МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЛАНУВАННЯ ТА ОБЛІКУ ПЕРЕГЛЯДУ ВІДЕО КОНТЕНТА
SOFTWARE DEVELOPMENT FOR SCHEDULING AND TRACKING THE VIEWS OF THE VIDEO CONTENT

Виконав: студент 4 курсу, групи КНТ-218сп
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Комп'ютерні науки

Коротєєв Д.А
(прізвище та ініціали)
Керівник Голуб Т.В
(прізвище та ініціали)
Рецензент Терещенко Е.В
(прізвище та ініціали)

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
"____" _____20___року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Коротєєва Дениса Анатолійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення для планування та обліку перегляду відео контенту
Software Development for Scheduling and Tracking the Views of the Video Content
керівник проєкту (роботи) Голуб Т.В ., к.т.н. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від " 30 " березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) 14 травня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити 1. Аналіз предметної області. 2. Розробка программної частини системи. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Голуб Т.В ., старш викладач.		
Нормоконтроль	Андреєв М.О., асистент		

7. Дата видачі завдання “ 04 ” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування.	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент _____ Коротєєв Д.А.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)_____ Голуб Т.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 77 с., 20 рис., 3 табл., 3 дод., 11 джерел.

РОЗРОБКА, ПРОЕКТУВАННЯ, ОНЛАЙН ЩОДЕННИК, ДИЗАЙН, JAVA SCRIPT.

Об’єкт дослідження – процес розробки та функціонування програмного забезпечення, призначеного для роботи з відеоконтентом.

Предмет дослідження – методи розробки та підтримки функціонування програмного забезпечення, призначеного для роботи з відеоконтентом.

Метою роботи є програмна реалізація додатку, призначеного для роботи з відеоконтентом на прикладі створеня можливості планування та ведення обліку переглянутих роликів, фільмів, відео, тощо.

Матеріали, методи та технічні засоби: мова програмування JavaScript, фреймворк React Native, система керування аналог mongoDB.

Результати. Розроблено додаток, що пропонує користувачу переглядати та записувати свої враження з приводу фільмів або ж планувати їх перегляд в електроному форматі. Розробку реалізовано у вигляді андроїд додатку, що реалізує перегляд матеріалу відео контенту та запису думки щодо переглянутого фільму.

Практична цінність роботи полягає в розробці кроссплатформного додатку для перегляду відео файлів, записних сторінок в цифровому форматі, що реалізує зручні інтерфейси для взаємодії з користувачем та огляду вже переглянутого відео контента і планування його подальшого перегляду.

Висновки. Розроблено програмне забезпечення, що реалізує додаток для планування та перегляду відео контенту.

Галузь використання – користувачі Інтернету, глядачі відео.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 77 pages, 20 figures, 3 tables, 3 appendixes, 11 sources.

DEVELOPMENT, PLANNING, ON-LINE DIARY, DESIGN, JAVA SCRIPT.

A research object is a development and functioning of the software intended for work with videocontent process.

The article of research is methods of development and support of functioning of the software intended for work with videocontent.

The aim of work is programmatic realization to addition, intended for work with videocontent on the example of the creating opportunities planning and maintenance account of the revised rollers, films, videos, and others like that.

Materials, methods and technical equipments: programming of JavaScript language, framework of React Native, control system by the base of mongoDB.

Results. Addition that offers to the user to look over and write down the impressions concerning films or plan their revision in an electro-noma format is worked out. Development is realized as to android addition that will realize the revision of material of video of content and record of idea in relation to the looked over film.

The practical value of work consists in development to cross-platform addition for the revision of video of files, record pages in a digital format that will realize comfortable interfaces for co-operating with an user and review of the already revised video of content and planning of him further revision.

Conclusions. Software that will realize addition for planning and revision of video of content is worked out.

Industry of the use is users of the Internet, audience of video.

ЗМІСТ

ВСТУП

Велике прагнення кожної людини досягти максимального комфорта в кожної з сфер життя, торкнулося і Інтернету. Користувач, бажаючи завжди залишатися в мережі, використовує у якості комунікатора, телефон.

Це зумовило появу мобільного Інтернету. При перебуванні зовні удома, або під час подорожей і відряджень можна замість ноутбука підключатися за допомогою планшета або аналогічного виду техніки. Ефективність і функціональність "міні" комп'ютерів не була б доведена до такого високого рівня без спеціалізованих застосувань.

Розробка мобільних застосувань, яка проводиться виключно фахівцями сфери, розрахована на певне призначення. Деякі програми дозволяють всюди здійснювати з'єднання з мережею, інші вказують маршрут, треті надають допомогу в пошуку магазину або необхідного товару. Є софтвери, які здійснюють замовлення їжі додому. Утиліти лягли в основу повсюдного обміну даними і інформацією, що дозволяє економити дорогоцінний час і ресурси кожного.

Усі основні застосування діляться на ті, які потрібні для приємного проведення часу, і ті, які використовуються виключно в робочих цілях. Перша група включає іграшки і розважальні програми, софтвер для відтворення відео і аудіо матеріалів, засоби для комунікації і багато іншого. Другий напрям розрахований на комплексне рішення певного завдання. Зокрема деякі утиліти здатні контролювати протікання бізнес процесів і складати аналітичні звіти.

Створення мобільних додатків другого типу поширеніше. Продукти діяльності щільно увійшли до таких життєвих напрямів як медицина, державні організації і навіть виробничі компанії. Розважальні утиліти можуть відіграти роль інструментів маркетингу для більшості підприємства, але навіть це не дозволяє їм скласти конкуренцію по сфері застосування діловому напряму.

Впродовж останнього року, показник покупок мобільних пристроїв зріс в рази. Ці дані постійно збільшуються, і нині статистика не міняється. Актуальність і доцільність мобільних застосувань очевидна. Головне, щоб напередодні розробки були чітко поставлені цілі софтвера і його застосування. Утиліта повинна приносити користь, тільки так її роль у комп'ютерному світі буде помітна.

Темою для розробки додатка для дипломного проєкту було обрано розробити мобільне кросплатформове застосування на базі фреймворка React Native з використанням мови javascript і так само нативних мов в міру необхідності для окремих модулів додатка. Цей застосунок - середній по розмірах, за нинішніми мірками, додаток-щоденник для планування, ведення обліку про проглянуті фільми, серіали, мультфільми та інші відеоматеріали з розділенням користувачів, додавання новинок, з відповідною датою виходу і своєчасним нагадуванням. Ці аналоги можна зустріти на звичайних сайтах з фільмами, додатках ніби Netflix і так далі. Особливістю даного застосунку є можливість перегляду бібліотеки фільмів з додатка, також я залишив то що було мені зручно і додав у своє застосування той функціонал, якого мені бракувало і максимально спростив UX до інтуїтивного рівня.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Задачі додатків для перегляду відеоконтенту

В сучасному світі при зростанні темпів життя все менше часу залишається на організацію дозвілля. Тому стає актуальним пошук шляхів для полегшення вирішення даної задачі. Одним із таких шляхів є використання мобільного дотатку, який дозволяє планувати розваги. До датих додатків відносять такі, що спрямовані на організацію перегляду відео контенту.

Загальні задачі такого типу додатку спрямовані на забезпечення наступних можливостей.

1. Перелік відео контенту, який вже було переглянуто.

2. Перелік відео контенту, який заплановано до перегляду.

3. Анонс майбутнього відео контенту чи планування до перегляду майбутнього сюжету.

Перелік відео контенту, який вже переглянуто, дозволяє не тільки згадати попередні перегляди, а й можна залишити свій коментар (або думку) щодо переглянутого контенту.

Перелік відео контенту, який заплановано до перегляду дозволяє користувачу запланувати те, що він хоче продивитися наступним або в інший час, тут можно запланувати перегляд відео контенту у будь-який час.

Анонс майбутнього відео контенту забезпечує додаткову можливість планувати заздалегідь перегляд майбутнього відео, поставити дату та час і нагадування в день, в який виходить контент.

1.2 Аналогічні сучасні системи

Аналогом застосування насправді є звичайна суміш стандартних застосувань в телефоні – замітки і нагадування. По скільки аналога повного функціонала як в моєму додатоку, на жаль, не знайшов, що мене власне і підштовхнуло до розробки. Частковий функціонал є в Netflix, де можна знайти,

подивитися улюблений фільм і додати його в обране, або ж скласти список для перегляду на майбутнє, і досить ще багато аналогів, але ні в одному з них не можна кастомізувати і додати до перегляду те, чого немає на цьому ресурсі. Але ця можливість реалізоавується мобільному застосуванні, що я вважаю украй зручною функцією і обов'язковою до впровадження на аналогічних платформах.

На аналогових платформах є ще один мінус - оплата. Велика частина контенту доступна по підписці і купівлі. Також є можливість узяти в оренду фільм на декілька днів. Менша частина контенту доступна для перегляду безкоштовно по рекламній моделі.

Джерелами монетизації є: коротка відеореклама, яка демонструється користувачам перед переглядом фільму, після закінчення фільму і при натисненні паузи; поштучна купівля фільмів-новинок(розділ "Прем'єри") і аудіокниг; різні по наповненню підписки "Кіно і ТБ".

Монетизація потрібна тому що створення будь-якого сайту та додатку пов'язане з витратами: створення, оплата домену та хостингу, наповнення контентом. При створенні додатку на безкоштовному конструкторі користувачі витрачають свій час. Заробіток дозволяє окупити його зміст і отримати додатковий прибуток, який з часом може перетворитися на основний дохід.

Дане застосування із-за обмеженого функціонала буде позбавлене таких неприємних моментів як підписка або покупки, оскільки все спрощено і кастомізовано для звичайних користувачів для щоденного, зручного і швидкого використання.

1.3 Основні теоретичні положення розробки систем

Сучасні компанії і організації функціонують в умовах великого об'єму постійно змініючийся інформації, яку необхідно оперативно аналізувати і приймати правильні рішення. Бурхливо розвивається обчислювальна техніка і інформаційні технології. Важко знайти зараз компанію, що не займається розвитком інформаційних технологій. Сучасні керівники фірм повністю

усвідомлюють те, що нині успішність і прибутковість компанії повністю залежать у тому числі, і від рівня розвитку IT- технології, швидкості і якості обробки інформації, обгрунтованості і зваженості що приймаються рішенні.

Великою помилкою являється позиція керівників компанії, які, впровадивши одного разу інформаційну систему, перестають нею займатися. Тому процес проектування інформаційних систем нині став обов'язковим. В даному випадку, якщо цей процес не уперше здійснюється компанією, то термін проектування прирівнюється до поняття розвиток інформаційної системи. Цим пояснюється бурхливий розвиток технології проектування інформаційних систем (ІС) останніми роками. Передусім, створення CASE - технології, які у багато скорочують терміни проектування ІС, дозволяють організувати одночасну колективну роботу, оперативно вносити зміни і швидко реагувати на зміну обстановки на підприємстві.

Особливе місце займають сучасні інформаційні технології ведення електронної комерції, робота із замовниками і постачальниками. І в цьому напрямі проектування і розвиток інформаційних систем не можливе без знання основних методології і програмних засобів, що дозволяють в найкоротші терміни і без помилок управляти цими процесами. Компанії отримують колосальні конкурентні переваги, якщо рівень розвитку інформаційних систем відповідає рівню розвитку підприємства. Покращується інвестиційна привабливість компанії, основні бізнес-процес стають прозорими і зрозумілими для контролю і управління, виключаються помилки, брак і пов'язані з ними втрати і часу, і засобів, кінець кінцем, збільшується прибуток компанії.

Основні поняття останніми роками не зазнали сильних змін, формулювання стали точнішими і лаконічнішими, такими, що виключають неоднозначність понять.

Інформація – "зведення(повідомлення, дані) незалежно від форми їх представлення".

Інформаційні технології – "процеси, методи пошуку, збору, зберігання, обробки, надання, поширення інформації і способи здійснення таких процесів і методів".

Інформаційна система – "сукупність інформації яка міститься у базах даних і що забезпечують її обробку інформаційних технологій і технічних засобів".

Проектування інформаційних систем – це впорядкована сукупність методології і засобів створення або модернізації інформаційних систем.

Управління інформаційних системам – "застосування методів управління процесами планування, аналізу, дизайна, створення, впровадження і експлуатації інформаційний системи організації для досягнення її метою".

Життєвий цикл інформаційних систем – "розвиток розглянутої системи в часі, починаючи від задуму і кінчаючи списанням".

Модель життєвого циклу – "структурна основа процесів і дій, що відносяться до життєвого циклу, яка служить в якості загальної посилання для встановлення зв’язків і взаєморозуміння сторін".

Архітектура інформаційних систем – це концепція, що визначає модель, структуру, виконувані функції і взаємозв'язок компонентів інформаційної системи.

Бізнес-процес – це ланцюжок взаємозв'язаних дій, спрямованих на створення товарної продукції або послуги.

Регламент бізнес-процесу – це чітко певний порядок виконання бізнес-процесу, визначальний склад і дій учасників.

Модель даних – це система організації даних і управління ними.

Методологія проектування інформаційних систем - це сукупність принципів проектування(моделювання), виражена впевної концепції.

Засоби моделювання - це програми опису і моделювання систем.

Нотації – це певні способи представлення елементів інформаційної системи.

Реінжиніринг бізнес-процесів - це фундаментальна реорганізація бізнес-процесів з метою підвищення їх ефективності.

Системний підхід – процес розгляду будь-який системи в якості сукупності взаємозв'язаних елементів.

Процесний підхід – представлення любої системи в якості сукупності процесів.

Функціональний підхід – передбачає чітке закріплення за кожної структурної одиниці набору функцій.

Технічне завдання – документ, використовуваний замовником в якості засобу для опису і визначення завдань, що виконуються при реалізації договору.

Типове проектне рішення (ТПР) – це багаторазово використовуване проектне рішення.

Основні процеси життєвого циклу складаються з п'яти процесів, які реалізуються під керуванням основних сторін, залучених в життєвий цикл програмних засобів. Під основної стороною розуміють одну з тих організації, які ініціюють або виконують розробку, експлуатацію або супровід програмних продуктів. Основними сторонами є замовник, постачальник, розробник, оператор і персонал супроводу програмних продуктів. Основними процесами є:

–процес замовлення. Визначає роботи замовника, тобто організації, яка придбаває систему, програмний продукт або програмну послугу;

–процес поставки. Визначає роботи постачальника, тобто організації, яка поставляє систему, програмний продукт або програмну послугу замовникові;

–процес розробки. Визначає роботи розробника, тобто організації, яка проектує і розробляє програмний продукт;

–процес експлуатації. Визначає роботи оператора, тобто організації, яка забезпечує експлуатаційне обслуговування обчислювальної системи в заданих умовах в інтересах користувачів;

–процес супроводу. Визначає роботи персоналу супроводу, тобто організації, яка надає послуги з супроводу програмного продукту, що полягають

в контрольованій зміні програмного продукту з метою збереження його початкового стану і функціональних можливостей. Данний процес охоплює перенесення і зняття з експлуатації програмного продукту.

