Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом»

Виконав: студент II курсу,
групи 3ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Мельник Денис Олександрович

Керівник: к.т.н., доц кафедри ПЗ Коваленко. О.О.
«13» грудня 2023 р.

Опонент: к.т.н., проф. кафедри ЗІ Кондратенко Н.Р.
«13» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н, проф. Романюк О.Н.
(прізвище та ініціали)
«13» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«19» вересня 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙУ РОБОТУ СТУДЕНТУ

Мельнику Денису Олександровичу

1. Тема роботи – Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом.

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. №247.

2. Строк подання студентом роботи: 5 грудня 2023 р.

3. Вихідні дані до роботи: середовища розробки – Visual Studio Code, фреймворк Next.js, , методи та засоби створення мережі спільнот з інтегрованим штучним інтелектом, інструменти на базі штучного інтелекту – OpenAI API, Tensoflor; тренована модель штучного інтелекту – Toxicity; система управління базами даних – PlanetScale; мова запитів SQL.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз предметної галузі та постановка задач дослідження; розробка методів, алгоритмів та архітектури мережі спільнот; реалізація методів і програмних засобів для створення мережі спільнот з інтегрованим штучним інтелектом; тестування програмного додатку; економічна частина; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність т
мета, об'єкт та предмет дослідження; задачі дослідження; новизна отрим
результатів; практична цінність одержаних результатів; порівня
характеристика аналогів; метод та блок-схема алгоритму модерації кон
шляхом використання тренованої моделі ШІ; метод та блок-схема алгор
рекомендації шляхом визначення рейтенгу публікацій; проєктув
програмного забзепечення; використані технології під час розробки засто
приклад сторінок мережі спільнот; тестування додатку; апробація та публі
результатів роботи; кінцевий слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдан прийн
1-4	Коваленко О.О., к.т.н., доцент. кафедри ПЗ	19.09.2023	05.12.2
5	Причепа І.В., к.е.н., доцент кафедри ЕПВМ	21.11.2023	01.12.2

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Прим
1	Аналіз предметної галузі та постановка задач дослідження	20.09.2023 – 30.09. 2023	вик
2	Розробка методів, алгоритмів та архітектури мережі спільнот	01.10. 2023 – 10.10. 2023	вик
3	Реалізація методів і програмних засобів для створення мережі спільнот з інтегрованим штучним інтелектом	11.10. 2023 – 25.10. 2023	вик
4	Тестування програмного додатку	26.10.2023 – 20.11.2023	вик
5	Економічна частина	21.11.2023 – 01.12.2023	вик

Студент _____ Мельник Д.О.
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Коваленко О.
(підпис) (прізвище та ініціали

АНОТАЦІЯ

УДК 004.42

Мельник Д.О. Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023, 115 с.

На укр. мові. Бібліогр.: назв вик. дж. : 38; рис. : 49; табл. 15.

Магістерська кваліфікаційна робота присвячена підвищенню рівня зручності та якості взаємодії користувачів між собою у мережі спільнот шляхом використання можливостей інтегрованого штучного інтелекту за рахунок впровадження нових методів та засобів взаємодії з штучним інтелектом. Проведено аналіз сфери мережі спільнот з інтегрованим штучним інтелектом, досліджено проблеми, які виникають на шляху розробки, проведено аналіз методів інтеграції штучного інтелекту.

У магістерській кваліфікаційній роботі розроблено метод динамічної модерації контенту шляхом використання тренованої моделі штучного інтелекту в поєднанні з бібліотекою машинного навчання та метод рекомендацій контенту цільовій аудиторії шляхом визначення рейтингу публікації враховуючи дані публікацій. Методи складаються з запропонованих моделей та алгоритмів.

У роботі спроектована архітектура мережі спільнот та діаграми роботи ключових модулів додатку. Обгрунтовано вибір мови програмування та всієї відповідної екосистеми, бібліотеки машинного навчання та штучного інтелекту. Описано процес розробки графічного інтерфейсу користувача та ключових компонентів програмного забезпечення. Описано процеси розробки клієнтської та серверної частини додатку. Виконано тестування програми та наведено тестування розробленого ПЗ.

Ключові слова: мережа спільнот, штучний інтелект, тренована модель, Next.js, OpenAI.

ABSTRACT

UDC 004.42

Melnyk D.O. Development of methods and software tools for a network of communities with integrated artificial intelligence. Master's qualification work in specialty 121 - Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2023, 115 p.

In Ukrainian. Bibliography: 38 titles; Figures: 49; Table 15.

The master's thesis is devoted to improving the level of convenience and quality of user interaction in a community network by using the capabilities of integrated artificial intelligence through the introduction of new methods and means of interaction with artificial intelligence. The author analyzed the field of community networks with integrated artificial intelligence, studied the problems that arise on the way of development, and analyzed the methods of artificial intelligence integration.

The master's thesis developed a method of dynamic content moderation by using a trained artificial intelligence model in combination with a machine learning library and a method of recommending content to the target audience by determining the rating of a publication based on publication data. The methods consist of the proposed models and algorithms.

The architecture of the community network and the diagrams of the key modules of the application are designed. The choice of the programming language and the entire relevant ecosystem, machine learning and artificial intelligence libraries is substantiated. The process of developing a graphical user interface and key software components is described. The processes of developing the client and server parts of the application are described. The program is tested and the testing of the developed software is presented.

Keywords: community network, artificial intelligence, trained model, Next.js, OpenAI.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Розвиток сучасних технологій та поширення використання Інтернету в сучасному світі суттєво змінили парадигму спілкування та взаємодії між людьми. Зв'язок, який раніше був обмежений географічними та соціокультурними бар'єрами, тепер перетворився в глобальну мережу, де спілкування та обмін інформацією можливі на відстані кліку миші або дотику до сенсорного екрану. Цей великий крок у передовій технологічної революції відкриває безмежні можливості для створення і розвитку онлайн-спільнот [1].

Однак цей розвиток також створив нові виклики та можливості для спільнот, які формуються в онлайн-просторі. Спільноти стають осередками обміну інформацією, обговорення ідей, спільної діяльності та розвитку спільних інтересів. Вони можуть охоплювати все, від форумів для обговорення наукових тем до соціальних мереж, що об'єднують людей, які цікавляться конкретною тематикою або хобі.

У майбутньому онлайн-спільноти можуть стати ще більш персоналізованими та інтегрованими в інші аспекти нашого життя. Наприклад, люди можуть використовувати онлайн-спільноти для пошуку інформації, навчання, роботи та навіть особистих стосунків.

У світлі цих тенденцій, розробка програмного забезпечення для мереж спільнот стає актуальним завданням. Програми, які об'єднують користувачів на платформах для обговорення, обміну думками, співпраці та розвитку інтересів, стають основою цього нового виміру соціального життя.

Зараз, коли штучний інтелект набуває все більшого значення в різних галузях, можливості для розробки програмного забезпечення з інтегрованим ШІ у мережах спільнот стають безмежними. Інтелектуальні алгоритми, нейронні мережі, машинне навчання і аналіз даних можуть допомогти забезпечити ефективне функціонування спільнот, прогнозувати їхні потреби, рекомендувати вміст та забезпечити безпеку користувачів [2].

Штучний інтелект також може підтримувати процеси взаємодії та співпраці в онлайн-спільнотах, надаючи інструменти для ефективної комунікації та спільної роботи. Наприклад, автоматизовані системи можуть полегшити пошук інформації, взаємодію в групах чи партнерство в проектах.

Поєднання розвитку онлайн-спільнот та використання штучного інтелекту відкриває нові перспективи для створення інноваційних та динамічних віртуальних просторів, де персоналізація та ефективність взаємодії стають не лише можливістю, але і реальністю нашого онлайн-життя.

Отже, актуальність пояснюється тим, що спільноти в Інтернеті стають осередками обміну інформацією, обговорення ідей, спільної діяльності та розвитку спільних інтересів, і програми, що об'єднують користувачів в мережі, допомагають забезпечити якісне функціонування цих спільнот. З використанням штучного інтелекту можливо покращити безпеку користувачів, ефективність взаємодії та рекомендаційний вміст, сприяючи подальшому розвитку цього нового виміру соціального життя в онлайн-середовищі.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення рівня якості взаємодії користувачів між собою у мережі спільнот шляхом використання можливостей інтегрованого штучного інтелекту та за рахунок впровадження нових методів та засобів динамічної модерації та рекомендації контенту цільовій аудиторії.

Основними **задачами** дослідження є:

– провести дослідження можливостей сучасних методів інтеграції штучного інтелекту в мережі спільнот;

– розробити метод створення мережі спільнот з використанням штучного інтелекту;

– розробити метод інтеграції модулю генерації тексту з використанням штучного інтелекту;

– розробити метод модерації контенту шляхом використання тренованої моделі штучного інтелекту;

– розробити метод рекомендації шляхом визначення рейтингу публікацій;

– провести проектування архітектури мережі спільнот;

– розробити програмні компоненти та модулі для запропонованих методів;

– провести тестування розробленого програмного засобу.

Об’єкт дослідження – процеси створення мережі спільнот з штучним інтелектом.

Предмет дослідження – методи та програмні засоби для інтеграції штучного інтелекту з мережами спільнот.

Методи дослідження. У процесі досліджень використовувались: теорія множин – для формування математичних моделей; аналізу – для визначення особливостей існуючих наукових та практичних підходів до створення мережі спільнот та використання елементів штучного інтелекту; інтеграції штучного інтелекту; модерації контенту шляхом використання тренованої моделі штучного інтелекту; методи проектування програмного забезпечення; методи побудови компонентів програми та їхніх взаємозв’язків для створення мережі спільнот – для проєктування та програмної реалізації мережі спільнот; методи тестування – для перевірки роботи програмних модулів.

Наукова новизна отриманих результатів:

– Подальшого розвитку отримав метод динамічної модерації контенту, який, на відміну від існуючих, дозволяє отримати більш високий рівень достовірності та безпеки використання контенту шляхом використання тренованої моделі штучного інтелекту в поєднанні з бібліотекою машинного навчання для автоматизованої перевірки вмісту публікацій на наявність ненормативного наповнення, що забезпечує більш високий рівень забезпечення прозорості та безпеки використання контенту.

– Подальшого розвитку отримав метод рекомендацій контенту цільовій аудиторії, який, на відміну від існуючих, дозволяє відокремлювати особливо популярні теми для обговорення на платформі шляхом визначення рейтингу публікації, враховуючи дані публікацій, що, в свою чергу, дозволяє більш точно визначати цільову аудиторію для визначених обговорень, проєктів, співпраці.

Практична цінність отриманих результатів. Практична цінність результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано програмні модулі та моделі мережі спільнот, які можуть бути запроваджені для мереж спільнот з метою забезпечення автоматизованої модерації та рекомендації контенту.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях автору належать такі результати: правила інтеграції штучного інтелекту [3]; модель використання штучного інтелекту у комп'ютерній візуалізації [4].

Апробація матеріалів магістерської кваліфікаційної роботи. Результати роботи доповідалися на таких конференціях:

– "Електронні інформаційні ресурси: створення, використання, доступ" 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023.

– CXXXV міжнародна інтернет — конференція «наукові підсумки 2023 року», м. Запоріжжя, 2023.

Публікації. Основні результати досліджень опубліковано в 2 наукових працях у збірниках матеріалів конференцій:

– Мельник Д.О. Правила інтеграції штучного інтелекту. Міжнародна науково-практична Інтернет конференція "Електронні інформаційні ресурси: створення, використання, доступ" 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. 323с.

– Мельник Д.О. Модель використання штучного інтелекту у комп'ютерній візуалізації. CXXXV міжнародна інтернет — конференція «наукові підсумки 2023 року», м. Запоріжжя. 2023.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз сфери мереж спільнот з інтегрованим штучним інтелектом

Аналіз мереж спільнот з інтегрованим штучним інтелектом – це підхід до вивчення взаємин між різними групами користувачів в соціальних мережах з використанням методів машинного навчання та штучного інтелекту. Цей підхід може бути застосований для різних цілей, таких як визначення впливу окремих користувачів на мережу, прогнозування поведінки користувачів та ідентифікації важливих груп у мережі.

Переваги використання інтегрованого штучного інтелекту для аналізу мереж спільнот полягають у тому, що цей підхід дозволяє виявляти складні залежності між користувачами та групами, а також варіативність поведінки користувачів у соціальних мережах.

Інтегрований підхід поєднує в собі різні методи машинного навчання та статистичного аналізу, такі як кластерний аналіз, моделювання тем, рекомендаційні системи та інші. Один з найпопулярніших методів для аналізу мереж спільнот з використанням штучного інтелекту - це аналіз тематичних груп.

Для аналізу тематичних груп в мережі використовується метод LDA (Latent Dirichlet Allocation), який дозволяє виявляти теми, що об’єднують групи користувачів. Цей метод використовує статистичні методи для виявлення складових тематичних груп, які пов’язані одна з одною.

Для вивчення взаємин між різними групами користувачів в мережі використовуються методи кластерного аналізу та аналізу графів. Кластерний аналіз дозволяє об'єднати користувачів у групи за спільними ознаками, такими як зацікавленість у тих же темах. Аналіз графів дозволяє визначити, в якій мережі користувачів утворюють групи, залежно від зв'язків та взаємодії між ними у мережі.

Штучний інтелект також може бути застосований для прогнозування поведінки користувачів у мережі. Наприклад, на основі даних про попередню поведінку користувачів, можна визначити, які акції які користувачі здійснять в майбутньому. Це дає можливість зробити більш точні прогнози щодо будь-яких заходів в мережі та залучити більше користувачів до взаємодії у мережі.

Інтегрований підхід до аналізу мереж спільнот з використанням штучного інтелекту дозволяє виявити складні взаємини між різними групами користувачів у соціальних мережах. Такі підходи дозволяють краще розуміти поведінку та взаємини між користувачами та групами у мережі, що може бути корисним для розробки нових стратегій взаємодії в соціальних мережах.

Область інтеграції штучного інтелекту з соціальною мережею об'єднує дві ключові складові: мережі спільнот і штучний інтелект, сприяючи покращенню спілкування, розвитку спільнот та поліпшенню користувацького досвіду.

Програмне забезпечення мереж спільнот стало невід'ємною частиною життя в інтернет-середовищі. Вони охоплюють різноманітні платформи, від соціальних мереж до спеціалізованих форумів та додатків для співпраці. Однак з ростом обсягу інформації і активності користувачів, важливість інтеграції штучного інтелекту зросла значно.

Штучний інтелект додає нові можливості до програм спільнот, забезпечуючи автоматизацію багатьох процесів. Це включає в себе інтелектуальні алгоритми для персоналізації вмісту, рекомендацій, виявлення порушень правил і модерації, а також аналіз настроїв та сентименту у текстовому вмісті.

Метою досліджень в напряму використання ШІ в спільнотах є поліпшення взаємодії між користувачами у спільнотах, збільшення залученості та задоволення від користування, а також забезпечення безпеки і ефективності в цифровому просторі. Програми з інтегрованим ШІ можуть автоматизувати рутинні завдання, що звільняє час для більш цікавого та значущого спілкування, сприяючи розвитку спільнот.

Проте існують виклики, пов'язані з розробкою такого програмного забезпечення, включаючи етичні питання, приватність даних, а також ризики

пов'язані із неправильною або нейтральною поведінкою штучного інтелекту. Регулювання і стандартизація таких систем є важливими аспектами для забезпечення ефективності і безпеки в мережах спільнот.

Усе це робить предметну область програмного забезпечення мереж спільнот з інтегрованим штучним інтелектом динамічною та важливою для подальшого розвитку цифрового співтовариства, що прагне покращити спілкування, обмін знаннями та розвиток спільних інтересів у віртуальному просторі.

Враховуючи всі вищезгадані аспекти сучасної області програмного забезпечення для мереж спільнот з інтегрованим штучним інтелектом, наявність розроблювального додатка, який зможе ефективно вирішити всі ці виклики і проблеми, є важливим кроком вперед. Нове програмне забезпечення має потенціал поліпшити користувацький досвід та забезпечити більш ефективну та зручну взаємодію між учасниками спільноти.

Інтегрований штучний інтелект в новому додатку може забезпечити високий рівень персоналізації, пропонуючи користувачам вміст, який точно відповідає їхнім інтересам та потребам. Він здатний навчитися і пристосовуватися до змінних уподобань користувачів, надаючи їм найкращий досвід у спільноті.

Крім того, розроблюваний додаток може виправити обмеженість аналітичних можливостей і підвищити якість взаємодії, надаючи користувачам доступ до більш розширених інструментів аналізу даних і забезпечуючи зручні умови для спільної роботи в спільноті.

З цим новим додатком, користувачі матимуть можливість насолоджуватися більш персоналізованим та ефективним досвідом у своїй мережі спільноти. Такий підхід може вирішити численні проблеми, які існують в існуючих аналогах, і відкрити нові можливості для розвитку та зростання спільнот у цифровому світі.

1.2 Порівняняльний аналіз додатків

Великі соціальні мережі, такі як Instagram, Twitter і Facebook, використовують штучний інтелект для модерації контенту і забезпечення безпеки своїх платформ. Але останнім часом спостерігається інтерес до більш глибокої інтеграції штучного інтелекту безпосередньо в додатки соціальних мереж для покращення взаємодії з користувачами та надання їм додаткової підтримки у вирішенні різних завдань.

Інтеграція штучного інтелекту безпосередньо в додатки соціальних мереж є справжньою інновацією в цій сфері. Вона дозволяє розширити функціональність платформи, надаючи користувачам інструменти для створення контенту, генерації тексту. Це робить взаємодію користувачів з платформою більш динамічною і заохочує їх активніше використовувати сервіс.

Наприклад, додатки для соціальних мереж, які використовують інтегрований штучний інтелект, можуть допомогти користувачам у створенні цікавих постів, генерації тексту за кілька клацань. Вони можуть надавати рекомендації щодо контенту або допомагати в аналізі статистики та взаємодії зі спільнотою. Це розширює можливості користувачів і робить їхній досвід більш цікавим та зручним.

Такі інновації в сфері соціальних мереж підкреслюють значущість штучного інтелекту і його потенціал для поліпшення якості взаємодії та забезпечення покращеного користувацького досвіду в цифровому світі. Вони також свідчать про те, що розробники активно досліджують нові способи використання інтелектуальних технологій для вирішення потреб користувачів у соціальних мережах.

Розглянемо існуючі аналоги та порівняємо їх з власною мережею спільнот з інтегрованим штучним інтелектом та їхню реалізацію можливостей ШІ.

Twitter використовує ШІ різними способами. Одним із способів є використання алгоритмів машинного навчання для підбору персоналізованого контенту для часової шкали кожного користувача [5]. Ці алгоритми аналізують

минулу поведінку користувача, наприклад облікові записи, на які вони підписані, твіти, з якими вони взаємодіють, і теми, якими вони цікавляться, щоб визначити, який вміст їм показувати. Крім того, Twitter використовує штучний інтелект для вдосконалення своїх систем рекомендацій, пропонуючи облікові записи для підписки або твіти для взаємодії на основі інтересів і вподобань користувача. Але як було згадано вище Twitter не реалізував інструменти взаємодії з користувачами для покращення інтеракції з додатком. Інтерфейс користувача даного додатку зображено на фото 1.1.

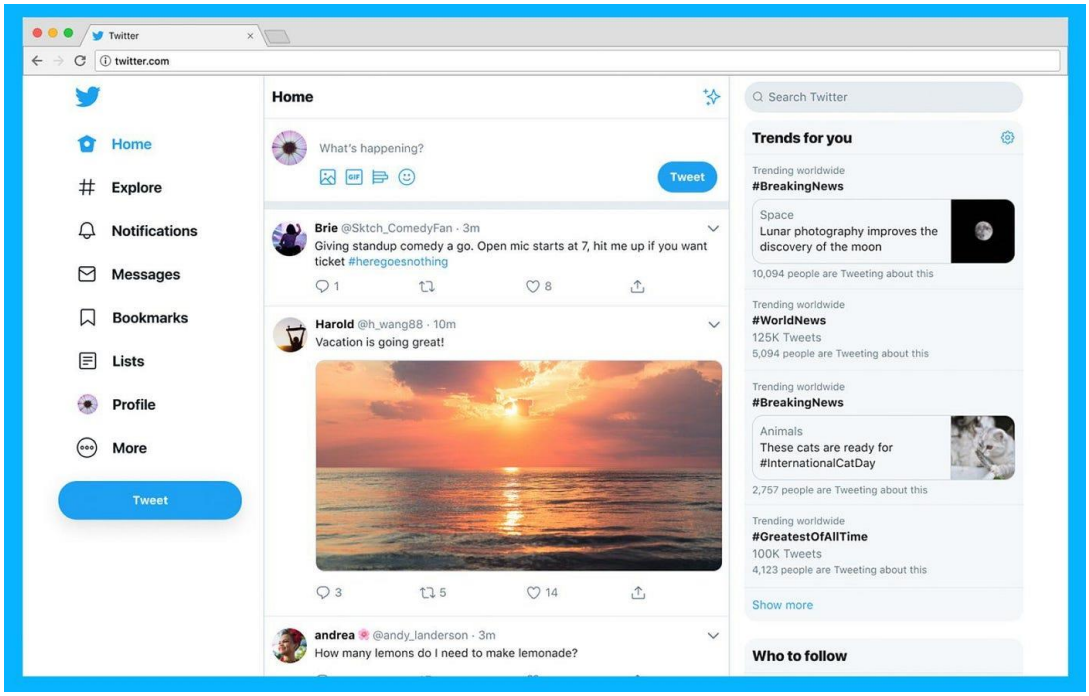

Рисунок 1.1 – користувацький інтерфейс Twitter

Facebook є однією з найбільш популярної соціальною мережою, яка почала задіювати ШІ для аналітики [6]. Facebook використовує алгоритми машинного навчання, щоб аналізувати активність користувачів та надавати персоналізовані рекомендації стосовно контенту, такого як повідомлення в новинному стрічці, сторінки для підписки, і рекламні оголошення. Це допомагає покращити залученість користувачів та забезпечити їм цікавий контент. Також мережа використовує алгоритми для виявлення і фільтрації неприпустимого контенту, що порушує правила, такого як спам, образливі повідомлення, фейки

тощо. Автоматизовані алгоритми сприяють збереженню безпеки та чистоти платформи. Щодо аналітики, Facebook надає користувачам та підприємствам інструменти для аналізу даних та відстеження метрик ефективності. ШІ допомагає обробляти великі обсяги інформації та надавати звіти для прийняття рішень. Підсумовуючи, Facebook досить вдало використовує можливості ШІ, але все ж не реалізована можливість самих користувачів взаємодіяти з даною технологією. Інтерфейс користувача даного додатку зображено на фото 1.2.

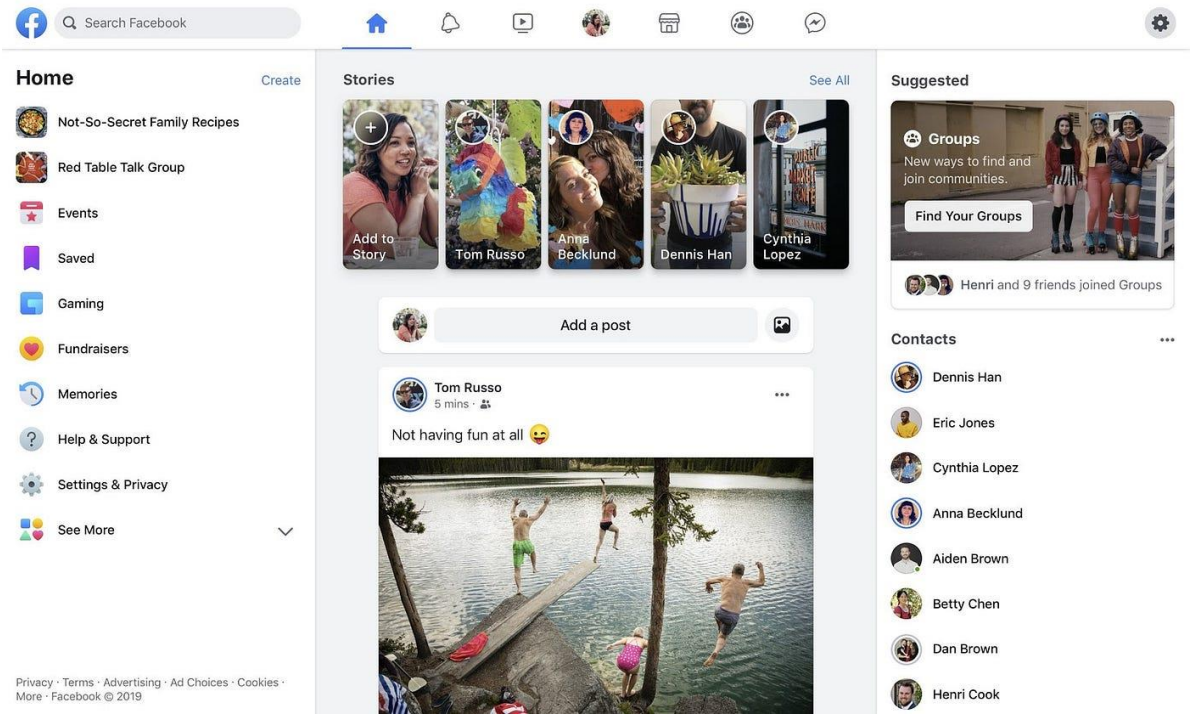

Рисунок 1.2 – користувацький інтерфейс Facebook

Instagram використовує технологію штучного інтелекту для забезпечення різноманітних функцій і алгоритмів на своїй платформі [7]. Штучний інтелект допомагає персоналізувати взаємодію з користувачем, а також покращувати точність розпізнавання зображень і відео. За допомогою штучного інтелекту Instagram прагне у майбутньому підвищити залучення користувачів, захистити безпеку користувачів і забезпечити більш приємний досвід роботи в соціальних мережах. Instagram використовує алгоритми штучного інтелекту для аналізу поведінки користувачів, зокрема їхніх лайків, коментарів і взаємодії, щоб

пропонувати персоналізований контент. Розуміючи вподобання та інтереси користувача, штучний інтелект допомагає Instagram рекомендувати дописи, історії та облікові записи, які більш доречні для людини. Ця функція рекомендацій щодо вмісту на основі штучного інтелекту спрямована на те, щоб зацікавити користувачів, демонструючи вміст, який їм, ймовірно, сподобається. Головним недоліком даної мережі є відсутність взаємодії користувачів з функціоналом ШІ. Інтерфейс користувача даного додатку зображено на фото 1.3.

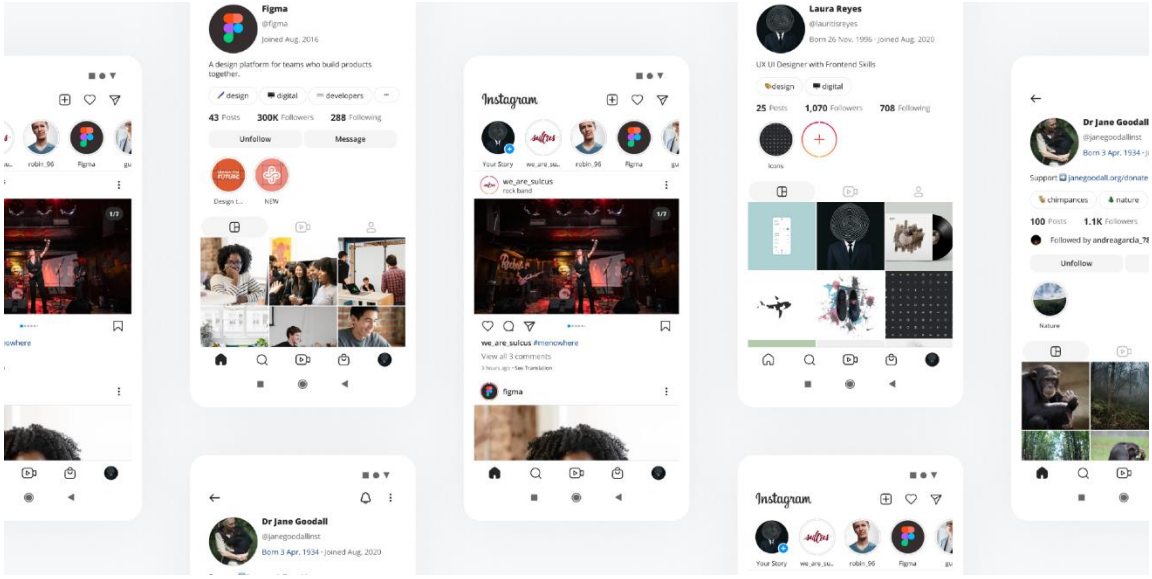

Рисунок 1.3 – користувацький інтерфейс Instagram

Усі вище згадані додатки чудово опановують можливості, які надає ШІ, але вони дещо забули про подальшу інтеграцію його можливостей та надання самим користувачами право на взаємодію з технологією. Зазначена проблема вирішена у розроблюваному додатку шляхом інтеграції можливостей ШІ та надання користувачам можливості на взаємодію з ним.

Зведемо у таблицю результати аналізу усіх аналогів розроблюваної системи, визначимо основні недоліки та переваги їх функцій прогнозування розміру витрат користувача та зрівняємо із аналогічною функцією розроблюваної мережі спільнот з інтегрованим штучним інтелектом під назвою “ComunnAI”. Результат порівняння наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	Twitter	Instagram	Facebook	ComunnAI
Генерація тексту	0	0	1	1
Взаємодія користувача з ШІ	0	0	0	1
Можливість створення спільнот	1	1	1	1
Відсутність платних функцій	0	0	0	1
Автоматизована модерація контенту	1	1	0	1
Підсумковий результат	2	2	2	5

Отже, розібравши існуючі аналоги та їхні можливості, реалізації та функціонал, було виявлено певні недоліки та бажані покращення, які реалізовані у розроблюваному додатку, а також зроблено висновок, що інтеграція штучного інтелекту (ШІ) розширює можливості платформи, надаючи користувачам засоби для творчості, створення контенту, а також генерації тексту. Це стимулює активність користувачів і робить їх взаємодію з платформою більш живою та захопливою.

1.3 Аналіз сучасних методів інтеграції штучного інтелекту в мережі спільнот

Інтеграція штучного інтелекту (ШІ) в мережі спільнот - це процес, що передбачає застосування різноманітних методів та підходів для впровадження ШІ-технологій у ці мережі. Ця інтеграція включає в себе розробку та впровадження різних інструментів, які покликані полегшити різні аспекти взаємодії і комунікації між учасниками мережі спільнот.

Одним з ключових методів інтеграції ШІ в мережі спільнот є використання аналітичних систем. Ці системи дозволяють аналізувати текстову інформацію, яка генерується в мережах спільнот, виявляти тренди, настрої та ключові слова,

що допомагають розуміти, що цікавить учасників спільноти та які питання актуальні.

Інший підхід - використання чат-ботів та інших інтерактивних систем на основі ШІ. Ці роботи можуть надавати відповіді на запити користувачів, надаючи інформацію або допомагаючи вирішувати проблеми.

