

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Система адміністрування застосунку психологічної підтримки»

Виконав студент групи ПЗк-21-1
КАЛІНОВСЬКИЙ Денис Юрійович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ЧИЖМОТРИЯ Олексій Володимирович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КАЛІНОВСЬКИЙ Денис Юрійович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Система адміністрування застосунку психологічної підтримки»**,

затверджена Наказом закладу вищої освіти від «23» лютого 2024 р. №74с

Вихідні дані для роботи: метою є розробка системи адміністрування застосунку психологічної підтримки. Об'єктом дослідження є процес листування терапевтів з пацієнтами у реальному часі та адміністрування наповнення застосунку, яке пов'язане з поліпшенням психологічного стану користувача. Предметом дослідження є підходи та інструменти для забезпечення листування в реальному часі та адміністрування наповнення застосунку психологічної підтримки.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р.С.	27.02.2024р.	27.02.2024р.
2	Савіцький Р.С.	27.02.2024р.	27.02.2024р.
3	Савіцький Р.С.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота складається з програмної реалізації системи адміністрування застосунку психологічної підтримки, а також пояснювальної записки. Пояснювальна записка до дипломної роботи містить 133 сторінок, 47 ілюстрацій, 21 таблицю, 2 додатки та 56 літературних джерел.

Метою роботи є дослідження та розробка системи адміністрування, яка надає користувачам набір інструментів, що оптимізують, пришвидшують та спрощують адміністрування застосунку психологічної підтримки.

Основним інструментарієм при розробці є мови програмування JavaScript та Python, фреймворки Next.js та FastAPI, нативні реалізації протоколу Websockets кожної з мов програмування для забезпечення зв'язку між застосунками. Для реалізації єдиного, зрозумілого та ергономічного інтерфейсу використано бібліотеки TailwindCSS, Radix UI, Shadcn UI та Lucide.

Для оптимального збереження даних з великою кількістю зв'язків обрано базу даних PostgreSQL, а для безпечного доступу до даних та декларативного визначення структури даних обрано бібліотеки Prisma та SQLAlchemy для мов програмування JavaScript та Python відповідно.

В роботі описано процес аналізу технічного завдання та вимог системи, оглянуто існуюче програмне забезпечення за тематикою проєкту. Також описано процес розробки бази даних та реалізації системи адміністрування. Далі зазначено порядок встановлення та налаштування системи, опис інтерфейсу застосунку та приклади тестування.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, ІНСТРУМЕНТИ АДМІНІСТРУВАННЯ, ЧАТ, ПСИХОЛОГІЧНА ПІДТРИМКА, FASTAPI, NEXTJS, WEBSOCKETS.

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	«Система адміністрування застосунку психологічної підтримки» Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Каліновський Д.					3	133
Керівник		Савіцький Р.С.						
Рецензент		Чижмотря О.В..				Житомирська політехніка, група ІПЗк-21-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The final qualification work consists of a software implementation of a administration system of psychological help application, as well as an explanatory note. The explanatory note to the thesis contains 133 pages, 47 illustrations, 21 table, 2 appendices, and 56 references.

The purpose of the work is to research and develop an administration system that provides users with a certain set of tools with which they can optimize, accelerate, and ease administration of a psychological help application.

The main tools used in the development are JavaScript and Python programming languages, Next.js and FastAPI frameworks, and native implementations of the Websockets protocol of each programming language to ensure communication between applications. The TailwindCSS, Radix UI, Shadcn UI, and Lucide libraries were used to implement a unified, clear, and ergonomic interface.

For optimal data storage with a large number of relationships, the PostgreSQL database was chosen, and for secure data access and declarative data structure definition, the Prisma and SQLAlchemy libraries for JavaScript and Python were chosen, respectively.

The paper describes the process of analyzing the terms of reference and requirements of the system, reviews existing software on the subject of the project. The process of developing the database and implementing the administration system is also described. Further, the procedure for installing and configuring the system, description of the application interface, and examples of testing are given.

KEYWORDS: WEB APPLICATION, ADMINISTRATION TOOLS, CHAT, PSYCHOLOGICAL SUPPORT, FASTAPI, NEXTJS, WEBSOCKETS.

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	8
1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту	12
1.3 Вибір архітектури системи адміністрування застосунку психологічної підтримки	21
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	25
Висновки до першого розділу	33
2. ПРОЄКТУВАННЯ СИСТЕМИ АДМІНІСТРУВАННЯ ЗАСТОСУНКУ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ	35
2.1 Визначення варіантів використання та структури системи	35
2.2 Розробка бази даних системи	41
2.3 Проєктування та реалізація алгоритмів роботи системи.....	57
2.4 Реалізація системи адміністрування застосунку психологічної підтримки	72
Висновки до другого розділу	81
3. ІНТЕРФЕЙС ТА ПОРЯДОК ВЗАЄМОДІЇ З СИСТЕМОЮ АДМІНІСТРУВАННЯ ЗАСТОСУНКУ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ	83
3.1 Порядок встановлення та налаштування системи.....	83
3.2 Опис інтерфейсу програмного продукту	87
3.3 Тестування системи адміністрування застосунку психологічної підтримки	103
Висновки до третього розділу	111
ВИСНОВКИ	112
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	114
ДОДАТКИ.....	119

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface – інтерфейс програмування застосунків, дозволяє різним програмам взаємодіяти між собою.

AWS – Amazon Web Services – набір хмарних сервісів від Amazon, який надає віртуальні сервери, зберігання даних тощо.

CI/CD – Continuous Integration, Continuous Delivery – неперервна інтеграція і неперервне розгортання, практики в програмній інженерії, які забезпечують автоматизацію випуску і розгортання програмного забезпечення.

CSS – Cascading Style Sheets – мова каскадних таблиць стилів, використовується для оформлення вебсторінок.

CUID – Collision-resistant Unique Identifier – колізійно-стійкий унікальний ідентифікатор, використовується для створення унікальних ключів в базах даних.

DOM – Document Object Model – об'єктна модель документа, що дозволяє програмам змінювати структуру, стиль і зміст вебсторінок.

HTML – HyperText Markup Language – мова розмітки гіпертексту, використовується для створення структури вебсторінок.

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту, основний протокол для передачі даних у Інтернеті.

NPM – Node Package Manager – менеджер пакетів для мови програмування JavaScript, який спрощує управління залежностями в проєктах.

PyPI – Python Package Index, репозиторій програмного забезпечення для мови програмування Python.

RPC – Remote Procedure Call – віддалений виклик процедур, технологія, яка дозволяє комп'ютерним програмам викликати процедури на іншому комп'ютері.

SSG – Static Site Generation – генерація статичних вебсторінок, яка передбачає створення файлів розмітки гіпертексту заздалегідь під час розгортання сайту.

SQL – Structured Query Language – мова структурованих запитів, використовується для управління даними в реляційних базах даних.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

SSR – Server-Side Rendering – генерація вебсторінок на стороні сервера, забезпечує швидку початкову завантаження і кращу індексацію пошуковими системами.

UUID – Universally Unique Identifier – універсальний унікальний ідентифікатор, використовується для гарантування унікальності ідентифікаторів у розподілених системах.

OPM – об'єктно-реляційна модель – техніка програмування, яка дозволяє використовувати об'єктно-орієнтовані концепції для маніпулювання даними, збереженими в реляційних базах даних.

СУБД – система управління базами даних – забезпечує зберігання, зміну та вибірку даних.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Психологічна допомога стала актуальною у всьому світі в 21-му столітті, а в Україні, у зв'язку з бойовими діями, ментальне здоров'я усіх українців отримало удар, від якого не всі можуть оклигати самотужки, і з часом таких людей ставатиме все більше. Для допомоги згаданим особам виникла велика кількість сервісів, що допомагають знайти психологічну підтримку. Однак більшість таких сервісів мають велику ціну та націнку, а безкоштовні сервіси не мають достатнього фінансування для розробки повноцінного застосунку з системою адміністрування і потребують як від користувача, так і від терапевта, великої кількості дій, що не є оптимальним, коли користувач може мати погане самопочуття, а терапевт витрачає свій дорогоцінний час на заповнення та перегляд непотрібної інформації раз за разом.

Під час спілкування психотерапевти також можуть витрачати велику кількість часу на пошук інформації, посилок на джерела, відповідей на розповсюджені питання, формування рукописних нотаток та на звичайне дотримання процедури. Всі описані процеси мають високий потенціал оптимізації за допомогою інформаційних технологій.

Мета даної кваліфікаційної роботи полягає в розробці системи, яка б поєднала у собі сучасні підходи до листування у реальному часі та адміністрування застосунку психологічної підтримки, що допоможе професіоналам допомогти максимальній кількості людей, а користувачам отримати цю допомогу не витративши ні дорогоцінного часу, ні грошей.

Завданням даної кваліфікаційної роботи є аналіз існуючих систем адміністрування застосунків психологічної підтримки, запровадження сучасних підходів масштабування, стабільності і безпеки та створення системи адміністрування застосунку психологічної підтримки.

Важливим є проєктування архітектури застосунку для забезпечення стійкості та надійності, а також тестування архітектури для оптимізації та встановлення доцільності прийнятих рішень.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Для забезпечення повного супроводу застосунку, частиною завдання також є розробка плану тестування застосунку, що забезпечує високу якість, налаштування автоматизованого розгортання, яке зменшує необхідність додаткового персоналу для управління технічною підтримкою продукту, та інтеграція із сторонніми сервісами, впровадження якої надало можливість допомогти максимальній кількості людей.

Об'єктом дослідження є процес листування терапевтів з пацієнтами у реальному часі та адміністрування наповнення застосунку, яке пов'язане з поліпшенням психологічного стану користувача.

Предметом дослідження є підходи та інструменти для забезпечення листування в реальному часі та адміністрування наповнення застосунку психологічної підтримки. Інструменти включають візуальну реалізацію, тобто інтерфейс користувача, та реалізацію організації як листування так і адміністрування. Особливу увагу приділено оцінці та вдосконаленні інтерфейсу і функціональності системи.

Під час реалізації системи адміністрування було розглянуто реалізацію листування в реальному часі, використання штучного інтелекту в застосунку психологічної підтримки, а також спроможність штучного інтелекту створювати унікальну інформацію, бути об'єктивним та чи можна на нього покладатися, що особливо важливо в контексті такої чутливої теми, як психологічна підтримка. Були опубліковані тези: ІНТЕГРАЦІЯ МУЛЬТИПЛАТФОРМНИХ МЕСЕНДЖЕРІВ З ВИКОРИСТАННЯМ FASTAPI ТА WEBSOCKETS ДЛЯ СТВОРЕННЯ УНІФІКОВАНОЇ КОМУНІКАЦІЙНОЇ ПЛАТФОРМИ, XIV Міжнародна науково-технічна конференція «Інформаційно-комп'ютерні технології – 2024» [1], ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В СИСТЕМАХ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ, XV Міжнародна науково-практична конференція «Комп'ютерні системи та мережні технології» [2], UNVEILING AI's ILLUSION OF KNOWLEDGE, Current Trends in Young Scientists' Research: XI All Ukrainian Scientific and Practical Conference [3].

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

1.1 Постановка задачі

Кваліфікаційна робота бакалавра на тему «Розробка системи адміністрування застосунку психологічної підтримки» має на меті створення комплексної інфраструктури управління застосунком, який надає психологічну допомогу користувачам з використанням WebSockets та мікросервісів. Основна ціль проєкту полягає у розробці ефективної та гнучкої системи адміністрування, що дозволить моніторити, керувати та оптимізувати процеси взаємодії між користувачами та психологами, а також управління контентом, пов'язаним із підтримкою ментального здоров'я. Для спрощення та пришвидшення роботи операторів чату застосунком доповнено базою знань, яка надає можливість зберігати та оновлювати релевантну інформацію щодо психологічної підтримки.

Важливі аспекти системи включають розвиток модулів для управління сесіями психологічної допомоги та автоматизацію процесів публікації і модерації експертного контенту. Ключовим елементом системи є автоматизований чат, що швидко та ефективно переспрямовує повідомлення операторів системи та користувачів підтримуваних месенджерів, дозволяючи проєктувати та розробляти систему не відволікаючись на особливості окремих месенджерів.

Актуальність теми кваліфікаційної роботи бакалавра «Розробка системи адміністрування застосунку психологічної підтримки» визначається необхідністю створення вдосконалених інструментів для ефективного управління ресурсами та процесами в межах платформи психологічної підтримки. З огляду на зростання потреби в психологічній допомозі та виклики, пов'язані з адаптацією платформ до швидкозмінного суспільного контексту, розробка надійної системи адміністрування є критично важливою. Чат, що спрощує взаємодію з користувачами, не змушуючи їх покидати вже відомі платформи, дозволяє надавати послуги максимальній кількості людей, зберігаючи конфіденційність та якість послуг.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Об'єктом дослідження є чат, що пропонує абстракцію над підтримуваними месенджерами, а також система адміністрації застосунку психологічної підтримки, що включає інтерфейси керування, системи моніторингу та аналітики, інструменти для управління користувацьким контентом та взаємодією з користувачами. Дослідження спрямоване на аналіз існуючих підходів до адміністрування подібних платформ та розробку нових методів, функцій і технологій, що можуть підвищити ефективність та гнучкість управління складною системою.

Предметом дослідження є методи модуляризації та приховування особливостей різних сутностей за єдиним інтерфейсом, а також інструменти адміністрування, які забезпечують високий рівень контролю, гнучкості та адаптивності управління ресурсами застосунку психологічної підтримки, інструментів для аналізу даних великих обсягів, систем управління контентом та взаємодією.

Кваліфікаційна робота бакалавра також передбачає вивчення та адаптацію передових практик з безпеки даних та приватності користувачів, враховуючи високу чутливість інформації, що обробляється в рамках платформи психологічної підтримки. Це включає розробку аутентифікації, гнучкої авторизації користувачів та контролю доступу як до системи так і до окремих її частин, фокусуючись на автоматизації процесу.

Значна увага була приділена розробці інтерфейсу адміністративної частини застосунку, що мав бути інтуїтивно зрозумілим, легким у використанні та ефективним для управління всіма аспектами платформи. Це включає створення дашборду для візуалізації статистики використання застосунку та відстеження активності користувачів та психологів. Основною частиною адміністративної системи є управління наповненням вебсайту. Створюючи досвід користувача, що дозволяє йому швидко вивчити систему, за допомогою оптимізації процесів з концентрацією на зменшенні відмінностей між модулями системи.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Результатом кваліфікаційної роботи бакалавра стала комплексна платформа, що складається з чату, дозволив надати користувачам якісну та своєчасну підтримку, та системи адміністрування, яка дозволила оптимізувати роботу застосунку психологічної підтримки, вивести його ефективність та гнучкість на новий рівень.

1.2 Аналіз аналогів програмного продукту

Існує велика кількість систем, що пропонують психологічну підтримку, як в Україні так і за кордоном. Суттєвим недоліком більшості таких систем в умовах збільшення попиту на покращення ментального здоров'я є висока ціна за послуги. Група застосунків, що є прямими аналогами створеному програмному продукту, надають послуги психологічної допомоги онлайн, деякі з використанням текстового листування. Саме ці програмні продукти мають значно нижчу ціну на послуги та націлені на найбільшу групу користувачів, тому їх визначено прямими аналогами системи.

Для аналізу аналогів було обрано лише українські продукти, які пропонують психологічну підтримку онлайн, оскільки саме вони найбільше відповідають цілям та меті проєктованого продукту.

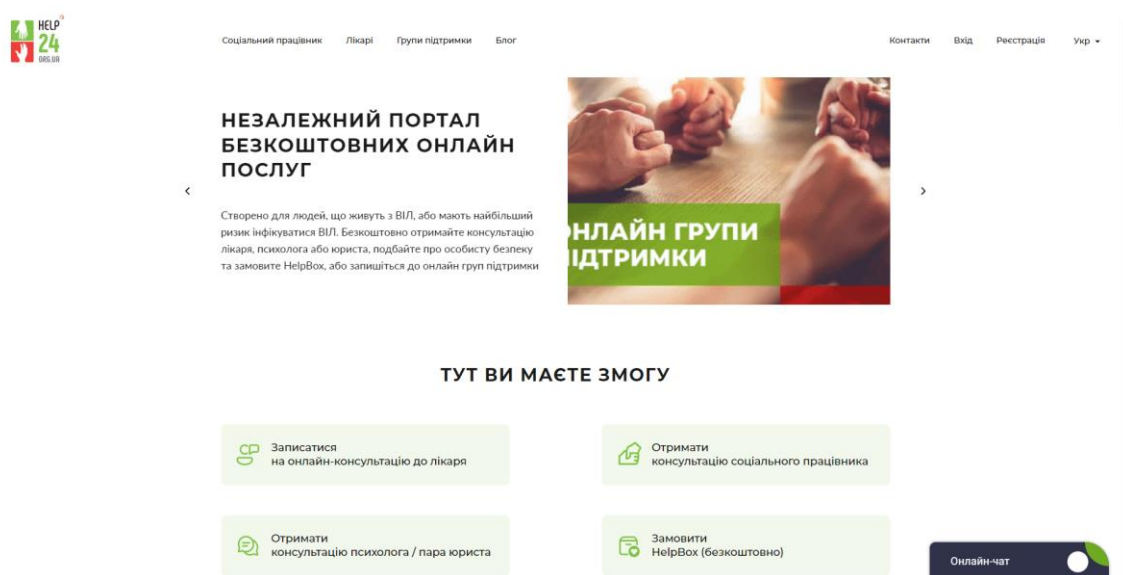


Рисунок 1.1. Головна сторінка сервісу.

На рисунку 1.1 зображено головну сторінку незалежного порталу безкоштовних онлайн послуг переважно пов'язаних з психологічним здоров'ям Help24 [4].

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Система надає широкий перелік послуг, пов'язаних із ментальним здоров'ям в першу чергу, в тому числі консультації та асистовані тестування від соціальних працівників, пошук та консультація з лікарями різних напрямків, адміністрування груп підтримки для людей з невиліковними хворобами. Система націлена охопити якнайбільшу аудиторію під егідою турботи та підтримки найменш захищених шарів населення.

Монетизація застосунку не має репрезентації на сайті, навіть при записі на консультації завчасно дізнатися ціну неможливо.

У застосунку є система авторизації та аутентифікації, однак використовується лише для базових потреб, як-от розміщення інформації адміністраторами, лікарями та авторами, а також збереження інформації про користувача. Решта дій, навіть звернення до лікарів, відбуваються в обхід системи авторизації адміністрацією системи. В особистому кабінеті користувач може бачити заплановані та проведені консультації.

Автоматизація процесів галузі у застосунку стосується лише поверхневих процесів та першої взаємодії користувача з організацією.

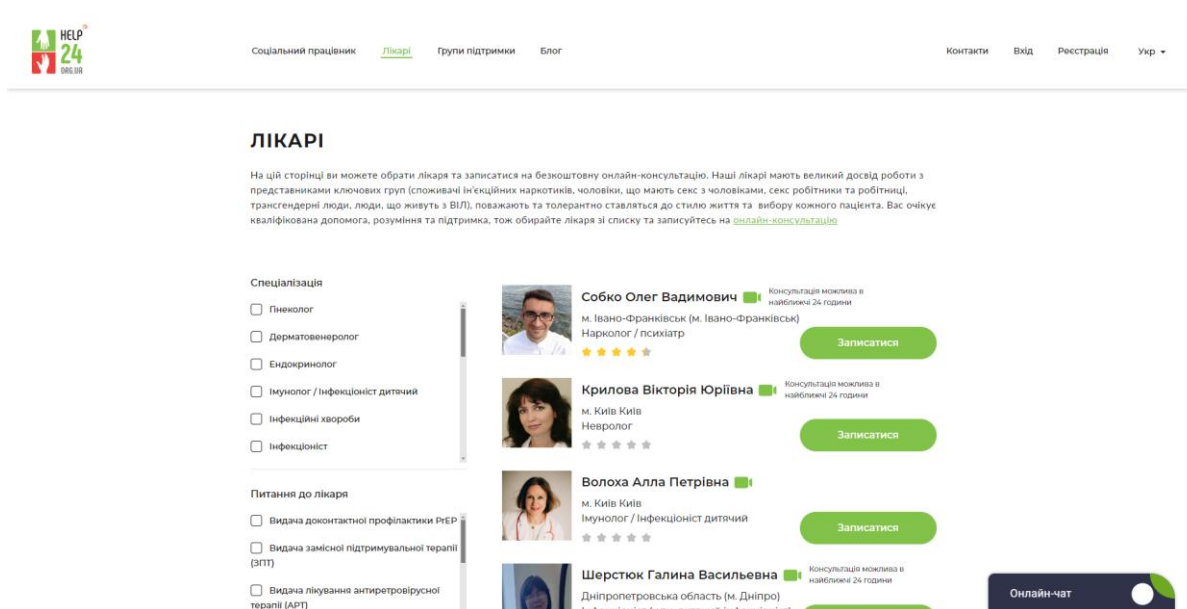


Рисунок 1.2. Сторінка пошуку лікаря.

Система адмініструє початкові та повторні звернення, однак не дозволяє лікарям комунікувати з клієнтами, чи навпаки, використовуючи платформу (рис.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2). Вони змушені користуватися сторонніми сервісами, які можуть не гарантувати конфіденційності.

Система не включає в себе адміністрування процесів, як-от створення та управління групами підтримки, тобто з моменту додання адміністраторами групи підтримки, управління нею повністю переходить в руки людей та паперові списки, замість електронної системи обліку.

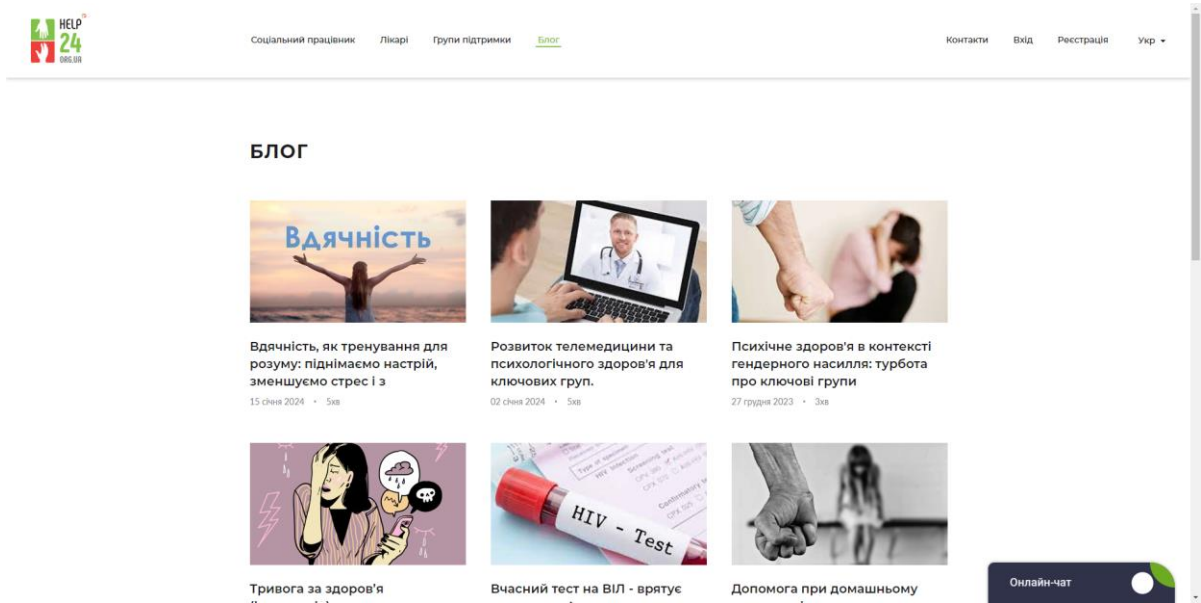


Рисунок 1.3. Сторінка «Блог».

Застосунок надає можливість лікарям та авторам створювати статті, збірки порад, інструкції зі збереження та покращення здоров'я та інше за допомогою інструменту Блог (рис. 1.3). Ця частина застосунку є однією з найважливіших для утримання уваги користувачів та потенційних користувачів, які ще не наважилися чи не мають змоги отримати консультацію лікарів чи інших соціальних працівників.

У вебзастосунку є сторінка контактів, що надає інформацію для написання фізичного листа організації, прямого текстового звернення (варто зазначити, що невідомо, коли з користувачем зв'яжуться). Для системи, що знаходиться у розробці, найважливішою частиною сторінки є посилання на Telegram та Viber боти. Особливості їх реалізації не було встановлено через небажання перешкоджати роботі та забирати час у працівників. Найімовірніше боти та чат на сайті реалізовані як дві окремі сутності, мають окремий програмний код та

адміністратори використовують їх окремо, оскільки розробка системи, подібної до запланованої у створюваному проєкті є занадто складною для застосунків, що не позиціонують спілкування із лікарями через онлайн-чат однією з основних функцій системи.

Перейдемо до платформи «Розкажи мені», що надає українцям безкоштовну психологічну допомогу [5].

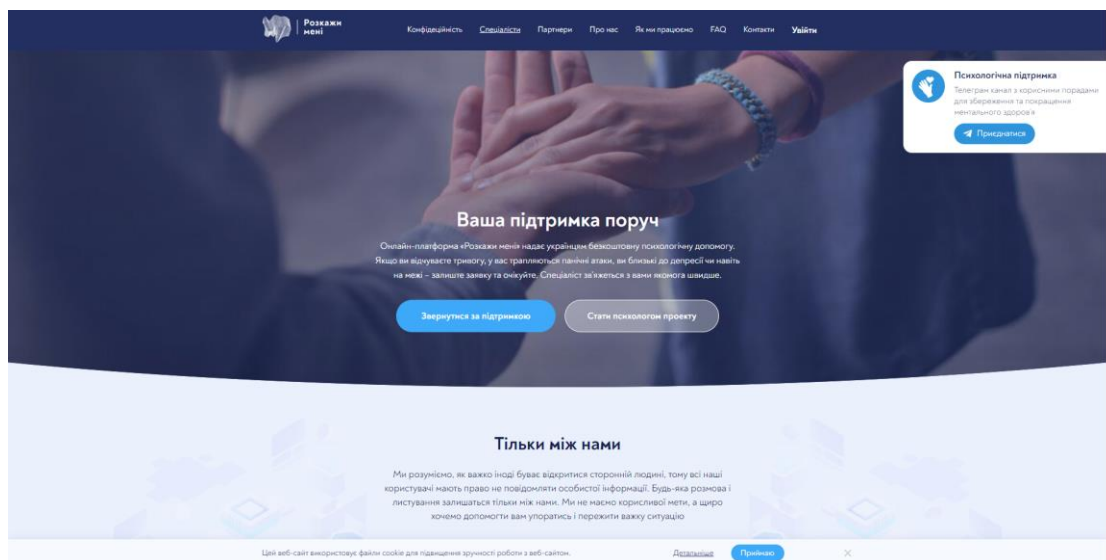


Рисунок 1.4. Головна сторінка сервісу.

На рисунку 1.4 зображено головну сторінку онлайн-платформи. Система надає послуги, пов'язаних із психологічною підтримкою, з особливою увагою до екстренної допомоги людям, що «знаходяться на межі». Система націлена допомогти людям у найскладніших ситуаціях, що потребують психологічної допомоги та не мають фінансової можливості скористатися платними консультаціями. Консультації відбуваються онлайн. Всі психологи проєкту є волонтерами.