Допоміжні процеси життєвого циклу складаються з восьми процесів. допоміжний процес є цілеспрямованої складовою частиною іншого процесу, забезпечує успішну реалізацію і якість виконання програмного проекту. Допоміжний процес, при необхідності, ініціюється і використовується іншим процесом. Допоміжними процесами є:

– процес документування. Визначає роботи по опису інформації, видається в процесі життєвого циклу;

– процес управління конфігурацією. Визначає роботи по управлінню конфигурацией;

– процес забезпечення якості. Визначає роботи по об'єктивному забезпеченню того, щоб програмні продукти і процеси відповідали вимогам, встановленим для них, і реалізовувалися у рамках затверджених планів. Спільні аналізи, аудиторські перевірки, верифікація і атестація можуть використовуватися як методи забезпечення якості;

– процес верифікації. Визначає роботи(замовника, постачальника або незалежної сторони) по верифікації програмних продуктів у міру реалізації програмного проекту;

– процес атестації. Визначає роботи(замовника, постачальника або незалежної сторони) по атестації програмних продуктів програмного проекту;

– процес спільного аналізу;

– визначає роботи за оцінкою стану і результатів будь-якої роботи. Данний процес може використовуватися двома будь-якими сторонами, коли одна із сторін(перевіряє) перевіряє іншу сторону(що перевіряється) на спільній нараді;

– процес аудита. Визначає роботи за визначенням відповідності вимогам, планам і договору. Данний процес може використовуватися двома сторонами,

коли одна із сторін(перевіряє) контролює програмні продукти або роботи інший сторони(перевіряється);

– процес рішення проблем. Визначає процес аналізу і усунення проблем(включаючи невідповідності), незалежно від їх характеру і джерела, які були виявлені під час здійснення розробки, експлуатації, супроводу або інших процесів.

1.4 Постановка завдання

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.

Метою є розробка додатка для перегляду відео контенту.

Програма має виконувати наступні функції:

–прийом та обробка запитів користувачів;

–можливість перегляду каталогу фільмів або мультфілмів для користувача (з можливістю фільтрації та редагуванням);

–можливість повідомляти користувача, коли буде вихід даного відеоконтенту, реалізація функції нагадування про реліз;

–можливість змінювати, додавати та видаляти інформацію про відеоконтент;

–можливість запису свох коментарів після перегляду відео контенту.

1.5 Висновки за розділом 1

В даному розділі виконано дослідження предметної області. Розглянуто та визначено основні поняття, які пов’язані з обраною предметною областю.

Також проаналізовані аналоги кросплатформ, було визначено їх переваги та недоліки.

Проведено порівняльний аналіз щодо основних теорій положення розробки систем для перегляду відео контенту. Сформульовано і визначено завдання до роботи.

2 РОЗРОБКА ПРОГРАММНОЇ ЧАСТИНИ СИСТЕМИ

2.1 Програмні засоби для розробки систем

Як вже згадувалося раніше додаток був написаний на мові JavaScript з використанням фреймворка React Native. Перше що треба для початку розробки - оточення(NodeJs). Node або Node.js - програмна платформа, заснована на движку V8(транслюючому JavaScript в машинний код), перетворююча JavaScript з вузькоспеціалізованої мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з облаштуваннями введення-виведення через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript- коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні застосування(за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери(наприклад, tessel, low.js і espruino). У основі Node.js лежить подієво-орієнтоване і асинхронне(чи реактивне) програмування з неблокуючим введенням/виведенням.

React Native це фреймворк для створення кроссплатформенних застосувань на мові JavaScript. Він дозволяє писати додатки для IOS, Android і навіть VR(на React VR можна створювати додатки для шоломів і окулярів віртуальної реальності")[2].

Творець React Native - Facebook, лідер в front - end, компанія, яка вкладає величезні ресурси в розвиток своїх технологій. Facebook активно розвиває React і React Native, створює навколо них цілу інфраструктуру і потужне IT-співтовариство.

React Native дозволяє створювати мобільні застосування, використовуючи при цьому тільки JavaScript з такою ж структурою, що і у React. Це дає можливість складати багатофункціональний мобільний UI із застосуванням декларативних компонентів.

Таблиця 2.1 – Порівняння

Критерії порівняння	React Native	Flutter
Початковий випуск	Травень 2015	Травень 2017
Розробники	Facebook	Google
Мова	JavaScript	Dart
Продуктивність додатків	Close to native	Fast. 60 fps
IDE підтримка	Range of IDE’s and tools with JS support	Android Studio, Visual Studio Code, IntelliJ IDEA
Рідна зовнішність	Lower. Dependency on third-party apps	Better. Has access to device core functionalities
Гаряче перезавантаження	Так	Так
Час виходу на ринок	Повільніше ніж Flutter	Швидше
Повторне використання коду	90% коду багаторазового використання	50-90% коду багаторазового використання
Програми	Tesla, Discord, Instagram	KlasterMe, PostMuse Reflectiy

rXcode інтегроване середовище розробки(IDE) програмного забезпечення для платформ macOS, iOS, watchOS і tvOS, розроблена корпорацією Apple. Перша версія випущена в 2003 році. Стабільні версії поширюються безкоштовно через Mac App Store.

Android Studio – інтегроване середовище розробки(IDE) для роботи з платформою Android, анонсована 16 травня 2013 року на конференції Google I/O.

Ця IDE знаходилася у вільному доступі починаючи з версії 0.1, опублікованою в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася підтримка плагіну Android Development Tools(ADT) для Eclipse.

Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, - офіційний засіб розробки Android додатків. Це середовище розробки доступне для Windows, macOS і GNU/Linux. 17 травня 2017, на щорічній конференції Google I/O, Google анонсував підтримку мови Kotlin, використовуваної в Android Studio, як офіційної мови програмування для платформи Android на додаток до Java і З++.

Завдання мого дипломного проекту і головне метою мого застосунку є розробка програмного забезпечення для планування і для обліку проглянутого відео контенту з окремим розділенням для кожного користувача. Мій додаток повинен мати зрозумілий та зрусний UI інтерфейс та працювати без багів та помилок на смартфонах iOS та Android. Для цього я вивчив мобільні застосування і ринок конкурентних систем схожого характеру і тематики для виявив їх сильні і слабкі сторони і перейняв до свого застосунку тільки найзручніший, корисніший і необхідніший функціонал для досягнення продуктивного і унікального рішення. Адже робота програміста це не коддинг, а вирішення проблеми, яке відповідає попиту споживачів і яке буде максимально зручне, швидке, ефективне і інтуїтивно зрозуміле, і звичайно ж стильне і сучасне для більшого залучення споживачів і підвищенню попиту.

Для розробки додатку був використаний macBook Pro 2017 13 inch. з такими системними характеристиками:

– процесор: Intel Core i5;

– кількість ядер процесора: 4;

–внутрішня тактова частота: 2.3 ГГц (up to 3.8 ГГц) - Технологія пам'яті: LPDDR3 SDRAM;

–швидкість пам'яті: 2133МГц(PC3-17000);

–ємність встановленого ОЗУ: 8 ГБ - Об'єм пам'яті: 128Gb.

Для мінімальної розробки на андроїд на будь-якій ОС буде досить наступних характеристик:

–оперативна пам’ять: 4 ГБ (мінімум), 8 ГБ (рекомендується); +2 ГБ для Android Emulator;

–вільне місце на диску: 2 ГБ мінімум (500 МБ для IDE + 1.5 ГБ для Android SDK и образа системи емулятора), 4 ГБ SSD рекомендується;

–версія JDK: Java Development Kit 8;

–возширення екрану: 1280 x 800 (мінімум).

2.2 Загальна структура програми

У android знаходяться усі файли і модулі що належать до андроиду, в ios – усі файли і модулі відносяться до ios. У теці src знаходиться увесь javascript код який безпосередньо розбитий на такі розділи: components, assets(json анімації, зображення, шрифти), screens, utils, redux(actions, reducers, sagas), constants. Сам додаток React Native базується на архітектурі мостів(Bridges) між javascript потоком і Native UI потоком [4].

React Native – це приклад рішення, яке можна назвати "не-залежним". В результаті виявляється, що головна мета фреймворка полягає в тому, щоб дозволити розробникам писати JavaScript- код, що використовує механізми React. Завдання React Native полягає в тому, щоб забезпечити перетворення дерева React- компонентів в щось таке, що виявиться працездатним на різних мобільних платформах. Це означає наступне: коректне формування інтерфейсу, забезпечення доступу до нативних можливостей різних платформ.

За спілкування між двома потоками відповідає так званий міст(bridge) – це ядро React Native. Міст дозволяє потокам спілкуватися найкращим, оптимізованим способом. Він служить посередником, який регулює запити і дані, що входять, від двох потоків. Такий підхід дозволяє їм спілкуватися асинхронно, що призводить до стабільної роботи, тому що потоки ніколи не зможуть заблокувати друг-друга. [6]

У кожному React Native-додатку паралельно виконуються три наступних потоку:

– потік JS. Це - потік, в якому здійснюється читання і компіляція JavaScript- коду. Тут же виконується основна частина логіки додатка. Бандлер

Metro комбінує увесь JS- код в єдиний файл, відповідає за обробку JSX - і TS-конструкцій. Потім отриманий код відправляють движку JavaScriptCore, засобами якого цей код може бути запущений;

– потік Native. Тут робиться виконання нативного коду. Цей потік взаємодіє з JavaScript- потоком тоді, коли треба оновити інтерфейс або звернутися до нативних функцій. Цей потік можна розділити на дві частини. Перша, Native UI, відповідає за використання нативних засобів формування інтерфейсу. Друга, Native Modules, надає доступ до особливих можливостей платформи, на якій працює додаток. Вона готова до роботи після запуску додатка. Тобто, наприклад, якщо React Native використовує Bluetooth- модуль, він завжди буде активним, навіть у тому випадку, якщо він, насправді, не використовується;

– потік Shadow. У нім виконується перерахунок макету додатка. Тут використовується движок Yoga, що є власною розробкою Facebook. У цьому потоці виконуються обчислення, пов'язані з формуванням інтерфейсу додатка. Результати цих обчислень відправляють потоку, відповідальному за виведення інтерфейсу.

Для організації взаємодії між потоками JS і Native використовується модуль Bridge(міст). Цей модуль написаний на C++, в його основі лежить асинхронна черга. Коли міст отримує дані від однієї із сторін, він сериализует ці дані, перетворюючи в строковій вид, і передає їх через чергу. Коли дані прибувають в пункт призначення, вони десеріалізуются.

Це означає, що усі потоки покладаються на асинхронні JSON-повідомлення, що передаються через міст. Кожна із сторін відправляє ці повідомлення, чекаючи(але не маючи гарантії) того, що колись в майбутньому на ці повідомлення буде отримана відповідь. При такій схемі обміну даними існує ризик перевантаження каналу зв'язку.

Ось приклад, який зазвичай приводять, щоб проілюструвати те, як подібна схема обміну даними здатна викликати проблеми з продуктивністю додатка. Припустимо, що користувач додатка виконує прокрутку величезного

списку. Коли в нативному середовищі відбувається подія onScroll, інформація асинхронно передається в JavaScript- оточення. Але нативні механізми не чекають до тих пір, поки JavaScript- частина додатка зробить свою справу і їм про це відзвітує. Через це виникає затримка, що виражається в появі в списку, перед виведенням його вмісту, порожнього простору.

Аналогічно, перед тим, як результати перерахунку макету дійдуть до екрану, системі належить виконати декілька сеансів обміну даними. Так, перш ніж дані макету потраплять в нативне середовище, їх треба обробити в Yoga. А це, зрозуміло, означає, що такі дані теж доводиться передавати через міст.

Як видно, відправка JSON-даних з використанням асинхронних механізмів, а також сериализация і десериализация цих даних, викликають проблеми з продуктивністю. Цей механізм недосконалий.

В ході того, що перепроектувало архітектури React Native виконується поступова відмова від функцій моста, на зміну яким приходить новий механізм, званий JavaScript Interface (JSI).

Застосування JSI відкриває можливості для деяких чудових поліпшень.

Перше таке поліпшення полягає в тому, що JS- бандл більше не покладається на JSC. Іншими словами, движок JSC тепер легко можна замінити на щось інше, цілком можливо, що відрізняється більш високою продуктивність. Наприклад – на движок V8.

Друге поліпшення – це те, що лежить в основі нової архітектури React Native. Воно полягає в тому, що, завдяки використанню JSI, в JavaScript можна зберігати посилання на C++-об’єкти (Host Objects) і викликати методи цих об'єктів. В результаті виявляється, що JavaScript – частина додатка і нативні механізми знатимуть один про одного значно більше, чим раніше.

Іншими словами, JSI дозволяє забезпечити повну взаємодію між усіма потоками. Завдяки використанню концепції спільного володіння (shared ownership), JavaScript-код може запускати нативні методи безпосередньо з JS-потоку. При цьому немає необхідності в тому, щоб використати JSON для передачі даних. Це дозволяє позбавитися від проблем, характерних для

використання моста, пов'язаних з можливим переповнюванням черги і з асинхронною передачею даних.

Нова архітектура, дозволяє здійснювати безпосереднє управління нативними модулями. Це означає, що нативні модулі можна використати тоді, тоді вони потрібні, а не ініціалізувати їх все при запуску додатка. Це призводить до серйозного збільшення швидкості запуску додатків.

Цей новий механізм може відкрити дорогу до розробки нових проектів, здатних на те, що було недоступне старим RN- застосуванням. Річ у тому, що тепер в нашому розпорядженні опинилася потужність C++. А це означає, що на базі React Native тепер можна буде створювати значно більше різновидів додатків, чим раніше.

Генератори можуть породжувати (yield) безліч значень одно за іншим, в міру необхідності. Генератори відмінно працюють з перебираними об'єктами і дозволяють легко створювати потоки даних.

Дивлячись з боку кожної окремої платформи ми отримаємо два застосування. Перше на Objective C для ios і друге на Java для андроид.

Для стейт менеджменту в своєму додатку я вибрав Redux-Saga.

Redux-Saga – це бібліотека, яка покликана спростити і поліпшити побічні ефекти(тобто такі дії, як асинхронні операції, наприклад, завантаження даних, і "брудні" дії, такі, як доступ до браузерного кешу), зробити легенями в тестуванні і краще справлятися з помилками.

Можна представити це так, що saga - це як окремий потік у вашому застосуванні, який відповідає за побічні ефекти. redux - saga – це мидлвар redux, що означає, що цей потік може запускатися, зупинятися і відмінятися з основного застосування за допомогою звичайних дій redux, воно має доступ до повного стану redux застосування і також може диспатчить дії redux.