Підвищення безпеки в мережах спільнот є ще одним важливим аспектом інтеграції ШІ. Системи ШІ можуть виявляти загрози для безпеки, такі як кібератаки або спроби розповсюдження дезінформації, що допомагає захищати користувачів та інформацію в мережах спільнот.

Також, інтеграція ШІ включає в себе автоматизовану модерацію контенту. Це дозволяє виявляти порушення правил та вилучати несанкціонований вміст.

Необхідно відзначити, що інтеграція ШІ в мережі спільнот вимагає вивчення та аналізу даних користувачів, що створює питання щодо приватності. Тому важливо розробляти стратегії для забезпечення захисту особистих даних та використання ШІ відповідально.

Усі ці методи інтеграції ШІ в мережі спільнот спрямовані на покращення якості взаємодії між учасниками мережі, полегшення пошуку інформації та підвищення безпеки, забезпечуючи більш зручне та ефективне використання мереж спільнот.

Одним з методів інтеграції штучного інтелекту в мережі спільнот є застосування методів машинного навчання, таких як нейронні мережі та дерева рішень. Ці методи дозволяють аналізувати великі обсяги даних та автоматично виявляти складні залежності між ознаками та поведінкою користувачів у соціальному середовищі [8].

Розглянемо ще один метод інтеграції штучного інтелекту в мережі спільнот - застосування алгоритмів класифікації для автоматичного розпізнавання та сортування користувачів у мережі на основі спільних ознак та інтересів. Цей метод дозволяє автоматизувати процес аналізу та розуміння поведінки користувачів у соціальному середовищі. Інтеграція штучного інтелекту в мережі спільнот також може включати застосування аналізу

сентименту для визначення настрою та емоцій користувачів у мережі та їх впливу на інших користувачів [8].

Один з популярний методів інтеграції штучного інтелекту в мережі спільнот – застосування рекомендаційних систем для підбору контенту та розміщення його на сторінках користувачів у відповідності до їх інтересів. Однією з основних переваг інтеграції штучного інтелекту в мережі спільнот є забезпечення більш точної та швидкої аналітики соціальної мережі та розвиток більш ефективних стратегій моніторингу та аналізу поведінки користувачів [8].

Отже, інтеграція штучного інтелекту в спільноти соціальних мереж дозволяє аналізувати та розуміти поведінку користувачів. Такі методи, як машинне навчання й алгоритми класифікації, можуть аналізувати великі набори даних і виявляти складні закономірності в поведінці користувачів. Аналіз настроїв також можна використовувати для визначення настроїв і емоцій користувачів, а системи рекомендацій можна використовувати для персоналізованого контенту. Головною перевагою є точніший і швидший аналіз соціальних мереж, що веде до більш ефективних стратегій моніторингу та аналізу поведінки.

1.4 Постановка задачі дослідження та удосконалення методів та засобів автоматизації управління кадровими ресурсами

Для ефективної розробки програмного забезпечення мережі спільнот з інтегрованим штучним інтелектом було визначено наступні задачі, які необхідно здійснити під час створення програмного застосунку:

– провести дослідження можливостей сучасних методів інтеграції штучного інтелекту в мережі спільнот;

– розробити метод створення мережі спільнот з використанням штучного інтелекту;

– розробити метод інтеграції модулю генерації тексту з використанням штучного інтелекту;

– розробити метод модерації контенту шляхом використання тренованої моделі штучного інтелекту;

– розробити метод рекомендації шляхом визначення рейтингу публікацій;

– провести проектування архітектури мережі спільнот;

– розробити програмні компоненти та модулі для запропонованих методів;

– провести тестування розробленого програмного засобу.

Основні функціональні задачі розробки додатку повинні бути такі, як:

Реєстрація та аутентифікація користувачів: Розробка механізму реєстрації нових користувачів та забезпечення безпеки та конфіденційності їхніх облікових записів. Аутентифікація повинна бути надійною та забезпечувати захист від несанкціонованого доступу.

Розробка системи профілів користувачів, яка дозволяє їм створювати та налаштовувати свої профілі.

Розробка алгоритму рейтингу окремих форумів для виділення популярних тем використовуючи наявні дані та інтенсивність інтеракції користувачів з системою.

Розробка механізму для створення спільнот та створення і публікації відповідних постів з різноманітним контентом. Забезпечення можливості редагування, видалення та організації контенту у зручний спосіб.

Розроблювальний додаток має бути легко масштабованим, щоб забезпечити у подальшому розробку додаткових функцій, розширень та можливостей платформи для відповіді на зростаючі потреби та вимоги користувачів. Маштабованість поминна бути реалізована для впорядкування збільшення обсягів даних та користувацького трафіку.

Розробка механізму кешування даних з використанням сучасної технології. Кешування полягає в тимчасовому зберіганні обчислених або отриманих даних у швидкодіючій пам'яті, щоб майбутні запити до цих даних

виконувалися швидше, без необхідності повторного обчислення або запиту до джерела даних.

Розробка методу мережі спільнот з інтегрованим штучним інтелектом для надання використання можливостей ШІ, щоб забезпечити зручність викроистання додатку та модерування контенту.

Отже, сформовано основні функціональні, які повинні бути реалізовані в межах поставлених задач під час розробки програмного забезпечення мережі спільнот з інтегрованим штучним інтелектом.

1.5 Висновки розділу 1

У першому розділі було проведено аналіз сфери мереж спільнот з інтегрованим штучним інтелектом, порівняно аналоги існуючих реалізацій з розроблюваним додатком, проведено аналіз методів інтеграції штучного інтелекту у подібні додатки та визначено основні функції, які повинні бути втілені в рамках поставлених завдань під час розробки програмного забезпечення для мережі спільнот з інтегрованим штучним інтелектом. Доведено, що інтеграція штучного інтелекту (ШІ) розширює можливості платформи, надаючи користувачам засоби для творчості, створення контенту, а також генерації тексту.

2 РОЗРОБКА МЕТОДІВ, АЛГОРИТМІВ ТА АРХІТЕКТУРИ МЕРЕЖІ СПІЛЬНОТ

2.1 Розробка методу мережі спільнот з використанням можливостей штучного інтелекту

Розробка програмного засобу може виконуватися за різними методами, в залежності від особливостей проекту, вимог та внутрішніх процесів розробки.

Метод створення ПЗ мережі спільнот з інтегрованим ШІ включає в себе:

– визначення найбільш підходящої моделі розробки програмного забезпечення відповідно до вимог;

– визначення логіки роботи програмного застосунку та покращення недолків існуючих аналогів;

– проектування архітектури системи з усіма модулями, компонентами, підсистемами, а також опис та встановлення зв’язку між ними;

– розробка алгоритмів роботи застосунку, для розуміння усіх процесів додатку крок за кроком.

Для початку розробки ПЗ для мережі спільнот з інтегрованим ШІ необхідно вибрати модель розробки, порівнявши аналоги, було обрано ітераційну модель розробки, основні принципи якої зображено на рисунку 2.1 [9].

Рисунок 2.1 – Модель ітераційного розвитку ПЗ

Модель ітераційного розвитку була обрана для розробки мережі спільнот з інтегрованим штучним інтелектом з наступних причин:

Ця модель надає гнучкість в змінах вимог, що особливо важливо в контексті мереж спільнот, де вимоги можуть змінюватися в залежності від поведінки користувачів та розвитку спільноти.

Модель ітераційного розвитку передбачає розбиття проекту на ітерації, дозволяючи швидко впроваджувати нові функції та отримувати фідбек від користувачів.

У мережах спільнот, де важливо забезпечити безпеку та правильну роботу функцій, ітеративний підхід дозволяє більше уваги приділити тестуванню та валідації кожної ітерації перед впровадженням. Також модель сприяє спільній роботі з користувачами на ранніх етапах розробки та врахуванню їхніх потреб.

Цей підхід дозволяє збирати та аналізувати дані про використання додатку для постійного вдосконалення, а також швидко реагувати на ринкові зміни та залишати вашу мережу спільнот актуальною та конкурентоздатною.

Розробка алгоритмів є фундаментальним етапом у створенні програмного забезпечення і має важливе значення з наступних причин.

Алгоритми – це послідовність точно визначених дій, які призводять до вирішення поставленої задачі чи певного завдання і на сьогодні уже створено величезну кількість алгоритмів для вирішення важких задач, що полегшують написання коду будь-якому програмісту, особливо початківцям [10].

До методу розробки подіного роду застосунків особливу увагу приділялють алгоритмам, адже це основна складова частина будь-якої програми. Вони визначають логіку роботи програмного застосування та спосіб взаємодії з даними та користувачем. Правильно спроектовані алгоритми допомагають забезпечити правильну та ефективну роботу програми, зменшуючи ймовірність помилок та некоректної поведінки [11].

Також алгоритми визначають якість програмного забезпечення. Некоректно розроблені алгоритми можуть призвести до вразливостей і помилок, які можуть бути використані зловмисниками для атак на систему. Грамотні

алгоритми забезпечують безпеку та надійність програми, допомагаючи уникати таких проблем. Розробка алгоритмів сприяє структурованому та системному підходу до роботи над програмою. Вона вимагає аналізу та розуміння проблеми з визначенням послідовних етапів її вирішення. Цей підхід допомагає команді розробників краще розподіляти завдання, спрощує комунікацію та полегшує відслідковування прогресу.

Основний алгоритм роботи розроблюваного додатку включає в себе наступні етапи:

Створення облікового запису: Користувачі реєструються на платформі, створюючи свій обліковий запис. Обліковий запис дозволяє їм додавати контент, створювати форуми, коментувати та голосувати за інші повідомлення.

Створення тем і додавання контенту: Користувачі можуть створювати нові теми (або "пости") для обговорення різних тем. Вони можуть додавати текстовий контент, зображення, посилання і багато іншого.

Генерація тексту: Користувач має змогу використовувати вбудований ШІ для спрощення написання потсів шляхом генерації тексту з використанням можливостей ШІ.

Модерація контенту: Модерація контенту відбувається за рахунок використання модулю ШІ, який використовує треновану модель даних для зчитування текстового наповнення публікацій та попередження користувачів про неприпустимий контент.

Голосування: Користувачі мають можливість голосувати за теми і коментарі. Голоси поділяються на "вгору" (позитивний голос) і "вниз" (негативний голос). Це дозволяє визначити популярність контенту.

Обговорення та коментування: Користувачі можуть залишати коментарі під темами. Це створює обговорення і сприяє взаємодії користувачів.

Пошук і підписка: Користувачі можуть шукати теми за ключовими словами та підписуватися на субредділи, які їх цікавлять, для отримання оновлень у своєму стрічці.

Персоналізація: Додаток надає можливість користувачам налаштовувати свій досвід, включаючи вибір підписаних форумів, зміну тем оформлення та інші налаштування.

Також сервер займається збереженням даних користувача, для використання в наступних сесіях.

Для кращого розуміння даного алгоритму було розроблену діаграму послідовності, яку представлено на рисунку 2.2 [12].

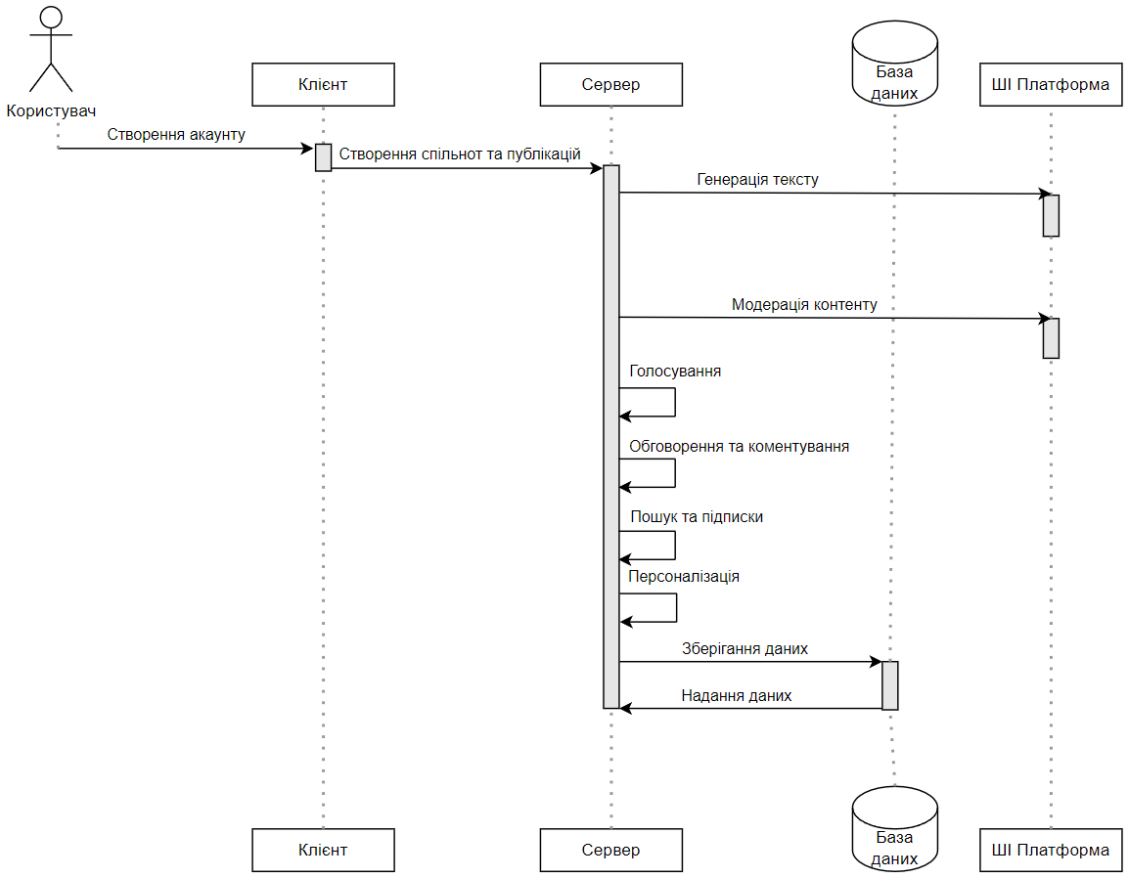

Рисунок 2.2 – Діаграма послідовності роботи додатку

Ця діаграма ілюструє потік процесів починаючи з реєстрації користувача в мережі спільнот. Користувач створює обліковий запис, а потім у серверній частині виконуються різні кроки, такі як створення спільнот, генерування тексту, голосування, увімкнення обговорень і коментарів. Сервер взаємодіє з базою даних для зберігання даних користувача.

Розробка методів є важливим етапом у процесі розробки програмного забезпечення, який вимагає глибокого розуміння задачі та відповідального підходу до її вирішення.

Отже, поданий метод дозволяє створити програний продукт, який дозволяє користувачам обмінюватися інформацією, обговорювати теми та долучатися до різноманітних спільнот зі спільними інтересами.

2.2 Метод модерації контенту шляхом використання тренованої моделі штучного інтелекту

Модерація контенту в соціальних мережах допомагає створити безпечне, інклюзивне та законне середовище для всіх користувачів [13]. Вона допомагає видаляти контент, який може завдати шкоди, наприклад, контент, що містить насильство, ненависть, дезінформацію або рекламу. Модерація контенту також допомагає рекомендувати користувачам контент, який відповідає їхнім інтересам [14]. Для покращення ефективності модерації контенту було прийнято рішення про використання тренованої моделі штучного інтелекту [15].

Насамперед необхідно визначати токсичність контенту публікацій, для запобігання його перегляду певній категорії користувачів.

Розглянемо ключові сутності, які відносяться до даної моделі.

Модель: Вираз алгоритму, який був навчений на даних розпізнавати закономірності і може робити прогнози, перетворення або реагувати.

Поріг: Мінімальна достовірність прогнозу, яку ми допускаємо.

Мітки: Масив категорій - у нашому випадку, образ.

Передбачення: Масив об'єктів, який показує необроблені ймовірності для кожного входу, а відповідність вказує на істинність або хибність. Однак, якщо жодне з передбачень не перевищує порогового значення, відповідність повертає нуль.

Модель токсичності визначає, чи підпадає текст під такі категорії: мова погроз, образи, нецензурні вирази, ненависть на основі ідентичності або відверто сексуальна лексика.

Отже, для наших цілей ми дамо моделі речення у вигляді рядка, і вона класифікує, чи порушує воно одну з перерахованих вище категорій, чи ні.

Для прогнозування буде використовуватися відсоток ймовірності для кожного рядка в масиві. Ці відсотки представлені у вигляді двох значень типу Float32 від 0 до 1. Наприклад, користувач у публікації вжив ненормативну лексику, як комусь може здатись, у певному реченні, якщо ми пропустимо цей рядок через модель і отримаємо масив, що має вигляд [0.7630404233932495, 0.2369595468044281], це на 76% не є порушенням і на 24% - ймовірно, є порушенням.

Представимо модель як множину оцінок, яка формується як функція на основі множини кількості порушень в реченні за ймовірністю порушень:

$$P = k \cdot P1 + (1 - k) \cdot P2 \tag{2.1}$$

, де:

– P - оцінка ймовірності порушення (від 0 до 1);

– P1 і P2 - відсотки ймовірності для кожного рядка в масиві (значення типу Float32 від 0 до 1);

– k - коефіцієнт важливості для текстового контенту (від 0 до 1).

Коефіцієнт k визначає, наскільки важливим є текстового контент для загального прогнозу. Якщо k більше, то перший рядок вважатиметься більш важливим для визначення загальної ймовірності порушення.

Ця формула дає можливість враховувати ймовірності для текстового контенту та вагувати його відповідно до важливості.

Модель токсичності визначає, чи містить текст токсичний контент, такий як погрози, образи, непристойності, ненависть на основі ідентичності або відверті висловлювання сексуального характеру. Модель навчалася на базі даних

користувацьких коментарів, яка містить близько 2 мільйонів коментарів, позначених як токсичні.

Розглянемо класифікацію модельованого контенту:

Ненормативна лексика: Містить лайливі слова, прокльони або іншу нецензурну лексику.

Відверта лексика: Містить посилання на відверті дії або частини тіла, або інший непристойний контент.

Напад на основі ідентичності: Негативний, дискримінаційний або сповнений ненависті коментар про людину або групу людей на основі критеріїв, що включають расову або етнічну приналежність, релігію, стать, національність або громадянство, інвалідність, вік або сексуальну орієнтацію.

Образливі: Образливий, підбурювальний або негативний коментар на адресу людини або групи людей.

Погроза: описує бажання або намір завдати болю, травми або насильства особі чи групі осіб.

Керуючись даною моделлю розглянемо приклад перевірки певної публікації на наявність ненормативної лексики:

Користувач опублікував фото з підписом “Він схожий на печерну людину, тільки набагато менш розумну!”. Дане повідомлення потрапляє до тренованої моделі, обробляється та сервер отримує відповідь від моделі про її токсичність у даному вигляді:

- Ненормативна лексика/непристойність: Ні;
- Відверта ненормативна лексика: Ні;
- Атаки на основі особистих даних: Ні;
- Образа: Так;
- Погрози: Ні.

На підставі цих даних публікація позначається спеціальною іконкою про неприпустимий контент, що надає привід подумати іншим користувачам перед тим, як перейти до публікації.

На основі поданого методу розроблено блок-схему алгоритму для мережі спільнот з інтегрованим модулем модерації контенту з використанням штучного інтелекту, яку зображено на рисунку 2.3.

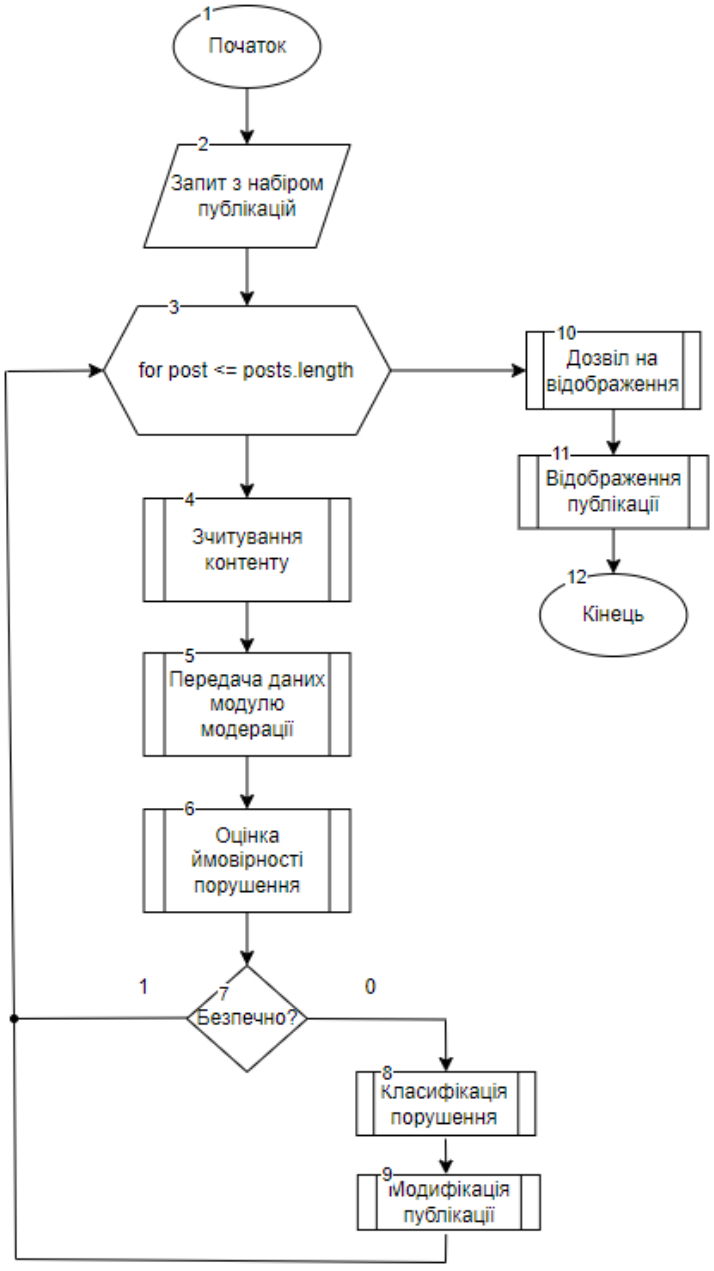

Рисунок 2.3 – Блок-схема алгоритму модерації контенту

Крок 1: Початок.

Крок 2: Сервер отримує запит з набором публікацій відповідно до обраної спільноти.

Крок 3: Програма ітерує по кожній публікації для подальшого набору дій.

Крок 4-5: Зчитується текстове наповнення публікацій та передається модулю модерації для подальшого опрацювання.

Крок 6: За допомоги бібліотеки TensorFlow.js проводиться модерація контенту шляхом використання тренованої моделі ШІ.

Крок 7-8: Якщо ймовірність висока, модуль переходить до калсифікації порушення

Крок 9: Після класифікації порушення, публікація модифікується класифікованим записом для подальшого відображення на стороні клієнту.

Крок 10: Надсилання набору публікацій на сторону клієнта з дозволом на відображення.

Крок 11: Відображення публікацій у додатку зі спеціальним ідентифіікатором.

Крок 12: Кінець.

Отже, у даному розділі було розроблено метод модерації контенту шляхом використання тренованої моделі штучного інтелекту, а також розроблено блок-схему для алгоритму усіх відповідних процесів методу.

2.3 Метод інтеграції модулю генерації тексту шляхом використання ШІ

Щоб інтегрувати ШІ для генерації тексту в додаток, потрібно виконати певні кроки.

Перш за все, потрібно вибрати ШІ-платформу. На ринку існує багато різних платформ, які пропонують різні функції та ціни. При виборі платформи слід враховувати такі фактори, як:

- Обсяг генерованого тексту;
- Якість генерованого тексту;
- Можливість налаштування параметрів генерації тексту;
- Ціна.

Після вибору ШІ-платформи потрібно зареєструватися на їхньому веб-сайті та отримати API ключ. API ключ є унікальним кодом, який дозволяє додатку взаємодіяти з ШІ-сервісом.

Наступним кроком є отриманні API ключа, який необхідно інтегрувати в додаток. Це можна зробити за допомогою різних методів, залежно від того, на якій платформі розроблено додаток.

Після інтеграції API в додаток, надається дозвіл створювати запити до ШІ-сервісу. Запити можуть включати в себе такі параметри, як опис зображення, стиль, розмір та інші.

ШІ-сервіс повинен відповідати на запит згенерованим текстом. Додаток може використовувати цей текст в інтерфейсі додатку, наприклад, для генерації контенту, відповідей або рекомендацій.

Рекомендується реалізувати обробку помилок та винятків на стороні серверу. Це допоможе запобігти виникненню проблем.

Після інтеграції ШІ, важливо протестувати та відлагодити функцію генерації тексту. Це допоможе переконатися, що функція працює належним чином.

Також важливо забезпечити документацію та моніторинг. Документація допоможе іншим розробникам використовувати функцію, а моніторинг допоможе вам відстежувати використання та продуктивність ШІ-сервісу.

Нарешті, важливо пам'ятати про забезпечення безпеки та конфіденційності. ШІ-сервіси можуть обробляти особисті дані користувачів, тому важливо вжити заходів для захисту цих даних.

Для кращого розуміння даного методу було розроблену діаграму моделі інтеграції, яку представлено на рисунку 2.4.

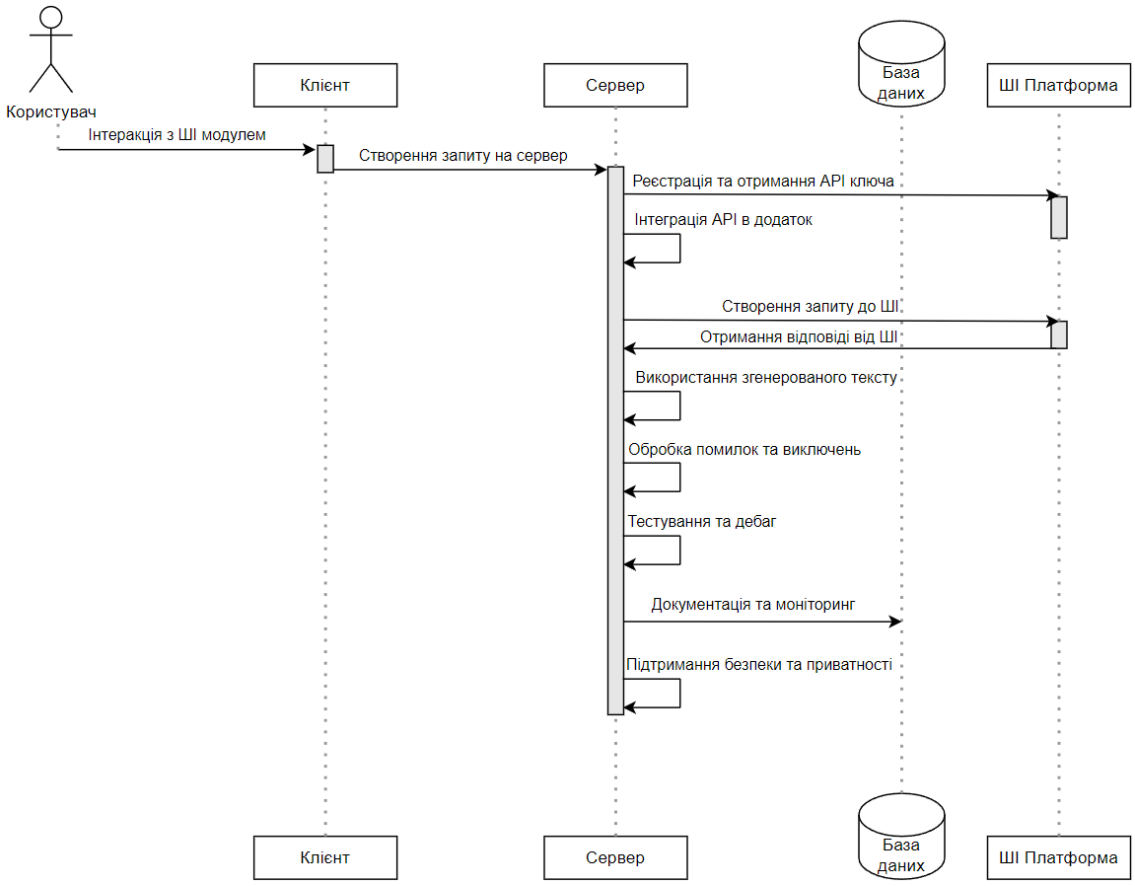

Рисунок 2.4 – Діаграма послідовності методу інтеграції ШІ

На цій діаграмі показано інтеграцію системи штучного інтелекту для генерування тексту в існуючий додаток. Потік взаємодії користувача з модулем ШІ зображено у першій частині діаграми, а кроки для інтеграції системи ШІ додано у другій частині. Сервер взаємодіє з платформою ШІ шляхом реєстрації та отримання ключа API, інтеграції API в додаток, створення запиту до ШІ, отримання відповіді та обробки помилок і винятків. Крім того, бекенд виконує тестування, налагодження, документування, моніторинг і забезпечує безпеку та конфіденційність.

Розглянемо більш детально, що відбувається під час взаємодії серверної частини додатку та ШІ платформи.

Для цього побудуємо блок-схему, яка описує процес отримання запиту з клієнту від користувача, його модерацію, обробку та оперування отриманими даними шляхом надсилання їх до використованого нами API.

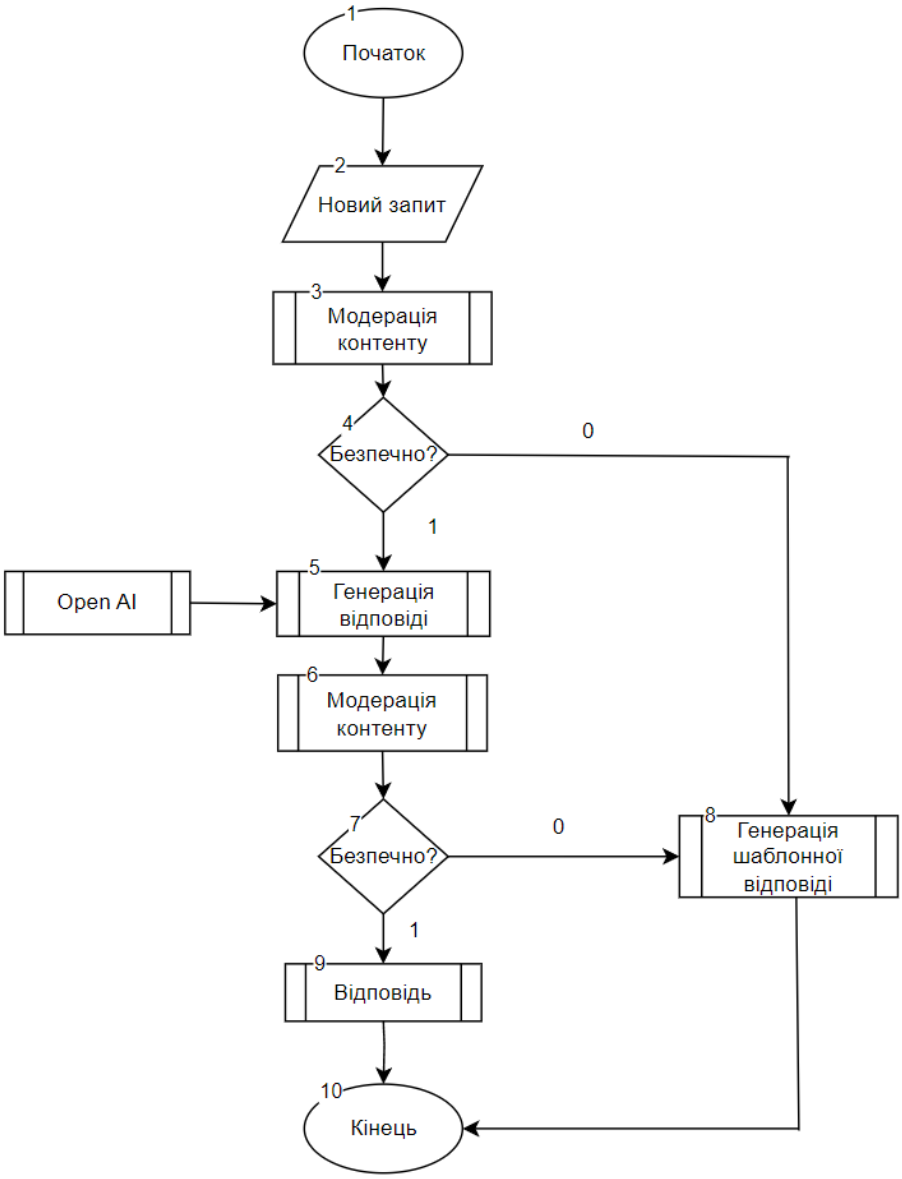

Рисунок 2.5 – Блок-схема генерації тексту

Крок 1: Початок.

