Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Кузнєцової Ганни Олексіївни*
(ПІБ)

академічної групи *122-20з-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-додатку для надання та отримання психологічної підтримки з використанням мови програмування Typescript*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Алексєєв М.О.*			
розділів:				
спеціальний	*проф. Алексєєв М.О.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 202 4 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20з-1* *Кузнєцової Г. О.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-додатку для надання та та абітурієнтів «НТУ Дніпровська політехніка» отримання психологічної підтримки з використанням мови програмування Typescript*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *470-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2024 р***

Завдання видав _____ *проф. Алексєєв М.О.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняла до виконання _____ *Кузнєцова Г.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 61 с., 11 рис., 0 табл., 2 дод., 30 джерел.

Об'єктом дослідження є платформа для анонімного спілкування та надання психологічної підтримки, а предметом дослідження - розробка такої платформи.

Метою кваліфікаційної роботи є створення сучасної, надійної та зручної платформи, що дозволяє користувачам анонімно обмінюватися досвідом, отримувати та надавати психологічну підтримку. Це дозволить зменшити відчуття ізоляції та підвищити психологічне благополуччя користувачів в умовах національної кризи, спричиненої війною.

Методи дослідження та апаратура: використано сучасні веб-технології, такі як Nest.js для серверної частини, React.js для клієнтської частини, та MongoDB як база даних. Для тестування та розгортання застосовувались інструменти Visual Studio Code та хмарні сервіси.

Основні конструктивні, технологічні й техніко-експлуатаційні характеристики та показники:

1. Платформа забезпечує анонімність користувачів.
2. Реєстрація та авторизація відбувається за допомогою електронної пошти та паролю.
3. Миттєвий обмін повідомленнями у реальному часі.
4. Автоматичний підбір співбесідників.
5. Зберігання історії спілкування у базі даних.
6. Використання технологій шифрування для захисту даних.

Практичне значення роботи полягає у можливості впровадження її результатів для надання психологічної підтримки широкому колу користувачів в умовах кризи, підвищуючи рівень їхнього емоційного благополуччя.

Прогнозні припущення про розвиток об’єкту дослідження або розробки: Очікується, що платформа буде використовуватися як індивідуально громадянами, так і інтегруватися в програми соціальної підтримки державними та недержавними організаціями. Можливі подальші покращення функціоналу та розширення можливостей платформи.

Перелік ключових слів: ПЛАТФОРМА ДЛЯ СПІЛКУВАННЯ, ПСИХОЛОГІЧНА ПІДТРИМКА, Nest.js, React.js, MongoDB.

ABSTRACT

Explanatory note: 61 pages, 11 figures, 0 tables, 2 appendices, 30 sources.

The object of the research is a platform for anonymous communication and providing psychological support, and the subject of the research is the development of such a platform.

The purpose of the qualification work is to create a modern, reliable, and convenient platform that allows users to anonymously share experiences, receive and provide psychological support. This will reduce the feeling of isolation and improve the psychological well-being of users in the context of a national crisis caused by the war.

Research methods and equipment: modern web technologies such as Nest.js for the server part, React.js for the client part, and MongoDB as the database were used. Tools such as Visual Studio Code and cloud services were used for testing and deployment.

Main design, technological, and technical-operational characteristics and indicators:

1. The platform ensures user anonymity.
2. Registration and authorization are carried out using email and password.
3. Instant messaging in real-time.
4. Automatic selection of interlocutors.
5. Storage of communication history in the database.
6. Use of encryption technologies for data protection.

The practical significance of the work lies in the possibility of implementing its results to provide psychological support to a wide range of users in crisis conditions, thereby improving their emotional well-being.

Forecast assumptions about the development of the research object or development: It is expected that the platform will be used both individually by citizens and integrated into social support programs by governmental and non-governmental organizations. Further improvements in functionality and expansion of the platform's capabilities are possible.

List of keywords: COMMUNICATION PLATFORM, PSYCHOLOGICAL SUPPORT, Nest.js, React.js, MongoDB.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - База даних

DI - Dependency Injection (ін'єкція залежностей)

HTML - HyperText Markup Language (мова розмітки гіпертексту)

JWT - JSON Web Token (токен на основі JSON)

SHA-256 - Secure Hash Algorithm 256-bit (алгоритм хешування)

SPA - Single Page Application (односторінковий застосунок)

VS Code - Visual Studio Code

W3C - World Wide Web Consortium (Всесвітній веб-консорціум)

WS - WebSocket (веб-сокет)

BSON - Binary JSON (бінарний формат JSON)

AppConfig - Application Configuration (конфігурація додатку)

HTTP - Hypertext Transfer Protocol (протокол передачі гіпертексту)

REST - Representational State Transfer (передача репрезентативного стану)

CSS - Cascading Style Sheets (каскадні таблиці стилів)

ВСТУП

В умовах повномасштабного вторгнення Росії в Україну, українське суспільство стикається з непростими викликами, серед яких одним з найбільш нагальних є забезпечення психологічної підтримки людей, які зазнали моральних травм та стресових станів. Наслідки війни непокоять не тільки тих, хто безпосередньо зіткнувся з військовими діями, але й широке коло громадян, кожен з яких переживає цю кризу по-своєму. У такі часи потреба в кваліфікованій психологічній допомозі значно перевищує можливості існуючої системи охорони здоров'я.

Розуміючи гостроту цієї проблеми, ми вирішили створити платформу для спілкування, яка надає можливість людям анонімно ділитися своїми переживаннями, знаходити взаєморозуміння та підтримку в середовищі однодумців. Ця платформа покликана стати містком між тими, хто шукає підтримки, та тими, хто готовий її надавати, зменшуючи тим самим відчуття ізоляції та безсилля перед обличчям загальнонаціональної кризи.

Проблема психологічної підтримки є актуальною для фахівців напряму "Комп'ютерні науки", оскільки інформаційні технології можуть значно сприяти розвитку систем підтримки, які забезпечують взаємодію між людьми. Використання сучасних технологій дозволяє створювати ефективні рішення для підтримки психологічного здоров'я населення, що є важливою складовою соціальної стабільності.

Метою кваліфікаційної роботи є створення сучасної, надійної та зручної платформи, що дозволяє користувачам анонімно обмінюватися досвідом, отримувати та надавати психологічну підтримку. Галузь застосування цієї платформи охоплює як індивідуальне використання громадянами, так і інтеграцію в програми соціальної підтримки, що здійснюються державними та недержавними організаціями.

Актуальність теми обумовлена поточною ситуацією в Україні, де значна кількість людей потребує психологічної підтримки через наслідки військових дій. Традиційні методи надання психологічної допомоги часто не здатні охопити

всі верстви населення, особливо в умовах обмежених ресурсів. Використання інформаційних технологій для створення платформи анонімної психологічної підтримки дозволяє значно розширити доступність допомоги.

Конкретизація постановки завдання включає розробку веб-додатку, який забезпечує безпечне зберігання даних користувачів та їхню анонімність, швидкий обмін повідомленнями, а також інтеграцію з іншими сервісами для забезпечення безперервної підтримки. Додаток повинен включати ефективні механізми для реєстрації, входу, пошуку співбесідників та відправлення повідомлень.

Розроблений з використанням сучасних технологій, включаючи Nest.js для серверної частини, React.js для клієнтської частини, та MongoDB як базу даних, наш застосунок є надійним і зручним інструментом для забезпечення емоційної підтримки та психологічної допомоги. У цьому документі ми детально описуємо архітектуру застосунку, його компоненти та принципи взаємодії між ними, прагнучи забезпечити легкість використання та доступність сервісу для всіх, хто потребує допомоги в цей складний час.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

На сучасному ринку існує низка застосунків та платформ, спрямованих на надання психологічної допомоги. Більшість із цих рішень зосереджена на з'єднанні користувачів із професійними психологами, пропонуючи можливості для особистих консультацій, як платних, так і безкоштовних. Такі платформи відіграють важливу роль у забезпеченні доступу до кваліфікованої психологічної допомоги, особливо в умовах глобальних криз та пандемій.

Проте, хоча ці рішення і є цінними, вони не в повній мірі задовольняють потреби в психологічній підтримці всіх верств населення. Основна проблема полягає в тому, що доступ до професійних психологів може бути обмеженим через фінансові бар'єри, обмежену кількість спеціалістів, а також через стигматизацію психічного здоров'я, що ускладнює звернення за допомогою.

Наше рішення спрямоване на створення платформи для анонімного спілкування та надання психологічної підтримки, яка дозволяє користувачам обмінюватися досвідом і отримувати підтримку від інших учасників спільноти. Це дозволяє забезпечити більш доступну і масову психологічну підтримку, що є особливо важливим в умовах національної кризи, спричиненої війною.

Назва програми: Платформа для анонімного спілкування та надання психологічної підтримки.

Характеристика галузі застосування: Програма призначена для використання у галузі психологічної підтримки та соціальної адаптації. Вона може бути застосована як інструмент для підтримки емоційного благополуччя населення в умовах кризових ситуацій.

Об'єкт використання програми: Платформа використовується широким колом користувачів, які шукають психологічну підтримку або готові надавати її

іншим. Вона забезпечує зручний та безпечний простір для анонімного обміну повідомленнями і досвідом між користувачами.

Таким чином, наша платформа розширює можливості психологічної підтримки, пропонуючи новий підхід, який компліментарний до існуючих професійних психологічних служб та водночас відкриває додаткові шляхи для психологічної допомоги.

1.2. Призначення розробки та область застосування

Повна назва розробленої системи: Supportie - платформа для анонімного спілкування та надання психологічної підтримки.

Причини виникнення необхідності розробки програмного забезпечення: В умовах сучасних викликів, зокрема в умовах війни, значно зростає потреба у психологічній підтримці населення. Існуючі сервіси часто не здатні повністю задовольнити цю потребу через обмежену кількість спеціалістів, фінансові бар'єри та стигматизацію психічного здоров'я. Розробка анонімної платформи для психологічної підтримки дозволяє знизити ці бар'єри, забезпечуючи доступність допомоги для широкого кола користувачів.

Галузі застосування системи:

1. Особисте використання: підтримка емоційного благополуччя індивідуальних користувачів;
2. Криза: надання допомоги під час кризових ситуацій для зниження стресу та покращення психологічного стану;
3. Взаємопідтримка: сприяння створенню мережі взаємної підтримки серед користувачів.

Система забезпечує зручний та безпечний простір для обміну повідомленнями та досвідом між користувачами, сприяючи покращенню їх емоційного благополуччя.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

- Освітня програма за спеціальністю 122 «Комп’ютерні Науки»;
- Графік навчального процесу та навчальний план;
- Наказ ректора Національного технічного університету «Дніпровська політехніка» № 470-с від 23.05.2024 р.;
- Завдання на кваліфікаційну роботу на тему «Розробка веб-додатку для надання та отримання психологічної підтримки з використанням мови програмування Typescript”.

1.4. Постановка завдання

Метою даної роботи є розробка веб-додатку для надання та отримання психологічної підтримки з використанням мови програмування TypeScript. Додаток повинен забезпечити користувачам можливість анонімного обміну досвідом, отримання та надання психологічної підтримки, що зменшить відчуття ізоляції та підвищить психологічне благополуччя користувачів в умовах національної кризи, спричиненої війною.

Техніко-економічна сутність завдання та обґрунтування доцільності його рішення:

Розробка веб-додатку дозволяє забезпечити доступ до психологічної підтримки широкому колу користувачів. Використання сучасних технологій (Nest.js, React.js, MongoDB) сприяє створенню надійного та зручного сервісу. Ефективність рішення обумовлена можливістю масштабування та інтеграції з іншими сервісами, що дозволяє підтримувати зростаючий потік користувачів та забезпечувати їхні потреби.

Система буде використовуватися для управління психологічною підтримкою серед наступних об'єктів:

- Користувачі, що шукають психологічну підтримку;

- Користувачі, що надають психологічну підтримку.