Бібліотека використовує концепцію ES6, під назвою генератори, для того, щоб зробити ці асинхронні потоки легкими для читання, написання і тестування. Тим самим, ці асинхронні потоки виглядають, як ваш стандартний синхронний JavaScript-код. (на кшталт async/await, але генератори мають декілька

прекрасних можливостей, необхідних нам). Звичайні функції повертають тільки одно-єдине значення(чи нічого).

Redux — бібліотека для JavaScript з відкритим початковим кодом, призначена для управління станом додатка. Найчастіше використовується в зв'язці з React або Angular для розробки клієнтської частини. Містить ряд інструментів, що дозволяють значно спростити передачу даних сховища через контекст.

Для зберігання даних і розділення юзерів, і спілкування я використав AsyncStorage.

AsyncStorage – це незашифрована, асинхронна, постійна система зберігання "ключ-значення", глобальна для додатка. Його слід використати замість LocalStorage.

Рекомендується використати абстракцію поверх AsyncStorage замість безпосередньо AsyncStorage для чого-небудь, окрім легкого використання, оскільки вона працює глобально.

У iOS AsyncStorage підтримується власним кодом, який зберігає невеликі значення в серіалізованному словнику, а великі значення - в окремих файлах. У Android AsyncStorage використовуватиме або RocksDB, або SQLite залежно від того, що доступно.

Код JavaScript AsyncStorage – це фасад, який надає зрозумілий API JavaScript, реальні об'єкти помилок і не-мульти-функції. Кожен метод в API повертає об'єкт Promise.

Усі компоненти і екрани додатка побудовані за принципом MVC.

Для двосторонньої взаємодії потрібні динамічні сторінки. MVC-ключ до розуміння розробки динамічних застосувань, тому розробникові треба знати цю модель.

MVC розшифровується як модель-представлення-контролер(від англ. model - view - controller). Це спосіб організації коду, який припускає виділення блоків, що відповідають за рішення різних завдань. Один блок відповідає за ці

застосування, інший відповідає за зовнішній вигляд, а третій контролює роботу додатка. Компоненти MVC:

– модель - цей компонент відповідає за дані, а також визначає структуру додатка. Наприклад, якщо ви створюєте To-Do додаток, код компонента model визначатиме список завдань і окремі завдання;

– модання – цей компонент відповідає за взаємодію з користувачем. Тобто код компонента view визначає зовнішній вигляд додатка і способи його використання;

– контролер – цей компонент відповідає за зв'язок між model і view. Код компонента controller визначає, як сайт реагує на дії користувача. По суті, це мозок MVC-додатка.

2.3 Модуль управління користувачами

Це найважливіший і основний модуль, в якому прописано ядро логіки управління усім потоком інформації усередині додатка.

База даних в цьому застосуванні не була потрібна, був написаний власний модуль аналогічний кістяку сучасної mongoDb.

NoSQL став дуже популярною темою у великомасштабному розгортанні, масштабованість і напівструктуровані дані призводять до вимог для баз даних, які якраз задовольняє NoSQL. MongoDb - це один з продуктів NoSQL, який вражає своєю зручністю використання і легкістю розуміння її базової архітектури.

MongoDb відрізняється від СУБД таким чином:

В відмінність від запису СУБД, яка являється "flat"(постійне число простого типу даних), основна одиниця MongoDb – "документ", який є "вкладеним" і може містити поля-мультизначення(масиви, хеш).

В відмінність від СУБД, де усі записи, що зберігаються в таблиці, мають бути обмежені схемою таблиці, документи будь-якої структури можуть зберігатися в одному сховищі.

В запитах немає операції "join". В цілому, дані можуть бути організованими більше денормалізованим способом, і на розробників додатків лягає більше навантаження для забезпечення несуперечності даних.

В MongoDb немає ніякого поняття "транзакції". "Атомарність" гарантується тільки на рівні документу(ніякого часткового оновлення документу статися не може).

Нема поняття "ізоляції", будь-які дані, які прочитуються одним клієнтом, можуть бути модифіковані іншим паралельним клієнтом.

Шляхом видалення деяких функцій, які надає класична СУБД, MongoDb став легшим і більше масштабованим в обробці великих даних.

MongoDb належить типу документо-орієнтованої бази даних. У цій моделі дані організовані як документ JSON і зберігаються в колекції. Колекцію можна сприймати як таблицю, а Документ еквівалентний записам у світі СУБД.

Так само було впроваджено індексування.Індекс може також бути створений на багатозначному атрибуті, такому як масив. В цьому випадку у кожного елементу в масиві буде окремий вузол в BTree.

Створення індексу може бути здійснене як в оффлайновом, так і онлайновому пріоритетному режимі. Пріоритетний режим обробки завдань пройде набагато швидше, але база даних не може бути доступна впродовж періоду створення індексу. Якщо система працює в Replica Set, рекомендується повернути кожну задіяну базу даних оффлайн і створити індекс в пріоритетному режимі.

Якщо в запиті декілька критеріїв відбору, MongoDb намагається використати один кращий індекс, щоб вибрати передбачуваний набір і потім послідовно виконує ітерації по них, щоб оцінити інші критерії.

Якщо є декілька індексів доступних для колекції. При обробці запиту вперше, MongoDb створить плани виконання (по одному для кожного

доступного індексу) і дозволить їм мінятися(впродовж певного числа тиків) для виконання, поки найшвидший план не закінчиться. Буде повернений результат найшвидшого виконавця, а система запам'ятає відповідний індекс, який він використав. Подальший запит використовуватиме індекс, що запам'ятав, поки в колекції не станеться певне число оновлень, тоді система повторює процес, щоб з'ясувати, який індекс є кращим на той момент.

Оскільки тільки один індекс використовуватиметься, важливо шукати критерії пошуку або сортування і створювати додатковий складений індекс, який краще буде відповідати запиту. Підтримка індексу не беззатратно, оскільки індекс повинен оновлюватися, коли документи створюються, віддаляються і оновлюються, що перевантажує операції оновлення. Щоб підтримати оптимальний баланс, ми повинні періодично вимірювати ефективність наявності індексу(наприклад, співвідношення читання-запису) і видаляти менш ефективні індекси.

Відповідно модуль управління користувачами працює на аналогічній основі. Існує розділ якої контролює зберігання і перевірку вже створених або нових токенов для підтримки користувачам в додатку і повторного логіна. Розділ який контролює реєстрацію і логін користувачів і контролює унікальність даних кожного користувача і контролює індексування для швидкого доступу до кожного їх них про зверненні до сховища. Відповідно це прискорює пошук, сортування і фільтрацію за нашими даними.

2.3.1 Модуль JWT

Веб-токен JSON, або JWT(вимовляється "jot"), є стандартизований, в деяких випадках підписаний і/або зашифрований формат упаковки даних, який використовується для безпечної передачі інформації між двома сторонами.

Проаналізуємо це формулювання.

Формат упаковки даних.

JWT визначає особливу структуру інформації, яка вирушає по мережі. Вона представлена в двох формах - серіалізовані і десеріализовані. Перша використовується безпосередньо для передачі даних із запитами і відповідями. З іншого боку, щоб читати і записувати інформацію в токен, потрібна його десеріализация.

Десеріализована форма.

У несеріализованой виді JWT складається із заголовка і корисного навантаження, які є звичайними JSON- об'єктами.

Заголовок(заголовок JOSE) в основному використовується для опису криптографічних функцій, які застосовуються для підпису і/або шифрування токена. Тут також можна вказати додаткові властивості, наприклад, тип вмісту, хоча це рідко потрібно.

Якщо JWT підписаний і/або зашифрований, в заголовку вказується ім'я алгоритму шифрування. Для цього призначена заявка alg.

Слово "заявка" в специфікації означає просто частину інформації і аналогічна ключу JSON- об'єкту. Вона представлена у вигляді пари ім'я: значення, де ім'я завжди є рядком. Значенням заявки може бути будь-який серіализуємий тип даних. Наприклад наступний об’єкт JSON складається з трьох заявок: iss, exp и http://example.com/is_root.

Заявки бувають службовими і призначеними для користувача. Перші зазвичай є частиною якого-небудь стандарту, наприклад, реєстру JSON Web Token Claims, і мають певні значення. Найбільш поширені службові заявки:

–iss видавець токена;

–sub описуваний об’єкт;

–aud одержувачі;

–exp дата закінчення терміну дії;

–iat час створення.

Тема опису криптографічні операції, які застосовуються до веб-токену. Але в деяких випадках підпис і шифрування можуть бути відсутніми. Зазвичай це відбувається, коли JWT є частиною деякої вже зашифрованої структури даних. У заголовку такого непідписаного токена заявка alg має бути рівна none.

Корисне навантаження.

Корисні дані – частина токена, в якій розміщується уся необхідна призначена для користувача інформація. Як і заголовок, корисне навантаження є звичайним об'єктом JSON. Тут жодне поле не є обов'язковим. Зазвичай використовуються вже розглянуті службові заявки iss, sub і aud, а також специфічні для додатки дані.

Процес серіалізації JWT складається з кодування заголовка, корисного навантаження і підпису, якщо вона є, за допомогою алгоритму base64url. Це проста варіація base64, яка використовує URL- безпечний символ _ замість небезпечних + і /. Річ у тому, що деякі інструменти обробки даних розпізнають символи, що управляють, в рядку, тому їх бути не повинно.

JSON токени може використати будь-який процес, пов'язаний з обміном даних через мережу. Наприклад, просте клієнт-серверне застосування або група з декількох пов'язаних серверів і клієнтів. Відмінним прикладом складних процесів з множиною споживачів даних служать фреймворки авторизації, такі як AuthO і OpenID.

Найчастіше використовуються клієнтські сеанси без збереження стану. При цьому уся інформація розміщується на стороні клієнта і передається на сервер з кожним запитом. Саме тут використовується JSON web token, який забезпечує компактний і захищений контейнер для даних.

На відміну від серверних сесій, дані клієнтських сеансів зберігаються на машині користувача і вирушають з кожним запитом. Один із способів реалізації

цього механізму - використання файлів cookie. У них можна зберігати не лише ідентифікатор сеансу, але і самі дані. Куки-файли дуже зручно використати, оскільки вони автоматично обробляються веб-браузерами. Проте це не єдиний спосіб передавати інформацію сеансу. Це також можна робити через HTTP-заголовки, щоб уникнути проблем CORS, або за допомогою URL-параметрів.

З сеансами на стороні клієнта знімається проблема з масштабованістю. Проте з'являються уразливості безпеки. Не можна гарантувати, що клієнт не змінює ці сесії. Наприклад, якщо ідентифікатор користувача зберігається у файлах cookie, його легко змінити. Це дозволить отримати доступ до чужого облікового запису. Щоб запобігти подібним діям, треба обернути дані в захищений від несанкціонованої дії пакет.

Саме для цього і потрібний JWT. Завдяки тому, що JSON токени мають підпис, сервер може перевіряти достовірність даних сеансу і довіряти ім. При необхідності їх можна навіть зашифрувати, щоб захиститися від читання і зміни.

Втім, у сеансів на стороні клієнта також є свої мінуси. Наприклад, додаток може вимагати великого об'єму призначених для користувача даних, і їх доведеться відправляти туди-назад для кожного запиту. Це може навіть перекрити усі переваги простоти і масштабованості. Таким чином, у реальному світі додатка частенько поєднують клієнтські і серверні сеанси.

Мета підпису полягає в тому, щоб дати можливість одній або декільком сторонам встановити достовірність токена. Пам'ятайте приклад підробки ідентифікатора користувача з cookie для діставання доступу до чужого облікового запису? Токен можна підписати, щоб перевірити, чи не були змінені дані, що містяться в нім. Проте підпис не заважає іншим сторонам читати вміст JWT, цю вже справу шифрування. Підписаний веб-токен відомий як JWS(JSON Web Signature). У компактній серіалізованій формі у нього з'являється третій сегмент - підпис.

Найпоширеніший алгоритм підпис – HMAC. Він об'єднує корисне навантаження з секретом, використовуючи криптографічну хеш-функції(частіше усього SHA - 256). За допомогою отриманого унікального підпису можна

веріфіцировать дані. Це схема називається розділенням секрету, оскільки він відомий обом сторонам: творцеві і одержувачеві. Таким чином, і той, і інший можуть генерувати нове підписане повідомлення.

Інший алгоритм підпису - RSASSA. На відміну від HMAC від дозволяє приймаючій стороні тільки перевіряти достовірність повідомлення, але не генерувати його. Алгоритм заснований на схемі відкритого і закритого ключів. Закритий ключ може використовуватися як для створення підписаного повідомлення, так і для перевірки. Відкритий ключ, навпаки, дозволяє лише перевірити достовірність. Це важливо у багатьох сценаріях підписки, таких як Single - Sign On, де є тільки один творець повідомлення і багато одержувачів. Таким чином, ніякий зловмисний споживач даних не зможе їх змінити [8].

2.3.2 Модулі HOC

Я написав ряд таких модулів для спрощення розробки, що відповідно породило окремий модуль по управлінню конструкцією додатка.

Компонент вищого порядку(Higher - Order Component, HOC) – це один з просунутих способів для повторного використання логіки. HOC не є частиною API React, але часто застосовуються із-за композиційної природи компонентів.

Говорячи просто, компонент вищого порядку – це функція, яка приймає компонент і повертає новий компонент.

Якщо звичайний компонент перетворить пропси в UI, то компонент вищого порядку перетворить компонент в інший компонент.

HOC часто зустрічаються в сторонніх бібліотеках, наприклад connect в Redux і createFragmentContainer в Relay.

Традиційні компоненти мають на увазі багатократне використання, але не дозволяють з легкістю вирішити деякі проблеми.

HOC нічого не міняє і не наслідує поведінку компонента, що обертається, замість цього HOC обертає оригінальний компонент в контейнер за допомогою композиції. HOC є чистою функцією без побічних ефектів.

Компонент, що обертається, отримує усі пропси, передані контейнеру, а також проп data. Для HOC не важливо, як використовуватимуться дані, а компоненту, що обертається, не важливо, звідки вони беруться.

Оскільки withSubscription - це звичайна функція, то ми можемо прибрати або додати будь-яку кількість аргументів. Наприклад, ми могли б зробити таким, що конфігурується назву пропа data і ще більше ізолювати HOC від компонента, що обертається. Також ми можемо додати аргумент для конфігурації shouldComponentUpdate або джерела даних. Усе це можливо, тому що HOC повністю контролює процес створення компонента.

Взаємодія між withSubscription і компонентом, що обертається, здійснюється за допомогою пропсів, так само як і між звичайними компонентами. Завдяки цьому ми можемо з легкістю замінити один HOC на інший, за умови, що вони передають одні і ті ж пропси в компонент, що обертається. Це може згодитися якщо, наприклад, ми вирішимо поміняти бібліотеку отримання даних.

