НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки
Кафедра Автоматики та управління в технічних системах

До захисту допущено
Завідувач кафедрою
_____ Ролік О. І.
(підпис) (ініціали, прізвище)

«__»_____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 « Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Кінематографічна соціальна мережа з використанням хмарних сервісів»

Виконав:
студент IV курсу, групи ІА-62
Тряпко Олексій Олексійович _____

Керівник: асист.
Дорога-Іванюк Олена Олександрівна

_____

Рецензент: _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 151 Автоматизація та комп'ютерно-інтегровані технології
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Ролік О. І.
(підпис) (ініціали, прізвище)

«__» _____ 2020 р.

ЗАВДАННЯ

на дипломний проект студенту

Тряпко Олексію Олексійовичу

1. Тема проекту «Кінематографічна соціальна мережа з використанням хмарних сервісів», керівник проекту Дорога-Іванюк Олена Олександрівна, асистент, затверджені наказом по університету від «__» _____ 2020 р. №___

2. Термін подання студентом проекту 9 червня 2020 року

3. Вихідні дані до проекту:

Мова програмування JavaScript, середовище програмування VisualStudio Code, обрана для розробки клієнут бібліотека - React

4. Зміст пояснювальної записки: 1. Вступ 2. Огляд існуючих рішень 3. Реалізація 4. Інструкція користувача 5. Висновки

5. Перелік графічного матеріалу: діаграма структури баз данних, діаграма класів, діаграма розгортання, структурна схема

6. Дата видачі завдання: 11 березня 2020 року.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів проекту	Примітка
1	Аналіз теоретичних матеріалів та вивчення предметної області	15.04.2020 р.	
2	Вибір технологій та засобів для реалізації задачі	19.04.2020 р.	
3	Огляд існуючих рішень з тематики роботи	27.04.2020 р.	
4	Реалізація проекту	05.05.2020 р.	
5	Оформлення текстової документації	08.06.2020 р.	

Студент _____ Тряпко О.О.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Дорога-Іванюк О.О.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Тряпко О.О. Кінематографічна соціальна мережа з використанням хмарних сервісів. КПІ ім. Ігоря Сікорського, Київ, 2020.

Пояснювальна записка містить 61 с. тексту, 11 рисунків, 3 додатоки та 11 літературних джерел.

Ключові слова: соціальна мережа, кіно, хмарні сервіси, комунікація.

Об’єктом розробки є кінематографічна соціальна мережа з використанням хмарних сервісів.

Мета розробки – створення спецалізованої мережі для розвитку та комунікації однодумців.

Дипломний проект присвячено проектуванню та розробці кінематографічної соціальної мережі з використанням хмарних сервісів, що являє собою суміш соціальної мережі та кінематографічного довідника. Було проведено аналіз існуючих рішень та розроблено діючий додаток для використання з великою кількістю функціионалу для різних цілей.

SUMMARY

Tryapko O.O. Cinematic social network using cloud services. Igor Sikorsky KPI, Kyiv, 2020.

The explanatory note contains 61 pages of text, 11 images, 3 additions and 11 literary sources.

Key words: social network, cinema, cloud services, communication.

The object of development is the Cinematic social network with the use of cloud services.

The purpose of the development is to create a specialized network for the groving and networking of like-minded people.

The diploma project is devoted to the design and development of a cinematic social network using cloud services, which is a mix of a social network and a cinematic guide. An analysis of existing solutions was performed and an existing application was developed for use with a large number of features for various purposes.

Пояснювальна записка
до дипломного проекту
на тему: « Кінематографічна соціальна мережа з використанням хмарних сервісів»

Київ – 2020 рік

ЗМІСТ

ВСТУП

У наші дні без перебільшення можна сказати, що багато хто з нас проводять велику частину свого життя в соціальних мережах. Соціальні мережі змінили спосіб взаємодії один з одним. Такі сервіси дають можливість отримувати інформацію відносно ваших інтересів. Ви можете слідкувати за друзями, їхніми постами, фотографіями та відео.

Соціальні медіа не просто важливі для нашого особистого життя, вони також є чудовим інструментом для бізнесу, або тих хто намагається розвинути свою професійну мережу або шукає роботу. З кожним роком з'являються все більше соціальних мереж. Кожна з них може запропонувати щось різне і може принести користь вам у багатьох різних направленнях.

Дуже важливо витрачати свій час з користю та отримувати цікаву та необхідну вам інформацію. При використанні не специфічній певному напряму соціальної мережі ми витрачаємо багато часу на перегляд не корисної, чи не цікавої для нас інформації. Натомість при використанні соціальної мережі з конкретною спеціалізацією ми отримаємо специфічну інформацію в котрій ми зацікавлені.

За останні роки стрімко розвиваються стрімінгові сервіси, котрі мають купу унікальних кінофільмів та інших медіа. Користувач стикається з проблемою пошуку цікавого йому фільму чи тв-шоу. Для того, щоб оглянути існуючі пропозиції користувачу потрібно переглянути усі стрімінгові сервіси котрі йому цікаві та порівнювати запропоновані варіанти. Для порівняння обраних фільмів користувачу потрібно отримати рецензії, відгуки та враження інших людей чи ресурсів, тобто потрібно шукати в просторах інтернету, писати друзям, тощо. Після перегляду фільму користувач використовує ще одну соціальну мережу для того, щоб поділитися враженнями після перегляду фільму. Це лише частина потреб, для вирішення котрих потребується купа різних ресурсів та сервісів.

Переваги та доцільність використання спеціалізованої соціальної мережи з напрямом у кіно очевидна. У нас час існує велика кількість застосунків схожого характеру, але проблема таких соціальних мереж в тому, що в більшості випадків такі мережі - бази даних фільмів та інформації стосовно цих фільмів. Існуючі рішення не дають можливість користувачам комфортно спілкуватися, ділитися враженнями та інформацією.

Отже об’єктом дослідження дипломного проекту є застосунок, який поєднує у собі найбільш важливі функції для соціальної мережі спеціалізованій на кіно такі, як: база фільмів та інформації стосовно цих фільмів, можливість оцінки коментування та обговорення специфічного фільму, створення довгострокових та тимчасових публікацій, створення опитувань, створення власних списків фільмів, створення подій, розумний алгоритм рекомендацій та інше.

Метою дипломного проекту є розробка веб-сервісу спеціалізованої соціальної мережі з використанням хмарних сервісів, розподілених баз даних та розумними алгоритмами рекомендацій.

Для досягнення поставленої мети були сформульовані та вирішені наступні задачі:

— огляд та аналіз існуючих соціальних мереж;

— вибір стеку технологій та засобів для реалізації;

— сформувати вимоги до інтерфейсу та функціоналу;

— розроблення схеми бази даних;

— реалізація додатку.

Особливістю розроблюваної системи є використання розумного алгоритму рекомендацій фільмів, котрий базується на ваших інтересах. Також в додатку реалізовано алгоритм для оцінки настрою написаного відгуку.

Крім того цей додаток має інтеграцію з іншими сервісами, що дає можливість отримувати оновлену інформацію, та делігувати операції.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

На разі існує багато додатків котрі виконують функції кінематографічних довідників, платформ спілкування та поширення інформації. Ці додатки виконують одну чи декілька функцій одночасно. Отже розглянемо найпопулярніші з них.

1.1 IMDB

IMDB = це гігантський збірник фільмів, телевізійних шоу та відеоігор. Основне його використання - пошук детальної інформації про будь-якого актора, продюсера чи рейтингу фільму. Відкривши певний фільм, користувач бачить опис, трейлери, фотографії, список акторів, рейтинг, відгуки та інше. Саме кількість інформації робить IMDB дуже корисним. Ресурс дає інформацію не тільки про фільми, котрі вже давно вийшли в прокат, а також про новинки.

IMDB надає можливість користувачам створювати безкоштовні облікові записи. Авторизований користувач має можливість додавати фільми до персонального списку фільмів для перегляду. Цей список є приватним за замовчуванням, але його можна зробити публічними та отримати посилання для розповсюдження.

Сервіс дає можливість слідкувати за списками фільмів інших користувачів. Кількість списків не обмежена для одного користувача, також вони можуть бути рейтинговими. Авторизований користувач може створювати власні списки.

IMDB має блок с новинами. Ці новини пов’язані не лише напряму з кіно та тв-шоу, а також з кіноіндустрією в цілому. Новини дуже різноманітні та актуальні через те, що сервіс виконує роль агрегатора, тобто новини поступають з багатьох різних ресурсів.

Сервіс надає можливість оформити платну підписку. Платна підписка надає користувачу доступ до великої кількості професійної інформації. Підписка

націлена на розвиток професійних навичок: акторства, продюсерства, письменництва та інше. Також користувач отримує можливість виставляти фотографії та резюме для кастингу.

Переваги застосунку

— велика база фільмів;

— детальна інформація;

— авторитетні оцінки фільмів;

— актуальність даних;

— список побажань для власного перегляду;

— топи та списки;

— авторизація за допомогою різних ресурсів.

Недоліки застосунку

— орієнтовано на англомовну аудиторію;

— платна підписка;

— користувачі не мають можливості створювати персональні публікації;

— користувачі не мають можливості спілкуватися приватно;

— деякі частини додатку мають застарілий інтерфейс.

1.2 Letterboxd

Letterboxd – повноцінна соціальна мережа, що спеціалізується на кіно. Цей сервіс дає велику базу фільмів з додатковою інформацією. В додатку реалізовані списки для власного перегляду, також є публічні списки фільмів. Публічні списки можуть бути нумеровані.

В сервісі не реалізована реєстрація за допомогою соціальних мереж. Авторизований користувач може підписуватись на інших користувачів, активність цих користувачів формує сторінку активності.

В Letterboxd реалізовано панель користувача на котрій відображено основну інформацію відносно активності. Користувач бачить кількість

переглянутих фільмів, кількість підписників та підписчиків, кількість списків та інше. В додатку реалізована система тегів, авторизований користувач може додавати теги к фільмам та виконувати пошук фільмів та користувачів зацікавлених в цих фільмах за тегом. Є платна підписка, котра дозволяє отримувати персоналізовані рекомендації для перегляду, фільтрацію активності користувачів та інше.

Переваги застосунку

— функціонал підписок;

— статистика користувача;

— статистика фільмів;

— використання інформації стрімінгових сервісів.

Недоліки застосунку

— відсутність авторизації за допомогою соціальних мереж;

— орієнтовано на англомовну аудиторію;

— зміна ім’я дозволена лише при наявності платної підписки.

1.3 Rotten Tomatoes

Rotten Tomatoes – веб застосунок, що об'єднує думки критиків кіно і телевізійних критиків і підсумовує бал, який є "свіжим" або "гнилим". Користувачі використовувують Rotten Tomatoes для пошуку оглядів фільмів та квитків у кінотеатри. Сервіс розміщує оцінки - "Томатометр" поруч із списками квитків на фільми.

Оцінка "Томатометр" - заснована на думках критиків кіно та телебачення – є вимірюванням критичної оцінки рекомендації для користувачів. Оцінка являє собою відсоток відгуків професійних критиків, які є позитивними для даного фільму чи телешоу. Оцінка "Томатометр" розраховується для кіно або телешоу після того, як він отримає щонайменше п'ять оглядів.

Rotten Tomatoes має команду адміністраторів, котра відбирає рецензії на фільми і телепередачі. Команда відбирає огляди на фільми та телепередачі котрі в свою чергу формують оцінку "Томатометр". Крім відсіювання неякісних рецензій команда адміністраторів розподіляє рецензії на позитивні та негативні.

Окрім оцінки критиків сервіс має також окремий рейтинг заснований на оцінці звичайних користувачів. Рейтинг формується на основі проценту позитивних відгуків. Залишити відгук може лише користувач котрий придбав квиток на відповідний фільм, таким чином сервіс проводить верифікацію відгуків користувачів.

Переваги застосунку

— зручний інтерфейс;

— багато інформації відносно фільмів;

— фільми мають посилання на стрімінгові сервіси.

Недоліки застосунку

— відсутня авторизація за допомогою Google;

— орієнтовано на англомовну аудиторію;

— оцінки фільмів сформовані неправдиво;

— користувач не має можливості залишити відгук без квитка на фільм;

— сервіс не надає можливость комунікації між користувачами;

— користувачі не маєть можливості створювати власні публікації;

— оцінка якості відгуків проводиться мануально.

Висновки до розділу 1

У цьому розділі було розглянуто декілька існуючих рішень, котрі виконують деякі з вимог до дипломного проекту.

Було розглянуто три веб сервіси. Rotten Tomatoes не виконує всі задачі пов’язані з комунікацією, але має гарну систему оцінювання та застарілу систему підтвердження оцінок. IMDB на відміну від Rotten Tomatoes має ширший

функціонал для користувача, кращу систему оцінок та можливості для створення персональних матеріалів, але цей сервіс також не має повноцінної системи публікацій та комунікування між користувачами. Веб сервіс Letterboxd покриває більшість задач поставлених у дипломному проекті, але він не має авторизації через соціальні мережі та має обмеження які знімаються тільки завдяки платній підписці.

Отже, на даний момент не існує готового рішення котре би цілком покривало всі поставлені задачі. Більшість веб сервісів працюють як довідники та не мають повноцінної системи комунікування та розповсюдження інформації для користувачей. Ідеальний додаток повинен мати можливості популярний соціальних мереж таких, як facebook та instagram та в той самий час мати кінематографічну спрямованість з великою кількістю специфічних можливостей для цього направлення. Важливо зауважити додаток не повинен мати набір незадежних можливостей, котрі можуть надати більшість соціальних мереж, натомість цей додаток повинен зв’язувати можливсості кінематорграфічного довідника та соціальної мережі котра впроваджує поширення даних, інформаціїт та інструменти комунікування.

2 ФОРМУВАННЯ ВИМОГ ДО ФУНКЦІОНАЛУ ТА ІНТЕРФЕЙСУ

2.1 Функціональні вимоги

Функціональні вимоги оипсують та визначають які можливості повинна мати система у цілому та які можливості повинен мати користувач для взаємодії з данною системою.

Застосунок повинен налічувати засоби для отримання інформації відносно кінематографу та поширення Користувачі застосунку повинні мати можливість використувати додаток як у пізновальних цілях так і задля поширення інформації та комунікування. Ці вимоги були взяті у основу вимог до застосунку.

Виходячи з цього розроблений додаток повинен мати наступні функціональні вимоги:

— авторизація;

— каталог фільмів;

— користувацькі публікації;

— тимчасові користувацькі публікації;

— користувацькі налаштування;

— особисті сторінки користувачів;

— система підписок між користувачами;

— повідомлення;

— особистий список «Передевитися пізніше»;

— події;

— опитування;

— списки фільмів;

— глобальний пошук;

— огляди фільмів;

— система рекомендацій фільмів;

— особисті повідомлення;

— швидкісний пошук фільмів.

2.1.1 Авторизація

Головною вимогою до авторизації є створення умов для користувача за яких він може дуже швидко отримати доступ до основної частини додатку. Користувач повинен мати можливість отримати доступ до додатку зручним йому способом, тобто додаток повинен мати декілька способів авторизації та реєстрації. Обов'язковою вимог також є процедура відновлення паролю. Виходячи з цих вимог були побудовані наступні функціональні вимоги:

— кожна процедура повинна мати окрему сторінку з навігацією між собою. Тобто авторизація реєстрація та відновлення паролю – окремі сторінки між котрими повинна бути навігація;
— авторизація за допомогою email та паролю користувача;
— авторизація за допомогою Google;
— авторизація за допомогою Facebook;
— реєстрація за допомогою мінімальної кількості інформації користувача;
— реєстрація за дпомогою Google;
— реєстрація за допомогою Facebook;
— відновлення паролю через email повідомлення;
— усі форми повинні мати валідацію для уникнення відправки невалідних запросів на авторизація чи реєстрацію. Валідаційні помилки повинні мати опис конкретної помилки;
— при невдалій спробі авторизації чи реєстрації користувач повинен бути сповіщений причиною невдалого запиту;
— використання хмарного сервісу для делігування процесу авторизації.

2.1.2 Каталог фільмів

Додаток спеціалізований на кінематографі тому він повинен мати обширну базу фільмів та додаткову інформацію відносно них. База фільмів повинна бути

актуальною та обновлятися без участі користувачів чи адміністраторів. Пошук та навігація між фільмами повинна бути прозора та зрозуміла.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— використовувати зовнішній API для фільмів та інформації віносно них. повині регулярно оновлюватись та мати детальну інформацію відносно кожного фільму;

— обрати сховище для фільмів котре надає можливість найефективінше виконувати пошук з використанням складних фільтрів;

— для відображення фільмів потрібно зробити сторінку з усіма фільмами, яка б відображала постер та коротку інформацію відносно кожного фільму. Також кожен фільм повинен мати інтерфейс додавання цього фільма до списку перегляду.

— сторінка з усіма фільмами повинна мати пагінацію.

— для швидкого пошуку фільмів потрібно зробити розширений пошук, котрий охоплює осноновні данні фільму. Пошук може здійснюватися за такими параметрами як нозва, жанр, рейтинг, актори, режесери, рік виходу та інші;

— інформація на сторінці конкретного фільму повинна бути розбита по категоріям;

— кожна сторінка фільму повинна налічувати блок з дискусією для конкретного фільму. Кожен авторизований користувач повинен мати можливість відправити повідомлення за допомогою цього блоку.

2.1.3 Користувацькі публікації

Додаток повинен мати можливість ствоення довгострокові користувацькі публікації. Для поширення інформації та висловлювання власних думок користувач може створити публікацію.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— публікація повинна містити ім’я користувача та аватар, як посилання на сторінку користувача;

— основна частина публікації – текст;

— в текстову частину публікації повинна бути можливість додавати посилання на фільм. При створенні публікації користувач може ввести особливий символ та почати вводити назву фільму, після цього повинен з’явитися блок пошуку з варіантами котрі відходять під початок назви фільму, котра була введена користувачем. Публікація може містити безліч посилань такого роду. При натисканні на посилання покристувач буде перенаправлений на сторінку фільму;

— публікація може містити одне або декілька зображень. Після завантаження зобрадення користувач повинен мати можливість відредагувати зображення, а саме змінити розміри зображення;

— публікація може містити додаткові матеріали такі як опитування, списки фільмів, події. Прикріплений матеріал повинен мати коротку інформацію. Опитування повинно мати заголовок описання та зображення, список фільмів повинен мати заголовок, зображення та перші три фільму з списку, подія повинна містити заголовок, описання, зображення та дату проведення події. Кожен додатковий матеріал повинен бути посиланням, тобто якщо користувач натисне на матеріал його треба перенаправити на сторінку з конкретним матеріалом. Якщо публікація не містить зображень, то на місце зображення повинно стати зображення з додаткового матеріалу;

— користувач повинен мати можливість відреагувати на публікацію за допомогою реакцій. Кожна публікація повинна мати список реакцій котрі може обрати користувач. Користувач може залишити безліч унікальних реакцій під одною публікацією;

— користувач повинен мати можливість залишити коментар під публікацією та передивитися коментарі інших користувачів.
— Якщо публікація створена поточним користувачем він повинен мати можливість відредагувати її та видалити.