Система складається з наступних компонентів:

1. Серверна частина: реалізована на фреймворку Nest.js, відповідає за обробку запитів, управління даними та забезпечення безпеки;
2. Клієнтська частина: реалізована на React.js, надає інтерфейс користувачам для взаємодії з системою;
3. База даних: MongoDB для зберігання даних користувачів, чатів та повідомлень.

Показники, що характеризують стан системи:

1. Кількість активних користувачів;
2. Кількість створених чатів;
3. Середній час відповіді системи;
4. Рівень завантаження сервера.

Вихідна інформація включає:

1. Звіти про активність користувачів;
2. Логи помилок та інцидентів;
3. Статистика використання ресурсу.

Вхідна інформація повинна збиратися та передаватися в режимі реального часу з використанням протоколів HTTPS та WebSocket. Всі дані повинні бути зашифровані та передаватися через захищені канали для забезпечення конфіденційності та цілісності інформації.

Автоматизоване рішення припиняється у випадках:

- Відсутність з'єднання з сервером;
- Виявлення критичних помилок в роботі системи;
- Необхідність оновлення або технічного обслуговування.

Система інтегрується з іншими психологічними сервісами та платформами для надання допомоги, що дозволяє розширити коло підтримки та забезпечити більш ефективну допомогу користувачам.

Розподіл функцій між персоналом і технічними засобами:

- Персонал: адміністратори слідкують за роботою системи та вирішують технічні питання;
- Технічні засоби: сервери та база даних забезпечують зберігання та обробку даних, клієнтські додатки забезпечують інтерфейс для взаємодії користувачів із системою.

Ця структура дозволяє ефективно організувати роботу системи, забезпечуючи високу якість послуг та задоволення потреб користувачів.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Платформа повинна забезпечувати такі функціональні можливості:

1. Анонімність: Забезпечення повної анонімності користувачів, щоб співбесідники не могли отримати персональну або ідентифікуючу інформацію одне про одного з програмного продукту;
2. Реєстрація та авторизація: Користувачі повинні мати можливість створювати акаунти та входити в систему, використовуючи псевдонім та електронну пошту;
3. Миттєвий обмін повідомленнями: Користувачі повинні мати змогу спілкуватися в реальному часі. Час отримання повідомлення не повинен перевищувати 1 секунди за умови стабільного інтернет-з'єднання;
4. Пошук співбесідника: Можливість автоматичного підбору співбесідника на основі вибраної ролі (шукає підтримку чи бажає підтримати);
5. Зберігання історії спілкування: Збереження всіх повідомлень для подальшого доступу до них.

1.5.2. Вимоги до інформаційної безпеки

Програма повинна забезпечувати високий рівень інформаційної безпеки, включаючи:

1. Захист даних облікових записів: Використання хешування паролів з додаванням солі для захисту від несанкціонованого доступу;
2. Захист даних при передачі: Використання протоколу HTTPS для шифрування даних під час їх передачі між клієнтом і сервером;
3. Захист від несанкціонованого доступу: Впровадження JWT (JSON Web Token) для авторизації та аутентифікації користувачів;
4. Контроль цілісності даних: Регулярний контроль та перевірка цілісності даних для запобігання їх спотворенню чи втраті.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для забезпечення належної роботи програмного виробу необхідні такі технічні засоби:

1. Сервер: Віртуальний приватний сервер (VPS) з мінімальними характеристиками: процесор з частотою 2 ГГц, 4 ГБ оперативної пам'яті, 50 ГБ SSD-накопичувач;
2. Клієнтські пристрої: ПК, ноутбуки, планшети та смартфони з веб-браузерами, які підтримують сучасні веб-технології;
3. База даних: MongoDB, що забезпечує зберігання даних у форматі BSON та підтримує горизонтальне масштабування.

1.5.4. Вимоги до інформаційної та програмної сумісності

Програмний виріб повинен забезпечувати сумісність із наступними інформаційними структурами та програмними засобами:

1. Мови програмування: Використання TypeScript для серверної частини

(Nest.js) та клієнтської частини (React.js);

2. Інтеграція з API: Підтримка REST API для взаємодії між клієнтською та серверною частинами;
3. Сумісність з протоколами: Підтримка HTTPS для захищеної передачі даних та WebSocket для обміну повідомленнями в реальному часі.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Функціональне призначення:

1. Анонімне спілкування: Користувачі можуть обмінюватися повідомленнями без розкриття особистої інформації, що сприяє створенню безпечного середовища для обговорення особистих проблем;
2. Реєстрація та авторизація: Система дозволяє створювати акаунти, входити в систему, використовуючи псевдонім та електронну пошту, забезпечуючи при цьому захист даних користувачів;
3. Миттєвий обмін повідомленнями: Користувачі можуть обмінюватися повідомленнями в режимі реального часу. Повідомлення передаються з мінімальною затримкою, що забезпечує ефективне спілкування;
4. Пошук співбесідника: Платформа автоматично підбирає співбесідника на основі вибраної ролі (шукає підтримку чи бажає підтримати).
5. Зберігання історії спілкування: Всі повідомлення зберігаються для можливості подальшого доступу до них користувачів;

Експлуатаційне призначення:

1. Забезпечення психологічної підтримки: Платформа надає можливість широкому колу користувачів отримувати та надавати психологічну підтримку, що особливо важливо в умовах кризових ситуацій;
2. Зниження витрат на психологічну допомогу: Використання платформи зменшує потребу в особистих консультаціях з психологами, що знижує витрати на такі послуги;
3. Доступність допомоги: Платформа забезпечує доступність психологічної підтримки для користувачів незалежно від їхнього місця знаходження;

4. Автоматизація процесів: Платформа автоматизує процес підбору співбесідників та обміну повідомленнями, що знижує потребу в людських ресурсах для організації таких процесів;
5. Покращення емоційного благополуччя: Використання платформи сприяє покращенню психологічного стану користувачів, зменшуючи відчуття ізоляції та стресу.

Ця система створює нові можливості для підтримки емоційного благополуччя населення, надаючи інструменти для взаємодії та підтримки в умовах соціальних та психологічних криз.

2.2 Опис застосованих математичних методів

У даній роботі не використовувалися математичні методи.

2.3 Опис використаних технологій та мов програмування

В розробці сучасних веб-застосунків вибір правильних технологій та інструментів є ключовим фактором, який визначає якість кінцевого продукту, його ефективність, безпеку та зручність користування. Розробка застосунку для анонімної психологічної підтримки, який призначений для надання можливості користувачам спілкуватися та підтримувати один одного у складні часи, потребує особливої уваги до деталей в плані архітектури, безпеки, та взаємодії з користувачем.

Нижче представлений детальний огляд технологій та інструментів, що були обрані для розробки вказаного веб-застосунку.

2.3.1 Мова програмування

Це критично важливе рішення, яке впливає на багато аспектів проєкту, від розробки до розгортання та подальшого супроводу. Для мого проєкту,

основними критеріями вибору стали ефективність розробки, підтримка сучасних веб-технологій, гнучкість та спроможність до масштабування. З огляду на ці критерії, вибір було зроблено на користь JavaScript.

JavaScript є універсальною мовою, яка підтримується всіма сучасними веб-браузерами, дозволяючи розробникам створювати як клієнтську, так і серверну частину веб-додатків. Це забезпечує високу ефективність розробки та спрощує процес інтеграції та тестування.

Також було прийнято рішення використовувати TypeScript для створення цього застосунку. Це додає додаткову цінність, оскільки TypeScript, надмножина JavaScript, вносить строгу типізацію, що значно покращує якість коду, полегшує його розуміння та спрощує виявлення та виправлення помилок на ранніх стадіях розробки.

Вибір JavaScript як мови програмування для розробки суттєво спрощує процес створення веб-додатків, оскільки він дозволяє використовувати єдину мову як для клієнтської, так і для серверної частин. Це не тільки зменшує кількість необхідних технологій, які потрібно вивчати та використовувати в проєкті, але й сприяє підвищенню продуктивності розробки завдяки можливості перевикористання коду і компонентів між різними частинами додатку. Крім того, наявність широкої екосистеми бібліотек та інструментів для JavaScript гарантує гнучкість у виборі рішень для будь-яких потреб проєкту.

2.3.2 Серверна частина, фреймворк Nest.js

Цей вибір для розробки серверної частини веб-додатку має низку переконливих причин, які випливають з особливостей його архітектури та функціональності.

Nest.js є фреймворком для створення ефективних, надійних і масштабованих серверних додатків на платформі Node.js.

Цей фреймворк пропонує модульну структуру, яка дозволяє легко організувати код в межах проєкту, сприяє його перевикористанню та забезпечує легкість управління залежностями. Він розроблений з використанням TypeScript,

що надає всі переваги строгої типізації.

Також використання Nest.js робить можливим застосування декількох патернів проєктування які спрощують та пришвидшують розробку програмних продуктів:

1. Dependency Injection (DI): Dependency Injection є центральним патерном у Nest.js, який дозволяє зменшити залежність між компонентами програми. Фреймворк автоматично управляє життєвим циклом об'єктів, що ін'єктуються, забезпечуючи ефективне використання ресурсів;
2. Модульність: модульність у Nest.js дозволяє організовувати код у логічні блоки, кожен з яких відповідає за певний аспект функціональності додатку. Модулі можуть містити контролери, провайдери та інші модулі, що дозволяє створювати зрозумілу та легко масштабовану структуру проєкту. Такий підхід спрощує підтримку та модифікацію проєкту в довгостроковій перспективі.
3. Контролери: контролери в Nest.js відповідають за обробку вхідних запитів та повернення відповідей клієнту. Цей патерн сприяє розділенню логіки прийому запитів від логіки бізнес-процесів, що забезпечує чітку організацію коду та полегшує його тестування та розширення.
4. Сервіси: сервіси (часто називаються також провайдерами) у Nest.js використовуються для інкапсуляції бізнес-логіки, доступу до бази даних або інтеграції з зовнішніми сервісами. Ці компоненти є відокремленими від контролерів, що дозволяє забезпечити легкість перевикористання.
5. Middleware: Middleware у Nest.js дозволяє виконувати код перед або після маршрутизованих запитів, надаючи можливість модифікації запитів або відповідей, ведення логів, валідації даних тощо. Цей патерн забезпечує гнучке управління потоком обробки запитів.

Вибір Nest.js для серверної частини цього застосунку не тільки відображає сучасні тенденції в розробці програмного забезпечення, але й надає унікальну комбінацію продуктивності, масштабованості та легкості у використанні, що

важко знайти в інших фреймворках. Завдяки глибокій інтеграції з TypeScript, Nest.js забезпечує потужні інструменти для створення зрозумілого та ефективного коду, спрощуючи процеси розробки та тестування. Його архітектурні принципи, засновані на використанні відомих патернів проєктування, роблять Nest.js ідеальною основою для розробки комплексних і водночас гнучких веб-застосунків. Це, у свою чергу, дозволяє забезпечити високий рівень абстракції, необхідної для створення різноманітних функціональних можливостей мого застосунку для спілкування та підтримки, роблячи взаємодію користувачів ефективнішою та приємнішою.

2.3.3 Серверна частина, система управління базою даних MongoDB

Обравши MongoDB як систему управління базою даних для мого комунікаційного застосунку, я скористалася низкою її ключових переваг, які ідеально відповідали потребам мого проєкту. Особливо цінною виявилася гнучкість MongoDB у моделюванні даних, яка дозволила мені легко зберігати різноманітну інформацію - від профілів користувачів до повідомлень- без потреби в постійному перегляді та зміні схеми бази даних.

Також варто взяти до уваги здатність MongoDB до горизонтального масштабування через шардування, що стало критично важливим для підтримки зростаючої кількості користувачів і даних. Це забезпечило мій застосунок високою доступністю та продуктивністю, навіть під час пікових навантажень.

Крім того, активна спільнота MongoDB та широкий спектр доступних ресурсів і інструментів стали великою підтримкою протягом усього процесу розробки. Вони забезпечили мені важливі знання та інструменти для вирішення будь-яких викликів, з якими я міг зіткнутися.

