Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія управління соціальною мережею»

Виконав: студент 2-го курсу, групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Саїнський Б.Д.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. КН
_____ Озеранський В.С.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: к.т.н., доц. каф. САІТ
_____ Варчук І.В.
(прізвище та ініціали)
«07» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«8» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти II-й (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 122 – Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
_____
(підпис)
"29" 08 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Саїнському Борису Дмитровичу

1 Тема роботи: «Інформаційна технологія управління соціальною мережею».
Керівник роботи: Озеранський Володимир Сергійович, к.т.н., доц.
затверджені наказом вищого навчального закладу «18» 09 2023 року № 247

2 Строк подання студентом роботи 13.11.2023

3 Вихідні дані до роботи: використання об'єктно-орієнтованої мови програмування; клієнт-серверна архітектура; база даних типу NoSQL; кількість колекцій в базі даних - не менше 2шт; кількість полів для використання в колекції - не менше 3шт.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасних технологій управління соціальною мережею, моделювання інформаційної технології управління соціальною мережею, програмна реалізація інформаційної технології управління соціальною мережею, економічна частина, висновки, перелік використаних джерел, додатки

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму функціонування інформаційної технології управління соціальною мережею, Структура інформаційної технології управління соціальною мережею, Приклади роботи програми.

6 Консультанти розділів проекту (роботи)

Консультанти розділів роботи в таблиці 1.

Таблиця 1 - Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Озеранський В. С., к.т.н., доц. каф. КН	29.08.23	01.11.23
4	Адлер О.О., к.т.н., доцент каф. ЕПВМ	01.11.23	02.11.23

7 Дата видачі завдання 29.08.23

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних технологій управління соціальною мережею	01.09.23 - 05.09.23	
2	Моделювання інформаційної технології управління соціальною мережею	06.09.23 - 16.09.23	
3	Програмна реалізація інформаційної технології управління соціальною мережею	17.09.23 - 07.10.23	
4	тестування та аналіз результатів роботи розробленої програми	08.10.23 - 23.10.23	
5	Розробка інструкції користувача	24.10.23-01.11.23	
6	Оформлення матеріалів до захисту МКР	02.11.23-10.11.23	

Студент _____ Саїнський Б.Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Озеранський В. С
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.415

Саїнський Б.Д. Інформаційна технологія управління соціальною мережею. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2023. 100с.

На укр. мові. Бібліогр.: 20 назв; рис.: 25; табл. 13.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології управління соціальною мережею.

У магістерській кваліфікаційній роботі проведено аналіз та порівняння різноманітних методів та технологій призначених для створення та управління соціальними мережами. Проведено дослідження сучасних технологій які дозволяють створювати Web-сайти та соціальні мережі, здійснено порівняння найпопулярніших соціальних мереж.

Дана робота допомагає проаналізувати питання, які стосуються розробки соціальних мереж, та допомагає актуально підібрати технологію, яку буде найзручніше застосовувати для розробки інформаційної технології організації соціальної мережі.

В розділі економіки було виконано розрахунок загальних витрат на завершення науково-технічної роботи та оформлення її результатів, які складають 281172 гривень, період окупності витрат складає 1,93 року.

Ключові слова: інформаційна технологія, соціальні мережі, спілкування.

ABSTRACT

Sainsky B.D. Information technology of social network management. Master's thesis on specialty 122 - computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2023. 100p.

In Ukrainian speech Bibliography: 20 titles; Fig.: 25; table 13.

This master's thesis is devoted to the development of information technology for social network management.

In the master's qualification work, an analysis and comparison of various methods and technologies designed for the creation and management of social networks was carried out. A study of modern technologies that allow creating websites and social networks was carried out, a comparison of the most popular social networks was carried out.

This work helps to analyze the issues related to the development of social networks and helps to choose the technology that will be most convenient to use for the development of information technology of the social network organization.

In the economy section, the calculation of the total costs for the completion of the scientific and technical work and the registration of its results was performed, which amount to 281,172 hryvnias, the payback period of the costs is 1.93 years.

Keywords: information technology, social networks, communication.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Для десятків мільйонів людей XIX століття життя стало неможливе без технологічного явища − соціальних мереж. Світ змінюється, а разом із ним змінюються й пріоритети людини. Розробки не стоять на місці і відповідають нашим потребам та вимогам. Саме тому деякі соціальні мережі більш популярні, а якісь менш популярні. Соціальні мережі надзвичайно потужно впливають на розумову діяльність людини, що призводить до серйозних відхилень у розвитку організму та викликають не лише звикання, але й різноманітні захворювання. Необхідно чітко знати і розуміти, які наслідки можуть бути від постійного користування соціальними мережами. Через це необхідно проводити наукові дослідження у цій сфері.

Соціальна мережа (від англ. Social networking service) − платформа, онлайн-сервіс або веб-сайт, призначені для побудови, відображення і організації соціальних взаємовідносин.

Кінець XX − початок XXI століття відзначено бурхливим розвитком Всесвітньої павутини. У міру її зростання стають все поширенішими і різні новітні інформаційні технології − використання Інтернету стає звичним у багатьох сферах життя і діяльності людини, раніше далеких від активного використання ЕОМ.

Соціальні мережі − основна причина, по якій сьогодні зростає кількість часу, проведеного в Інтернеті. Головні їх переваги − можливість користувачів заявляти про свої інтереси, і розділяти їх з оточуючими. І це дає підстави стверджувати, що соціальні мережі є не тільки засобом для спілкування, а й потужним маркетинговим інструментом, більш того, дослідники вважають, що незабаром вони стануть необхідним інструментом для ведення діяльності. Соціальні мережі служать майданчиком для неформального спілкування, допомагають створювати нову музику, служать серйозним інструментом для пошуку співробітників і партнерів.

Соціальні мережі є потужним інструментом маркетингових досліджень, оскільки користувачі добровільно публікують інформацію про себе, свої погляди, інтереси, переваги і так далі. Зважаючи на це рекламодавці можуть досить чітко

визначати, яких саме користувачів зацікавить їх оголошення, і направити свої рекламні оголошення конкретним користувачам, в залежності від інформації в їх профілях (вік, стать, місце проживання та інше).

Зв'язок роботи з науковими програмами, планами, темами. Магістрська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 – «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікації » та плану навчально-методичної та наукової роботи кафедри.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей програмного забезпечення управління соціальною мережею.

Для досягнення поставленної мети необхідно розв'язати такі завдання:

- аналіз та обґрунтування доцільності розробки інформаційної технології управління соціальною мережею;
- аналіз та вибір методів та технологій для управління соціальною мережею;
- розробити інформаційну технологію управління соціальною мережею;
- розробити структурну організацію соціальної мережі та здійснити її програмну реалізацію;
- провести тестування програми та проаналізувати отримані результати;
- економічно обґрунтувати доцільність розробки нової інформаційної технології управління соціальною мережею.

Об'єкт дослідження – процес управління соціальною мережею.

Предмет дослідження – програмні засоби управління соціальною мережею.

Методи дослідження. У роботі використані такі методи наукових досліджень: аналіз інформаційної системи соціальної мережі, методи оброблення цифрової інформації, методи взаємодії між серверами, методи комунікації з базою даних, об'єктно-орієнтованого програмування для отримання подальших результатів роботи програми.

Наукова новизна одержаних результатів полягає в наступному:

– вдосконалено інформаційну технологію управління соціальною мережею, яка відрізняється від існуючих використанням нейромережевих технологій управління соціальною мережею, що забезпечує розширення функціональних можливостей програмного забезпечення управління соціальною мережею.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено алгоритм управління соціальною мережею;
2. Розроблено програмне забезпечення управління соціальною мережею.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, чітким та лаконічним виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати роботи були апробовані на конференції «LI Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2023)» (м. Вінниця, Україна, 2023 р.).

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано тезу доповіді на конференції «LI Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2023)» [1].

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ СОЦІАЛЬНОЮ МЕРЕЖЕЮ

1.1 Аналіз існуючих соціальних мереж

На даний час в світі існує більше 20 відомих соціальних мереж. Використовуються вони різними віковими, та ставтевими прошарками, починаючи від людей віком 16 років, які використовую соціальні мережі для самовираження та розваг, закінчуючи людьми і старшим віком, для яких соціальні мережі залишаються як основний спосіб отримання новин та цікавинок зі всього світу. Зі всього різноманіття соціальних мереж можна виділити декілька основних:

- Facebook;
- Instagram;
- TikTok;
- Twitter;
- Reddit;
- LinkedIn.

Facebook − найбільша соціальна мережа у світі та була запущена 4 лютого 2004 року як мережа для студентів кількох американських університетів. Засновник і голова служби - Марк Цукерберг. За даними Alexa, сайт Facebook.com займає третє місце в світі за обсягом трафіку.

У червні 2017 року кількість українських користувачів Facebook зросла після заборони на російські Інтернет-ресурси, такі як соціальні мережі Одноклассники та ВКонтакте. В результаті кількість облікових записів Facebook в Україні досягла 10 мільйонів, а в липні того ж року Facebook став найпопулярнішою соціальною мережею в Україні.

Сторінка аутентифікації і реєстрації (рис. 1.1), завантажується швидко і виконана в синьо-сірих тонах (с градієнтом білого), блок реєстрації розташований посередині і виділений великим шрифтом, що сприяє концентрації уваги користувача на важливою для нього інформації.

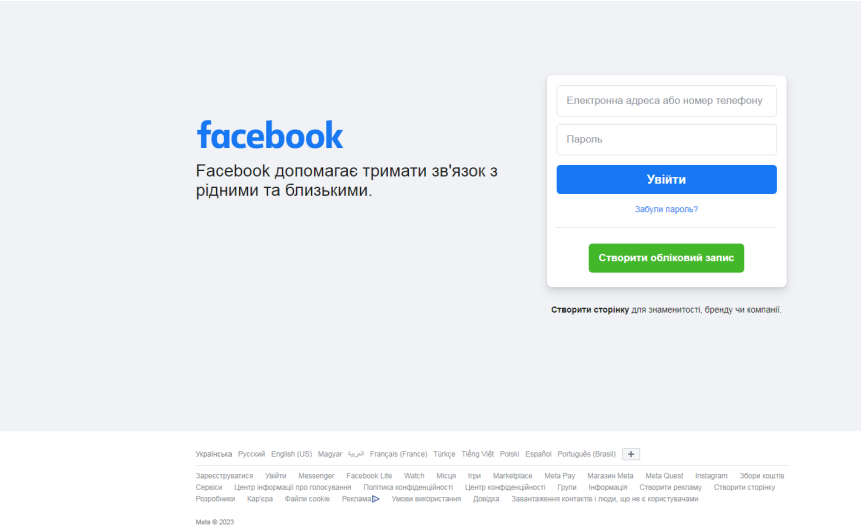

Рисунок 1.1 – Сторінка аутентифікації та реєстрації соціальної мережі «Facebook»

Блок реєстрації розроблений дуже зручно і дозволяє відразу ввести всю необхідну для користувача інформацію. Вказані посилання на «Послуги використання» і «Політику використання даних». Також присутня можливість реєстрації публічної особи чи організації. У процесі реєстрації надається допомога користувачеві у вигляді спливаючих підказок і попереджень.

Особиста сторінка користувача захаращена зайвими візуальними компонентами (рис. 1.2). Загальну ергономічність сторінки можна розцінювати як середню. Незручно розташовані пошукові посилання [2].

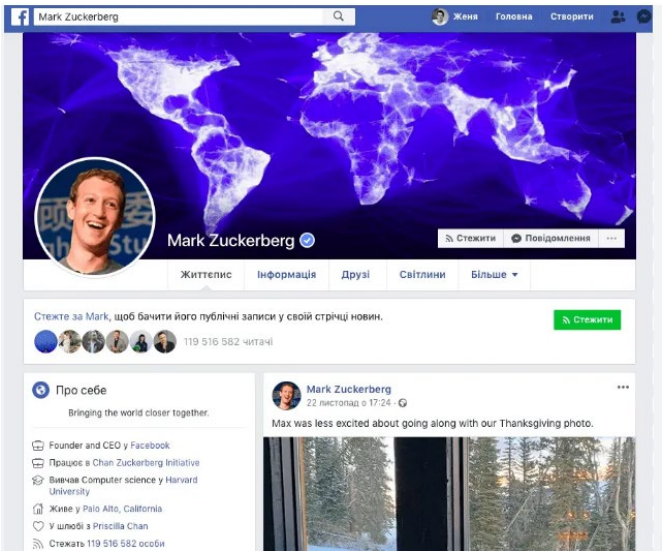

Рисунок 1.2 – Особиста сторінка користувача соціальної мережі «Facebook»

Instagram – це соціальна мережа, заснована на обміні фотографіями, що дозволяє користувачам робити фотографії, застосовувати фільтри та розповсюджувати їх через сервіси та інші соціальні мережі (рис. 1.3). Один з найпопулярніших сервісів для iPhone. Instagram робить знімки у формі квадрата, як фотокамера Kodak Instamatic або Polaroid. Більшість мобільних редакторів фотографій використовують співвідношення сторін 3: 2.

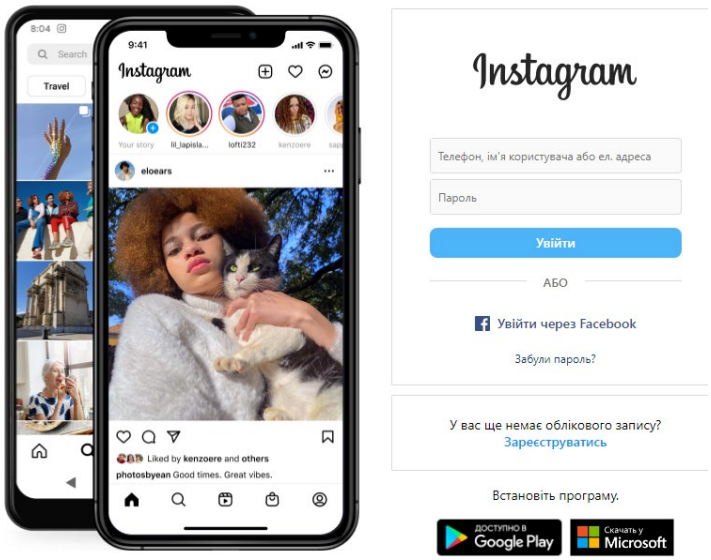

Рисунок 1.3 – Сторінка мережі «Instagram»

Застосунок сумісний зі специфікацією IPhone, IPad і IPod на IOS 4,0 і вище, а також зі смартфонами на Android 4,2,2 і вище з підтримкою OpenGL ES 2, а також доступний для смартфонів, на яких встановлена Windows Phone 8 (8.1). Поширюється воно через App Store, Google Play, Windows Marketplace відповідно [3].

TikTok (відомий у Китаї як **Douyin**) — програма соціальних медіа для створення та розповсюдження відеофайлів та онлайн-трансляцій. Розроблений китайською компанією ByteDance, запущений у вересні 2016 року, є найпопулярнішою в Азії відео платформою, і швидко поширилвся в інші частини світу (рис. 1.4).

За допомогою цієї програми ви можете створювати відео з музикою, танцями, комедіями тощо протягом 15 секунд. Програма охопила 500 мільйонів користувачів із 150 країн.

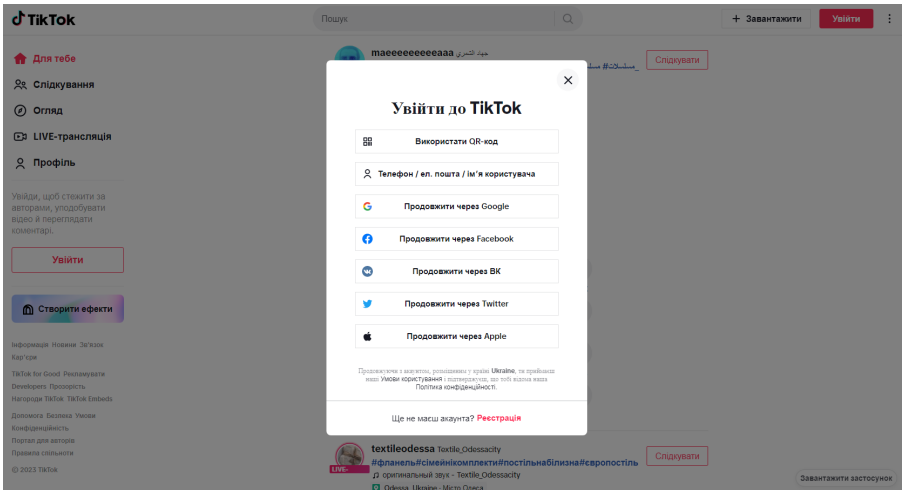

Рисунок 1.4 – Сторінка мережі «TikTok»

Застосунок надає користувачам можливість виставити свої акаунти у приватний режим, дозволивши перегляд контенту лише вибраним користувачам. Користувачі також можуть дозволити всім або лише друзям надсилання їм коментарі або повідомлення, а також «duet» з чи «react» на їхні відеофайли. Функція «duet» дозволяє користувачам створювати нові відео поряд з вже таким, що існує, а функція «react» — поміщати нове відео у менший фрейм, який можна перетягнути на вже записаний відеофайл [4].

Твітер – соціальна мережа мікроблогів, для надсилання коротких текстових повідомлень за допомогою повідамлень, служб миттєвого обміну повідомленнями та сторонніх клієнтських програм. Сегмент «Література Twitter» створив коротку текстову літературу, як Twitterature.

Twitter належить Twitter, Inc., штаб-квартира якого знаходиться у Сан-Франциско, Каліфорнія. Twitter, Inc. також має сервери та офіси в Сан-Антоніо, Техасі та Бостоні, штат Массачусетс. Станом на вересень 2019 року в компанії працює понад 4600 працівників (понад 900 у 2012 році).

Створений Джеком Дорсі в 2006 році, Twitter швидко став популярним у всьому світі. Станом на 1 січня 2011 року сервіс налічує понад 200 мільйонів користувачів. 100 мільйонів користувачів заходять щонайменше один раз на місяць, з них 50 мільйонів користуються Twitter щодня. 55% використовують Twitter для мобільних гаджетів, а близько 400 мільйонів унікальних дозволів здійснюються безпосередньо на сайті twitter.com щомісяця. З 18 по 19 листопада 2014 року хакери з анонімної спільноти пошкодили сайт і зробили його недоступним протягом кількох годин. Але проблему швидко вирішили, і сайт доступний для користувачів знову.

У 2010 році дохід Twitter, Inc. (від продажів в Інтернеті) досяг 45 мільйонів доларів.

До 7 листопада 2017 року максимальна кількість символів у повідомленні становила 140 символів. Збільшення ліміту до 280 знаків було прокоментовано розробниками як таке, що допоможе користувачам з латинськими та кириличними мовами рідше досягати ліміту під час написання твіта, чого майже не стається з користувачами ієрогліфічних мов [5].

23 липня 2023 Твітер оновлено до X - соціальної мережі, яка є мережею мікроблогів та надає можливість користувачам надсилати короткі текстові повідомлення (рис. 1.6).

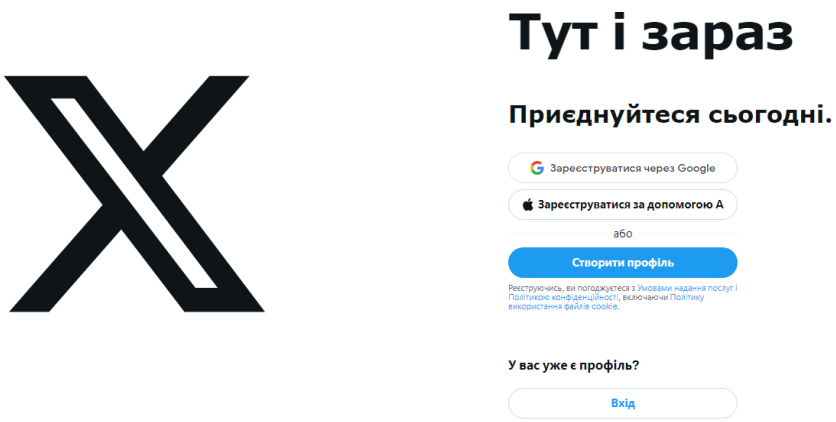

Рисунок 1.6 – Сторінка мережі «X»

Reddit – це розважальна програма, онлайн-сервіс новин та Інтернет-ЗМІ. Зареєстровані користувачі можуть додавати вміст, такий як текстові повідомлення та

прямі посилання для їх обговорення (BBS). Як і багато інших подібних сайтів, Reddit підтримує систему голосування за ваші улюблені повідомлення - найпопулярніші з них з'являються на головній сторінці сайту. Також найпопулярніші публікації відображаються вгорі кожної категорії. Вміст організовується через субреддіти. Тематика Subreddit досить широка: новини, ігри, фільми, музика, книги, спорт, їжа, фотографія та інше.

У 2015 році Reddit мав 542 мільйони користувачів щомісяця (234 мільйони унікальних користувачів), що робить 14-м у списку найбільш відвідуваних сайтів у США та 36-м у світі. Протягом 2015 року Reddit обслуговував 82,54 мільярда переглядів сторінок, 725,58 мільйонів коментарів та 6,89 мільярда рейтингів після публікації.

Reddit було засновано двома однокурсниками з Вірджинського університету Стівом Гафманом та Алексісом Оханяном у 2005 році. Condé Nast Publications придбало сайт у жовтні 2006-го. У вересні 2011-го Reddit став дочірньою компанією компанії-власника Condé Nast's, компанії Advance Publications. В серпні 2012 р. Reddit працював як незалежна одиниця, хоча Advance і досі є основним власником. Reddit базується в Сан Франциско. У жовтні 2014 року Reddit отримав 50 млн доларів інвестицій під керівництвом генерального директора бізнес-інкубатора Y Combinator Сема Олтмена та інвесторів Марка Ендерсона, Пітера Тіля, Рона Конвея, Снуп Дога та Джареда Лето. Їхні інвестиції оцінюються в 500 млн доларів [6].

LinkedIn − соціальна мережа для пошуку та встановлення ділових контактів. LinkedIn має 400 мільйонів зареєстрованих користувачів (станом на 2015 рік), що представляють 150 компаній у 200 країнах. Сайт відображається англійською, французькою, німецькою, італійською, португальською, іспанською та російською мовами.

LinkedIn також є платформою для обміну ідеями та знань. Користувачі можуть ділитися статтями, публікаціями, фотографіями та відео, що стосуються їхньої професійної сфери. Це створює можливість для обговорення тенденцій, отримання фахових порад та підвищення професійної видимості. Можна використовувати LinkedIn як інструмент для пошуку роботи та рекрутингу. Компанії розміщують

вакансії, а користувачі можуть шукати роботу з урахуванням свого досвіду та навичок. Рекрутери використовують платформу для знаходження кандидатів та взаємодії з потенційними працівниками. Крім цього LinkedIn надає можливість приєднатися до груп, спеціалізованих за галузями чи інтересами. Це створює унікальну можливість обговорювати теми, отримувати оновлення та розширювати свої знання в рамках конкретної індустрії.

13 червня 2016 року Microsoft оголосила про придбання LinkedIn за 26,2 мільярда доларів. Транзакція завершилася до кінця 2016 року.

Соціальна мережа LinkedIn була заснована Рейдом Хоффманом у грудні 2002 року, запущена в травні 2003 року. Нинішній керівник компанії — Джефф Вейнер, раніше представник правління Yahoo! Inc. Штаб-квартира компанії знаходиться в Маунтін-В'ю, Каліфорнія, США. LinkedIn має також офіси в Омасі, Чикаго, Нью-Йорку і Лондоні. Компанія була фінансована такими інвестиційними фондами як: Грейлок, секвоя Капітал, Бейн Капітал, Бессемер. 19 травня 2011 р. Linkedln нарешті провела заплановане первинне розміщення акцій (IPO) на Нью-Йоркській фондовій біржі, в ході якого інвесторам було продано 7,84 мільйона акцій по $45 за штуку [7].