2.1.4 Тимчасові користувацькі публікації

Для поширення інформації в тимчасовому, швидкому форматі користувач повинен мати можливість створювати тимчасові публікації – історії. Публікації повинні мати можливість розміщувати додаткові матеріали та інтерактивні елементи. Відносно кожної публікації користувач повинен мати можливість висловитися.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— публікація може містити зображення. Після завантаження зобрадення користувач повинен мати можливість відредагувати зображення, а саме змінити розміри зображення;
— користувач повинен мати можливість змінювати колір фону публікації за дпомогою спецільного інструменту – палетки. Палетка повинна містити заготовлені кольори, також користувач повинен мати можливість обрати свій унікальний колір;
— публікація може містити один або декілька блоків з текстом. Кожен блок користувач повинен мати можливість перетягнути у бажане місце. Кожен блок тексту може мати різний колір, колір можна змінити за допомгою палетки так само як і фон;
— публікація може містити прикріплений фільм. Постер фільму повинен зайняти основне місце в публікації під текстовими блоками. В публікації з фільмом повинен бути інтерфейс додавання цього фільму

до списку перегляду. Також у доданому фільмі повинен бути інтерфейс виставлення йому рейтингу;

— публікація може містити додаткові матеріали так само як і довгострокві публікації. На відміну від довгострокових публікацій в історіях додатковий матеріал повинен мати лише заголовок. Заголовок додаткової інформації повинен бути посиланням. Якщо користувач натисне на додатковий матеріал його буде перенаправлено на сторінку цього матеріалу;
— користувач повинен мати можливість відреагувати на історію. Реакція може бути текстовою, або швидкою реакцією у вигляді смайлу. Реакція на історію буде відправлена автору цієї історії в особисте повідомлення;
— користувач повинен мати можливість додати швидке опитування до публікації. Опитування може містити одне питання та одну чи більше відповідей. Коли користувач натискає на відповідну відповідь він бачить як відповіли інші користувачі на це питання у відсотках відносно відповіді.

2.1.5 Користувацькі налаштування

Додаток повинен надавати можливість користувачу змінювати основні данні аккаунту. Перш за все користувач повинен мати можливість змінити емейл та пароль облікового запису. Також користувач повинен мати можливість видалити аккаунт. Так як, додаток налічує повідомлення, як внутрішні так і емейл сповіщення треба реалізувати налаштування повідомленнь, тобто дати можливість відрегулювати які повідомлення можуть приходити. За замовчуванням усі користувачі можуть бачити публікації та данні розміщенні користувачем. Потрібно реалізувати налаштування приватності.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— користувач повинен мати сторінку з налаштуваннями розбитими на блоки: налаштування аккаунту, налаштування приватності, налаштування приватності;
— користувач повинен мати можливість змінити поштову адресу аккаунту;
— користувач повинен мати можливість змінити пароль аккаунту;
— користувач повинен мати можливість видалити аккаунт;
— користувач повинен мати можливість змінити налаштування сповіщень, як емейл так і внтурішніх;
— користувач повинен мати можливість змінити налаштування приватності своїх публікацій та персональних даних. Кожен тип медіа повинен мати окремі налаштування на вибір: доступно всім, доступно підписникам, нікому не доступно.

2.1.6 Особисті сторінки користувачів

Кожен користувач повинен мати персональну сторінку, яка б відображала основну інформацію користувача та медія створені ним. Сторінка користувача повинна бути розбита на таби за складом.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— додавання та зміна персональної інформації користувача;
— можливість створення та перегляду створених публікацій, подій, опитуваннь, оглядів, списків;
— можливість підписатися та відписатися на користувача;
— можливість відправити персональне повідомлення користувачу;
— можливість роботи з списком для перегляду фільмів.

Користувачі повинні мати можливість підписатися на інших користувачів. На основі підписок повинні з’являтися медіа ресурси при відповідних

налаштуваннях приватності. Свої підписки та підписників можна передивитися на сторінці користувача, та цій же сторінці повинна бути можливість підписатися, або відписатися від користувача.

2.1.7 Повідомлення

Додаток повинен мати поввідомлення. Повідомлення повинні бути різних типів. Перший тип повідомлень – заснований на активності користувача, та інших користувачів відносно публікацій поточного. Вони повинні відображатися в інтерфейсі додатку, як активність з роз'ясненнями та посиланнями та подію. Ці повідомлення повині дублюватися push повідомленнями при згоді користувача. Другий тип повідомленнь – email повідомлення. Ці повідомлення можуть, або відносно нововведень в додатку, або через певні дії користувача.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— push повідомлення за допомогою хмарних сервісів;

— повідомлення в додатку;

— емейл повідомлення;

— повідомлення повинні бути прив'язані до налаштуваннь користувача.

2.1.8 Особистий список «Передевитися пізніше»

Кожен користувач повинен мати список фільмів для перегляту пізніше. Фільми можна додавати з основного каталогу фільмів та з усіх місць де є превью фільму, тобто в історії, списку фільмів, сторінці фільму та інших місцях повинен бути інтерфейс керуванням конкретного фільму у цьому списку. Фільм може бути не в списку, доданий до списку та переглянутий. Необхідно розробити зрозумілий та функціональний інтефейс для регулювання статусу певного

фільму. Список фільмів користувач може передивитися на персональній сторінці користувача.

2.1.9 Події

Події – публікації котрі повинні розширювати межі додатку. Мета цього типу публікацій вивести комунікацію у реальний світ із інтернету. Користувачі можуть створювати події та приєднуватися до існуючих. Події присвячені кіно, наприклад, обговорення нового фільму, чи перегляд улюбленного.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— створити інтефейс створення та редагування подій. Конструктор подій повинен містити інтерфейс додавання зображень, місця проведення часу та дати;
— підключити сторонній ресурс для вибору місця проведення події;
— подія повинна мати два списки людей, преший – список зацікавлених людей, другий – список людей котрі націлені ідти;
— подія може бути приватною чи публічною;
— сторінка події повинна мати чат для обговорення.

2.1.10 Опитування

Опитування – публікаціїї з комплексними опитуваннями. Кожен користувач може створити власне опитування, або пройти чиєсь. Опитування налічує різні види питаннь такі як: питання з однією відповіддю, питання з багатьма відповідями, питання з розгорнутою відповіддю, питання зі шкалою. Автор поитування повинен мати можливість продивитися статистику відповідей у анонімному форматі. Автор повинен бачити, як загальну статистику по всім питанням так і відповіді окремих користувачів на всі питання.

На основі цих вимог були розроблені наступні функціональні вимоги до додатку:

— користувач може стоврювати опитування;
— користувач може додати зображення до опитування, опис опитування та заголовок;
— кожне питання може мати заголовок, писання та зображення;
— користувач може дублювати питання для пришвидшення створення опитування;
— питання може бути позначено як обов’язкове. Якщо на обов’язкове питання відповідь не була надана, опитування не можна завершити;
— користувач може подивитися прев’ю опитування, до його публікації;
— опитування може складатися з безлічі питанньь;
— автор опитування може закрити його;
— автор опитування може передивитися статистику відповідей на питання. Статистика повинна бути двох видів. Загальна статистика – статиска показує як відповіли всі користувачі на всі питання та індивідуальна показує як відповіли окремі користувачі на всі питання, при цьому статистика анонімна.

2.1.11 Списки фільмів

Користувачі можуть створювати списки фільмів, як нумеровані(рейтингові) так і просто списки. Користувач може додати список в улюблені в такому разі він з’явиться у нього на сторінці. Кожен фільм у списку може мати коментар від автора списку. Фільми у списку повинні матиінтерфейс для додавання їх у список для переглядів пізніше.

2.1.12 Глобальний пошук

Додаток повинен мати пошук закріплений у шапці веб-сайту. Пошук повинен виконуватись скрізь усю інформацію в додатку. Користувач повинен мати можливість швидко знайти іншого користувача чи список фільмів за назвою. Пошук повинен мати фільтри за шуканими категоріями.

2.1.13 Огляди фільмів

Кожен авторизований користувач повинен мати можливість залишити огляд до фільму. Огляд повинен пройти автоматичну оцінку настрою, та отримати відповідний фон, якщо настрій огляду поганий – фон червоний, якщо добрий – зелений. Користувачі повинні мати можливість оцінити огляд, залишив відгук у вигляді позитивної, або негативної оцінки відповідного огляду. Усі огляди до конкретного фільму повинні відображатися на сторінці цього фільму.

2.1.14 Система рекомендацій фільмів

На основі активності користувача потрібно сформувати три фільми котрих він не бачив та запропонувати до перегляду на відповідній сторінці. На основі списку переглянутих фільмів, публікацій, оглядів, списків фільмів та ін. додаток повинен сформувати три унікальні фільми, котрі підходять користувачу.

2.1.15 Особисті повідомлення

Кожен користувач може відправити особисте повідомлення іншому користувачу. Повідомлення повинні бути структуровані за діалогами. Користувач може видалити, або відредагувати своє повідомлення. Повідомлення повинні працювати у реальному часі, тобто без оновлення сторінки повинні

з’являтися нові повідомлення. Користувач повинен мати можливість відправляти зображення. Поточний користувач повинен бачити всі свої діалоги, та мати можливість вільно виконувати навігацію між ними.

2.1.16 Швидкісний пошук фільмів

Для того щоб користувач не витрачав час на очікування результатів пошуку фільмів треба створити окремий ресурс з фільмами, котрий надвав би високу швидкість відповіді та надавав можливість виконувати запити з фільтрами.

2.2 Вимоги до інтерфейсу

Перш за все інтерфейс додатку повинен бути доступним. Доступність в дизайні дозволяє користувачам різноманітних можливостей орієнтуватися, розуміти та використовувати інтерфейс додатку. Основні принципи котрі повинен зберігати інтерфейс додатку: чистотат, стійкість, специфічність. Розглянемо детальніше кожен з них. Чисті інтерфейси допомогають користувачам орієнтуватися, створюючи чіткі інтерфейси з чіткими закликами до дії ми робимо прозорим взаємодію користувача та додатку. Стійкість інтерфейсу заключається в тому, що додаток повиен відповідати великій кількості користувачів однаково. Специфічність заключається в підтримці допоміжних технології, характерних для платформи, наприклад підтримка способів внесення інформації за допомогою дотику, клавіатури та миші.

Навігація на платформі поивнна бути ієрархічною. Коли навігація проста та ієрархічна, користувачі розуміють, де вони знаходяться у додатку. Навігація повинна мати чіткі потоки переміщень з мінімальними кроками з легким розташуванням елементів керування та чітким маркування. Контроль

фокусування або можливість керування фокусом на клавіатурі для навігації можуть бути реалізовані для часто використовуваних елементів.

Щоб підкреслити, яка інформація важлива, інтерфейс повинен мати чіткі візуальні та текстових сигнали, такі як колір, форма, текст та рух, ці сигнали додають чіткість. Кожна додана кнопка, зображення та рядок тексту збільшують складність інтерфейсу користувача. Користувач повинен легко розуміти інтерфейсу додатку, для цього потрібно використовувати:

— чіткі, видимі елементи;

— достатня контрастність та розмір елементів;

— чітка ієрархія важливості;

— ключову інформацію, повинна бути зрозуміла з першого погляду.

Колір та контраст можна використовувати, щоб допомогти користувачам бачити та інтерпретувати вміст програми, взаємодіяти з потрібними елементами та розуміти дії. Більш доступний колір. Колір може допомогти спілкуватися з настроєм, тоном та критичною інформацією. Первинні, вторинні та акцентні кольори можна вибрати для підтримки зручності використання. Достатній контраст кольорів між елементами може допомогти користувачам зі слабким зором бачити та використовувати додаток.

Гнучкі та чутливі до розміру екрану інтерфейси допомагають масштабувати вміст залежно від розміру екрана. Це допомагає уникнути обрізання вмісту, який може не відображатися повністю на певних типах пристроїв або відображатися по різному. Пов'язані елементи можна групувати поблизу один до одного для поліпшення читабельності.

Для покращення читабельності користувачі можуть збільшити розмір шрифту. Мобільні пристрої та браузери містять функції, які дозволяють користувачам коригувати розмір шрифту у всій системі. Додаток повинен в повній мірі відображати весь контент навіть при збільшенні розміру шрифту, також потрібно перевірити чи є достатньо місця для великих шрифтів на іноземних мовах.

Висновки до розділу 2

В даному розділу було розглянуто основні вимоги до функціоналу та інтерфейсу. Вимоги до функціоналу описують основні можливості додатку, описаний функціонал передбачає можливість розширення при розробці, але на основі цих вимог повинна базуватися основна частина функціоналу та напрям розробки. Для зменшення навантаження на основні частини додатку, а саме клієнт та сервер було приянете рішення використовувати хмарні сервіси для делугвання виконання задач. На разі формування вимог до функціоналу було вирішено використовувати хмарні сервіси для: авторизації, реєстрації та відновленню пароля, файлове сховище для зберігання зображеннь, котрі завантажують користувачі, та медіа файлів котрі енобхідні для роботи додатку, сервіс для збереження бази фільмів з можливістю пагінації даних та складних фільтрів.

Крім описаних вимог до інтерфейсу повинні виконуватися усі основні вимоги, котрі являються стандартом при розробці сучасних інтерфейсів. Важливо наголосити що інтерфейс повинен бути простим, зрозумілим та доступним, задля легкого занурення новими користувачами у проект.

3 РЕАЛІЗАЦІЯ

3.1 Архітектура проекту

Загальна архітектура проекту зображена на діаграмі ІА62.280БАК.006 Д4. Розглянемо детально складові архітектури додатку та їх функціональне призначення.

Додаток використовує в основі клієнт–серверну архутектуру. Ця ахітектрура надає можливість багатьом клієнтським додаткам відправляти запити та отримують відповіді від сервера. Клієнтські комп’ютери надають інтерфейс, що дозволяє користувачеві комп'ютера запитувати послуги сервера та відображати результати, які сервер повертає. Сервери чекають надходження запитів від клієнтів, а потім відповідають на них. В ідеалі, сервер забезпечує стандартизований прозорий інтерфейс для клієнтів, щоб клієнти не були в курсі специфіки системи (тобто апаратного та програмного забезпечення), що надає послугу. Ця архітектура особливо ефективна, коли клієнти та сервер мають різні завдання, які вони регулярно виконують. Наприклад, в нашому додатку клієнтська частина відповідає за рендер інтерфейсу та перетворення клієнтський запитів у HTTP/HTTPS запити. Багато клієнтів можуть отримати доступ до інформації сервера одночасно, і в той же час клієнтський комп'ютер може виконувати інші завдання, наприклад, надсилання електронної пошти.

Для збільшення можливостей додатку було прийнято рішення використовувати додаткові хмарні сервіси для делігування деяких задач. Для відправки повідомлень було використано Firebase Cloud Messaging [1].

Firebase Cloud Messaging (FCM) - це рішення для платформ обміну повідомленнями між платформами, яке дозволяє надійно надсилати повідомлення без будь-яких витрат. Використовуючи FCM, в додатку стало реально повідомити клієнтську програму про те, що нова електронна пошта чи інші дані доступні для синхронізації. Ви можете надсилати повідомлення-сповіщення для повторного залучення та утримання користувачів.

Для зберігання файлів було прийнято рішення використовувати Amazone S3. Amazon Simple Storage Service (Amazon S3) [2] - це сервіс зберігання об'єктів, який дає можливість масштабування, доступність даних, безпеку та продуктивність. Це означає, що ми можемо використовувати його для зберігання та захисту будь-якого обсягу даних для різних випадків використання, таких як завантаження зображень, резервне копіювання та відновлення,. Amazon S3 надає прості у використанні функції управління, завдяки яким ви могли упорядкувати свої дані та налаштувати контролі доступу.

Для зберігання бази фільмів ми використали ElasticSearch [3]. Elasticsearch - это распределенный поисковый и аналитический RESTful движок, способный работать с растущим числом вариантов использования. Являясь сердцем Elastic Stack, он централизованно хранит данные для молниеносного поиска, точной настройки релевантности и мощной аналитики, которая легко масштабируется. Это позволило улучшить скорость поиска фильмов.

В якості основної бази даних було вибрано PostgreSQL [4]. PostgreSQL - одна з найстаріших, але найбільш досконалих реляційних систем управління базами даних з відкритим вихідним кодом. Вона управляється і підтримується активним і незалежним спільнотою і активно забезпечує підтримку і допомагає вирішувати проблеми при роботі. Вона може працювати на різних платформах, таких як Windows, Linux і Mac.

3.2 Клієнтська частина додатку

Клієнтська частина – SPA побудована на базі React [5]. Клієнстький додаток був створений за допомогою create-react-app інструменту. Create React App [6] - це офіційно підтримуваний спосіб створення односторінкових програм React. Він пропонує сучасну настройку збірки без конфігурації. Через використання цього інструменту не потрібно було встановлювати чи налаштовувати такі інструменти, як webpack чи Babel. Вони заздалегідь

налаштовані та приховані, щоб можно було зосередитись на коді. Інструмент надає можливість створити новий проект використовуючи заготовлений темплейт. Так як ми обрали TypeScript [7] для розробки додатку при створенні преокту ми використали наступну команду (рис. 3.1).

```
aatry@AlexeyTryapko MINGW64 /d/Projects/bsa-2019-popcorn/server (master)
$ npx create-react-app client --template typescript
```

Рисунок 3.1 - Створення клієнтського додатку

Після виконання команди у директорії з'являється повніістю сконфігурований стартовий шаблон додатку, готовий для запуску та подальшого розширення.

Для покращення якості коду та уникнення помилок було сформовано конфігураційний файл для TypeScript, котрий налічуває збірку правил котрі повинен виконувати код. Це допомогло уникнути багатбох помилок на етапі збірки проекту.

React зробив легким створення інтерактивних інтерфейсів користувача. Основна ідея при використанні цієї бібіліотеки – побудова якнайбільш простих компонентів без власного стану це дозволяє ефективніше працювати та перевикористовувати компоненти. React ефективно оновлює та змінює лише потрібні компоненти, коли дані зміняться. Декларативні компоненти роблять код більш передбачуваним та простішим налагодження. Бібліотека несе в собі компонентний підхід, це означає що більшість компонентів були розроблені для перевикористання, це допомогло зпростити питання підтримки та відлагоджування коду.

Параметри необхідні компонентам було описані за допомогою інтерфейсів. Такий підхід дозволив зменшити кількість помилок та покращити розуміння компонентів (рис. 3.2).

```
interface IProps {
  match: {
    params: {
      id: string;
    };
  };
  fetchMessages: (userId: string, chatId: string) => void;
  chat: any;
  userId: string;
  isLoadingMessages: boolean;
}
```

Рисунок 3.2 - Приклад інтерфейсу параметрів компоненту

Більшість компонентів зі станом була написано в функціональному стилі використовуючи React Hooks, це дозволило збільшити ефективність компонентів та зменшити темплейт компонентів.

