуванням їхніх індивідуальних уподобань та інтересів.

Методи дослідження. У процесі досліджень використовувались:

- методи та засоби реалізації мобільних застосунків мовою Kotlin з використанням Android SDK для розробки логіки системи та графічного інтерфейсу користувача;
- методи побудови UML-діаграм для моделювання основних функціональних компонентів системи та їх взаємодії;
- методи оптимізації коду та використання бібліотек Retrofit і Room для забезпечення ефективного виконання HTTP-запитів і роботи з локальною базою даних;
- методи захисту даних за допомогою шифрування завантажених відеофайлів для забезпечення безпеки користувачів;
- методи тестування для перевірки працездатності та відповідності мобільної системи поставленим вимогам.

Новизна отриманих результатів:

1. Подальшого розвитку набув метод персоналізованого підбору відеоконтенту, який, на відміну від існуючих, враховує індивідуальні вподобання користувачів, історію переглядів та інші дії, що дозволяє надавати персоналізовані рекомендації, які максимально відповідають очікуванням користувачів та підвищують їх задоволення від використання додатку.

2. Подальшого розвитку набула модель зберігання та захисту відеоконтенту, яка, на відміну від існуючих, включає шифрування завантажених відеофайлів для захисту від несанкціонованого доступу та забезпечує можливість завантаження контенту для перегляду в офлайн-режимі, що значно підвищує зручність роботи користувачів та забезпечує високу доступність відеоконтенту.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в можливості використання розробленої програми, щоб значно заощадити час користувачів, які шукають відеоконтент, а також дозволить розширити їхні інтереси через персоналізовані рекомендації. Крім того, система може бути використана в навчальних та рекламних цілях, що робить її значним активом у сучасному інформаційному середовищі.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані особисто автором. У друкованій праці[3], опублікованій у співавторстві, автору належать такі результати: розробка методу та програмних засобів для персоналізованого підбору та завантаження відеоконтенту на платформі Android, включаючи реалізацію алгоритмів для рекомендованої системи, розробку модуля для завантаження відеоконтенту та забезпечення безпеки даних за допомогою шифрування відеофайлів.

Апробація матеріалів бакалаврської дипломної роботи Матеріали бакалаврської дипломної роботи доповідалися на конференції: LIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2024).

Публікації. За тематикою дослідження опубліковано 1 наукову працю у збірнику матеріалів LIII Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2024) [3].

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містять 27 найменувань, 4 додатка. Робота містить 28 ілюстрацій та 3 таблиці.

1 АНАЛІЗ РОЗВИТКУ МОБІЛЬНИХ СИСТЕМ ДЛЯ ПІДБОРУ ВІДЕОКОНТЕНТУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану мобільних систем для підбору відеоконтенту

Зважаючи на швидкий розвиток технологій та зростаючий попит на відеоконтент серед користувачів мобільних пристроїв, мобільні системи для підбору відеоконтенту пройшли значний шлях еволюції протягом останніх кількох років. Ось деякі ключові аспекти цього розвитку:

1. Персоналізовані рекомендації. Сучасні мобільні системи для підбору відеоконтенту активно використовують алгоритми машинного навчання та штучного інтелекту для аналізу виборів користувачів та створення персоналізованих рекомендацій. Це дозволяє кожному користувачеві отримувати рекомендації, які відповідають їхнім індивідуальним смакам та інтересам.
2. Зростання кількості контенту та постачальників. З кожним роком зростає кількість вмісту, доступного для перегляду через мобільні додатки. На ринку з'являються нові сервіси стрімінгового відео, а також вміст від традиційних медіа-компаній, інтернет-контент-провайдерів, аматорських відеоблогерів та ін.
3. Технологічні інновації. Завдяки прогресу в області штучного інтелекту, обробки великих обсягів даних та комп'ютерного зору, мобільні системи для підбору відеоконтенту стають все більш інтелектуальними та здатними адаптуватися до індивідуальних потреб користувачів.
4. Мультимедійні можливості мобільних пристроїв. З розвитком мобільних пристроїв з'явилися нові можливості для споживання відеоконтенту, такі як підтримка високої роздільності (HD, 4K), покращені звукові технології та розширений діапазон відображення кольорів.
5. Зростання мобільної аудиторії. Кількість користувачів, які переглядають відео на мобільних пристроях, стабільно зростає. Це створює нові

можливості для рекламодавців і контент-постачальників, а також вимагає покращення в якості і зручності доступу до відеоконтенту.

Загалом, розвиток мобільних систем для підбору відеоконтенту свідчить про постійні зусилля розробників удосконалювати технології, розширювати вміст та покращувати користувацький досвід з метою задоволення зростаючих потреб аудиторії.

1.2 Порівняльний аналіз аналогів

Поява мобільних систем для підбору відеоконтенту є відображенням сучасних тенденцій у споживанні медіа. Відтак, спостерігається зростання популярності цих систем, оскільки вони відповідають зростаючим потребам споживачів у зручному та доступному способі споживання контенту.

Кожна з цих систем має свої унікальні особливості, що роблять їх привабливими для різних груп користувачів.

Деякі системи акцентують увагу на індивідуальному підборі контенту з урахуванням персональних вподобань користувача, забезпечуючи надзвичайно персоналізований досвід. Інші системи можуть надавати широкий спектр контенту з різних джерел та жанрів, спрощуючи процес вибору та забезпечуючи більш широкий вибір для користувачів.

Наявність різних систем дозволяє користувачам вибирати той, що найбільше відповідає їхнім потребам та вподобанням. Однак, вибір оптимальної системи може бути складним завданням, оскільки потреби користувачів можуть змінюватися з часом та в залежності від конкретних ситуацій.

Отже, мобільні системи для підбору відеоконтенту представляють собою важливий елемент сучасного медіа-ландшафту, пристосований до потреб та вимог сучасних споживачів. Їхнє різноманіття та унікальні особливості дозволяють задовольняти різноманітні потреби та вподобання користувачів, роблячи процес споживання відеоконтенту більш зручним та цікавим.

До найбільш відомих рішень відносять:

- YouTube;
- Netflix;
- Hulu;
- Amazon Prime Video;
- Disney+.

YouTube [4] (рис.1.1) – це велика онлайн-платформа для безкоштовного перегляду, завантаження та розповсюдження відео. Користувачі можуть знайти великий асортимент відеоконтенту, від коротких відеороликів і влогів до повноформатних фільмів та серіалів. Платформа пропонує різноманітні категорії контенту, включаючи музику, розваги, освіту, спорт, науку і багато іншого. Що робить YouTube унікальним, це велика кількість користувачів-контент-творців, що створюють оригінальний вміст у будь-якій тематиці.

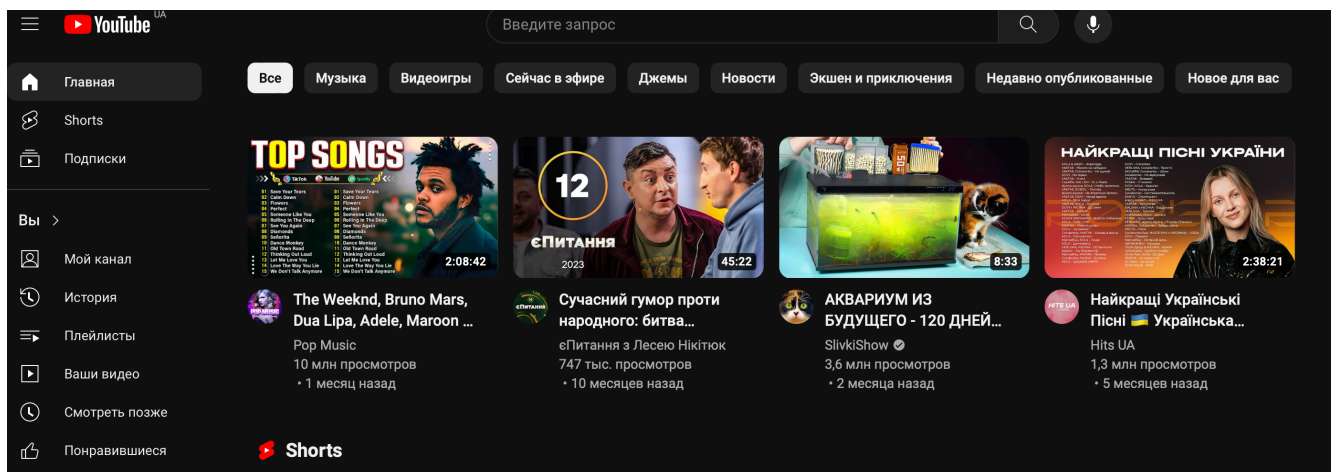


Рисунок 1.1 – Приклад роботи платформи YouTube

Інший приклад – Netflix [5] (рис.1.2) провідна світова платформа для стрімінгового відео, яка пропонує широкий вибір фільмів, серіалів і оригінального контенту. Підписники можуть насолоджуватися безліччю вмісту у різних жанрах, включаючи драму, комедію, жахи, наукову фантастику та багато іншого. Однією з ключових особливостей Netflix є його власний оригінальний контент, який неможливо побачити на інших платформах. Це включає у себе серіали, фільми, документальні фільми і комедійні шоу, створені спеціально для Netflix.

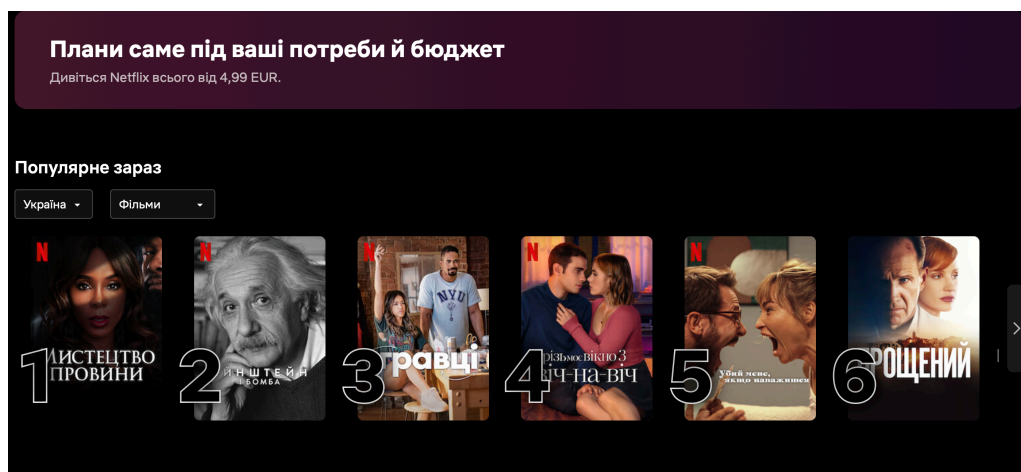


Рисунок 1.2 – Приклад роботи платформи Netflix

Hulu [6] (рис. 1.3) – це платформа для стрімінгового відео, яка пропонує широкий вибір телевізійних шоу, оригінальних серіалів, фільмів і документальних фільмів. Він також має можливість перегляду епізодів телешоу в день їхнього виходу. Однією з унікальних особливостей Hulu є його співпраця з телевізійними мережами, яка дозволяє йому швидко надавати доступ до нових епізодів телешоу.

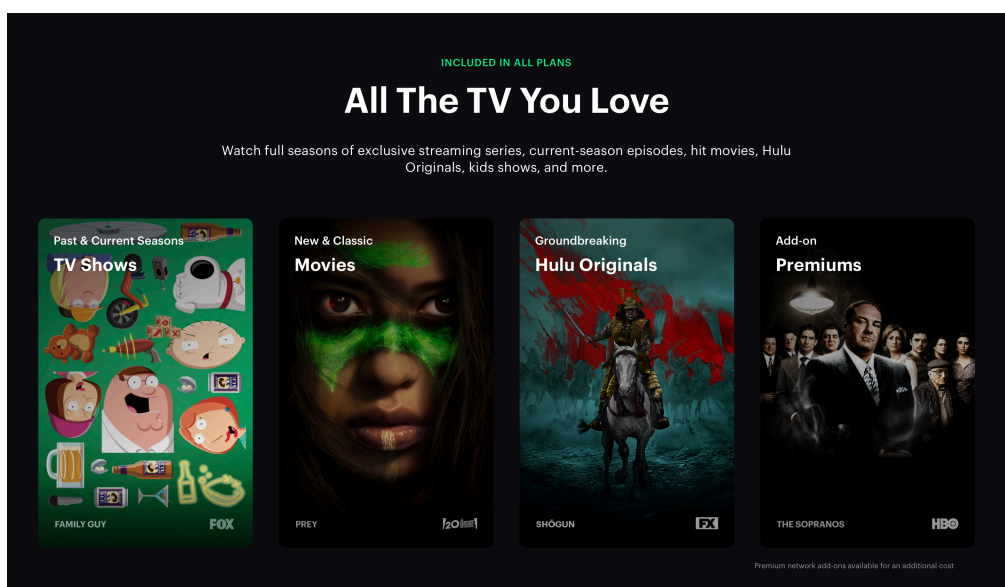


Рисунок 1.3 – Приклад роботи платформи Hulu

Amazon Prime Video [7] (рис. 1.4) – це частина платформи Amazon Prime, яка пропонує широкий вибір фільмів, серіалів і оригінального вмісту. Однією з

унікальних особливостей Amazon Prime Video є те, що вона включена в Amazon Prime, що дозволяє підписникам насолоджуватися безкоштовною доставкою, доступом до Amazon Music та іншими перевагами.

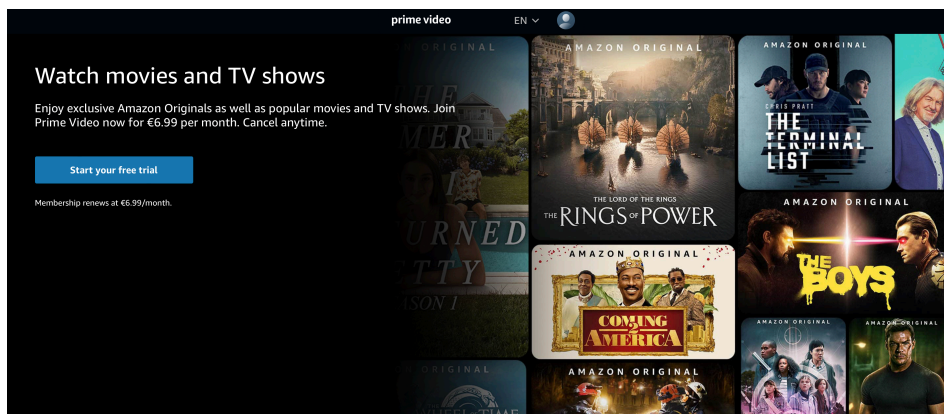


Рисунок 1.4 – Приклад роботи платформи Amazon Prime Video

Disney+ [8] (рис. 1.5) – це стрімінговий сервіс, який містить в собі широкий вибір вмісту від Disney, Pixar, Marvel, Star Wars та National Geographic. Однією з унікальних особливостей Disney+ є доступ до ексклюзивного оригінального контенту, такого як серіали та фільми, які створені спеціально для платформи Disney+.

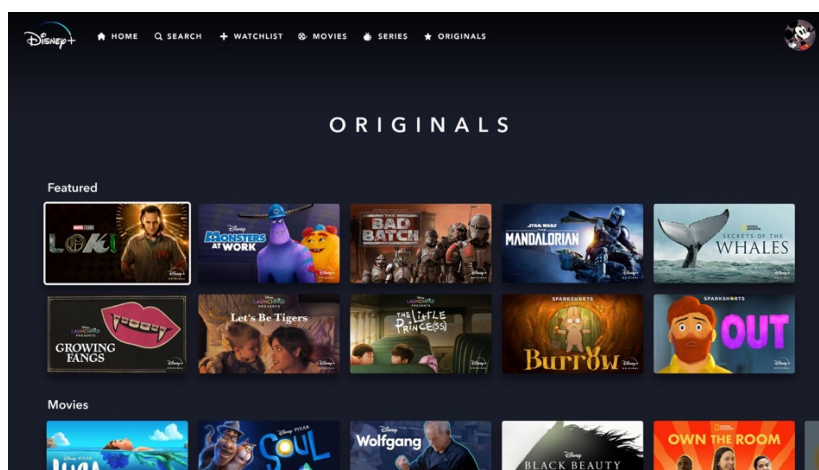


Рисунок 1.5 – Приклад роботи додатку Snapchat Spectacles

Розробка програмних модулів для впровадження мобільної системи підбору відеоконтенту - це складний та багатокроковий процес, який передбачає створення

різних компонентів, які забезпечують функціональність, ефективність та зручність для користувачів та їх інтеграцію в єдину функціональну систему.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристики онлайн платформ

Критерії	YouTube	Netflix	Hulu	Amazon Prime Video	Disney+	Topical Films
Різноманітність контенту	+	-	+	-	-	+
Безкоштовний повний доступ до контенту	-	-	-	-	-	+
Доступність для мобільних девайсів	+	+	+	+	+	+
Можливості персоналізації	+	+	+	-	-	+
Сучасні алгоритми рекомендацій	+	+	-	+	+	+
Підтримка оф-лайн режиму	+	-	-	-	-	+
Загальна оцінка	83%	50%	50%	33%	33%	100%

Відповідно до таблиці порівняльних характеристик (табл.1.1) розробка власної мобільної системи для підбору відеоконтенту є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити новий досвід з пошуку та підбору відеоконтенту.

