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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ СТВОРЕННЯ МЕСЕНДЖЕРА З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент ІІ курсу
групи 1ПІ-22М спеціальності
121 – Інженерія програмного забезпечення
Грабарчук Антон Володимирович

Керівник: к.т.н., доц. каф. ПЗ Майданюк В. П.
«13» грудня 2023 р.

Опонент: к.т.н., доц. каф. ОТ Кожемяко А. В.
«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
Романюк О. Н.
«19» вересня 2023 р.

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

Грабарчуку Антону Володимировичу

1. Тема роботи – Методи та програмні засоби створення месенджера з елементами штучного інтелекту.

Керівник роботи: Майданюк Володимир Павлович, к.т.н., 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:

Потужність множини користувачів - не менше 2; мінімальна кількість записів в базі даних - не менше 10, база даних - реляційна, мова програмування - веб-орієнтована.

4. Зміст текстової частини: вступ; аналіз існуючих аналогів та обгрунтування задачі; розробка архітектури та графічного інтерфейсу додатку; розробка нейронної мережі та методу отримання повідомлень; програмна реалізація месенджера; тестування програмного додатку; економічна частина; висновки; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність дослідження; мета та завдання; об'єкт, предмет та методи дослідження, наукова новизна, практична цінність отриманих результатів, існуючі системи, недоліки

існуючих систем, метод застосування класифікації повідомлень, розр архітектури додатку, розробка архітектури бази даних, математична мо алгоритму розпізнавання природної мови, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-5	Майданюк В. П., к.т.н., доцент каф. ПЗ	19.09.2023	05.12.2023
6	Причепа І. В. к.е.н., доцент каф. ЕПВМ	17.11.2023	25.11.2023

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Обгрунтування вибору методу розробки та постановка задач	20.09.23 – 02.10.23	вик
2	Розробка архітектури та алгоритмів програмного продукту	03.10.23 – 23.10.23	вик
3	Розробка архітектури нейронної мережі	16.10.23 – 23.10.23	вик
4	Розробка програмного продукту	24.10.23 – 13.11.23	вик
5	Тестування програми	14.11.23 – 21.11.23	вик
6	Розробка економічної частини	17.11.23 –25.11.23	вик
7	Оформлення матеріалів до захисту МКР	22.11.23 – 01.12.23	вик

Студент (підпис) **Грабарчук А. В.** (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи (підпис) **Майданюк В. П.** (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Грабарчук А.В. Методи та програмні засоби створення месенджера з елементами штучного інтелекту: магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 100 с.

На укр. мові. Бібліогр. : 31 назв ; рис. : 28; табл. 19.

У магістерській кваліфікаційній роботі розроблено месенджер з інтеграцією штучного інтелекту, штучний інтелект використовується для декількох функцій, перша, це використання нейронної мережі для пріоритезації повідомлень, на важливі та не важливі й таким чином виведення важливих першими, друга це визначення, чи не є повідомлення шахрайськими.

Розробка виконана мовами програмування JavaScript, Typescript та Python. З використанням бібліотек React, Nest.js, TensorFlow. База даних реляційна PostgreSQL, з використанням мови SQL Розроблений застосунок є простим та надійним у використанні, повністю задовольняючи при цьому всі висунуті до нього вимоги. Додаток був протестований, та перевірений на відсутність багів, в ході тестування констатовано досягнення мети роботи.

Отримані результати можуть бути використати для створення схожих застосунків в яких необхідна автоматизація обробки та пріоритезації повідомлень.

Ключові слова: Machine learning, deep learning, natural language processing, месенджер.

ANNOTATION

Hrabarchuk A.V. "Methods and Software Tools for Creating a Messenger with Elements of Artificial Intelligence: Master's Qualification Work in the Specialty 121 - Software Engineering, Educational Program - Software Engineering. Vinnytsia: VNTU, 2023. 100 p.

In English. Bibliography: 31 titles; fig.: 28; tab. 19.

This master's qualification work presents the development of a messenger with the integration of artificial intelligence. Artificial intelligence is utilized for several functions. Firstly, a neural network is employed to classify messages into important and unimportant categories, presenting important messages first. Secondly, the neural network identifies whether a message may be fraudulent. The developed application is user-friendly, meeting all specified requirements.

The development was done using the programming languages JavaScript, TypeScript, and Python, utilizing the React, Nest.js, and TensorFlow libraries. The relational PostgreSQL database was employed with the SQL language. The developed application is simple and user-friendly, fully meeting all specified requirements. The application has been tested and verified for the absence of bugs, and the testing process confirmed the achievement of the project's objectives.

The obtained results can be utilized for creating similar applications where automation of message processing and prioritization is necessary.

Keywords: web system, crowdfunding, crowdfunding platform, crowdfunding campaign, natural language processing, artificial intelligence.

Keywords: Machine learning, deep learning, natural language processing, messenger.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі, на фоні стрімкого розвитку інформаційних технологій та зростаючої ролі комунікаційних платформ, штучний інтелект визначає новий рівень функціональності та взаємодії у сфері месенджерів. Завдяки небаченим досі можливостям аналізу даних, машинного навчання та обробки природної мови, використання штучного інтелекту у месенджерах стає ключовим фактором у вдосконаленні способів спілкування, надаючи користувачам інноваційні та персоналізовані рішення.

Вивчення цієї теми має стратегічне значення для розвитку месенджерів, оскільки дозволяє не лише поліпшити користувацький досвід, а й створює нові можливості для бізнесу та соціальної взаємодії.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно з планом виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення рівня автоматизації пріоритезації повідомлень та виявлення спаму шляхом застосування методів штучного інтелекту.

Основними задачами дослідження є:

- Проаналізувати поняття обміну повідомленнями та месенджерів.
- Провести аналіз існуючих систем обміну повідомленнями.
- Спроєктувати архітектуру розроблюваної системи.
- Обрати тип штучного інтелекту.
- Провести тренування алгоритму машинного навчання.
- Розробити дизайн системи.
- Розробити код додатка.
- Провести тестування створеної системи.

- Виконати розрахунки доведення економічної доцільності розробки.

Об’єктом дослідження - процес обміну повідомленнями в Інтернет.

Предметом дослідження – методи та програмні засоби створення месенджера з елементами штучного інтелекту.

Методи дослідження. У процесі досліджень використовувались: основи теорії штучного інтелекту, теорія розпізнавання образів, теорія алгоритмів, методи WEB-програмування для розробки моделей та методів побудови архітектури програмного забезпечення месенджера з елементами штучного інтелекту; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Вперше запропоновано метод класифікації повідомлень в месенджерах, особливістю якого є застосування штучного інтелекту для пріоритезації повідомлень та виявлення шкідливих, що підвищує зручність використання месенджера.

2. Подальшого розвитку отримав метод обробки текстової інформації, у якому, на відміну від існуючих, використано NLP технологію, що дозволяє сформувати векторний простір для подальшої пріоритезації повідомлень.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби підвищення ефективності використання месенджерів і захисту користувачів.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: використання штучного інтелекту для розпізнавання тексту в зображеннях [1];

класифікація повідомлень з використанням NLP [2].

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях: Науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ) (Вінниця, 2022 р.); Міжнародна науково-практична конференція «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023).

Публікації. Основні результати досліджень опубліковано в 2 наукових працях у матеріалах конференцій.

1. АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ ТА ОБҐРУНТУВАННЯ ЗАДАЧІ

1.1 Аналіз сучасного стану питання

Месенджери стали невід’ємною частиною сучасного спілкування і вплинули на наше повсякденне життя в ряді способів:

- Швидке та зручне спілкування: Месенджери надають можливість обмінюватися повідомленнями в режимі реального часу, що дозволяє швидко та зручно спілкуватися навіть на великі відстані.
- Мультимедійний контент: Вони дозволяють відправляти не лише текстові повідомлення, але й мультимедійний контент, такий як фотографії, відео, аудіозаписи та документи.
- Груповий чат: Функції групового чату у месенджерах забезпечують зручне обговорення та спільне планування подій для групи людей.
- Віртуальні виклики та відеодзвінки: Багато месенджерів дозволяють здійснювати аудіо- та відеодзвінки, що допомагає відчувати близькість у віддалених розташуваннях.
- Інтеграція з іншими сервісами: Месенджери нерідко інтегруються з іншими сервісами, такими як платежі, бронювання, а також із соціальними мережами.
- Збереження історії чатів: Вони автоматично зберігають історію переписки, що робить можливим повернення до попередніх повідомлень та інформації.
- Безкоштовне спілкування через Інтернет: Можливість взаємодії безкоштовно через Інтернет зменшила залежність від традиційних мобільних операторів для обміну повідомленнями.
- Безпека і приватність: Деякі месенджери надають зашифровані чати та інші засоби для збереження приватності користувачів.

Месенджери значно полегшили і розширили можливості спілкування, забезпечуючи широкий спектр функцій, які відповідають потребам сучасного користувача. Вони стали не лише інструментом для обміну повідомленнями, але й платформою для віртуального життя, роблячи наше спілкування більш різноманітним та доступним.

Історія створення та розвитку месенджерів є захоплюючою подорожжю, яка розпочалася зі спроб полегшити спілкування між людьми. Від початку телефонії до сучасних інноваційних месенджерів, ця еволюція визначала зміни в способах, якими ми взаємодіємо один з одним.

SMS і MMS (до 1990-х): Перші етапи месенджерів пов'язані з короткими текстовими повідомленнями (SMS) та повідомленнями з мультимедійним вмістом (MMS). SMS став популярним засобом спілкування через мобільні телефони, але він мав обмежений обсяг символів.

ICQ та AIM (1990-2000): З появою інтернету месенджери для особистих комп'ютерів стали популярними. ICQ (I Seek You) та AOL Instant Messenger (AIM) входять в історію як одні з перших інтернет-месенджерів, що надавали можливість спілкування в режимі реального часу.

MSN Messenger та Yahoo Messenger (2000-2010): MSN Messenger та Yahoo Messenger стали ключовими гравцями на ринку месенджерів у першому десятилітті 2000-х. Вони підняли планку, надаючи можливості групового чату, файлового обміну та інші функції.

Skype (2003): Skype з'явився як програма для голосового та відеочату. Він відрізнявся високою якістю аудіо та відео зв'язку і став дуже популярним серед користувачів для віртуальних зустрічей.

WhatsApp (2009): WhatsApp визначив новий етап розвитку месенджерів, надаючи безкоштовне обмін текстовими повідомленнями через Інтернет. Його

простий інтерфейс та широкий охоплення зробили його одним із найпопулярніших месенджерів у світі.

Viber, Telegram, і Signal (2010-нині): Інші месенджери, такі як Viber, Telegram і Signal, вибралися на фоні конкуренції, пропонуючи різні функції та забезпечуючи більшу приватність та безпеку.

Facebook Messenger, iMessage (2010-нині): Соціальні мережі також розширюють свої можливості у сфері месенджерів. Facebook Messenger став окремим застосунком, а iMessage входить у стандартні можливості iOS-пристроїв.

Сучасні інновації (нині): Сучасні месенджери активно впроваджують інновації, такі як зашифровані чати, віртуальні асистенти, групові відеодзвінки, віртуальні стікери та багато іншого.

Месенджери і їхні функції продовжують швидко розвиватися, пристосовуючись до сучасних потреб користувачів і впливу технологічних інновацій.

Штучний інтелект (artificial intelligence, AI) — це метод змусити комп'ютер чи програмне забезпечення «мислити» як людський мозок. Це досягається шляхом вивчення закономірностей роботи людського мозку та аналізу когнітивних процесів. Результатом цих досліджень є розробка інтелектуального програмного забезпечення та систем [3].

Аналіз сучасного стану питання розробки месенджера з елементами штучного інтелекту допоможе визначити поточні тенденції та виклики у цій області. Використання сканеру відбитків пальців, Face ID у телефоні та додатку Дія, друкування тексту з допомогою T9, спілкування із чат-ботом у якомусь онлайн-магазині — взаємодія з artificial intelligence. Ще приклади штучного інтелекту: ChatGPT, голосові помічники Siri чи Alexa, система «Розумний будинок», автопілоти в машинах тощо[4].

Сучасні месенджери починають використовувати штучний інтелект для покращення користувацького досвіду. Це може охоплювати автоматичні відповіді, чат-боти для обслуговування клієнтів, функції автоматичного перекладу та багато інших функцій, що розширюють можливості користувача.

ШІ може допомогти аналізувати поведінку користувачів та надавати персоналізовані рекомендації щодо вмісту або послуг. Це стає все важливішим аспектом для залучення користувачів. Варіативність персоналізації на основі активності користувача покращить користувацький досвід та зробить будь-який додаток зручнішим для більшої вибірки користувачів.

Аналіз тексту та обробка мовлення за допомогою ШІ дозволяє автоматизувати визначення ключових слів, тем, настроїв користувачів та інших факторів, які можуть бути використані для покращення взаємодії.

Сучасні розробники месенджерів з ШІ ретельно розглядають питання безпеки та конфіденційності даних. Важливо забезпечити, щоб особисті дані користувачів були належним чином захищені. Також важливо враховувати етичні питання, пов'язані з використанням ШІ в месенджерах. Яскравим прикладом є цензурування неетичного контенту спрямованого на розпалювання ненависті та дискредитації певних осіб. Завдяки ШІ, цензурування некоректних висловів стає більш гнучким та дозволяє швидше блокувати образливі повідомлення в групових чатах та збирати статистику про порушників правил використання сервісу.

Конкурентність є одним з найголовніших факторів на ринку споживачів. Ринок месенджерів є досить конкурентним. Популярність і успішність нового месенджера можуть значно залежати від того, наскільки він виходить за межі традиційних месенджерів і надає додаткові функції завдяки ШІ. Ексклюзивність та задоволення потреб користувача робить додаток більш популярним. Критикувати

Таким чином актуальним є питання розробки месенджеру з елементами штучного інтелекту.

1.2 Огляд існуючих аналогів

Штучний інтелект став невід'ємною частиною багатьох сучасних месенджерів, які використовують ШІ для покращення функціональності та користувацького досвіду. Можливості залучення ШІ до своїх проєктів були реалізовані великими продуктовими компаніями ще на самому початку розвитку ШІ, через що більшість месенджерів вже давно мають велику кількість функцій автоматизації:

- Facebook Messenger: це соціальна мережа, яка дозволяє людям спілкуватися, ділитися інформацією та створювати спільноти. Вона була заснована Марком Цукербергом у 2004 році і швидко стала однією з найпопулярніших соціальних мереж у світі. Месенджер цієї соціальної мережі використовує ботів і ШІ для автоматизації відповідей на повідомлення, створення чат-ботів для бізнесів та запуску ігор.
- WhatsApp: це безкоштовний месенджер, який дозволяє користувачам надсилати текстові повідомлення, фотографії, відео та голосові повідомлення іншим користувачам, які також мають WhatsApp. Він був заснований Брайаном Ектоном і Яном Кумом у 2009 році і швидко став одним з найпопулярніших месенджерів у світі. WhatsApp, який також належить Facebook, використовує ШІ для автоматичного перекладу повідомлень, а також для покращення функціональності групових чатів.
- Telegram: це месенджер, який дозволяє користувачам надсилати текстові повідомлення, фотографії, відео, голосові повідомлення, файли та створювати групи та канали. Telegram володіє чат-ботами, які дозволяють розробникам створювати різноманітні боти для різних цілей, від новин до ігор.
- WeChat: це мобільна платформа для обміну повідомленнями, соціальних медіа та електронної комерції, розроблена компанією Tencent. WeChat використовує

ШІ для запуску чат-ботів та розширення функціональності, включаючи послуги замовлення продуктів та послуг.

- Slack: це хмарна платформа для обміну повідомленнями, яка дозволяє командам спілкуватися, співпрацювати та отримувати доступ до інформації в одному місці. Slack, популярний серед команд для спільної роботи, використовує ШІ для автоматизації завдань, сповіщень та іншого.
- Microsoft Teams: це платформа для роботи в командах, яка об'єднує спілкування, зустрічі, файли, завдання та інші інструменти для спільної роботи в єдиному просторі. Microsoft Teams, який також використовується для спільної роботи, має інтеграцію з ШІ для автоматизації рутинних операцій та створення чат-ботів.
- Skype: це програма, яка дозволяє людям спілкуватися один з одним за допомогою голосових, відео та текстових повідомлень. Skype використовує ШІ для автоматичного перекладу повідомлень та аудіо-дзвінків.

Аналіз існуючих аналогів показав, що жоден з аналогів не використовує штучний інтелект для класифікації повідомлень по важливості і захисту від шкідливих листів.

1.3 Обґрунтування використання штучного інтелекту

Враховуючи вище перерахований аналіз існуючих месенджерів, можемо стверджувати, що штучний інтелект – необхідний елемент сучасного месенджера, котрий здатен внести покращення функціональності та конкурентоспроможності. ШІ дозволяє автоматизувати велику кількість завдань, підвищити ефективність використання месенджера.

Для активних користувачів аналогів є актуальною проблемою велика кількість повідомлень, через які можна пропустити дійсно важливу інформацію, також не менш важливим є захист користувачів від спаму і шкідливих повідомлень,

в той час, як інша популярна технологія ШІ, комп’ютерний зір[1] є недоречною у використанні[2].

Штучний інтелект може допомогти в пріоритезації. Це стає ключовим аспектом взаємодії з користувачами та збереження їхнього інтересу до проекта. З плином часу ШІ в месенджерах стає стандартом, і месенджери без ШІ можуть втрачати конкурентоспроможність на ринку. Розробка месенджера з ШІ допоможе займати підходящу позицію в цьому конкурентному середовищі.

Зростання популярності чат-ботів, які базуються на ШІ, стають популярними для бізнесів та користувачів. Вони можуть надавати послуги обслуговування клієнтів, автоматизовані операції та інші корисні функції. Розвиток ШІ продовжується, і разом з ним росте потенціал для розширення функціональності месенджерів. ШІ може допомогти вирішувати нові завдання та надавати інноваційні можливості для користувачів.

З огляду на проведений аналіз та відсутність використання ШІ аналогами для аналізу повідомлень з метою пріоритезації повідомлень, розробка месенджера з використанням ШІ має обґрунтоване підґрунтя. Цей проєкт має потенціал покращити взаємодію користувачів, надати конкурентну перевагу та надати інноваційні можливості в сфері месенджерів. Також важливо дотримуватися найвищих стандартів безпеки та етики під час розробки та використання ШІ.

Використання штучного інтелекту (ШІ) в месенджерах обґрунтовано необхідністю з кількох ключових причин. По-перше, ШІ додає значний рівень інтелектуальної функціональності до месенджерів, що поліпшує користувацький досвід. Він дозволяє розуміти та аналізувати поведінку користувачів, робити персоналізовані рекомендації та оптимізувати взаємодію з додатком. Крім того, ШІ забезпечує можливість автоматизувати багато рутинних завдань, такі як відповіді на повідомлення або навіть управління розкладом зустрічей, що спрощує життя користувачів та робить месенджер більш корисним і ефективним. Важливим

аспектом є також можливість використовувати ШІ для аналізу величезної кількості даних та виявлення трендів в комунікації користувачів, що дозволяє розробникам адаптувати та вдосконалювати месенджер відповідно до змінних потреб користувачів. Ще однією важливою перевагою є підтримка користувачів - ШІ може виконувати роль віртуального помічника або служби підтримки, що відповідає на питання та надає допомогу цілодобово. Не менш важливим є факт, що ШІ допомагає забезпечити безпеку месенджера, виявляючи підозрілу активність та фільтруючи спам, що забезпечує високий рівень безпеки та комфорту для користувачів. З усіма цими перевагами використання ШІ в месенджерах стає ключовим для створення продукту, який відповідає сучасним вимогам користувачів і забезпечує їм зручність та ефективність в комунікації.

1.4 Висновки

1. Аналіз сучасних засобів комунікації між людьми показав, що месенджери стали невід’ємною частиною повсякденного життя.

2. Аналіз існуючих аналогів показав, що жоден з аналогів не використовує штучний інтелект для класифікації повідомлень по важливості і захисту від шкідливих листів, що знижує ефективність використання месенджера та захист користувачів.

3. Тому актуальним є розробка нового месенджера, який би виконував класифікацію повідомлень за важливістю, що сприяло б підвищенню зручності використання месенджера.

2. РОЗРОБКА АРХІТЕКТУР ТА ГРАФІЧНОГО ІНТЕРФЕЙСУ ДОДАТКУ

2.1 Визначення необхідних етапів розробки та методології роботи

Оскільки наш месенджер є вебсистемою то його розробка потребує виконання всіх тих самих етапів, що й інших вебсистем

Розробка веб-системи - це процес, який складається з п'яти етапів[5]:

1. Аналіз вимог і планування. На цьому етапі розробники розуміють, що потрібно зробити, скільки часу і ресурсів це займе. Вони збирають і аналізують вимоги клієнта, розробляють специфікацію вимог, планують проект і визначають технічні вимоги.

2. Проектування. На цьому етапі розробники створюють дизайн і архітектуру системи, вибирають технології і розробляють прототипи. Вони створюють детальний план розробки і прототипи сторінок веб-сайту.

3. Розробка. На цьому етапі розробники пишуть код і створюють функціональні можливості системи. Вони дотримуються кращих практик програмування і використовують найбільш ефективні технології.

4. Тестування. На цьому етапі перевіряється правильність роботи системи, виконуються різні тести на безпеку, швидкість і надійність.

Отже, для досягнення результату нам необхідно виконати всі ці умови.

Тепер визначимо методологію розробки.

Методологія програмного забезпечення - це спосіб розробки програмного забезпечення, який допомагає зробити його якісним, ефективним і надійним.

Основні причини використання методологій програмного забезпечення[6]:

- Покращення якості: Методології допомагають виявити і виправити помилки на ранніх стадіях розробки, що робить програмне забезпечення більш якісним.

- Ефективність: Методології допомагають оптимізувати використання часу і ресурсів, що знижує вартість розробки.
- Надійність: Методології допомагають зробити програмне забезпечення більш надійним, зменшуючи ймовірність виникнення помилок в процесі його використання.
- Спільна робота: Методології допомагають команді розробників працювати разом більш ефективно, полегшуючи комунікацію.
- Підтримка та розвиток: Методології допомагають зробити програмне забезпечення більш простим для розуміння і підтримки в майбутньому.

Основні методології розробки програмного забезпечення

Існує безліч методологій розробки програмного забезпечення (ПЗ), кожна з яких має свої переваги та недоліки. До основних методологій можна віднести:

- Водоспад.
- Ітераційно-інкрементна розробка.
- Адаптивні методології.

