Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

Інститут деревообробних та комп'ютерних технологій і дизайну

(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій

(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

другий (магістерський)

(рівень вищої освіти)

на тему:

Розроблення соціальної мережі для віддаленої роботи бізнес-підприємства

Виконав: студент 6 курсу, групи КН-61

напряму підготовки

122 – "Комп'ютерні науки"

(шифр і назва напряму підготовки, спеціальності)

Ревура Володимир Іванович

(прізвище та ініціали)

Керівник: Процах Наталія Петрівна

(прізвище та ініціали)

Рецензент_____

(прізвище та ініціали)

Львів – 2021 року

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Освітньо-кваліфікаційний рівень другий (магістерський)

Напрямок 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри _____Крошний І. М."____"
_____202_ року

З А В Д А Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Ревура Володимир Іванович
(прізвище, ім'я, по батькові)

1. **Тема роботи**: Розроблення соціальної мережі для віддаленої роботи бізнес- підприємства

керівник роботи:д.ф.-м.н., проф. Процах Наталія Петрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "31"_грудня_____2020року № С-593
2. Термін подання студентом роботи:
3. Вихідні дані до роботи: пристрій з доступом до інтернету
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити:

Вступ
Розділ 1: Стан проблемної області
Розділ 2: Інформаційне та математичне забезпечення
Розділ 3: Програмне та технічне забезпечення
Висновки

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

Слайди доповіді, актуальність теми, постановка завдання, скріншоти соціальної мережі, опис функціональності, висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	**Процах Н.П., професор кафедри ІТ**		

7. Дата видачі завдання: 18 грудня 2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел та інтернет ресурсів, пошук загальної інформації за вказаною темою	04.02.2021 20.02.2021	
2	Вибір технологій, підходів і фреймворків для поставленої задачі	09.03.2021 21.03.2021	
3	Проектування програмного забезпечення, налаштування середовища	03.04.2021 20.04.2021	
4	Створення користувацького інтерфейсу	01.06.2021 20.06.2021	
5	Імплементація функціональності соціальної мережі, завершення створення дизайну	24.06.2021 30.07.2021	
6	Тестування програми та аналіз результатів	02.08.2021 26.08.2021	
7	Оформлення записки до дипломної роботи	28.10.2021 30.11.2021	
8	Здача пояснювальної записки на рецензування	01.12.2021 10.12.2021	
9	Створення презентації та підготовка до захисту	11.12.2021 15.12.2021	

Студент: Ревура Володимир Іванович

Керівник роботи: Процах Наталія Петрівна

Технічне завдання

Розроблення соціальної мережі для віддаленої роботи бізнес-підприємства.

Вдосконалення соціальної мережі, імплементація нового функціоналу для віддаленої роботи:

- Створення сповіщень (вподобання допису, додавання в друзі, чат сповіщення);
- Авторизація з правами адміністратора;
- Функціонал для адміністратора (видалення постів, блокування/видалення користувача, видалення коментарів);
- Створення чату;
- Рекомендовані друзі для спілкування;
- Інше.

Можливість користування даною мережею на будь-якому пристрої, зручний, простий і інтуїтивно зрозумілий інтерфейс.

РЕФЕРАТ

Магістерська дипломна робота (проект): пояснювальна записка: 61 стор., 21 рис., 3 табл., 1 додаток, 12 джерел; графічний матеріал – 15 аркушів.

У данійдипломній роботі було використано сучасні фреймоврки для створення мережі, а також популярний протокол Websocketдля обміну повідомленнями в режимі реального часу. Перевірено швидкодію надсилання повідомлень з різних девайсів і з кількома учасниками одночасно.

Програмне забезпечення реалізовано за допомогою JavaScript, React, Redux, Node.js, Websocket.

Ключові слова:соціальна мережа, автентифікація, адаптивність, гнучкий дизайн, веб-переглядач, React, Redux, JavaScript, Node.js, Express, MongoDB, TypeScript, Socket.io

ABSTRACT

Master's thesis (project): explanatory note: 61 pages, 21 pictures, 3 tables, 1 application, 12 sources; graphic material – 15 sheets.

In this master’s project were used modern frameworks to create a network, andthe popular Websocket protocol for real-time messaging. The speed of sending messages from different devices and with several participants at the same time was also tested.

The software was developed usingJavaScript, React, Redux, Node.js, Websocket.

Keywords:social network, authentication, responsive and flexible design, web-viewer, React, Redux, JavaScript, Node.js, Express, MongoDB, TypeScript, Socket.io

Зміст

Вступ

Соціальні мережі –найпопулярніший спосіб проведення вільного часу. Майже кожен з нас перебуває в Інтернеті на роботі, в школі чи вдома. Причина, через яку реклама через соціальні мережі настільки популярна, є логічною.

Традиційний Інтернет-маркетинг в основному працює з продуктами, які вже користуються попитом. Якщо товар є новим продуктом, і його ніхто не шукає в пошукових системах, які можуть успішно просувати товар, успіху не буде. У цьому випадку соціальні мережі спочатку реагують на просування новизни, створюючи тим самим нові вимоги.

Актуальність проблеми.Залежно від цілей компанії, соціальні мережі можуть використовуватися як канали для побудови іміджу, сервісні столи, платформи зв'язку або інструменти прямого продажу. Також, якщо взяти для прикладу сьогоднішню ситуацію, час пандемії, карантину, то мабуть без соціальних мереж неможливо обійтись навіть декілька днів. Усі обговорення відбуваються на різних платформах, відео-чатах і тому подібне. Хіба б не чудово було б мати можливість обмінюватись повідомленнями, ділитись файлами, брати участь у відео-конференціях і це все лише в одній програмі?

Об’єкт дослідження – методи та засоби створення соціальних мереж, розробка месенджера.

Предмет дослідження – соціальні мережі.

Мета роботи– Розроблення соціальної мережі для віддаленої роботи бізнес підприємства.

У результаті виконання магістерської роботи реалізовано соціальну мережу, яка дозволяє користувачам зареєструватись в системі, слідкувати за новинами друзів, ділитись своїми новинами, редагувати профіль і інше. Мережа підтримує один тип користувача, ким може стати будь-яка людина,

що пройшла реєстрацію, використавши форму для збереження основних даних (ім’я та прізвище, емейл, пароль), або, скориставшись сервісом отримання даних через пошту (Gmail). Використання програми дозволяє користувачам просто і комфортно користуватись усім функціоналом соціальної мережі завдяки зручному інтерфейсу, який є доступним для користувачів пристроїв з будь-якими розмірами екранів. Для програмної реалізації цієї системи обрано мову програмування JavaScript, для створення користувацького інтерфейсу – мова розмітки JSX та препроцесорна мова SCSS, для асинхронного обміну даними з сервером – мову JavaScriptз фреймворком – React. Для роботи з базою даних –NodeJS і MongoDb.

Новизна роботи. Новизна полягає у тому, що крім розробленої соціальної мережі для слідкування за новинами і створенням власних постів, було імлементовано можливість обміну повідомленнями всередині цієї мережі. Не потрібно скачувати додаткових програм, не потрібно відкривати декілька сторінок у браузері і не обов’язково заходити тільки з комп’ютера чи планшета. Завдяки гнучкому дизайну просто користуватись навіть з телефона.

Практичне значення одержаних результатів. Розроблено соціальну мережу включно з месенджером, тестування, яке проводилося локально з невеликою кількістю людей одночасно, пройшло без жодних проблем.

1. Стан проблемної області

1.1 Поняття про соціальні мережі і месенджери

В умовах глобального розвитку Інтернету ефективне використання можливостей, що надаються Інтернетом, може стати ключовим фактором успіху. Більшість українців хочуть отримати необхідну інформацію в Інтернеті. Існує безліч варіантів використання Інтернет-ресурсів: створити власний сайт бібліотеки, реклами на інших сайтах.

Також мережі допомагають створювати постійну аудиторію сайту: підписники регулярно бачитимуть оновлення на своїй стрічці новин, і тому будуть старатись слідкувати за новинами, заходячи знову і знову в мережу. Також соціальні мережі дозволяють легко отримати зворотній зв’язок з читачами завдяки повідомленню, в якому вони можуть висловити свою думку щодо якості роботи мережі, вказати на можливі недоліки, запропонувати ідеї щодо кращого обслуговування, або задати питання.

Соціальна мережа – це соціальна структура, сформована окремими особами чи організаціями. В Інтернеті соціальна мережа виглядає як сайт, де користувачі можуть створювати персональні сторінки та спільноти для обміну інформацією та розповсюдження її серед великої кількості людей. Більше половини користувачів Інтернету зареєструвались принаймні в одній соціальній мережі, що свідчить про їхню популярність. Тому їх використання для популяризації бібліотеки є логічним рішенням для пошуку нових читачів та підтримання зв’язку з існуючими читачами.

1.1.1 Соціальні мережі

Однією з характеристик мережі є те, що нова інформація постійно надходить від друзів, спільнот, на які користувачі підписуються тощо. Це призводить до необхідності йти в ногу з часом, необхідним для обслуговування сторінок бібліотеки: новини швидко старіють і можуть бути легко загублені на інших сторінках та в групах без постійного оновлення.

У соціальних мережах є можливість сподобати або поширити опубліковані новини. Кількість вподобань дозволяє оцінити актуальність новин серед користувачів, а поширення новин ще більше розширює натовп читаючих новин. Тому використання різноманітних соціальних мереж дозволяє бібліотекам віддалено розширювати канали спілкування зі своїми користувачами. Соціальні мережі дають можливість швидко повідомляти про події в мережі, отримувати відгуки, стежити за коментарями громадськості та аналізувати значення мережного персоналу в різних сферах.

Завдяки розвитку технологій соціальні мережі перестали бути єдиним місцем для спілкування. В наш час існує безліч мобільних додатків-месенджерів, які можуть задовольнити більш комфортні потреби у спілкуванні, тому їх популярність з кожним днем зростає. Розуміючи потенціал цих мобільних додатків, більшість компаній почали використовувати Messenger як допоміжний інструмент для спілкування зі споживачами. Зрештою, смартфони доступні цілодобово, а це означає, що вхідні повідомлення будуть помічені з майже 100% ймовірністю.

До переваг Messenger належать: охоплення більшої частки користувачів, можливість обміну різними типами вмісту (від великих текстових файлів, зображень, відео до географічного розташування) та максимальна швидкість відповіді на повідомлення користувачів (push-повідомлення) .

Крім того, основними перевагами месенджера є можливість завжди залишатися на зв'язку, забезпечувати цілодобову підтримку та забезпечувати найвищий рівень конфіденційності при вирішенні проблем користувачів.

Незважаючи на популярність месенджерів, більшість українців (67%) витрачають менше години на день, спілкуючись в месенджерах. 24% людей проводять близько 3 годин на день. Лише 3% користувачів проводять у Месенджері більше 6 годин.

1.1.2 Популярні месенджери

Цікавою особливістю є те, що жінки, як правило, проводять більше часу у месенджерах, ніж чоловіки. Люди похилого віку рідше спілкуються в месенджері, 79%молодих людей у віці від 45 до 59 років проводять менше однієї години, спілкуючись у месенджері на день. На відміну від них, молоді люди у віці 16-29 років проводять в середньому 3 години на день у месенджерах.

1.2 Створення месенджера

Для того щоб створити свій власний обмінник інформацією потрібно дотримуватись певних кроків і імплементувати основні функції.

Авторизація

Для користувачів час реєстрації не повинен перевищувати двох хвилин. Щоб пришвидшити процес, можна додати інтеграцію до соціальних мереж, зареєструватися за номером телефону або електронною адресою. Потім користувач зможе вибрати фотографію профілю та ім’я.

Доступ до контактів

Ця функція значно спрощує процес авторизації, оскільки користувачі можуть зареєструватися, синхронізувати свої контакти та негайно надіслати комусь повідомлення.

Обмін повідомленнями

Ця функція є суттю вашої програми. Для того, щоб користувачі успішно спілкувались у чаті, вам потрібно: приватний чат та груповий чат, надсилання та скасування повідомлень, стан доставки, історія чату та редагування тексту. Також можна добавити голосові та відео повідомлення.

Обмін файлами

Можливість обміну мультимедійними файлами - ще одна причина, через яку люди регулярно використовують вашу послугу. Найкраще додати можливість надсилати фотографії, відео, GIF-файли та документи один одному, і тоді багато людей використовуватимуть цей Messenger.

Push-повідомлення

Як створити Messenger без функції обміну повідомленнями? Це неможливо! У цю епоху, коли ми боїмося пропустити важливі речі, користувачі повинні негайно знати, що вони отримали нове повідомлення. Можливо, кожен з нас божеволіє хоча б раз, очікування зворотного зв'язку з

потенційними роботодавцями, що надсилають повідомлення в чаті, дуже важливо.

Захист данних

Конфіденційність важлива для будь-якого спілкування (ділового чи особистого). Дані користувача зазвичай зберігаються на сервері, який належить додатку, а повідомлення шифруються та передаються між пристроями за допомогою різних протоколів зв'язку.