Отже, розробка мобільної системи для підбору відеоконтенту відкриває значні можливості та перспективи для користувачів та розробників. За допомогою алгоритмів рекомендацій, мобільна система може створити персоналізований досвід перегляду відеоконтенту для кожного користувача, що підвищить задоволення від використання та залучить нових користувачів. Мобільність дає можливість користувачам отримувати доступ до відеоконтенту з будь-якого місця та в будь-який час, що сприяє збільшенню аудиторії та залученню нових користувачів. Розробка мобільної системи дозволить досягти більшої аудиторії, оскільки користувачі все частіше вибирають мобільні пристрої для споживання відеоконтенту. Для виробників відеоконтенту це відкриває можливості для розширення аудиторії та монетизації контенту через рекламу, підписки або преміальний вміст.

1.3 Аналіз методів розв'язання задачі

Розробка ефективних методів та засобів для реалізації мобільної системи для підбору відеоконтенту вимагає комплексного та детального підходу до вирішення цієї проблеми. Одним із найважливіших етапів цього процесу є аналіз та вибір оптимальних стратегій інтеграції різноманітних методик та технологій. Цей розділ присвячено ретельному розгляду найбільш перспективних підходів до розробки та впровадження систем, спрямованих на покращення користувацького досвіду у сфері споживання відео.

Перш за все, важливо визначити ключові цілі та завдання мобільної системи. Це дозволить зорієнтуватися на те, які функції та можливості повинні бути вбудовані в додаток для найбільшого комфорту та задоволення користувачів. Наприклад, серед основних функціональних можливостей є автоматизований підбір відеоконтенту з урахуванням індивідуальних інтересів користувача, підтримка різноманітних форматів відео, можливість відстеження оновлень та рекомендацій, завантаження відеоконтенту на пристрій, а також інтерактивність та зручний інтерфейс.

Далі, в процесі розробки системи слід уважно вивчити та оцінити різноманітні технології та методики, які можуть бути застосовані для досягнення поставлених цілей. Наприклад, для ефективного аналізу відеоданих та побудови рекомендаційних систем можуть використовуватися методи машинного навчання, включаючи нейронні мережі та алгоритми класифікації. Крім того, важливо враховувати особливості мобільних платформ та забезпечити оптимальну продуктивність додатку при мінімальному споживанні ресурсів пристрою.

Не менш важливим є питання забезпечення безпеки та конфіденційності користувачів у процесі використання мобільної системи. Застосування сучасних методів шифрування та заходів захисту даних дозволить уникнути можливих загроз та забезпечити надійність та безпеку інформації.

Під час розробки системи підбору відеоконтенту важливо розглянути широкий спектр аспектів, які можуть впливати на якість та ефективність продукту. Окрім основних аспектів, таких як використання машинного навчання та аналіз текстових та аудіо-візуальних даних, варто звернути увагу на додаткові аспекти, які можуть покращити користувацький досвід та функціональність системи.

Машинне навчання [9] відіграє важливу роль у підборі контенту, допомагаючи класифікувати та групувати відеоматеріали за різними параметрами. Проте, націлене застосування алгоритмів рекомендацій, які враховують індивідуальні уподобання користувача та аналізують його історію переглядів, забезпечить персоналізований підбір контенту, що підвищить його цінність для кожного користувача.

Аналіз текстових та аудіо-візуальних даних також є ключовим елементом у підвищенні якості рекомендацій. Використання методів обробки природної мови для аналізу коментарів користувачів та описів відео допомагає відфільтрувати та категоризувати контент з більшою точністю. Застосування технологій розпізнавання образів та голосу також сприяє автоматизації процесу та покращує користувацький інтерфейс [10].

Оптимізація алгоритмів рекомендацій є необхідним етапом у підвищенні ефективності системи. Розробка алгоритмів, що враховують не лише індивідуальні

вподобання користувачів, але й контекстні фактори, такі як час доби та локація, може значно покращити рекомендаційний механізм.

Нарешті, вибір засобів розробки має велике значення для успішної імплементації системи. Використання потужних мов програмування та фреймворків дозволить розробникам ефективно втілити не лише базовий функціонал, але й додаткові можливості, які підвищують конкурентоспроможність продукту на ринку медіа-сервісів.

Вибір засобів розробки - використання мови програмування Python та фреймворків TensorFlow [11] для машинного навчання.

Розробка мобільних додатків за допомогою Android Studio [] для Android, використовуючи мову програмування Kotlin[12].

Поглиблений аналіз різноманітних методів та інструментів, застосовуваних у розробці мобільних систем для підбору відеоконтенту, дозволяє виявити найбільш оптимальний підхід, спрямований на забезпечення якісних та індивідуалізованих рекомендацій для кінцевих користувачів. Цей глибокий аналіз відкриває нові перспективи та можливості у створенні мобільних додатків, що мають потенціал стати ключовими гравцями на ринку медіа-сервісів.

Однією з вагомих переваг такого підходу є можливість реалізації повноцінного програмного продукту без потреби в складних та дорогих апаратних рішеннях. Це відкриває широкі можливості для стартапів та інноваційних команд, що працюють над розробкою мобільних додатків, зменшуючи фінансові бар'єри та спрощуючи шлях до ринку.

Крім того, застосування високотехнологічних методів розробки дозволяє створити продукт, який відповідає найвищим стандартам якості та відповідає потребам сучасного споживача. Він забезпечує зручний та інтуїтивно зрозумілий інтерфейс, швидку та надійну роботу, а також широкий функціонал, що задовольняє потреби різних категорій користувачів.

Унікальність платформи та її функціоналу стає ключовим конкурентним перевагою на ринку. Шляхом впровадження інноваційних ідей та рішень, що враховують найсучасніші тенденції в сфері розробки мобільних додатків, можна

створити продукт, який вирізнятиметься серед інших та приверне увагу великої аудиторії користувачів.

1.4 Постановка задач для розробки мобільної системи для підбору відеоконтенту

Проаналізувавши переваги та недоліки існуючих системи для підбору відеоконтенту було визначено наступні завдання, які необхідно виконати для розробки мобільної системи:

- розробити метод і модель системи;
- розробити алгоритми рекомендаційного підбору контенту;
- розробити інтерфейс користувача;
- створення бази даних;
- розробити функціонал для завантаження відеоконтенту на пристрій користувача;
- тестування та відлагодження;
- впровадження та підтримка.

1.5 Висновки

У першому розділі дослідження зосередились на аналізі стану ринку мобільних систем для підбору відеоконтенту. Було докладно розглянуто кілька відомих платформ, що включають в себе такі гігантів, як YouTube, Netflix, Hulu, Amazon Prime Video та Disney+.

Під час порівняльного аналізу цих платформ було виявлено, що вони здобули велику популярність серед користувачів завдяки своїй легкості використання та широкій доступності. Проте, також виявилися деякі обмеження, зокрема, відсутність можливості повного безкоштовного доступу до контенту, відсутність підтримки оф-лайн режиму та несуттєві обмеження у сучасних алгоритмах рекомендацій.

На підставі отриманої інформації отримано висновок розроблений власне програмне рішення переважає у функціональності розглянуті аналоги, а також відповідає потребам сучасного ринку медіа-сервісів. Для досягнення цієї мети було сформовано перелік завдань, які необхідно виконати на шляху розробки власної мобільної платформи. Такий підхід дозволив створити продукт, який відповідає найвищим стандартам якості та задовольняти потреби навіть найвибагливіших користувачів.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Для реалізації мобільної системи для підбору відеоконтенту потрібно мати наступні дані:

1. Інформація про користувачів:

- Дані про історію переглядів.
- Вподобання та оцінки відеоконтенту.
- Демографічні дані (вік, стать, місце проживання тощо).

2. Інформація про відеоконтент:

- Метадані про відео (назва, опис, жанр, тривалість тощо).
- Популярність відео (кількість переглядів, рейтинги, коментарі).

3. Контекстуальні дані:

- Інформація про час та місцезнаходження користувача.
- Соціальні та погодні дані, які можуть впливати на вибір відеоконтенту.

Для успішної реалізації нашої мобільної системи для підбору відеоконтенту було обрано використання мови програмування Kotlin та інтегрованого середовища розробки Android Studio. Цей вибір обумовлений декількома вагомими причинами, які дозволять нам максимально ефективно реалізувати наші цілі та завдання.

Почнемо з Kotlin. Ця мова програмування визнана не лише зручною, але й надзвичайно ефективною для розробки мобільних додатків під платформу Android. Будучи офіційною мовою програмування для цієї платформи, Kotlin має широкий набір функцій, які спрощують розробку та підтримку коду. Її сучасний синтаксис дозволяє писати чистий та зрозумілий код, що робить процес розробки більш продуктивним та приємним [13].

Щодо Android Studio, це інтегроване середовище розробки спеціально призначене для створення мобільних додатків для платформи Android. Воно наділене широким спектром інструментів та функцій, що роблять процес розробки,

налагодження та тестування додатків більш простим та ефективним. Багатофункціональний інтерфейс дозволяє розробникам концентруватися на творчому процесі, забезпечуючи при цьому високий рівень продуктивності та якості результату [14].

Етапи реалізації мобільної системи для підбору відеоконтенту є ключовими для створення успішного продукту, що задовольнить потреби користувачів. Розглянемо кожен з них детальніше.

Аналіз та проектування. Починаємо з докладного вивчення потреб та очікувань користувачів. Це включає в себе аналіз ринку, визначення конкурентних переваг, а також проектування архітектури системи з урахуванням зручності використання та масштабованості.

Розробка інтерфейсу користувача. Створюємо інтуїтивно зрозумілий та привабливий інтерфейс, що сприяє зручному взаємодії з додатком. Важливо враховувати ергономіку, дизайн та унікальність, які зроблять додаток привабливим для користувачів.

Розробка бекенду та бази даних. На цьому етапі було створено надійну та ефективну серверну частину системи, яка забезпечує швидку обробку запитів користувачів та безпечне зберігання їх даних.

Реалізація алгоритмів рекомендацій та обробки даних. Розробляємо складні алгоритми для аналізу великої кількості даних користувачів та контенту. Це включає в себе використання штучного інтелекту та машинного навчання для надання персоналізованих рекомендацій.

Тестування та налагодження. Проводимо всебічне тестування системи для виявлення та виправлення будь-яких помилок або недоліків у функціоналі. Важливо забезпечити високу якість продукту перед випуском його на ринок.

Впровадження та підтримка. Після успішного завершення всіх попередніх етапів було розгорнуто систему та забезпечуємо її доступність для користувачів. Крім цього, було забезпечено постійну підтримку та оновлення системи, щоб вона залишалася актуальною та функціональною з часом.

Отже, комбінація Kotlin та Android Studio надає найкращі умови для успішної розробки власного мобільного додатку, що задовольнятиме потреби наших користувачів та стане відмінним прикладом сучасного та функціонального програмного забезпечення[15].

2.2 Розробка моделі системи

Для кращого розуміння роботи програмних засобів системи для підбору відеоконтенту було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 2.1). Згідно вказаної діаграми першим ділом користувач повинен увійти в обліковий запис або зареєструватися.

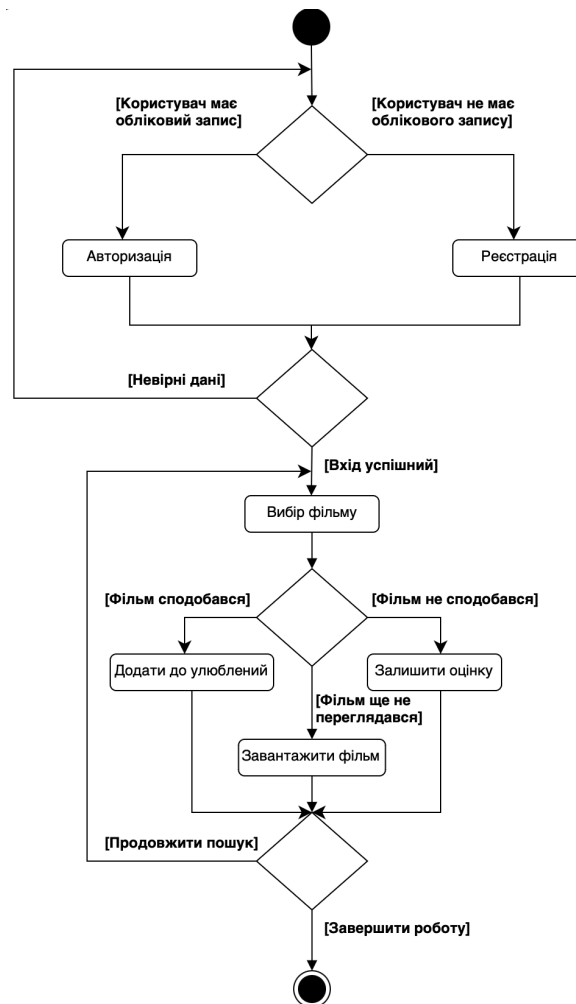


Рисунок 2.1 – Модель системи у вигляді діаграми діяльності

Наступним кроком користувач повинен обрати фільм серед рекомендованих йому. Далі якщо користувач переглядав цей фільм раніше він може додати його для

улюблених або залити оцінку на основі своїх вражень. Якщо користувач ще не переглядав фільм він може завантажити його для перегляду в майбутньому.

Діаграма діяльності системи показує загальну роботу програми на високому рівні абстракції для легшого розуміння робочих процесів.

2.3 Розробка методу персоналізованого підбору відеоконтенту

Ключовою особливістю розроблюваної системи є надання користувачеві можливості отримання персоналізованих відеорекомендацій та завантаження відео для офлайн перегляду, що є перевагою над проаналізованими аналогами.

Метод персоналізованого підбору відеоконтенту дозволяє користувачу отримувати індивідуальні рекомендації на основі історії переглядів та вподобань, а також завантажувати відео для перегляду без підключення до Інтернету.

Метод включає такий функціонал:

1. Авторизація користувача:

- Після запуску програми користувачеві пропонується увійти в обліковий запис або створити новий.
- Клас `AuthViewModel` обробляє введені облікові дані та здійснює перевірку через клас `DatabaseConnection`.
- Якщо дані вірні, користувач успішно входить у систему.

2. Збір та аналіз історії переглядів:

- Після авторизації програма збирає дані про попередні перегляди користувача.
- Клас `HistoryViewModel` зчитує історію переглядів з бази даних за допомогою класу `SqlDataReader` і зберігає її в колекцію `ObservableCollection<Video>`.
- Алгоритм машинного навчання в класі `RecommendationEngine` аналізує зібрані дані та формує профіль вподобань користувача.

3. Генерація персоналізованих рекомендацій:

- На основі профілю вподобань користувача клас `RecommendationEngine` генерує список відеорекомендацій.
- Отримані рекомендації додаються в колекцію `ObservableCollection<VideoRecommendation>` та відображаються користувачеві за допомогою класу `RecommendationsView`.

4. Вибір відео з рекомендацій:

- Користувач обирає відео для перегляду або завантаження.
- Клас `VideoSelectionViewModel` обробляє вибір користувача та надає детальну інформацію про обране відео.

5. Завантаження відео:

- Користувач може обрати відео для завантаження.
- Клас `DownloadManager` розпочинає процес завантаження, використовуючи клас `FileHandler` для збереження файлу.
- Прогрес завантаження відображається в інтерфейсі користувача за допомогою класу `DownloadProgressView`.

6. Перегляд завантажених відео офлайн:

- Завантажені відео зберігаються у внутрішній пам'яті пристрою.
- Клас `OfflineStorage` зберігає інформацію про завантажені відео та забезпечує їх відображення в інтерфейсі користувача через клас `OfflineVideosView`.
- Користувач може переглядати завантажені відео без підключення до Інтернету.

Розглянемо алгоритм роботи запропонованого методу:

1. Запуск програми.
2. Авторизація користувача або реєстрація нового облікового запису.
3. Збір та аналіз даних про попередні перегляди.
4. Генерація персоналізованих рекомендацій.
5. Вибір відео з рекомендацій.
6. Завантаження відео для перегляду офлайн.
7. Перегляд завантажених відео офлайн.

8. Завершення роботи програми.

Наведемо опис класів системи:

1. MainActivity:

- Призначення – головний клас, який запускає програму.
- Функції – відображає головний екран програми та забезпечує навігацію між іншими екранами.

2. AuthViewModel:

- Призначення – клас для обробки авторизації та реєстрації користувачів.
- Функції:
 - login(): Авторизація користувача за допомогою введених облікових даних.
 - register(): Реєстрація нового користувача та створення нового облікового запису.

3. DatabaseConnection:

- Призначення – клас для встановлення з'єднання з базою даних.
- Функції:
 - connect(): Встановлення з'єднання з базою даних.
 - executeQuery(): Виконання запитів до бази даних.

4. HistoryViewModel:

- Призначення – клас для збору та управління історією переглядів користувача.
- Функції:
 - getHistory(): Отримання історії переглядів з бази даних.
 - updateHistory(): Оновлення історії переглядів.

5. RecommendationEngine:

- Призначення – клас для аналізу даних та генерації персоналізованих рекомендацій.
- Функції:
 - analyzeHistory(): Аналіз історії переглядів користувача.

- generateRecommendations(): Генерація списку відеорекомендацій.

6. VideoSelectionViewModel:

- Призначення – клас для управління вибором відео користувачем.
- Функції:
 - selectVideo(): Обробка вибору відео користувачем.
 - getVideoDetails(): Отримання детальної інформації про відео.

7. DownloadManager:

- Призначення – клас для управління завантаженнями відеоконтенту.
- Функції:
 - startDownload(): Початок завантаження відео.
 - pauseDownload(): Призупинення завантаження.
 - resumeDownload(): Відновлення завантаження.
 - cancelDownload(): Скасування завантаження.

8. FileHandler:

- Призначення – клас для обробки файлів під час завантаження.
- Функції:
 - saveFile(): Збереження завантаженого файлу.
 - readFile(): Читання збереженого файлу.

9. OfflineStorage:

- Призначення – клас для зберігання завантажених відеофайлів у внутрішній пам'яті пристрою.

- Функції:
 - saveVideo(): Збереження завантаженого відео.
 - getVideo(): Отримання збереженого відео для перегляду офлайн.