Водоспад - це класична методологія розробки, яка передбачає послідовне виконання етапів: аналіз вимог, проектування, розробка, тестування та впровадження. Ця методологія добре підходить для розробки великих і складних проектів, де вимоги добре відомі і стабільні. Однак, водоспадний підхід може бути неефективним для розробки проектів з невизначеними або мінливими вимогами.

Ітераційно-інкрементна розробка - це методологія, яка передбачає розробку програмного забезпечення в ітераціях, кожна з яких включає в себе цикл розробки, тестування та впровадження. Цей методологія дозволяє розробникам отримувати готовий продукт на кожній ітерації, що дозволяє отримати відгуки користувачів і вносити необхідні зміни. Ітераційно-інкрементна розробка добре підходить для розробки проектів з невизначеними або мінливими вимогами.

Адаптивні методології - це методології, які дозволяють розробникам адаптуватися до змін в ході розробки. До адаптивних методологій відносяться, наприклад, скрам і едж-ін. Адаптивні методології добре підходять для розробки проектів з невизначеними або мінливими вимогами.

Враховуючи переваги та недоліки всіх методологій, було вирішено обрати адаптивну методологію. Її використання будемо проводити на основі скраму.

SCRUM - це адаптивна методологія, яка заснована на циклічному повторенні ітерацій, кожна з яких називається спринтом. Спринт триває зазвичай 1-4 тижні. В рамках кожного спринту розробники працюють над набором задач, які називаються історіями користувача. Історії користувача повинні бути короткими і зрозумілими, щоб вони могли бути завершені в рамках одного спринту.

Скрам має ряд переваг, які роблять його популярним вибором для розробки програмного забезпечення:

- Адаптивність: Скрам дозволяє розробникам адаптуватися до змін в ході розробки.
- Команда: Скрам фокусується на командній роботі і спільному плануванні.
- Прозорість: Скрам вимагає від розробників регулярного спілкування з користувачами і замовниками.

Скрам добре підходить для розробки проектів з невизначеними або мінливими вимогами. Ця методологія дозволяє розробникам отримувати відгуки користувачів і вносити необхідні зміни на ранніх стадіях розробки. Скрам також добре підходить для проектів, де важлива командна робота і прозорість.

Ось кілька прикладів того, як скрам може бути використаний для розробки програмного забезпечення:

- Розробка нового веб-сайту.
- Розробка нового додатка для мобільних пристроїв.

- Розробка нового програмного забезпечення для бізнесу.

2.2 Розробка архітектури месенджера

Для створення месенджера, особливо з використовуванням штучного інтелекту, найкращим вибором буде розробка за допомогою клієнт-серверної архітектури. Цей підхід передбачає відокремлення функціональності мобільного додатку на клієнтську та серверну сторони, що дозволяє забезпечити більшу масштабованість, ефективність та безпеку. Перевагами такої архітектури є масштабованість, вищий рівень безпеки, ефективність, ресурсів сервера вистачає, щоб ефективно використовувати штучний інтелект та не навантажувати пристрій користувача, а також можливість внесення оновлень та сумісність з більшою кількістю пристроїв[7] Архітектуру проєкту зображено на рисунку 2.1.

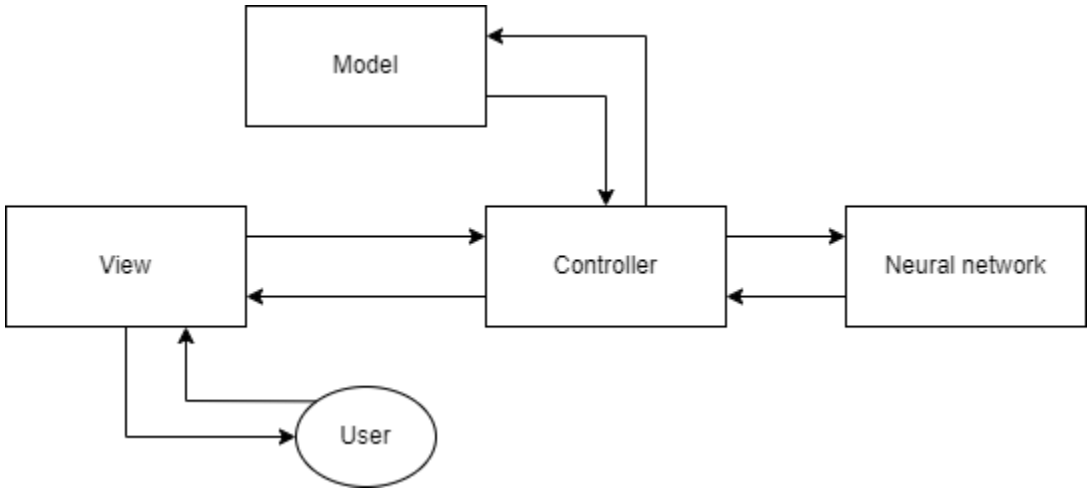


Рисунок 2.1 – Архітектура месенджера

Окрім клієнт-серверної архітектури було вирішено використовувати мікросервісну архітектуру. Мікросервісна архітектура - це підхід до розробки програмного забезпечення, при якому програма розбивається на невеликі незалежні компоненти, які називаються мікросервісами. Кожен мікросервіс відповідає за

виконання певної обмеженої функціональності та має своє власне API для взаємодії з іншими мікросервісами[8].

Клієнтська частина проекту і серверна частина (зазвичай, хмарний сервер) можуть розгортатися окремо. Це дозволяє легко масштабувати серверну інфраструктуру для обробки багатьох користувачів і підтримувати високий рівень завантаження.

Клієнт-серверна архітектура дозволяє зберігати конфіденційні дані на сервері, а не на самому пристрої користувача. Це допомагає захищати дані користувачів від втрати або крадіжки. Якщо використовувати штучний інтелект для аналізу даних, сервер може забезпечити необхідну потужність для цього, тоді як клієнтська частина може забезпечити відображення результатів на пристрої користувача.

Ця архітектура спрощує процес оновлення програмного забезпечення, оскільки ви можете вносити зміни на серверній стороні без необхідності оновлювати кожен окремий мобільний додаток на пристроях користувачів.

В наш час хмарні технології стають все доступнішими для залучення в малих проектах та покращують процес розробки мобільного додатку. Багато платформ і інструментів, таких як Firebase, AWS (Amazon Web Services), та Azure, надають зручні засоби для створення серверної інфраструктури для мобільних додатків[9].

Тож було прийнято рішення використовувати два сервера, один з який створений спеціально для розміщення нейронної мережі, реалізований мовою Python, що забезпечить кращу маштабізацію проекту та покращить обмін вхідними та вихідними даними з нейронної мережі.

Тепер необхідно обрати підхід до створення бекенду котрий виконуватиме запити, а саме обрати чи буде наш сервер Stateful, чи Stateless.

Stateful сервер - це сервер, який зберігає стан між запитами від клієнтів. Це означає, що сервер може зберігати інформацію про користувачів, сеанси, корзини покупок та інші дані, які можуть бути необхідні для обробки запитів.

Stateless сервер - це сервер, який не зберігає стан між запитами від клієнтів. Це означає, що сервер повинен отримувати всю необхідну інформацію з запиту від клієнта.

Stateful сервери зазвичай використовуються для додатків, які вимагають збереження стану між запитами. Наприклад, веб-сайти з корзиною покупок, ігри та соціальні мережі зазвичай використовують stateful сервери.

Stateless сервери зазвичай використовуються для додатків, які не вимагають збереження стану між запитами. Наприклад, веб-сайти з статичним вмістом, API та інші додатків, які не вимагають зберігання будь-якої інформації про користувачів, зазвичай використовують stateless сервери.

Ось деякі з переваг і недоліків stateful і stateless серверів:

Stateful сервери:

Переваги:

- Можуть зберігати стан між запитами, що може бути корисно для додатків, які вимагають цього.
- Можуть бути більш ефективними, ніж stateless сервери, оскільки вони не повинні отримувати всю необхідну інформацію з запиту від клієнта.

Недоліки:

- Можуть бути складнішими у розробці та підтримці, ніж stateless сервери.
- Можуть бути менш масштабованими, ніж stateless сервери, оскільки вони повинні зберігати стан для кожного клієнта.

Stateless сервери:

Переваги:

- Простіші у розробці та підтримці, ніж stateful сервери.
- Більш масштабовані, ніж stateful сервери, оскільки вони не повинні зберігати стан для кожного клієнта.

Недоліки:

- Не можуть зберігати стан між запитами, що може бути обмеженням для деяких додатків.
- Можуть бути менш ефективними, ніж stateful сервери, оскільки вони повинні отримувати всю необхідну інформацію з запиту від клієнта.

Отже, враховуючи особливості нашого додатку, недоліки Stateful сервера не будуть відчутними для нас, в той час, як використання даного підходу пришвидшить розробку і зробить код більш зрозумілим для читання.

2.3 Розробка бази даних

База даних - це систематична колекція даних, які зберігаються та обробляються за допомогою комп'ютерної програми. Бази даних використовуються для зберігання різноманітних типів даних, таких як текст, числа, графіки та зображення.

База даних складається з трьох основних компонентів[10]:

- Дані: Дані - це фактична інформація, яка зберігається в базі даних. Дані можуть бути текстовими, числовими, графічними або в іншому вигляді.
- Структура: Структура визначає, як дані зберігаються в базі даних. Структура може бути простою або складною, залежно від типу даних, які зберігаються в базі даних.
- Програмне забезпечення: Програмне забезпечення - це програма, яка використовується для доступу до даних у базі даних та їх обробки.

Тепер необхідно обрати базу даних. Бази даних поділяються на 2 основних типи, а саме SQL та NoSQL, розглянемо їх.

SQL бази даних - це тип баз даних, які використовують мову структурованих запитів (SQL) для доступу до даних. SQL - це декларативна мова програмування, яка означає, що ви описуєте, що хочете зробити з даними, а не те, як це зробити.

SQL бази даних складаються з таблиць, які містять дані. Кожна таблиця має рядки (записи) та стовпці (поля). Рядки представляють окремі одиниці даних, а стовпці представляють типи даних, які містяться в кожному рядку[11].

Використовувати SQL можна для виконання різних операцій із даними в SQL базі даних, таких як:

- Вставка даних у таблицю
- Оновлення даних у таблиці
- Видалення даних із таблиці
- Запитування даних із таблиці

SQL бази даних широко використовуються в різних галузях, включаючи бізнес, уряд та освіту. Вони є популярним вибором для зберігання великих обсягів даних, оскільки вони ефективні та надійні.

Переваги використання SQL баз даних:

- Ефективність: SQL бази даних є ефективними для зберігання та доступу до даних.
- Надійність: SQL бази даних є надійними та можуть зберігати великі обсяги даних.
- Розширюваність: SQL бази дані можна легко розширювати, щоб підтримувати зростання обсягів даних.

Недоліки використання SQL баз даних:

- Складність: SQL є складною мовою програмування, яку може бути важко вивчити.
- Безпека: SQL бази дані можуть бути вразливими до атак, якщо вони не захищені належним чином.
- Вартість: SQL бази дані можуть бути дорогими в експлуатації, особливо для великих баз даних.

NoSQL бази даних - це тип баз даних, які не використовують мову структурованих запитів (SQL) для доступу до даних. NoSQL бази даних зазвичай використовують більш гнучкі моделі даних, ніж SQL бази даних, що дозволяє їм краще підтримувати різні типи даних та запитів.

NoSQL бази даних часто використовуються для зберігання великих обсягів неструктурованих або слабо структурованих даних. Вони також можуть бути корисні для зберігання даних, які часто змінюються або які потрібно обробляти швидко.

Переваги використання NoSQL баз даних:

- Гнучкість: NoSQL бази дані дозволяють зберігати різні типи даних та запитів.
- Швидкість: NoSQL бази дані можуть бути ефективними для обробки даних, які часто змінюються або які потрібно обробляти швидко.
- Розширюваність: NoSQL бази дані можна легко розширювати, щоб підтримувати зростання обсягів даних.

Недоліки використання NoSQL баз даних:

- Складність: NoSQL бази дані можуть бути складними для вивчення та використання.
- Безпека: NoSQL бази дані можуть бути вразливими до атак, якщо вони не захищені належним чином.
- Вартість: NoSQL бази дані можуть бути дорогими в експлуатації, особливо для великих баз даних.

SQL бази даних є більш поширеними та добре зрозумілими, а також вони добре підходять для виконання складних структурованих запитів. NoSQL бази даних є хорошим вибором для додатків, які вимагають зберігання великих обсягів неструктурованих або слабо структурованих даних. Враховуючи, що наш додаток

не матиме великого обсягу інформації, але вимагатиме виконання складних запитів, було вирішено обрати SQL базу даних.

Тепер необхідно обрати безпосередньо базу даних. Є дві найбільш поширених SQL бази даних, а саме MySQL та PostgreSQL.

MySQL та PostgreSQL - це дві найпопулярніші реляційні бази даних (RDBMS) у світі. Вони обидві підтримують мову SQL для доступу до даних, але між ними є деякі ключові відмінності.

MySQL - це відкрите програмне забезпечення, розроблене компанією Oracle. Воно є популярним вибором для веб-додатків, оскільки воно є швидким і легким у використанні. MySQL підтримує широкий спектр функцій, включаючи:

- Реляційна модель даних
- Мова SQL
- Підтримка транзакцій
- Автоматичне відновлення
- Швидкість і масштабованість

PostgreSQL також є відкритим програмним забезпеченням, але його розробляє некомерційна організація PostgreSQL Global Development Group. PostgreSQL підтримує більше функцій, ніж MySQL, включаючи:

- Об'єктно-реляційна модель даних
- Підтримка транзакцій ACID
- Автоматичне відновлення
- Концепція ролей
- Розширюваність

Переваги MySQL:

- Швидкість: MySQL є одним із найшвидших RDBMS на ринку. Це пов'язано з тим, що він використовує оптимізований механізм зберігання та індексування.

- Легкість використання: MySQL є відносно простим у використанні. Це пов'язано з тим, що він має невелику кількість функцій та опцій.
- Вартість: MySQL є безкоштовним для використання та поширення.

Недоліки MySQL

- Обмежена функціональність: MySQL не підтримує всі функції, які доступні в інших RDBMS, таких як PostgreSQL.
- Безпека: MySQL може бути менш безпечним, ніж інші RDBMS, якщо його не налаштувати належним чином.
- Розширюваність: MySQL може бути менш масштабованим, ніж інші RDBMS, при обробці великих обсягів даних.

Переваги PostgreSQL:

- Функціональність: PostgreSQL підтримує більше функцій, ніж MySQL. Це робить його більш гнучким і придатним для широкого спектру додатків.
- Безпека: PostgreSQL підтримує більше функцій безпеки, ніж MySQL. Це допомагає захистити дані від несанкціонованого доступу.
- Розширюваність: PostgreSQL є більш масштабованим, ніж MySQL, при обробці великих обсягів даних.

Недоліки PostgreSQL

- Швидкість: PostgreSQL може бути менш швидким, ніж MySQL, для простих запитів.
- Складність: PostgreSQL може бути складнішим у використанні, ніж MySQL, оскільки він має більше функцій і опцій.
- Вартість: PostgreSQL є безкоштовним для використання та поширення, але може вимагати додаткових витрат на підтримку та розробку.

Враховуючи переваги та недоліки обох СУБД було вирішено обрати PostgreSQL, завдяки його функціональності, безпеці та масштабованості ця СУБД є безумовно найкращим рішенням для нашого додатка. Котре надасть найкращу

функціональність для досягнення цілей з побудови бази даних для нашого проєкту з створення месенджера з елементами штучного інтелекту.

Розглянемо таблиці бази даних(Рис. 2.2). Як можемо спостерігати у нас є 3 таблиці:

- Users - таблиця котра містить інформацію про користувачів;
- Chats - таблиця котра містить інформацію про чати. По суті, виконує роль проміжної таблиці для зв’язку n:n між користувачами і повідомленнями;
- Messages - таблиця котра містить інформацію про повідомлення.

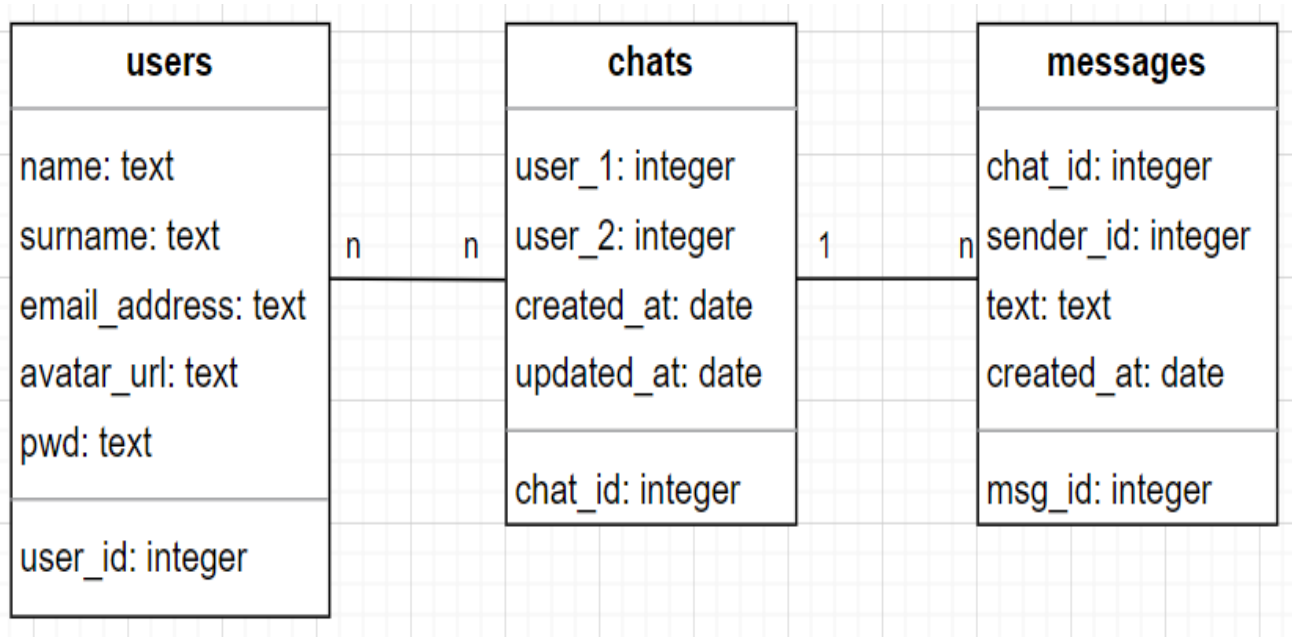


Рисунок 2.2 - Таблиці додатку

Розглянемо детальніше таблиці й інформацію котру вони містять. Почнемо з таблиці users(Таб. 2.1)

Таблиця 2.1 - Таблиця Users

Назва	Тип	Опис
user_id	integer(первинний ключ)	Унікальний ідентифікатор користувача
name	text	Ім’я користувача
surname	text	Прізвище користувача
email_adress	text	Емейл адреса користувача (є унікальним значенням)
avatar_url	text	Посилання на фото користувача
pwd	text	Пароль користувача

Тепер розглянемо таблицю Chats(Таб. 2.2)

Таблиця 2.2 - Таблиця Chats

Назва	Тип	Опис
chat_id	Integer(первинний ключ)	Унікальний ідентифікатор чату
user_1	integer (зовнішній ключ)	Унікальний ідентифікатор користувача
user_2	integer(зовнішній ключ)	Унікальний ідентифікатор користувача

Продовження таблиці 2.2.

created_at	date	Час створення чату
updated_at	date	Час крайнього оновлення чату

Тепер розглянемо таблицю Messages(Таб. 2.3)

Таблиця 2.3 - Таблиця Messages

Назва	Тип	Опис
msg_id	Integer(первинний ключ)	Унікальний ідентифікатор повідомлення
chat_id	integer (зовнішній ключ)	Унікальний ідентифікатор чату
sender_id	integer (зовнішній ключ)	Унікальний ідентифікатор користувача
text	text	Текст повідомлення
created_at	date	Час створення повідомлення

Отже, нами була розроблена база даних додатку.

2.4 Розробка графічного інтерфейсу

Графічний інтерфейс (GUI) — це спосіб взаємодії користувача з комп'ютерною програмою чи операційною системою, використовуючи графічні елементи, такі як вікна, ікони та кнопки, замість текстових команд. GUI робить взаємодію із системою більш інтуїтивно зрозумілою для користувача, спрощуючи введення та взаємодію[12].

Першим етапом розробки графічного інтерфейсу розроблюваного проекту стали планування і дизайн. На цьому етапі було визначено функціональні вимоги до інтерфейсу та проведено дизайн-роботи. Було визначено структуру і вигляд додатку, розміщення елементів інтерфейсу, кольорову палітру та шрифти. Користувацький досвід (UX) і дизайн (UI) важливі для створення привабливого та зручного інтерфейсу.

Розглянемо існуючі графічні редактори для вибору найбільш підходящого для нас варіанту.

Adobe Photoshop, Adobe Illustrator, Sketch і Figma - це все популярні програми для створення дизайну, які використовуються для створення веб-сайтів, мобільних додатків, логотипів, інфографіки та інших графічних елементів.

Adobe Photoshop - це растровий графічний редактор, який використовується для редагування фотографій, створення ілюстрацій і дизайну інтерфейсів. Він має широкий спектр функцій, включаючи інструменти для редагування зображень, створення векторних об'єктів, малювання, створення анімації та багато іншого.

Adobe Illustrator - це векторний графічний редактор, який використовується для створення логотипів, ілюстрацій, дизайну інтерфейсів і інших графічних елементів. Він має широкий спектр функцій, включаючи інструменти для створення векторних об'єктів, малювання, створення шрифтів і багато іншого.

Sketch - це векторний графічний редактор, який був розроблений спеціально для створення дизайнів веб-сайтів і мобільних додатків. Він має простий у

використанні інтерфейс і широкий спектр функцій, включаючи інструменти для створення макетів, стилізації елементів, додавання інтерактивних елементів і багато іншого.

Figma - це хмарна графічна програма, яка використовується для створення дизайнів веб-сайтів, мобільних додатків і інших графічних елементів. Вона має широкий спектр функцій, додавання інтерактивних елементів і багато іншого.

Тепер порівняємо переваги та недоліки цих програмних продуктів(Таб. 2.4)

Таблиця 2.4 - Порівняння графічних редакторів

Характеристика	Adobe Photoshop	Adobe Illustrator	Sketch	Figma
Тип графіки	Растрова	Векторна	Векторна	Векторна
Інтерфейс	Настільний	Настільний	Настільний	Хмарний
Функціональність	Редагування зображень, створення векторних об'єктів, малювання, створення анімації	Створення векторних об'єктів, малювання, створення шрифтів	Створення макетів, стилізація елементів, додавання інтерактивних елементів	Створення макетів, стилізація елементів, додавання інтерактивних елементів
Ціна	24,99 доларів США на місяць	29,99 доларів США на місяць	99 доларів США на рік	Безкоштовно для команд до 3 користувачів

Отже, враховуючи переваги та недоліки всіх додатків було вирішено обрати Figma, завдяки її безкоштовності, зручності та наявності вебверсії.