1.2 Аналіз технологій для управління соціальною мережею

На даний момент у світі близько 6 найбільших соціальних мереж мають левову частку на ринку. Потрібна унікальна фішка і помітна назва для майбутньої мережі.

Для реалізації завдання є багато платформ керування та створення мережі:

1) Joomla! – відкрита універсальна система керування вмістом для публікації інформації в інтернеті. Підходить для створення малих та великих корпоративних сайтів, інтернет-порталів, інтернет-магазинів, сайтів громад та персональних сторінок. Особливості Joomla включають гнучкі інструменти управління обліковими записами, інтерфейс управління медіа, підтримка створення багатомовної сторінки, система управління рекламою, адресна книга користувача, голосування, вбудований пошук, класифікація посилань та облік кліків, редактор WYSIWYG, система

шаблонів, меню. Підтримка, управління каналами новин, XML-RPC API для інтеграції з іншими системами, підтримка кешування сторінок, великий набір додаткових додатків [8].

Joomla! написана на мові PHP з використанням архітектури MVC. Для збереження інформації використовується база даних MySQL, PostgreSQL чи MS SQL.

2) Drupal - це високопопулярна вільна та відкрита система управління контентом (CMS), яка базується на модульності і розроблена мовою програмування PHP. Започаткована у 2000 році, Drupal здобула велику популярність завдяки своїй гнучкості, розширюваності та потужним можливостям для створення різноманітних веб-сайтів.

Однією з ключових особливостей Drupal є його модульна структура, яка дозволяє розробникам і користувачам легко розширювати функціональність за допомогою модулів. Модулі в Drupal представляють собою невеликі блоки коду, які можна включати або виключати в залежності від конкретних потреб веб-сайту.

Крім того, Drupal відомий своєю потужною системою управління правами доступу, що дає можливість точно налаштовувати, хто і як може взаємодіяти з різним вмістом веб-сайту. Це особливо корисно для великих та складних проєктів з численними користувачами і адміністраторами.

Drupal підтримує широкий спектр типів веб-сайтів, від особистих блогів до корпоративних порталів та інтернет-магазинів. Його спільнота розробників активно співпрацює над постійним вдосконаленням системи, виправленням помилок та випуском нових версій для вдосконалення користувацького досвіду та забезпечення високої якості веб-проєктів. Drupal може фактично працювати на будь-якій платформі, яка підтримує веб-сервери Apache, Nginx, Lighttpd або Microsoft IIS. Також потрібна система управління базами даних MySQL / MariaDB, PostgreSQL 8.3, SQLite або інші комерційні продукти [8].

3) Social Engine - це потужна система управління вмістом, розроблена на мові програмування PHP і спеціально орієнтована на створення та управління соціальними мережами. Завдяки своїм розширеним можливостям та гнучкості, SocialEngine став вибором для тих, хто прагне створити власну унікальну соціальну платформу.

Основні функції SocialEngine включають не лише ефективне управління соціальною мережею, але й широкий спектр параметрів налаштувань для персоналізації та пристосування. Незашифрований код дозволяє розробникам вносити зміни та розширювати функціональність відповідно до унікальних потреб проєкту.

Однією з переваг є багатомовні функції, які роблять SocialEngine глобально орієнтованою платформою, дозволяючи користувачам спілкуватися і співпрацювати мовами свого вибору.

SocialEngine підтримує розширення функціональності через плагіни та віджети. Це дозволяє розробникам та адміністраторам впроваджувати нові функції, що відповідають конкретним потребам їхніх соціальних мереж.

Крім того, наявні численні шаблони та доповнення, які полегшують розгортання та розробку. Це дозволяє адміністраторам швидко налаштовувати вигляд та функціоналість своєї соціальної мережі, використовуючи готові рішення або вносячи індивідуальні зміни.

SocialEngine написаний на PHP та використовує базу даних MySQL. Крім того, на вашому веб-сервері повинні бути встановлені бібліотеки GD та mod_rewrite.

Але хоча цей рушій не позбавлений недоліків, на мою думку, головне - платити за багато додатків, плагінів та скриптів, без яких неможливо було б створити ресурси на цьому рівні.

Незважаючи на все різноманіття рушіїв для створення соціальних мереж, найбільш широкі можливості з розробки індивідуальних соціальних мереж надати може лише веб-програмування вручну. Індивідуально розроблені для соціальних мереж системи управління дозволяють здійснювати створення соціальної мережі, яка максимально повно відповідає її цілям і завданням, запропонувати майбутнім учасникам зовсім нові можливості [8].

1.3 Постановка задачі дослідження

Задача даної роботи полягає у розробці інформаційної технології управління соціальною мережею. Проаналізувавши переваги та не доліки програм аналогів управління соціальною мережею, сформуємо вимоги до клієнтської та серверної частин інформаційної технології управління соціальносю мережею.

Вимоги до клієнту:

- вікно реєстрації;
- вікно входу;
- головне вікно користувача;
- фото профілю;
- блог;
- інформація користувача;
- можливість редагування інформації;
- завершення сесії (вихід).

Вимоги до серверу:

- зберігання даних користувача через зв’язок з базою даних;
- генерація токена для дозволу запитів залогованого користувача;
- видача необхідних даних по запиту від клієнта.

1.4 Висновки до розділу 1

В даному розділі проаналізовано та досліджено використання соціальних мереж, наведений їхній опис та цільова аудиторія використання.

Проведено аналіз існуючих соціальних мереж, описано їх переваги та недоліки. Описано основні програмні засоби для створення та управління соціальними мережами. Здійснено постановку задачі дослідження, описано вимоги для клієнта, сервера.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ СОЦІАЛЬНОЮ МЕРЕЖЕЮ

2.1 Обгрунтування засобів розробки соціальної мережі

Протягом десятиліть розробники намагалися помістити пов’язані, напівструктуровані набори даних у реляційні бази даних. Це пов’язано з переходом паперових форм і табличних структур на оцифровану версію, що дозволяє швидко шукати всі дані та вводити нові відповідно до певної структури – це те, що реляційні бази даних роблять виключно добре. Моделювання даних за конкретними правилами їх зберігання дозволяє уникнути помилок при введенні нових даних і розвитку міжпанельної комунікації.

2.1.1 Аналіз типів баз даних

Однак у ситуаціях, коли набір даних стає більш складним і менш однорідним, реляційна модель перевантажується великими таблицями, і зазвичай рядки не завжди мають заповненими всі стовпці, що призводить до перевірки записів – логіки перевірки нуля. У міру зростання такої бази даних покращені з’єднання знижують продуктивність і ускладнюють розробку існуючої моделі відповідно до потреб бізнесу (рис. 2.1) [7].

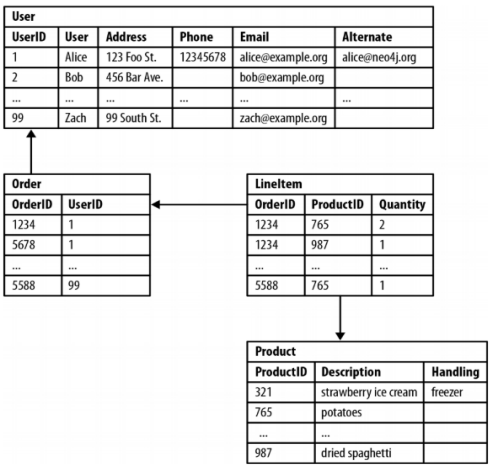

Рисунок 2.1– Реляційна база даних

Реляційні бази даних мають сильно пов’язані домени. Слід розуміти, що така архітектура має високу вартість виконання пов’язаних запитів у реляційній базі даних. Розглянемо кілька запитів у домені соціальної мережі [7].

Слабкість реляційної бази проявляється в необхідності переходу від однієї схеми до іншої, коли програма набуває нових масштабів.

Таким чином, реляційна база даних є ідеальним рішенням для проектів, які мають точну структуру даних, яка може лише змінюватися з часом, але не змінюватися, в якій потрібно мати зразки у вигляді статистики, обліку тощо. Однак використання реляційних баз даних не доцільно для побудови додатків, які працюють з неструктурованими даними, в яких важливо не об’єднання записів, а швидкість їх повернення сервером користувачеві на екран його смартфона.

Існує ще один тип баз даних, який повністю відрізняється від реляційних баз даних - база даних NOSql або бази даних, подібних до документів.

Більшість баз даних NOSQL орієнтовані на ключ-значення, документ або стовпець. Як і будь-який тип баз даних, ця модель має свої позитивні та негативні сторони при застосуванні [8].

На відміну від реляційних баз даних, документоподібні бази даних не мають жорсткої системи однотипної структури даних. Замість таблиць існують певні «каталоги», які містять масиви об’єктів, кожен із власним ключем для прямого доступу. Отже, якщо ми хочемо отримати дані для конкретного користувача соціальної мережі, знаючи його ідентифікатор, ми можемо отримати його дані безпосередньо, посилаючись на його індекс масиву [8]. Завдяки цій архітектурі ми можемо швидко, без додаткової обробки сервера, отримати всі дані для заповнення інтерфейсу користувача. Для веб-додатків він ідеально підходить для роботи в мережі. Адже всі дані надходять із сервера у вигляді об’єктів у форматі JSON, з яким дуже зручно працювати в JavaScript, їх структура в більшості випадків не потребує розбору [8].

Ця архітектура ускладнює їх використання для пов’язаних даних і графіків. Однією з добре відомих стратегій додавання зв’язків до таких сховищ є вбудовування ідентифікатора агрегату в поле, що належить іншому агрегату, – ефективне введення

зовнішніх ключів. Однак для цього потрібно підключати блоки на програмному рівні, що швидко стає непомірно дорогим [4].

У цій системі є ще одна слабкість. Оскільки не існує ідентифікаторів для кожного запису як зовнішнього ключа, вбудованого в базу даних у рамках процесу об’єднання різних даних, розробник втрачає можливість створювати запити відображення в базі даних. У більшості випадків програми вимагають не стільки даних, скільки окремих моделей, скільки їх зв’язку з іншим інтерфейсом. Тому нам доводиться постійно змінювати структуру в процесі вбудовування нових функцій у робочий процес програми, що призводить до альтернативних рішень, таких як дублювання даних, що призводить до необхідності збільшення пам’яті для клонування ідентичних даних, але іншої логіки. Наприклад, запит-рекомендація на отримання певного виду товару на основі попередніх записів подібних покупок. Побудова такого запиту в базі даних NoSQL буде дуже витратною з точки зору процесора, який був розроблений з урахуванням підходу «отримувати-віддавати». Зрештою, весь алгоритм буде залежати від оперативної пам’яті сервера.

Для такої системи, як соціальна мережа, кількість користувачів дуже значна, тому при недостатніх ресурсах пам’яті може не вистачити на всі з’єднання, що в свою чергу може призвести до збою всієї системи.

Якщо ми все ще хочемо використовувати це рішення для роботи з даними, необхідно експортувати набір даних і обробити його якоюсь зовнішньою обчислювальною інфраструктурою для обчислення результатів, що, у свою чергу, призводить до нового ланцюжка подій у коді. Також варто відзначити, що передача та обробка даних на сервері завжди супроводжується низкою перевірок помилок. Мінімальна помилка - це втрата зв'язку між клієнтом і сервером. Накладення перевірок на такі помилки слід передбачити в кожному методі обробки даних. Збільшення кількості таких перевірок суттєво знизить ефективність виконання коду за одиницю часу, який не реагує жодною стороною. Обробка помилок завжди є слабким місцем будь-якої архітектури. Тому їх накопичення вимагає лише переробки алгоритму операцій з даними. Крім того, ми можемо ретроспективно вставити інструкції від зовнішніх блоків у протилежному напрямку, а потім запитати

результат. У будь-якому випадку результати будуть приховані. Окремі документи з даними не підтримують узгодженість їхніх відносин і не підтримують так званий індекс вільного сусідства, у якому елементи містять прямі посилання на своїх сусідів [8].

Попередні приклади мали справу з неявно пов’язаними даними. Як користувачі, ми отримуємо семантичні зв’язки між сутностями, але самі моделі даних і бази даних невидимі для цих зв’язків. Програма повинна створити мережу з найбільш доступними даними. Чого ми дійсно хочемо досягти, так це узгодженого перегляду даних, включаючи зв’язки між елементами. На відміну від сховищ даних, які описані в реляційних базах даних і NoSQL, в системі графічних баз даних з'єднання даних зберігаються як самі реальні дані. Де є з'єднання в домені і є дані в сховищі бази даних. Для прикладу розглянемо соціальну мережу, показану на рис. 2.2 [5].

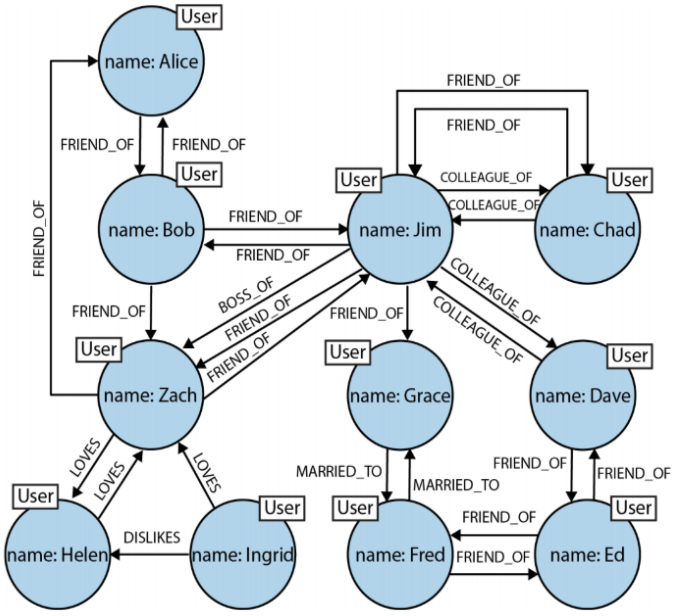

Рисунок 2.2 – Модель графового представлення бази даних

У цій соціальній мережі, як і в багатьох реальних випадках зв’язаних даних, зв’язки між сутностями не демонструють однорідності в межах домену, дані структуровані. Соціальна мережа є популярним прикладом тісно пов’язаних даних, мережа зі змінною структурою, яка може не існувати в єдиній схемі, яка підходить для всіх, або зручно розподілятися між роз’єднаними гілками даних. Наша проста

мережа друзів зросла, тепер є потенційні друзі до шести рівнів глибини запитів і лише більший набір даних. Гнучкість графічної моделі дозволила нам додавати нові вузли та нові зв'язки без шкоди для існуючої мережі чи міграції даних – вихідні дані та їх архітектура залишаються незмінними.

Система керування базою даних діаграм — це онлайн-система керування базою даних із методами створення, читання, оновлення та видалення, які розкривають модель даних діаграми. Графічні бази даних зазвичай будуються для використання з транзакційними системами [9].

Відносини є основним модулем Графічної моделі даних. Збираючи прості абстракції вузлів і зв'язків у пов'язані структури, бази даних графів дозволяють нам створювати як завгодно складні моделі, які тісно пов'язані з нашою проблемною областю. Отримані моделі є простішими та виразнішими, ніж моделі, створені з використанням традиційних реляційних баз даних та інших репозиторіїв NOSQL [9].

База даних діаграм пропонує набагато глибшу картину розвитку мережі з подібними комбінаціями даних. Ми бачимо, хто кого «любить» і чи взаємна ця «любов».Ми бачимо, хто такий COLLEAGUE_OF, хто і хто BOSS_OF з усіх користувачів. У наших соціальних мережах ми навіть бачимо антисоціальні елементи, представлені асоціацією DISLIKES. Маючи такий графік, ми маємо можливість розглянути переваги графічних баз даних у роботі пов'язаних даних [9].

Ми часто хочемо класифікувати вузли в наших мережах відповідно до їх ролі в графіках. Наприклад, деякі сайти можуть представляти користувачів, а інші можуть відображати замовлення або продукти. У системі баз даних графіків Neo4j ми використовуємо мітки для представлення ролей, які виконує вузол у графі. Оскільки вузол може мати кілька різних ролей у графі, Neo4j дозволяє нам додати більше однієї мітки до вузла.

Збираючи прості абстракції вузлів і зв'язків у пов'язані структури, бази даних графів дозволяють нам створювати як завгодно складні моделі, які тісно пов'язані з нашою проблемною областю. Отримані моделі є простішими та виразнішими, ніж ті, що створюються за допомогою традиційних реляційних баз даних та інших сховищ NOSQL.

Використовуючи мітки таким чином, ми можемо згрупувати вузли. Ми можемо отримати інформацію в базі даних, наприклад, де знайти всі вузли, позначені як «Користувач».

Зв’язки в діаграмі можна легко інтуїтивно побудувати, зрозумівши, як дані просто пов’язані з точкою зору людини як генератором базової моделі. Завдяки орієнтованості на правила характеру моделі даних більшість операцій з базою даних графів на основі шляхів зрозумілі як програмісту, так і пересічній людині, яка бажає створити таку базу даних самостійно, що робить їх надзвичайно ефективними.

Той факт, що графічна база даних забезпечує потужну, але нову техніку моделювання даних, сам по собі недостатній для того, щоб виправдати заміну добре відомої, добре відомої платформи даних; також мають бути негайні й дуже значні практичні вигоди. На додаток до цих переваг продуктивності, графічні бази даних пропонують надзвичайно гнучку модель даних, а також метод доставки, що відповідає сучасним практичним методам доставки програмного забезпечення [9].

Однією з важливих причин вибору графічної бази даних є підвищення продуктивності під час роботи з пов’язаними даними та реляційними базами даних і сховищами NOSQL.На відміну від реляційних баз даних, де продуктивність запитів через з’єднання значно погіршується в міру зростання набору даних, продуктивність баз даних графіків, як правило, залишається відносно постійною навіть у міру зростання набору даних. Це тому, що запити локалізуються після частини розкладу. В результаті час виконання кожного запиту пропорційний лише розміру частини діаграми, яка відповідає цьому запиту, а не розміру всієї діаграми [9].

Ми хочемо мати можливість розвивати нашу модель даних відповідно до решти нашої програми, використовуючи технології відповідно до сучасних, покрокових і ітеративних методів доставки програмного забезпечення. Сучасні графічні бази даних дозволяють нам розробляти та підтримувати системи. Зокрема, схематичний характер графічної моделі даних у поєднанні з перевіреністю інтерфейсу програмування програми графічної бази даних та мови запитів дає змогу розробляти програму більш керовано.

2.1.2 Використання нейронних мереж для управління соціальною мережею

Нейронні мережі — це математичні моделі, які працюють за принципом мережі нервових клітин тваринного тіла. Щоб полегшити сприйняття, нейрон можна розглядати як клітинку з багатьма входами та одним виходом. Алгоритм розрахунку визначає кількість створених вхідних сигналів. Кожному нейронному входу призначаються ефективні значення, які потім поширюються через міжнервні зв'язки - реферати. Синапси мають один параметр — вагу, тому вхідна інформація змінюється під час переходу від одного нейрона до іншого. Найпростіший принцип роботи нейронних мереж можна проілюструвати на прикладі змішування кольорів. Синій, зелений і червоний нейрони мають різну вагу. Інформація про нейрон, вага якого буде домінувати в наступному нейроні [10].

Основною перевагою нейронних мереж перед звичайними обчислювальними алгоритмами є їх здатність до навчання. Загалом наука полягає в пошуку правильних коефіцієнтів зв’язку між нейронами, а також узагальненні даних та ідентифікації складних взаємозв’язків між вхідними та вихідними сигналами. Насправді, успішне навчання нейронних мереж означає, що система зможе виявити правильний результат на основі відсутніх даних у навчальній вибірці.

Залежно від діапазону нейронної мережі його можна інтерпретувати по-різному, наприклад, у сфері машинного навчання, ANN є методом розпізнавання шаблонів. З математичної точки зору - це багатопараметричне завдання. З точки зору кібернетики - модель управління адаптивною робототехнікою.У випадку AI, INS є ключовим компонентом моделювання природного інтелекту за допомогою обчислювальних алгоритмів.

Кожному нейронному входу призначаються ефективні значення, які потім поширюються через міжнервні зв'язки - реферати. Синапси мають один параметр — вагу, тому вхідна інформація змінюється під час переходу від одного нейрона до іншого. Найпростіший принцип роботи нейронних мереж можна проілюструвати на

прикладі змішування кольорів. Синій, зелений і червоний нейрони мають різну вагу. Інформація про нейрон, вага якого буде домінувати в наступному нейроні [10].

Щоб використовувати архітектуру нейронної мережі в JavaScript, потрібно підключити окрему бібліотеку програмування – TensorFlow.

TensorFlow - це багатозадачна бібліотека машинного навчання з відкритим вихідним кодом, розроблена Google для задоволення потреб систем, здатних будувати й навчати нейронні мережі для виявлення й декодування зображень і кореляцій, подібних до навчання та розуміння людини. Нині Google використовується як для досліджень, так і для розробки продуктів [11].

У той час як еталонна реалізація працює на окремих пристроях, TensorFlow може працювати на кількох ЦП і графічних процесорах, включаючи додаткові обчислювальні розширення загального призначення на GPU. TensorFlow доступний для 64-розрядних операційних систем Linux, macOS, Windows і мобільних обчислювальних платформ, включаючи Android та iOS [11].

Обчислення TensorFlow виражаються як стани графіка потоку даних. Назва TensorFlow походить від операцій, які нейронні мережі виконують над багатовимірними наборами даних. Ці багатовимірні масиви називаються «тензорами» [12].

2.1.3 Асинхронний спосіб роботи з даними в соціальній мережі

Перед створенням соціальної мережі розглянемо важливе архітектурне рішення для роботи з даними, наприклад використання асинхронних функцій.