10. VideoPlayer:

- Призначення – клас для відтворення відеоконтенту.
- Функції:
 - play(): Відтворення відео.
 - pause(): Призупинення відтворення.

•stop(): Зупинка відтворення.

Алгоритм роботи цього методу наведено на рисунку 2.2.

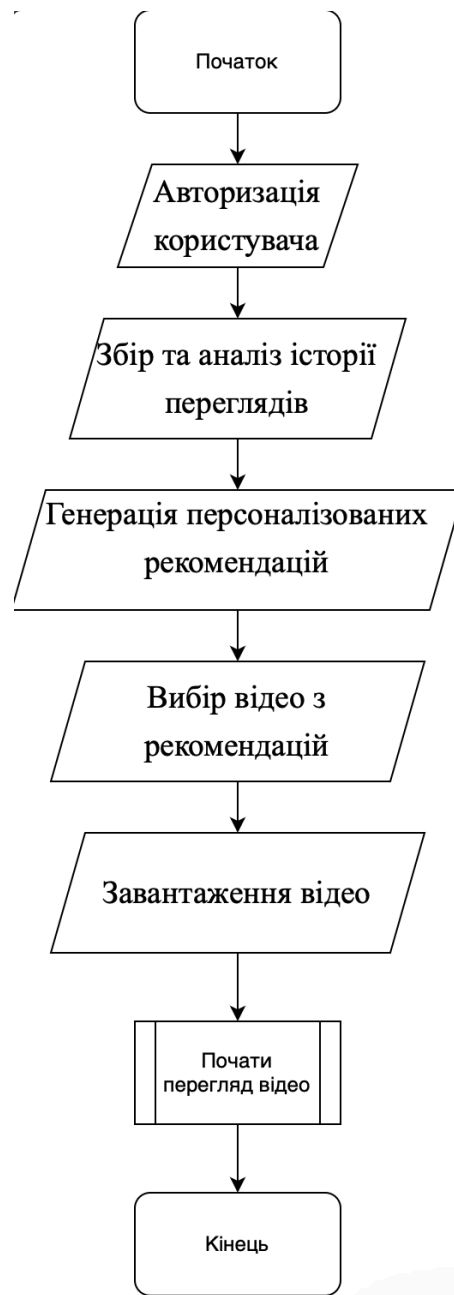


Рисунок 2.2 – Блок-схема алгоритму роботи методу персоналізованого підбору відеоконтенту

Описаний метод включає кроки приєднання до бази даних, формування та надсилання запитів, створення відповідних колекцій та графічних відображень, вибір користувача, завантаження відео та відображення результату користувачеві.

2.4 Розробка алгоритмів роботи системи

У цьому розділі було розглянуто ключові алгоритми, які лежать в основі функціонування системи (рис. 2.3). Кожен з них відповідає за певний аспект роботи системи та забезпечує виконання відповідних завдань.

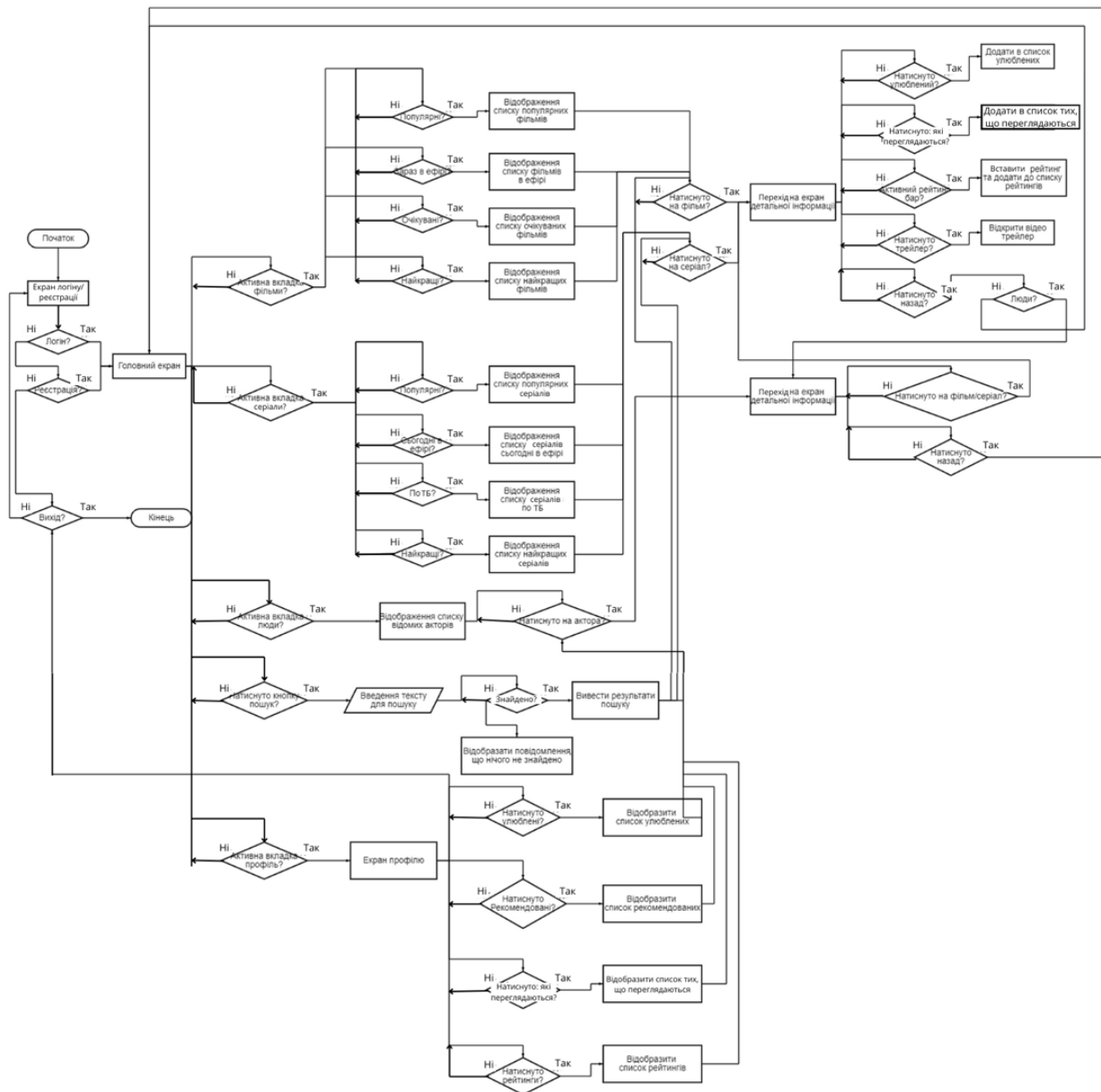


Рисунок 2.3 – Блок-схема алгоритму роботи мобільної системи підбору відеоконтенту

1. Алгоритм формування списку рекомендацій. Алгоритм формування списку рекомендацій використовує складні моделі машинного навчання для аналізу користувацьких даних. Він ретельно вивчає історію перегляду кожного користувача, а також його оцінки та відгуки до вже переглянутого контенту. На основі цього аналізу алгоритм визначає переваги та уподобання кожного користувача для подальшого рекомендування відеоконтенту. Він також враховує популярність та рейтинги контенту серед інших користувачів, щоб забезпечити більш об'єктивні рекомендації. Цей алгоритм неперервно оновлюється та покращується на основі нових даних та оновлень платформи. Він має можливість адаптуватися до змін у виборі користувачів та швидко враховувати їхні зміни у вподобаннях. Алгоритм також враховує контекстуальні фактори, такі як час доби, місцезнаходження та соціокультурні особливості користувача. Він може попередньо рекомендувати відеоконтент, який відповідає найпопулярнішим та актуальним тематикам. Цей алгоритм спрямований на максимізацію задоволення користувачів та забезпечення їм цікавого та різноманітного відеоконтенту. Він є важливою складовою для покращення користувацького досвіду та збільшення активності на платформі.

2. Алгоритм відображення детальної інформації. Алгоритм відображення детальної інформації є ключовою частиною системи та відповідає за представлення обраного фільму або серіалу на екрані "Деталі". Він програмований для ефективного та зручного відображення дати випуску контенту, щоб користувач міг легко зорієнтуватися у часі виходу. Алгоритм також включає в себе функціонал, який дозволяє користувачеві отримати загальний опис сюжету фільму чи серіалу, щоб отримати загальне уявлення про контент. Його завдання включає відображення рейтингу контенту, який може бути важливим критерієм для вибору перегляду для користувача. Алгоритм також надає можливість користувачу залишити власну оцінку, щоб він міг відзначити свої враження та думки про контент. Окрім цього, він забезпечує можливість переглянути трейлер фільму або серіалу, щоб користувач міг отримати уявлення про його зміст та стиль. Алгоритм орієнтується на зручність та доступність інформації, щоб користувач міг швидко та

легко знайти необхідні дані. Він враховує інтерфейсні особливості та дизайн системи, щоб інформація була представлена у привабливому та зрозумілому вигляді. Цей алгоритм також має можливість оновлювати дані про фільми та серіали для забезпечення актуальності інформації. В цілому, алгоритм відображення детальної інформації є важливою складовою системи, яка спрощує користування та робить перегляд контенту приємним та зручним.

2.5 Аналіз функціоналу системи для завантаження відеоконтенту

У цьому розділі було розглянуто розробку функціоналу, який дозволяє користувачам мобільної системи "Tropical Films" завантажувати відеоконтент на пристрій для подальшого перегляду в оф-лайн режимі [16]. Це забезпечує можливість доступу до персоналізованого відеоконтенту без необхідності постійного підключення до інтернету, що є особливо зручним для користувачів, які часто подорожують або мають обмежений доступ до мережі.

Перед розробкою функціоналу необхідно визначити основні вимоги до нього:

- Зручність використання. Інтерфейс для завантаження відео повинен бути інтуїтивно зрозумілим та легким у використанні.
- Швидкість завантаження. Система повинна забезпечувати максимальну швидкість завантаження відеоконтенту, враховуючи доступну пропускну здатність мережі.
- Управління завантаженнями. Користувачі повинні мати можливість керувати завантаженими відео, включаючи перегляд списку завантажень, видалення непотрібного контенту та перевірку статусу завантажень.
- Безпека та захист даних. Забезпечення захисту особистих даних користувачів та дотримання правил ліцензування відеоконтенту.

Архітектура функціоналу завантаження відеоконтенту включає базові компоненти.

Інтерфейс користувача (UI) [17]:

- Елементи управління для запуску, призупинення та відновлення завантажень.
- Індикація статусу завантаження (процент завантаженого контенту, швидкість завантаження).

Модуль керування завантаженнями:

- Обробка запитів на завантаження від користувача.
- Управління чергою завантажень та пріоритетами.

Модуль збереження відеоконтенту:

- Збереження завантаженого контенту у внутрішній пам'яті пристрою або на зовнішніх носіях.
- Забезпечення доступу до завантаженого контенту для офлайн перегляду.

Модуль безпеки:

- Шифрування завантажених даних для захисту від несанкціонованого доступу.
- Дотримання правил ліцензування та авторських прав.

Основні алгоритми, які забезпечують роботу функціоналу завантаження.

1. Алгоритм старту завантаження:

- користувач вибирає відеоконтент для завантаження;
- система перевіряє наявність вільного місця на пристрої;
- якщо місця достатньо, система додає запит на завантаження до черги;
- розпочинається завантаження контенту з сервера;
- під час завантаження відображається індикатор прогресу;
- після завершення завантаження контент зберігається на пристрої.

2. Алгоритм призупинення/відновлення завантаження:

- користувач може призупинити завантаження у будь-який момент;
- система зберігає поточний статус завантаження;
- при відновленні завантаження, система продовжує завантаження з останнього.

3. Алгоритм видалення завантаженого контенту:

- користувач вибирає завантажений контент для видалення;
- система видаляє файл контенту з пам'яті пристрою;
- система оновлює статус завантажень та звільняє місце на пристрої.

Таким чином, розробка функціоналу для завантаження відеоконтенту у мобільній системі "Tropical Films" дозволяє значно покращити користувацький досвід, забезпечуючи доступ до персоналізованого контенту в будь-який час та у будь-якому місці.

2.6 Висновки

У другому розділі було розглянуто ключові алгоритми, що використовуються у функціонуванні системи. Ці алгоритми відповідають за різноманітні аспекти роботи системи, включаючи відображення контенту, формування рекомендацій, управління списками перегляду та взаємодію з базою даних.

Аналізуючи описані алгоритми, можна зрозуміти, що система розроблена з використанням передових технологій та методів, таких як машинне навчання, аналіз даних і оптимізація процесів. Це дозволяє системі ефективно працювати з великим обсягом даних, персоналізувати взаємодію з користувачами та забезпечувати швидкий та зручний доступ до контенту [18].

Крім того, описані алгоритми забезпечують користувачам широкі можливості для організації своїх переглядів, отримання персоналізованих рекомендацій та взаємодії з контентом. Це сприяє покращенню користувацького досвіду та задоволенню потреб кожного користувача.

Особлива увага була приділена розробці функціоналу для завантаження відеоконтенту. Цей функціонал дозволяє користувачам завантажувати підібраний для них відеоконтент на пристрій для подальшого перегляду в офлайн режимі. Така можливість забезпечує доступ до персоналізованого контенту навіть за відсутності підключення до інтернету, що значно підвищує зручність та привабливість системи для користувачів. Впровадження цієї функції потребувало розробки спеціальних

алгоритмів для управління завантаженнями, збереження відеоконтенту та забезпечення його безпеки [19].

Наведено блок-схеми алгоритму роботи розробленого методу та загального алгоритму роботи системи.

Загалом, описані алгоритми грають важливу роль у функціонуванні системи, забезпечуючи її ефективну та надійну роботу. Вони є ключовими складовими успішного функціонування та взаємодії системи з користувачами, що робить їх важливим об'єктом подальшого вдосконалення та розвитку.

Також розроблено метод і модель роботи системи, орієнтованої на потреби користувача.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЙ МОБІЛЬНОЇ СИСТЕМИ ДЛЯ ПІДБОРУ ВІДЕОКОНТЕНТУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

У цьому розділі розглядається варіантний аналіз і обґрунтування вибору засобів для розробки програмного забезпечення, спрямованого на підбір відеоконтенту за допомогою мобільної системи.

Для розробки мобільного додатку обрано мову програмування Kotlin. Це сучасна мова, що працює на платформі Java Virtual Machine (JVM) [20] та має високий рівень сумісності з Java. Kotlin пропонує безпеку типів, простоту синтаксису та підтримку функцій вищого порядку, що сприяє швидкій і продуктивній розробці. Kotlin є відкритою мовою програмування, яка здатна працювати на будь-яких платформах, що підтримують JVM, включаючи Android. Використання Kotlin дозволяє писати більш безпечний і надійний код завдяки системі типізації та іншим механізмам захисту від помилок. Kotlin надає розробникам зручний та ергономічний синтаксис, який спрощує розробку та робить код більш зрозумілим для інших розробників. Однією з переваг Kotlin є його висока сумісність з Java [21], що дозволяє розробникам використовувати існуючий Java-код і бібліотеки у своєму проєкті. Kotlin підтримує функціональне програмування, що дозволяє розробникам використовувати функції вищого порядку та інші конструкції для реалізації складних алгоритмів. Можливість використовувати Kotlin на платформі JVM робить його ідеальним вибором для розробки Android-додатків, оскільки велика частина розробки для Android базується на Java. Kotlin активно розвивається та підтримується корпорацією JetBrains, що забезпечує його подальше вдосконалення та підтримку. Розробка на Kotlin дозволяє ефективно використовувати сучасні технології та практики програмування, такі як реактивне програмування та корутини. Kotlin має широку спільноту розробників, що активно діляться досвідом та розробляють корисні

бібліотеки та інструменти. Враховуючи всі ці переваги, вибір Kotlin для розробки мобільного додатку є обґрунтованим та дозволить забезпечити високу якість та продуктивність розробки [22].

Для розробки мобільного додатку використовується Android Studio - офіційне інтегроване середовище розробки (IDE) від Google для платформи Android. Android Studio надає широкий набір інструментів для розробки, тестування та налагодження Android-додатків, включаючи підтримку Kotlin, емулятори Android-пристроїв та інші корисні функції. Android Studio забезпечує розробникам широкий спектр можливостей для створення високоякісних та ефективних мобільних додатків під платформу Android. Інтегроване середовище розробки Android Studio забезпечує зручний та ергономічний інтерфейс, що сприяє швидкій та продуктивній розробці додатків. Android Studio надає широкий набір інструментів для автоматизації рутинних задач розробки, що дозволяє ефективно використовувати час розробника. Однією з переваг Android Studio є підтримка Kotlin як офіційної мови програмування для розробки Android-додатків, що дозволяє використовувати всі переваги цієї мови у процесі розробки. Емулятори Android-пристроїв, доступні у складі Android Studio, дозволяють розробникам тестувати свої додатки на різних конфігураціях пристроїв без необхідності фізичного наявності цих пристроїв. Android Studio підтримує широкий спектр інтегрованих інструментів для аналізу та оптимізації продуктивності додатків, що дозволяє створювати додатки з високою швидкодією та ефективністю. Завдяки активному співтовариству розробників та постійному оновленню, Android Studio постійно розвивається та вдосконалюється, що забезпечує розробникам доступ до новітніх технологій та інструментів. Android Studio підтримує інтеграцію з такими сервісами як Google Play, що спрощує процес публікації та розгортання додатків на мобільний ринок. Інтегрована підтримка версій у Android Studio дозволяє розробникам ефективно керувати версіями своїх проєктів та спільно працювати над ними в команді. Загальна інтегрованість і багатофункціональність Android Studio роблять його ідеальним інструментом для розробки мобільних додатків для платформи Android [23].