3.2.1 Обробка стану додатку

При розробці клієнтського додатку для обробки стану додатку було прийнято рішення використовувати бібліотеку Redux [8]. State, термін від React, - це об'єкт, який містить дані, що містяться в компоненті. Він визначає, як компонент поводить себе та відтворює інтерфейс. State є центральним компонентом створення динамічних сторінок за допомогою виконання певних умов. Простий спосіб зрозуміти цю концепцію - зрозуміти інтерфейс користувача як функцію стану, тобто розробник може змінити зовнішній вигляд веб-програми залежно від даних, що зберігаються у State.

Управління станом - це управління станом декількох елементів керування або компонентів інтерфейсу користувача. Оскільки я працював над великим додатком, складність обробки стану програми збільшувалась, було прийнято рішення використовувати зовнішні інструменти для кращого керування станом програми. Щоб полегшити управління станом, було використано бібліотеку для управління станом, це дозволило створювати модель стану додатків, оновлювати

стан компонентів, контролювати та спостерігати за змінами стану та читати значення стану.

Оскільки стан певних об'єктів може впливати на інтерфейс у різних частинах додатку, наявність єдиного постачальника стану – Redux, дозволяє тримати данні консистентними у всьому додатку. На приклад, користувач вирішив змінити ім'я в аккаунті, після зміни був відправлений запит на сервер та отримана успішна відповідь, у цьому разі треба відновити данні про ім'я користувача у всьому додатку для зберігання даних оновленими. Інший приклад – в розроблюваній соціальній мережі багато пов'язано з фільмами, це означає що запити на списки фільмів будуть робитися доволі часто, для того щоб не підгружати данні, котрі вже завантажені можна використовувати Redux State як єдиний постачальний інформації, та оновлювати його при необхідності. Виділимо позитивні сторони використання такої бібліотеки:

— стан зберігається разом в одному місці, яке називається "store". Хоча вам не потрібно зберігати всі змінні стану у "store", особливо важливо, коли данні із "store" використовуються кількома компонентами або більш складними архітектрурними рішеннями. У міру того, як програма збільшується, визначити джерело даних стану може бути складніше, тому "store" корисний, як єдине джерело правди. Данні зі "store" можна легко отримати в будь-якому компоненті;

— redux - це передбачуваний контейнер стану для додатків Оскільки "reducers" – функції для зміни стану додатку це чисті функції, той самий результат завжди буде вироблятися при зміні стану та дії. Крім того, фрагменти стану визначені нами з самого початку, що робить потік даних більш передбачуваним;

— redux надає сувору структуру для управління кодом та станом, що робить архітектуру легкою для копіювання та масштабування для тих, хто має попередній досвід роботи з Redux.

Розглянемо основні концепти роботи з Redux. Для управлінням станом додатку Redux використовує actions, action creators, reducers, та stores. Ці концепції використовуються для створення простої архітектури управління станом (рис. 3.3).

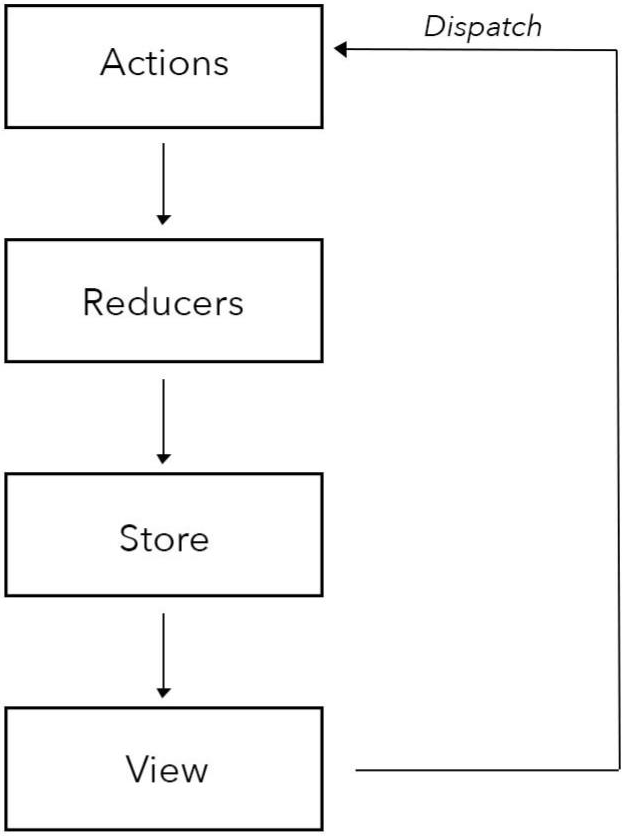

Рисунок 3.3 - Архітектура управління станом

За замовчуванням використовуючи Redux не можна відправляти асинхронні запити до серверу. Для вирішення цього завдання було використано бібліотеку redux-saga. це бібліотека, яка має на меті зробити можливим виконувати асинхронні запити в додатку такі як запити до серверу, чи отримання кешу додатку. Ця бібліотек простіше в управлінні та тестуванні за конкурентні це надає можливість швидшого виявлення та усунення несправностей в обробці запитів. Основна ідея цієї бібліотек полягає в тому, що сага - це як окрема нитка у вашій програмі, яка відповідає виключно за побічні ефекти. redux-saga - це

програмне забезпечення надає можливість змінювати стан додатку, можна запускати, призупиняти та скасовувати запити з основної програми за допомогою звичайних дій redux. Для запитів данна бібліотека використовує функцію ES6 під назвою генератори для полегшення читання, запису та тестування цих асинхронних потоків. Асинхронні запити виглядають як стандартний синхронний код JavaScript.

Разом з redux-saga було прийнято рішення використовувати паттерн для reducer під назвою redux-saga-routine. Цей паттерн описує етапи виконання запитів у наступній послідовності: TRIGGER, REQUEST, SUCCESS або FAILURE та FULFILL. Відповідно на кожен з цих станів при запиті описуються функція в reducer, котра змінює стан додатку. Розглянемо детальніше кожен стан. При виклику action, викликається TRIGGER, після створення та відправки запиту викликається REQUEST, при успішному виконанні запиту виконується SUCCESS, а при виникненні помилки FAILURE, завершуючий стан FULFILL. Ця бібліотека дозволяє уникнути дублювання коду при описуванні станів запитів, та саме головне цей патерн дозволяє сповіщувати клієнта про статус його запиту. Наприклад, якщо запит виконується показувати спінер, чи якщо виникла помилка показувати відповідне повідомлення. В Додатку А зображен reducer, котрий займається обробкою завантаження користувача.

Додаток використовує динамічні store, тобто одночасно додаток не зберігає данні всіх модулей додатку, а лише найнеобхіжніші данні та данні необхідні для активної частини додатку. Коли користувач заходить на сторінку з функціоналом котрий має окремий store цей модуль ініціалізується та заповнюється данним, відповідно після того як користувач виходить з певної сторінки данні видаляються. Цей підхід дозволяє покращити швидкість роботи додатку та зменшити кількість запитів при старті проекту.

3.2.2 Навігація додатку

Так як React не має інструментів для реалізації роутінгу було використано сторонню бібліотеку react-router. Компоненти - це серце потужної декларативної моделі програмування React. React Router [9] - це сукупність навігаційних компонентів, які декларативно складаються разом із програмою.

На структурі проекту, котра зображена в ІА62.280БАК.006 Д4 можна побачити основні навігаційні шляхи в додатку. Основна частина додатку потребує авторизації, та перенаправлення на сторінки авторизації, якшо неавторизований користувач намагається отримати доступ до данних. Завдяки компонентам впровадженим цією бібліотекую я реалізував список роутів доступних неавторизованим користувачам, та перенаправлення на сторінки авторизації (рис. 3.4).

Коли користувач пробує перейти на говону сторінку виконується перевірка наяності токену авторизації, та при його відсутності за допомогою компонента Redirect впровадженого бібліотекою react-router користувач перенаправляється на сторінку авторизації.

```
<Switch>
  <Route
    exact
    path="/login"
    component={() => (
      <Login
        loginError={loginError}
        isAuthorized={isAuthorized}
        onSubmit={authorize}
        authWithSocial={authWithSocial}
      />
    )}
  />
  <Route
    exact
    path="/registration"
    component={() => (
      <Registration
        registerError={registerError}
        isAuthorized={isAuthorized}
        registration={registration}
      />
    )}
  />
  <Route path="/" component={Main} />
  {/* Not found route */}
  <Route path="*" exact component={NotFound} />
</Switch>
```

Рисунок 3.4 - Приклад навігації в додатку

Також було реалізовано сторінку котра відображається коли користувач хоче перейти на сторінку шлях котрої не був описаний. Тобто якщо користувач хоче отримати данні за неіснуючим шляхом його буде перенаправлено на спецальну сторінку з навігацією на основну сторінку.

Роутер дозволяє передавати параметри при переході на іншу сторінку та вказувати query параметри у посиланняї. Використовуючи ці можливості було реалізовано навігацію на сторінки специфічних матеріалів, тобто конкретного фільму чи опитування.

3.2.3 Відловлювання помилок

Помилка JavaScript у частині інтерфейсу не повинна зламати всю програму. Щоб вирішити цю проблему для користувачів React, React вводить нове поняття "error boundary". Error boundary - це компоненти React, які

вловлюють помилки JavaScript в будь-якому місці їхнього дочірнього дерева компонентів, записують ці помилки та відображають резервний інтерфейс користувача замість дерева компонентів, в яких виникнули помилки. Error boundary уловлюють помилки під час візуалізації, у методах життєвого циклу та в конструкторах цілого дерева під ними.

Для відновлення усіх помилок у додатку у корневий елемент я додав компонент котрий налічує метод життєвого циклу componentDidCatch при виникненні помилки у дочірніх компонентів спрацьовує цей метод та в залежності від змінного оточення буде відображатися або стек викликів котрий спровакував помилку, або інтерфейс котрий сповіщує користувача о деякій помилці. Тобто якщо додаток має оточення розробки, користувач бачить інформацію про помилку, а при продакшн оточенні користувач бачить інтерфейс кажучий о помлці без детальної інформації.

3.2.4 Зв’язок із сервером

Для отримання данних від серверної частини використовується HTTP запити, Запити виконуються лише через actions впроваджені react-saga, завдяки цьому запити можна легко відлагоджувати та знаходити помилки.

Додаток має низку модулів які потребують двостороннього зв’язку з сервером. Для особистих повідомлень, коментарів, публікацій важливо мати актуальні данні, тобто при зміні данних одразу відображати нові. Це питання вирішелось завдяки web-sockets. Клієнтський додаток має з’єднання з сервером так коли сервер відправляє подію про нове повідомлення чи коментар клієнтський додаток робить відповідний HTTP запит для отримання нових данних. Такий підхід дозволяє відновлювати данні у реальному часі.

Клієнт отримує повідомлення на основі кімнат в котрих знаходиться його з’єднання, це дозволяє дуже наглядно сповіщувати усіх членів кімнати про нову подію. За допомогою використання кімнат дуже просто реалізувати

налаштування повідомлень, тобто при виконанні відповідних умов користувач потрапляє у кімнату та отпримує повідомлення, чи навпаки.

3.2.5 Структура проекту

Клієнтський додаток має архітектуру засновану на поєданні компоентів та файлів за доменною областю. Це означає, що кожен компонет котрий відноситься до відповідного модуля занходиться в одній директорії. Також усі сервіси та файли котрі відносяться до одного модуля згнаходяться теж в одній директорії. Винятком є компоненти та сервіси котрі виликаються не лише в одній, специфічній доменній області, такі файли розміщуються в спільній директорії.

Розглянемо струтуру проекту зображену на рисунку 3.5. Спочатку розглянемо наступні файли tsconfig.json, prettier.config.js, .prettierrc це конфігураційні файли, котрі не несуть ніякої цінності коли проект вже збудовано, але при розробці дозволяють уникнути багатьох проблем. Файл index.tsx – точка входу додатку, у цьому файлі ініціалізується сервіси Firebase та основна частина додатку. App.tsx має підключення роутера, Error boundary та redux store, цей файл відповідає за підключення основних допоміжних сервісів та редірект на сторінку роутінгу. Директорія styles налічує стилі, котрі обїявлені глобально. Директорія services потрібна для загальних сервісів, котрі використовуються в багатьох місцях, сервіс в свою чергу налічує функції які реалізують запити на сервер.

Директорія redux налічує файли для конфігурації Redux store, відповідно для додавання нових redux-saga, reducers, stores та routines потрібно розширювати файли в цій директорії.

Директорія helpers відповідає за синхронні функції котрі використовуються в багатьох місцях додатку. На відміну від сервісів helpers повинні бути синхронними та не виконувати запитів на сервер.

Директорія containers налічує файли з роутінгом, саме в цих деректоріях виконується основна навігація скрізь додаток. Директорія налічує дві дочірні директорії: Routing – описує публічні роути, Main – описує основні шляхи навігації.

Директорія components налічує низку піддиректорій розбитих за модулями – доменними областями. Кожна доменна область має свої сторінки та компоненти. Витримувалась ідея коли тільки сторінки мають свій стан, а компоненти повинні бути без свого стану, та не мати доступ до загального store додатку, тобто працювати лише з власними параметрами.

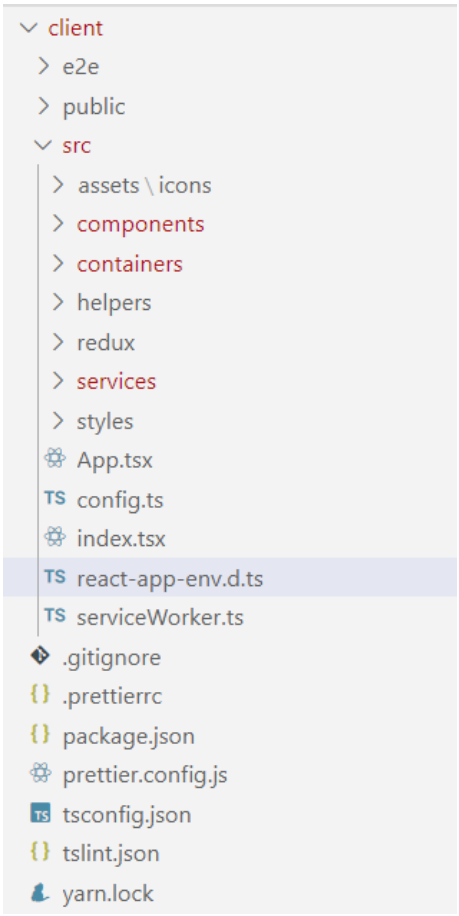

Рисунок 3.5 - Стуктура клієнтської частини

Розглянемо детальніше одну з доменних областей додатку. На прикладі доменної області чатів зображених на рисунку 3.6 я розповім про окремі доменні області.

Кожна доменна область має точку входу, котра регулює подальшу поведінку додатку у обраному модулі. У випадку з особистими повідомленнями це ChatPage.tsx. При необхідності можуть існувати стилі як ChatPage.scss, але важливо зауважити, що стилі повинні бути ізольовані та не створювати не впливати на стилі із інших доменних областей. Для обробки стану відповідного модуля доменна область повинна мати власний store при необхідності.

Важливо зауважити, що точки входу в доменні області повинні асинхронно завантажуватись, а не в початковому бандлі. Такий підхід щменшує час очікування користувачем відгуку від додатку.

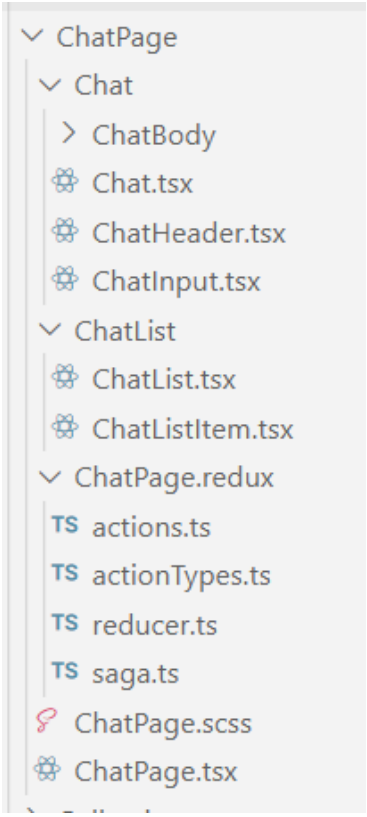

Рисунок 3.6 - Приклад окремої доменної області

3.3 Сервер

Серверна частина представляє собою RESTfull API Express [10] та socket сервер. REST - це абревіатура для Representational State Transfer. Це

архітектурний стиль для розподілених гіпермедіа-систем і вперше був представлений Роєм Філдінгом у 2000 році у своїй знаменитій дисертації. Як і будь-який інший архітектурний стиль, REST також має свої 6 керівних обмежень, які повинні бути задоволені, для того щоб інтерфейс можна було називати RESTful. Ці принципи перераховані нижче.

— клієнт-сервер - відокремлюючи проблеми користувальницького інтерфейсу від проблем зберігання даних, ми покращуємо портативність користувальницького інтерфейсу на кількох платформах та покращуємо масштабованість, спрощуючи серверні компоненти;
— без стану - кожен запит від клієнта до сервера повинен містити всю інформацію, необхідну для розуміння запиту, і не може скористатися будь-яким збереженим контекстом на сервері. Тому стан сесії повністю зберігається на клієнті;
— кешованість - обмеження кешу вимагають, щоб дані у відповіді на запит неявно або явно позначалися як кешовані або не кешовані. Якщо відповідь є кешованою, то клієнтський кеш має право повторно використовувати ці дані відповіді для наступних, рівнозначних запитів;
— уніфікований інтерфейс - застосовуючи принцип спільності програмного забезпечення на компонентному інтерфейсі, спрощується загальна архітектура системи та покращується видимість взаємодій; щоб отримати єдиний інтерфейс, для керування поведінкою компонентів потрібно кілька архітектурних обмежень. REST визначається чотирма обмеженнями інтерфейсу: ідентифікація ресурсів; маніпулювання ресурсами через представництва; повідомлення з самоописанням; і, гіпермедіа як двигун стану застосування;
— багатошарова система - шаруватий стиль системи дозволяє архітектурі складатися з ієрархічних шарів, обмежуючи поведінку компонентів

таким чином, щоб кожен компонент не міг «бачити» поза безпосереднім шаром, з яким вони взаємодіють.

3.3.1 Структура серверу

Серверна частина додатку розита на шари. Розглянемо за що відопвідає кожен шар, та як в цілому реалізован сервер. На рисунку 3.7 зображено структуру серверної частини додатку.

index.ts – вхідна частина додатку, у цьому файлі цініціалізуються усі middleware, підключаються роути, встановлюється зв’язок з базою даних та запускається сервер.

Розглянемо окремі директорії. Почнемо з config, ця директорія була створена для того щоб зібрати основні конфігурації для сервісів в одному місці. Конфігурація серверної частини додатку залежить від змінного оточення, для легкої роботи з змінними оточенням було використано бібліотеку dotenv.

