Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

# Розробка соціальної мережі

**Текстова частина до курсової роботи**
**за спеціальністю „Інженерія програмного забезпечення” 121**

Керівник курсової роботи
ст.викладач, Борозенний С.О
_____
*(підпис)*
“____” _____ 2020 р.

Виконав студент
Назар Т.І
“____” _____ 2020 р.

Київ 2020

**Календарний план виконання роботи:**

| № п/п | Назва етапу курсової роботи | Термін виконання етапу | Примітка |
|---|---|---|---|
| 1. | Отримання теми курсової роботи. | 10.11.2019 | |
| 2. | Вибір стеку для розробки застосунку | 10.12.2019 | |
| 3. | Створення базової архітектури застосунку | 20.01.2020 | |
| 4. | Написання перших двох розділів | 10.02.2020 | |
| 5. | Створення веб-застосунку | 30.03.2020 | |
| 6. | Написання останнього розділу | 10.04.2020 | |
| 7. | Створення презентації | 12.04.2020 | |

Студенту Назару Т.І

Керівник Борозенний С.О

“_____”_____

# Зміст

*Анотація*

У роботі розглянуто принципи написання веб застосунків за допомогою мови JavaScript та фреймворку React. Розроблено соціальну мережу, що поєднює в собі риси вже існуючих соціальних мереж та надає деякі нові можливості.

# ВСТУП

Сьогодні ми не можемо уявити свого існування без інтернету. Із розвитком технологій вихід в світову павутину стає все більш доступним та все більше людей щоденно витрачають багато часу будучи онлайн. Також інтернет вже давно перестав бути лише засобом для пошуку інформації, зараз він поєднує у собі як і науково-дослідницьке застосування так і розваги.

Зараз важко знайти людину яка б користувалася мережею інтернет, але не була б зареєстрована в жодній із численних соціальних мереж. Соціальні мережі в сучасному світі це сайти з одним із найбільших трафіком в мережі. І це недивно адже, завдяки їм можна дізнаватися свіжі новини, спілкуватися із друзями та просто вбивати час гортаючи стрічку новин. Зараз існує багато соціальних мереж таких як Facebook, Twitter, Instagram і всі вони користуються неабиякою популярністю.

За мету курсової роботи було поставлено створити прототип соціальної мережі, який б поєднував в собі популярні рішення вже існуючих та додавав деякий новий функціонал.

# РОЗДІЛ 1: ОПИС ІСНУЮЧИХ СОЦІАЛЬНИХ МЕРЕЖ

## 1.1 ІСТОРІЯ СОЦІАЛЬНИХ МЕРЕЖ

Небагато людей знає що поняття соціальна мережа з'явилося ще в далекому 1954 році і ввів його соціолог Джеймс Барнс. Тоді цей термін мав зовсім інше значення, а саме, соціальною мережею називали соціальну структуру, яка складалася з групи вузлів, якими є соціальні об'єкти(люди чи організації), і зв'язки між ними(соціальні відносини). З появленням інтернету в 1969 році наукова концепція Джеймса Барнса почала набирати популярність і це призвело до розвитку соціальних мереж в світовій павутині.

Цікаво те, що концепцію соціальних мереж передбачив російський письменник і філософ Володимир Одоевский ще в 1835 році в своєму романі «4338-й рік». Але перша мережа з використанням комп'ютерної техніки з'явилася в 1971 році вона використовувалась воєнними для передачі інфрмації через ARPANET(комп'ютерна мережа створена в 1969 році в США агентством міністерства оборони по перспективних дослідженнях яка була прототипом мережі інтернет). Через 17 років, в 1988 році, фінський учений Ярко Ойкаринен винайшов протокол «IRC» - ретрансльований інтерне-чат – і програмне забезпечення для його реалізації. Після цього стало можливо спілкуватись один з одним в реальному часі. Однак справжню популярність соціальні мережі получили в 1995 році. Тоді американець Ренді Конрадс створив Classmates.com – першу соцмережу в її сучасному розумінні. В ній зареєстровані користувачі получають доступ до каталогу випускників різних навчальних закладів. Таким чином будь-хто бажаючий може знайти однокласників чи однокурсників. Варто відмітити що Classmates.com відразу стала дуже популярною. До речі її популярність не падає і сьогодні – соціальною мережею користується більш ніж 50 млн людей, більш відомі нам «Однокласники» являються російським аналогом цієї мережі.

Зараз найпопулярнішою мережею в світі є Facebook. Компанія Facebook.inc була заснована 4 лютого 2004 року чотирма студентами, які навчалися в

Гарвардському університеті: Марком Цукенбергом, Едуардо Саверином, Дастіном Московіцем і Крісом Хьюзом. В той же час з'явився однойменний сайт. Спочатку він був доступний лише студентам Гарварду. Трохи пізніше реєстрацію відкрили для студентів університетів Бостона, а потім для всіх американців, які мали електронний адрес в домені .edu. Починаючи з вересня 2006 року Facebook став доступним для всіх користувачів інтернета віком від 16 років. Сьогодні він входить в п'ятірку найбільш відвідуваних сайтів світу. Недивно, що місячна аудиторія мережі складає 1,968 мільярдів людей.

| Rank | Website | Change | Avg. Visit Duration | Pages / Visit | Bounce Rate |
|---|---|---|---|---|---|
| 1 | google.com | = | 00:10:53 | 8.34 | 29.37% |
| 2 | youtube.com | = | 00:21:59 | 9.22 | 25.32% |
| 3 | facebook.com | = | 00:11:16 | 9.32 | 32.48% |
| 4 | twitter.com | +1 | 00:10:26 | 10.90 | 31.05% |
| 5 | baidu.com | -1 | 00:08:10 | 8.24 | 31.97% |
| 6 | instagram.com | = | 00:07:01 | 11.07 | 36.29% |
| 7 | xvideos.com | = | 00:12:26 | 9.19 | 26.39% |
| 8 | wikipedia.org | = | 00:03:51 | 2.95 | 58.42% |
| 9 | yahoo.com | +2 | 00:07:56 | 7.13 | 35.27% |
| 10 | pornhub.com | -1 | 00:09:38 | 7.32 | 28.90% |
| 11 | xnxx.com | -1 | 00:15:21 | 11.54 | 18.44% |
| 12 | yandex.ru | = | 00:10:41 | 8.72 | 28.60% |
| 13 | vk.com | = | 00:18:13 | 21.67 | 23.11% |
| 14 | amazon.com | = | 00:07:06 | 9.12 | 35.67% |
| 15 | live.com | = | 00:07:46 | 8.19 | 19.41% |

Десятого жовтня 2006 року в Росії з'являється аналог Facebook – соціальна мережа «ВКонтакті». Її створив Павло Дуров, сайт доступний на багатьох мовах, однак його основна аудиторія це російськомовні користувачі. Спочатку вона була задумана як ресурс для студентів та випускників російських вузів, але через деякий час він став себе позиціонувати як «сучасний, швидкий і естетичний спосіб спілкування в мережі».

Історія ще одної мегапопулярної сьогодні мережі Twitter почалася в березні 2006 року. Спочатку сервіс використовувася для спілкування між працівниками одноіменної компанії. А 15 липня 2006 року став доступним для публічного обміну повідомленнями: відправлені в програмі твіти зразу відображаються на сторінці користувача і приходять підписникам. Перше повідомлення відправив власник системи Джек Дорсі, він написав: «Just setting up my twtt». Сьогодні Twitter дуже популярний серед користувачів інтернет і займає 4 місце в рейтингу SimilarWeb.