Машинне навчання використовується для аналізування історії перегляду користувачів та надання персоналізованих рекомендацій. Застосування алгоритмів машинного навчання дозволяє автоматично адаптувати пропозиції до вподобань та інтересів користувачів, покращуючи загальний досвід використання додатку. Машинне навчання відіграє ключову роль у покращенні функціоналу нашого мобільного додатку, дозволяючи аналізувати поведінку користувачів та надавати персоналізовані рекомендації. Застосування алгоритмів машинного навчання дозволяє автоматично адаптувати пропозиції контенту до індивідуальних вподобань та інтересів кожного користувача. Моделі машинного навчання, які використовуються в нашому додатку, постійно навчаються на нових даних, що дозволяє їм підтримувати актуальність та точність рекомендацій. Завдяки машинному навчанню, наш додаток може ефективно виявляти зміни у смаках та інтересах користувачів і адаптувати рекомендації відповідно. Використання алгоритмів машинного навчання сприяє покращенню залучення користувачів та збільшенню часу, проведеного ними у додатку. Машинне навчання дозволяє нашому додатку автоматично аналізувати та розуміти поведінку користувачів, що допомагає покращувати якість наданих рекомендацій. Аналіз історії перегляду користувачів за допомогою машинного навчання допомагає виявляти та прогнозувати їхні переваги та тенденції. Машинне навчання спрощує процес прийняття рішень щодо рекомендацій, надаючи аналітичні дані та інсайти, необхідні для підтримки прийняття рішень. Інтеграція машинного навчання в наш додаток дозволяє збільшити рівень задоволеності користувачів, адаптуючи вміст до їхніх індивідуальних потреб. Машинне навчання є ключовим компонентом додатку, який допомагає забезпечувати користувачів персоналізованим та цікавим відеоконтентом.

Отже, у цьому розділі було визначено, що використання мови програмування Kotlin є оптимальним рішенням завдяки її сучасності, безпеці типів та простоті синтаксису. Android Studio було обрано в якості інтегрованого середовища розробки, оскільки воно забезпечує широкий функціонал для розробки, тестування та налагодження Android-додатків, включаючи підтримку Kotlin та емулятори

Android-пристроїв [24]. Використання машинного навчання в додатку дозволяє адаптувати рекомендації до індивідуальних потреб користувачів та покращує їхній загальний досвід використання додатку. Застосування цих засобів дозволить створити ефективний та інноваційний продукт, який задовольнить потреби сучасних користувачів в підборі якісного відеоконтенту.

3.2 Розробка модуля відображення контенту у мобільній системі підбору відеоконтенту

Головною задачею мобільної системи підбору відеоконтенту є забезпечення користувачам можливості ефективно відображати відеоконтент на своїх пристроях. Розробка модуля відображення контенту включає в себе ряд етапів та методів, які дозволяють забезпечити якісне та зручне відображення відеоматеріалів на екрані користувача.

Вибір технологій відображення: Першим кроком є вибір технологій, які будуть використовуватися для відображення відеоконтенту на мобільних пристроях. Це може включати в себе використання стандартних бібліотек або спеціалізованих фреймворків для відео.

Розробка інтерфейсу користувача: Для зручного користування мобільним додатком потрібно розробити інтуїтивно зрозумілий і ергономічний інтерфейс. Це включає в себе створення елементів керування, відображення інформації про відео та інші важливі елементи інтерфейсу.

Оптимізація відображення: Для ефективного відображення відеоконтенту на мобільних пристроях необхідно провести оптимізацію роботи модуля. Це включає в себе оптимізацію швидкості завантаження відео, оптимізацію використання ресурсів пристрою та забезпечення стабільної роботи під час відтворення.

Підтримка різних форматів відео: Модуль повинен підтримувати різні формати відео, щоб користувачі могли переглядати різноманітний контент без обмежень.

Для відображення відеоконтенту використані спеціалізовані бібліотека – ExoPlayer (рис. 3.1). ExoPlayer [25]автоматично підтримує різні формати відео, такі як MP4, HLS, DASH та інші. Також можна легко відтворювати різноманітний відеоконтент без додаткових налаштувань.

```
val player = ExoPlayer.Builder(context).build()
player.setMediaItem(MediaItem.fromUri(videoUri))
player.prepare()
player.play()
```

Рисунок 3.1 – Реалізація використання бібліотеки ExoPlayer

Для створення інтуїтивно зрозумілого і ергономічного інтерфейсу було використувано XML розмітку (рис. 3.2).

```
<Button
    android:id="@+id/playButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Play" />

<TextView
    android:id="@+id/videoInfoTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Video Title"
    android:visibility="invisible" />
```

Рисунок 3.2 – Реалізація створеної XML розмітки

Для оптимізації відображення відео на мобільних пристроях важливо використовувати асинхронне завантаження та кешування відеоданих. Також було

використовано механізми підкастингу (prefetching) (рис.3.3) для завантаження відео заздалегідь.

```
val dataSourceFactory = DefaultDataSourceFactory(context, "Topical Films")
val mediaSource = ProgressiveMediaSource.Factory(dataSourceFactory)
    .createMediaSource(MediaItem.fromUri(videoUri))
```

Рисунок 3.3 – Реалізація функції prefetching

3.3 Розробка модуля для кешування відеоконтенту в мобільній системі підбору відеоконтенту

Імпорт бібліотек (рис. 3.5): Починаємо з імпорту необхідних бібліотек для роботи з відеоконтентом. У нашому випадку, було використано бібліотеку ExoPlayer, яка є потужним інструментом для відтворення медіа-контенту в мобільних додатках на платформі Android.

```
import com.google.android.exoplayer2.upstream.DefaultDataSourceFactory
import com.google.android.exoplayer2.source.ProgressiveMediaSource
import com.google.android.exoplayer2.MediaItem
```

Рисунок 3.5 – Імпорт бібліотек

Створення джерела даних для кешування: Використовуємо DefaultDataSourceFactory (рис. 3.6), який є частиною ExoPlayer, для створення джерела даних. Це джерело даних дозволяє ExoPlayer отримувати вміст з мережі або з кешу, якщо він вже був завантажений.

```
val dataSourceFactory = DefaultDataSourceFactory(context, "Topical Films")
```

Рисунок 3.6 – Реалізація класу DefaultDataSourceFactory

Створення медіа-джерела з кешуванням: За допомогою ProgressiveMediaSource.Factory (рис. 3.7) було створено медіа-джерело з

кешуванням. Це медіа-джерело буде використовувати наше джерело даних для кешування відеоконтенту.

```
val mediaSource = ProgressiveMediaSource.Factory(dataSourceFactory)
    .createMediaSource(MediaItem.fromUri(videoUri))
```

Рисунок 3.7 – Реалізація медіа-джерела ProgressiveMediaSource.Factory

3.4 Розробка модуля для завантаження відеоконтенту

У цьому розділі було розглянуто розробку модуля, який дозволяє користувачам мобільної системи "Topical Films" завантажувати відеоконтент на свої пристрої для подальшого перегляду в офлайн режимі. Такий функціонал забезпечує доступ до персоналізованого контенту навіть за відсутності підключення до інтернету, що значно підвищує зручність та привабливість системи для користувачів.

Для розробки модуля завантаження відеоконтенту на платформі Android були використані ряд технологій та інструментів, які забезпечили ефективність та надійність розробки. Ось детальніша інформація про кожен з використаних компонентів:

1. Мова програмування - Kotlin. Kotlin - це сучасна та потужна мова програмування, що надає широкі можливості для розробки Android-додатків. Вона відмінно поєднується з Android SDK, надаючи зручний синтаксис та підтримку важливих функцій, таких як корутини для асинхронного програмування.
2. Фреймворк для мобільної розробки - Android SDK. Android SDK - це набір інструментів, бібліотек і ресурсів, які дозволяють розробникам створювати програми для операційної системи Android. Цей фреймворк надає доступ до різноманітних функцій, включаючи роботу з графічним інтерфейсом, мережеві операції, роботу з базами даних та багато іншого.

3. Бібліотеки: - Retrofit для виконання HTTP-запитів. Retrofit [25] - це дуже потужна бібліотека для роботи з мережевими запитами у Android-додатках. Вона спрощує процес виконання HTTP-запитів, дозволяючи зосередитися на логіці додатка. Room для роботи з локальною базою даних. Room - це бібліотека, що надає абстракцію бази даних SQLite для Android-додатків. Вона дозволяє зручно взаємодіяти з базою даних, використовуючи анотації та компоненти доступу до даних.
4. Сервіси для зберігання відеоконтенту – Firebase для зберігання та доставки відеофайлів. Firebase - це платформа розробки мобільних та веб-додатків, яка надає широкий спектр інструментів та сервісів, включаючи зберігання та доставку відеофайлів. Firebase пропонує простий та ефективний спосіб управління контентом, забезпечуючи високу швидкість та надійність.

Розробка користувацького інтерфейсу для управління завантаженнями відеоконтенту включала багатоетапний процес, що передбачав аналіз потреб користувачів, проектування відповідного інтерфейсу та імплементацію функціональності з урахуванням найсучасніших рекомендацій та практик дизайну. Під час цього процесу враховувалися не лише технічні аспекти, а й психологічні та ергономічні особливості, що мають велике значення для забезпечення зручного та ефективного взаємодії користувача з додатком.

Розробка UI для управління завантаженнями включала створення інтерфейсу користувача з можливістю ініціювання, призупинення, відновлення та видалення завантажень. Додавання індикаторів прогресу дозволяло зручно відображати користувачам статус завантажень. За реалізацію інтерфейсу користувача відповідає Activity з назвою DownloadActivity. Клас DownloadActivity успадковується від AppCompatActivity. Оголошуються змінні для кнопки завантаження (downloadButton) та прогрес-бару (progressBar). Метод onCreate викликається при створенні активності. Встановлюється макет activity_download. Ініціалізуються змінні downloadButton та progressBar через їхні ідентифікатори в

макеті. Встановлюється слухач подій для кнопки `downloadButton`, який викликає метод `startDownload` при натисканні. Метод `startDownload` запускає процес завантаження, викликаючи метод `startDownload` з `DownloadManager` з URL відео. Метод `updateProgress` оновлює прогрес-бар відповідно до поточного значення прогресу.

Крім того, для оптимізації управління завантаженнями та встановлення пріоритетів був використаний `DownloadManager` (рис. 3.8). Цей інструмент надав можливість структурувати чергу завантажень та керувати пріоритетами, що сприяло оптимальному розподілу ресурсів та підвищенню ефективності завантажень.

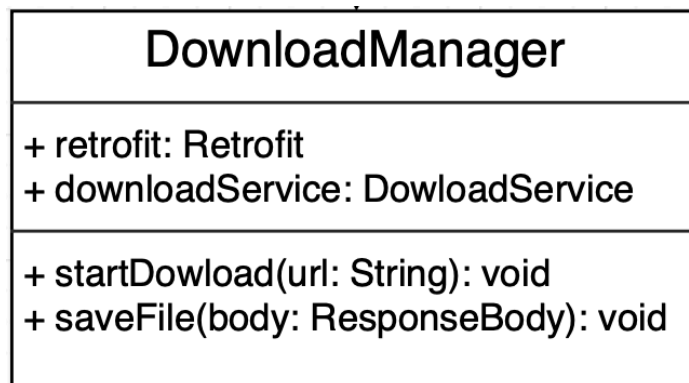


Рисунок 3.8 – Діаграма класу `DownloadManager`

За реалізацію модуля керування завантаженнями, відповідає `DownloadManager` (рис. 3.9). Цей модуль забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувачів, де вони можуть легко керувати процесом завантаження відеоконтенту, включаючи ініціювання, призупинення, відновлення та видалення завантажень.

`DownloadManager` оголошується як об'єкт (синглтон), що означає, що буде створено лише один екземпляр цього класу. Створюється екземпляр `Retrofit` з базовим URL. Створюється сервіс завантаження з використанням інтерфейсу `DownloadService`. Метод `startDownload` приймає URL як параметр. Викликається метод `downloadFile` з сервісу завантаження. Використовується метод `enqueue` для

виконання асинхронного запиту. В методі `onResponse` перевіряється успішність відповіді, і якщо відповідь успішна, викликається метод `saveFile` для збереження отриманого файлу. В методі `onFailure` здійснюється обробка помилок. Метод `saveFile` відповідає за збереження отриманого файлу у внутрішню пам'ять пристрою.

Такий комплексний підхід до розробки дозволив створити надійний та функціональний модуль для завантаження відеоконтенту на платформі Android, що відповідає сучасним стандартам та вимогам користувачів.

```
object DownloadManager {

    private val retrofit = Retrofit.Builder()
        .baseUrl("https://example.com")
        .build()

    private val downloadService = retrofit.create(DownloadService::class.java)

    fun startDownload(url: String) {
        val call = downloadService.downloadFile(url)
        call.enqueue(object : Callback<ResponseBody> {
            override fun onResponse(call: Call<ResponseBody>, response: Response<ResponseBody>) {
                if (response.isSuccessful) {
                    response.body()?.let { body ->
                        saveFile(body)
                    }
                }
            }
            override fun onFailure(call: Call<ResponseBody>, t: Throwable) {
                // Обробка помилки
            }
        })
    }

    private fun saveFile(body: ResponseBody) {
        // Логіка для збереження файлу у внутрішню пам'ять пристрою
    }
}
```

Рисунок 3.9 – Реалізація класу `DownloadManager`

Реалізація модуля збереження відеоконтенту була важливим етапом у розробці мобільного додатку, оскільки цей модуль забезпечував ефективне та надійне зберігання відеофайлів і їхніх метаданих. Для цього використовувалася низка технологій і підходів, які гарантували оптимальну роботу додатку та зручність для користувачів.

Основним компонентом для збереження метаданих про завантажені відеофайли була бібліотека `Room`. `Room` є частиною `Android Jetpack` і надає абстракцію для роботи з базою даних `SQLite`. Використання `Room` дозволило

створити надійну та ефективну локальну базу даних, де зберігалися всі необхідні метадані про завантажені відеофайли, такі як назва, розмір, дата завантаження та шлях до файлу. Це значно спростило процес доступу до інформації про завантаження та забезпечило високу швидкість операцій з базою даних.

Крім того, самі відеофайли зберігалися у внутрішній пам'яті пристрою. Внутрішня пам'ять пристрою була обрана через її надійність і безпеку, що гарантувало збереження контенту навіть у випадку відсутності доступу до мережі або зовнішніх сховищ.

За загальне зберігання даних у додатку відповідав компонент під назвою `AppDatabase`. `AppDatabase` виконував роль центрального вузла для управління всіма операціями з базою даних і гарантував їхню синхронізацію та узгодженість. Цей компонент забезпечував інтерфейс для взаємодії з базою даних, включаючи додавання, оновлення, видалення та отримання даних. Крім того, `AppDatabase` відповідав за ініціалізацію бази даних і налаштування всіх необхідних параметрів для її коректної роботи.

Таким чином, реалізація модуля збереження відеоконтенту включала використання `Room` для зберігання метаданих про завантажені відеофайли в локальній базі даних та збереження самих відеофайлів у внутрішній пам'яті пристрою. За зберігання даних і управління базою даних відповідав компонент `AppDatabase`, який забезпечував надійну та ефективну роботу додатку (рис. 3.10).

Клас `Download` позначається як сутність (`@Entity`) для таблиці `downloads`. Поля `id`, `url`, `filePath`, `status` визначаються як стовпці таблиці, причому `id` є первинним ключем, що генерується автоматично. Інтерфейс `DownloadDao` позначається як `Data Access Object (DAO)` для доступу до даних таблиці `downloads`. Методи `insert`, `update`, `delete`, `getAllDownloads` забезпечують вставку, оновлення, видалення записів та отримання всіх записів з таблиці відповідно. Клас `AppDatabase` позначається як база даних `Room (@Database)`, що містить сутність `Download`. Визначається метод `downloadDao()`, який повертає інтерфейс `DownloadDao` для доступу до методів DAO.

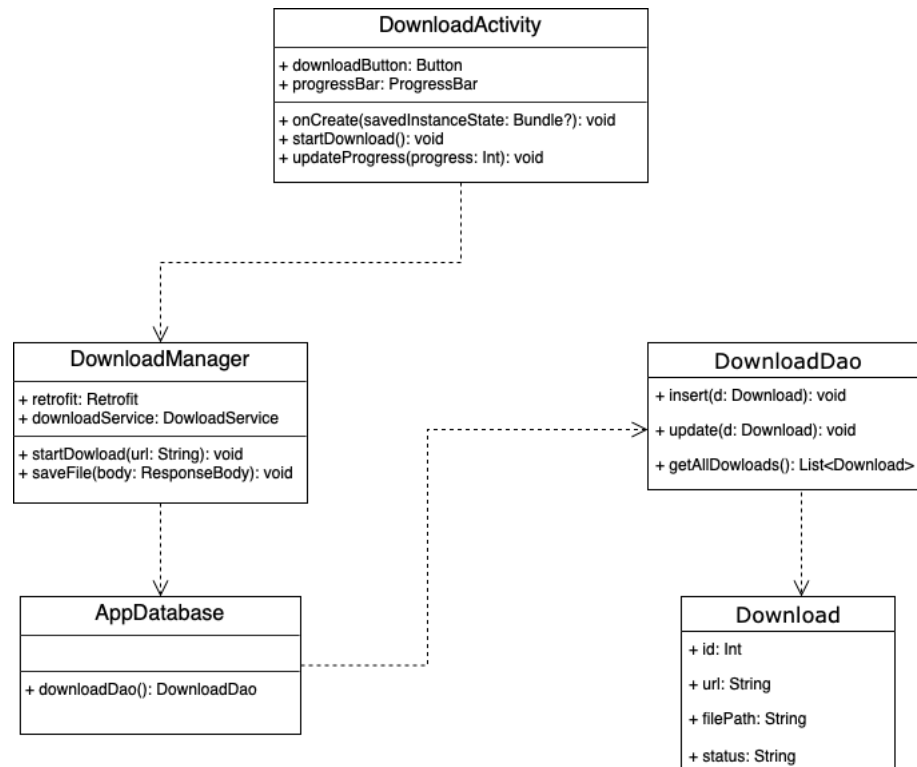


Рисунок 3.10 – Діаграма класів для керування завантаженням відеоконтенту

Цей підхід дозволив створити потужний та зручний інструмент для роботи з відеоконтентом на платформі Android, що відповідає всім сучасним вимогам і стандартам.