Монетизація у системі повністю відсутня, всі консультації проводяться на волонтерських засадах професійними психологами, а адміністрування системи волонтерами, що започаткували проєкт.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

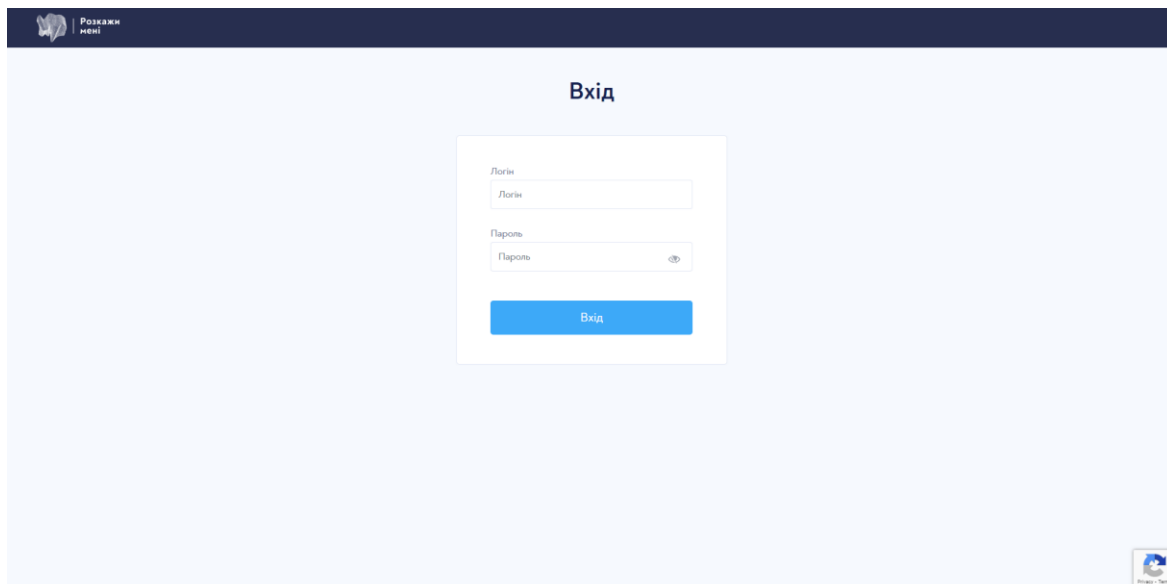


Рисунок 1.5. Сторінка входу в панель адміністратора.

Функціонал авторизації доступний лише адміністраторам та слугує для контролю доступу до редагування інформації на сайті (рис. 1.5). Користувачі можуть лише залишити свої дані, погодившись на умови платформи, аби з ними могли зв'язатися психологи.

Окрім зазначеного решта вебсайту виконує функцію запевнення користувача у тому, що йому варто звернутися за допомогою. Вторинною функцією також є донесення до користувача різноманітної інформації щодо принципу роботи платформи, пояснень з внесення заявки та з взаємодії з лікарями.

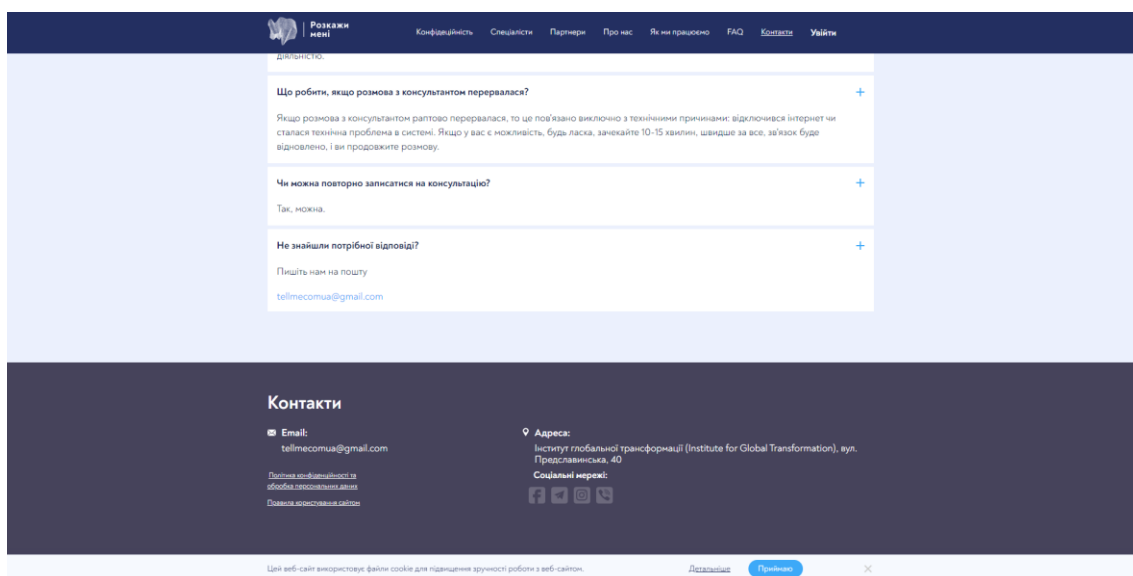


Рисунок 1.6. Частина сторінки із контактами та відповідями на запитання.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Система надає інформацію про різноманітні соціальні мережі застосунку, електронну пошту та фізичну адресу (рис. 1.6). Можливостей звернутися за підтримкою в текстовому чаті в системі не передбачено.

Далі розглянемо платформу Terlo, що сконцентрована на наданні легкодоступних платних послуг психологів онлайн [6].

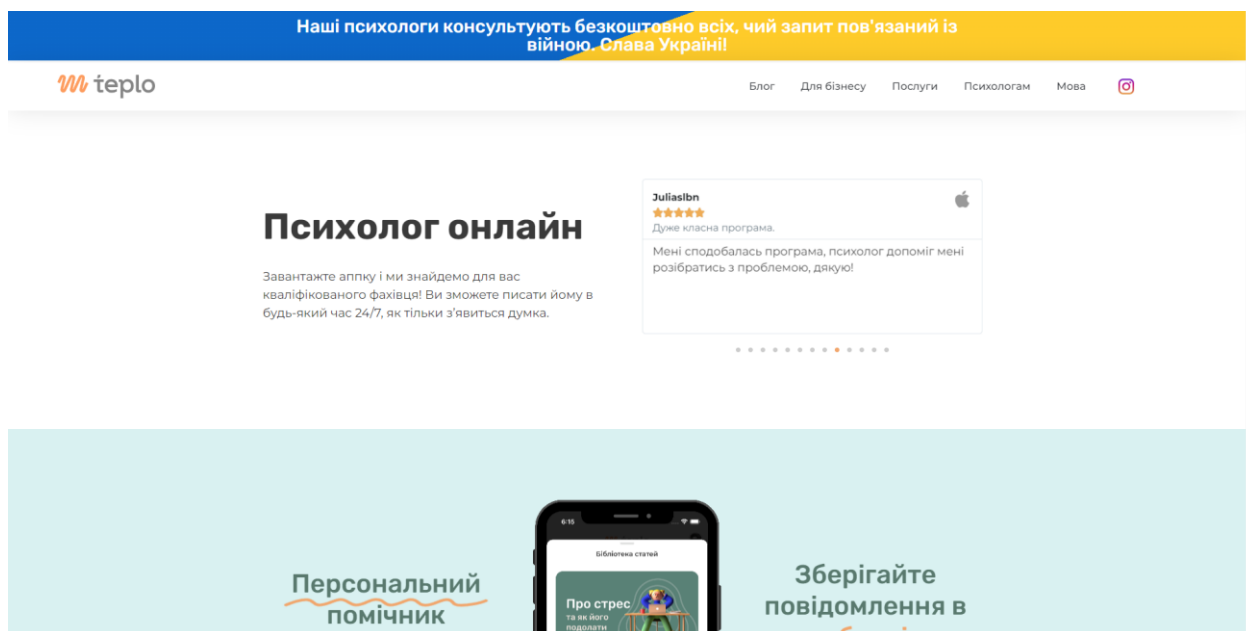


Рисунок 1.7. Головна сторінка сервісу.

На рисунку 1.7 зображено головну сторінку платформи. Система надає широкий перелік послуг, пов'язаних із психологічною допомогою, в тому числі онлайн консультації в електронному листуванні або відео-виклику та послуги психологічної допомоги для бізнесу. Система націлена охопити велику аудиторію через низьку ціну та велику кількість технологічних послуг, що зменшують ціну надання таких послуг.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

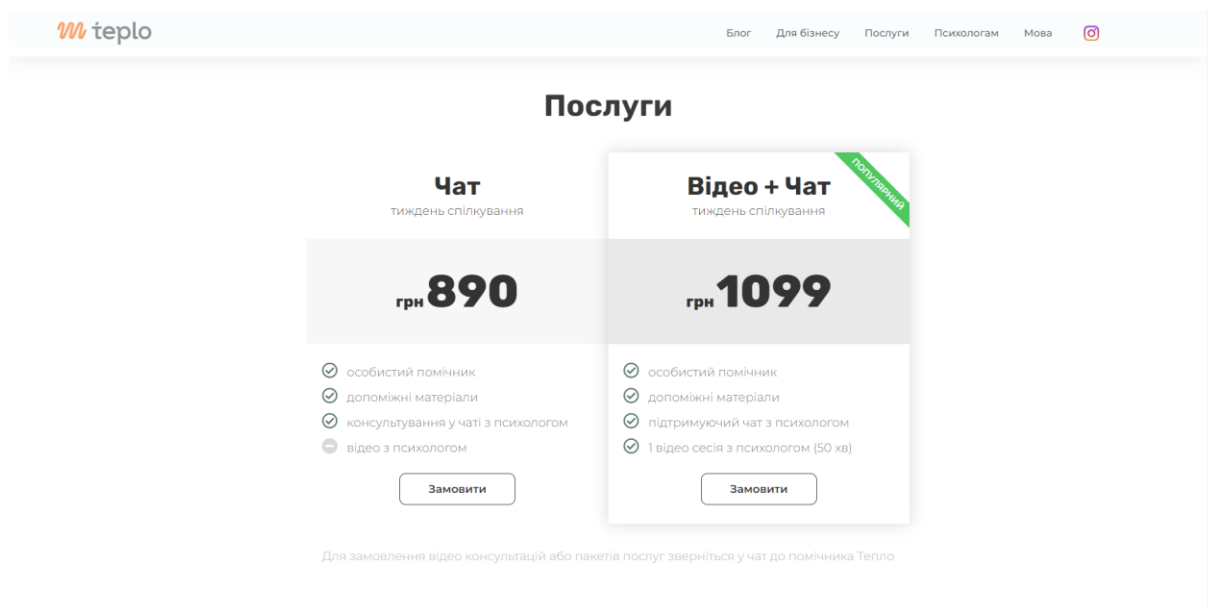


Рисунок 1.8. Монетизація сервісу.

Монетизація застосунку є основною ціллю. Система пропонує пересічному користувачу 2 типи послуг без відео-спілкування та з ним відповідно (рис. 1.8). Також пропонуються «пакети» відео-консультацій. У всі сервіси входять допоміжні матеріали та «особистий помічник», який є штучним інтелектом.

Вебсайт використовується виключно у маркетингових цілях, а уся бізнес-логіка винесена у мобільний застосунок. У застосунку є система авторизації та аутентифікації користувачів, яка активно використовується для контролю доступу користувачів до системи. Також доступна авторизація для авторів та адміністраторів для створення блогів.

Автоматизація процесів галузі у застосунку на високому рівні, оскільки включає в себе оплату, зв'язок між користувачем та працівниками компанії, розміщення та перегляд блогів та матеріалів, а також штучний інтелект.

Застосунок надає можливість лікарям та авторам створювати статті, збірки порад, інструкції зі збереження та покращення психологічного здоров'я та інше за допомогою інструменту Блог, однак мало використовується та має вторинне значення, оскільки основний перелік матеріалів доступний платним користувачам у застосунку.

У вебзастосунку є сторінка контактів, що надає інформацію про різноманітні сервіси, лікарні та індивідуальні практики лікарів-психіатрів та

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

лікарів-психологів, однак сторінка не відповідає законодавству України, оскільки має лише версію іноземною мовою.

Додатково було розглянуто сайт UAMH [7], організації, що «підкується про ментальне здоров'я українців». Система включає швидкий спосіб звернутися за допомогою (однак не отримати її, оскільки після запиту психолог ймовірно вийде на зв'язок лише через деякий час), повноцінну візитівку, можливість долучитися до організації з усією потрібною інформацією, сторінкою зі звітністю організації, оскільки вона фінансується меценатами. Окремо варто розглянути сторінку з поширеними питаннями та відповідями, що покриває широкий спектр можливих питань, однак не дозволяє користувачам задати нові питання. Однак весь функціонал окрім суто інформаційного забезпечується зовнішніми сервісами, тобто система майже не автоматизує процеси в предметній області. У зв'язку з зазначеними недоліками сайт виключено з порівняльної характеристики.

Таблиця 1.1

Порівняння функціональних можливостей

Функція	Help24	Розкажи мені	Терло	Розроблений застосунок
Чат з психологом	Ні	Ні	Так	Так
Інтеграція зі сторонніми платформами	Ні	Ні	Ні	Так
Вбудований відео-чат	Ні	Ні	Так	Ні
Матеріали для підтримки психологічного здоров'я	Так	Ні	Так	Так
Можливість вибору лікаря	Так	Ні	Ні	Ні
Контакти	Так	Так	Ні	Ні

Продовження таблиці 1.1

Відповіді на часті запитання	Ні	Так	Ні	Так
Авторизація користувачів	Так	Ні	Так	Так
Можливість сплатити за послуги	Ні	Ні	Так	Так
Мобільний застосунок	Ні	Ні	Так	Ні
Повний набір функцій при використанні стільникової версії	Так	Так	Ні	Так
Інструменти для збереження релевантної інформації, доступні психологам	Ні	Ні	Ні	Так

У таблиці 1.1 відображено виокремлені особливості кожної з систем. Всі аналоги мають свої особливості та переваги, однак найбільші досягнення з автоматизації процесів зі сторони користувача серед розглянутих аналогів розробленого застосунку показав сервіс Терпо, оскільки пропонує користувачу найбільше цифрових рішень в мобільному застосунку, будучи єдиним сервісом, що має власний програмний продукт для мобільних пристроїв. Водночас розроблений застосунок поєднує найкращі інструменти для адміністраторів та психологів, виводячи досвід персоналу на новий рівень.

Отже, ключовими можливостями в новій системі психологічної підтримки повинні бути матеріали для підтримки психологічного здоров'я, чат з психологом, контакти пов'язаних установ, відповіді на часті запитання, авторизація адміністраторів, а за умови рішення монетизації системи також і авторизація користувачів. Система повинна мати повний функціонал при використанні стільникової версії, оскільки саме у ній найбільші можливості з надання цифрових рішень.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савицький Р.С.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3 Вибір архітектури системи адміністрування застосунку психологічної підтримки

Враховуючи вимоги до системи, прийнято рішення розділити систему на дві частини: вебзастосунок та роутер повідомлень, оскільки це дозволить повноцінно розділити систему за логікою та цілями. Це спростить та напряду покращить реалізацію системи як одного цілого.

Для розробки вебзастосунку обрано монолітну архітектуру. Програмне забезпечення включає в себе чат, базу знань, статті, а також багато інших функцій. Вибір архітектури є стратегічним рішенням, що базується на комплексному аналізі технічних та бізнес-потреб проєкту. Вибір монолітної архітектури підкріплюється рядом ключових переваг, які відповідають особливостям та цілям даного застосунку.

Монолітна архітектура дозволяє централізовано управляти типізацією та валідацією даних, що є критично важливим для забезпечення цілісності та безпеки інформації у системі психологічної підтримки. Це сприяє уніфікації обробки даних на всіх рівнях застосунку, зменшуючи ймовірність помилок та спрощуючи процес розробки та тестування.

Використання tRPC у монолітній архітектурі забезпечує ефективну інтеграцію між різними частинами системи, дозволяючи розробникам легко створювати типізовані та надійні application programming interface (API) ендпойнти. tRPC спрощує взаємодію між клієнтською та серверною частинами, підтримуючи строгую типізацію даних, що забезпечує додатковий рівень безпеки та знижує час розробки за рахунок автоматизації багатьох процесів [8].

Монолітна архітектура сприяє високій надійності системи, оскільки всі компоненти застосунку тісно інтегровані та працюють у єдиному виконавчому середовищі. Це уможливорює легке виявлення та швидке усунення помилок.

Монолітна архітектура спрощує процес розробки та підтримки системи, оскільки розробники працюють з єдиною кодовою базою. Це значно знижує складність інтеграції та тестування різних компонентів системи, а також уніфікує

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

процес розгортання. Така єдність досвіду розробки сприяє кращому розумінню системи в цілому та полегшує діагностику та виправлення помилок.

У монолітних системах керування залежностями є менш складним, ніж у мікросервісних архітектурах, де кожен мікросервіс може мати свій набір залежностей. Це знижує ризик конфліктів залежностей та спрощує процес оновлення бібліотек та фреймворків [9].

Монолітна архітектура дозволяє консолідувати заходи безпеки у єдиному місці, замість того, щоб розділяти та дублювати їх серед багатьох сервісів. Це полегшує управління доступом, аутентифікацією, шифруванням даних та іншими аспектами безпеки, забезпечуючи комплексний захист всієї системи.

Враховуючи специфіку проекту веборієнтованої системи психологічної підтримки, яка вимагає високої надійності, масштабованості та зручності використання, монолітна архітектура є оптимальним вибором. Вона не тільки сприяє ефективній реалізації поставлених завдань, але й забезпечує міцну основу для стабільного та безпечного розвитку системи в майбутньому. Монолітна архітектура дозволяє зосередитися на якісній реалі

Вибір проксі-архітектури для реалізації роутера повідомлень, основними завданнями якого є створення та зберігання чатів і повідомлень, обґрунтований низкою важливих переваг та характеристик, які ідентифікуються в умовах сучасних вимог до систем реального часу та ефективності обробки великих об'ємів даних.

Перш за все, проксі-архітектура забезпечує оптимізований шлях комунікації між клієнтами та сервером основного застосунку, що дозволяє мінімізувати затримки у передачі повідомлень та підвищує загальну реактивність системи. Використання протоколу WebSocket в застосунку, що має проксі-архітектуру, дозволяє здійснювати двосторонній обмін даними в реальному часі, що є критично важливим для систем миттєвого обміну повідомленнями [10].

Розмежування логіки роботи з повідомленнями і чатами в окремий компонент (проксі) сприяє покращенню масштабованості та легкості

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

обслуговування системи. Це дозволяє централізовано управляти з'єднаннями, а також ефективно масштабувати систему зі збільшенням навантаження, без необхідності модифікації основного монолітного застосунку.

Архітектурна ізоляція компонента, що відповідає за реалізацію функціоналу чатів і повідомлень, сприяє також підвищенню безпеки системи. Проксі може виконувати роль додаткового шару безпеки, аналізуючи та фільтруючи вхідні з'єднання та запити перед їх потраплянням до системи.

Додатково, проксі-архітектура надає гнучкість в інтеграції з іншими системами та сервісами, що може бути корисним для розширення функціоналу, наприклад, інтеграції з системами аналітики, моніторингу чи зовнішніми API. Обрана архітектура також дозволяє додавати чи прибирати підтримку платформ обміну повідомлення жодним чином не модифікуючи, ні ядро, ні модулі сервера.

Враховуючи наведені аргументи, можна зробити висновок, що вибір проксі-архітектури для реалізації роутера повідомлень є обґрунтованим та ефективним рішенням, що відповідає актуальним тенденціям розробки високопродуктивних та масштабованих систем миттєвого обміну повідомленнями. Така архітектура не лише забезпечує високу пропускну спроможність та низьку затримку в обміні повідомленнями, але й сприяє підвищенню надійності системи через можливість реалізації розподілених технік відновлення при відмовах та балансування навантаження.

Додатково, архітектура з використанням проксі дозволяє оптимізувати управління ресурсами сервера. Завдяки можливості масштабування проксі-серверів, система може адаптуватися до змінних обсягів трафіку, розподіляючи навантаження для забезпечення безперервності та якості обслуговування. Це особливо актуально в умовах різких піків активності користувачів або при розгортанні нових функцій, що вимагають додаткових ресурсів.

З погляду розробки та підтримки, використання проксі-сервера як окремого компонента спрощує процес оновлення та впровадження змін в системі. Розробники можуть вносити модифікації до проксі-сервера або основної

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

системи незалежно, зменшуючи ризики порушення роботи системи та спрощуючи процес тестування.

Також, важливим аспектом є відкритість системи для інтеграції з майбутніми технологічними нововведеннями. Архітектурна гнучкість, що надається проксі-сервером, гарантує, що система залишиться сумісною з розвитком нових протоколів комунікацій, стандартів безпеки та методів шифрування, що з часом можуть стати невід'ємною частиною інфраструктури обміну повідомленнями.

У підсумку, стратегічний вибір проксі-архітектури для роутера повідомлень не тільки відображає передові практики у сфері програмної інженерії, але й забезпечує міцну основу для майбутнього розвитку та оптимізації системи. Це рішення демонструє глибоке розуміння потреб бізнесу у створенні ефективних, безпечних та легко масштабованих рішень для миттєвого обміну повідомленнями.

Наступним кроком визначено модель бази даних. В умовах розробки системи психологічної підтримки, обрано реляційну базу даних, спираючись на набір вирішальних критеріїв, які відповідають вимогам до проєкту. Реляційні бази даних надають виняткові можливості для забезпечення суворого дотримання структур даних, що має важливе значення для систем, де важлива точність, цілісність та детерміністична структуризація інформації.

Реляційні бази даних відрізняються високим рівнем забезпечення строгої відповідності даних. Досягається зазначена відповідність за допомогою обмежень ідентифікації, таких як первинні та зовнішні ключі, обмеження унікальності та обов'язковості даних, а також строгої статичної типізації значень. Це дозволяє забезпечити точність та консистентність даних, що зберігаються, обробляються і курсують у системі.

Важливим недоліком реляційних баз даних є обмеження з точки зору горизонтального масштабування порівняно з нереляційними базами. Обмеження часто компенсуються їх здатністю ефективно обробляти складні запити.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

Таким чином, враховуючи цілісний аналіз потреб проєкту, вибір реляційної бази даних є стратегічно важливим та обґрунтованим рішенням, спрямованим на створення міцної, надійної та безпечної системи.

Враховуючи викладене, комбінація монолітної архітектури та проксі-сервера для системи психологічної підтримки є не лише виправданою з технічної точки зору, але й стратегічною з погляду довгострокового розвитку та адаптації до майбутніх викликів. Можливості обраної архітектури підсилюються реляційною базою даних, що дозволить проєктувати сутності з великою кількістю взаємозв'язків, ефективного структурування даних та мінімізації дуплікації інформації.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Застосунок розділено на дві частини. Таким чином роутер повідомлень, що виконує перенаправлення повідомлень між оператором та користувачем, абстрагуючись від конкретного месенджера, тобто має найбільше навантаження, буде виконана у вигляді окремого застосунку, що дозволить полегшити та автоматизувати масштабування обох частин системи.

Надалі наведено обґрунтування вибору технологій для основної частини системи. В першу чергу варто зазначити, що початковий набір технологій було запозичено з офіційного прикладу Next.js [11], однак з часом розширено актуальним стеком ТЗ [12], та додатково розширено технологіями, необхідними конкретно розробленому застосунку.

Виходячи з досвіду в комерційній розробці, вибір мови програмування JavaScript є стратегічним рішенням. Мова загального використання та найбільше використовується у вебзастосунках, оскільки вбудована в усі браузери за замовчуванням [13].

Корисною надбудовою над обраною мовою є TypeScript, що інтегрує легкість та поширеність JavaScript із перевагами, характерними для мов програмування зі строгою типізацією [14]. TypeScript значною мірою є лише

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

«помічником» у підсвічуванні помилок та недоліків у коді та потребує розширень для забезпечення найкращого досвіду розробки.

Основне з таких покращень – ESLint, що додає набір власних правил та дозволяє гнучко налаштувати підсвічування у середовищі розробки під потреби проєкту та розробника [15]. Додані правила є не строгими щодо форматування, а у проєкті, де більше одного розробника, воно стає помітною проблемою. Її рішенням є Prettier, що чітко визначає правила форматування та вбудовується в ESLint для забезпечення сповіщення розробників про невідповідність їх коду прийнятому стандарту форматування [16]. Для довершення набору покращення досвіду розробки використано Husky – модуль, що дозволяє легко створювати та керувати хуками Git [17]. Таким чином забезпечено обов’язкову перевірку коду репозиторія на відповідність стандартам перед тим, як розробник зможе зберегти свої зміни у коміті.

JavaScript також широко застосовуваний на стороні сервера, що дозволило відділення двигуна JavaScript, V8, розробленого компанією Google для браузера Google Chrome. Середовище виконання Node.js – це двигун V8 разом з реалізацією набору інструментів та решти компонентів середовища виконання [18]. Зазначене середовище має велику кількість зобов’язань щодо зворотної сумісності, оскільки розробляється вже 15 років. У зв’язку з цим за останні роки з’явилася велика кількість конкуруючих проєктів, що намагаються покращити досвід розробки, швидкість, ефективність та ряд інших параметрів. Одним з таких є Bun, що є швидшим та ефективнішим на кілька порядків за існуючі аналоги [19]. Додатково середовище пропонує безпроблемний та простий досвід використання. Ключовим у виборі стала нативна підтримка TypeScript, що вирішує проблеми мови пов’язані з необхідністю транспіляції в JavaScript для виконання.

Менеджером пакетів у застосунку прийнято рішення використовувати Node Package Manager Registry (NPM), хоча він і має перелік недоліків, пов’язаних з віком як і у Node.js, на момент написання роботи він не має повноцінних аналогів [20].

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

Оскільки TypeScript лише покращує досвід розробки додаючи типи до вихідного коду, він не додає жодної типізації в працюючий застосунок, тож виникає потреба також впевнюватися в типах даних під час виконання. Для виконання задачі обрано бібліотеку Zod, яка серед конкурентів є однією з небагатьох, що інтегрується з Typescript та є широко використовуваною, що робить її найкращою в цій ніші [21]. Модулем розширення обрано Zod-i18n, що дозволяє перекласти помилки українською мовою, оскільки це основна натуральна мова проєкту [22]. Для забезпечення повноцінної перевірки непримітивних типів використано Validator.js [23], який надає широкий спектр функцій-валідаторів.

React як бібліотека для розробки лицевих вибрана через її велику популярність та активну спільноту, що забезпечує доступ до великої кількості ресурсів, навчальних матеріалів та готових до використання компонентів [24]. React дозволяє створювати інтерактивні користувацькі інтерфейси з високим рівнем абстракції компонентів, спрощуючи процес розробки та підтримки застосунків. Особливістю React є використання віртуального Document Object Model (DOM), що забезпечує високу продуктивність шляхом мінімізації дорогих операцій з реальним DOM.

Next.js використовується як мета-фреймворк для спрощення та пришвидшення фулстек-розробки завдяки його підтримці серверного рендерингу (SSR), статичної генерації сторінок (Static Site Generation – SSG) та інших оптимізацій, що покращують продуктивність [25]. Next.js спрощує маршрутизацію, впровадження API-ендпоінтів та інтеграцію з серверними функціями, роблячи його вибором номер один для розробки сучасних вебзастосунків, які потребують швидкого завантаження та оптимізації для пошукових систем.

Найбільшою перевагою в контексті архітектури застосунку, Next.js є розділення застосунку на статичні та динамічні компоненти під час розгортання. Наступним кроком перші розгортаються на Amazon S3, а другі – на Amazon Lambda, що максимізує масштабованість. Ця особливість дозволяє зберегти всі

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

переваги монолітного застосунку та отримати більшість переваг мікросервісної архітектури, взамін додаючи невелику кількість обов'язкових паттернів під час розробки (як-от явний динамічний імпорт більшості бібліотек лицевіці, оскільки під час статичної генерації сторінок не існує об'єкта window та багатьох інших сутностей).

Використання Cascading Style Sheets (CSS) фреймворку TailwindCSS стає логічним рішенням, враховуючи його синергію з обраними технологіями та загальну архітектурну стратегію проєкту [26]. Tailwind CSS, будучи утилітарним першокласним CSS-фреймворком, пропонує розробникам гнучкий підхід до стилювання вебінтерфейсів, дозволяючи швидко застосовувати дизайнерські рішення без необхідності писати велику кількість CSS. Ця особливість Tailwind CSS ідеально вписується в стратегію мінімізації навантаження на застосунок та спрощення процесу розробки, дозволяючи розробникам зосередитись на бізнес-логіці, а не на деталях реалізації стилів.

TailwindCSS є міцним фундаментом для розробки лицевіці, однак для швидшого ітерування обрано також бібліотеку з примітивними Radix UI [27]. Набір модулів Radix надає примітивні компоненти як-от кнопки, поля та інші компоненти інтерфейсу.

Окрім примітивних компонентів важливо також зберігати уніфікований загальний вигляд застосунку. Для вирішення задачі обрано Shadcn UI, що є набором як примітивних так і складних компонентів, які часто є надбудовою над компонентами з Radix UI [28]. Бібліотека дозволить притримуватися єдиного дизайну та айдентики у всьому застосунку, також зменшуючи кількість місць, де потрібно вирішувати однотипні задачі, надаючи для них готові рішення.