Простого надсилання повідомлень та фотографій недостатньо, щоб завоювати велику аудиторію. Тільки завдяки спеціальним функціям, які ще не широко використовуються або повністю унікальні, ви можете запам'ятати та отримати завантаження. Наприклад:

- Секретний чат. Через деякий час повідомлення на обох пристроях буде видалено.
- Знімки екрана сповіщень - ще один варіант безпеки. Якщо хтось із одержувачів вирішить заскрінити ваше повідомлення, ви отримаєте повідомлення і будете продовжувати обережно ділитися файлами з цією особою.
- Можливість ділитися місцезнаходженням з друзями.
- Чат-боти. Особливо це стосується електронної комерції.
- Тимчасовий вміст, такий як статус чи історія.
- GIF як аватар.
- Можливість створювати власні наклейки.
- Відкладені повідомлення які прийдуть в час вказаний людиною.
- Ігри в месенджері.
- Інше.

1.3 **Месенджери: основні поняття, принципи роботи та особливості**

Миттєві повідомлення(англ. Instant messaging, скорочено IM) - Telecom Послуги для обміну текстовими повідомленнями між комп'ютерами або іншими комп'ютерами.

Користувач проходить через пристрій у комп’ютерній мережі (як правило, Інтернет). Зазвичай з самого початку це невеликі текстові повідомлення. Але з розвитком система також додає інші функції, такі як передача файлів, зображень, аудіосигнали та повідомлення, відео та спільні дії, такі як як живопис чи гра. Для використання цього типу зв'язку потрібна клієнтська програма. Клієнт обміну миттєвими повідомленнями зазвичай називають Інтернет-пейджером або Messenger.

Різниця між миттєвими повідомленнями та електронною поштою - доставка повідомлень відбувається в режимі реального часу.

Електронна пошта зберігається в поштовій скриньці на сервері. Для того, щоб отримати повідомлення, одержувач повинен особисто перевірити свою поштову скриньку та забрати її.

Система обміну миттєвими повідомленнями працює за певними протоколами. Протокол зв'язку сервер або немає сервера. Найпоширенішим є серверний протокол, коли месенджер не працює самостійно, а підключений до центру комп’ютер, який називається сервером у мережі обміну повідомленнями. і так месенджер ще називають клієнтом (клієнтською програмою).

2. Інформаційне забезпечення

2.1. Етапи проектування

Проектування - відповідальний процес, що вимагає знання законів суспільного розвитку. Воно не повинно обмежуватися (орієнтуватися) суб'єктивними бажаннями. Позбутися від однобічності, суб'єктивізму в проектуванні можна, тільки спираючись на наукові методи, до яких, перш за все, відноситься метод використання матриці ідей, коли на основі декількох незалежних змінних складаються різні варіанти рішень. Зазвичай розробка соціального проекту залежить від складності та першочерговості поставлених завдань, від термінів, в межах яких потрібно здійснити задум, а також від матеріальних, трудових і фінансових ресурсів. Прораховуючи варіанти на основі цих змінних, можна визначити найбільш ефективний шлях реалізації проекту в заданих умовах. Це важлива вимога застосовується, як правило, в умовах обмежених можливостей (в умовах так званої області свободи).

На етапі пошуку, обґрунтування проекту необхідно хоча б у найзагальнішому вигляді уявити той резерв і ресурс часу, які відводяться для його виконання. Одночасно уточнюється мета розробки соціального проекту, дається характеристика того стану, якого хотілося б досягти виходячи з вимог соціального прогресу.

Сформулювавши мету, приступають до етапу збору необхідної інформації, звертаючись до будь-яких джерел, включаючи і наукові дослідження. Пріоритет віддається тим відомостям, в яких зафіксовано сучасний рівень пізнання соціального процесу. Крім того, треба брати до уваги аргументи як "за", так і "проти" тих рішень, які є в світовій практиці.

Етап складання завдання на проектування передбачає використання отриманої інформації та визначення того, якими параметрами воно повинно

задовольняти (або відповідати), і тут велику роль відіграє вироблення принципово нових ідей, які розкривають шляхи ефективного досягнення намічених цілей. Формулювання завдання розглядається як частина концепції, яка може включати різні варіанти рішень. Краще, якщо ці варіанти представлені у вигляді матриці, що передбачає відбір найбільш прийнятних рішень з урахуванням зміни тих чи інших змінних: термінів, матеріальних, фінансових і трудових ресурсів, першочерговості та важливості реалізації окремих елементів проекту. Концепція проекту зазвичай перевіряється двояким чином: за допомогою теоретичного аналізу та за допомогою експерименту.

На заключній стадії проектування приймається рішення, що виступає у вигляді конкретної програми дій, яка в заданих параметрах намічає ті чи інші засоби досягнення поставленої мети при наявності певних обмежень. Рішення містить також можливі варіанти, терміни, основні етапи та послідовність операцій.

Загальний функціонал соціальної мережі:

- Профіль користувача
- Швидка авторизація
- E-commerce
- Рекламні інструменти
- Технічна підтримка
- Налаштування конфіденційності

2.2.Засоби проектування

2.2.1 Javascript

JavaScript - це мультипарадигменна мова програмування, яка зазвичай застосовується як вбудований інструмент для програмного доступу до різних об'єктів додатків. З погляду веб-розробки без знань цієї технології неможливо займатися створенням сучасних інтерактивних сайтів.

Мова JS – це те, що «оживляє» розмітку сторінок (HTML) та користувальницький функціонал (CMS) сайтів. За допомогою цієї мови реалізується можливість реакції сторінки або її елементів на дії відвідувача. Наразі JavaScript є базовою мовою програмування для браузерів. Він повністю сумісний із операційними системами Windows, Linux, Mac OS, а також усіма популярними мобільними платформами.

2.2.2 Платформа Node.js

Node.js – це система, яка виконує JavaScript окремо від вашого браузера. Можна сказати, що це самостійне середовище для виконання JavaScript. Node.js можна встановити на сервер (так само, як Python) і виконувати на ньому ваш код, віддаючи результат виконання користувачам. На ньому можна робити окремі програми, використовуючи додаткові фреймворки.

З Node простіше масштабуватись. При одночасному підключенні до сервера тисяч користувачів Node працює асинхронно, тобто ставить пріоритети та розподіляє ресурси грамотніше. Java ж, наприклад, виділяє на кожне підключення окремий потік.

Node.js має наступні властивості:

- асинхронна одно-ниткова модель виконання запитів;
- неблокуючий ввід/вивід;
- commoJS - система модулів;

- двигун Google V8;

2.2.3 Express.js

Express.js – це структура Node.js. Це найпопулярніший фреймворк на даний момент (найпопулярніший на NPM).

Він побудований навколо конфігурації та детальної простоти проміжного програмного забезпечення Connect. Деякі люди порівнюють Express.js з Ruby Sinatra порівняно з громіздким та самовпевненим Ruby на Rails.

Якщо вам не потрібно повторювати один і той самий код знову і знову. Node.js це низькорівневий механізм введення-виводу, який має модуль HTTP. Якщо ви просто використовуєте модуль HTTP, велика робота, така як аналіз корисного навантаження, файли cookie, зберігання сеансів (у пам'яті або Redis), вибір правильного шаблону маршруту на основі регулярних виразів, повинна бути повторно реалізована. Із Express.js він просто існує для вас.

2.2.4 База даних MongoDB

MongoDB - система управління базами даних, яка працює з документоорієнтованою моделлю даних. На відміну від реляційних СУБД, MongoDB не потребує таблиці, схеми або окремої мови запитів. Інформація зберігається як документ чи колекція.

Розробники позиціонують продукт як проміжну ланку між класичними СУБД та NoSQL. MongoDB не використовує схеми, як це роблять реляційні бази даних, що підвищує продуктивність усієї системи.

MongoDB застосовують як сховище у сфері машинного навчання та штучного інтелекту. MongoDB належить до класу NoSQL СУБД та працює з документами, а не із записами. Це кросплатформовий продукт, який легко впроваджується у будь-яку операційну систему. Ряд унікальних

особливостей дозволяє використовувати СУБД під певні завдання, у яких вона забезпечує максимальну продуктивність та надійність.

2.2.5 Websocket

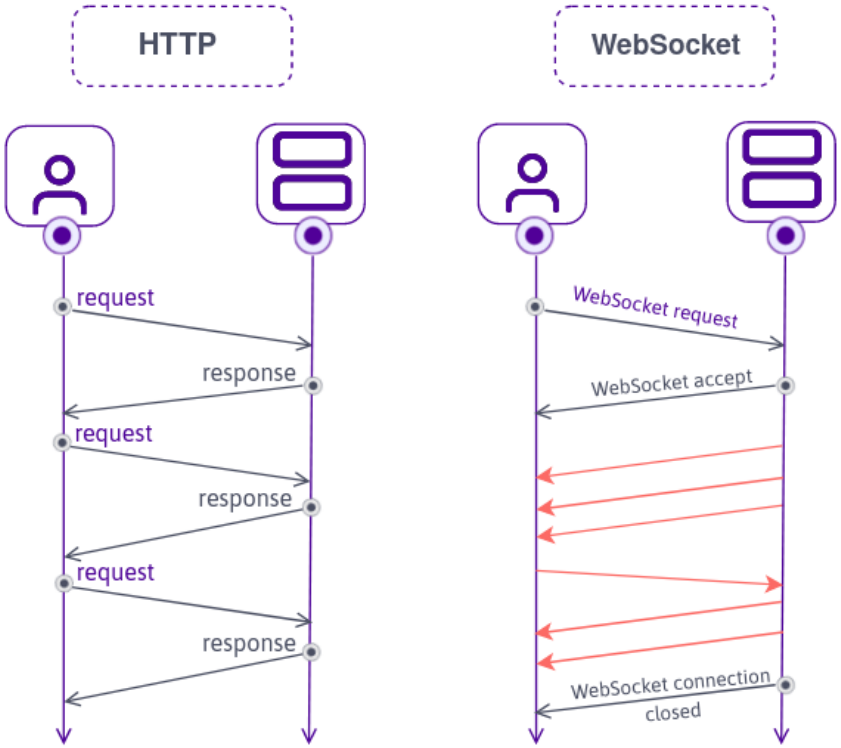
WebSocket - це найбільш ретельне розширення з моменту зародження протоколу HTTP. Це зміна в парадигмі HTTP. Спочатку синхронний протокол, заснований на моделі «запит-відповідь», став повністю асинхронним та симетричним. Немає фіксованих ролей клієнта та сервера, але є два рівних учасника обміну даними. Кожен працює самостійно і при необхідності надсилає дані іншим. Відправлено-продовжуйте, нічого не чекайте. Інша людина відповість у разі потреби - може відповісти не відразу, або взагалі не відповість. Протокол забезпечує повну свободу обміну даними, і ви можете вирішити, як ним користуватися.

2.3. Використання сокетів

Websocket - це протокол, який забезпечує двосторонній канал зв'язку між браузером та веб-сервером. Різниця між ним та звичайним способом передачі даних полягає в тому, що з'єднання не буде розірвано після отримання відповіді, але залишиться відкритим. Отже, інформацією можна обмінюватися в режимі реального часу, особливо ви можете отримувати повідомлення від сервера без додаткових запитів.

Коли і де використовувати веб-сокети:

- Соціальні мережі, включаючи чати та сповіщення про дії інших користувачів в мережі (лайки, коментарі, публікації тощо)
- Фінансові повідомлення або інші повідомлення, важливі для інформації. Під час роботи з документами важливо знати, які зміни в даний час вносить інша сторона.
- Програми, де важливі поточні дані про місцезнаходження. Одним із важливих критеріїв онлайн-освіти є зв’язок в Інтернеті між студентами та викладачами чи студентами.
- В онлайн-іграх дуже важлива взаємодія з гравцями онлайн.



2.3.1 Структура веб-сокета

Більшість веб-сокетів реалізовані за допомогою веб-сервера Node.js та Tornado. Це тому, що php розроблений як мова програмування веб-сторінок, які прийняли формат відповіді на запит. Однак бібліотека Ratchet дозволяє обійти ці обмеження та використовувати php для реалізації серверної підтримки Websockets.

Npm дозволяє нам дуже швидко встановлювати пакети з одним рядком коду, тому спершу перейдіть до свого каталогу та завантажте необхідні пакети npm:

- npm install express socket.io

Тепер ми можемо розпочати побудову контролера сервера для обслуговування домашньої сторінки. Весь код сервера ми зберігаємо у файлі "server.js", який буде виконуватися Node.js.