Розробка графічного інтерфейсу була здійснена в мінімалістичному стилі. Мінімалістичний дизайн має декілька переваг для месенджера:

- Мінімалістичний дизайн спрощує користувацький інтерфейс, що робить його легким у сприйнятті. Користувачі з легкістю знаходять потрібні функції та інструменти, не розсіюючись багатьма зайвими деталями.
- Мінімалістичний дизайн дозволяє звернути увагу на контенті і важливих повідомленнях, а не на зайвому візуальному оформленні. Це особливо важливо для месенджера, оскільки основна мета - це комунікація.
- Мінімалістичний інтерфейс вимагає менше ресурсів, що дозволяє програмі працювати швидше і більш стабільно на різних пристроях, включаючи старіші та менш потужні.
- Мінімалістичний дизайн, зазвичай, добре пристосовується до різних платформ і розмірів екрану. Це дозволяє побудувати єдиний дизайн, який може використовуватися як на мобільних пристроях, так і на десктопах.
- Простий і зрозумілий дизайн полегшує користувачам взаємодію з месенджером, що збільшує його зручність. Користувачі легше розуміють, як використовувати додаток.
- Це підвищує ефективність використання месенджера за рахунок зменшення або відсутності необхідності навчання користувача використовувати наш додаток.

Тепер розробімо дизайн нашого додатка використовуючи графічний редактор для створення дизайну figma.

Почнімо зі сторінки логіну та реєстрації(Рис. 2.3).

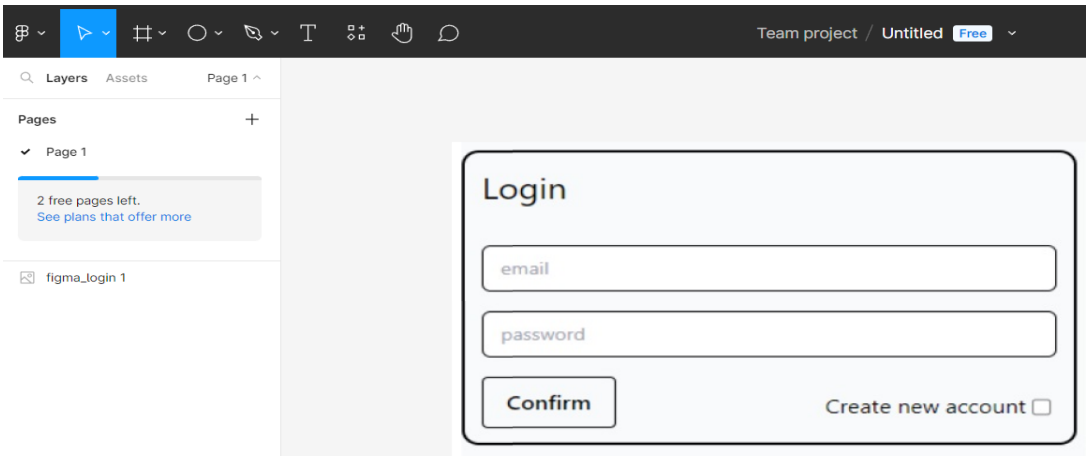


Рисунок 2.3 - Дизайн сторінки логіну та реєстрації

Як можемо бачити на дизайні є два поля, один для емейла інший для пароля, котрі і виступають ідентифікуючою інформацією користувача, щоб зайти в систему користувачу достатньо натиснути кнопку “Confirm”. Якщо ж у користувача немає створеного акаунта, то достатньо лиш поставити галочку в чекбоксі “Create new account” і ввести свої дані в вищеописані поля, таким чином автоматично буде створено новий акаунт, але не відбудеться редіректу на основну сторінку додатку, тому потрібно буде зробити ще раз дії необхідні для логіну.

Далі розробимо дизайн сторінки чату(Рис. 2.4).

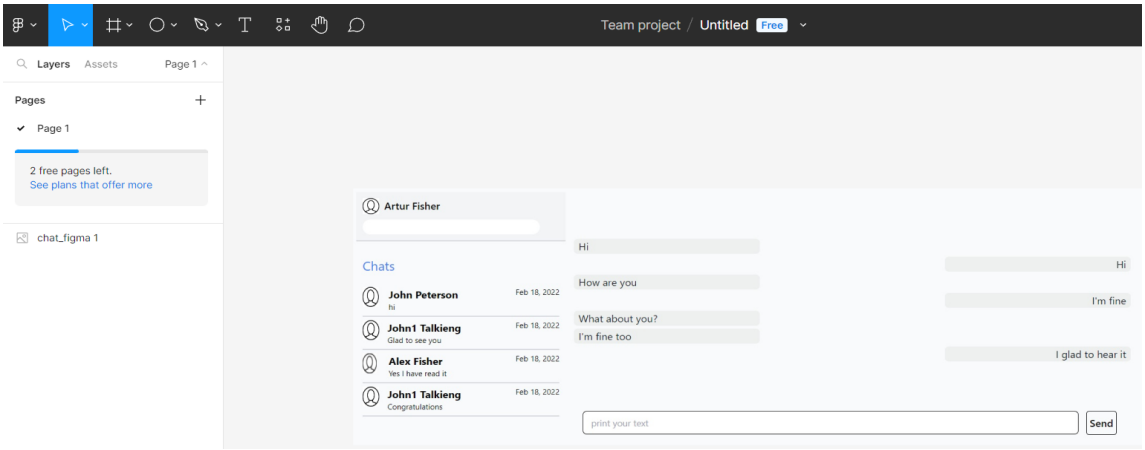


Рисунок 2.4 - Дизайн сторінки чату

Як можемо бачити, зліва у нас список чатів і інформація про користувача з пошуковою стрічкою для пошуку нових користувачів. Справа історія переписки з обраним користувачем, і нижче поле для вводу повідомлення й кнопка відправити котра і відправляє створене користувачем повідомлення.

Отже, ми розробили підхід до створення дизайну додатку, й нами був розроблений сам дизайн додатка, а також опис його функціоналу.

2.5 Розробка алгоритму застосування класифікації повідомлень

Алгоритм - це чітко визначений набір інструкцій або кроків, які виконуються для розв'язання конкретної задачі або виконання певної операції.

Отже, нам необхідно розробити алгоритм роботи нашого додатку, а саме алгоритм класифікації повідомлень.

Основні вимоги до алгоритму, це показ користувачу важливих повідомлень першим, і приховування шкідливих повідомлень, для захисту користувача. Рисунок 2.5.

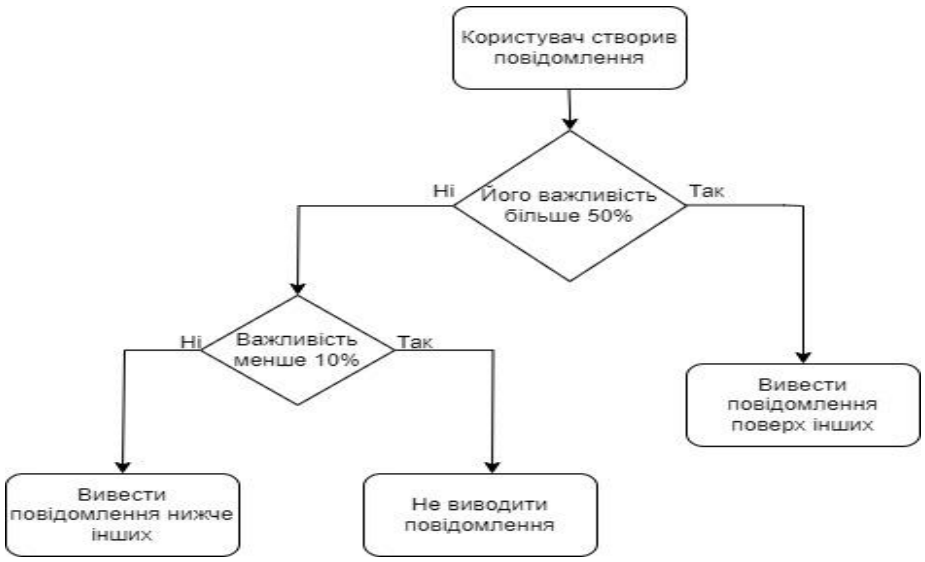


Рисунок 2.5 - Принцип класифікації повідомлень

Отже, ми розробили метод застосування штучного інтелекту класифікації повідомлень й захисту користувача від шкідливих повідомлень та спаму в месенджері.

2.5 Висновки

1. Обрано методологію розробки програмного забезпечення SCRUM, оскільки ця методологія доволі гнучка і орієнтована на інкрементальний підхід до розробки.
2. В якості архітектури додатку обрано архітектуру MVC, оскільки ця архітектура ефективно відокремлює Business Logic від користувальницького інтерфейсу програми.
3. В якості СУБД обрано СУБД PostgreSQL, оскільки ця СУБД характеризується надійністю та відкритим вихідним кодом.
4. З використанням СУБД PostgreSQL розроблено структуру бази даних повідомлень та користувачів.

3 РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ ТА МЕТОДУ ОТРИМАННЯ ПОВІДОМЛЕНЬ

3.1 Визначення необхідних кроків та постановка задачі

Ідея створення машин, які можуть мислити і діяти самостійно, існує вже багато століть. Перші розробки в галузі штучного інтелекту (ШІ) були зроблені в 1950-х роках. У цей час були розроблені перші алгоритми машинного навчання, які дозволяли машинам навчатися на прикладах.

У 1960-х роках ШІ переживав період бурхливого розвитку. Були розроблені нові алгоритми машинного навчання, які були більш ефективними і дозволяли машинам виконувати складні завдання. Однак у 1970-х роках розвиток ШІ призупинився через ряд проблем, включаючи недостатність обчислювальної потужності і брак даних для навчання.

У 1980-х роках ШІ знову почав розвиватися. Були розроблені нові алгоритми машинного навчання, які були більш ефективними і дозволяли машинам виконувати більш складні завдання. Однак, попри ці успіхи, ШІ все ще не міг конкурувати з людським розумом.

У 1990-х роках ШІ став більш доступним для широкого загалу. Були розроблені нові технології, які дозволяли б створювати ШІ-системи більш простими і ефективними. Це призвело до широкого поширення ШІ в різних галузях, включаючи комп'ютерні ігри, розпізнавання образів і машинний переклад.

У 2000-х роках ШІ продовжував розвиватися. Були розроблені нові алгоритми машинного навчання, які були більш ефективними і дозволяли машинам виконувати більш складні завдання. Це призвело до того, що ШІ став здатний виконувати завдання, які раніше вважалися виключно людськими, наприклад, грати в шахи.

У XXI столітті ШІ продовжує розвиватися і ставати все більш потужним. ШІ вже використовується в багатьох галузях, включаючи медицину, виробництво і транспорт. У майбутньому ШІ буде використовуватися в ще більш широкому спектрі областей, включаючи освіту, розваги й управління[13].

Однією з найбільш перспективних областей застосування ШІ є машинне навчання. Машинове навчання дозволяє машинам навчатися на прикладах і самостійно знаходити закономірності в даних. Це дозволяє машинам виконувати завдання, які раніше вважалися неможливими для автоматизації.

Іншою перспективною областю застосування ШІ є штучний інтелект на основі нейронних мереж. Нейронні мережі - це моделі машинного навчання, які імітують роботу людського мозку. Вони дозволяють машинам навчатися на прикладах і виконувати складні завдання, такі як розпізнавання образів і розпізнавання мови.

ШІ має потенціал змінити світ в багатьох аспектах. ШІ може використовуватися для вирішення складних проблем, таких як зміни клімату і бідність. ШІ також може використовуватися для створення нових продуктів і послуг, які покращать наше життя.

Однак, у ШІ також є ряд потенційних ризиків. Наприклад, ШІ може використовуватися для створення автономних збройних систем, які можуть завдати шкоди людям. ШІ також може використовуватися для створення пропаганди і дезінформації.

Важливо усвідомлювати потенційні ризики ШІ і розробити заходи для їх мінімізації. Однак, в цілому, ШІ має потенціал зробити світ кращим.

Створення нейронної мережі - це завдання, яке вимагає великої уваги до деталей та знань у галузі машинного навчання, а також врахування особливостей месенджерів та їхніх користувачів.

Створення нейронної мережі для використання в месенджері можна поділити на основні кроки[14]:

- Спочатку визначається, для якого завдання буде використовуватися нейронна мережа в месенджері. Це може бути автоматичний переклад тексту, розпізнавання емоцій у повідомленнях, фільтрація спаму, автоматична генерація відповідей, сортування повідомлень тощо.
- Залежно від завдання, збираються та підготуються дані для тренування мережі. Наприклад, для навчання мережі для автоматичного перекладу, вам знадобиться паралельний корпус текстів на різних мовах. Для тренування підходять лише перевірені та чисті дані.
- В залежності від обраного завдання, що буде виконувати мережа, обирається архітектура, за якою буде спроектована мережа. Для обробки тексту, рекурентні нейронні мережі (RNN) або трансформери можуть бути корисними. Для обробки зображень - сверточні нейронні мережі (CNN).
- Після вибору архітектури, має бути створена структура мережі, включаючи шари, функції активації та зв'язки. Для обробки тексту це може охоплювати використання векторних представлень слів, як Word2Vec[15] або GloVe. Для зображень - сверточні та пулінгові шари.
- Тренування моделі відбувається використовуючи підготовлені дані. Необхідно визначити функцію втрат та алгоритм оптимізації. Навчання мережі проходить на тренувальних даних та валідація її на тестових даних.
- Після тренування і валідації, вже навчена мережа має бути інтегрована в месенджер. Це може включати в себе створення API для взаємодії з мережею та обробку вхідних і вихідних повідомлень.

Особливу увагу необхідно виділити на безпеку користувацьких даних, особливо якщо мережа обробляє особисту інформацію користувачів.

Враховуючи вимоги і особливості нашого завдання, необхідно зробити наступні кроки:

1. Підготовка датасету;
2. Опрацювання датасету методами NLP;
3. Проведення тренування моделі на отриманих даних;
4. Валідація отриманих результатів.

3.2 Вибір алгоритму обробки природної мови

Natural Language Processing (NLP) - це галузь штучного інтелекту, що спрямована на розуміння та обробку природної мови людей. В основі NLP лежить намагання надати комп'ютерам здатність аналізувати, інтерпретувати та відтворювати мовлення на рівні, зрозумілому для людини. Це має безліч практичних застосувань, включаючи обробку текстових документів, автоматизовані відповіді на запити користувачів, машинний переклад, аналіз настроїв, класифікацію текстів та багато інших[16].

В основі NLP лежать алгоритми та моделі машинного навчання, які навчаються взаємодіяти з текстовою інформацією. До цих моделей включаються рекурентні нейронні мережі (RNN), відомі своєю здатністю аналізувати послідовний текст, а також трансформери, які здатні працювати з контекстом та встановлювати зв'язки між словами та фразами у тексті.

Однією з ключових задач NLP є розуміння тексту, включаючи семантику, синтаксис та смислові зв'язки між словами. Завдяки цьому, системи NLP можуть розрізняти наміри користувачів, що дозволяє створювати інтелектуальні асистенти, які реагують на запити[17].

NLP також використовується для виправлення орфографічних помилок, аналізу тональності текстів, визначення ключових слів та звітності з тексту. Вона допомагає покращити пошукові системи, робити аналітичні висновки з текстової

інформації, а також сприяє автоматизації багатьох видів рутинних завдань, пов'язаних з обробкою тексту.

"Bag of Words" (BoW) є одним з основних алгоритмів у сфері обробки природної мови (Natural Language Processing, NLP), і його важливість при створенні нейронних мереж важко переоцінити. Цей алгоритм дозволяє перетворити текстові дані, які комп'ютери не можуть розуміти напряму, у структуровану форму, яку можна використовувати для аналізу та використання в нейронних мережах.

BoW діє наступним чином: текст розглядається як набір слів, і для кожного слова створюється "мішок" (Bag), де кожен "мішок" представляється у вигляді числа, що відповідає кількості входжень цього слова в текст. Це перетворення дозволяє представити текст у форматі, зрозумілому для нейронних мереж.

BoW важливий для створення нейронної мережі з кількох причин. По-перше, він надає можливість векторизувати текстові дані, що дозволяє їм бути обробленими нейронною мережею. Нейронна мережа працює з числовими даними, тому BoW допомагає перетворити текст в числовий формат.

По-друге, BoW спрощує дані. Велика кількість слів у тексті може призвести до великої розмірності векторів, що може бути проблематичним для нейронної мережі. BoW допомагає зменшити розмірність даних, використовуючи лише важливі слова та їх кількість у тексті.

По-третє, BoW служить важливою основою для подальших методів векторизації тексту. Наприклад, методи, які використовують інформацію про контекст та семантику слів, можуть бути побудовані на основі BoW, дозволяючи нейронній мережі краще розуміти текст.

У розвитку нейронної мережі BoW відіграє роль важливої стартової точки для обробки та аналізу тексту. Він допомагає змінити природну мову в числовий формат, що може бути обробленим нейронною мережею, і створює основу для подальших досліджень та розвитку у галузі NLP. Таким чином, BoW є необхідним

етапом при роботі з текстовими даними в контексті нейронних мереж та важливим засобом для їх розвитку.

3.3 Покращення методу та алгоритму обробки повідомлення з використанням NLP технологій

Хоч існуючі методи обробки людських мов і є хорошими, але все ще мають великі недоліки, зокрема з точністю визначення слів, як було сказано вище принцип роботи алгоритму BOW полягає в тому, щоб представити текстовий документ (наприклад, речення чи документ) у вигляді вектору, враховуючи кількість входжень кожного слова у текст. Тобто ми маємо вектор з нулів і одиниць, де кожна позиція відповідає одному з поширених слів, якщо слово рідкісне, то воно потрапляє в категорію “інші”.

Розглянемо основні недоліки методу Bag of Words:

- Втрата порядку слів: BoW не враховує порядок слів у тексті, втрачаючи важливий синтаксичний контекст та порушуючи семантичне значення.
- Рівноправність слів, Кожне слово у BoW розглядається як незалежний елемент, не враховуючи його важливості або унікальності в контексті тексту.
- Великий обсяг даних, Матриці документ-слів можуть бути об'ємними та розрідженими, потребуючи значних обчислювальних ресурсів та простору для зберігання.

Враховуючи недоліки існуючого методу було вирішено покращити його створивши власний метод на його основі.

В основу методу ляже вимірювання частоти з якою зустрічається певне слово. Для вимірювання, наскільки часто слово зустрічається у конкретному документі, ми використовуємо відношення кількості входжень слова до загальної кількості слів у документі. Математична модель формули наведена в формулі 3.1

$$V = \frac{\text{частота слова в тексті}}{\text{загальна кількість слів}} \quad (3.1)$$

Тепер виміряємо, частоту унікального слова в документі. Зменшення значення для загальних слів допомагає виділяти ключові слова. Математичну модель наведено у формулі 3.2.

$$R = log(\frac{\text{загальна кількість слів}}{\text{частота слова в тексті}}) + 1 \quad (3.2)$$

Значення VR для кожного слова обчислюється як добуток його V та R, формула 3.3:

$$VR = V \times R \quad (3.3)$$

Тепер створюється вектор, який містить значення V-R для кожного слова. Це векторне представлення тексту забезпечує більш точну репрезентацію семантичного змісту та важливості слів.

Отже, Метод V-R вирішує недоліки BoW, забезпечуючи кращу вагову репрезентацію тексту. Його застосування системах обробки природної мови дозволяє ефективніше враховувати семантичний контекст та поліпшує точність аналізу текстової інформації.

3.4 Вибір алгоритму штучного інтелекту для класифікації повідомлень

Після обробки повідомлення через Natural Language Processing використовуючи алгоритм Bag of Words інформація використовується для машинного навчання, котрий і буде класифікувати пріоритетність повідомлень.

Отже, нам необхідно обрати алгоритм машинного навчання, для цього проаналізуємо існуючі алгоритми класифікації.

Класифікація повідомлень є важливою задачею в області обробки природної мови та аналітиці тексту. Надалі ми розглянемо декілька алгоритмів машинного навчання, включаючи Support Vector Machines (SVM), Random Forest, Naive Bayes, та Deep Neural Network (DNN), та оцінимо їх ефективність для даної задачі.

Support Vector Machines (SVM) - це алгоритм машинного навчання, який використовується для класифікації і регресії. Основна ідея полягає в тому, щоб знайти гіперплощину (наприклад, лінію або гіперплощину в багатовимірному просторі), яка найкращим чином розділяє дані на різні класи. Головною метою SVM є максимізація маржі між класами, тобто мінімізація відстані між найближчими точками кожного класу та гіперплощиною, що їх розділяє. Така гіперплощина називається "рішучою гіперплощиною". SVM може працювати як з лінійно роздільними даними, так і з нелінійно роздільними даними, використовуючи метод ядра для перетворення даних в більш високорозмірний простір, де вони стають лінійно роздільними. Цей алгоритм є дуже популярним для завдань класифікації та регресії через його здатність до роботи зі складними та нелінійними залежностями в даних.

Переваги:

- Висока точність класифікації, особливо для лінійно роздільних даних.
- Здатність робити багатокласову класифікацію.
- Відсутність проблеми перенавчання при великій кількості ознак.

Недоліки:

- Відносно повільне навчання для великих наборів даних.
- Складно здійснити вирішення завдань, де залучені тексти невеликих розмірів.

Random Forest - це ансамбль алгоритмів машинного навчання, який базується на деревах рішень. Основна ідея Random Forest полягає в тому, щоб створити багато дерев рішень під час навчання і об'єднати їх для отримання кінцевого класифікаційного або регресійного результату. Кожне дерево рішень навчається на випадковому піднаборі даних та випадковому піднаборі ознак.

Основні особливості Random Forest включають випадковий вибір піднаборів даних і ознак: Під час навчання кожне дерево випадковим чином вибирає піднабір навчальних даних та піднабір ознак для кожного вузла дерева. Це дозволяє зменшити перенавчання та зробити алгоритм стійким до шуму в даних. Багато дерев: Random Forest використовує багато дерев для прийняття рішення. Коли потрібно здійснити класифікацію або регресію, кожне дерево вносить свій внесок у результат, і результат обчислюється шляхом голосування (для класифікації) або середнього значення (для регресії) від усіх дерев.

Бутстрепні вибірки: Для кожного дерева генерується випадковий набір даних шляхом бутстрепу, що дозволяє кожному дереву бачити різні підмножини даних.