Не мутуйте компонент, що обертається. Використайте композицію. Якщо ми захочемо обернути EnhancedComponent в іншій HOC, який теж міняє componentDidUpdate, то ми зітремо функціональність задану першим HOC! Більше того, EnhancedComponent не працює з функціональними компонентами, тому що у них відсутні методи життєвого циклу.

Замість мутації, компоненти вищого порядку повинні застосовувати композицію, обертаючи компонент в контейнер. Цей HOC має таку ж функціональність, як і попередній, але не створює конфліктів з іншими HOC і працює як з функціональними, так і з класовими компонентами. Більше того, HOC реалізований за допомогою чистої функції, тому його можна поєднувати з іншими HOC, або навіть самого з собою.

За допомогою контейнерів ми зазвичай розділяємо загальну функціональність від приватної. Наприклад, в контейнері ми управлятимемо внутрішнім станом або підпискою на зовнішні ресурси, і через пропси передавати дані в компоненти, відповідальні за рендер UI. При реалізації HOC

ми теж використовуємо контейнери. Можна сказати що HOC - це інструмент для параметризованого створення контейнерів.

HOC додають компонентам функціональність, але вони не повинні міняти їх оригінальне призначення. Очікується, що інтерфейс компонента, який ви повертаєте з HOC, буде схожий на інтерфейс компонента, що обертається.

Пропси, які безпосередньо не пов'язані з функціональністю HOC, повинні передаватися без змін компоненту, що обертається.

connect — це функція вищого порядку, яка повертає компонент вищого порядку!

Така форма може здатися заплутаною і непотрібною, але є і переваги. Виклик connect повертає HOC з підписом Component => Component. Функції з однаковим типом результату і єдиного аргументу легко поєднуються в композиції. Тому ми можемо використати connect і що інші, що розширюють функціональність HOC в якості експериментальних JavaScript декораторів.)

Алгоритм порівняння React(відомий як узгодження або reconciliation) використовує тотожність компонентів щоб визначити чи треба оновити існуюче поддерево, або прибрати і монтувати замість нього нове. Якщо компонент, отриманий з render, ідентичний (===) компоненту з попереднього рендера, то React рекурсивно продовжить порівнювати поддерево. Якщо компоненти не рівні, React повністю видалить і замінить старе поддерево [10].

Проблема не лише в продуктивності. Повторне монтування компонента обнуляє його стан, а також стан його дочірніх компонентів.

При необхідності(у окремих випадках) можна динамічно застосовувати HOC в методах життєвого циклу або конструкторі компонента.

За угодою компоненти вищого порядку передають компоненту, що обертається, усі пропси, окрім рефов. ref насправді не проп, як, наприклад, key, і тому інакше обробляється React. Реф елементу, створеного компонентом з HOC, вказуватиме на екземпляр найближчого в ієрархії контейнера, а не на компонент, що обертається.

2.3.3 Загальна структура додатку

На основі проведеного аналізу та з урахуванням поставленої задачі розроблено загальну структуру додатку (рис. 2.1).

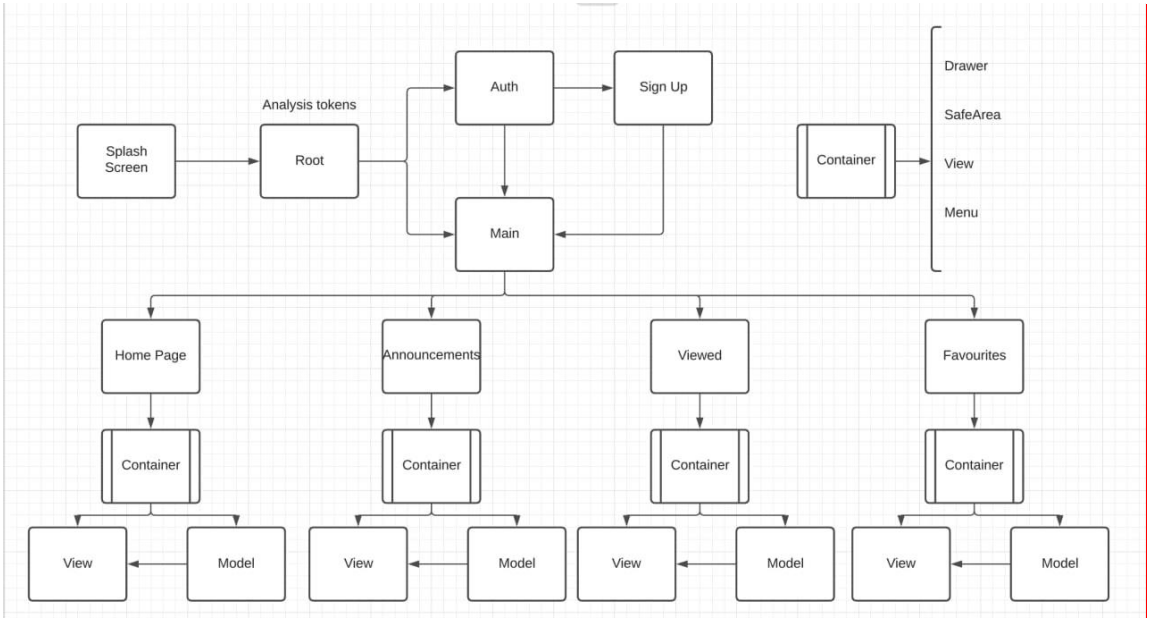

Рисунок 2.1 – Загальна структура додатку

Додаток включає в себе модулі.

Homepage (домашня сторінка) – це одна з назв головної сторінки додатку англійською мовою.

Announcement (оголошення) – це модуль, в якому подається потрібна інформація, адресована певному колу зацікавлених осіб.

Viewed (перегляд) – це модуль, який відповідає за функцію перегляду контенту, або відображення вже переглянутого відео.

Favourites (обране) – це модуль, який відповідає за вибраний користувачем контент, який найбільше сподобався.

2.4 Висновки за розділом 2

В даному розділі були розглянуті різні мови програмування та середовища розробки, які дозволяють розробляти програмні продукти. Було

проведено їх аналіз, виділено основні переваги і недоліки та складено порівняльну характеристику. В ході цієї роботи було встановлено, що для програмного продукту найбільш ефективною мовою програмування є Java Script.

Також було розроблено структурну схему програмного продукту і

Виконано аналіз фреймворків React Native і Flutter, який показав, що для вирішення поставленої задачі доцільно використовувари React Native.

Було описано модулі управління користувачами, розглянуто модуль JWT і модуль HOC.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Опис функціонування програми

База даних розташовується у файлі userController.js.

Після запуску додатку спочатку запускається файл Root.js у якому береться токен та ведуться розрахунки та отримується інформація про користувача та потім по його _id ми отримуємо його бібліотеку фільмів та доступ до функціоналу. Якщо ж немає токену або не вдалося отримати користувача відбудеться редірект на екран логіну.

3.2 Використані моделі даних

Моделювання даних – це один із найважливіших етапів побудови баз даних та додатку користувача.

Моделювання (проектування) баз даних складається з трьох етапів:

– концептуального;

– логічного;

– фізичного.

Найпершим і найважливішим етапом проектування баз даних являється етап концептуального моделювання предметної області. Завданням цього етапу є збір, аналіз і редагування вимог до даних. Також цей етап виконує обстеження предметної області, вивчення її інформаційної структури, виявлення всіх фрагментів предметної області. Після закінчення даного етапу отримуємо концептуальну модель, інваріантну до структури баз даних. Цей етап виконується аналітиками.

Наступним етапом є логічне проектування баз даних. На даному етапі діаграми перетворюються в набір таблиць під час чого відбувається їх нормалізація. На виході отримуємо структуру бази даних. Даний етап виконується програмистами. Спроектована діаграма зображена на рисунку 3.1.

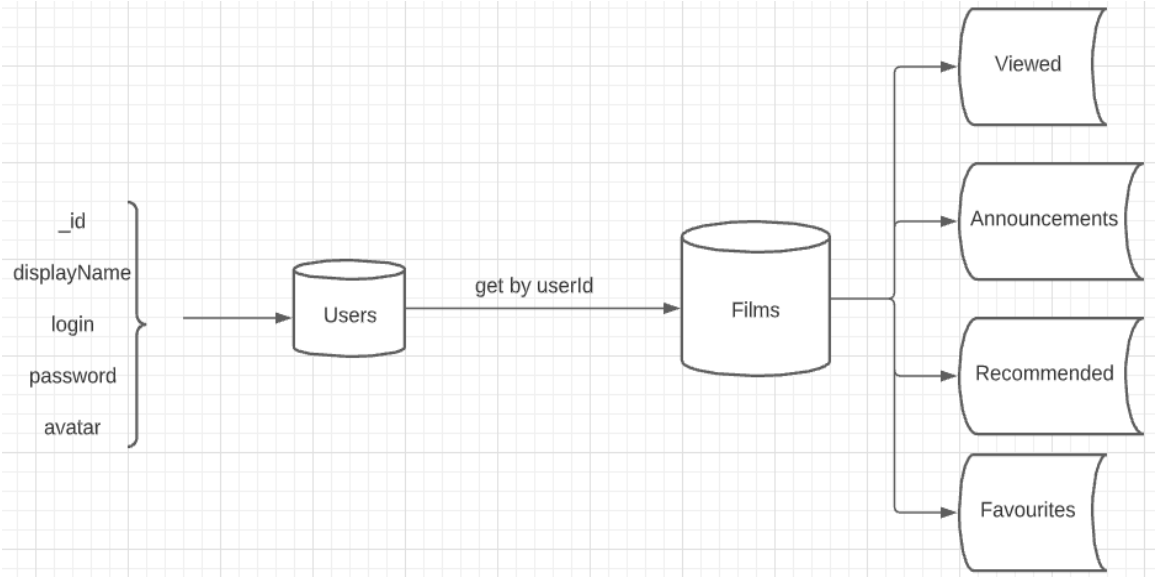

Рисунок 3.1 – Діаграма системи, що проєктується

Завершальним етапом є фізичне проєктування баз даних, на якому відбувається адаптація логічної моделі під обрану програмну платформу, вибір та побудова індексів, організація протоколів. Завершує проєктування адміністратор.

У mongodb та mongo-подібних система немає зв’язків, їх можна створити але вони будуть не справжніми а створені за допомогою індексації.

Сутність «Users» (_id, displayName, login, password, avatar). Сутність призначена для зберігання інформації про користувачів. В реалізації сутність представлено таблицею «Users» (табл. 3.1).

Таблиця 3.1 – Структура таблиці «Users»

Назва поля	Тип поля	Розмір поля	Ключ	Опис
_id	uniqueMongoId	< 32	Так	Призначене для унікальності записів у полях
displayName	Текстовий	15	—	Призначене для збереження Нікнейму користувача
login	Текстовий	12	—	Призначене для зберігання логіну користувача
password	Текстовий (md5)	18	—	Призначене для зберігання шифрованого паролю користувача
avatar	Тексовий	256	—	Призначене для зберігання посилання на аватар користувача

Сутність «Film» (_id, title, description, announceDate, rate, image). Сутність призначена для зберігання інформації про фільми. В реалізації сутність представлено таблицею «Film» (табл. 3.2).

Таблиця 3.2 – Структура таблиці «Film»

Назва поля	Тип поля	Розмір поля	Ключ	Опис
_id	uniqueMongoId	< 32	Так	Призначене для унікальності записів у полях
title	Текстовий	24	—	Призначене для збереження Назви фільму
announceDate	Date	—	—	Призначене для зберігання дати виходу фільму та таймеру
description	Текстовий	128	—	Призначене для зберігання опису фільму
rate	Чисельний	1	—	Призначене для зберігання рейтингу фільму
image	Текстовий	256	—	Призначений для зберігання посилання на зображення фільму

3.3 Опис логічної структури програми

При запуску програми завантажується головний модуль програми App.js + middleware. Тут перевіряється наявність записів а базі, перевіряється чи всі модулі працездатні та встановлюється базова архітектура та токени користувача.

Головний модуль розташований у файлі userController.js. Цей модуль контролює всі таблиці, запис, читання та валідацію всіх даних. Він керує користувачами, створенням нових та перевіркою.

Модуль API має в собі вже прописані параметри для спілкування та обміном даними зі стороннім сервісом для отримання та зміни нових фільмів, жанрів та іншого.

Модуль Config контролює в якому середовищі це все буде працювати, керую ключами та конфігураціями для всіх інших модулей.

Модуль Films дозволяє працювати з таблицею фільмів, переміщувати їх в інші категорії, створювати інші та інше.

Також у додатку існує окремий сервіс який працює у бєкграунді та перевіряє всі ваші таймери та слідкує за повідомленнями щоб своєчасно нагадати

вам про те що скоро вийде новий фільм який ви додали до анонсів чи до улюблених.

В програмі використані наступні бібліотеки:

–@react-native-community/async-storage – дозволяє працювати зі сховищем;

–Lodash – набір готових методів для JavaScript;

–Lottie-ios + lottie-react-native – бібліотека для анімованих json та svg файлів;

–React-native-date-picker – біблотека для нативного вибору часу та дати;

–React-native-drawer – бібліотека для бокового меню;

–React-native-image-picker – бібліотека для вибору зображення з пам’яті пристрою;

–React-native-keyboard-aware-scrollview – бібліотека для перевірки та зміщення контенту у разі перекриття клавіатурою контенту додатку;

–React-antive-size-matters – бібліотека для використання відносних величин для одного дизайну та розміру на всіх діагналях;

–React-native-swipeout – бібліотека для контролювання і додавання нових функцій при пролистуванні об’єктів;

–React-native-vector-icons – бібліотека для підключення усіх існуючих іконок створених для дизайну (шрифти).

3.4 Висновки за розділом 3

В даному розділі розроблено діаграму системи, що проєктується, а також сутність логічних структур, що входять в систему.

В розділі також виконано опис функціонування програми і типи використаних моделей даних, представлено опис процесу моделювання бази даних.

4 КЕРІВНИЦТВО КОРИСТУВАЧА

4.1 Призначення програми

Застосування – середні по розмірах, за нинішніми мірками, додаток-щоденник для планування, ведення обліку о проглянутих фільму, серіалах, мультфільмах і інших відеоматеріалів з розділенням користувачів, додавання новинок які ще не вийдуть з відповідним звітом часу і даті виходу і своєчасним нагадуванням. Ці аналоги можна зустріти на звичайних сайтах з фільмами, додатках ніби Netflix і так далі. Я спростив можливість перегляду фільму з додатка, залишив то що було мені зручно і додав у своє застосування той функціонал якого мені не хватає у аналог і максимально спростив UX до інтуїтивного рівня.