3 Математичне забезпечення

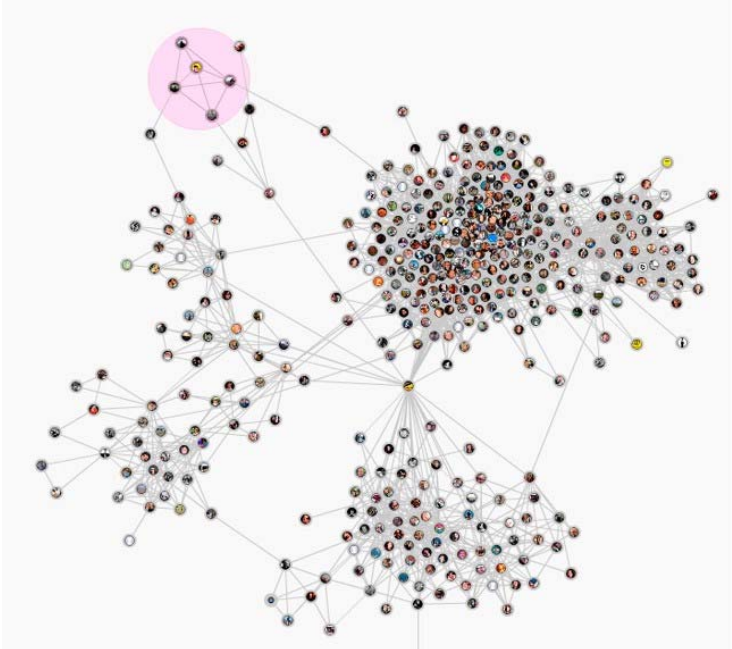
3.1.Графи в комп’ютерному моделюванні

Теорія графів — розділ дискретної математики, що вивчає властивості графів. Як правило, граф представляється як набір вершин (вузлів), з’єднаних ребрами.

Наприклад, теорія графів використовується в геоінформаційних системах. Існуючі або новостворені будівлі, споруди, населені пункти тощо розглядаються як гори, а дороги, інженерні комунікації, лінії електропередач тощо – як краї.

Використовуючи різні розрахунки, виконані на карті цього типу, наприклад, ви можете знайти найкоротший об’їзд або найближчий продуктовий магазин, щоб спланувати найкращий маршрут.

Під час використання графів для опису графів найчастіше використовується система символів наступна: вершини графів представлені точками, прямокутниками та еліпсами, а вміст вершин відображається всередині графів (граф потоку алгоритму діаграма).

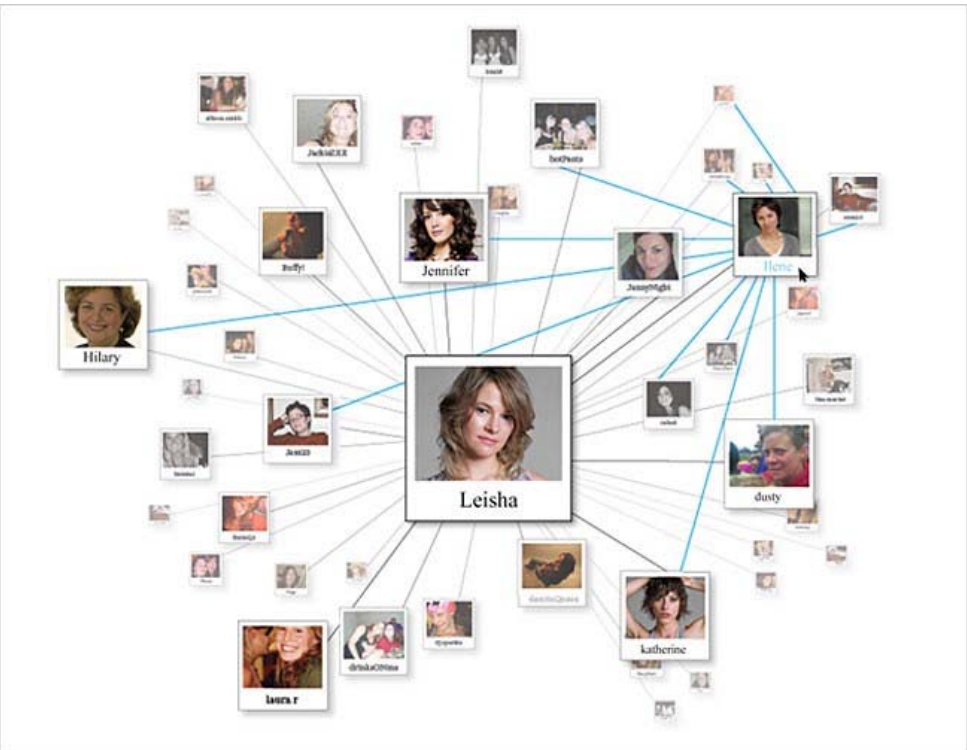


3.1.1 Зображення графа

Якщо між вершинами є ребра, то відповідні точки (графи) з'єднуються відрізками або дугами. У разі орієнтованого графа дуга замінюється стрілкою або чітко вказує напрямок ребра. Зображення фігури не слід плутати з самою фігурою (абстрактною структурою), оскільки одну фігуру можна порівняти з кількома графічними зображеннями. Це зображення використовується лише для того, щоб показати, які пари вершин з'єднані ребрами, а які ні.

Соціальний граф

Соціальний граф — це граф, вузли якого представлені соціальними об'єктами, такими як профілі користувачів із різними атрибутами (наприклад, ім'я, день народження, рідне місто тощо), спільноти, засоби масової інформації, вміст тощо, а також межі соціальних зв'язків. між ними



3.1.2 Соціальний граф

Неявні соціальні графи — це графи, які можна сформувати на основі взаємодії користувача з його групами «друзів» і «друзів» у соціальних

мережах. У цій колонці, на відміну від звичайних соціальних графів, «друзі» чітко не вказуються, тобто немає чітких соціальних відносин.

Задачі за допомогою соціальних графів:

-Ідентифікація користувача;

-Соціальний пошук;

-Створення рекомендацій щодо вибору «друзів», медіаконтенту та новин;

-Визначення "реальних" зв'язків або збирання відкритої інформації для графічного моделювання.

Обробка даних соціальних графів пов’язана з багатьма проблемами, такими як відмінності в соціальних мережах та конфіденційність соціальних даних.

Граф інтересів

Графінтересів — це онлайн-подання інтересів конкретної особи, яке виходить з урахуванням його діяльності в соціальній мережі. Вершини графа є фіксацією особистості, або це може бути особиста інформація людини в соціальній мережі, а краї графа відображають зв’язок між вершинами графа.

3.1.3 Граф інтересів

За допомогою графів інтересів можна зрозуміти, чим людина хоче займатися, що купити, куди хоче піти, кого може побачити, новини про те, кого вона цікавить, за кого вона збирається голосувати. Графи інтересів тісно пов’язані з соціальними графами, але вони не однакові.

Граф інтересів використовується для створення мережі інтересів людей. Хоча Facebook та інші соціальні мережі організовані навколо друзів людини (тобто навколо соціального графа), мережі хобі створюються навколо інтересів особи (тобто графа інтересів). Подібно до того, як соціальний граф є картою відносин між особою та людьми, які «слідують» за ними в Інтернеті, граф інтересів також є зв’язком із онлайн-інтересами особи.

Таким чином, захоплення людини, представлені у вигляді графа інтересів, забезпечують засоби для подальшої персоналізації веб-простору, що ґрунтується на перетині графа інтересів з веб-контентом. Граф інтересів або мережа інтересів у деяких випадках можуть бути отримані із соціального графа або соціальної мережі та можуть підтримувати та оновлювати зв'язки між вершинами на основі цієї соціальної мережі.

Граф інтересів має бути точним і виразним, він повинен брати до уваги явно оголошені інтереси, наприклад, «Like» на Facebook або «інтереси» у профілі на LinkedIn, а також неявні інтереси, виведені на основі активності користувача, наприклад, такі як клацання мишею , коментарі, теги до фото та чек-іни. Соціальні мережі найчастіше є джерелом цієї інформації.

3.2.Шифрування повідомлень

У багатьох сучасних месенджерах є шифрування як end-to-end. Це означає, що повідомлення шифруються на одному пристрої і через сервер надсилаються іншій людині, яка може їх розшифрувати. Крім відправника та одержувача, прочитати таке повідомлення ніхто не може, тому що вони використовують стійке симетричне шифрування. Простіше кажучи,повідомлення шифрується якимось ключем і розшифровується таким самим ключем.

Цей спосіб має один мінус — треба якось заздалегідь обмінятися спільним ключем. Для вирішення цієї проблеми використовують асиметричне шифрування. Воно дозволяє виробити спільний ключ і передати його у зашифрованому вигляді кожній стороні навіть за умов прослуховування каналу передачі.

Головна задача месенджера при першому зверненні – отримати спільний секретний ключ. Це можна зробити за допомогою протоколу Діффі-Хеллмана.

Принцип роботи цього протоколу Вітфілд Діффі та Мартін Хеллман опублікували в 1976 році, і з того часу він лежить в основі всіх подібних алгоритмів отримання загального секретного ключа.

Допустимо, Користувач1 хоче таємно переписуватися з Користувачем2, але вони можуть користуватися лише незахищеним каналом зв'язку. Це означає, що все, що вони відправляють може перехопити зловмисник і спробувати розшифрувати. У таких умовах вони не можуть просто надіслати один одному секретний ключ, бо його одразу дізнається той, хто за ними стежить.

У цьому випадку Користувач1 каже Користувачу2: придумай собі велике просте число і нікому його не кажи, а я поки що придумаю своє. Які

числа виберуть Користувач1 та Користувач2 — ніхто не знає, бо вони їх не пересилають одне одному, а просто домовляються про вибір. Це буде для кожного свій секретний ключ, який вони використовують для асиметричного шифрування, a – для Користувача1 та b – для Користувача2.

Коли обидва вибрали собі ключ, Користувач1 каже: Користувач2, ось два простих числа, P і G, працюватимемо з ними. У цей момент той, хто прослуховує канал, може отримати ці цифри, але це йому нічого не дасть.

Коли Користувач2 отримав ці числа, Користувач1 каже до Користувача2:

- Зведи G у ступінь свого секретного числа b
- Знайди залишок від ділення цього результату за модулем P

  Gb mod P = B ← ось це число Користувач2 і відправляє Користувачу1.

Користувач1 одночасно з Користувачем2робить те саме зі своїм секретним числом і відправляє результат Користувачу2:

Ga mod P = A ← ось це число Користувач1 відправляє Користувачу2.

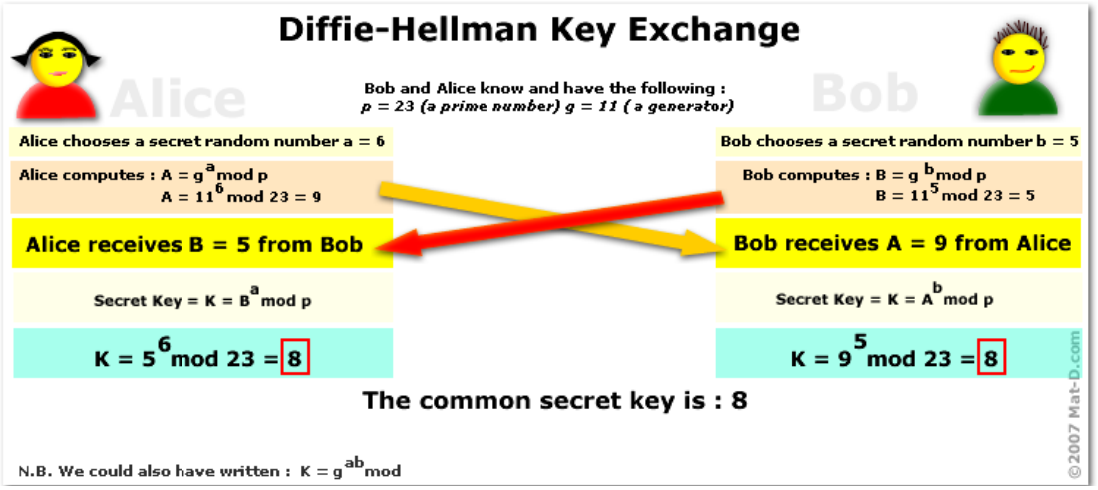
Коли Користувач1 і Користувач2 отримали один від одного їх залишки від ділення по модулю, вони застосовують ту саму операцію до цього залишку. Навіть якщо хтось перехопить ці залишки від ділення, він не відновить секретні ключікористувачів.

У результаті обидва співрозмовники одержують одне загальне число, яке можна використовувати як загальний ключ для симетричного шифрування. Як у них виходить однакове число – стежте за математичною магією:

- Користувач1 бере результат Користувача2 B і так: Ba mod P = K.
- Але Ba mod P = Gba mod P → запам'ятаємо це.

- Користувач2 бере результат Користувача1 і робить так: Ab mod P = K.
- Але Ab mod P = Gab mod P → запам'ятаємо це.

Від перестановки множників результат не змінюється, а обох випадках ми підносимо G до ступеня ab, тому у користувачів виходить у результаті те саме число. Це і є їхній спільний ключ для симетричного шифрування, яким вони тепер можуть користуватися.



3.2.1 Зображення алгоритму Діффі-Хелмана

Секрет в тому, що без секретних ключів користувачів хтось інший не зможе обчислити цей ключ, навіть якщо перехопить усі повідомлення із проміжними результатами. А все тому, що ніхто з них своїх секретних ключів нікуди не передавав.

Сам алгоритм необхідний лише для створення пари ключів. Якщо хтось посередині все підслуховуватиме, він може перехопити всі проміжні результати і підмінити їх своїми. У результаті користувачі встановлять проміжний секретний зв'язок через зловмисника, який читатиме все їхнє листування, а користувачі нічого не помітять.

Для запобіганню перехопленню повідомлень зловмисником використовують алгоритми автентифікації та перевірки цифрового підпису.