Крок 2: Користувач вводить повне питання, наприклад: "Поясніть, як працює алгоритм класифікації".

Крок 3: Питання надсилається на компонент модерації контенту. Цей компонент гарантує, що питання не порушує правила безпеки та фільтрує невідповідні питання.

Крок 4: Якщо вхідні дані пройшли контент-модерацію, вони надсилаються до моделі chatGPT. Якщо вхідні дані не проходять контент-модерацію, вони переходять безпосередньо до генерації шаблонних відповідей.

Крок 5-6: Після того, як модель згенерує відповідь, вона знову надсилається до компонента модерації контенту. Це гарантує, що згенерована відповідь є безпечною, нешкідливою, неупередженою тощо.

Крок 7-8: Якщо вхідні дані пройшли контент-модерацію, вони показуються користувачеві.

Крок 9: Якщо вхідні дані не пройшли контент-модерацію, вони переходять до генерації шаблонних відповідей і показуються користувачеві у вигляді шаблонної відповіді.

Крок 10: Початок.

Отже, після розробки методу було реалізовано алгоритм з використанням відкритих бібліотек, розроблено діаграму та блок-схему модулю генерації тексту з використанням ШІ. Даний прототип може слугувати прикладом для подальшого вдосконалення, притримуючись усіх принципів масштабованості.

2.4 Удосконалення методу та моделі рекомендації шляхом визначення рейтингу публікацій

Враховуючи той факт, що метою роботи є підвищення якості взаємодії користувачів між собою у мережі спільнот розглянемо покращену модель визначення рейтингу публікацій, шляхом використання комбінації голосування, взаємодії користувачів та часових факторів для визначення рейтингу публікацій і коментарів в мережі спільнот з кількох причин:

– Це забезпечує більш точне та справедливе ранжування контенту. Голосування є чітким показником того, що користувачі вважають контент цікавим або корисним.

– Це допомагає користувачам знаходити цікавий контент. Рейтинги контенту допомагають користувачам швидко і легко знаходити контент, який, ймовірно, буде їм цікавий.

– Це допомагає підвищити залучення користувачів. Коли користувачі бачать, що їхній контент отримує високі рейтинги, вони, швидше за все, будуть

продовжувати публікувати контент. Це може призвести до більшої активності в мережі спільнот і підвищення залучення користувачів.

Найпростіший алгоритм визначення популярності публікації може бути таким [16]:

– Визначте кількість голосів "за" та "проти".
– Визначення різниці між кількістю голосів "за" та "проти".
– Чим більша різниця, тим популярніша публікація.

Формула для розрахунку рейтингу публікації за цим алгоритмом може бути такою:

$$r = L – D \tag{2.2}$$

Ця формула враховує лише кількість голосів "за", як “L” і "проти", як “D”. Вона не враховує час публікації та взаємодію користувачів. Вона є досить примітивна та не задовільняє потреби додатку.

Удосконалимо дану модель шляхом використання певного набору інформації.

Необхідно використати комбінацію голосування, взаємодії користувачів та часових факторів для визначення рейтингу публікацій і коментарів на своєму ресурсі.

У математичній нотації алгоритм виглядає так:

Враховуючи час публікації запису “A” та час 7:46:43 ранку.

8 грудня 2023 “B”, ми маємо “t”, як їх різницю в секундах:

$$t_2 = A – B \tag{2.3}$$

і x як різниця між кількістю голосів “за” - U та кількість голосів проти - “D”:

$$x = U\text{-}D \tag{2.4}$$

де “y” € {-1,0,1}

$$y = \begin{cases} 1 \text{ якщо } x > 0 \\ 0 \text{ якщо } x = 0 \\ -1 \text{ якщо } x < 0 \end{cases} \tag{2.5}$$

і “z” як максимальне значення абсолютного значення “x” і “1”

$$z = \begin{cases} |x| \text{ якщо } |x| \geq 1 \\ 1 \text{ якщо } |x| < 1 \end{cases} \tag{2.6}$$

ми маємо рейтинг, як функцію f(t, y, z)

$$\int(t, y, z) = \log_{10} z + \frac{yt}{45000} \tag{2.7}$$

Про час подачі можна сказати наступне, що стосується ранжування історій:

– Час подачі має великий вплив на рейтинг, і алгоритм буде оцінювати нові історії вище, ніж старі.

– Оцінка не буде зменшуватися з плином часу, але нові історії отримають вищу оцінку, ніж старі. Це відмінний підхід від алгоритму, який зменшує оцінку з плином часу.

Час подачі - дуже важливий параметр, зазвичай новіші історії мають вищий рейтинг, ніж старіші.

Перші 10 голосів зараховуються так само, як і наступні 100. Наприклад, історія, яка отримала 10 вподобань, і історія, яка отримала 50 вподобань, матимуть однаковий рейтинг.

Суперечливі історії, які отримують однакову кількість голосів "за" і "проти", матимуть нижчий рейтинг порівняно з історіями, які отримують переважно голоси "за" і не будуть рекомендуватись цільовій аудиторії.

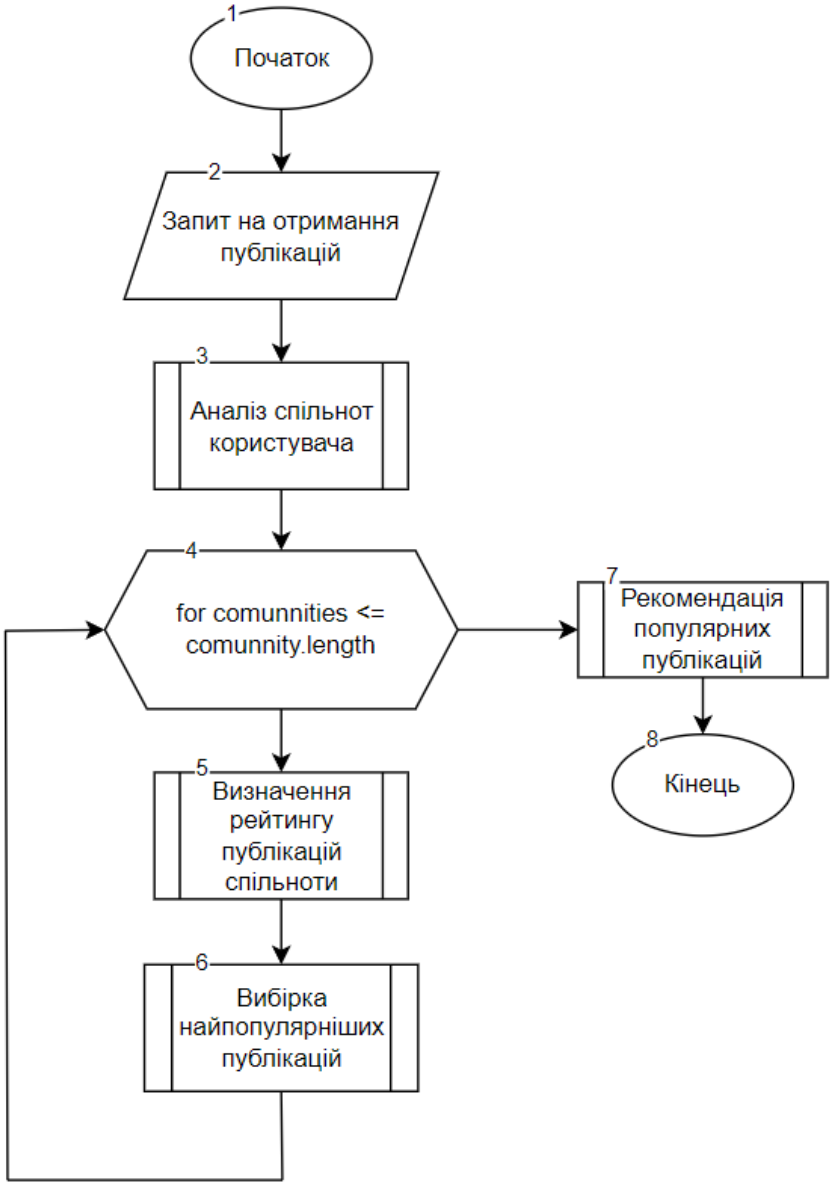

Рисунок 2.6 – Блок-схема алгоритму рекомендації публікацій

На основі поданого методу розроблено блок-схему алгоритму для рекомендації публікацій на основі його вподобань та комбінації голосування, взаємодії користувачів та часових факторів для визначення рейтингу публікацій і коментарів.

Крок 1: Початок.

Крок 2: Здійснюється запит зі сторони клієнту на отримання публікацій та їхнє відображення.

Крок 3: Системою здійснюється аналіз спільнот до яких належить користувач та на основі цих даних здійснюється підбір схожих спільнот.

Крок 4: Програма ітерує по кожній спільноті для подальших дій.

Крок 5: Здійснюється визначення рейтингу публікацій відповідно до методу.

Крок 6: З кожної спільноти відокремлюються найпопулярніші публікації.

Крок 7: Здійснюється відображення рекомендованих публікацій на стороні клієнту.

Крок 8: Кінець.

Отже, було покращено метод визначення рейтингу публікації враховуючи час та кількість голосів, що покращує рівень рекомендацій цільовій аудиторії.

2.5 Проектування архітектури мережі спільнот

Розробка загальної моделі системи є важливим кроком при створенні програмного продукту.

Така модель визначає стратегічні рішення та основні параметри системи, включаючи її структуру та взаємодію між компонентами.

Загальна модель системи визначає, як різні компоненти додатку взаємодіють між собою, включаючи розподіл функцій між серверним і клієнтським кодом, а також базову архітектуру системи.

Правильно спроектована структура дозволяє ефективно розробляти та підтримувати додаток.

Загальна модель також допомагає визначити, як компоненти системи обмінюються даними та взаємодіють один з одним. Це дозволяє забезпечити правильну і надійну роботу додатку, уникнути конфліктів та помилок взаємодії.

Крім того, правильно спроектована архітектура дає можливість легко розширювати функціональність додатку та вносити зміни, що важливо для адаптації системи до змінних потреб користувачів та ринку.

Ефективна архітектура також допомагає покращити продуктивність системи та забезпечити її безпеку.

Високорівневий дизайн для повностекового додатку відноситься до загальної архітектури та структури програмного забезпечення [17]. Він дає уявлення про те, як різні компоненти програми будуть взаємодіяти один з одним для досягнення бажаної функціональності.

Добре спроектований додаток має вирішальне значення для забезпечення безперебійної роботи користувача, мінімізації витрат на розробку та забезпечення масштабованості.

Щоб створити високорівневий дизайн для повностекового додатку, потрібно враховувати різні фактори, такі як схема бази даних, користувальницький інтерфейс, логіка додатку, зовнішні інтеграції та заходи безпеки.

Необхідно також визначити стек технологій, мови програмування та фреймворк, які найкраще відповідають вимогам проекту.

Високорівневий проектний документ зазвичай включає схему архітектури системи, яка ілюструє компоненти додатку, включаючи інтерфейс, бекенд і рівень бази даних.

Високорівневий проектний документ має важливе значення, оскільки він слугує планом для розробників, менеджерів проектів та зацікавлених сторін, а також полегшує комунікацію та співпрацю протягом усього процесу розробки.

Загалом, високорівневий дизайн для повностекового додатку є критично важливим першим кроком у розробці програмного забезпечення і гарантує створення високоякісного, масштабованого та зручного для користувача додатку.

Розроблювальна архітектура повинна бути розділена на три секції:

– Мережевий рівень - як запит буде реагувати на додаток;
– Наступний екземпляр програми, SSR;
– Зовнішні служби.

Беручи до уваги усе вище сказане розглянемо високорівневий проектний документ у вигляді рисунку 2.7.

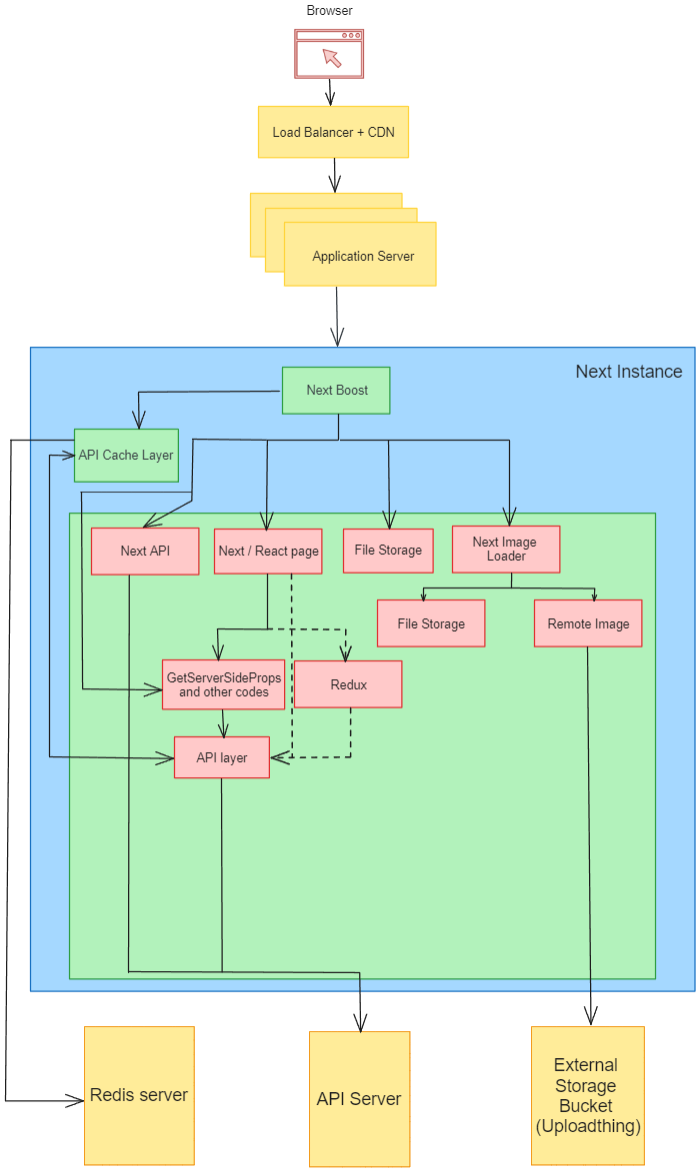

Рисунок 2.7 – Високорівневий дизайн архітектури додатку

Розглянемо мережевий рівень.

Коли користувач звертається до веб-сайту, в цьому розділі обговорюється, як цей запит потрапляє на сервер і який протокол ми повинні використовувати.

Коли користувач запитує додаток, запит надсилається до балансувальника навантаження; якщо запит стосується завантаження HTML-

контенту веб-сайту, запит надсилається до сервера додатків; якщо запит стосується завантаження статичного контенту, такого як зображення, завантажені з адмін-панелі, файли збірки, шрифти тощо, запит надсилається до CDN, і контент завантажується з CDN.

Таким чином, ми зможемо доставляти весь статичний контент з периферії, зменшуючи таким чином навантаження статичних запитів на сервер додатків.

В обох випадках ми повинні використовувати HTTP/2 або HTTP/3, причому HTTP/3 є кращим, ніж HTTP/2.

Наступний екземпляр програми, SSR.

У цьому розділі описано код додатку і те, як він має працювати; всі запити будуть проходити через користувацький сервер кешування next-boot, який буде кешувати будь-який вхідний запит; причина, чому ми використовуємо next-boost, полягає в тому, що це високопродуктивна бібліотека кешування NextJs, яка працює на базі даних SQLite; як результат, пропускна здатність дуже висока, і налаштувати цей рівень легко; запити в основному складаються з п'яти різних типів запитів.

1. Запит на сторінку. Це здебільшого запит сторінки SSR, в якому наступна програма завантажує весь матеріал і генерує HTML-версію запитуваної сторінки. Якщо нам потрібно завантажити динамічні дані разом з HTML, ми повинні використовувати реалізацію NextJs, а саме getServerSideProps.

2. Цей запит буде зроблено, коли користувач переходить з однієї сторінки на іншу, і якщо друга сторінка містить getServerSideProps, то цей запит буде зроблено. Next-boost не буде кешувати цей запит, тому нам знадобиться наш власний рівень кешування для цього коду, який може бути кешем redis.

3. Next API запити - це запити до наших визначень API NextJs, і нам зазвичай не потрібно кешувати цей рівень, якщо немає конкретного випадку використання, тому що платформа буде робити запит більшу частину часу на рівні браузера.

4. Next Image - Для кожного зображення, що завантажується на сайті, ми повинні використовувати NextImage. NextImage - це завантажувач, який перекодує зображення в певний розмір і формат, щоб мінімізувати час передачі через інтернет. Ми можемо ввімкнути кеш next-boost і вказати тут більшість зображень у форматах webp і avif.

5. File Request, цей маршрут не знадобиться, якщо нам не потрібно обслуговувати динамічний статичний контент; як зазначалося раніше, весь статичний контент буде переміщено на CDN, тому цей маршрут не знадобиться; однак, якщо нам потрібно обслуговувати карту сайту, яка є дуже динамічною і автоматично оновлюється на регулярній основі, нам може знадобитися зберігати її разом з самим сайтом; в цих випадках ми можемо використовувати цей маршрут.

Останній екземпляр програми – це зовнішній сервіс.

Сюди входять, наприклад, всі зовнішні сервіси, які ми використовуємо. Движок бази даних, redis та інші хмарні сервіси Google.

Кроки по відношенню до браузера:

Розмітка HTML:

1. Як тільки користувач вводить адресу веб-сайту, браузер надсилає перший початковий запит сторінки на сервер.

2. Балансувальник навантаження отримує цей запит.

3. Балансувальник навантаження перенаправляє запит на спеціальний сервер NextApplication.

4. Якщо файл вже був закешований, кастомний сервер перевіряє його наявність і повертає закешовані дані.

5. Якщо ні, завантажте наступний додаток і запустіть виконуваний файл Next Server, наприклад GetServerSideProps, потім завантажте повний наступний додаток і, нарешті, перетворіть цю сторінку в HTML.

6. Закешуйте HTML-файл, який було проаналізовано

7. Поверніть дані клієнту

Наступні виклики:

1. Завантажується HTML-розмітка і завантажується в браузер.

2. Після цього надсилається запит на завантаження всіх необхідних файлів, таких як CSS, JS, Fonts, Images і так далі.

3. Балансувальник навантаження отримує ці запити.

4. Балансувальник навантаження надсилає цей запит до CDN та до сховища, де зберігаються файли.

5. Ми можемо створити піддомен і підключити CDN до сховища безпосередньо, минаючи Application LoadBalancer.

6. Реактивна частина веб-сайту ініціалізується на стороні клієнта, як тільки все це буде завантажено.

7. Якщо для динамічного завантаження потрібні виклики інтерфейсного API, react зробить виклик і збереже дані в redux.

Високорівневий дизайн системи для повностекового додатку грає важливу роль у розробці програмного продукту, оскільки він визначає ключові архітектурні рішення та параметри системи.

Отже, високорівневий дизайн системи є важливою частиною розробки повностекових додатків, яка допомагає забезпечити якість та успіх проекту.

2.6 Висновки розділу 2

Отже, було розглянуто метод створення мережі спільнот з інтегрованим штучним інтелектом. Подано алгоритм генерації тексту Даний прототип може слугувати прикладом для подальшого вдосконалення та реалізації модулю генерації фото, притримуючись усіх принципів масштабованості. Розроблено алгоритм визначення рейтингу публікації враховуючи час та кількість голосів, було розроблено високорівневий дизайн системи, який є важливою частиною розробки додатків, яка допомагає забезпечити якість та успіх проекту.

3 РЕАЛІЗАЦІЯ МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ МЕРЕЖІ СПІЛЬНОТ З ІНТЕГРОВАНИМ ШТУЧНИМ ІНТЕЛЕКТОМ

3.1 Варіантний аналіз та обґрунтування вибору засобів реалізації

Так, як поставлені задачі та вимоги до розроблюваного програмного забезпечення мережі спільнот з інтегрованим штучним інтелектом є досить унікальні та беручи до уваги відсутність існуючих аналогів з реалізацією подібного функціоналу, було вирішено розробити клієнтську та серверну частини самостійно використовуючи відповідні технології.

Розробка фуллстек додатків, які включають в себе як клієнтську (frontend) частину, так і серверну (backend) частину, має численні переваги з перспективи розробника [18]:

Універсальність. Є можливість працювати з усіма аспектами додатка, від користувацького інтерфейсу до бази даних та серверної логіки. Це дозволяє розробникам бути більш універсальними та самостійними у роботі.

Збереження часу. Присутня опція виконувати різноманітні завдання без необхідності чекати на інших спеціалістів. Це дозволяє прискорити розробку та реагувати на зміни вимог швидше.

Краще розуміння всієї системи. Змога більше можливостей розуміти всю систему як цілісність, що сприяє зменшенню ризиків та поліпшенню якості коду.

Ефективне вирішення проблем. Ефективне вирішення проблеми, що виникають на різних рівнях додатка, замість того, щоб передавати їх іншим спеціалістам.

Збереження ресурсів. Можливість працювати на проєктах з обмеженим бюджетом, оскільки нема потреби додаткових спеціалістів.

Широкий спектр навичок. Потреба розвивати різноманітні навички, щоб мати змогу рівноцінно працювати як над клієнтською частиною додатку, так і над серверною та взаємодія з базами даних та різноманітними апі.

Звісно, розробка фуллстек додатків може бути вимогливою, інколи вимагати великого обсягу знань, але вона має численні переваги для розробників, які готові брати на себе більше відповідальностей та розвивати багатогранні навички.

Для розробки програмного засобу мережі спільнот було необхідно обрати сучасні, масштабовані та іноватівні засоби розробки програмного забезпечення.

В першу чергу необхідно порівняти існуючі стеки розробки фуллстек додатків. Існує багато популярних стеків для фуллстек розробки, де використовується JavaScript (або його варіації, такі як TypeScript) для розробки як клієнтської, так і серверної частин додатків. Ось декілька з найбільш популярних стеків [19]:

1. MERN Stack:

– MongoDB: Документ-орієнтована NoSQL база даних.

– Express.js: Мінімалістичний і швидкий backend фреймворк для Node.js.

– React: Популярна бібліотека для розробки клієнтського інтерфейсу.

– Node.js: Платформа для серверної розробки на JavaScript.

2. MEAN Stack:

– MongoDB: NoSQL база даних, схожа на MERN Stack.

– Express.js: Той самий швидкий backend фреймворк для Node.js.

– Angular: Розширений frontend фреймворк з великим спільноту і багатьма можливостями.

– Node.js: Популярна платформа для серверної розробки на JavaScript.

3. MEVN Stack:

– MongoDB: Знову ж таки, NoSQL база даних, як у MERN Stack та MEAN Stack.

– Express.js: Backend фреймворк для Node.js.

– Vue.js: Простий і легкий frontend фреймворк з активною спільнотою.

– Node.js: Платформа для серверної розробки на JavaScript.

Розглянувши дані стеки технологій для побудови клієнтської та серверної частини додатків, варто зазначити, що присутнє розподілення технологій відповідно до частини додатку. Використання різних засобів для побудови клієнтської та серверної частин додатку має кілька недоліків:

Складність інтеграції. Коли ви використовуєте різні технології для клієнтської та серверної частини, інтеграція між ними може бути важкою та часомісткою. Вам доведеться розробляти власні методи комунікації та забезпечувати сумісність даних.

Складність управління проєктом. Розробка та управління проєктом з використанням різних технологій може бути складнішою, оскільки потребує різних наборів навичок та інструментів.

Потреба в додатковому навчанні. Розробники повинні оволодіти двома окремими стеками для клієнтської та серверної частини, що може вимагати додаткового часу на навчання.

Складність утримання. Після завершення розробки проєкту, утримання та вдосконалення додатку може бути складнішим через використання різних технологій.

Насупроти цьому, фуллстек фреймворки, такі як Django (Python), Ruby on Rails (Ruby), Laravel (PHP), ASP.NET (C#) та Spring (Java), пропонують уніфіковане середовище для розробки як клієнтської, так і серверної частини додатку. Вони надають готові інструменти для швидкої розробки, інтеграції та управління проєктом, що робить їх привабливими для багатьох розробників. Такі фреймворки спрощують життя розробників і можуть прискорити час виготовлення та зменшити складність утримання додатків.

Повностековий фреймворк розробки - це набір програмного забезпечення, який забезпечує повне комплексне рішення для веб-розробки. Термін "повний стек" відноситься до фреймворку, що містить всі інструменти, необхідні для створення повноцінного веб-додатку, від інтерфейсу користувача до внутрішньої бази даних.

Хоча існує багато різних фреймворків повного стека, всі вони мають певні спільні риси, такі як веб-сервер, сервер додатків і база даних. Крім того, більшість фреймворків повного стека також включають різні інструменти для розробки, тестування та розгортання веб-додатків. Хоча повностекові фреймворки можна використовувати для розробки будь-якого типу веб-додатків, вони особливо добре підходять для складних додатків або тих, що вимагають високого ступеня інтеграції.

Саме такого виду фреймворк підходить під задачі розробки програмного забезпечення мережі спільнот з використанням штучного інтелекту. На підставі усього вищесказаного було обрано відповідний набір технологій.

Next.js 13: Next.js є потужним фреймворком для розробки веб-додатків, який надає зручні інструменти для побудови якісних та продуктивних додатків з використанням реактивного підходу [20].

Tailwind CSS: Tailwind CSS - це CSS-фреймворк, який дозволяє легко створювати стильовий дизайн інтерфейсів шляхом застосування класів безпосередньо в HTML-коді [21].

Next Auth: Next Auth - система аутентифікації (Auth) дозволяє забезпечити безпеку та автентичність користувачів платформи, забезпечуючи контроль над доступом до функціональності [22].

UploadThing: UploadThing - це платформа, яка дозволяє розробникам інтегрувати функцію завантаження зображень у свої веб-додатки. Це може допомогти спростити процес надання користувачам можливості завантажувати зображення у свої акаунти або профілі. Платформа пропонує такі функції, як перетягування, генерація ескізів зображень та перевірка формату файлів [23].

PlanetScale: PlanetScale - це MySQL-сумісна безсерверна база даних, яка забезпечує масштабування, продуктивність і надійність без шкоди для досвіду розробників. З PlanetScale ви отримуєте можливість горизонтального шардінгу, неблокуючих змін схем і багато інших потужних функцій бази даних без болю при їх впровадженні [24].

Prisma: Prisma - це ORM (Object-Relational Mapping) для баз даних, яке спрощує взаємодію з базою даних і дозволяє виконувати операції читання та запису з використанням стандартних мов програмування, таких як JavaScript [25].

MySQL: MySQL - це реляційна система управління базами даних, яка використовується для зберігання та організації даних, які використовуються в додатку [26].

JavaScript: JavaScript - це мова програмування, яка використовується для створення інтерактивних веб-додатків. Вона дозволяє виконувати клієнтський та серверний код та взаємодіяти з іншими технологіями для створення повнофункціонального програмного засобу [27].

Redis: Redis (Remote Dictionary Server) - це дуже популярна ін-меморі база даних з відкритим кодом, яка використовується для зберігання і управління даними. Вона найчастіше використовується як кеш, але також може використовуватися для інших цілей, таких як сесійне управління, реалізація черги повідомлень, аналітика і багато іншого [28].

OpenAI: OpenAI API - це набір послуг та інтерфейсів, наданий компанією OpenAI, дослідницькою організацією та компанією зі штучного інтелекту, які дозволяють розробникам інтегрувати та використовувати різні моделі та можливості штучного інтелекту у своїх додатках, продуктах або сервісах. OpenAI пропонує різноманітні моделі штучного інтелекту, які можуть виконувати завдання, такі як розуміння природної мови, генерація тексту, переклад та інше [29].

TensorFlow.js: TensorFlow.js - це бібліотека JavaScript з відкритим вихідним кодом, розроблена Google, яка дозволяє визначати, навчати та запускати моделі машинного навчання безпосередньо в браузері або на Node.js. Вона є частиною екосистеми TensorFlow, яка є популярним фреймворком для машинного навчання [30].

Вибір цих конкретних технологій був обгрунтований наступним чином:

Next.js 13 - це сучасний і популярний фреймворк для розробки веб-додатків, який дозволяє швидко та ефективно створювати високоякісний користувацький інтерфейс. Він включають в себе набір інструментів для роботи з компонентами, стейтом та роутингом, що спрощує розробку та підтримку додатків.

Tailwind CSS надає можливість створювати стильовий дизайн з використанням набору готових класів, що полегшує розробку та забезпечує консистентний вигляд інтерфейсу.

Використання системи аутентифікації (Auth) спрощує реалізацію безпеки та автентифікації користувачів, що є важливим для забезпечення доступу до спільноти та контролю за ним.

Prisma, як ORM, спрощує взаємодію з базою даних MySQL в поєднанні PlanetScale , надаючи зручний інтерфейс для операцій читання і запису даних.

Redis є потужним та високопродуктивним інструментом для зберігання та роботи з даними в реальному часі. Його швидкодія та розширюваність роблять його популярним великою кількістю веб-застосунків та систем обробки даних.

OpenAI API та TensorFlow.js надають доступ до моделей та можливостей штучного інтелекту для вирішення різноманітних завдань і створення інноваційних додатків та сервісів.

Дане API ідеально підходить для розробки нашого програмного забезпечення, адже вона надає можливість використовувати штучний інтелект для генерації тексту за допомоги GPT 3. GPT-3: Це одна з флагманських моделей OpenAI, яка означає "Generative Pre-trained Transformer 3". GPT-3 - це мовна модель, яка може розуміти та генерувати текст, подібний до людського. DALL-E: Модель DALL-E створена для генерації зображень на основі текстового опису. Вона може створювати унікальні зображення, які відповідають текстовим описам.