4.2 Умови виконання програми

Для роботи мого додатку потрібен iPhone з версією iOS 9.0 та вище, тобто iPhone починаючи с 5s підтримує мій додаток. Або смартфон на Android з версією вище ніж 9 та з мінімальною версією SDK 18.

Також для того що б додаток працював повністю у вашого смартфона повинна бути підтримка Google Services, тому смартфони Хуавей мають не повний функціонал мого додатку.

4.3 Робота з програмою

При запуску додатка ми бачимо Splash Screen. Це екран під час зображення якого робляться фонові завантаження в додатку необхідні спершу роботи. Далі ми бачимо екран логіна (рис. 4.1).

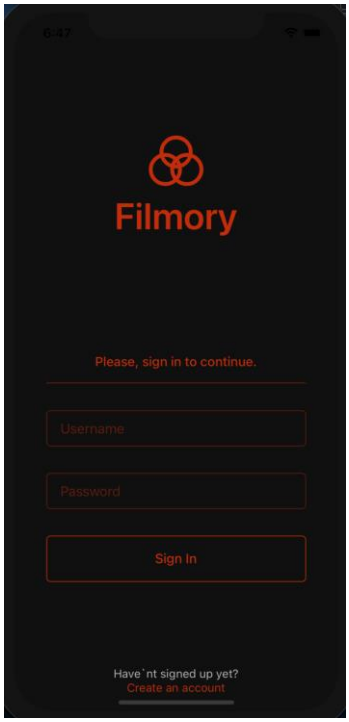

Рисунок 4.1 – Екран логіну

На цьому екрані ми можемо авторизуватися у свій аккаунт, є присутніми валідатори і повідомлення про помилки. Звідси також можна перейти на аналогічний екран реєстрації нового користувача (рис. 4.2). На нім можна заповнити ім'я користувача, логін і пароль. Валідатори налагоджені так що ім'я користувача і логін ніколи не повторюватимуться.

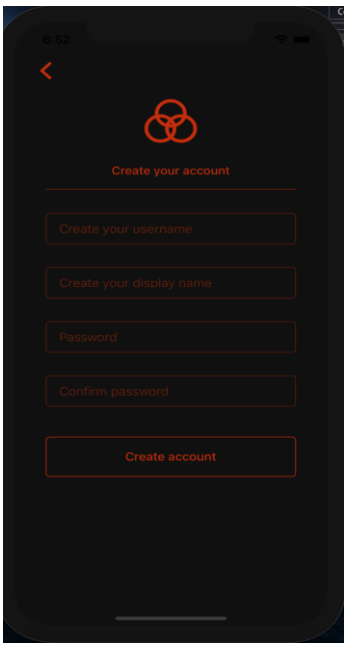

Рисунок 4.2 – Екран авторизації

Далі по умовчанню ми побачимо екран з рекомендованими фільмами і мультфільмами до перегляду з кращим рейтингом і новинки, які в додаток приходять з окремої API і завжди оновлюються залежно від оновлення (рис. 4.3).

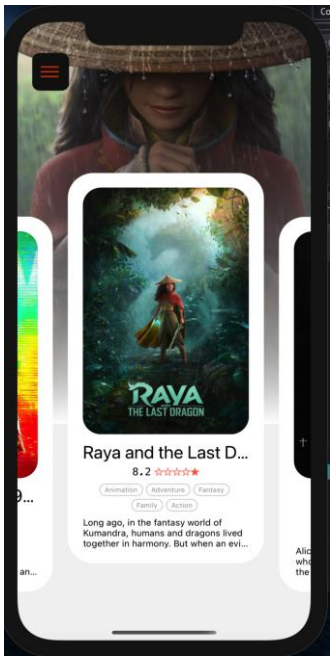

Рисунок 4.3 – Екран списку відео

Також в додатку бічне бургер меню з вкладками з додатковими екранами (рис. 4.4).

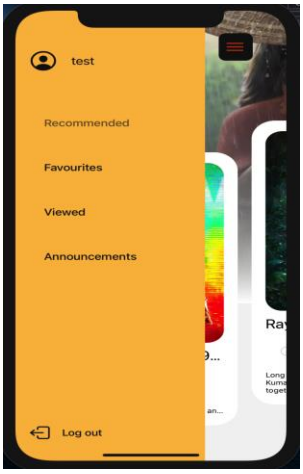

Рисунок 4.4 – Екран бокового меню

Один з головних екранів це анонси. Там можна створювати будь-які замітки з приводу фільмів, переглядати вже готові з таймером і рейтингом і так

само є можливість переміщення до проглянутих або улюблених фільмів (рис. 4.5).

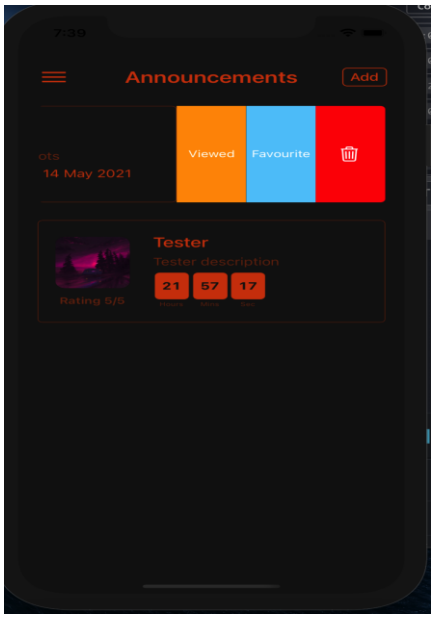

Рисунок 4.5 – Екран анонсів

Є кнопка "Add" яка перенаправить користувача на екран де можна створити новий анонс і далі перемістити його у будь-який інший розділ (рис. 4.6).

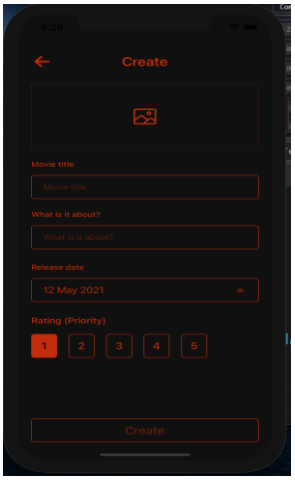

Рисунок 4.6 – Екран створення нового анонса

Екран з проглянутими має той же функціонал і є бібліотекою для зберігання історії проглянутих відеоматеріалів.

4.4 Висновки за розділом 4

В даному розділі розкрито особливості функціонування програми, умови її використиня, наведений інтерфейс додатку та опис принципів його роботи.

ВИСНОВКИ

Під час виконання роботи проведено дослідження предметної області, розглянуто та визначено основні поняття, які пов’язані з обраною предметною областю. Також були проаналізовані та досліджені існуючі методи, кожен з яких можливо застосовувати для вирішення поставленої задачі, було визначено їх переваги та недоліки в рамках вирішення задачі автоматизації ведення обліку відеоконтенту.

Було сформульовано і визначено завдання до роботи.

Метою роботи є створення програмної системи для оптимізації та ведення обліку відеоконтенту с розмеженням прав користувачів.

Для виконання поставленної мети було виконано наступні завдання:

– створено метод для забезпечення авторизації користувачів;

– створено метод для додавання інформації про нові ролики;

– створено метод для перегляду анонсів;

– створено метод для перегляду вже переглянутих роликів;

– створено метод для перегляду улюбленних роликів;

– створено метод для перегляду популярних та щойно випущених новинок;

– створено метод для підключення до стороннього API для отримання нових роликів.

Також в роботі були розглянуті різні мови програмування та середовища розробки, які дозволяють розробляти програмні продукти. Було проведено їх аналіз, виділено основні переваги і недоліки та складено порівняльну характеристику. В ході цієї роботи було встановлено, що для програмного продукту найбільш ефективною мовою програмування є JavaScript, тому що вона більш проста і працює з достатньою швидкістю. Також найбільш ефективним середовищем розробки було обрано середовище React Native, тому що воно є офіційним для мови програмування JavaScript, не потребує великих обсягів

апаратних ресурсів, а також дає можливість без перешкод розробити додаток відразу для всіх платформ.

В рамках роботи було розроблено структурну схему програмного продукту. Встановлено, що для зберігання вхідних даних найкращий тип сховища є аналог mongodb розробленний в проекті на основі базових методів та технологій React Native.

Таким чином, в рамках дипломної роботи було розроблено андроїд додаток, який дозволяє користувачу переглядати відео контент, ознайомлюватись і додавати коментарі з його приводу, отримувати додаткову інформацію про відео, а також планувати його перегляд в електроному форматі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Адітья Бхаргава Грокаем Алгоритми / Б. Адітья. – СПб. : «Питер», 1996. – 305 с.

2. David Flanaghan JavaScript / D. Flanaghan. – O'REILLY, 1995. – 981 с. [Electronic resource] – Access mode: http://kharchuk.ru/JavaScript.pdf

3. Джон Резіг JavaScript ніндзя / Д. Резіг. – YAKABOO.UA, 2013. – 416 с.

4. Мартін Фаулер Рефакторінг кода на JavaScript / М. Фаулер. – YAKABOO.UA, 2016. – 464 с. [Electronic resource] – Access mode: https://www.yakaboo.ua/refaktoring-koda-na-javascript-uluchshenie-proekta-suschestvujuschego-koda-2104307.html

5. Марк Тіленс Томас React в дії / М. Т. Томас – Manning, 2019. – 368 с.

6. Nader Dabit React Native in Acti / N. Dabit. – Manning, 2019. – 336 с. [Electronic resource] Access mode : https://revall.info/react-native-in-action.html

7. Charles Petzold Code: The Hidden Language of Computer Hardware and Software / C. Petzold. – Microsoft Press, 2000. – 400 с.

8. Addy Osmani Javascript Design Patterns / A. Osmani. – O'REILLY, 2012. – 574 с. [Electronic resource] – Access mode: https://addyosmani.com/resources/essentialjsdesignpatterns/book/

9. Cory Althoff The Self-Taught Programmer: The Definitive Guide to Programming Professionally / C. Althoff. – Self-Taught Media, 2016. – 301с.

10. Баас Л. Архитектура программного обеспечения на практике». / Л. Баас – СПб. : «Питер», 2006. – 574с. [Electronic resource] – Access mode: https://cdnpdf.com/pdf-19733-arhitektura-programmnogo-obespecheniya-na-praktike-2-e

11. Freeman E. Head First Design Patterns. / E. Freeman – O.REILLY, 2004. – 694с.

ДОДАТОК А

Технічне завдання

Вступ

Метою проекту є розробка додатку для перегляду відео контенту, спрямованого на реалізацію можливості зберігання історії та планування подальшого перегляду відео, що підвищує користувацький комфорт.

А.1 Підстава для розробки

Підставою для розробки слугує завдання на випускну роботу на тему «Розробка програмного забезпечення для планування та обліку перегляду відео контенту», затверджене наказом Національного університету «Запорізька політехніка» № 103 від 30 березня 2021 р.

А.2 Призначення розробки

Система спрямована на підвищення комфорту планування дозвілля. Система призначена для ведення обліку переглянутих відео, а також для планування перегляду того контенту, реліз яких на даний момент планується, з можливістю нагадування користувачеві про наближення відповідної дати.

Програма має виконувати наступні функції:

– прийом та обробка запитів користувачів;

– можливість перегляду каталогу фільмів або мультфілмів для користувача (з можливістю фільтрації та редагуванням);

– можливість повідомляти користувача, коли буде вихід даного відеоконтенту, реалізація функції нагадування про реліз;

– можливість змінювати, додавати та видаляти інформацію про відеоконтент;

– можливість запису свох коментарів після перегляду відео контенту.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Функціональні вимоги:

– система повинна приймати та обробляти запити користувачів;

– система повинна виводити каталог фільмів користувачу;

– система повинна надавати змогу змінювати, додавати та видаляти інформацію про відеоконтент;

– система повинна підтримувати можливість надання інформації про відео контент.

А.3.2 Вимоги до надійності

Програма повинна коректно реагувати на наступні виняткові ситуації: невірність введення логіну або паролю.

База даних з даними додатку повинна буди захищеною паролем, який відомий лише користувачу. Повинні бути збережені наступні властивості інформації: конфіденційність, цілісність, доступність. Пароль повинен шифруватися за допомогою технології S-DES.

А.3.3 Умови експлуатації

Програма може зберігатися та переноситися на будь-яких існуючих носіях інформації.

Експлуатація програми здійснюється згідно експлуатаційної документації на розроблену програму, яка відповідає стандартам і містить інформацію, необхідну для її освоєння та експлуатації.

А.3.4 Вимоги до складу та параметрів технічний засобів

Мінімальні вимоги до клієнтської частини:

– пристрій, що підтримує операційну систему Андроїд не ниже 6 версії, або ios не ниже версії 9.0;

– також щоб на Андроїду був Google Service;

– 64-bit CPU з як мінімум 4x Arm cores;

– 1.5 Гб або більше оперативної пам’яті;

– розмір вільної пам’яті пристрою не менше 8Мб;

– підключення до мережі Internet.

А.3.5 Вимоги до маркування і пакування

Програма може бути записана на переносний носій або розповсюджуватися через хмарні сховища. На впакуванні повинна бути назва програми – «Filmory».

А.3.6 Вимоги до транспортування й зберігання

Вимоги до транспортування й зберігання аналогічні вимогам, пропонованим до носія, на якому записана програма.

А.3.7 Вимоги до програмної документації

Програмний продукт повинен поставлятися у наступному складі:

– технічне завдання;

– опис програми;

– текст програми.

А.4 Стадії та етапи розробки

Виділяють такі етапи в розробці програмного забезпечення:

а) збирання та аналіз вимог. Проводиться збір вимог до програмного забезпечення, їх систематизація, документування, аналіз, виявлення протиріч, неповноти, рішення конфліктів в процесі розробки програмного забезпечення;

б) розробка ТЗ. На основі раніше зібраних вимог формується вихідний документ (ТЗ) на проектування програмного забезпечення. Описується всі вимоги до створюваного продукту;

в) створення логотипу та дизайн основних сторінок;

г) налаштування БД;

д) розробка структури. Дозволяє оцінити інформативність та компоновку елементів системи на початковому етапі розробки;

е) підключення адміністративного інтерфейсу. Має містити повне управління настройками системи, окремих модулів та контентом сайту;

ж) робота над програмним кодом. Розробка основного функціоналу системи;

з) наповнення та тестування. Професійні тестувальники знаходять «баги» у системі та фіксують їх. Заповнюється базовій «контент», щоб сайт мав привабливіший вигляд для перших користувачів;

и) розміщення та відкритий доступ, технічна підтримка. Соціальна мережа розміщується на сайті або інтернет-магазину та стає доступною для всіх користувачів;

к) оформлення документації.

А.5 Порядок контролю та приймання системи

Приймання системи здійснюється відповідно до складу та змісту робіт, наведених у Технічному завданні, і відповідно до умов контракту.