Коротко: є спеціальні алгоритми, які дозволяють перевірити, чи цей ключ відправив Користувач1, а не хтось інший замість нього.

4. Програмне забезпечення

4.1 Загальна структура мережі

UNFUgram це сучасна соціальна мережа, яка використовує сучасні підходи і фреймоврки, яка постійно покращується і розширюється з оновленнямм нових технологій.

UNFUgramстворений на основі відомої мережі Instagram. Реалізовано основні функції притаманні соціальним мережам, а саме:

- Авторизація:

Реалізована за домопогою MERN стеку, запис даних відбувається в базу даних MongoDB. Швидка і зручна реєстрація.

- Профіль користувача:

Після процесу логінації користувач може авторизуватись у системі. Користувач може побачити профіль в мережі, оскільки він рахується новим учасником мережі, слід заповнити деякі особисті дані (аватарка, статус та інше). Стрічка новин, рекомендації друзів будуть недоступні, оскільки потрібно підписатись на оновлення інших користувачів.

- Налаштування профіля:

В цих налаштуваннях кожен користувач можете налаштувати особисті дані, також дані які відображаються на сторінці користувача. Можливість зміни імені, паролю та подібного.

Головна мета створення цієї соціальної мережі, це згрупувати людей в одну так би мовити групу, з можливістю обміну новинами, постами, інформацією, повідомленнями та багато іншого. Кожного дня технології розвиваються, створюється щось нове, корисне, і відповідно до того, ця соціальна мережа також розширюється, збільшується функціонал, додаються нові можливості.

4.2 Результати проектування

Розроблений месенджер для соціальної мережі включає наступні елементи:

4.2.1 Діалогове вікно приватних повідомлень

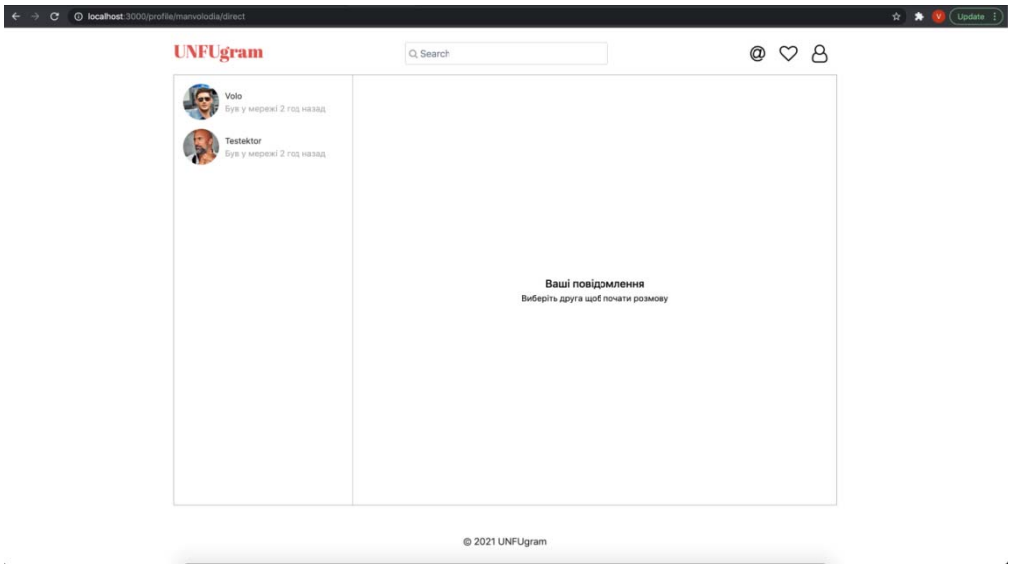


Рисунок 4.2.1.1 Діалогове вікно

Після переходу в direct можна вибрати друзів(видно список людей за яким стежиш) і почати спілкування

Після вибору друга, можна надсилати повідомлення, ділитися відео і тому подібне.

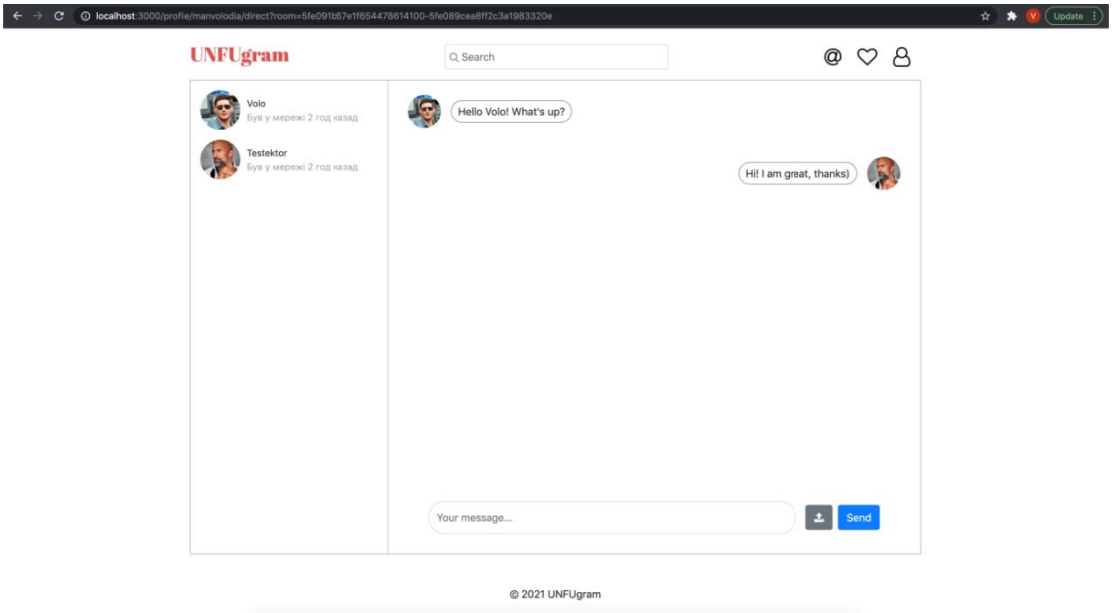


Рисунок 4.2.1.2Чат з другом

В полі для вводу повідомлення пишемо свій текст і надсилаємо користувачу

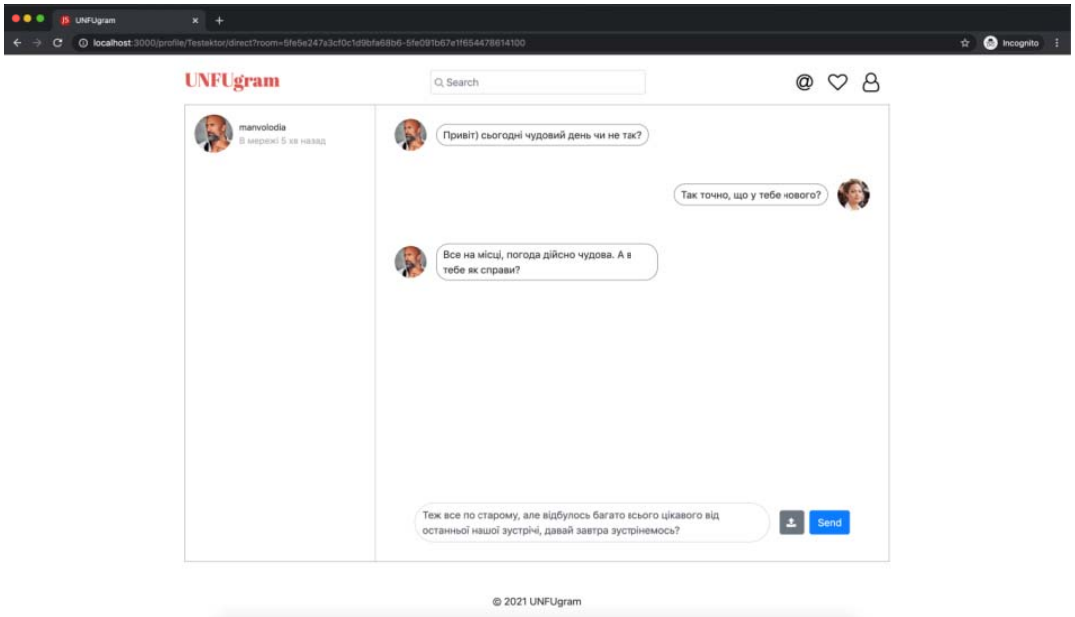


Рисунок 4.2.1.3 Обмін повідомлень з другом

і якщо цей користувач є зараз в мережі то він отримає push-сповіщення про отримання нового повідомлення