У підсумку, MongoDB виявилася ідеальним вибором для мого проєкту, забезпечивши необхідну продуктивність, масштабованість та гнучкість, які дозволили моєму застосунку ефективно розвиватися.

2.3.4 Комунікація серверної та клієнтської частини додатку

У моєму проекті, я приділила особливу увагу забезпеченню безпеки та ефективності взаємодії між користувачами, вибравши HTTPS та WebSocket (WS) як основні протоколи комунікації.

Використання HTTPS є критично важливим для захисту даних користувачів під час їх передачі через Інтернет. Цей протокол використовує шифрування для забезпечення конфіденційності та цілісності даних, запобігаючи їх перехопленню або модифікації зловмисниками. Використання HTTPS допомагає підвищити довіру користувачів до мого застосунку, гарантуючи, що їхня приватна інформація, така як особисті повідомлення, залишається захищеною.

Протокол комунікації WebSocket використовується для підтримки двостороннього зв'язку в реальному часі між клієнтом та сервером, що є ідеальним для моєї платформи спілкування. WS дозволяє користувачам негайно отримувати повідомлення та оновлення статусів без необхідності постійного перезавантаження сторінки або виконання запитів до сервера для перевірки наявності нових даних.

Обираючи HTTPS та WebSocket для мого проекту, я змогла створити безпечне та ефективне середовище для користувачів, які шукають та надають підтримку один одному. Це дозволило мені забезпечити надійний та зручний засіб спілкування, відповідаючи основним потребам і очікуванням користувачів моєї платформи.

2.3.5 Клієнтська частина

Для клієнтської частини мого комунікаційного застосунку я обрала React.js, рішення, що виходило з потреби створення інтерактивного та гнучкого інтерфейсу, який міг би забезпечити користувачам плавний та інтуїтивно зрозумілий досвід користування. React.js, як бібліотека для побудови

користувацьких інтерфейсів, розроблена Facebook, виявилася ідеальним вибором з кількох причин.

Перш за все, React.js сприяє ефективній розробці завдяки компонентному підходу, який дозволяє побудувати складні інтерфейси з використанням повторно використовуваних компонентів. Це не лише спрощує процес розробки та підтримку коду, але й забезпечує консистентність інтерфейсу застосунку.

Далі, використання віртуального DOM у React.js значно покращує продуктивність застосунку, мінімізуючи кількість маніпуляцій з реальним DOM та оптимізуючи процеси оновлення інтерфейсу. Це особливо важливо для застосунків з високим рівнем інтерактивності, де швидкість відгуку та плавність оновлень є ключовими для забезпечення якісного користувацького досвіду.

Також, React.js підтримує широкий спектр екосистеми інструментів та бібліотек, що дозволяє легко інтегрувати додаткові функції, такі як маршрутизація, управління станом та оптимізацію для пошукових систем (SEO). Ця гнучкість дозволила мені адаптувати застосунок до різних потреб та вимог без зайвих зусиль.

Врешті-решт, широка спільнота розробників та доступність ресурсів для навчання та підтримки сприяють легкості використання React.js та вирішенню будь-яких виникаючих питань чи проблем, що забезпечує плавний розвиток проекту.

Обравши React.js, я отримала не лише інструмент для ефективної розробки, але й змогла забезпечити користувачам мого застосунку високоякісний та інтерактивний досвід користування, що є вирішальним для успіху будь-якої сучасної комунікаційної платформи

2.3.6 HTML

HTML є абревіатурою від "HyperText Markup Language", що перекладається як "мова розмітки гіпертексту". Вона дає змогу користувачам створювати структурований контент для веб-сторінок, включаючи розділи,

абзаци, заголовки та посилання. Хоча HTML сама по собі не є мовою програмування і не може забезпечувати динамічні функції, вона використовується для організації та форматування контенту в документах подібно до того, як це робить Microsoft Word. HTML використовує теги та атрибути для визначення структури веб-сторінки, наприклад, <p> для абзаців, <h1> до <h6> для заголовків, <img> для зображень тощо. Ці теги слугують основою для вмісту сайту або додатку, а HTML-файли як правило є базою, з якої починається вся веб-сторінка. Ця мова розмітки є досить простою для навчання, навіть для тих, хто лише починає працювати з веб-технологіями.

2.3.7 CSS

CSS, що розшифровується як "Cascading Style Sheets" або "каскадні таблиці стилів", служить для оформлення веб-сторінок. Ця мова дозволяє контролювати візуальний аспект сторінки, такий як колір тексту, стиль шрифтів, відступи між абзацами, розміри елементів та їх розташування, а також фонові зображення. CSS дозволяє вирішувати, як елементи будуть виглядати на різних пристроях та розмірах екранів, додавати різноманітні візуальні ефекти.

Разом HTML і CSS формують фундамент веб-розробки: HTML відповідає за структуру та зміст, тоді як CSS відповідає за форму та презентацію. Коли ви розпочинаєте розробку додатку, ви використовуєте HTML для створення базових елементів веб-сторінки, а потім застосовуєте CSS для їх стилізації, забезпечуючи привабливий і професійний вигляд. Обидві технології є важливими для створення реактивних, інтерактивних та доступних веб-додатків.

2.3.8 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) — це безкоштовний, легкий і потужний редактор коду, розроблений компанією Microsoft. Він підтримує багато мов програмування та має багатий набір функцій для покращення продуктивності розробників. Основні компоненти включають редактор коду з підсвічуванням синтаксису та автозаповненням (IntelliSense), бічну панель з такими вкладками, як Explorer для навігації по файлах і папках проєкту, Search для швидкого пошуку та заміни тексту, Source Control для інтеграції з системами контролю версій (наприклад, Git), Run and Debug для запуску та налагодження коду, а також Extensions для керування розширеннями. Вбудований термінал дозволяє виконувати командний рядок прямо в редакторі, підтримуючи різні оболонки, такі як bash і PowerShell. Панель стану відображає корисну інформацію про поточний файл і налаштування, включаючи гілку Git, кодування файлу та позицію курсора. Командна панель, яка викликається комбінацією клавіш, дозволяє швидко знаходити та виконувати команди.

Однією з головних особливостей VS Code є його розширюваність. Підтримка тисяч розширень дозволяє додавати нові мови, теми, налагоджувачі, інструменти та інші функції. Вбудована підтримка Git дозволяє виконувати всі основні операції прямо з редактора. Потужні інструменти для налагоджування коду включають підтримку брекпойнтів, перегляд змінних та викликів стеку. IntelliSense надає контекстно-залежні підказки для коду, включаючи визначення змінних, функцій та модулів. Підтримка фрагментів коду (snippets) забезпечує швидке вставлення часто використовуваних блоків коду, а функція живого перегляду (Live Preview) дозволяє переглядати зміни в реальному часі для веб-розробки. Підтримка хмарних середовищ розробки через сервіси, такі як GitHub Codespaces, додає додаткову гнучкість.

2.4 Опис структури системи та алгоритмів її функціонування

2.4.1 Логічна структура системи

Платформа для анонімного спілкування та надання психологічної підтримки складається з трьох основних компонентів:

1. Серверна частина: Відповідає за обробку запитів, управління даними, аутентифікацію та авторизацію користувачів;
2. Клієнтська частина: Забезпечує інтерфейс для користувачів, дозволяє їм реєструватися, входити в систему, обмінюватися повідомленнями;
3. База даних: Використовується для зберігання інформації про користувачів, повідомлення, журнали дій тощо.

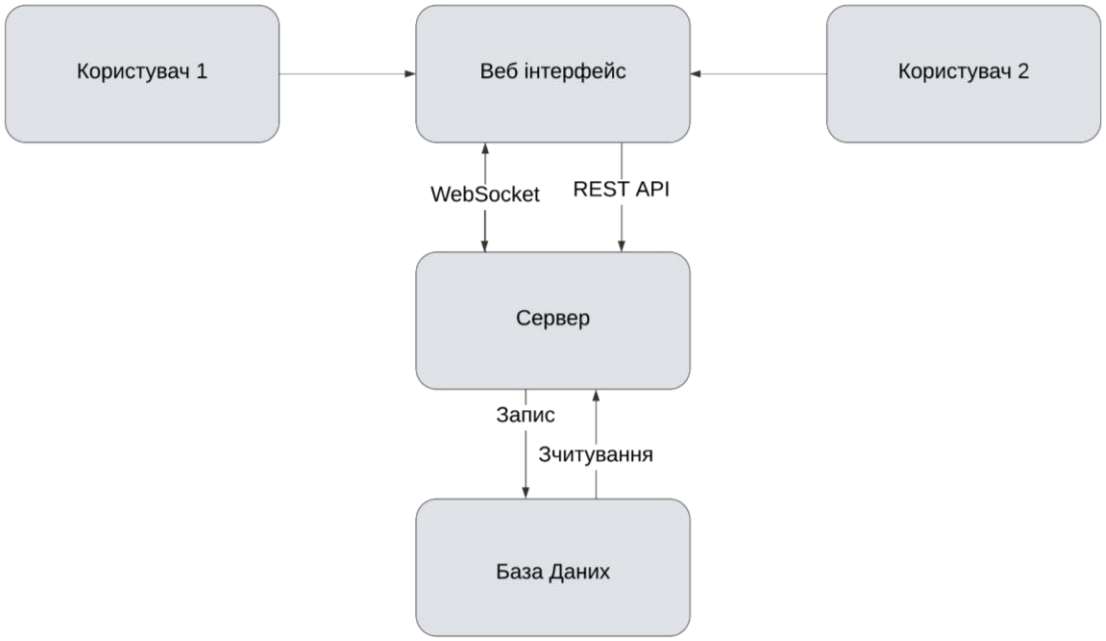

Рис 2.1 архітектура системи

Опис алгоритму функціонування:

1. Реєстрація користувача: Користувач заповнює форму реєстрації, дані передаються на сервер, де перевіряється унікальність електронної пошти та створюється новий обліковий запис;

2. Авторизація користувача: Користувач вводить свої облікові дані, сервер перевіряє їх та, у разі успішного входу, видає JWT-токен;
3. Пошук співбесідника: Користувач вибирає роль (шукає підтримку чи бажає підтримати), система автоматично підбирає співбесідника з протилежною роллю;
4. Обмін повідомленнями: Користувачі обмінюються повідомленнями в реальному часі через WebSocket;
5. Зберігання історії спілкування: Всі повідомлення зберігаються у базі даних, доступ до них можна отримати в будь-який момент.

2.4.2 Серверна Частина

Сервер виконує функціонал аутентифікації користувачів, управління профілями та безпосередньо опрацюванням повідомлень.

Головний файл проєкту відіграє роль кореневого модуля, який інтегрує та координує роботу всіх інших модулів системи. Він забезпечує інтеграцію з MongoDB через MongooseModule, який налаштовано асинхронно. Ця настройка дозволяє динамічно конфігурувати з'єднання з базою даних на основі конфігураційних параметрів з модулю CommonModule.

Модулі AuthModule, UsersModule, CommonModule, ChatsModule та StatusModule відповідають за різні функціональні аспекти додатку, включаючи аутентифікацію користувачів, управління профілями, спільні служби, обробку чатів. Ці модулі допомагають структурувати додаток та забезпечують чітке розмежування відповідальності між різними частинами системи.

AuthModule виконує забезпечує аутентифікацію та авторизацію у веб-додатку. Цей модуль охоплює всі основні аспекти безпеки, включаючи реєстрацію користувачів, вхід в систему та перевірку доступу до ресурсів. З огляду на його глобальну природу в архітектурі додатку, AuthModule доступний в усіх частинах системи.

В центрі AuthModule знаходяться контролер AuthController та сервіс AuthService, які разом управляють логікою аутентифікації та авторизації.

AuthController відповідає за прийом та обробку HTTP-запитів на реєстрацію нових та вхід існуючих користувачів у систему. Процес реєстрації включає створення нового користувача з наданими даними, в той час як процес входу забезпечує перевірку облікових даних користувача та видачу JWT для подальшої аутентифікації запитів.