Random Forest є потужним алгоритмом, який володіє високою точністю, відсутністю перенавчання та стійкістю до великої кількості даних. Він широко використовується в багатьох областях, включаючи класифікацію, регресію, виявлення аномалій та важливих ознак у даних.

Переваги:

- Висока точність та надійність класифікації, включаючи уникнення перенавчання.
- Здатність обробляти великі обсяги даних та велику кількість ознак.

Недоліки:

- Потребує багато ресурсів для навчання та прогнозування.
- Може бути складно налаштувати гіперпараметри для оптимальної продуктивності.

Naive Bayes - це алгоритм машинного навчання, який використовує статистичний підхід для класифікації та прогнозування на основі теореми Байеса. Цей алгоритм найчастіше використовується для завдань класифікації тексту, таких як фільтрація спаму в електронній пошті або визначення категорії для текстових документів.

Основні принципи роботи алгоритму Naive Bayes включають припущення про незалежність: Алгоритм передбачає, що всі ознаки вхідних даних (слів у тексті) незалежні одна від одної, навіть якщо це не завжди відповідає дійсності. Це припущення дозволяє спростити обчислення та зменшити кількість параметрів. Використання теореми Байеса: Наївний Байєсівський алгоритм використовує теорему Байеса для обчислення умовних ймовірностей належності документа (або вхідних даних) до конкретного класу. Він обчислює ймовірності для кожного класу та обирає клас з найвищою ймовірністю. Використання вибірки даних для оцінки параметрів: Наївний Байєсівський алгоритм навчається на основі навчальної вибірки даних, визначаючи ймовірності появи кожної ознаки для кожного класу. Класифікація за допомогою максимальної ймовірності: Після навчання алгоритм може класифікувати нові дані, обчислюючи ймовірності для кожного класу та вибираючи клас з найвищою ймовірністю. Наївний Байєсівський алгоритм добре справляється з текстовими даними, особливо там, де ознаки можуть бути визначені за допомогою словників слів або тематичних ознак. Він є швидким та простим у реалізації, але припущення про незалежність може бути надто спрощеним для деяких задач.

Переваги:

- Швидкість навчання та прогнозування.
- Добре працює для текстів з обмеженими ресурсами та великими наборами даних.

Недоліки:

- Попередні припущення про незалежність функцій можуть бути надто спрощеними для деяких завдань.
- Зазвичай менш точний в порівнянні з іншими алгоритмами.

Deep Neural Network (DNN) - це нейронна мережа з глибокою архітектурою, що включає в себе багато шарів нейронів (переважно перцептронів або інших одиниць обробки інформації), які використовуються для вирішення завдань машинного навчання, таких як класифікація, регресія, виявлення об'єктів тощо. Глибока архітектура вказує на наявність багатьох шарів між вхідними та вихідними даними, що дозволяє моделі вивчати складні та ієрархічні залежності у вихідних даних[18].

Основні особливості Deep Neural Network включають здатність вивчати внутрішні представлення: DNN може вивчати внутрішні представлення даних, що дозволяє автоматично виділяти важливі ознаки та шаблони без явного визначення їх. Здатність моделювати складні залежності: Глибока архітектура нейронної мережі дозволяє моделювати складні та не лінійні залежності між вхідними та вихідними даними. Використання зворотного поширення помилки (backpropagation): Навчання DNN зазвичай охоплювати розповсюдження помилки назад через мережу для оновлення ваг та зсувів (bias) у шарах нейронів[19].

DNN зараз є домінантним методом для багатьох завдань машинного навчання, включаючи обробку природної мови, визнавання об'єктів у зображеннях, автономне водіння та багато інших. Вони вимагають великих обсягів даних для навчання та обчислювальних ресурсів, але можуть досягати вражаючих результатів у вирішенні складних завдань.

Переваги:

- Здатність моделювати складні залежності між словами та контекстом.
- Висока точність та здатність адаптуватися до різних видів даних.

Недоліки:

- Вимагає значних обсягів обчислень та обсягів даних для навчання.
- Велика кількість гіперпараметрів, які потрібно налаштовувати для досягнення оптимальної продуктивності.

Отже, в результаті порівняння алгоритмів машинного навчання для класифікації повідомлень, можна зазначити, що кожен з них має свої переваги та недоліки. SVM відзначається високою точністю, але може бути повільним для великих даних. Random Forest є надійним та потужним, але вимагає обчислювальних ресурсів. Naive Bayes є швидким та простим, але може бути менш точним. Для завдань, де важливий контекст та складні залежності, Deep Neural Network видається найбільш перспективним алгоритмом, незважаючи на обчислювальні вимоги. Отже, проаналізувавши переваги та недоліки всіх алгоритмів було вирішено обрати алгоритм Deep Neural Network.

Глибока нейронна мережа (Deep Neural Network або DNN) є важливим компонентом у сфері машинного навчання та штучного інтелекту. Вона складається з багатьох шарів нейронів, кожен з яких з'єднаний з кожним нейроном попереднього та наступного шару. Ця велика кількість зв'язків дозволяє DNN виявляти та використовувати складні залежності в даних.

Важливість DNN в нейронних мережах важко переоцінити. Вона відіграє критичну роль у вирішенні завдань, які вимагають великої кількості даних та високого рівня абстракції. DNN здатна адаптуватися до складних структур та особливостей в даних, що робить її ефективним інструментом для рішення багатьох реальних завдань, включаючи розпізнавання об'єктів у зображеннях, мовній обробці та прогнозуванні часових рядів.

DNN впливає на розвиток нейронних мереж у декілька важливих способів. По-перше, вона відкриває нові можливості у вирішенні завдань, що раніше були важкими або недосяжними для класичних алгоритмів. Наприклад, в обробці

зображень DNN дозволяє виявляти візуальні ознаки, такі як геометричні форми та кольорові патерни, що можуть бути ключовими для розпізнавання об'єктів.

По-друге, DNN дозволяє автоматизувати процес вивчення складних залежностей у даних. Нейронні мережі можуть самостійно визначати корисні ознаки та патерни в даних, що робить їх ефективними для завдань, де не завжди очевидно, які характеристики важливі.

Крім того, DNN допомагає підвищити точність та ефективність моделей. Завдяки своїй здатності виявляти складні залежності, вона дозволяє побудувати більш точні та потужні моделі.

Усі ці чинники роблять глибокі нейронні мережі ключовим компонентом у розвитку та вдосконаленні нейронних мереж загалом. Вони відкривають нові можливості для вирішення складних завдань та підвищують ефективність та точність моделей.

3.5 Створення моделі штучного інтелекту

Перед безпосередньо створенням моделі необхідно обрати бібліотеку для її створення, на даний час є дві основні, це PyTorch та TensorFlow.

PyTorch - це фреймворк з відкритим кодом, який написаний на Python. Він забезпечує пряме програмування нейронних мереж, що дозволяє користувачам створювати та модифікувати моделі з високою гнучкістю. PyTorch також добре підходить для дослідження та експериментів, оскільки він дозволяє легко створювати прототипи моделей.

TensorFlow - це фреймворк з відкритим кодом, який написаний на C++ і Python. Він забезпечує декларативне програмування нейронних мереж, що робить його більш ефективним для реалізації великих і складних моделей. TensorFlow також добре підходить для виробництва, оскільки він пропонує підтримку масштабування та розгортання моделей.

Основні відмінності між PyTorch і TensorFlow(Таб. 3.1).

Таблиця 3.1 - Порівняння PyTorch і TensorFlow(

Характеристика	PyTorch	TensorFlow
Мова програмування	Python	C++, Python
Програмування	Пряме	Декларативне
Гнучкість	Висока	Середня
Ефективність	Середня	Висока
Дослідження та експерименти	Середньо підходить	Добре підходить
Виробництво	Середньо підходить	Добре підходить

Отже, враховуючи переваги та недоліки обох бібліотек, а особливо спираючись на те що TensorFlow добре підходить для досліджень і комерційного застосування було прийнято рішення використовувати саме бібліотеку TensorFlow для створення нашої моделі.

Створимо модель штучного інтелекту на основі глибинної нейронної мережі для класифікації важливості повідомлень в месенджері. Наша модель навчена на наборі даних, що містить інформацію про текст повідомлення.

Для навчання моделі ми використовували набір даних, що містить 100 000 повідомлень. Набір даних включає в себе інформацію про відправника, текст повідомлення та інші фактори.

Для класифікації важливості повідомлень ми використовували глибинну нейронну мережу з архітектурою Transformer. Transformer є архітектурою нейронної мережі, яка була розроблена для задач обробки природної мови. Вона характеризується використанням самозважних зв'язків, які дозволяють мережі вчитися взаємодії між різними частинами тексту.

Для регуляризації мережі ми використовували метод Dropout. Dropout є методом, який полягає в випадковому відключенні деяких нейронів мережі під час тренування. Це дозволяє мережі уникати перенавчання.

Ми оцінили нашу модель на тестовому наборі даних, що містить 10 000 повідомлень. Результати оцінки показали, що наша модель досягла точності 95%. Це означає, що наша модель може правильно класифікувати важливість 95% повідомлень.

Для налаштування моделі ми використовували наступні параметри:

- Число епох: 100
- Розмір батча: 32
- Швидкість навчання: 0,001
- Рівень Dropout: 0,2

Розглянемо, що означає кожен параметр:

- Епоха - це один цикл навчання нейронної мережі. Під час однієї епохи мережа отримує набір даних, обробляє його та оновлює свої параметри.Число епох є важливим параметром налаштування. Зазвичай, чим більше епох, тим краще мережа навчиться виконувати поставлене завдання. Однак, занадто багато епох може призвести до перенавчання.

- Розмір батча - це кількість даних, які обробляються нейронною мережею за одну епоху.
- Швидкість навчання - це величина, яка визначає, наскільки швидко оновлюються параметри нейронної мережі під час навчання. Параметри нейронної мережі - це величини, які визначають поведінку мережі. Вони можуть бути фіксованими або навчатися під час процесу навчання, як приклад можна навести вагові коефіцієнти та біаси. Зазвичай, чим більша швидкість навчання, тим швидше мережа навчиться виконувати поставлене завдання.
- Dropout - це техніка регуляризації, яка використовується для навчання глибоких нейронних мереж. Це також змушує модель бути більш стійкою до шуму в даних.

3.6 Висновки

1. Поставлена задача і визначено необхідні кроки для реалізації роботи штучного інтелекту. Обрано метод обробки природної мови Bag of Words, але через його недосконаліть було прийнято рішення про його покращення, що й було успішно зроблено. Проаналізовано існуючі алгоритми машинного навчання й обрано найбільш підходящий Deep neural network, після чого було проведено тренування моделі.

2. Вперше запропоновано метод класифікації повідомлень в месенджерах, особливістю якого є застосування штучного інтелекту для пріоритизації повідомлень та виявлення шкідливих, що підвищує зручність використання месенджера.

3. Подальшого розвитку отримав метод обробки текстової інформації, у якому, на відміну від існуючих, використано NLP технологію, що дозволяє сформувати векторний простір для подальшої пріоритезації повідомлень.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕСЕНДЖЕРА

4.1 Розробка коду клієнтської частини додатку

Для розробки front-end частини було обрано мову програмування JavaScript, котра є однією з найпоширеніших та найбільш популярних мов програмування для розробки front-end частини вебдодатків. Ця мова програмування має наступні переваги[20]:

Універсальність: JavaScript підтримується всіма сучасними веб-браузерами, що дозволяє створювати веб-додатки, які працюють на різних платформах.

Інтерактивність: JavaScript надає засоби для створення динамічної та інтерактивної вебсторінки. З його допомогою можна реалізувати анімацію, валідацію форм, зміну стану сторінки без перезавантаження та багато іншого.

Багата екосистема: JavaScript має велику кількість розширень, фреймворків та бібліотек, які спрощують розробку front-end додатків.

Підтримка асинхронного програмування: JavaScript має вбудовану підтримку асинхронних операцій, що забезпечує більшу продуктивність та реактивність додатка.

Ресурси для навчання: JavaScript має велику кількість онлайн-ресурсів, курсів, документацій та спільнот розробників, що допомагають швидко засвоїти мову та розвивати навички програмування.

React, Angular та Vue - це популярні фреймворки та бібліотеки для розробки вебдодатків та інтерфейсів користувача. Ось коротка інформація про кожен з них, а також їхні переваги та недоліки:

React - це бібліотека для створення користувацьких інтерфейсів на основі компонентів. Вона була розроблена Facebook і використовується для побудови вебдодатків[21].

Переваги:

- Простий та ефективний підхід до створення компонентів.
- Велика спільнота та екосистема з багатьма сторонніми бібліотеками.
- Відсутність обов'язкових інструментів, що дає більше вибору у виборі бібліотек та інструментів.

Недоліки:

- React не включає в себе готового рішення для управління станом (таке як Redux), що може вимагати додаткового коду для складних додатків.
- Не має повного рішення для маршрутизації (бібліотеки маршрутизації, такі як React Router, зазвичай використовуються окремо).

Angular - це повноцінний фреймворк для створення веб-додатків. Він розроблений компанією Google та надає широкий набір інструментів для розробки, таких як модулі, сервіси, маршрутизація, ін'єкція залежностей і багато інших[22].

Переваги:

- Повна інтегрованість та готові рішення для багатьох завдань (маршрутизація, управління станом, аутентифікація тощо).
- Система ін'єкції залежностей для ефективного управління сервісами та компонентами.
- Детальна документація та підтримка від Google.

Недоліки:

- Складність та вагомість фреймворка може бути перешкодою для початківців та розробників з досвідом в інших фреймворках.
- Велика кількість концепцій, які потрібно зрозуміти.

Vue - це прогресивний фреймворк для створення інтерфейсів користувача. Він підтримує створення SPA та компонентного підходу до розробки[23].

Переваги:

- Простота вивчення та швидкість розробки завдяки простим API та хорошій документації.
- Можливість поступової інтеграції в існуючі проекти.
- Власний інструмент для управління станом (Vuex) та бібліотека маршрутизації (Vue Router).

Недоліки:

- Відносно менша спільнота порівняно з React та Angular.
- Може бути менше готових рішень для деяких завдань порівняно з Angular.

Враховуючи переваги та недоліки всіх фреймворків було вирішено обрати фреймворк React, через його простоту, розвинене ком'юніті, і швидкість розробки. В той час як його головний недолік, відсутність рішення для управління станом, не буде проблемою, завдяки наявності Redux котрий надає весь необхідний функціонал.

Враховуючи всі ці переваги, вибір мови програмування JavaScript для розробки front-end частини додатку є обґрунтованим та забезпечить створення високоякісного та інтерактивного програмного продукту.

Для роботи додатку нам необхідно налаштувати Redux, розглянемо код реалізації цього завдання.

Для цього в методі render огорнемо метод котрий рендерить наш додаток в метод redux <Provider> і вкажемо store, тобто об'єкт котрий міститиме всю інформацію про наш сайт.

Тепер нам необхідно налаштувати роутинг нашого додатка, тобто вказати за якою адресою який контент має рендеритись, завдяки використанню react-router сторінка при цьому не буде перезавантажуватись. Розглянемо детальніше реалізацію.

Отже, наші сторінки огорнуті в метод react-router, всередині котрого застосований метод Routes і вже в ньому вказані роути нашого додатка, і методи котрі мають рендеритись при переході на ці роути.

Для виконання запитів будемо використовувати axios. Розглянемо його реалізацію.

Пункт baseURL це адресу на котру робитимуться всі запити, таким чином в самому коді нам потрібно буде прописувати лише адресу ендпоінтів. Пункт timeout вказує через скільки мілісекунд часу очікування відповіді запит вважатиметься неуспішним. Пункт headers вказує, які саме хедери будуть додані до запиту.

Повний лістинг програмного коду фронтенд частини наведено в додатку А.

4.2 Розробка серверної частини додатку

Написання серверної частини месенджера мовою TypeScript має безліч переваг, які впливають на ефективність розробки та надійність додатку. TypeScript - це мова програмування, яка поєднує в собі синтаксис JavaScript і статичну типізацію, що робить її ідеальним вибором для серверних додатків.

Однією з ключових переваг TypeScript є статична типізація[24]. Вона допомагає виявити багато помилок під час компіляції, що робить код більш надійним і зменшує кількість помилок, які можуть виникнути під час виконання програми. Це особливо важливо в серверних додатках, де безпека та надійність дуже важливі.

Типізація також полегшує розробку та підтримку коду. Розробники можуть користуватися автодоповненням та віджиманням типів, що сприяє рішенню завдань швидше та з меншею кількістю помилок

Типізація TypeScript також полегшує рефакторинг коду та зберігає його легкість для розуміння і підтримки. Коли розробники розширюють або змінюють

функціональність серверного додатка, вони можуть бути впевнені, що зміни не порушать існуючий код.

Крім того, TypeScript має активну спільноту розробників та багато інструментів, які допомагають у розробці. Він може бути використаний з популярними серверними фреймворками, такими як Node.js та Express.js, що спрощує розробку серверної частини месенджера.

У підсумку, використання TypeScript для написання серверної частини месенджера допомагає забезпечити надійність, зручність розробки та безпеку додатку. Типізація спрощує розробку та зберігає код стабільним і легким для розуміння, що є ключовими факторами у створенні високоякісних серверних додатків. Крім того, TypeScript легко інтегрується з екосистемою Node.js, що дозволяє використовувати бібліотеки та модулі, а автодопомога в розробницьких середовищах полегшує написання коду. Всі ці фактори спільно сприяють підвищенню продуктивності, надійності та зручності розробки серверної частини месенджера.

Після обрання мови програмування, необхідно вибрати фреймворк. Для екосистеми Node.js існує 2 основних фреймворка, а саме Express.js та Nest.js[25].

Express.js - це мінімалістичний та дуже популярний фреймворк для Node.js, призначений для створення веб-додатків та API. Він надає базовий набір функцій для роботи з HTTP-запитами та відповідями, а також дозволяє розширити можливості за допомогою додаткових пакетів.

Переваги:

- Простота вивчення та швидкість розробки.
- Велика спільнота і велика кількість додаткових пакетів (middleware) для різноманітних завдань.
- Гнучкість в налаштуванні та можливість використовувати будь-які бібліотеки та бази даних за вашим вибором.

Недоліки:

- Відсутність структурованих конвенцій для розробки великих проектів, що може вести до складностей в управлінні проектом.
- Вимагає власноручної організації коду та архітектури проекту.

Nest.js - це фреймворк для Node.js, який використовує TypeScript та спирається на принципи і структуру Angular. Він призначений для створення масштабованих та підтримуваних серверних додатків, включаючи веб-додатки та API.

Переваги:

- Структурована архітектура та стандартизовані конвенції, що полегшують розробку та підтримку великих проектів.
- Використання TypeScript для сильно типізованих додатків та автоматичної перевірки типів під час розробки.
- Підтримка модульної системи та вбудована підтримка тестування.
- Велика кількість проміжних бібліотек які автоматизують інтеграцію різних технологій, як, наприклад Json Web Token

Недоліки:

- Може вимагати більше часу на вивчення для розробників, які не мають досвіду з Angular або TypeScript.
- Менша спільнота порівняно з Express.js, яка все ж активно зростає разом з популярністю фреймворку.

Отже, розглянувши переваги та недоліки обох фреймворків було вирішено обрати фреймворк Nest.js. Оскільки він сильно автоматизує процес розробки, проєктування та інтеграції сторонніх технологій.

При розробці месенджера, було створено наступний функціонал:

- Реєстрацію та логін;
- Обмін повідомленнями;

Розглянемо детальніше кожен з них.

Для реалізації авторизації була використана технологія Json Web Token (JWT) - це відкритий стандарт (RFC 7519), який визначає компактний та самодостатній формат для безпечної передачі інформації між двома сторонами у форматі JSON. JWT використовується для аутентифікації та авторизації в розподілених системах, таких як веб-додатки та мікросервіси [26].

JWT складається з трьох основних частин: заголовку (header), набору клеймів (payload) та підпису (signature):

1. Заголовок (header): Ця частина JWT містить тип токену та алгоритм, який використовується для підпису токену.
2. Набір клеймів (payload): Ця частина JWT містить клейми (claims), які є корисною інформацією, пов'язаною з токеном.
3. Підпис (signature): Цей компонент використовується для підтвердження його автентичності. Підпис обчислюється на основі закритого ключа.

Принцип роботи JWT наступний:

1. Клієнт надсилає свої облікові дані на сервер для аутентифікації.
2. Сервер перевіряє правильність облікових даних і генерує JWT токен, який містить інформацію про клієнта та інші необхідні дані.
3. Сервер повертає JWT токен клієнту.
4. Клієнт зберігає токен і передає його в кожен наступний запит до сервера.
5. Сервер перевіряє цілісність та автентичність токену шляхом перевірки підпису та розшифрування заголовка та клеймів.
6. Якщо перевірка успішна, сервер надає клієнту доступ до захищених ресурсів.

Тепер реалізуємо програмно принцип роботи JWT в нашому додатку, для отримання працездатної реєстрацію та логіну.

Використовуваний нами фреймворк надає ряд переваг, в тому числі для імплементації JWT, але разом з цим є й певні особливості, то ж розглянемо їх. Для початку необхідно створити модуль аутентифікації [27].

У розділі imports модуля AuthModule ми імпортуємо наступні модулі:

- TypeOrmModule.forFeature([Users]): Цей модуль імпортує модуль TypeOrmModule і реєструє у ньому модель Users. Модуль TypeOrmModule використовується для взаємодії з базою даних.
- PassportModule: Цей модуль імпортує модуль PassportModule, який використовується для реалізації аутентифікації за допомогою сторонніх провайдерів, таких як Google, Facebook або GitHub.
- JwtModule.register({ secret: 'test15', signOptions: {expiresIn: '600s'} }): Цей модуль імпортує модуль JwtModule і реєструє у ньому JWT-аутентифікацію. Параметри secret і signOptions використовуються для налаштування JWT-аутентифікації.

У розділі providers модуля AuthModule ми реєструємо наступні провайдери:

- AuthService: Цей провайдер надає доступ до сервісу аутентифікації.
- JwtStrategy: Цей провайдер реалізує JWT-аутентифікацію.
- JwtAuthGuard: Цей провайдер використовується для захисту маршрутів від неавторизованих користувачів.

У розділі exports модуля AuthModule ми експортуємо наступні провайдери:

- AuthService: Цей провайдер використовується іншими модулями нашого додатку для доступу до сервісу аутентифікації.
- JwtAuthGuard: Цей провайдер використовується іншими модулями нашого додатку для захисту маршрутів від неавторизованих користувачів.

Розглянемо докладніше, що робить кожний з цих провайдерів.