Використання JavaScript як основної мови програмування дозволяє побудувати повнофункціональний додаток з високою реактивністю та можливістю взаємодії з іншими сервісами та бібліотеками.

Усі ці технології були обрані через їхню сумісність, продуктивність та зручність для розробки додатків мережі спільнот.

Ці технології разом утворюють потужний стек для розробки мережі спільнот, який надає можливості для розширення функціональності, забезпечення високої продуктивності та створення користувацьких інтерфейсів з сучасним дизайном.

Отже, було надано перелік технологій за допомги яких реалізовано програмний застосунок, також обгрунтований їхній вибір та описано реалізацію.

3.2 Розробка користувацького інтерфейсу мережі спільнот

Процес розробки інтерфейсу мережі спільнот включав в себе кілька етапів, включаючи створення логотипів за допомогою Adobe Photoshop та розробку загального дизайну за допомогою Figma [31].

Створення логотипів було першим кроком в процесі розробки інтерфейсу [32]. Adobe Photoshop був обраний для цієї задачі через свою потужну функціональність та можливості роботи з растровою графікою.

Під час розробки логотипів було враховано багато аспектів, таких як брендовий стиль, кольорова палітра та символіка, які відображали суть мережі спільнот.

Photoshop дозволив створити професійні та якісні логотипи, забезпечивши різноманітність варіантів для вибору. Було розроблено логотип:

Рисунок 3.1 – Логотип

Після створення логотипів було вирішено розробити загальний дизайн мережі спільнот та інтерфейсу веб-додатка.

Figma був обраний для цього завдання через його зручний інтерфейс для колаборативної роботи, можливість створювати прототипи та дизайн-системи.

В Figma були створені макети сторінок, включаючи дизайн елементів інтерфейсу, розташування контенту, кольорову палітру та інші дизайн-рішення.

Усі подані рисунку знизу були реалізовані для додатку шляхом використання фреймворку Next.js та Tailwind CSS.

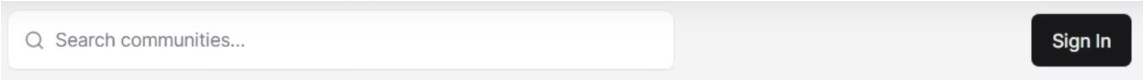

Рисунок 3.2 – Елементи застосунку “Navbar”

На даних рисунках зображено елементи додатку “Navbar” та“Post”. Перший відповідає за пошук форумів за ключовими словами, які вводить користувач. Також присутня кнопка логіну, яка дозволить користувачеві увійти до системи.

Рисунок 3.3 – Елементи застосунку “Post”

Компонент пост містить тег, за яким він доступний “r/uno”. Стрілку голосування, тобто кожен користувач буде мати змогу голосувати, чи корисна дана тема чи ні, тим самим роблячи тему популярнішою або навпаки.

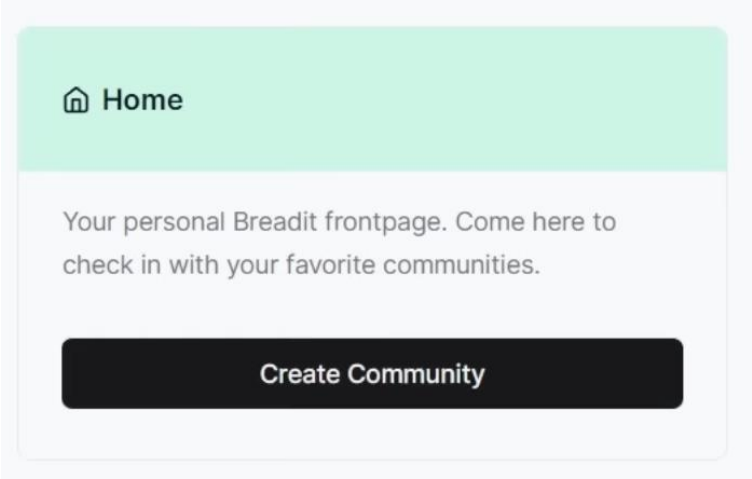

Рисунок 3.4 – Елементи застосунку “Suggestion”

Розглянемо два інші компоненти “Suggestion” та “Comment”. Перший відповідає за пропозицію користувачеів створити власний форум, куди будуть відноситись певні дискусії відповідно до тематики форуму. Також елемент містить стрічку відображєення поточною сторінки, у даному випадку це “Home”.

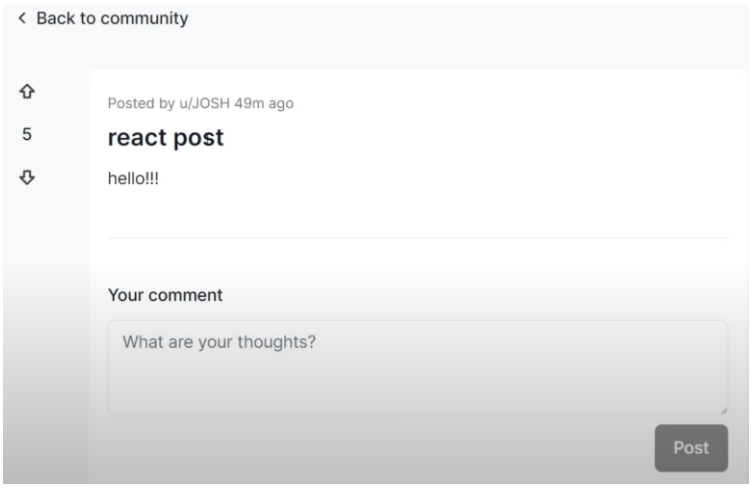

Рисунок 3.5 – Елементи застосунку “Comment”

Елемент “Comment” є певний інпут, в який інтегровано ШІ. Починаючи писати текст, користувач може відкрити модальне вікно та запитати допомоги у ШІ для генерації тексту чи фото.

Прийнявши запропоновану інформацію ШІ, користувач може вирішити чи постити її.

В цілому поєднавши згаді вище компоненти було намальовано дизайн головної сторінки додатку. На сторінці користувач має можливість перейти до свого профілю для його налаштування.

Також користувачі мають змогу переглядати увесь наявний контент за підписками на спільноти, а також взаємодія з ними за допомоги голосування та коментування. Інтерфейс головної сторінки додатку зображено на рисунку 3.6.

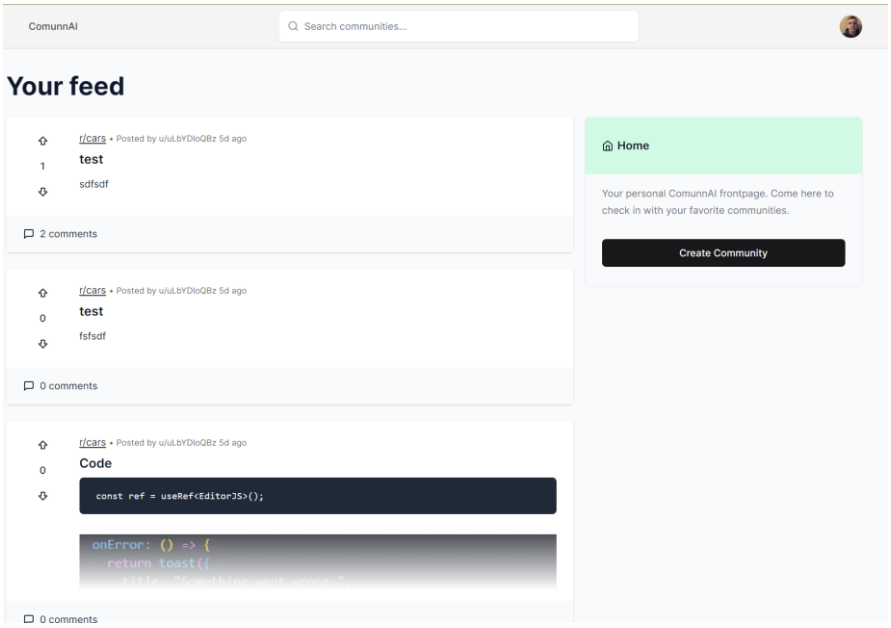

Рисунок 3.6 – Елементи застосунку “Home page”

Важливим елементом додатку є меню, де користувач бачить вказаний собою при реєстрації поштовий адрес, а також навігацію до ленти новин, сторінки створення спільнот, а також налаштування.

Також користувач має можливість вийти зі свого облікового запису нажавши на відповідний пункт в меню. Інтерфейс даного меню зображено на рисунку 3.7.

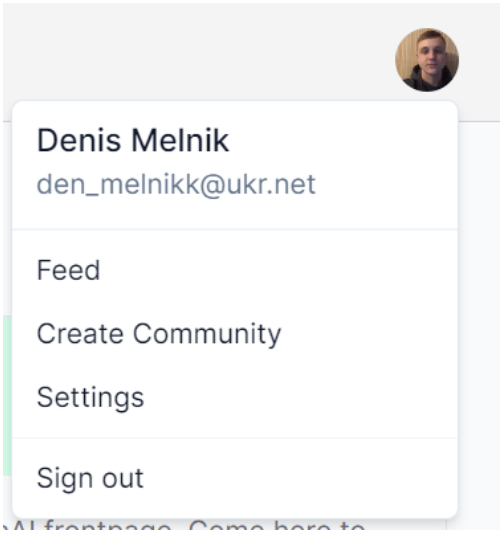

Рисунок 3.7 – Елементи застосунку “Menu”

Одним з ключовим компонентів додатку є інтерактивне поле для створення постів з набором різноманітних інструментів для зручного редагування. Інтерфейс даного меню зображено на рисунку 3.8.

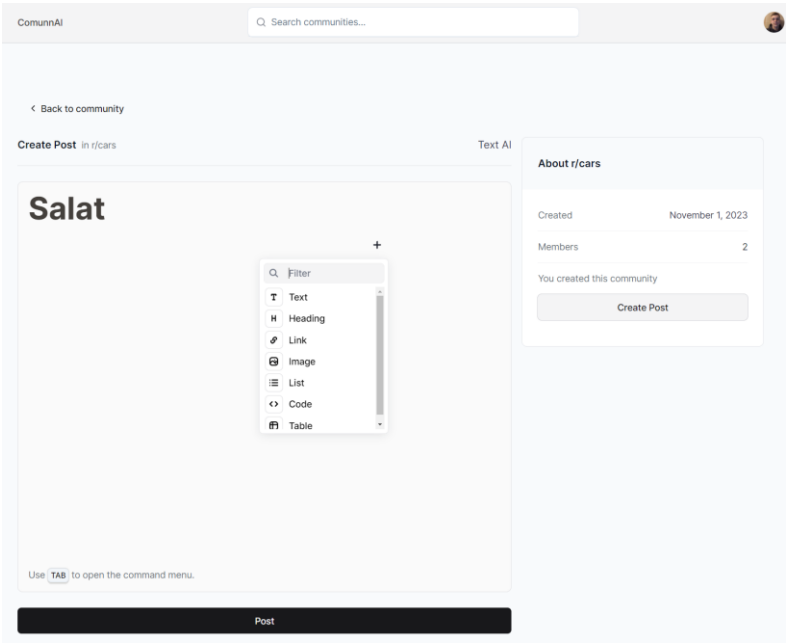

Рисунок 3.8 – Елементи застосунку “Edit area”

Редагуючи текст та нажавши на табуляцію, користувач побачить вікно з певними опціями та набором інструментів для швидкого та комфортного редагування коненту.

Також у додатку присутнє модальне вікно для виклику асистенту з написання тексту до постів, який реалізовано за допомоги штучного інтелекту.

Інтерфейс даного модального вікна зображено на рисунку 3.9.

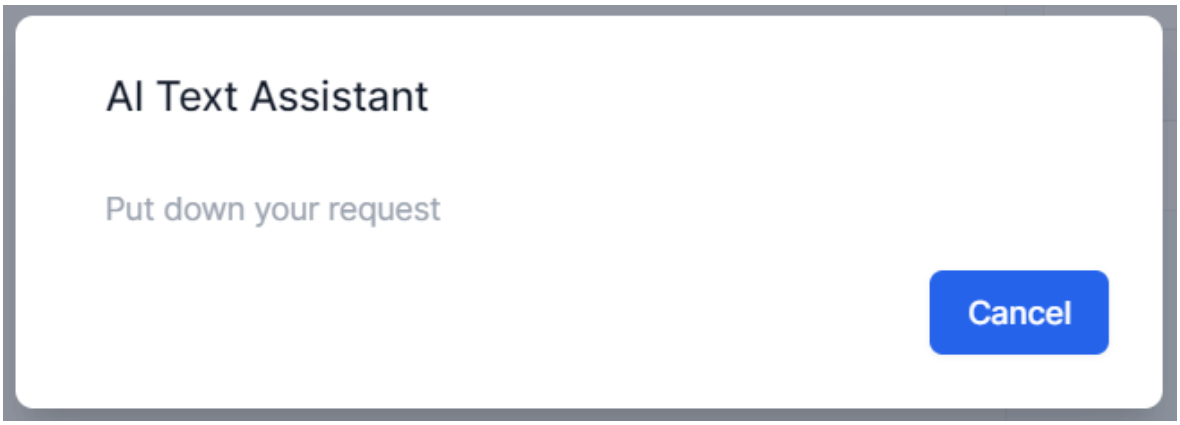

Рисунок 3.9 – Елементи застосунку “AI Text Assistant”

Adobe Photoshop був обраний для створення логотипів через його потужність у маніпулюванні графікою та можливість створення растрових зображень в високій якості.

Figma був обраний для розробки дизайну та інтерфейсу через його онлайн-платформу, що дозволило спільно працювати над проектом, а також через його можливості для створення прототипів та легкої інтеграції з іншими інструментами для розробки.

В цілому, обрані технології відображають потреби проекту, надавали можливості для якісної розробки інтерфейсу та логотипів, і сприяли ефективній спільній роботі команди.

Отже, було обрано певні засоби такі, як Figma та Adobe Photoshop для створення логотипу та побудови інтерфейсу користувача. Розроблено прототипи логотипу відповідно до назви додатку та можливої кольорової палітри. Розроблено певні компоненти інтерфейсу відповідно до завдання та надано певні прототипи реалізації.

3.3 Розробка серверної частини мережі спільнот

Щоб створити додаток, в якому потрібні дані і ресурси, що надаються і зберігаються зовнішнім джерелом, знадобиться API для запиту цих ресурсів [33]. По суті, потрібні два різних додатки, які взаємодіють один з одним для обміну даними.

У сучасній веб-розробці це зазвичай робиться шляхом створення двох різних додатків (припускаючи, що ви володієте додатком, який зберігає дані), клієнтського додатку, який працює в браузері, і серверного додатку, який працює на сервері.

Але так, як для розробки додатку було обрано Next.js фреймворк – є можливість написання і клієнтської і серверної частини в одному репозиторію за допомоги Next.js API Routes, що відповідає за серверну частину додатку.

По суті, Next.js API Routes усувають необхідність у створенні додаткового бекенд-сервера у ваших повностекових веб-додатках. За допомогою Next.js API Routes є можливість отримувати доступ до даних у базі даних або зберігати їх так само, як якщо б використовувати окремий бекенд-додаток.

Створення маршрутів API в Next.js схоже на створення маршрутів сторінок в Next.js.

Єдина відмінність полягає в тому, що ці маршрути API створюються в папці api, яка знаходиться в папці pages вашого додатку, і будь-який файл, знайдений в папці pages/api, буде розглядатися як кінцева точка API.

Тобто, якщо у є файл з ім'ям example.js у папці pages/api, є змога отримати доступ до маршруту API у коді, зробивши запит до /api/example.

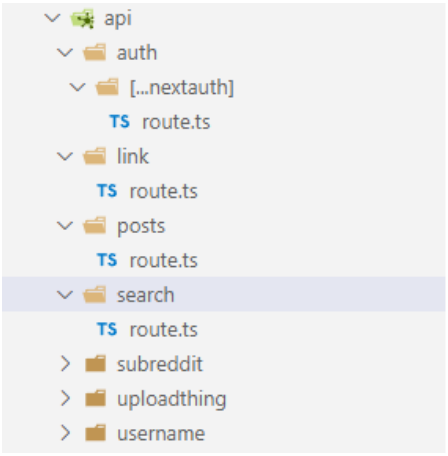

Рисунок 3.10 – Файлова структура серверної частини додатку

Маршрути API в Next.js також можуть бути динамічними. Це схоже на те, як працюють звичайні динамічні сторінки в Next.js. Динамічні маршрути API дозволяють нам надсилати відповідь на різні запити до кінцевої точки API. Для кращого розуміння маршрутів API в Next.js побудуємо табличку, де буде описано декілька з них.

Таблиця 3.1 – таблиця маршрутів API мережі спільнот

Файлова структура	Кінцевий маршрут	Функціонал
/api/ai/text/route.ts	/api/ai/text	Генерація тексту з використанням штучного інтелекту
/api/subreddit/post/comment/vote/route.ts	/api/subreddit/post/comment/vote	Голосування щодо публікації користувачів
/api/auth/[…nextAuth]/route.ts	/api/auth	Аутентифікація користувача
/api/subreddit/subscire	/api/subreddit/subscribe	Підписка на спільноту

Продовження таблиці 3.1

/api/subreddit/unsubscire	/api/subreddit/unsubscribe	
/api/posts	/api/posts?limit=${INFINITE_SCROLL_PAGINATION_RESULTS}&page=${pageParam}	Динамічний маршрут автоматичного завантаження публікацій

Для розуміння як пишуться модулі, що відповідають за серверну частину додатку, розглянемо один з них на рисунку 3.11.

```
export async function POST(req: Request) {
  try {
    const body = await req.json()

    const { title, content, subredditId } = PostValidator.parse(body)

    const session = await getAuthSession()

    if (!session?.user) {
      return new Response('Unauthorized', { status: 401 })
    }

    // verify user is subscribed to passed subreddit id
    const subscription = await db.subscription.findFirst({···
    })

    if (!subscription) {
      return new Response('Subscribe to post', { status: 403 })
    }

    await db.post.create({···
    })

    return new Response('OK')
  } catch (error) {···
  }
}
```

Рисунок 3.11 – Модуль створення публікацій

Детально розберемо модуль та його функціонал.

Кожен маршрут API або кінцева точка повинна експортувати функцію за замовчуванням, яка обробляє запити, зроблені до цієї кінцевої точки для надання можливості клієнтській частині додатку встановити зв’язок з даним маршрутом.

Функція отримує два параметри:

– req: екземпляр http.IncomingMessage, який також включає деякі вбудовані помічники - або проміжне програмне забезпечення - такі як req.body, req.query і так далі, для розбору вхідного запиту;

– res: екземпляр http.ServerResponse, який включає деякі Express.js-подібні допоміжні методи, такі як res.send, res.body і так далі.

Спочатку отримуємо дані від клієнту і на місці валідуємо їх. У випадку невідповідності формату даних – відсилаємо помилку на клієнт.

Далі, помістивши код у блок try{}, catch(e){}, для того щоб відловити помилку у випадку невдалого запиту, виконуємо асинхронні запити до бази даних.

Перш за все провіряємо, чи користувач увійшов до системи, якщо ні – повідомляємо йому про необхідність логінізації.

Якщо користувач увійшов до системи, ми беремо ідентифікаційний номер спільноти та робимо запит до бази даних для отримання колекції відповідних постів.

Якщо користувач намагається розмістити публікацію не підписавшись на спільноту – він отримає повідомлення про необхідність підписки для надання змоги публікації власних постів.

У випадку підписки користувача на відповідну спільноту – йому надається право створити пост, після чого він побачить відповідне повідомлення про вдале створення публікації.

Далі розглянемо ключовий модуль додатку, а саме генерацію тексту з використанням штучного інтелекту.

Щоб інтегрувати ChatGPT у додаток Next.js, потрібно використовувати API OpenAI, який надає програмний інтерфейс для взаємодії з моделлю ChatGPT. Спочатку потрібно отримати API-ключ OpenAI, а потім використовувати клієнтську API-бібліотеку, наприклад npm-пакет openai, для виконання запитів до API з коду Next.js. Модуль зображено на рисунку 3.12.

```
export async function POST(req: Request) {
  const { messages } = await req.json();

  const parsedMessage = MessageArraySchema.parse(messages);

  let options = {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
      Authorization:
        'Bearer ${process.env.OPENAI_API_KEY}',
    },
    body: JSON.stringify({
      model: "gpt-3.5-turbo",
      messages: [
        {
          role: "user",
          content: parsedMessage,
        },
      ],
    }),
  };

  const response = await fetch(
    "https://api.openai.com/v1/chat/completions",
    options
  );
  const data = await response.json();

  return new Response(data.choices[0].message.content);
}
```

Рисунок 3.12 – Модуль генерації тексту з використанням ШІ

Після отримання API-ключа на сайті, його необхідно зберегти всередині проєкту. Для цього використовується файл “.env”. Файли з розширенням .env використовуються для зберігання конфіденційної інформації та налаштувань, які використовуються в програмному забезпеченні. Даний файл зображено на рисунку 3.13.

```
.env
1  DATABASE_URL='mysql://r2cw0rztm1hurh31act4:pscale_pw_A3TKuksnZNs1s7coiaFfuhe5vTfzdlT1d4bAECvuRoZ@aws.connect.psdb.cloud/cominniai?sslaccept=strict'
2  NEXTAUTH_SECRET=mysecret
3
4  GOOGLE_CLIENT_ID=109409043932-q3jps12hnh3vabtt35hsco0j7ept02uf.apps.googleusercontent.com
5  GOOGLE_CLIENT_SECRET=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
6
7  UPLOADTHING_SECRET=sk_live_9fa127a1188958ff189a539af4e756d023e8c0e6494e22e41d796b141fbbfb4a
8  UPLOADTHING_APP_ID=sg2xtn9c9b
9
10 REDIS_URL=https://eu2-maximum-crayfish-30297.upstash.io
11 REDIS_SECRET=AXZZACQgMDdkNjYxMmYtNTEzMS00NjkwLWJmOGQtZWU0ODdlYzMxODA1YWJlMDhiMmQ5YWE2NDk2ZGE5OTQ2NzkzZGIxNWU5Zjc=
12
13 RAPID_API_KEY='099c532cdcmsh2c8c6cc75c4b605p1a7b57jsn51304c12edf6'
14 RAPID_API_HOST='ai-writer1.p.rapidapi.com'
15
16 OPENAI_API_KEY=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
```

Рисунок 3.13 – Файл .env з конфіденційною інформацією

Їх головною метою є зберігання конфіденційних даних, таких як паролі, ключі API, токени доступу та інша приватна інформація, яка не повинна зберігатися у відкритому вигляді у вашому коді або в репозиторії версійного контролю, такому як Git.

Далі, отримавши запит від користувача з клієнтської частини додатку, ми робимо парс повідомлення для подальшої роботи з ним за допомоги методу parse класу “ MessageArraySchema ”.

Після чого необхідно заповнити певні опції та дані для надання серверу нашого API. Для цього використовується об’єкт “options”. Всередину даного об’єкту необхідно помістити формат передачі даних, а також ключ. У тіло необхідно помістити версію API, у нашому випадку це "gpt-3.5-turbo", а також повідомлення, а саме запит користувача.

Потім необхідно надіслати наш запит з усіма вказаними даними та опціями до серверу і дочекатися відповіді, перетворивши її в необхідний формат та відіславши на клієнт для відображення користувачеві в такому вигляді:

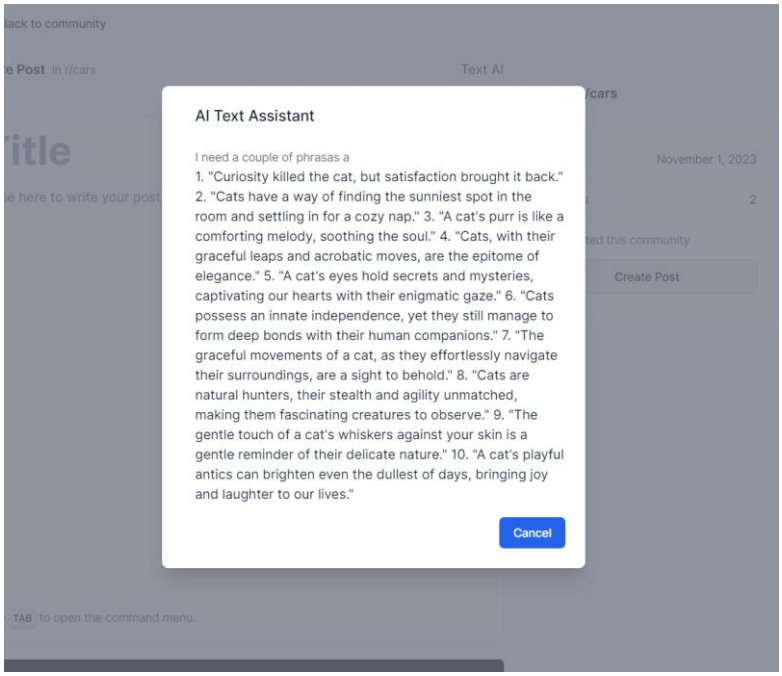

Рисунок 3.14 – Вдале відпрацювання генерації тексту та відображення відповіді користувачеві

Наступним кроком реалізуємо алгоритм визначення рейтингу. Алгоритм за замовчуванням, який називається гарячим рейтингом, реалізований так:

```
const { floor, log, max, abs } = Math;
const epoch = new Date(1970, 0, 1);
function epochSeconds(date) {
  const diff = date - epoch;
  const secondsInDay = 86400;
  const microsecondsToSeconds = 1e-6;
  return (
    floor(diff / 1000) * secondsInDay
    + (diff % 1000) * microsecondsToSeconds
  );
}
function score(ups, downs) {
  return ups - downs;
}
function hot(ups, downs, date) {
  const s = score(ups, downs);
  const order = log(max(abs(s), 1), 10);
  const sign = s > 0 ? 1 : s < 0 ? -1 : 0;
  const seconds = epochSeconds(date) - 1134028003;
  return round(sign * order + seconds / 45000, 7);
}
const ups = 10;
const downs = 5;
const date = new Date(2023, 11, 11);
const hotness = hot(ups, downs, date);
```

Рисунок 3.15 – Реалізацію алгоритму визначення рейтингу

Розглянемо функції окремо:

epochSeconds(date): Ця функція обчислює кількість секунд, що минули з початку "епохи" (1970, 1, 1) до заданої дати date. Вона враховує дні, секунди та мікросекунди.

score(ups, downs): Ця функція обчислює рейтинг публікації на основі кількості голосів "за" (ups) та "проти" (downs). Рейтинг просто обчислюється як різниця між голосами "за" та "проти".

hot(ups, downs, date): Ця функція обчислює гарячість публікації, використовуючи рейтинг публікації та дату публікації. Вона враховує важливість рейтингу (чим більше рейтинг, тим більше "гарячість") і час з моменту появи публікації.

s - рейтинг публікації.

order - логарифм рейтингу за основою 10 (значення має бути хоча б 1, щоб уникнути ділення на нуль).

sign - визначає, чи публікація має позитивний (+1), негативний (-1) чи нейтральний (0) рейтинг.

seconds - кількість секунд, які минули з моменту "епохи" до дати публікації.

Результат обчислення - вагова сума order і часу (в секундах), поділена на 45000.

Цей код використовується для індикації популярності публікації, де публікації з більшим рейтингом і більшим часом з моменту появи вгорі списку "гарячих" публікацій.

Для збереження даних було обрано PlanetScale - це платформа для керування реляційними базами даних, яка надає серверні послуги без необхідності самостійного адміністрування та підтримки MySQL-сумісних баз даних. Основним продуктом PlanetScale є публічна послуга PlanetScaleDB, яка пропонує можливість створювати та управляти серверними базами даних з підтримкою MySQL без необхідності власного обладнання або складного адміністрування.

Було розроблено таблиці бази даних де описі усі поля. Інтерфейс платформи з таблицями додатку зображено на рисунку 3.15.

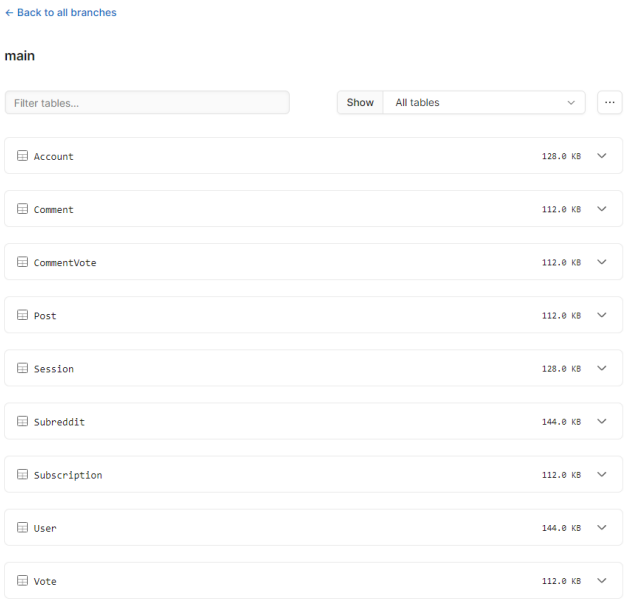

Рисунок 3.16 – таблиці бази даних мережі спільнот

Для змоги декларативно визначати схему бази даних за допомогою схеми та отримувати дані з PlanetScale з повною безпекою типів було використано Prisma.

Prisma - це ORM з відкритим вихідним кодом, яка легко інтегрується з PlanetScale і підтримує повний цикл розробки.

Використовуючи їх разом, ви отримуєте всі визнані переваги реляційних баз даних на додаток до сучасного досвіду розробника, безпечних запитів, нульових операцій та нескінченного масштабування.

Схема Prisma використовує мову моделювання Prisma для визначення схеми вашої бази даних. Це робить моделювання даних простим та інтуїтивно зрозумілим, особливо коли мова йде про моделювання відношень.

Синтаксис схеми Prisma значною мірою натхненний GraphQL SDL. Для прикладу розглянемо одну зі схем бази даних на рисунку 3.16.

```
model User {
  id                String         @id @default(cuid())
  name              String?
  email             String?        @unique
  emailVerified     DateTime?
  createdSubreddits Subreddit[]    @relation("CreatedBy")
  subscriptions     Subscription[]
  votes             Vote[]

  username String? @unique

  image       String?
  accounts    Account[]
  sessions    Session[]
  Post        Post[]
  Comment     Comment[]
  CommentVote CommentVote[]
}
```

Рисунок 3.17 – схеми даних User

Наданий код - це схема моделі користувача (User) в системі баз даних, яка використовує ORM-фреймворк Prisma для взаємодії з базою даних. Основні елементи цієї схеми визначають властивості та зв'язки користувачів у вашому додатку. Давайте розглянемо ключові частини схеми:

1. ”id”: Унікальний ідентифікатор користувача, який використовується як первинний ключ. Властивість має тип String, і вона автоматично генерується за допомогою cuid(), якщо значення не вказане.

2. ” name”: Ім'я користувача (String).

3. ”email”: Електронна пошта користувача, позначена анотацією @unique, що робить її унікальною для кожного користувача.

4. ” emailVerified”: Дата та час підтвердження адреси електронної пошти (DateTime).