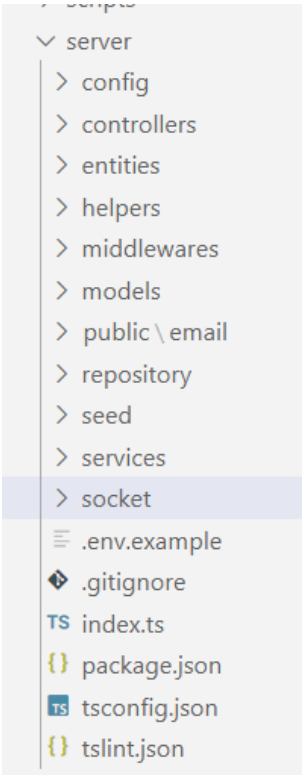

Рисунок 3.7 - Структура серверу

Розглянемо окремі конфігураційні файли та їх призначення. Першим розглянемо facebook.config.ts, цей файл відповідальний за експорт конфігураційних змінних для підключення авторизації через фейсбук. Наступні файли: firebase.config.ts, google.config.ts, imgur.config.ts виконують таку саму функцію вони відповідні лише за експорт змінних для підключення відповідних сервісів.

jwt.config.ts – конфігураційний файл котрий експортує змінні для конфігурації jwt токену за допомогою котрого виконується авторизація в додатку. Файл має наступні конфігурації: серкретний ключ, термін дії токену. Секретний ключ береться з змінного оточення та потрібен для створення токену, та його розшифровки. Термін дії токену описує час впродовж якого токен є дійсним, я визначив що токен після створення дійсний 24 години. Після того як токен перестає бути дійсним клієнтська частина відправляє запит на відновлення токену використовуючи для цього токен оновлення, котрий має довший тремін дії, а саме 6 місяців. У разі коли токен відновлення є недійсним користувач повинен знову авторизуватися в системі для оновлення обох токенів.

В додатку Б зображено файл з конфігурацію підключення до бази даних. Якщо проект в продакшині, створюється посилання на базу даних за допомогою змінних взятих з оточення. Коли додаток знаходиться в розробницькому оточенні послиання на підключення бази даних береться зі змінного оточення.

Наступний конфігураційний файл - passport.config.ts в цьому файлі створюються стратегії авторизації. Реалізовані наступні стратегії: авторизація, реєстрація, відновлення токену, перевірка валідності токену, авторизація та реєстрація за допомогою гугл, авторизація та реєстрація за допомогою фейсбук. Кожна стратегія виконує свою логіку та перевіряє валідність переданого токену чи даних.

routes-white-list.config.ts – останній конфігураційний файл. В цьому файлі визнача.ться роути котрі не потребують авторизації користувача. Це роути запитів пов’язаних з авторизацією та відновленням паролю.

Наступна директорія для розгляду - middlewares. З назви зрозуміло що ця директорія відповідає за мідлвари. Мідлвари – це ланцюх фунуцій котрі мають настпну сигнатуру - req, res, next. В req потрапляє запит до серверу, в аргументі res формується відповідь серверу, мідлвар може надіслати відповідь клієнту наприклад якщо авторизація прошла не успішно, або виникла помилка. Для переходу до наступного мідлвару викликається колбек next.

Розглянемо які мідлвари реалізовані в додатку. В файлі authentication.middleware.ts реалізована логіка обробки запиту на авторизацію користувача. Коли користувач відправляє запити за наступним шляхом /api/auth/login викликаєть цей мідлвар, в котрому в свою чергу викликається стратегія для авторизації користувача, котра була описана раніше.

Наступний мідлвар - authorization.middleware виконується кожен раз коли на сервер потрапляє запит. В цьому мідлварі перевіряється чи знаходиться роут по якому проводиться запит у списку роутів котрі не потребують авторизації, якщо роут знаходиться в цьому спику – викликається наступний роут, в іншому випадку викликається стратегія перевірки валідності та актуальності авторизаційного токену за допомогою мідлвару jwt.middleware, котрий в свою чергу викликає jwt стратегію.

Мідлвари google.middleware та facebook.middleware реадізовують схожий функціонал з різницею в сервісі. Ці мідлвари потрібні для реалізації авторизації на відповідних сервісах. Розглянемо детальніше на прикладі google.middleware в цьому файлі є два мідлвари googleMiddleware та googleCallbackMiddleware. Перший виконується при запиті на роут /api/auth/google він викликає стратегію авторизаціїї google, та описує скоупи необхідних данних від сервісу. Настпний мідлвар обробляє колбек сервісу автооризації після завершення авторизації на стороні сервісу. При виникненні помилок авторизації описується шлях для редіректу при помилці, в данному випадку було зазначено сторінку авторизації.

registration.middleware – цей мідлвар відповідно назві виконується при реєстраціїї користувача. Коли користувач відправляє запит /api/auth/register викликається цей мідлвар, котрий в свою чергу викликає стратегію для реєстрації.

Одним з найважливіших мідлварів є error-handler.middleware. Сигнатура цього мідлвару відрізняється від інших. Першим аргументом цей мідлвар приймає помилку, а потім req, res, next. Помилка – об'єкт з настпними полями: status та message, за замовчуванням поля мають значення 500 та пуста строка відповідно. Якщо помилка не була передана в аргументах чи має не істинне значення викликається наступний мідлвар, в протилежному стані сервер відправляє відповідь з статусом помилки та повідомленням.

Для подальшого розгляду ахітектури та розуміння реалізації треба розглянути директорію models. Ця директорія зберігає в собі класи усіх сутностей бізнес моделі. Ці класи використовуються в усій серверній частині для специфікації типів для змінних та аргументів. Взаємозв'язок цих класів зображено на ІА62.280БАК.004 Д2.

Розгляемо директорію controllers вона відповідає за декларування усіх API ендпоінтів серверу. Ця директорія налічує головний файл - root.controller, в цьому файлі реєструються всі контролери та описуються ендпоінти. На рисунку 3.8 зображено приклад реєстрації контролерів.

```
import authRoutes from "./auth.controller";
import imageRouters from "./image.controller";
import votingRoutes from "./voting.controller";

// register all routes
export default app => {
  app.use("/api/auth", authRoutes);
  app.use("/api/voting", votingRoutes);
  app.use("/api/image", imageRouters);
};
```

Рисунок 3.8 – Приклад реєстрації контролерів

Розглянемо реалізацію ендпоінту на рисунку 3.9 зображено фрагмент реалізації ендпоінтів, що відповідають за роботу с довгостроковими публікаціями. router – екземпляр класу Router котрий впроваджує express. Задопомогою ланцюжкового синтаксису декларуються типи запитів котрі приймає сервер, перший аргумент методу приймає строку, котра декларує шлях до конкретного ендпоінту також у цій строці може бути описано параметри які приймає ендпоінт, наприклад ":id", далі метод приймає функцію – мідлвар. В цій функції я викликаю наступний шар серверу – сервіс. При вдачній виконанні запиту севісу відправляється відповідь на запит, та при невдачній процесс переходить до мідлвару обробки помилок. Ми можемо бачити на прикладі запиту на створення посту, що окрім http відповіді викликається метод бібліотеки socket.io, котрий сповіщує користувачів про те що створилася нова публікація. Всі роутери експортуються та реєструються в головному роутері, котрий в свою чергу реєструється в index.ts.

```
router
  .get("/", (req, res, next) =>
    postService
      .getPosts()
      .then((posts: any[]) => res.send(posts))
      .catch(next)
  )
  .get("/:id", (req, res, next) =>
    postService
      .getPostById(req.params.id)
      .then((post: Post) => res.send(post))
      .catch(next)
  )
  .post("/", (req: Request & { io: any }, res, next) =
    postService
      .createPost(req.body)
      .then(post => {
        res.send(post);
        req.io.emit("new-post", post);
      })
      .catch(next)
  );
```

Рисунок 3.9 – Приклад релізації конкретного ендпоінту

Наступний шар після контролерів - сервіси. Цей шар потрібен для реалізації основної бізнес логіки на сервері. Методи сервісу оброблюють

отриманні данні, форматують їх до потрібного вигляду після чого викликають методи інших сервісів, або репозиторієв – шар відповідний за записи у базу данних. Після виконання всіх операцій сервіс повинен повернути Promise , це зумовлено сигнатурою контролерів, інакше винекне помилка.

Розглянемо метод сервісу, котрий відповідає за відправку письма привітання після реєстрації. На рисунку 3.10 зображено функцію котра приймає в аргументах емейл користувача, після чого формує об’єкт для відправки пошти. Важливо зауважити для отримання емейлу відправника використовується щмінне оточення, тобто сервіс викликає не лише репозиторії, а й отримаж інші сторонні данні. Після формування необхідного об’єкта функція викликає метод emailRepository репозиторію для відправки повідомлення. Метод репозиторія повертає Promise, в свою чергу функція sendWelcomeEmail провертає результат виконання цієї функції.

```
export const sendWelcomeEmail = email => {
  const msg = {
    to: email,
    from: process.env.USER_MAIL,
    subject: "Congratulation with registration",
    html: emailView.welcome()
  };
  return emailRepository.send(msg);
};
```

Рисунок 3.10 – Метод сервісу відправки пошти

Наступний шар - репозиторії. Цей шар потрібен для виклику методів хмарний сервісів, чи наприклад методів ORM для роботи с базою данних. Тобто шар репозиторій регулює процеси читання та запису данних в додатку.

Усі репозиторії, котрі працюють с базою данних Postgresql наслідуються від класу Repository, котрий впроваджує ORM. Цей клас надає можливості використання усіх методів для роботи з базою данних без використання sql запитів. З одного боку, це дуже поліпшило та прискорило розробку, але такі запити є дуже не ефективні. Запити до бази данних за допомогою ORM

виконують велику кількість надлишкових команд, при підвищенні складності та специфічності потреб від ORM запити стають все більш не ефективними. Через те що ORM дозволила дуже прискорити роботу на початковому етапі, я використав її можливості.

Відповідно до сервісів при виникненні помилки в роботі функції репозиторія помилка оброблюється та викликається настпний мідлвар – обробник помилок.

Type ORM [11] дозволяє за допомогою звичайних класів typescript описати сутності для бази данних. На основі створених сутностей автоматично генеруються міграції для бази данних.

На рисунку 3.11 зображен прикад сутності списку улюблених фільмів. Окрім звичайно описаних полів класу, тут є описання моделі для бази данних. Розглянемо деякі з них PrimaryGeneratedColumn – головна колонка в таблиці з головним ключем. Ідентифікатор Column описує звичайну колонку. ManyToOne описує взаємозв'язок між фільмом у списку та користувачем, тобто у користувача може бути безліч фільмів у списку.

На основі таких сутностей та їх взаємозв'язків генеруються міграції, для оновлення бази данних. При підключенні серверу до бази виконується порівняння міграцій котрі були виконані та котрі наявні. У випадку коли є невиконані міграції база оновлюєтсья.

```
@Entity()
export class FavoriteList {
  @PrimaryGeneratedColumn("uuid")
  id: string;

  @Column()
  movieId: number;

  @ManyToOne(type => User, { onUpdate: "CASCADE", onDelete: "CASCADE" })
  user: User;
}
```

Рисунок 3.11 – Приклад сутності

Детальна база даних зображена у ІА62.280БАК.003 Д1. При розробці моделі бази данних були використані перші три рівня нормалізації. Для того щоб не вводити тестові данні кожного разу при розгортанні проекту були створені сіди – міграції для заповнення таблиць тестовими данними.

Сервер налічує не лише HTTP сервер, а й сокет сервер. Для сповіщення користувачів про появу чи оновлення данних використовуються сокет події. Завдяки такій техніці ми оновлюємо данні в реальному часі. Це використовуються для публікацій, повідомленнь та особистих повідомлень.

3.4 Сервіс для зберігання фільмів

Для зберігання фільмів було прийнято рішення виокристати ElasticSearch. Чому саме Elasticsearch через швидкість і масштабованість Elasticsearch та його здатність до індексації багатьох типів контенту означають, що він може бути використаний для ряду випадків використання:

— пошук фільмів;

— пошук з використанням скалдних фільтрів;

— моніторинг ефективності додатку;

— аналітика безпеки;

— бізнес аналітика.

Сирі дані надходять у Elasticsearch з різних джерел, включаючи журнали, системні показники та веб-додатки. Прийом даних - це процес, за допомогою якого ці необроблені дані аналізуються, нормалізуються та збагачуються до того, як вони будуть індексовані в Elasticsearch. Після індексації в Elasticsearch, користувачі можуть запускати складні запити щодо своїх даних та використовувати агрегації для отримання складних композицій своїх даних. Саме це відповідає потребам нашого додатку, адже саме завдяки швидкій обробці та композиції великої кількості данних додаток зможе мати велику кількість фільмів та додаткової інформації стосовно цих фільмів. Крім того

ElasticSearch дозволяє дуже легко працювати з пагінацією та всілякими фільтрами.

Висновки до розділу 3

У цьому розділі були детально розглянуті основні вузли роботи додатку. Основний огляд реалізації був зосереджений на сервері та клієнті. Ці вузли були побудовані з огляду на масштабування. На цей час додаток налічує дуже багато функціоналу, дуже важливо розширювати цей функціонал не збільшуючи складність проекту.

Рішення при обиранні архітектури для клієнтської частини були основані на цій цілі. Доменна модель архітектури робить окремі блоки функціоналу незалженими один від одного, це дозволяє не ускладнювати архітектуру додатку.

Основан задача серверної частини - представлення стійкого API для клієнтської частини. Я вважаю, шарувата архітектура – те що потрібно для уникнення глобальних проблем з сервером. Так як помилки відловлюються на кожному рівні та ці рівні незалежні один від одного серверна частина залишается стійкою.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Інструкція до авторизації, реєстрації

4.1.1 Авторизація

При відкритті застосунку не авторизований раніше користувач потрапляє на сторінку авторизації. Сторінка зображена у додатку Г.1. Для авторизації за допомогою емейлу та паролю користувач повинен заповнити відповідні поля. Після заповнення полів користувач може натиснути кнопку “Sign In” для авторизації.

Якщо користувач ввів дані котрі не відповідають правилам валідації він побачить надписи, котрі відповідають помилці валідації приклад зображено у додатку Г.2.

При не успішній авторизації користувач побачить сповіщення з причиною помилки авторизації. Приклад оповіщення представлений у додатку Г.3. Якщо користувач був успішно авторизован він буде перенаправлен на головну сторінку застосунку.

Користувач має можливість авторизуватися за допомогою Google чи Facebook. Для авторизації через Google потрібно натиснути кнопку “Continue with Google” (додаток Г.1), відповідно для авторизації через Facebook треба натиснути “Continue with Facebook” (додаток Г.1). Після натискання однієї з цих кнопок користувач буде перенаправлений на сторінку авторизації, котру впровадив он з цих сервісів. Після успішної авторизації користувач буде перенаправлений на головну сторінку застосунку.

4.1.2 Відновлення паролю

Для відновлення паролю користувач повинен натиснути на посилання “Forget password?” (додаток Г.1) після чого його буде перенаправлено на сторінку відновлення паролю представлену у додатку Г.4. Для створення запиту

на відновлення паролю потрібно ввести в поле емейл від відповідного аккаунту, та натиснути на кнопку “Reset” (додаток Г.4). При невалідних чи некоректних даних користувач побачить повідомлення, котре описує проблему так само, як у додатку Г.2 та Г.3. Для повернення на сторінку авторизації потрібно натиснути на посилання “Login” (додаток Г.4).

Після успішного запиту користувач отримує повідомлення на введену емейл адресу з посиланням на сторінку відновлення паролю (додаток. Г.5). На цій сторінці потрібно ввести пароль у відповідне поле та натиснути на кнопку “Restore”. Після успішного відновлення паролю користувач буде перенаправлений на головну сторінку застосунку.

Якщо пароль не відповідає правилам валідації користувач побачить відповідне повідомлення так як зображено у додатку Г.2.

З сторінки відновлення паролю користувач може повернутись на сторінку авторизації для цього потрібно натиснути кнопку “Login” (додаток Г.5).

4.1.3 Реєстрація

Для переходу на сторінку реєстрації потрібно натиснути на посилання “Register” (додаток Г.1).

Сторінка реєстрації зображена у додатку Г.6. Користувач має можливість зареєструватися на основі ім’я, емейлу та паролю для цього потрібно заповнити відповідні поля. Для того, щоб відправити запит на реєстрацію потрібно натиснути кнопку “Sing Up”. Після успішної реєстрації користувач буде перенаправлений на головну сторінку застосунку.

При виникненні помилок валідації чи реєстрації користувач побачить відповідні помилки так само як у додатках Г.1 та Г.2.

Користувач може зареєструватися використовуючи Google чи Facebook натиская відповідні кнопки “Continue with Google” та “Continue with Facebook”, Після натискання користувач перенаправляється на сторінку авторизації

відповідних сервісів. За умови успішної авторизації користувача буде перенаправлено на головну сторінку.

Для того, щоб повернутися на сторінку авторизації потрібно натиснути на посилання “Login”.

4.2 Головна сторінка

Після успішної авторизації користувач потрапляє на головну сторінку зображену у додатку Г.7. Головна сторінка відображає основні елементи навігації та інформацію сформовану на активності користувачів.

4.2.1 Історії

В верхній частині головної сторінки розташовані історії - тимчасові користувацькі публікації (додаток Г.8). Для того, щоб переміщувати потік історій користувачу потрібно зажати одну із історій та перетащити потік в бажану сторону.

Для того, щоб відкрити окрему історію потрібно натиснути на неї. Після натискання відкриється модальне вікно з обраною публікацією (додаток Г.9).

У модальному вікні (додаток Г.9) є декілька органів управління. З правої та лівої сторони є стрілки вліво та право, за допомогою котрих користувач може виконувати навігацію між історіями. В правому верхньому куті модального вікна є хрестик, при натисканні на цей крестик модальне вікно закриється.

Користувач може відправити свою реакцію на обрану історію. Для цього потрібно натиснути на поле для вводу тексту, після чого відкривається додаткова панель з смайлами (додаток Г.10).

Є можливість відправити текстову реакцію на історію. Для відправки такої реакції необхідно ввести текст у відповідне поле та натиснути на кнопку з

іконкою літака чи натиснути клавішу Enter. Після відправки реакції автор історії отримає відповідне персональне повідомлення (додаток Г.11).

Користувач також може відреагувати натиснув на один з смайликів у відкритій панелі (додаток Г.11). При натисканні на смайл автор історії теж отримає персональне повідомлення, але замість тексту відобразиться обраний смайл (додаток Г.12).

Для створення нової історії потрібно натиснути на перший блок зображений у додатку Г.8. Після натискання відкриється модальне вікно конструктора історій (додаток. Г.13).

Для того щоб створити історію потрібно натиснути на кнопку “Share”. Після чого нового історія у реальному часі з’явиться у інших користувачів.

У верхній частині конструктора історій (додаток Г.13) знаходиться блок інструментів за допомогою котрих користувач може створити свою унікальну історію.

Перший елементи в блоці інструментів відповідає за колір фону. При натисканні на нього відкривається палітра кольорів (додаток Г.14), яка дозволяє вибрати бажаний колір.

Другий блок відповідає за додавання тексту на історію. Натискаючи на цей блок в історії з’являється поле для вводу тексту (додаток Г.15). Текст можна перетягувати по історії. Для зміни кольору тексту потрібно натиснути на третій блок, після чого відкриється палітра кольорів.