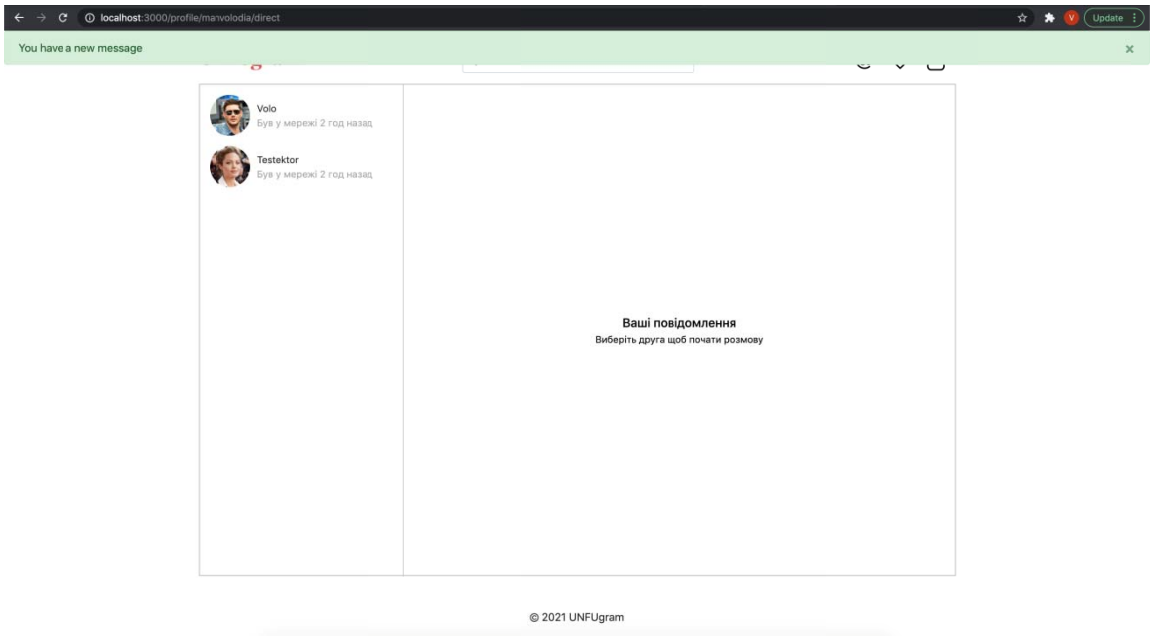


Рисунок 4.2.1.4 Сповіщення про повідомлення

Нажавши на відповідну кнопку, можемо вибрати файл з комп’ютера

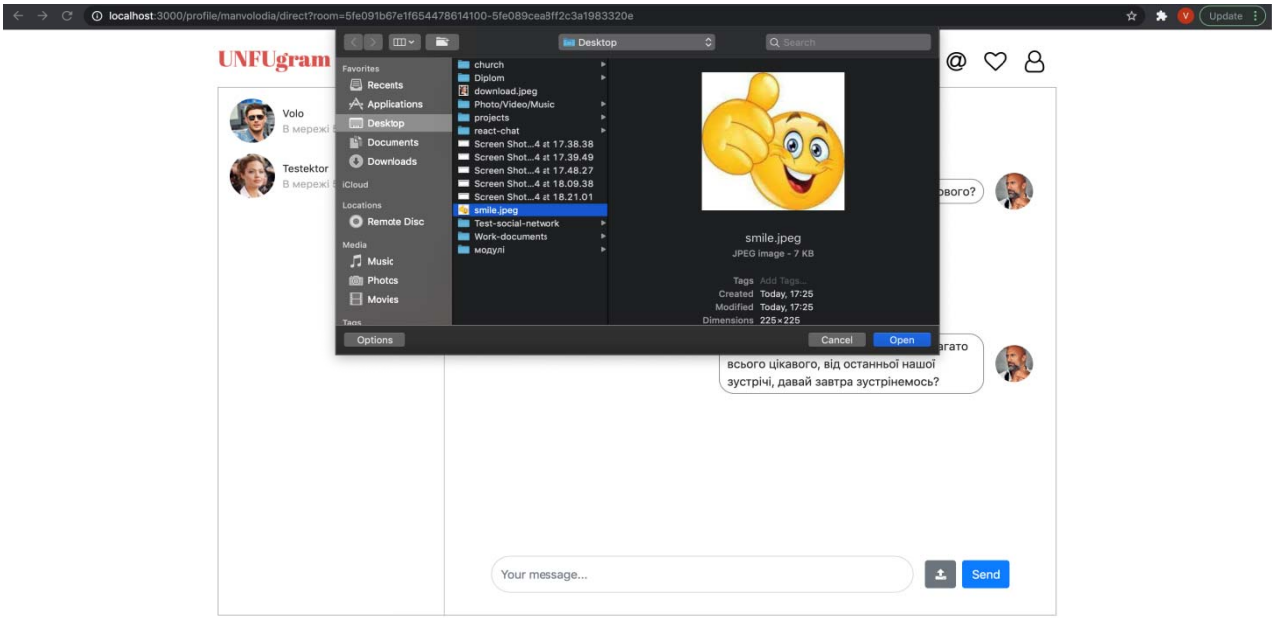


Рисунок 4.2.1.5 Вибір файлу

Відображається файл як і звичайні повідомлення

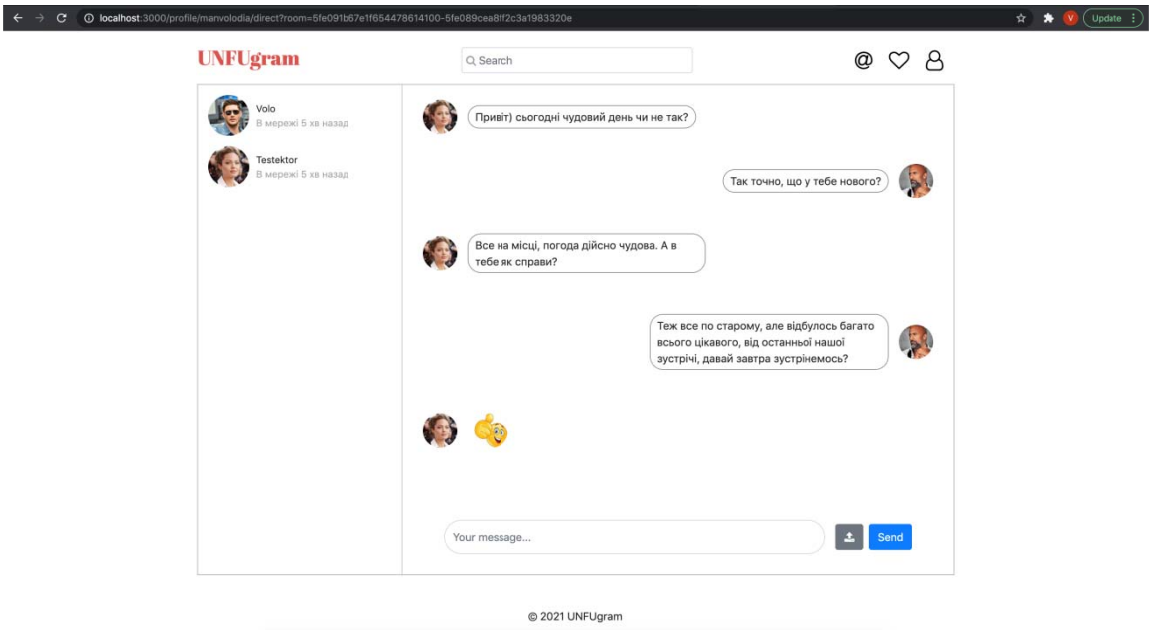


Рисунок 4.2.1.6 Відображення файла

Також крім звичайної картинки можна надсилати відео, гіфку тощо. Завантаження відбувається так само як і з картинками, у разі великої кількості повідомлень з’являється скрол для прокручування всієї історії повідомлень

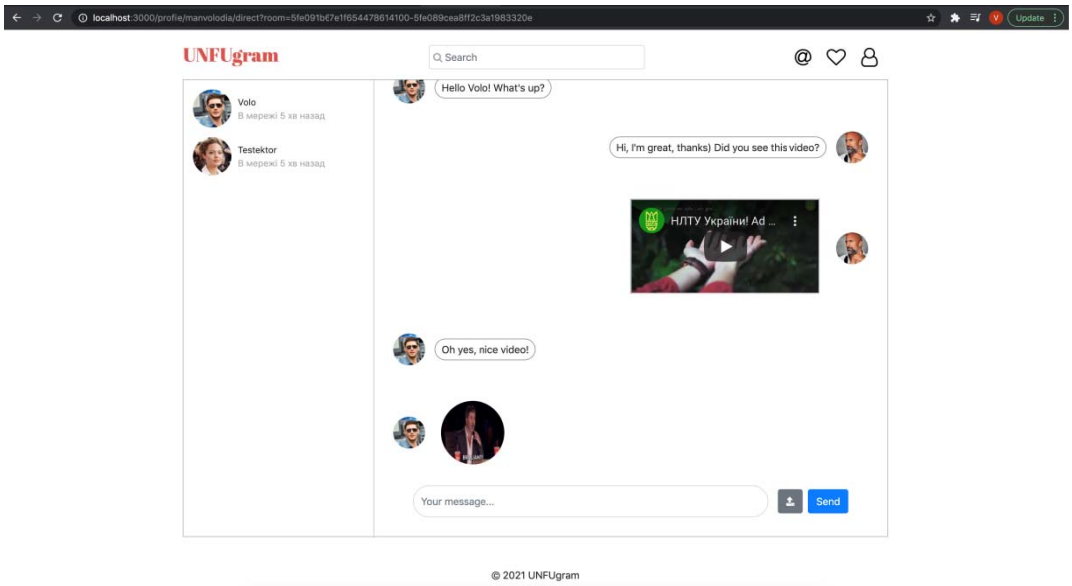


Рисунок 4.2.1.7 Відображення відео, гіфки

Після написання повідомлення, його можна редагувати, виправити помилки в слова або ж взагалі видалити з діалогу

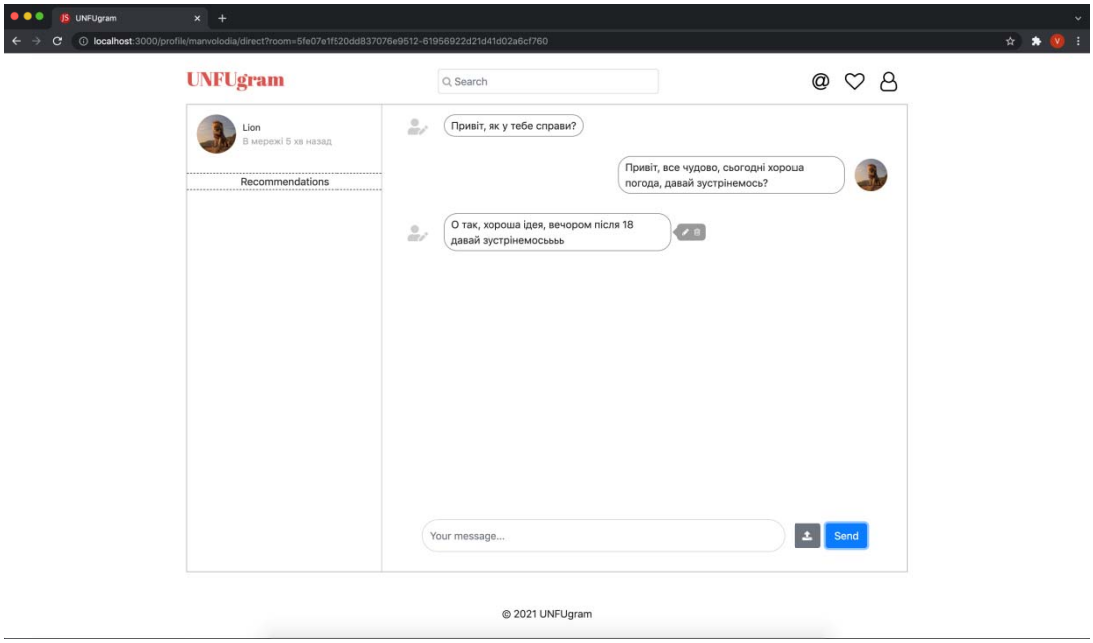


Рисунок 4.2.1.8 Tooltipредагування/видалення власного повідомлення

Після натискання на олівчик, повідомлення з’являється в текстовому полі

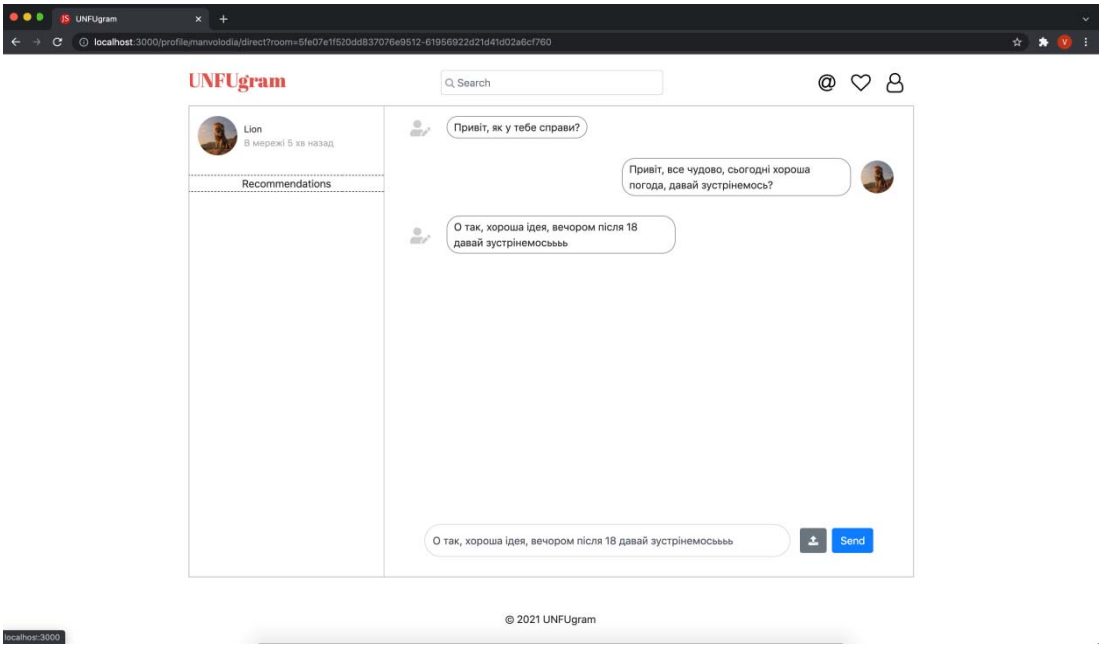


Рисунок 4.2.1.9 Вигляд для редагування повідомлення

Після закінчення редагування і нажимання кнопки бачимо змінене повідомлення

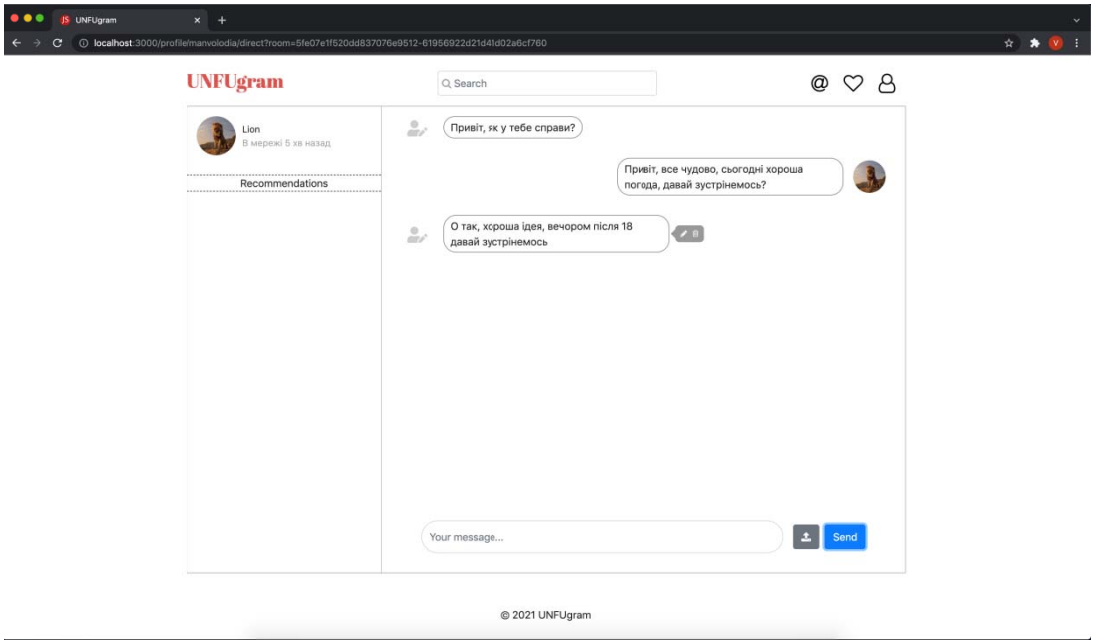


Рисунок 4.2.1.10 Вигляд редагованого повідомлення

Вище на малюнку також в нашому тултіпі бачимо корзинку, після нажимання на яку, повідомлення буде видалено