У зв'язку з необхідністю ведення статистики у застосунку, прийнято рішення використати бібліотеку для візуалізації графіків React Charts, яка обгортає модуль Charts.js [29]. Бібліотека оптимізує відображення серіалізованої інформації як-от хронологія повідомлень чи активних чатів.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

Для покращення користувацького досвіду також використано модуль `Date-fns` [30] для форматування та маніпуляції датами і модуль `Use-sound` [31] для програвання звуків операторам під час отримання нових повідомлень у чаті.

`Prisma` обрана як бібліотека-ОРВ для спрощення взаємодії з базами даних завдяки її високому рівню абстракції та підтримці багатьох систем управління базами даних (СУБД) [32]. `Prisma` пропонує потужний набір інструментів для моделювання даних, міграцій бази даних та зручного формування запитів через свій типізований API, що дозволяє розробникам ефективно працювати з даними, забезпечуючи безпеку та масштабованість застосунків. Вибір бібліотеки підкріплений в першу чергу її фокусом на підтримці реляційних баз даних. Головним недоліком є помилкові рішення під час проєктування як-от відсутність нативних `Join` операцій, що погіршує продуктивність усієї системи.

Для забезпечення використання найкращих практик аутентифікації користувачів обрано сучасну бібліотеку `NextAuth` [33]. Модуль дозволяє використовувати десятки різних методів аутентифікації, однак було обрано найбільш поширені серед цільової аудиторії – `Google`, `Facebook` та магічне посилення (аутентифікація за електронною поштою). Бібліотека інтегрується з `Prisma` за допомогою офіційного адаптера та не потребує додаткового налаштування, окрім чітко задокументованих змін, пов'язаних з підключенням окремих провайдерів аутентифікації.

Для тиловиці основного застосунку використано `tRPC` [34]. Бібліотека дозволяє спроектувати «сузір'я» процедур, тобто вони можуть залежні одна від одної. Таким чином у архітектурі `Remote Procedure Call (RPC)` у цій бібліотеці реалізовано `middleware`. Використовуючи цю та інші переваги `tRPC` (як-от мета-дані процедур) можна реалізувати підсистеми авторизації, аутентифікації та різноманітної модифікації отримуваних запитів у вигляді їх контексту.

Оскільки система спирається на сервіси `Amazon` у аспектах збереження зображень та надсилання повідомлень на електронну пошту – використано набір модулів з бібліотеки `AWS SDK v3` [35], а також консольний застосунок для

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

спрощення розробки та тестування взаємодії з сервісами Amazon Web Services (AWS) [36].

Надалі наведено обґрунтування вибору технологій для частини системи, що переспрямовує повідомлення між месенджерами та вебзастосунком.

Python як основна мова програмування вибрана за її універсальність, читабельність та велику кількість бібліотек, що дозволяє швидко розробляти складні застосунки з різноманітною функціональністю [37]. Python підтримує різні парадигми програмування (об'єктно-орієнтовану, процедурну, функціональну), що робить його гнучким у використанні для різних типів проєктів. Широке використання Python в наукових дослідженнях, веброзробці, автоматизації, аналітиці даних та штучному інтелекті забезпечує велику спільноту та підтримку, що є важливими факторами при виборі мови програмування для кваліфікаційної роботи бакалавра.

Частиною екосистеми обраної мови є Python Package Index (PyPI) [38], який дозволяє отримувати необхідні модулі та контролювати їх версію. Також частиною системи є консольний застосунок `pip`, однак його не було використано в роботі через низьку спроможність контролю версій за пакунків. Для вирішення цих проблем використано `Poetry` [39], застосунок, що надає повний контроль над пакетами та їх версіями, що дозволяє притримуватися стабільної версії залежностей та гарантувати повторюваність розгортань на різних системах та пристроях.

`FastAPI` обрано для розробки тиловиці завдяки його високій продуктивності, простоті у використанні та потужним можливостям для створення асинхронних API [40]. `FastAPI` автоматично генерує документацію за допомогою `Swagger UI` та `ReDoc`, що спрощує розробку та тестування API. Бібліотека підтримує сучасні стандарти Python, такі як асинхронність (`asyncio`) та `type hints`, що дозволяє писати більш чистий та безпечний код. Ці особливості роблять `FastAPI` ідеальним вибором для швидкої розробки ефективних та легко масштабованих вебсервісів.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

Pydantic використовується для валідації, санітизації та типізації даних, що дозволяє забезпечити високу якість вхідних та вихідних даних API [41]. Ця бібліотека інтенсивно використовує Python type hints, що сприяє розробці більш безпечного та легко підтримуваного коду. Pydantic спрощує обробку та перевірку складних даних, автоматично перетворюючи їх у Python об'єкти, що значно підвищує продуктивність розробника та зменшує кількість помилок у застосунку.

SQLAlchemy є бібліотекою-ОРВ (об'єктно-реляційне відображення), яка спрощує взаємодію з базами даних, дозволяючи розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід [42]. SQLAlchemy підтримує багато різних СУБД, надаючи єдиний, уніфікований API для взаємодії з ними, що робить код більш портативним та легко адаптованим під різні бази даних. Особливості, такі як потужний механізм запитів, підтримка транзакцій та можливість розширення, роблять SQLAlchemy вибором для розробників, які шукають гнучкість та ефективність при роботі з реляційними базами даних.

Для взаємодії з сервісами AWS із застосунку використано офіційний модуль boto3 [43]. Модуль передбачає аналогічний набір функціоналу до JavaScript версії.

Оскільки усі обрані інструменти підтримують широкий перелік СУБД, прийнято рішення порівняти кілька найпопулярніших для здійснення вибору.

Таблиця 1.2

Порівняння особливостей СУБД

Особливість	PostgreSQL	MySQL	MSSQL Server
Тип	Реляційна (RDBMS)	Реляційна (RDBMS)	Реляційна (RDBMS)
Відкрите ПЗ	Так	Так	Ні
Відповідність ACID	Так	Так (з двигуном зберігання InnoDB)	Так
Реплікація	Master-Slave, Multi-Master	Master-Slave, Master-Master	Master-Slave
Партціонування	Так	Так	Так
Підтримка JSON	Так	Так	Так
Геопросторова підтримка	Так	Так (з розширеннями)	Так

Ліцензія	Ліцензія PostgreSQL (схожа на MIT)	GPL (v2) або Пропрієтарна	Пропрієтарна
Популярні випадки використання	Вебзастосунки, Аналітика, Геопросторові бази даних	Вебзастосунки, Електронна комерція, Логування застосунків	Корпоративні застосунки, Бізнес-аналітика, Обробка транзакцій

У таблиці 1.2 приведено порівняння трьох поширених СУБД, що мають набір інструментів для комплексних застосунків, необхідний в контексті проєктованого застосунку.

Оскільки PostgreSQL є найвідкритішою серед конкурентів, має вільну ліцензію, найшвидше розвивається, має найбільшу підтримку в будь-якій сфері, окрім екосистеми C#, та є дуже популярною, обрано саме її [44].

Для забезпечення найкращого досвіду розробки обрано лідерів у сфері середовищ розробки. Для основного застосунку це WebStorm [45], який забезпечує підтримку не лише обраної мови програмування, а й більшості обраних модулів як-от Next.js, Prisma та інших. Для роутера повідомлень обрано PyCharm [46], що також надає розширену підтримку використовуваних модулів. Згадані середовища поділяють єдине ядро інтерфейсу, тому не вимагають вивчення кожного окремо, що пришвидшує розробку при постійному перемиканні між проєктуванням двох застосунків, особливо під час інтеграції між ними.

Використання найкращих практик зобов'язує реалізувати систему постійного розгортання та інтеграції системи.

Основний застосунок має перевагу: він використовує Next.js, який розробляється компанією Vercel, яка фокусується на монетизації розробницького досвіду, надаючи найкращий інструментарій розгортання застосунків на інфраструктурі AWS. Роутер повідомлень не має подібної переваги, однак використовує набір інструментів, що спрощують розгортання на будь-якій розповсюдженій системі.

Код застосунків зберігається у GitLab, який надає інтерфейс для запуску власних «раннерів» [47], що можуть проводити будь-які bash скрипти, що

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

використовують спеціальний формат, розроблений GitLab [48]. Основний застосунок не потребує додаткових налаштувань чи застосунків для розгортання, достатньо налаштувати раннер та створити конфігураційний файл розгортання. Для роутера повідомлень окрім зазначеного також необхідно підготувати Docker [49], та включити в конфігураційний файл передання змінних середовища.

Отже, для реалізації системи обрано наступні інструменти:

1. Основна мова програмування: Typescript.
2. Бібліотека для валідації, санітизації та типізації: Zod.
3. Бібліотека для розробки лицевих: React.
4. Бібліотеки React-компонентів для пришвидшення розробки лицевих: Radix UI, Shadcn UI, React Charts.
5. Мета-фреймворк для спрощення та пришвидшення фулстек-розробки: Nextjs.
6. Бібліотека-ОРВ для спрощення взаємодії із СУБД: Prisma.
7. Бібліотека для забезпечення аутентифікації: NextAuth.
8. Бібліотека для взаємодії із AWS: AWS SDK for JavaScript v3.

Для написання програмного коду застосунку роутера повідомлень обрано наступні інструменти:

1. Основна мова програмування: Python.
2. Бібліотека для розробки тилових: FastAPI.
3. Бібліотека для валідації, санітизації та типізації: Pydantic.
4. Бібліотека-ОРВ для спрощення взаємодії із СУБД: SQLAlchemy.
5. Бібліотека для взаємодії із AWS: Boto3.

Системою управління баз даних обрано PostgreSQL. Для розгортання системи використано GitLab Runner, Docker та сервіс Vercel.

Висновки до першого розділу

Проаналізувавши наявні підходи до адміністрування застосунків психологічної підтримки, можна зробити висновок про необхідність створення рішення, яке б відповідало сучасним вимогам забезпечення ментального здоров'я користувачів. Важливість проєкту підкріплюється зростаючим попитом

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

на психологічні послуги та необхідністю інтеграції новітніх інформаційних технологій для оптимізації взаємодії між користувачами та фахівцями. Розроблення ефективної системи адміністрування дозволить не тільки поліпшити якість обслуговування, але й значно спростити процес управління контентом та інтеракціями в рамках застосунку.

Обрано технології для розробки системи згідно з вимогами та цілями проєктованої системи.

Окремо слід відзначити рішення про розробку модульних функцій, які забезпечать адаптивність та масштабованість системи. Реалізація автоматизованого чату та модулів управління сесіями, реєстрації користувачів та публікації контенту відіграватиме ключову роль у створенні зручного та інтуїтивно зрозумілого інтерфейсу. Такий підхід до проєктування та розробки системи не лише задовольнить поточні потреби користувачів, але й гарантує легкість внесення змін та додавання нових функціональних можливостей у майбутньому.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

2. ПРОЄКТУВАННЯ СИСТЕМИ АДМІНІСТРУВАННЯ ЗАСТОСУНКУ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

2.1 Визначення варіантів використання та структури системи

Основні цілі системи адміністрування вебзастосунку психологічної підтримки – надати ергономічний інтерфейс з широким набором можливостей управління сутностями системи. У першу чергу система спрямована на оптимізацію процесів адміністрування наповнення комплексної системи наповнення застосунку. Це дозволяє персоналу платформи мати повний контроль над пропонованою інформацією, покращуючи їх досвід.

У розробленому застосунку також інтегровано чат для надання психологічної підтримки, його основна ціль – забезпечувати високоякісний та ефективний спосіб спілкування між користувачами та операторами. Перш за все, система спрямована на інтеграцію різноманітних платформ для комунікації, включно з популярними месенджерами та власною платформою, забезпечуючи легкий доступ до чату. Це дозволяє користувачам платформи отримати швидке та зручне з'єднання з операторами та необхідними ресурсами щодо психологічної підтримки.

Також, система акцентує увагу на особистісному підході, дозволяючи операторам самостійно оцінювати свою зайнятість для оптимізації розподілу робочого навантаження.

Внутрішня база знань служить як засіб для поліпшення кваліфікації операторів, надаючи їм доступ до актуальної інформації у сфері. Модуль відтворює спрощену версію «хмарних» дисків, однак фокусується на полегшенні взаємодії та найкращому досвіді при роботі саме з документами.

Система розподілу чатів базується на алгоритмах, що враховують метрики швидкості відповіді, кількості нещодавніх чатів, кількості нещодавніх повідомлень та суб'єктивної оцінки зайнятості безпосередньо оператором, що сприяє збалансованому та справедливому розподілу завдань серед операторів.

На рівні адміністрації персоналу, основні завдання включають управління, від створення до видалення облікових записів, а також деталізоване

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

налаштування доступу до функціоналу системи. Збір та аналіз статистичних даних виконується для визначення продуктивності операторів та ефективності системи в цілому, що включає моніторинг середнього часу відповіді та кількості чатів.

Користувачі вебзастосунку психологічної підтримки поділяються на кілька різних категорій за рівнем доступу та відповідальності, а саме: анонімні користувачі, автентифіковані користувачі, оператори та адміністратори. Це умовний поділ, оскільки система використовує гнучку модель дозволів, яка дозволяє забезпечити гнучкість в налаштуванні та адаптації доступу залежно від потреб застосунку та його користувачів.

Анонімні користувачі мають можливість доступу до чату через різні месенджери, що дозволяє їм звертатися за підтримкою без необхідності реєстрації чи авторизації безпосередньо у системі, забезпечуючи конфіденційність та доступність послуг.

Оператори мають доступ до всіх перелічених вище можливостей комунікації, а також можуть самостійно оцінювати свою зайнятість, що впливає на алгоритм розподілення нових чатів. Оператори мають доступ до внутрішньої бази знань, де зібрана інформація про методики лікування та вправи для релаксації.

Адміністратори мають найвищий рівень доступу та контролю в системі. Вони займаються адмініструванням персоналу, включаючи створення, перегляд, редагування, деактивацію та видалення облікових записів. Крім того, адміністратори відповідають за гранульоване налаштування доступу до всіх функцій системи, що забезпечує гнучке управління правами та безпекою.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				36
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.1. Варіанти використання застосунку анонімними користувачами.

На рисунку 2.1 зображено варіанти використання системи анонімним користувачем. Оскільки метою є адміністративна частина застосунку, анонімні користувачі мають можливість користуватися системою лише за допомогою підтримуваних месенджерів.

Для користувача є різниця у тому, чи є звернення першим, адже оператор обирається за зайнятістю при першому зверненні, а при повторних система намагається обрати вже «знайомого» користувачу оператора. Якщо такого оператора немає онлайн – знову обирається новий за звичайним алгоритмом.

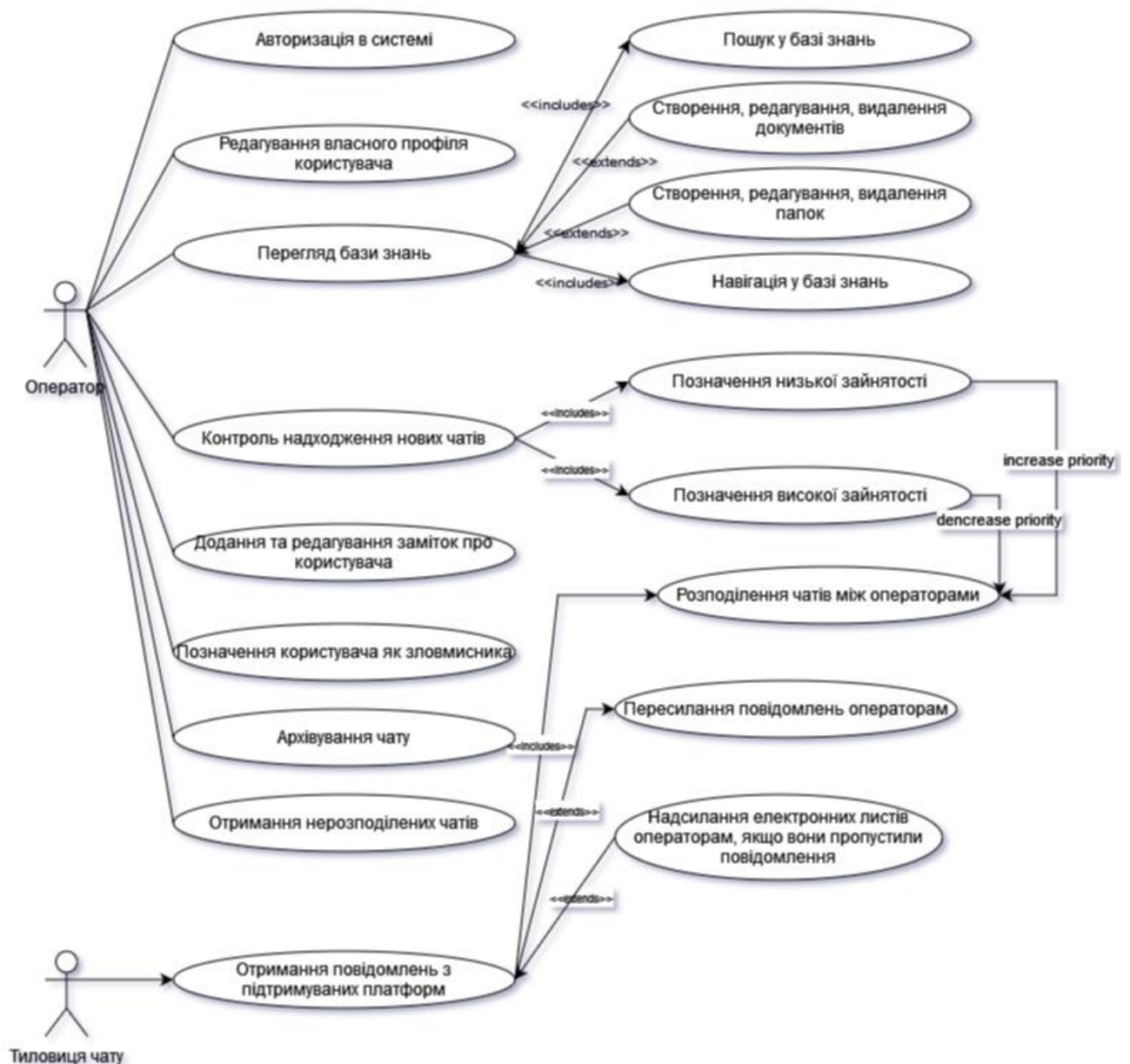


Рисунок 2.2. Варіанти використання застосунку операторами та роутером повідомлень.

На діаграмі варіантів використання, зображених на рисунку 2.2, зосереджено увагу на функціях оператора в системі цифрового обміну повідомленнями. Як центральний актор, оператор має можливість авторизуватися в системі, що є першим кроком для доступу до інших варіантів використання.

Після авторизації, оператор може змінювати та налаштовувати свій профіль. Також оператор має здатність контролювати надходження нових чатів та вносити редагування нотаток про користувачів та має активну роль у моніторингу та управлінні діалогами.

На діаграмі також відображено варіанти використання, пов'язані з визначенням пріоритетності чатів, включаючи позначення чатів як низької або високої важливості, що відображає гнучкість системи у відповіді на змінювані потреби користувачів та операторів.

Центральна функція діаграми – розподілення чатів між операторами – включає в себе механізми контролю пріоритетності операторів. Це дозволяє адаптувати робоче навантаження операторів згідно з їхніми можливостями та поточною зайнятістю. Крім того, оператори можуть отримувати повідомлення, переспрямовані через застосунок-роутер повідомлень, що діє як автоматизований актор, забезпечуючи автоматизацію комунікації між операторами та користувачами, фігуруючи медіумом між платформами що кожен з них використовує.

Варто зазначити, що роутер повідомлень окрім автоматизації доставки повідомлень також відправляє електронні листи за потреби для сповіщення персоналу про пропущені повідомлення.

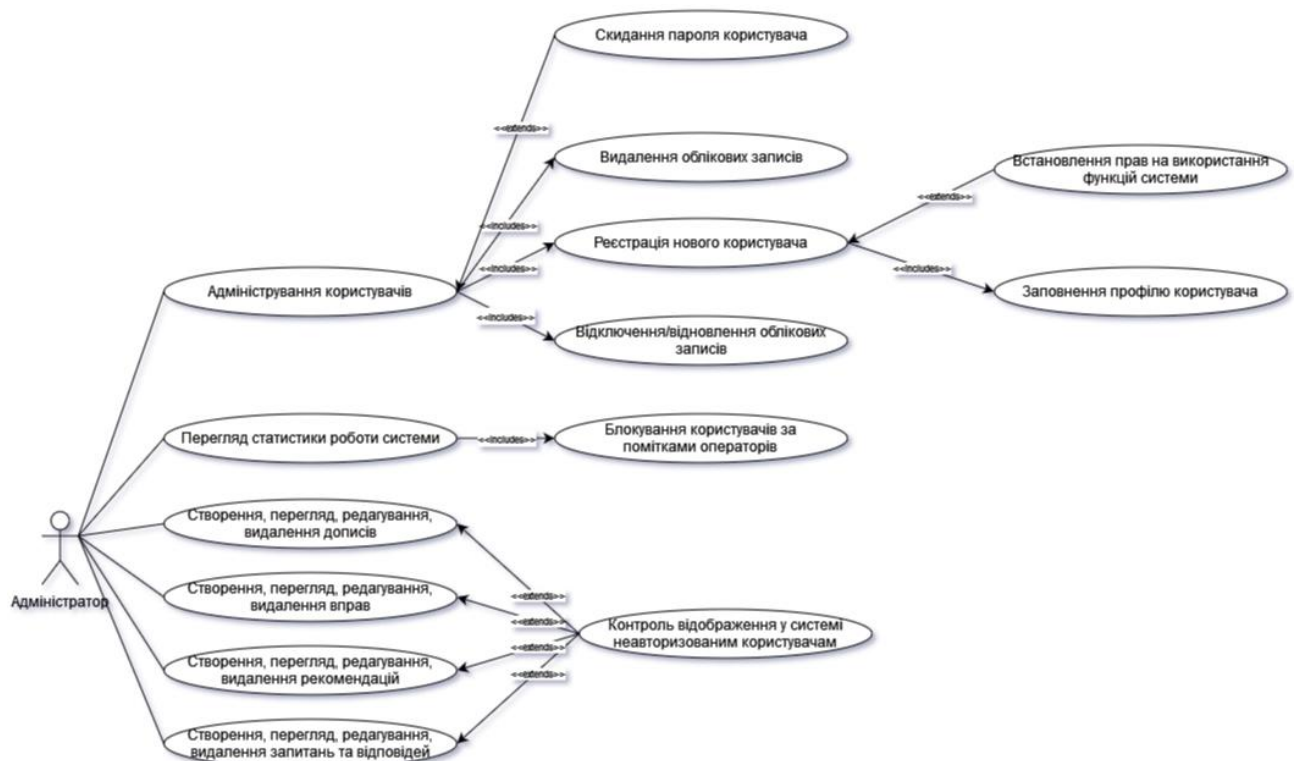


Рисунок 2.3. Варіанти використання застосунку адміністраторами.

В рамках дослідження інтерфейсів адміністративного управління, створено діаграму на рисунку 2.3, яка демонструє можливості взаємодії адміністратора з платформою.

Комплексний підхід до управління профілями користувачів доповнено можливістю блокування користувачів за порушення правил, що забезпечує дотримання етичних та операційних стандартів. Перегляд статистики роботи системи виконується для аналізу загальної ефективності та виявлення потенційних зон для покращення.

Скидання пароля користувача забезпечується через процес, який дозволяє користувачам відновлювати доступ до акаунтів у разі втрати пароля, звернувшись до адміністратора.

Доступ до окремих функцій системи забезпечується системою дозволів. Адміністратор має змогу налаштовувати індивідуальні можливості кожного користувача відповідно до його потреб та ролі у організації, не надаючи зайвих прав, що може скомпрометувати систему, де велика частина інформації не підлягає розголошенню.

Створення, перегляд, редагування, та видалення записів сутностей, що відносяться до інформаційного наповнення застосунку, дозволяє забезпечити актуальність і точність інформації в системі. Зокрема адміністратор може приховати чи опублікувати кожен сутність.

Ці процеси інтегровані в одну систему, що дозволяє адміністратору ефективно керувати аспектами взаємодії користувачів, їхніми правами, та безпекою системи загалом.

Частина варіантів використання видалена з діаграми, щоб запобігти дублюванню діаграми на рисунку 2.2, оскільки адміністратор має доступ до всіх функцій.

Окремо варто зазначити, що адміністратор у даному випадку – користувач з повним доступом до системи, що забезпечується спеціальною роллю. Будь-якому користувачу можна надати довільний набір дозволів на використання функціоналу.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

Зображені варіанти використання наголошують на інтегрованості ролей та процесів у системі, що надає операторам необхідні інструменти для ефективного управління комунікаціями та розподілом робочого навантаження.

2.2 Розробка бази даних системи

У якості системи управління базами даних обрано PostgreSQL, оскільки при первинному аналізі проєкту визначено високу потребу у зв'язках та інших особливостях, що надають лише реляційні бази даних, а саме обрана СУБД є стандартом для усього спектру вебпроєктів [50].

Діаграма на рисунку 2.4 відображає усі таблиці в базі даних та зв'язки між ними. Діаграму згенеровано за допомогою PgAdmin, що є частиною інструментарію екосистеми PostgreSQL [51].

Варто зазначити, що на діаграмі відображено також і таблиці (Role, Tag та похідні від них), що не повністю задіяні у представленій версії застосунку. Ці сутності – частина майбутнього оновлення, вони не розглядаються як частина системи у її поточній версії, однак вже інтегровані в тиловицю застосунку, тому не виключені з деяких діаграм та вихідного коду застосунку).

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

Опис моделей Prisma, що відповідають таблицям в базі даних

Модель	Опис
User	Загальна модель користувача, яка включає базові дані (ім'я, опис, електронну адресу), статус онлайн, ролі, сесії, облікові записи та зв'язки з чатами та контентом.
VerificationToken	Використовується для зберігання токенів верифікації, що містять ідентифікатор, унікальний токен і дату закінчення.
Account	Модель для зберігання інформації про облікові записи соціальних мереж або інших зовнішніх провайдерів. Включає токени, типи акаунтів та ідентифікатори.
Session	Зберігає дані про сесії користувачів, такі як токен сесії, ідентифікатор користувача та термін дії сесії.
Permission	Визначає дозволи, які можуть бути призначені користувачам або ролям.
Chat	Відображає чати між користувачами та персоналом, зберігаючи пов'язані повідомлення та дати створення.
Message	Модель для зберігання тексту повідомлень у чаті, включаючи інформацію про те, чи було повідомлення від користувача.
ArchivedChat	Зберігає інформацію про архівовані чати, включаючи час завершення та пов'язані повідомлення.
ArchivedMessage	Модель для зберігання архівованих повідомлень, що належать до архівованих чатів.
Recommendation	Модель рекомендацій, що може містити заголовки, описи та зображення.

Продовження таблиці 2.1

Post	Публікації користувачів, які можуть містити опис, зображення та теги для категоризації.
Exercise	Відображає вправи, які включають назву, опис, теги та кроки вправ.
ExerciseStep	Описує окремі кроки в рамках вправи, включаючи заголовок, опис та зображення.
Document	Модель для документів, які можуть бути частиною папок документів і містити теги для категоризації.
DocumentFolder	Папки для організації документів, можуть містити батьківські папки та дочірні папки.
QandA	Зберігає запитання та відповіді для розділу Q&A, включаючи інформацію про автора.
gptMessage	Відповідає за облік звернень до OpenAI API. Використовується для відображення попередніх звернень та заборони повторних.

Надалі усі сутності бази даних, описані у таблиці 2.1 буде детально описано, включно з полями та функціоналом, який вони забезпечують.

Таблиця 2.2

Допоміжні таблиці в базі даних

Таблиця	Опис
_PermissionToUser	Пов'язує дозволи та користувачів
_ExerciseToExerciseStep	Пов'язує вправи та їх кроки

У таблиці 2.2, виділено допоміжні таблиці, що створюються Prisma для зв'язків типу багато-до-багатьох та не мають власних моделей, оскільки створюються «неявним» оголошенням у моделях, які вони пов'язують [52].

В архітектурі бази даних, розробленій за допомогою Prisma для PostgreSQL, представлено декілька основних моделей, зокрема User, Account та Session, які відіграють ключову роль у системі аутентифікації користувачів.