Забезпечення безпеки даних є критично важливим аспектом у розробці будь-якого програмного забезпечення, особливо коли йдеться про завантаження та зберігання відеофайлів. Шифрування завантажених відеофайлів дозволяє зберігати їх у зашифрованому вигляді, що гарантує, що лише авторизовані користувачі можуть отримати доступ до вмісту.

Для реалізації шифрування відеофайлів використовуються сучасні криптографічні алгоритми, такі як AES (Advanced Encryption Standard) [26], який забезпечує високий рівень безпеки. Процес шифрування включає перетворення відеофайлів у зашифрований формат під час завантаження. Це гарантує, що навіть якщо файл буде перехоплений під час передачі або несанкціоновано завантажений, його

Для реалізації шифрування в мобільному додатку була використана бібліотека для роботи з криптографією, а саме "javax.crypto". Ця бібліотека є частиною стандартної бібліотеки Java і забезпечує потужні та надійні інструменти для виконання криптографічних операцій, таких як шифрування та дешифрування даних. Зокрема, "javax.crypto" дозволяє використовувати різноманітні алгоритми шифрування, включаючи AES (Advanced Encryption Standard), який відомий своєю високою безпекою та ефективністю.

За шифрування файлів відповідає функція `encryptFile`. Функція `generateKey()` - створює ключ для AES шифрування. Використовується 256-бітний ключ. `encryptFile(file: File, key: Key)` цей метод приймає файл та ключ, ініціалізує шифрування за допомогою AES і створює шифрований файл.

За дешифрування файлів відповідає функція `decryptFile` (рис. 3.11). цей метод приймає зашифрований файл та ключ, ініціалізує дешифрування за допомогою AES і створює оригінальний файл.

```
private fun decryptFile(file: File, key: Key) {  
    val cipher = Cipher.getInstance("AES")  
    cipher.init(Cipher.DECRYPT_MODE, key)  
  
    val inputStream = FileInputStream(file)  
    val decryptedFile = File(file.parent, "decrypted_${file.name}")  
    val outputStream = FileOutputStream(decryptedFile)  
    val cipherInputStream = CipherInputStream(inputStream, cipher)  
  
    cipherInputStream.use { input ->  
        outputStream.use { output ->  
            input.copyTo(output)  
        }  
    }  
}
```

Рисунок 3.11 – Реалізація функції `decryptFile`

3.5 Розробка інтерфейсу програмного додатку

Під час розробки інтерфейсу програмного додатку було визначено стратегію, спрямовану на створення максимально привабливого та інтуїтивно зрозумілого середовища для користувачів. Основний колір інтерфейсу був обраний темно-синій, оскільки він відомий своєю нейтральністю та здатністю зменшувати

втомлення очей під час тривалого користування. Для створення контрасту та покращення візуальної здатності вибраного основного кольору, було використано блакитний та білий як допоміжні кольори. Це дозволило нам досягти гармонійного злиття кольорів, що сприяє приємному та зручному користуванню додатком [14]. Крім того, було приділено увагу деталям та ергономіці інтерфейсу, щоб забезпечити максимальний комфорт для користувачів у всіх видах взаємодії з додатком.

При відкритті додатку, перше, що зустрічає користувача, - це привітальний екран (рис. 3.12), який створено з метою викликати позитивні емоції та забезпечити комфортне та приємне використання системи. Цей екран вітає користувача логотипом та назвою додатку, виконані в приємних для ока кольорах, що сприяють створенню привабливого та дружнього середовища. Такий підхід сприяє позитивному першому враженню від використання додатку та створює основу для подальшої приємної взаємодії з ним.



Рисунок 3.12 – Екран завантаження

Оскільки однією з ключових функцій нашої системи є можливість реєстрації окремих користувачів, було розроблено та впроваджено спеціальний екран реєстрації. На цьому екрані користувачеві необхідно ввести свої персональні дані

та обрати надійний пароль. Після успішної реєстрації особистого кабінету, користувач переходить на екран авторизації (рис. 3.13), де може ввести свої аутентифікаційні дані для входу в систему додатка. Пройшовши авторизацію, користувач відкриває для себе широкі можливості, серед яких - обирання цікавих для нього жанрів фільмів або серіалів та можливість підписатися на нові відеоконтент від улюблених акторів.

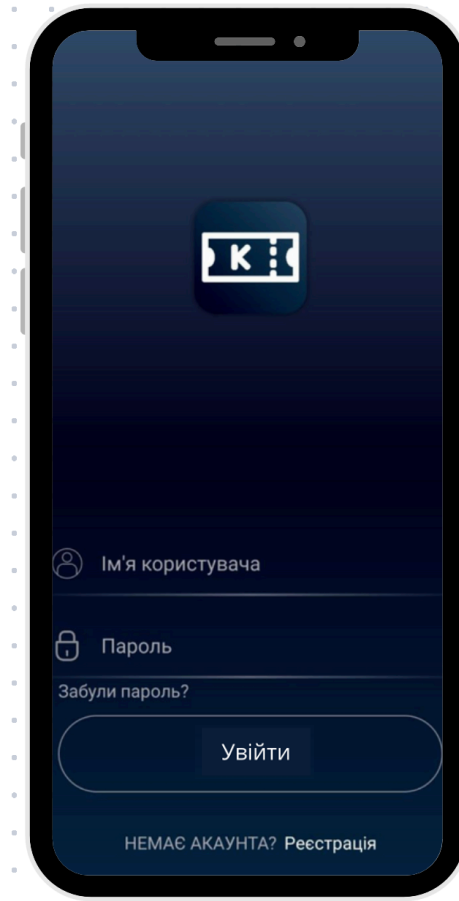


Рисунок 3.13 – Сторінка авторизації користувача

На головному екрані (рис. 3.14) користувач має можливість відкрити широкий асортимент відеоконтенту, який система пропонує йому. Тут він знаходиться в світі розваг та розваг, маючи можливість не лише насолоджуватися переглядом фільмів, але і глибше зануритися у світ серіалів, щоб насолодитися довгими годинами відпочинку. Однак крім цього, у верхній частині екрану розташовані інші цікаві можливості, які розширюють взаємодію користувача з

додатком. Тут він може здійснити вибір серед фільмів, серіалів, а також відкрити список улюблених акторів та актрис, щоб ознайомитися з їхнім творчим доробком. Крім того, в один клік користувач може перейти на особистий профіль, де він має можливість налаштувати свої вподобання та переглянути свою активність у системі, що додає йому більше можливостей для особистого налаштування досвіду користування.

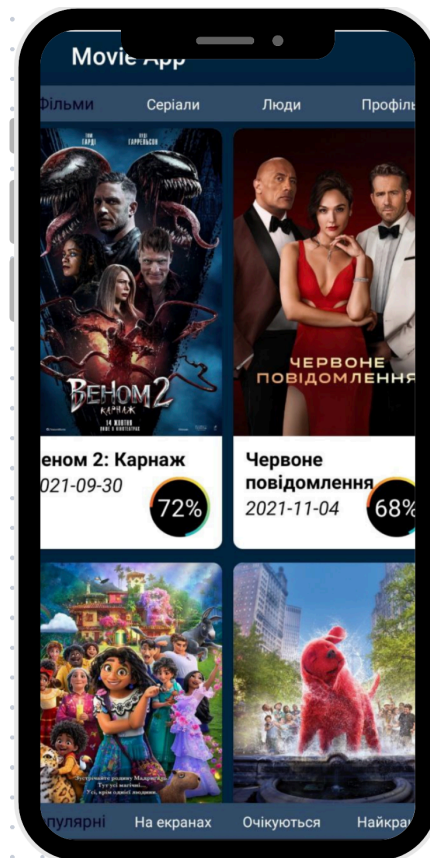


Рисунок 3.14 – Головний екран

У нижній частині екрану, яка відображається на пристрої користувача, розташовані різноманітні опції, що становлять великий інтерес для користувача. Серед цих опцій можна виділити такі категорії, як "Популярні", "На екранах", "Очікуються" та "Найкращі". Ці категорії надають користувачеві можливість швидко орієнтуватися в доступному контенті та знайти те, що найбільше відповідає його потребам та уподобанням.

Перейшовши на екран, що присвячений серіалам (рис. 3.15), користувач отримує унікальну можливість ознайомитися з широким спектром цікавих телесеріалів. Тут він може вивчити їхні рейтинги, ознайомитися з докладними описами, а також дізнатися про дату виходу нових серій. Кожен серіал представлений з урахуванням різноманітних вимог користувача, щоб забезпечити йому найкращий досвід перегляду та вибору відповідно до його смаку та попередніх уподобань.

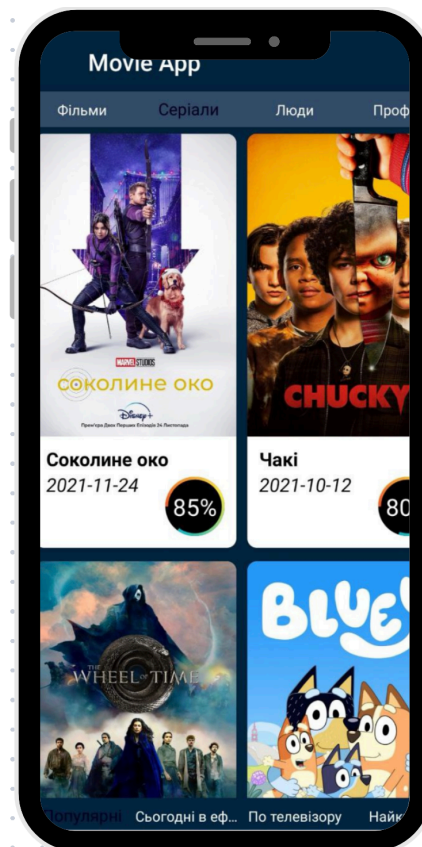


Рисунок 3.15 – Екран «Серіали»

Екран "Деталі" (рис. 3.16) відіграє ключову роль у наданні користувачеві повної та докладної інформації про конкретний фільм або серіал. Тут користувач може знайти вичерпність інформації про вибраний контент, включаючи його опис, основні характеристики та можливості. Окрім того, на цьому екрані надається можливість переглянути трейлер у вбудованому відео плеєрі, щоб отримати загальне уявлення про зміст та атмосферу твору.

Важливим функціоналом екрану "Деталі" є можливість додавання фільму або серіалу до списку улюблених або до списку для подальшого перегляду. Це забезпечує користувачеві зручність та швидкість доступу до обраного контенту у майбутньому. Додавання фільму або серіалу до улюблених або списку перегляду виконується простими діями за допомогою відповідних кнопок, розташованих на екрані. Такий інтуїтивно зрозумілий інтерфейс дозволяє користувачеві легко керувати своїм персональним вибором та вподобаннями прямо з одного екрану.

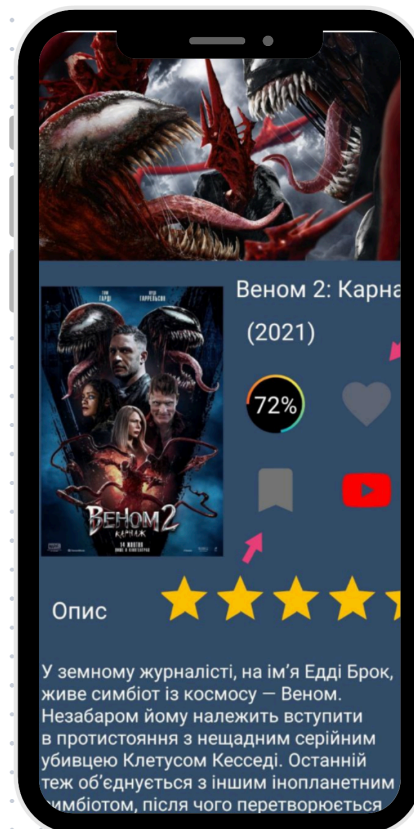


Рисунок 3.16 – Екран «Деталі»

На екрані «Профіль» (рис. 3.17) користувач може встановити зображення аватарки, перейти на екран вподобаних фільмів та серіалів, а також на екран, де відображаються рекомендації, створені алгоритмами для конкретного користувача. Крім того, він має можливість перейти на екран, де можна переглянути список вже переглянутих раніше фільмів та серіалів. Користувач може відкрити екран вподобаних та переглянути список, а також перейти на екран деталей, щоб

ознайомитися з контентом. Також він може видалити зі списку вподобаних елементи. При переході на екран рекомендацій, користувач може вибрати між фільмами, серіалами або мультфільмами. Клікнувши на певний контент, відкриється екран з детальною інформацією. Перейшовши на екран перегляду, користувач отримує список контенту, який він вже переглянув, що дозволяє йому знову ознайомитися з ним. Перейшовши на екран рейтингів, користувач отримує список контенту, відсортований за рейтингом, а також контент, якому він особисто виставив свою оцінку.

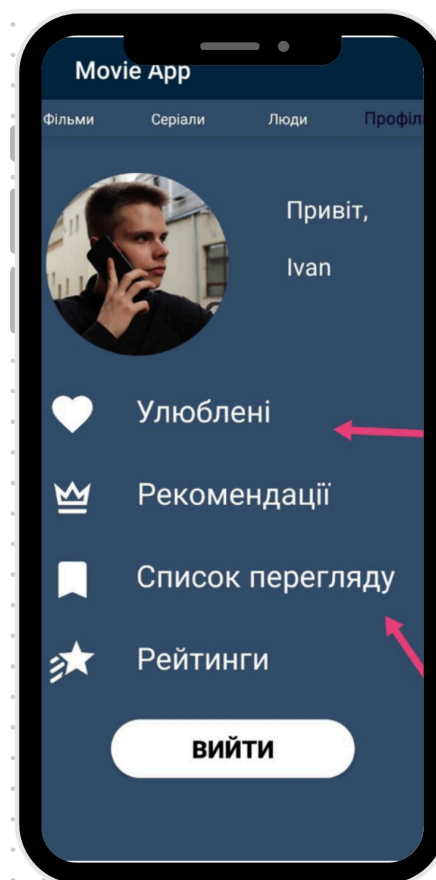


Рисунок 3.17 – Екран «Профіль»

Розробка графічного інтерфейсу мобільної системи була проведена у середовищі розробки Android Studio [16].

3.6 Висновки

У третьому розділі було обґрунтовано вибір мови програмування та технологій для розробки системи. Розробка модулів для мобільної системи підбору відеоконтенту є важливим етапом у створенні функціонального та зручного застосунку для користувачів. Використання сучасних технологій, таких як Kotlin та Android Studio, дозволяє ефективно втілювати функціонал та забезпечувати високу якість додатку.

Особлива увага була приділена реалізації модуля для завантаження відеоконтенту. Цей модуль забезпечує користувачам можливість завантажувати відеофайли на свої пристрої для подальшого перегляду в офлайн-режимі, що значно підвищує зручність та гнучкість використання додатку. Завдяки використанню бібліотеки Retrofit для виконання HTTP-запитів, процес завантаження відеоконтенту є надійним і швидким. Крім того, було реалізовано шифрування завантажених файлів для захисту від несанкціонованого доступу, що підвищує безпеку даних користувачів.

Таким чином, комплексний підхід до розробки, включаючи використання сучасних технологій і впровадження додаткових функціональних можливостей, таких як завантаження та шифрування відеоконтенту, забезпечує створення високоякісного та безпечного мобільного додатку, який відповідає потребам сучасних користувачів.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування системи мобільного підбору відеоконтенту є невід'ємною частиною розробки, спрямованою на забезпечення якості та надійності додатку перед його випуском. Цей розділ описує методи та підходи, використані для тестування різних аспектів системи.

1. Функціональне тестування. Функціональне тестування включає в себе перевірку функціональності кожного модуля та можливості системи в цілому відповідно до вимог та очікувань. Це включає в себе тестування відтворення відеоконтенту, функціональність інтерфейсу користувача, роботу алгоритмів підбору контенту та інші ключові функції додатку.

2. Інтеграційне тестування. Інтеграційне тестування перевіряє взаємодію різних модулів та компонентів системи. Це дозволяє виявити та усунути проблеми, пов'язані зі зламами взаємодії між різними частинами системи, що можуть виникнути під час реального використання.

3. Тестування продуктивності. Тестування продуктивності оцінює швидкодію та продуктивність системи в різних умовах. Це включає в себе аналіз часу відгуку додатку, швидкість завантаження відеоконтенту, використання ресурсів пристрою та інші метрики продуктивності.

4. Тестування на відмовостійкість. Тестування на відмовостійкість оцінює, як система реагує на негативні сценарії та виклики. Це включає в себе тестування додатку на стійкість до відключень мережі, введення некоректних даних користувачем та інші потенційні проблеми.

5. Тестування на різних пристроях. Тестування на різних пристроях забезпечує сумісність та оптимальну роботу додатку на різних моделях та версіях мобільних пристроїв. Це включає в себе тестування на пристроях з різними версіями операційної системи Android та різною апаратною специфікацією. Результат тестування зображено на рисунку 4.1.