Безпатний застосунок для обмыну фотографіями і відеозаписами – Instagram – з'явилося в магазині AppStore 6 жовтня 2010 року. Спочатку Instagram називався по-іншому – Burbn (проект названий в честь бурбона, який дуже любив один із засновників мережі). Він дозволя користувачам «чекінитися» в різних місцях, планувати зустрічі з друзями і публікувати фотографії. Проаналізвавши статистику, засновники Кевін Сістром і Майк Крігер зрозуміли, що люди не чекіняться, а лише обмінюються фотографіями. В зв'язку з цим вони вирішили позбавитися від усіх функцій, залишивши лише публікацію фото. Хід спрацював і мережа набрала нереальну популярність. Сьогодні мережа займає 6 місце в рейтингу SimilarWeb і має більш ніж 200 мільйонів активних користувачів, які загрузили більш ніж 16 мільярдів різних фотографій і відеороликів.

# РОЗДІЛ 2: ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

## 2.1 ОГЛЯД ФРОНТ-ЕНД ФРЕЙМВОРКУ REACT

React – це інструмент для створення UI(інтерфейс користувача), який був розроблений та зараз підтримується компанією Facebook. Це frontend бібліотека яка розширює та робить функціонал чистого JavaScript зручнішим та більш зрозумілим. Тобто головне його завдання це вивід на екран того що ми можемо бачити на веб-сторінках. React значно полегшує створення інтерфейсів завдяки розбивці кожної сторінки на невеликі фрагменти, які називаються компонентами. Нижче наведено приклад розбиття сторінки на компоненти.


Ще одним плюсом компонентної структури є те що кожен з компонентів може бути використаний в будь-якому місці веб-застосунку. Існує два види компонентів в React, перший це класові компоненти які мають власний стан, а інші це функціональні які характеризують відсутністю стану.

Що ж таке компонент в React, це якщо пояснювати просто, частина коду, яка представляє собою певну ділянку сторінки. Кожен компонент це JavaScript-функція яка повертає код, що представляє собою частину веб-сторінки та відображається в браузері. Для того щоб сформувати сторінку ці функції-компоненти викликаються в певній послідовності. Власне React відповідає лише

за відображення інформації у вікні браузера тому для створення повноцінних веб застосунків потрібно використовувати багато допоміжних бібліотек таких як React Router для налаштування і реалізації роутингу по застосунку, чи Redux для керування станом і даними.

Також однією із технологій, що використовує React є віртуальний DOM(Document Object Model). React створює в пам'яті кеш того що зараз відображено на сторінці і коли відбувається якась зміна старий кеш порівнюється із новим, якщо є якісь відмінності то відбувається оновлення того, що відображено у вікні браузера. Це дозволяє програмісту писати код так ніби при зміні оновлюється все, що знаходиться на сторінці, хоча насправді відбувається це лише з тими копонентами до яких було внесено зміни. Завдяки цьому сайти написані на React працюють швидко.

Ще одним зручним інструментом який надається розробникам для спрощення роботи є lifecycle методи, вони дозволяють виконувати певний код на різних етапах життєвого циклу компонента. Вони доступні лише в класових компонентах. Зараз можна використовувати такі методи:

- shouldComponentUpdate – цей метод дозволяє уникнути непотрібного ререндеренгу компонента, якщо він поверне false то компонент не буле оновлено
- componentDidMount – викликається коли компонент вперше відображається на сторінці, його часто використовують для асинхронних операцій таких як загрузка даних із сервера чи стороннього API.
- componentWillUnmount – викликається перед тим як компонент буде видалено із DOM, його використовують для видалення залежностей які прив'язані до даного компонента та не зникнути при його видаленні, наприклад setInterval() чи eventListener який існував через цей компонент.
- render – це найважливіший компонент життєвого циклу і він є обов'язковим для будь-якого компоненту реалізованого за допомогою

класу. Він викликається кожного разу коли змінюється стан компоненту і повертає новий вигляд компоненту.

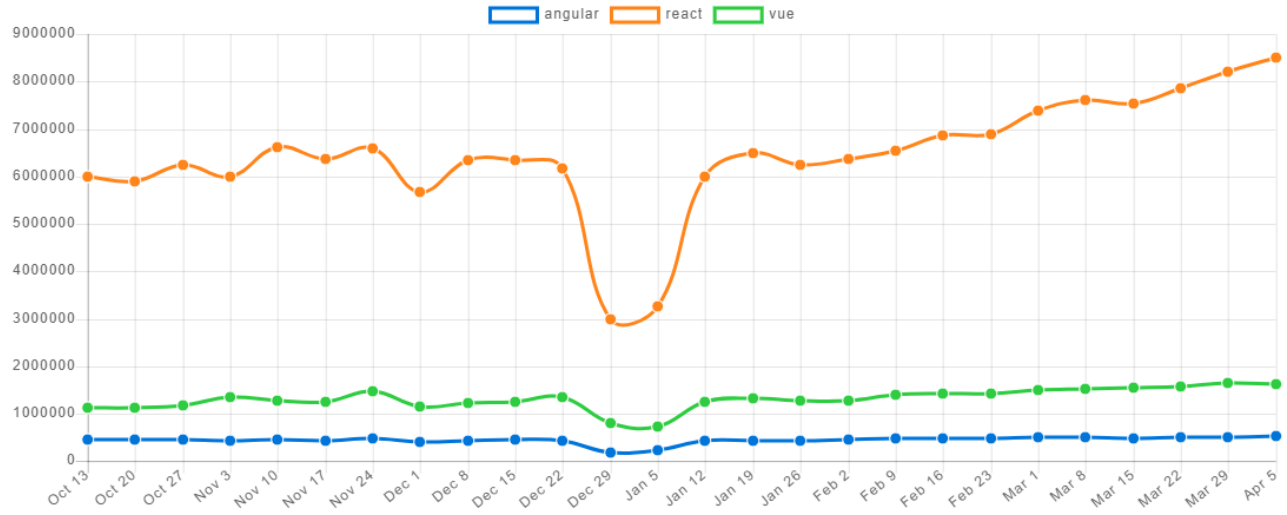
При роботі з React часто використовується розширення для синтаксису мови JavaScript – JSX або JavaScript XML. За своїм виглядом він нагадує HTML, JSX надає спосіб для структуризації рендерингу компонента за допомогою синтаксису який є добре відомий майже усім розробникам. Найчастіше компоненти в React реалізуються за допомогою JSX, але це необов'язково їх також можна писати і використовуючи чистий JavaScript. JSX схожий на розширення яке створив Facebook для PHP, XHP. Також кожен компонент повинен повертати лише один елемент, тобто всі елементи які включає компонент повинні бути загорнуті в контейнер, такий як  чи інші. Браузери не розуміють коду написаного на JSX тому для його перетворення використовується Babel, який робить його зрозумілим на етапі збору застосунку, перед його запуском.