Таблиця 2.3

Опис стовбців таблиці User

Назва	Тип	Опис
id	text	Унікальний ідентифікатор користувача в системі. Рядок, згенерований за алгоритмом Collision-resistant unique identifier (CUID) [53].
name	text	Ім'я користувача або псевдонім.
description	text	Короткий опис або біографія користувача.
email	text	Електронна адреса користувача.
emailVerified	timestamp	Чи підтверджена електронна адреса користувача.
image	text	Посилання на зображення профілю користувача.
notes	text	Примітки адміністратора чи інші важливі замітки про користувача.
password	text	Хеш пароля користувача для аутентифікації.
suspended	boolean	Чи заблоковано користувача в системі.
createdAt	timestamp	Дата та час створення облікового запису користувача.
busyness	integer	Індикатор зайнятості користувача, використовується при розподіленні нових вхідних чатів між операторами.
reportCount	integer	Кількість скарг, поданих на користувача.
isOAuth	boolean	Чи було створено обліковий запис користувача через OAuth аутентифікацію.

Таблиця в базі даних (представлена у таблиці 2.3) User є похідною від моделі, що є центральною для керування ідентифікацією користувачів. Кожен користувач отримає унікальний ідентифікатор, який генерується за

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

специфікацією CUID, що є аналогом Universally unique identifier (UUID), але фокусується на вирішенні проблеми однакових незалежно-згенерованих ідентифікаторів [53].

Ця модель містить поля для збереження інформації про ім'я, електронну адресу, час перевірки електронної адреси, зображення профілю, а також час створення та оновлення запису. Ці поля можуть приймати значення null, що дозволяє гнучко управляти інформацією про користувачів. Додатково, модель включає інформацію про стан користувача, наприклад, чи є він онлайн, а також інші специфічні для домену атрибути, такі як ролі та права доступу, пов'язані з різними видами діяльності на платформі.

Таблиця 2.4

Опис стовбців таблиці Account

Назва	Тип	Опис
id	text	Унікальний ідентифікатор запису в таблиці.
user_id	text	Ідентифікатор користувача, асоційований з цим записом.
type	text	Тип облікового запису, який визначає його функціональність або роль.
provider	text	Провайдер аутентифікації, наприклад Google, Facebook тощо.
provider_account_id	text	Ідентифікатор облікового запису у провайдера.
refresh_token	text	Оновлюваний токен, який використовується для поновлення access_token після його закінчення.
access_token	text	Токен доступу, який використовується для аутентифікації запитів від користувача.
expires_at	integer	Час закінчення дії токена доступу.

Продовження таблиці 2.4

token_type	text	Тип токена, який вказує на метод його використання.
scope	text	Область дозволів, наданих токеном. Використовується лише в підсистемі аутентифікації, для авторизації використовується виключно підсистема дозволів.
id_token	text	ID токен, який використовується для ідентифікації користувача.
session_state	text	Стан сесії, який може використовуватись для контролю стану авторизації.
oauth_token_secret	text	Секретний токен OAuth, використовується в деяких типах авторизації.
oauth_token	text	OAuth токен, який може використовуватися для доступу до API провайдера.

Модель Account (табл. 2.4) визначає облікові записи, асоційовані з користувачами, і є важливою для інтеграції зовнішніх сервісів аутентифікації, таких як соціальні мережі або email-провайдери. Кожен запис містить тип провайдера та ідентифікатор облікового запису провайдера, а також може включати токени доступу, терміни дії токенів та іншу інформацію, що стосується OAuth аутентифікації. Відносини між користувачем та його обліковими записами забезпечуються через зв'язки на рівні бази даних, що дозволяє ефективно управляти каскадним читанням, редагуванням та видаленням даних.

Таблиця 2.5

Опис стовбців таблиці Session

Назва	Тип	Опис
id	text	Зберігає унікальний ідентифікатор для кожної сесії.

Продовження таблиці 2.5

session_token	text	Токен сесії, який використовується для верифікації користувача під час запитів.
user_id	text	Ідентифікатор користувача, якому належить сесія.
expires	timestamp	Час закінчення сесії. Після закінчення часу сесія вважається недійсною.
createdAt	timestamp	Час створення запису сесії в базі даних.

Модель Session (табл. 2.5) відповідає за управління сесіями користувачів. Сутність є критично важливою для підтримки безпеки та стійкості як системи, так і користувацького досвіду. Кожна сесія має унікальний токен сесії та мітку часу закінчення дії. Ця модель дозволяє слідкувати за активністю користувача та їх автентифікаційним статусом на сервері. Також модель зв'язується з моделлю User, що дозволяє відстежувати всі сесії, пов'язані з конкретним користувачем та керувати ними у разі потреби.

Таблиця 2.6

Опис стовбців таблиці VerificationToken

Назва	Тип	Опис
id	integer	Ідентифікатор токена.
identifier	text	Ідентифікатор сутності, що верифікується (користувача)
token	text	Унікальний токен
expires	timestamp	Дата та час закінчення терміну дії токена

Модель VerificationToken (табл. 2.6) призначена для гарантування безпеки та верифікації нових користувачів, використовується для управління тимчасовими токенами, які мають ключове значення під час процесів автентифікації, таких як зміна пароля або перевірка електронної пошти. Ця модель містить унікальний ідентифікатор, ідентифікатор, що пов'язує токен з певним аспектом користувацького облікового запису, сам токен, який є унікальним і забезпечує одноразовий доступ до певних функцій системи, та дату

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

закінчення дії токена, що дозволяє обмежити часовий проміжок його дійсності. Використання унікальності пари ідентифікатор-токен забезпечує додатковий рівень захисту, мінімізуючи можливість несанкціонованого використання токенів та доступу зловмисників до облікових записів користувачів.

Кожна з моделей Session, Account, User та VerificationToken спроектована з урахуванням вимог сучасних систем аутентифікації та управління користувацькими даними, надаючи можливість безпечно та ефективно керувати доступом до ресурсів та послуг.

У сучасних системах управління доступом ідентифікація і авторизація користувачів відіграють ключову роль у захисті ресурсів та даних. Розглянуті сутності досконально виконують задачу аутентифікації користувача, однак в умовах, де чутливі дані користувачів постійно курсують системою, необхідною є підсистема гнучкої та гранульованої авторизації. Для виконання задачі обрано підхід дозволів на кожну дію в системі. В контексті спроектованої архітектури застосунку, модель «Permission» є фундаментальною для реалізації таких механізмів.

Таблиця 2.7

Опис стовбців таблиці Permission

Назва	Тип	Опис
id	integer	Ідентифікатор запису, який зазвичай використовується для унікального визначення кожного запису в таблиці.
title	text	Заголовок запису, який зазвичай використовується для короткого опису змісту запису або як назва статті.

Модель «Permission» (табл. 2.7) представляє дозволи, які можуть бути призначені користувачам. Дозвіл характеризується унікальним ідентифікатором «id» та назвою «title», яка вказує на конкретну дію або набір дій, що дозволені. Наприклад, дозвіл може визначати можливість читання, запису або модифікації даних.

При розгортанні системи набір можливих дозволів перевіряється на актуальність, застарілі дозволи видаляються, а нові додаються, та якщо немає дозволів, що потрібно додати чи видалити, в базі даних змін не проводиться. Набір ролей визначається назвами роутерів, а також типами та назвами процедур, створених за допомогою tRPC та що включають проміжне програмне забезпечення, що перевіряє наявність відповідного дозволу [34] [54].

Таблиця 2.8

Опис стовбців таблиці Chat

Назва	Тип	Опис
id	integer	Ідентифікатор чату.
createdAt	timestamp	Дата і час створення запису.
userId	text	Ідентифікатор користувача, який почав чат.
personnelId	text	Ідентифікатор персоналу, який отримав чат.

В рамках структуризації взаємодій між користувачами, система використовує кілька ключових моделей для управління чатами та повідомленнями. Модель «Chat» (табл. 2.8) відіграє центральну роль у встановленні комунікаційного каналу між двома користувачами: ініціатором (записано як «user», «userId») та приймаючою стороною (записано як «personnel», «personnelId»). Ця модель включає унікальний ідентифікатор чату, дату створення, ідентифікатори учасників і зберігає посилання на повідомлення, якими учасники обмінюються в рамках даного чату.

Таблиця 2.9

Опис стовбців таблиці Message та ArchivedMessage

Назва	Тип	Опис
id	integer	Унікальний ідентифікатор повідомлення.
createdAt	timestamp	Дата та час створення повідомлення.
text	text	Текст повідомлення.
chatId	integer	Ідентифікатор чату, до якого належить повідомлення.

isFromUser	boolean	Позначка, що вказує, чи було повідомлення відправлене користувачем.
------------	---------	---

Модель «Message» (табл. 2.9) служить для збереження інформації про конкретне повідомлення, яке включає текст повідомлення, час створення та вказівку, чи було воно відправлено користувачем чи персоналом. Кожне повідомлення асоційоване з конкретним чатом.

Паралельно до моделей активного чату та повідомлень у ньому, існують моделі для їх архівування: «ArchivedChat» і «ArchivedMessage». Модель «ArchivedChat» зберігає інформацію про чати, які були завершені, та має додаткове поле у порівнянні з основною моделлю – час завершення спілкування. Таблиця «ArchivedMessage» має ідентичну до «Message» структуру, але є призначеною для збереження інформації про повідомлення вже в архівованому чаті.

Отримана архітектура дозволяє ефективно управляти «живими» та архівними даними без змішування контекстів, забезпечуючи цілісність і легкість доступу до історичних даних, коли це необхідно. Також такий підхід дозволить майже «безболісно» масштабувати систему у майбутньому, розділивши базу даних на сервери за таблицями [55].

Таблиця 2.10

Опис стовбців таблиці gptMessage

Назва	Тип	Опис
id	integer	Унікальний ідентифікатор повідомлення.
createdAt	timestamp	Час і дата створення повідомлення.
text	text	Текст повідомлення.
chatId	integer	Ідентифікатор чату, до якого належить повідомлення.
messageId	boolean	Ідентифікатор повідомлення, що використовується для відстеження в межах чату.

Модель gptMessage (табл. 2.10) в Prisma забезпечує облік звернень до OpenAI API [56]. Ця модель дозволяє відображати історію запитів і забороняти повторні звернення. Ідентифікатор кожного запису генерується автоматично і зберігається в полі id. Поле createdAt визначає час створення запису та автоматично встановлюється в поточний момент створення. Звернення містить текст запиту в полі text. Кожне звернення асоціюється з конкретним чатом через поле chat, яке використовує зовнішній ключ chatId і забезпечує каскадне видалення. Опціонально, звернення може мати зв'язок з іншим повідомленням через поле message, яке використовує зовнішній ключ messageId та дозволяє встановити належність до попереднього повідомлення, з можливістю його видалення без видалення повідомлення.

Таблиця 2.11

Опис стовбців таблиці Document

Назва	Тип	Опис
id	text	Цілочисловий унікальний ідентифікатор документа.
title	text	Назва документа.
description	text	Текстовий опис змісту документа.
authorId	text	Цілочисловий ідентифікатор автора документа.
createdAt	timestamp	Дата і час створення документа.
updatedAt	timestamp	Дата і час останнього оновлення документа.
parentId	text	Цілочисловий ідентифікатор батьківського документа, якщо такий існує.

Модель Document (табл. 2.11) представляє собою одиницю інформації у внутрішній системі знань, призначену для допомоги операторам чату в пошуку та зберіганні відомостей про психологічне здоров'я. Кожен документ має унікальний ідентифікатор (id) і може бути пов'язаним з папкою документів через поле parentId. Документ включає назву (title) та наповнення (description). Документ має поля, що зв'язують його з автором (authorId і author), та мітки часу створення та останнього оновлення документа.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

Опис стовбців таблиці DocumentFolder

Назва	Тип	Опис
id	text	Унікальний ідентифікатор папки.
title	text	Назва папки, яка може містити інформацію про зміст або призначення папки.
parentId	text	Ідентифікатор батьківської папки, який вказує на ієрархічну структуру папок.
authorId	text	Ідентифікатор автора папки, який створив цю папку.
createdAt	timestamp	Дата і час створення папки.
updatedAt	timestamp	Дата і час останнього оновлення папки.

Модель DocumentFolder (табл. 2.12) описує папку, яка може містити документи та інші папки, створюючи ієрархічну структуру для кращої організації інформації. Кожна папка також має унікальний ідентифікатор (id) і заголовок (title). Папки можуть включати масив документів (documents), з якими вони пов'язані за допомогою зв'язку один-до-багатьох, що забезпечує зовнішній ключ на стороні документів. Подібно до моделі Document, папка може мати поле parentId, яке пов'язує її з батьківською папкою для створення вкладеної структури. Папки також мають поля для визначення автора (authorId і author) і мітки часу, які позначають коли папка була створена та коли вона була оновлена.

Опис стовбців таблиці Post

Назва	Тип	Опис
id	text	CUID поста.
title	text	Заголовок поста.
description	text	Опис поста.
image	text	Шлях до зображення поста.
published	boolean	Чи опубліковано пост.

authorId	text	ID автора поста.
createdAt	timestamp	Дата та час створення поста.
updatedAt	timestamp	Дата та час оновлення поста.

Модель Post (табл. 2.13) описує публікації, які містять заголовок, опис та опціонально зображення. Публікації зазвичай пов'язані з автором. Поле published дозволяє контролювати видимість публікації на випадок відкладеної публікації чи при виявленні неточної інформації у публікації. Час створення та останнього оновлення також відстежуються.

Таблиця 2.14

Опис стовбців таблиці Recommendation

Назва	Тип	Опис
id	text	Унікальний ідентифікатор рекомендації.
title	text	Назва рекомендації.
description	text	Детальний опис рекомендації.
published	boolean	Чи опублікована рекомендація.
authorId	text	Ідентифікатор автора рекомендації.
createdAt	timestamp	Дата і час створення запису.
updatedAt	timestamp	Дата і час останнього оновлення запису.

Модель Recommendation (табл. 2.14) визначає рекомендації, що відображаються пліч-о-пліч з дописами. Кожна рекомендація має унікальний ідентифікатор, назву, опис, і може бути пов'язана з автором через зв'язок з таблицею користувачів. Властивість published вказує, чи є рекомендація доступною для публіки. Дати створення та останнього оновлення відстежуються автоматично, що дозволяє забезпечити актуальність інформації. Рекомендації відображаються користувачам випадковим чином.

Таблиця 2.15

Опис стовбців таблиці Exercise

Назва	Тип	Опис
id	text	Ціле число, що унікально ідентифікує кожне вправу в базі даних.
title	text	Назва вправи.
description	text	Детальний опис вправи.
image	text	Зображення, що ілюструє вправу.
published	boolean	Булевий індикатор, який показує, чи опублікована вправа.
authorId	text	Ідентифікатор автора вправи.
createdAt	timestamp	Дата і час створення запису.
updatedAt	timestamp	Дата і час останнього оновлення запису.

Модель Exercise (табл. 2.15) призначена для управління вправами на сайті. Кожна вправа включає заголовок, опис та можливо зображення. Вправи можуть бути теговані та містити кроки, що представлені моделлю ExerciseStep. Аналогічно іншим моделям, вправа має поля для відстеження авторства, публікації та часових відміток створення та оновлення.

Таблиця 2.16

Опис стовбців таблиці ExerciseStep

Назва	Тип	Опис
id	text	ідентифікує кожен крок у вправі.
title	text	Назва кроку вправи.
description	text	Детальний опис кроку вправи.
image	text	Зображення, що ілюструє крок.
timeSeconds	integer	Час у секундах, необхідний для виконання кроку.
authorId	text	ID автора, який створив крок.
createdAt	timestamp	Час створення запису кроку вправи.

updatedAt	timestamp	Час останнього оновлення запису кроку вправи.
-----------	-----------	---

Модель ExerciseStep (табл. 2.16) використовується для деталізації кроків у вправах. Кожен крок містить заголовок, опис, опціонально зображення, і може містити тривалість виконання у секундах, що дозволяє користувачу зберігати ритм чи навпаки затримувати його на кроках як-от глибоке, спокійне дихання. Кроки можуть бути пов'язані з конкретними вправами, а також містять інформацію про автора та час створення.

Таблиця 2.17

Опис стовбців таблиці QandA

Назва	Тип	Опис
id	integer	Ціле число, що унікально ідентифікує кожне запитання та відповідь в базі даних.
question	text	Текст запитання, поставленого користувачем.
answer	text	Текст відповіді на запитання.
authorEmail	text	Електронна пошта автора, який створив запис.
authorName	text	Ім'я автора, який створив запис.
published	boolean	Логічне значення, що показує, чи опубліковано запис.
createdAt	timestamp	Часова мітка, що позначає коли запис було створено.
updatedAt	timestamp	Часова мітка, що позначає коли запис було оновлено.

Модель QandA (табл. 2.17) використовується для зберігання питання та слідує за ним відповіді від психологів та адміністрації сайту психологічної підтримки. Кожен запис містить унікальний ідентифікатор id, який автоматично збільшується з кожним новим записом. Поле question зберігає текст питання, а answer — текст відповіді, який може бути незаповненим, поки відповідь не буде надана. Додатково зберігаються authorEmail та authorName для інформування користувача щодо відповіді на його запитання за наданою адресою електронної пошти. Поле published визначає, чи є питання публічно доступним. Час

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

створення і останнього оновлення запису фіксується в полях `createdAt` та `updatedAt` відповідно.

Результатом проєктування стала база даних, що відповідає встановленим вимогам до системи та потребам спроектованої системи адміністрування застосунку психологічної підтримки. Загалом створено 21 таблицю, що відображають сутності у самій системі та дозволяють побудувати повноцінну систему на їх основі уникаючи компромісів у функціоналі, безпеці чи надійності.

2.3 Проєктування та реалізація алгоритмів роботи системи

Спроектовано та реалізовано ключові алгоритми у системі, що відкрило можливість побудувати систему адміністрування. Для визначення важливих частин застосунку, що потребують першочергового виконання, створено діаграми дій системи для унікальних компонентів для візуалізації необхідного результату у зрозумілій формі. На діаграмах будуть відображені високорівневі дії користувачів, що мають різні доступи, цілі та можливості у системі.

Варто зазначити, що будь-яке розподілення за ролями умовне, оскільки система дозволяє гнучко налаштувати доступи будь-якого аутентифікованого користувача. Доступ анонімних користувачів можна повністю обмежити, гранульоване налаштування для таких акторів неможливе.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

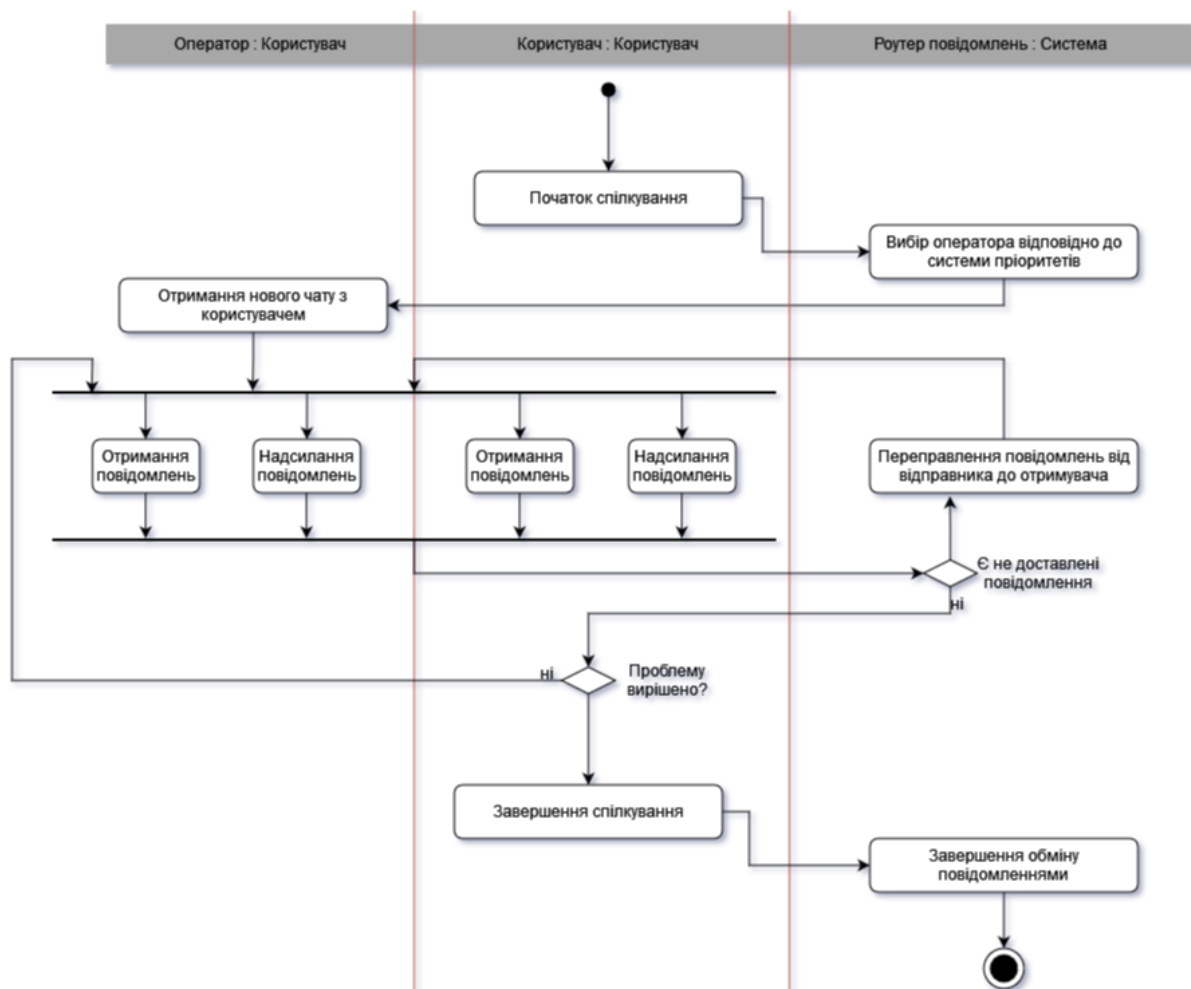


Рисунок 2.5. Діаграма дій системи для операторів та користувачів у чаті.

На рисунку 2.5 зображено дії, які доступні користувачам умовно розділеним на операторів та анонімних користувачів у чаті. Це включає початок спілкування, обмін повідомлення поки проблему не буде вирішено та завершення чату.

Забезпеченням успішності листування на діаграмі займається автоматизований актор, який при зверненні користувача обирає йому оператора та слугує проксі для швидкого обміну повідомленнями між користувачами месенджерів та операторами, які використовують для спілкування саму платформу.

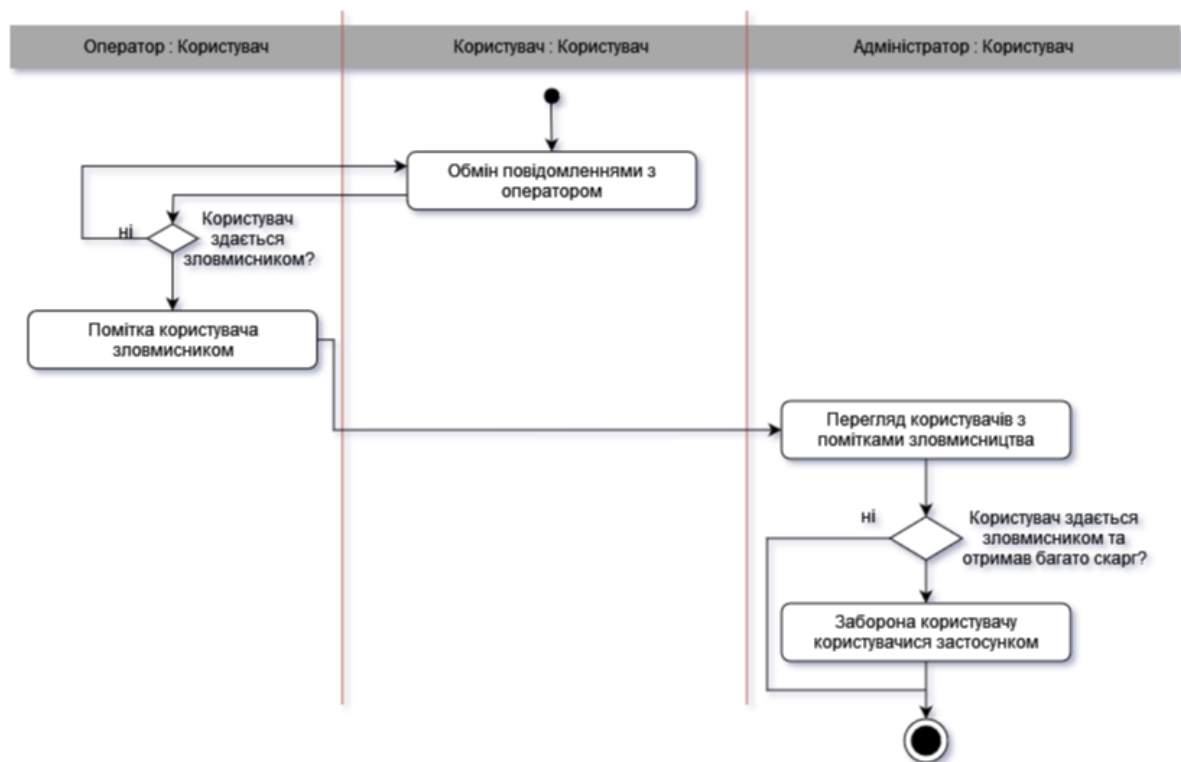


Рисунок 2.6. Діаграма дій системи для операторів при неприпустимій поведінці користувача.

На рисунку 2.6 зображено можливі дії оператора та адміністратора у випадку, коли дії користувача у чаті є неприпустимими. Це включає обмін повідомленнями та створення скарги на користувача. Адміністратор у асинхронному режимі переглядає активних користувачів та кількість скарг на них від операторів та може моментально заблокувати користувача від взаємодії із системою.

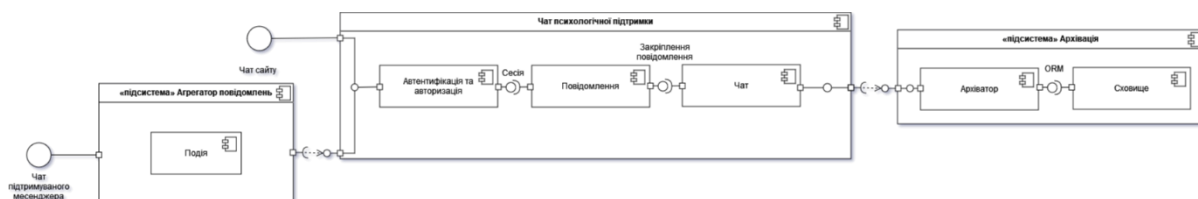


Рисунок 2.7. Діаграма компонентів чату психологічної підтримки.

На рисунку 2.7 зображено діаграму компонентів чату у системі. На рисунку виділено три підсистеми, кожна з яких відповідальна за частину життєвого циклу чату, це – створення чату, надсилання повідомлень та архівація як останній крок.