Для того, щоб додати зображення користувачу потрібно натиснути на четвертий блок. Позицію завантаженого зображення можна змінювати за допомогою перетягування.

Користувач має можливість прикріпити додаткові матеріали до історії натиснувши на останній блок. Після натискання відкриється інтефейс додаткових матеріалів (додаток Г.16). На інтерфейсі зображено список всіх додаткових матеріалів. При натисканні на окремий з них користувач буде перенаправлений на сторінку вибору, чи створення додаткового матеріалу. Усі

інтерфейси додаткових матеріалів крім голосування передбачають вибір із існуючих матеріалів, або перехід до створення певного матеріалу.

Якщо користувач натискає на кнопку “Add survey” він перенаправляється до інтерфейсу вибору опитування (додаток Г.17). Для додавання опитування потрібно натиснути на необхідне в списку. Якщо необхідного опитування не існує користувач може створити його, натиснув на кнопку “Create survey” користувач буде перенаправлений до сторінки створення нових опитування. За допомогою ідентичного інтерфейсу користувач може додати топи та події. Інтерфейс додавання голосування зображено у додатку Г.18.

Для створення нового голосування користувачу потрібно заповнити поле питання та варіантів відповідей. Для того, щоб додати варіант відповіді потрібно натиснути на кнопку “Add Option”. Після завершення потрібно натиснути на кнопку “SHOW PREVIEW”.

Голосування відображається у створеній історії (додаток Г.19), для того щоб проголосувати користувач повинен натиснути на відповідну частину блоку голосування.

Для додавання фільму користувач повинен знайти необхідний фільм, для цього в полі пошуку (додаток Г.20) потрібно ввести назву фільму, або частину від назви та вибрати із запропонованих. Обраний фільм з’явиться в інтерфейсі, при натисканні прапорця фільм буде доданий до вашого списку для перегляду. При натисканні на фільм він додається до історії та його постер заповнює фон.

Додаткові матеріали такі як опитування, події та топи відображаються як блок з заголовком (додаток Г.21) при натисканні на котрий користувач перенаправляється на сторінку відповідного матеріалу.

4.2.2 Пости

У центральній частині головної сторінки знаходиться потік з постами (рис. 5.8). Пости - довгострокові публікації (додаток Г.22).

Користувач має можливість залишити свою реакцію відносно коментарів, для цього потрібно навести курсор на смайл під описом публікації (додаток Г.22). Після наведення з’явиться панель з смайлами на вибір (додаток Г.23). Для того, щоб залишити реакцію потрібно натиснути на бажаний смайл.

Під постом існують коментарі (додаток Г.22). Для того щоб створити новий коментар треба заповнити поле для вводу коментаря та натиснути на кнопку відправлення, або на клавішу Enter.

Пост створений користувачем має панель для керування постом (додаток г.24). Для відкриття панелі потрібно натиснути на іконку, котра має вигляд трьох крапок. Відкрита панель має опції: “Edit” - відкриває інтерфейс редагування поста, “Delete” - видаляє пост.

Для стоврення нового поста необхідно натиснути на кнопку “Add post” (додаток 5.7). У відкритому модальному вікні зображеному у додатку Г.25 зображені основні інструменти для створення нової публікації. Для створення публікації потрібно натиснути на кнопку “Add Post”. Користувач може закрити модальне вікно натиском на хрестик.

Для створення публікації достатньо заповнити поле описання та натиснути кнопку “Add Post”. Така публікація матиме наступний вигляд (додаток Г.26).

До публікації можливо прикріпити додаткові матеріали. Користувач має можливість прикріпити опитування натиснув на кнопку “Survey” (додаток Г.25). Відкриється інтерфейс вибору опитування (додаток Г.27). Якщо бажане опитування вже створено користувач може вибрати зі списку, чи натиснути кнопку “Create survey” для створення нового.

Після вибору опитування користувач повернеться до конструктору постів. Потрібно відкадрувати зображення прикріплене до опитування за допомогою кропера (додаток Г.28). Перетягуючи рамку користувач може змінювати розміри зображення. Якщо формат зображення відповідає побажанням користувача потрібно натиснути на іконку підтвердження, якщо натиснути на іконку відміни зображення не буде прикріплено до публікації.

Для видалення додаткового матеріалу потрібно натиснути на хрестик, котрий відноситься до певного матеріалу.

Створений пост з прикріпленим опитуванням зображений у додатку Г.29. Якщо користувач натисне на прикріплене опитування - його буде перенаправлено на сторінку проходження цього опитування.

Додавання топів та подій мають такий самий алгоритм, як і опитування. Основна різниця - превью. Превью топів складається з назви та декількох трьох перших фільмів (додаток Г.30). Превью подій складається з назви події та дати проведення (додаток Г.31).

Користувач може додати посилання на фільм у описі поста, для цього потрібно написати спеціальний символ “$” та почати вводити назву фільму. Відкриється список знайдених фільмів так як зображено у додатку Г.32. Користувач може додати безліч фільмів до опису.

Якщо в описі є фільм, він буде підкреслений помаранчевим кольором та при натисканні на нього користувач буде перенаправлений на сторінку фільму.

Для редагування постів необхідно натиснути на кнопку “Edit” (додаток Г.24). Інформація обраної публікації заповнить конструктор постів (додаток Г.33). Після завершення редагування потрібно натиснути кнопку “Edit Post”, Для того, щоб скасувати зміни потрібно натиснути на хрестик.

4.2.3 Рекомендації

У право куті головної сторінки (додаток Г.7) знаходиться блок рекомендацій - актуальний, або популярний контент. В цьому блоці зібрані чотири рекомендованих матеріала різного типу.

Перший рекомендований матеріал – опитування (додаток Г.34). Якщо користувач натисне на опитування - він буде перенаправлений на сторінку з опитуванням.

Другим рекомендованим матеріалом є подія (додаток Г.35). Подія має схожу логіку перенаправлення з опитуванням, але для того, щоб перейти на сторінку події потрібно натиснути на заголовок події. При натиску на превью конкретного учасника події користувач буде перенаправлений на профіль цього учасника.

Третій рекомендований матеріал - топ. Логіка перенаправлення схожа з логікою у рекомендованій події.

Останній рекомендований матеріал – огляд (додаток Г.36). Колір огляду відповідає настрою тексту, тобто добра оцінка - зелений, а погана - красний. В лівому верхньому куті знаходиться автор огляду, при натисканні користувач буде перенаправлений в профіль автора. Зправа від автора знаходиться кіно до якого відноситься цей огляд, клік на цей елемент перенаправить користувача на сторінку фільму. Для того, щоб подивитися огляд повністю треба натиснути на кнопку “read more”. Користувач має можливість залишити свою оцінку на огляд для цього потрібно натиснути на один з пальців в нижній частині огляду.

4.3 Пошук

На всіх сторінках додатку в верхній частині є поле для пошуку (додаток Г.37). Пошук відбувається не лише по фільмам, але й по всім іншим даним: користувачі, події, опитування, топи. При натисканні на знайдений елемент користувач перенаправляється на сторінку цього елементу. Для зручності користування пошуком існує фільтр пошуку (додаток Г.38).

4.4 Повідомлення

У верхній частині екрану знаходяться повідомлення для користувача (додаток Г.39). Повідомлення формуються на основі активності відносно аккаунта користувача та всіх матеріалів створених ним.

При наявності непрочитанних повідомлень над іконкою повідомлень з’являється помаранчеве коло. Для того щоб відмітити повідомлення як прочитане потрібно навести курсор на необхідне повідомлення. Прочитані повідомлення мають світліший відтінок.

Для того, щоб перейти до причини повідомлення, тобто особистого повідомлення, коментаря тощо, потрібно натиснути на нього.

4.5 Основна навігація

Основна навігація (додаток Г.40) розташована з лівої сторони інтерфейсу додатку, за винятком сторінки фільмів. При натисканні на логотип користувач потрапляє на головну сторінку (додаток Г.7). Під логотипом розташований блок з осноними сторінками. Активна сторінка підкреслюється.

4.6 Користувацькі налаштування

Для переходу до користувацьких налаштувань користувач повинен навести на іконку зі своїм ім’ям у правому верхньому куті додатку. Та натиснути на кнопку “Settings” у відкритому блоці (додаток Г.41).

Інтерфейс налаштувань складається з трьох розділів (додаток Г.42): управління аккаунтом, управління повідомленнями та налаштування приватності. Навігація між розділами здійснюється за допомогою кліку на необхідний. Активний розділ має помаранчеву лінію зверху.

Першим розділом налаштувань є налаштування аккаунту. В цьому розділі користувач має можливість зміни емейл, оновити пароль та видалити аккаунт.

Для зміни емейлу користувач повинен заповнити поле з бажаним емейлом, ввести пароль (додаток Г.43) та натиснути на кнопку “Save”. При виникненні валідаційних помилок кнопка збереження бажаних змін буде заблокована та користувач побачить повідомлення з помилками валідації. Якщо запит на

оновлення емейлу не пройшов успішно користувач також побачить повідомлення з описанням помилки.

Для зміни пароля користувачу потрібно заповнити поля зображені у додатку Г.44 та натиснути кнопку “Save”. Так само як при оновленні паролю користувач буде бачити валідаційні помилки та помилки зв’язані з запитом на оновлення.

Для видалення аккаунту потрібно натиснути кнопку “Permanently delete this account” зображену у додатку Г.45. Важливо зауважити, користувач не може відновити свій аккаунт після його видалення.

Повідомлення поділяються на дві групи: внутрішні повідомлення на сайті, емейл повідомлення. Користувач має можливість окремо налаштувати, які повідомлення він хоче отримувати з кожної групи.

Інтерфейс налаштування зображено у додатку Г.46. Для того щоб змінити налаштування потрібно зняти, або поставити флаг біля бажаних категорій повідомлень, після чого натиснути на копку “Save”.

За замовчуванням всі дані користувачів публічні, тобто кожен інший користувач може побачити історії або огляди кожного з користувачів. Для того, щоб змінити цю поведінку потрібно змінити налаштування приватності відповідної категорії.

З правої сторони від кожної категорії (додатку Г.47) є селект з варіантами налаштувань. Користувач може змінити необхідні йому параметри, після чого натиснути кнопку “Save” для збереження змін.

4.7 Профіль користувача

Кожен користувач має особистий профіль (додаток Г.48), інтерфейс профіля поділен на секції. Головна секція - “Profile” ця секція відображає інформацію користувача, надає можливість написати персональне повідомлення обраному користувачу та підписатися на нього.

Для написання персонального повідомлення потрібно натиснути на кнопку “Send message” після цього користувач буде перенаправлений на сторінку повідомлень. Користувач має можливість додати іншого до своїх підписок, для цього потрібно натиснути на кнопку “Follow”.

Профіль користувача може мати список улюблених фільмів. При кліку на фільм користувач буде перенаправлений на сторінку обраного фільма.

Для того щоб подивитись підписників чи підписки користувача потрібно натиснути на відповідний блок під аватаром користувача. Після натискання відкриється модальне вікно (додаток Г.49). При натисканні на певного користувача буде виконано перенаправлення на його сторінку. Для закриття модального вікна потрібно натиснути хрестик.

У профілі поточного користувача (додаток Г.50) є можливість редагування. Для того щоб почати редагування профіль потрібно натиснути на ікноку редагування в правій частині інтерфейсу.

Інтерфейс редагування користувача зображено у додатку Г.51. Для збереження змін необхідно натиснути на кнопку “Save”, відповідно для відміни “Cancel”, Користувач має можливість додавати улюблені фільми для цього у відповідному полі потрібно вписати назву фільму та обрати з результатів пошуку. Для видалення фільму з списку улюблених потрібно натиснути на хрестик у правій частині конкретного фільму.

Висновки до розділу 4

В данному розділі було розглянуто базовий функціонал реалізованого додатку та способи взаємодії з ним. Основна частина функціоналу додатку має схожий інтерфейс та алгоритми взаємодії з ним. Детально було розглянуто основну сторінку додатку за допомогою основної сторінки відбувається найбільш комунікації та розповсюдження інформаціїї між користувачами.

ВИСНОВКИ

В ході данного дипломного проекту було реалізовано спецалізовану соціальну мережу з використанням змарних сервісів. Реалізований додаток налічує функціонал для комунікації, поширення інформації, розвитку власного бренду чи бізнесу.

На основі огляду існуючих рішень можно сказати, що реалізований додаток ввібрал в себе найцікавіший та корисний функціонал сучасних аналогів. При розробці додатку були використані новітні інструменти для покращення стабільності, швидкодії та досвіду користування додатком. Також треба зауважити що додаток добре захищає данні користувача, через вибір користування сервісами Google та Facebook для авторизації.

Підсумовуючи сказане, в створеному проекті реалзовані всі вимоги до функціоналу та інтерфейсу. Додаток надає чудову можливість розвиватися та знаходити корисну інформацію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Firebase messaging – хмарний сервіс для повідомлень [Електронний ресурс]: Режим доступу: https://firebase.google.com/docs/cloud-messaging/?gclid=CjwKCAjwlZf3BRABEiwA8Q0qq9lpR9xMCeCfoWPEv5NpkZKr7xtm-IY4JPCHBtFN3d93KLHTDVZViRoCiuEQAvD_BwE
2. Документація Amazon S3 – сервіст для зберігання об'єктів [Електронний ресурс]: Режим доступу: https://docs.aws.amazon.com/s3/index.html
3. Документація ElasticSearch – хмарний сервіс для зберігання даних [Електронний ресурс]: Режим доступу: https://www.elastic.co/guide/index.html
4. Документація PostgreSQL – реляційна база данних [Електронний ресурс]: Режим доступу: https://www.postgresql.org/docs/
5. Документація React – бібліотека для розробки клієнтської частини [Електронний ресурс]: Режим доступу: https://reactjs.org/
6. Документація Create react app – cli для створення нових проектів [Електронний ресурс]: Режим доступу: https://github.com/facebook/create-react-app
7. Документація TypeScript – типізований суперсет JavaScript [Електронний ресурс]: Режим доступу: https://www.typescriptlang.org/
8. Документація Redux – бібліотека для обробки стану [Електронний ресурс]: Режим доступу: https://redux.js.org/
9. Документація react-router – бібліотека для створення навігації [Електронний ресурс]: Режим доступу: https://github.com/ReactTraining/react-router
10. Документація Express – фреймворк для Node.js серверверу [Електронний ресурс]: Режим доступу: https://expressjs.com/
11. Документація TypeOrm – типізована orm для Node.js [Електронний ресурс]: Режим доступу: https://typeorm.io/

Додаток А – Приклад reducer

```
export default (state = initialState, action) => {
  switch (action.type) {
    case fetchUser.TRIGGER:
      return {
        ...state,
        loading: true
      };
    case fetchUser.SUCCESS:
      return {
        ...state,
        profileInfo: action.payload.data.user
      };
    case fetchUser.FAILURE:
      return {
        ...state,
        error: action.payload.erorr
      };
    case fetchUser.FULFILL:
      return {
        ...state,
        loading: false
      };
    default:
      return state;
  }
}
```

Додаток Б – Конфігурація підключення бази даних

```
import { ConnectionOptions } from "typeorm";
import entities from "../entities/index";
import migrations from "../seed/index";

const {
  DATABASE_URL,
  DB_HOST,
  DB_USERNAME,
  DB_PASSWORD,
  DB_NAME
} = process.env;

const url =
  DATABASE_URL ||
  `postgresql://${DB_USERNAME}:${DB_PASSWORD}@${DB_HOST}/${DB_NAME}`;

const dbConfig: ConnectionOptions = {
  type: "postgres",
  url,
  synchronize: true,
  logging: false,
  entities,
  migrations
};

export default dbConfig;
```