В останніх версіях React було додано React Hooks вони дозволяють розробникам проникнути в середину компонента і його життєвого цикл. Вони не працюють в компонентах створених на базі класу, тому для їх роботи потрібні функціональні компоненти. Власне завдяки хукам можна додати до функціональних компонентів які за замовчуванням є stateless стан і методи життєвого циклу. React надає декілька базових хуків, найбільш поширеними серед них є useState за допомогою якого здійснюється контроль за станом компоненту, та useEffect який використовується для контролю за побічними ефектами. Також є декілька правил які потрібно виконувати коли користуєшся хуками:

3 Хуки мають використовуватись лише на верхньому рівні компонента, тобто не можна їх використовувати в циклі чи в if блоках
4 Хуки повинні викликатися лише з функціональних компонентів

Також є можливість створювати свої власні хуки, це дозволяє винести логіку компонента в окрему функцію яка потім може бути використана ще раз. Назва функції хуку має починатися із use та в ній можуть використовуватись інші хуки.

React зараз це найпопулярніше рішення для фронт-енд розробки, це можна побачити на графіках наведених нижче, на який показано кількість встановлень кожного з цих фреймворків. React з великим відривом переганяє своїх конкурентів.




Звідси і великий плюс це наявність великого, давно сформованого комьюніті, це значно спрощує розробку, адже часто можна знайти вже готове рішення для якоїсь із своїх проблем із детальним поясненням. Також базовий React досить легко освоїти, завдяки детальній документації та покроковому туторіалу, який дозволяє за наявності базових знань JavaScript, HTML, CSS досить швидко розібратися в основах фреймворку. React має першокласну підтримку PWA(Progressive Web App) завдяки генератору програм react-create-app, він дозволяє створити застосунок з усіма необхідними базовими налаштуваннями лише запустивши одну команду. Ще одним вагомим плюсом є те що навики які ви отримуєте при використанні React потім можуть бути використані для розробки нативних мобільних застосунків з використанням фреймворку React Native адже він розроблений використовуючи ті самі підходи що і в React, лише з невеликими змінами.

Тепер декілька слів про недоліки React. Те, що зараз відбувається відхід від класової побудови компонентів до функціональної може спричиняти дискомфорт для деяких розробників які привикли працювати з об’єктно орієнтованим програмуванням. Також при першому знайомстві змішування

шаблонів із логікою (використання JSX) може збити з пантелику деяких розробників.

## 2.1 ОГЛЯД ДОПОМІЖНИХ БІБЛІОТЕК

Розробляти високоякісні застосунки використовуючи тільки функціонал який надає React практично неможливо, тому він зазвичай використовується із рядом допоміжних бібліотек. Далі ми розглянемо декілька найбільш популярних та корисних із них.

Перш за все майже в кожному веб-застосунку потрібно реалізовувати роутинг, в React для цього завдання є зручний інструмент – react-router. Його можна встановити та додати до свого проекту використовуючи будь-який популярний менеджер пакетів. Ця бібліотека надає два види роутерів, BrowserRouter – його слід використовувати якщо ваш сайт буде використовувати запити до сервера та HashRouter – його використовують лише на статичних сторінках. В останній версії роути прописуються у вигляді компонентів, що полегшує розуміння бібліотеки для розробників. Плюсами цієї бібліотеки є легке використання та великий функціонал. Серед недоліків можна зазначити те, що якщо у вас недостатньо досвіду можна запутатися у написанні роутів.

Ще однією не менш важливою бібліотекою є Redux. Це бібліотека для керування станом застосунку. Він дозволяє получати та змінювати стан програми із будь якого компоненту.  В Redux є три основних поняття це: reducer, action і store. Store – це власне контейнер в якому зберігаються дані про стан нашого застосунку, actions – це прості функції що повертають об’єкт який містить два поля, type – назва і payload – якісь дані які ми хочемо модифікувати, reducer – відслідковує те як створюються actions і в залежності від типу створеного екшну проводить певні маніпуляції із даними що зберігаються в store. Великим плюсом redux є те, що він надає зручні Redux DevTools які дозволяють відслідковувати створення екшенів та зміни значень у сторі, це значно спрощує дебаггінг та виявлення дефектів у роботі програми.

Також використовуються різні middleware для виконання асинхронних операцій, таких як запити до серверу. Найпопулярнішими є thunks і sagas, при написанні свого застосунку я використовував останні. Це на сьогоднішній момент

найновіший підхід до обробки асинхронних операцій, він на працює на основі функцій-генераторів. Плюсом цього підходу є те що існують так звані saga-effects які дозволяють контролювати, що буде відбуватися якщо наприклад ми отримаємо декілька однакових екшенів одночасно.

## 2.2 ОГЛЯД БЕК-ЕНД ПІДХОДУ REST

При створені серверної частини використовувався підхід REST(Representational State Transfer). Цей термін вперше був введений Роєм Філдінгом, який також є одним із творців протоколу HTTP. REST – це стиль архітектури програмного забезпечення для розподілених систем, таких як WEB, який як правило використовується для побудови веб-служб. В загальному REST це дуже простий інтерфейс управління інформацією без використання будь-яких додаткових внутрішніх прослойок. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким як URL. Кожна URL в свою чергу має строго визначений формат.

Відсутність додаткових внутрішніх прослойок означає передачу даних в тому ж вигляді, що і самі дані, тобто ми не завертаємо дані в XML, як це робиться в SOAP.

Кожна одиниця інформації однозначно визначається URL – це означає, що URL по суті являється первинним ключем для одиниці даних. Тобто наприклад третя книжка на поличці буде мати вигляд /books/3, а 43 сторінка в цій книзі - /books/3/page/35. Звідси і получається строго заданий формат. Причому зовсім не має значення, в якому форматі надходять дані за адресом /books/3/page/35 – це можу бути і HTML, і скан-копія у вигляді картинки чи документ.

Як відбувається управління інформацією – це цілком і повністю базується на протоколі передачі даних. Найчастіше використовується HTTP, і взаємодія з сервісом відбувається за допомогою методів цього протоколу:

- GET – використовується для отримання даних
- POST – метод використовується для відправки на сервер об’єкту і часто створення нових даних
- PUT – метод використовується для модифікації вже існуючих даних
- DELETE – використовується для видалення даних

Ці методи використовуються найчастіше, також є і інші менш популярні: HEAD, CONNECT, PATCH, OPTIONS, TRACE.

Також творець REST визначив деякі архітектурні обмеження яких мають притримуватись розробники застосунків. Перше обмеження вказує, що мережа повинна складатися із клієнтів і серверів. Сервер – це комп'ютер, який має необхідні ресурси, а клієнт – це комп'ютер, якому потрібно взаємодіяти з ресурсами, які зберігаються на сервері. Тобто коли ми копаємось в інтернеті, наш комп'ютер виступає в ролі клієнта і відправляє HTTP запити серверу, щоб получати доступ до інформації і працювати з нею. RESTful-система має виконувати операції в клієнт-серверній моделі, навіть якщо компонент періодично веде себе то як клієнт, то як сервер. Альтернативою клієнт-серверної архітектури, побудованій без REST – це інтеграція, заснована на подіях. В цій моделі, кожен компонент неперервно передає події, перехоплюючи відповідні події із інших компонентів. В ній немає взаємодії один до одного, тільки передача і перехват. REST потребує взаємодії один до одного, тому архітектура, заснована на подіях не відповідає вимогам RESTful.

Ще однією вимогою є відсутність стану. Поняття без стану не означає, що сервіси і клієнти його не мають, у них просто немає необхідності відслідковувати стан один одного. Коли клієнт не взаємодіє з сервером, сервер не має представлення про його існування. Сервер також не веде облік минулих запитів. Кожен запит розглядається самостійно.