Синхронна програма складається з послідовності викликів синхронних функцій, при цьому одночасно виконується лише одна з синхронних функцій. Коли ми переходимо до наступної функції в послідовності, ми завжди знаємо результат попередньої. Цей процес показано на рис. 2.3 [13].

```
console.log("sync 1"); /* output */ sync 1
console.log("sync 2"); sync 2
console.log("sync 3"); sync 3
console.log("sync 4"); sync 4
```

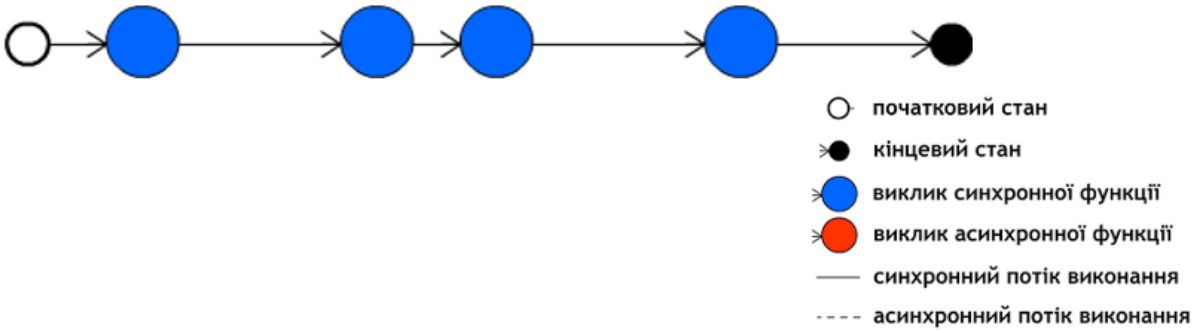

Рисунок 2.3 – Схема синхронної роботи фрагменту програми

Деякі функції у послідовності викликів можуть мати асинхронний характер. Цей процес зображено на рис. 2.4.

```
console.log(1);  // sync
console.log(2);  // sync
setTimeout(() => console.log("Async 1"), 0);  // async function
console.log(3);  // sync
```

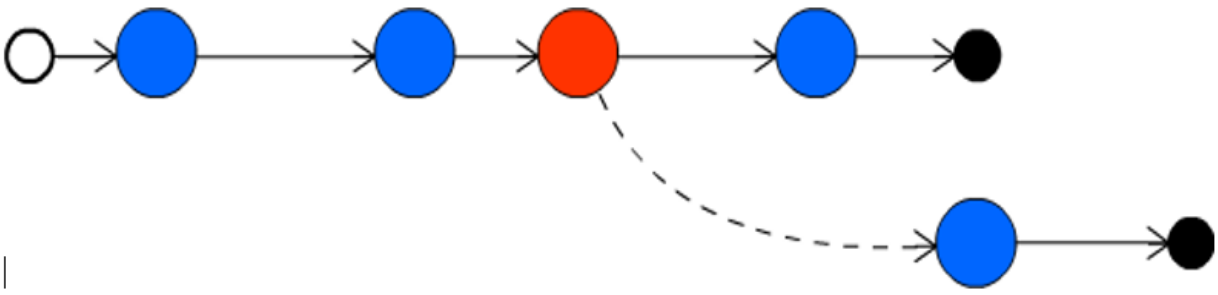

Рисунок 2.4 – Схема асинхронної роботи фрагменту програми

Такі функції додатково запускають виконання асинхронної операції, результат якої доступний не відразу, а обчислюватиметься паралельно. Виклик асинхронної функції не чекає завершення асинхронної операції, а повертає синхронний результат (завершення виклику) і продовжує основний синхронний потік програми.

Можна перевірити, чи є функція асинхронною, у відповідній документації, але зазвичай асинхронні функції використовують одну або кілька функцій Callback як один з аргументів для обробки результату асинхронної операції.

Поширеною практикою в JavaScript є передача Callback функції, як функції двох значень: err — опис можливої помилки виконання та date — результат асинхронної операції, якщо її виконання не призвело до помилки [13].

2.1.4 Масштабування сервера

Характерною проблемою для додатків соціальних мереж є швидкий приріст користувачів системи, що в свою чергу збільшує кількість мережевих підключень до сервера прямо пропорційно для отримання результатів.

Якщо архітектура проекту не передбачає процесу масштабування сервера - його вихід з ладу гарантований при високому навантаженні.

Код сервера, що дозволяє розвивати розвинену соціальну мережу, заснований на Javascript з ядром V8 від Google. JavaScript на стороні сервера, відомий як NodeJS, має відповідні модулі для масштабування ЦП машини, на якій вона працює.

Сервер, написаний на Node.js, зазвичай дуже швидкий, частково завдяки неблокуючому вводу/виводу (введення/виведення), а також завдяки добре оптимізованому механізму v8. Єдиний момент полягає в тому, що JavaScript використовує лише одне ядро ЦП, але ви можете виправити це і користуватися всіма перевагами багатопроцесорного середовища [14].

Слабким місцем Node.js є загальність стану та колективної пам’яті. Існує рішення проблеми спільного використання пам’яті [14].

Коли програма виконується не в одному процесі, а в багатьох, відстеження цього стану більше не є безмежним, це абсолютно нетривіальне завдання. В ідеальному світі програма повністю без громадянства, і клієнта зовсім не хвилює процес, за допомогою якого надсилаються його запити. На практиці такої поведінки досягти дуже важко: сеанси користувачів, падіння пулу підключень тощо [15].

Для вирішення таких проблем Node.js надає можливість обмінюватися повідомленнями між процесами. Це можна легко використовувати для простих програм - ми будемо надсилати сповіщення всім процесам зміни статусу.

Цей код може використовуватися у вашій програмі, але він не гарантує безпеку. Наприклад, якщо Worker втрачає повідомлення або одержимий його отриманням, ми отримуємо помилку, що стан стану програми відсутній, причину якого нелегко зрозуміти, а ще важче виявити. Оскільки наша програма розростається від одного процесу до кластера процесів, складність реалізації такого зв'язку сповіщень стає нетривіальним завданням.

Найкраще рішення — налаштувати функціональність для роботи зі станами у серверній частині. Таким чином ми переносимо цю роботу з рівня програми на рівень пам'яті, що дозволить розподілити обов'язки між процесорами.

Масштабування в Node.js не запізнюється. Це те, що захоплено в ядрі середовища виконання. Вузол називається Node, щоб підкреслити, що програма Node повинна містити кілька невеликих розподілених вузлів, які взаємодіють один з одним [13].

Ми запускаємо процес Node на кожному ядрі ЦП виробничих машин і балансуємо навантаження всіх запитів між ними. У Node є вбудований модуль для роботи з такою архітектурою. Модуль кластера Node не тільки надає нестандартне рішення для використання повної потужності ЦП машини, але також допомагає підвищити доступність процесів Node і надає можливість перезапускати всю програму з нульовим простоєм [13].

Ми також можемо масштабувати програму, розкладаючи її на основі функціональності та послуг. Це означає наявність кількох різних програм з різними кодами, а іноді і власними спеціальними базами даних та інтерфейсами користувача.

Цю стратегію зазвичай пов'язують з терміном «мікросервіс», де мікро – означає, що цих сервісів має бути якомога менше, але розмір сервісу не має особливого значення, а скоріше забезпечення вільного зв'язку та високої узгодженості між сервісами. Реалізація цієї стратегії часто буває нелегкою і може

призвести до довгострокових несподіваних проблем, але якщо все робити правильно, переваги величезні.

Ми також можемо розділити програму на кілька модулів, коли кожен екземпляр відповідає лише за частину даних програми. Цю стратегію часто називають горизонтальним розділенням або фрагментацією в базах даних. Розділення даних вимагає кроку пошуку перед кожною операцією, щоб визначити, який екземпляр програми використовувати.

2.2 Розробка діаграм варіантів використання

При створенні соціальної мережі буде створено кілька класів користувачів: гість, зареєстрований користувач, адміністратор.

Гість може лише переглядати сторінки та шукати інформацію, він не може ділитися своїми повідомленнями (рис. 2.5).

Зареєстрований користувач може створювати, редагувати, видаляти повідомлення, додавати контент, спілкуватися, знаходити необхідну інформацію та переглядати сторінки (рис. 2.6).

Основними обов'язками адміністратора можна назвати інформаційну безпеку, налаштування бази даних, визначення ефективності баз даних (рис. 2.7).

Діаграма варіантів використання представлена у вигляді графіка.

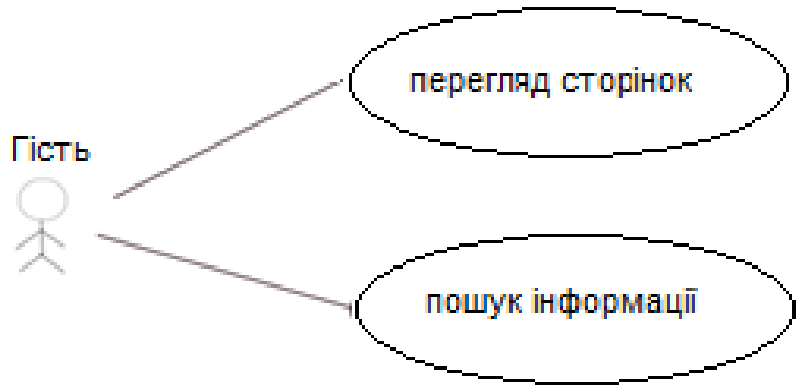

Рисунок 2.5 – Діаграма варіантів використання для класу «Гість»

Рисунок 2.6 – Діаграма варіантів використання для класу «Зареєстрований користувач»

Рисунок 2.7 – Діаграма варіантів використання для класу «Адміністратор»

2.3 Розробка структури інформаційної технології управління соціальною мережею

Структура веб-додатку показує складові додатку, всі його сторінки, ресурси, використані стилі, їх взаємодія між собою та зв’язок. Вона повинна бути чіткою та логічною, уникати надлишковості, зайвих посилань. При цьому треба забезпечити доступ до основних підрозділів веб-додатку з будь-якої його сторінки (рис. 2.8).

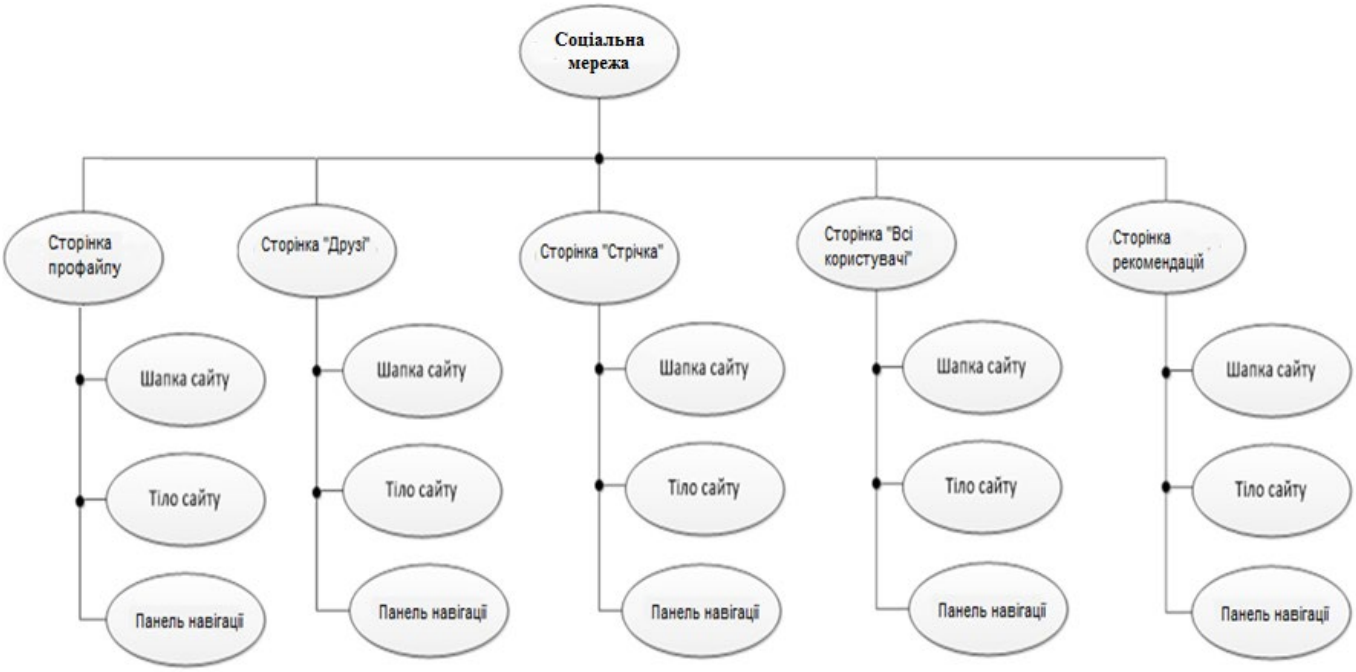

Рисунок 2.8 – Структура розроблювального програмного забезпечення

Отже, інформаційна технологія управління соціальною мережею буде реалізована у вигляді веб-ресурсу, структура якого та навігація між сторінками є зрозумілою для користувача, що робитиме роботу з ним зручною та ефективною.

2.4 Розробка бази даних для соціальної мережі

Діаграма «сутність-зв’зок» або ER diagram - це модель даних, яка дозволяє описувати концептуальні діаграми за допомогою узагальнених конструкцій. Основні

конструкції цієї моделі - сутності і зв'язки. ER Модель E - це метамодель даних, тобто спосіб опису моделей даних. Існує багато моделей представлення знань, але одним із найзручніших інструментів для уніфікованого представлення даних, незалежно від програмного забезпечення, що його реалізує, є модель зв’язку сутність (рис. 2.9).

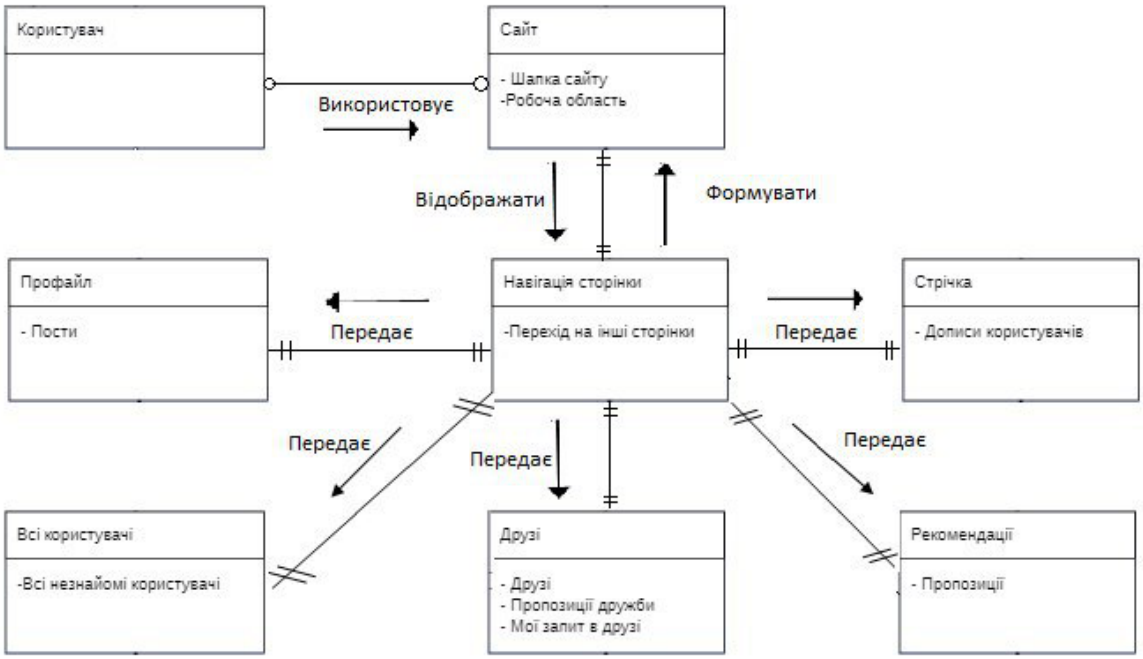

Рисунок 2.9 – Діаграма «сутність-зв'язок» інформаційної технології управління соціальною мережею

Щоб створити базу даних, потрібно запустити хостинг, який підтримуватиме онлайн-середовище. Будемо використовувати графову базу даних Neo4j (рис. 2.10). Для створення середовища використовуємо безкоштовний ресурс https://app.graphenedb.com/.

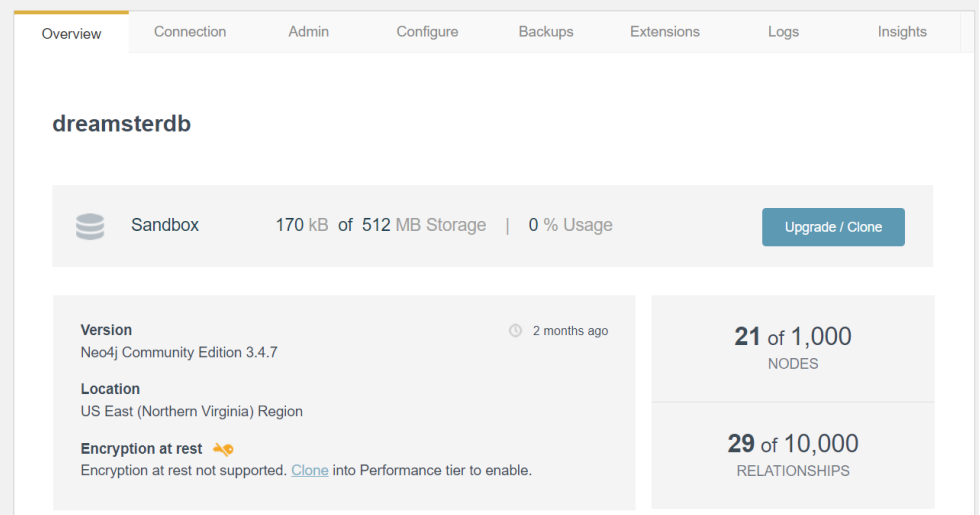

Рисунок 2.10 – Середовище створення графової бази даних

Це середовище забезпечує візуальний інтерфейс для розробки даних у моделі діаграми. Але для створення самих доменів діаграми необхідно використовувати мову Cypher. Для підключення сервера до середовища баз даних використовується пакет драйверів, який бере на себе функції URL-адреси бази даних, пароля та ключа доступу до API.

const url = 'bolt://hobby-elikibjimasdedcl.dbs.graphenedb.com:2773';

const driver = neo4j.drivers(url, neo4j.auth.basic('us-admin', 'b.07qax7z5R.Fgf35jasdaHzNU'));

const session = drivers.session();

Для формування записів використаємо мову Cypher – мову запитів до графової бази даних. Спочатку створимо користувачів соціальної мережі (рис. 2.11):

MERGE (u:User {name:'Boris', email:'Britva@gmail.com', picture:'https://google.image', idToken:'2sssala55'}) RETURN u`)

Даний скрипт створить модель даних з інформацією про певного користувача: ім’я, електронну пошту, картинку для аватарки та унікальний його токен для доступу. Після цього запит поверне цей запис для подальшої роботи з ним.

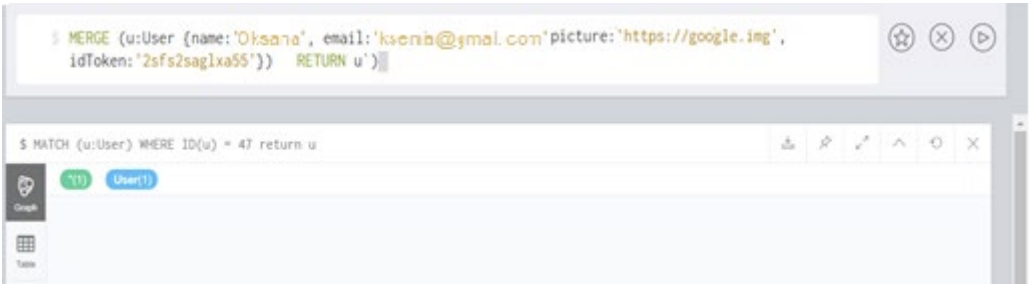

Рисунок 2.11 – Створення нового запису в графовій базі даних

Варто відзначити, що користувачі в базі даних мають свій особливий тип домену - User. Завдяки різним типам доменів модель діаграми має чітке розуміння того, на чому їй потрібно буде зосередитися в майбутніх записах.

Тепер потрібно створити запис для існуючого користувача - Boris.

Цей запит має створити новий запис із текстом «Я йду на концерт» і автоматичним часом, згенерованим базовим середовищем (рис. 2.12):

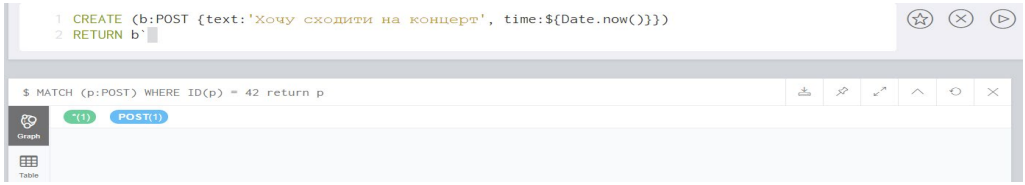

Рисунок 2.12 – Створення допису користувача в графовій базі даних

Після того як у нас є користувач та його запис, необхідно ці дані з’єднати між собою в такому відношенні: користувач Boris є “власником” запису (рис. 2.13).

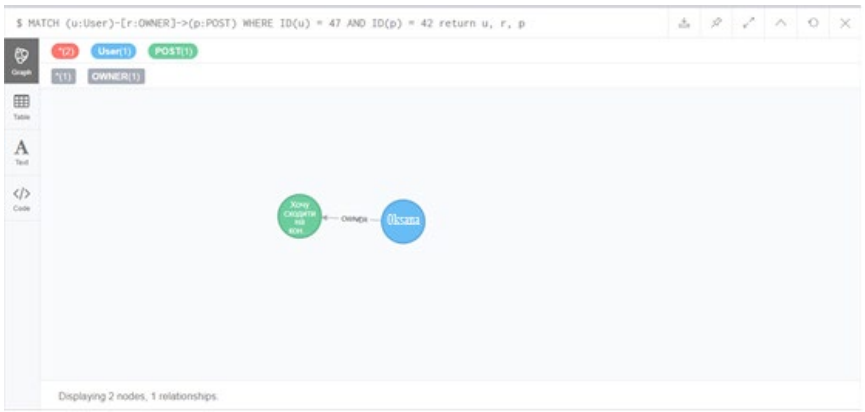

Рисунок 2.13 – Створення зв’язку між користувачем і його записом в графовій базі

Ця схема не є інтерфейсом для відображення підключень даних, це їх власна архітектурна комбінація. Два домени "POST" і "User" пов'язані відношенням "OWNER", яке залишає користувача та переходить до запису. Траєкторія спілкування дуже важлива, тому що, коли ми запитуємо дані, ця траєкторія впливає на результати.

Основним алгоритмом куревання соціальною мережею є підключення користувачів у відношеннях «друзі». Також варто відзначити, що цей зв’язок може бути багатостороннім або тільки одностороннім. Тому необхідно розробити обидва випадки об’єднання таких даних.

Маючи ідентифікатори користувачів, ми повинні побудувати між ними зв’язок (рис. 2.14).