AuthService відіграє центральну роль у генерації та валідації JWT, що є ключовими для забезпечення безпечного доступу до ресурсів додатку. Сервіс використовує конфігураційні параметри з AppConfig для налаштування секретного ключа, який необхідний для підпису та перевірки токенів. Цей механізм дозволяє ідентифікувати та верифікувати користувачів на основі токенів, які вони надсилають разом зі своїми запитами, тим самим мінімізуючи потребу в повторних перевірках стану аутентифікації.

AuthGuard у AuthModule діє як бар'єр для доступу до різних частин системи, вимагаючи від користувачів надати дійсний токен для виконання запитів. Цей механізм захисту використовує методи з AuthService для перевірки токенів та визначення, чи має користувач право на доступ до запитаних ресурсів. Такий підхід сприяє безпеці додатку, оскільки забезпечує, що всі захищені ресурси можуть бути доступні лише авторизованим користувачам.

UsersModule виконує роль збереження інформації про користувачів, створення та зчитування даних облікових записів. Головним компонентом цього модуля є UsersService який реалізує описаний вище функціонал. На відміну від AuthModule цей модуль не має контролера який надає доступ до сервера через протокол HTTP. Саме тому він інтегрується із іншими частинами додатку, зокрема з AuthModule, для реєстрації нових та верифікації існуючих користувачів. Отже, UsersModule використовується іншими модулями як сервісний шар, що надає методи для роботи з даними. Це дозволяє інкапсулювати логіку доступу до даних, забезпечуючи високий рівень абстракції та сприяючи більшій гнучкості в коді.

Розглянемо детальніше деякі важливі аспекти обробки даних облікових записів:

- для створення запису користувача необхідно щоб були виконані умови унікальності юзернейму (псевдоніму) та електронної пошти;

- при створенні профілю важливо переконатися в безпечності зберігання чутливих даних, таких як пароль, тому для цього застосовуються хеш-функції, які перетворюють вихідний пароль на унікальний набір символів фіксованої довжини, залежно від алгоритму хешування, такого як SHA-256 або bcrypt. Цей процес є одностороннім, що означає, що з хешу неможливо відновити оригінальний пароль. Це значно ускладнює завдання зловмисникам, які намагаються отримати доступ до паролів, оскільки навіть якщо база даних буде скомпрометована, витягнуті хеші не дають зловмиснику корисної інформації без додаткового обчислювального зусилля. Крім того, використання солі — унікального випадково згенерованого рядка, який додається до пароля перед хешуванням, — додатково збільшує безпеку, оскільки навіть однакові паролі будуть мати різні хеші, що робить неможливим використання готових таблиць відповідностей (rainbow tables) для злому паролів.

Таким чином, UsersModule служить надійним фундаментом для управління даними користувачів, відіграючи ключову роль у загальній структурі безпеки додатку. Він забезпечує не тільки зберігання інформації, але й функціональність, необхідну для підтримки безпечної взаємодії в рамках веб-додатку.

CommonModule це простий модуль який тримає інформацію необхідну для функціонування інших компонентів додатку. До цих налаштувань входить рядок посилання на базу даних, рядок випадкових символів (сіль) що необхідний для збільшення надійності хешування паролів, а також рядок випадкових символів який є секретним ключем для генерації JWT що необхідний для процесу аутентифікації.

У підсумку, CommonModule відіграє важливу роль у забезпеченні безпеки та конфігурації додатку, зосереджуючись на централізації управління ключовими налаштуваннями, які використовуються різними компонентами системи. Цей модуль допомагає уникнути дублювання коду та зберігає критичні

параметри конфіденційності в одному місці, що сприяє більшій модульності та легшому управлінню конфігурацією.

ChatsModule - найскладніший компонент цієї системи, оскільки для його коректного функціонування необхідно задіяти всі вищеописані модулі. Також в ньому присутня логіка створення чатів, надсилання та обробку повідомлень від користувачів, зберігання та зчитування історії листування.

Розглянемо детальніше усі частини цього модуля.

ChatsController надає доступ через REST API до двох кінцевих шляхів

- GET /chats: цей метод забезпечує отримання списку чатів для певного користувача. Це здійснюється шляхом виклику методу getUserChats з ChatsService, передаючи унікальний ідентифікатор користувача, отриманий з об'єкта запиту. Це дозволяє користувачам переглядати всі чати, в яких вони беруть участь;
- POST /chats/create: цей метод використовується для створення нового чату з ініціатором та типом чату, визначеними в тілі запиту. Метод createChat з ChatsService викликається для обробки цієї операції;

ChatsService містить декілька ключових методів для управління чатами:

- createChat: створює новий чат або додає користувача до існуючого чату, якщо всі необхідні умови виконані. Спочатку перевіряється наявність користувача через UsersService, а потім шукається можливість додати користувача до існуючого чату перед створенням нового. Цей метод також відповідає за встановлення ролей учасників у чаті;
- getUserChats: забирає всі чати, в яких задіяний певний користувач, використовуючи userId. Це забезпечує перегляд всіх активних чатів користувача;
- updateChatMessages: дозволяє додавати нові повідомлення до визначеного чату, що підтримує інтерактивність діалогів між учасниками;
- getChatById: зчитує конкретний чат за його ідентифікатором, що необхідно для операцій, які вимагають роботи з конкретним чатом.

ChatsGateway це шлюз який є компонентом веб-сокетів у фреймворку NestJS, він використовує бібліотеку Socket.io для забезпечення двосторонньої асинхронної комунікації між клієнтами та сервером.
Цей веб-сокет шлюз організовано з підтримкою неймспейсу “messaging”.

Неймспейс використовується для створення виділених каналів комунікації. Кожен неймспейс може мати свої окремі події та логіку обробки, що дозволяє організовувати масштабовані та ефективні системи, де різні типи спілкування мають власні "простори" для обміну повідомленнями. В даному випадку існує лише один такий простір який має строковий ключ “messaging”.

Розглянемо основні функції ChatsGateway.

Ініціалізація та логування:

- При створенні ChatsGateway ініціалізується компонент логування, що фіксує події і дії всередині шлюзу, зокрема моменти з'єднання клієнтів та отримання повідомлень від користувачів.

Обробка підключень:

- Метод handleConnection логує факт підключення нового клієнта, допомагаючи відстежувати активність користувачів у системі.

Обробка повідомлень та команд:

- handleError: реагує на помилки, що виникають у веб-сокеті, логуючи їх для подальшого аналізу та вирішення;
- handleJoinChat: Займається обробкою запитів від клієнтів на приєднання до специфічних каналів меседжингу. Клієнти використовують ID користувача як канал, що дозволяє їм приєднуватися до індивідуальних чатів;
- handleMessage: Відправляє повідомлення до конкретного чату, залучаючи сервіс ChatsService для збереження повідомлення у базі даних та відправлення його всім учасникам чату через сервер Socket.io. Кожне повідомлення розсилається в реальному часі до всіх учасників відповідного чату.

Отже, ChatsModule відіграє ключову роль у забезпеченні функціональності чату в межах мого веб-додатку, дозволяючи користувачам здійснювати інтерактивне спілкування в реальному часі. Модуль інтегрує основні компоненти, такі як ChatsService та ChatsGateway, які разом управляють логікою створення чатів, обробки повідомлень і керування учасниками чату. Він також забезпечує зберігання повідомлень учасників, використовуючи базу даних для постійного збереження історії спілкування, для доступу до цих даних використовується ChatsController.

Модуль статусу, представлений в коді через StatusController, виконує просту, але важливу роль у веб-додатку, використовуючи фреймворк NestJS. Цей модуль призначений для надання відповіді на запити про стан системи, зазвичай використовуючись для моніторингу здоров'я додатку та його доступності.

Основна функціональність:

StatusController містить один метод getStatus, який обробляє HTTP GET запити на маршрут /status. Цей метод просто повертає текстовий рядок "OK", що є загальноприйнятим способом підтвердження того, що додаток функціонує нормально і може обробляти запити. Такий підхід використовується у системах моніторингу, які регулярно перевіряють кінцеву точку, щоб забезпечити, що додаток працює та доступний для користувачів.

У підсумку, у веб-додатку, розробленому на базі фреймворку NestJS, використання різноманітних модулів, таких як AuthModule, UsersModule, CommonModule, ChatsModule, і StatusModule, дозволяє ефективно організувати код, розподіляючи відповідальність за різні аспекти системи між спеціалізованими блоками. Всі окремі компоненти формують єдину систему яка виконує роль сервера для мого додатку.

2.4.8 Клієнтська частина

Клієнтська частина платформи розроблена з використанням бібліотеки React.js, яка дозволяє створювати компонентний підхід до розробки інтерфейсу користувача. Основні компоненти клієнтської частини:

1. App Component: Головний компонент, який містить основну логіку маршрутизації та відображення інших компонентів;
2. Authentication Components:
    - Login Component: Форма для введення електронної пошти та паролю користувача для входу в систему;
    - Register Component: Форма для створення нового облікового запису, включаючи поля для введення псевдоніму, електронної пошти та паролю.
3. Main Page Components:
    - Header Component: Відображає заголовок платформи, навігаційні посилання та кнопку виходу з системи;
    - Sidebar Component: Містить список доступних співбесідників та дозволяє користувачам вибирати, з ким вони бажають спілкуватися;
    - Chat Component: Основне вікно для обміну повідомленнями між користувачами, включаючи поле для введення тексту та область для відображення історії чату.
4. Notification Components:
    - Notification Component: Відображає сповіщення про нові повідомлення або інші важливі події.

Для написання застосунку з подібним функціоналом майже неможливо обійтись без застосування патернів проєктування, що спрощують написання програмного коду. Розглянемо деякі з них що використовувались для написання веб сайту:

1. Патерн компонентів. Це основний патерн без якого неможливо навіть уявити написання програмного коду з використанням фреймворку React.js. Він дозволяє створювати каркас із HTML елементів та універсальною логікою що обробляє вхідні дані та на їх основі змінює відображення

інтерфейсу.

Чудовим прикладом є компонент повідомлення. В залежності від того хто надіслав повідомлення визначається його позиція (зліва або справа). А також відображається безпосередньо текст повідомлення.

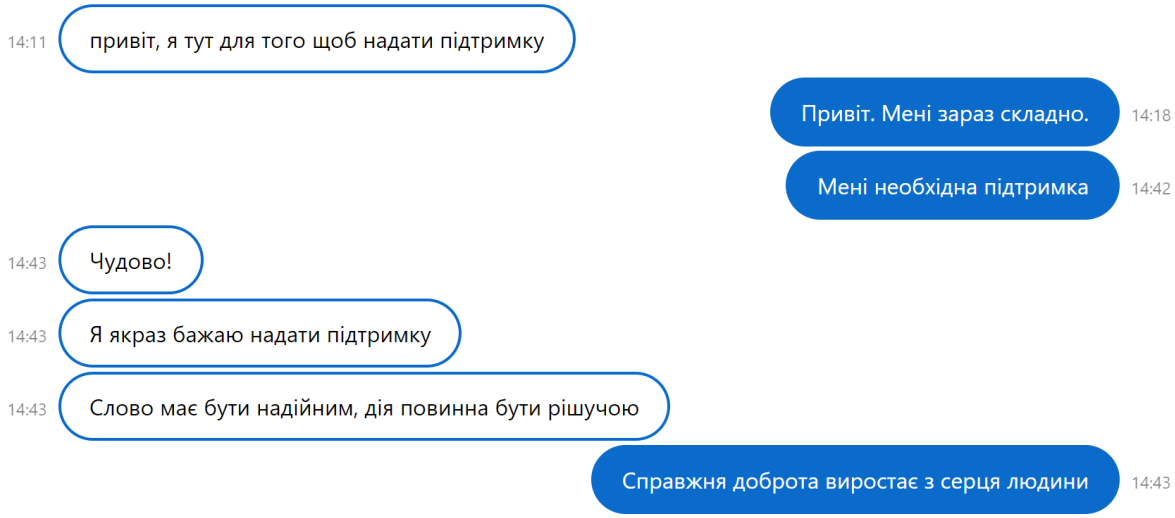

Рис. 2.1 Компонент повідомлення

2. Патерн композиції. Композиція дозволяє об'єднувати кілька компонентів для створення більш складних інтерфейсів. Це досягається шляхом вкладення компонентів один в одного, що дозволяє будувати гнучкі та динамічні структури.