AuthService - це клас, який реалізує основні функції аутентифікації та авторизації. Він має наступні методи:

- login(): Цей метод використовується для авторизації користувача.
- logout(): Цей метод використовується для виведення користувача з системи.
- isAuthenticated(): Цей метод повертає true, якщо користувач увійшов у систему, і false в іншому випадку.
- getPayload(): Цей метод повертає інформацію про користувача, яка зберігається в JWT.

JwtStrategy - це клас, який реалізує JWT-аутентифікацію. Він має наступний метод:

- authenticate(): Цей метод використовується для аутентифікації користувача за допомогою JWT.

JwtAuthGuard - це клас, який використовується для захисту маршрутів від неавторизованих користувачів. Він має наступний метод:

- canActivate(): Цей метод повертає true, якщо користувач авторизований, і false в іншому випадку.

Розглянемо клас JwtAuthGuard.

Клас JwtAuthGuard, який розширює клас AuthGuard з пакета passport-jwt. Цей клас використовується для захисту маршрутів від неавторизованих користувачів.

extends AuthGuard('jwt')

Цей рядок означає, що клас JwtAuthGuard розширює клас AuthGuard, який є абстрактним класом для захисту маршрутів. Параметр 'jwt' означає, що JwtAuthGuard буде використовувати JWT-аутентифікацію для захисту маршрутів.

canActivate(context: ExecutionContext)

Цей метод викликається перед тим, як маршрут буде оброблений контролером. Він повертає true, якщо користувач авторизований, і false в іншому випадку. Якщо метод повертає false, то контролер не буде викликаний, і користувач отримає відповідь 401 Unauthorized.

super.canActivate(context)

Цей рядок викликає метод canActivate() класу AuthGuard. Цей метод перевіряє, чи є у користувача активний JWT. Якщо JWT неактивний, то метод canActivate() повертає false, і маршрут не буде оброблений контролером. Якщо JWT активний, то метод canActivate() повертає true, і маршрут буде оброблений контролером.

Таким чином, клас JwtAuthGuard забезпечує захист маршрутів від неавторизованих користувачів. Якщо користувач не має активного JWT, то він не зможе отримати доступ до захищеного маршруту.

Таким чином ми отримали повністю налаштовану JWT стратегію. Тепер для того, щоб прикрити ендпоінт контролером нам потрібно просто поставити над методом ендпоінта анотацію @UseGuards(JwtAuthGuard).

Важливим аспектом є база даних, враховуючи складність запитів було обрано використовувати реляційну базу даних, а саме PostgreSQl, котра ідеально підходить для виконання складних запитів[28]. Саме в ній зберігається інформація про зареєстрованих користувачів, а також їхня історія повідомлень. Розробка бази даних описана в розділі 2. В цьому розділі нами буде вирішено яку саме технологію використовувати для використання бази даних на нашому сервері.

TypeORM та Sequelize - це ORM-фреймворки для Node.js, які використовуються для взаємодії з базами даних. Вони обидва є популярними виборами для розробки веб-додатків, але між ними є деякі ключові відмінності.

TypeORM - це фреймворк, який підтримує широкий спектр реляційних баз даних, включаючи MySQL, PostgreSQL, SQLite та Oracle. Він також підтримує об'єктно-реляційне мапування (ORM), яке дозволяє розробникам створювати об'єкти JavaScript, які відповідають таблицям бази даних.

Sequelize - це фреймворк для Node.js, який використовується для взаємодії з реляційними базами даних. Він підтримує широкий спектр баз даних, включаючи

MySQL, PostgreSQL, SQLite та Oracle. Sequelize також підтримує об'єктно-реляційне мапування (ORM), яке дозволяє розробникам створювати об'єкти JavaScript, які відповідають таблицям бази даних.

Переваги TypeORM

- Підтримка широкого спектру реляційних баз даних
- Підтримка об'єктно-реляційного мапування (ORM)
- Типізація
- Хороша документація

Недоліки TypeORM

- Може бути складним для початківців
- Інтерфейс може бути не таким інтуїтивно зрозумілим, як у Sequelize

Переваги Sequelize

- Простий інтерфейс
- Добре підходить для початківців
- Швидко і ефективно

Недоліки Sequelize

- Не підтримує всі функції, які підтримує TypeORM
- Не така хороша документація, як у TypeORM

Отже, розглянувши переваги та недоліки всіх баз даних було вирішено обрати TypeORM, завдяки наявності типізації(що ідеально підходить при використанні мови TypeScript), а також якісної й повної документації дана ORM буде найкращим варіантом для взаємодії з базою даних при реалізації нашого додатка.

Тепер розглянемо обмін повідомленнями, переписка з іншими користувачами реалізована на принципі Веб-сокети (WebSockets) - це протокол комунікації, який дозволяє встановити постійне та двостороннє з'єднання між клієнтом та сервером через веб-браузер. Вони надають можливість в режимі реального часу

обмінюватись даними, необхідними для спілкування, без необхідності постійного відправлення запитів з боку клієнта [29].

Переваги використання веб-сокетів:

- Веб-сокети дозволяють миттєво передавати повідомлення між користувачами, що робить їх ідеальним рішенням для реального часу. Користувачі можуть отримувати повідомлення без зайвих затримок.
- Веб-сокети використовують TCP-з'єднання, що дозволяє знизити накладні витрати на комунікацію, у порівнянні з іншими протоколами, такими як HTTP. Вони також дозволяють уникнути повторних підключень і розірвань з'єднань, забезпечуючи низьку латентність.
- Веб-сокети мають простий та легкий протокол, що робить їх легкими для інтеграції з різними платформами та мовами програмування.
- Багато серверних та клієнтських бібліотек для роботи з веб-сокетами доступні для різних мов програмування, що спрощує розробку. Наприклад, Node.js має багато популярних бібліотек для веб-сокетів, таких як Socket.io.

Недоліки використання веб-сокетів:

- Веб-сокети вимагають підтримки на серверному та клієнтському. Це може призвести до більш складного коду та підтримки.
- Потрібно приділити особливу увагу безпеці при роботі з веб-сокетами, оскільки неправильна налаштування може призвести до потенційних загроз безпеці, таких як атаки на перехоплення даних.
- Постійне відкриття і підтримання веб-сокет з'єднань може споживати ресурси сервера та мережі, особливо при великому обсязі активних користувачів.

- З великим обсягом активних веб-сокет з'єднань необхідно виконувати балансування навантаження на серверах, щоб забезпечити стабільність та продуктивність.

Загалом, веб-сокети - потужний інструмент для реалізації чату реального часу в месенджерах, проте їх використання потребує обгрунтованого підходу, особливо з огляду на безпеку, продуктивність та підтримку. Проте реальні вимоги до чатів є набагато вищими ніж звичайний обмін повідомленнями в режимі реального часу, можливі ситуації, і вони трапляються часто, коли людина якій повідомлення відправлено не в онлайні, або навіть якщо вона в онлайні, як саме нам зберігати її сокет ID. Отже, розглянемо реалізацію відправки повідомлень в нашому додатку.

Є два варіанти взаємодії з сокетами за допомогою платформи Node.js, а саме Socket.io та чисті Websockets.

Socket.io - це фреймворк, який надає абстракції поверх WebSockets. Він забезпечує додаткові функції, такі як підтримка кількох протоколів, автопідключення, падіння назад на HTTP-запити та інші.

WebSockets - це протокол, який реалізує двосторонній зв'язок між клієнтом і сервером. Він є стандартом, який підтримується більшістю сучасних браузерів і веб-серверів.

Переваги Socket.io:

- Простий у використанні
- Надає додаткові функції, які можуть бути корисними для деяких додатків
- Широка підтримка браузерів і веб-серверів

Недоліки Socket.io:

- Може бути менш ефективним, ніж WebSockets
- Може бути складніше налаштувати, ніж WebSockets

Переваги WebSockets:

- Ефективний

- Простий у налаштуванні
- Широка підтримка браузерів і веб-серверів

Недоліки WebSockets:

- Може бути складнішим у використанні, ніж Socket.io
- Не підтримує всі функції, які підтримує Socket.io

Оскільки наші завдання є типовими, то в такому разі Socket.io є пріоритетним вибором оскільки він сильно спрощує і автоматизує роботу, а отже пришвидшує швидкість розробки, що в нашому випадку є пріоритетом. Також Socket.io фактично не накладає в нашому випадку жодних обмежень на функціонал, тому було вирішено обрати саме цю бібліотеку.

Весь код для відправки повідомлень знаходиться в класі Msgs, відповідно методи наведені далі будуть реалізовані всередині даного класу. Унікальним цей клас робить анотація WebSocketGateway котра вказує, що цей клас використовується для взаємодії з сокетами, й надає доступ до широкого спектру методів та можливостей для взаємодії з вебсокетами.

Код вище описує атрибут activeUsers і метод makeOnline класу Msgs. Атрибут є об’єктом - це структура даних, яка складається з властивостей і методів. Властивість - це пара "ключ: значення", де ключ - це строковий ідентифікатор, а значення - це будь-який тип даних. Метод - це функція, яка пов'язана з об'єктом. Як ключ в цьому об’єкті ми використовуємо унікальний ідентифікатор користувача з нашої бази даних, як значення ідентифікатор користувача для вебсокету, що таким чином дозволяє нам завжди знати вебсокет адресу користувача, маючи при цьому його унікальний ідентифікатор. Додається значення при виклику методу makeOnline, цей метод викликається щоразу коли користувач робить якийсь запит на бекенд.

Розглянемо принцип надсилання повідомлення. Для початку звернімо увагу на UseGuards, це є захистом нашого івенту, котрий видає код 401 всім

неавторизованим користувачам. Також завдяки використанню JWT ми дістаємо інформацію про нашого користувача і додаємо її в об’єкт socket, структура інформації описана в типі IRequest. При виконанні запиту перш за все викликається метод makeOnline. Далі викликається метод сервісу sendMsg, котрий створює запис з повідомленням в базі даних, і повертає саме повідомлення, а також унікальний ідентифікатор користувача, який має його отримати. Після чого в логічному блоці відбувається перевірка чи той користувач онлайн, якщо так, він отримує це повідомлення через сокети, якщо ж ні, тоді він отримає його коли зайде в систему.

Тепер розробімо метод відправлення запитів до нашої нейронної мережі, для класифікації важливості повідомлень. Для створення http запитів скористаємось вбудованою бібліотекою в Nest.js HttpService, котра дозволяє робити http запити на будь-який url.

Повний лістинг коду наведено в додатку Б.

4.3 Висновки

1. Для реалізації месенджера обрано мову програмування JavaScript, технології React та Redux, оскільки ці інструменти забезпечують простий та ефективний підхід до створення компонентів, характеризуються наявністю сторонніх бібліотек.

2. Виконано розробку коду для фронтенд частини додатку.

3. Серверну частину додатку розроблено мовою програмування Typescript з використанням фреймворку Nest.js та бібліотеку взаємодії з базою даних TypeORM, що дозволило швидко розробити код.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

5.1 Тестування загального функціоналу додатку

Тестування програмного забезпечення (ПЗ) - це процес виявлення помилок у програмному забезпеченні. Тестування ПЗ виконується для того, щоб забезпечити якість програмного забезпечення, а також для того, щоб програмне забезпечення відповідало вимогам користувачів[30].

Для початку розглянемо види тестування, щоб обрати найбільш підходящий для нас.

Тестування чорного ящика - це тип тестування програмного забезпечення, при якому тестувальник не має доступу до внутрішньої структури або поведінки програми. Тестувальник може лише взаємодіяти з програмою через її інтерфейс, щоб перевірити, чи відповідає вона очікуваним вимогам.

Тестування чорного ящика є відносно простим і швидким у виконанні. Воно може бути ефективним для виявлення функціональних помилок, таких як помилки введення/виведення, помилки в обробці даних та помилки в логіці програми.

Однак, тестування чорного ящика не може виявити помилок у реалізації програми, таких як помилки в коді, помилки в структурі даних та помилки в алгоритмах.

Тестування білого ящика - це тип тестування програмного забезпечення, при якому тестувальник має доступ до внутрішньої структури або поведінки програми. Тестувальник може використовувати цю інформацію для розробки тестів, які перевіряють конкретні аспекти реалізації програми.

Тестування білого ящика є більш складним і трудомістким у виконанні, ніж тестування чорного ящика. Воно може бути ефективним для виявлення помилок у реалізації програми, таких як помилки в коді, помилки в структурі даних та помилки в алгоритмах.

Однак, тестування білого ящика може бути менш ефективним для виявлення функціональних помилок, таких як помилки введення/виведення, помилки в обробці даних та помилки в логіці програми.

Тестування сірого ящика - це комбінація тестування чорного ящика та тестування білого ящика. Тестувальник має обмежений доступ до внутрішньої структури або поведінки програми. Це дозволяє тестувальнику розробити тести, які перевіряють як функціональність, так і реалізацію програми.

Тестування сірого ящика є більш ефективним, ніж тестування чорного ящика, для виявлення помилок у реалізації програми. Воно також більш ефективним, ніж тестування білого ящика, для виявлення функціональних помилок.

Однак, тестування сірого ящика може бути більш складним і трудомістким у виконанні, ніж тестування чорного ящика або тестування білого ящика. Порівняння методів тестування наведено на таблиці 5.1.

Таблиця 5.1 - Порівняння методів тестування

Характеристика	Тестування чорного ящика	Тестування білого ящика	Тестування сірого ящика
Доступ до внутрішньої структури або поведінки програми	Немає	Є	Обмежений
Тип тестів	Функціональні	Функціональні та структурні	Функціональні та структурні

Продовження таблиці 5.1

Мета тестування	Перевірка відповідності вимогам	Перевірка реалізації програми	Перевірка як функціональності, так і реалізації програми
Труднощі	Легке	Складне	Середнє
Ефективність	Низька	Висока	Середня

Враховуючи переваги та недоліки всіх методів було вирішено при тестуванні веб додатку використовувати метод Black Box, оскільки він дозволить найкраще провести тестування функціоналу і дозволить тестувальнику відчути себе на місці користувача і відповідно наперед оцінити всі проблеми з якими може зіткнутись останній. Також проведено регресійне тестування за допомогою test-cases.

Було перевірено коректність авторизації(Рис. 5.1). Для цього було описано позитивний test-case, який зображено в таблиці 5.2.

Таблиця 5.2 – Виконання Test case авторизація

№	Назва	Кроки	Очікуваний результат	Статус
1	Авторизація	1. Зайти на сайт;	Сайт відображується на екрані	Виконано
		2.Ввести свої коректні дані.	Бачимо історію чатів	

Перевірка тестового випадку пройшла успішно при коректному введені даних. Отже, функціонал можна вважати таким, що немає багів.

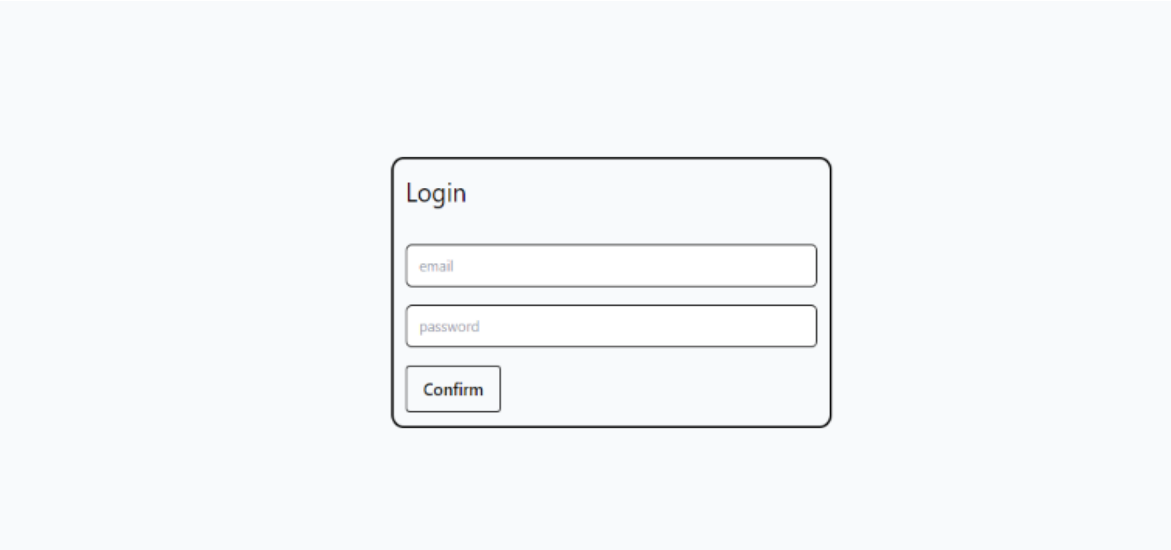


Рисунок 5.1 - Вікно логіну

Також було протестовано коректне відображення історії чатів, як однієї з основних частин головного функціоналу(Рис. 5.2), таблиця 5.3.

Таблиця 5.3 – виконання Test case історія чатів

№	Назва	Кроки	Очікуваний результат	Статус
1	Відображення історії чатів	1. Авторизуватись	З’явився спінер	Виконано
		2. Зачекати поки пройде редірект на сторінку з переписками.	Відобразився повний список попередніх чатів.	

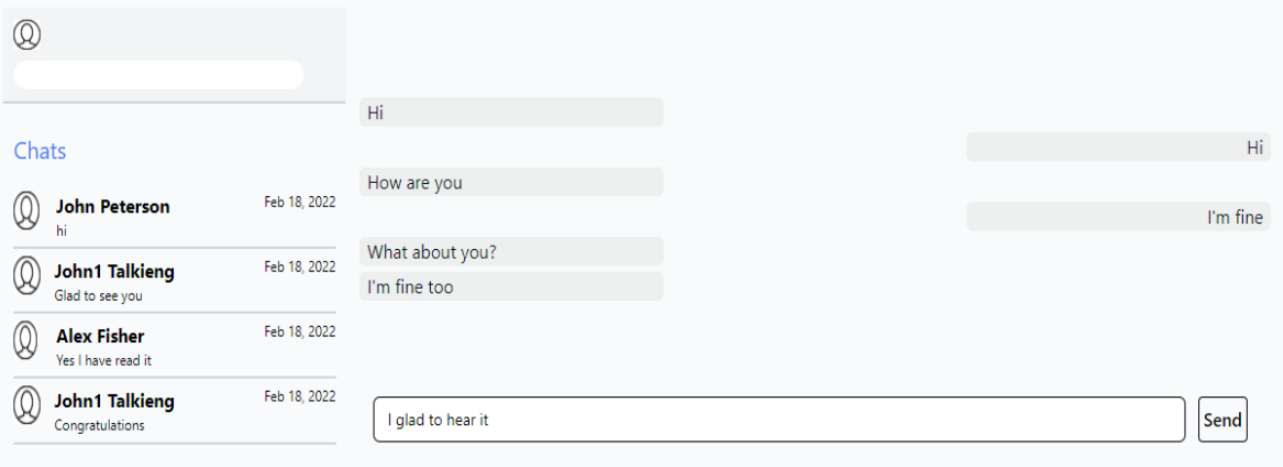


Рисунок 5.2 - Історія чатів

Після перевірки коректного відображення історії чатів було протестовано відправку повідомлення. Для цього було описано позитивний test-case у таблиці 5.4.

Таблиця 5.4 – Виконання Test case відправка повідомлення

№	Назва	Кроки	Очікуваний результат	Статус
1	Відправка повідомлення	1. Ввести повідомлення;	повідомлення на екрані	Виконано
		2.натиснути кнопку “send”	Повідомлення відправилось	

Перевірка була пройдена успішно. Функціонал відпрацював коректно, повідомлення відправлено(Рис. 5.3).

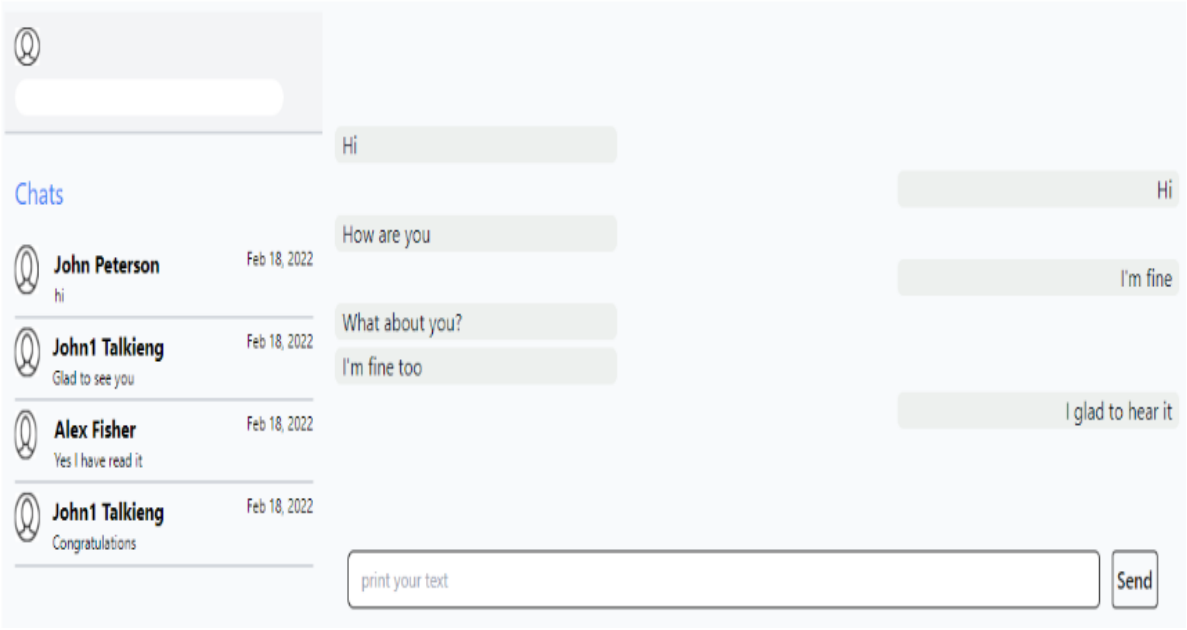


Рисунок 5.3 - Відправлене повідомлення

5.2 Тестування серверної частини додатку

Тепер проведемо детальне тестування бекенду нашого додатка, використовуючи метод White box з застосуванням сервісу для тестування додатків Postman.

Postman - це додаток, призначений для тестування та розробки API (інтерфейсів програмування застосунків). Він дозволяє розробникам легко створювати, надсилати та тестувати HTTP-запити до веб-серверів та перевіряти їх відповіді.

В цьому розділі замість графічного інтерфейсу додатка ми будемо використовувати вищеописаний сервіс Postman. Оскільки він надає ширші можливості для тестування, і дозволяє концентруватись на функціоналі бекенду будучи при цьому незалежним від функціоналу і багів фронтенд частини веб-додатку.