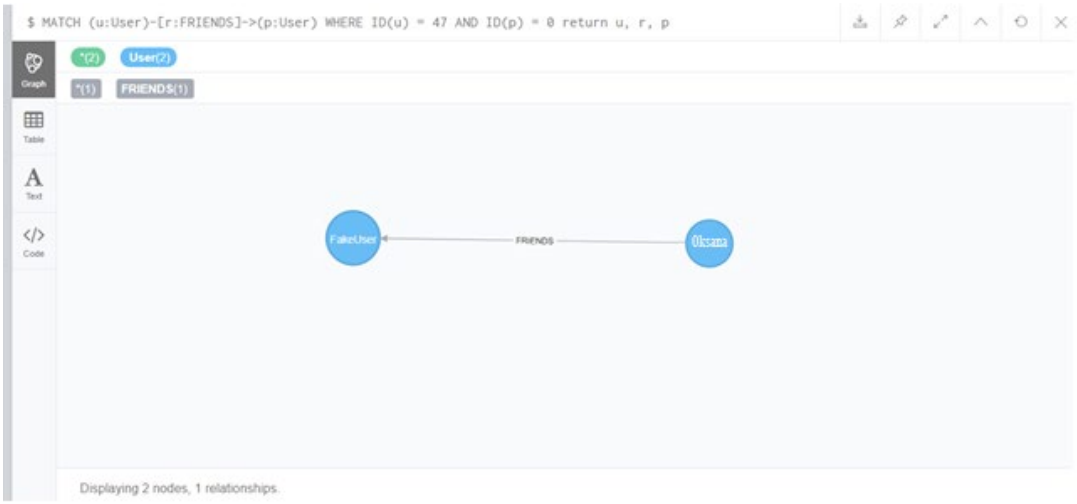

Рисунок 2.14 – Створення зв’язку між користувачами

Скрипт для поєднання у такий “дружній” зв’язок має вигляд (рис. 2.15):

MATCH (u:Users), (u2:Users) WHERE ID(u) = ${me} AND ID(u2) = ${friends_id} MERGE (u)<-[r:FRIEND]->(u2) return u, r, p

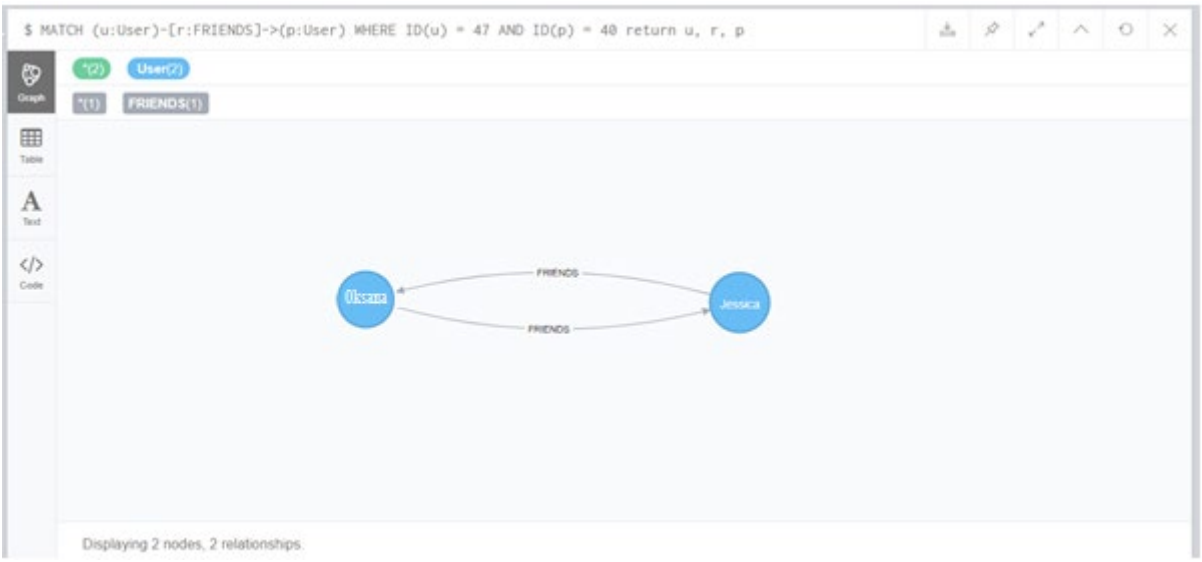

Рисунок 2.15 – Пошук відношення «друзі» в базі даних

Інший домен, існування якого пов’язане з подальшим використанням нейронної мережі як рекомендаційної системи, — це створення різноманітних напрямків роботи користувачів або компаній, розташованих у соціальній мережі. Ці «сфери» потрібно позначити, щоб нейронна мережа могла навчитися будувати нові зв’язки між даними. Теги слів слід розділити на різні типи словотворення, такі як: об’єкти, дієслова, вказівники. Коли користувач створює допис, парсер, створений на сервері, повинен проаналізувати текст і розбити його на різні частини речення.

Але головна мета домену графа - мати правильні теговані слова, щоб ми створювали робочий простір для компаній і користувачів. Домен, наприклад, може відповідати за організацію покупки квитків на концерти (рис. 2.16).

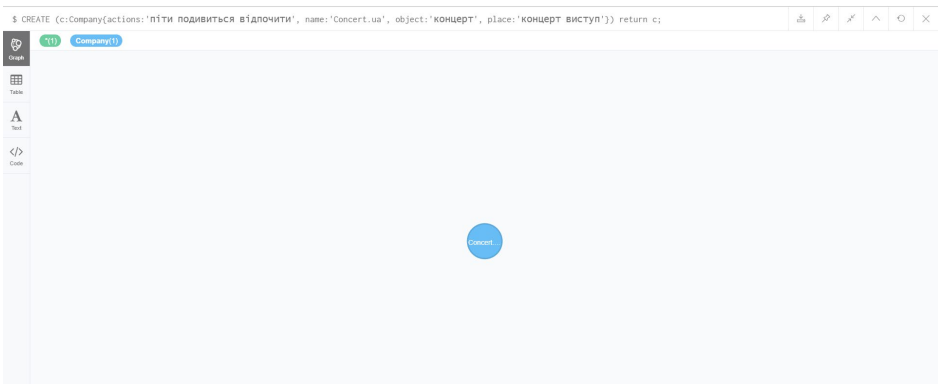

Рисунок 2.16 – Створення інформації про компанію

В результаті роботи таких запитів на створення певних записів та зв’язківк у базі даних, ми отримаємо повну картину графової моделі даних (рис. 2.17).

Рисунок 2.17 – Повна графова модель бази даних

Згенерованих даних достатньо, щоб отримати загальну картину архітектури бази даних і перевірити середовище з точки зору виконання запитів шифрування. Ця схема забезпечує всі необхідні підключення даних, а також відповідні домени діаграми.

Якщо є зміни в бізнес-логіці, проблеми зі зміною зв’язків або структури даних не вирішуються. Тому модель діаграм ідеально підходить для побудови архітектури даних соціальних мереж.

2.5 Розробка файлової структури програмного забезпечення управління соціальною мережею

Цей проект міститиме файли і серверної частини і клієнтської частин. Клієнтська частина міститиме всі статичні файли, необхідні для інтерфейсу програми та його фактичної логіки інтерфейсу, а також необхідний список пакетів. Статичні файли включають зображення, значки, файли стилів і файли JavaScript.

Серверна частина програми матиме один файл як точку входу в наш API - сервіс і зібраний проект для реалізації клієнтської частини.

Завдяки цьому наш сервер має прямий доступ до домену веб-додатку, щоб зробити всі статичні файли програми доступними для користувачів.

Папка FrontEnd м істить всю архітектуру клієнтської частини програми (рис. 2.18).

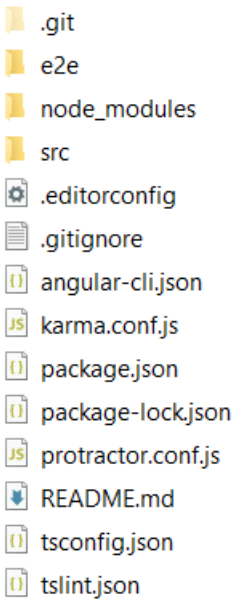

Рисунок 2.18 – Файлова структура front-end частини веб-додатку

Файли в папці BackEnd містять всю бізнес-логіку сервера та моделі обробки даних, а також скомпільовані та згорнуті файли інтерфейсу програми (рис. 2.19).

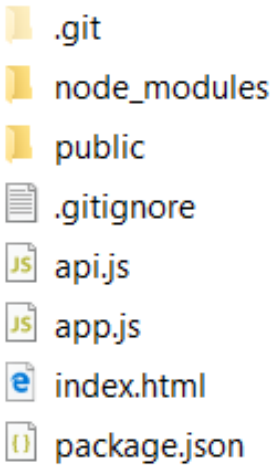

Рисунок 2.19 – Файлова структура back-end частини

Варто зазначити, що папка «public» — це прихована частина, яка призначена для клієнтської частини, але мінімальна і готова до роботи на сервері.

Для реалізації серверної частини програми буде використовуватися NodeJS - сервер JavaScript, тобто його фреймворк - Express.

Щоб легко працювати зі створенням коду для серверної обробки, вам потрібно налаштувати середовище.

Встановлюємо пакет Nodemon, що дозволить нам відстежувати зміни у файлах проекту та автоматично перекомпілювати проект (рис. 2.20).

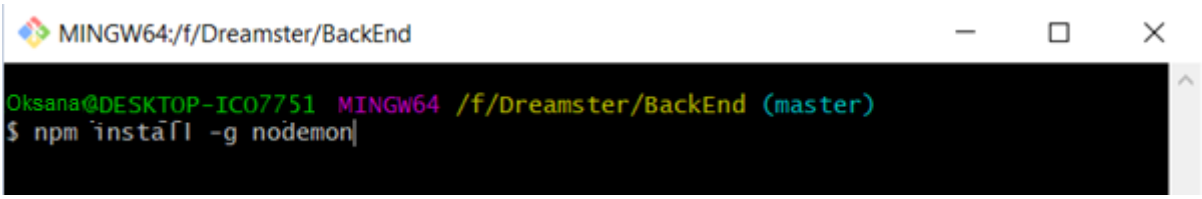

Рисунок 2.20 – Встановлення пакету Nodemon

Потім необхідно запустити сам пакет, перед цим вказати точку входу для створеного сервера api.js, nodemon api.js (рис. 2.21).

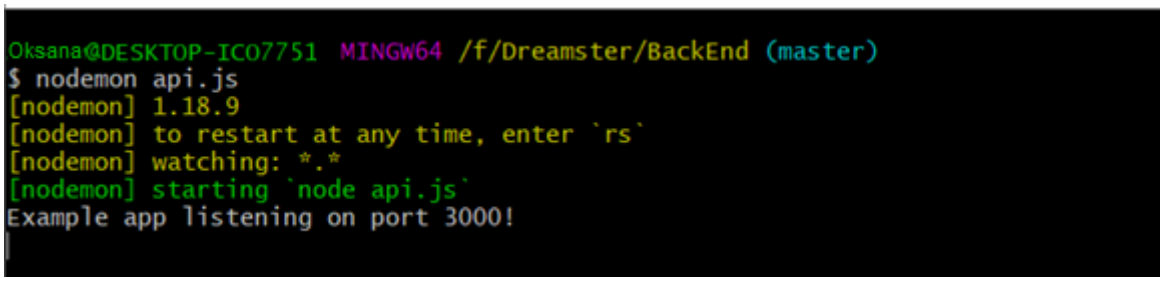

Рисунок 2.21 – Запуск сервера

2.6 Розробка алгоритму автентифікації користувача в соціальній мережі

Побудуємо логіку запитів автентифікації користувача в соціальній мережі. Для цього скористаємося API-системою Google Auth, алгорим роботи якої зображено на рис. 2.22.

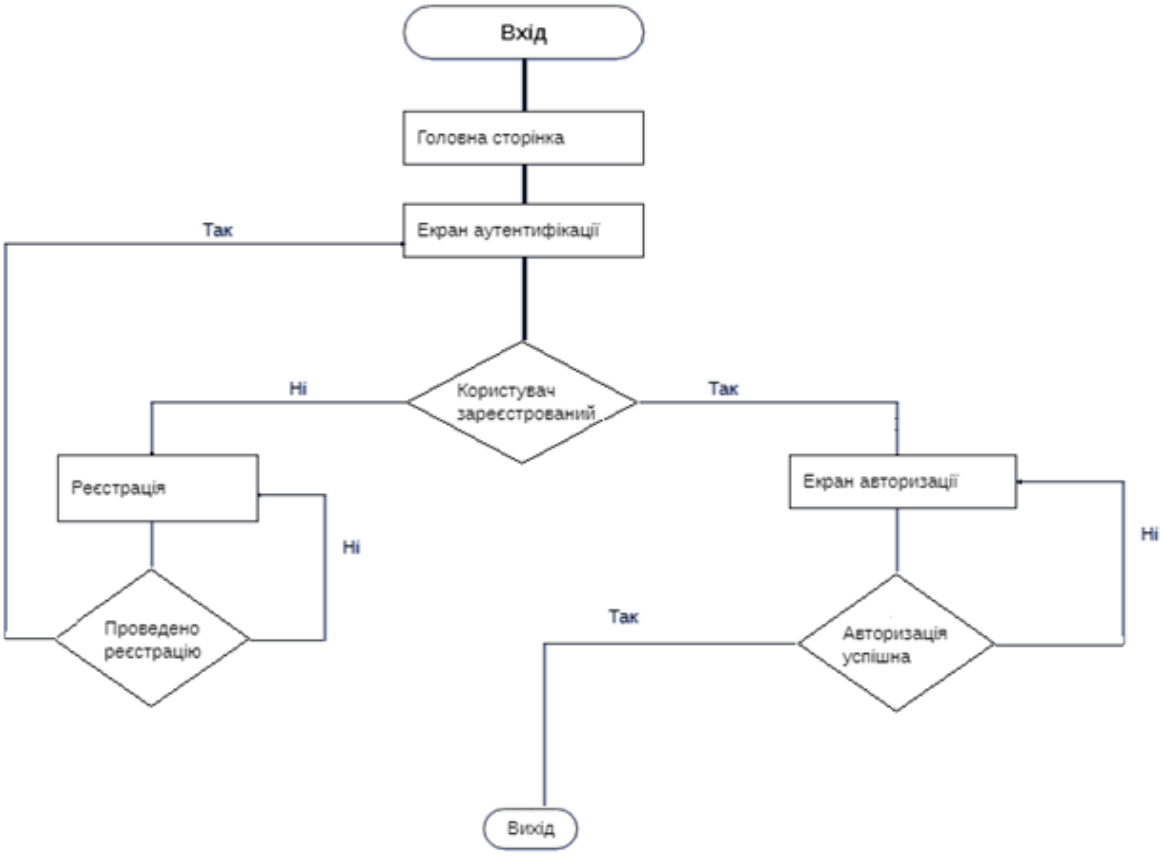

Рисунок 2.22 - Алгоритм автентифікації користувача в соціальній мережі

Пов’язаний пакет «google-auth-library» має асинхронну функцію, яка запускає логіку аутентифікації користувача Google, а саме запускає вікно з полями для заповнення та списком доступу, який вимагає сайт [14].

Коли користувач входить в систему, у нього є власний маркер, який підтверджує, що він пов’язаний з його обліковим записом Google. Завдяки цьому унікальному маркеру ми можемо ідентифікувати дані користувача в базі даних. Крім того, наявність цього токена дозволяє нам обмежити запити між різними користувачами. Наприклад, нам потрібно отримати онлайн-список даних нашого друга, але ми повинні мати доступ до такого запиту. Тому створимо функцію, яка перевірить, чи є клієнт «другом» іншого клієнта.

```
try{
if(req.params.friends_id == 'my' || req.param.friends_id == res.local.usersID){
 next()}
else{
```

```
session.run(`MATCH (u:User)-[r:FRIEND]-(f:User)WHERE ID(u) = ${res.local.usersID} AND ID(f) = ${req.param.friends_id}
RETURN f`)
.then(data => {session.close()
if(data.record.length < 0){let r = data.record.map(item => item.get(2)['property'])
next()}else{
res.send({status:'not friends', code:'411'})} })}}
catch(err){ res.send({status:`not friends`, code:'502'}) }
```

Ця функція перевірить, чи дійсно у нас є зв’язок в базі даних між користувачами типу «друзі», і якщо так, функція перейде далі в логіку запиту, якщо ні, вона поверне клієнту помилку «502» яка є прапорцем статусу " not friends".

Крім того, оскільки кожен із наших запитів API повинен мати відповідні дозволи для отримання результатів, нам потрібно розробити функцію, яка фактично буде контролювати ці доступи. Ця функція отримує токен, з яким користувач надсилає запит, і перевіряє його на правильність, функція jwt.verify перетворює хешовані дані на запис, який містить необхідну інформацію для підтвердження даних запиту. Слід відзначити, що при вході в систему користувач отримує файл cookie. Цей файл містить інформацію про стан мережі клієнта, але це особливий тип файлу cookie, який може редагувати та читати лише сервер, який його створив. Це означає, що якщо користувач захоче вручну внести дані до файлу, це буде помилкою [14].

Ми можемо переконатися в цьому, запустивши консоль у браузері, виконавши команду document.cookie, яка повинна повернути всі доступні йому файли cookie. Ми бачимо лише файли, створені Google для власних цілей, але нашого файлу-«токену» немає (рис. 2.23).

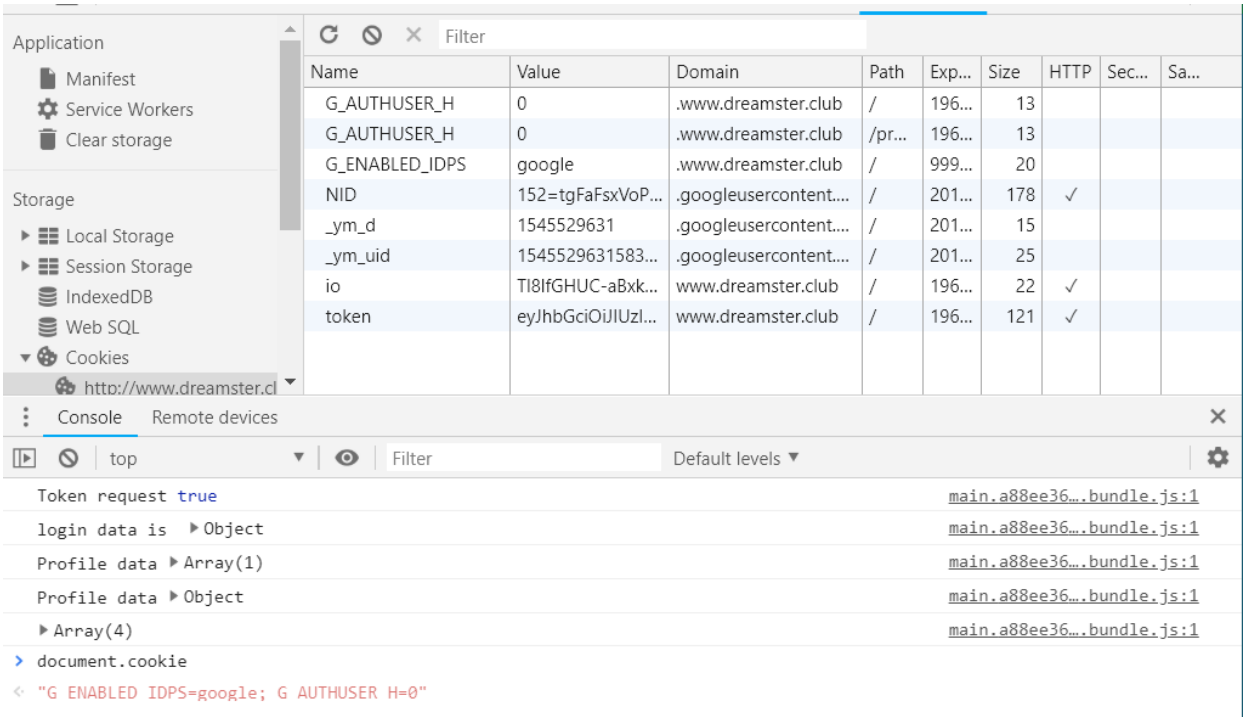

Рисунок 2.23 – Виконання команди document.cookie

2.7 Розробка front-end частини програмного забезпечення управління соціальною мережею

Спочаику налаштовуємо середовище розробки, підлкючаємо всі необхідні програмні пакети та встановлюємо компілятори для конвертування ts-файлів в js-файли.

Встановлюємо середовище роботи з typescript:

npm i - g typescript

Встановлюємо Angular-cli для зручної роботи з фреймворком Angular:

npm i - g angular - cli@latest

Після встановлення всіх пакетів і створення ієрархії файлів та папок у проекті маємо кінцеву структуру (рис. 2.24).

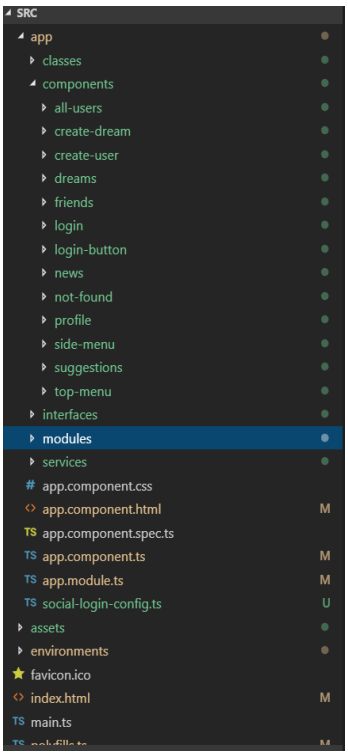

Рисунок 2.24 – Структура front-end частини програмного забезпечення

Кожна папка проекту має назву відповідно до вмісту її файлу. Таким чином, папка services містить всі залежності, необхідні для зовнішніх компонентів, а саме: налаштування роботи з мережевими сокетами, файли для налаштування роботи з http-запитами, файли для налаштування доступних URL-шляхів додатків і файл для підключення зовнішнього сервісу для роботи. з бібліотеками Google для авторизації.

Об’єднуємо всі бібліотеки, які нам потрібні для роботи із запитами Angular. Потім нам потрібно створити методи для різних типів запитів до сервера, наприклад: GET, POST, UPDATE, DELETE. Кожен із цих методів можна використовувати будь-де на веб-сайті, якщо сервіс підключений до відповідного компонента. Щоб послуга була доступна ззовні, її повинен експортувати оператор класу експорту. Більше того, завдяки архітектурі Dependency Injection ми можемо перехопити методи описаного сервісу в інших компонентах [12].

Розглянемо вміст папки «класи», вона містить розроблені файли класів програмного забезпечення, які перевернуть архітектуру нашої програми, додавши методи роботи з конкретними типами даних (рис. 2.25).

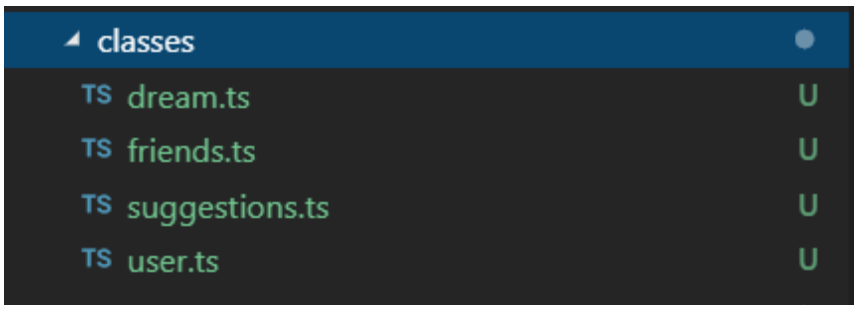

Рисунок 2.25 – Файли класів проекту

Файл "user.ts" описує всі методи, які вам знадобляться для роботи з даними користувача.

Створений клас має методи, що дозволяють отримати дані про профіль користувача, а також виконувати операції «входу» та «виходу» з мережі. Зауважимо, що цей клас включає в себе описаний раніше метод роботи із запитами від файлових служб /rest-service/rest.service. Його методи передаються в конструкторі класу до параметра _restService. Це дозволяє нам викликати всі методи, посилаючись на цей параметр. І будь-які зміни, які потрібно буде вносити в процесі модифікації запитів до сервера, потрібно буде вносити лише в одному місці – у службовому файлі, який відповідає за запити.

Вся система адресації контролюється на стороні клієнта програми. Тому нам потрібно вказати всі можливі адреси, на які може перейти користувач або яку сторінку відобразити, якщо адреса не існує. Ця конфігурація закодована у файлі модуля modules /routing-module.routing.ts.

Кожна адреса повинна запускати відповідний компонент, налаштування також визначають параметр data {depth}, він використовується для запуску анімації під час переміщення між сторінками. Якщо користувач знайшов неіснуючу адресу, йому слід перейти на сторінку, яка повідомить його і перенаправить на шлях/профіль. Параметр canActivate: [AuthGuard] вказує, що лише якщо користувач пройшов вказану перевірку в сервісі AuthGuard, він може перейти за посиланням, інакше він буде перенаправлений на сторінку авторизації, оскільки модуль AuthGuard перевіряє токен користувача. Для побудови елементів інтерфейсу програми необхідно розбитий макет

сторінки на невеликі частини-компоненти, кожна з яких повинна мати мінімум бізнес-логіки.