   Прикладом використання композиції в моєму застосунку можна навести контейнер що містить повідомлення та форму відправки повідомлень. Тут використовуються декілька різних компонентів об’єднані в один більший який має семантично цілісну структуру.

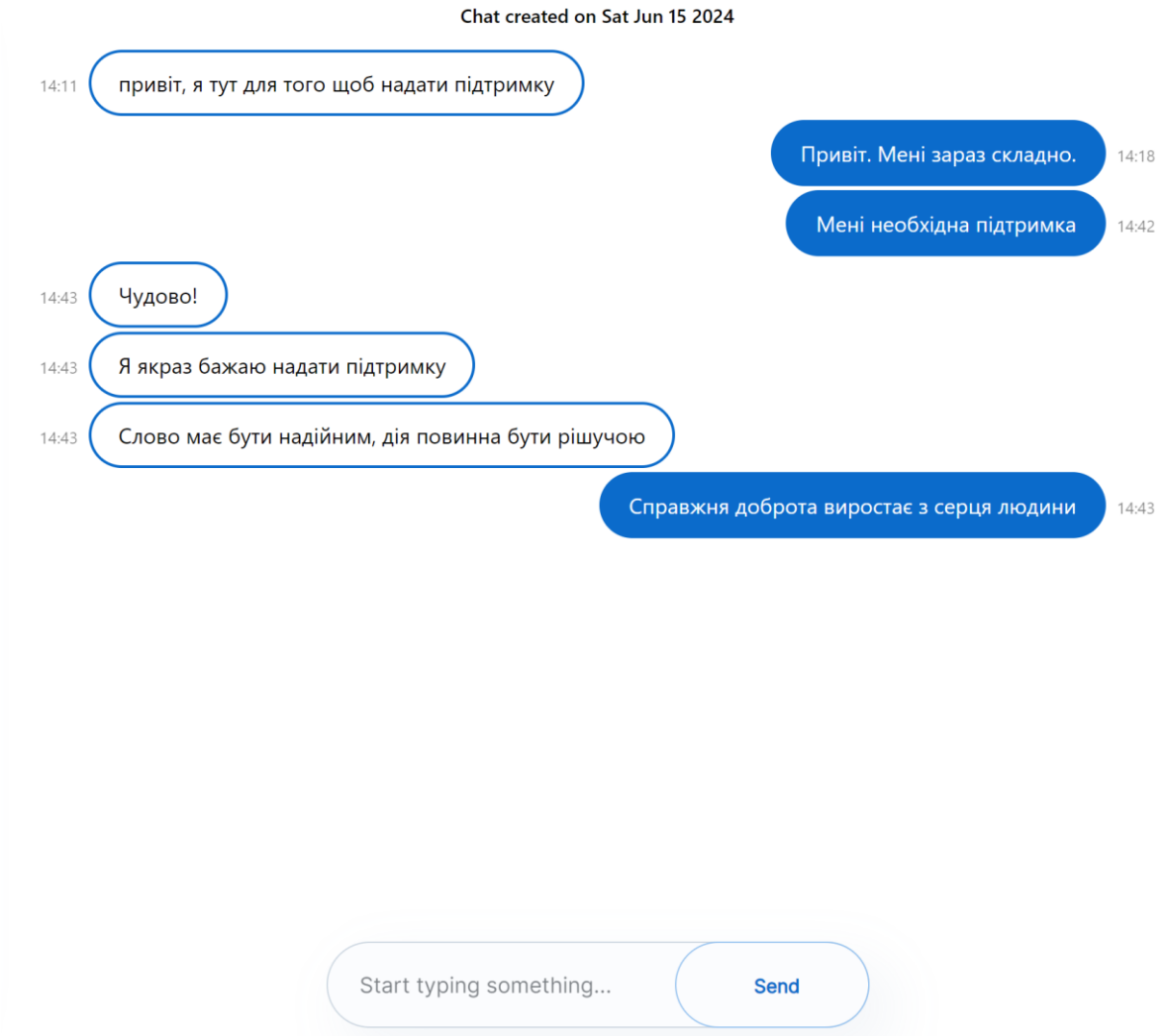

Рис. 2.2 Композиція компонентів

3. Контейнерний патерн. Компоненти поділяються на два типи: контейнерні та презентаційні. Контейнерні компоненти займаються бізнес-логікою та роботою з даними, тоді як презентаційні компоненти відповідають за графічне відображення.
З цього можна зрозуміти що контейнерні компоненти неможливо побачити в інтерфейсі, хоча вони і є невід'ємною складовою застосунку.

Для прикладу можна навести компонент входу в застосунок. Він може називатися контейнерним бо сам не має ніякого відображення. Але основною відповідальністю цього компоненту є забезпечення обробки введених користувачем даних облікового запису, відправка запиту на

сервер та обробку відповіді з подальшим перенаправленням на інші сторінки застосунку. Проте варто зазначити що в ньому також використовується патерн композиції, і хоч сам компонент входу не має графічного відображення - він складається з інших компонентів які видно користувачу (форма введення даних та інші).

4. Патерн контексту. Контекст використовується для передачі даних через дерево компонентів без необхідності передавати їх на кожному рівні. Це зручно для управління глобальним станом що рідко змінюється. Загалом цей патерн схожий на попередній тим що в контекстних компонентів немає графічного відображення.

   Для прикладу можна навести контекст авторизації що відповідає за збереження токену від серверу, обмеження доступу до функціоналу додатку неавторизованих користувачів.

5. Патерн сервіс. Це важлива архітектурна практика в розробці програмного забезпечення, зокрема в React-додатках. Цей патерн передбачає використання окремих класів або модулів для інкапсуляції логіки, пов'язаної з доступом до зовнішніх ресурсів, таких як API, бази даних або інші сервіси.

   До прикладу в моєму застосунку є сервіс чатів який відповідає за надсилання та отримання повідомлень, запити про історію листування та список чатів користувача.

2.4.9 Структура бази даних

Майже кожен застосунок використовує ту чи іншу базу даних. В моєму випадку це документоорієнтована MongoDB. В описі засобів розробки серверної частини вже було наведено обґрунтування вибору саме цієї БД. Проте також важливим є структура в якій зберігаються ці дані. Оскільки MongoDB працює з документами у форматі BSON (Binary JSON), структура даних може бути досить

гнучкою та динамічною. Документи у MongoDB організовані в колекції, які є аналогом таблиць у реляційних базах даних. Кожен документ може мати різні поля, що дозволяє зберігати об'єкти з різними структурами в одній колекції. Для мого застосунку основними колекціями є користувачі та чати.

Розглянемо структуру кожного з типів документу.

Документ користувача:

- _id: поле унікального ідентифікатора користувача;
- email: електронна пошта;
- password: поле для збереження захешованого паролю;
- username: псевдонім;
- __v: версія документу (скільки разів оновлювався).

Документ чату:

- _id: унікальний ідентифікатор чату;
- initaiatorId: ідентифікатор користувача який ініціював створення чату;
- participants: масив що містить вкладені дані учасників чату:
  1. userId: ідентифікатор користувача;
  2. username: псевдонім користувача;
  3. role: роль в якій користувач бере участь при спілкуванні.
- messages: масив повідомлень в чаті:
  1. _id: унікальний ідентифікатор повідомлення;
  2. from: ідентифікатор відправника;
  3. createdAt: дата та час відправки повідомлення.
- createdAt: дата та час створення чату;
- updatedAt: дата та час останнього оновлення;
- __v: версія документу.

2.4.10 Хмарна інфраструктура

Хмарна інфраструктура мого додатку складається з кількох основних компонентів, які працюють разом для забезпечення стабільної та ефективної роботи застосунку. Для хостингу сторінки використовується Firebase, що

дозволяє швидко та безпечно розміщувати статичні файли, забезпечуючи при цьому високу доступність і масштабованість. Firebase також пропонує вбудовані функції для управління вмістом та інтеграції з іншими сервісами Google, що робить його ідеальним вибором для фронтенд частини додатку.

Для хостингу сервера я використала Digital Ocean Droplet, який забезпечує надійну платформу для запуску серверної частини. Droplets є віртуальними приватними серверами, що дозволяють легко налаштовувати та масштабувати ресурси залежно від потреб застосунку. Digital Ocean надає простий у використанні інтерфейс для управління серверами, а також підтримку різних операційних систем та додатків, що робить його гнучким рішенням для хостингу бекенд компонентів.

Хостинг бази даних реалізований через Cloud MongoDB, який забезпечує надійне і масштабоване сховище даних. Використання MongoDB у хмарі дозволяє ефективно обробляти великі обсяги даних, забезпечуючи при цьому високу продуктивність і доступність. Cloud MongoDB також пропонує інструменти для моніторингу та управління базами даних, що полегшує адміністрування і забезпечує захист даних.

2.5 Обґрунтування та організація вхідних та вихідних даних програми

2.5.1 Організація вхідних даних

Вхідні дані вводяться користувачами через веб-інтерфейс у діалоговому режимі. Основні види вхідних даних включають:

1. Реєстраційні дані: Псевдонім, електронна пошта, пароль;
2. Авторизаційні дані: Електронна пошта, пароль;
3. Текст повідомлень: Текст, що вводиться у чаті.

2.5.2 Організація вихідних даних

Вихідні дані представляють собою відповіді сервера на запити користувачів:

1. Підтвердження реєстрації/авторизації: Повідомлення про успішну реєстрацію або авторизацію;
2. Список співбесідників: Інформація про знайдених співбесідників;
3. Повідомлення в чаті: Текст повідомлення, час відправлення.

2.5.3 Формат та спосіб кодування даних

1. Формат даних: JSON для передачі даних між клієнтом і сервером;
2. Кодування даних: UTF-8 для текстових даних, хешування паролів за допомогою алгоритму SHA-256.

2.6 Опис розробленої системи

2.6.1 Використані технічні засоби

Для роботи системи використовуються такі технічні засоби:

1. Сервер: Віртуальний приватний сервер (VPS) з процесором 2 ГГц, 4 ГБ оперативної пам'яті, 50 ГБ SSD;
2. Клієнтські пристрої: ПК, ноутбуки, планшети, смартфони.

2.6.2 Використані програмні засоби

1. Операційна система: Linux для серверної частини;

2. Програмні засоби: Nest.js (серверна частина), React.js (клієнтська частина), MongoDB (база даних);

2.6.3 Виклик та завантаження програми

1. Виклик програми: Запуск серверної частини здійснюється командою npm start у середовищі Node.js;
2. Завантаження програми: Клієнтська частина завантажується у веб-браузері шляхом введення URL платформи.

2.6.4. Опис інтерфейсу користувача

Клієнтська частина являє собою веб сайт який надає безпосередній доступ користувачам до функціоналу додатку. Так само як і в серверній частині програмний код застосунку розділений на окремі модулі, кожен з яких відповідальний за визначений функціонал.

Це односторінкові сайт (Single Page Application, SPA) на основі React.js що працює за принципом компонентної архітектури. React.js використовує компоненти як основні будівельні блоки, що інкапсулюють логіку, користувацький інтерфейс та стилі. Кожен компонент відповідає за певну частину інтерфейсу, що дозволяє легко управляти та повторно використовувати код.

Односторінковий рендеринг передбачає завантаження всього контенту один раз при початковому завантаженні сторінки. Після цього додаток динамічно оновлює інтерфейс без перезавантаження сторінки, що забезпечує швидкість і плавність взаємодії з сайтом.

Для створення односторінкового сайту з кількома "сторінками" використовують бібліотеки на кшталт React Router. Вони дозволяють визначати маршрути, які відповідають за рендеринг різних компонентів залежно від URL без перезавантаження сторінки.

SPA часто взаємодіють з сервером за допомогою асинхронних запитів. Це дозволяє динамічно завантажувати дані без перезавантаження сторінки, що робить додаток більш інтерактивним і зручним для користувача.