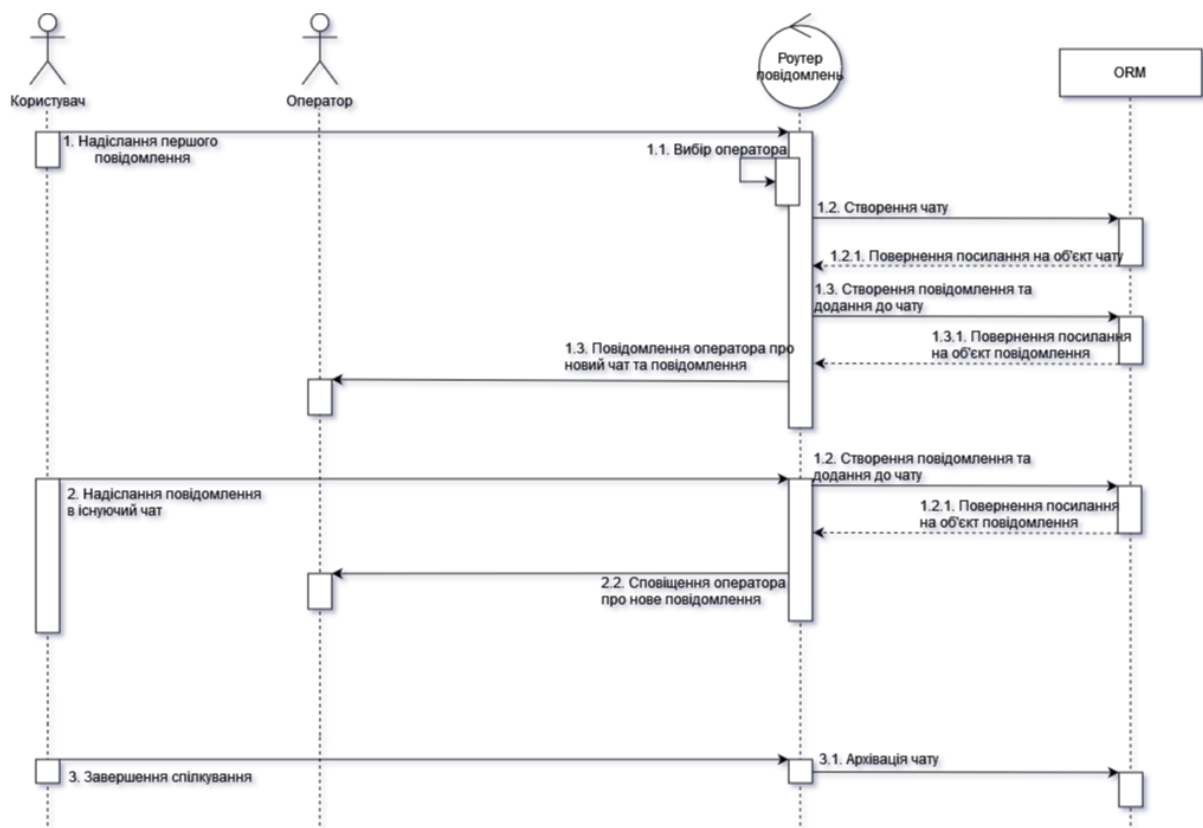


Рисунок 2.8. Діаграма послідовності листування у чаті.

На рисунку 2.8 зображено послідовність дій акторів у системі при листуванні. Навідміну від рисунка 2.5, на даній діаграмі зображені процеси та залежності у більш технічній формі. Відображені взаємодії з ОРВ, яка є абстракцією бази даних. Виклики включають створення чату, створення повідомлень та архівація чату. Викладені процеси на стороні роутера повідомлень, що включають вибір оператора, надсилання та отримання інформації з бази даних та архівація чату.

Надалі розглянуто реалізацію архітектури, структури та алгоритмів у системі. Архітектурою основного застосунку обрано моноліт, у якому більшість запитів виконується у вигляді Remote procedure call (RPC), що реалізовано бібліотекою tRPC. У системі використовуються також і POST запити, однак це виключення для специфічного функціоналу.

Для імплементації та структуризації застосунку згідно з найкращими практиками, використано приклади використання з офіційної документації для полегшення початку проєктування та реалізації. Згідно з документацією, процедури повинні бути розділені на колекції, які називаються роутерами. В

застосунку прийнято рішення закріпити роутери за сутностями, для взаємодії з якими призначені процедури в них. Після визначення роутерів, вони об'єднуються в роутер застосунку.

Таблиця 2.18

Опис роутерів

Назва	Опис
chat	Процедури, що використовуються на сторінці чату. Включає в себе перегляд, отримання, архівацію чатів та отримку допомоги від ІІІ.
user	Процедури, що пов'язані з користувачами. Включає в себе перегляд, отримання, редагування, видалення користувачів, оновлення пароля та отримання актуального токена доступу.
permission	Процедури, що пов'язані з управлінням доступом до системи. В першу чергу надає можливість встановлювати доступи користувачів.
stats	Надає перелік статистичних даних щодо роботи чату, включно з метриками зайнятості операторів.
documentFolder	Надає доступ до створення, перегляду, редагування та видалення директорій в базі знань. Дозволяє отримувати та шукати наповнення папок.
document	Процедури, призначені для управління документами (створення, перегляд, редагування, видалення).
recommendation	Відповідає за адміністрування рекомендацій
post	Відповідає за адміністрування дописів
exercise	Відповідає за адміністрування вправ
qanda	Відповідає за адміністрування модуля питань та відповідей

У таблиці 2.18 наведено повний перелік реалізованих у системі роутерів з описом функціоналу процедур, які вони в себе включають.

Розглянемо реалізований алгоритм забезпечення авторизації у системі. Значною мірою можливість інтегрування такої важливої та складної системи у великий застосунок забезпечується створеною на основі tRPC структурою застосунку. Кожну процедуру можна налаштувати окремо, додаючи дії, що будуть виконані перед виконанням запиту або мета-дані, що дозволяють виконувати дії взагалі поза системою.

На основі згаданих особливостей побудовано набір процедур, що включають в себе від публічного рівня доступу до приватного, доступного лише за наявності дозволу на виконання конкретної процедури. Усі проміжні варіанти як-от редагування лише власних дописів за умови аутентифікації, але можливість редагувати будь-чий допис за наявності відповідного дозволу.

Для забезпечення виконання згаданих умов та автоматичного генерування дозволів необхідно налаштувати контекст tRPC за допомогою наступного коду:

```
const t = initTRPC
  .context<typeof createTRPCContext>()
  .meta<Meta>()
  .create({
    transformer: superjson,
    errorFormatter({ shape, error }) {
      return {
        ...shape,
        data: {
          ...shape.data,
          zodError:
            error.cause instanceof ZodError
              ? error.cause.flatten()
              : null,
        },
      },
    },
  });
  defaultMeta: {
    hasPermissionProtection: false,
    hasSpaProtection: false,
    hasMultilevelProtection: false,
  },
});
```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

Варто звернути увагу на мета-дані, на яких значною мірою покладається система дозволів для забезпечення автоматизації процесів і зниження навантаження на розробника щодо управління доступом.

Основні процедури мають наступний вигляд:

```
export const protectedProcedure = publicProcedure.use(({ ctx, next
}) => {
  if (!ctx.session || !ctx.session.user) {
    throw new TRPCError({ code: 'UNAUTHORIZED' });
  }
  return next({
    ctx: {
      session: { ...ctx.session },
    },
  });
});

export const permissionProcedure = protectedProcedure
  .use(({ ctx, next }) => {
    const accessType = getAccessType(
      ctx.session.user.permissions,
      ctx.entity,
      ctx.action,
    );
    if (accessType === AccessType.NONE) {
      throw new TRPCError({ code: 'UNAUTHORIZED' });
    }
    return next({
      ctx: {
        isFullAccess: accessType === AccessType.ALL,
      },
    });
  })
  .meta({
    hasPermissionProtection: true,
  });
```

Додаткові процедури мають наступний код:

```
export const publicPermissionProcedure = publicProcedure.meta({
  hasPermissionProtection: true,
});

export const publicMultilevelPermissionProcedure =
  publicPermissionProcedure.meta({
    hasMultilevelProtection: true,
  });

export const publicSpaProcedure = publicProcedure.meta({
  hasSpaProtection: true,
```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

});

export const publicMultilevelSpaProcedure =
  publicMultilevelPermissionProcedure.meta({
    hasSpaProtection: true,
  });

export const multilevelPermissionProcedure =
  permissionProcedure.meta({
    hasMultilevelProtection: true,
  });

// Single permission access
export const spaProcedure = permissionProcedure.meta({
  hasSpaProtection: true,
});

export const multilevelSpaProcedure =
  multilevelPermissionProcedure.meta({
    hasSpaProtection: true,
  });

```

Зазначені примітиви процедур виконують функції захисту коду від неавторизованого виконання та декларації факту використання ними дозволів. Залишається нерозглянутою частина автоматичного створення переліку дозволів та адміністрування їх в базі даних. За цю частину відповідає скрипт, що запускається під час розгортання застосунку.

Згаданий скрипт поділяється на дві частини. Перша частина – генерація набору актуальних дозволів, вона має наступний код:

```

function getRelevantPermissions() {
  const result: Record<string, { action: string; multilevel:
boolean }[]> =
    {};

  interface AppRouterStructure {
    [key: string]: {
      [key: string]: {
        _def?: {
          meta?: Meta;
        };
      };
    };
  }

  const typedAppRouter = appRouter as unknown as
AppRouterStructure;
  for (const entity in typedAppRouter) {

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (entity.startsWith('_')) continue;
        const actions = [];
        let hasSpaProtection = false;
        for (const action in typedAppRouter[entity]) {
            const meta =
typedAppRouter[entity][action]?._def?.meta;
            if (meta?.hasSpaProtection) hasSpaProtection = true;
            else if (meta?.hasPermissionProtection) {
                actions.push({
                    action,
                    multilevel: meta?.hasMultilevelProtection ??
false,
                });
            }
        }
        if (actions.length > 0 || hasSpaProtection) {
            result[entity] = actions;
        }
    }

    const permissions = new Set<string>();
    for (const entity in result) {
        let hasMultilevel = false;
        for (let x = 0; x < result[entity].length; x++) {
            const { action, multilevel } = result[entity][x];

            permissions.add(`${entity}:${action}`);

            if (multilevel) {
                permissions.add(`${entity}:${action}:*`);
                hasMultilevel = true;
            }
        }
        permissions.add(`${entity}:*`);
        if (hasMultilevel) {
            permissions.add(`${entity}:*:*)`);
        }
    }

    return Array.from(permissions.add('*:*:*'));
}

```

Функція перебирає усі роутери та процедури у них в пошуках тих, що використовують систему дозволів, після чого на основі мета-даних, розглянутих вище, генерує відповідні дозволи, необхідні знайденим процедурам.

Друга частина скрипта – видалення застарілих дозволів з бази даних якщо такі є та створення нових, якщо такі є. Код має наступний вигляд:

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export async function updatePermissions() {
  console.log(
    'Warning: renamed permissions are treated as deleted and
added again, so any users with the old permissions will lose
them.',
  );

  const relevantPermissions = getRelevantPermissions();

  console.log(relevantPermissions);

  logWithDivider('Relevant permissions:',
relevantPermissions.length);

  const currentPermissionRows = await db.permission.findMany({
    select: {
      title: true,
    },
  });

  const currentPermissions = currentPermissionRows.map((item) =>
item.title);

  logWithDivider('Current permissions:',
currentPermissions.length);

  const itemsToAdd = relevantPermissions.filter(
    (item) => !currentPermissions.includes(item),
  );

  const itemsToRemove = currentPermissions.filter(
    (item) => !relevantPermissions.includes(item),
  );

  logWithDivider('Items to add:', itemsToAdd.length);
  logWithDivider('Items to remove:', itemsToRemove.length);

  let r = await db.permission.createMany({
    data: itemsToAdd.map((item) => ({ title: item })),
    skipDuplicates: true,
  });

  logWithDivider('Added permissions:', r.count);

  r = await db.permission.deleteMany({
    where: {
      title: {
        in: itemsToRemove,
      },
    },
  });
}

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    logWithDivider('Removed permissions:', r.count);
}

```

Важливу роль у забезпеченні дотримання обмежень встановлених дозволами виконують наступні функції:

```

export const getAccessType = (
  userPermissions: string[],
  entity: string,
  action: string,
) => {
  const specificPermission = `${entity}:${action}`;
  const generalPermission = `${entity}:*`;
  // const actionPermission = `*:${action}`; // This is not used
  in the current implementation
  const allOwnPermission = '*:*';
  const allPermission = '*:*:*';

  if (
    userPermissions.includes(`${specificPermission}:*`) ||
    userPermissions.includes(`${generalPermission}:*`) ||
    // userPermissions.includes(`${actionPermission}:*`) ||
    userPermissions.includes(allPermission)
  ) {
    return AccessType.ALL;
  }

  if (
    userPermissions.includes(specificPermission) ||
    userPermissions.includes(generalPermission) ||
    // userPermissions.includes(actionPermission) ||
    userPermissions.includes(allOwnPermission)
  ) {
    return AccessType.OWN;
  }

  return AccessType.NONE;
};

export const hasAccess = (
  userPermissions: string[],
  entity: string,
  action: string,
) => {
  return getAccessType(userPermissions, entity, action) !==
  AccessType.NONE;
};

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

Вони відповідають за перевірку наявності у користувача відповідних дозволів на виконання певної дії над даним ресурсом та класифікацію доступу на повний, частковий та відсутній.

У зв'язку з неможливістю ефективно формувати складні запити до бази даних використовуючи обрані ОРВ, прийнято рішення комплексну логіку, пов'язану з даними, реалізувати на стороні бази даних за допомогою Structured Query Language (SQL) функцій та запитів. Реалізовано їх у роутері повідомлень та використано за допомогою функцій SQLAlchemy, які підвищують безпеку запитів, прибирають ризики пов'язані з використанням вводу користувача у запитах санітизуючи ввід. ОРВ також санітизує та структурує результати запитів.

Як додатковий рівень покращення досвіду розробки використання зазначеного методу надає можливість утриматися від мануальних змін в базі даних, надаючи можливість легко розгорнути проєкт на новій базі даних. Для доповнення зазначеного також створено сценарії, що адмініструють базу даних зі сторони головного застосунку.

В першу чергу важливим є процедура, що створюється в базі даних для можливості доступу з обох застосунків. Вона об'єднує 4 основних метрики та визначає об'єктивну зайнятість оператора, що гарантує збалансоване розподілення чатів. Код створення функції:

```
create_get_personnel_stats_function = text("""
DROP FUNCTION IF EXISTS get_personnel_stats();
CREATE OR REPLACE FUNCTION get_personnel_stats()
RETURNS TABLE (
    personnelId TEXT,
    name TEXT,
    totalChats BIGINT,
    normalizedChats DOUBLE PRECISION,
    totalMessages NUMERIC,
    normalizedMessages DOUBLE PRECISION,
    averageResponseTimeSeconds NUMERIC,
    normalizedResponseTime DOUBLE PRECISION,
    perceivedBusyness INT,
    normalizedBusyness DOUBLE PRECISION,
    normalizedScore DOUBLE PRECISION
) AS $$
BEGIN
    RETURN QUERY
    WITH Personnel AS (
```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

SELECT DISTINCT u.id AS "personnelId"
FROM "User" u
LEFT JOIN "_PermissionToUser" up ON u.id = up."B"
LEFT JOIN "Permission" p ON up."A" = p.id
LEFT JOIN "_RoleToUser" ur ON u.id = ur."B"
LEFT JOIN "Role" r ON ur."A" = r.id
LEFT JOIN "_PermissionToRole" pr ON r.id = pr."B"
LEFT JOIN "Permission" p2 ON pr."A" = p2.id
WHERE p.title = ANY(ARRAY['chat:*', 'chat:*:*', '*:*', '*:*:*']) OR p2.title = ANY(ARRAY['chat:*', 'chat:*:*', '*:*', '*:*:*'])
),
TotalChats AS (
SELECT
    "personnelId",
    COUNT(*) AS "totalChats"
FROM (
    SELECT "personnelId"
    FROM "Chat"
    WHERE "personnelId" IS NOT NULL AND "createdAt" >=
NOW() - INTERVAL '1 YEAR'
    UNION ALL
    SELECT "personnelId"
    FROM "ArchivedChat"
    WHERE "personnelId" IS NOT NULL AND "endedAt" >=
NOW() - INTERVAL '1 YEAR'
) AS combined_chats
GROUP BY "personnelId"
),
MessagesLastHour AS (
SELECT "personnelId", COALESCE(SUM("messageCount"), 0)
AS "totalMessages"
FROM (
    SELECT "personnelId", COUNT(*) AS "messageCount"
    FROM "Message"
    JOIN "Chat" ON "Message"."chatId" = "Chat"."id"
    WHERE "Message"."createdAt" > NOW() - INTERVAL '1
YEAR'

    GROUP BY "Chat"."personnelId"
    UNION ALL
    SELECT "personnelId", COUNT(*) AS "messageCount"
    FROM "ArchivedMessage"
    JOIN "ArchivedChat" ON "ArchivedMessage"."chatId"
= "ArchivedChat"."id"
    WHERE "ArchivedMessage"."createdAt" > NOW() -
INTERVAL '1 YEAR'
    GROUP BY "ArchivedChat"."personnelId"
) AS messages
GROUP BY "personnelId"
),
ResponseTime AS (
SELECT
    "personnelId",

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        AVG(EXTRACT(EPOCH FROM ("nextMessageTime" -
"createdAt")) AS "averageResponseTimeSeconds"
        FROM (
            SELECT
                "personnelId",
                "createdAt",
                LEAD("createdAt") OVER (PARTITION BY "chatId"
ORDER BY "createdAt") AS "nextMessageTime"
            FROM (
                SELECT "personnelId", "chatId", m."createdAt",
"isFromUser" FROM "Message" m
                JOIN "Chat" c ON "chatId" = c.id
                WHERE m."createdAt" > CURRENT_TIMESTAMP -
INTERVAL '1 YEAR'
                UNION ALL
                SELECT "personnelId", "chatId",
am."createdAt", "isFromUser" FROM "ArchivedMessage" am
                JOIN "ArchivedChat" ac ON "chatId" = ac.id
                WHERE am."createdAt" > CURRENT_TIMESTAMP -
INTERVAL '1 YEAR'
            ) AS messages
            WHERE "isFromUser" = TRUE
        ) AS filtered_messages
        GROUP BY "personnelId"
    ),
    PerceivedBusyness AS (
        SELECT id AS "personnelId", "busyness" AS
"perceivedBusyness"
        FROM "User"
    ),
    MaxValues AS (
        SELECT
            GREATEST(MAX("totalChats"), 1) AS maxChats,
            GREATEST(MAX("totalMessages"), 1) AS maxMessages,
            GREATEST(MAX("averageResponseTimeSeconds"), 1) AS
maxResponseTime,
            GREATEST(MAX("perceivedBusyness"), 1) AS
maxBusyness
        FROM Personnel
        LEFT JOIN TotalChats USING ("personnelId")
        LEFT JOIN MessagesLastHour USING ("personnelId")
        LEFT JOIN ResponseTime USING ("personnelId")
        LEFT JOIN PerceivedBusyness USING ("personnelId")
    )
    SELECT
        p."personnelId",
        u.name,
        COALESCE(t."totalChats", 0) AS "totalChats",
        COALESCE(t."totalChats", 0)::float / mv.maxChats AS
normalizedChats,
        COALESCE(m."totalMessages", 0) AS "totalMessages",
        COALESCE(m."totalMessages", 0)::float / mv.maxMessages
AS normalizedMessages,

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        COALESCE(r."averageResponseTimeSeconds", 0) AS
"averageResponseTimeSeconds",
        COALESCE(r."averageResponseTimeSeconds"::float /
mv.maxResponseTime, 0) AS normalizedResponseTime,
        COALESCE(pb."perceivedBusyness", 0) AS
"perceivedBusyness",
        COALESCE(pb."perceivedBusyness"::float /
mv.maxBusyness, 0) AS normalizedBusyness,
        (COALESCE(t."totalChats"::float / mv.maxChats, 0) +
COALESCE(m."totalMessages"::float / mv.maxMessages, 0) +
        COALESCE(r."averageResponseTimeSeconds"::float /
mv.maxResponseTime, 0) + COALESCE(pb."perceivedBusyness"::float /
mv.maxBusyness, 0)) AS normalizedScore
    FROM Personnel p
    LEFT JOIN "User" u ON p."personnelId" = u."id"
    LEFT JOIN TotalChats t ON p."personnelId" =
t."personnelId"
    LEFT JOIN MessagesLastHour m ON p."personnelId" =
m."personnelId"
    LEFT JOIN ResponseTime r ON p."personnelId" =
r."personnelId"
    LEFT JOIN PerceivedBusyness pb ON p."personnelId" =
pb."personnelId"
    CROSS JOIN MaxValues mv
    ORDER BY normalizedScore ASC;
END;
$$ LANGUAGE plpgsql;
""")

```

Функція об'єднує дані з таблиць користувачів, повідомлень та чатів, визначаючи швидкість відповіді на повідомлення користувачів, кількість нещодавніх повідомлень та чатів, а також оцінку власної зайнятості оператором. Функція нормалізує отримані значення (приводить до значень на проміжку від 0 до 1), що дозволяє об'єктивно визначити відносну зайнятість операторів та зробити найкращий вибір при розподіленні чатів.

Перед використанням функцій, їх треба передати базі даних, за це відповідає наступний код:

```

with engine.connect() as connection:
    for query in register_queries:
        connection.execute(query)
    connection.commit()

```

Для виконання створеної на стороні бази даних функції використовується наступний код:

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

get_personnel_stats_query = text("""
    SELECT personnelId FROM get_personnel_stats() WHERE
    personnelId = ANY(:available_personnel_ids);
""")

def get_least_busy_personnel_id(session: Session,
    available_personnel_ids: list[str]):
    personnel_by_busyness =
    session.execute(get_personnel_stats_query,
        {'available_personnel_ids':
        available_personnel_ids}).scalars().all() or []
    unmentioned_online_personnel =
    list(set(available_personnel_ids) - set(personnel_by_busyness))

    return unmentioned_online_personnel[0] if
    unmentioned_online_personnel else personnel_by_busyness[0]

```

У розглянутому коді створюється запит до створеної функції, йому надається список персоналу онлайн та отримується один оператор що має найменшу зайнятість.

Отже, створено важливі для системи алгоритми, обрані архітектурні та структурні рішення та розглянуто відповідний програмний код.

2.4 Реалізація системи адміністрування застосунку психологічної підтримки

Розглянемо реалізацію роутера користувачів як прикладу, оскільки одним з пріоритетів було збереження консистентності назв та структури роутерів. Вихідний код основних процедур роутера:

```

get: publicMultilevelPermissionProcedure
    .input(StringIdSchema.optional())
    .query(async ({ input, ctx }) => {
        const accessType = getAccessTypeFromContext(ctx);
        if (
            input !== ctx.session?.user?.id &&
            accessType === AccessType.NONE
        ) {
            throw new Error('Access denied');
        }

        const id = input || ctx.session?.user?.id;

        if (!id) {
            throw new Error('User not found');
        }
    })

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return ctx.db.user.findUnique({
            where: {
                id,
            },
            select: getProjection(accessType === AccessType.ALL),
        });
    })),
    getAccessToken: protectedProcedure.query(async ({ ctx }) => {
        return (
            await ctx.db.session.findFirst({
                where: {
                    userId: ctx.session.user.id,
                    expires: {
                        gte: new Date(),
                    },
                },
                select: {
                    sessionToken: true,
                },
            })
        )?.sessionToken;
    }),

    create: permissionProcedure
        .input(CreateUserSchema)
        .mutation(async ({ ctx, input: data }) => {
            const password = randomUUID();

            const user = await ctx.db.user.create({
                data: {
                    ...data,
                    password: await bcrypt.hash(password,
env.SALT_ROUNDS),
                    isOauth: false,
                },
            });

            await sendRegEmail(data.email, data.name, password,
ctx.host);

            return user;
        }),
    delete: permissionProcedure
        .input(StringIdSchema)
        .mutation(async ({ ctx, input: id }) => {
            if (ctx.session.user.id === id) {
                throw new Error('Неможливо видалити власний запис');
            }

            return ctx.db.user.delete({
                where: {
                    id,
                },
            });
        });

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```
    });  
  },
```

Ці процедури виконують створення, перегляд, редагування та видалення операторів. У роутері також наявні інші процедури, які дозволяють більш гнучко отримувати користувачів, оновлювати пароль (як користувачу, так і адміністратору) та блокувати користувача, однак їх код буде викладено у додатку А через великі об'єми.

Бібліотека tRPC у комбінації з Zod дозволяє мінімізувати кількість коду для перевірки запиту та сфокусуватися на основній цілі процедур, не турбуючись про несанкціонований доступ чи помилкові дані. Відсутність кроку генерації коду та повної типізації бібліотек створено досвід розробки, який дозволив збільшити продуктивність при написанні логіки роботи з даними.

Для демонстрації компонентів лицевіці у застосунку обрано відповідальний за відображення оператору його оцінки власної зайнятості, який також дає можливість встановити нове значення. Код компонента:

```
import * as React from 'react';  
import { useSession } from 'next-auth/react';  
import { BusynessEmoji } from '~/utils/types';  
import { api } from '~/utils/api';  
  
export const Busyness = () => {  
  const { data: session, status } = useSession();  
  const { client: apiClient } = api.useUtils();  
  const [busyness, setBusyness] = React.useState<number>(-1);  
  
  React.useEffect(() => {  
    if (status !== 'authenticated') return;  
    setBusyness(session.user.busyness);  
  }, [status]);  
  
  function setBusynessHandler(index: number) {  
    setBusyness(index);  
    void apiClient.chat.setBusyness.mutate(index);  
  }  
  
  return (  
    <div className="flex h-10 justify-between px-4">  
      {Object.keys(BusynessEmoji).map((emoji, index) => (  
        <button  
          key={index}  
          className="rounded-full border-transparent p-2
```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

disabled:border-gray-500 disabled:bg-neutral-200"
      disabled={busyness === index}
      onClick={() => setBusynessHandler(index)}
    >
      {emoji}
    </button>
  )}}
</div>
);
};

```

У компоненті для модифікації зайнятості використано незвичайний спосіб, що включає створення хука для роутера всього застосунку. Зазвичай для виклику процедур використовується метод з наступного компонента:

```

import React from 'react';
import { api, type RouterOutputs } from '~/utils/api';
import { TableCell } from '../../ui/table';
import {
  Popover,
  PopoverTrigger,
  PopoverContent,
} from '~/components/ui/popover';
import { Button } from '../../ui/button';
import { defaultFormatDateTime } from '~/utils/dates';
import { useToast } from '~/components/ui/use-toast';
import Image from 'next/image';
import { MoreHorizontalIcon } from 'lucide-react';

export const Single: React.FC<{
  entity: RouterOutputs['user']['list']['values'][number];
  edit: (arg: string) => void;
  refetch: () => void;
}> = ({ entity, edit, refetch }) => {
  const resetPassword = api.user.resetPassword.useMutation();
  const suspend = api.user.suspend.useMutation();
  const delete_ = api.user.delete.useMutation();
  const { toast } = useToast();

  function successToast(description: string) {
    toast({
      title: 'Успіх',
      description,
    });
  }

  async function resetPasswordHandler() {
    try {
      await resetPassword.mutateAsync(entity.id);
      successToast('Пароль скинуто');
    } catch (e) {

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        console.error(e);
        toast({
            title: 'Помилка',
            description:
                e instanceof Error ? e.message : 'Невідома
помилка',
            variant: 'destructive',
        });
    }
    void refetch();
}

async function suspendHandler() {
    try {
        await suspend.mutateAsync({
            id: entity.id,
            value: !entity.suspended,
        });
        successToast('Статус змінено');
    } catch (e) {
        console.error(e);
        toast({
            title: 'Помилка',
            description:
                e instanceof Error ? e.message : 'Невідома
помилка',
            variant: 'destructive',
        });
    }
    void refetch();
}

async function deleteHandler() {
    try {
        await delete_.mutateAsync(entity.id);
        successToast('Запис видалено');
    } catch (e) {
        console.error(e);
        toast({
            title: 'Помилка',
            description:
                e instanceof Error ? e.message : 'Невідома
помилка',
            variant: 'destructive',
        });
    }
    void refetch();
}

return (
    <>
    <TableCell>
    <Image

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        className="rounded-full"
        src={
          entity.image ? entity.image :
'/images/placeholder.svg'
        }
        alt={entity.name ?? 'Аватар'}
        width={32}
        height={32}
      />
    </TableCell>
    <TableCell>{entity.id}</TableCell>
    <TableCell>{entity.email ?? '📧'}</TableCell>
    <TableCell>{entity.name ?? '👤'}</TableCell>

    <TableCell>{defaultFormatDateTime(entity.createdAt)}</TableCell>

    <TableCell>{defaultFormatDateTime(entity.updatedAt)}</TableCell>
    <TableCell>{entity.reportCount}</TableCell>
    <TableCell>{entity.suspended ? '🛑' :
'✅'}</TableCell>
    <TableCell className="text-right">
      <Popover>
        <PopoverTrigger asChild>
          <Button size="icon" variant="ghost">
            <MoreHorizontalIcon className="h-4 w-
4" />
            <span className="sr-only">Дії</span>
          </Button>
        </PopoverTrigger>
        <PopoverContent className="flex flex-col gap-2
transition-colors">
          <Button onClick={() => edit(entity.id)}>
            Редагувати
          </Button>
          <Button
            variant="outline"
            onClick={resetPasswordHandler}
          >
            Скинути пароль
          </Button>
          <Button variant="outline"
onClick={suspendHandler}>
            {entity.suspended ? 'Увімкнути' :
'Відключити'}{' '}
            запис
          </Button>
          <Button variant={'destructive'}
onClick={deleteHandler}>
            Видалити
          </Button>
        </PopoverContent>
      </Popover>

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </TableCell>
    </>
);
};

```

Компонент відповідає за відображення одного користувача у таблиці, що на панелі адміністратора. Серед іншого компонент дозволяє запустити процес редагування користувача, скинути пароль, заблокувати користувача у системі та видалити запис з бази даних.