Також важливим обмеженням є одноманітність інтерфейсу. Обмеження гарантує, що між серверами і клієнтами існує спільна мова, яка дозволяє кожній частині бути замінною і змінною, без порушення цілісності системи. Це досягається через 4 підобмеження: ідентифікація ресурсів, маніпуляція ресурсами через представлення, «самодостатні» повідомлення і медіа. Перше підобмеження «уніфікованого інтерфейсу» впливає на те, як ідентифікуються ресурси. В термінології REST ресурсом може бути будь-що. Кожен ресурс повинен бути унікально визначеним постійним ідентифікатором. «Постійний» означає, що ідентифікатор не зміниться за час обміну даними і навіть коли зміниться стан ресурсу. Якщо ресурсу присвоюється інший ідентифікатор, сервер повинен повідомити клієнта, що запит був невдалий і дати посилання на

новий адрес. Другим підобмеження каже, що клієнт керує ресурсами, направляючи серверу представлення, зазвичай у вигляді JSON-об'єкта, який містить контент, який він хотів б додати, видалити чи змінити. В REST у сервера повний контроль над ресурсами, і він відповідає за будь-які зміни. Коли клієнт хоче внести зміни в ресурси, він надсилає серверу представлення того, яким він бачить кінцевий ресурс. Сервер приймає запит як пропозицію, але за ним все таки залишається повний контроль. «Самодостатні» повідомлення – це ще одне підобмеження, яке гарантує уніфікованість інтерфейсу у клієнтів і серверів. Тільки самодостатні повідомлення містить всю інформацію, яка необхідна для розуміння його отримувачем. В окремій документації чи повідомленні не повинно бути додаткової інформації. І останнє підобмеження – гіпермедіа, це поняття для зазначення даних, які містять інформацію про те, що клієнту потрібно робити далі, іншими словами, які ще запити він може зробити. В REST сервери повинні надсилати клієнтам тільки гіпермедіа. Коли система має ідентифікатори для кожного ресурсу, керує ними через направлення представлень від клієнта до сервера, містить самодостатні повідомлення і складається із гіпермедіа, то кажуть що в неї уніфікований інтерфейс.

Також є інші обмеження: кешування, система шарів і код за вимогою. Кешування: відповіді сервера повинні помічатись як кешовані і некешовані. Кешування відбувається, коли клієнт зберігає відповіді, получені раніше від сервера. Коли ці дані потрібні знову, кешування дозволяє позбутися повного проходу даних по мережі. Можливість кешування існує завдяки самодостатнім повідомленням. Клієнту не потрібно переживати про те, що випадково закешується частина необхідної інформації, а інші частини втратяться.

Система шарів припускає наявність більшої частини компонентів, ніж клієнт і сервер. В системі може бути більше одного шару. Тим не менш, кожен компонент обмежений можливістю бачити тільки сусідній шар і взаємодіяти тільки з ним. Проксі – це додатковий компонент, він ретранслює HTTP-запити на сервер чи інші проксі. Проксі сервера можуть бути корисними для балансування нагрузки і перевірки безпеки. Шлюз – це ще один додатковий

компонент, він переводить HTTP-запит в інший протокол, розповсюджує цей запит, а потім переводить отриману відповідь назад в HTTP.

Код за вимогою – єдине необов'язкове обмеження, яке передбачає відправлення сервером клієнту коду для виконання.

В цілому, система RESTful – це будь яка мережа, яка відповідає цим обмеженням. RESTful-система повинна бути достатньо гнучкою для різних сценаріїв використання, масштабованою для розміщення великої кількості користувачів і компонентів, а також адаптованою з плином часу.

# РОЗДІЛ 3: ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

## 3.1 ОПИС ОСНОВНОГО ФУНКЦІОНАЛУ ЗАСТОСУНКУ

Мною було розроблено веб-застосунок, прототип соціальної мережі, яка поєднує в собі класичні функції соціальних мереж та деякий новий функціонал.

В основі застосунку лежать всім відомі вже соціальні мережі, з такими класичними вже можливостями, як створення сторіс, постів, написання коментарів та залишення реакцій, система підписок. Також в якості розширеного «нестандартного функціоналу» було вибрана можливість пошуку фільмів, та перегляд інформації про них. Таке рішення було прийнято у зв'язку з тим, що майже всі люди люблять переглядати фільми та цікавляться ними, а серед відомих аналогів, не було знайдено таких які б комбінували як і базові можливості соцмереж так і сервісів з пошуку фільмів.

В якості основних технологій було вибрано React для фронт-енд частини та Node.js і Express бек-енд. Також на бек-енді використовується TypeScript – суперсет над JavaScript, який дає багато корисних функцій таких як статична типізація яка значно полегшує дебаггінг та траблшутінг.

Також в якості основної бази даних використовується MondoDB. І ще є додаткова база даних в якій зберігаються всі дані про фільми, для неї використовується ElasticSearch, для швидкої фільтрації фільмів та пошуку.

### 3.1.1 ОСНОВНОГО ФУНКЦІОНАЛУ ЗАСТОСУНКУ

При першому відвідуванні сайту коричтувачу необхідно сворити новий аккаунт, заповнивши просту форму із необхідними даними такими як прізвище і ім’я, адрес електронної пошти і пароль.


*Рис 3.1 Сторінка реєстрації*

У випадку якщо це вже не перший візит користувача можна перейти на сторінку входу нажавши кнопку Login і ввівши пароль та адресу електронної пошти потрапити на вже існуючий аккаунт.

Реєстрація і аутентифікація користувачів на сервері реалізовані використовуючи популярну бібліотеку passport.js і jsonwebtoken. При логіні користувача генерується унікальний токен і відправляється користувачу, після чого він зберігається в локальному сховищі браузера. Таким чином при закритті сторінки чи браузера користувач не буде викинутий із системи і непотрібно буде повторно входити. Токен являється валідним двадцять чотири години, після проходження цього часу необхідно буде знову пройти процес лог-іну.

### 3.1.2 ОПИС ГОЛОВНОЇ СТОРІНКИ

Після проходу процесу реєстрації або логіну, користувач потрапляє на основну сторінку сайту, де можна побачити головну інформацію.