Додаток Г – Ілюстрації до інструкції користувача

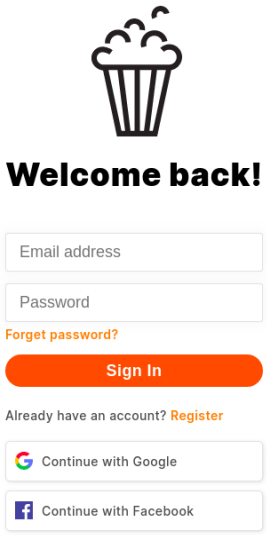

Рисунок Г.1 - Сторінка авторизації

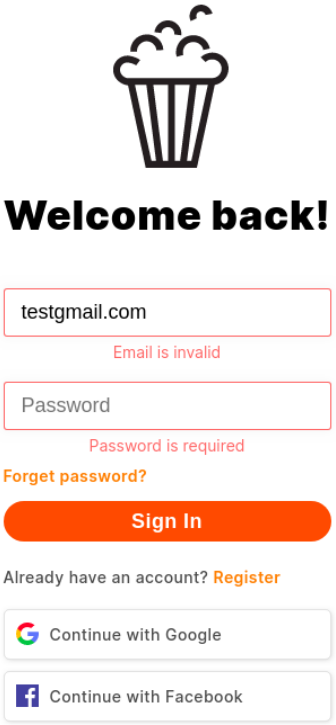

Рисунок Г.2 - Помилка валідації даних

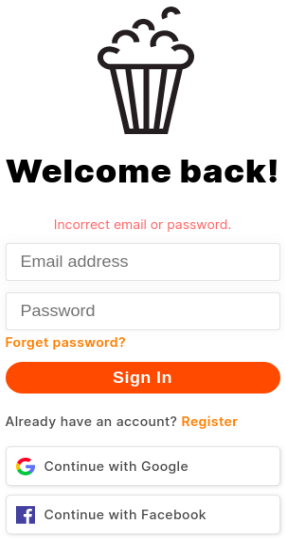

Рисунок Г.3 - Помилка авторизації

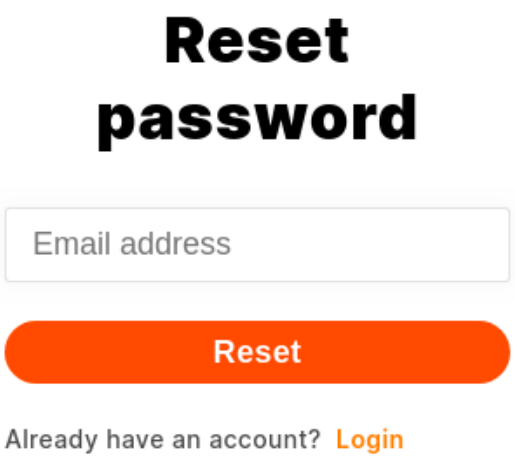

Рисунок Г.4 - Сторінка підтвердження відновлення паролю

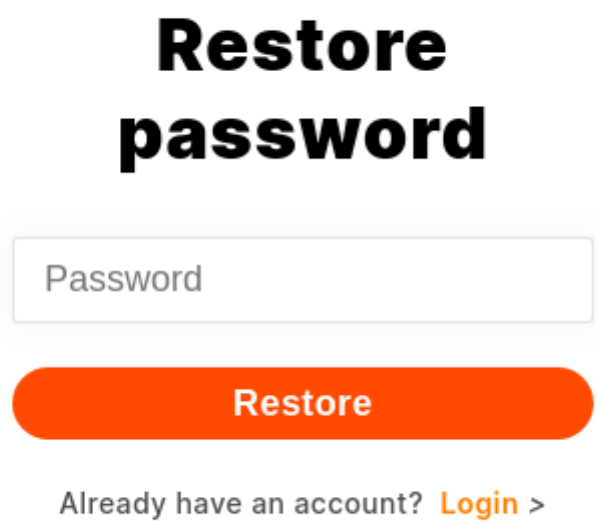

Рисунок Г.5 - Сторінка відновлення паролю

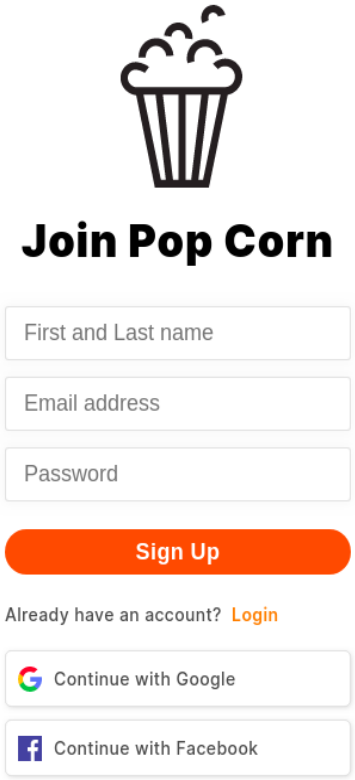

Рисунок Г.6 - Сторінка реєстрації

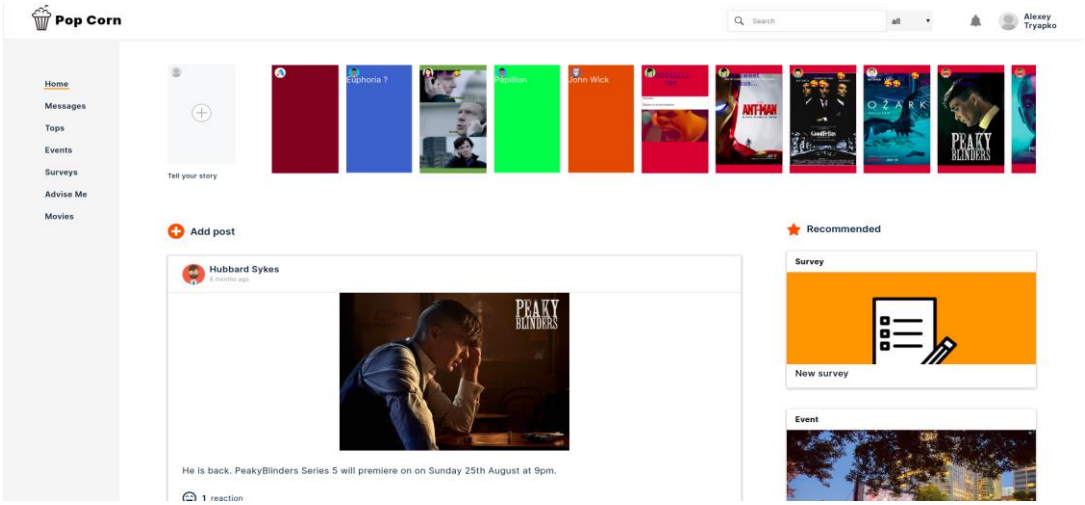

Рисунок Г.7 - Головна сторінка

Рисунок Г.8 - Блок з історіями

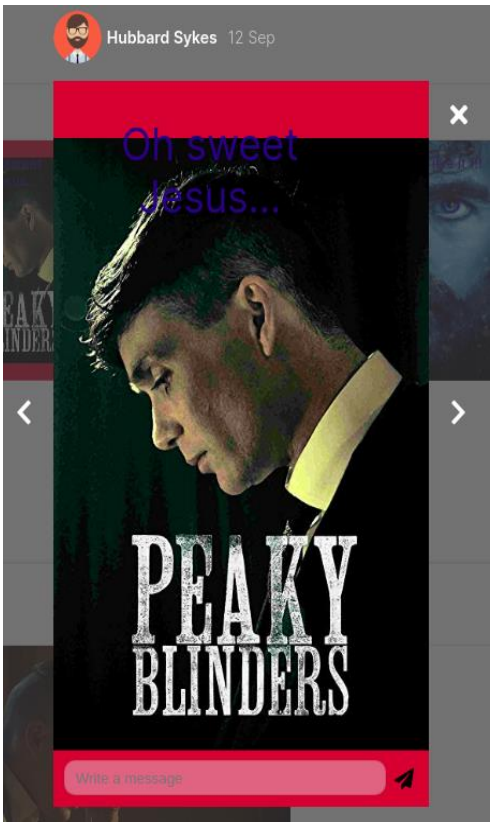

Рисунок Г.9 - Відкрита історія

Рисунок 5.10 - Поле реакцій на історію

Рисунок Г.11 - Повідомлення з текстовою реакцією

Рисунок Г.12 - Повідомлення при використанні смайлу

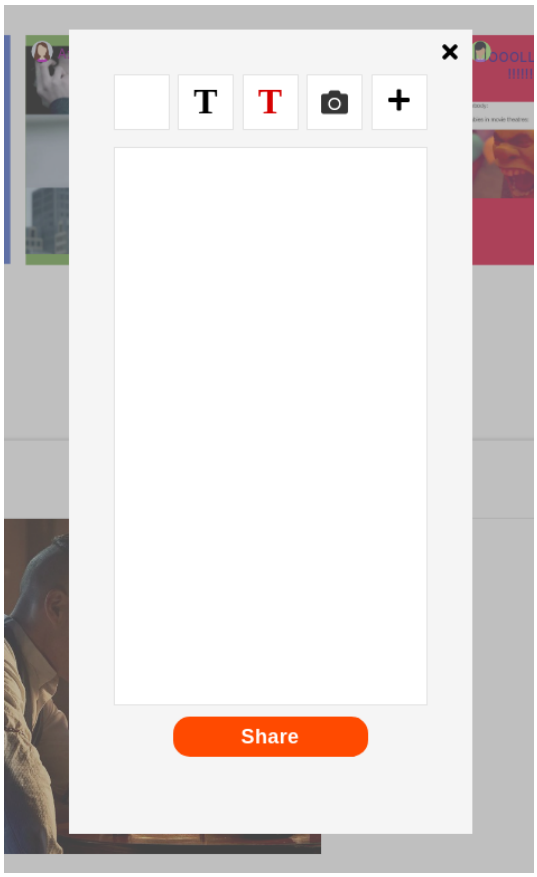

Рисунок Г.13 = Конструктор історій

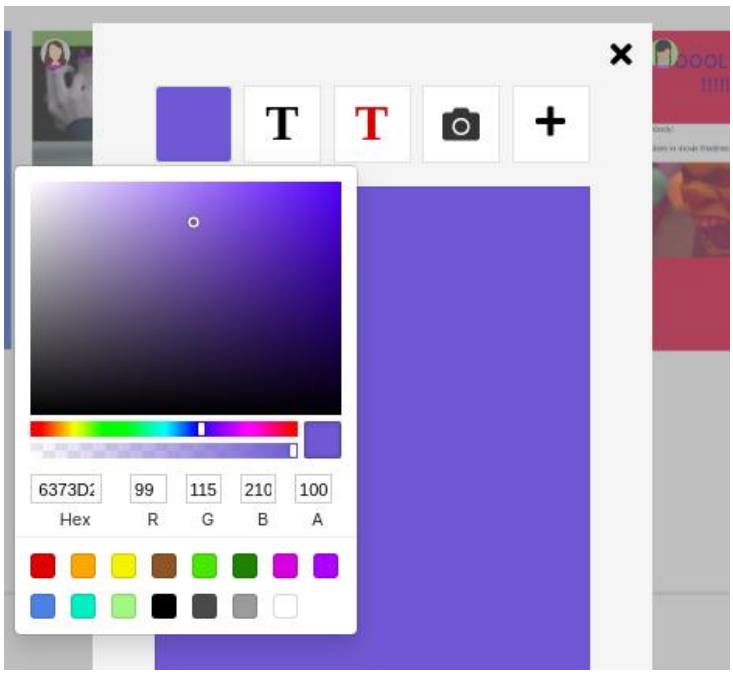

Рисунок Г.14 - Вибір кольору фону

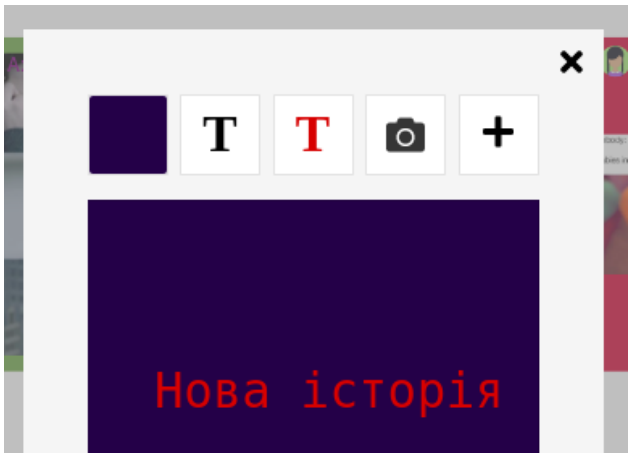

Рисунок Г.15 - Текст на історії

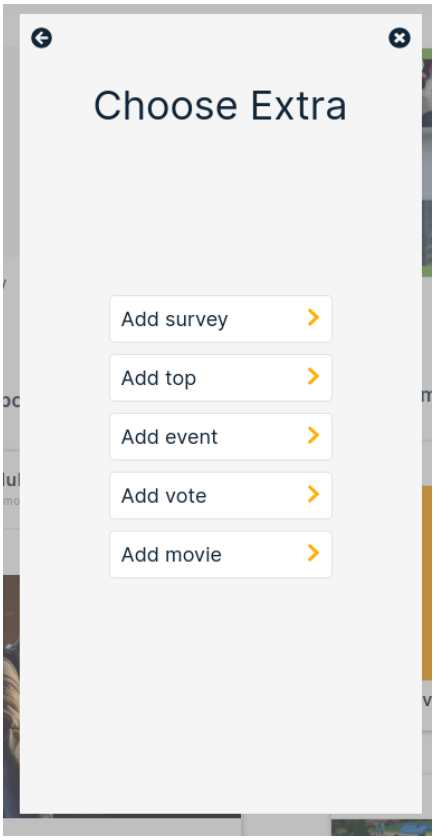

Рисунок Г.17 - Додаткові матеріали для історії

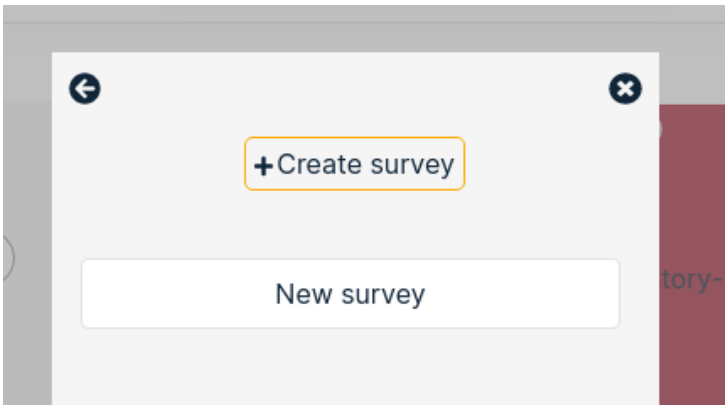

Рисунок Г.17 - Вибір опитування

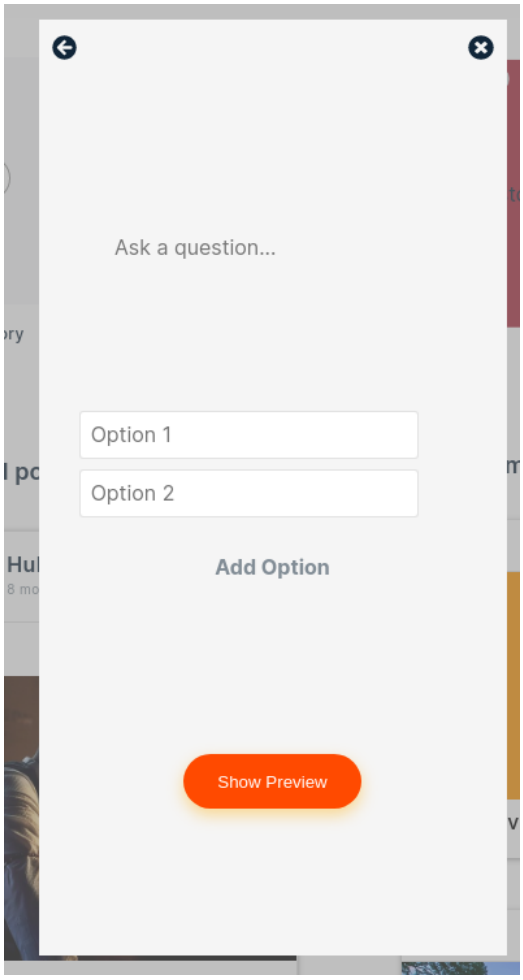