Такі веб-сайти визначають статус користувача і відповідно до нього показують відповідний контент - у моєму випадку сайт автоматично перенаправляє неавторизованих користувачів на сторінку входу.

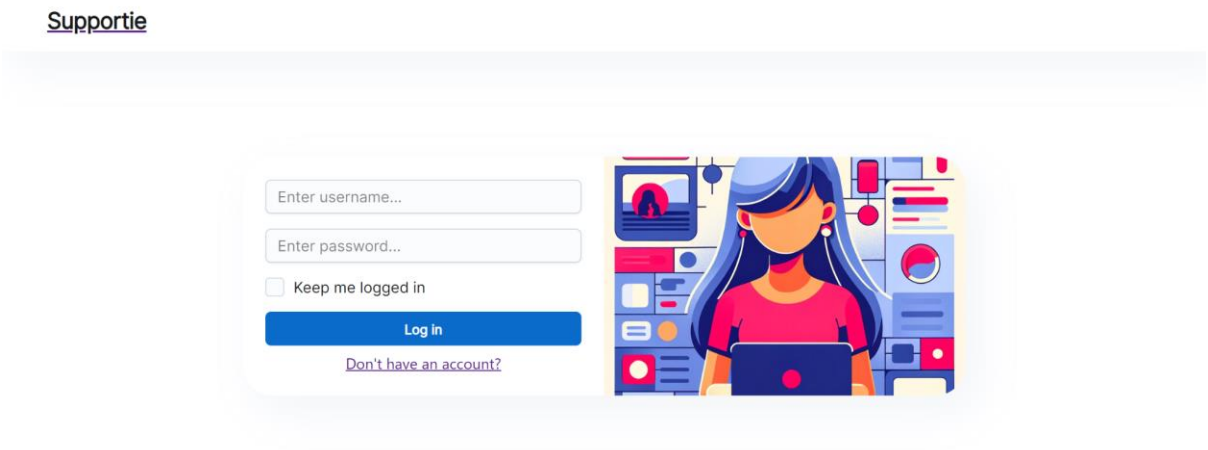

Рис. 2.3 сторінка авторизації

На зображенні ви бачите сторінку входу веб-застосунку "Supportie", яка включає наступні елементи:

- Логотип застосунку "Supportie" - розташований у верхній частині сторінки, служить як кнопка для повернення на головну сторінку.
- Форма входу:
- Поле для введення імені користувача - текстове поле з підказкою "Enter username...", куди користувачі вводять своє ім'я для входу.
- Поле для введення пароля - текстове поле з підказкою "Enter password...", призначене для введення пароля.

- Прапорець "Keep me logged in" - дозволяє користувачам залишатися в системі після закінчення сесії.
- Кнопка "Log in" - велика синя кнопка, яка запускає процес авторизації користувача.
- Посилання для реєстрації - під кнопкою входу знаходиться гіпертекстове посилання "Don't have an account?", яке веде на сторінку реєстрації для нових користувачів.

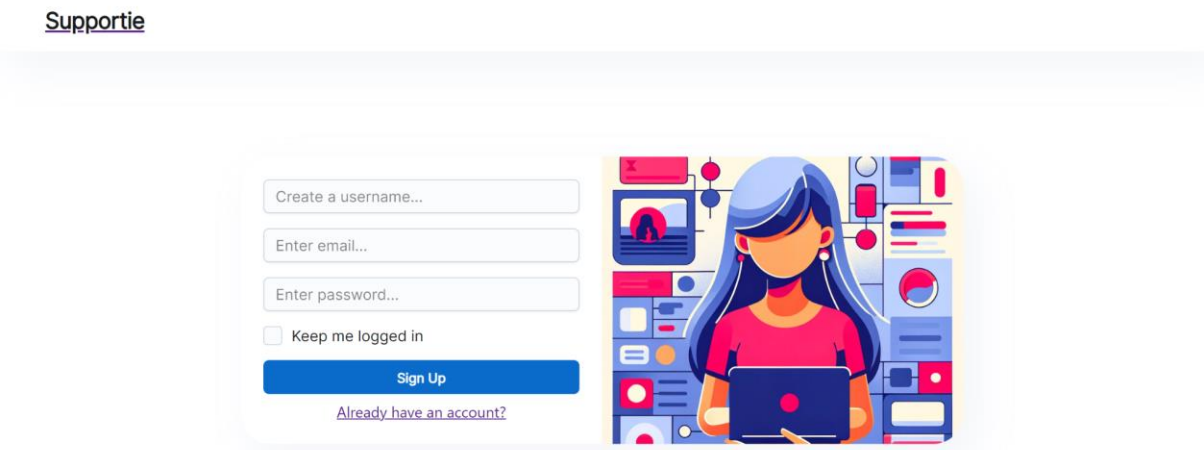

Рис. 2.4 сторінка реєстрації

На зображенні представлена сторінка реєстрації веб-застосунку "Supportie". Вона має наступні компоненти:

- Логотип застосунку - в верхній частині сторінки розташований логотип "Supportie", що служить також як домашня кнопка.
- Форма реєстрації:
- Поле для створення імені користувача - велике текстове поле з підказкою "Create a username...".
- Поле для введення електронної адреси - нижче текстове поле з підказкою "Enter email...".

- Поле для введення пароля - ще одне текстове поле, що має підказку "Enter password...".
- Прапорець "Keep me logged in" - опція, яка дозволяє залишатися у системі після закінчення сесії.
- Кнопка "Sign Up" - велика синя кнопка для подання форми реєстрації.
- Посилання для входу - під кнопкою реєстрації знаходиться гіпертекстове посилання "Already have an account?", яке веде на сторінку входу в систему.

Усі видимі елементи на веб сайті створені за допомогою бібліотеки React.js. Вони семантично розділені на так звані компоненти, які можна перевикористовувати з різним набором параметрів.

Для початку розглянемо компоненти на сторінці реєстрації, багато з них будуть повторюватись на інших сторінках веб застосунку, тому не будемо зупинятись на них двічі.

Перший елемент на сторінці - навігаційна панель яка має гіперпосилання на головну сторінку додатку, вона перевикористовується на всіх екранах додатку, якщо користувач не авторизований.

Наступним можна побачити контейнер з полями для вводу даних та ілюстративним зображенням дівчини, що використовує портативний персональний комп’ютер. Цей компонент використовується також на сторінці входу, а його вміст (поля для вводу даних користувача) змінний. Таким чином було досягнуто ефективне використання коду, за рахунок його перевикористання.

Всередині контейнеру лежить форма для введення псевдоніму, електронної пошти та паролю. Після введення всіх необхідних даних у форму реєстрації користувач має натиснути на кнопку "Sign Up," що за допомогою JavaScript надішле запит на сервер з щойно введеними даними та створить обліковий запис у випадку якщо такого ще не існує. У відповідь сервер надішле інформацію користувацького акаунту щоб підтвердити його створення. Одразу після цього користувачу буде надано доступ безпосередньо до функціоналу застосунку.

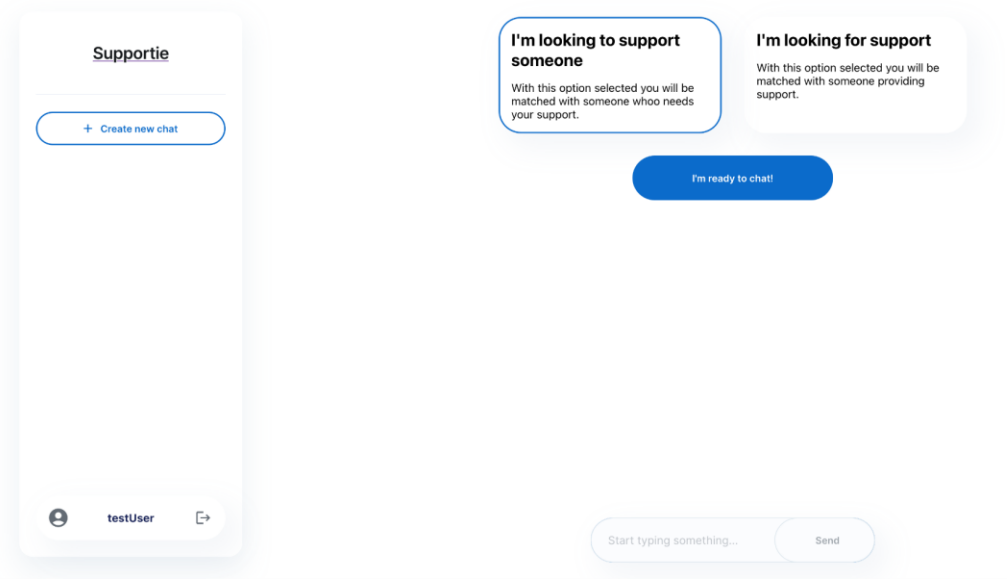

Рис. 2.5 Сторінка з основним функціоналом

Зліва знаходиться зручна навігаційна панель яка має логотип застосунку та кнопку створення нового чату. Оскільки користувач щойно зареєструвався в нього поки що немає жодних чатів в яких він бере участь, тому по центру знаходиться компонент що допомагає створити новий чат.

Це два контейнери з текстовим описом того який тип чату буде створений після натискання на кнопку “I’m ready to chat.” За замовчуванням обраний тип чату має синій край що візуально виділяє елемент. Після натискання на кнопку “I’m ready to chat” буде надіслано запит на сервер з обраною конфігурацією.

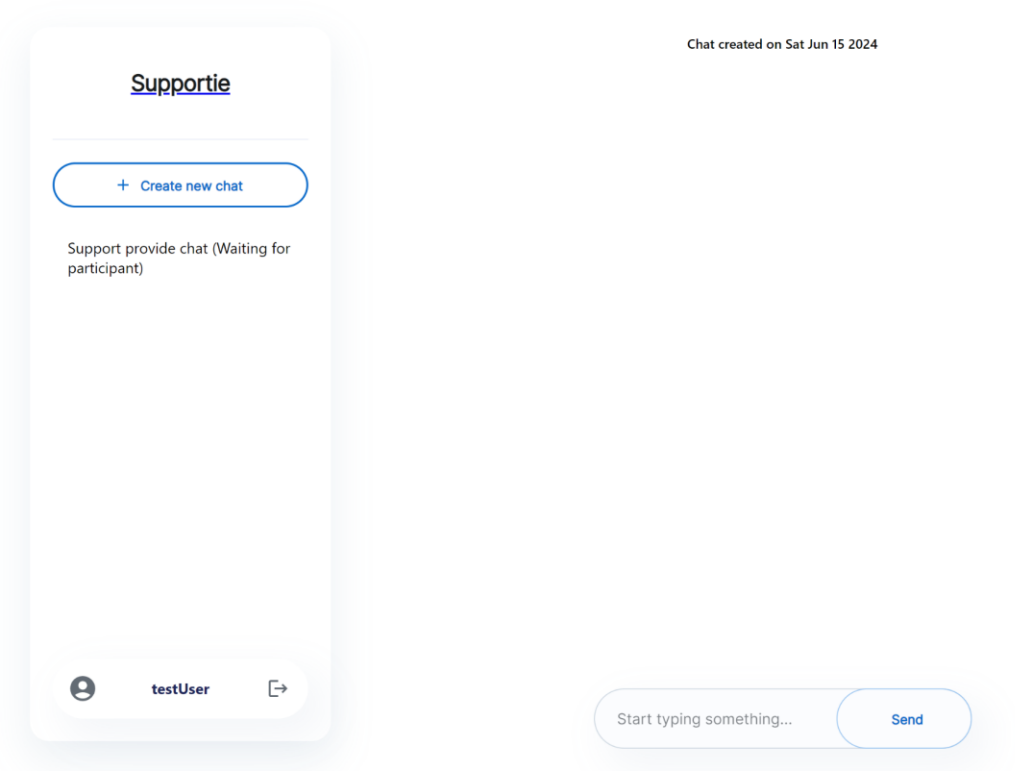

Рис. 2.6 Результат створення чату

Після отримання успішної відповіді від сервера з результатом створення чату клієнт відобразить його в списку на навігаційній панелі. Назва матиме мету чату (надання або отримання підтримки), а також буде відображати статус очікування співбесідника аж до моменту поки система не знайде співрозмовника та не доєднає його до чату.