Розглянутий компонент є репрезентативним відображенням більшості компонентів, оскільки використовує більшість типів взаємодії з застосунком, валідує дані, відображає повідомлення про помилки та використовує бібліотеки лицевих, обрані у минулих розділах.

Перейдемо до роутера повідомлень. Основна функція, що слугує вхідною точкою для анонімних користувачів має наступний код:

```

@app.post(webhook_path)
async def webhook_callback(request: Request):
    try:
        event = await EventFactory.create_event(request)
    except ValueError as e:
        logging.warning(f"Error: {e}")
        return HTTPException(status_code=400, detail=str(e))

    user_id = event.user_unique_id
    with Session() as session:
        user: User = session.get(User, user_id)
        if not user:
            user = User(
                id=user_id,
            )
            session.add(user)
            session.commit()

        if user.suspended:
            event.send_message("Ви були заблоковані. Якщо вважаєте, що це помилка - зверніться на пошту unban@soulful.pp.ua для розблокування.")
            return

        chat = session.execute(select(Chat).where(Chat.user_id == user_id)).scalar_one_or_none()
        if not chat:
            # The following could be used to continuously verify user access to chat, probably unnecessary
            # personnel = get_personnel(session, personnel_ids)

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if not ws_manager.get_client_ids():
            no_personnel_error(event, user_id)

        acq_chat_id = get_acquainted_chat(session,
ws_manager.get_client_ids(), user_id)

        if acq_chat_id:
            chat = Chat()
            chat.id = unarchive_chat(session, acq_chat_id)
            event.send_message("Вітаємо! Ваше попереднє
звернення було відновлено. Як ми можемо вам допомогти?")
        else:
            personnel_id =
get_least_busy_personnel_id(session, ws_manager.get_client_ids())
            chat = Chat(
                user_id=user_id,
                personnel_id=personnel_id,
            )
            session.add(chat)
            session.commit()
            if personnel_id:
                event.send_message("Привіт! Як ми можемо вам
допомогти? Оператор незабаром відповість вам.")

        message = Message(
            text=event.text,
            is_from_user=True,
            chat_id=chat.id,
        )
        session.add(message)
        session.commit()

        if chat.personnel_id not in ws_manager.get_client_ids():
            if chat.personnel_id:
                no_personnel_error(event, user_id,
is_assigned=True)

        send_missed_a_message_email(get_user_email(session,
chat.personnel_id), chat.id)
        return

    try:
        await ws_manager.send_json(chat.personnel_id, {
            'id': message.id,
            'text': message.text,
            'createdAt': message.created_at.timestamp(),
            'chatId': chat.id,
            'isFromUser': message.is_from_user,
        })
    except Exception as e:
        if e == WebSocketDisconnect:
            await ws_manager.disconnect(chat.personnel_id)

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        logging.error(f"Unable to reach an operator who was
previously connected: {e}")

    return "OK"

```

Функція валідує повідомлення та встановлює платформу відправника. Описане відбувається завдяки реалізованому патерну адаптер, код та приклад втілення викладено у додатку Б.

Наступним кроком створюється обліковий запис в базі даних, якщо користувач звертається вперше. Далі створюється або віднаходиться чат. Якщо це нове звернення, обраховується зайнятість операторів за допомогою розглянутої в розділі 2.3 функції та визначається, кого призначити до нового чату. У випадку відсутності операторів онлайн, чат створюється без оператора, а користувача повідомляється про затримку. Коли новий оператор приєднається, у нього буде змога отримати цей чат. Після розподілення чату, оператору надсилається повідомлення по WebSockets, за умови, що він підключений.

Саме використання WebSockets дозволяє мінімізувати кількість повільних запитів використовуючи Hypertext Transfer Protocol (HTTP) та запитів до бази даних, максимізуючи швидкість доставки повідомлень. Для отримання можливості пересилання повідомлень цим способом необхідно створити точку для приєднання операторів, яка має наступний вихідний код:

```

@app.websocket("/ws/{personnel_token}")
async def websocket_endpoint(websocket: WebSocket,
personnel_token: str):
    with Session() as session:
        personnel_id = session.execute(text("SELECT user_id FROM
\"Session\" WHERE session_token = :token"), {'token':
personnel_token}).scalars().one_or_none()

        user = get_personnel(session, [personnel_id])

        if not personnel_id or not user:
            return HTTPException(status_code=401,
detail="Unauthorized")

    await ws_manager.connect(personnel_id, websocket)

    while ws_manager.get_client(personnel_id):
        data = await ws_manager.receive_text(personnel_id)

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if not data:
    continue

data = json.loads(data)

with Session() as session:
    message = Message(
        text=data['text'],
        chat_id=data['chatId'],
        is_from_user=data['isFromUser'],
    )
    session.add(message)
    session.commit()

send_message(data['userId'], data['text'])

await ws_manager.send_json(personnel_id, {
    'id': message.id,
    'text': message.text,
    'createdAt': message.created_at.timestamp(),
    'chatId': message.chat_id,
    'isFromUser': message.is_from_user,
})

```

При підключенні оператор використовує у шляху кінцевої точки власний токен доступу, який перевіряється на актуальність та на факт наявності у оператора необхідних дозволів для використання системи. Далі функція переходить у режим очікування повідомлень від оператора та пересилання їх кінцевому отримувачу.

Таким чином успішно реалізовано систему адміністрування застосунку психологічної підтримки. Розглянуто основні типи процесів, модулів та компонентів системи та принципи їх роботи.

Висновки до другого розділу

У результаті аналізу розробленої системи адміністрування вебзастосунку психологічної підтримки було встановлено, що основні цілі, такі як надання ергономічного інтерфейсу та інтеграція різноманітних платформ комунікації, успішно досягнуті. Чат-функціональність, що включає розподіл завдань серед операторів за допомогою алгоритмів, які враховують метрики швидкості відповіді та кількості чатів, забезпечує ефективність у взаємодії та оптимізацію

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

робочого навантаження. Це сприяє підвищенню загальної продуктивності системи та забезпечує адаптацію до індивідуальних потреб користувачів, підтримуючи високий рівень особистісного підходу у наданні психологічної підтримки.

Важливим аспектом стала також внутрішня база знань, що слугує динамічним ресурсом для операторів, надаючи актуальну інформацію та ресурси для навчання. Адміністративні можливості системи, включаючи деталізоване управління профілями користувачів та контроль доступу, відіграють ключову роль у забезпеченні ефективності та безпеки в роботі системи. Моніторинг та аналіз статистичних даних дозволяють виявляти потенційні зони для подальшого вдосконалення, що є основою для розробки стратегій оптимізації взаємодій всередині системи та збільшення загальної задоволеності користувачів.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

3. ІНТЕРФЕЙС ТА ПОРЯДОК ВЗАЄМОДІЇ З СИСТЕМОЮ АДМІНІСТРУВАННЯ ЗАСТОСУНКУ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

3.1 Порядок встановлення та налаштування системи

Система має два режими роботи: для розробки та для повноцінної роботи. Для виконання у другому виконується ряд оптимізацій та сам процес розгортання застосунку на сервері.

Основний застосунок має інструкцію як з локального запуску, так і з розгортання застосунку. Щоб запустити проєкт слідуючи йому, не потрібні жодні додаткові ресурси.

В першу чергу розглянемо застосунок у режимі для розробки. В ньому основна частина працює як монолітний застосунок. Завдяки Next.js сервер моментально перезапущається після внесення змін в код, дозволяючи швидко бачити поведінку системи після змін та результати цих змін.

Застосунок працює на порті 80, що не є стандартом, однак спрощує тестування за умови можливості доступу до комп'ютера з глобальної мережі.

Сторінки доповнюються компонентами для налагодження системи. Вони в тому числі відображають помилки у зрозумілому форматі, перезавантажують сторінку після внесення змін у вихідний код та відновлюють з'єднання при перезапуску сервера.

На стороні роутера повідомлень забезпечується подібний досвід, однак не має кроку компіляції (у зв'язку з чим перезапуск відбувається швидше) та забезпечується бібліотекою FastAPI. Працює на 443 порті та завжди має Secure Sockets Layer (SSL) сертифікат для забезпечення шифрування комунікацій, оскільки частина підтримуваних месенджерів мають таку вимогу.

Розглянемо процес розгортання застосунку для запуску в режимі повноцінної роботи. Для повного розуміння роботи системи та її компонентів у режимі, розглянемо діаграму розгортання.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

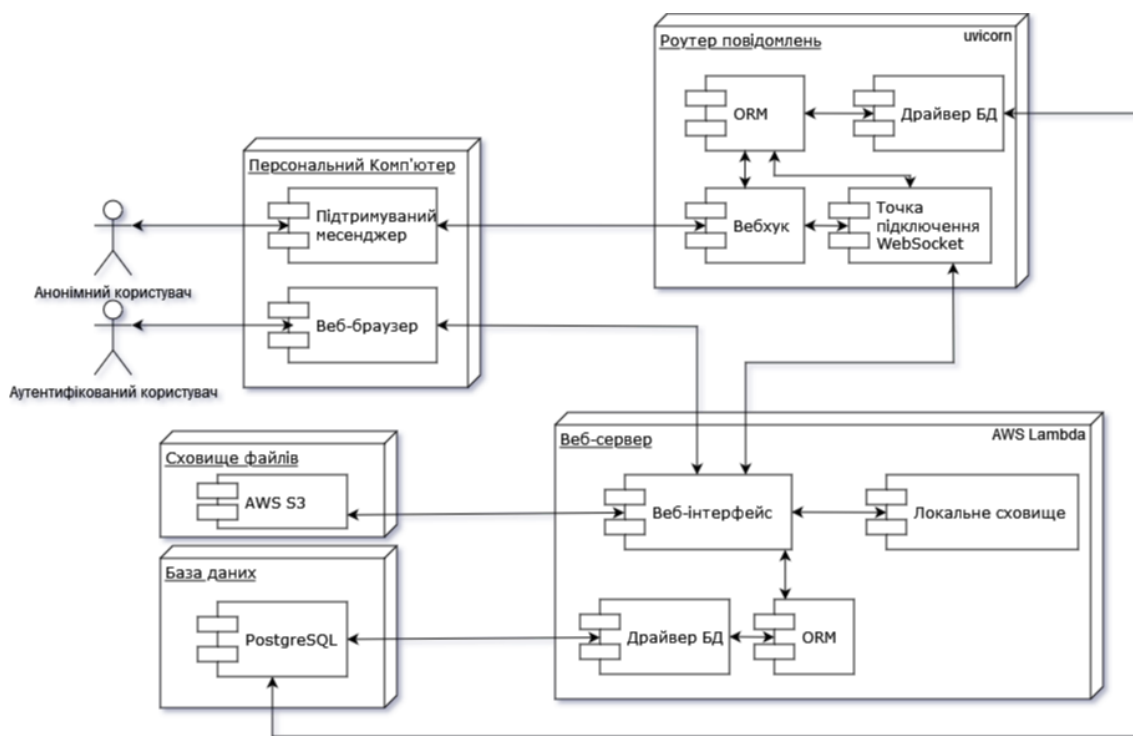


Рисунок 3.1. Діаграма розгортання застосунку.

На рисунку 3.1 зображено компоненти системи у вигляді, який вони мають після розгортання. Серед суттєвих відмінностей від локального запуску – запуск коду основного застосунку на мікросервісній архітектурі AWS Lambda та роутера повідомлень у режимі продуктивності. Обидва застосунки втрачають будь-які функції пов'язані зі спрощенням розробки, тобто не фіксують зміни та не перезапускаються, не надають повних повідомлень про помилки. Результатом втрати вказаного є значно підвищена продуктивність роботи застосунків.

На діаграмі окрім зазначеного також відображено базу даних, до якої підключаються застосунки, а також віддалене сховище файлів, де основний застосунок зберігає медіа користувачів.

Актори на діаграмі – анонімний користувач, що використовує підтримуваний месенджер та отримує доступ до системи за допомогою роутера повідомлень та аутентифікований користувач, який отримує доступ до системи напряду через основний застосунок.

Для забезпечення автоматичного розгортання необхідно використати GitLab continuous integration, continuous delivery (CI/CD) pipelines та створити

конфігураційний файл, який буде запускати GitLab Runner. Для основного застосунку він має наступний вигляд:

```
stages:
  - deploy

deploy_preview:
  stage: deploy
  except:
    - main
  script:
    - vercel deploy --yes --token=$VERCEL_TOKEN

deploy_production:
  stage: deploy
  only:
    - main
  script:
    - vercel deploy --yes --prod --token=$VERCEL_TOKEN
```

Завдяки використанню сервісу Vercel, конфігурацію розгортання зменшено до мінімуму – одного запиту сервісу з секретним ключем.

Для роутера повідомлень конфігурація складніше, оскільки розгортається він в Docker контейнер на віддаленому сервері. Конфігураційний файл має таке наповнення:

```
stages:
  - deploy

variables:
  SSL_DIR: ./ssl
  SSL_KEYFILE: $SSL_DIR/key.pem
  SSL_CERTFILE: $SSL_DIR/cert.pem

deploy:
  stage: deploy
  only:
    - main
  script:
    - mkdir $SSL_DIR

    - touch "$SSL_CERTFILE"
    - printf "%s" "$SSL_CERT" > "$SSL_CERTFILE"

    - touch "$SSL_KEYFILE"
    - printf "%s" "$SSL_KEY" > "$SSL_KEYFILE"

    - sudo docker stop message-router && sudo docker rm
message-router
```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

- sudo docker buildx build -t message-router .

- sudo docker run -d --restart=unless-stopped --network
nginx_bridge --ip 172.23.0.69 -p "${PORT}:${PORT}" --name message-
router -e "PORT=${PORT}" -e "SSL_KEYFILE=${SSL_KEYFILE}" -e
"SSL_CERTFILE=${SSL_CERTFILE}" -e "API_URL=${API_URL}" -e
"WEBHOOK_PATH=${WEBHOOK_PATH}" -e
"TELEGRAM_TOKEN=${TELEGRAM_TOKEN}" -e
"TELEGRAM_VERIFICATION_TOKEN=${TELEGRAM_VERIFICATION_TOKEN}" -e
"VIBER_TOKEN=${VIBER_TOKEN}" -e
"VIBER_MIN_API_VERSION=${VIBER_MIN_API_VERSION}" -e
"FACEBOOK_VERIFICATION_TOKEN=${FACEBOOK_VERIFICATION_TOKEN}" -e
"FACEBOOK_APP_SECRET=${FACEBOOK_APP_SECRET}" -e
"FACEBOOK_PAGE_ID=${FACEBOOK_PAGE_ID}" -e
"FACEBOOK_PAGE_TOKEN=${FACEBOOK_PAGE_TOKEN}" -e
"FACEBOOK_API_VERSION=${FACEBOOK_API_VERSION}" -e
"LOCAL_STORAGE_PATH=${LOCAL_STORAGE_PATH}" -e
"DB_CONNECTION_STRING=${DB_CONNECTION_STRING}" -e "STAGE=${STAGE}"
-e "AWS_REGION=${AWS_REGION}" -e
"AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}" -e
"AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}" -e
"AWS_SES_FROM_IDENTITY=${AWS_SES_FROM_IDENTITY}" message-router

```

Оскільки використовується також Docker, для нього створено наступний Dockerfile:

```

ARG WORKDIR=.

FROM python:3.11-alpine AS builder

ARG WORKDIR

ENV PYTHONDONTWRITEBYTECODE 1 \
    PYTHONUNBUFFERED 1 \
    POETRY_NO_INTERACTION=1 \
    POETRY_VIRTUALENVS_IN_PROJECT=1 \
    POETRY_VIRTUALENVS_CREATE=1 \
    POETRY_CACHE_DIR=/tmp/poetry_cache \
    STAGE=prod

ADD pyproject.toml poetry.lock ${WORKDIR}/

RUN apk add build-base libffi-dev

RUN pip install poetry
RUN poetry config virtualenvs.in-project true

WORKDIR ${WORKDIR}
COPY . .

RUN poetry install

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```
FROM python:3.11-alpine as runtime

ARG WORKDIR
WORKDIR ${WORKDIR}

COPY --from=builder ${WORKDIR} .
ADD . .