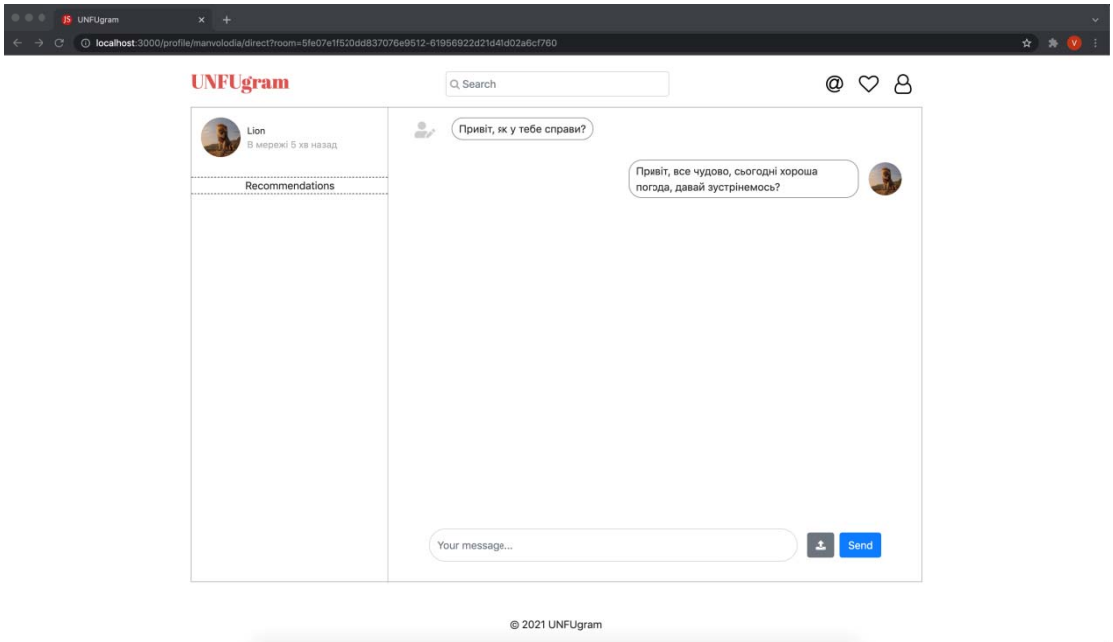


Рисунок 4.2.1.11 Видалення повідомлення

Авторизувавшись як адміністратор, маємо можливість видалити користувача з мережі

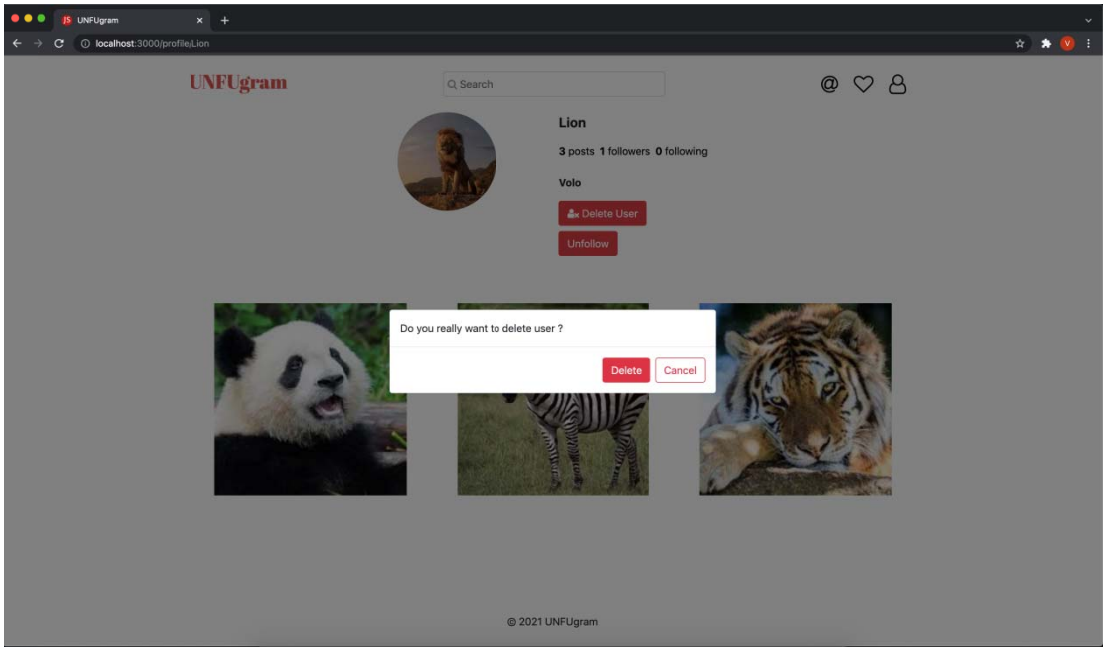


Рисунок 4.2.1.11 Видалення користувача

Крім цього як адміністратор ми маємо можливість видаляти пости будь-яких користувачів, а також видаляти коментарі до постів

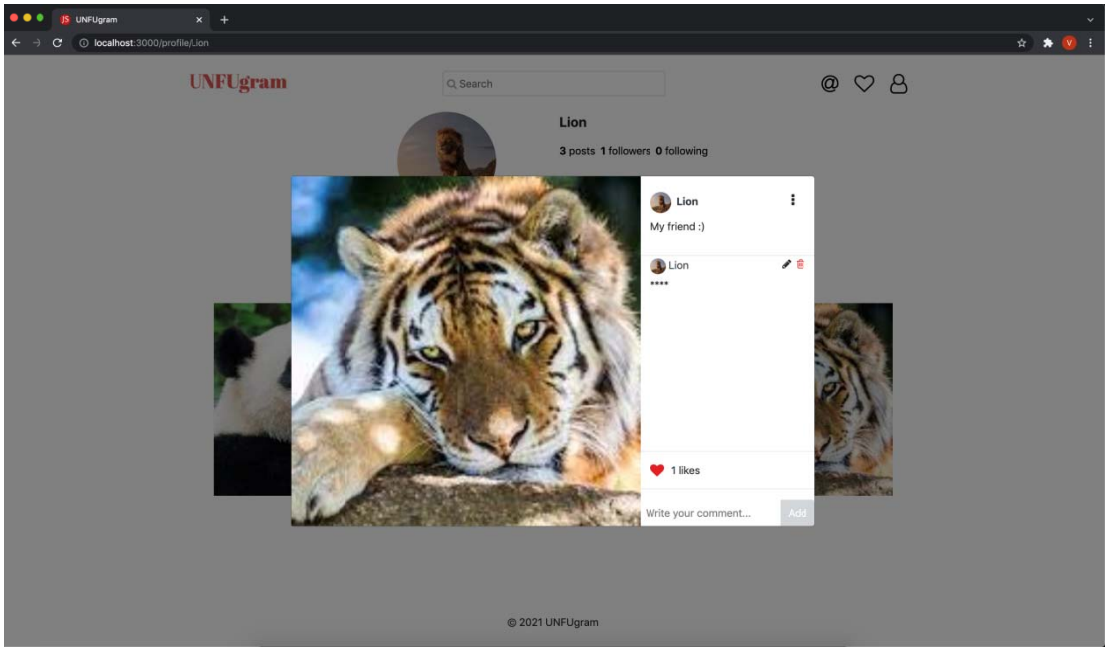


Рисунок 4.2.1.12 Видалення коментаря

Нажавши на бургер меню (3 вертикальні крапочки) маємо можливість видалити пост чи змінити його вміст

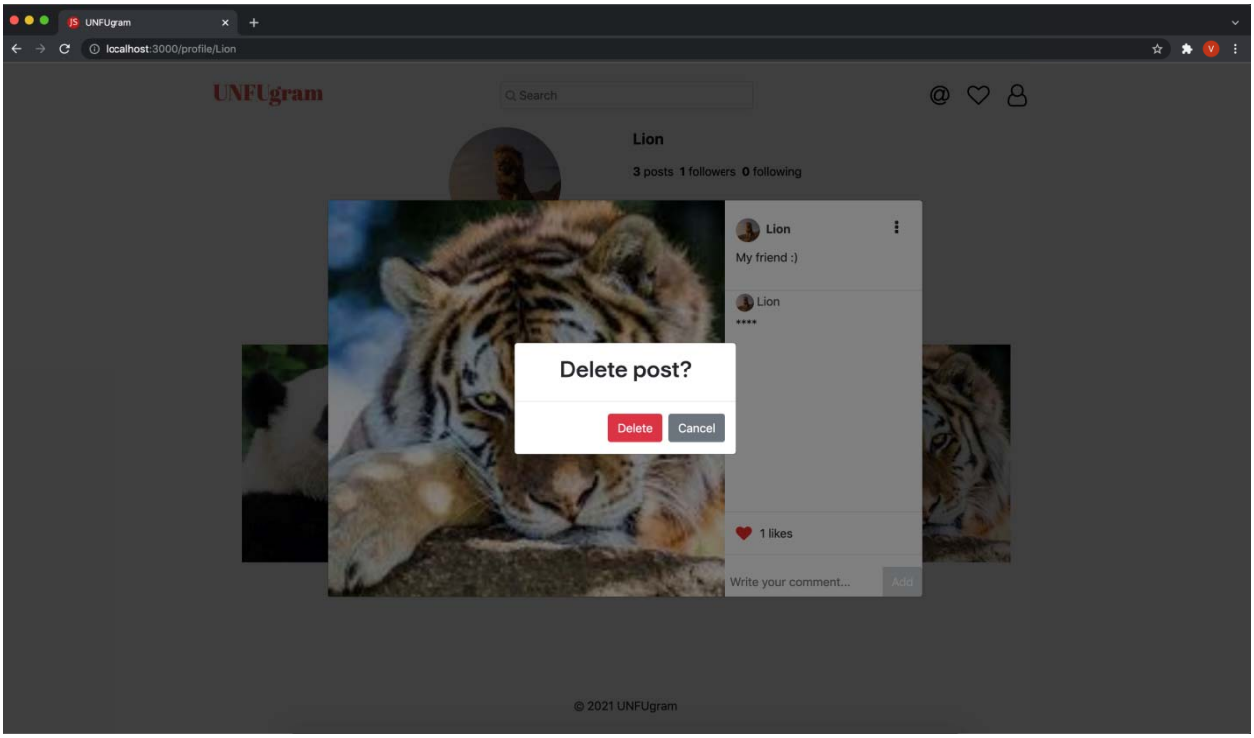


Рисунок 4.2.1.13 Видалення поста

5. Розроблення стартап-проекту

5.1 Опис ідеї стартапу

Сьогодні ідея створення чогось власного не виглядає неможливою. Але створити насправді хороший продукт, який буде подобатись людям і буде дійсно корисним для інших є досить не простою задачею. Слід добре розуміти з чого потрібно почати і що потрібно робити, також розуміти, що досягти успіху не вийде за один день і не зупинятись після невдалих спроб.

Тому перед початком розголошення про проект як новий стартап, потрібно виконати деякі дії. Основним кроком є створення спеціальної інформаційної карти про проект. Ця карта містить всі необхідні та короткі відомості про проект, а саме «UNFUgram» - соціальна мережа для спілкування з друзями, роботи бізнес-підприємства, діленням цікавої інформації та інше. Сьогодні можна знайти багато соцільних мереж чи то просто зручних месенджерів які будуть виконувати ті ж самі дії, але чи не було б зручно користуватись лиш однієї мережею для усіх потреб які виникають, не відчуваючи жодного дискомфорту. Хотілось б додати що створення стартапу це насамперед ідея, а вже далі реалізації цієї ідеї, але для успішної робити потрібне і одне і друге звичайно. Нижче опишу короткі відомості про проект.

1. Назва номінації - Web application (Інтернет-застосунок)
2. Назва проекту – UNFUgramсоціальна мережа для віддаленої роботи бізнес- підприємства
3. Назва ВНЗ, факультету, спеціальності - НЛТУ, кафедра інформаційних технологій, комп’ютерні науки
4. Автор – Ревура Володимир Іванович

5. Цілі і задачі проекту – розроблена мережа має виконувати наступні функції:
   - створення, видалення, редагування постів, коментарів
   - створення, видалення, редагування приватних повідомлень
   - можливість отримувати пуш сповіщення на деякі дії всередині мережі
   - заповнення, а також можливість змінення особистої інформації
   - рекомендації друзів, пошук людей по іменах, пошук хештегів
   - мережа повинна працювати на будь-яких пристроях безперебійно, дизайн динамічний і під будь які екрани буде виглядати відповідно(контент не буде вилазити за межі видимості, тексти, блоки, картинки і інше буде зменшуватись чи переміщатись для кращого вигляду)
   - та інше
6. Короткий зміст - проект розроблявся як зручний спосіб комунікації між людьми, особливо сьогодні без цього дуже важко уявити наше життя. Швидкі повідомлення, вчасні сповіщення про отримання нових повідомлень, це все дає змогу в реальному часі спілкуватись з тієї чи іншою людиною. Загалом проект націлений на людей різного віку, від молоді і старших, кожен може користуватись мережею. Людина може просувати якусь продукцію, тобто свого роду рекламу і завдяки великій кількості користувачів багато хто зможе її побачити і можливо навіть когось зацікавить. Тому це потужний інструмент для поширення власних ідей, інформації чи чогось іншого, обмінну повідомленнями та багато іншого.
7. Термін виконання проекту –6 місяців

5.2Розроблення ринкової стратегії

В цьому пункті сформовано декілька стратегій, як правильно використовувати не лише програмне забезпечення, а й особистісні підходи до рекламодавців, клієнтів та, власне, цільову аудиторію, для якої і подається реклама.