Звіт про проведену роботу направляється Замовнику на паперовому носії в 2-х екземплярах, підписаних належним чином, а також на електронному носії.

Вимоги до оформлення проміжних звітів і підсумкового звіту узгоджуються з Виконавцем при підписанні контракту.

Одночасно з підсумковим звітом про виконання робіт Виконавець надає Замовнику підписаний зі сторони Виконавця Акт здачі-приймання виконаної роботи.

ДОДАТОК Б

Текст програми

userController.js

```
//We have in AsyncStorage [users, ids, userIds, userLogins, usernames, usersIds, films]
import AsyncStorage from '@react-native-community/async-storage';
import _ from 'lodash';

export const generateId = () => '_' + Math.random().toString(36).substr(2, 9);

export const getToken = async () => {
 const token = await AsyncStorage.getItem('token');

 return token ? JSON.parse(token) : null;
};

export const setToken = async (token) => await AsyncStorage.setItem('token', JSON.stringify(token));

export const clearToken = async () => await AsyncStorage.removeItem('token');

export const getItemFromAsyncStorage = async (item) => {
 const list = await AsyncStorage.getItem(item);

 return list ? JSON.parse(list) : item === 'films' ? {} : [];
};

export const setItemToAsyncStorage = async (item, value) => await AsyncStorage.setItem(item,
JSON.stringify(value));

export const addItemToExistingAsyncStorage = async (item, value) => {
 const list = await getItemFromAsyncStorage(item);
  const updatedList = [...list, value];
 await setItemToAsyncStorage(item, updatedList);
};

const getUserFilms = async (id) => {
 const films = await getItemFromAsyncStorage('films');

 return films[id];
};

export const getUserById = async (id) => {
 const ids = await getItemFromAsyncStorage('ids');
 const users = await getItemFromAsyncStorage('users');

 if (!ids.includes(id)) return { status: 404, message: 'User not found!' };

 const index = ids.indexOf(id);
 const user = users[index];

 if (user) {
   const films = await getUserFilms(user._id);
   return { status: 200, user: { ..._.omit(user, 'password'), ...films } };
 }

 return { status: 404, message: 'User not found!' };
```

```
};

export const loginController = async (login, password) => {
 const users = await getItemFromAsyncStorage('users');

 if (users.length === 0) return { status: 404, message: 'User was not found. Please, create your account!' };

 let user = null;
 let emailExists = false;
 let wrongPassword = false;

 users?.map(_ => {
  if (_.login === login) {
    emailExists = true;
    password === _.password ? user = _ : wrongPassword = true
  };
 });

 if (!emailExists) return { status: 404, message: 'User was not found. Please, create your account!' };

 if (wrongPassword) return { status: 401, message: 'Wrong username or password. Please, try again!' }

 if (user) {
  const { user: constuctorUser } = await getUserById(user._id);
  return { status: 200, user: { ...constuctorUser } };
 }

 return { status: 400, message: 'Unknown error. Try again later.' };
};

export const registerController = async (login, password, displayName) => {
 const userLogins = await getItemFromAsyncStorage('userLogins');
 const usernames = await getItemFromAsyncStorage('usernames');

 if (userLogins.includes(login)) {
  return { status: 400, message: 'Your username is already taken.' }
 };

 if (usernames.includes(displayName)) {
  return { status: 400, message: 'Your display name is already taken.' }
 };

 const user = {
  _id: generateId(),
  login,
  password,
  displayName,
 };

 const defaultFilms = await createFilms(user._id);
 await addItemToExistingAsyncStorage('users', user);
 await addItemToExistingAsyncStorage('ids', user._id);
 await addItemToExistingAsyncStorage('userLogins', login);
 await addItemToExistingAsyncStorage('usernames', displayName);
```

```
 return { status: 200, user: { ..._.omit(user, 'password'), ...defaultFilms} };

 //lists: all, favourite, anons, viewed, want to view(wishes)
};

export const createFilms = async (id) => {
 const newList = {
   favourites: [],
   announcements: [],
   viewed: [],
   toWatch: [],
 };
 const films = await getItemFromAsyncStorage('films');
 films[id] = { ...newList };

 await setItemToAsyncStorage('films', films);
  return newList;
};

export const addAnnouncements = async (announcement) => {
 const id = await getToken();
 const films = await getItemFromAsyncStorage('films');

 films[id].announcements = [ ...films[id].announcements, announcement ];
 await setItemToAsyncStorage('films', films);

 return films[id];
};

export const deleteAnnouncements = async (_id) => {
 const id = await getToken();
 const films = await getItemFromAsyncStorage('films');

 films[id].announcements = films[id].announcements.filter(_ => _._id !== _id);
 await setItemToAsyncStorage('films', films);

 return { status: 200, success: true };
};

export const moveToViewed = async (film) => {
 const id = await getToken();
 const films = await getItemFromAsyncStorage('films');

 films[id].viewed = [...films[id].viewed, _.omit(film, 'release')];
 await setItemToAsyncStorage('films', films);
 await deleteAnnouncements(film._id);

 return { status: 200, success: true };
};

export const deleteViewed = async (_id) => {
 const id = await getToken();
 const films = await getItemFromAsyncStorage('films');
```

```
 films[id].viewed = films[id].viewed.filter(_ => _._id !== _id);
 await setItemToAsyncStorage('films', films);

 return { status: 200, success: true };
};

export const moveToFavourites = async (film) => {
 const id = await getToken();
 const films = await getItemFromAsyncStorage('films');

 films[id].favourites = [...films[id].favourites, _.omit(film, 'release')];
 await setItemToAsyncStorage('films', films);
 await deleteAnnouncements(film._id);

 return { status: 200, success: true };
};

export const deleteFavourites = async (_id) => {
 const id = await getToken();
 const films = await getItemFromAsyncStorage('films');

 films[id].favourites = films[id].favourites.filter(_ => _._id !== _id);
 await setItemToAsyncStorage('films', films);

 return { status: 200, success: true };
};
```

api.js

```
import { API_KEY } from './config';
const genres = {
 12: 'Adventure',
 14: 'Fantasy',
 16: 'Animation',
 18: 'Drama',
 27: 'Horror',
 28: 'Action',
 35: 'Comedy',
 36: 'History',
 37: 'Western',
 53: 'Thriller',
 80: 'Crime',
 99: 'Documentary',
 878: 'Science Fiction',
 9648: 'Mystery',
 10402: 'Music',
 10749: 'Romance',
 10751: 'Family',
 10752: 'War',
 10770: 'TV Movie',
};

const API_URL = `https://api.themoviedb.org/3/discover/movie?api_key=${API_KEY}&sort_by=popularity.desc`;
```

```
const getImagePath = (path) =>
 `https://image.tmdb.org/t/p/w440_and_h660_face${path}`;
const getBackdropPath = (path) =>
 `https://image.tmdb.org/t/p/w370_and_h556_multi_faces${path}`;

export const getMovies = async () => {
 const { results } = await fetch(API_URL).then((x) => x.json());
 const movies = results.map(
  ({
    id,
    original_title,
    poster_path,
    backdrop_path,
    vote_average,
    overview,
    release_date,
    genre_ids,
  }) => ({
    key: String(id),
    title: original_title,
    poster: getImagePath(poster_path),
    backdrop: getBackdropPath(backdrop_path),
    rating: vote_average,
    description: overview,
    releaseDate: release_date,
    genres: genre_ids.map((genre) => genres[genre]),
  })
 );

 return movies;
};

Gallery.js (Recommended)
import * as React from 'react';
import {
 StatusBar,
 Text,
 View,
 StyleSheet,
 FlatList,
 Image,
 Dimensions,
 Animated,
 TouchableOpacity,
 Platform,
} from 'react-native';
const { width, height } = Dimensions.get('window');
import { getMovies } from '../../../utils/api';
import Genres from './Genres';
import Rating from './Rating';
import LinearGradient from 'react-native-linear-gradient';
import Icon from 'react-native-vector-icons/Ionicons';

import Drawer from 'react-native-drawer';
```

```
import DrawerView from '../../../components/Drawer';

import styles from './styles';
import COLORS from '../../../constants/colors';

const SPACING = 10;
const ITEM_SIZE = Platform.OS === 'ios' ? width * 0.72 : width * 0.74;
const EMPTY_ITEM_SIZE = (width - ITEM_SIZE) / 2;
const BACKDROP_HEIGHT = height * 0.65;

const Loading = () => (
 <View style={styles.loadingContainer}>
  <Text style={styles.paragraph}>Loading...</Text>
 </View>
);

const Backdrop = ({ movies, scrollX }) => {
 return (
  <View style={{ height: BACKDROP_HEIGHT, width, position: 'absolute' }}>
   <FlatList
    data={movies.reverse()}
    keyExtractor={(item) => item.key + '-backdrop'}
    removeClippedSubviews={false}
    contentContainerStyle={{ width, height: BACKDROP_HEIGHT }}
    renderItem={({ item, index }) => {
     if (!item.backdrop) {
      return null;
     }
     const translateX = scrollX.interpolate({
      inputRange: [(index - 2) * ITEM_SIZE, (index - 1) * ITEM_SIZE],
      outputRange: [0, width + 10],
      // extrapolate:'clamp'
     });
     return (
      <Animated.View
       removeClippedSubviews={false}
       style={{
        position: 'absolute',
        width: translateX,
        height,
        overflow: 'hidden',
       }}
      >
       <Image
        source={{ uri: item.backdrop }}
        style={{
         width,
         height: BACKDROP_HEIGHT,
         position: 'absolute',
        }}
       />
      </Animated.View>
     );
    }}
```

```
    />
    <LinearGradient
      colors={['rgba(0, 0, 0, 0)', 'white']}
      style={{
        height: BACKDROP_HEIGHT,
        width,
        position: 'absolute',
        bottom: 0,
      }}
    />
  </View>
);
};

export default function App({ navigation }) {
 const [movies, setMovies] = React.useState([]);
 const scrollX = React.useRef(new Animated.Value(0)).current;
 const _drawer = React.useRef(null);

 const openMenu = () => _drawer.current.open();

 React.useEffect(() => {
  const fetchData = async () => {
    const movies = await getMovies();
    // Add empty items to create fake space
    // [empty_item, ...movies, empty_item]
    setMovies([{ key: 'empty-left' }, ...movies, { key: 'empty-right' }]);
  };

  if (movies.length === 0) {
    fetchData(movies);
  }
 }, [movies]);

 if (movies.length === 0) {
  return <Loading />;
 };

 const burger = () => (
  <View style={styles.burder}>
    <TouchableOpacity
      activeOpacity={0.7}
      hitSlop={{ top: 15, right: 15, bottom: 15, left: 15 }}
      onPress={openMenu}
      style={styles.burderTouch}
    >
      <Icon name='menu-outline' color={COLORS.orange} size={35}/>
    </TouchableOpacity>
  </View>
 );

 return (
  <Drawer
    ref={_drawer}
```

```
  content={<DrawerView current='Home' navigation={navigation}/>}
  openDrawerOffset={0.3}
  tapToClose={true}
  side='left'
>
  <View style={styles.container}>
    { burger() }
    <Backdrop movies={movies} scrollX={scrollX} />
    <StatusBar hidden />
    <Animated.FlatList
      showsHorizontalScrollIndicator={false}
      data={movies}
      keyExtractor={(item) => item.key}
      horizontal
      bounces={false}
      decelerationRate={Platform.OS === 'ios' ? 0 : 0.98}
      renderToHardwareTextureAndroid
      contentContainerStyle={{ alignItems: 'center' }}
      snapToInterval={ITEM_SIZE}
      snapToAlignment='start'
      onScroll={Animated.event(
        [{ nativeEvent: { contentOffset: { x: scrollX } } }],
        { useNativeDriver: false }
      )}
      scrollEventThrottle={16}
      renderItem={({ item, index }) => {
        if (!item.poster) {
          return <View style={{ width: EMPTY_ITEM_SIZE }} />;
        }

        const inputRange = [
          (index - 2) * ITEM_SIZE,
          (index - 1) * ITEM_SIZE,
          index * ITEM_SIZE,
        ];

        const translateY = scrollX.interpolate({
          inputRange,
          outputRange: [100, 50, 100],
          extrapolate: 'clamp',
        });

        return (
          <View style={{ width: ITEM_SIZE }}>
            <Animated.View
              style={{
                marginHorizontal: SPACING,
                padding: SPACING * 2,
                alignItems: 'center',
                transform: [{ translateY }],
                backgroundColor: 'white',
                borderRadius: 34,
              }}
            >
```

```
            <Image
              source={{ uri: item.poster }}
              style={[styles.posterImage, { height: ITEM_SIZE * 1.2, }]}
            />
            <Text style={{ fontSize: 24 }} numberOfLines={1}>
              {item.title}
            </Text>
            <Rating rating={item.rating} />
            <Genres genres={item.genres} />
            <Text style={{ fontSize: 12 }} numberOfLines={3}>
              {item.description}
            </Text>
          </Animated.View>
        </View>
      );
    }}
  />
 </View>
 </Drawer>
);
};
```