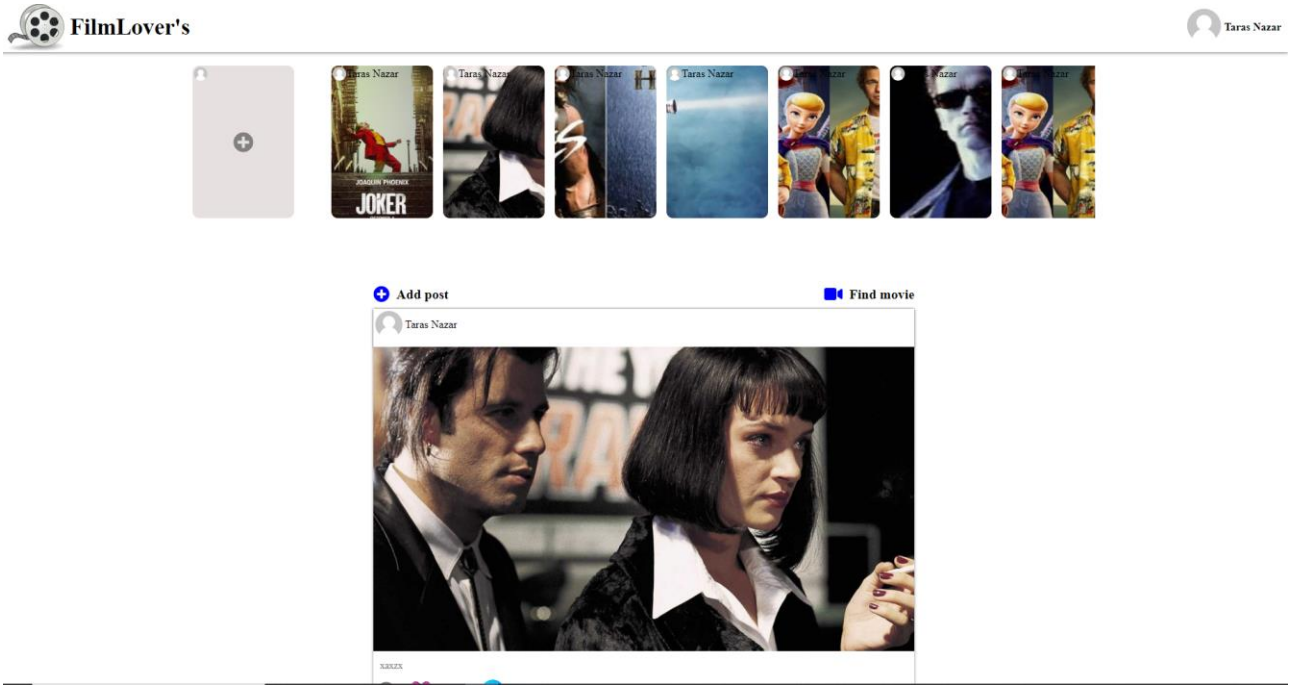



*Рис 3.2 Головна сторінка*

Основна сторінка складається із трьох базових компонентів, зверху знаходиться хедер із лого сайту та інформацією про користувача, також є компонент в якому знаходиться список сторіс створених користувачами та кнопка для створення нового сторіс і компонент із списком постів. На сторінці є ще дві кнопки: Add post і Find movie, перша відкриває конструктор для створення посту, а друга переводить на сторінку із пошуком фільмів.

### 3.1.3 ОПИС СТВОРЕННЯ ТА РОБОТИ СТОРІС

Зараз кожна сучасна соціальна мережа надає своїм користувачам можливість ділитися інформацією за допомогою сторіс – це зображення або короткі відео. Це дуже зручний спосіб поділитися чимось цікавим, адже зазвичай список сторіс знаходиться на самому верху сторінки і таким чином не втрачається серед динамічного потоку постів. В своїй соцмережі я також реалізував спрощний прототип тих сторіс, які ми звикли бачити в Instagram і Facebook.

Всі сторіс створені користувачами знаходяться у списку відразу під хедером, це можна побачити на *Рис 3.2.* Вони відображаються у вигляді карточок і в якості фону використовується вибрана користувачем картинка чи колір. На початку списку є кнопка – плюс, яка дозволяє створити новий запис. При натисканні на неї відкривається конструктор для створення сторіс.

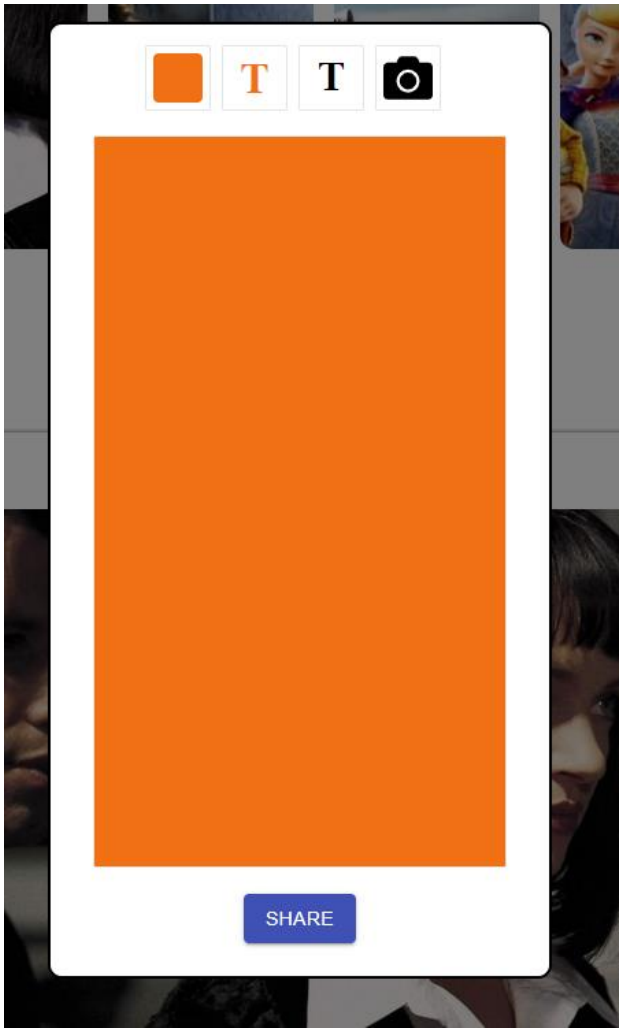



*Рис 3.3 Конструктор для стоврення сторіс*

Наверху знаходяться основні кнопки управління конструктуром. Перша дозволяє вибрати фоновий колір для сторіс, далі йде дві кнопки для додавання тексту та зміни його кольору і остання дозволяє завантажити картинку. При додаванні тексту його можна розмістити будь-де на дооступному просторі, для цього використовується механізм drag and drop. Також після завантаження зображення спочатку користувач повинен обрізати його використовуючи кроппер, це зроблено для того, щоб пізніше при перегляді сторіс не відбувалось стискання чи розтягнення картинки без збереження пропорцій і відповідно не втрачалась якість зображення. Коли користувач закінчив створення свого запису потрібно лише натиснути кнопку Share і запис відразу з’явиться в списку сторі. При виборі будь-якої сторі із списку вже існуючих користувач потрапить до переглядача.


*Рис 3.4 Переглядач сторі*

Тут можна побачити власне саму сторі. Для переходу до наступного чи попереднього запису можна використовувати стрілки розміщені справа та зліва від вікна перегляду.

### 3.1.4 ОПИС СТВОРЕННЯ ТА РОБОТИ ПОСТІВ

Пости – це базовий спосіб ділитись інформацією та цікавими речами в інтернеті і жодна сучасна соціальна мережа не може існувати без цієї функції. В розробленій мною системі ця функція також присутня, список усіх постів відображається на головній сторінці.

Конструктор для створення нового посту викликається натисканням відповідної кнопки на головній сторінці – «Add post». Він виглядає ідентично до описаного в попередньому розділі конструктору сторіс, лише з дещо зміненими функціями. На картинці вибраній для створення посту не можна розмістити текст, натомість, є спеціальне поля в яке можна додати опис для посту.

Для відображення посту є створений компонент Posts і саме список цих компонентів відображається на говловній сторінці.