Першим кроком для успішної реалізації проекту – це створення впізнаваного бренду. Немає значення – це звичайна маленька мережа чи велика платформа – такий сайт повинен стати популярним. Спершу потрібно інвестувати в рекламу мережі. Подбати про його хороші риси, які спершу, звісно, виробити. Ці риси подавати ненав'язливо, адже люди не люблять одноманітності та нав'язування. Потрібно про свої перспективні сторони бізнесу натякати, щоб клієнт сам зробив відповідні висновки. Розповісти про переваги чому слід користуватись саме моєю мережею, що в ній особливого і чим вона краща за інші. Така методика вплине на користувачів набагато ефективніше, адже так тоді він гадатиме, що сам дійшов до такого рішення.

Після того, як про ваш бренд дізнаються, слід подбати про дизайн, перше враження користувача так би мовити. Людям необхідно відчувати себе в зоні комфорту – тоді вони повертатимуться знову. Відвідавши один раз сайт людина зразу зможе зробити висновок, тому дуже важливо, щоб перше враження було позитивне, щоб користування було зручним і зрозумілим і головне, щоб все виглядало пристойно. Початкова загрузка сайту не повинна бути повільною, користувач не хоче довго чекати і є ймовірність, що він просто закриє вкладу. Також такі речі як реєстрація, створення профілю і тому подібне не повинно забирати в людей багато часу, все має швидко і зрозуміло, це допоможе зберегти більше людей і залучити нових

Зацікавивши людей відвідувати мережу знову і знову можна переходити до наступного етапу – залучення рекламодавців. Як правило, при стрімкому розвитку бізнесу рекламодавці самі починають виходити на зв’язок, аби рекламувати свої продукти використовуючи вашу платформу.

Важливо пам’ятати, що все має свої обмеження, особливо реклама. Мало хто з користувачів любить рекламу, тому вони найменше хочуть її бачити. Тому не варто зловживати. Не завжди потрібно показувати рекламу повністю. Звісно, сама реклама – це власна справа рекламодавця, але спробуйте вибрати більш цікаві варіанти. Іншими словами, слухаючи рекламу, у вас самого має виникати бажання стати клієнтом бізнесу рекламодавця. Важливо, щоб реклама не була одноманітною, адже вона швидко набридне, викличе лише дискомфорт і злість людей, що лише відразу призведе до продукту рекламодавця, що суперечить меті апріорної реклами. Останній етап – це аналіз, який насправді є статистикою. Ця статистика важлива не тільки для рекламодавців, а й для вас. Він показує, які рекламодавці не гідні розміщувати свої оголошення у вашій організації, а інші, навпаки – потрібно заохочувати, а при правильній аргументації ще й отримувати додаткові бонуси за успішність розміщення реклами продукту у вашому закладі.

Отже, дотримуючись усіх вищенаведених стратегій ведення бізнесу, вибірки рекламодавців і реклами,також поступове виконання необхідних кроків дозволить успішно отримувати дохід з реклами, що буде сприяти появі нових користувачів

5.3Розробка маркетингової програми

Першочерговим завданням маркетингу є досягнення прибутку та його зростання за рахунок вашого бізнесу. В нашому випадку – це соціальна мережа. Перші кроки для цього були озвучені в пункті 5.2, проте в таблиці 1 сформовано більш точний опис.

Таблиця 5.3. Визначення переваг проекту

№ п/п	Потреба	Вигода проекту	Ключові переваги перед конкурентами
1	Простий та гнучкий сучасний інтерфейс	Зрозумілий, кросплатформовий інтерфейс, з яким зручно взаємодіяти	Підтримка будь-яких пристроїв
2	Повідомлення/ сповіщення	Можливість вчасно отримувати інформацію	Хороший і зручний функціонал і інтерфейс для обмінну повідомленнями, вчасні сповіщення
3	Розіграші/Бонуси	Залучення більшої кількості аудиторії	Постійні розіграші певних предметів, сертифікатів і тому подібне

Процес отримання доходу від реклами (монетизація) – буде чинником грошового обігу в умовах використання стартапу. Отримання коштів від рекламодавців обумовлене власне розміщенням їхньої реклами всередині вашої мережі. Зручність, зрозумілість та доступність є основними перевагами над конкурентами. Завдяки простоті користувачі будуть задоволенні цією мережею і будуть запрошувати друзів приєднатись, а радість в людей, хіба ж це не важливо, приносити якусь користь чи задоволення власним продуктом.

5.4Вимоги до технічного та програмного забезпечення

Для відображення та користування даною соціальною мережею достатньо зайти у будь-який браузер(Google Chrome, Mozilla, Safari, etc.)

Від швидкості інтернету залежить швидкість самої мережі і здійснення запитів на бек-енд і отримання відповідей на клієнтській стороні. Тому для стабільної і безперебійної роботи, слід мати хороше підключення до інтернету.

Крім цього користуватись UNFUgram-ом можна навіть і з телефону, оскільки гнучкий дизайн дає змогу переглядати мережу на різних пристроях(мінімальна підтримка для пристроїв з екраном 360px).

Висновки

Протягом розробки даної соціальної мережі для бізнес підприємства були застосовані сучасні підходи і технології для досягнення даної мети. Також протягом усього процесу окремі файли були покриті тестами для кращого тестування і уникнення помилок в майбутньому. Для цього були використано звязку Jest і Enzyme. Деякі файли були переписані на Typescript що дає значну перевагу у уникненні багатьох помилок чи їх вияву на ранньому етапі.

Враховуючи все вище мною описане, технології, підходи та інше було створено UNFUgram. Він багатий функціональністю, але як і будь-який інший додаток чи соціальну мережу можна вдосконали новими фічами, що можна сказати і про цю мережу теж.

Також можна виділити, що за допомогою сучасного протоколу Websocketбуло імплементовано месенджер для кращого обміну інформацією, файлами, без чого зараз дуже важко обійтись, особливо в час пандемії дуже зручно спілкуватись не виходячи і з дому, ділитись інформацією і працювати над вирішенням спільних проблем.

Список використаної літератури

1. HTML and CSS: Design and Build Websites by Jon Duckett
2. Complete Concepts and Techniques by Gary B. Shelly
3. Eloquent JavaScript: A Modern Introduction to Programming by Marjin Haverbeke
4. You don’t know JS by Kyle Simpson
5. The Definitive Guide by David Flanagan
6. https://uk.reactjs.org/docs/getting-started.html
7. https://docs.mongodb.com/manual/
8. https://nodejs.org/uk/docs/
9. https://redux.js.org/introduction/getting-started
10. https://knowledge.allbest.ua/programming/3c0b65625b3bd69b5d53b89421206d37
11. Ревура В., Процах Н.П.Розробка соціальної мережі «CloneInstagram» з використанням React, Redux технологій// Комп’ютерне моделювання та інформаційні технології: матеріали другої науково-практичної конференції студентів, аспірантів та молодих вчених. – Львів: кафедра інформаційних технологій НЛТУ України, 2020. – С. 68-69
12. Ревура В., Процах Н.П.Розроблення соціальної мережі для віддаленої роботи бізнес підприємства// Комп’ютерне моделювання та інформаційні технології: матеріали третьої науково-практичної конференції студентів, аспірантів та молодих вчених (Львів, 14-16 жовтня 2021 р.). – Львів: кафедра інформаційних технологій НЛТУ України, 2021. – С. 67-69

Додатки

Реєстрація

RegisterContainer.tsx

```
import React from 'react';
import Register from '../../components/Register';
import { reduxForm } from 'redux-form';
import { connect } from 'react-redux';
import * as action from '../../store/register/actions';
import validate from '../../utils/validation';
import { FormProps } from 'reactstrap';
import { IUser } from '../../store/commonInterfaces/commonInterfaces';

const RegisterContainer = ({ registerUser, handleSubmit, submitting, invalid
}: FormProps): JSX.Element => {
    const onSubmit = (user: IUser): void => {
        registerUser(user);
    };

    return (
<Register
            handleSubmit={handleSubmit}
            onSubmit={onSubmit}
            submitting={submitting}
            invalid={invalid}
        />
    );
};

export default connect(
    null,
    {registerUser: action.registerUser},
)(
    reduxForm({
        form: 'registerForm',
        validate,
    })(RegisterContainer),
);
```

Register.ts

```
import { showAlert } from '../alert/actions';
import {API} from '../api';
import { reset } from 'redux-form';
import { Dispatch } from 'redux';
import { IUser } from '../commonInterfaces/commonInterfaces';

export const registerUser = (user: IUser): (dispatch: Dispatch) =>
Promise<void> =>
    async (dispatch: Dispatch): Promise<void> => {
        try {
            const res = await API.post('/user', user);
            dispatch(showAlert(res.data.status, 'success'));
            dispatch(reset('registerForm'));
```

```
        } catch (e) {
            dispatch(showAlert(e.response.data.message, 'danger'));
        }
    };
```

Логінація

LoginContainer.ts

```
import React from 'react';
import Login from '../../components/Login';
import { reduxForm } from 'redux-form';
import { connect } from 'react-redux';
import { loginUser } from '../../store/login/actions';
import validate from '../../utils/validation';
import { FormProps } from 'reactstrap';
import { IUser } from '../../store/commonInterfaces/commonInterfaces';
import { setToken } from '../../store/login/setToken.helper';
import { history } from '../../history';

export class LoginContainer extends React.Component<any> {
    public componentWillMount(): void {
        const {token}: { token: string } = this.props.match.params;
        if (token) {
            setToken(token);
history.push('/feed');
        }
    }

    public onSubmit = (user: IUser): void => this.props.loginUser(user);

    public render(): JSX.Element {
        const {handleSubmit, submitting}: FormProps = this.props;

        return (
<Login
                handleSubmit={handleSubmit}
                onSubmit={this.onSubmit}
                submitting={submitting}
            />
        );
    }
}

export default connect(
    null,
    {loginUser},
)(
    reduxForm({
        form: 'loginForm',
```

```
        validate,
    })(LoginContainer),
);
```

Login.tsx

```
import React from 'react';
import { Field } from 'redux-form';
import { Link } from 'react-router-dom';
import { Spinner } from 'reactstrap';
import { renderField } from '../CommonComponents/ReduxFormFields';
import { Button, Form, FormGroup } from 'reactstrap';
import '../../styles/style.scss';
import logo from '../../assets/logo.png';
import { IUser } from '../../store/commonInterfaces/commonInterfaces';

interface IProps {
    handleSubmit: (onSubmit: any) => any;
    submitting: boolean | undefined;
    onSubmit: (user: IUser) => void;
}

const Login = ({handleSubmit, onSubmit, submitting}: IProps ): JSX.Element =>
{
    return (
<div className='container-fluid'>
<div className='row justify-content-center align-items-center'>
<div className='col-sm-8 col-md-6 col-xl-5'>
<Form className='mt-4 bg-white' onSubmit={handleSubmit(onSubmit)}>
<div className='border'>
<FormGroup className='col-lg-10 offset-lg-1 text-center'>
<Link to='/'>
<img className='picture img-fluid mt-4 mb-2' src={logo} alt='logo'/>
</Link>
<Field
                                    className='form-control form-control-lg'
                                    type='text'
                                    name='email'
                                    placeholder='E-mail'
                                    component={renderField}
                                />
<Field
                                    className='form-control form-control-lg
mt-3'
                                    type='password'
                                    name='password'
                                    placeholder='Password'
                                    component={renderField}
                                />
<Button
                                    className='mt-3'
                                    color='danger'
                                    disabled={submitting}
                                    size='lg'
                                    block
>
```

```
                                    {submitting ? <Spinner color='light'/> :
'Log in'}
</Button>
</FormGroup>
<div className='or-devider'>
<span />OR<span />
</div>
<div className='text-center mt-2'>
<p>
<i className='fa fa-google'/>
<a href={process.env.REACT_APP_GOOGLE_AUTH_URL}
                                    className='text-danger login-google
pl-2'
>
                                     Log in with Google
</a>
</p>
<p>
<i className='fa fa-facebook'/>
<a href={process.env.REACT_APP_FACEBOOK_AUTH_URL}
                                    className='text-danger login-google
pl-2'
>
                                     Log in with Facebook
</a>
</p>
<p>
<Link to='/password-reset' className='text-danger pl-1'>
                                     Forgot password?</Link>
</p>
</div>
</div>
</Form>
<Form className='bg-white mt-3'>
<div className='border'>
<FormGroup className='text-center register-acc mt-2'>
<p className='pt-2'>
                                  Still don't have an account?
<Link to='/register' className='pl-1 text-danger'>
                                     Register
</Link>
</p>
</FormGroup>
</div>
</Form>
</div>
</div>
</div>
    );
};

export default Login;
```