Кожен компонент просто повинен знати, де отримати дані і де їх розмістити в інтерфейсі користувача. Логіка обробки запитів до сервера та інших джерел даних повинна бути в розробленому класі сервісів. Таким чином, кожен компонент можна повторно використовувати в різних частинах веб-сайту з даними, що динамічно змінюються. Тому логіку подрібнення компонентів необхідно розбити на окремі модулі.

Компонент як окремий блок складається з файлу розмітки html, файлу стилю css і фактичного файлу класу, в якому логіка передачі даних між контролером і представленням. Також у деяких випадках, коли логіка компонента дещо складніша і вимагає проходження тестів, створюється тестовий файл. Описані класи компонентів повинні мати всі необхідні комбінації залежностей. Тому вкрай важливо забезпечити всю ін’єкцію залежностей в окремих компонентах, які цього потребують.

2.8 Компіляція front-end логіки програмного забезпечення управління соціальною мережею

Розроблені та протестовані компоненти готові до компіляції та мінімізації їх коду для подальшого розгортання на сервері.

Командний рядок angular-cli дозволяє автоматично створювати проект з найкращою мінімізацією коду та продуктивністю за допомогою команди ng build --aot --prod. Перемикач --aot дозволяє компілювати проект з паралельним процесом запуску файлів на сервері, --prod - замінює відповідні значення в змінних проекту іншим середовищем запуску програми. "ng build -prod -aot –output -path ./ Back End/ public" (рис. 2.26).

```
{
  "name": "dreamster",
  "version": "1.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve --aot",
    "build": "ng build --prod",
    "joinback": "ng build --prod --aot --output-path ../BackEnd/public",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
```

Рисунок 2.26 – Налаштування консолі розробника програмного забезпечення

В результаті виконання даної команди отримуємо відомості щодо статусу виконаної роботи, розміру утворених файлів, і т.д (рис. 2.27).

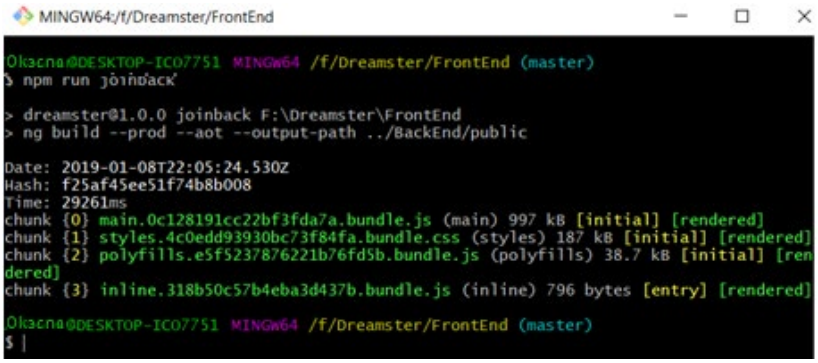

Рисунок 2.27 – Генерація серверної частини кінцевого проекту

Якщо операція пройшла успішно, в папку логіки сервера ми отримаємо нову, додаткову «загальнодоступну» папку з усієї колекції кодів інтерфейсу. Коли користувач переходить до каталогу прямих адрес домену, сервер повинен надати Yomi весь необхідний вміст із цієї папки.

2.9 Використання TensorFlow для управління соціальною мережею

Розвинена архітектура зв'язків даних у створеній графовій базі даних забезпечує з’єднання між користувачем та його рекомендаційним постом, відповідно до сформованого ним повідомлення (рис. 2.28).

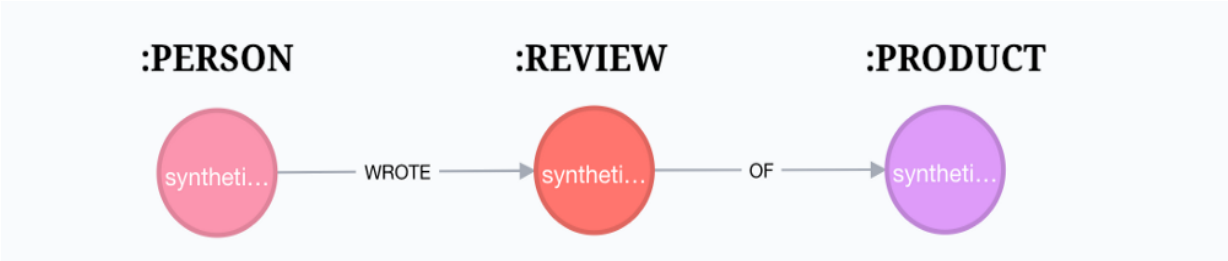

Рисунок 2.28 – Модель даних в графовій базі даних

Ми можемо використовувати розміщену базу даних або згенерувати дані у нашому власному прикладі Neo4j, використовуючи нашу кодову базу.

Набір даних синтетичний – ми згенерували його з імовірнісної моделі.

Якщо ми застосуємо неперевірений метод до невідомих даних, ми не зможемо навчити нейронну мережу, ми не зможемо визначити, чи є проблема в алгоритмі, чи в даних, чи в моделі. Синтезуючи дані, одне невідоме видаляється, і ми можемо зосередитися на пошуку успішної моделі [12].

Синтетичний набір даних має обмеження: у ньому відсутні нерівності та помилки, типові для даних реального світу. Для цієї навчальної вправи дуже корисний синтетичний набір даних, але будь-якій реальній системі знадобиться більше етапів очищення даних та експериментів, щоб знайти підходящу модель.

Оскільки кожна людина перевіряє 40 випадково вибраних продуктів, дуже ймовірно, що вони оцінять по одному продукту в кожному з шести стилів, тому наше прогнозування проблем перегляду дуже обмежене, і ми повинні мати можливість наблизитися до 100%.

Вхідними параметрами для нашої моделі є ідентифікатор особи та ідентифікатор продукту. Результатом моделі є оглядова оцінка.

Першим кроком нашої моделі є перетворення ідентичностей та ідентифікаторів продуктів у тензори впливу та стилю, які можна оцінити. На щастя, TensorFlow досить простий.

Налаштування та стилі зберігаються для всіх користувачів і їх записів у вигляді двох змінних: фігури [number-of-ids, width-of-tensor]:

```
product = tf.get-variable("product", [n-product, embedding-width])
person  = tf.get-variable("person",  [n-person,  embedding-width])
```

Використаємо tf.nn.embedding-lookup (продукт, product-id) для конвертації ідентифікатора в тензор-форми [width-of-tensor]

Модель для прогнозування має такий вигляд:

product = tf.get-variable("product",[n-product, embedding-width])

person = tf.get-variable("person",[n-person, embedding-width])

product-emb = tf.nn.embedding-lookup(product, product-id)

person-emb = tf.nn.embedding-lookup(person, person-id)

m = tf.multiply(product-emb, person-emb)

m = tf.reduce-sum(m, axis=-3)

review-score = tf.layers-dense(m, (1), tf.nn-sigmoid)

Для оцінювання втрат використаємо вбудовану середню квадратичну оцінку:

loss = mean-squared-error(pred-review-score, label-review-score)

Для навчання мережі використаємо вбудований оптимізатор Адама, який дає змогу мінімізувати втрати:

Train-op = tf.train-AdamOptimizer(params["lr"])-minimize(loss)

Використаємо Cypher запит для отримання даних з нашої графової бази даних:

MATCH p=(person:PERSON) - [:WROTE]->(review:REVEW

{dataset-name:"article-1", test:{test}})-[:OF]->(product:PRODUCT) RETURN person-id as person-id, product-id as product-id, review-core as review-core

2.10 Висновки до розділу 2

В даному розділі було обгрунтовано звибір засобів розробки керування соціальною мережею. Проаналізовано типи баз даних, які можна використовувати в розробці нового програмного забезпечення, в результаті чого обрано графову модель бази даних. Обгрунтовано застосування бібліотеки TensorFlow.

Здійснено розробку бази даних та створення записів до неї. Розроблено файлову структуру серверної та клієнтської частин інформаційної технології управління соціальною мережею.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ СОЦІАЛЬНОЮ МЕРЕЖЕЮ

3.1 Обґрунтування вибору мови програмування

Основні вимоги до мови програмування та середовища розробки такі:

- Мова має бути об'єктно-орієнтованою, оскільки програма є громіздкою, а реалізувати її процедурною мовою програмування було б важко та застаріло;
- Мова повинна підтримувати використання готових компонентів і мати сильну їх бібліотеку.

Для розробки програмних модулів такого рівня розглянемо три мови програмування: JavaScript, Java, Python.

У веб-розробці використовується багато спеціальних мов, як розмітки, так і веб-програмування, які часто поєднуються розробниками в одному проекті. Кожен з них має різний рівень складності, і одним з найбільш використовуваних є JavaScript [2].

Динамічна мова програмування, яка від самого початку носила назву Mocha, була написана групою програмістів на чолі з Брендоном Ейком у 1995 році. Мову було створено за 10 днів для компанії Netscape, яка потребувала скриптову мову для свого браузера з однойменною назвою.

З Mocha мову було перейменовано у LiveScript, а потім вже у JavaScript через високу популярність у той час мови Java (хоча ці мови мало що об’єднує). Статистика відразу кількох відомих ресурсів, таких як GitHub, StackOverflow та інших, стверджують, що JavaScript є мовою програмування, яка швидше за все набирає популярність, а результати опитувань розробників по всьому світові й взагалі ставлять JavaScript на перше місце серед найпопулярніших мов програмування.

JavaScript було вигадано як мову для браузерів, але сьогодні на ній можна створювати сервер, розробляти веб-додатки, мобільні додатки, ігри та багато іншого. Якщо раніше JavaScript входив у обов’язковий список вмінь для front end-розробки, то зараз він впевнено відчуває себе й на back end.

Кількість вакансій для JavaScript розробників збільшується не по рокам, а з кожним місяцем, що пов’язане з кількома факторами відразу: зростаючими можливостями мови, відносною простотою у вивченні, а також проникненням інтернету у всі сфери людського життя.

JavaScript - це об'єктно-орієнтована мова сценаріїв, яка використовується для реалізації інтерактивних елементів на сторінці. Використовуючи JavaScript, можна значно розширити функціональність веб-сайту, буквально «оживити» його, встановивши таким чином зворотний зв’язок з відвідувачем [2].

Варто зазначити, що JavaScript дуже компактний, але його явно не можна назвати легким для написання порівняно з тим же HTML-кодом. «Java» вивчити складніше, але її можливості набагато більше.

Різниця між цією мовою програмування та іншою популярною – PHP – полягає в тому, що вона працює на стороні клієнта, а не на стороні сервера. Точніше – за допомогою свого браузера. Це створює певні ризики безпеки, але в той же час знижує навантаження на сервер [3]. Переваги JavaScript:

- Жоден сучасний браузер не обходиться без підтримки JavaScript;
- Навіть фахівець не зможе використовувати плагіни і скрипти, написані на JavaScript;
- Корисні функціональні налаштування;
- Мова постійно вдосконалюється – зараз розробляється бета-версія проекту JavaScript;
- Взаємодія з додатком може відбуватися навіть із застосуванням текстових редакторів - Microsoft Office та Open Office [2].

Якщо ми хочемо працювати в Інтернеті, найкращий варіант - JavaScript. Якщо ми хочемо писати «десктопні» програми, Java - непоганий вибір.

Джеймс Гослінг почав розробляти проект мови програмування JAVA в липні 1991 року, для використання його в одному із своїх багаточисельних проектів set – top box. Мова спочатку називалась Oak “Дуб” на честь дуба, який ріс перед офісом Гослінга, але в кінці вибір був зупинений на JAVA, назва була вибрана із списку випадковим чином.

Гослінг вирішив запропонувати як додаток до віртуальної машини, яка буде мати стиль С С++. Sun випустила свій продукт під іменем JAVA 1.0. Девіз звучав (пишеш один раз запускаєш всюди). Слід відзначити і налаштовану безпеку, яка дозволяє використовувати межу на рівні файлів доступа.

Більшість веб–браузерів володіли можливістю запускати JAVA– аплети (програмні компонети в двоїчному коді які виконуються в вікні браузера) на веб–сторінках. Завдяки всьому цьому JAVA в дуже короткий час стала дуже популярною мовою.

В грудні 1998 року появилась JAVA 2. Нова версія пропонувала більшість конфігурацій, створених спеціально для різних типів платформ, наприклад JDEE додаток типу enterprise в той же час як stripped+down Sun ME був придуманий для мобільних платформ.

Sun переймінувала нові версії JD2 наступним чином: JAVA EE; JAVA ME; JAVA SE;

JAVA залишається стандартом, яким керується JAVA Community Process (процес який дозволяє заінтересованим лицям приймати участь у формуванні майбутніх специфікацій JAVA). Sun запропонувала більшість частин JAVA безкоштовно, незалежно на статус власника програмного забезпечення. Доходи від JAVA поступають за рахунок продажі спецалізованих продуктів, таких як Enterprise Java System.

Java - це мова програмування загального призначення, яка дотримується парадигми об’єктно-орієнтованого програмування та підходу одноразового запису й використання всюди. Java використовується в настільних, веб-, мобільних і корпоративних додатках. Java не настільки зручна для розробників, як Python, але вона досить проста для будь-якого розробника з базовим розумінням фреймворків, пакетів, класів та об’єктів. Він простий, стандартизований і передбачуваний, що дозволяє навчитися мислити в правильному напрямку [14].

Python проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell. З іншого боку, вона краще за С обробляє помилки, і, будучи мовою дуже

високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C потребує значних витрат часу.

Завдяки більш загальним типам даних, Python застосовують до більш широкого кола задач, ніж Awk і навіть Perl, у той ж час багато речей на мові Python робляться настільки ж просто.

Python дозволяє розбивати програми на модулі, що потім можуть бути використані в інших програмах. Python поставляється з великою бібліотекою стандартних модулів, які можна використовувати як основу для нових програм або як приклади при вивченні мови. Стандартні модулі надають засоби для роботи з файлами, системними викликами, мережними з'єднаннями і навіть інтерфейсами до різних графічних бібліотек.

Python – це інтерпретована мова, що дозволяє заощадити значну кількість часу, що зазвичай витрачається на компіляцію. Інтерпретатор можна використовувати інтерактивно, що дозволяє експериментувати з можливостями мови, писати шаблони програм або тестувати функції при розробці “знизу-вверх”. Він також зручний як настільний калькулятор. Python дозволяє писати дуже компактні й зручні для читання програми.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строго динамічною типізацією [8]. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів [14].

Його основними перевагами є: чистий синтаксис, портативність програми, стандартний дистрибутив має велику кількість корисних модулів, можливість використання Python в режимі діалогу [14].

Недоліком цієї мови є відносно повільна швидкість виконання Python через його інтерпретацію.

Ці та інші особливості Python роблять розгортання додатків надзвичайно швидким. Продуктивність створеного додатку залежить від його особливостей. Звичайно, для чисельного алгоритму, що виконує звичайну арифметику цілого числа в циклі 'for', неважливо, на якій мові він написаний. Але для “середнього” додатка,

збільшення продуктивності може бути просто дивовижним. Один недолік Python, у порівнянні з найбільш традиційними мовами, полягає в тому, що це - не цілком компільована мова; замість цього, вона частково транслює програму до внутрішньої форми байт-коду, і цей байт-код виконується інтерпретатором Python. Однак, у перспективі – сучасні комп'ютери мають так багато невикористовуваного обчислювального потенціалу, що для 90% додатків швидкодія зв'язана з вибором мови. Java теж компілюється в байт-код, але в даний час працює повільніше ніж Python у більшості випадків. Крім того, дуже просто об'єднати Python з модулями, написаними на C або C++, які можна використовувати, щоб збільшити швидкість роботи програм в критичних ділянках.

Після порівняння трьох мов програмування для реалізації цієї системи було обрано JavaScript.

Термін IDE (Integrated Development Environment) – «інтегроване середовище розробки» означає редактор, який розширив багато «приємностей», здатний працювати з допоміжними системами, такими як баг-трекер, контроль версій [15].

Як правило, IDE завантажує весь проект, тому може забезпечити функції автозаповнення для всього проекту, зручну навігацію по його файлам тощо.

JavaScript — потужна і примхлива мова. З одного боку, багато фреймворків і бібліотек, з іншого боку, не найпростіший синтаксис і небезпека «динаміки». Тому важливо вибрати редактора для роботи з ним. Правильний вибір забезпечить чистоту коду, високу швидкість розробки, мінімум помилок і задоволення від роботи [7].

Для роботи було обрано середовище Microsoft Visual Code версії 1.3.

Visual Studio Code (VS Code) – це спрощений, але потужний редактор вихідного коду, який запускається на комп'ютері й доступний для Windows, macOS і Linux. Вона підтримує JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов (наприклад, C++, C#, Java, Python, PHP і Go) і середовищ виконання (наприклад, .NET). VS Code надає додаткові можливості завдяки розширенням. Розширення VS Code можуть вносити додаткові функції до наявного набору. Після випуску цієї функції з’явилася можливість використовувати розширення VS Code для роботи із порталами Power Apps.

Особливості Visual Studio Code:

- VS Code дозволяє розробляти як консольні програми, так і програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-програми, веб-служби як у рідному, так і в керованому кодах для всіх платформ;

- у редакторі присутні вбудований відладчик, інструменти для роботи з Git та засоби рефакторингу, навігації за кодом, автодоповнення типових конструкцій та контекстної підказки;

- продукт підтримує розробку для платформ ASP.NET і Node.js, і вважається легковажним рішенням, яке дозволяє обійтися без повного інтегрованого середовища розробки.

Можливості Visual Studio Code:

- вбудовані інструменти інтеграції з GitHub, GIT, а також Visual Studio Team Services для швидкого тестування, складання, пакування та розгортання різних типів програм;

- зручність роботи з Unity-проектами;

- підтримка TypeScript та JavaScript;

- підтримка майже всіх мов програмування;

- написання коду для конкретної задачі з його подальшою інтеграцією до проекту (з надбудовою або безпосередньо);

- велика бібліотека шаблонів, готових фрагментів коду та сніпетів з можливістю додавання своїх елементів;

- одночасна робота з кількома проектами (у кількох вікнах).

Visual Studio включає редактор вихідного коду з підтримкою IntelliSense і простим рефакторингом коду [2]. Вбудований налагоджувач може запускати як налагоджувач вихідного коду, так і налагоджувач на рівні машини. Інші інструменти включають редактор форм для спрощення графічного інтерфейсу користувача програми, веб-редактор, конструктор класів і конструктор схем бази даних.Visual Studio дозволяє створювати та комбінувати сторонні програми (плагіни), щоб розширити функціональність практично на будь-який рівень, включаючи додавання

підтримки систем контролю вихідного коду (таких як Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, редагування та візуальний дизайн) код тематичними мовами програмування) або інструменти для інших аспектів процесу розробки програмного забезпечення (наприклад, Team Explorer для роботи з Team Foundation Server) [7].

Visual Studio — чудовий вибір, оскільки редактор виділяє синтаксис і виконує форматування коду, що, у свою чергу, робить код більш читабельним. Крім того, редактор MVS автоматично завершує деякі структури коду, слід почати вводити, наприклад, оператор вибору перемикача, редактор завершить його [7].

3.2 Тестування програмного забезпечення управління соціальною мережею

Щоб авторизувати користувача в мережі, скористайтеся формою входу в Google (рис. 3.1). Заповніть результати, які відображатимуться як маркер для наступних викликів сервера API.

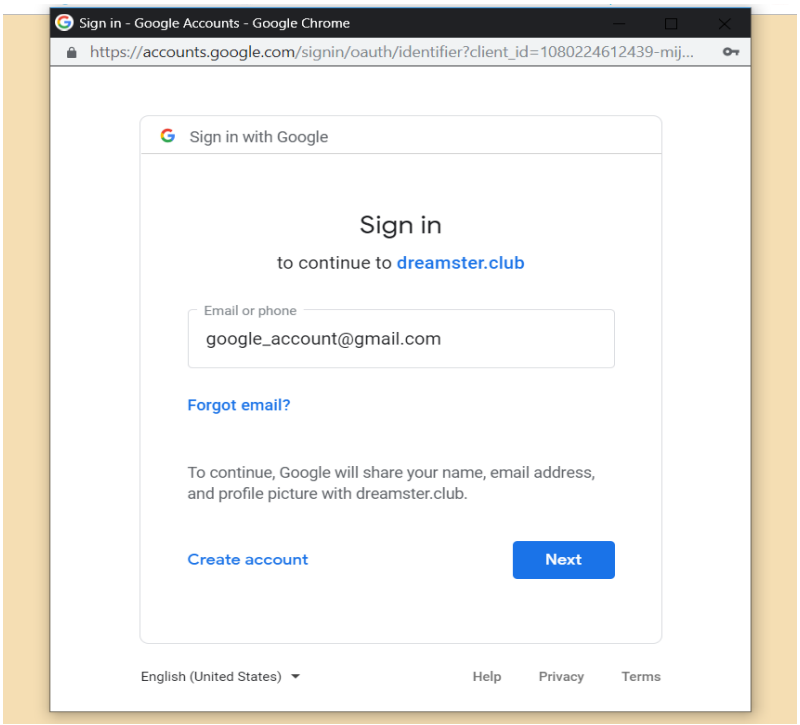

Рисунок 3.1 – Вікно авторизації корисувача соціальної мережі

Отже, ми можемо ідентифікувати користувачів в соціальній мережі та надавати їм певні права доступу (рис. 3.2)

```
async function verify(userTokenID) {
  const ticket = await client.verifyIdToken({
    idToken: userTokenID,
    audience: CLIENT_ID,
  });
  const payload = ticket.getPayload();
  return payload;
}
```

Рисунок 3.2 – Токен авторизації корисувача соціальної мережі

Створений токен має термін дії, і якщо він закінчився, сервер згенерує запит на поновлення токену в Google-системі.

Після авторизації користувача ми отримуємо його ідентифікатор в системі Google, його зображення профілю та адресу електронної пошти. Доступ для отримання цих параметрів вказується користувачем у формі авторизації.

Після успішної авторизації користувач переходить на сторінку свого профілю, де він може створити новий пост і переглянути наявні (рис. 3.3).

Щоб створити цей екран, потрібно створити такі запити API:

/ api / profile /: idUser - для отримання інформації з профілю користувача

/ api / lastDreams /: id - для отримання наявних записів користувача.

Якщо користувач перебуває за адресою профілю зі своїм ідентифікатором або порожньою адресою, яка ідентична адресі профілю, то потрібно створити нову форму повідомлення. Для цього згенеруємо компонент за умови, що власником є користувач:<create-dream *ngIf = "isOwner"> </create-dream>

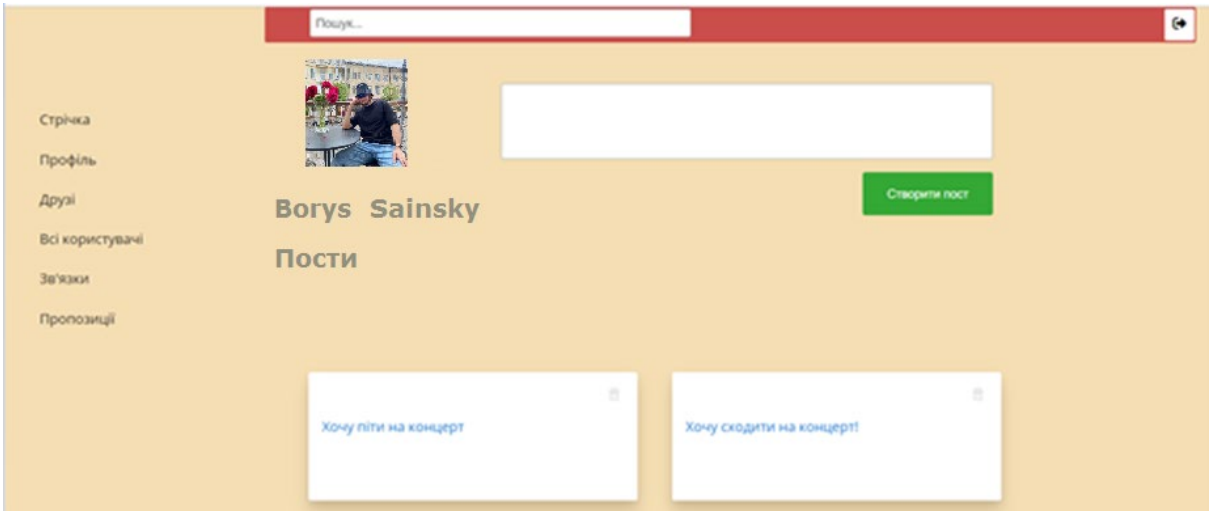

Рисунок 3.3 – Сторінка профілю користувача соціальної мережі

Кожний користувач має змогу переглядати всі свої записи в порядку їх дати формування (рис. 3.4).