6. “subscriptions”: Список підписок користувача на різні сабреддіти (категорії).

7. ”votes”: Список голосів користувача, які він залишив на постах і коментарях.

8. “username”: Ім'я користувача, яке також позначено анотацією @unique.

9. “image”: Посилання на зображення або аватар користувача (String).

10. “accounts”: Зв'язок з іншими обліковими записами користувача (може бути корисним, якщо користувач має багато облікових записів).

11. “sessions”: Зв'язок з сеансами користувача.

12. “Post”: Список постів, які були створені користувачем.

13. “Comment”: Список коментарів, які були створені користувачем.

14. ”CommentVote”: Список голосів користувача, залишених на коментарях.

Ця схема дозволяє моделювати дані користувачів та їх взаємодію в системі, включаючи створення, підписки, голосування та інші дії, які користувачі можуть виконувати. Prisma використовує цю схему для генерації SQL-запитів та взаємодії з базою даних, спрощуючи розробку додатку.

Отже, у даному розділі було розібрано реалізацію основних елементів серверної частини додатку, наведено приклади та детально і покроково прокоментовано дії та моменти реалізації.

3.4 Розробка клієнтської частини мережі спільнот

Клієнтські компоненти дозволяють додати інтерактивність на стороні клієнта до додатку. У Next.js вони попередньо рендерингуються на сервері та гідратуються на клієнті. Директива "use client" - це конвенція, яка визначає межу між графом серверного та клієнтського компонентів [34].

При клієнтському рендерингу (CSR) браузер завантажує мінімальну HTML-сторінку та необхідний для неї JavaScript. Потім JavaScript використовується для оновлення DOM і рендерингу сторінки. Коли додаток вперше завантажується, користувач може помітити невелику затримку перед тим, як побачити повну сторінку, це відбувається тому, що сторінка не відображається повністю, поки весь JavaScript не буде завантажено, проаналізовано та виконано. Розберемо компонент “UseAuthForm” для відображення форми аутентифікації з використанням директиви “use client” та його функціонал на рисунку 3.18.

```
interface UserAuthFormProps extends React.HTMLAttributes<HTMLDivElement> {}
const UserAuthForm: FC<UserAuthFormProps> = ({ className, ...props }) => {
  const { toast } = useToast()
  const [isLoading, setIsLoading] = React.useState<boolean>(false)
  const loginWithGoogle = async () => {
    setIsLoading(true)
    try {
      await signIn('google')
    } catch (error) {
      toast({
        title: 'Error',
        description: 'There was an error logging in with Google',
        variant: 'destructive',
      })
    } finally {
      setIsLoading(false)
    }
  }
  return (
    <div className={cn('flex justify-center', className)} {...props}>
      <Button
        isLoading={isLoading}···
        disabled={isLoading}>
        {isLoading ? null : <Icons.google className='h-4 w-4 mr-2' />}
        Google
      </Button>
    </div>
  )
}
```

Рисунок 3.18 – компонент UseAuthForm

Наданий код представляє собою компонент Next.js, давайте розглянемо ключові частини цього компоненту:

1. Імпорти:

– cn: Це функція, яка імпортується з @/lib/utils. Вона використовується для генерації CSS-класів з допомогою бібліотеки classnames.

– signIn: Імпорт функції signIn з пакету next-auth/react. Вона використовується для виконання авторизації через Google за допомогою NextAuth.js.

– Інші імпорти включають компоненти, функції та інтерфейси, які використовуються в компоненті.

2. Оголошення інтерфейсу:

– Оголошення інтерфейсу UserAuthFormProps, який розширює стандартний HTMLAttributes<HTMLDivElement>. Це дозволяє передавати додаткові HTML-атрибути в компонент.

3. Оголошення компоненту:

– Оголошення компоненту UserAuthForm як функціонального компонента (Functional Component - FC). Він приймає UserAuthFormProps як аргумент та розширює його можливості, включаючи клас компоненту та інші HTML-атрибути.

4. Створення екземпляру useToast:

– За допомогою const { toast } = useToast() створюється екземпляр функції toast, яка використовується для відображення сповіщень на екрані.

5. Стан isLoading:

– За допомогою React.useState створюється стан isLoading, який визначає, чи триває завантаження.

6. Функція loginWithGoogle:

– Ця функція викликається при натисканні на кнопку "Google". Вона встановлює isLoading в true, намагається виконати авторизацію через Google за допомогою signIn('google'), обробляє помилки, якщо вони виникають, і встановлює isLoading в false після завершення дії.

7. Повернення JSX-коду:

– Компонент повертає JSX-код, який містить кнопку "Google" з іконкою Google, яка відображається, коли isLoading - false. Кнопка активує функцію loginWithGoogle, коли її натискають.

Цей компонент призначений для відображення кнопки авторизації через Google та відповідає за її логіку, включаючи відображення повідомлень про помилки за допомогою “useToast”.

Перейдемо до розробки компоненту, який відповідає за інтерактивне меню при натисканні на табуляцію, розроблене за допомоги бібліотеки Editor.js, що дозволяє користувачеві використовувати наявні опції при створення публікації.

```
const initializeEditor = useCallback(async () => {
  const EditorJS = (await import('@editorjs/editorjs')).default  212.7k (gzipped: 56.3k)
  const Header = (await import('@editorjs/header')).default  15.6k (gzipped: 5.7k)
  const Embed = (await import('@editorjs/embed')).default  20.5k (gzipped: 7.1k)
  const Table = (await import('@editorjs/table')).default  31k (gzipped: 9.7k)
  const List = (await import('@editorjs/list')).default  14k (gzipped: 5.2k)
  const Code = (await import('@editorjs/code')).default  9.2k (gzipped: 4k)
  const LinkTool = (await import('@editorjs/link')).default  46k (gzipped: 14.8k)
  const InlineCode = (await import('@editorjs/inline-code')).default  10.4k (gzipped: 4.4k)
  const ImageTool = (await import('@editorjs/image')).default  45.5k (gzipped: 14.2k)

  if (!ref.current) {
    const editor = new EditorJS({
      holder: 'editor',
      onReady() {
        ref.current = editor
      },
      placeholder: 'Type here to write your post...',
      inlineToolbar: true,
      data: { blocks: [] },
      tools: {···
      },
    })
  }
}, [])
```

Рисунок 3.19 – компонент Editor

Наданий код ініціалізує та налаштовує інтеграцію Editor.js в додатку. Editor.js - це бібліотека для створення та редагування вмісту на веб-сторінках. Давайте розглянемо ключові деталі цього коду:

1. const initializeEditor = useCallback(async () => {...}: Ця функція є функцією ініціалізації редактора і використовує useCallback для оптимізації. Вона буде викликана при першому рендері компоненту.

2. import збережених модулів: Код використовує динамічний import для завантаження необхідних модулів Editor.js та його розширень з зовнішніх пакетів.

3. Створення екземпляра EditorJS:

– Спочатку перевіряється наявність ref.current (посилання на поточний екземпляр редактора).

– За допомогою new EditorJS({ ... }) створюється новий екземпляр редактора з наступними налаштуваннями:

holder: Ідентифікатор DOM-елементу, де буде відображатися редактор.

onReady(): Функція зворотнього виклику, яка встановлює ref.current після того, як редактор готовий до використання.

placeholder: Текст-заглушка, який відображається у редакторі, коли немає контенту.

inlineToolbar: Увімкнення плаваючого панелі інструментів для редагування тексту.

data: Початковий вміст редактора, у цьому випадку, пустий об'єкт блоків.

tools: Налаштування доступних інструментів редактора, таких як заголовок, посилання, зображення тощо.

4. Налаштування окремих інструментів:

– Для інструментів, таких як linkTool, image, налаштовані конфігурації, які вказують шляхи до серверних точок для завантаження зображень та обробки посилань.

Цей код встановлює і налаштовує редактор Editor.js для створення та редагування вмісту. Editor.js надає зручний інтерфейс для створення структурованого контенту та підтримує різні типи блоків і інструментів для редагування.

Наступний важливий компонент розроблено для відображення публікацій, на які підписаний користувач та являє собою так звану стрічку публікацій, код якого зображено на рисунку 3.19.

```
const CustomFeed = async () => {
  const session = await getAuthSession()
  if (!session) return notFound()
  const followedCommunities = await db.subscription.findMany({
    where: {
      userId: session.user.id,
    },
    include: {
      subreddit: true,
    },
  })
  const posts = await db.post.findMany({
    where: {
      subreddit: {
        name: {
          in: followedCommunities.map((sub) => sub.subreddit.name),
        },
      },
    },
    orderBy: {···
    },
    include: {···
    },
    take: INFINITE_SCROLL_PAGINATION_RESULTS,
  })
  return <PostFeed initialPosts={posts} />
}
```

Рисунок 3.20 – компонент CustomFeed

Наданий код визначає компонент CustomFeed, який відповідає за відображення призначеного для користувача живого стріму публікацій із спільнот, на які він підписаний. Давайте розглянемо ключові деталі цього коду:

1. import імпортує необхідні залежності та модулі:

– INFINITE_SCROLL_PAGINATION_RESULTS: Імпорт константи, яка визначає кількість результатів, що відображаються під час безкінечної прокрутки.

– getAuthSession: Функція для отримання аутентифікованої сесії користувача.

– db та PostFeed: Імпорт бази даних та компоненту PostFeed.

– notFound з пакету next/navigation: Функція для відображення сторінки "Not Found".

2. CustomFeed - це асинхронна функція, яка виконує наступні дії:

– Отримує аутентифіковану сесію користувача за допомогою getAuthSession() та перевіряє наявність сесії .

– Отримує список спільнот, на які підписаний користувач, з допомогою запиту до бази даних. Ці спільноти зберігаються в followedCommunities.

– Отримує публікації з обраних спільнот за допомогою другого запиту до бази даних. Результати впорядковані за датою створення у зворотному порядку (нові публікації вгорі).

– Передає отримані публікації компоненту PostFeed у вигляді initialPosts.

3. PostFeed компонент призначений для відображення стріму публікацій та використовує initialPosts для відображення публікацій на сторінці.

Цей код відображає адаптовану стрім-сторінку з публікаціями для аутентифікованого користувача, який підписаний на певні спільноти, результат якого зображено на рисунку 3.21.

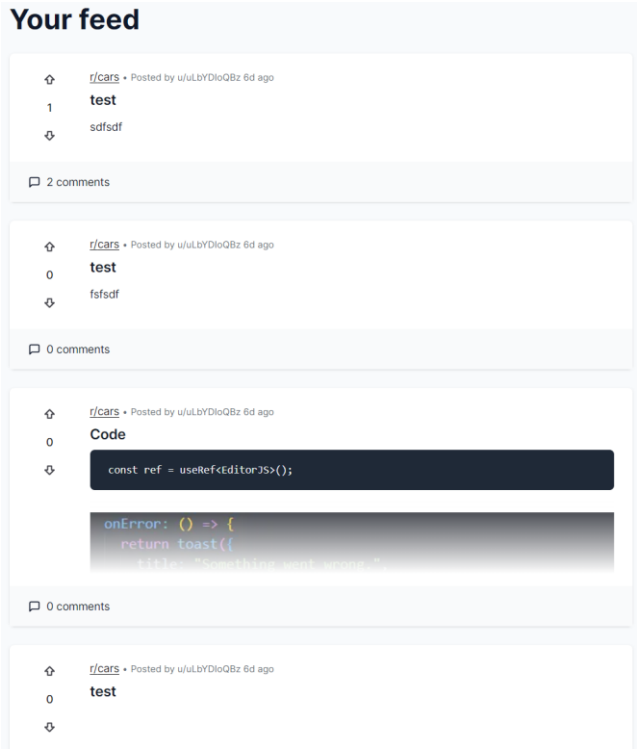

Рисунок 3.21 – адаптована сторінка з піблікаціями

Далі розглянемо ключовий компонент додатку “Post”, який є ключовою сутністю додатку. Та використовую згадані вище компоненти інтерактивного меню та асистента з використанням ШІ.

```
'use client'
type PartialVote = Pick<Vote, 'type'>
interface PostProps {
  post: Post & {
    author: User
    votes: Vote[]
  }
  votesAmt: number
  subredditName: string
  currentVote?: PartialVote
  commentAmt: number
}
const Post: FC<PostProps> = ({
  post,
  votesAmt: _votesAmt,
  currentVote: _currentVote,
  subredditName,
  commentAmt,
}) => {
  const pRef = useRef<HTMLParagraphElement>(null)
  return (
    <div className='rounded-md bg-white shadow'>
      <div className='px-6 py-4 flex justify-between'>···
      </div>

      <div className='bg-gray-50 z-20 text-sm px-4 py-4 sm:px-6'>···
      </div>
    </div>
  )
}
export default Post
```

Рисунок 3.22 – Компонент публікації

Наданий код представляє компонент Next.js з назвою Post, який відповідає за відображення окремого посту на сторінці. Давайте розглянемо ключові деталі цього коду:

1. Імпорти та Типи:

– PartialVote - це тип даних, який представляє частковий об'єкт голосування Vote з вибором поля type.

– PostProps - це інтерфейс, який описує властивості, що передаються у компонент Post. Властивості включають об'єкт post, кількість голосів (votesAmt), назву спільноти (subredditName), частковий об'єкт голосування (currentVote) і кількість коментарів (commentAmt).

2. Відображення поста:

– Внутрішність компонента Post містить відображення окремого поста. Це включає в себе інформацію про автора, кількість голосів, назву спільноти, час створення, назву та вміст поста.

3. Голосування за постом:

– Для голосування за постом використовується компонент PostVoteClient, якому передаються postId, initialVotesAmt та initialVote.

4. Відображення інформації про пост:

– Інформація про пост включає в себе назву спільноти, автора поста та час створення.

5. Вміст поста:

– Вміст поста відображається за допомогою компонента EditorOutput, який відображає текст та інші блоки, які можуть бути частиною поста. Якщо вміст поста занадто довгий, він може бути вирізаний знизу, і встановлюється спеціальний ефект розмиття.

6. Кількість коментарів:

– В компоненті відображається кількість коментарів, і це виводиться разом з посиланням на сторінку поста.

Отже, у даному розділі було розібрано реалізацію основних елементів клієнтської частини додатку, наведено приклади та детально прокоментовано дії та моменти реалізації.

3.5 Висновки розділу 3

Отже, у розділі було надано детальний огляд реалізації програмного застосунку, включаючи вибір та використання різних технологій. Для створення логотипу та інтерфейсу користувача були використані засоби, такі як Figma та Adobe Photoshop, а також були розроблені прототипи для дизайну та інтерфейсу.

У розділах про серверну та клієнтську частину було ретельно розглянуто реалізацію основних елементів програми, надано приклади коду та надано докладні пояснення щодо кожного кроку та дії.

Загальний висновок полягає в тому, що програмний застосунок був успішно розроблений та реалізований з використанням відповідних технологій та інструментів для дизайну та розробки як серверної, так і клієнтської частини.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Аналіз методів та засобів тестування

Тестування програмного забезпечення важливе з кількох причин. Воно допомагає виявити та виправити помилки перед випуском продукту, що поліпшує його якість і задоволення користувачів. Також тестування забезпечує стабільну роботу програми в різних умовах і виявляє потенційні проблеми безпеки.

Важливо також переконатися, що програмне забезпечення відповідає встановленим вимогам і функціям, щоб задовольнити очікування користувачів і бізнесу. Тестування також допомагає оптимізувати використання ресурсів, таких як пам'ять і процесор, забезпечуючи ефективність програмного продукту. Загалом, воно грає важливу роль у забезпеченні успішного випуску програмного продукту на ринок [35].

Розглянемо три основних метода тестування, які можуть підійти для мережі спільнот:

1. Тестування "чорного ящика":

Тестування "чорного ящика" - це тип тестування програмного забезпечення, при якому функціональність програмного забезпечення не відома. Тестування проводиться без внутрішніх знань про продукти. Його також називають функціональним тестуванням. Тестування "чорного ящика" фокусується на зовнішніх атрибутах і поведінці програмного забезпечення. Цей тип тестування розглядає очікувану поведінку програмного забезпечення програми з точки зору користувача.

2. Тестування білого ящика:

Тестування білої скриньки або тестування скляної скриньки - це техніка тестування програмного забезпечення, яка тестує програмне забезпечення, використовуючи знання внутрішніх структур даних, фізичного логічного потоку та архітектури на рівні вихідного коду. Це тестування працює, дивлячись на

тестування з точки зору розробника. Це тестування також відоме як тестування "скляної коробки", тестування "чистої коробки", структурне тестування або нефункціональне тестування.

3. Тестування сірого ящика:

Тестування сірого ящика - це поєднання техніки тестування чорного ящика і техніки тестування білого ящика в тестуванні програмного забезпечення. Тестування сірого ящика включає в себе входи і виходи програми для цілей тестування, але дизайн тесту перевіряється за допомогою інформації про код. Тестування сірого ящика добре підходить для тестування веб-додатків, оскільки воно враховує високорівневе середовище проектування та умови сумісності.

Давайте розглянемо відмінності між ними на таблиці 4.1.

Таблиця 4.1. – розбіжності між методами тестування

№	Тестування чорного ящика	Тестування сірого ящика	Тестування білого ящика
1.	Тестування має низьку деталізацію.	Тестування має середній рівень деталізації.	Тестування має високий рівень деталізації.
2.	Роблять кінцеві користувачі, а також тестувальники та розробники.	Виконується кінцевими користувачами (називається тестуванням прийнятності користувача), а також тестувальниками та розробниками.	Зазвичай це роблять тестувальники та розробники.
3.	Не потрібно знати внутрішні дані.	Відомі внутрішні елементи, що стосуються тестування.	Відомий внутрішній код програми та бази даних.

Продовження таблиці 4.1

4.	Він буде менш вичерпним, ніж два інших.	Це щось посередині.	Найбільш вичерпний серед усіх трьох.
5.	Засновано на вимогах і тестових випадках на функціональних специфікаціях, оскільки внутрішні елементи невідомі.	Забезпечує кращу різноманітність/глибину тестових випадків завдяки високому рівню знань внутрішніх органів.	Може використовувати код із різними даними.
6.	Якщо тестування алгоритму не підходить для цього найкраще.	Якщо використовується тестування алгоритму, то для цього також не підходить найкраще.	Якщо тестування алгоритму є, то воно найкраще для цього підходить.
7.	Підходить для функціонального або бізнес-тестування.	Підходить для глибокого функціонального або бізнес-доменного тестування.	Використовується для всіх.
8.	Тестування Black Box забезпечує стійкість і захист від вірусних атак.	Тестування Grey Box не забезпечує стійкість і захист від вірусних атак.	Тестування White Box не забезпечує стійкість і безпеку проти вірусних атак.

У порівнянні трьох методів тестування програмного забезпечення, тестування сірого ящика видається найбільш збалансованим та важливим підходом. Воно поєднує переваги тестування чорного та білого ящика, надаючи середній рівень деталізації та дозволяючи використовувати як зовнішні, так і внутрішні аспекти програмного продукту.

Тестування сірого ящика виконується як кінцевими користувачами, так і тестувальниками з розробниками. Воно орієнтоване на функціональність та бізнес-тестування, забезпечуючи різноманітність тестових випадків. Також, воно

не вимагає повного знання внутрішніх деталей програмного забезпечення, що робить його ефективним та економічно затратним методом.

Порівняно з іншими методами, тестування сірого ящика може бути оптимальним вибором, забезпечуючи необхідний рівень деталізації та глибини тестування для забезпечення надійності та ефективності програмного продукту.

Для тестування буде розроблена декілька тест-кейсів, які покриють ключовий функціонал мережі спільнот.

Кейси тестування програмного забезпечення (test cases) – це документовані інструкції, які визначають кроки для виконання конкретного тесту, щоб перевірити, чи працює програмне забезпечення належним чином та відповідає вимогам [36].

Отже, розібравши методи тестування для програмного засобу мережі спільнот з інтегрованим штучним інтелектом, було надано перевагу методу сірого ящика.

4.2 Тестування клієнтської частини додатку

Тестування клієнтської частини грає важливу роль у забезпеченні якості програмного продукту та задоволення потреб користувачів, адже саме через клієнтську частину користувач взаємодіє з програмним продуктом. Якість цієї взаємодії безпосередньо впливає на загальний враження від використання додатку. Якщо інтерфейс користувача простий, зрозумілий та ефективний, а функціонал працює без завдання, це позитивно впливає на задоволення користувача.

Почнемо тестування з аутентифікації користувача перейшовши на головну сторінку додатку, та натиснувши на кнопку “Sign in” – відкриється модальне вікно з пропозицією зареєструватися, або ж, якщо користувач вже зареєстрований – увійти до системи.

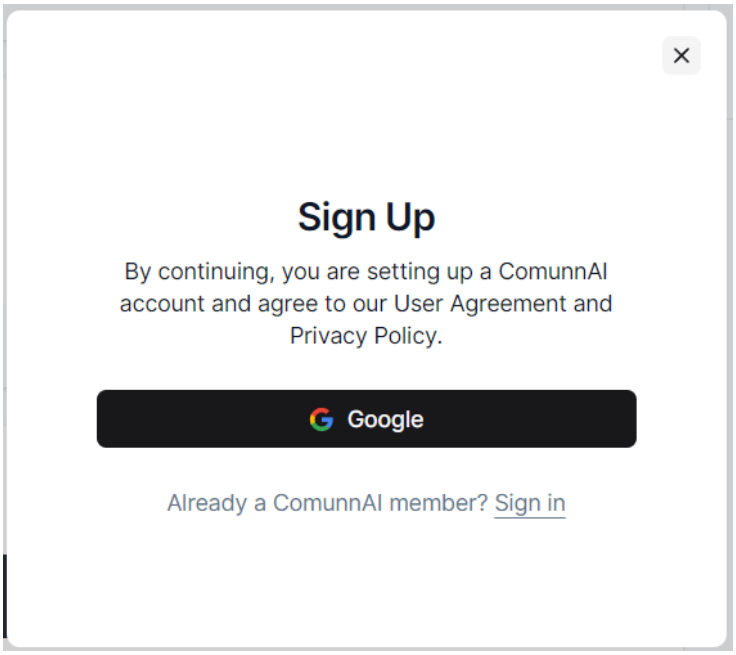

Рисунок 4.1 – Сторінка аутентифікації

Увійшовши до системи знайдемо цікаву нам спільноту скориставшись полем пошуку зверху головної сторінки.

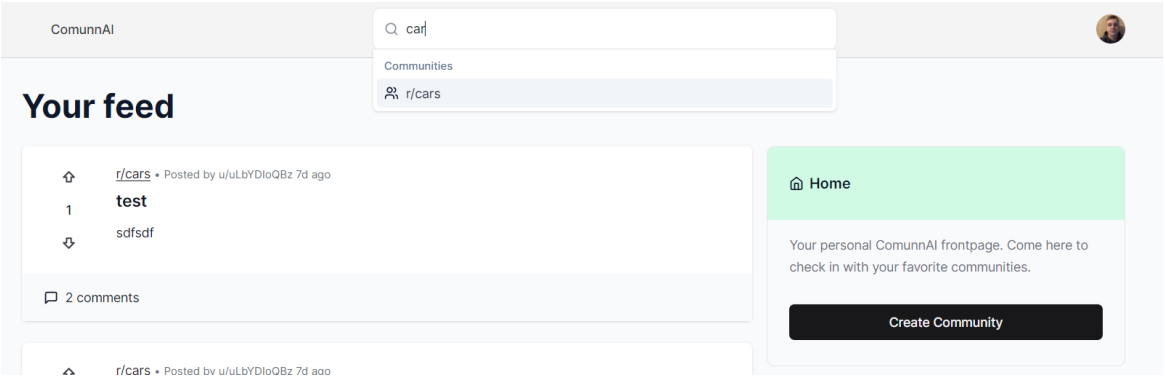

Рисунок 4.2 – Головна сторінка додатку

Як бачимо, під часу того, як користувач вписує назву спільноти, яка йому цікава, додаток пропонує найбільш відповідні варіанти запиту, у випадку тесту “cars” у відповідь на “car”.

Після виконання запиту відображають усі пости відповідно до обраної спільноти у такому вигляді, як зображено на рисунку 4.3.

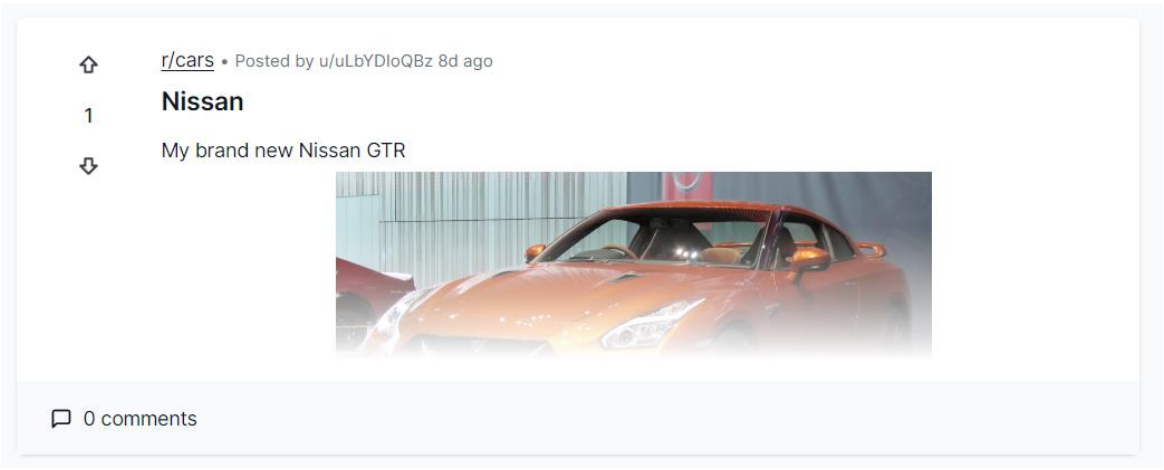

Рисунок 4.3 – Результат пошуку спільноти

Користувачеві надається змога передивлятись публікації різних користувачів відповідно до спільноти. Перейдемо до сторінки публікації натиснувши на неї.

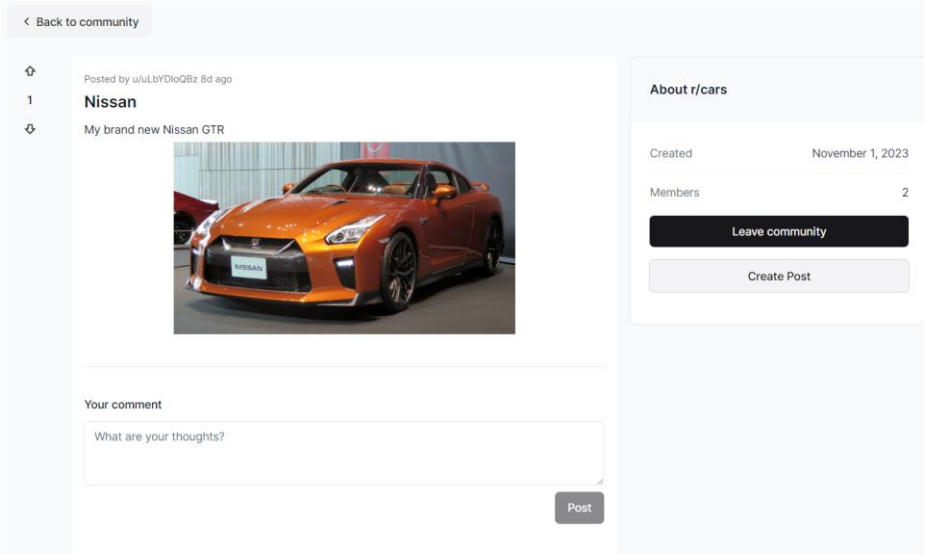

Рисунок 4.4 – Сторінка публікації

На даній сторінці користувач має змогу переглянути детальні дані щодо публікації, а саме – автора, дату публікації, коментарі, тощо.

Протестуємо можливість написання коментаря та реакції на публікацію.

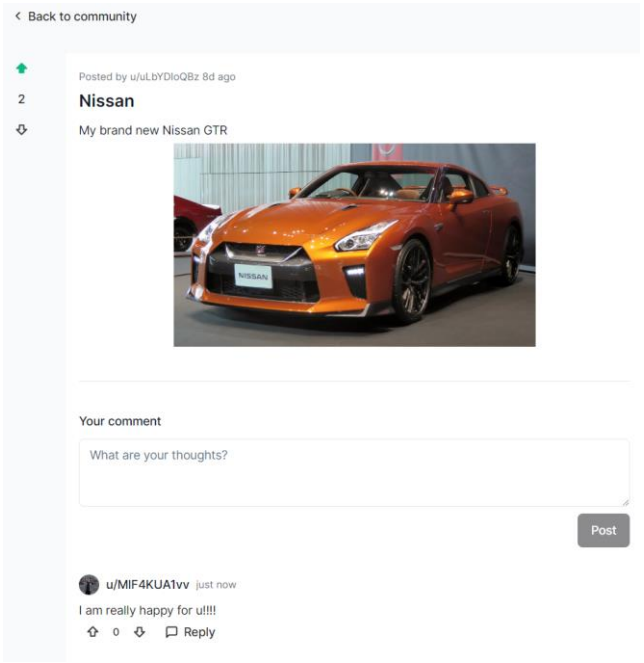

Рисунок 4.5 – Реакція та коментар до публікації

Як бачимо, користувач залишив коментар до публікації, а також відреагував на нього натиснувши на стрілку догори, яка отримала зелений колір та матиме вплив на рейтинг публікації.

Провіримо взаємодію між користувачами шляхом відповіді на коментарі, тим самим протестуємо реалізацію дерева коментарів.

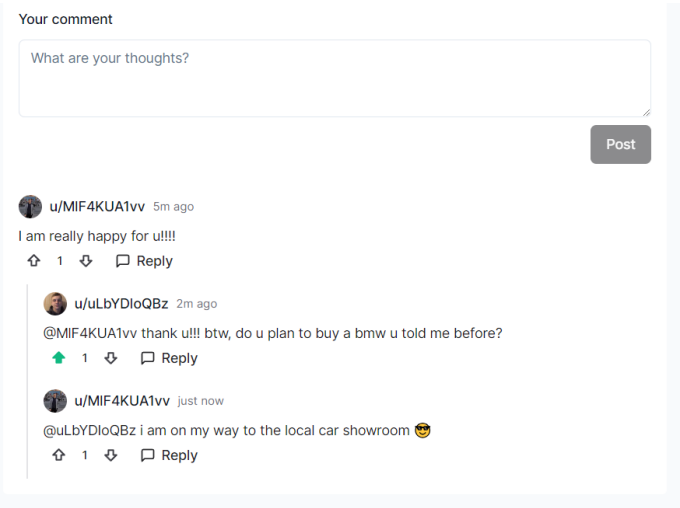

Рисунок 4.6 – Дерево коментарів та реакція на них

Після вдалого тестування взаємодії з постами інших користувачів перейдемо до тестування створення публікацій. Для цього необхідно перейди до бажаної спільноти та натиснути кнопку “Create Post”.

Таким чином користувач перейде до сторінки створення публікації де у нього буде певний набір інструментів для цього. Набір інструментів зображено на рисунку 4.7.