AddAnnouncements.js

```
import React, { useState, useRef } from 'react';
import {
 View,
 Text,
 Image,
 Animated,
 ScrollView,
 TouchableOpacity,
} from 'react-native';
import Icon from 'react-native-vector-icons/Ionicons';
import DatePicker from 'react-native-date-picker'
import { launchImageLibrary } from 'react-native-image-picker';
import moment from 'moment';

import Input from '../../../../components/DefaultInput';

import COLORS from '../../../../constants/colors';

import styles from './styles';

const options = {
 title: 'Select Image',
 mediaType: 'photo',
 maxWidth: 1280,
 maxHeight: 720,
 storageOptions: {
  skipBackup: true,
  path: 'images',
 },
};
```

```
const Add = ({ create, enabled, date, _date, image, _image, about, title, rating, _rating, onChangeTitle, onChangeAbout,
showOriginal, toggleOriginalImage }) => {
 const [isOpened, _isOpened] = useState(false);
 const animatedDate = useRef(new Animated.Value(0)).current;

 const open = () => Animated.timing(animatedDate, { toValue: 1, duration: 500 }).start();

 const close = () => Animated.timing(animatedDate, { toValue: 0, duration: 500 }).start(() => _isOpened(false));

 const pickImage = () => launchImageLibrary(options, (response) => {
   if (response.didCancel) {
    console.log('User cancelled image picker');
  } else if (response.error) {
    console.log('ImagePicker Error: ', response.error);
  } else if (response.customButton) {
    console.log('User tapped custom button: ', response.customButton);
  } else {
    _image(response.uri);
  }
 });

 const renderPicker = () => (
  <TouchableOpacity
    activeOpacity={0.7}
    onPress={pickImage}
    style={styles.pickImage}
  >
    <Icon name='image-outline' size={40} color={COLORS.orange}/>
  </TouchableOpacity>
 );

 const picker = () => {
  if (image) return renderImage();

  return renderPicker();
 };

 const renderImage = () => (
  <>
    <View style={styles.showRow}>
      <TouchableOpacity
        activeOpacity={0.7}
        hitSlop={{ top: 10, right: 5, left: 10 }}
        onPress={toggleOriginalImage}
        style={styles.showOriginalTouch}
      >
        <Text style={styles.showOriginal}>{ showOriginal ? 'Show smooth' : 'Show original'}</Text>
      </TouchableOpacity>
    </View>
    <TouchableOpacity
      activeOpacity={0.7}
      onPress={pickImage}
      style={styles.imageTouch}
```

```
    >
      <Image
        source={{ uri: image }}
        resizeMode={showOriginal ? 'contain' : 'cover'}
        style={styles.image}
      />
    </TouchableOpacity>
  </>
)

const renderRating = () => Array(5).fill(null).map((_, i) =>
  <TouchableOpacity
    activeOpacity={0.7}
    onPress={() => _rating(i + 1)}
    style={[styles.ratingContainer, { backgroundColor: i + 1 <= rating ? COLORS.orange : COLORS.black }]}
  >
    <Text style={[styles.rating, { color: i + 1 <= rating ? COLORS.black : COLORS.orange }]}>{ i + 1 }</Text>
  </TouchableOpacity>
);

const bottom = () => (
  <View style={styles.bottom}>
    <TouchableOpacity
      activeOpacity={0.7}
      disabled={!enabled}
      hitSlop={{ top: 15, right: 15, bottom: 15, left: 15 }}
      onPress={create}
      style={[styles.bottomTouch]}
    >
      <Text style={[styles.create, { opacity: enabled ? 1 : 0.5 }]}>Create</Text>
    </TouchableOpacity>
  </View>
);

const toggleDate = () => {
  if (isOpened) {
    // _isOpened(!isOpened);
    close();
  } else {
    _isOpened(!isOpened);
    open();
  }
}

const renderDate = () => {
  const scaleX = animatedDate.interpolate({
    inputRange: [0, 1],
    outputRange: [60, 180],
  });

  const opacity = animatedDate.interpolate({
    inputRange: [0, 1],
    outputRange: [0, 1],
  });
```

```
  const rotate = animatedDate.interpolate({
    inputRange: [0, 1],
    outputRange: ['0deg', '180deg'],
  });

  return (
    <Animated.View style={[styles.animatedDate, { height: scaleX  }]}>
      <Text style={styles.dateLabel}>Release date</Text>
      <TouchableOpacity
        activeOpacity={0.7}
        onPress={toggleDate}
        style={styles.dateContainer}
      >
        <Text style={styles.date}>{ moment(date).format('DD MMM YYYY') }</Text>
        <Animated.View style={[styles.arrow, { transform: [{ rotate }] }]}>
          <Icon name='caret-up-outline' size={20} color={COLORS.light_orange}/>
        </Animated.View>
      </TouchableOpacity>
      <Animated.View style={{ opacity, alignItems: 'center', width: '100%' }}>
        { isOpened && (<DatePicker
          date={date}
          mode="date"
          textColor={COLORS.orange}
          onDateChange={_date}
          style={{ height: 135 }}
          minimumDate={new Date()}
          disabled={true}
        />)}
      </Animated.View>
    </Animated.View>
  );
};

return (
  <View style={styles.container}>
    <ScrollView showsVerticalScrollIndicator={false} contentContainerStyle={styles.contentContainerStyle}>
      { picker() }

      <Input
        label='Movie title'
        value={title}
        onChangeText={onChangeTitle}
        placeholder='Movie title'
      />
      <Input
        label='What is it about?'
        value={about}
        onChangeText={onChangeAbout}
        placeholder='What is it about?'
        containerStyles={{
          marginTop: 21
        }}
      />
```

```
      { renderDate() }

      <Text style={styles.ratingLabel}>Rating (Priority)</Text>
      <View style={styles.ratingRow}>
        { renderRating() }
      </View>

    </ScrollView>
    { bottom() }
  </View>
 );
};

export default Add;
```

ДОДАТОК В

Слайди презентації

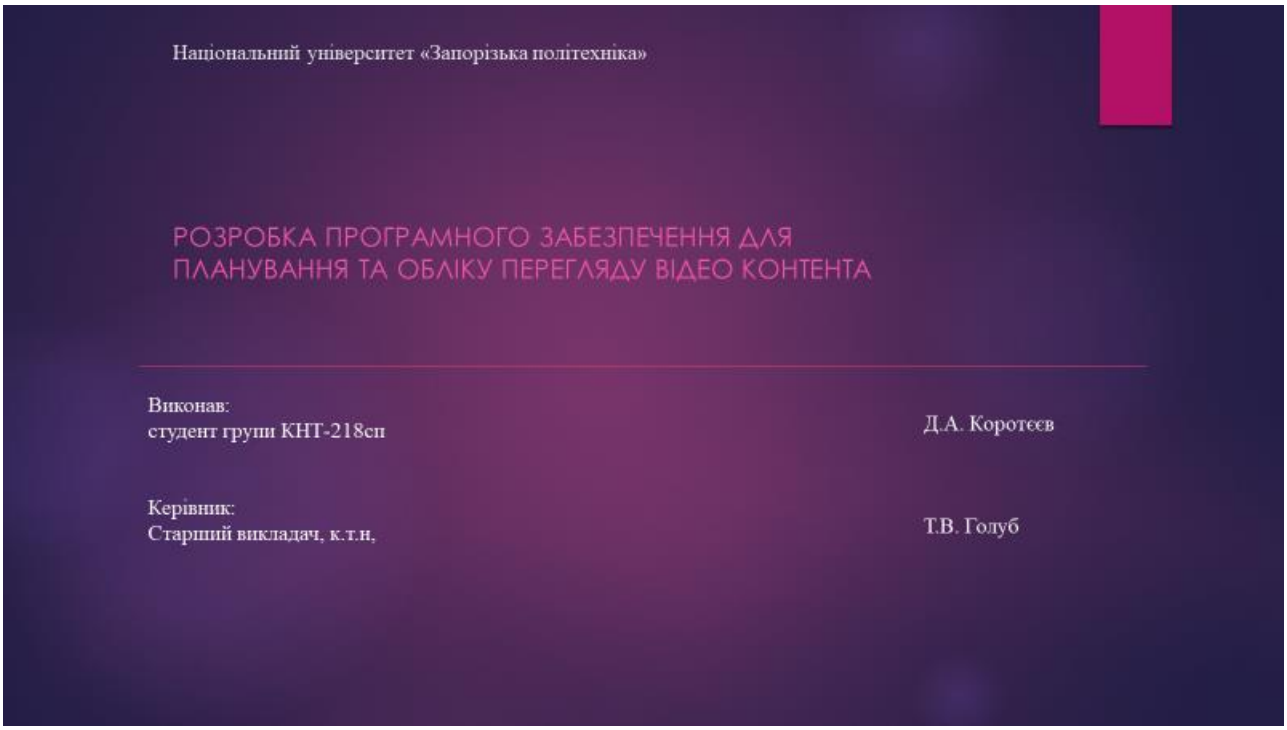

Рисунок В.1 – Слайд № 1

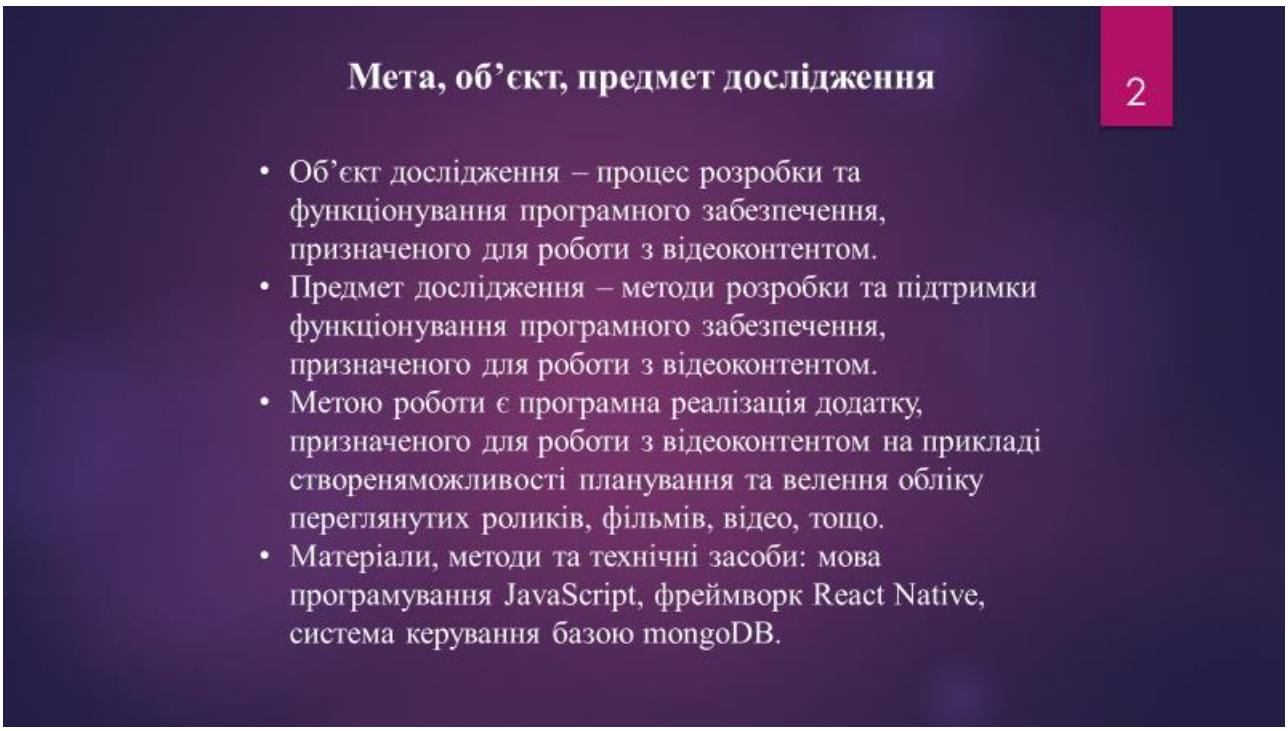

Рисунок В.2 – Слайд № 2

Порівняльна таблиця React Native і Flutter

3

Критерії порівняння	React Native	Flutter
початковий випуск	Травень 2015	Травень 2017
розробники	Facebook	Google
Мова	JavaScript	Dart
продуктивність додатків	Close to native	Fast. 60 fps
IDE підтримка	Range of IDE's and tools with JS support	Android Studio, Visual Studio Code, IntelliJ IDEA
Рідна зовнішність	Lower. Dependency on third-party apps	Better. Has access to device core functionalities
Гаряче перезавантаження	Так	Так
час виходу на ринок	Повільніше ніж Flutter	Швидше
Повторне використання коду	90% коду багаторазового використання	50-90% коду багаторазового використання
Програми	Tesla, Discord, Instagram	KlasterMe, PostMuse Reflectiy

Рисунок В.3 – Слайд № 3

Загальна структура додатку

4

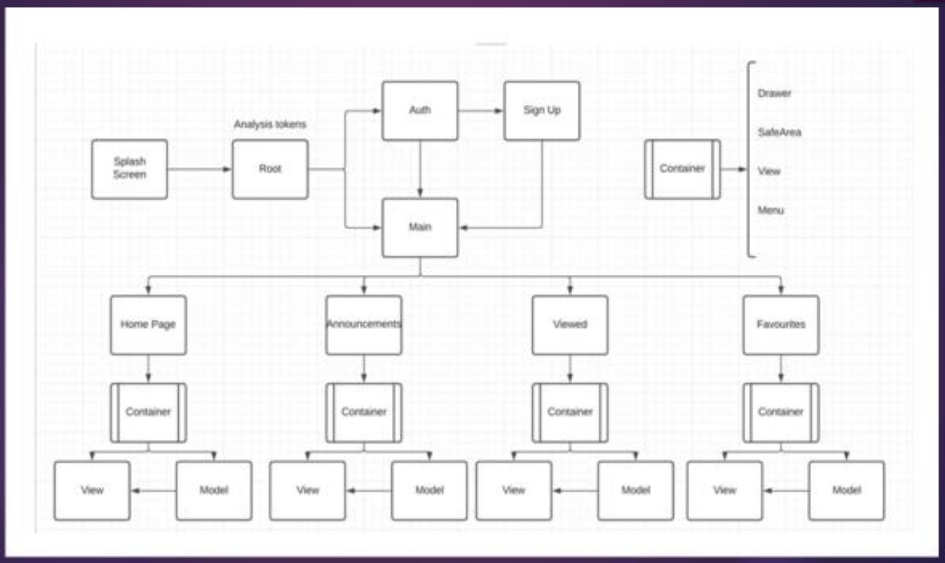

Рисунок В.4 – Слайд № 4

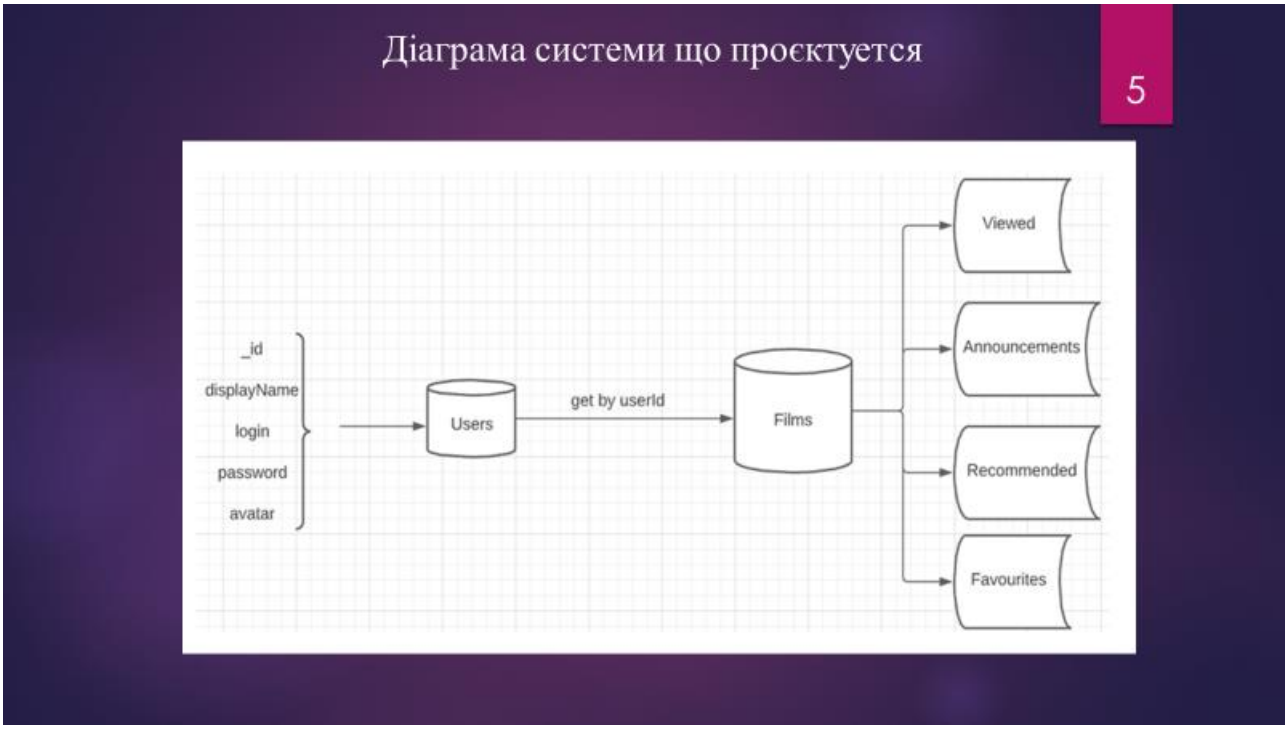

Рисунок В.5 – Слайд № 5

Назва поля	Тип поля	Розмір поля	Ключ	Опис
_id	uniqueMongoId	< 32	Так	Призначене для унікальності записів у полях
title	Текстовий	24	—	Призначене для збереження Назви фільму
announceDate	Date	-	—	Призначене для зберігання дати виходу фільму та таймеру
description	Текстовий	128	—	Призначене для зберігання опису фільму
rate	Чисельний	1	—	Призначене для зберігання рейтингу фільму
image	Текстовий	256	-	Призначений для зберігання посилання на зображення фільму