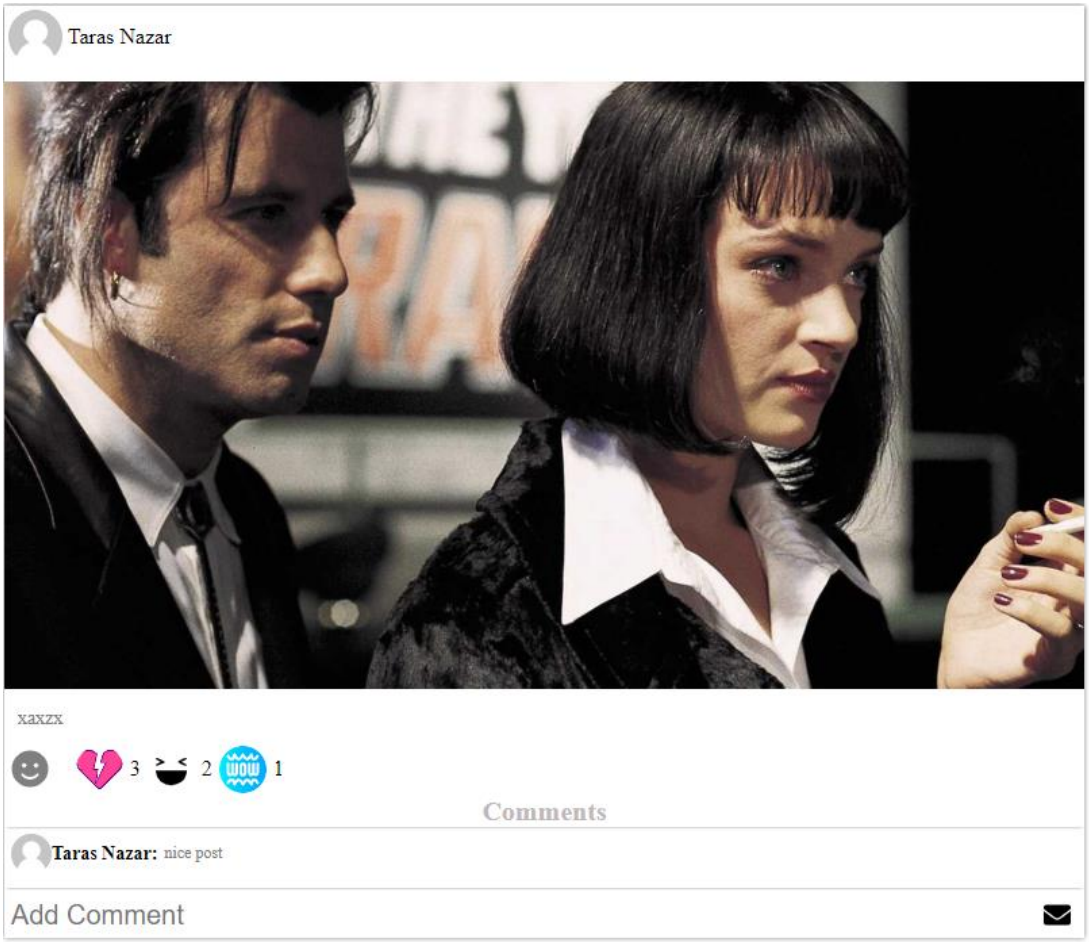



*Рис 3.5 Пост*

Кожен пост складається із зображення та опису. Жодна з цих частин не є обов’язковою, у випадку відсутності картинки буде відображено дефолтне зображення. Також на верху кожного посту є інформація про автора. Ще однією функцією без якої неможливо уявити пости це емоджі, реакції. Для користувачів є доступний набір емоджі які дозволять  відображати свої враження від публікації. Доступні такі емоції: like, dislike, haha, wow, sad і angry. Всі залишені реакції відображаються під постом разом із кількістю людей, що так відреагували.

Кожен користувач також може залишити коментар до публікації і почати спілкування з іншими юзерами. Кожен коментар складається з часу його створення, користувача, що його залишив і тексту.

### 3.1.5 ОПИС ФУНКЦІОНАЛУ ПОВ'ЯЗАНОГО З ФІЛЬМАМИ

Крім стандартного для всіх більшості соцмереж функціоналу, розроблений мною застосунок також надає можливість шукати фільми та переглядати інформацію про них. Фільми зберігаються в окремій базі даних на базі ElasticSearch. Для наповнення бази даних було використано дані із The Movie Database, це сервіс схожий до IMDb, проте із зручним API. Було написано скріпт який дістає дані із TMDB API, записує їх в JSON файл, який потім імпортується в мою базу даних. ElasticSearch розміщений використовуючи appbase.io – це сервіс який дозволяє задеплоїти свою базу даних безкоштовно (є обмеження по кількості запитів), також сервіс надає зручний візуальний інтерфейс для відображення даних та управління ними, також можна переглянути історію запитів та результати їх виконання, дуже корисно при дебаггінгу. Ще одною зручною та корисною функцію яку забезпечує сервіс це плейграунд для надсилання квері, де можна потренуватися в написання квері та протестувати їх.

Користувач може перейти на сторінку пошуку фільмів натиснувши кнопку Find movie на головній сторінці.

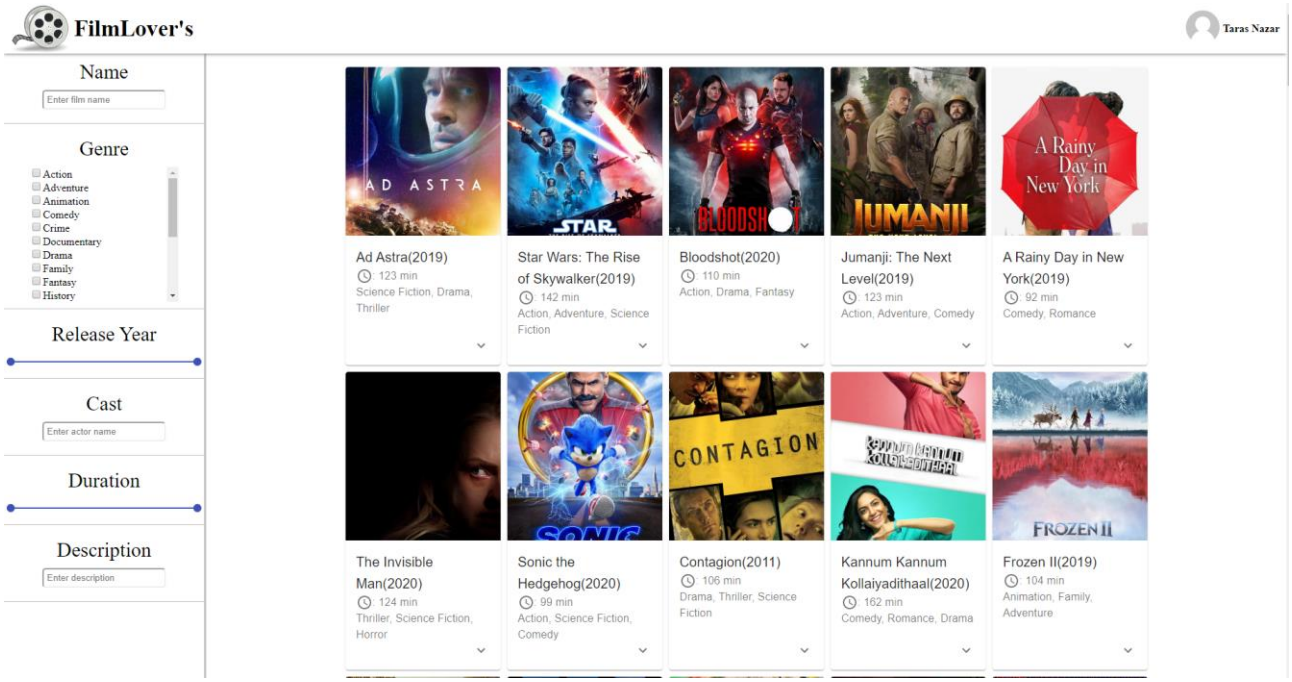



*Рис 3.6 Сторінка пошуку фільмів*

Сторінка пошуку фільмів складається із сайдбару із набором доступних фільтрів і контейнера в якому відображаються список фільмів, що задовільняють вибрані критерії. Серед доступних фільтрів є:

- Назва – дозволяє шукати фільми за їх назвою або частиною назви, потрібно просто ввести символи у відповідний інпут
- Жанр – дозволяє фільтрувати фільми за жанрами, всі доступні жанри відображаються у вигляді списку із чекбоксами потрібно вибрати необхідні пункти, можна вибирати більше ніж один тоді будуть відображені фільми, що відповідають хоч одному із вибраних жанрів. Також можна здійснити пошуку жанрів у списку
- Рік випуску – дозволяє фільтрувати фільми за датою їх виходу, потрібно задати проміжок у роках
- Каст – можна шукати за актором, потрібно ввести ім’я актора і буде відображено фільми в яких він брав участь. Можна вводити лише одного актора.
- Опис – кожен фільм має короткий опис який характеризує його, можна ввести пене слово чи фразу і буде відображено фільми опис яких містить ці слова
- Тривалість – за цей фільтр відповідає слайдер, потрібно вказати проміжок у хвилинах і будуть відображені фільми із такою тривалістю

Фільми, що задовільняють вибрані фільтри відображаються лівіше у вигляді карточок, кожна з яких містить основну інформацію про фільм: постер, назву і рік випуску, перелік жанрів і тривалість.

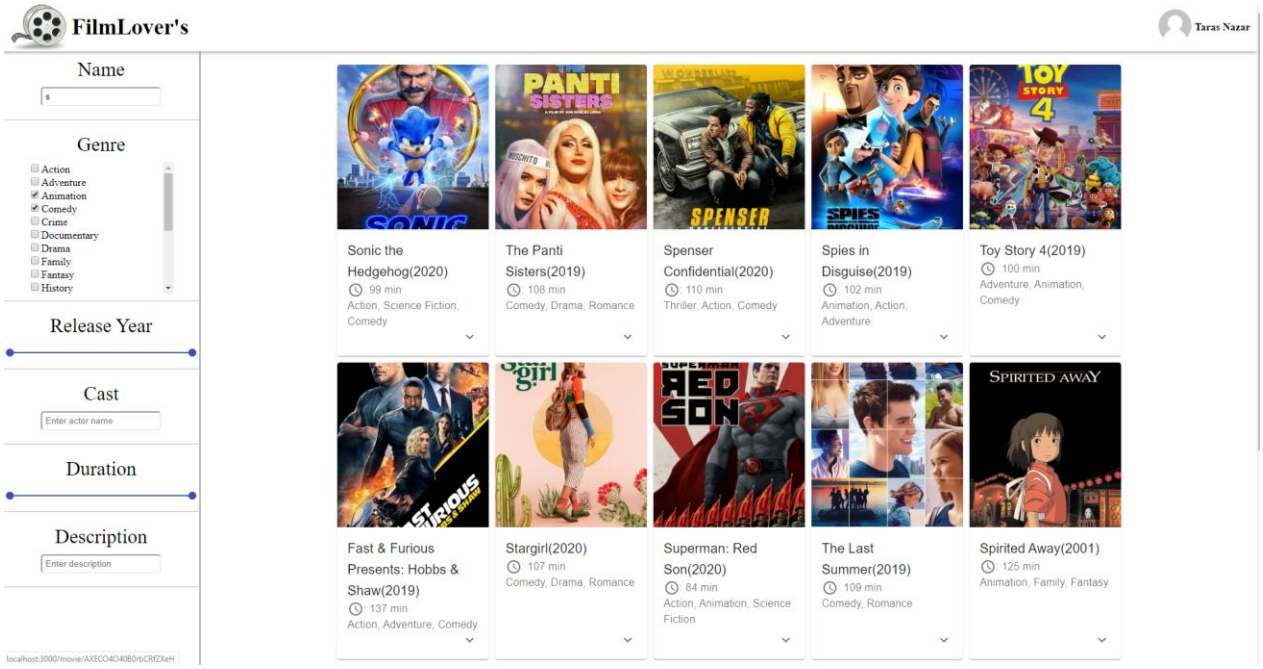



*Рис 3.7 Приклад роботи фільтрів*

Після того як користувач знайшов потрібний фільм, він може переглянути більш детальну інформацію про нього, натиснувши на карточку користувач буде переадресований на сторінку вибраної стрічки.

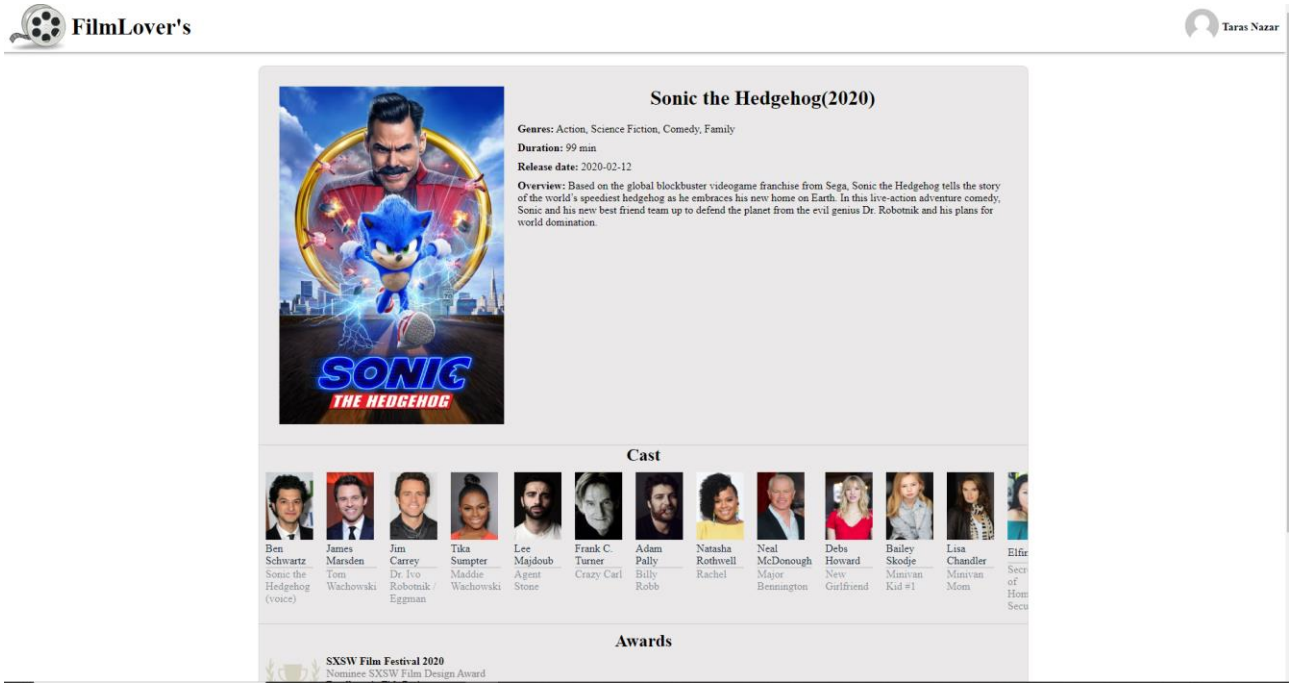



*Рис 3.8 Сторінка фільму*

Сторінка фільму складається із кількох секцій, на першій – базовій відображаються основна інформація про фільм, така як постер, назва, рік випуску, жанр, тривалість, опис, бюджет.