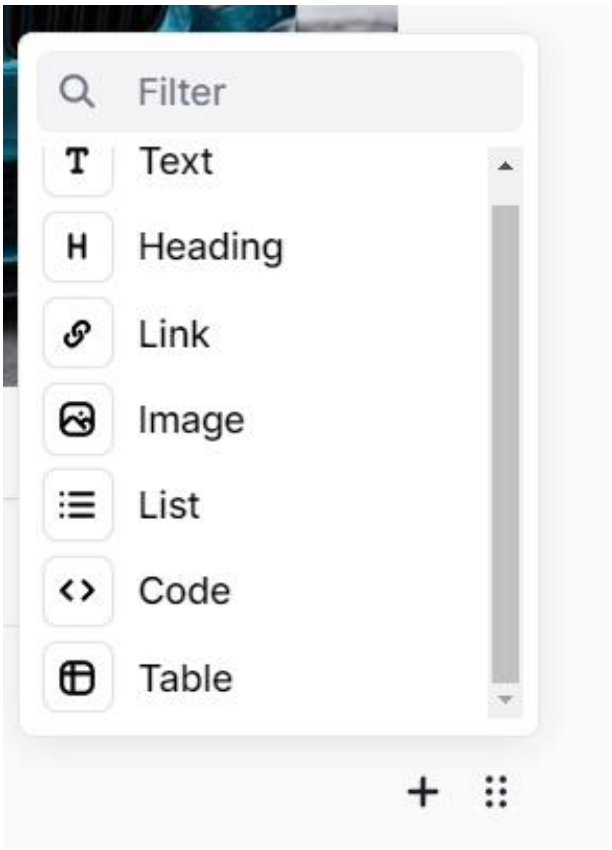

Рисунок 4.7 – Інтерфейс меню інструментів

Скориставшись підказкою знизу редактору публікацій, користувач, натиснувши на табуляцію, відкриває меню інструментів, де у нього є набір опцій щодо вставлення списків, коду, посилань, фото, тексту, заголовку та таблиці.

Розберемо усе детально дивлячись на рисунок 4.8.

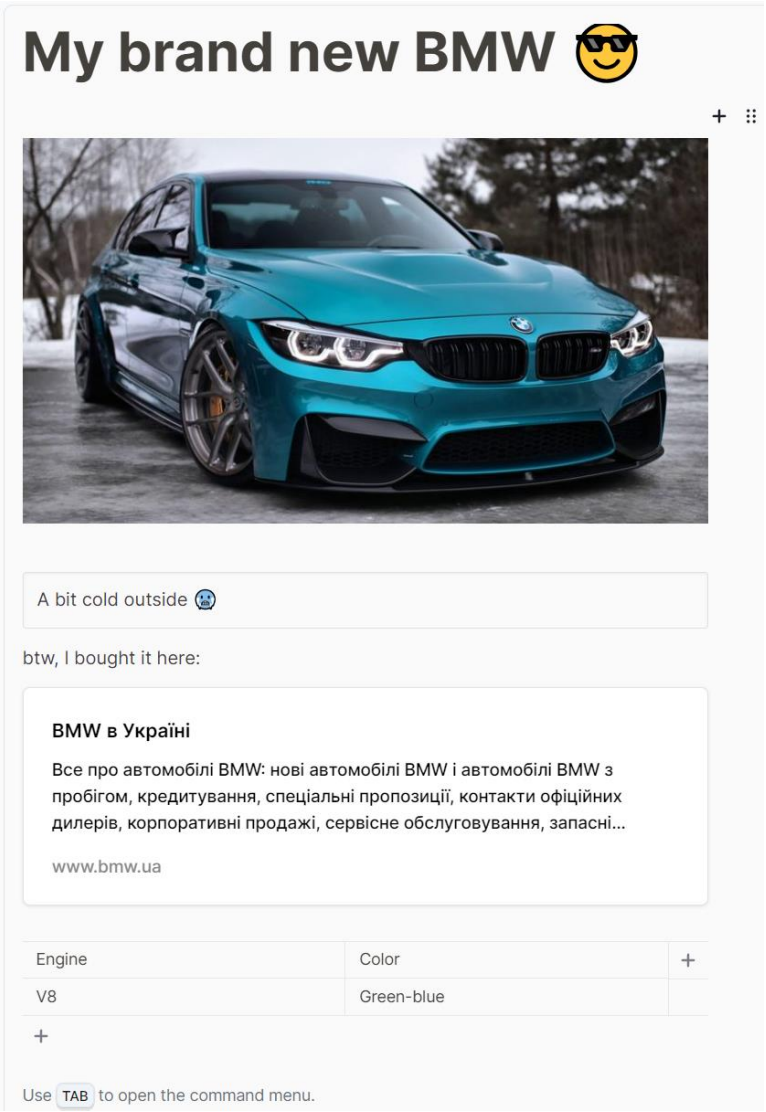

Рисунок 4.8 – Інтерфейс сторінки створення публікації

Створення публікації необхідно почати з заголовка, у даному випадку це “My brand new BMW”. Далі користувач обрав інструмент з меню команд для вставлення фото, після чого обравши фото завантажив його у редактор публікації. Знизу від фото користувач вписав короткий допис. Потім, обравши інструмент “Link” він додав посилання до публікації, а також з використанням інструменту “Table”, було додано табличку з певними даними до публікації.

Після заповнення публікації, її було опубліковано. Усі вище написані операції було виконано без помилок системи. Переглянемо наявність публікації в стрічці спільноти з іншого облікового запису.

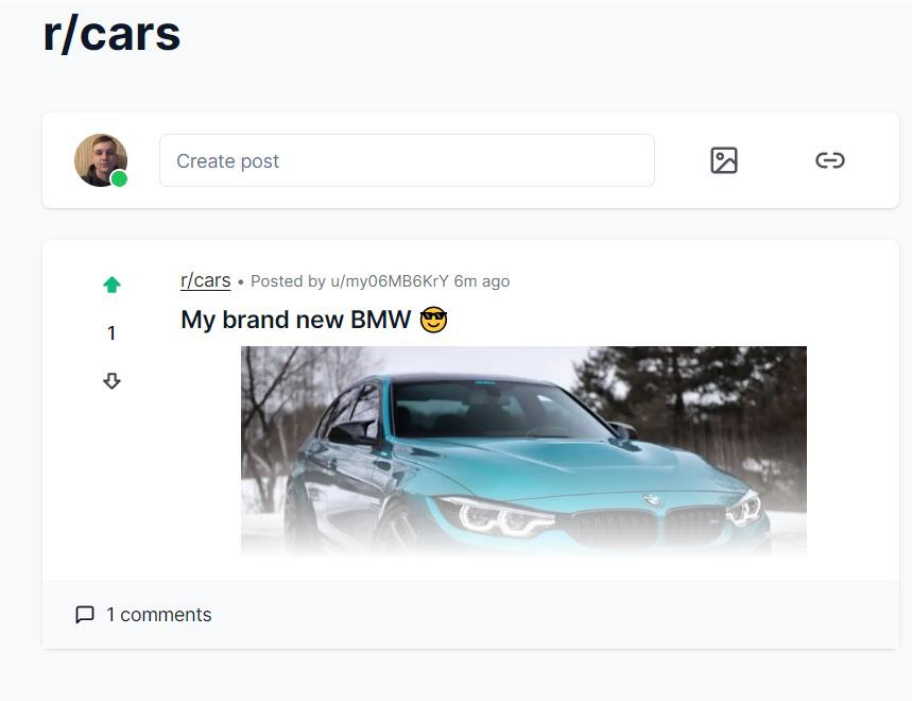

Рисунок 4.9 – Наявність створеної публікації в стрічці

Після вдало створеної публікації перейдемо до текстового асистента з інтегрованим штучним інтелектом. Для цього повернемося на сторінку створення публікації на натиснемо на відповідну кнопку у правому верхньому кутку. Після натискання відповідне модальне вікно з’явиться перед користувачем.

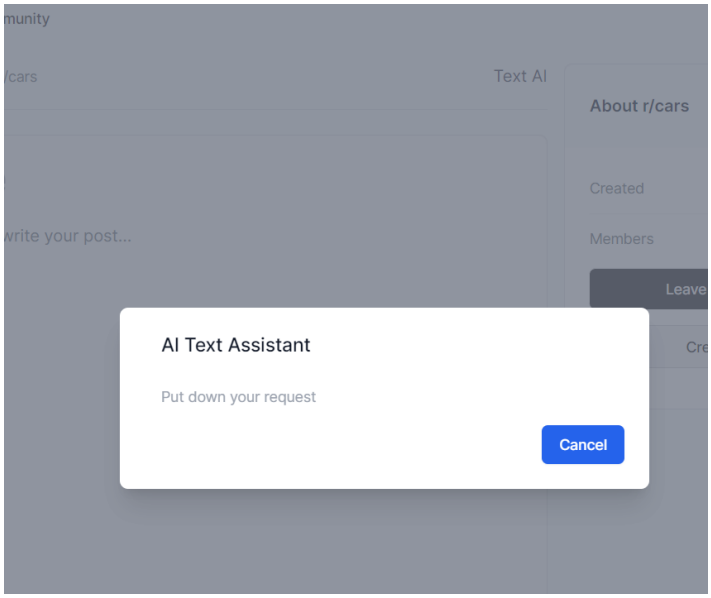

Рисунок 4.10 – Модальне вікно для текстового асистента

Впишемо наш запит до текстового поля на надішлемо запит натиснувши “Enter”. Перевіремо наявність відповіді згенерованої штучним інтелектом на рисунку 4.11.

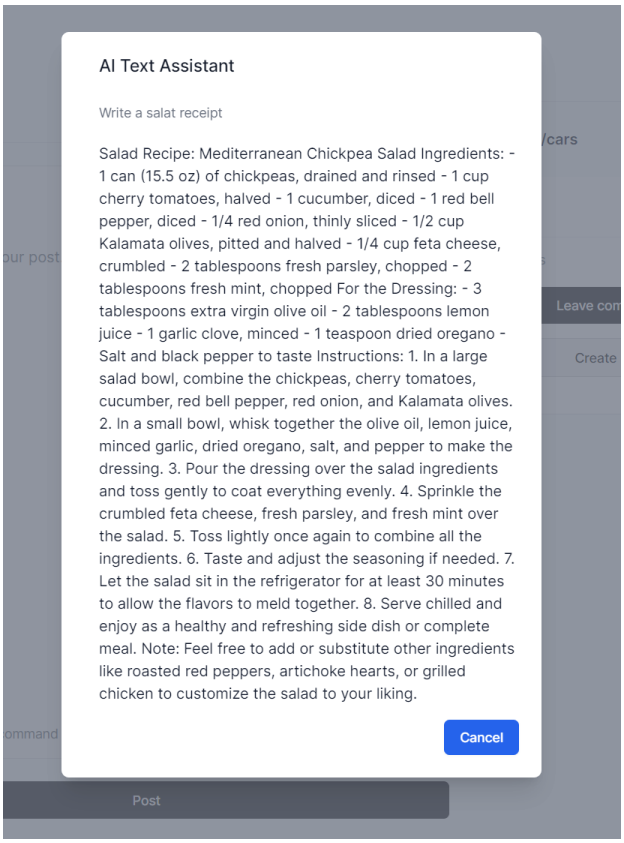

Рисунок 4.11 – Відповідь текстового асистента

Отримавши відповідь ми можемо скопіювати її та використати для нової публікації в подальшому.

Далі протестуємо функціонал, який відповідає за автоматизовану модерацію вмісту опублікованих публікацій користувачів. Для цього перейдемо на сторінку створення публікації та впише ненормативний текстовий контент після чого здійснимо публікацію.

Таким чином ми переконаємось чи відпрацьовує наша тренована модель штучного інтелекту та бібліотека, яка керується аналітикою моделі та слугує середньою ланкою між серверною частиною та модулем штучного інтелекту. Результат тестування зображено на рисунку 4.12.

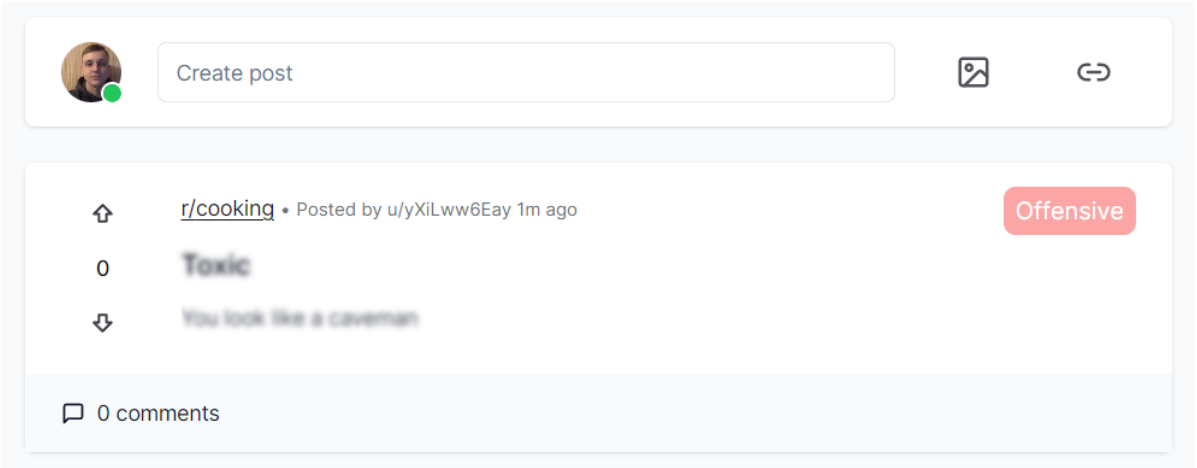

Рисунок 4.12 – Відображення класифікованого попередження щодо ненормативного контенту

Можемо бачити на рисунку вдале розпізнавання та класифікацію поданого в публікації контенту, що говорить про правильну та повну реалізацію поставленої задача даного модулю.

Протестуємо правильність роботи рекомендацій публікацій шляхом підписки на певні спільноти та запису коментарів, реакцією на пости. Таким чином ми переконаємось у правильності алгоритму визначення цільової аудиторії та коректної працездатності модулю.

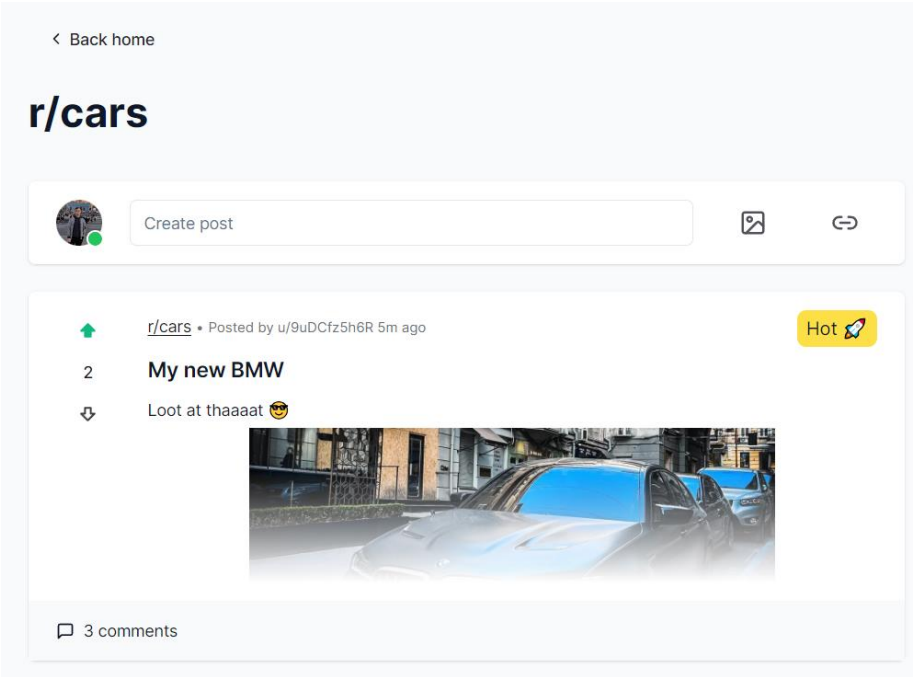

Рисунок 4.13 – Відображення рекомендованої публікації

На рисунку 4.13 видно, що після інтеракції користувачів з нещодавно опублікованим постом він отримав іконку, яка свідчить про його популярність. У свою чергу користувач підписаний на подібного роду спільноти, тому для нього відображення такої публікації буде доречна у його стрічці перегляду.

Отже, у даному розділі було протестовано клієнтську частину мережі спільнот з інтегрованим штучним інтелектом, наведено потік дій з перспективи користувача та протестовано їх без помилок системи.

4.3 Тестування серверної частини додатку

Тестування серверної частини додатку визнається ключовим етапом розробки програмного забезпечення, оскільки воно впливає на низку критичних аспектів та має величезне значення для успіху та надійності програмного продукту.

Почнемо тестування з перевірки створення таблиць бази даних за допомоги PlanetScale та Prisma. Запустимо скрипт з наявними схемами Prisma виконавши команду “npx prisma db push” та перевіремо наявність таблиць на платформі PlanetScale.

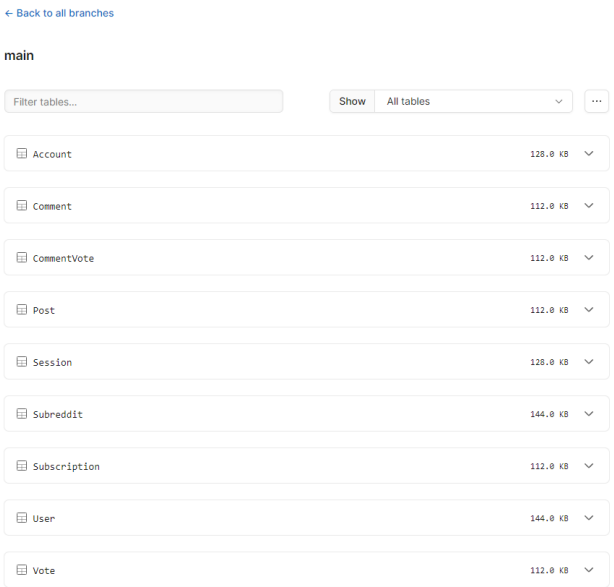

Рисунок 4.14 – Вдало створено таблиці бази даних

Після тестування клієнтської частини та створення публікацій до бази даних повинні потрапити відповідні записи. Переконаємося в їхній наявності перейшовши до платформи PlanetScale та перевіримо відповідний дашборд з діаграмами.

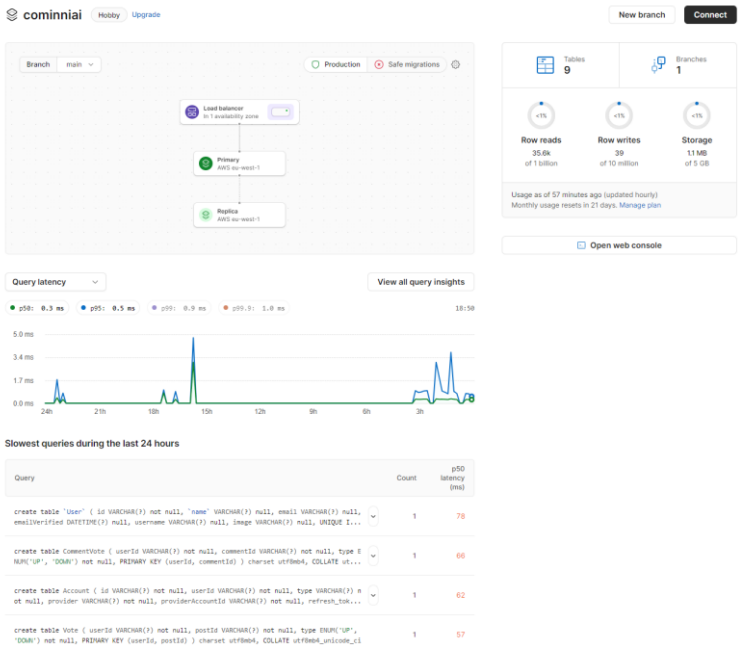

Рисунок 4.15 – Вдало відпрацьовані запити бази даних

Як можемо бачити усі дані були вдало збережені до бази даних, а записи про їхню маніпуляцію відстежено платформою та зображено у вигляді діаграми.

Перейдемо до наступного модуля, який відповідає за завантаження та кешування фото до нашого додатку.

Після того, як ми тестували створення публікації з фото, це фото надсилається на сервіс “Uploadthing”, де воно зберігається та буде кешоване у наступних запитах користувачів, що є досить вагомим покращенням та оптимізацією системи для економії витрати ресурсів та трафіку.

Перевіремо наявність фото після наших запитів у системі “Uploadthing”, де їхнє відображення повинно бути подане у вигляді назви файлу та дати збереження у системі.

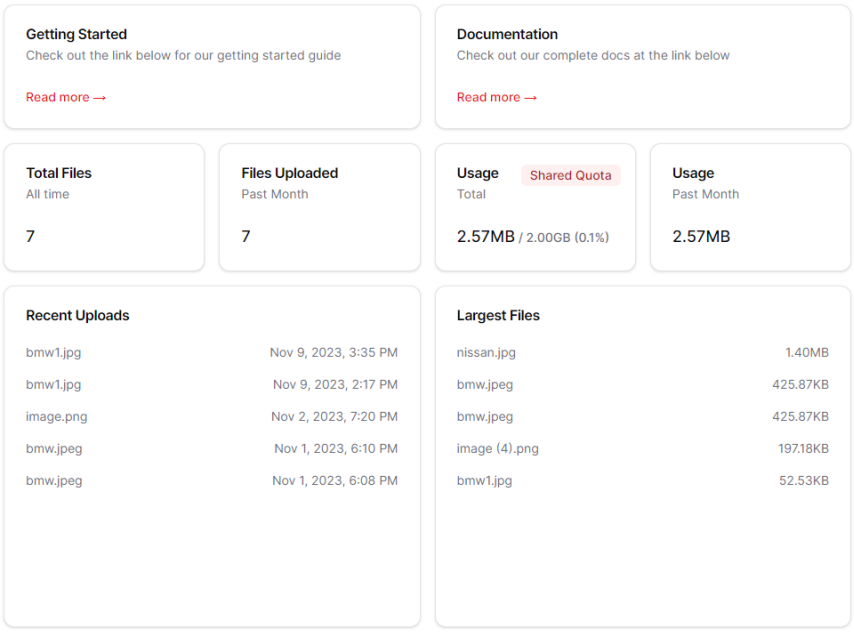

Рисунок 4.16 – Збережені фото на платформі “Uploadthing”

Переконавшись у відмінній роботі логіки, перейдемо до наступного модулю, який відповідає за кешування даних користувачів.

Кешування даних допомагає покращити швидкість завантаження сторінок, а також зберігати копії ресурсів на більш доступних місцях, таких як локальний комп'ютер користувача.

У нашому випадку ми кешуємо дані, щодо інтеракції користувачів з публікаціями, що оптимізовує систему, та забезпечує запобігання використання надмірної кількості запитів до сервери чи бази даних, що у свою чергу досить ресурсозатратно для системи.

Таким чином, при повторній інтеракції користувача з певною публікацією, оновляться лише ті дані, які були змінені, що буде свідчити про вірне відпрацювання логіки та модулю в цілому.

Для цього було використано сервіс “Upstash”, перейдемо до нього та переконаємось, що наші дані обробляються та кешуються.

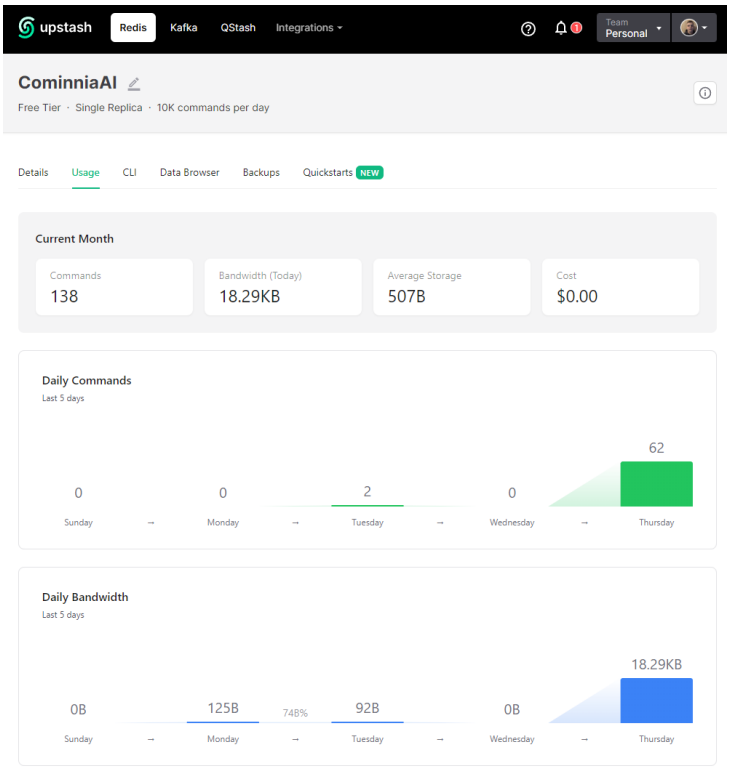

Рисунок 4.17 – Діаграма кешування даних

Відображені діаграми свідчать про вдалу роботу сервісу та коректну взаємодію з серверною частиною додатку.

Отже, було вдало протестовано ключові модулі серверної частини додатку, описано їхню роботу та кроки тестування після чого не було виявлено жодних помилок у роботі системи.

4.4 Висновки розділу 4

Підсумовуючи, після детального аналізу методів тестування для програмного засобу мережі спільнот з інтегрованим штучним інтелектом, визначено, що вибір був зроблений на користь методу сірого ящика. В рамках цього методу було успішно протестовано клієнтську та серверну частини додатку, включаючи інтегрований штучний інтелект.

Отже, можна зробити висновок, що тестування виявилося успішним, і програмний засіб готовий до ефективного використання як мережа спільнот з інтегрованим штучним інтелектом.

5 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 «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом» є оцінювання науково-технічного рівня та рівня

комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [37].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
	Активна конкуренція великих	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

Практична здійсненність					
	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
2	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	4	4
2. Ринкові переваги (наявність аналогів)	4	4	4
3. Ринкові переваги (ціна продукту)	3	4	3
4. Ринкові переваги (технічні властивості)	4	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	41	41	40
Середньоарифметична сума балів $СБ_c$	40,7		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки.

При цьому використаємо рекомендації, наведені в табл. 5.3.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній

Продовження таблиці 5.3

11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом» становить 40,7 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «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.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [37]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=21 день.

$$З_о = 22000{,}00 \cdot 60 / 21 = 62857{,}14 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	22000	1047,62	60	62857,14
Інженер-розробник програмного забезпечення	20000	952,38	55	52380,95
Консультант	18000	857,14	20	17142,86
Всього				132380,95

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [37] ;

K_c – 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.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}1 \cdot 1{,}65 / (21 \cdot 8) = 72{,}38$ грн.

$З_{р1} = 72{,}38 \cdot 4{,}00 = 289{,}54$грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка електронно-обчислювального обладнання	4	2	1,1	72,38	289,54

Продовження таблиці 5.5.

Підготовка робочого місця дослідника	3	2	1,1	72,38	217,15
Інсталяція програмного забезпечення	4	5	1,7	111,87	447,46
Всього					954,15

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

$З_{дод} = (132380,95 + 954,15) \cdot 12 / 100\% = 16000,21$грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн = (132380,95 + 954,15 + 16000,21) \cdot 22 / 100\% = 32853,76$грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (5.6)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 2 \cdot 205{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 451{,}0$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір Calipso Plus A4-500-80	205	2	0	0	451
Папір для записів Calipso Parers Light A5	110	1	0	0	121

Продовження таблиці 5.6

Канцелярське приладдя (набір офісного працівника)	155	2	0	0	341
Картридж для принтера Canon LBP6500	1100	1	0	0	1210
Всього					2123

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_е$)*, які використовують при проведенні НДР на тему «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (5.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = *20000· 3 · 1,1 = 66000 грн.*

Отримані результати зведемо до таблиці

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Ноутбук HP Laptop 15s-eq2053ua (4A7N8EA)	3	20000	66000
Маршрутизатор	1	1700	1870
Всього			67870

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *1300,00 · 1 · 1,12 = 1456 грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Операційна система Microsoft Windows 10 Pro for Workstations	3	15000	50400
Графічний редактор Adobe Photoshop	1	1300	1456
Upstash	1	100	112
Uploadthing	1	360	403,2
Всього			52371,2

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (66000 \cdot 3) / (3 \cdot 12) =$ 5500,00 *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук HP Laptop 15s-eq2053ua (4A7N8EA)	66000	3	3	5500,0
Маршрутизатор	1870	3	3	155,83
Операційна система Microsoft Windows 10 Pro for Workstations	50400	5	3	2520,0
Графічний редактор Adobe Photoshop	1456	4	3	91,00
Upstash	112	3	3	9,33
Uploadthing	403,2	3	3	33,60
Всього				8309,77

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot K_{вni}}{\eta_i}, \qquad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 7,50$ грн;

$К_{вні}$ – коефіцієнт, що враховує використання потужності, $К_{вні} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,2 \cdot 480,0 \cdot 7,50 \cdot 0,95 / 0,97 = 705,15$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук HP Laptop 15s-eq2053ua (4A7N8EA)	0,2	480	705,15
Ноутбук HP Laptop 15s-eq2053ua (4A7N8EA)	0,2	440	646,39
Ноутбук HP Laptop 15s-eq2053ua (4A7N8EA)	0,2	160	235,05
Блок живлення маршрутизатора	0,02	60	8,81
Всього			1595,41

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та

приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (5.11)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (132380,95+ 954,15) · 20 / 100% = 26667,02 грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{H_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 40%.

$B_{сп}$ = (132380,95+ 954,15) · 40 / 100% = 53334,04 грн.

5.2.11 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» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 75%.

$I_в$ = (132380,95+ 954,15) · 75 / 100% = 100001,33 грн.

5.2.12 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» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 150%.

$В_{нзв}$ = (132380,95+ 954,15) · 150 / 100% = 200 002,66грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (5.15)$$

$B_{заг}$= *694463,51 грн.*

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

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,6.

ЗВ = 694463,51/ 0,6= 1157439,19 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

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 [38].

Результати дослідження проведені за темою «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 2000 користувачів;

2-й рік – 3000 користувачів;

3-й рік – 2500 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 10000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 180000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 20000 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [37]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 25\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (20000,00 \cdot 10000,00 + 200000,00 \cdot 2000) \cdot 0,83 \cdot 0,25 \cdot (1\text{-}0,18/100\%) =$

102459000грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(20000,00·10000,00+200000,00·(2000+3000))·0,83·0,25·(1-0,18/100%)=204918000грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(20000,00·10000,00+2000000,00·(2000+3000+2500))·0,83·0,25·(1-0,18/100%)= 290300500 грн.

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

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.18)$$

де $\Delta П_i$ – 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;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,3;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ = 102459000/(1+0,3)^1+204918000/(1+0,3)^2+290300500/(1+0,3)^3=332202826,58 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=4;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1157439,19грн.