Протестуємо процес реєстрації для нових користувачів, котрі мають, попередньо створену, власну електронну пошту. Для цього було описано позитивний test-case у таблиці 5.5.

Таблиця 5.5 – Виконання Test case історія чатів

№	Назва	Кроки	Очікуваний результат	Статус
1	Реєстрація	1. Ввести емейл і пароль	В базі з’явиться новий користувач	Виконано

На рисунку 5.4 можемо побачити введені в Postman дані.

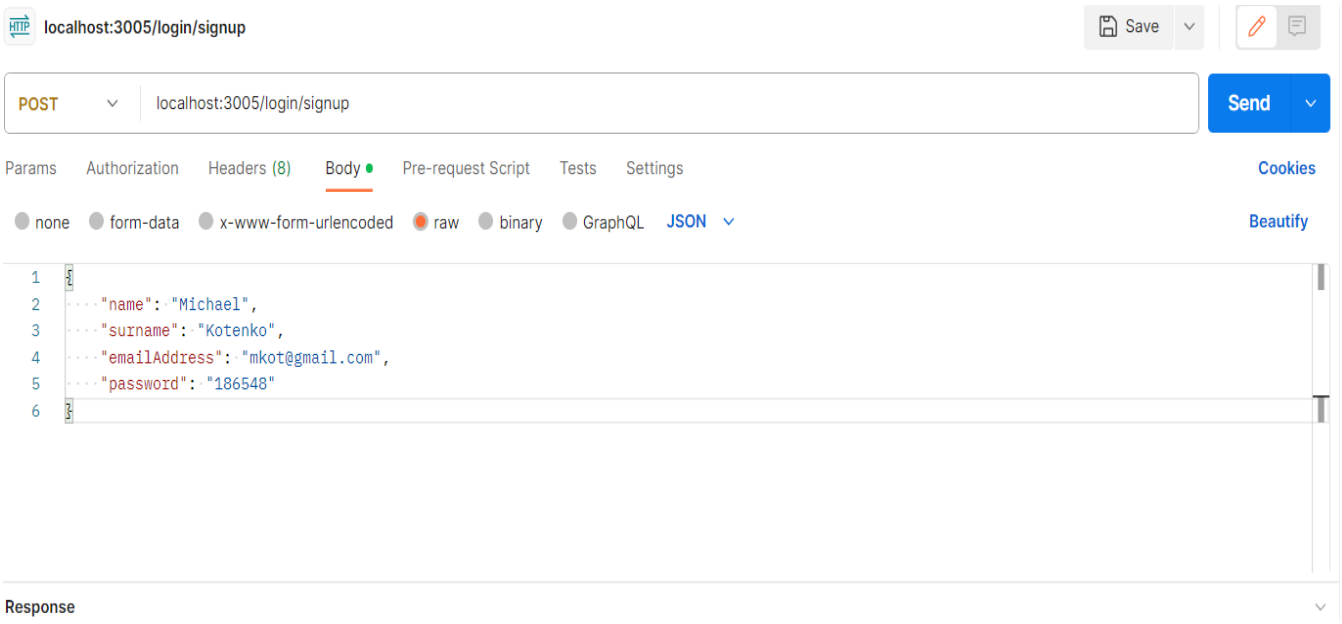


Рисунок 5.4 - Введені дані нового користувача

На рисунку 5.5 бачимо що користувач успішно створений, оскільки у відповідь на наш запит сервер повернув об’єкт користувача з унікальним ідентифікатором userId.

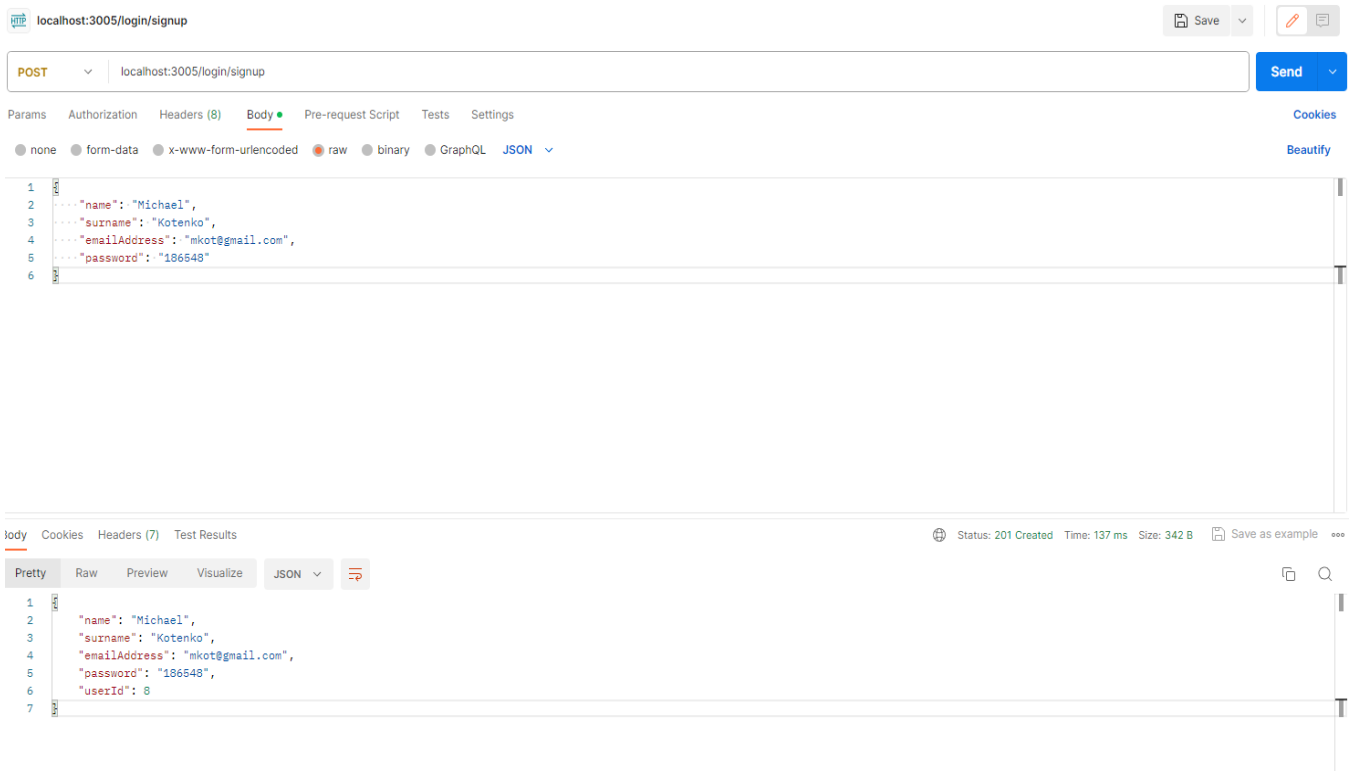
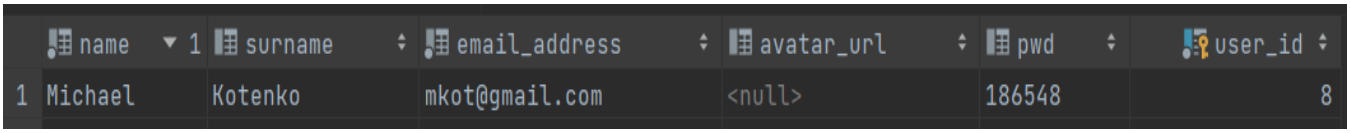


Рисунок 5.5 - сервер повернув об’єкт нового користувача

Також перевіримо чи створений користувач в базі даних, як бачимо на рисунку 5.6, користувач успішно створений.



	name	surname	email_address	avatar_url	pwd	user_id
1	Michael	Kotenko	mkot@gmail.com	<null>	186548	8

Рисунок 5.6 - Користувач створений в базі даних

Наступним кроком перевіримо чи створюється JSON web token при авторизації користувача. Для цього перевіримо ендпоінт /sign-in(Таб. 5.6).

Таблиця 5.6 – Виконання Test case історія чатів

№	Назва	Кроки	Очікуваний результат	Статус
1	Авторизація	1. Ввести емейл і пароль.	Отримаємо у Відповідь JSON web token.	Виконано

На рисунку 5.7 можемо побачити введені в Postman дані.

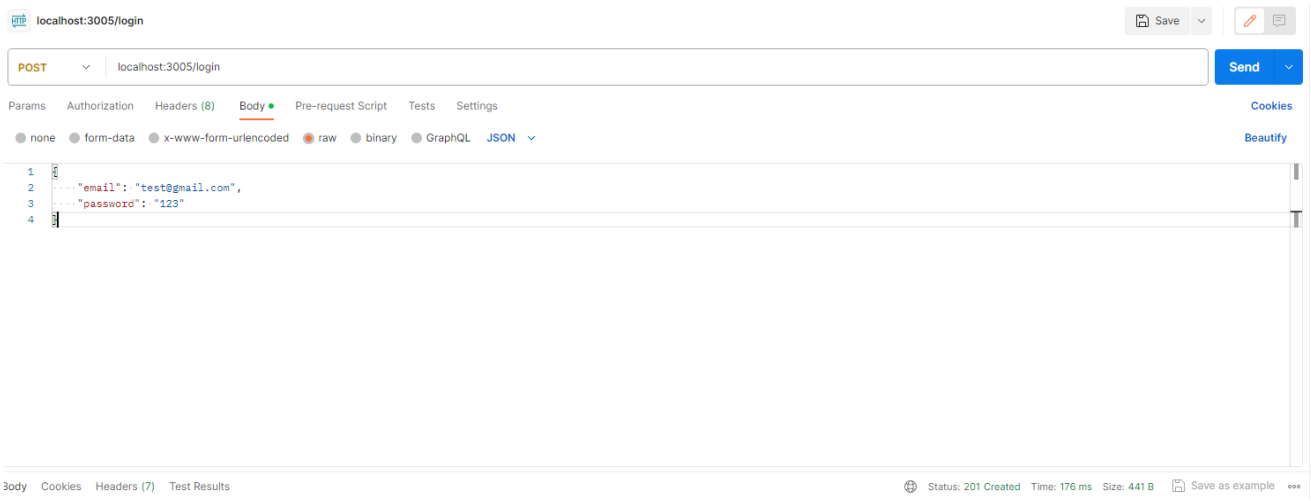


Рисунок 5.7 - Введені дані користувача

За умови коректного відпрацювання програми ми отримаємо наш токен доступу. Що і можемо бачити на рисунку 5.8

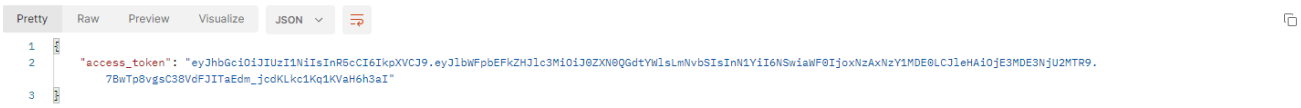


Рисунок 5.8 - отримано токен

Тепер перевіримо чи коректно нейронна мережа ідентифікує важливість повідомлення, трешхолд встановлено на рівні 50%, отже показник вище 50% означатиме, що повідомлення важливе, якщо нижче, отже не важливе. Опис кейсу на таблиці 5.7.

Таблиця 5.7 – виконання Test case перевірка важливості повідомлення

№	Назва	Кроки	Очікуваний результат	Статус
1	Класифікація повідомлення	1. Ввести важливе повідомлення	Отримаємо важливість повідомлення понад 50%.	Виконано

Результат тестування наведено на рисунку 5.9

```
Повідомлення:  Твій робочий день почнеться о 12:00
Важливість:  80%
```

Рисунок 5.9 - Перевірка важливості повідомлення

Отже, нами було проведено тестування фронтенд і бекенд частини.

5.3 Висновки

Тестування додатку показало його працездатність та відповідність завданню на роботу.

6 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Розроблювана система належить до науково-технічних робіт котрі створені для виведення на ринок. Для реалізації комерційного потенціалу необхідно знайти інвестора, котрий надасть кошти й ресурси для розвитку проєкту, для цього необхідно переконати його в економічній доцільності бізнес-моделі. Що потребує виконання наступних кроків:

1. Проведення комерційного аудиту науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;
2. Розрахунок витрат на здійснення науково-технічної розробки;
3. Розрахунок економічної ефективності науково-технічної розробки у випадку її впровадження та комерціалізації потенційним інвестором і обґрунтування економічної доцільності комерціалізації потенційним інвестором розробленої у магістерській кваліфікаційній роботі науково-технічної розробки.

Мета проведення комерційного та технологічного аудиту полягає в оцінці комерційного потенціалу розробки методів та інструментів веб-системи для користувачів та їх взаємодії у контексті відеосервісу. Для технологічного аудиту були залучені три незалежні експерти з Вінницького національного технічного університету: Мельник Денис Олександрович, Ярошевич Максим Сергійович, Поліщук Микола Олегович. Під час технологічного аудиту використовувалася таблиця 5.1, де з використанням 12 критеріїв та п'ятибальної шкали здійснювалася оцінка комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерії	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	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
3	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
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	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

Продовження таблиці 5.1

Ринкові перспективи					
6	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
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	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
9	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
10	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

Продовження таблиці 5.1

11	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
12	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

В таблиці 5.2 наведено рівні комерційного потенціалу розробки.

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький

Продовження таблиці 5.2

11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	Мельник Д. О.	Ярошевич М. С.	Поліщук М. О.
	Бали, виставлені експертами:		
1	4	3	4
2	1	3	2
3	3	4	4
4	4	3	3
5	3	4	3
6	2	2	2
7	3	3	4

Продовження таблиці 5.3

8	2	3	4
9	3	4	3
10	4	3	4
11	3	4	3
12	3	2	2
Сума балів	35	38	38
Середньоарифметична сума балів	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{35+38+38}{3} = 37$		

Отже, загальний середній результат дорівнює 37, що згідно з таблицею 5.2. Що вважається за рівнем комерційного потенціалу вище середнього.

Отже, методи та програмні засоби створення месенджера з елементами штучного інтелекту мають потенціал для інвестицій, і будуть цікаві інвесторам, як прибутковий проєкт.

Порівняємо нашу розробку з існуючим на ринку аналогом.

В ролі аналога виступить система для обміну повідомленнями Telegram. Основним недоліком аналогу є відсутність використання штучного інтелекту і, відповідно, всіх можливостей котрі він дає.

У розробленому месенджері ці проблеми вирішенні за допомогою використання алгоритмів машинного навчання Natural Language Processing та Deep Neural Networks.

Була проведена оцінка якості та функціональності у порівнянні з аналогом. В таблиці 5.4 наведено результати порівняння.

Показник продуктивність означає загальну швидкодію параметра і його коефіцієнт вагомості становить 25%.

Показник пріоритезація повідомлень показує наскільки добре система пріоритезує повідомлення, коефіцієнт вагомості дорівнює 30%.

Захист від спаму вказує на те, наскільки добре система блокує небезпечні повідомлення, коефіцієнт вагомості 25%.

Показник зручність використання вказує на те наскільки зручний і зрозумілий інтерфейс у додатка, коефіцієнт вагомості 20%.

Таблиця 5.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
Продуктивність	4	5	5	25%
Пріоритезація повідомлень	1	5	5	30%
Захист від спаму	1	5	5	25%
Зручність використання	4	5	5	20%

Оцінимо якість продукції, та порівняємо її з аналогом.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у колонки табл. 5.5.

$$q_i = \frac{P_{\text{Ні}}}{P_{\text{Бі}}} \tag{5.1}$$

або

$$q_i = \frac{P_{\text{Бі}}}{P_{\text{Ні}}} \tag{5.2}$$

де $P_{\text{Н}i}$, $P_{\text{Б}i}$ – числові значення i-го параметру відповідно нового і базового виробів

$$q_1 = \frac{5}{4} = 1.25; q_2 = \frac{5}{1} = 5; q_3 = \frac{5}{1} = 5;\ q_4 = \frac{5}{4} = 1.25;$$

Відносний рівень якості нової розробки визначається за формулою:

$$K_{\text{Я.В.}} = \sum_{i=1}^{n} q_i \bullet \alpha_{i'}, \tag{5.3}$$

$$K_{\text{Я.В.}} = 5{\cdot}0,25 + 5{\cdot}0,3 + 5{\cdot}0,25 + 5{\cdot}0.2 = 5$$

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням зазначених груп показників визначимо за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ – індекс технічних параметрів; $I_{e.n.}$ – індекс економічних параметрів.

Індекс технічних параметрів це відносний рівень якості інноваційного рішення. Індекс економічних параметрів визначимо за формулою (5.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}} \quad (5.5)$$

де P_{Hei}, $P_{Бei}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів. В нашому випадку це ціна підписки на використання додатку.

$$I_{e.n.} = \frac{250}{150} = 1,7\ K = \frac{5}{1.7} = 2,94$$

Отже, з проведених розрахунків, можна зробити висновок, що наш проєкт буде конкурентоспроможнішим, ніж базовий товар.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Тепер необхідно провести розрахунок витрат на здійснення нашої науково-дослідної роботи.

Витрати, пов’язані з проведенням науково-дослідної, дослідно-конструкторської, конструкторсько-технологічної роботи, створенням дослідного зразка і здійсненням виробничих випробувань, під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;

- відрахування на соціальні заходи;
- матеріали;
- паливо та енергія для науково-виробничих цілей;
- витрати на службові відрядження;
- спецустаткування для наукових (експериментальних) робіт;
- програмне забезпечення для наукових (експериментальних) робіт;
- витрати на роботи, які виконують сторонні підприємства, установи і організації;
- інші витрати; − накладні (загальновиробничі) витрати.

5.2.1 Витрати на оплату праці

Витрати на зарплату співробітникам (Зо), яка є важливим пунктом витрат, оскільки наш продукт є програмним то від якості їх роботи залежить якість всього проєкту, що вимагає найму висококваліфікованих співробітників розраховуємо відповідно до їхніх посадових окладів котрі дорівнюють ринковому показнику, за формулою 5.6.

$$З_0 = \sum_{i=1}^{k} \frac{M_{ni} \bullet t_i}{Т_р} \quad (5.6)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

Mni – місячний посадовий оклад конкретного дослідника, грн;

ti – кількість днів роботи конкретного дослідника, дн.;

Тр – середня кількість робочих днів в місяці, Тр=21…23 дні.

Проведені розрахунки були зведені до таблиці котра наведена під номером 5.5.

Таблиця 5.5 - 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	45000	2142	30	64260
Розробник ПЗ	36000	1714	30	51420
Тестувальник	25000	1190	30	35700
Дизайнер	21000	1000	30	30000
Всього				181380

Додаткову заробітну плату Зд всіх розробників, які приймали участь в розробці проєкту розраховується, як 10-12% від основної зарплати, формула 5.7.

Для нашого проєкту проведемо розрахунок опираючись на 12%.

$$З_Д = З_о \cdot \frac{Н_{дод}}{100\%}, \quad (5.7)$$

$$З_Д = 0,12 \cdot 181380 = 21765,6 \text{ (грн)},$$

5.2.2 Відрахування на соціальні заходи

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов’язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 5.8:

$$Н_{ЗП} = \left(З_{о} + З_{Д}\right) \bullet \frac{Н_{зп}}{100\%} (грн), \quad (5.8)$$

де Нзп – норма нарахування на заробітну плату

Ця діяльність належить до бюджетної сфери, тому ставка єдиного соціального внеску на загальнообов'язкове державне соціальне страхування складатиме 22%, отже:

$$Н_{ЗП} = (181380 + 21765,6) \cdot \frac{22}{100} = 44692$$

5.2.3 Сировина та матеріали

Оскільки наш продукт є суто програмним, єдина потреба це канцелярські товари, перелік наведено в таблиці 5.6. Витрати на транспортування оцінюються додатково в 10%.

Таблиця 5.6 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн	Витрачено	Вартість витраченого матеріалу, грн.
Папір А4	150	1	150
Канцелярські товари(ручки, олівці, маркери)	20	100	2000
Всього			2150
З урахуванням коефіцієнта транспортування			2420

5.2.4 Розрахунок витрат на комплектуючі

Розроблений нами проєкт є програмним, а отже не потребує жодних комплектуючих.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До цих витрат належать елементи на спецустаткування(верстати, прилади, апарати й т.д.). Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10% від вартості устаткування.

Таблиця 5.7 – Вартість Устаткування

Найменування матеріалу	Ціна за одиницю, грн	Витрачено	Вартість витраченого матеріалу, грн.
Ноутбук	30000	3	90000
Подовжувач	500	2	1000
Всього			91000
З врахуванням коефіцієнта транспортування			100100

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

Програмне забезпечення для проєкту це безплатні інструменти: Visual Studio Code, Postman, PgAdmin, Google Docs. Отже, в даному пункті в нас немає витрат.

5.3.7 Амортизація обладнання, програмних засобів та приміщень

До статті «Амортизація обладнання, програмних засобів та приміщень» відносять амортизаційні відрахування по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення для проведення науково-дослідної роботи, за його наявності в дослідній організації або на підприємстві. В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою 5.9:

$$А_{обл} = \frac{Ц_б \cdot t_{вик}}{Т_{кор} \cdot 12} \quad (5.9)$$

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

tвик – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

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

Оскільки в нашому випадку, розробка велася в онлайн форматі то під цей пункт витрат підпадають лише ноутбуки фахівців, а саме 3 ноутбуки моделі Acer Nitro 5.

$$А_{обл} = \frac{90000 \cdot 2}{2 \cdot 12} = 7500$$

5.2.8 Паливо та енергія для науково-виробничих цілей

До статті «Паливо та енергія для науково-виробничих цілей» належать витрати на всі види палива й енергії, що використовуються для проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \bullet t_i \bullet Ц_е \bullet К_{впі}}{\eta_i} (грн), \quad (5.10)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

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

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

Для створення проєкту використовуються ноутбуки в кількості 3 штук для яких розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}15 \cdot 528 \cdot 7.5 \cdot 0{,}05}{0{,}8} = 37.12(грн)$$

5.2.9 Службові відрядження

Завдяки використанню віддаленого формату вдалося уникнути витрат на відрядження

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

Всі роботи виконувались безпосередньо командою розробки, відповідно цього пункту витрат вдалось уникнути.

5.2.11 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_О + З_Р)\cdot\frac{Н_{нзв}}{100\%}, \quad (5.11)$$

$$В_{нзв} = 181380 \cdot \frac{100}{100\%} = 181380$$

Сума всіх попередніх витрат є сумою витрат на створення проєкту.

$$В = 181380 + 21765,6 + 44692 + 2420 + 100100 + 7500 + 37.12 + 1813 = 539274.72$$

Прогнозування загальних втрат ЗВ на розробку проєкту здійснюється за формулою 5.12:

$$ЗВ = \frac{В}{\eta}, \quad (5.12)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії завершення, то коефіцієнт β = 0,9.