В центрі сторінки згори відображено початок чату надписом з датою його створення. Вже на цьому етапі користувач може почати писати повідомлення, оскільки вони будуть збережені в історії листування та новий учасник чату буде мати до них доступ.

Для того щоб відправити повідомлення потрібно ввести його зміст в поле для вводу тексту знизу і натиснути на кнопку “Send.” Після цієї дії буде надіслано запит на сервер з вмістом повідомлення та даними відправника. Сервер обробить цю інформацію та розподілить це повідомлення серед усіх учасників чату.

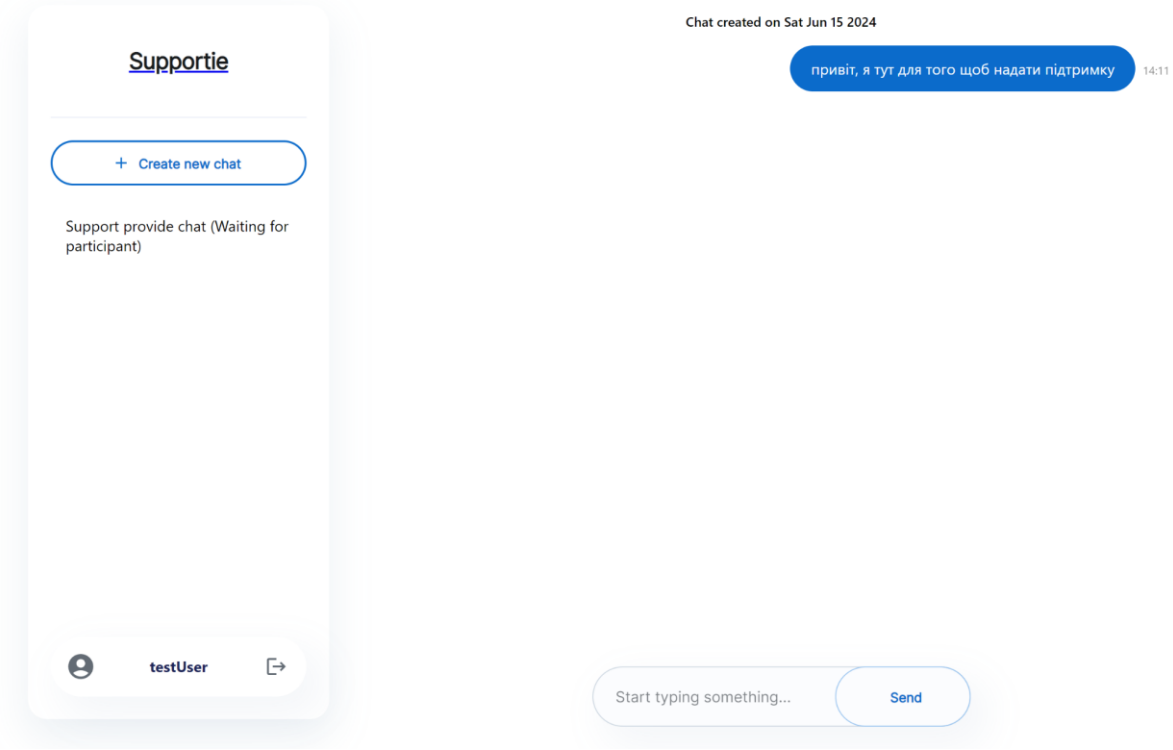

Рис. 2.7 Результат надсилання повідомлення

Через деякий час до розмови з користувачем testUser долучається інший співрозмовник який також пройшов реєстрацію, проте обрав чат для того щоб отримати підтримку.

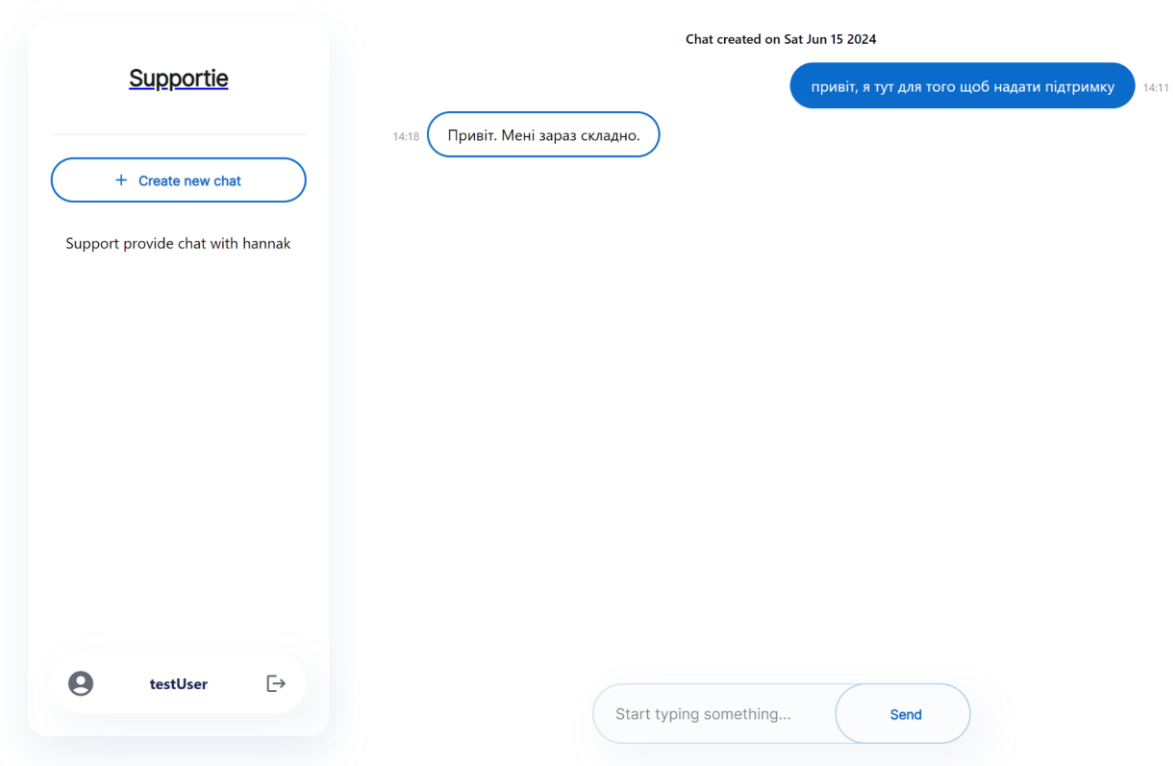

Рис. 2.8 Інший користувач приєднався до розмови

Після приєднання користувача до чату змінюється його назва на панелі навігації. Замість статусу очікування співрозмовника там описано мету чату та псевдонім користувача з яким ведеться листування. Також інший користувач надсилає повідомлення які відображаються зліва в контейнері
чату та мають інший фон.

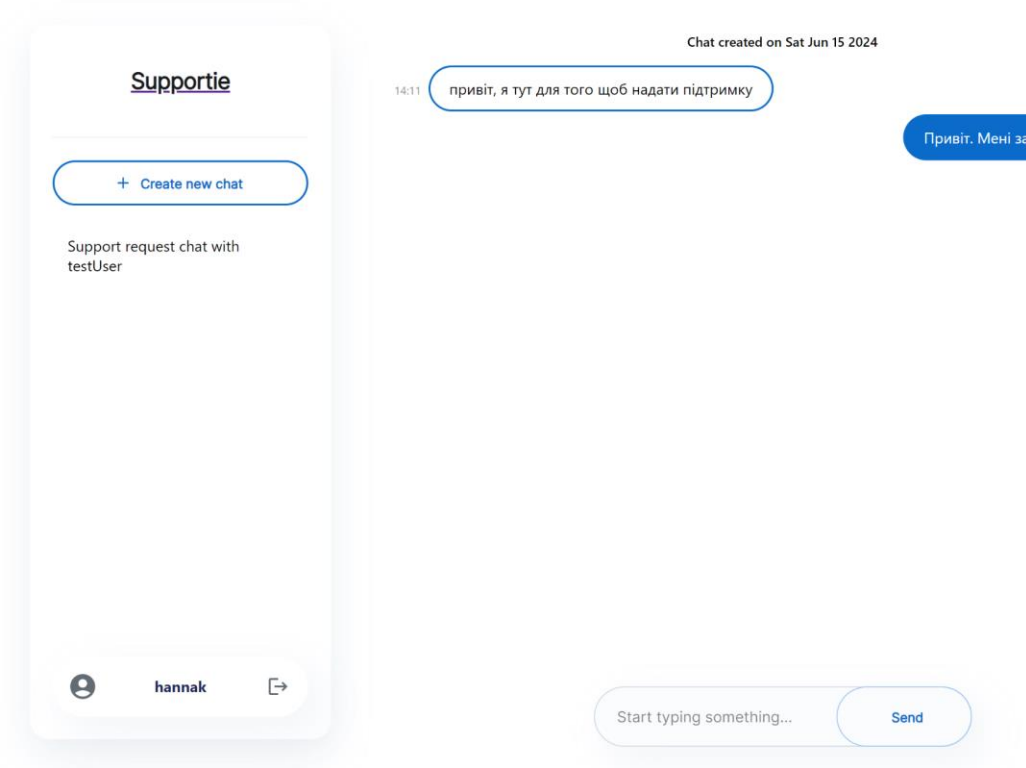

Рис. 2.9 Інтерфейс другого користувача

Після тривалого спілкування в чаті накопичуються повідомлення які не поміщаються в звичайний контейнер чату. Тому після з’являється вертикальний скрол який допомагає відображати лише останні повідомлення та за необхідності прогортати до попередніх повідомлень. Загалом інтерфейс чату дуже загальний та інтуїтивно зрозумілий будь якому користувачу що користувався подібним функціоналом деінде.

Також весь сайт адаптується під різні розміри екрану що робить його зручним для використання на будь яких пристроях.

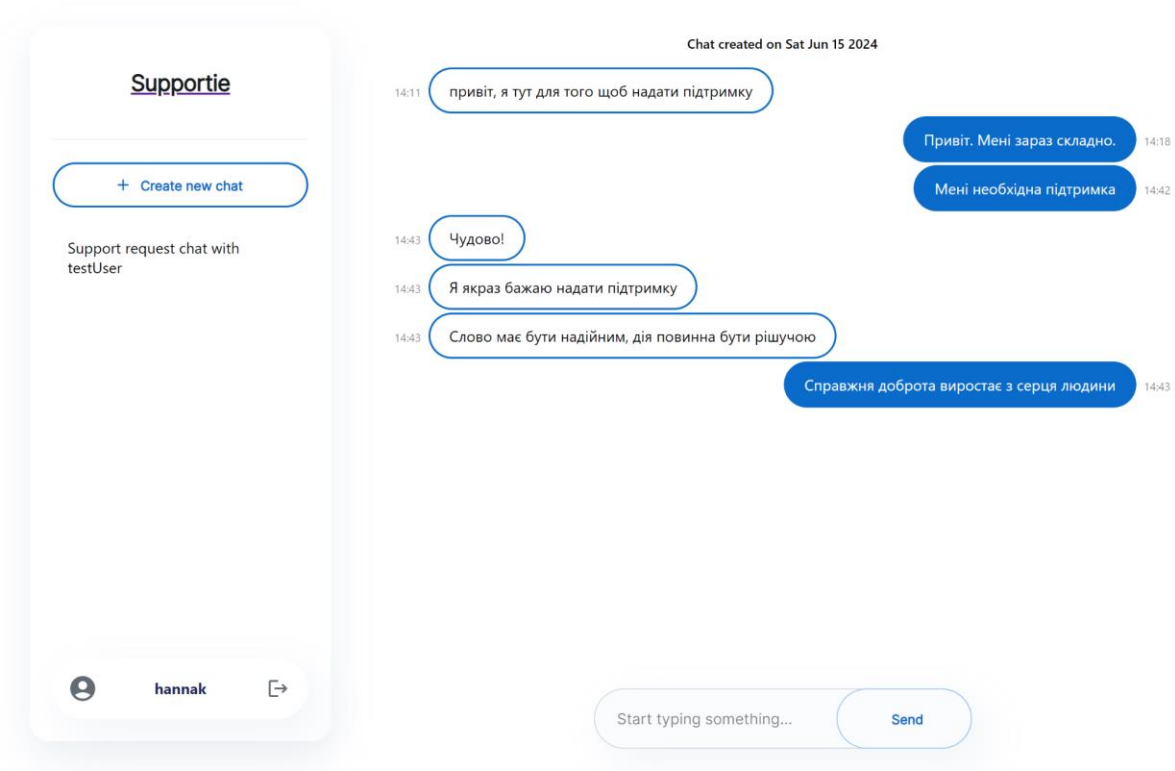

Рис. 2.10 Більше повідомлень

Також варто окремо виділити нижній елемент на навігаційній панелі. На ньому розміщено іконку користувача та псевдонім що користувач обрав під час створення облікового запису. Справа іконка при натисканні на яку користувач зможе вийти з авторизованого стану. Після цього додаток автоматично перенаправить на сторінку з логіном.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Визначення трудомісткості та вартості розробки програмного продукту

Задані дані:

1. передбачуване число операторів (підпрограм) – 1500;
2. коефіцієнт складності програми – 1,5;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста, грн/год – 130,84;[1]
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,0;
7. вартість машино-години ЕОМ, грн/год – 12.