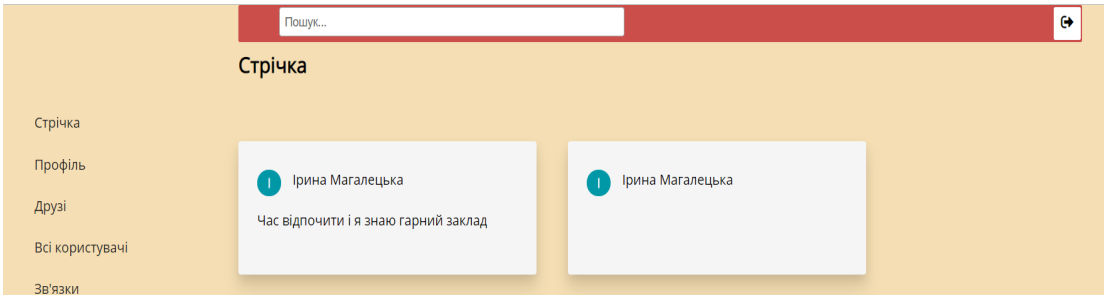

Рисунок 3.4 – Вкладинка «стрічка»

Для цієї сторінки програмного модуля потрібно створити запит /API/News API. Варто зазначити, що неавторизований користувач отримає помилку доступу до результату запиту.

Якщо запит буде успішним, ми отримаємо список друзів користувачів та їхні публікації. Записи сортуються в порядку, починаючи з першого пункту в списку - останнього створеного повідомлення. При натисканні на елемент публікації користувач потрапляє на сторінку профілю автора.

Користувач може перейти на сторінку зі списком його «друзів» у мережі. Сторінка містить три вкладки, кожна з яких вимагає запитів на окрему адресу API.

Перша вкладка містить користувачів, з якими ми маємо відношення :FRIENDS в обидві сторони (рис. 3.5).

Наступна адреса дає нам такі результати - / api / getFriends /: friend-id.

Список пропозицій дружби містить список користувачів, які надіслали нам запит на дружбу і чекають нашого підтвердження або відхилення.

Вкладка «Пропозиція дружби» містить список користувачів, яким ми надіслали запит «Дружба». Тому вони мають наш додаток на вкладці «Пропозиція дружби».

Результати запиту на кожній вкладці ідентичні, але мають різні пріоритети, тому ми отримуємо ім’я користувача та зображення профілю.

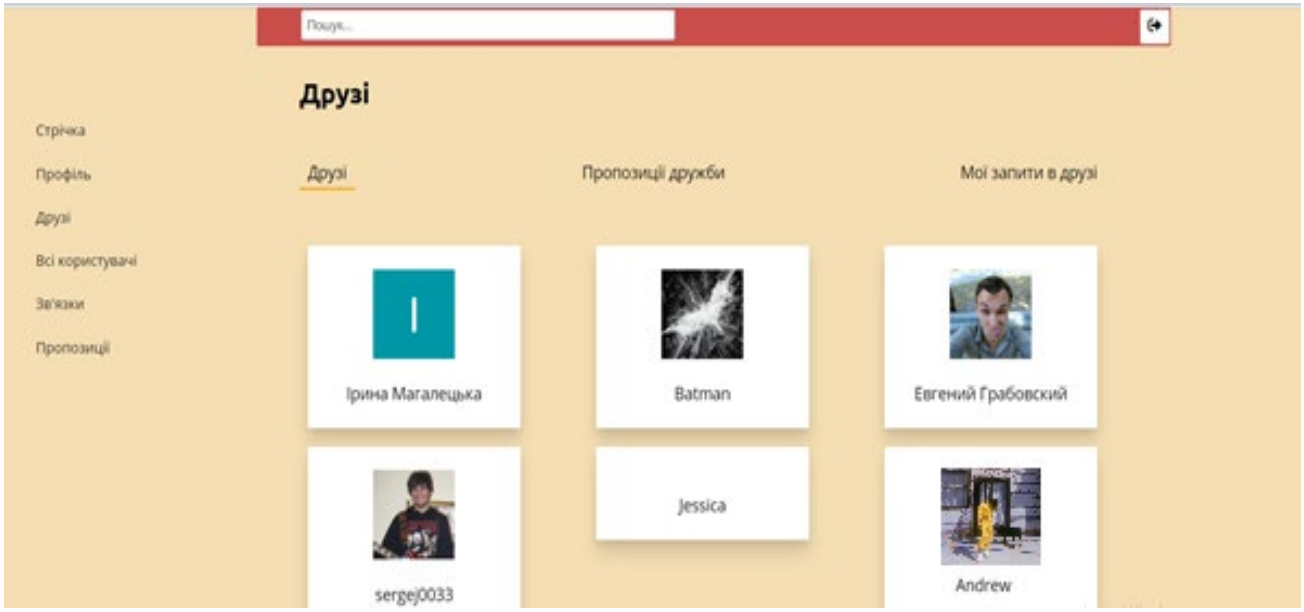

Рисунок 3.5 – Вкладинка «Друзі»

Користувач може отримати список усіх користувачів, які уповноважені в мережі додавати їх до «друзів» (рис. 3.6). Ця функція отримує поточний ідентифікатор користувача, невідомий ідентифікатор користувача та створює зв’язок даних між ними в графовій базі даних.

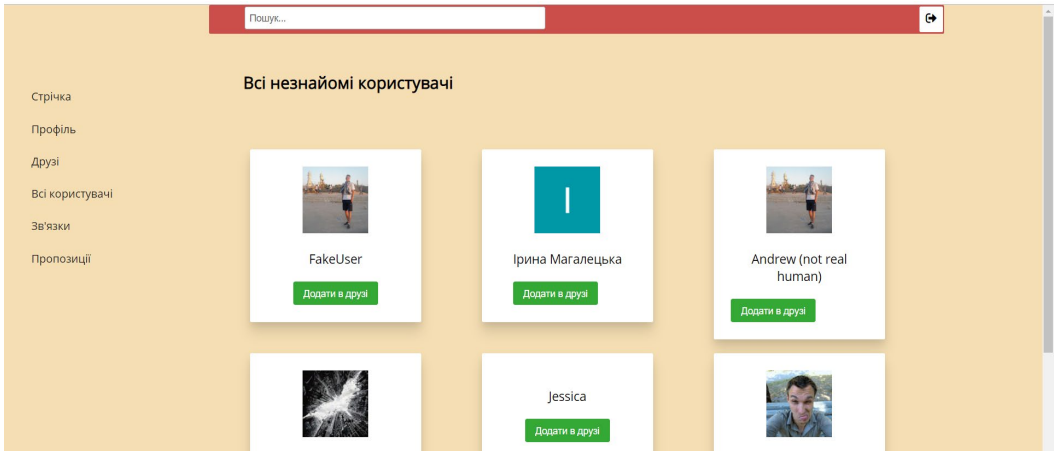

Рисунок 3.6 – Вкладинка «Всі користувачі» системи

Після створення запису користувач соціальної мережі перенаправляється на вкладинку публікації рекомендацій (рис. 3.7)

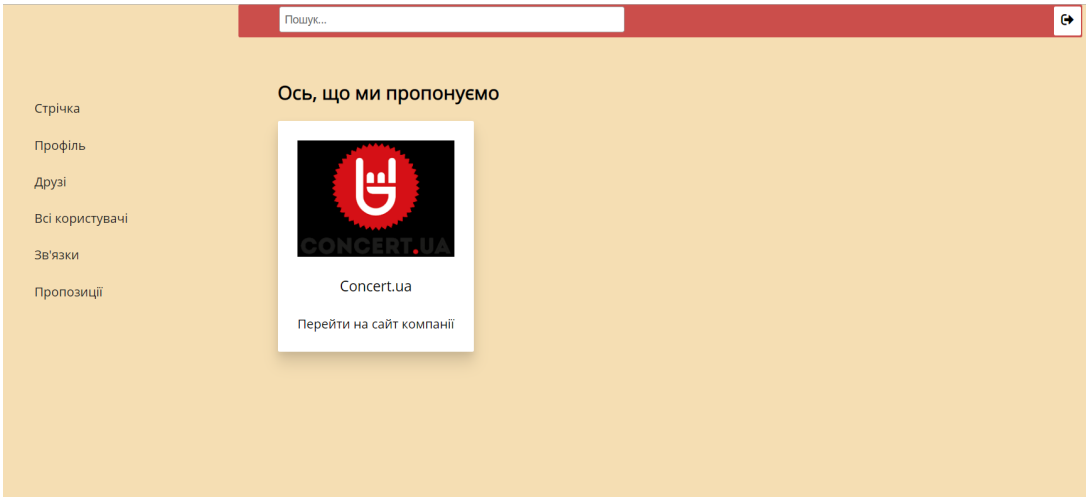

Рисунок 3.7 – Вкладинка публікації рекомендацій

Система розбиває речення на частини та створює процес узгодження нейронної мережі, який, у свою чергу, бере наявні дані від різних компаній і знаходить відповідності в публікаціях різних користувачів і в кінцевих результатах.

Якщо є співпадіння, ми отримаємо список компаній, які відповідають нашій публікації. В іншому випадку соціальна мережа повідомить, що рекомендацій не знайдено.

З метою тестування програмного модуля керування соціальною мережею булопроведено реєстрацію для 5 користувачів, при цьому кожен користувач створив

по 5 дописів. Відповідно до проведених досліджень порівняємо розроблене програмне забезпечення з раніше розглянутими системами аналогами за наступними критеріями: зручна реєстрація, зручна особиста сторінка, наявність окремого блогу, створення груп друзів по інтересам. Порівняння наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння програм для керування соціальною мережею

Назва програми	Зручна реєстрація	Зручна особиста сторінка	Наявність окремого блогу	Створення груп друзів по інтересам
Facebook	+	+	+	-
Pinterest	-	-	-	-
Розроблений веб-додаток	+	+	+	+

Таким чином в результаті тестування програмного модуля та відповідно до порівняльної таблиці програм для керування соціальною мережею, можна зробити висновок про те що було розширено функціонал соціальної мережі, що свідчить про те що мету дослідження було досягнуто.

3.3 Висновки до розділу 3

У третьому розділі було обґрунтовано вибір мови та середовища програмування Розроблене програмне забезпечення управління соціальною мережею було протестований, що підтвердило його правильну роботу та додавання нових функцій у вигляді отримання рекомендацій від користувачів з кола своїх друзів.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Інформаційна технологія управління соціальною мережею» відноситься до науково-технічних робіт, які плануються для використання не тільки самим розробником (замовником), а і іншими користувачами.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 4.1.

За результатами розрахунків, наведених в таблиці 4.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології управління соціальною мережею – середній.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу засобу поляриметричного аналізу оптично активних рідни

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	2	1	2
Ринкові переваги (наявність аналогів)	3	2	2
Ринкові переваги (ціна продукту)	4	3	3
Ринкові переваги (технічні властивості)	3	2	3
Ринкові переваги (експлуатаційні витрати)	2	1	1
Ринкові перспективи (розмір ринку)	3	2	3
Ринкові перспективи (конкуренція)	4	3	3
Практична здійсненність (наявність фахівців)	3	2	2
Практична здійсненність (наявність фінансів)	2	1	2
Практична здійсненність (необхідність нових матеріалів)	2	2	2
Практична здійсненність (термін реалізації)	2	3	3
Практична здійсненність (розробка документів)	2	1	1
Сума балів	32	23	27
Середньоарифметична сума балів, СБ	27		

4.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з

чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2.

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	12000	545	25	13625
Розробник	9250	420	25	10500
Консультанти	6700	305	10	3050
Всього:	27175			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2023 році $М_м$=6700 грн; $К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні; $t_{зм}$ – тривалість зміни, год.

Зроблені розрахунки зводимо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Програмування	20	1	38,1	1,0	762
Налагодження	100	3	44,9	1,18	4490
Тестування	5	2	41,5	1,09	207,5
Всього					5459,5

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (27175 + 5459{,}5) = 3263{,}5 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (27175 + 5459{,}5 + 3263{,}5) = 7897{,}6 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Розрахунок витрат на матеріали. Витрати на матеріали М, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$М = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де $Н_і$ – кількість матеріалів і-го виду, шт.; $Ц_і$ – ціна матеріалів і-го виду, грн.; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); n – кількість видів матеріалів.

Зроблені розрахунки зводимо до таблиці 4.4.

Таблиця 4.4 – Матеріали, що використані на розробку

Найменування матеріалів	Ціна за одиницю, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Папір А4	270	1	270
Фарба для друку	50	1	50
Всього, з врахуванням коефіцієнта транспортних витрат			352

Спецустаткування для наукових (експериментальних) робіт. Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами.

$$В_{спец} = \sum_1^к Ц_і \cdot С_{пр.і} \cdot К_і, \quad (4.5)$$

де $Ц_і$ – ціна придбання спецустаткування і-го виду, грн.; $С_{пр.і}$ – кількість одиниць спецустаткування відповідного виду, шт.; $К_і$ – коефіцієнт транспортних

витрат, $К_і$ = (1,1…1,15); n – кількість видів спецустаткування.

Зроблені розрахунки зводимо до таблиці 4.5.

Таблиця 4.5 – Витрати на придбання спецустаткування

Найменування спецустаткування	Ціна за одиницю, грн.	Витрачено	Вартість спецустаткування, грн.
Проектор для презентацій	15800	1	15800
Екран для проектора	5300	1	5300
Стіл офісний	4300	1	4300
Всього, з врахуванням коефіцієнта транспортних витрат			27940

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_1^к Ц_{іпрг} \cdot С_{прг.і} \cdot К_і, \quad (4.6)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Зроблені розрахунки зводимо до таблиці 4.6.

Таблиця 4.6 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Середовище розробки Intellij Idea	20000	1	20000
ПЗ для підготовки записки Microsoft Office	4000	1	4000
Всього, з врахуванням коефіцієнта інсталяції та налагодження			26400

Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (4.7)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки.

Зроблені розрахунки зводимо до таблиці 4.7.

Таблиця 4.7 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Ноутбук (Lenovo IdeaPad Gaming 316IAH7)	30000	3	2	2500
Принтер HP1020	2000	3	2	111
Всього	2611			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД} = \frac{0{,}4 \cdot 200 \cdot 7{,}5 \cdot 0{,}85}{0{,}98} + \frac{0{,}75 \cdot 20 \cdot 7{,}5 \cdot 0{,}85}{0{,}98}$$

$$= 618 \text{ грн.},$$

W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Зроблені розрахунки зводимо до таблиці 4.8.

Таблиця 4.8 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Ноутбук (Lenovo IdeaPad Gaming 316IAH7)	0.4	200
Принтер HP1020	0.75	20

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (27175 + 5459{,}5) \cdot \frac{63}{100} = 20560 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (27175 + 5459{,}5) \cdot \frac{137}{100} = 44709 \text{ грн.},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е +$$
$$+ I_в + В_{нзв} = 27175 + 5459{,}5 + 3263{,}5 + 7897{,}6 + 352 + 27940 + 2611 + 618$$
$$+ 20560 + 44709 = 140586 \text{ грн.}$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{140586}{0{,}5} = 281172 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії розробки дослідного зразка, то η=0,5.

4.3 Розрахунок економічної ефективності науково-технічної розробки від її впровадження безпосередньо розробником (замовником)

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_і)_і \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (4.8)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який

визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((15000 - 9500) \cdot 1000 - (1000 - 100) \cdot 9500) \cdot 0{,}8333 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 668316 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{668316}{(1+0{,}1)^1} = 607560 \text{ грн.}, \quad (4.9)$$

де $\Delta П$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо T=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний

інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot \text{ЗВ} = 2 \cdot 281172 = 562344 \text{ грн.}$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{\text{інв}}$=2...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{\text{абс}} = \text{ПП} - PV = 607560 - 562344 = 45216 \text{ грн.,}$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $E_{\text{абс}}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар’єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} = \sqrt[1]{1 + \frac{45216}{607560}} = 1{,}03, \quad (4.10)$$

де $Т_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Далі розраховуємо період окупності інвестицій $Т_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$Т_о = \frac{1}{E_в} = \frac{1}{1{,}03} = 0{,}97 \text{ року.}$$

Оскільки $Т_о$=0,97<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.4 Висновок до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія управління соціальною мережею» становить 42,0 бала, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,76 рази.

Також термін окупності становить 0,97 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати розробника до впровадження даної розробки при отриманні ефекту в розмірі 45216 грн.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія управління соціальною мережею».

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було розроблено інформаційну технологію управління соціальною мережею.

В роботі проаналізовано процес передачі даних у мережі, проаналізовано технлогії управління соціальними мережами, визначено актуальність та доцільність розробки інформаційної технології управління соціальною мережею із застосуванням технології нейронних мереж. Під час аналізу об’єкта проектування було сформульовано задачі розробки інформаційної технології управління соціальною мережею з використанням технології нейронних мереж, а саме визначено поняття соціальної мережі, визначено основні завдання, напрямки розвитку та необхідний функціонал програми. Проаналізовано програми аналоги. Навіть найвідоміші програми в цій галузі мають деякі недоліки. Тому доцільно створити власну вдосконалену програму управління соціальною мережею.

Обгрунтовано вибір засобів розробки інформаційної технології управління соціальною мережею. Проаналізовано типи баз даних, які можна використовувати в розробці нового програмного забезпечення, в результаті чого обрано графову модель бази даних. Обгрунтовано застосування бібліотеки TensorFlow. Здійснено розробку бази даних та створення записів до неї. Розроблено файлову структуру серверної та клієнтської частин інформаційної технології управління соціальною мережею.

Обґрунтовано вибір мови та середовища програмування. Розроблене програмне забезпечення управління соціальною мережею було протестовано, що підтвердило його правильну роботу та додавання 2 нових функцій у вигляді отримання рекомендацій від користувачів з кола своїх друзів.

Також було виконано розрахунок загальних витрат на завершення науково-технічної роботи та оформлення її результатів, які складають 281172 гривень, період окупності витрат складає 1,93 року.

Отже всі поставлені задачі виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Саїнський Б.Д. "Особливості створення соціальних мереж" в Матеріали конференції «LI Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2023)», Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15858/13358
2. Від Netflix до «Такфлікс»: великий гід стримінговими сервісами. The Village Україна. URL: https://www.the-village.com.ua/village/culture/tv/293007-tv-netflix-amazon-disney-plus-apple-tv-plus-hbo-max-peacock-streaming.
3. Які соціальні мережі популярні в світі [Електронний ресурс] – Режим доступу: https://marketer.ua/ua/top-social-media-2018/
4. Вплив соціальних мереж на суспільство [Електронний ресурс]. Режим доступу: http://esnuir.eenu.edu.ua/bitstream/123456789/10267/1/Yurieva.pdf
5. Створення соціальної мережі [Електронний ресурс]. Режим доступу: http://webstudio2u.net/ua/webdesign/569-sozdanie-sotcialnoj-seti.html
6. Соціальні мережі: поняття, історія виникнення [Електронний ресурс]. Режим доступу: https://zounb.zp.ua/resourse/zaporizkyy-kray/zaporizhzhya-bibliotechne/fahova-osvita/socialni-merezhi-piv
7. Як створити соціальну мережу [Електронний ресурс]. Режим доступу: http://megasite.in.ua/80343-yak-stvoriti-socialnu-merezhu.html
8. Деркаченко Я. А. Соціальні мережі, як середовище для технологій маніпулятивного впливу [Електронний ресурс]. Режим доступу: https://journals.dut.edu.ua/index.php/dataprotect/article/view/531/493
9. Community builder [Electronic resource] – Mode of access: https://extensions.joomla.org/extension/community-builder/
10. Бази даних і системи підтримки рішень [Електронний ресурс] – Режим доступу: http://dss-bi.com.ua/System/бази-даних-і-системи-підтримки-прийня/

11. The most popular social media platforms [Electronic resourse] – Mode of access: https://www.digitalinformationworld.com/2019/01/most-popular-global-social-networks-apps-infographic.html
12. Інтернет технології в світі: Перспективи використання соціальних мереж [Електронний ресурс]. Режим доступу: http://internet-texnologyeducation.blogspot.com/2016/05/blog-post_5.html
13. Марутян Р. Соціальні мережі, як виклик національній безпеці, 2012. [Електронний ресурс]. Режим доступу: http://www.dsaua.org/index.php/option=com_content&view=article&id=154:2012-03-15-21-44-19&catid=66:2010-13-08-48-53&ltemid=90&lang=en
14. Facebook визнали втрачу даних [Електронний ресурс]. Режим доступу: https://news.rambler.ru/internet/39534694-v-facebook-priznali-peredachu-dannyh-polzovateley/
15. NoSQL Databases Explained [Electronic resourse] – Mode of access: https://www.mongodb.com/nosql-explained
16. TensorFlow.js [Electronic resourse] – Mode of access: https://www.freecodecamp.org/news/get-to-know-tensorflow-js-in-7-minutes-afcd0dfd3d2f/
17. Manyika James, Chui Michael, Brad Brown, Bughin Jacques, Dobbs Richard, Roxburgh Charles, Hung Byers Angela. Report of McKinsey Global Institute, Neo4J: The next frontier for innovation, competition, and productivity. [Electronic resourse] – Mode of access: https://neo4j.com/developer/get-started/
18. PHP: Hypertext preprocessor [Електронний ресурс]. Режим доступу: http://php.net/
19. Довідник з PHP [Електронний ресурс]. Режим досутупу: https://uk.wikipedia.org/wiki/PHP
20. Кавецький В.В. Економічне обґрунтування інноваційних рішень: практикум / В.В. Кавецький, В.О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113с.

ДОДАТКИ

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія управління соціальною мережею

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 84,8% · Схожість 15,2%

Аналіз звіту подібності (відмітити потрібне):

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

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Саїнський Б.Д.