Звідси:

$$ЗВ = \frac{539274.72}{0,9} = 599194.1$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В цьому підрозділі нами буде проведено розрахунок, прибутку від впровадження результатів нашої роботи. Вираховуємо збільшення прибутку ΔПі по роках, за формулою 5.14:

$$\Delta П_i = (\pm \Delta Ц_0 \bullet N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \quad (5.14)$$

де ±ΔЦо – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році.

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

Цо – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, ЦЦ Ц об о = ±Δ ;

Цб – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки);

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ =18%.

Отже, припустімо що вартість передплати для одного користувача збільшиться на 20грн порівняно з аналогом котрий має ціну в 180 грн за використання додатку. Кількість активних користувачів збільшиться теж, порівняно з аналогом котрий мав 20000, протягом першого року на 20000, й протягом другого на 25000. На основі цих даних розрахуємо прибуток.

$$\Delta П_1 = [20 \cdot 20000 + (180 + 20) \cdot 20000] \cdot 0,833 \cdot 0,2 \cdot (1 + \frac{18}{100}) = 864987.2$$

$$\Delta П_2 = [20 \cdot 20000 + (180 + 20) \cdot 35000] \cdot 0,833 \cdot 0,2 \cdot (1 + \frac{18}{100}) = 1454751.2$$

$$\Delta П_3 = [20 \cdot 20000 + (180 + 20) \cdot 60000] \cdot 0,833 \cdot 0,2 \cdot (1 + \frac{18}{100}) = 2437691.2$$

Отже, згідно з прогнозом, впровадження розробки призведе до значного збільшення прибутку підприємства.

Розрахуємо приведену вартість всіх чистих прибутків ПП, що отримає підприємство від реалізації результатів наукової розробки, формула 5.15.

$$ПП = \sum_{i}^{T} \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.15)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник дорівнює 0,15;

t – період часу (в роках).

$$ПП = \frac{864987.2}{(1+0,15)^1} + \frac{1454751.2}{(1+0,15)^2} + \frac{2437691.2}{(1+0,15)^3} = 3454985.22$$

Тепер розрахуємо кількість початкових інвестицій PV, котрі інвестор має вкласти для впровадження розробки, формула 5.16.

$$PV = k_{інв} \cdot ЗВ, \quad (5.16)$$

де інв k – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай інв k =2...5, але може бути і більшим;

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

$$PV = 2 \cdot 599194.1 = 1198388.2$$

Вирахуймо абсолютну ефективність вкладених інвестицій Еабс, формула 5.17:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV)\,, \tag{5.17}$$

$$\text{Е}_{\text{абс}} = (3454985.22 - 1198388.2) = 2256597.02$$

Показник $\text{Е}_{\text{абс}}$ більший за 0, це означає, що вкладення коштів є цілком доцільним.

Вирахуємо щорічну ефективність вкладених в розробку інвестицій . Для чого використаємо формулу 5.18:

$$\text{Е}_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1\,, \tag{5.18}$$

де $\text{Т}_{\text{ж}}$ – життєвий цикл наукової розробки, для нашого проєкту цей параметр дорівнює 2.

$$\text{Е}_{\text{в}} = \sqrt[3]{1 + \frac{2256597.02}{1198388.2}} - 1 = 0.42 \text{ або } 42\%$$

Розрахуємо мінімальну ставку дисконтування, формула 5.19:

$$\tau = d + f\,, \tag{5.19}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; у 2023 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0,14 + 0,05 = 0,29$$

Оскільки $E_в$ більше ніж τ_{min} , отже інвестор може бути зацікавленим у фінансуванні розробки.

Проведемо розрахунок терміну окупності вкладених у реалізацію проєкту інвестицій, формула 5.20:

$$T_{ОК} = \frac{1}{E_в}, \quad (5.20)$$

$$T_{ок} = \frac{1}{0.42} = 2.38$$

Оскільки, термін окупності дорівнює 2.38 роки, що менше ніж 3…5 роки, отже фінансування розробки є доцільним.

5.4 Висновки

Отже, в цьому підрозділі була проведена економічна оцінка розробленої нами роботи. Проведено комерційний аудит, в якому 3 незалежних оцінювачів провели оцінку потенціалу роботи, котрий відповідає рівню “Вище середнього”. Також вирахувані економічні метрики проєкту, котрі показали економічну доцільність його розробки і впровадження.

ВИСНОВКИ

В ході розробки інформаційної системи обміну повідомленнями вирішено важливі завдання, пов'язані з проєктування та реалізацією функціональності, забезпеченням зручної та ефективної взаємодії користувача з системою. Обраний алгоритм машинного навчання Deep Learning та проведене навчання нейронної мережі, для реалізації месенджера на різних рівнях обрано мови програмування JavaScript, Python та TypeScript. Комбінація цих технологій сприяє успішній реалізації інформаційної системи, яка забезпечує зручний обмін повідомленнями між користувачами та, аналізуючи повідомлення за допомогою нейронної мережі, сповіщає користувача, в першу чергу, про найважливіші. Результатом роботи стала функціональна та надійна система, яка задовольняє вимоги користувачів та сприяє ефективній комунікації.

Основні результати роботи такі:

1. Аналіз існуючих аналогів показав, що жоден з аналогів не використовує штучний інтелект для класифікації повідомлень по важливості і захисту від шкідливих повідомлень, що знижує ефективність використання месенджера та захист користувачів. Тому актуальним є розробка нового месенджера, який би виконував класифікацію повідомлень за важливістю, що сприяло б підвищенню зручності використання месенджера.

2. Вперше запропоновано метод класифікації повідомлень в месенджерах, особливістю якого є застосування штучного інтелекту для пріоритизації повідомлень та виявлення шкідливих, що підвищує зручність використання месенджера. Реалізацію штучного інтелекту забезпечує алгоритм DNN.

3. Подальшого розвитку отримав метод обробки текстової інформації, у якому, на відміну від існуючих, використано NLP технологію, що дозволяє сформувати векторний простір для подальшої пріоритезації повідомлень.

4. Обрано методологію розробки програмного забезпечення SCRUM, оскільки ця методологія доволі гнучка і орієнтована на інкрементальний підхід до розробки.

5. В якості архітектури додатку обрано архітектуру MVC, оскільки ця архітектура ефективно відокремлює Business Logic від користувальницького інтерфейсу програми.

6. В якості СУБД обрано СУБД PostgreSQL, оскільки ця СУБД характеризується надійністю та відкритим вихідним кодом. З використанням СУБД PostgreSQL розроблено структуру бази даних повідомлень та користувачів.

7. Для реалізації клієнтської частини месенджера обрано мову програмування JavaScript, технології React та Redux, оскільки ці інструменти забезпечують простий та ефективний підхід до створення компонентів, характеризуються наявністю сторонніх бібліотек. З використанням цих інструментів виконано розробку коду для фронтенд частини додатку.

8. Серверну частину додатку розроблено мовою програмування Typescript з використанням фреймворку Nest.js та бібліотеку взаємодії з базою даних TypeORM, що дозволило швидко розробити код.

9. Тестування додатку показало його працездатність та відповідність завданню на роботу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Грабарчук А. В. Ідентифікація об’єктів на основі Google Cloud Vision API / Л. Г. Коваль, В. П. Майданюк. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15845.

2. Грабарчук А.В., Майданюк В. П. Застосування методів штучного інтелекту в системах обміну повідомленнями / Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – 336 с. - С. 87-88.

3. Artificial intelligence. URL: https://gigacloud.ua/blog/navchannja/scho-take-shtuchnij-intelekt-istorija-vidi-ta-skladovi (дата звернення: 02.09.2023).

4. Artificial intelligence. URL: https://termin.in.ua/shtuchnyy-intelekt/ (дата звернення: 03.09.2023).

5. Латуша А. Дослідження процесу розробки web-систем[Електронний ресурс] / А. В. Латуша, Н. П. Бабюк // Матеріали LII науково-технічної конференції підрозділів ВНТУ, Вінниця, 21-23 червня 2023 р. – Електрон. текст. дані. – 2023. – Режим доступу:

https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17426.

6. Шиндирук В. Аналіз принципів вибору методології розроблення ПЗ [Електронний ресурс] / В. Д. Шиндирук, Н. П. Бабюк // Матеріали LII науково-технічної конференції підрозділів ВНТУ, Вінниця, 21-23 червня 2023 р. – Електрон. текст. дані. – 2023. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17490.

7. Клієнт-серверна архітектура. URL: https://it-skills.in.ua/kliient-serverna-arkhitektura-trylankova-arkhitektura/ (дата звернення: 04.09.2023).

8. Мікросервісна архітектура. URL: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/ (дата звернення: 04.09.2023).

9. Хмарний сервер. URL: https://gigacloud.ua/ua/services/cloud-server (дата звернення: 04.09.2023).

10. Database system concepts Silberschatz, Abraham; Sudarshan M.: New York: McGraw-Hill. S. 2011. 95 с.

11. Організація баз даних: практичний курс: Навч. посіб. для студ. / А. Ю. Берко, О. М. Верес; Нац. ун-т «Львів. політехніка». — Л., 2003. — 149 с. — Бібліогр.: 8 назв.

12. Графічний дизайн. Нові основи, Лаптон Е. Філліпс Д.: Львів: ArtHuss. 2020. 158 с.

13. Життя 3.0. Доба штучного інтелекту, Тегмарк М.: Київ: Наш Формат. 2019. 238 с.

14. Artificial Intelligence: A Modern Approach third edition, Global Edition, Russel S. Norvig P.: New York: Pearson. 2016. 168с с.

15. Efficient Estimation of Word Representations in Vector Space. URL: https://arxiv.org/abs/1301.3781 (дата звернення: 04.09.2023).

16. Natural Language Processing. URL: https://www.ibm.com/topics/natural-language-processing (дата звернення: 04.09.2023).

17. Foundations of Statistical Natural Language Processing, Manning C. Schutze H.: New York: Random House Publishing Group. 1999. 185 с.

18. Machine Learning (McGraw-Hill International Editions Computer Science Series), Mitchell T.: Washington: McGraw-Hill. 1997. 325 с.

19. Jurgen Schmidhuber Lugano Unversity. URL: https://arxiv.org/pdf/1404.7828v4.pdf (дата звернення: 10.09.23)

20. JavaScript. URL: https://astwellsoft.com/uk/blog/tehnology/javascript.html (дата звернення: 05.09.2023).

21. React 18 Design Patterns and Best Practices: Design, build, and deploy production-ready web applications with React by leveraging industry-best practices 4th Edition, Roldan C.: Washington: Packt Publishing. 2023. 320 с.

22. Angular: Up and Running: Learning Angular, Step by Step 1st Edition, Seshadri S.: Springfield: O'Reilly. 2018. 218 с.

23. Vue.js: Up and Running: Building Accessible and Performant Web Apps 1st Edition, Macrae C.: Springfield: O'Reilly. 2018. 119 с.

24. TypeScript. URL: https://www.typescriptlang.org/ (дата звернення: 05.09.2023).

25. Node.js: The Comprehensive Guide to Server-Side JavaScript Programming, Springer S.: Bonn: Rheinwerk Computing. 2022. 834 с.

26. JSON Web Token URL: https://www.json.org/json-ua.html (дата звернення: 10.09.23).

27. Nest.js URL: https://docs.nestjs.com/security/authentication (дата звернення: 10.09.23)

28. PostgreSql URL: https://www.postgresql.org/ (дата звернення: 10.09.23).

29. WebSocket URL: https://developer.mozilla.org/en/docs/Web/API/WebSocket (дата звернення: 10.09.23).

30. How Google Tests Software Уітакер Д. Каролло Д. / Уітакер Д. Каролло Д. M.: Addison-Wesley Professional, 2012. 156 с.

31. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. Вінниця : ВНТУ, 2022. – 50с.

102

Додаток А
(обов'язковий)

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

ЗАТВЕРДЖУЮ
Завідувач кафедрою ПЗ
д.т.н., проф.
_____ О. Н. Романюк
"20" вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Методи та програмні засоби створення месенджера з елементами штучного інтелекту» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доцент В. П. Майданюк
"20" вересня 2023 р.
Виконав:
_____ студент гр. 1ПІ-22м А. В. Грабарчук
"20" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи та програмні засоби створення месенджера з елементами штучного інтелекту».

Галузь застосування – месенджери, комунікація через Інтернет.

2. Підстава для розробки.

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

3. Мета та призначення розробки.

Метою роботи є підвищення рівня автоматизації пріоритезації повідомлень та виявлення спаму шляхом застосування методів штучного інтелекту.

Робота призначена для забезпечення спілкування з використанням мережі Інтернет.

4. Вихідні дані для проведення НДР

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

1. Грабарчук А. В. Ідентифікація об’єктів на основі Google Cloud Vision API / Л. Г. Коваль, В. П. Майданюк. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/158
2. Грабарчук А.В., Майданюк В. П. Застосування методів штучного інтелекту в системах обміну повідомленнями / Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – 336 с. - С. 87-88.

3. Artificial Intelligence: A Modern Approach third edition, Global Edition, Russel S. Norvig P.: New York: Pearson. 2016. 168c c.
4. Efficient Estimation of Word Representations in Vector Space. URL: https://arxiv.org/abs/1301.3781 (дата звернення: 04.09.2023).
5. Natural Language Processing. URL: https://www.ibm.com/topics/natural-language-processing (дата звернення: 04.09.2023).
6. Foundations of Statistical Natural Language Processing, Manning C. Schutze H.: New York: Random House Publishing Group. 1999. 185 c.
7. Machine Learning (McGraw-Hill International Editions Computer Science Series), Mitchell T.: Washington: McGraw-Hill. 1997. 325 c.

5. Технічні вимоги

- середовище розробки – WebStorm, PyCharm;
- мова програмування – JavaScript, TypeScript, Python;
- тип нейронної мережі – Deep Neural Network;
- Операційна система – кросплатформена;
- час навчання мережі – 180 сек.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

1. Пояснювальна записка до МКР;
2. Технічне завдання;
3. 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	Обґрунтування вибору методу розробки та постановка задач	20.09.23 – 02.10.23
2	Розробка архітектури та алгоритмів програмного продукту	03.10.23 – 23.10.23
3	Розробка архітектури нейронної мережі	16.10.23 – 23.10.23
4	Розробка програмного продукту	24.10.23 – 13.11.23
5	Тестування програми	14.11.23 – 21.11.23
6	Розробка економічної частини	17.11.23 –25.11.23
7	Оформлення матеріалів до захисту МКР	22.11.23 – 01.12.23

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

106

Додаток Б
(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Майданюк В. П.

Оригінальність	96,7 %
Схожість	3,3 %

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.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Грабарчук А. В.

Керівник роботи Майданюк В. П.

Додаток В

(довідниковий)

Лістинг програмного коду клієнтської частини

App.js

```
import {Route, Routes, BrowserRouter as Router} from "react-router-dom";
import ChatsPage from "./pages/chatsPage";
import LoginPage from "./pages/login";

function App() {
 return (
  <div className="max-w-screen-xl bg-[#f8fafc] w-[95%] mx-auto py-[1%] h-screen">
    <Router>
       <Routes>
          <Route path={'/'} element={<ChatsPage/>} />
          <Route path={'login'} element={<LoginPage/>} />
       </Routes>
    </Router>
  </div>
 );
}

export default App;
```

ChatsPage.js

```
import UserInfo from "../components/userInfo";
import ChatsList from "../components/chatsList/chatsList";
import ChatComponent from "../components/chatComponent/chatComponent";
```

```
export default function ChatsPage() {
  return (
    <div className={'flex justify-center'}>
      <div className={'flex flex-col w-1/3 min-w-[280px] max-w-[321px]'}>
        <UserInfo />
        <ChatsList />
      </div>
      <div className={'w-2/3 mt-[75px] ml-[1%]'}>
        <ChatComponent/>
      </div>
    </div>
  )
}
```

Login.js

```
export default function LoginPage() {
  const fixedInputClass="mb-[16px] rounded-md appearance-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 focus:outline-none focus:ring-purple-500 focus:border-purple-500 focus:z-10 sm:text-sm"
  return <div className="w-1/3 flex justify-center flex-col ">
    <div className="border-solid border-2 p-3 rounded-xl border-indigo-600">
      <p className="mb-[32px] text-2xl">Login</p>
      <input className={fixedInputClass} placeholder="email"/>
      <input className={fixedInputClass} placeholder="password"/>
      <button className="bg-transparent hover:bg-blue-500 text-blue-700 font-semibold hover:text-white py-2 px-4 border border-blue-500 hover:border-transparent rounded">Confirm</button>
    </div>
  </div>
```

```
}
```

UserInfo.js

```
export default function UserInfo() {

  return (
    <div className={'flex flex-col w-full bg-greybg p-[3%] border-b-2 border-greyborder'}>
      <img alt={'avatar'} className={'w-1/12'} src={'/assets/img_no_avatar.png'}/>
      <input className={'mt-[3%] rounded-xl w-[90%]'}/>
    </div>
  )
}
```

ChatsList.js

```
import ChatItem from "../chatItem/chatItem";
import {useEffect, useState} from "react";
import axios from "axios";
import {useDispatch, useSelector} from "react-redux";
import {setCommunicatorUsersList} from '../../redux/actions';

export default function ChatsList() {
  const users = useSelector((state) => state.userReducer.comunicatorUsersList);
  const dispatch = useDispatch();
  console.log('users', users)
  // const [users, setUsers] = useState([])
  useEffect(() => {
    async function fetchData() {
      const result = [{communicatorId: 1, name: "John", surname: "Peterson", messages: [{text:
'hi'}]}, {communicatorId: 2, name: "John1", surname: "Talkieng", messages: [{text: 'Glad to see you'}]},
```

```
{communicatorId: 2, name: "Alex", surname: "Fisher", messages: [{text: 'Yes I have read it'}]},
{communicatorId: 2, name: "John1", surname: "Talkieng", messages: [{text: 'Congratulations'}]}]
//(await axios.get('history')).data;
                console.log('result', result)
                dispatch(setCommunicatorUsersList(result))
            }
            if(users.length === 0) {
                fetchData();
            }
        }, [dispatch, setCommunicatorUsersList, axios])
        console.log('dispatch users', users);
        return <div className={'flex flex-col p-[3%] '}>

            <h1 className={'my-[15px] text-[#3b82f6] text-xl'}>Chats</h1>
            { users.map((user) => <ChatItem key={user.communicatorId} userInfo={user} />)}
        </div>
    }
```

ChatComponent.js

```
import MsgsList from "../msgsList/msgsList";
import SendMsgComponent from "../sendMsgComponent/sendMsgComponent";


export default function ChatComponent() {
    return (
        <div className={'flex flex-col'}>
            <MsgsList/>
            <SendMsgComponent/>
        </div>
    )
}
```

sendMsgComponent.js

```
export default function SendMsgComponent() {
  return <div className="w-[99%] mt-[75px] flex justify-center">
    <input className='w-[90%] mb-[16px] rounded-md appearance-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 focus:outline-none focus:ring-purple-500 focus:border-purple-500 focus:z-10 sm:text-sm' placeholder="print your text" />
    <button className="max-h-[38px] bg-transparent hover:bg-blue-500 text-blue-700 font-semibold hover:text-white ml-3 py-[1px] px-1 border border-blue-500 hover:border-transparent rounded">Send</button>
  </div>
}
```

ChatItem.js

```
import {useDispatch} from "react-redux";
import {setCommunicatorId} from "../../redux/actions";

export default function ChatItem({userInfo}) {
  // const testCount = useSelector((state) => state.userReducer.testCount)
  const dispatch = useDispatch();
  const {name, surname, communicatorId} = userInfo;
  const lastMsg = userInfo?.messages[userInfo?.messages?.length - 1] || {text: 'are you here'};
  return (<div onClick={() => {dispatch(setCommunicatorId(communicatorId));}} className={'flex justify-between border-b-2 border-greyborder py-[2%] min-h-30 min-w-[100%] cursor-pointer'}>
    <div className={'flex justify-between max-w-[50%]'}>
      <img className={'w-1/6 h-8 mr-2'} src={'/assets/img_no_avatar.png'}/>
      <div className={'flex flex-col'}>
        <p className={'font-bold'}>
```

```
                {name} {surname}
              </p>
              <p className={'text-xs'}>{lastMsg.text}</p>
            </div>
        </div>
        <p className={'text-xs'}>Feb 18, 2022</p>
    </div>)
}
```

actions.js

```
export function setCommunicatorId(communicatorId) {
  return {
      type: 'setCurrentCommunicatorUser',
      payload: {
          communicatorId
      }
  }
}

export function setCommunicatorUsersList(usersList) {
  return {
      type: 'setCommunicatorUsersList',
      payload: {
          usersList
      }
  }
}
```

userReducer.js

```
const initialState = {
  userId: 4,
```

```
  comunicatorUsersList: [],
  currentCommunicatorUser: null,
};

export const userReducer = (state=initialState, action) => {
  switch (action.type) {
    case 'setUserId':
      return {...state, userId: action.payload.userId}
      break;
    case 'setCurrentCommunicatorUser':
      if (!action.payload.communicatorId && action.payload.communicatorId !== 0) {
        return state;
      }
      // const newState = {...state};
      state.currentCommunicatorUser = action.payload.communicatorId;
      return state;
      break;
    case 'setCommunicatorUsersList':
      state.comunicatorUsersList = action.payload.usersList;
      console.log('dispatched', action.payload.usersList)
      return {...state};
    default:
      return state;
  }
}
```

Додаток Г

(довідниковий)

Лістинг програмного коду серверної частини

Main.ts

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
 const app = await NestFactory.create(AppModule);
 await app.listen(3005);
}
bootstrap();
```

App.module.ts

```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { Chats } from "./entities/Chats";
import { Messages } from "./entities/Messages";
import { Users } from "./entities/User";
import { LoginModule } from './modules/login/login.module';
import { MessagesModule } from "./modules/messages/messages.module";

@Module({
 imports: [TypeOrmModule.forRoot({
   "type": "postgres",
   "host": "localhost",
   "port": 5432,
   "username": "postgres",
   "password": "pwd1",
   "database": "recruit_test",
   "entities": [Users, Chats, Messages],
```

```
  "synchronize": false,}
 ), LoginModule, MessagesModule],
 controllers: [],
 providers: [],
})
export class AppModule {}
```

JwtAuthGuard.ts

```
export class JwtAuthGuard extends AuthGuard('jwt') {
 canActivate(context: ExecutionContext) {
  return super.canActivate(context);
 }
}
```

WsJwtAuthGuard.ts