Рисунок Г.18 - Інтерфейс додавання голосувань

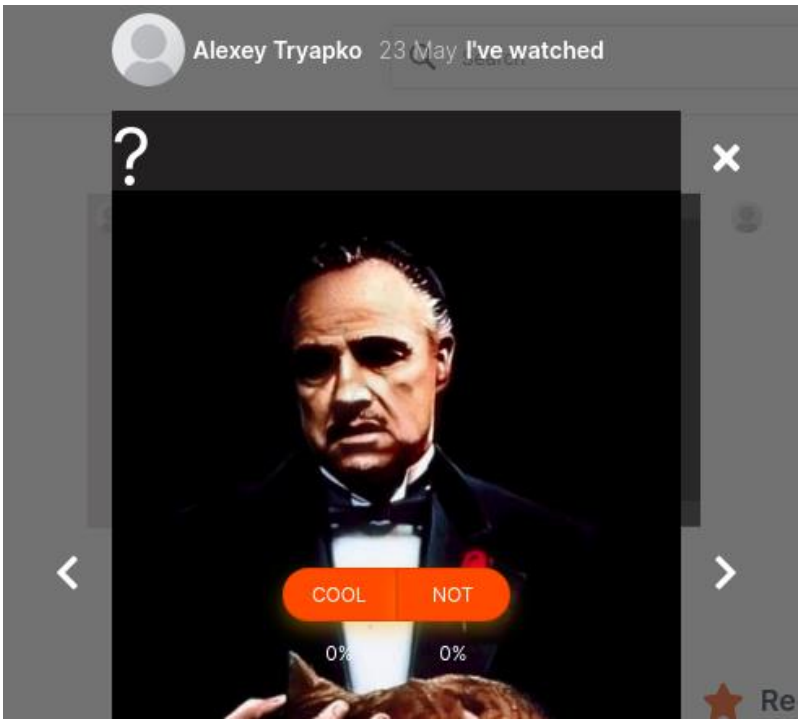

Рисунок Г.19 - Голосування в історії

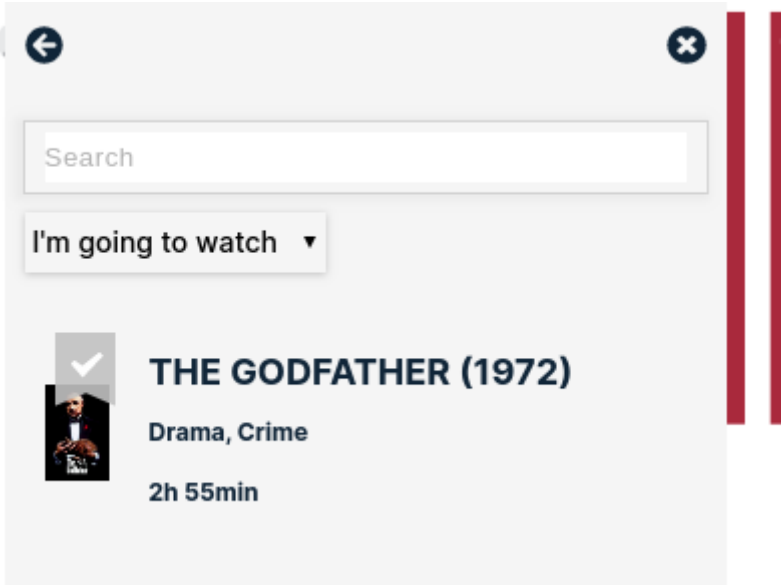

Рисунок Г.20 - Інтерфейс додавання фільму

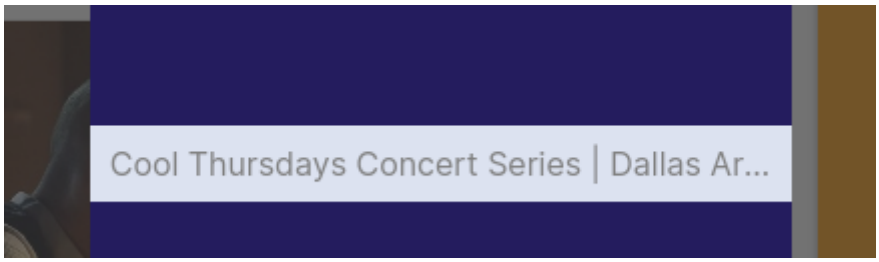

Рисунок Г.21 - Блок додаткової інформації

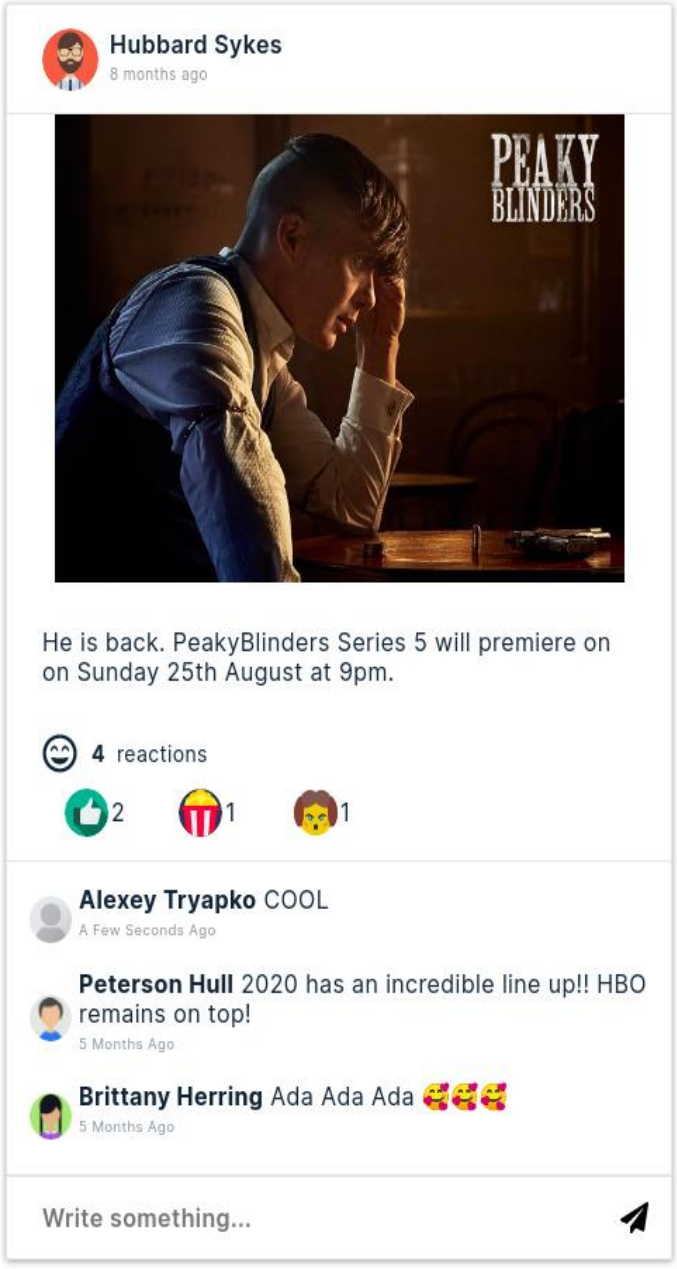

Рисунок Г.22 - Пост

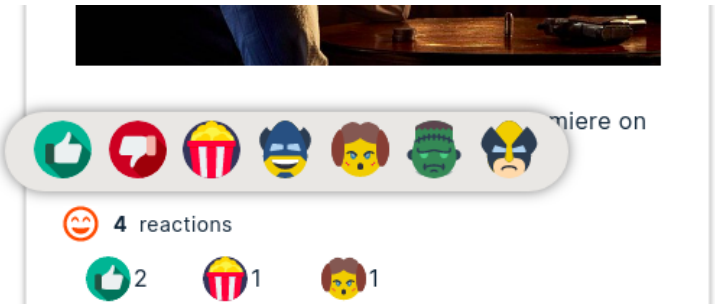

Рисунок Г.23 - Реакції на пости

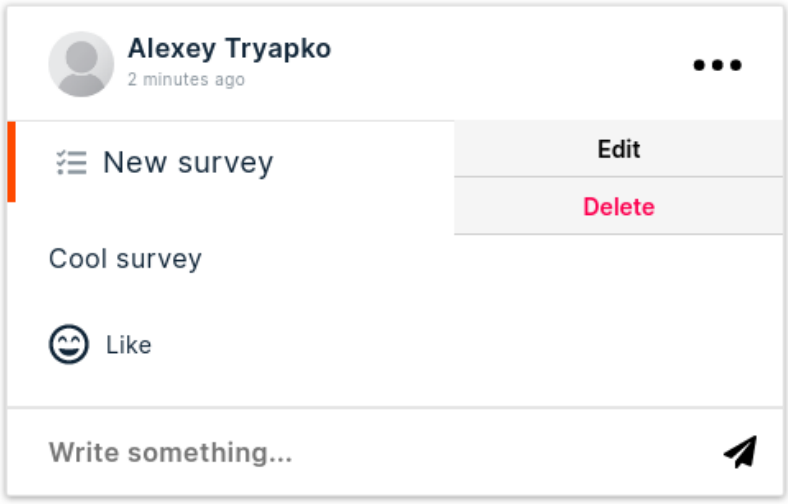

Рисунок Г.24 - Пост поточного користувача

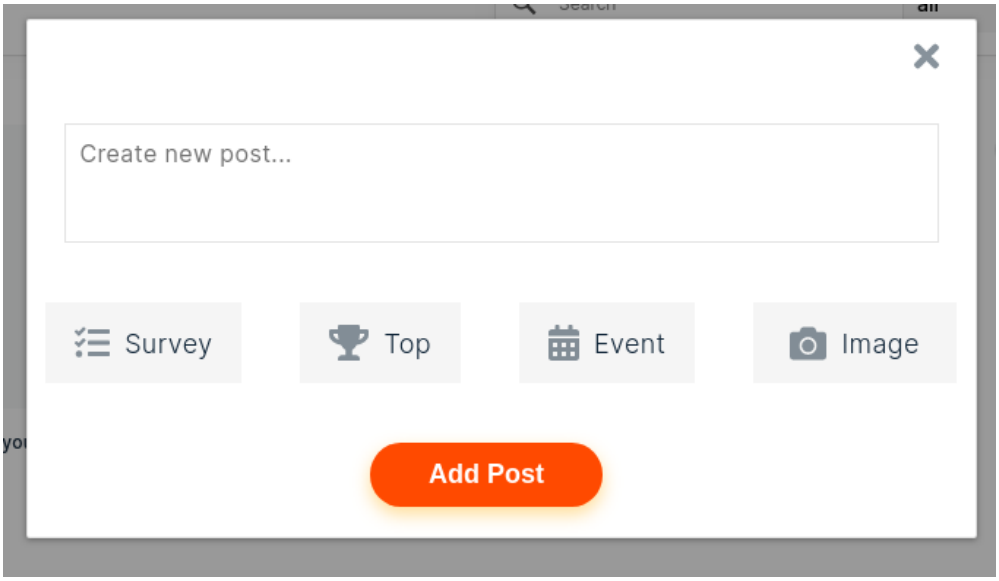

Рисунок Г.25 - Конструктор постів

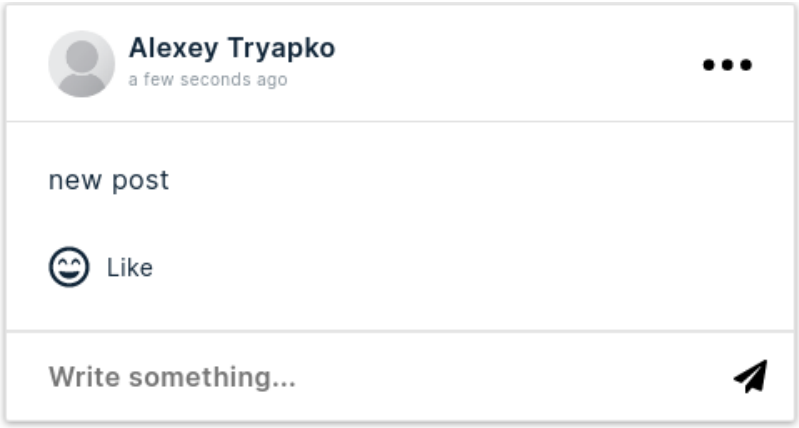

Рисунок Г.26 - Пост з використанням лише тексту

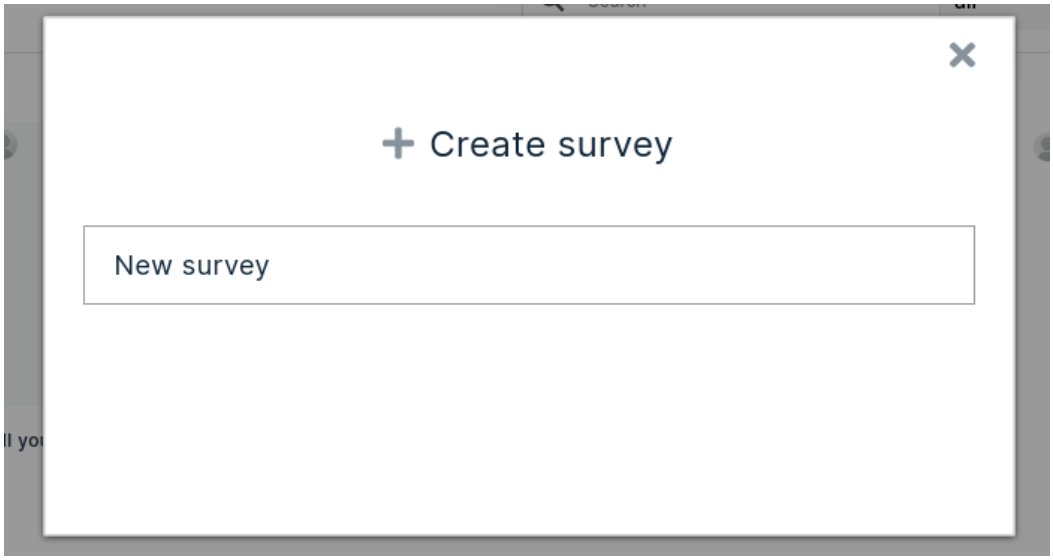

Рисунок Г.27 - Вибір опитування

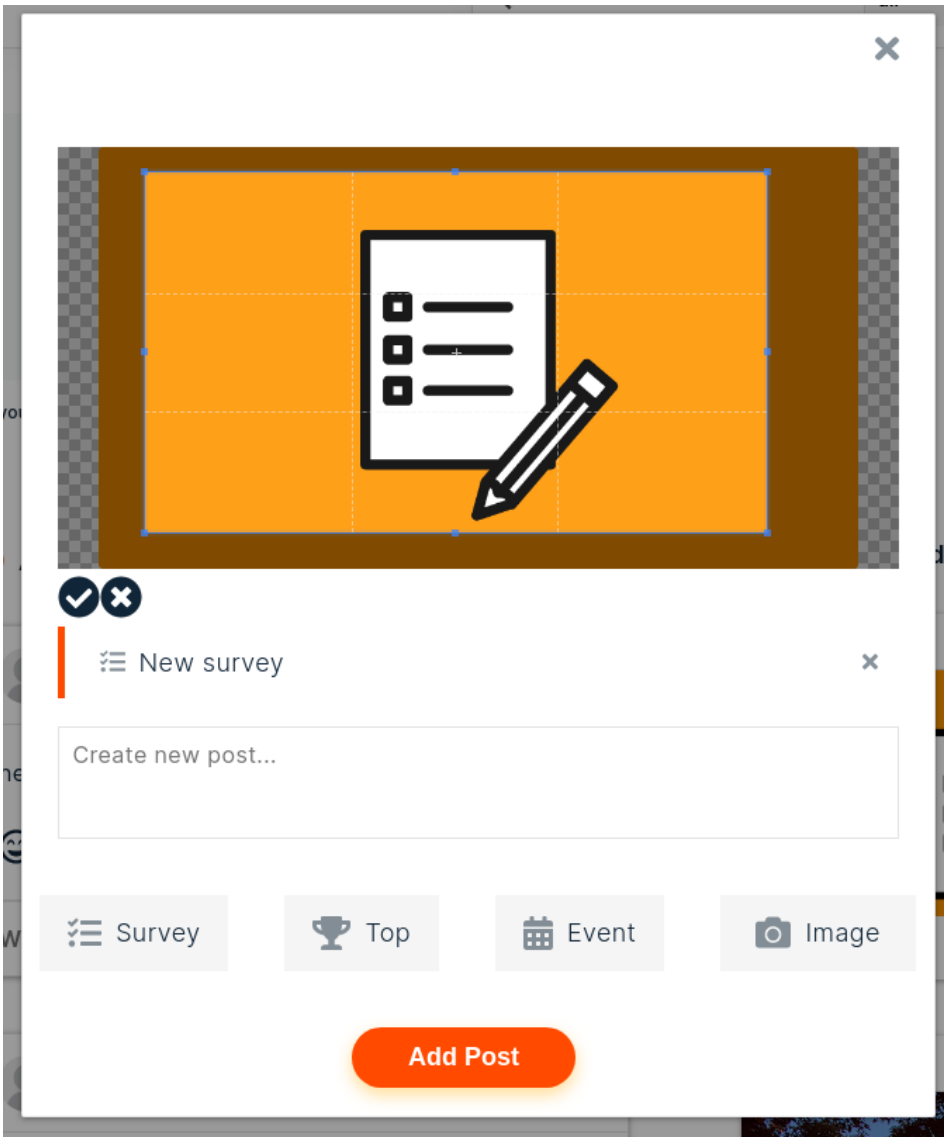

Рисунок Г.28 - Кадрування зображення

Рисунок Г.29 - Пост з прикріпленим опитуванням

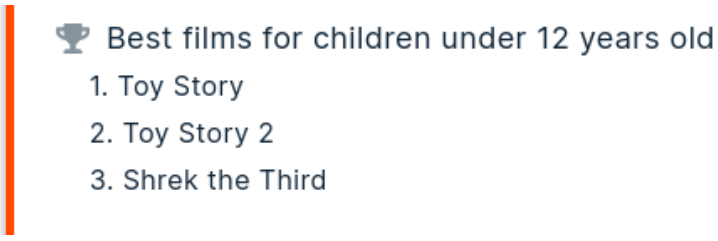

Рисунок Г.30 - Превью топа прикріпленного до поста

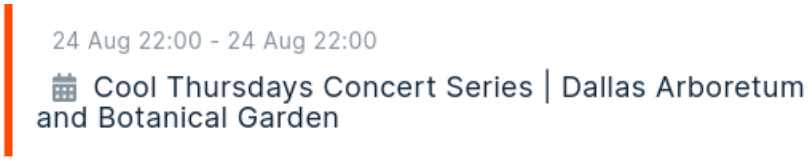

Рисунок Г.31 - Превью події прикріпленої до поста

Рисунок Г.32 - Додавання фільму до опису поста

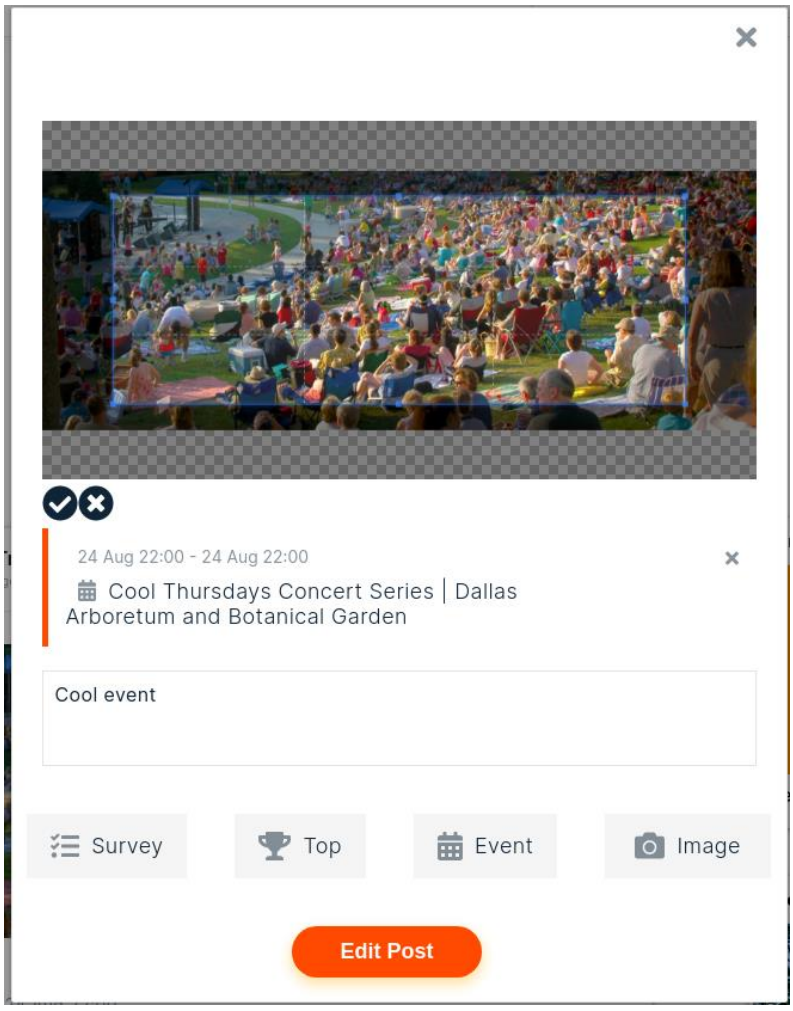

Рисунок Г.33 - Інтерфейс редагування постів

Рисунок Г.34 - Приклад рекомендованого опитування