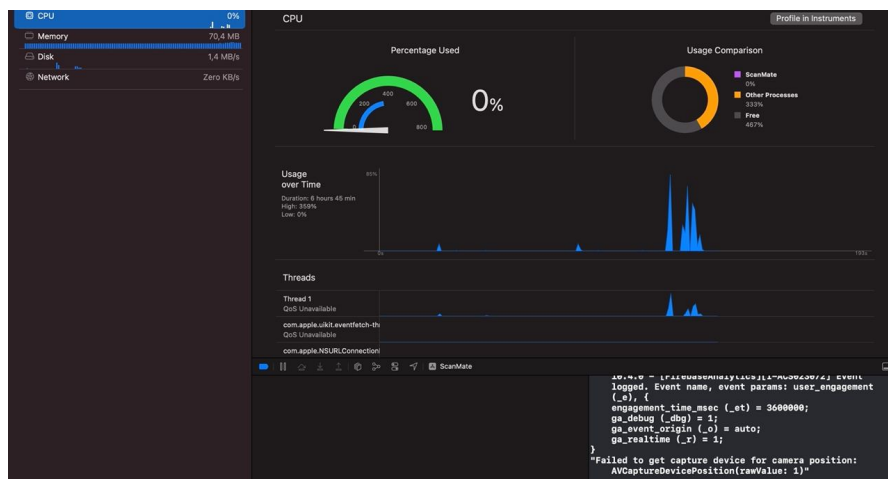


Рисунок 4.1 – Тестування продуктивності системи

Наступним етапом буде проведення перевірки екрану реєстрації користувача через спробу зареєструвати його без введення імені. У випадку спроби реєстрації без обов'язкового введення інформації про ім'я, система повідомить про помилку та запропонує ввести необхідні дані. Це відповідає стандартам правильної роботи системи, оскільки інформація про ім'я є необхідною для подальшого користування. Результати проведеного тестування можна побачити на рисунку 4.2.

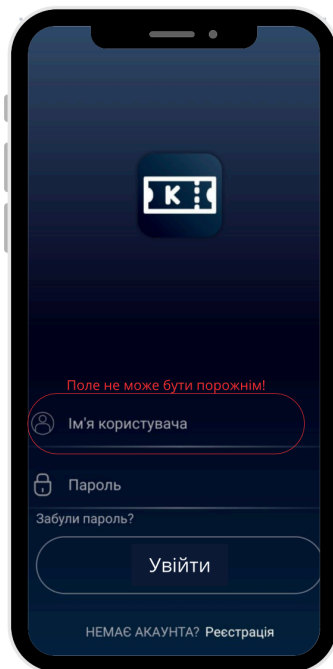


Рисунок 4.2 – Реєстрація користувача без імені

Після переконання в тому, що система блокує можливість реєстрації користувача без обов'язкових полів, було спробовано ввести некоректне значення

у поле електронної адреси. Після завершення тестування було отримано сповіщення про некоректність введеної електронної адреси, що відображено на рисунку 4.3.

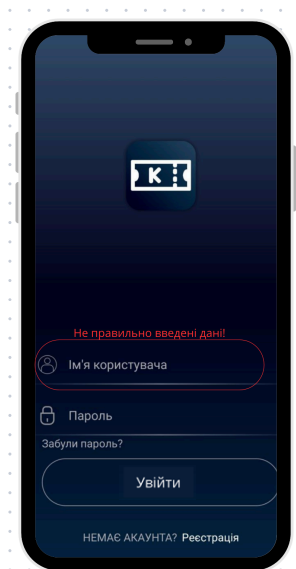


Рисунок 4.3 – Реєстрація користувача з неправильним електронним адресом

Перевіривши введення не дійсних даних, спробуємо заповнити всі поля вірно і перевірити чи зареєструє система користувача (рис. 4.4).

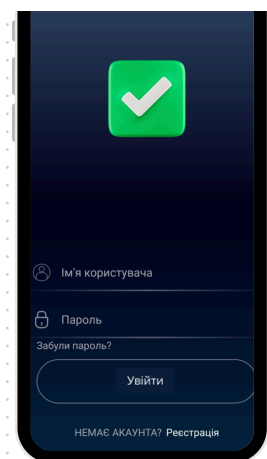


Рисунок 4.4 – Результат успішної реєстрації користувача

У результаті перевірки система перенаправила користувача на сторінку авторизації, отже користувача було успішно зареєстровано.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації мобільного девайсу можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	Qualcomm Snapdragon 400
Об'єм оперативної пам'яті	2 ГБ
Місце на жорсткому диску	1 ГБ
Операційна система	Android KitKat

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	Qualcomm Snapdragon 800
Об'єм оперативної пам'яті	4 ГБ
Місце на жорсткому диску	4 ГБ
Операційна система	Android 12

Інструкція користувача для мобільної системи підбору відеоконтенту.

Завантаження та встановлення додатку:

Відкрийте магазин додатків на своєму мобільному пристрої.

У полі пошуку введіть "Tropical Films" або назву додатку.

Оберіть додаток зі списку результатів та натисніть кнопку "Встановити".

Після завершення завантаження та встановлення натисніть кнопку "Відкрити".

Реєстрація або вхід в обліковий запис:

Якщо у вас вже є обліковий запис, введіть свої дані (адреса електронної пошти та пароль) і натисніть кнопку "Увійти".

Якщо у вас немає облікового запису, натисніть кнопку "Зареєструватися" і дотримуйтесь інструкцій для створення нового облікового запису.

Перегляд відеоконтенту:

На головному екрані додатку буде продемонстровано список рекомендованого відеоконтенту.

Натисніть на відео, щоб розгорнути його та розпочати перегляд.

Використовуйте елементи керування (пауза, відтворення, перемотка) для керування відтворенням відео.

Пошук відеоконтенту:

Використовуйте функцію пошуку для знаходження конкретних відео за назвою, жанром або актором.

Введіть запит у поле пошуку і натисніть кнопку "Пошук".

Додавання відео до списку улюблених:

Під час перегляду відео натисніть на відповідний значок або кнопку, щоб додати відео до списку улюблених.

Керування обліковим записом:

В розділі "Налаштування" буде можливість змінити свій пароль, адресу електронної пошти та інші особисті дані.

Також тут можна вибрати параметри відображення відеоконтенту та налаштування приватності.

4.3 Висновки

У четвертому розділі було проведено тестування мобільної системи підбору відеоконтенту "Tropical Films", доказано що вона є стабільною, функціональною та зручною для користувачів. Інструкція користувача дозволяє легко встановити та користуватися додатком.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено мобільну систему підбору відеоконтенту «Topical Films», що дозволяє користувачам швидко та зручно знаходити відеоматеріали, які відповідають їхнім індивідуальним уподобанням та інтересам. Розробка такого додатку є важливим кроком у вдосконаленні інтернет-середовища та розширенні можливостей користувачів у споживанні відеоконтенту, що відповідає сучасним вимогам і тенденціям розвитку інформаційних технологій.

Проведено аналіз стану існуючих систем для підбору відеоконтенту, розглянуто популярні аналоги, такі як YouTube, Netflix, Amazon Prime Video, Hulu та Disney+. Визначено їхні переваги та недоліки в порівнянні з мобільною системою «Topical Films» та встановлено, що розроблений програмний засіб переважає у функціональності розглянуті аналоги, зокрема, завдяки можливості завантаження відеоконтенту для перегляду в офлайн-режимі.

Базуючись на результатах аналізу та порівняння, були поставлені основні задачі бакалаврської дипломної роботи та обрано методи розробки мобільної системи. Проведено варіантний аналіз та обґрунтування вибору засобів реалізації програмного продукту. В якості мови програмування було обрано Kotlin, а фреймворк для мобільної розробки - Android SDK. Використовувались бібліотеки Retrofit для виконання HTTP-запитів та Room для роботи з локальною базою даних. Для зберігання та доставки відеофайлів було застосовано Firebase.

Відповідно до поставлених задач було:

- розроблено блок схеми загального алгоритму роботи мобільної системи для підбору відеоконтенту, яка описує функціонал платформи;
- розроблено алгоритми основних програмних компонентів застосунку;
- реалізовано функціонал завантаження відеоконтенту на пристрій користувача для подальшого перегляду в офлайн-режимі;
- розроблено зручний та легкий інтерфейс користувача для мобільного додатку;

- описано програмну реалізацію основних модулів, включаючи модуль шифрування завантажених файлів для забезпечення безпеки даних;
- забезпечено підтримку української мови та доступність мобільної системи на платформі Android;
- проведено тестування програми, яке показало цілковиту працездатність програмного продукту та відповідність поставленому технічному завданню.

Крім того, було розроблено метод формування персоналізованих рекомендацій, який дозволяє аналізувати дані про перегляди, вподобання та інші дії користувачів для надання індивідуальних рекомендацій. Це сприяє покращенню користувацького досвіду та задоволенню потреб кожного користувача.

Також було розроблено інструкцію користувача, що забезпечує легкість освоєння та використання мобільного додатку.

Бакалаврську дипломну роботу було оформлено відповідно до методичних вказівок [27].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зростання активності в інтернеті [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/46359/>
2. Офлайн перегляд фільмів [Електронний ресурс] – Режим доступу до ресурсу: <https://popcorn.politeka.net/uk/425100-kak-na-smartfone-posmotret-filmy-esli-net-interneta-poleznye-sovety>
3. Бабійчук І.С. Розробка мобільної системи підбору відеоконтенту “Topical Films” / В. В. Войтко, А. В. Денисюк, О. В. Гаврилюк, Н. Є. Барчук, І. С. Бабійчук // Матеріали ЛІІІ Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: <https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20436/16967>
4. “YouTube” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.youtube.com>
5. “Netflix” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.netflix.com/ua/>
6. “Hulu” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.hulu.com/welcome>
7. “Amazon Prime Video” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.primevideo.com/offers/nonprimehomepage>
8. “Disney+” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.preview.disneyplus.com/unavailable>
9. “Android Developer: AI” [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/ai>
10. “Android Developer Documentation” [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/>
11. “TensorFlow” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.tensorflow.org>

12. "Kotlin Programming Language Documentation" [Электронный ресурс] – Режим доступа до ресурсу: <https://kotlinlang.org/docs/home.html>
13. "Google Developers: Android Studio" [Электронный ресурс] – Режим доступа до ресурсу: <https://developer.android.com/studio>
14. "Android Developers Blog" [Электронный ресурс] – Режим доступа до ресурсу: <https://android-developers.googleblog.com/>
15. "W3Schools: Android Tutorial" [Электронный ресурс] – Режим доступа до ресурсу: <https://www.w3schools.com/android/>
16. "Stack Overflow: Android" [Электронный ресурс] – Режим доступа до ресурсу: <https://stackoverflow.com/questions/tagged/android>
17. "GitHub: Kotlin" [Электронный ресурс] – Режим доступа до ресурсу: <https://github.com/JetBrains/kotlin>
18. "YouTube: Android Developers" [Электронный ресурс] – Режим доступа до ресурсу: <https://www.youtube.com/user/androiddevelopers>
19. "Medium: Android Developers" [Электронный ресурс] – Режим доступа до ресурсу: <https://medium.com/androiddevelopers>
20. "Ray Wenderlich: Android Tutorials" [Электронный ресурс] – Режим доступа до ресурсу: <https://www.raywenderlich.com/android>
21. "Android Arsenal: Open-Source Android Apps" [Электронный ресурс] – Режим доступа до ресурсу: <https://android-arsenal.com/>
22. "Java Virtual Machine; JVM" [Электронный ресурс] – Режим доступа до ресурсу: https://uk.wikipedia.org/wiki/Віртуальна_машина_Java
23. "Java" [Электронный ресурс] – Режим доступа до ресурсу: <https://www.java.com/ru/>
24. "CodePath: Android Guides" [Электронный ресурс] – Режим доступа до ресурсу: <https://guides.codepath.com/android>
25. "Android Weekly Newsletter" [Электронный ресурс] – Режим доступа до ресурсу: <https://androidweekly.net/>
26. "Vogella: Android Tutorials" [Электронный ресурс] – Режим доступа до ресурсу: <https://www.vogella.com/tutorials/android.html>

27.Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 12 » березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка методу та програмних засобів реалізації мобільної системи
підбору відеоконтенту "Tropical Films"»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. Войтко В. В.
« 12 » березня 2024 р.

Виконав:
_____ студент гр. ЗПІ-206 Бабійчук І. С.
« 12 » березня 2024 р.

Вінниця – 2024

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методу та програмних засобів реалізації мобільної системи підбору відеоконтенту "Topical Films"».

Галузь застосування – медіа та розвоги.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від 20 березня 2024 року.

3. Мета та призначення розробки.

Метою бакалаврської дипломної є покращення можливостей підбору відеоконтенту шляхом розробки та використання спеціалізованого програмного продукту, який дозволяє проводити аналіз переглядів користувача, визначати їх вподобання та надавати рекомендації й здійснювати пошук потрібного відеоконтенту, що дозволяє персоналізувати процес підбору відеоконтенту для потреб конкретного користувача.

Призначення роботи – розробка програмного продукту для підбору відеоконтенту.

4. Вихідні дані для проведення НДР

1. Phillips B., Stewart C., Hardy K. Android Programming: The Big Nerd Ranch Guide – 2021. – 688 с.

2. Meier R. Professional Android – 2018. – 928 с.

3. Nash G. Kotlin Programming: The Big Nerd Ranch Guide – 2019. – 600 с.

5. Технічні вимоги

Смартфон з процесором Qualcomm Snapdragon 400. Об'єм оперативної пам'яті 2 ГБ, розмір вільного місця на жорсткому диску 1 ГБ. Операційна система Android KitKat.

6. Конструктивні вимоги.

Додаток має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку мобільних систем для підбору відеоконтенту та постановка задач дослідження	14.03.24 – 24.03.24
2	Розробка методу, моделі та алгоритмів програмного продукту	25.03.24 – 15.04.24
3	Розробка програмних модулів реалізації мобільної системи для підбору відеоконтенту	15.04.24 – 30.04.24
4	Тестування програми	01.05.24 – 10.05.24
5	Оформлення матеріалів до захисту БДР	11.05.24 – 30.05.24

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Додаток Б – ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ
РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка методу та програмних засобів реалізації мобільної системи підбору відеоконтенту "Topical Films"»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Войтко В.В.

Оригінальність	94,7%
Схожість	5,3%

Аналіз звіту подібності

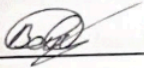
■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

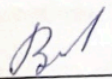
□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку:  Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unichack

Автор роботи:  Бабійчук І. С

Керівник роботи:  Войтко В.В

Додаток В – Лістинг програми

```
package com.example.movie.ui.films

import android.annotation.SuppressLint
import android.content.Intent
import android.content.res.Configuration
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.core.app.ActivityOptionsCompat
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.lifecycle.lifecycleScope
import androidx.recyclerview.widget.DefaultItemAnimator
import androidx.recyclerview.widget.GridLayoutManager
import com.bumptech.glide.RequestManager
import com.example.movie.R
import com.example.movie.adapter.FilmAdapter
import com.example.movie.databinding.BestFilmFragmentBinding
import com.example.movie.model.result.ResultFilm
import com.example.movie.util.launchWhenStarted
import com.example.movie.util.toastError
import com.example.movie.util.viewBinding
import com.example.movie.view.DetailsActivity
import com.example.movie.viewmodel.FilmFragmentViewModel
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject
import kotlinx.coroutines.flow.onEach

@AndroidEntryPoint
class BestFilmFragment : Fragment(R.layout.best_film_fragment) {
```

@Inject

lateinit var glideRequestManager: RequestManager

private val binding by viewBinding(BestFilmFragmentBinding::inflate)

private lateinit var resultFilmArrayList: ArrayList<ResultFilm>

private val fragmentViewModel by activityViewModels<FilmFragmentViewModel>()

override fun onCreateView(

inflater: LayoutInflater,

container: ViewGroup?,

savedInstanceState: Bundle?