Керівник роботи _____ Озеранський В.С.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

Серверна частина

```
      const itemController = require('../controllers/item.controller');
      const multer  = require("multer");
      const authCheck = require('../middlewares/session.checker');
      const storage = multer.diskStorage({
import {Injectable} from '@angular/core';
import {RestService} from '../services/rest-service/rest.service';

import {environment} from '../../environments/environment';
import { Observable } from 'rxjs';

import {User} from '../interfaces/interfaces';
import {Dream} from '../interfaces/interfaces';

@Injectable()
export class DreamClass {

    private readonly domain:string = environment.rest_domain;
    constructor(private _restService:RestService){}
    createDream(dream:Dream):Observable<any>{
        console.log(dream)
        return this._restService.post([this.domain+'/api/Dream', dream])
    }
    // get News for user
    getNews(){
        return this._restService.get(this.domain+'/api/News')
    }
    //ID of Dream
    getDream(id){
        return this._restService.get(this.domain+'/api/Dream/'+id)
    }
    getDreams(user_id = 'me'){
        return this._restService.get(this.domain+'/api/Dreams/'+user_id)
    }
    removeDream(id){
        console.log('we are here', id)
        return this._restService.delete(this.domain+'/api/Dream/'+id)
    }
}
      import {Injectable} from '@angular/core';
      import {RestService} from '../services/rest-service/rest.service';
      import {environment} from '../../environments/environment';
      import { Observable } from 'rxjs';
      import {User} from '../interfaces/interfaces';
```

```
@Injectable()
export class Friends {
private readonly domain:string = environment.rest_domain;

    constructor(private _restService:RestService){}
    addFriend(id:Number):Observable<any>{
        return this._restService.post([this.domain+'/api/addFriend', {id}])
    }
    getFriends(id = 'me'):Observable<any>{
        return this._restService.get(this.domain+'/api/getFriends/'+id)
    }
    getFollowers(id = 'me'):Observable<any>{
        return this._restService.get(this.domain+'/api/getFollowers/'+id)
    }
    getFollowings(id = 'me'):Observable<any>{
        return this._restService.get(this.domain+'/api/getFollowings/'+id)
    }
    removeFriend(id = 'me'):Observable<any>{
        return this._restService.delete(this.domain+'/api/Friend/'+id)
    }
}
import {Injectable} from '@angular/core';
import {RestService} from '../services/rest-service/rest.service';

import {environment} from '../../environments/environment';
import { Observable } from 'rxjs';
@Injectable()
export class Suggestions {

    private readonly domain:string = environment.rest_domain;
          constructor(private _restService:RestService){}
    getSuggestions(id:number):Observable<any>{
        console.log('dream_id', id)
        return this._restService.get(this.domain+'/api/suggestion/'+id)
    }
}
import {Injectable} from '@angular/core';
import {RestService} from '../services/rest-service/rest.service';
import {environment} from '../../environments/environment';
import { Observable } from 'rxjs';

import {User} from '../interfaces/interfaces';
@Injectable()
```

```
export class UserClass{
    private readonly domain:string = environment.rest_domain;

    constructor(private _restService:RestService){}

    signUp(googleToken):Observable<any>{
        console.log('googleToken', googleToken)
        return this._restService.post([this.domain+'/api/login',
{googleToken}])
    }
    returnProfile(user_id = 'me'):Observable<any>{
        return this._restService.get(this.domain+'/api/profile/'+user_id)
    }
        getAllUsers(){
        return this._restService.get(this.domain+'/api/Users')
    }
}
```