$PV = k_{інв} \cdot ЗВ$ = 4 · 1157439,19= 4629756,748 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 332202826,58грн;

PV – теперішня вартість початкових інвестицій, 4629756,748грн.

$E_{абс} = ПП - PV$ = 332202826,58 - 4629756,748= 327573069,83грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.21}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 327573069,83грн;

PV – теперішня вартість початкових інвестицій, 4629756,748грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+327573069{,}83/4629756{,}748)^{1/3} = 3{,}16.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \qquad (5.22)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,11+0,18 = 0,29 < 0,68 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \qquad (5.23)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 3,16= 0,32 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки розділу 5

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом» становить 40,7 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

Також термін окупності становить 0,32 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом».

ВИСНОВКИ

В магістерській кваліфікаційній роботі була проведена розробка методів та засобів мережі спільнот з інтегрованим штучним інтелектом, яка надає змогу користувачам використовувати можливості ШІ у додатку.

Виконано аналіз сучасних методів інтеграції штучного інтелекту в мережі спільнот. Було проведено дослідження предметної області взаємодії між користувачами та інтеграції штучного інтелекту, під час виконання якого визначено цілі використання й виконано порівняльний аналіз додатків зі схожим функціоналом.

Розроблено метод інтеграції генерованого тексту штучного інтелекту з використанням відкритих бібліотек, розроблено модуль генерації тексту з використанням штучного інтелекту. Даний модуль може слугувати прикладом для подальшого вдосконалення, притримуючись усіх принципів масштабованості.

Розроблено метод динамічної модерації контенту шляхом використання тренованої моделі штучного інтелекту в поєднанні з бібліотекою машинного навчання для автоматизованої перевірки вмісту публікацій на наявність ненормативного наповнення, для забезпечення прозорості та безпеки використання програмного засобу.

Покращено модель рекомендацій контенту цільовій аудиторії шляхом визначення рейтингу публікації враховуючи час та кількість голосів, що дозволяє відокремлювати особливо популярні теми для обговорення на платформі, тим самим, заохочувати більше людей для обговорювання та заволікати відповідну аудиторію.

Проведено архітектурне проектування додатку, яке дозволило створити добре організований, модульний та масштабований додаток, що в результаті забезпечує зручну та ефективну взаємодію з користувачами. У якості моделі розробки ПЗ було обрано ітераційну модель розробки, що надає достатню гнучкість під час розробки.

За домогою таких засобів, як Figma та Adobe Photoshop, було розроблено логотипи, а також дизайн компонентів мережі спільнот.

Фреймворк Next дозволив побудувати потужний серверний рендеринг та забезпечити швидкість завантаження сторінок. Використання Prisma та Planetscale дозволило зберігати, управляти та ефективно взаємодіяти з даними платформи, забезпечуючи надійність та масштабованість системи.

Після здійснення розробки програмного забезпечення було проведено тестування сірої скриньки. Під час проведення тестування було здійснено аналіз головних компонент додатку, і створено відповідні послідовності їх тестування. Після виконання тестування було доведено, що система відповідає своїй головній задачі, а також виконує усі функції коректно.

У результаті проведення роботи було створено застосунок за допомогою Next.js та набором технологій, які разом надають потужні можливості для створення подібного роду програмних продуктів.

Отримані в магістерській кваліфікаційній роботі наукові та практичні результати можна використати для розробки мереж спільнот з використанням штучного інтелекту

Результати роботи опубліковані в 2 наукових публікаціях.

Практична цінність одержаних результатів полягає в розроблених програмних компонентах, які були протестовані та можуть бути використані для інших проектів, адже мають потужну масштабованість та легку інтеграцію.

Задачі магістерської кваліфікаційної роботи виконано в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kraut R. Building Successful Online Communities: Evidence-Based Social Design, 2012. 309 с.

2. Kulshrestha R. A Beginner’s Guide to Latent Dirichlet Allocation(LDA). URL: https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2 (дата звернення: 11.10.2023).

3. Мельник Д.О. Правила інтеграції штучного інтелекту. Міжнародна науково-практична Інтернет конференція “Електронні інформаційні ресурси: створення, використання, доступ” 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. 323с.

4. Мельник Д.О. Модель використання штучного інтелекту у комп’ютерній візуалізації. CXXXV міжнародна інтернет — конференція «наукові підсумки 2023 року», м. Запоріжжя. 2023.

5. Bhatti U. All of Twitter’s Latest Features and how Businesses are Using Them. URL: https://buffer.com/resources/all-of-twitters-latest-features-and-how-to-use-them/ (дата звернення: 12.10.2023).

6. Clifford C. How to Use Facebook: A Beginner’s Guide. URL: https://blog.hubspot.com/marketing/how-to-use-facebook (дата звернення: 12.10.2023).

7. Pokrop J. What’s New on Instagram in 2023: New Features and Updates. URL: https://napoleoncat.com/blog/instagram-new-features-and-updates/ (дата звернення: 13.10.2023).

8. Terence B. Applying AI in Software Development: Best Practices and Examples. URL: https://blog.dreamfactory.com/applying-ai-in-software-development-best-practices-and-examples/ (дата звернення: 12.10.2023).

9. Bittner K. Managing Iterative Software Development Projects, 2018. 672 с.

10. Коваленко О.О. Використання алгоритмів у бібліотеках мов програмування. Міжнародна науково-практична Інтернет-конференція, 9-10 листопада 2020, м. Суми/Вінниця. 2020.

11. Bhargava A. Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People, 2016. 256 с.

12. Rumpe B. Modeling with UML: Language, Concepts, Methods, 2016. 295 с.

13. Stackpole T. Content Moderation Is Terrible by Design. URL: https://hbr.org/2022/11/content-moderation-is-terrible-by-design (дата звернення: 13.10.2023).

14. Griffin R. Algorithmic Content Moderation Brings New Opportunities and Risks. URL: https://www.cigionline.org/articles/algorithmic-content-moderation-brings-new-opportunities-and-risks/ (дата звернення: 13.10.2023).

15. Tarud J. AI Models: How Does It Work? URL: https://www.koombea.com/blog/ai-models/ (дата звернення: 13.10.2023).

16. Dorcas A. Everything you need to know about social media algorithms. URL: https://sproutsocial.com/insights/social-media-algorithms/ (дата звернення: 13.10.2023).

17. Elliott J. Understanding Behavioral Synthesis: A Practical Guide to High-Level Design, 2013. 337с.

18. Mirza A. What is Full Stack in Software Development? URL: https://levelup.gitconnected.com/what-is-full-stack-in-software-development-67caf2fbaec8 (дата звернення: 14.10.2023).

19. Dayanand A. The Ultimate Guide to Becoming a Full-Stack Developer in 2023. URL: https://medium.com/@adarsh-d/the-ultimate-guide-to-becoming-a-full-stack-developer-in-2023-c0358c8d8f60 (дата звернення: 14.10.2023).

20. Camden R. The Jamstack Book: Beyond static sites with JavaScript, APIs, and markup, 2022. 280 с.

21. Derek A. Why Tailwind CSS Became So Popular: A Developer’s Guide. URL: https://betterprogramming.pub/why-tailwind-css-became-so-popular-a-developers-guide-11213c08fa46 (дата звернення: 14.10.2023).

22. Jayakody P. Authentication with Next.js 13 and Next Auth. URL: https://medium.com/ascentic-technology/authentication-with-next-js-13-and-next-auth-9c69d55d6bfd (дата звернення: 14.10.2023).

23. Aayushmaan A. Gain an understanding of Uploadthing. URL: https://medium.com/@apurvkashyappurnea/gain-an-understanding-of-uploadthing-33ad6d0b553a (дата звернення: 14.10.2023).

24. YLD. Planet Scale and Vitess: A Technical Deepdive. URL: https://medium.com/yld-blog/planet-scale-and-vitess-a-technical-deepdive-f1e039d443aa (дата звернення: 14.10.2023).

25. Tepes A. Introduction for firther dive into Prisma. URL: https://medium.com/@tepes_alexandru/introduction-to-prisma-d587e95d2a07 (дата звернення: 14.10.2023).

26. Richard B. What Is MySQL and How Does It Work. URL: https://www.hostinger.com/tutorials/what-is-mysql (дата звернення: 14.10.2023).

27. Flanagan D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language, 2020. 704c.

28. Maxwell D. Redis Essentials: Harness the power of Redis to integrate and manage your projects efficiently, 2015. 230c.

29. Hanane D. Complete Process for Fine Tuning GPT-3.5-Turbo using OpenAI API. URL: https://medium.com/ai-advances/complete-process-for-fine-tuning-gpt-3-5-turbo-using-openai-api-db4a50b3de1a (дата звернення: 14.10.2023).

30. Karan A. Ultimate Guide to Getting Started with TensorFlow.js. URL: https://medium.com/@karanaryan/ultimate-guide-to-getting-started-with-tensorflow-js-7db1d3f3954e (дата звернення: 14.10.2023).

31. Conrad C. Adobe Photoshop Classroom in a Book, 2022. 416c.

32. Mischook S. Web Design - Start Here: A No-Nonsense, Jargon Free Guide to the Fundamentals of Web Design, 2015. 224c.

33. Marquez-Soto P. Backend Developer in 30 Days: Acquire Skills on API Designing, Data Management, Application Testing, Deployment, Security and Performance Optimization, 2022. 678c.

34. Riva M. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production, 2022. 366c.

35. Khawaja G. Practical Web Penetration Testing: Secure web applications using Burp Suite, Nmap, Metasploit, and more, 2018. 294c.

36. Пилипенко Д. Ю., Коваленко О.О. Методика оцінювання рівня покриття процесів тестування програмних продуктів. XVI МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ, 19-20 ЖОВТНЯ 2023 р., м. Одеса. 2023.

37. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

38. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

ДОДАТОК А
(обов'язковий)
Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«19» вересня 2023 року

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доц Коваленко. О.О.
«19» вересня 2023 р.
Виконав:
студент гр. 3ПІ-22м Мельник Д.О.
«19» вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом».

Галузь застосування – комунікація та соціальне життя; професійні та цільові спільноти.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №247 від 18 вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення рівня якості взаємодії користувачів між собою у мережі спільнот шляхом використання можливостей інтегрованого штучного інтелекту та за рахунок впровадження нових методів та засобів динамічної модерації та рекомендації контенту цільовій аудиторії.

Призначення роботи – розробка методів і програмних засобів для підвищення ефективності комунікації користувачів надаючи засоби для творчості та створення контенту.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Kraut R. Building Successful Online Communities: Evidence-Based Social Design, 2012. 309 c.

2. Terence B. Applying AI in Software Development: Best Practices and Examples. URL:https://blog.dreamfactory.com/applying-ai-in-software-development-best-practices-and-examples/ (дата звернення: 12.10.2023).

3. Griffin R. Algorithmic Content Moderation Brings New Opportunities and Risks. URL: https://www.cigionline.org/articles/algorithmic-content-moderation-brings-new-opportunities-and-risks/ (дата звернення: 13.10.2023).

4. Camden R. The Jamstack Book: Beyond static sites with JavaScript, APIs, and markup, 2022. 280 с.

5. Mischook S. Web Design - Start Here: A No-Nonsense, Jargon Free Guide to the Fundamentals of Web Design, 2015. 224с.

5. Технічні вимоги

Вихідні дані до роботи: середовища розробки – Visual Studio Code, фреймворк Next.js, , методи та засоби створення мережі спільнот з інтегрованим штучним інтелектом, інструменти на базі штучного інтелекту – OpenAI API, Tensoflor; тренована модель штучного інтелекту – Toxicity; система управління базами даних – PlanetScale; мова запитів SQL.

6. Конструктивні вимоги.

Розроблене програмне забезпечення повинно відповідати естетичним та ергономічним вимогам, бути зручним в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз предметної галузі та постановка задач дослідження	20.09.2023 – 30.09. 2023
2	Розробка методів, алгоритмів та архітектури мережі спільнот	01.10. 2023 – 10.10. 2023
3	Реалізація методів і програмних засобів для створення мережі спільнот з інтегрованим штучним інтелектом	11.10. 2023 – 25.10. 2023
4	Тестування програмного додатку	26.10.2023 – 20.11.2023
5	Економічна частина	21.11.2023 – 01.12.2023

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б

(обов'язковий)

Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Розробка методів і програмних засобів мережі спільнот з інтегрованим штучним інтелектом

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ЗПІ – 22м

Науковий керівник: Коваленко О.О.

Unicheck	
Оригінальність	**97%**
Схожість	3%

Аналіз звіту подібності

■ **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 автора. Роботу направити на доопрацювання.

□ 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.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Мельник Д.О.

Керівник роботи _____ Коваленко О.О.

ДОДАТОК В

(довідниковий)

Лістинг коду

Editor.tsx

```
'use client'

import EditorJS from '@editorjs/editorjs'
import { zodResolver } from '@hookform/resolvers/zod'
import { usePathname, useRouter } from 'next/navigation'
import { useCallback, useEffect, useRef, useState } from 'react'
import { useForm } from 'react-hook-form'
import TextareaAutosize from 'react-textarea-autosize'
import { z } from 'zod'

import { toast } from '@/hooks/use-toast'
import { uploadFiles } from '@/lib/uploadthing'
import { PostCreationRequest, PostValidator } from '@/lib/validators/post'
import { useMutation } from '@tanstack/react-query'
import axios from 'axios'

import '@/styles/editor.css'

type FormData = z.infer<typeof PostValidator>

interface EditorProps {
  subredditId: string
}

export const Editor: React.FC<EditorProps> = ({ subredditId }) => {
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm<FormData>({
    resolver: zodResolver(PostValidator),
    defaultValues: {
      subredditId,
      title: '',
      content: null,
    },
  })
  const ref = useRef<EditorJS>()
  const _titleRef = useRef<HTMLTextAreaElement>(null)
  const router = useRouter()
  const [isMounted, setIsMounted] = useState<boolean>(false)
  const pathname = usePathname()

  const { mutate: createPost } = useMutation({
    mutationFn: async ({
      title,
      content,
      subredditId,
    }: PostCreationRequest) => {
      const payload: PostCreationRequest = { title, content, subredditId }
```

```
      const { data } = await axios.post('/api/subreddit/post/create', payload)
      return data
    },
    onError: () => {
      return toast({
        title: 'Something went wrong.',
        description: 'Your post was not published. Please try again.',
        variant: 'destructive',
      })
    },
    onSuccess: () => {
      // turn pathname /r/mycommunity/submit into /r/mycommunity
      const newPathname = pathname.split('/').slice(0, -1).join('/')
      router.push(newPathname)

      router.refresh()

      return toast({
        description: 'Your post has been published.',
      })
    },
  })

  const initializeEditor = useCallback(async () => {
    const EditorJS = (await import('@editorjs/editorjs')).default
    const Header = (await import('@editorjs/header')).default
    const Embed = (await import('@editorjs/embed')).default
    const Table = (await import('@editorjs/table')).default
    const List = (await import('@editorjs/list')).default
    const Code = (await import('@editorjs/code')).default
    const LinkTool = (await import('@editorjs/link')).default
    const InlineCode = (await import('@editorjs/inline-code')).default
    const ImageTool = (await import('@editorjs/image')).default

    if (!ref.current) {
      const editor = new EditorJS({
        holder: 'editor',
        onReady() {
          ref.current = editor
        },
        placeholder: 'Type here to write your post...',
        inlineToolbar: true,
        data: { blocks: [] },
        tools: {
          header: Header,
          linkTool: {
            class: LinkTool,
            config: {
              endpoint: '/api/link',
            },
          },
          image: {
            class: ImageTool,
            config: {
              uploader: {
                async uploadByFile(file: File) {
                  // upload to uploadthing
                  const [res] = await uploadFiles([file], 'imageUploader')

                  return {
                    success: 1,
                    file: {
```

```
                    url: res.fileUrl,
                  },
                }
              },
            },
          },
        },
        list: List,
        code: Code,
        inlineCode: InlineCode,
        table: Table,
        embed: Embed,
      },
    })
  }
}, [])

useEffect(() => {
  if (Object.keys(errors).length) {
    for (const [_key, value] of Object.entries(errors)) {
      value
      toast({
        title: 'Something went wrong.',
        description: (value as { message: string }).message,
        variant: 'destructive',
      })
    }
  }
}, [errors])

useEffect(() => {
  if (typeof window !== 'undefined') {
    setIsMounted(true)
  }
}, [])

useEffect(() => {
  const init = async () => {
    await initializeEditor()

    setTimeout(() => {
      _titleRef?.current?.focus()
    }, 0)
  }

  if (isMounted) {
    init()

    return () => {
      ref.current?.destroy()
      ref.current = undefined
    }
  }
}, [isMounted, initializeEditor])

async function onSubmit(data: FormData) {
  const blocks = await ref.current?.save()

  const payload: PostCreationRequest = {
    title: data.title,
    content: blocks,
    subredditId,
```

```
    }

    createPost(payload)
  }

  if (!isMounted) {
    return null
  }

  const { ref: titleRef, ...rest } = register('title')

  return (
    <div className='w-full p-4 bg-zinc-50 rounded-lg border border-zinc-200'>
      <form
        id='subreddit-post-form'
        className='w-fit'
        onSubmit={handleSubmit(onSubmit)}>
        <div className='prose prose-stone dark:prose-invert'>
          <TextareaAutosize
            ref={(e) => {
              titleRef(e)
              // @ts-ignore
              _titleRef.current = e
            }}
            {...rest}
            placeholder='Title'
            className='w-full resize-none appearance-none overflow-hidden bg-
transparent text-5xl font-bold focus:outline-none'
          />
          <div id='editor' className='min-h-[500px]' />
          <p className='text-sm text-gray-500'>
            Use{' '}
            <kbd className='rounded-md border bg-muted px-1 text-xs uppercase'>
              Tab
            </kbd>{' '}
            to open the command menu.
          </p>
        </div>
      </form>
    </div>
  )
}
```

Post.tsx

```
"use client";

import { formatTimeToNow } from "@/lib/utils";
import { Post, User, Vote } from "@prisma/client";
import { MessageSquare } from "lucide-react";
import Link from "next/link";
import { FC, useRef } from "react";
import EditorOutput from "./EditorOutput";
import PostVoteClient from "./post-vote/PostVoteClient";
import { Toxicity } from "./ui/Toxicity";

type PartialVote = Pick<Vote, "type">;

interface PostProps {
  post: Post & {
    author: User;
```

```
    votes: Vote[];
  };
  votesAmt: number;
  subredditName: string;
  currentVote?: PartialVote;
  commentAmt: number;
}

const Post: FC<PostProps> = ({
  post,
  votesAmt: _votesAmt,
  currentVote: _currentVote,
  subredditName,
  commentAmt,
}) => {
  const pRef = useRef<HTMLParagraphElement>(null);

  return (
    <div className="rounded-md bg-white shadow">
      <div className="px-6 py-4 flex justify-between">
        <PostVoteClient
          postId={post.id}
          initialVotesAmt={_votesAmt}
          initialVote={_currentVote?.type}
        />

        <div className="w-0 flex-1">
          <div className="flex justify-between">
            <div className="max-h-40 mt-1 text-xs text-gray-500">
              {subredditName ? (
                <>
                  <a
                    className="underline text-zinc-900 text-sm underline-offset-2"
                    href={`/r/${subredditName}`}
                  >
                    r/{subredditName}
                  </a>
                  <span className="px-1">•</span>
                </>
              ) : null}
              <span>Posted by u/{post.author.username}</span>{" "}
              {formatTimeToNow(new Date(post.createdAt))}
            </div>
            <Toxicity post={post} />
          </div>
          <a href={`/r/${subredditName}/post/${post.id}`}>
            <h1 className="text-lg font-semibold py-2 leading-6 text-gray-900">
              {post.title}
            </h1>
          </a>

          <div
            className="relative text-sm max-h-40 w-full overflow-clip"
            ref={pRef}
          >
            <EditorOutput content={post.content} />
            {pRef.current?.clientHeight === 160 ? (
              // blur bottom if content is too long
              <div className="absolute bottom-0 left-0 h-24 w-full bg-gradient-to-t
from-white to-transparent"></div>
            ) : null}
          </div>
```

```
        </div>
      </div>

      <div className="bg-gray-50 z-20 text-sm px-4 py-4 sm:px-6">
        <Link
          href={`/r/${subredditName}/post/${post.id}`}
          className="w-fit flex items-center gap-2"
        >
          <MessageSquare className="h-4 w-4" /> {commentAmt} comments
        </Link>
      </div>
    </div>
  );
};
export default Post;
```

UserAuthForm.tsx

```
'use client'

import { cn } from '@/lib/utils'
import { signIn } from 'next-auth/react'
import * as React from 'react'
import { FC } from 'react'
import { Button } from '@/components/ui/Button'
import { useToast } from '@/hooks/use-toast'
import { Icons } from './Icons'

interface UserAuthFormProps extends React.HTMLAttributes<HTMLDivElement> {}

const UserAuthForm: FC<UserAuthFormProps> = ({ className, ...props }) => {
  const { toast } = useToast()
  const [isLoading, setIsLoading] = React.useState<boolean>(false)

  const loginWithGoogle = async () => {
    setIsLoading(true)

    try {
      await signIn('google')
    } catch (error) {
      toast({
        title: 'Error',
        description: 'There was an error logging in with Google',
        variant: 'destructive',
      })
    } finally {
      setIsLoading(false)
    }
  }

  return (
    <div className={cn('flex justify-center', className)} {...props}>
      <Button
        isLoading={isLoading}
        type='button'
        size='sm'
        className='w-full'
        onClick={loginWithGoogle}
        disabled={isLoading}>
        {isLoading ? null : <Icons.google className='h-4 w-4 mr-2' />}
        Google
```

```
      </Button>
    </div>
  )
}

export default UserAuthForm
```

Toxicity.tsx

```
"use client";

import { FC, useState } from "react";
import * as toxicity from "@tensorflow-models/toxicity";

interface ToxicityProps {
  post: any;
}

const Toxicity: FC<ToxicityProps> = ({ post }) => {
    console.log(post);

  const filteredPosts = post.content.blocks.filter(
    (el: any) => el.type == "paragraph"
  );
  let text: any;
  const [status, setStatus] = useState<string>("");
  if (!!filteredPosts.length) {
    text = filteredPosts[0].data.text;
    console.log(text);
    const threshold = 0.9;

    toxicity.load(threshold).then((model) => {
      const sentences = [text];

      model.classify(sentences).then((predictions) => {
        let res = predictions.filter((el) => el.results[0].match == true);
        if (!!res.length) {
          let status = predictions.filter(
            (el) => el.results[0].match == true
          )[0].label;
          setStatus(status);
          console.log(status);
        }
      });
    });
  }

  return <div>{status}</div>;
  return (
    <div className="bg-red-300 text-white px-2 py-1 rounded-lg">Offensive</div>
  );
};

export { Toxicity };
```

AiTextAssistant.tsx

```
"use client";
```

```
import TextAssistant from "@/components/text-generation/TextAssistant";
import Link from "next/link";

interface pageProps {
  params: {
    slug: string;
  };

}

const BasicModal = async ({ params }: pageProps) => {
  return (
    <div
      className="fixed z-10 inset-0 overflow-y-auto"
      id="error-modal"
      aria-labelledby="modal-title"
      role="dialog"
      aria-modal="true"
    >
      <div className="flex items-end justify-center min-h-screen pt-4 px-4 pb-20 text-
center sm:block sm:p-0">
        <div
          className="fixed inset-0 bg-gray-500 bg-opacity-75 transition-opacity"
          aria-hidden="true"
        ></div>
        <span
          className="hidden sm:inline-block sm:align-middle sm:h-screen"
          aria-hidden="true"
        >
          &#8203;
        </span>
        <div className="inline-block align-bottom bg-white rounded-lg px-4 pt-5 pb-4
text-left overflow-hidden shadow-xl transform transition-all sm:my-8 sm:align-middle
sm:max-w-lg sm:w-full sm:p-6">
          <div className="sm:flex sm:items-start">
            <div className="mt-3 text-center sm:mt-0 sm:ml-4 sm:text-left">
              <h3
                className="text-lg leading-6 font-medium text-gray-900"
                id="modal-title"
              >
                AI Text Assistant
              </h3>
              <div className="mt-2">
                <TextAssistant />
              </div>
            </div>
          </div>
          <div className="mt-5 sm:mt-4 sm:flex sm:flex-row-reverse">
            <Link
              href={`r/${params.slug}/submit`}
              type="button"
              className="w-full inline-flex justify-center rounded-md border border-
transparent shadow-sm px-4 py-2 bg-blue-600 text-base font-medium text-white hover:bg-
blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-red-500 sm:ml-3
sm:w-auto sm:text-sm"
            >
              Cancel
            </Link>
          </div>
        </div>
      </div>
    </div>
```

```
  );
};

export default BasicModal;
```

CommentsSection.tsx

```
import { getAuthSession } from '@/lib/auth'
import { db } from '@/lib/db'
import { Comment, CommentVote, User } from '@prisma/client'
import CreateComment from './CreateComment'
import PostComment from './comments/PostComment'

type ExtendedComment = Comment & {
  votes: CommentVote[]
  author: User
  replies: ReplyComment[]
}

type ReplyComment = Comment & {
  votes: CommentVote[]
  author: User
}

interface CommentsSectionProps {
  postId: string
  comments: ExtendedComment[]
}

const CommentsSection = async ({ postId }: CommentsSectionProps) => {
  const session = await getAuthSession()

  const comments = await db.comment.findMany({
    where: {
      postId: postId,
      replyToId: null,
    },
    include: {
      author: true,
      votes: true,
      replies: {
        include: {
          author: true,
          votes: true,
        },
      },
    },
  })

  return (
    <div className='flex flex-col gap-y-4 mt-4'>
      <hr className='w-full h-px my-6' />

      <CreateComment postId={postId} />

      <div className='flex flex-col gap-y-6 mt-4'>
        {comments
          .filter((comment) => !comment.replyToId)
          .map((topLevelComment) => {
            const topLevelCommentVotesAmt = topLevelComment.votes.reduce(
              (acc, vote) => {
```

```
                if (vote.type === 'UP') return acc + 1
                if (vote.type === 'DOWN') return acc - 1
                return acc
              },
              0
            )

            const topLevelCommentVote = topLevelComment.votes.find(
              (vote) => vote.userId === session?.user.id
            )

            return (
              <div key={topLevelComment.id} className='flex flex-col'>
                <div className='mb-2'>
                  <PostComment
                    comment={topLevelComment}
                    currentVote={topLevelCommentVote}
                    votesAmt={topLevelCommentVotesAmt}
                    postId={postId}
                  />
                </div>

                {topLevelComment.replies
                  .sort((a, b) => b.votes.length - a.votes.length)
                  .map((reply) => {
                    const replyVotesAmt = reply.votes.reduce((acc, vote) => {
                      if (vote.type === 'UP') return acc + 1
                      if (vote.type === 'DOWN') return acc - 1
                      return acc
                    }, 0)

                    const replyVote = reply.votes.find(
                      (vote) => vote.userId === session?.user.id
                    )

                    return (
                      <div
                        key={reply.id}
                        className='ml-2 py-2 pl-4 border-l-2 border-zinc-200'>
                        <PostComment
                          comment={reply}
                          currentVote={replyVote}
                          votesAmt={replyVotesAmt}
                          postId={postId}
                        />
                      </div>
                    )
                  })}
              </div>
            )
          })}
      </div>
    </div>
  )
}

export default CommentsSection
```

MiniCreatePost.tsx

```
'use client'

import { Button } from '@/components/ui/Button'
import { Input } from '@/components/ui/Input'
import { Image as ImageIcon, Link2 } from 'lucide-react'
import { FC } from 'react'
import { UserAvatar } from './UserAvatar'
import type { Session } from 'next-auth'
import { usePathname, useRouter } from 'next/navigation'

interface MiniCreatePostProps {
  session: Session | null
}

const MiniCreatePost: FC<MiniCreatePostProps> = ({ session }) => {
  const router = useRouter()
  const pathname = usePathname()

  return (
    <li className='overflow-hidden rounded-md bg-white shadow'>
      <div className='h-full px-6 py-4 flex justify-between gap-6'>
        <div className='relative'>
          <UserAvatar
            user={{
              name: session?.user.name || null,
              image: session?.user.image || null,
            }}
          />

          <span className='absolute bottom-0 right-0 rounded-full w-3 h-3 bg-green-500
outline outline-2 outline-white' />
        </div>
        <Input
          onClick={() => router.push(pathname + '/submit')}
          readOnly
          placeholder='Create post'
        />
        <Button
          onClick={() => router.push(pathname + '/submit')}
          variant='ghost'>
          <ImageIcon className='text-zinc-600' />
        </Button>
        <Button
          onClick={() => router.push(pathname + '/submit')}
          variant='ghost'>
          <Link2 className='text-zinc-600' />
        </Button>
      </div>
    </li>
  )
}

export default MiniCreatePost
```

SearchBar.tsx

```
'use client'

import { Prisma, Subreddit } from '@prisma/client'
```

```tsx
import { useQuery } from '@tanstack/react-query'
import axios from 'axios'
import debounce from 'lodash.debounce'
import { usePathname, useRouter } from 'next/navigation'
import { FC, useCallback, useEffect, useRef, useState } from 'react'

import {
  Command,
  CommandEmpty,
  CommandGroup,
  CommandInput,
  CommandItem,
  CommandList,
} from '@/components/ui/Command'
import { useOnClickOutside } from '@/hooks/use-on-click-outside'
import { Users } from 'lucide-react'

interface SearchBarProps {}

const SearchBar: FC<SearchBarProps> = ({}) => {
  const [input, setInput] = useState<string>('')
  const pathname = usePathname()
  const commandRef = useRef<HTMLDivElement>(null)
  const router = useRouter()

  useOnClickOutside(commandRef, () => {
    setInput('')
  })

  const request = debounce(async () => {
    refetch()
  }, 300)

  const debounceRequest = useCallback(() => {
    request()

    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [])

  const {
    isFetching,
    data: queryResults,
    refetch,
    isFetched,
  } = useQuery({
    queryFn: async () => {
      if (!input) return []
      const { data } = await axios.get(`/api/search?q=${input}`)
      return data as (Subreddit & {
        _count: Prisma.SubredditCountOutputType
      })[]
    },
    queryKey: ['search-query'],
    enabled: false,
  })

  useEffect(() => {
    setInput('')
  }, [pathname])

  return (
    <Command
```

```
      ref={commandRef}
      className='relative rounded-lg border max-w-lg z-50 overflow-visible'>
      <CommandInput
        isLoading={isFetching}
        onValueChange={(text) => {
          setInput(text)
          debounceRequest()
        }}
        value={input}
        className='outline-none border-none focus:border-none focus:outline-none ring-
0'
        placeholder='Search communities...'
      />

      {input.length > 0 && (
        <CommandList className='absolute bg-white top-full inset-x-0 shadow rounded-b-
md'>
          {isFetched && <CommandEmpty>No results found.</CommandEmpty>}
          {(queryResults?.length ?? 0) > 0 ? (
            <CommandGroup heading='Communities'>
              {queryResults?.map((subreddit) => (
                <CommandItem
                  onSelect={(e) => {
                    router.push(`/r/${e}`)
                    router.refresh()
                  }}
                  key={subreddit.id}
                  value={subreddit.name}>
                  <Users className='mr-2 h-4 w-4' />
                  <a href={`/r/${subreddit.name}`}>r/{subreddit.name}</a>
                </CommandItem>
              ))}
            </CommandGroup>
          ) : null}
        </CommandList>
      )}
    </Command>
  )
}

export default SearchBar
```