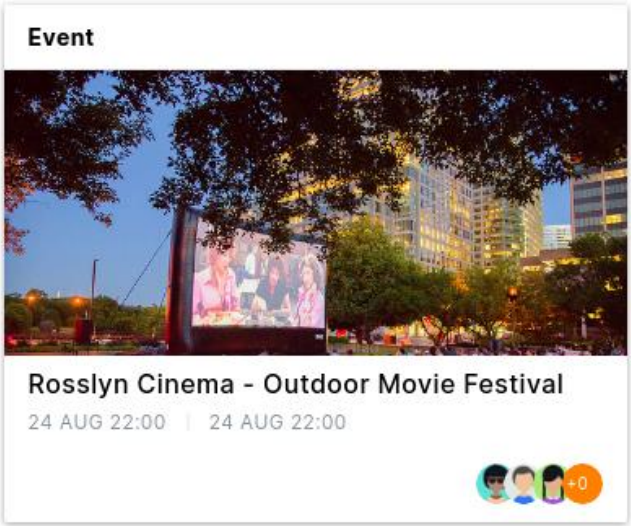

Рисунок 5.35 - приклад рекомендованої події

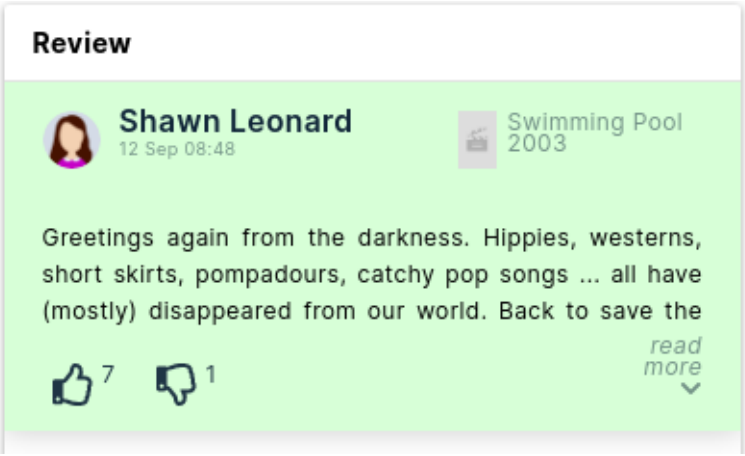

Рисунок Г.36 - Приклад рекомендованого огляду

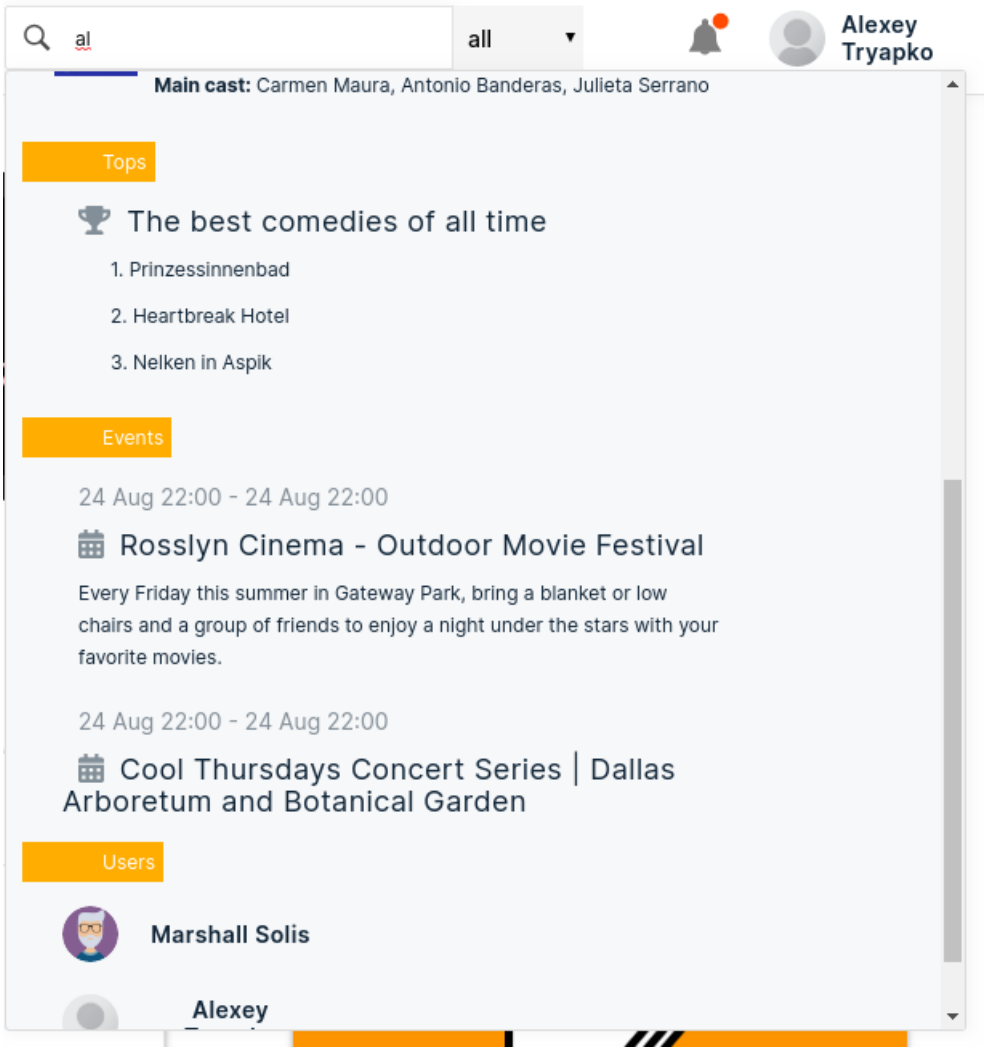

Рисунок Г.37 - Глобальний пошук

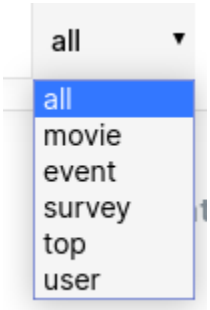

Рисунок Г.38 - Фільтр пошуку

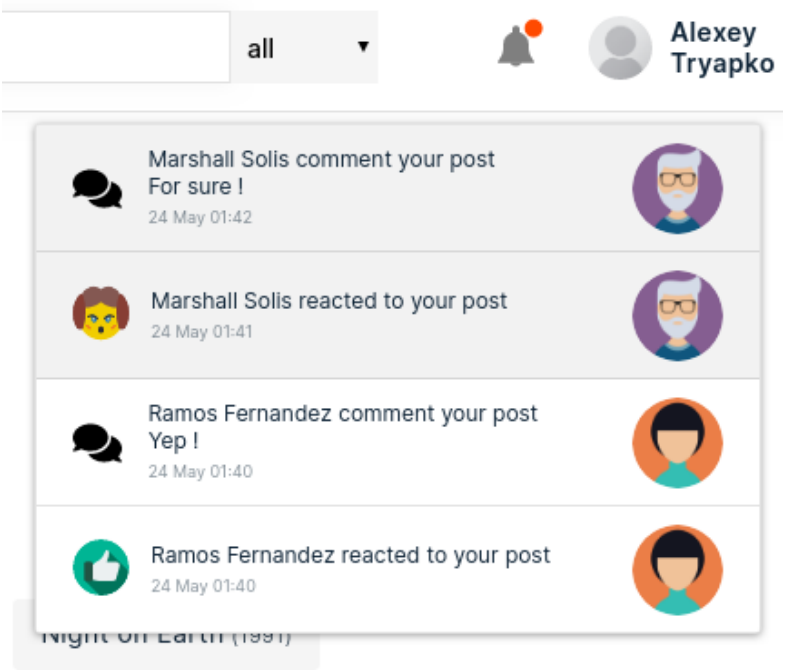

Рисунок Г.39 - Інтерфейс повідомлень

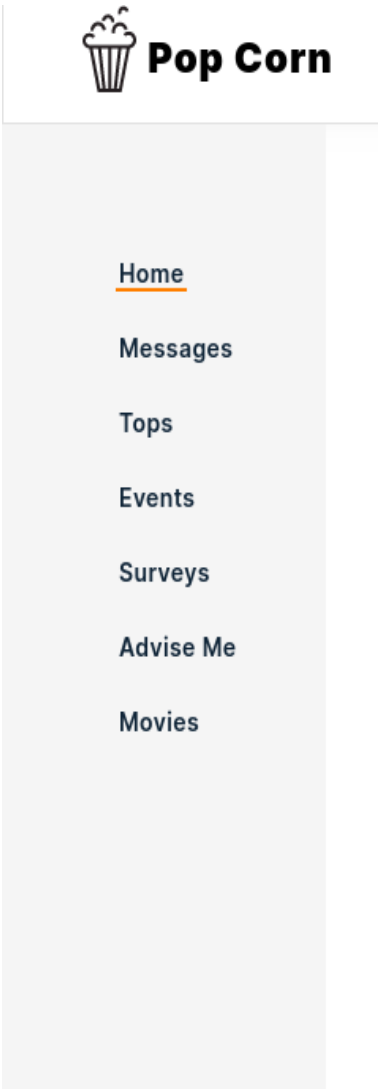

Рисунок Г.40 - Навігація

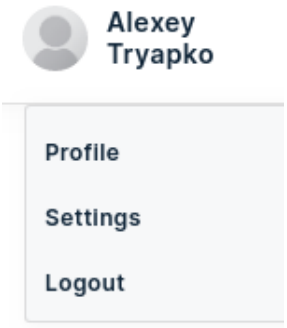

Рисунок Г.41 - Користувацький блок

Рисунок Г.42 - Розділи налаштувань додатку

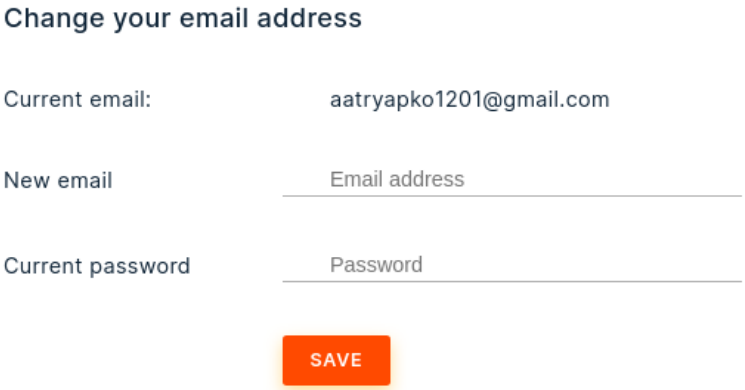

Рисунок Г.43 - Зміна емейл

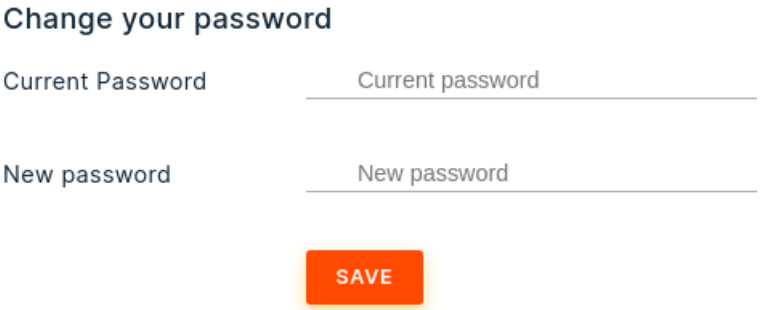

Рисунок Г.44 - Зміна паролю

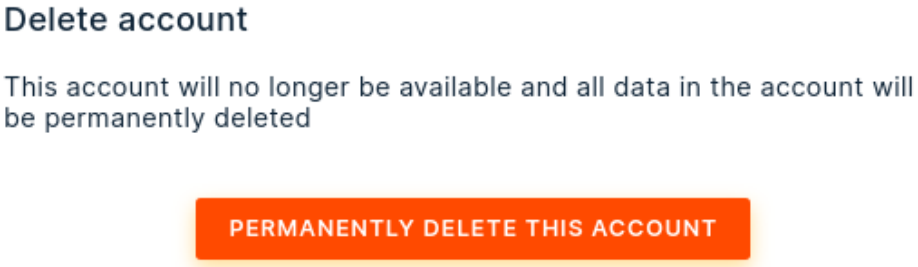

Рисунок Г.45 - Видалення аккаунту

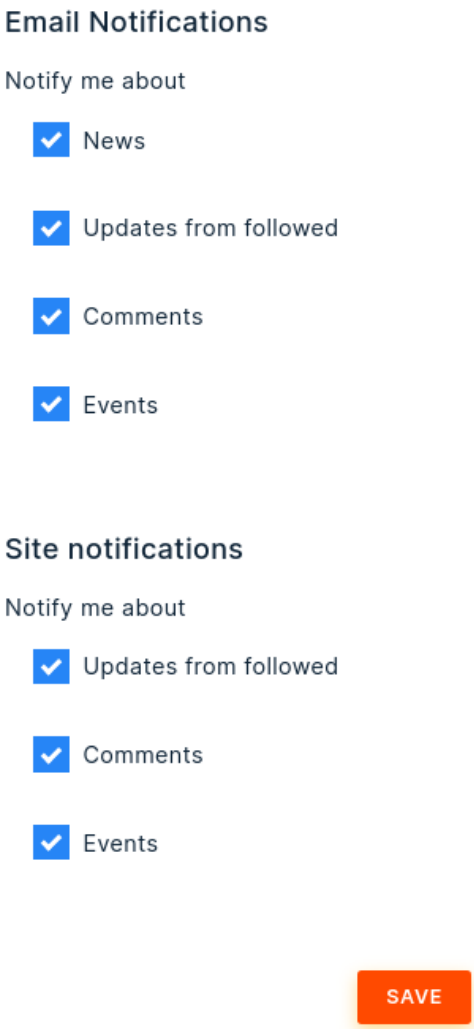

Рисунок Г.46 - Налаштування повідомлень

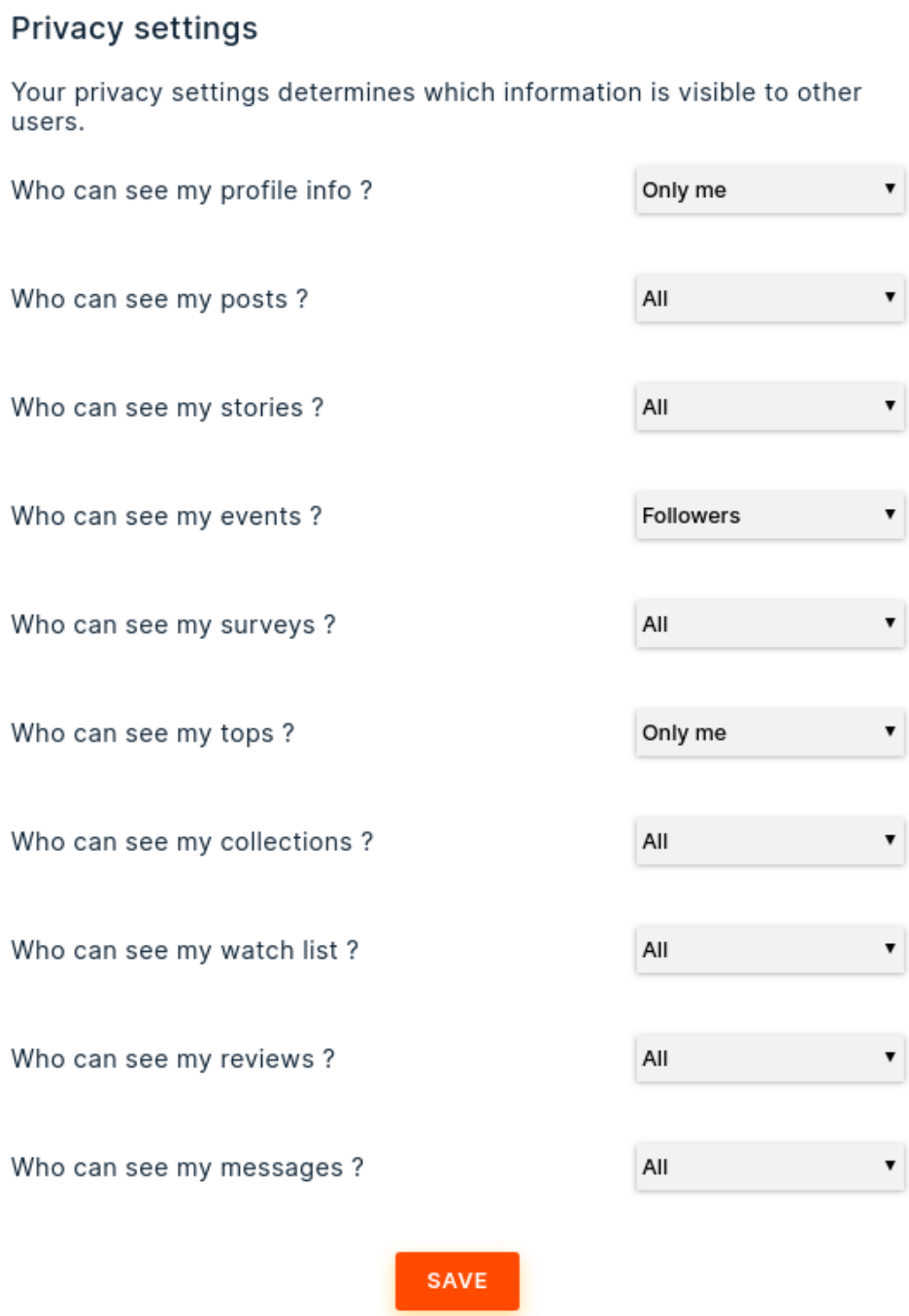

Рисунок Г.47 - Налаштування приватності

Рисунок Г.48 - Профіль користувача (не поточного)

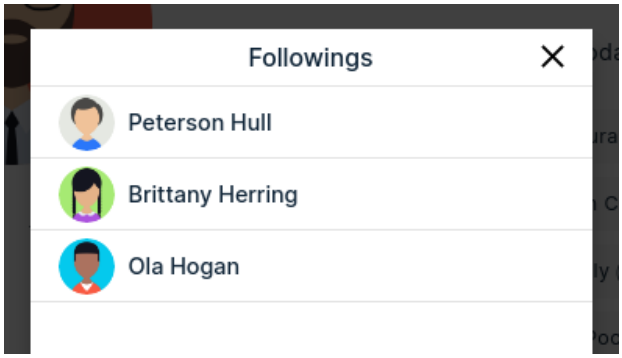

Рисунок Г.49 - Інтерфейс списку підписок

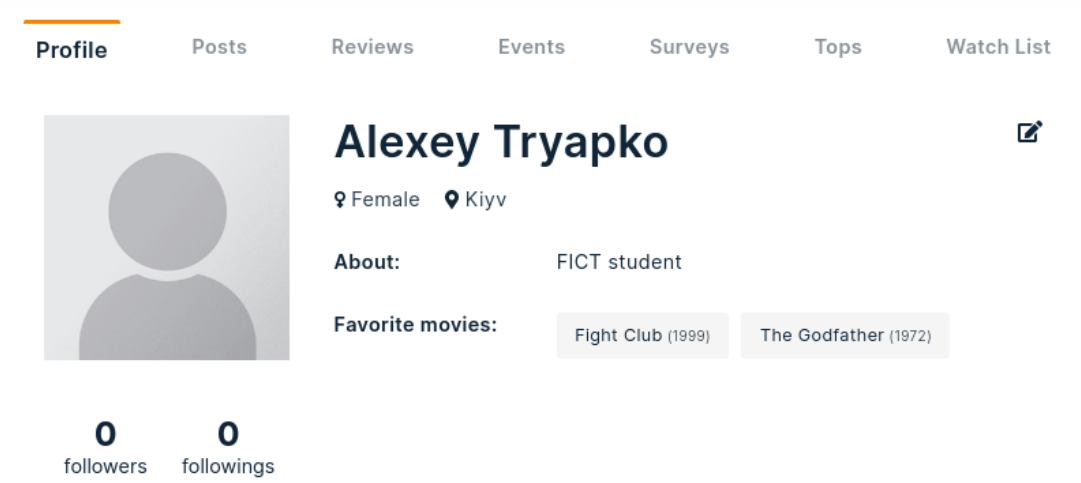

Рисунок Г.50 - Профіль поточного користувача

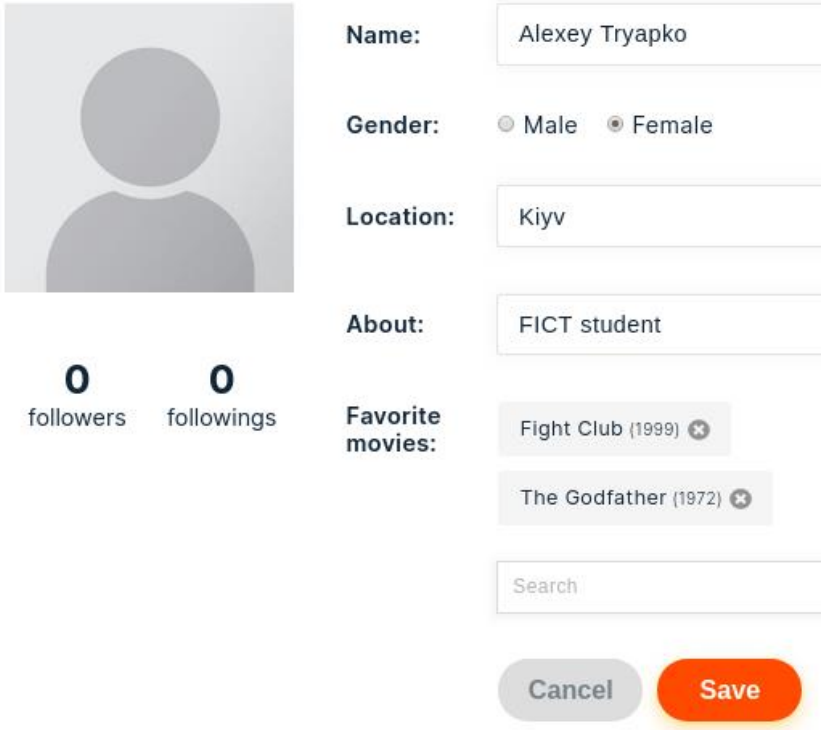

Рисунок Г.51 - Редагування профілю користувача