В секції Cast можна побачити інформацію про акторів які брали участь в зйомці фільму та ролі які вони виконували. Інформація відображається у вигляді списку карток, які містять фото людини, її ім’я та роль.

У секції Awards можна знайти список нагород які завоював або на які було номіновано фільм. На жаль основне API яке я використовував для заповнення бази із фільмами – TMDB API не надає такої інформації, тому довелось використовувати ще одне API – MyApiFilms, воно дістає інформацію про нагороди за imdbId фільму. Це API парсить сторінку фільму на IMDb та повертає необхідну інформацію. Кожна нагорода відображається у вигляді айтема який складається із іконки (якщо вона кольорова – це означає що фільм завоював нагороду, якщо сіра – був лише номінований), назви підрозділу нагороди, та якщо є, списку людей яким вона була присвоєна.

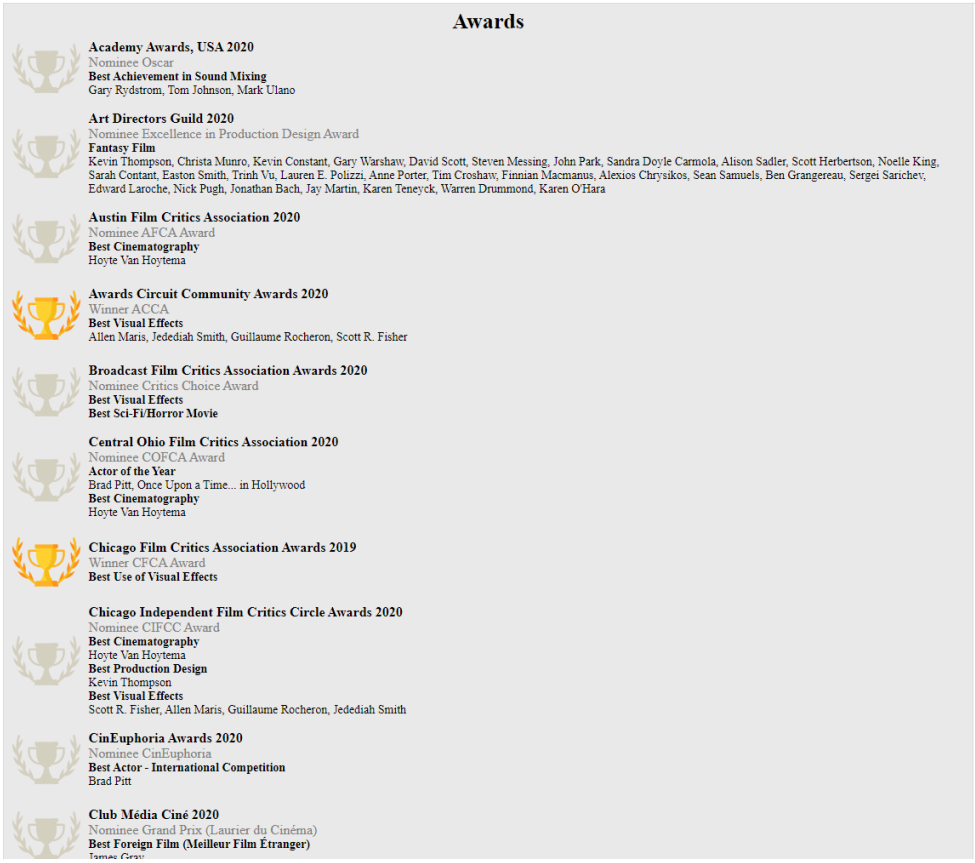



*Рис 3.9 Секція із нагородами*

У секції Review відображаються анонімні відгуки інших користувачів, також є кнопка та поле для додавання свого відгуку.

### 3.1.6 ОГЛЯД СТОРІНКИ КОРИСТУВАЧА

І звичайно найбільш базовою функцією, яку надає кожна соцмереража є перегляд сторінок інших користувачів.

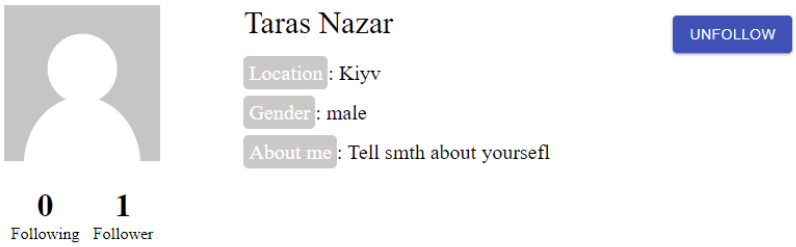



*Рис 3.10 Сторінка користувача*

На сторінці користувача можна знайти інформацію про вибраного користувача, таку як аватар, ім’я, місце проживання, стать і те що сам користувач хоче сказати іншим про себе. Також можна підписатися на користувача, або якщо ви вже підписані то відписатися. Під аватаром користувача можна подивитися кількість користувачів на яких він підписаний, а також кількість його підписників. Нажавши на цю кількість відкриється модальне вікно, в якому можна побачити список цих користувачів.

# Висновки

Веб-застосунки зараз є одними із найпопулярніших застосунків у світі. Мільярди користувачів щодня користуються ними, а соціальні мережі, як вид таких застосунків, займають одне з перших місць за популярністю. Також для веб-розробки існує багато корисних та зрозумілих бібліотек і фреймоврків, а також підходів до розробки, які значно спрощують процес написання коду.

Результатом виконання курсової роботи є створений веб-застосунок додаток – соціальна мережа. Розроблений веб-сайт надає базовий функціонал, такий як створення постів і сторіс, взаємодію із іншими користувачами через коментарі, реакції до постів, відгуки до фільмів та підписки. Також був реалізований функціонал для зручного пошуку фільмів за фільтрами і перегляду повної інформації про фільми.

## Список використаної літератури

1.React.js documentation[Електронний ресурс]

https://uk.reactjs.org/docs/

2.Redux documentation[Електронний ресурс]

https://redux.js.org/api/api-reference

3.ElasticSearch documentation[Електронний ресурс]

https://www.elastic.co/

4. MongoDB documentation[Електронний ресурс]

https://www.mongodb.com/

5. Introduction to REST API - RESTful Web Services [Електронний ресурс]

https://www.springboottutorial.com/introduction-to-rest-api

6. Social networks history [Електронний ресурс]

https://www.antevenio.com/usa/a-brief-history-of-social-networks/

7. David Flanagan. “JavaScript: The Definitive Guide: Activate Your Web Pages(Definitive Guides)”