): View {

return binding.root

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

with(binding) {

pbBestFilm.show()

fragmentViewModel.updateBestFilm()

fragmentViewModel.getSuccessRetrofitRequest().onEach { success ->

success?.let {

fragmentViewModel.getBestFilmFlow().onEach { results ->

results?.let {

resultFilmArrayList = it as ArrayList<ResultFilm>

bestFilmSwipeRefresh.isRefreshing = false

pbBestFilm.hide()

tvNotConnect.visibility = View.GONE

btnRetryConnect.visibility = View.GONE

rvBestFilm.visibility = View.VISIBLE

fillRecyclerView()

}

}.launchWhenStarted(lifecycleScope)

}

```

    }.launchWhenStarted(lifecycleScope)
    fragmentViewModel.getErrorRetrofitRequest().onEach {
        it?.let {
            it.toastError(requireContext())
            bestFilmSwipeRefresh.isRefreshing = false
            pbBestFilm.hide()
            tvNotConnect.visibility = View.VISIBLE
            btnRetryConnect.visibility = View.VISIBLE
            rvBestFilm.visibility = View.GONE
        }
    }.launchWhenStarted(lifecycleScope)

    btnRetryConnect.setOnClickListener {
        fragmentViewModel.updateBestFilm()
        tvNotConnect.visibility = View.GONE
        btnRetryConnect.visibility = View.GONE
        pbBestFilm.show()
    }

    bestFilmSwipeRefresh.setColorSchemeResources(R.color.colorPrimary)
    bestFilmSwipeRefresh.setOnRefreshListener { fragmentViewModel.updateBestFilm()
}

}

}

@SuppressLint("NotifyDataSetChanged")
private fun fillRecyclerView() {

    val filmAdapter = FilmAdapter(
        resultFilmArrayList,
        glideRequestManager
    ) { film, view, text ->
        startDetails(film, view, text)
    }
}

```

```

with(binding) {
    if (resources.configuration.orientation ==
        Configuration.ORIENTATION_PORTRAIT
    ) {
        rvBestFilm.layoutManager = GridLayoutManager(context, 2)
    } else {
        rvBestFilm.layoutManager = GridLayoutManager(context, 4)
    }

    rvBestFilm.itemAnimator = DefaultItemAnimator()
    rvBestFilm.adapter = filmAdapter
    filmAdapter.notifyDataSetChanged()
    bestFilmSwipeRefresh.isRefreshing = false
}
}

private fun startDetails(film: ResultFilm, view: View, text: TextView) {
    val intent = Intent(context, DetailsActivity::class.java)

    val ivPair = androidx.core.util.Pair.create<View, String>(view, "ivMovieToDetail")
    val tvPair = androidx.core.util.Pair.create<View, String>(text, "tvTitleMovieToDetail")

    val options = ActivityOptionsCompat.makeSceneTransitionAnimation(
        this.requireActivity(),
        ivPair,
        tvPair
    )
    intent.putExtra(DetailsActivity.MOVIE_DATA_EXTRA, film.id)
    startActivity(intent, options.toBundle())
}
}

```

```

package com.example.movie.ui.films

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.NavigationUI
import com.example.movie.R
import com.example.movie.databinding.FilmsFragmentBinding
import com.example.movie.util.viewBinding

class FilmsFragment : Fragment(R.layout.films_fragment) {

    private val binding by viewBinding(FilmsFragmentBinding::inflate)

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val navController =
            (childFragmentManager.findFragmentById(R.id.fcvFilms)
NavHostFragment).navController
        NavigationUI.setupWithNavController(binding.navigation, navController)
    }
}

```

as

```

package com.example.movie.ui.films

import android.annotation.SuppressLint
import android.content.Intent
import android.content.res.Configuration
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.core.app.ActivityOptionsCompat
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.lifecycle.lifecycleScope
import androidx.recyclerview.widget.DefaultItemAnimator
import androidx.recyclerview.widget.GridLayoutManager
import com.bumptech.glide.RequestManager
import com.example.movie.R
import com.example.movie.adapter.FilmAdapter
import com.example.movie.databinding.PopularFilmFragmentBinding
import com.example.movie.model.result.ResultFilm
import com.example.movie.util.launchWhenStarted
import com.example.movie.util.toastError
import com.example.movie.util.viewBinding
import com.example.movie.view.DetailsActivity
import com.example.movie.viewmodel.FilmFragmentViewModel
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject
import kotlinx.coroutines.flow.onEach

@AndroidEntryPoint
class PopularFilmFragment : Fragment(R.layout.popular_film_fragment) {

    @Inject
    lateinit var glideRequestManager: RequestManager

```

```

private val binding by viewBinding(PopularFilmFragmentBinding::inflate)

private lateinit var resultFilmArrayList: ArrayList<ResultFilm>
private val fragmentViewModel by activityViewModels<FilmFragmentViewModel>()

override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    with(binding) {
        pbPopularFilm.show()
        fragmentViewModel.updatePopularFilm()
        fragmentViewModel.getSuccessRetrofitRequest().onEach {
            it?.let {
                fragmentViewModel.getPopularMovieFlow().onEach { results ->
                    resultFilmArrayList = results as ArrayList<ResultFilm>
                    popularFilmSwipeRefresh.isRefreshing = false
                    pbPopularFilm.hide()
                    tvNotConnect.visibility = View.GONE
                    btnRetryConnect.visibility = View.GONE
                    rvPopularFilm.visibility = View.VISIBLE
                    fillRecyclerView()
                }.launchWhenStarted(lifecycleScope)
            }
        }.launchWhenStarted(lifecycleScope)
        fragmentViewModel.getErrorRetrofitRequest().onEach {
            it?.let {
                it.toastError(requireContext())
            }
        }
    }
}

```

```

        popularFilmSwipeRefresh.isRefreshing = false
        pbPopularFilm.hide()
        tvNotConnect.visibility = View.VISIBLE
        btnRetryConnect.visibility = View.VISIBLE
        rvPopularFilm.visibility = View.GONE
    }
}

.launchWhenStarted(lifecycleScope)

btnRetryConnect.setOnClickListener {
    fragmentViewModel.updatePopularFilm()
    tvNotConnect.visibility = View.GONE
    btnRetryConnect.visibility = View.GONE
    pbPopularFilm.show()
}

popularFilmSwipeRefresh.setColorSchemeResources(R.color.colorPrimary)
popularFilmSwipeRefresh.setOnRefreshListener {
    fragmentViewModel.updatePopularFilm() }
}
}

@SuppressLint("NotifyDataSetChanged")
private fun fillRecyclerView() {

    val filmAdapter = FilmAdapter(
        resultFilmArrayList,
        glideRequestManager
    ) { film, view, text ->
        startDetails(film, view, text)
    }

    with(binding) {
        if (resources.configuration.orientation ==
            Configuration.ORIENTATION_PORTRAIT
        ) {

```



```

        rvPopularFilm.layoutManager = GridLayoutManager(context, 2)
    } else {
        rvPopularFilm.layoutManager = GridLayoutManager(context, 4)
    }

    rvPopularFilm.itemAnimator = DefaultItemAnimator()
    rvPopularFilm.adapter = filmAdapter
    filmAdapter.notifyDataSetChanged()
    popularFilmSwipeRefresh.isRefreshing = false
}
}

private fun startDetails(film: ResultFilm, view: View, text: TextView) {
    val intent = Intent(context, DetailsActivity::class.java)

    val ivPair = androidx.core.util.Pair.create<View, String>(view, "ivMovieToDetail")
    val tvPair = androidx.core.util.Pair.create<View, String>(text, "tvTitleMovieToDetail")

    val options = ActivityOptionsCompat.makeSceneTransitionAnimation(
        this.requireActivity(),
        ivPair,
        tvPair
    )
    intent.putExtra(DetailsActivity.MOVIE_DATA_EXTRA, film.id)
    startActivity(intent, options.toBundle())
}
}

package com.example.movie.ui

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup

```

```

import android.widget.AdapterView
import androidx.fragment.app.DialogFragment
import androidx.fragment.app.activityViewModels
import androidx.lifecycle.lifecycleScope
import com.example.movie.R
import com.example.movie.databinding.DialogFilterViewBinding
import com.example.movie.model.result.ResultGenre
import com.example.movie.util.*
import com.example.movie.viewmodel.FilmFragmentViewModel
import com.google.android.material.chip.Chip
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.onEach

@AndroidEntryPoint
class FilterDialogFragment : DialogFragment(R.layout.dialog_filter_view),
    AdapterView.OnItemClickListener {

    private val viewModel by activityViewModels<FilmFragmentViewModel>()

    private val binding by viewBinding(DialogFilterViewBinding::inflate)

    private lateinit var genreList: ArrayList<ResultGenre>

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        dialog?.window?.setBackgroundDrawableResource(R.drawable.round_corner)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        if (viewModel.getGenreFilmFlow().value == null) {

```

```

viewModel.getGenreFilm()
}
with(binding) {
    val genreIdActive = viewModel.getGenre()
    viewModel.getGenreFilmFlow().onEach { resultList ->
        if (resultList != null) {
            genreList = resultList as ArrayList<ResultGenre>
            genreList.forEach {
                val inflater = LayoutInflater.from(context)
                val newChip = inflater.inflate(
                    R.layout.chip_layout_filter, cgGenre,
                    false
                ) as Chip
                newChip.text = it.name
                if (it.id.toString() == genreIdActive) {
                    newChip.isChecked = true
                }
                cgGenre.addView(newChip)
            }
        }
    }.launchWhenStarted(lifecycleScope)
}

```

```

val inflater = LayoutInflater.from(context)
val nChip = inflater.inflate(
    R.layout.chip_layout_filter, cgGenre,
    false
) as Chip
nChip.text = KEY_GENRE_ALL
if (KEY_GENRE_ALL == genreIdActive) {
    nChip.isChecked = true
}
cgGenre.addView(nChip)

```

```

SORT.forEach {
    val inflater = LayoutInflater.from(context)

```

```

        val newChip = inflater.inflate(
            R.layout.chip_layout_filter, cgSort,
            false
        ) as Chip
        newChip.text = it
        if (it == viewModel.getSort()) {
            newChip.isChecked = true
        }
        cgSort.addView(newChip)
    }
    btnSave.setOnClickListener { saveFilterInfo() }
}

private fun saveFilterInfo() {
    with(binding) {
        val genreId = cgGenre.checkedChipId
        val sortId = cgSort.checkedChipId
        if (genreId >= 0) {
            val chipGenre: Chip = cgGenre.findViewById(genreId)

            if (chipGenre.text.toString() == KEY_GENRE_ALL) {
                viewModel.setGenreFilm(KEY_GENRE_ALL)
            } else {
                genreList.forEach {
                    if (it.name == chipGenre.text.toString()) {
                        viewModel.setGenreFilm(it.id.toString())
                    }
                }
            }
        }
    }

    if (sortId >= 0) {
        val chipSort: Chip = cgSort.findViewById(sortId)
        viewModel.setSortFilm(chipSort.text.toString())
    }
}

```

```

        }
        viewModel.updateUi()
        dismiss()
    }
}

override fun onStart() {
    super.onStart()
    val width = (resources.displayMetrics.widthPixels * 0.85).toInt()
    dialog!!.window?.setLayout(width, ViewGroup.LayoutParams.WRAP_CONTENT)
}

override fun onItemSelected(parent: AdapterView<*>?, view: View?, position: Int, id:
Long) {
    viewModel.setGenreFilm(GENRES[position])
}

override fun onNothingSelected(parent: AdapterView<*>?) {
}
}

package com.example.movie.data

import android.content.Context
import com.example.movie.util.*
import javax.inject.Inject

class Storage @Inject constructor(context: Context) {

    private val sharedPreferences = context.getSharedPreferences(
        PREFERENCE_NAME,
        Context.MODE_PRIVATE
    )

    fun addPreferencesSession(value: String) = with(sharedPreferences.edit()) {

```

```

        putString(KEY_SESSION, value).apply()
    }

    fun addPreferencesUserName(value: String) = with(sharedPreferences.edit()) {
        putString(KEY_USER_NAME, value).apply()
    }

    fun addPreferencesName(value: String) = with(sharedPreferences.edit()) {
        putString(KEY_NAME, value).apply()
    }

    fun addPreferencesId(value: String) = with(sharedPreferences.edit()) {
        putString(KEY_USER_ID, value).apply()
    }

    fun addPreferencesImageProfile(tmdb: String?) = with(sharedPreferences.edit()) {
        putString(KEY_USER_IMAGE_PROFILE, tmdb).apply()
    }

    fun readPreferencesSession() = sharedPreferences.getString(KEY_SESSION, "")
    fun readPreferencesUserName() = sharedPreferences.getString(KEY_USER_NAME, "")
    fun readPreferencesName() = sharedPreferences.getString(KEY_NAME, "")
    fun readPreferencesId() = sharedPreferences.getString(KEY_USER_ID, "")
    fun
        readPreferencesImageProfile()
    =
sharedPreferences.getString(KEY_USER_IMAGE_PROFILE, "")

    fun clearStorage() = sharedPreferences.edit().clear().apply()
}

package com.example.movie.viewmodel

import androidx.lifecycle.ViewModel
import com.example.movie.model.User
import com.example.movie.model.repository.MovieRepository
import com.example.movie.model.repository.SearchRepository

```

```

import com.example.movie.model.repository.UserRepository
import com.example.movie.model.result.ResultFilm
import com.example.movie.model.result.ResultGenre
import com.example.movie.model.search.MultiSearchApiResponse
import com.example.movie.util.KEY_GENRE_ALL
import com.example.movie.util.KEY_SORT_POPULARITY
import com.example.movie.util.RetrofitListener
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import javax.inject.Inject

```

```
@HiltViewModel
```

```

class FilmFragmentViewModel @Inject constructor(
    private val movieRepository: MovieRepository,
    private val userRepository: UserRepository,
    private val searchRepository: SearchRepository
) : ViewModel() {

```

```

        private val successRetrofitRequestFlow: MutableStateFlow<Boolean?> =
MutableStateFlow(null)

```

```

        private val errorRetrofitRequestFlow: MutableStateFlow<Throwable?> =
MutableStateFlow(null)

```

```

        private val listener = object : RetrofitListener {
            override fun onSuccess() {
                successRetrofitRequestFlow.value = true
                errorRetrofitRequestFlow.value = null
            }

```

```

            override fun onError(throwable: Throwable) {
                errorRetrofitRequestFlow.value = throwable
                successRetrofitRequestFlow.value = null
            }
        }
    }

```

```

private var genre = KEY_GENRE_ALL
private var sort = KEY_SORT_POPULARITY

fun getSuccessRetrofitRequest(): StateFlow<Boolean?> = successRetrofitRequestFlow
fun getErrorRetrofitRequest(): StateFlow<Throwable?> = errorRetrofitRequestFlow
fun          getWatchingNowFlow():          StateFlow<List<ResultFilm>?>          =
movieRepository.getWatchNowFlow()
fun          getWatchListFlow():          StateFlow<List<ResultFilm>?>          =
movieRepository.getWatchListFlow()
fun          getRatedFilmFlow():          StateFlow<List<ResultFilm>?>          =
movieRepository.getRatedMovieFlow()
fun          getGenreFilmFlow():          StateFlow<List<ResultGenre>?>          =
movieRepository.getGenreFlow()
fun          getSearchFlow():          StateFlow<MultiSearchApiResponse?>          =
searchRepository.getSearchFlow()
fun getUserFlow(): StateFlow<User?> = userRepository.getUserFlow()
fun getRecommendationsFilmFlow(): StateFlow<List<ResultFilm>?> =
    movieRepository.getRecommendationsFlow()

fun getExpectedFlow(): StateFlow<List<ResultFilm>?> =
    movieRepository.getExpectedMovieFlow()

fun getBestFilmFlow(): StateFlow<List<ResultFilm>?> =
    movieRepository.getTheBestMovieFlow()

fun getFavoriteFilmFlow(): StateFlow<List<ResultFilm>?> =
    movieRepository.getFavoriteFlow()

fun getPopularMovieFlow(): StateFlow<List<ResultFilm>?> =
    movieRepository.getPopularityMovieFlow()

fun updateUser() = userRepository.getAccount()

fun getGenreFilm() = movieRepository.getGenreMovies(listener)

```



```

fun updatePopularFilm() {
    if (genre == KEY_GENRE_ALL && sort == KEY_SORT_POPULARITY) {
        movieRepository.getPopularMovies(listener)
    } else {
        if (genre == KEY_GENRE_ALL) {
            movieRepository.getFilmBySort(sort, listener)
        } else {
            movieRepository.getFilmByGenreAndSort(genre, sort, listener)
        }
    }
}

```

```

fun updateWatchingNow() = movieRepository.getWatchingNow(listener)

```

```

fun updateExpected() = movieRepository.getExpected(listener)

```

```

fun updateBestFilm() = movieRepository.getTheBestFilm(listener)

```

```

fun updateFavoriteFilm(successListener: (status: Boolean) -> Unit) =
    movieRepository.getFavoriteMovies(listener, successListener)

```

```

fun updateRatedFilm() = movieRepository.getRatedMovies(listener)

```

```

fun updateWatchList() = movieRepository.getWatchList(listener)

```

```

fun                                updateRecommendationsFilm()                                =
movieRepository.getRecommendationsMovies(listener)

```

```

fun singOutUser() = userRepository.deleteSession()

```

```

fun multiSearch(textSearch: String) = searchRepository.multiSearch(textSearch)

```

```

fun updateUi() = updatePopularFilm()

```

```

fun setGenreFilm(genre: String) {
    this.genre = genre
}

fun setSortFilm(sort: String) {
    this.sort = sort
}

fun getGenre(): String = genre

fun getSort() = sort

}

package com.example.movie.view

import android.os.Bundle
import android.widget.Toast
import com.example.movie.databinding.YoutubeActivityBinding
import com.example.movie.model.result.ResultTrailerMovie
import com.example.movie.util.API_KEY_YOUTUBE
import com.google.android.youtube.player.YouTubeBaseActivity
import com.google.android.youtube.player.YouTubeInitializationResult
import com.google.android.youtube.player.YouTubePlayer

class YouTubeActivity : YouTubeBaseActivity() {
    companion object {
        const val YOUTUBE_EXTRA = "youTubeData"
    }

    private lateinit var resultTrailer: ResultTrailerMovie

    private lateinit var binding: YoutubeActivityBinding

```

```

private lateinit var youtubePlayerInit: YouTubePlayer.OnInitializedListener

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = YoutubeActivityBinding.inflate(layoutInflater)
    setContentView(binding.root)

    val intent = intent
    if (intent != null && intent.hasExtra(YOUTUBE_EXTRA)) {
        resultTrailer = intent.getParcelableExtra(YOUTUBE_EXTRA)!!
        with(binding) {
            youtubePlayerInit = object : YouTubePlayer.OnInitializedListener {
                override fun onInitializationSuccess(
                    p0: YouTubePlayer.Provider?,
                    p1: YouTubePlayer?,
                    p2: Boolean
                ) {
                    p1?.loadVideo(resultTrailer.key)
                }

                override fun onInitializationFailure(
                    p0: YouTubePlayer.Provider?,
                    p1: YouTubeInitializationResult?
                ) {
                    Toast.makeText(applicationContext, "Failed",
Toast.LENGTH_SHORT).show()
                }
            }
            youtubePlayerView.initialize(API_KEY_YOUTUBE, youtubePlayerInit)
        }
    }
}

```

ІЛЮСТРАТИВНА ЧАСТИНА

**РОЗРОБКА МЕТОДУ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ МОБІЛЬНОЇ
СИСТЕМИ ПІДБОРУ ВІДЕОКОНТЕНТУ "TOPICAL FILMS"**

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА КОМП'ЮТЕРНОЇ
ІНЖЕНЕРІЇ
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА НА
ТЕМУ:

«РОЗРОБКА МЕТОДУ ТА ПРОГРАМНИХ
ЗАСОБІВ РЕАЛІЗАЦІЇ МОБІЛЬНОЇ
СИСТЕМИ ПІДБОРУ
ВІДЕОКОНТЕНТУ "TOPICAL FILMS"»

Автор: ст. гр. ЗПІ-20Б Бабійчук І.С.
Науковий керівник: к.т.н., доц. каф. ПЗ Войтко В. В.