Рисунок В.6 – Слайд № 6

Структура таблиці «Users»

7

Назва поля	Тип поля	Розмір поля	Ключ	Опис
_id	uniqueMongoId	< 32	Так	Призначене для унікальності записів у полях
displayName	Текстовий	15	—	Призначене для збереження Нікнейму користувача
login	Текстовий	12	—	Призначене для зберігання логіну користувача
password	Текстовий (md5)	18	—	Призначене для зберігання шифрованого паролю користувача
avatar	Тексовий	256	—	Призначене для зберігання посилання на аватар користувача

Рисунок В.7 – Слайд № 7

Інтерфейсні вікна програми – «Вікно авторізації»

8

Назва програми

Рисунок В.8 – Слайд № 8

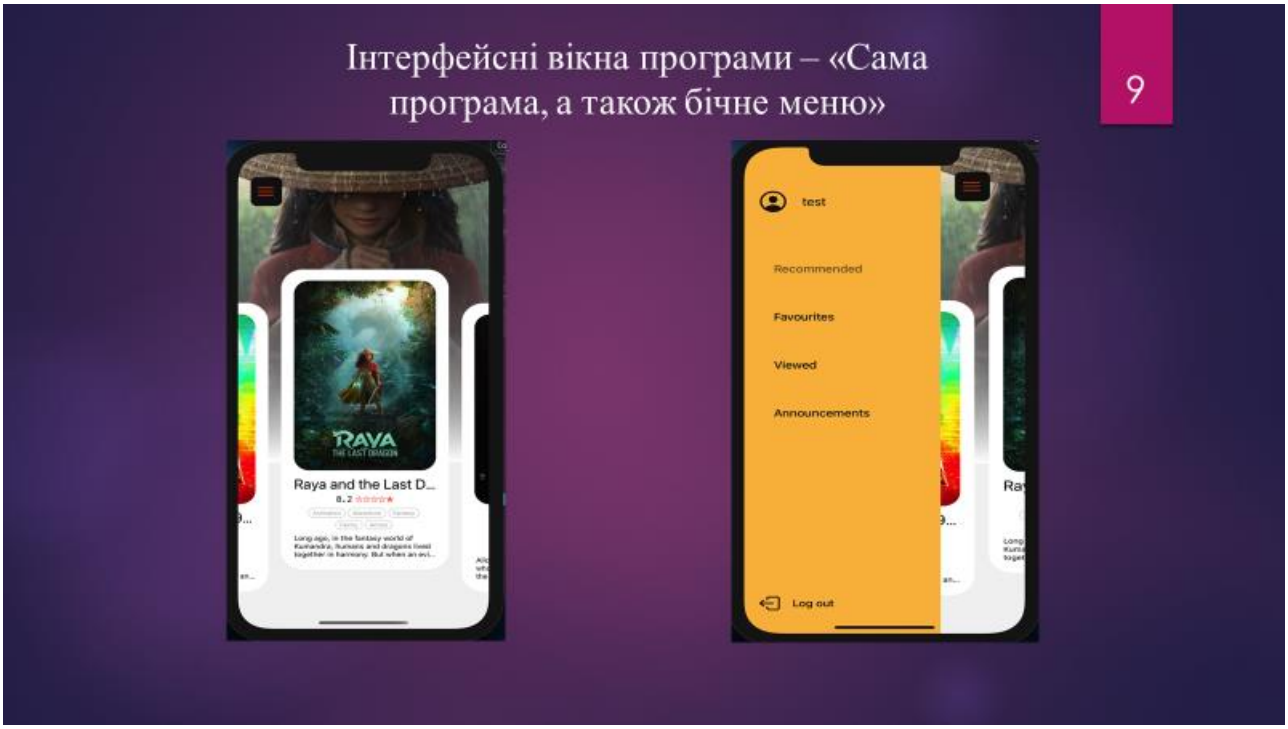

Рисунок В.9 – Слайд № 9

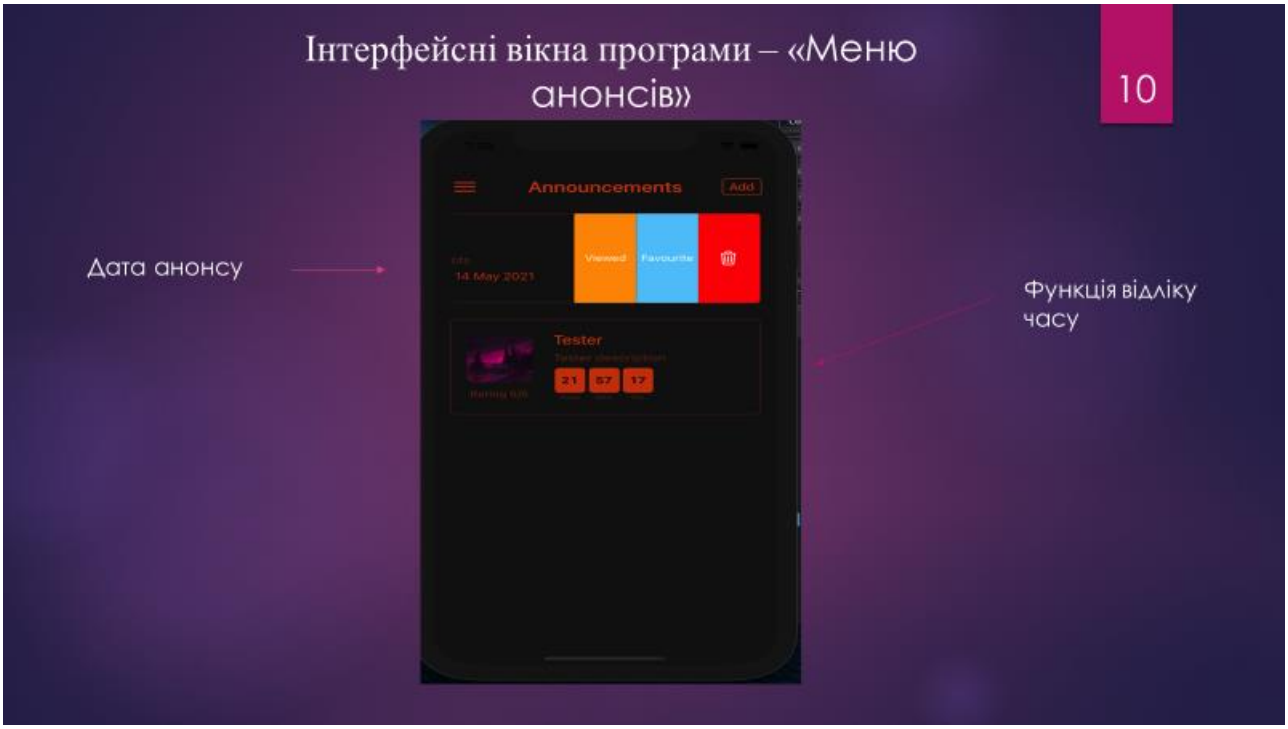

Рисунок В.10 – Слайд № 10

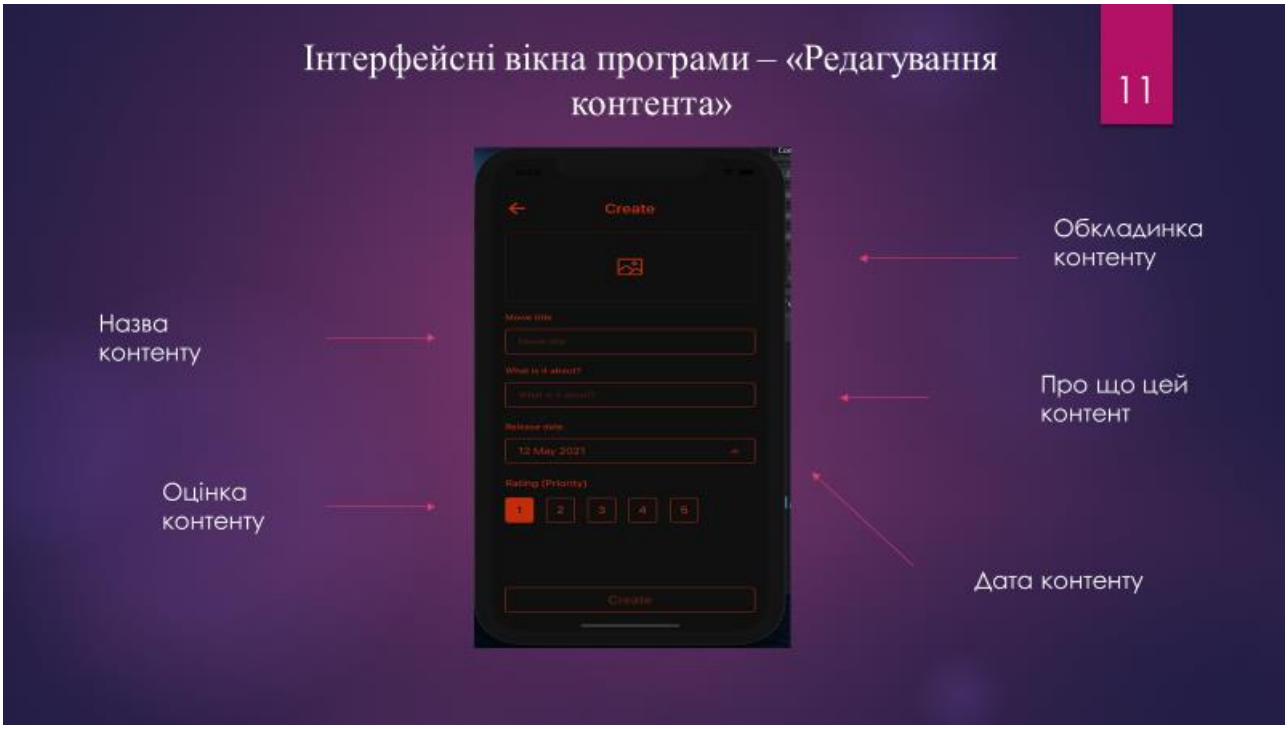

Рисунок В.11 – Слайд № 11

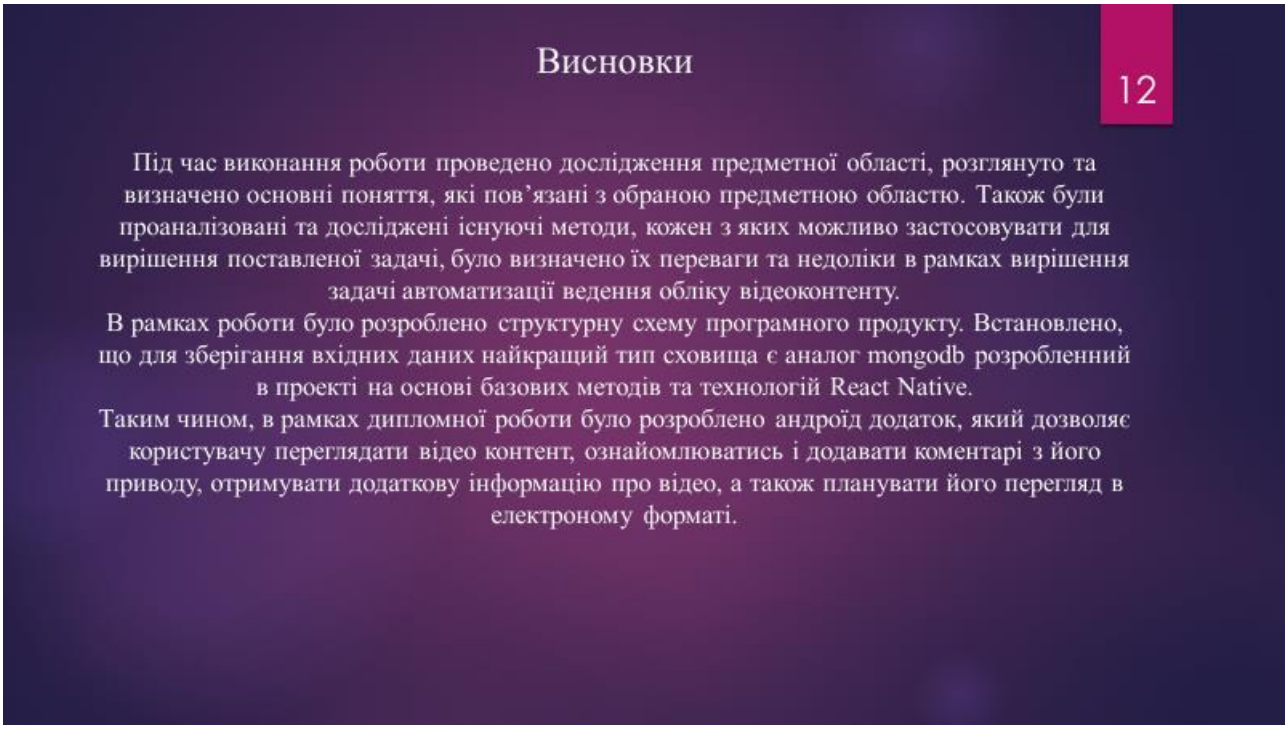

Рисунок В.12 – Слайд № 12