ENTRYPOINT ["poetry", "run", "python", "-m", "src.run"]
```

В ньому побудова застосунку поділяється на два кроки, що дозволяє ізолювати підготовку середовища, що включає в себе встановлення змінних середовища, встановлення Poetry та завантаження залежностей, від запуску самого сервера. Це дозволяє не перезавантажувати усе середовище при кожному розгортанні, оскільки зазвичай змінюється лише код застосунку.

3.2 Опис інтерфейсу програмного продукту

Розроблений застосунок має зрозумілий інтерфейс у єдиній темі. Почнемо розгляд сторінки панелі керування застосунком з сторінки користувачів.

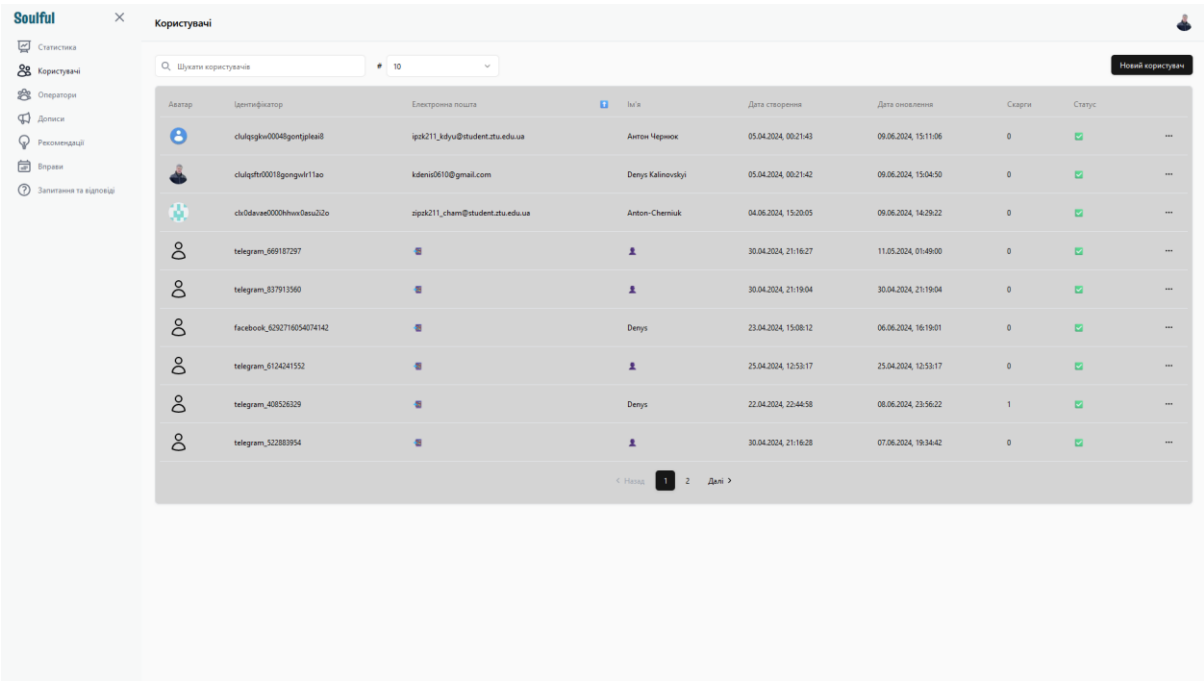


Рисунок 3.2. Сторінка «Користувачі» на панелі керування.

На рисунку 3.2 зображено таблицю, яка містить усіх користувачів системи включно з користувачами месенджерів (їх облікові записи позначені відповідною картинкою). Є можливість сортувати записи за обраною

характеристикою. На рисунку увімкнено сортування за електронною поштою у порядку зростання, що дозволяє переглянути в першу чергу користувачів з електронною поштою, зареєстрованих через застосунок.

The screenshot shows a web interface for managing users. On the left is a sidebar with navigation links: Статистика, Користувачі, Додатки, Рекомендації, Вправи, and Запитання та відповіді. The main area has a search bar with 'Denys' entered and a dropdown for 10 results. Below is a table of users.

Аватар	Ідентифікатор	Електронна пошта	Ім'я	Дата створення	Дата оновлення	Скарги	Статус	
	ckulqph00018gonguk11ao	kdeni0610@gmail.com	Denys Kalinovskyi	05.04.2024, 00:21:42	09.06.2024, 15:04:30	0		...
	viber_513v5QRECMoOduyT0bAc+		Denys	23.04.2024, 15:38:27	28.05.2024, 01:50:12	0		...
	telegram_408326329		Denys	22.04.2024, 22:44:58	08.06.2024, 23:56:22	1		...
	facebook_6292716054074142		Denys	23.04.2024, 15:08:12	06.06.2024, 16:19:01	0		...

Рисунок 3.3. Пошук на сторінці «Користувачі» на панелі керування.

Оскільки в системі передбачається велика кількість користувачів, передбачено пошук по більшості наявної інформації про користувачів. На рисунку 3.3 здійснено пошук за словом «Denys», та одразу кілька записів мають це слово у імені, тому їх було відображено.

The screenshot shows the same user list as Figure 3.3. A context menu is open over the first user, showing the following actions: Редагувати, Скинути пароль, Відключити запис, and Видалити.

Аватар	Ідентифікатор	Електронна пошта	Ім'я	Дата створення	Дата оновлення	Скарги	Статус	
	ckOdavee000hwx0asu2i2o	zipck211_cham@student.ztu.edu.ua	Anton-Cherniuk	04.06.2024, 15:20:05	09.06.2024, 15:16:39	0		...
	telegram_837913560			30.04.2024, 21:19:04	30.04.2024, 21:19:04	0		
	telegram_322883954			30.04.2024, 21:16:28	07.06.2024, 19:34:42	0		
	telegram_669187297			30.04.2024, 21:16:27	11.05.2024, 01:49:00	0		

Рисунок 3.4. Дії, доступні з обліковими записами користувачів.

На рисунку 3.4 зображено вікно, що дозволяє відредагувати обліковий запис та видалити його. Окремо варто зазначити можливість скидання пароля. У випадку використання функції, користувач отримує повідомлення на пошту із складним випадковим паролем, адміністратор не знає ні нового ні старого пароля.

Функція блокування дозволяє повністю обмежити доступ до системи, навіть якщо користувач лише спілкується у чаті. Одразу ж після натискання

кнопки користувач втрачає будь-який доступ як до платформи, так і до чату зі сторонніх месенджерів.

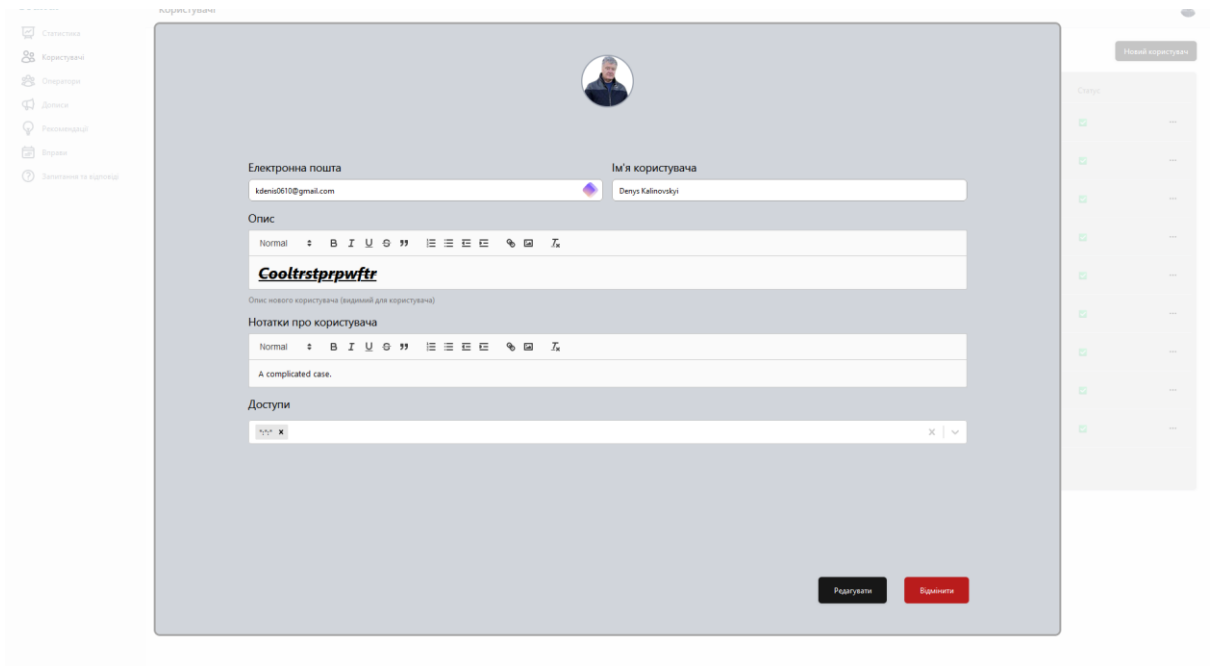


Рисунок 3.5. Вікно створення та редагування користувачів.

На рисунку 3.5 відображено вікно з інформацією одного з користувачів. Тут адміністратори мають змогу відредагувати інформацію про користувача та змінити його дозволи.

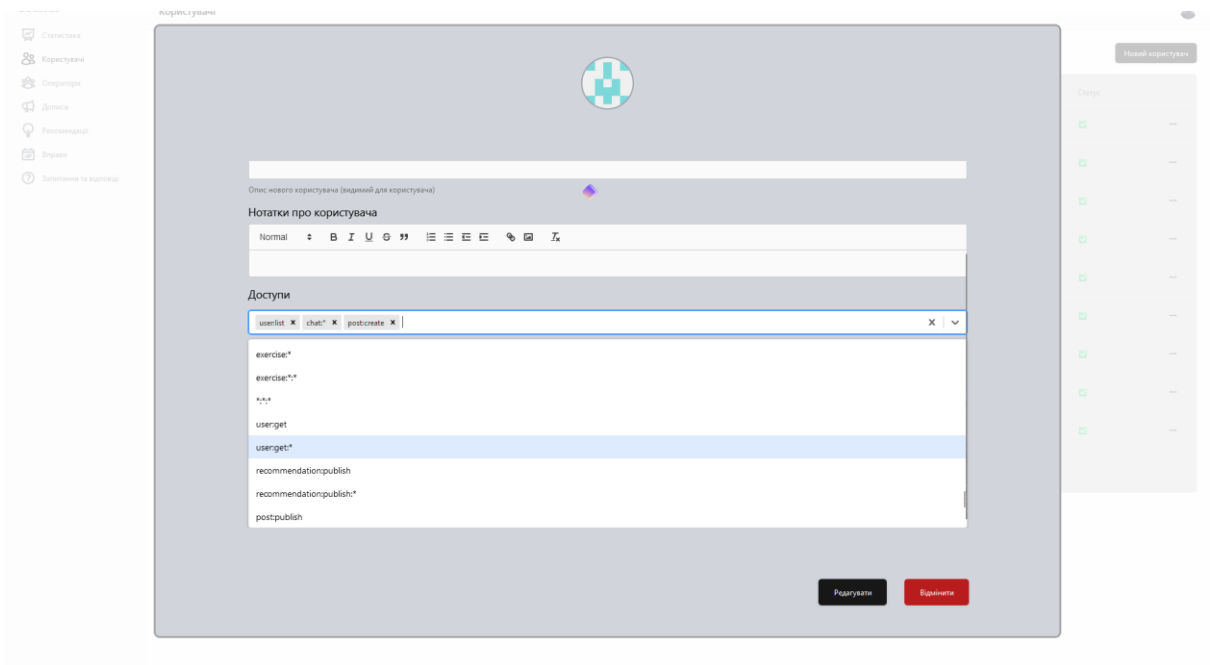


Рисунок 3.6. Налаштування дозволів користувача.

На рисунку 3.6 відображено наповнення поля, у якому користувачу можна встановити дозволи. Назва дозволу розділена на 3 частини, які мають наступне значення:

1. Сутність, щодо якої дію дозволено.
2. Дія, яку дозволено виконувати відносно сутності.
3. Рівень доступу, відсутність значення вказує на звичайний рівень доступу.

Варто зазначити, що будь-яку частину можна замінити на зірку «*», що має значення відсутності обмежень, тобто «chat:*» надає користувачу доступ до усіх дій в чаті, а «*:*:*» надає користувачу повний доступ до усіх можливостей в системі. У випадку з дозволами, що мають варіант з рівнем доступу, наприклад «post:list» та «post:list:*», вони надають доступ лише до власних прихованих дописів та доступ до усіх постів відповідно.

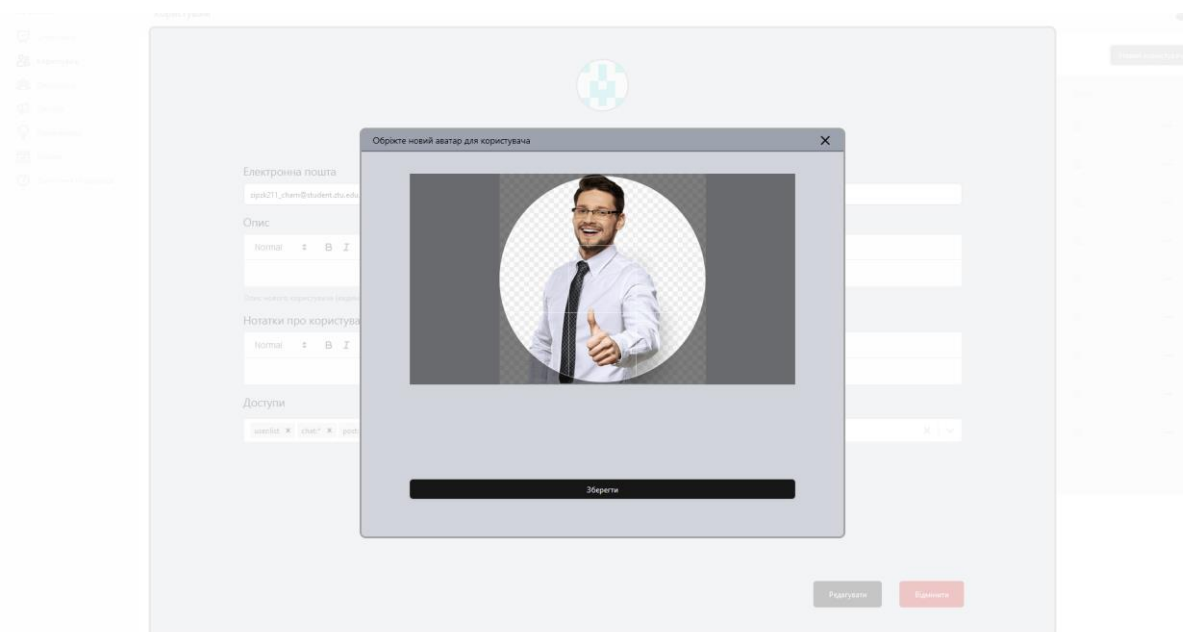


Рисунок 3.7. Вікно встановлення картинки профіля користувача.

На рисунку 3.7 зображено вікно, у якому користувач обрізає фото для забезпечення бажаного результату при збереженні єдиного формату між різними користувачами.

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відписки

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи</

Рисунок 3.8. Сторінка «Оператори» на панелі керування.

На рисунку 3.8 зображено таблицю, яка містить усіх користувачів системи, що мають доступ до чату, тобто операторів. На сторінці доступні усі функції, вже описані на сторінці «Користувачі».

Статистика

Користувачі

Оператори

Дописи

Рекомендації

Вправи

Застереження та відповіди

Пошук дописи

10

Новий допис

Зображення	Ідентифікатор	Заголовок	Автор	Дата створення	Дата оновлення	Статус
	cklubbm0003rccsc0279f99	Значення майнафунес для покращення ментального здоров'я		04.06.2024, 23:16:13	04.06.2024, 23:16:13	 ---
	cklubbm0003rccsc0279f99	Як встановлення між може покращити ваше ментальне здоров'я		04.06.2024, 23:16:13	04.06.2024, 23:16:13	 ---
	cklubbm0003rccsc0279f99	Стратегії самодіагностики для боротьби з тривогою: Практичні поради для покращення вашого ментального здоров'я		04.06.2024, 23:16:13	04.06.2024, 23:16:13	 ---
	cklubbm0003rccsc0279f99	Психологічне пильність: ключ до покращення ментального здоров'я		04.06.2024, 23:16:13	04.06.2024, 23:16:13	 ---
	cklubbm0003rccsc0279f99	Психологічне стійкість: Шлях до кращого ментального здоров'я		04.06.2024, 23:16:13	04.06.2024, 23:16:13	 ---
	cklubbm0003rccsc0279f99	Освоєння стресу через мистецтво усвідомленості: Практичні поради для покращення ментального здоров'я		04.06.2024, 23:16:13	06.06.2024, 16:09:27	 ---
	cklubbm0003rccsc0279f99	Важливість емоційної присутності для ментального здоров'я		04.06.2024, 23:16:13	06.06.2024, 16:09:40	 ---
	cklubbm0003rccsc0279f99	Важливість самовисловлення у покращенні ментального здоров'я		04.06.2024, 23:16:13	04.06.2024, 23:16:13	 ---
	cklubbm0003rccsc0279f99	Стратегії подолання тривоги: знаходимо спокій у неспокійному світі		04.06.2024, 23:16:13	04.06.2024, 23:16:13	 ---

Рисунок 3.9. Сторінка «Дописи» на панелі керування.

На рисунку 3.9 зображено таблицю, яка містить усі дописи. Є можливість сортувати записи за обраною характеристикою. Доступна решта схожого або аналогічного функціоналу з попередніх сторінок. Важливо зазначити, що замість блокування дописи можна приховати або опублікувати.

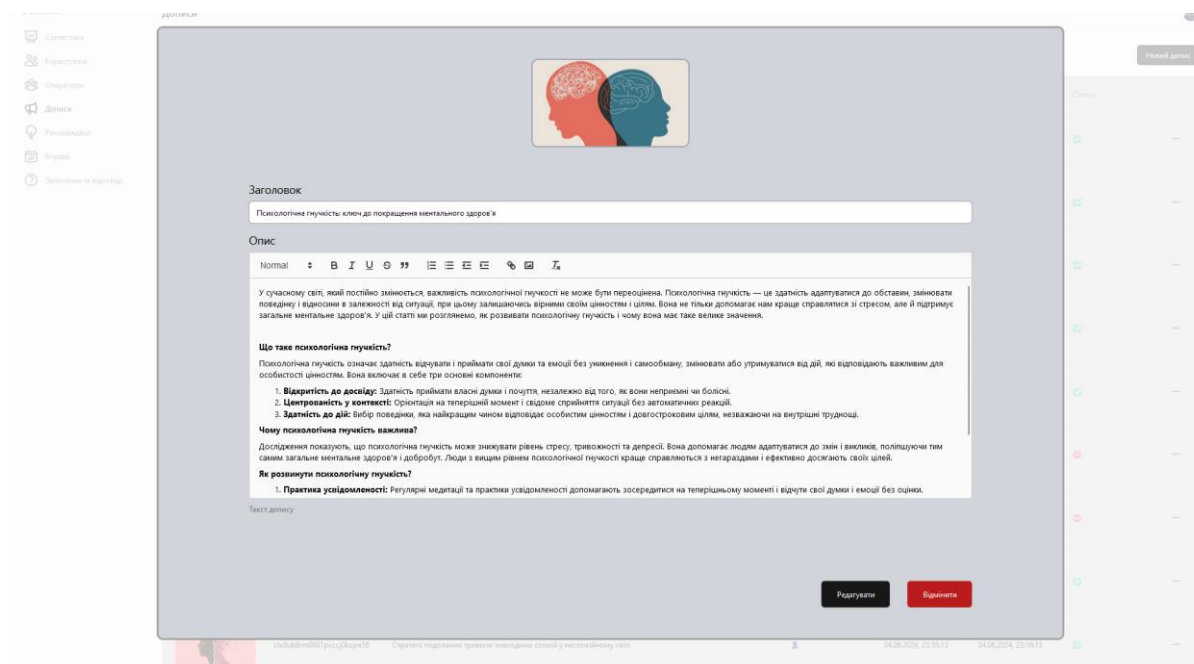


Рисунок 3.10. Вікно створення та редагування дописів.

На рисунку 3.10 відображено вікно з інформацією одного з дописів. Тут адміністратори мають змогу відредагувати інформацію про допис.

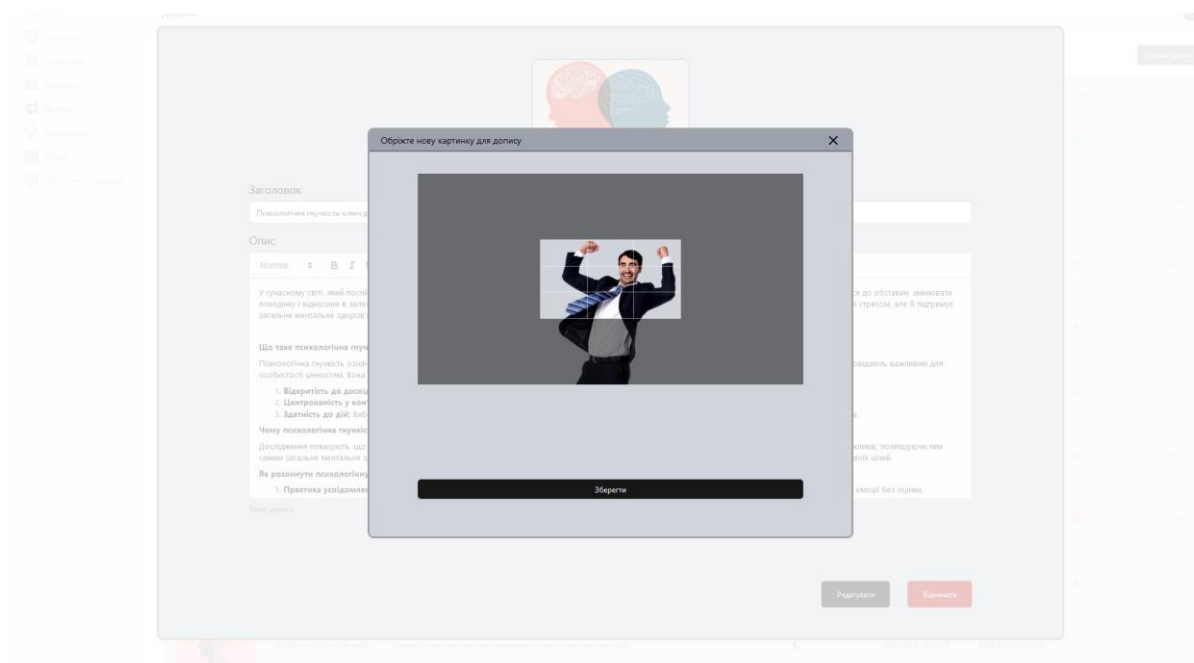


Рисунок 3.11. Вікно встановлення картинки допису.

На рисунку 3.11 зображено вікно, у якому користувач обрізає фото допису для забезпечення бажаного результату при збереженні єдиного формату між різними дописами.

Статистика

Користувачі

Додатки

Рекомендації

Вправи

Застосування та відслідковування

Рекомендації

Пошук рекомендацій

10

Нова рекомендація