Собівартість машино-години ЕОМ, грн/год:

$$M = \frac{S_1 + A + S_2 + S_3 + S_4 + S_5}{H}, \text{грн/год}, \quad (3.1)$$

де S_1 – річні витрати на заробітну плату обслуговуючого персоналу ЕОМ, грн;

A – річна сума амортизації, грн;

S_2 – річні витрати на електроенергію, грн; [2]

---

1 https://ua.jooble.org/salary/%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%96%D1%81%D1%82#:~:text=%D0%AF%D0%BA%D1%89%D0%BE%20%D0%BF%D0%BE%D0%B4%D0%B8%D0%B2%D0%B8%D1%82%D0%B8%D1%81%D1%8F%20%D0%BD%D0%B0%20%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D1%83%20%D0%B7%D0%B0%D1%80%D0%BE%D0%B1%D1%96%D1%82%D0%BD%D0%BE%D1%97,130%2C84%20%E2%82%B4%20%D0%BD%D0%B0%20%D0%B3%D0%BE%D0%B4%D0%B8%D0%BD%D1%83.

2 https://index.minfin.com.ua/tariff/electric/

S_3 – річні витрати на ремонтування та обслуговування обладнання, грн;

S_4 – річні витрати на матеріали, грн;

S_5 – річні накладні розходи, грн;

H – дійсний годовий фонд часу роботи, годин.

$$M = \frac{0 + 1500 + 3840 + 3000 + 1000 + 1000}{800} = 12{,}925 \frac{\text{грн}}{\text{год}}.$$

Трудомісткість розробки ПЗ можна розраховується за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.2)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p), \quad (3.3)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт корекції програми в ході її розробки.

$$Q = 1500 * 1{,}5 * (1 + 0{,}2) = 2700.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{(75..85) * k}, людино - годин. \quad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{2700 * 1}{85 * 1} = 31{,}76, людино - годин.$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) * k}, людино - годин, \quad (3.5)$$

$$t_a = \frac{2700}{25 * 1} = 108, людино - годин.$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20 \ldots 25) * k}, людино - годин, \quad (3.6)$$

$$t_n = \frac{2700}{25 * 1} = 108, людино - годин,$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) * k}, людино - годин, \quad (3.7)$$

$$t_{отл} = \frac{2700}{5 * 1} = 540, людино - годин;$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 * t_{отл}, \text{ людино-годин,} \quad (3.8)$$

$$t_{отл}^{k} = 1{,}5 * 540 = 810, \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.9)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{15..20 * k}, людино - годин, \quad (3.10)$$

$$t_{др} = \frac{2700}{20 * 1} = 135, людино - годин.$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 * t_{др}, \text{ людино-годин,} \quad (3.11)$$

$$t_{до} = 0{,}75 * 135 = 101{,}25 \text{ ,людино-годин,}$$

$$t_д = 135 + 101{,}25 = 236{,}25 \text{ ,людино-годин.}$$

Тепер розрахуємо трудомісткість ПЗ:

$$t = t_о + t_и + t_а + t_п + t_{отл} + t_д, \text{ людино-годин,}$$

$$t = 50 + 31{,}76 + 108 + 108 + 540 + 101{,}25 = 1280, \quad 61,$$

людино-годин.

3.2 Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{по} = З_{зп} + З_{мв}, \text{ грн.} \quad 3.12$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} = t * С_{пр}, \text{ грн,} \quad 3.13$$

де t - загальна трудомісткість, людино-годин;
$С_{пр}$ - середня годинна заробітна плата програміста, грн/година.

$$З_{зп} = 1280{,}61 * 130{,}84 = 167555{,}012, \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми:

$$З_{мв} = t_{отл} * С_{мч}, \text{ грн,} \quad 3.14$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год,
$С_{мч}$ - вартість машино-години ЕОМ, $12{,}925 \frac{\text{грн}}{\text{год}}$,

$$З_{мв} = 1280{,}61 * 12{,}925 = 16551{,}9, \text{ грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення ПЗ:

$$К_{по} = 167555{,}012 + 16551{,}9 = 184106{,}9, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, міс., \quad 3.15$$

де B_k - число виконавців (приймається 1),
F_p - місячний фонд робочого часу (40 годин на тиждень F_p = 176 годин).

$$T = \frac{1280{,}61}{1 * 176} = 7{,}27, міс.$$

Висновок: в даному розділі було визначено трудомісткість розробленої інформаційної системи на основі системи моделей з різною точністю оцінки (1280,61 людино-годин), проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення. Для

розробки програмного забезпечення у вигляді бази даних необхідно виділити 7,27 місяців та одноразові капітальні витрати в розмірі *184106,9* гривень.

ВИСНОВКИ

Ця робота описує процес розробки та впровадження платформи для анонімного спілкування і надання психологічної підтримки. В умовах сучасних викликів, зокрема в умовах війни, створення такого застосунку є важливим кроком для забезпечення психологічного благополуччя населення. Розробка включала ретельний аналіз предметної області, вибір відповідних технологій та інструментів, а також реалізацію різних модулів і компонентів для забезпечення функціональності додатку.

Серверна частина, побудована на базі фреймворку Nest.js, забезпечує надійне та масштабоване середовище для обробки запитів користувачів, а також інтеграцію з базою даних MongoDB. Це дозволяє зберігати та обробляти великі обсяги даних, забезпечуючи високу продуктивність та надійність. Використання різних патернів проектування в серверній частині сприяє чіткій організації коду та полегшує його підтримку і розширення.

Клієнтська частина, розроблена з використанням React.js, забезпечує інтерактивний та зручний інтерфейс для користувачів. Впровадження компонентної архітектури та використання патернів проектування дозволяє створювати гнучкі та повторно використовувані компоненти, що сприяє підвищенню ефективності розробки та підтримки коду.

Окрема увага була приділена вибору хмарної інфраструктури, яка включає Firebase для хостингу сторінки, Digital Ocean Droplet для хостингу сервера та Cloud MongoDB для хостингу бази даних. Це рішення забезпечує високу доступність, масштабованість та безпеку даних, що є критичними для успішної роботи платформи.

Загалом, розробка цього застосунку демонструє ефективне поєднання сучасних технологій і патернів проектування для створення масштабованої, надійної та зручної платформи для надання психологічної підтримки. Використання хмарної інфраструктури та передових інструментів дозволяє забезпечити високий рівень продуктивності та доступності сервісу для користувачів, які потребують допомоги в ці складні часи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бази даних та інформаційні системи: навчальний посібник / Н. О. Харів. – Рівне : НУВГП, 2018. – 127 с.
2. Анісімов А.В., Кулябко П.П. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. - Київ. – 2017. – 110 с.
3. Гайна Г. А. Основи проектування баз даних: Навчальний посібник. – К: КНУБА, 2005. – 204 с.
4. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп’ютерні системи» / О.Г. Вагонова, О.Б. Нікітіна, Н.Н. Романюк; М-во освіти і науки України, ДВНЗ «Нац. гірн. ун- т». – Д.: НГУ, 2013. – 11 с.
5. Методичні рекомендації до виконання кваліфікаційних робіт здобувачів першого рівня вищої освіти спеціальності 122 Комп'ютерні науки / В.В. Спірінцев, П.О. Іщук, О.С. Шевцова; Д : НТУ «Дніпровська політехніка», 2021. – 59 с.
6. Бази даних. Мови запитів, управління транзакціями, розподілена обробка даних: навчальний посібник / А.М. Пєтух, О.В. Романюк, О.Н. Романюк – Вінниця : ВНТУ, 2016 – 97 с.
7. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. – Електронне видання, 2018. – 118 с.
8. В. В. Булатецький, Л. В. Булатецька. Реляційна алгебра. Реляційне числення [Електронний ресурс]. ВНУ ім. Лесі Українки, 2020. – 36 с.
9. Трофименко О. Г. Організація баз даних : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. – Одеса : Фенікс, 2019. – 246 с.
10. Крещенко Л.Ф. Мова SQL: Навчально-методичний посібник. – Полтава: РВВ ПУСКУ, 2009. – 143 с.
11. Документація Nest.js. URL: https://docs.nestjs.com

12. Документація React.js. URL: https://reactjs.org/docs/getting-started.html
13. Документація MongoDB. URL: https://docs.mongodb.com
14. Алгоритм SHA-256. URL: https://en.wikipedia.org/wiki/SHA-2
15. Протокол WebSocket. URL: https://tools.ietf.org/html/rfc6455
16. Вступ до JSON Web Token (JWT). URL: https://jwt.io/introduction
17. Документація Visual Studio Code. URL: https://code.visualstudio.com/docs
18. Документація Node.js. URL: https://nodejs.org/en/docs/
19. Документація TypeScript. URL: https://www.typescriptlang.org/docs/
20. GitHub. URL: https://github.com
21. Специфікація HTML W3C. URL: https://www.w3.org/TR/html52/
22. Специфікація CSS W3C. URL: https://www.w3.org/Style/CSS/
23. Документація Firebase. URL: https://firebase.google.com/docs
24. Документація DigitalOcean Droplets. URL: https://www.digitalocean.com/docs/droplets/
25. Патерн Dependency Injection. URL: https://martinfowler.com/articles/injection.html
26. WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
27. Кращі практики безпеки JWT. URL: https://auth0.com/docs/security/best-practices
28. Кращі практики Node.js. URL: https://github.com/goldbergyoni/nodebestpractices
29. Сторінка сайту Jobble із даними про середню заробітну плату програміста. https://ua.jooble.org/salary/%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%96%D1%81%D1%82#:~:text=%D0%AF%D0%BA%D1%89%D0%BE%20%D0%BF%D0%BE%D0%B4%D0%B8%D0%B2%D0%B8%D1%82%D0%B8%D1%81%D1%8F%20%D0%BD%D0%B0%20%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D1%83%20%D0%B7%D0%B0%D1%80%D0%BE%D0%B1%D1%96%D1%82%D0%BD%D0%BE%D1%97,130%2C84%20%E2%82%B4%

20%D0%BD%D0%B0%20%D0%B3%D0%BE%D0%B4%D0%B8%D0%BD%D1%83.

30. Дані про тарифи на електроенергію Minfin. https://index.minfin.com.ua/tariff/electric/

ДОДАТОК А

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК Б

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Кузнєцова.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Кузнєцова.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
source.zip	Архів. Містить код програми
Презентація	
Презентація_Кузнєцова.pptx	Презентація кваліфікаційної роботи