```
export class WsJwtAuthGuard extends AuthGuard('jwt') {
 canActivate(context: ExecutionContext) {
  const token = context.switchToWs().getClient()?.handshake?.query.token;
  const request = context.switchToHttp().getRequest();
  request.headers = {authorization: token}
  return super.canActivate(context);
 }
}
```

JwtStrategy.ts

```
@Injectable()
export class JwtStrategy extends PassportStrategy(Strategy) {
 constructor() {
  super({
   jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
   ignoreExpiration: false,
```

```
    secretOrKey: 'test15'
  });
 }
 async validate(payload) {
  console.log('validate in strategy');
  return {userId: payload.sub, emailAddress: payload.emailAddress}
 }
}
```

AuthService.ts

```
@Injectable()
export class AuthService {
 @InjectRepository(Users)
 userRepository: Repository<Users>;
 constructor(private jwtService: JwtService) {}
 async validateUser(emailAddress: string, pass: string) {
  console.log('validate in service');
  const user = await this.userRepository.findOne({where: {emailAddress}});
  if(user && user.password === pass) {
    const {password, ...result} = user;
    return result;
  }
  return null;
 }
 async login(user) {
  const payload = {emailAddress: user.emailAddress, sub: user.userId}
  return {
    access_token: this.jwtService.sign(payload)
  }}}
```

LoginController.ts

```
@Controller('login')
export class LoginController {
 constructor(private loginService: LoginService,
          private authService: AuthService) {}
 @Post('signup')
 async signUpUser(@Body() signupReq: SignupReqDto): Promise<LoginResDto> {
  return await this.loginService.signUpUser(signupReq);
 }
 @Post()
 async loginUser(@Body() body) {
  const user = await this.authService.validateUser(body.email, body.password);
  return this.authService.login(user);
 }
 @UseGuards(JwtAuthGuard)
 @Get('profile')
 getProfile(@Request() req) {
  return req.user;
 }
}
```

LoginService.ts

```
@Injectable()
export class LoginService {

 @InjectRepository(Users)
 private usersRepository: Repository<Users>;

 async signUpUser(userData: SignupReqDto): Promise<LoginResDto> {
Return await this.usersRepository.save(this.usersRepository.create(userData));}
 async loginUser(userData: LoginReqDto) {
```

```
  return this.usersRepository.findOne({where: {emailAddress: userData.email, password:
userData.password}})}}
```

MessagesService.ts

```
@Injectable()
export class MessagesService {
 @InjectRepository(Users)
 usersRepository: Repository<Users>;
 @InjectRepository(Chats)
 chatsRepository: Repository<Chats>;
 @InjectRepository(Messages)
 msgsRepository: Repository<Messages>;
 async getAllChats(userId, step=0) {
  const chats: Chats[] = await this.chatsRepository.find({
   where: [
    {user1: userId},
    {user2: userId}
   ],
   order: {
    updatedAt: 'DESC'
   },
   skip: step*50,
   take: 50,
  });
  const result = await Promise.all(chats.map(async (chat: Chats) => {
   const communicatorInfo = await this.usersRepository.findOne(
    {
     select: {
      userId: true,
      name: true,
      surname: true,
      avatar_url: true
```

```
      },
      where: {userId: chat.user1 === userId ? chat.user2 : chat.user1}});
   const lastMsg = await this.msgsRepository.findOne({
     where: {chatId: chat.chatId},
     order: {
       createdAt: "DESC"
     }
   });
      console.log(chat.chatId)
   return {chatId: chat.chatId, lastMsg, communicatorInfo};
 })
 );
 return result;
}

async getMsgs(chatId, userId, step= 0) {
 const chat = await this.chatsRepository.findOne({where: [
    {chatId, user1: userId},
    {chatId, user2: userId}
  ]});

 if(!chat) {
   throw new WsException({ code: 401, msg: "unauthorized" })
 }

 const msgs = await this.msgsRepository.find({
   where: { chatId },
   skip: step * 50,
   take: 50,
   order: {createdAt: "DESC"}
 });
```

```
  return msgs;
 }
 async sendMsg(userId, data: SendMsgReqDto) {
  const {chatId, text} = data;
  const chat = await this.chatsRepository.findOne({where: [
     {chatId, user1: userId},
     {chatId, user2: userId}
   ]});
  if(!chat) {
    throw new WsException({ code: 401, msg: "unauthorized" })}
  const msg = this.msgsRepository.create({senderId: userId, ...data})
  const result = await this.msgsRepository.save(msg)
  return {msg: result, receiverId: chat.user1 == msg.senderId ? chat.user2 : chat.user1}}
 async createChat(user1, user2) {
  const chat = this.chatsRepository.create({user1, user2})
  return await this.chatsRepository.save(chat)}}
```

Msgs.ts

```
type IRequest =  Request & {
 user: UserInfo
}

export type UserInfo = {
 emailAddress: string,
 userId: string,
}
@WebSocketGateway(80, {path: '/' })
export class Msgs {
 activeUsers = {}
 makeOnline(sockedId, userId) {
  this.activeUsers[userId] = sockedId
```

```
 }
 constructor(private readonly msgsService: MessagesService) {
 }
 @WebSocketServer()
 server: Server;
 @UseGuards(WsJwtAuthGuard)
 @SubscribeMessage('getChats')
 async getChats(@ConnectedSocket() socket: Socket & IRequest,@MessageBody() data: GetChatsDto)
{
  this.makeOnline(socket.id, socket.user.userId)
  return {event: 'getChats', data: await this.msgsService.getAllChats(socket.user.userId)};
 }
 @UseGuards(WsJwtAuthGuard)
 @SubscribeMessage('getMsgs')
 async getMsgs(@ConnectedSocket() socket: Socket & IRequest, @MessageBody() data: {chatId:
number}) {
  this.makeOnline(socket.id, socket.user.userId)
  return {
   event: 'getMsgs',
   data: await this.msgsService.getMsgs(data.chatId, socket.user.userId)
  }
 }
 @UseGuards(WsJwtAuthGuard)
 @SubscribeMessage('createChat')
 async createChat(@ConnectedSocket() socket: Socket & IRequest, @MessageBody() data: {userId:
number}) {
  this.makeOnline(socket.id, socket.user.userId)
  return {
   event: 'createChat',
   data: await this.msgsService.getMsgs(data.userId, socket.user.userId)
  }
```

```
 }

 @UseGuards(WsJwtAuthGuard)
 @SubscribeMessage('sendMsg')
 async sendMsg(@ConnectedSocket() socket: Socket & IRequest, @MessageBody() data:
SendMsgReqDto) {
   this.makeOnline(socket.id, socket.user.userId)
   const result = await this.msgsService.sendMsg(socket.user.userId, data)
   if(this.activeUsers[result.receiverId])
this.server.to(this.activeUsers[result.receiverId]).emit('sendMsg', result.msg)
     return {
       event: 'sendMsg',
       data: result.msg
     };}}
```

UserActionsController.ts

```
@Controller('user-actions')
export class UserActionsController {
 constructor(private userActionsService: UserActionsService) {
 }
 @Post('findUser')
 async findUser(@Body() body) {
   return await this.userActionsService.findUser(body.emailAddress);
 }}
```

UserActionsService.ts

```
@Injectable()
export class UserActionsService {
 @InjectRepository(Users)
 private readonly usersRepository: Repository<Users>;
 async findUser(emailAddress: string) {
   return await this.usersRepository.findOne({select: {
       userId: true,
```

```
      emailAddress: true,
      avatar_url: true,
      name: true,
      surname: true
    },
  where: {emailAddress}});
}}
```

UserActionsModule.ts

```
import { Module } from '@nestjs/common';
import { UserActionsController } from './user-actions.controller';
import { UserActionsService } from './user-actions.service';
import { TypeOrmModule } from "@nestjs/typeorm";
import { Users } from "../../entities/User";
@Module({
 imports: [TypeOrmModule.forFeature([Users])],
 controllers: [UserActionsController],
 providers: [UserActionsService]
})
export class UserActionsModule {}
```

PriorityService .ts

```
@Injectable()
export class PriorityService {
  constructor(private readonly http: HttpService) {}
  async getPriority(text: string) {
    const response = await this.http.post(
      'https://localhost:3005/priority',
      { text });
    return response.body.priority;}}
```

Додаток Д

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ СТВОРЕННЯ МЕСЕНДЖЕРА З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Магістерська кваліфікаційна робота

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ СТВОРЕННЯ МЕСЕНДЖЕРА З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: Грабарчук Антон Володимирович (1ПІ-22м)
Керівник: к.т.н., доц. каф. ПЗ Майданюк В. П.
Опонент: к.т.н., доц. каф. ОТ Кожемяко А. В.

Вінниця 2023

Рисунок Д.1 - Слайд презентації №1

Актуальність дослідження

У сучасному світі, на фоні стрімкого розвитку інформаційних технологій та зростаючої ролі комунікаційних платформ, штучний інтелект визначає новий рівень функціональності та взаємодії у сфері месенджерів. Завдяки небаченим досі можливостям аналізу даних, машинного навчання та обробки природної мови, використання штучного інтелекту у месенджерах стає ключовим фактором у вдосконаленні способів спілкування, надаючи користувачам інноваційні та персоналізовані рішення.

Вивчення цієї теми має стратегічне значення для розвитку месенджерів, оскільки дозволяє не лише поліпшити користувацький досвід, а й створює нові можливості для бізнесу та соціальної взаємодії.

Рисунок Д.2 - Слайд презентації №2

Мета та завдання

Мета: підвищення рівня автоматизації пріоритизації повідомлень та виявлення спаму шляхом застосування методів штучного інтелекту.

Основними задачами дослідження є:

- Проаналізувати поняття обміну повідомленнями та месенджерів.
- Провести аналіз існуючих систем обміну повідомленнями.
- Спроєктувати архітектуру розроблюваної системи.
- Обрати тип штучного інтелекту.
- Провести тренування алгоритму машинного навчання.
- Розробити дизайн системи.
- Розробити код додатка.
- Провести тестування створеної системи.
- Виконати розрахунки доведення економічної доцільності розробки.

Рисунок Д.3 - Слайд презентації №3

Об'єкт, предмет та методи дослідження

Об'єктом дослідження - процес обміну повідомленнями в Інтернет.

Предметом дослідження – методи та програмні засоби створення месенджера з елементами штучного інтелекту.

Методи дослідження. У процесі досліджень використовувались: основи теорії штучного інтелекту, теорія розпізнавання образів, теорія алгоритмів, методи WEB-програмування для розробки моделей та методів побудови архітектури програмного забезпечення месенджера з елементами штучного інтелекту; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Рисунок Д.4 - Слайд презентації №4

Наукова новизна

1. Вперше запропоновано метод класифікації повідомлень в месенджерах, особливістю якого є застосування штучного інтелекту для пріоритезації повідомлень та виявлення шкідливих, що підвищує зручність використання месенджера.
2. Подальшого розвитку отримав метод обробки текстової інформації, у якому, на відміну від існуючих, використано NLP технологію, що дозволяє сформувати векторний простір для подальшої пріоритезації повідомлень.

Рисунок Д.5 - Слайд презентації №5

Практична цінність отриманих результатів

Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби підвищення ефективності використання месенджерів і захисту користувачів.

Рисунок Д.6 - Слайд презентації №6

Існуючі системи

Telegram — месенджер, відомий своїм високим ступенем шифрування повідомлень, широким функціоналом, таким як секретні чати та канали, а також можливістю надсилати великі файли. Зручний інтерфейс та надійність роблять його популярним серед користувачів з усього світу.

WhatsApp — месенджер, який відзначається простим інтерфейсом, безкоштовними текстовими, голосовими та відеовикликами через Інтернет. Забезпечує ефективне шифрування повідомлень для забезпечення приватності користувачів. Є одним із найпопулярніших месенджерів у світі.

Signal — відомий своєю високою конфіденційністю та безпекою. Він використовує end-to-end шифрування для всіх видів повідомлень та викликів, забезпечуючи високий рівень приватності користувачів. Signal також відомий своєю відкритістю та відсутністю рекламних елементів, фокусуючись на забезпеченні безпечного та анонімного зв'язку.

Рисунок Д.7 - Слайд презентації №7

Недоліки існуючих систем

Основним недоліком існуючих систем є відсутність інтеграції штучного інтелекту, і як наслідок недостатня ефективність використання цих додатків, а також слабкий захист від шкідливих повідомлень.

Рисунок Д.8 - Слайд презентації №8

Метод застосування класифікації повідомлень

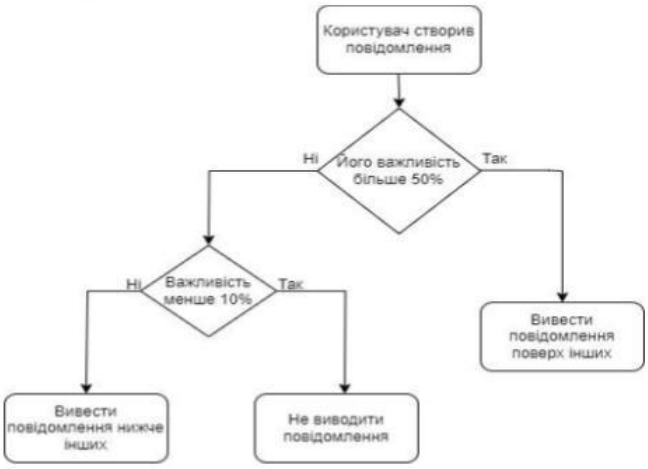


Рисунок Д.9 - Слайд презентації №9

Розробка архітектури додатку

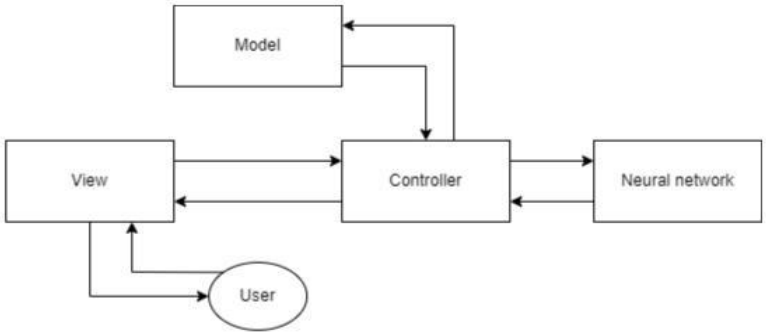


Рисунок Д.10 - Слайд презентації №10

Розробка архітектури бази даних

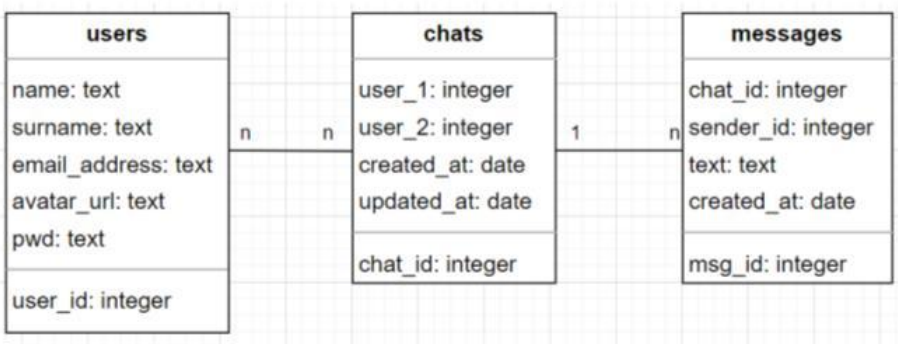


Рисунок Д.11 - Слайд презентації №11

Математична модель алгоритму розпізнавання природної мови

$$V = \frac{\text{кількість конкретного слово}}{\text{загальна кількість слів}}$$

$$R = log(\frac{\text{загальна кількість слів}}{\text{кількість конкретного слова}}) + 1$$

$$VR = V \times R$$

Рисунок Д.12 - Слайд презентації №12

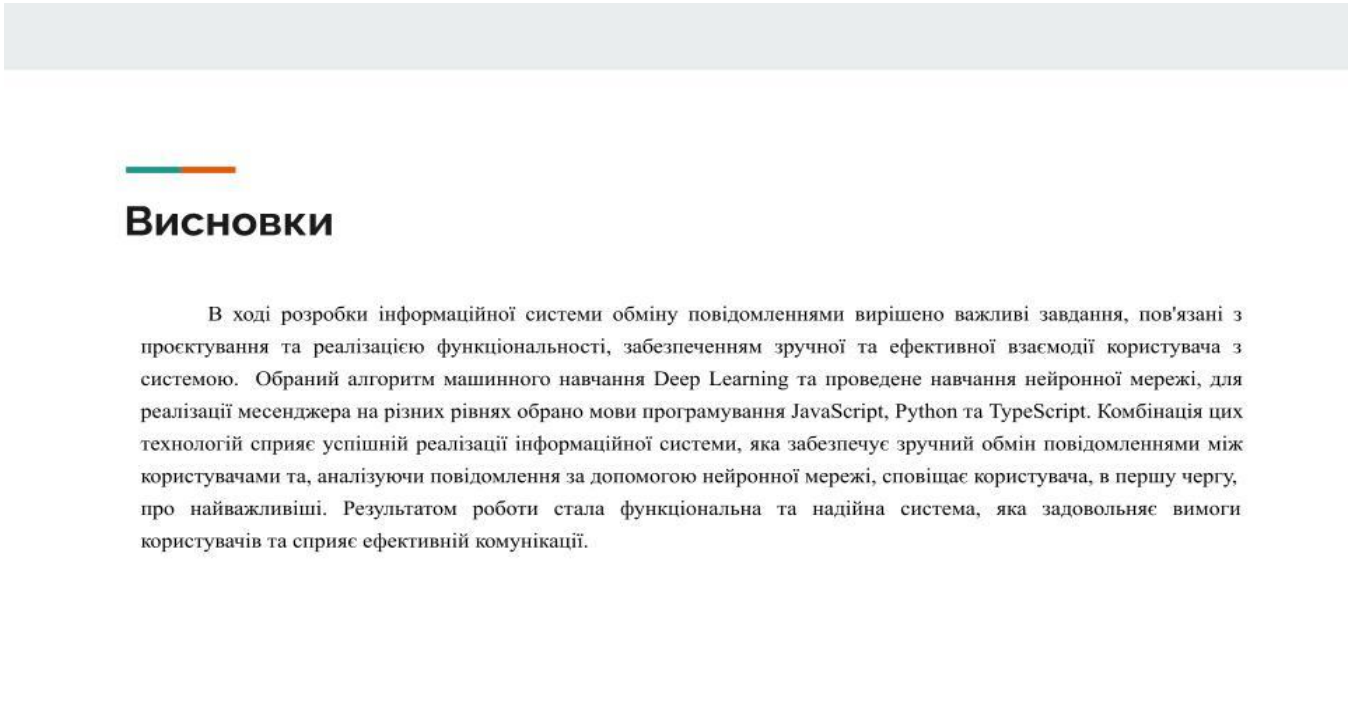


Рисунок Д.13 - Слайд презентації №13

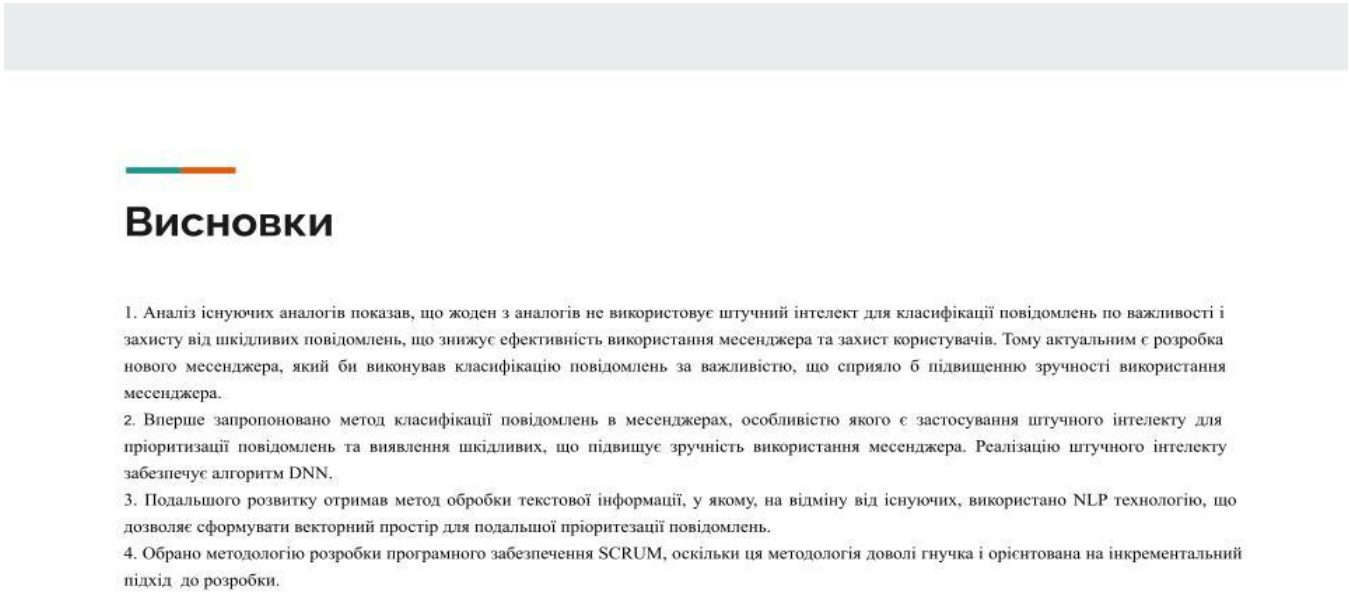


Рисунок Д.14 - Слайд презентації №14

Висновки

В якості архітектури додатку обрано архітектуру MVC, оскільки ця архітектура ефективно відокремлює Business Logic від користувальницького інтерфейсу програми.

6. В якості СУБД обрано СУБД PostgreSQL, оскільки ця СУБД характеризується надійністю та відкритим вихідним кодом. З використанням СУБД PostgreSQL розроблено структуру бази даних повідомлень та користувачів.

7. Для реалізації клієнтської частини месенджера обрано мову програмування JavaScript, технології React та Redux, оскільки ці інструменти забезпечують простий та ефективний підхід до створення компонентів, характеризуються наявністю сторонніх бібліотек. З використанням цих інструментів виконано розробку коду для фронтенд частини додатку.

8. Серверну частину додатку розроблено мовою програмування Typescript з використанням фреймворку Nest.js та бібліотеку взаємодії з базою даних TypeORM, що дозволило швидко розробити код.

9. Тестування додатку показало його працездатність та відповідність завданню на роботу.

Рисунок Д.15 - Слайд презентації №15

Дякую за увагу

Рисунок Д.14 - Слайд презентації №14