Серверні налаштування

```
// Start
const http = require('http');
const express = require('express');
const socketIO = require('socket.io');
// Secure
const helmet = require('helmet');

//Google AUTH
const {OAuth2Client} = require('google-auth-library');
const CLIENT_ID = "1080224612439-
mijm05g3tdgt7hq4rhah0e5tgv7s9a8e.apps.googleusercontent.com";
const client = new OAuth2Client(CLIENT_ID);
// AUTH
const cookieParser = require('cookie-parser');
const jwt = require('jsonwebtoken');
const token_secret = 'token-secret'; // TOKEN SECRET TO SIGN AND VERFIY
// PARSING
const bodyParser = require('body-parser');
const jsonParser = bodyParser.json()

//Hash
```

```
const crypto = require('crypto');

// DB
const neo4j = require('neo4j-driver').v1;
//Parsing dream
'use strict';
var DreamObject = /** @class */ (function () {
    function DreamObject(dreamText) {
        var _this = this;
        //Object with words and actions to make with date
        this.DateObject = [            {
                id: 0,
                word: 'завтра',
                addIng: '+1 day'            },            {
                id: 1,
                word: 'тижн',
                addIng: '+1 week'            },            {
                id: 2,
                word: 'місяц',
                addIng: '+1 month'            },            {
                id: 3,
                word: 'року',
                addIng: '+1 year'            },            {
                id: 4,
                word: 'рік',
                addIng: '+1 year'            }        ];
        // List of endsWith for Action detect
        this.endsWith = ['ти', 'тися', 'тись', 'ть'];
        //Words which help find date
        this.addictedDate = ['наступно', 'після', 'через', 'протягом'];
        this.PreymenukList = {
            All: ['в', 'у', 'на', 'до', 'від', 'під', 'над', 'за', 'крізь',
'через', 'при', 'перед', 'поза', 'з-за', 'біля', 'обабіч', 'коло', 'по', 'з', 'із',
'разом'],
            Places: ['в', 'у', 'на', 'до', 'від', 'під', 'над', 'за', 'крізь',
'через', 'при', 'перед', 'поза', 'з-за', 'біля', 'обабіч', 'коло', 'по'],
            Members: ['з', 'із', 'разом']
        };
        this.DreamStructure = { value: '', subStrings: [], words: [], Objects:
[], Places: [], Date: [], Time: [], Members: [], Preymunuku: [], Actions: [], Adds:
[], Character: [] };
        this.DreamStructure.value = dreamText;
        var removeArr = [];
```

```
            var arr = dreamText.split(' '); // convert to array
            this.DreamStructure.subStrings = dreamText.split('.'); //Creating
substring PROTOTYPE
            arr.map(function (x, index) { return _this.DreamStructure.words.push({
id: index, value: x }); }); //Creating new DreamObject
            this.DreamStructure.words.map(function (x, index) {
                if (x.value == "Я" || x.value == "хочу") {
                    removeArr.push(index);
                }        });    }
        //Return return word VALUES by ID
        DreamObject.prototype.getWordById = function (index) {
            if (this.DreamStructure.words[index])
                return this.DreamStructure.words[index]['value'];
            else
                return false;
        };
        // Get previous word if Exist else false
        DreamObject.prototype.getPrevWord = function (index) {
            if (this.DreamStructure.words[index - 1]) {
                return this.DreamStructure.words[index - 1]['value'];        }
            else
                return false;    };
        //Get next word
        DreamObject.prototype.getNextWord = function (index) {
            if (this.DreamStructure.words[index + 1]) {
                return this.DreamStructure.words[index + 1]['value'];
            }
            else
                return false;    };
        //Getiing members
        DreamObject.prototype.getMembers = function () {
            var _this = this;
            var removeArr = [];
            this.DreamStructure.words.map(function (x, index) {
                _this.PreymenukList['Members'].map(function (prumennuk) {
                    if (x.value == prumennuk) {
                        _this.DreamStructure.Members.push({   id:   index,   value:
_this.getNextWord(index) });
                        removeArr.push(index + 1); // HARD CODE ???
                    }            });        });
            return this.DreamStructure.Members;    };
        //Getting places
        DreamObject.prototype.getPlaces = function () {
```

```
            var _this = this;
            var removeArr = [];
            this.DreamStructure.words.map(function (x, index) {
                _this.PreymenukList['Places'].map(function (prumennuk) {
                    if (x.value == prumennuk) {
                        this.DreamStructure.Places.push(_this.getNextWord(index));
                        removeArr.push(index + 1); // HARD CODE ???
                    }            });        });
            return this.DreamStructure.Places.join(' ');
        };
        //Get Pruymennuks
        DreamObject.prototype.getPreymunuk = function () {
            var _this = this;
            var removeArr = [];
            this.DreamStructure.words.map(function (x, index) {
                _this.PreymenukList['All'].map(function (prumennuk) {
                    if (x.value == prumennuk) {
                        _this.DreamStructure.Preymunuku.push({  id:  index,  value:
x.value });
                        removeArr.push(index);
                        //getObjects(index); //Trying to get Object of Dream
                    }            });        });
            return this.DreamStructure.Preymunuku;
        };
        //Get Time
        DreamObject.prototype.getTime = function () {
            var _this = this;
            var removeArr = [];
            this.DreamStructure.words.map(function (x, index) {
                if (x.value == "o") {
                    _this.DreamStructure.Time.push({   id:   index   +   1,   value:
_this.getNextWord(index) });
                    //HERE WE DONT ADD letter 'o' to any specific data
                    removeArr.push(index + 1);
                }        });
            return this.DreamStructure.Time;
        };
        //Getting Upper date (words of space-date)
        // Get word definition by index word
        DreamObject.prototype.getSpaceDate = function () {
            var _this = this;
            var removeArr = [];
            this.DreamStructure.words.map(function (x, index) {
```

```
                _this.addictedDate.map(function (item) {
                    if (x.value.startsWith(item)) {
                        _this.DreamStructure.Date.push({ id: index, value: x.value
});
                        _this.DreamStructure.Date.push({ id: index + 1, value:
_this.getNextWord(index) });
                        removeArr.push(index);
                        removeArr.push(index + 1);
                    }            });         });
            return this.DreamStructure.Date;    };
        //Put actions from Dream to Object
        DreamObject.prototype.getActions = function () {
            var _this = this;
            var removeArr = [];
            this.DreamStructure.words.map(function (x, index) {
                _this.endsWith.map(function (end) {
                    if (x.value.endsWith(end)) {
                        _this.DreamStructure.Actions.push(x.value);
                        removeArr.push(index);
                    }
                });         });
            console.log('returen', this.DreamStructure.Actions);
            return this.DreamStructure.Actions.join(' ');
        };
        return DreamObject; }());
    //let Dream = 'Я хочу піти подивитися наступної середи в кіно разом з Петром.';
    // let Dream = 'Хочу піти до парку погуляти о 12:00';
    // console.log('getActions', dd.getActions());
    // console.log('getMembers', dd.getMembers());
    // console.log('getTime', dd.getTime());
    // console.log('getPlaces', dd.getPlaces());
    // console.log('getSpaceDate', dd.getSpaceDate());
    // Connect
    const url = 'bolt://hobby-elikijibbjimgbkedkeclcbl.dbs.graphenedb.com:24786';
    const     driver     =     neo4j.driver(url,     neo4j.auth.basic('user-admin',
'b.07cUDad7z50R.FgfXA35jp8FHwzNU'));
    const session = driver.session();
    // const url = 'bolt://hobby-oicnfgccoamjgbkejncodebl.dbs.graphenedb.com:24786';
    //    const    driver    =    neo4j.driver(url,    neo4j.auth.basic('user-admin',
'b.kalRNFzN1j60.WpqhlGvHNhu7MKs1'));
    const app = express();
    // Testing connection with sockets. Providing app to socket
    const server = http.createServer(app);
```

```
const io = socketIO(server);
app.use(helmet());

app.use(express.static(__dirname + '/public'))
// CORS
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "http://localhost:4200");
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, DELETE');
    res.header("Access-Control-Allow-Headers",    "Origin,    X-Requested-With,
Content-Type, Accept");
    res.header("Access-Control-Allow-Credentials", true);
    next();
});
// app.use(cors(corsOptions))
app.use(cookieParser());
io.on('connection', (socket) => {
    // Log whenever a user connects
    console.log('user connected');
    // Log whenever a client disconnects from our websocket server
    socket.on('disconnect', function(){
        console.log('user disconnected');
    });

    // When we receive a 'message' event from our client, print out
    // the contents of that message and then echo it back to our client
    // using `io.emit()`
    socket.on('message', (message) => {
        console.log("Message Received: " + message);
        io.emit('update', {type:'update_wall', dream_id: message});
    }); });
/Custom middleware for cookie detect
function ensureToken(req, res, next){
    console.log('req.cookies ', req.cookies.token)
    try{
        const payload = jwt.verify(req.cookies.token, token_secret)
        if(payload){
            console.log('verified');
            console.log('padload', payload.id)
            res.locals.userID = payload.id // User ID() in BD
            next()
        }
        else {
            console.log('Not valid token');
```

```
                res.send({response:{data:'not verified'}})
            }     }
        catch(err){
            console.log('Credentials error', err);
            res.sendStatus(401);
        } }
    async function verify(userTokenID) {
        const ticket = await client.verifyIdToken({
            idToken: userTokenID,
            audience: CLIENT_ID,  // Specify the CLIENT_ID of the app that accesses
the backend
            // Or, if multiple clients access the backend:
            //[CLIENT_ID_1, CLIENT_ID_2, CLIENT_ID_3]
        });
        const payload = ticket.getPayload();
        return payload;
        // If request specified a G Suite domain:
        //const domain = payload['hd'];
    }
    function isFriend(req, res, next){
        try{
            console.log('Friend middleware', res.locals.id);
            // User himself
            if(req.params.friend_id    ==    'me'    ||    req.params.friend_id    ==
res.locals.userID){
                console.log('User himself')
                next()
            }
            else{
                session.run(`MATCH (u:User)-[r:FRIENDS]-(f:User)
                             WHERE ID(u) = ${res.locals.userID} AND
                             ID(f) = ${req.params.friend_id}
                             RETURN f`)
                        .then(data => {
                            session.close()
                            if(data.records.length > 0){
                                console.log('lentgs      of      response      ',
data.records.length)
                                let      r      =      data.records.map(item      =>
item.get(0)['properties'])
                                next()
                            }
                            else{
```

```
                                console.log('not friends')
                                res.send({status:'not friends', code:'413'})
                            }               })          }     }
        catch(err){
            console.log(err)
            res.send({status:`error in 'try' block`, code:'500'})
        } }
    //return if token exist
    app.get('/api/checkToken', ensureToken, function(req, res){
        let response = {token:true}
        res.send(response)
    })
    app.post('/api/login', jsonParser, function(req, res){
        const user = req.body
        console.log('Incoming user', user)
        if(user.googleToken !== undefined){
            console.log('user.googleToken', user.googleToken)
            verify(user.googleToken).then( payload => { // All google info
                if(payload){
                    console.log('payload', payload);
                    const googleUserID = payload['sub'];
                    const googleUserName = payload['name'];
                    const googleUserEmail = payload['email'];
                    const                    googleUserPicture                    =
payload['picture'].replace('https://','');
                    const session = driver.session();
                    // CREATE OR GET EXIST
                    // IMAGE ISN OT VISIBLE FROM BROWSE DB, BUT IT EXIST IN TABLE
                    session.run(`MERGE      (u:User      {name:'${googleUserName}',
email:'${googleUserEmail}',                          picture:'${googleUserPicture}',
idToken:'${googleUserID}'})
                                RETURN u`)
                            .then(data => {
                                session.close()
                                console.log('return data', data)
                                console.log('0                                    record',
data.records[0].get(0)['identity']['low'])
                                let userID = data.records[0].get(0)['identity']['low']
// user ID() from db as token in cookies
                                // console.log('user.id', user.idToken)
                                const user = {id:userID} // for origin data structure;
                                const token = jwt.sign(user, token_secret); //sign for
cookies
```

```
                            console.log('Token = ', token);
                            // Signed JWT token
                            res.cookie('token', token, {httpOnly:true})
                               .status(200)
                               .send({response:'Вхід успішний'})
                        })
                        .catch( err => {
                            console.log(err)
                            res.send({reponse:err})
                        })            }
                else {
                    console.log('Verify on creating google token error')
                    res.statusCode(401)
                }
            }).catch( (err) => {
                console.log(err)
                res.send({response:{data:err}})
            })          }
        else {
            res.send({response:{data:'User not defined'}})
        } })
    app.get('/api/logout', function(req, res){
        res.clearCookie('token', {httpOnly:true});
        res.send({response:'removed'}) })
    pp.get('/api/profile/:idUser', ensureToken, function(req, res){
        let userID; // Token of user
        if(req.params.idUser == 'me'){
            userID = res.locals.userID
        } else {
            userID = req.params.idUser
        }
        console.log('USREDID', userID)
        const session = driver.session();
        session.run(`MATCH (u:User) WHERE ID(u) = ${userID} RETURN u`).then(data =>
{
            session.close()
            let r = data.records.map(item => item.get(0)['properties'])
            res.json(r);
        })
        .catch( err => {
            res.send(err)
        }) })
    // app.set('trust proxy', 1) // trust first proxy
```

```
      app.get('/api/Users', ensureToken, function (req, res) {
          let me = res.locals.userID
          console.log('Me', me)
          const session = driver.session(); // All users, but not me
          session.run(`MATCH (u:User)
                      WHERE NOT ID(u) = ${me}
                      RETURN u`).then(data => {
              session.close()
              console.log('data records', data.records)
              let r = data.records.map(item => {
                  return                                {user:item.get(0)['properties'],
userID:item.get(0)['identity']['low'] }
              })
              console.log('R', r)
              res.json(r);       })
          .catch( err => {
              res.send(err)
          }) });
      // Friends BLOCK
      app.get('/api/getFriends/:friend_id', ensureToken, isFriend, function (req, res)
{
          let userID;
          console.log('me id', res.locals.userID)
          if(req.params.friend_id == 'me') userID = res.locals.userID
          else userID = req.params.friend_id
          console.log('ID = ',userID)
          session.run(`MATCH (u:User)-[:FRIENDS]->(f:User)-[:FRIENDS]->(u:User)
                      WHERE ID(u) = ${userID}
                      RETURN DISTINCT f;`)
                  .then(data => {
                      let r = data.records.map(item => {
                       //Get ID of RECORD
                      return                            {user:item.get(0)['properties'],
userID:item.get(0)['identity']['low'] }
                      })
                      res.send(r)
                  })
                  .catch( err => {
                      res.send(err)
                  })
                  .finally(() => {
                      console.log('finally');
                      session.close()
```

```
                })})
      //get Followers
      app.get('/api/getFollowers/:friend_id', ensureToken, isFriend, function (req,
res) {
          let userID;
          console.log('me id', res.locals.userID)
          if(req.params.friend_id == 'me') userID = res.locals.userID
          else userID = req.params.friend_id
          console.log('ID = ',userID)
          session.run(`MATCH (u:User)<-[:FRIENDS]-(f:User)
                      WHERE ID(u) = ${userID} AND NOT (u)-[:FRIENDS]->(f)
                      RETURN DISTINCT f;`)
                  .then(data => {
                      let r = data.records.map(item => {
                       //Get ID of RECORD
                      return                          {user:item.get(0)['properties'],
userID:item.get(0)['identity']['low'] }
                      })
                      res.send(r)
                  })
                  .catch( err => {
                      res.send(err)
                  })
                  .finally(() => {
                      console.log('finally');
                      session.close()
                  })})
      //Get followings
      app.get('/api/getFollowings/:friend_id', ensureToken, isFriend, function (req,
res) {
          let userID;
          console.log('me id', res.locals.userID)
          if(req.params.friend_id == 'me') userID = res.locals.userID
          else userID = req.params.friend_id
          console.log('ID = ',userID)
          session.run(`MATCH (u:User)-[:FRIENDS]->(f:User)
                      WHERE ID(u) = ${userID} AND NOT (u)<-[:FRIENDS]-(f)
                      RETURN DISTINCT f;`)
                  .then(data => {
                      let r = data.records.map(item => {
                       //Get ID of RECORD
                      return                          {user:item.get(0)['properties'],
userID:item.get(0)['identity']['low'] }
```

```
    })
    res.send(r)
})
.catch( err => {
    res.send(err)
})
.finally(() => {
    console.log('finally');
    session.close()
})})
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ УПРАВЛІННЯ СОЦІАЛЬНОЮ МЕРЕЖЕЮ

Виконав: студент 2 курсу, групи 1КН-22м
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Саїнський Б.Д.
(прізвище та ініціали)
Керівник: к.т.н., доцент каф. КН
Озеранський В.С.
(прізвище та ініціали)
«07» 12 2023 р.

Вінниця ВНТУ – 2023 рік

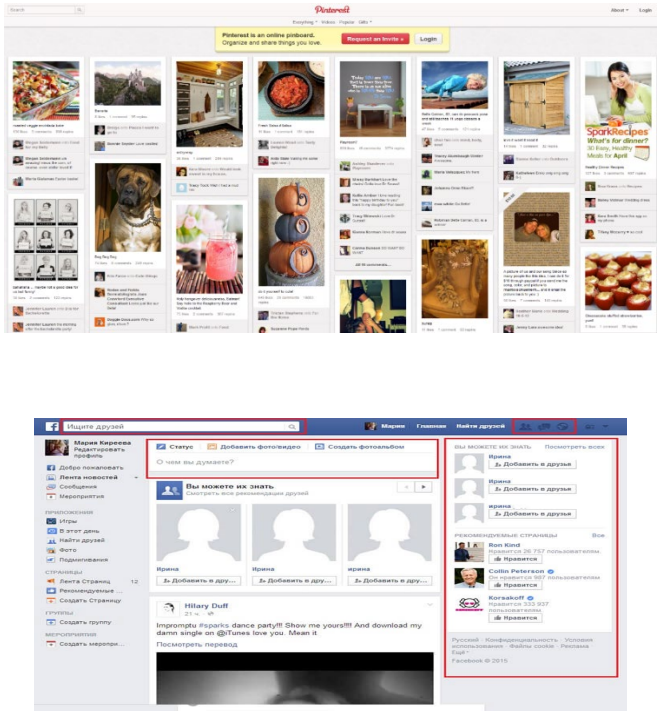

Рисунок В.1 - Програми аналоги управління соціальною мережею

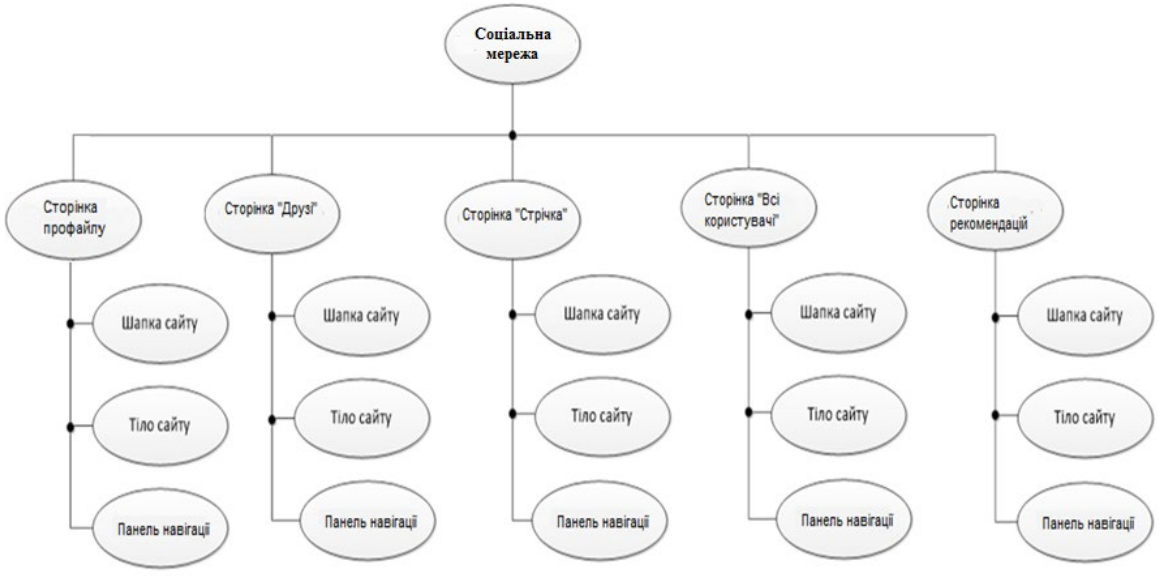

Рисунок В.2 - Структура інформаційної технології управління соціальною мережею

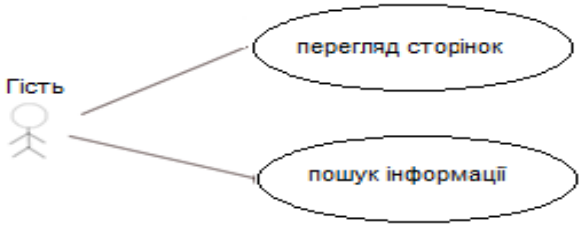

Рисунок В.3 - UML-діаграма варіантів використання для класу «Гість»

Рисунок В.4 - UML-діаграма варіантів використання для класу «Зареєстрований користувач»

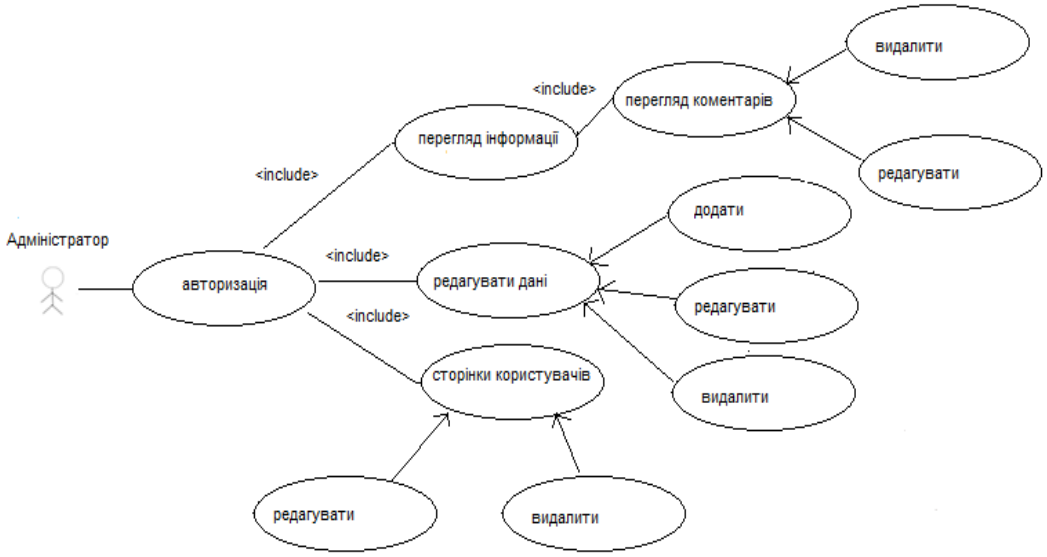

Рисунок В.5 - UML-діаграма варіантів використання для класу «Адміністратор»

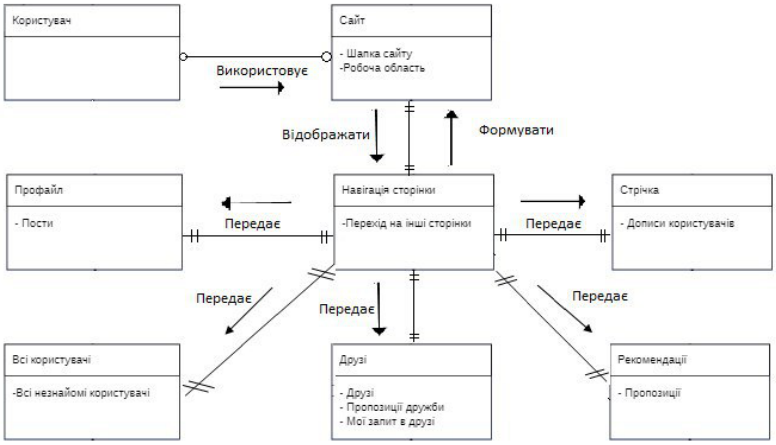

Рисунок В.6 - UML-діаграма «сутність-зв'язок» інформаційної технології управління соціальною мережею

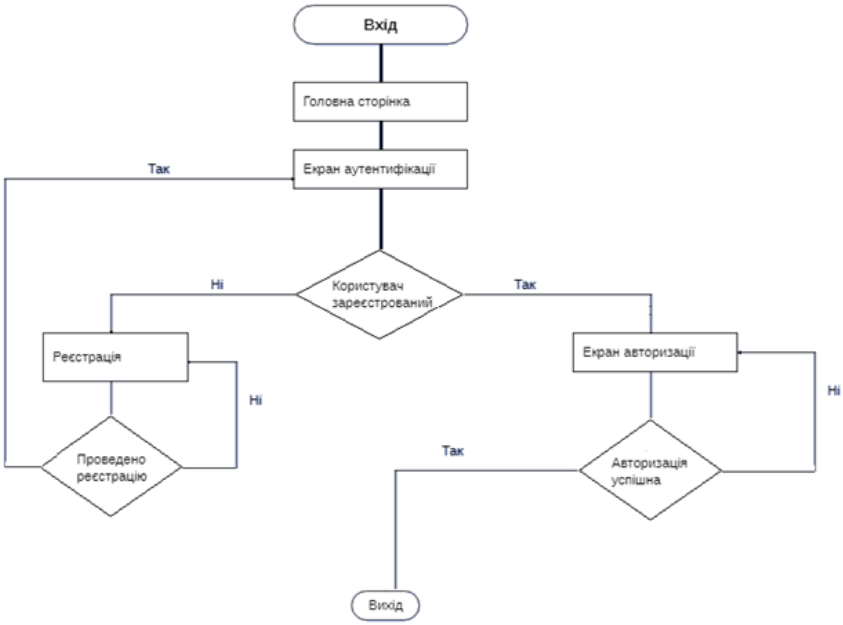

Рисунок В.7 - Алгоритм створення сесії користувача

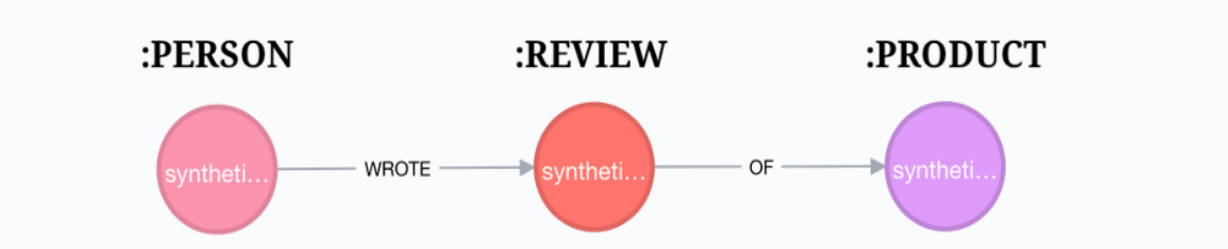

Рисунок В.8 - Використання бібліотеки TensorFlow для управління соціальною мережею

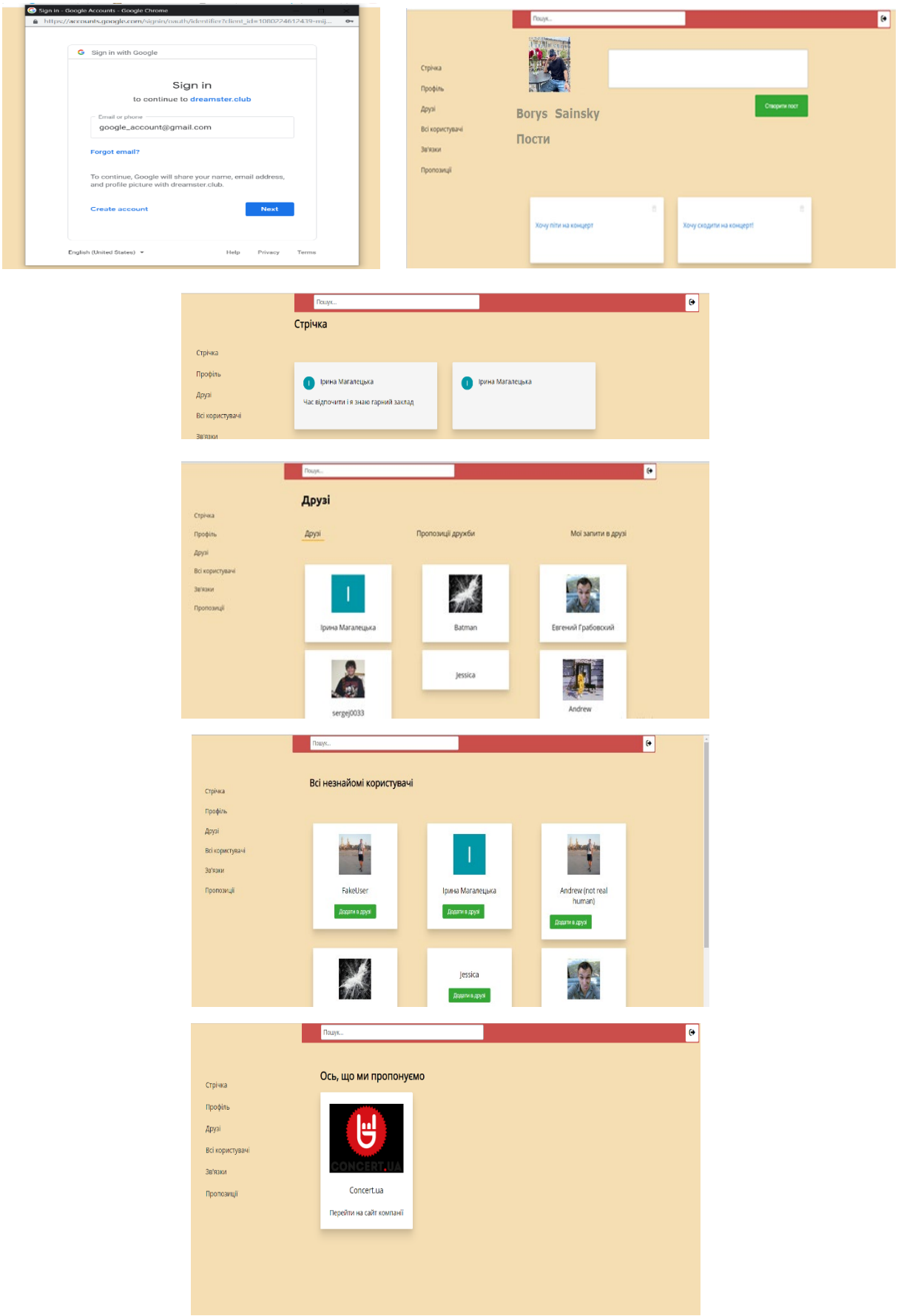

Рисунок В.9 - Інтерфейсна частина програмного забезпечення управління соціальною мережею

Додаток Г (довідниковий)

Інструкція користувача

Щоб авторизувати користувача в мережі, скористайтеся формою входу в Google (рис. Г.1).

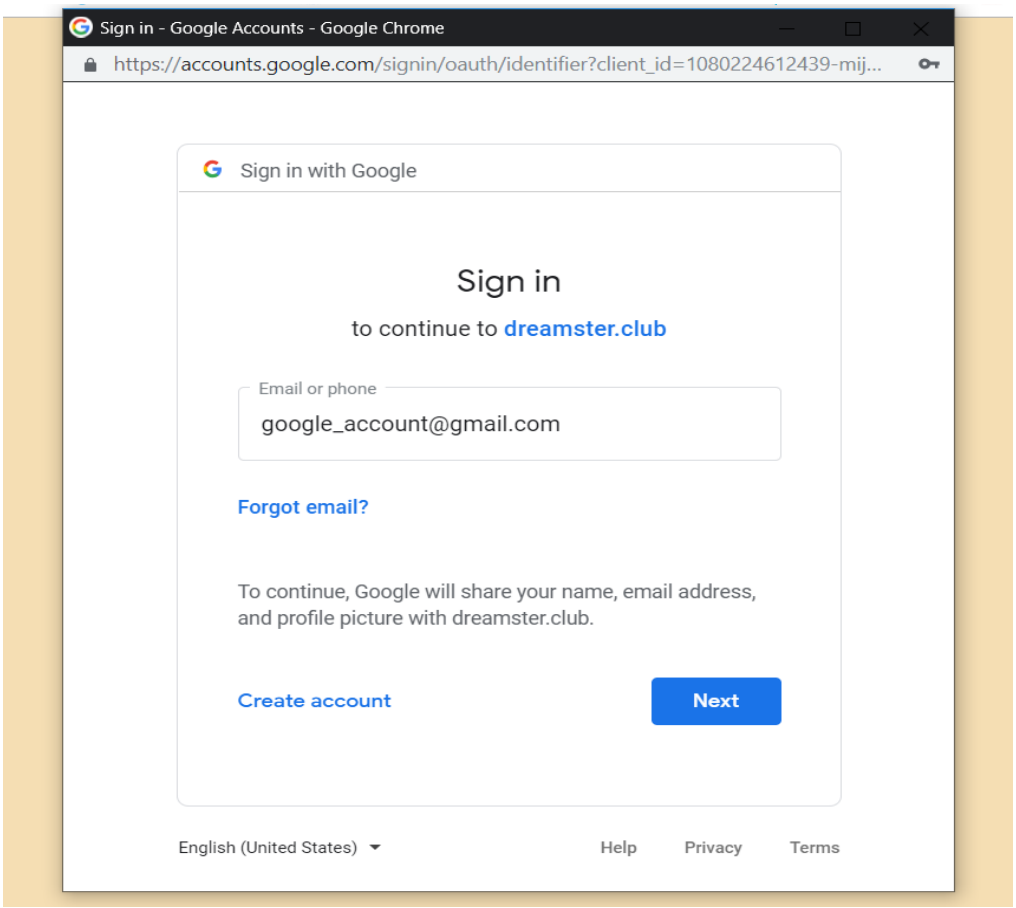

Рисунок Г.1 – Вікно авторизації корисувача соціальної мережі

Після успішної авторизації користувач переходить на сторінку свого профілю, де він може створити новий пост і переглянути наявні пости (рис. Г.2).

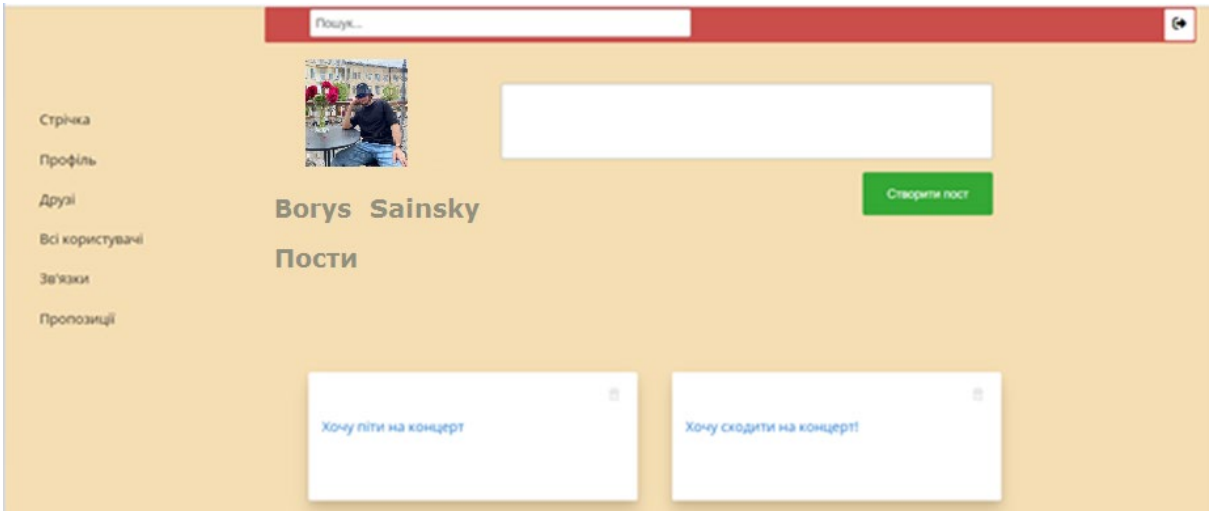

Рисунок Г.2 – Сторінка профілю користувача соціальної мережі

Кожний користувач має змогу переглядати всі свої записи в порядку їх дати формування (рис. Г.3).

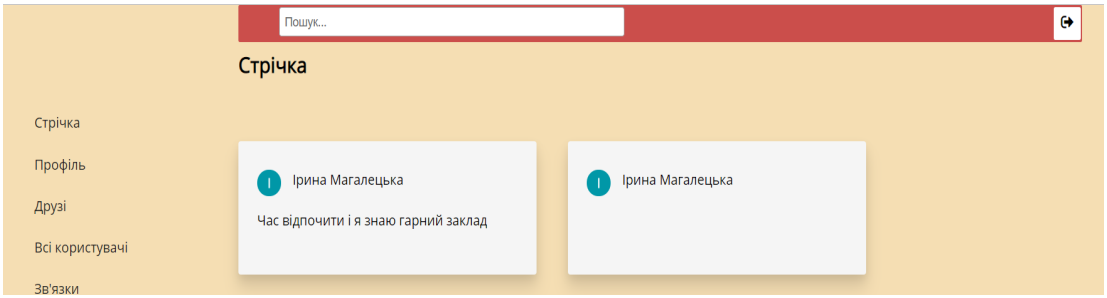

Рисунок Г.3 – Вкладинка «стрічка»

Якщо запит буде успішним, ми отримаємо список друзів користувачів та їхні публікації. Записи сортуються в порядку, починаючи з першого пункту в списку - останнього створеного повідомлення. При натисканні на елемент публікації користувач потрапляє на сторінку профілю автора.

Користувач може перейти на сторінку зі списком його «друзів» у мережі. Сторінка містить три вкладки, кожна з яких вимагає запитів на окрему адресу API. Перша вкладка містить користувачів, з якими ми маємо відношення :FRIENDS в обидві сторони (рис. Г.4).

Вкладка «Пропозиція дружби» містить список користувачів, яким ми надіслали запит «Дружба». Тому вони мають наш додаток на вкладці «Пропозиція дружби».

Результати запиту на кожній вкладці ідентичні, але мають різні пріоритети, тому ми отримуємо ім’я користувача та зображення профілю.

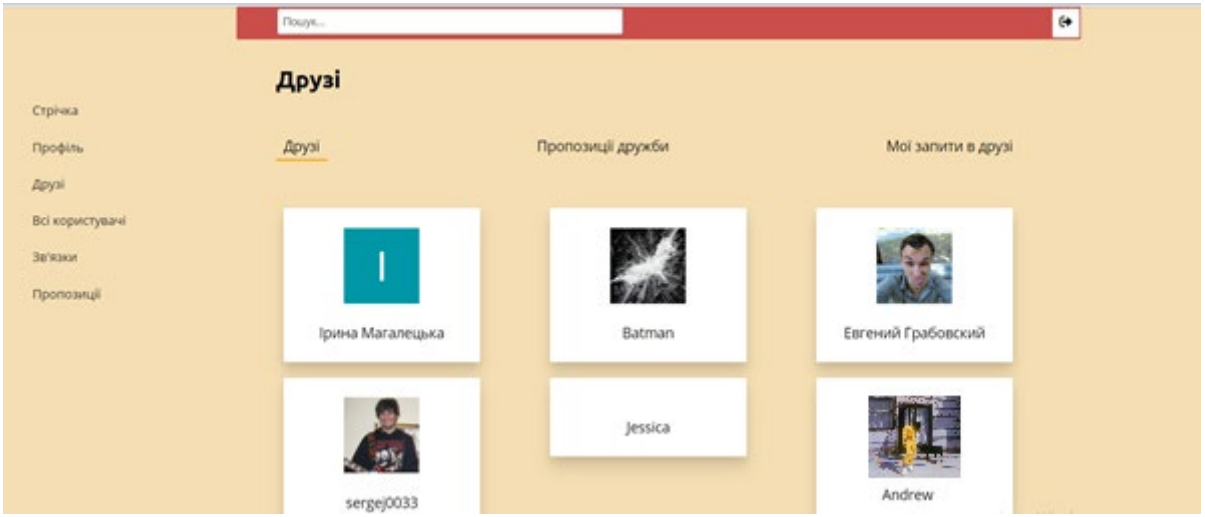

Рисунок Г.4 – Вкладинка «Друзі»

Користувач може отримати список усіх користувачів, які уповноважені в мережі додавати їх до «друзів» (рис. Г.5). Ця функція отримує поточний ідентифікатор користувача, невідомий ідентифікатор користувача та створює зв’язок даних між ними в графовій базі даних.

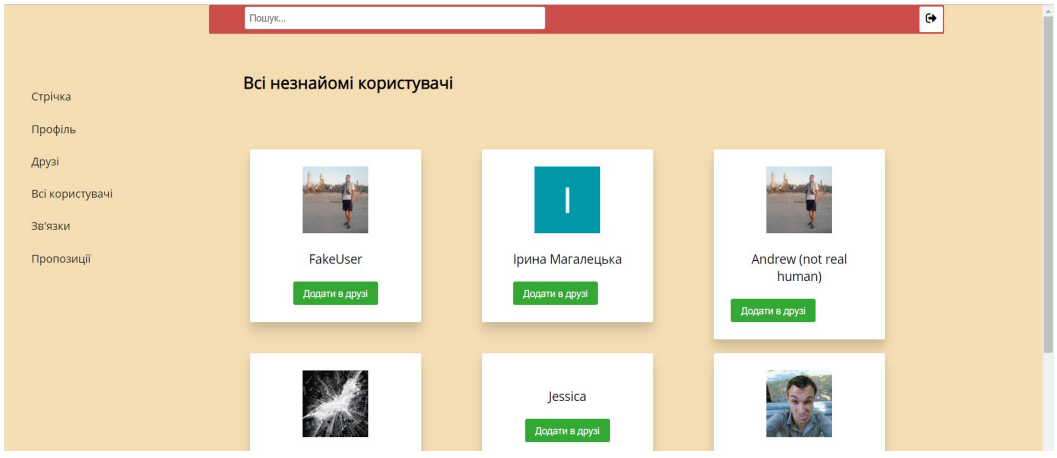

Рисунок Г.5 – Вкладинка «Всі користувачі» системи

Після створення запису користувач соціальної мережі перенаправляється на вкладинку публікації рекомендацій (рис. Г.6)

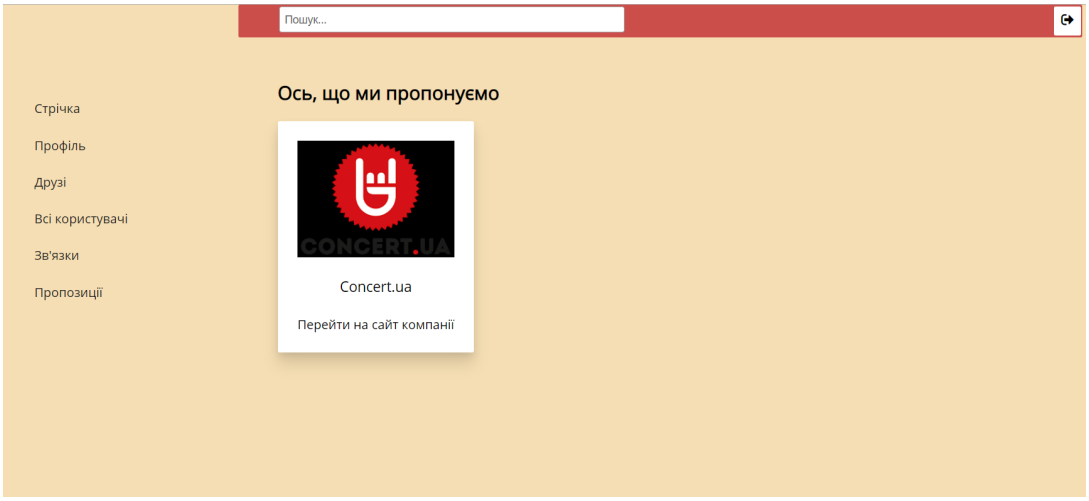

Рисунок Г.6 – Вкладинка публікації рекомендацій

Програма розбиває речення на частини та створює процес узгодження нейронної мережі, який, у свою чергу, бере наявні дані від різних компаній і знаходить відповідності в публікаціях різних користувачів і в кінцевих результатах.

Якщо є співпадіння, ми отримаємо список компаній, які відповідають нашій публікації. В іншому випадку соціальна мережа повідомить, що рекомендацій не знайдено.