PostFeed.tsx

```
'use client'

import { INFINITE_SCROLL_PAGINATION_RESULTS } from '@/config'
import { ExtendedPost } from '@/types/db'
import { useIntersection } from '@mantine/hooks'
import { useInfiniteQuery } from '@tanstack/react-query'
import axios from 'axios'
import { Loader2 } from 'lucide-react'
import { FC, useEffect, useRef } from 'react'
import Post from './Post'
import { useSession } from 'next-auth/react'

interface PostFeedProps {
  initialPosts: ExtendedPost[]
  subredditName?: string
}
```

```tsx
const PostFeed: FC<PostFeedProps> = ({ initialPosts, subredditName }) => {
  const lastPostRef = useRef<HTMLElement>(null)
  const { ref, entry } = useIntersection({
    root: lastPostRef.current,
    threshold: 1,
  })
  const { data: session } = useSession()

  const { data, fetchNextPage, isFetchingNextPage } = useInfiniteQuery(
    ['infinite-query'],
    async ({ pageParam = 1 }) => {
      const query =
        `/api/posts?limit=${INFINITE_SCROLL_PAGINATION_RESULTS}&page=${pageParam}` +
        (!!subredditName ? `&subredditName=${subredditName}` : '')

      const { data } = await axios.get(query)
      return data as ExtendedPost[]
    },

    {
      getNextPageParam: (_, pages) => {
        return pages.length + 1
      },
      initialData: { pages: [initialPosts], pageParams: [1] },
    }
  )

  useEffect(() => {
    if (entry?.isIntersecting) {
      fetchNextPage()
    }
  }, [entry, fetchNextPage])

  const posts = data?.pages.flatMap((page) => page) ?? initialPosts

  return (
    <ul className='flex flex-col col-span-2 space-y-6'>
      {posts.map((post, index) => {
        const votesAmt = post.votes.reduce((acc, vote) => {
          if (vote.type === 'UP') return acc + 1
          if (vote.type === 'DOWN') return acc - 1
          return acc
        }, 0)

        const currentVote = post.votes.find(
          (vote) => vote.userId === session?.user.id
        )

        if (index === posts.length - 1) {
          return (
            <li key={post.id} ref={ref}>
              <Post
                post={post}
                commentAmt={post.comments.length}
                subredditName={post.subreddit.name}
                votesAmt={votesAmt}
                currentVote={currentVote}
              />
            </li>
          )
        } else {
          return (
```

```
            <Post
              key={post.id}
              post={post}
              commentAmt={post.comments.length}
              subredditName={post.subreddit.name}
              votesAmt={votesAmt}
              currentVote={currentVote}
            />
          )
        }
      })}

      {isFetchingNextPage && (
        <li className='flex justify-center'>
          <Loader2 className='w-6 h-6 text-zinc-500 animate-spin' />
        </li>
      )}
    </ul>
  )
}

export default PostFeed
```

CommentVotes.tsx

```
'use client'
import { Button } from '@/components/ui/Button'
import { toast } from '@/hooks/use-toast'
import { useCustomToasts } from '@/hooks/use-custom-toasts'
import { cn } from '@/lib/utils'
import { CommentVoteRequest } from '@/lib/validators/vote'
import { usePrevious } from '@mantine/hooks'
import { CommentVote, VoteType } from '@prisma/client'
import { useMutation } from '@tanstack/react-query'
import axios, { AxiosError } from 'axios'
import { ArrowBigDown, ArrowBigUp } from 'lucide-react'
import { FC, useState } from 'react'

interface CommentVotesProps {
  commentId: string
  votesAmt: number
  currentVote?: PartialVote
}

type PartialVote = Pick<CommentVote, 'type'>

const CommentVotes: FC<CommentVotesProps> = ({
  commentId,
  votesAmt: _votesAmt,
  currentVote: _currentVote,
}) => {
  const { loginToast } = useCustomToasts()
  const [votesAmt, setVotesAmt] = useState<number>(_votesAmt)
  const [currentVote, setCurrentVote] = useState<PartialVote | undefined>(
    _currentVote
  )
  const prevVote = usePrevious(currentVote)

  const { mutate: vote } = useMutation({
    mutationFn: async (type: VoteType) => {
      const payload: CommentVoteRequest = {
```

```
        voteType: type,
        commentId,
      }

      await axios.patch('/api/subreddit/post/comment/vote', payload)
    },
    onError: (err, voteType) => {
      if (voteType === 'UP') setVotesAmt((prev) => prev - 1)
      else setVotesAmt((prev) => prev + 1)

      // reset current vote
      setCurrentVote(prevVote)

      if (err instanceof AxiosError) {
        if (err.response?.status === 401) {
          return loginToast()
        }
      }

      return toast({
        title: 'Something went wrong.',
        description: 'Your vote was not registered. Please try again.',
        variant: 'destructive',
      })
    },
    onMutate: (type: VoteType) => {
      if (currentVote?.type === type) {
        // User is voting the same way again, so remove their vote
        setCurrentVote(undefined)
        if (type === 'UP') setVotesAmt((prev) => prev - 1)
        else if (type === 'DOWN') setVotesAmt((prev) => prev + 1)
      } else {
        // User is voting in the opposite direction, so subtract 2
        setCurrentVote({ type })
        if (type === 'UP') setVotesAmt((prev) => prev + (currentVote ? 2 : 1))
        else if (type === 'DOWN')
          setVotesAmt((prev) => prev - (currentVote ? 2 : 1))
      }
    },
  })

  return (
    <div className='flex gap-1'>
      {/* upvote */}
      <Button
        onClick={() => vote('UP')}
        size='xs'
        variant='ghost'
        aria-label='upvote'>
        <ArrowBigUp
          className={cn('h-5 w-5 text-zinc-700', {
            'text-emerald-500 fill-emerald-500': currentVote?.type === 'UP',
          })}
        />
      </Button>

      {/* score */}
      <p className='text-center py-2 px-1 font-medium text-xs text-zinc-900'>
        {votesAmt}
      </p>

      {/* downvote */}
```

```
      <Button
        onClick={() => vote('DOWN')}
        size='xs'
        className={cn({
          'text-emerald-500': currentVote?.type === 'DOWN',
        })}
        variant='ghost'
        aria-label='downvote'>
        <ArrowBigDown
          className={cn('h-5 w-5 text-zinc-700', {
            'text-red-500 fill-red-500': currentVote?.type === 'DOWN',
          })}
        />
      </Button>
    </div>
  )
}

export default CommentVotes
```

UserNameForms.tsx

```
'use client'

import { zodResolver } from '@hookform/resolvers/zod'
import { User } from '@prisma/client'
import { useRouter } from 'next/navigation'
import * as React from 'react'
import { useForm } from 'react-hook-form'
import * as z from 'zod'

import { Button } from '@/components/ui/Button'
import {
  Card,
  CardContent,
  CardDescription,
  CardFooter,
  CardHeader,
  CardTitle,
} from '@/components/ui/Card'
import { Input } from '@/components/ui/Input'
import { Label } from '@/components/ui/Label'
import { toast } from '@/hooks/use-toast'
import { cn } from '@/lib/utils'
import { UsernameValidator } from '@/lib/validators/username'
import { useMutation } from '@tanstack/react-query'
import axios, { AxiosError } from 'axios'

interface UserNameFormProps extends React.HTMLAttributes<HTMLFormElement> {
  user: Pick<User, 'id' | 'username'>
}

type FormData = z.infer<typeof UsernameValidator>

export function UserNameForm({ user, className, ...props }: UserNameFormProps) {
  const router = useRouter()
  const {
    handleSubmit,
    register,
    formState: { errors },
  } = useForm<FormData>({
```

```
  resolver: zodResolver(UsernameValidator),
  defaultValues: {
    name: user?.username || '',
  },
})

const { mutate: updateUsername, isLoading } = useMutation({
  mutationFn: async ({ name }: FormData) => {
    const payload: FormData = { name }

    const { data } = await axios.patch(`/api/username/`, payload)
    return data
  },
  onError: (err) => {
    if (err instanceof AxiosError) {
      if (err.response?.status === 409) {
        return toast({
          title: 'Username already taken.',
          description: 'Please choose another username.',
          variant: 'destructive',
        })
      }
    }

    return toast({
      title: 'Something went wrong.',
      description: 'Your username was not updated. Please try again.',
      variant: 'destructive',
    })
  },
  onSuccess: () => {
    toast({
      description: 'Your username has been updated.',
    })
    router.refresh()
  },
})

return (
  <form
    className={cn(className)}
    onSubmit={handleSubmit((e) => updateUsername(e))}
    {...props}>
    <Card>
      <CardHeader>
        <CardTitle>Your username</CardTitle>
        <CardDescription>
          Please enter a display name you are comfortable with.
        </CardDescription>
      </CardHeader>
      <CardContent>
        <div className='relative grid gap-1'>
          <div className='absolute top-0 left-0 w-8 h-10 grid place-items-center'>
            <span className='text-sm text-zinc-400'>u/</span>
          </div>
          <Label className='sr-only' htmlFor='name'>
            Name
          </Label>
          <Input
            id='name'
            className='w-[400px] pl-6'
            size={32}
```

```
              {...register('name')}
            />
            {errors?.name && (
              <p className='px-1 text-xs text-red-600'>{errors.name.message}</p>
            )}
          </div>
        </CardContent>
        <CardFooter>
          <Button isLoading={isLoading}>Change name</Button>
        </CardFooter>
      </Card>
    </form>
  )
}
```

DropDownMenu.tsx

```
"use client"

import * as React from "react"
import * as DropdownMenuPrimitive from "@radix-ui/react-dropdown-menu"
import { Check, ChevronRight, Circle } from "lucide-react"

import { cn } from "@/lib/utils"

const DropdownMenu = DropdownMenuPrimitive.Root

const DropdownMenuTrigger = DropdownMenuPrimitive.Trigger

const DropdownMenuGroup = DropdownMenuPrimitive.Group

const DropdownMenuPortal = DropdownMenuPrimitive.Portal

const DropdownMenuSub = DropdownMenuPrimitive.Sub

const DropdownMenuRadioGroup = DropdownMenuPrimitive.RadioGroup

const DropdownMenuSubTrigger = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.SubTrigger>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubTrigger> & {
    inset?: boolean
  }
>(({ className, inset, children, ...props }, ref) => (
  <DropdownMenuPrimitive.SubTrigger
    ref={ref}
    className={cn(
      "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm
outline-none focus:bg-accent data-[state=open]:bg-accent",
      inset && "pl-8",
      className
    )}
    {...props}
  >
    {children}
    <ChevronRight className="ml-auto h-4 w-4" />
  </DropdownMenuPrimitive.SubTrigger>
))
DropdownMenuSubTrigger.displayName =
  DropdownMenuPrimitive.SubTrigger.displayName
```

```
const DropdownMenuSubContent = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.SubContent>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubContent>
>(({ className, ...props }, ref) => (
  <DropdownMenuPrimitive.SubContent
    ref={ref}
    {...props}
  />
))
DropdownMenuSubContent.displayName =
  DropdownMenuPrimitive.SubContent.displayName

const DropdownMenuContent = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Content>
>(({ className, sideOffset = 4, ...props }, ref) => (
  <DropdownMenuPrimitive.Portal>
    <DropdownMenuPrimitive.Content
      ref={ref}
      sideOffset={sideOffset}
      {...props}
    />
  </DropdownMenuPrimitive.Portal>
))
DropdownMenuContent.displayName = DropdownMenuPrimitive.Content.displayName

const DropdownMenuItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Item>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Item> & {
    inset?: boolean
  }
>(({ className, inset, ...props }, ref) => (
  <DropdownMenuPrimitive.Item
    ref={ref}
    className={cn(
      "relative flex select-none items-center rounded-sm px-2 py-1.5 text-sm outline-
none transition-colors focus:bg-accent focus:text-accent-foreground hover:bg-zinc-100
data-[disabled]:pointer-events-none data-[disabled]:opacity-50",
      inset && "pl-8",
      className
    )}
    {...props}
  />
))
DropdownMenuItem.displayName = DropdownMenuPrimitive.Item.displayName

const DropdownMenuCheckboxItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.CheckboxItem>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.CheckboxItem>
>(({ className, children, checked, ...props }, ref) => (
  <DropdownMenuPrimitive.CheckboxItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-
2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground
data-[disabled]:pointer-events-none data-[disabled]:opacity-50",
      className
    )}
    checked={checked}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
```

```
      <DropdownMenuPrimitive.ItemIndicator>
        <Check className="h-4 w-4" />
      </DropdownMenuPrimitive.ItemIndicator>
    </span>
    {children}
  </DropdownMenuPrimitive.CheckboxItem>
))
DropdownMenuCheckboxItem.displayName =
  DropdownMenuPrimitive.CheckboxItem.displayName

const DropdownMenuRadioItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.RadioItem>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.RadioItem>
>(({ className, children, ...props }, ref) => (
  <DropdownMenuPrimitive.RadioItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-
2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground
data-[disabled]:pointer-events-none data-[disabled]:opacity-50",
      className
    )}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <DropdownMenuPrimitive.ItemIndicator>
        <Circle className="h-2 w-2 fill-current" />
      </DropdownMenuPrimitive.ItemIndicator>
    </span>
    {children}
  </DropdownMenuPrimitive.RadioItem>
))
DropdownMenuRadioItem.displayName = DropdownMenuPrimitive.RadioItem.displayName
```

PostRating.js

```
const { floor, log, max, abs } = Math;
const epoch = new Date(1970, 0, 1);
function epochSeconds(date) {
  const diff = date - epoch;
  const secondsInDay = 86400;
  const microsecondsToSeconds = 1e-6;
  return (
    floor(diff / 1000) * secondsInDay
    + (diff % 1000) * microsecondsToSeconds
  );
}
function score(ups, downs) {
  return ups - downs;
}
function hot(ups, downs, date) {
  const s = score(ups, downs);
  const order = log(max(abs(s), 1), 10);
  const sign = s > 0 ? 1 : s < 0 ? -1 : 0;
  const seconds = epochSeconds(date) - 1134028003;
  return round(sign * order + seconds / 45000, 7);
}
const ups = 10;
const downs = 5;
const date = new Date(2023, 11, 11);
const hotness = hot(ups, downs, date);
```

ДОДАТОК Г
(довідниковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ МЕРЕЖІ СПІЛЬНОТ З ІНТЕГРОВАНИМ ШТУЧНИМ ІНТЕЛЕКТОМ

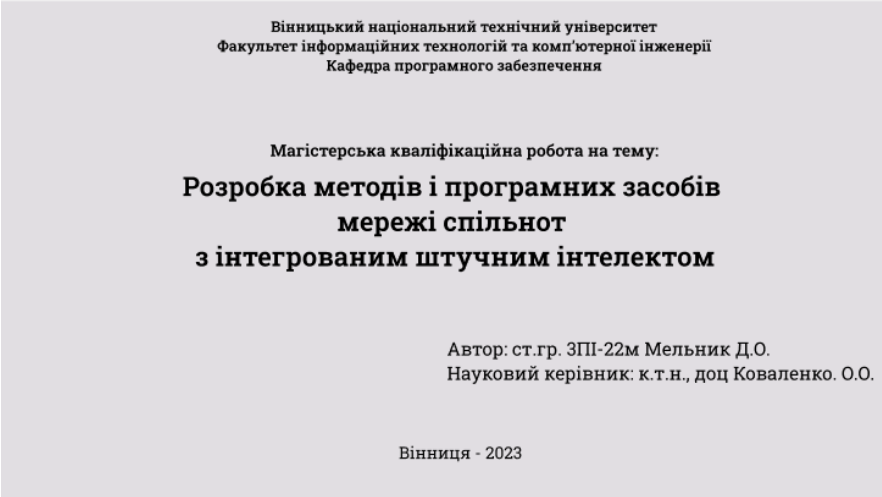

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність теми

Рисунок Г.3 – Мета, предмет та об’єкт дослідження

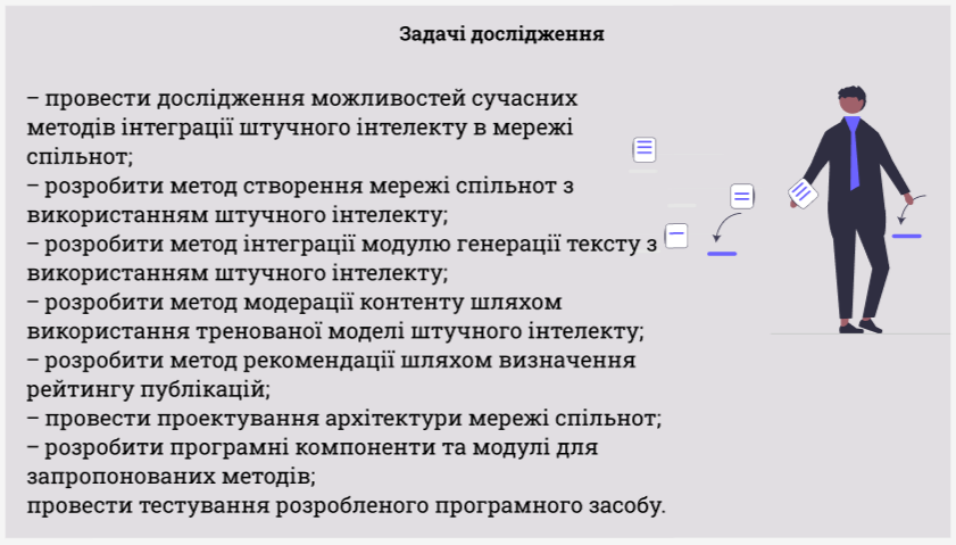

Рисунок Г.4 – Задачі дослідження

Рисунок Г.5 – Новизна отриманих результатів

Рисунок Г.6 – Практична цінність отриманих результатів

Порівняльна характеристика аналогів

Критерій	Twitter	Instagram	Facebook	ComunnAI
Генерація тексту	0	0	1	1
Взаємодія користувача з ШІ	0	0	0	1
Можливість створення спільнот	1	1	1	1
Відсутність платних функцій	0	0	0	1
Автоматизована модерація контенту	1	1	0	1
Підсумковий результат	2	2	2	5

Рисунок Г.7 – Порівняльна характеристика застосунків

Метод та блок-схема алгоритму модерації контенту шляхом використання тренованої моделі ШІ

Представимо модель як множину оцінок, яка формується як функція на основі множини кількості порушень в реченні за ймовірністю порушень:

$$P = k \cdot P1 + (1 - k) \cdot P2$$

– P - оцінка ймовірності порушення (від 0 до 1);

– P1 і P2 - відсотки ймовірності для кожного рядка в масиві (значення типу Float32 від 0 до 1);

– k - коефіцієнт важливості для текстового контенту (від 0 до 1). Коефіцієнт k визначає, наскільки важливим є текстового контент для загального прогнозу. Якщо k більше, то перший рядок вважатиметься більш важливим для визначення загальної ймовірності порушення.

Ця формула дає можливість враховувати ймовірності для текстового контенту та вагувати його відповідно до важливості.

Модель визначає, чи містить текст ненормативний контент, такий як погрози, образи, непристойності, ненависть на основі ідентичності або відверті висловлювання сексуального характеру.

Модель навчалася на базі даних користувацьких коментарів, яка містить близько 2 мільйонів коментарів, позначених як токсичні.

Етапи методу

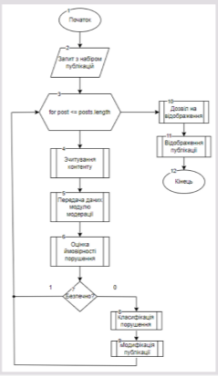

Рисунок Г.8 – Метод та блок-схема алгоритму модерації контенту шляхом використання тренованої моделі ШІ

Метод та блок-схема алгоритму рекомендації шляхом визначення рейтингу публікацій

Необхідно використати комбінацію голосування, взаємодії користувачів та часових факторів для визначення рейтингу публікацій і коментарів на своєму ресурсі.

Наприклад - враховуючи час публікації запису "A" та час 7:46:43 ранку.
8 грудня 2023 "B", ми маємо "t", як їх різницю в секундах:

$$t_0 = A - B$$

і x як різниця між кількістю голосів "за" - U та кількість голосів проти - "D":

$$x = U - D$$

де "y" ∈ {-1,0,1}

$$y = \begin{cases} 1 \text{ якщо } x > 0 \\ 0 \text{ якщо } x = 0 \\ -1 \text{ якщо } x < 0 \end{cases}$$

і "z" як максимальне значення абсолютного значення "x" і "t"

$$z = \begin{cases} |x| \text{ якщо } |x| \geq 1 \\ 1 \text{ якщо } |x| < 1 \end{cases}$$

ми маємо рейтинг, як функцію f(t, y, z)

$$f(t, y, z) = \log_{10} z + \frac{yt}{45000}$$

Етапи методу

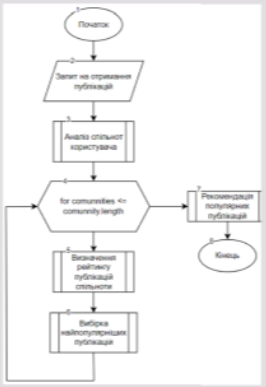

Рисунок Г.9 – Метод та блок-схема алгоритму рекомендації шляхом визначення рейтингу публікацій

Рисунок Г.10 – Проєктування програмного забезпечення

Рисунок Г.11 – Використані технології під час розробки застосунку

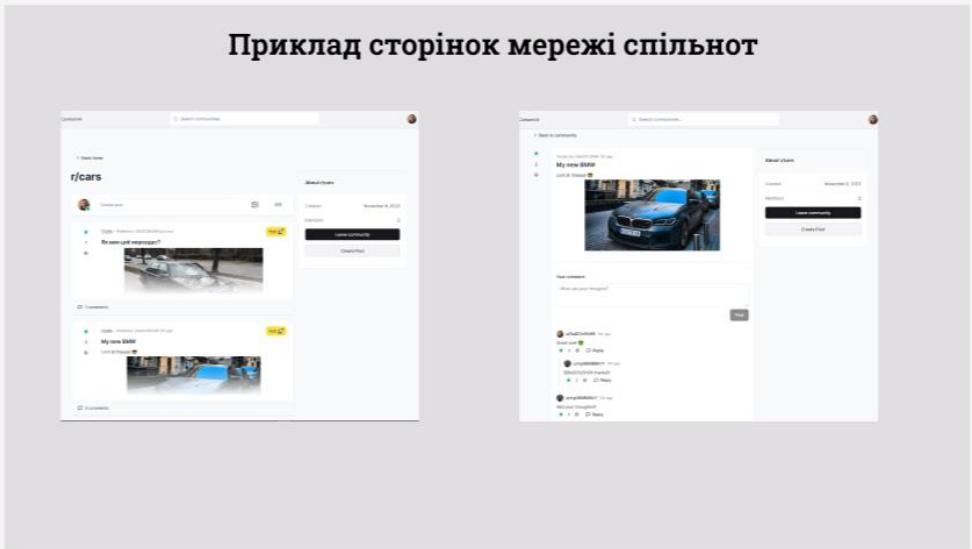

Рисунок Г.12 – Приклад сторінок мережі спільнот

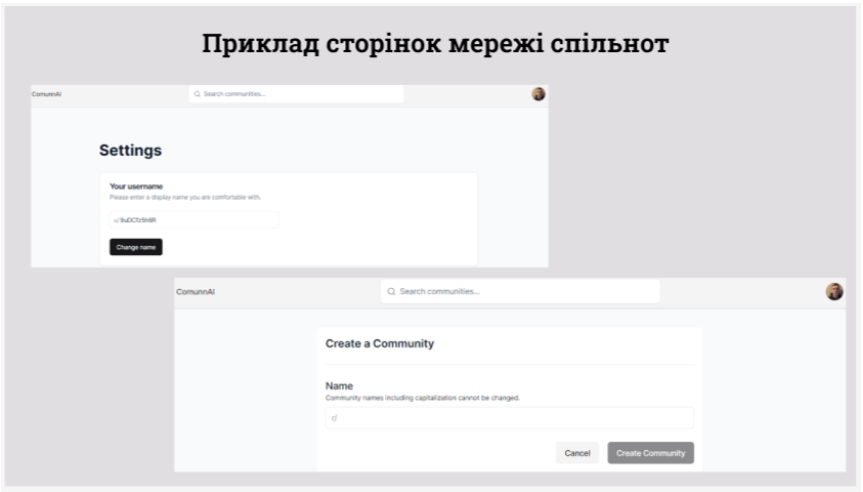

Рисунок Г.13 – Приклад сторінок мережі спільнот

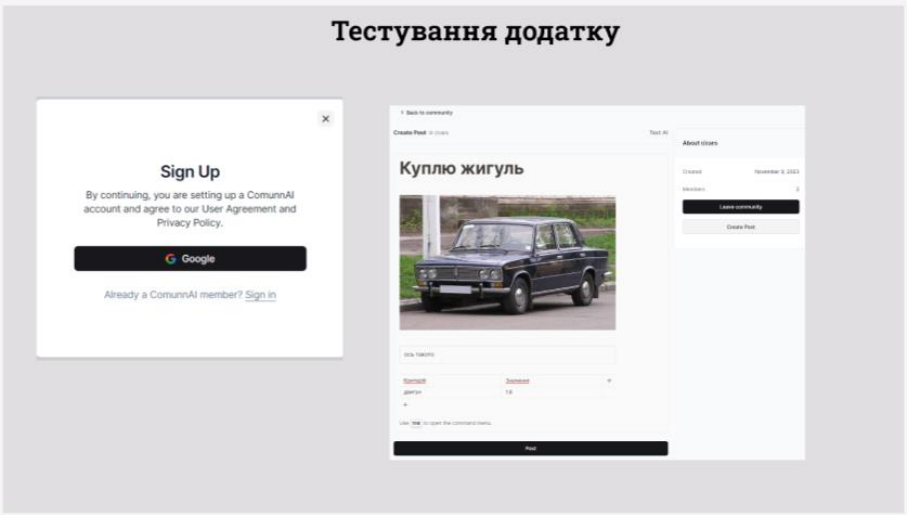

Рисунок Г.14 – Тестування додатку

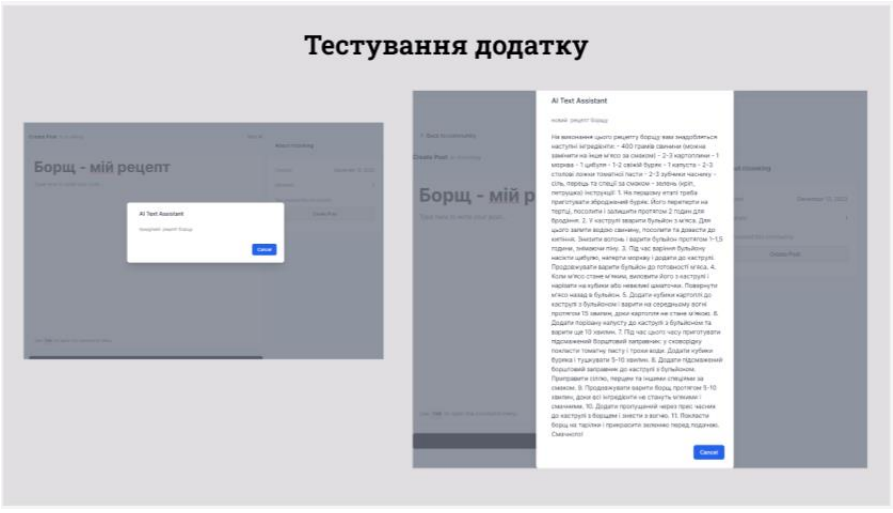

Рисунок Г.15 – Тестування додатку

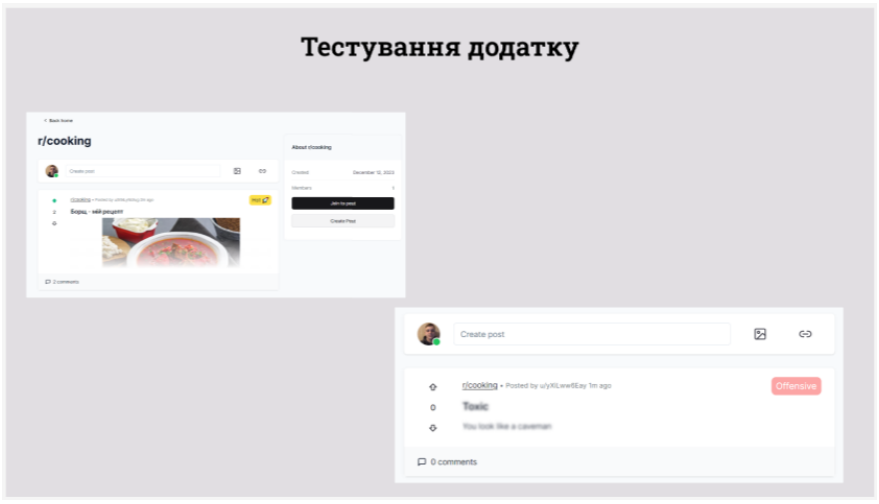

Рисунок Г.16 – Тестування додатку

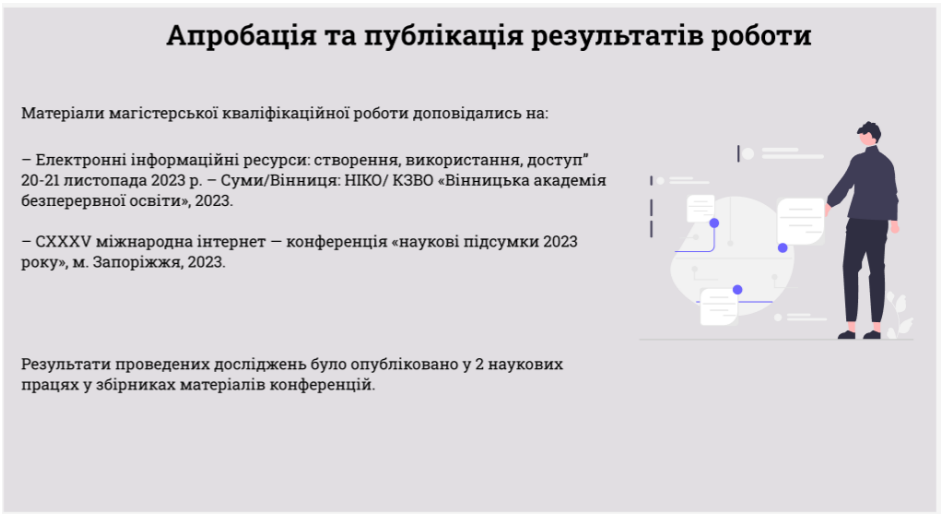

Рисунок Г.17 – Апробація та публікація результатів роботи

Рисунок Г.18 – Кінцевий слайд