Вінниця 2024



Рисунок Г.1 – Титульний слайд

АКТУАЛЬНІСТЬ РОЗРОБКИ

- зростання популярності онлайн-відеоконтенту;
- потреба у покращенні алгоритмів персоналізованих рекомендацій;
- інтеграція ефективних механізмів завантаження та зберігання відео дозволяє користувачам зберігати відео у внутрішній пам'яті пристрою, відстежувати прогрес завантаження та отримувати доступ до збережених відео без необхідності підключення до Інтернету;
- забезпечення безперервного доступу до відеоконтенту, навіть за відсутності Інтернет-з'єднання, значно покращує користувацький досвід.

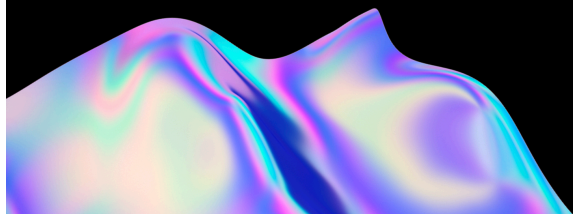


Рисунок Г.2 – Слайд «Актуальність розробки»

МЕТА, ПРЕДМЕТ ТА ОБ'ЄКТ ДОСЛІДЖЕННЯ

Мета

Метою бакалаврської дипломної роботи є покращення можливостей підбору відеоконтенту шляхом розробки та використання спеціалізованого програмного продукту, який дозволяє проводити аналіз переглядів користувача, визначати їх вподобання та надавати рекомендації й здійснювати пошук потрібного відеоконтенту, що дозволяє персоналізувати процес підбору відеоконтенту для потреб конкретного користувача.

Об'єкт дослідження

процес розробки та впровадження мобільної системи підбору відеоконтенту "Topical Films".

Предмет дослідження

методи та засоби, які дозволять ефективно відбирати та рекомендувати відеоконтент користувачам з урахуванням їхніх індивідуальних уподобань та інтересів.

Рисунок Г.3 – Слайд «Мета, предмет та об'єкт дослідження»

ПОСТАНОВКА ЗАДАЧІ

- провести аналіз розвитку мобільних систем для підбору відео-контенту та постановка задач дослідження;
- розробити програмні модулі реалізації мобільної системи для підбору відеоконтенту ;
- програмно реалізувати функціонал системи для підбору відеоконтенту;
- розробити модель системи для підбору відеоконтенту яка включає основний функціонал застосунку ;
- провести тестування розроблених засобів для підбору відеоконтенту;
- запропонувати новий метод рекомендацій та завантаження відеоконтенту;

Рисунок Г.4 – Слайд «Постановка задачі»

НОВИЗНА

Подальшого розвитку набув метод персоналізованого підбору відеоконтенту, який, на відміну від існуючих, враховує індивідуальні вподобання користувачів, історію переглядів та інші дії, що дозволяє надавати персоналізовані рекомендації, які максимально відповідають очікуванням користувачів та підвищують їх задоволення від використання додатку.

Подальшого розвитку набула модель зберігання та захисту відеоконтенту, яка, на відміну від існуючих, включає шифрування завантажених відеофайлів для захисту від несанкціонованого доступу та забезпечує можливість завантаження контенту для перегляду в офлайн-режимі, що значно підвищує зручність роботи користувачів та забезпечує високу доступність відеоконтенту.

Рисунок Г.5 – Слайд «Новизна»

ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Практична цінність одержаних результатів полягає в можливості використання розробленої програми, щоб значно заощадити час користувачів, які шукають відеоконтент, а також дозволить розширити їхні інтереси через персоналізовані рекомендації. Крім того, система може бути використана в навчальних та рекламних цілях, що робить її значним активом у сучасному інформаційному середовищі.



Рисунок Г.6 – Слайд «Практична цінність отриманих результатів»



Рисунок Г.7 – Слайд «Існуючі аналогі»

АНАЛІЗ АНАЛОГІВ

Критерії	YouTube	Netflix	Hulu	Amazon Prime Video	Disney+	Topical Films
Різноманітність контенту	+	-	+	-	-	+
Безкоштовний повний доступ до контенту	-	-	-	-	-	+
Доступність для мобільних девайсів	+	+	+	+	+	+
Можливості персоналізації	+	+	+	-	-	+
Сучасні алгоритми рекомендацій	+	+	-	+	+	+
Підтримка оф-лайн режиму	+	-	-	-	-	+
Загальна оцінка	83%	50%	50%	33%	33%	100%

Рисунок Г.8 – Слайд «Аналіз аналогів»

Метод персоналізованого підбору відеоконтенту (ч.1)

Авторизація користувача

- Після запуску програми користувачеві пропонується увійти в обліковий запис або створити новий.
- Клас `AuthViewModel` обробляє введені облікові дані та здійснює перевірку через клас `DatabaseConnection`.
- Якщо дані вірні, користувач успішно входить у систему.

Збір та аналіз історії переглядів

- Після авторизації програма збирає дані про попередні перегляди користувача.
- Клас `HistoryViewModel` зчитує історію переглядів з бази даних за допомогою класу `SqlDataReader` і зберігає її в колекцію `ObservableCollection<Video>`.
- Алгоритм машинного навчання в класі `RecommendationEngine` аналізує зібрані дані та формує профіль вподобань користувача.

Генерація персоналізованих рекомендацій

- На основі профілю вподобань користувача клас `RecommendationEngine` генерує список віорекомендацій.
- Отримані рекомендації додаються в колекцію `ObservableCollection<VideoRecommendation>` та відображаються користувачеві за допомогою класу `RecommendationsView`.

Рисунок Г.9 – Слайд «Метод персоналізованого підбору відеоконтенту (ч.1)»

Метод персоналізованого підбору відеоконтенту (ч.2)

Вибір відео з рекомендацій

- Користувач обирає відео для перегляду або завантаження.
- Клас `VideoSelectionViewModel` обробляє вибір користувача та надає детальну інформацію про обране відео.

Завантаження відео

- Користувач може обрати відео для завантаження.
- Клас `DownloadManager` розпочинає процес завантаження, використовуючи клас `FileHandler` для збереження файлу.
- Прогрес завантаження відображається в інтерфейсі користувача за допомогою класу `DownloadProgressView`.

Перегляд завантажених відео офлайн

- Завантажені відео зберігаються у внутрішній пам'яті пристрою.
- Клас `OfflineStorage` зберігає інформацію про завантажені відео та забезпечує їх відображення в інтерфейсі користувача через клас `OfflineVideosView`.
- Користувач може переглядати завантажені відео без підключення до Інтернету.

Рисунок Г.10 – Слайд «Метод персоналізованого підбору відеоконтенту (ч.2)»

Блок-схема алгоритму роботи методу персоналізованого підбору відеоконтенту



Рисунок Г.11 – Слайд «Блок-схема алгоритму роботи методу персоналізованого підбору відеоконтенту»

Модель системи у вигляді діаграми діяльності

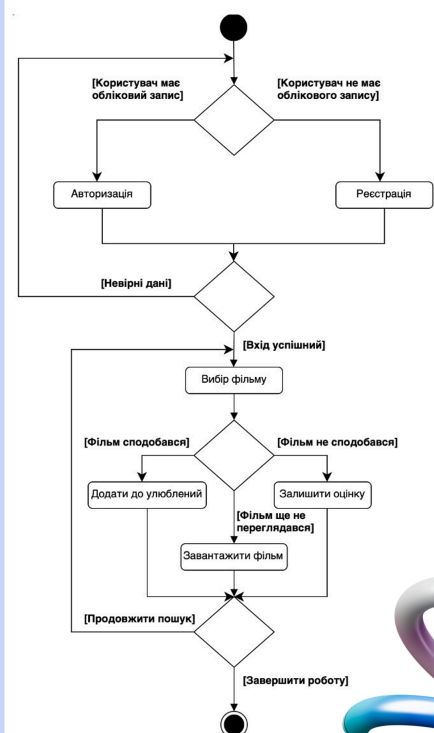


Рисунок Г.12 – Слайд «Модель системи у вигляді діаграми діяльності»

Блок-схема алгоритму роботи мобільної системи підбору відеоконтенту

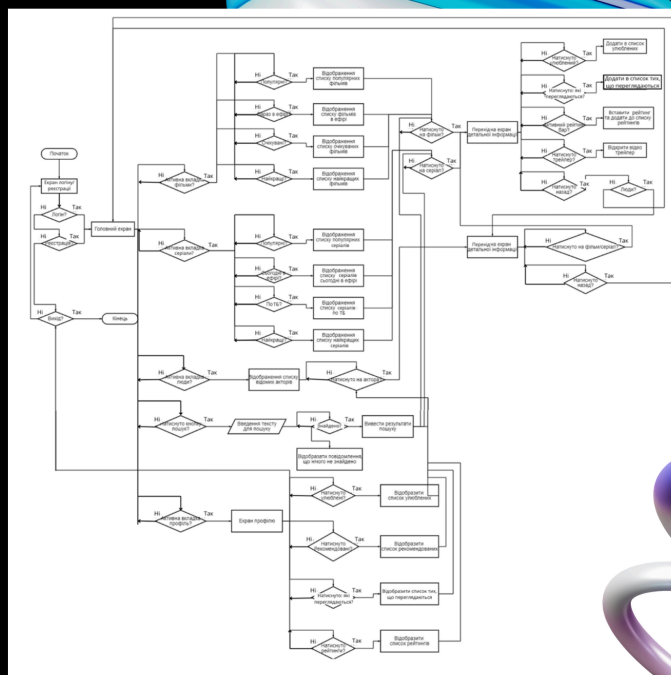


Рисунок Г.13 – Слайд «Блок-схема алгоритму роботи мобільної системи підбору відеоконтенту»

Розробка графічного інтерфейсу

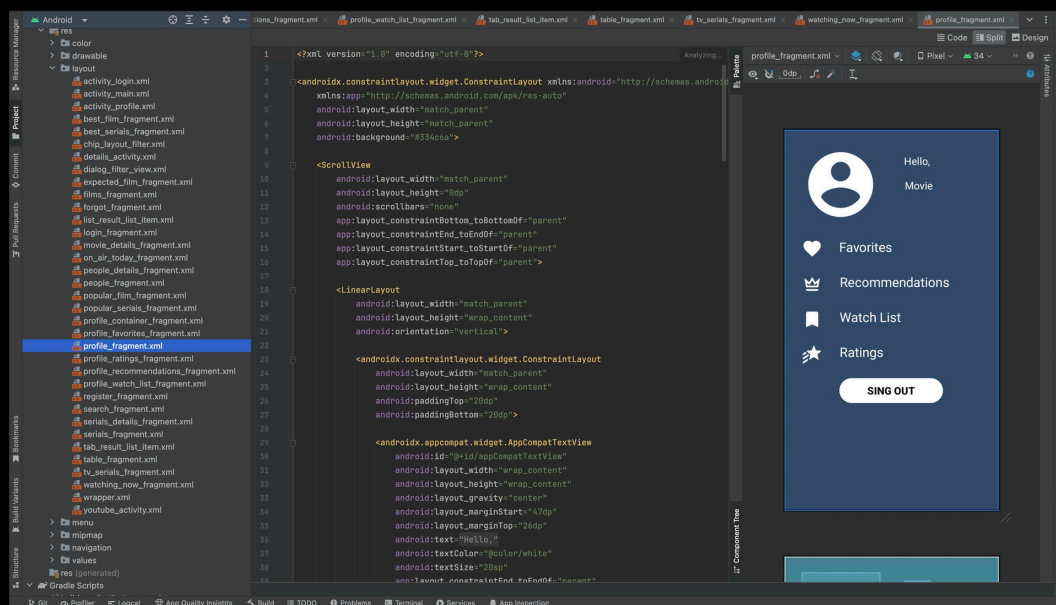


Рисунок Г.14 – Слайд «Розробка графічного інтерфейсу»

Використані технології



Рисунок Г.15 – Слайд «Використані технології»

Функціонал системи

Сторінка авторизації користувача

Екран «Профіль»

Екран «Фільми – Популярні»

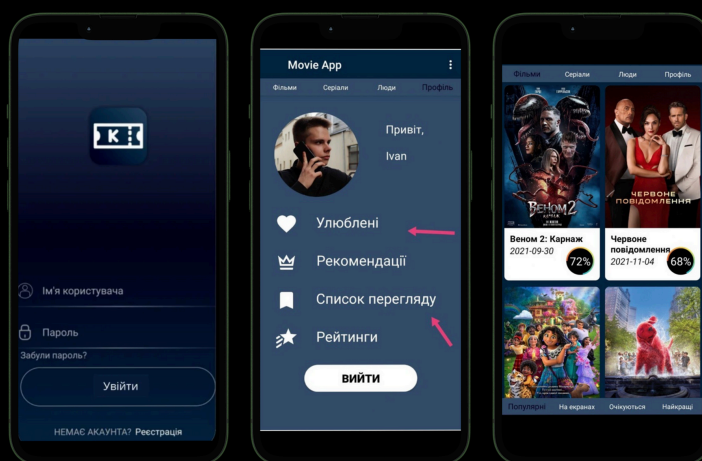


Рисунок Г.16 – Слайд «Функціонал системи»



Рисунок Г.17 – Слайд «Демонстрація роботи»

АПРОБАЦІЯ ТА ПУБЛІКАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

Матеріали бакалаврської дипломної роботи доповідалися на: LIII Науково-технічній конференції підрозділів Вінницького національного технічного університету (2024)». Вінниця, 2024. За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференції.

УДК 681.3.07

**В. В. Войтко
А. В. Денисюк
О. В. Гаврилюк
Н. С. Барчук
І. С. Бабійчук**

**РОЗРОБКА МОБІЛЬНОЇ СИСТЕМИ ПІДБОРУ
ВІДЕОКОНТЕНТУ "TORICAL FILMS"**

Вінницький національний технічний університет

Анотація
Проведено порівняльний аналіз аналогів мобільної системи для підбору відеоконтенту, спрямований на задоволення індивідуальних потреб користувачів. Визначено функціонал власної розробки. Побудований алгоритм роботи мобільної системи.
Ключові слова: мобільна система, автоматизація, відеоконтент.

Abstract
A comparative analysis of mobile system counterparts for video content curation aimed at meeting individual user needs has been conducted. The functionality of the proprietary development has been determined. An algorithm for the system's operation has been constructed.
Keywords: Mobile system, automation, video content.

Вступ

У сучасній цифровій епосі споживання відеоконтенту стало невід'ємною складовою щоденного життя для багатьох людей. З величезним розмаїттям відеоконтенту, доступного на різних платформах, користувачі часто почувують себе перенасиченими вибором, маючи складнощі у пошуку відео, яке відповідає їхнім інтересам та уподобанням. Щоб вирішити цю проблему, розробка інноваційних методів та інструментів для рекомендацій відеоконтенту стає все більш важливою.

Мобільна система "Tropical Films" являє собою передовий крок у цьому напрямку, метою якого є визначення способу, яким користувачі обирають відеоконтент для перегляду. Система спрямована на проєктування та реалізацію високотехнологічних засобів, здатних автоматично вибирати та рекомендувати відеоматеріали, які відповідають індивідуальним потребам й уподобанням користувачів. Використовуючи передові алгоритми та технології, "Tropical Films" має на меті спростити процес відкриття контенту, надаючи користувачам можливість персоналізованого вибору та комфортного перегляду.

Об'єктом дослідження є процес розробки мобільної системи для підбору відеоконтенту. Предметом дослідження є методи і засоби реалізації мобільної системи для підбору відеоконтенту. Головною задачею є розробка мобільної системи, завдяки якій користувачі зможуть ефективно керувати пошуком відеоматеріалів.

Порівняння аналогів та розробка мобільної системи підбору відеоконтенту

Перед розробкою мобільної системи "Tropical Films" було проведено аналіз аналогічних систем для підбору відеоконтенту. Основні аспекти порівняння включали функціонал, ефективність рекомендаційних алгоритмів, зручність інтерфейсу користувача, інтеграцію з популярними відеоплатформами та рівень захисту даних.

Враховуючи переваги й недоліки систем-аналогів, було визначено функціонал власної розробки системи підбору відеоконтенту.

Розроблена мобільна система надає користувачам змогу швидко знаходити фільми різних жанрів з різними акторами та іншими фільмами. Однією з головних особливостей розробленого програмного продукту є реалізація гнучких рекомендацій, що персоналізовано підлаштовуються під вподобання користувача, а також збереження та оцінка відеоматеріалів.

Рисунок Г.18 – Слайд «Апробація та публікація результатів роботи»

Висновки

На прикладі UML-діаграм діяльності розроблено модель системи для рекомендацій та завантаження відео-контенту, яка включає основний функціонал застосунку.

Програмно реалізовано функціонал системи для рекомендацій та завантаження відео-контенту.

Розроблено програмні компоненти та систему візуалізації на основі запропонованого методу.

Проведено тестування програми, яке показало цілковиту працездатність програмного продукту та відповідність поставленому технічному завданню.

Рисунок Г.19 – Слайд «Висновки»

Дякую за увагу

Бабійчук Іван Сергійович
+380957353016
babiichukzno@gmail.com
<https://github.com/Vanek1756>

Рисунок Г.20 – Фінальний слайд