Рекомендації

FriendsRecommendations.tsx

```
import React from 'react';
import { Link } from 'react-router-dom';
import { Button } from 'reactstrap';
import noAvatar from '../../assets/noAvatar.png';

export interface IUser {
    photoPath: string;
    _id: string;
    username: string;
    isAlreadyFollow: boolean;
}

export interface IFriendsRecommendation {
    users: IUser[];
    loading: boolean;
}

interface IProps {
    loggedUsername: string;
    friendsRecommendations: IFriendsRecommendation;
    changeUsersFollowing: (id: string, followType: string) => void;
}

export class FriendsRecomendations extends React.Component <IProps> {

    public followRecommendationUser = (_id: string): void => {
        this.props.changeUsersFollowing(_id, 'follow');
    };

    public unfollowRecommendationUser = (_id: string): void => {
        this.props.changeUsersFollowing(_id, 'unFollow');
    };

    public dynamicButton = (_id: string, isAlreadyFollow: boolean):
JSX.Element => {

        if (isAlreadyFollow) {
            return (
<Button className='btn align-self-center'
                        color='danger'
                        onClick={(): void =>
this.unfollowRecommendationUser(_id)}>
                    Unfollow
</Button>
            );
        }

        return (
<Button className='btn align-self-center'
                    color='danger'
                    onClick={(): void => this.followRecommendationUser(_id)}>
```

```
                Follow
</Button>
        );
    };

    public render(): JSX.Element {
        const USERS_PER_PAGE = 4;
        const topRecommendations =
this.props.friendsRecommendations.users.slice(0, USERS_PER_PAGE);
        return (
<div className='mt-5'>
<h4 className='text-center'>Suggestions for you:</h4>
            {topRecommendations.map((user: IUser) => (
<div className='d-flex justify-content-between mb-2' key={user._id}>
<div>
<Link to={`/profile/${user.username}`}>
<img
                                    src={user.photoPath || noAvatar}
                                    alt='avatar'
                                    width={64}
                                    height={64}
                                    className='img-fluid rounded-circle'
                                /></Link>
<Link to={`/profile/${user.username}`} className='mt-1 ml-3 mr-4
                                  text-dark'>{user.username.length > 10 ?
                                            `${user.username.substring(0,
10)}...` : user.username}</Link>
</div>
                            {this.dynamicButton(user._id,
user.isAlreadyFollow)}
</div>
                ),
            )}
            {this.props.friendsRecommendations.users.length >
topRecommendations.length &&
<Link to={`/profile/${this.props.loggedUsername}/recommendations`}
className='mr-2 text-decoration-none'>
                See all recommendations</Link>}
</div>
        );
    }
}
```

FriendsRecommendationList.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { Link } from 'react-router-dom';
import { Container, Button} from 'reactstrap';
import { changeUsersFollowing } from '../../store/newsFeed/actions';
import { getRecommendations } from '../../store/newsFeed/actions';
import { IUser, IFriendsRecommendation } from '../FriendsRecommendations';
import Menu from '../Menu';
import noAvatar from '../../assets/noAvatar.png';
```

```
interface IProps {
    friendsRecommendations: IFriendsRecommendation;
    getRecommendations: () => void;
    changeUsersFollowing: (_id: string, followType: string) => void;
}

export class FriendsRecommendationsList extends React.Component <IProps> {

    public componentDidMount(): void {
        this.props.getRecommendations();
    };

    public followRecommendationUser = (_id: string): void => {
        this.props.changeUsersFollowing(_id, 'follow');
    };

    public unfollowRecommendationUser = (_id: string): void => {
        this.props.changeUsersFollowing(_id, 'unFollow');
    };

    public dynamicButton = (_id: string, isAlreadyFollow: boolean):
JSX.Element => {

        if (isAlreadyFollow) {
            return (
<Button className='btn' color='danger' onClick={(): void =>
this.unfollowRecommendationUser(_id)}>
                    Unfollow
</Button>
            );
        }

        return (
<Button className='btn' color='danger' onClick={(): void =>
this.followRecommendationUser(_id)}>
                Follow
</Button>
        );
    };

    public render(): JSX.Element {

        const { friendsRecommendations: { users } }: IProps = this.props;

        return (
<Container>
<Menu/>
<div className='d-flex justify-content-center'>
<div className='follow-wrapper'>
<h4 className='text-center font-weight-light text-secondary text-uppercase'>
                            Recommendations:</h4>
                        {users.map((user: IUser) =>
<div className='d-flex mt-1 mb-3 justify-content-between' key={user._id}>
<div className='row'>
<img
                                        src={user.photoPath || noAvatar}
                                        alt='avatar'
```

```
                                        width={32}
                                        height={32}
                                        className='img-fluid rounded-circle
ml-2 mr-2 mt-1'
                                    />
<h6 className='align-self-end'>
<Link to={`/profile/${user.username}`}
                                          className='text-
dark'>{user.username}</Link>
</h6>
</div>
                                    {this.dynamicButton(user._id,
user.isAlreadyFollow)}
</div>,
                        )}
</div>
</div>
</Container>
        );
    }
}

const mapStateToProps = (state: any): any => ({
    friendsRecommendations: state.newsFeed.friendsRecommendations,
});

const mapDispatchToProps = {
    getRecommendations,
    changeUsersFollowing,
};

export default connect(mapStateToProps,
mapDispatchToProps)(FriendsRecommendationsList);
```

GetFriendsRecommendation.ts

```
import { IUser } from '../db.requests/getFriendsRecommendations.requests';

interface IGraph {
    [id: string]: string[];
}

interface IUsersFriendshipDegree {
    [id: string]: number;
}

interface IFriendsRecommendations {
    [id: string]: number;
}

export const createGraph = (user: IUser): IGraph => {
    const graph: IGraph = {
        [user._id]: user.following.map((fol: any) => fol._id),
    };
```

```
    return user.following.reduce((result: IGraph, fol: any) => {
        const updatedByFollowingResult = fol.following.reduce((res: IGraph,
follow: any) => {
            if (!res[follow]) {
                return {...res, [follow]: [] };
            }
            return res;
        }, result);

        return {...updatedByFollowingResult, [fol._id]: fol.following.map((f:
object) => f.toString())};
    }, graph);
};

const checkWhetherToRecommend = (queue: string[], friend: string,
usersFriendshipDegree: IUsersFriendshipDegree,
                                 friendsRecommendations:
IFriendsRecommendations, current: any): void => {
    // If this node is already in queue && distance to it !== distance to
current node,
    // which in our case means that this user is common for at least two root
user's friends
    if (queue.includes(friend) && usersFriendshipDegree[friend] !==
usersFriendshipDegree[current]) {
        // If this user is already in recommendations, increase it's score by
one,
        // else add it to recommendations and set score to 1
        if (friendsRecommendations[friend]) {
            friendsRecommendations[friend] += 1;
        } else {
            friendsRecommendations[friend] = 1;
        }
    }
};

const setDistanceAndPushToQueue = (queue: string[], friend: string,
                                   usersFriendshipDegree:
IUsersFriendshipDegree, current: any): void => {
    // If we haven't visited this node, set distance to it by adding 1 to the
current distance
    if (usersFriendshipDegree[friend] === Infinity) {
        usersFriendshipDegree[friend] = usersFriendshipDegree[current] + 1;
        // Push this node to the queue
        queue.push(friend);
    }
};

export const findRecommendations = (users: IGraph, rootUser: string):
IFriendsRecommendations => {
    // Object, where key - is node, value - distance from the root node
    // (in our case, distance 1 - they are friends, 2 - have mutual friend,
and so on)
    const usersFriendshipDegree: IUsersFriendshipDegree =
    // Take all nodes of graph and set distance to rootUser to Infinity,
which means, that node is not visited yet
Object.keys(users).reduce((result: object, item: string) => ({...result,
[item]: Infinity }), {});
```

```
    // Object, where key - user to be recommended, value - recommendation
score
    const friendsRecommendations: IFriendsRecommendations = {};
    // Set distance from root node to itself to 0
    usersFriendshipDegree[rootUser] = 0;
    // Keep track of nodes to visit
    const queue = [rootUser];
    // Keep track of node we are currently traversing
    let current: any;
    // Keep traversing until there is no node anymore to traverse
    while (queue.length !== 0) {
        // Take node from queue (at the beggining - rootUser)
        current = queue.shift();
        // Get all nodes connected to the current node
        const userFriends = users[current];
        // For each connected node
        userFriends.forEach((friend: string) => {
            checkWhetherToRecommend(queue, friend, usersFriendshipDegree,
friendsRecommendations, current);
            setDistanceAndPushToQueue(queue, friend, usersFriendshipDegree,
current);
        });
    }

    return friendsRecommendations;
};

export const sortFriendsRecommendations = (friendsRecommendations:
IFriendsRecommendations): string[] => {

    return Object.keys(friendsRecommendations)
           .sort((a: string, b: string) => friendsRecommendations[b] -
friendsRecommendations[a]);
};
```

Месенджер

Notification.ts

```
export class Notifications {
    private namespace: any;
    constructor(private name: string, public io: any) {
        this.namespace = io.of(`/${name}`);
        this.listenOnRoom();
    }
    private listenOnRoom(): void {
        this.namespace.on('connect', (socket: any) => {
            this.handleJoinRoom(socket);
            this.handleOnNewNotification(socket);
```

```
        });
    }

    private handleJoinRoom(socket: any): void {
        socket.on('join room', (loggedId: string): void => {
            socket.join(`room-${loggedId}`);
        });
    }

    private handleOnNewNotification(socket: any): void {
        socket.on('add new notification',
            ({userId, username, message}: {userId: string, username: string,
message: string}): void => {
                socket.join(`room-${userId}`);
                this.handleEmitNewNotification(socket, userId, username,
message);
                socket.leave(`room-${userId}`);
            });
    }
    private handleEmitNewNotification(socket: any, userId: string, username:
string, message: string): void {
        socket.broadcast.in(`room-${userId}`).emit('add new notification',
{username, message});
    }
}
```

SocketConnection.ts

```
import io from 'socket.io-client';

export class SocketAPI {
    private socket: any;

    constructor(private path: string) {
        this.path = path;
        this.connect();
    }

    public connect(): Promise<unknown> {
        this.socket =
io.connect(`${process.env.REACT_APP_BASE_API}/${this.path}`);
        return new Promise(((resolve: any, reject: any): void => {
            this.socket.on('connect', () => resolve());
            this.socket.on('connect error', (error: any) => reject(error));
        }));
    }

    public disconnect(): Promise<unknown> {
        return new Promise((resolve: any): void => {
            this.socket = null;
            resolve();
        });
    }

    public emit(event: any, data: any): Promise<unknown> {
        return new Promise(((resolve: any, reject: any): void => {
            if (!this.socket) {
                return reject('No socket connection');
```

```
            }

            return this.socket.emit(event, data);
        }));
    }

    public on(event: string, fun: (data: any) => void): Promise<unknown> {
        return new Promise((resolve: any, reject: any): void => {
            if (!this.socket) {
                return reject('No socket connection');
            }

            this.socket.on(event, fun);
            resolve();
        });
    }
}
```

DirectContainer.tsx

```
import React from 'react';
import { connect } from 'react-redux';

import * as newsFeedAction from '../../store/newsFeed/actions';

import { IFeedState as INewsFeeState } from '../../store/newsFeed/reducers';
import { IFeedState } from '../../store/feed/reducers';

import DirectList from '../../components/Direct';
import {IUserData} from '../../components/Profile';
import {ISubscribersProps} from '../../components/Subscribers';
import {getUser} from '../../store/profile/actions';
import {getFriends} from '../../store/subscribers/actions';
import {changeUsersFollowing as changeUsersFollowingAction} from
'../../store/newsFeed/actions';

interface IDirectState {
    profile: {
        user: IUserData;
    };
    feed: IFeedState;
    subscribers: ISubscribersProps;
    newsFeed: INewsFeeState;
}

interface IDirectProps {
    loggedId: string;
    subscribers: [];
    user: IUserData;
    urlUsername: string;
    getUser: (username: string) => void;
    getFriends: (loggedId: string, subscribers: string, urlUsername: string)
=> void;
    changeUsersFollowing: (id: string, followType: string) => void;
    getRecommendations: () => void;
```

```
}

interface IDirectLocalProps {
    loggedId: string;
    user: IUserData;
    subscribers: [];
    newsFeed: INewsFeeState;
    loggedUsername: string;
}

type DirectProps = IDirectLocalProps & IDirectProps;

const DirectContainer = (props: DirectProps): JSX.Element => {
    const {
        loggedId,
        loggedUsername,
        newsFeed,
        subscribers,
        user,
        urlUsername,
        getRecommendations,
    } = props;

    return (
<DirectList
            loggedId={loggedId}
            loggedUsername={loggedUsername}
            newsFeed={newsFeed}
            friendsRecommendations={newsFeed.friendsRecommendations}
            subscribers={subscribers}
            user={user}
            urlUsername={urlUsername}
            getUser={props.getUser}
            getFriends={props.getFriends}
            changeUsersFollowing={props.changeUsersFollowing}
            getRecommendations={getRecommendations}
        />
    );
};

export const mapStateToProps = (state: IDirectState): IDirectLocalProps => ({
    newsFeed: state.newsFeed,
    loggedUsername: state.feed.loggedUsername,
    loggedId: state.feed.loggedId,
    user: state.profile.user,
    subscribers: state.subscribers.subscribers,
});

const mapDispatchToProps = {
    getRecommendations: newsFeedAction.getRecommendations,
    getUser,
    getFriends,
    changeUsersFollowing: changeUsersFollowingAction,

};

export default connect(mapStateToProps, mapDispatchToProps)(DirectContainer);
```