Ідентифікатор	Заголовок	Наповнення	Автор	Дата створення	Дата оновлення	Статус	
skBr1my0041hmtg22gr2n7	Розвивайте свої творчі здібності	Творчість є чудовим способом вираження себе і зменшення стресу.		04.06.2024, 20:48:51	04.06.2024, 20:48:51		---
skBr1my0042hmtg3bqa7n	Піклуйтеся про своє тіло	Регулярний догляд за своїм тілом покращує самопочуття та підтримує психологічне здоров'я.		04.06.2024, 20:48:51	04.06.2024, 20:48:51		---
skBr1my0043hmtg3hucstm	Дозвольте собі помилитися	Прийняття невдач як частини навчального процесу може допомогти вам рости та розвиватися.		04.06.2024, 20:48:51	04.06.2024, 20:48:51		---
skBr1my0044hmtg38frbvd	Розвивайте свою інтуїцію	Слухаючи свою інтуїцію, ви можете краще розуміти свої інстинкти та емоції.		04.06.2024, 20:48:51	04.06.2024, 20:48:51		---
skBr1my0045hmtg3ouep7	Практикуйте конструктивне самокритику	Оцінюйте свої дії об'єктивно, але з доброзичливістю, щоб покращити свої навички та самоповагу.		04.06.2024, 20:48:51	04.06.2024, 20:48:51		---

< Назад

1

...

4

5

6

Далі >

Рисунок 3.12. Сторінка «Рекомендації» на панелі керування.

На рисунку 3.12 зображено таблицю, яка містить усі рекомендації. Є можливість сортувати записи за обраною характеристикою. Доступна решта схожого або аналогічного функціоналу з сторінки дописів.

Статистика

Користувачі

Додатки

Рекомендації

Вправи

Застосування та відслідковування

Рекомендації

Пошук рекомендацій

10

Нова рекомендація

Заголовок

Розвивайте свої творчі здібності

Наповнення

Творчість є чудовим способом вираження себе і зменшення стресу.

Редагувати

Вирішити

Рисунок 3.13. Вікно створення та редагування рекомендацій.

На рисунку 3.13 відображено вікно з інформацією однієї з рекомендацій. Тут адміністратори мають змогу відредагувати рекомендацію.

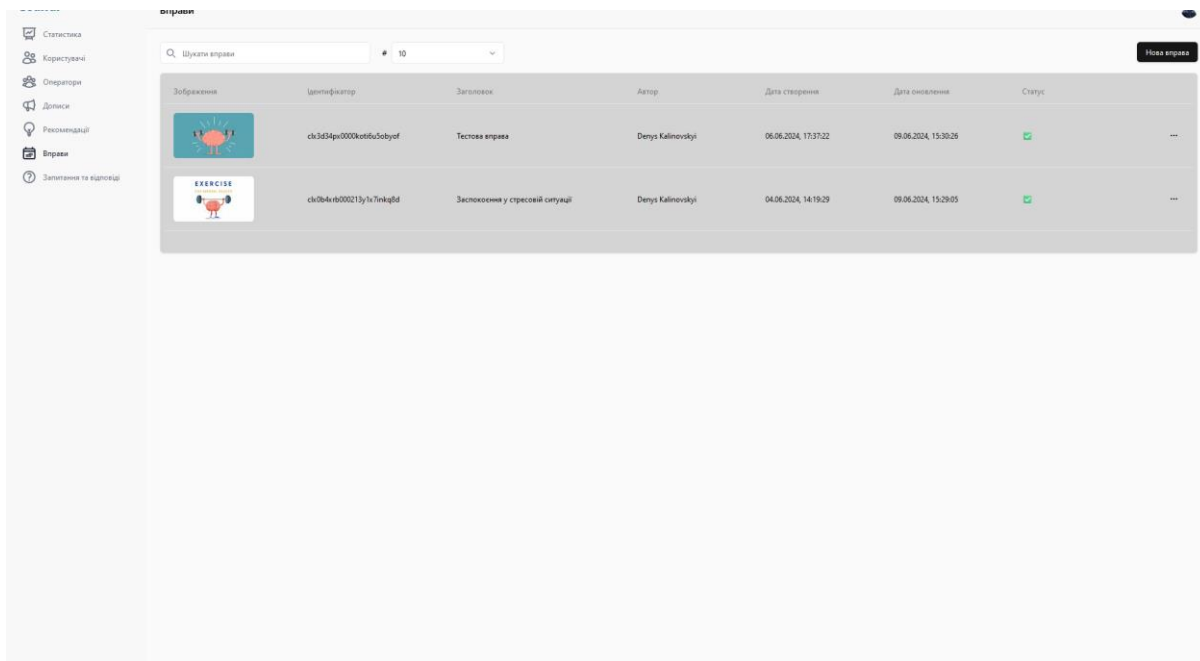


Рисунок 3.14. Сторінка «Вправи» на панелі керування.

На рисунку 3.14 зображено таблицю, яка містить усі вправи. Є можливість сортувати записи за обраною характеристикою. Доступна решта схожого або аналогічного функціоналу з сторінки дописів.

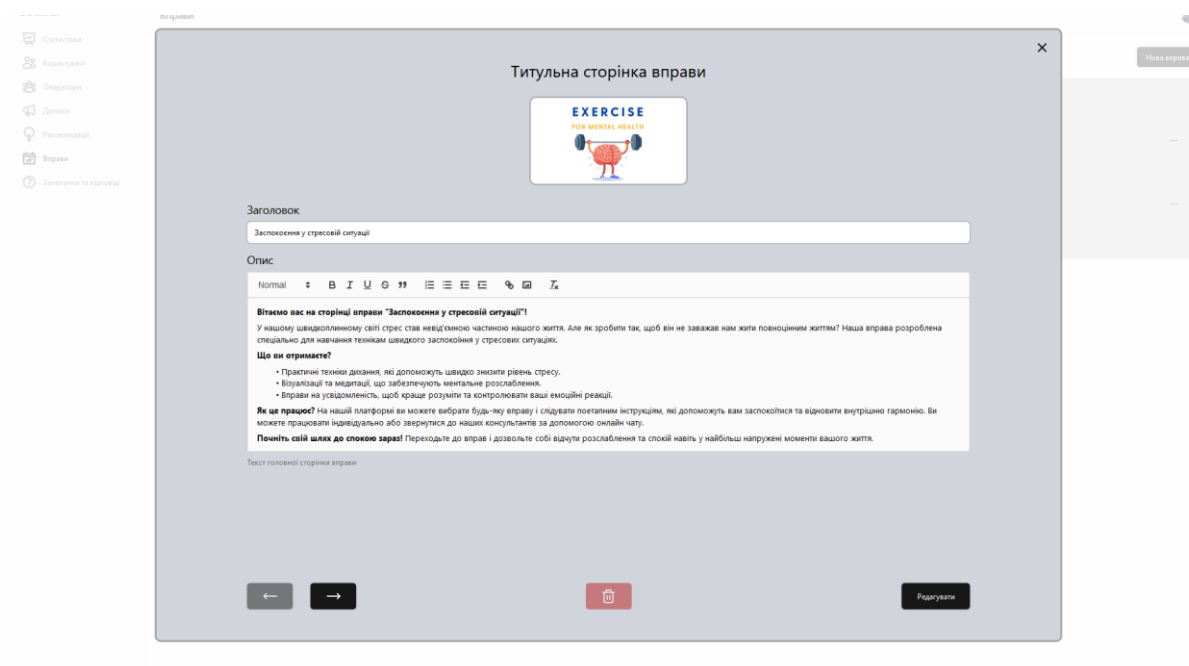


Рисунок 3.15. Вікно створення та редагування головної сторінки вправи.

На рисунку 3.15 відображено вікно з інформацією однієї з вправ. Тут адміністратори мають змогу відредагувати головну сторінку вправи.

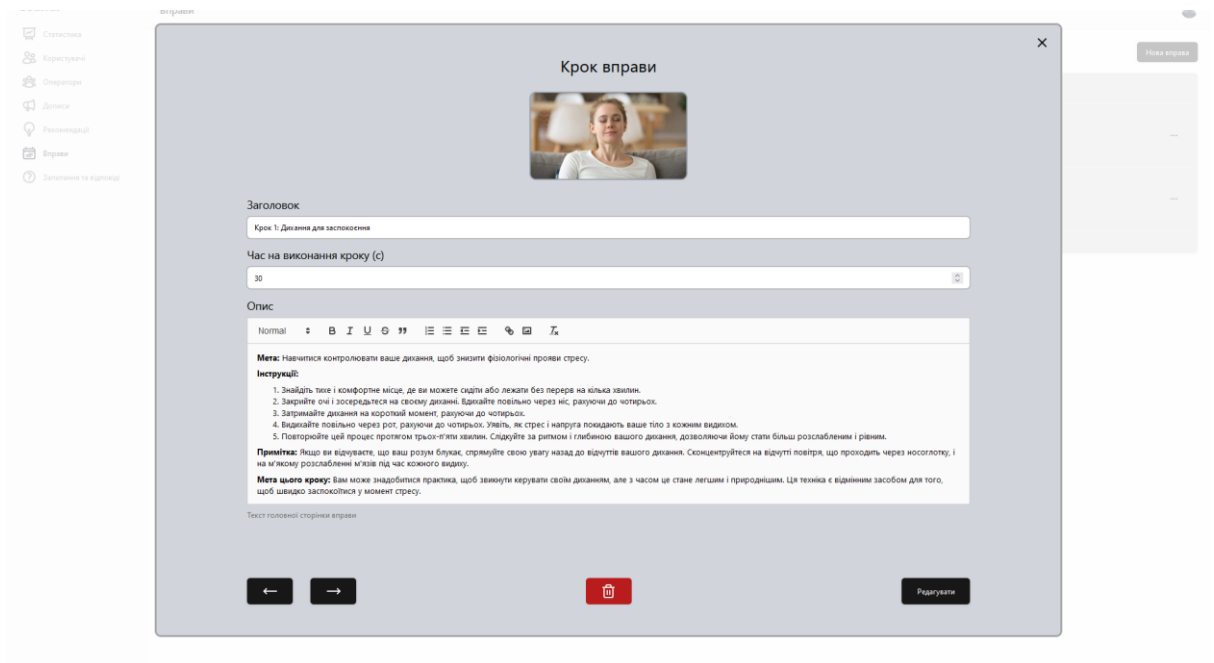


Рисунок 3.16. Вікно створення та редагування кроків вправи.

Вікно редагування вправ поділяється на дві частини: головна сторінка та кроки вправи. На рисунку 3.16 зображено частину редагування кроку вправи, який окрім інших змін, також має поле для введення кількості секунд, що потрібні на виконання вправи.

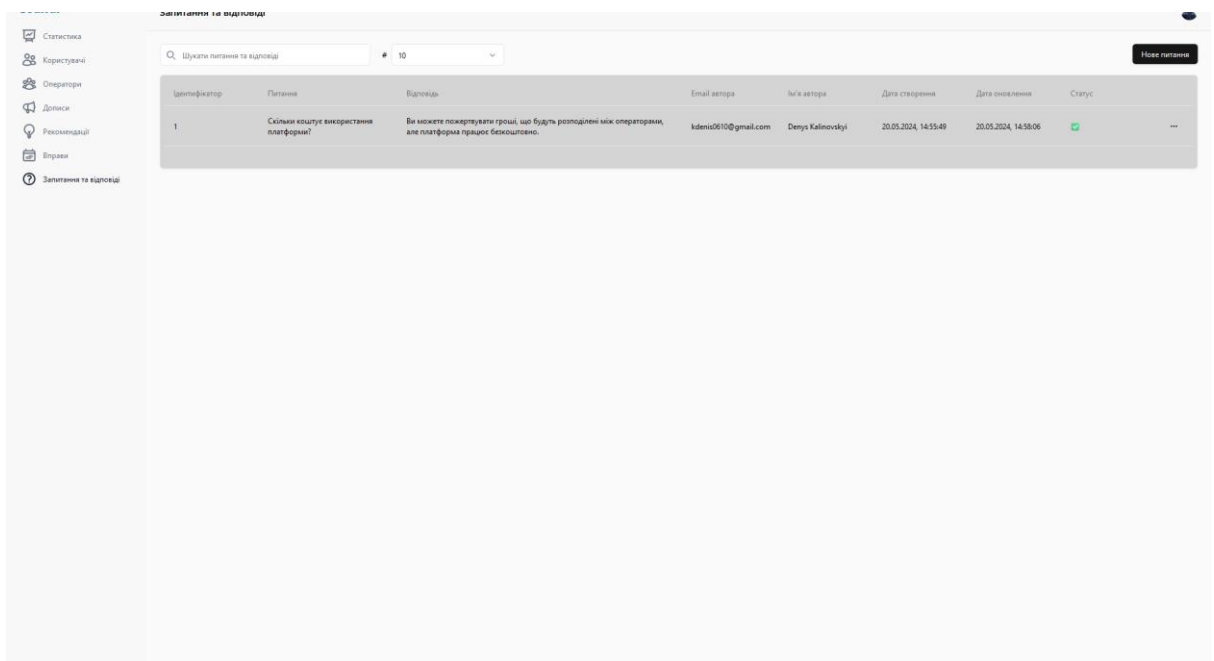


Рисунок 3.17. Сторінка «Запитання та відповіді» на панелі керування.

На рисунку 3.17 зображено таблицю, яка містить усі звернення користувачів з питаннями. Є можливість сортувати записи за обраною

характеристикою. Доступна решта схожого або аналогічного функціоналу з сторінки дописів.

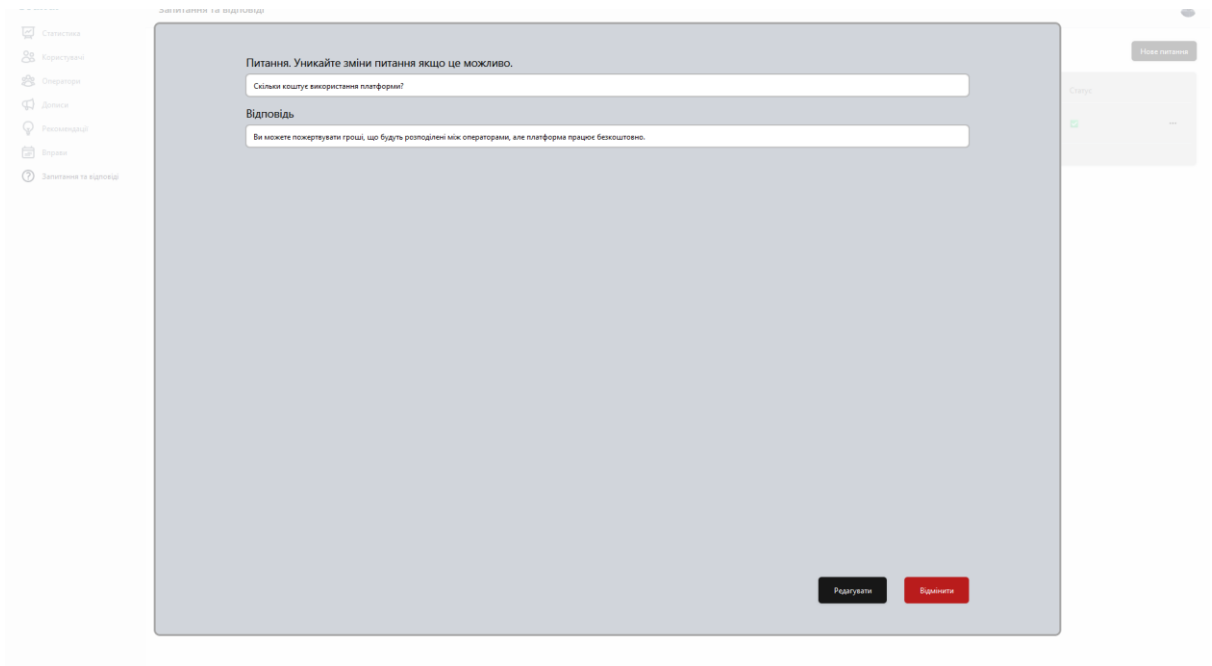


Рисунок 3.18. Вікно створення та редагування питання та відповіді.

На рисунку 3.18 відображено вікно з інформацією одного з запитань користувачів. Тут адміністратори мають змогу відредагувати як питання, так і відповідь. Коли питання отримує відповідь, його автор отримує електронний лист з відповіддю. Адміністратор має змогу опублікувати це запитання на відповідній сторінці застосунку.

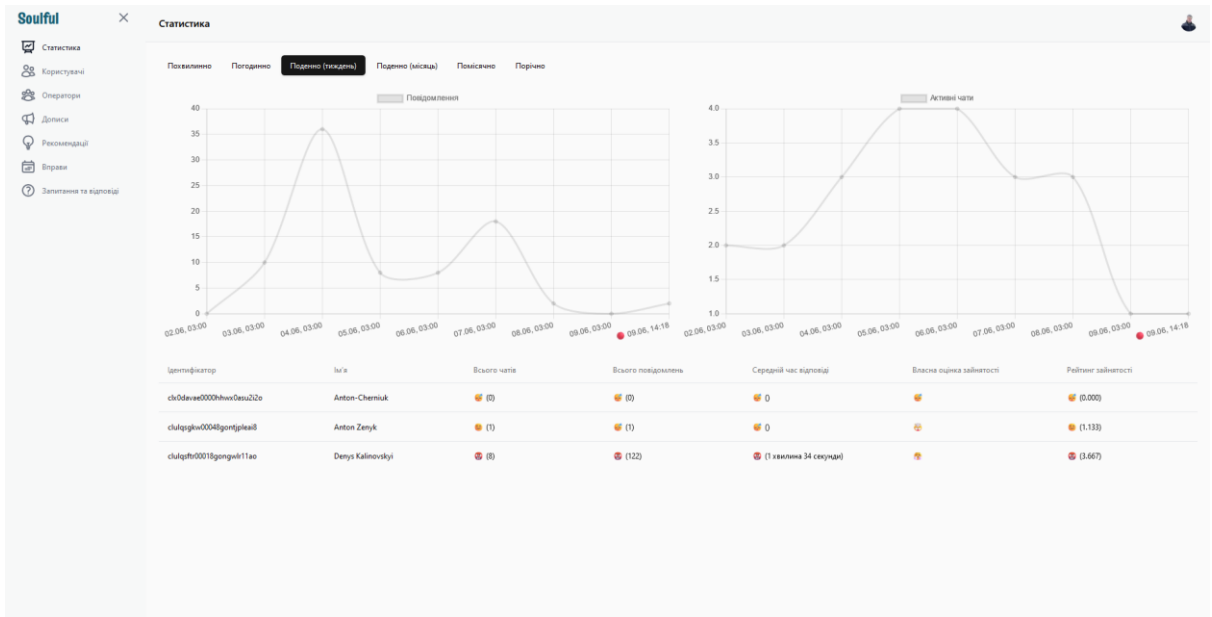


Рисунок 3.19. Сторінка «Статистика» на панелі керування.

На рисунку 3.19 зображено сторінку статистики чату. Графіки відображають кількість повідомлень та кількість відкритих чатів у відповідний період.

Під графіками зображено таблицю, що відображає статистику повідомлень, чатів, часу на відповідь та оцінки власної зайнятості оператором. Значення додатково нормалізовано (приведено до значень на проміжку від 0 до 1) та візуально позначено емодзі, що позначають рівень зайнятості (всього рівнів зайнятості 5). Оператори у таблиці відсортовані за збільшенням зайнятості.



Рисунок 3.20. Графіки на сторінці «Статистика» з іншими налаштуваннями.

На рисунку 3.20 зображено вигляд графіків при зміні налаштування на денні відрізки за останній місяць. Налаштування змінюються за допомогою кнопок у верхній частині сторінки.

Розгляд функціоналу адміністраторів завершено, наступним розглянуто функціонал, передбачений для операторів.

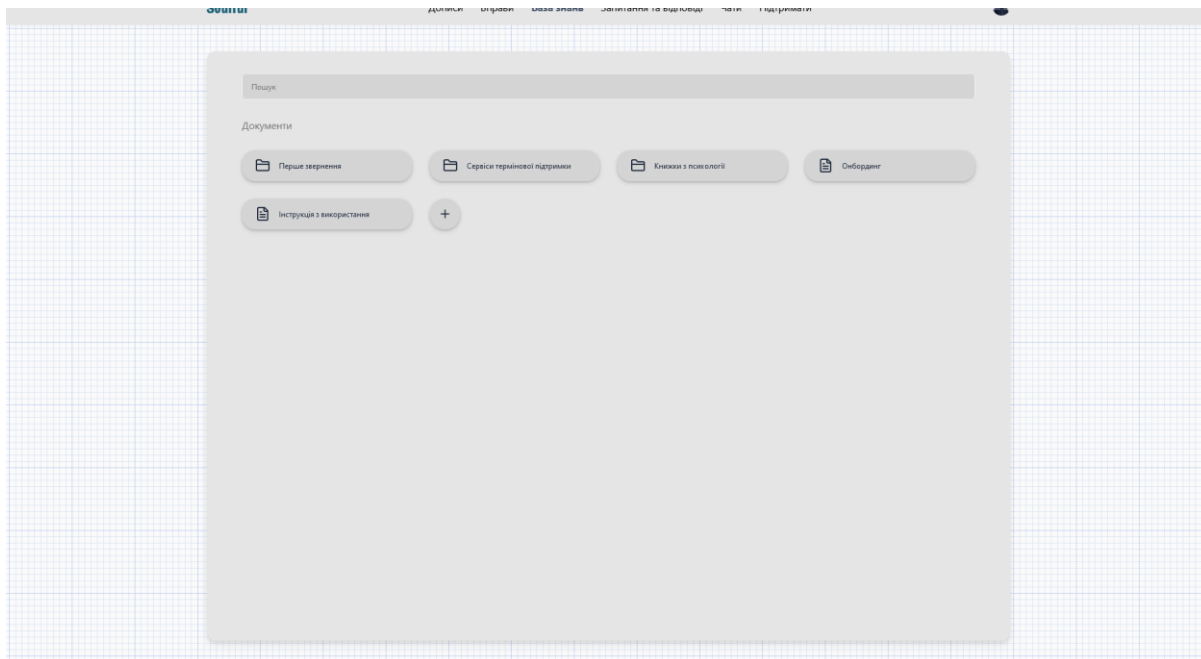


Рисунок 3.21. Модуль «База знань».

На рисунку 3.21 зображено інтерфейс бази знань, модуля, що дозволяє операторам отримувати, зберігати та обмінюватися інформацією. Для організації інформації користувачі мають змогу розміщати документи у папках. Допускається будь-який рівень вкладеності.

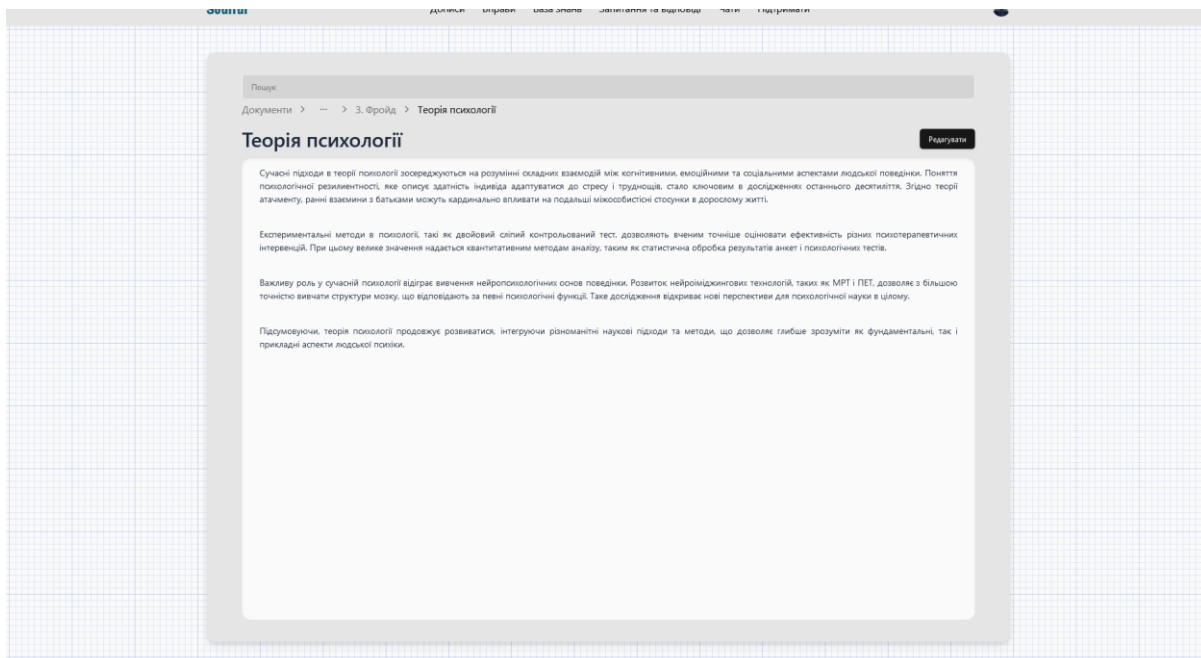


Рисунок 3.22. Перегляд документів у модулі «База знань».

На рисунку 3.22 зображено вигляд документа в базі знань. Додатково варто зазначити елемент навігації хлібні крихти, що дозволяють легко навігувати між директорій та файлів у базі знань. Завдяки використанню Next.js роутера під час

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

переходу між сторінками, повноцінна навігація доступна також і використовуючи перехід на попередню та наступну сторінку у браузері.

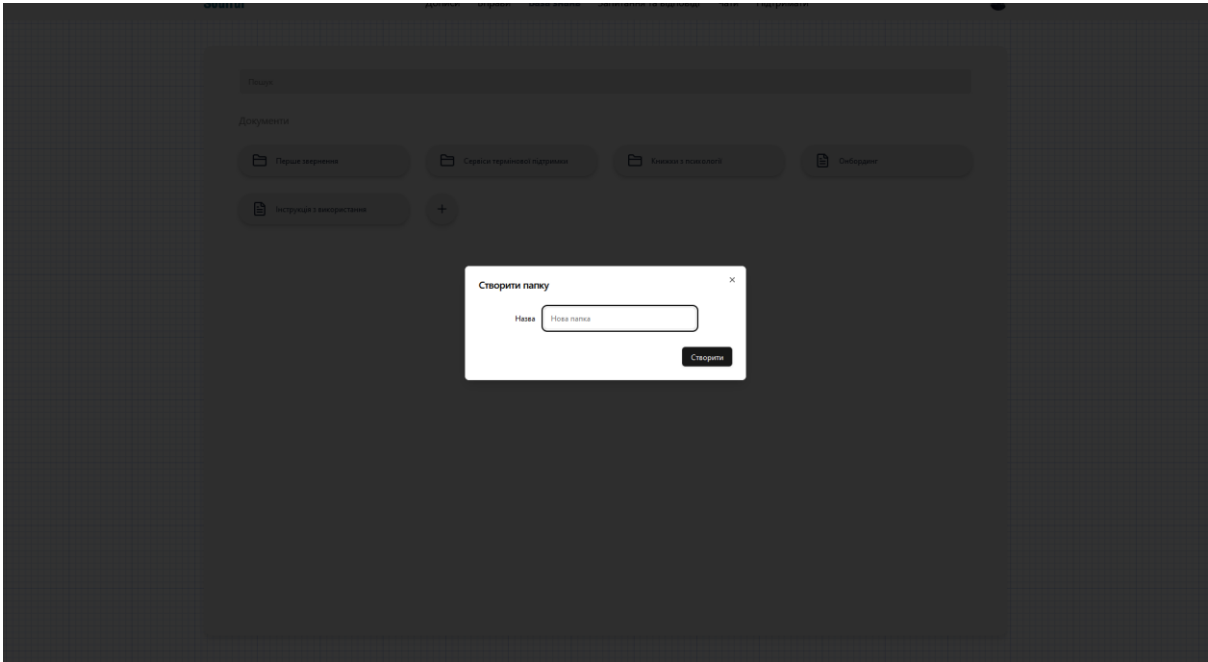


Рисунок 3.23. Створення нової сутності в модулі «База знань».

На рисунку 3.23 зображено вікно створення нової папки або файлу, яке відкривається за допомогою контекстного меню або при кліку на кнопку «+».

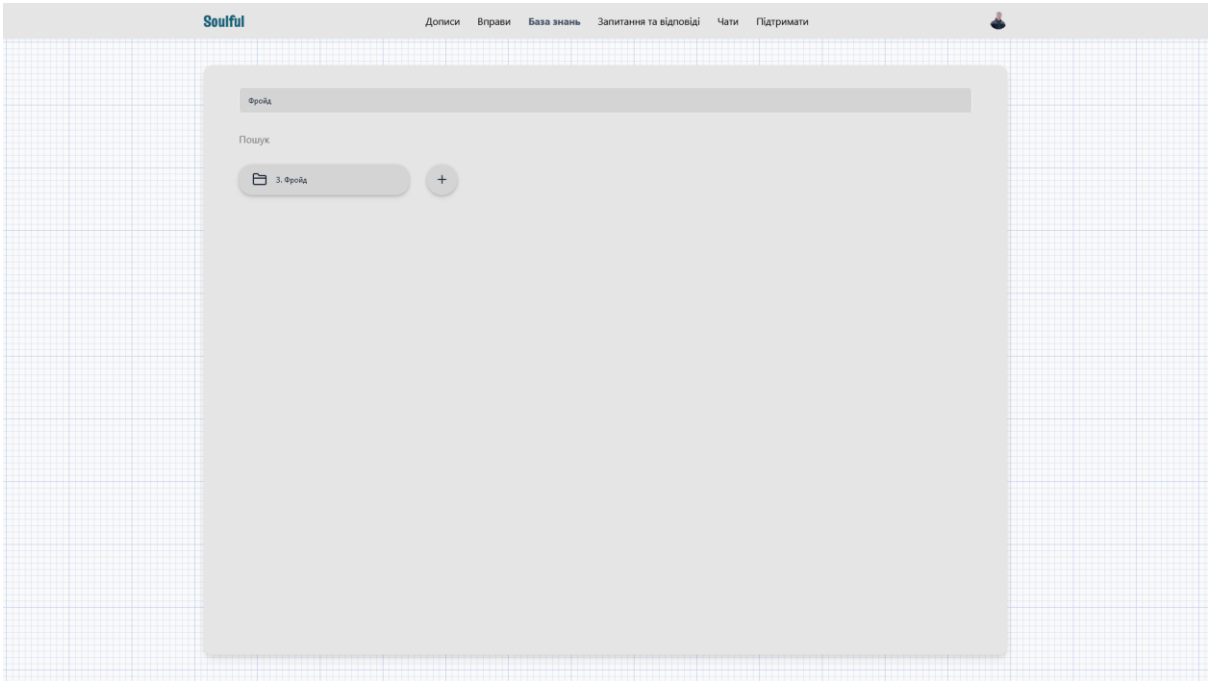


Рисунок 3.24. Пошук в модулі «База знань».

Оскільки модуль передбачається високонавантаженим, у ньому передбачено пошук (рис. 3.24). Якщо користувач вводить запит у поле пошуку з

головної папки – пошук відбувається серед усіх сутностей в базі знань. За умови пошуку в певній папці – буде відображено лише результати з обраної папки.

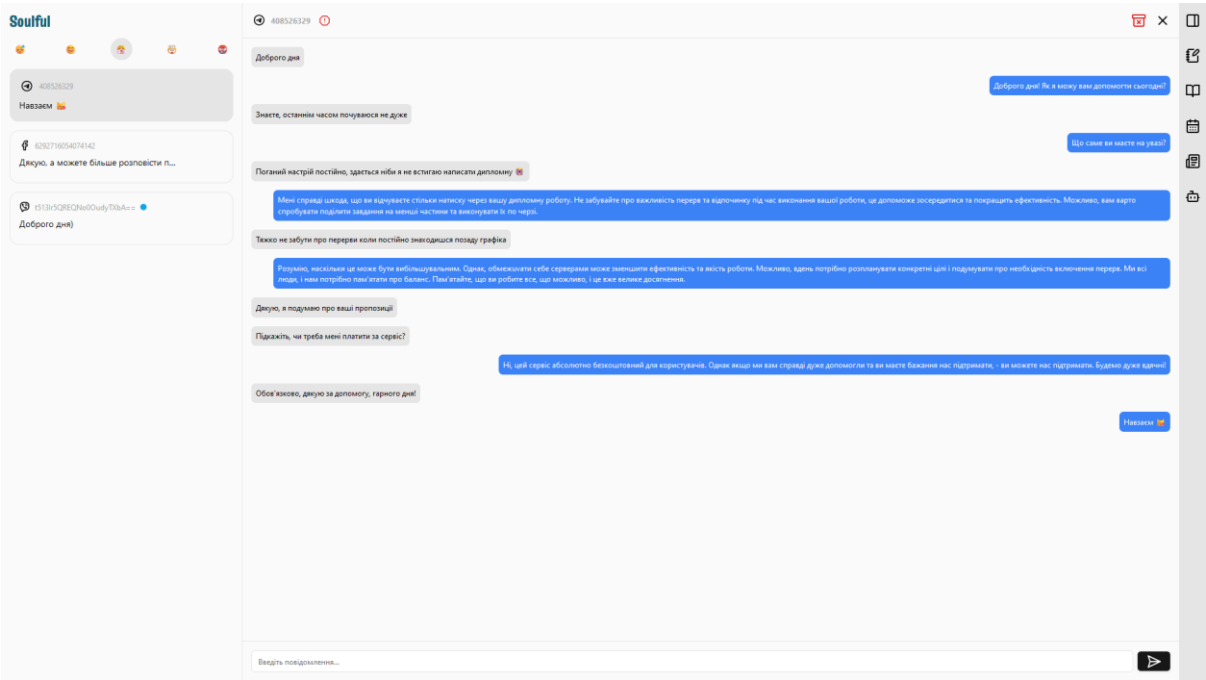


Рисунок 3.25. Чат оператора.

На рисунку 3.25 зображено зовнішній вигляд чату зі сторони оператора у застосунку. У чаті оператор отримує нові чати та повідомлення за алгоритмами, описаними у попередніх розділах.

Зліва оператор може визначити власну зайнятість, що вплине на розподілення чатів. Нижче можна побачити список активних чатів. У списку зазначено платформу та ідентифікатор користувача на ній.

Посередині сторінки відображаються повідомлення користувача (сірий колір) та оператора (синій колір). Справа знаходяться вкладки інструментів, доступних оператору.

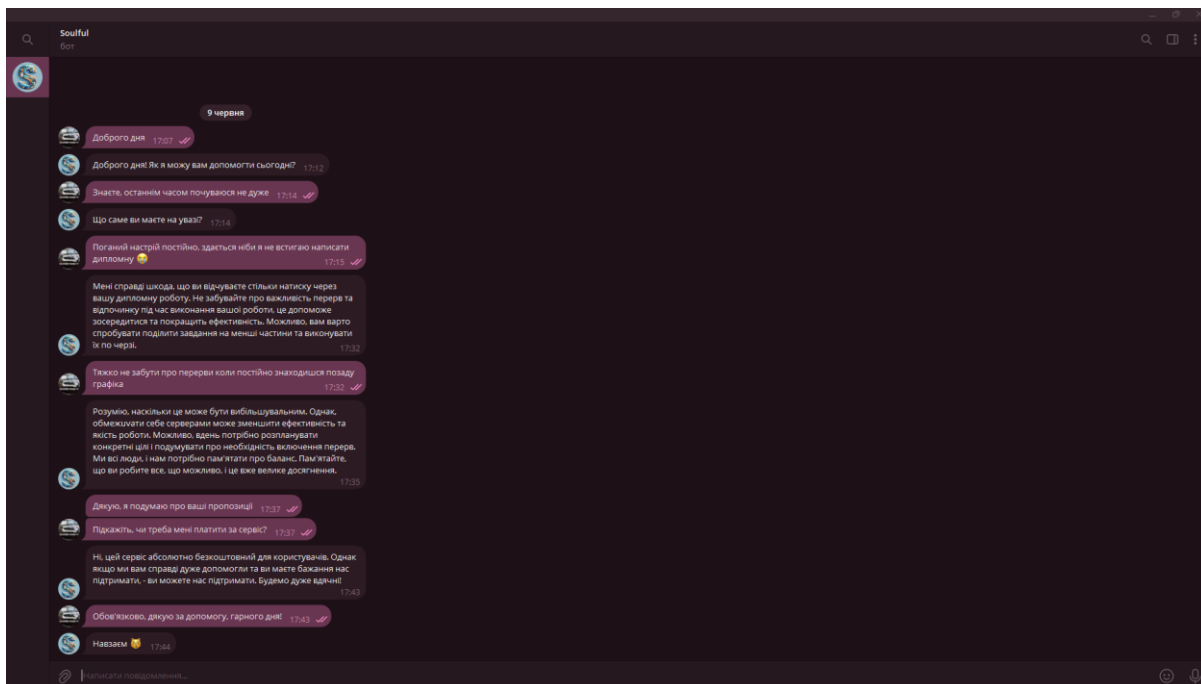


Рисунок 3.26. Чат користувача у месенджері Telegram.

Користувач може використовувати один з підтримуваних месенджерів для доступу до послух операторів системи. Для прикладу наведено спілкування за допомогою платформи Telegram (рис. 3.26).

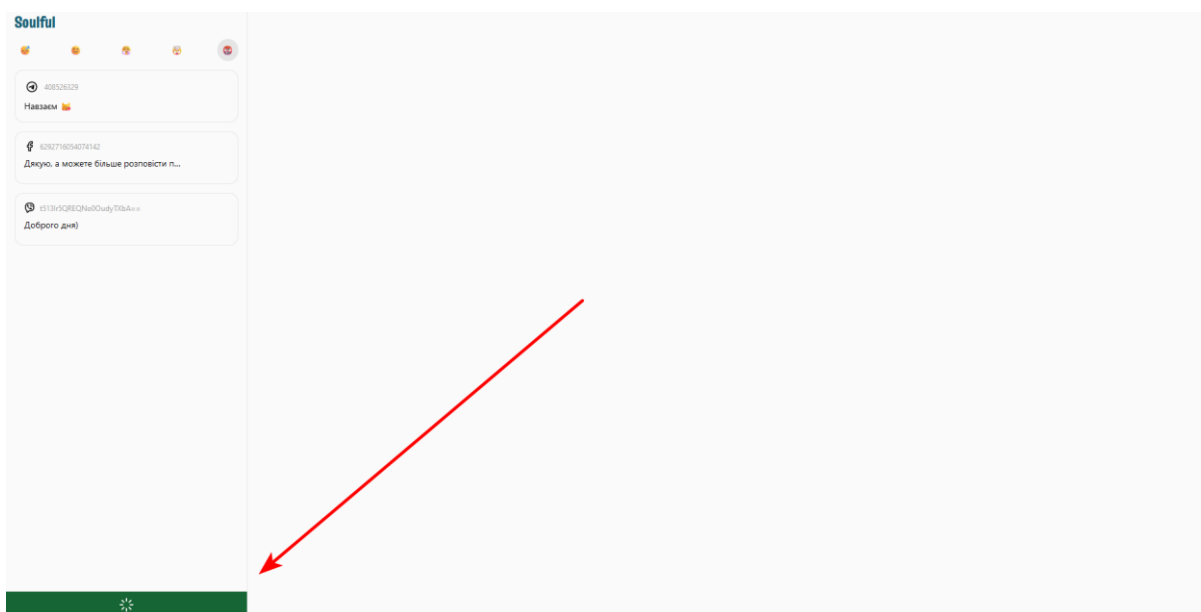


Рисунок 3.27. Індикатор-кнопка чату без оператора.

Якщо користувач звернеться за допомогою у час, коли немає операторів онлайн – усі оператори матимуть змогу отримати такий чат як тільки підключаться до чату за допомогою індикатора-кнопки, зображеного на рисунку 3.27.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

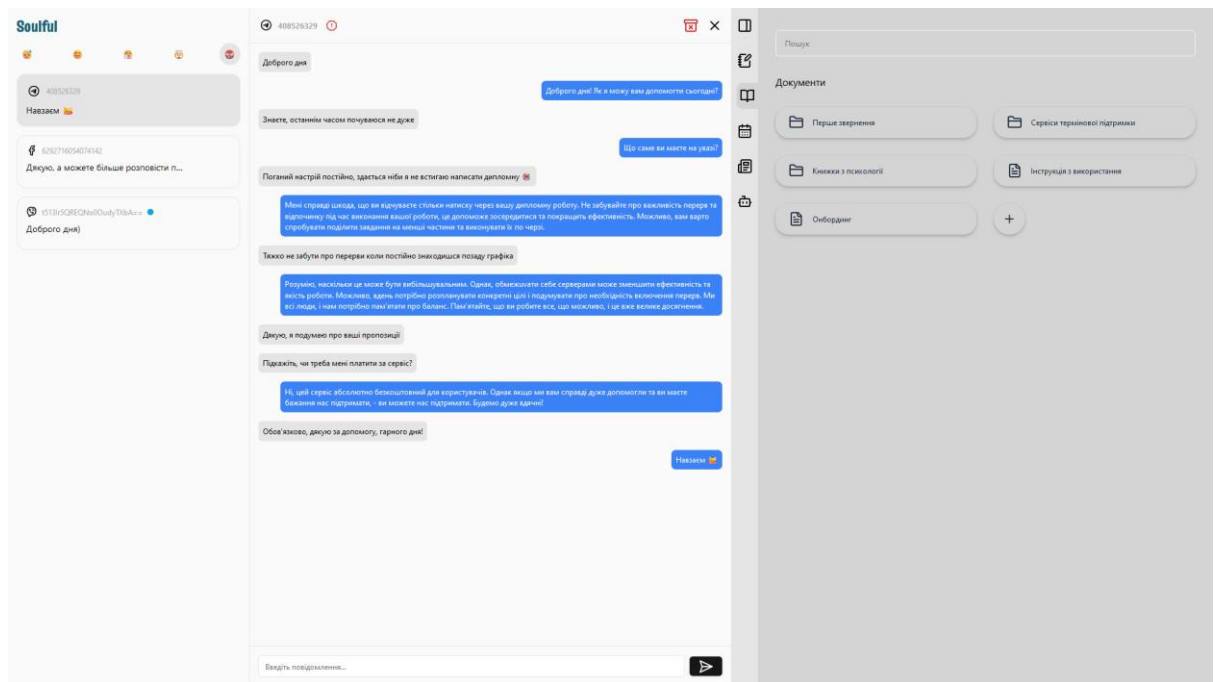


Рисунок 3.28. Модуль «База знань» в чаті.

У чаті також є доступ до бази знань (рис. 3.28). Оператор має змогу консультуватися з інструкціями та документацією за потреби.

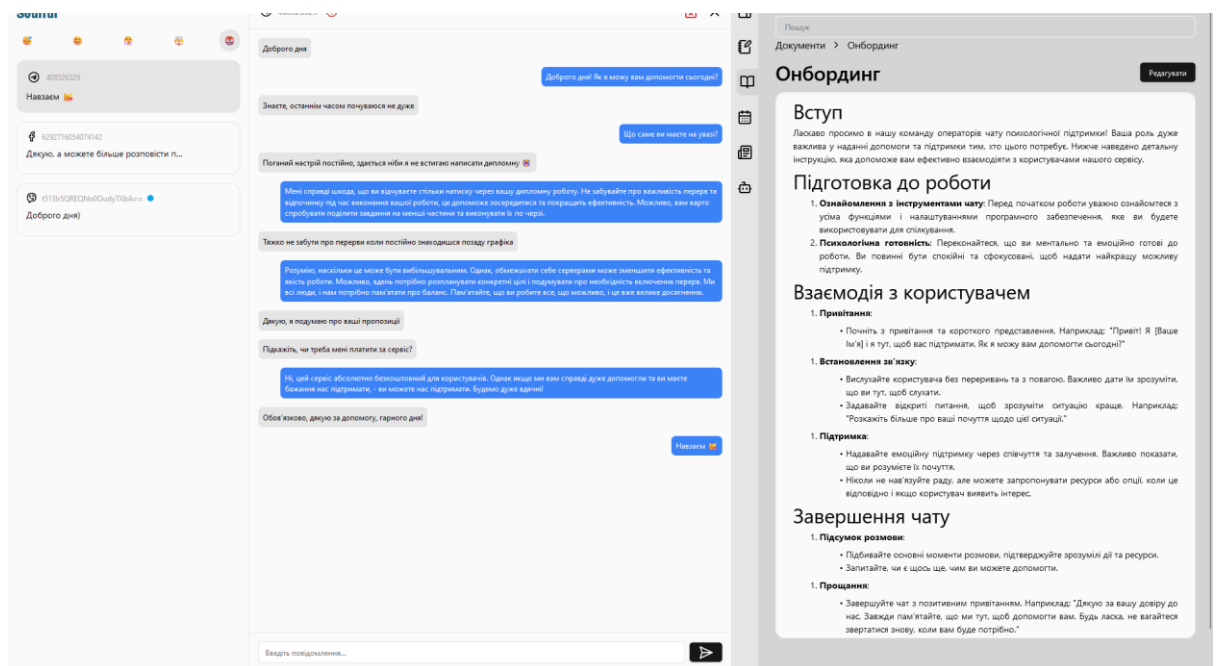


Рисунок 3.29. Документ, відкритий у чаті.

На рисунку 3.29 зображено приклад документу, який оператор може відкрити та перечитати.

		Каліновський Д.					Арк.
		Савицький Р.С.					102
Змн.	Арк.	№ докум.	Підпис	Дата	ІПЗ.КР.Б – 121 – 24 – ПЗ		

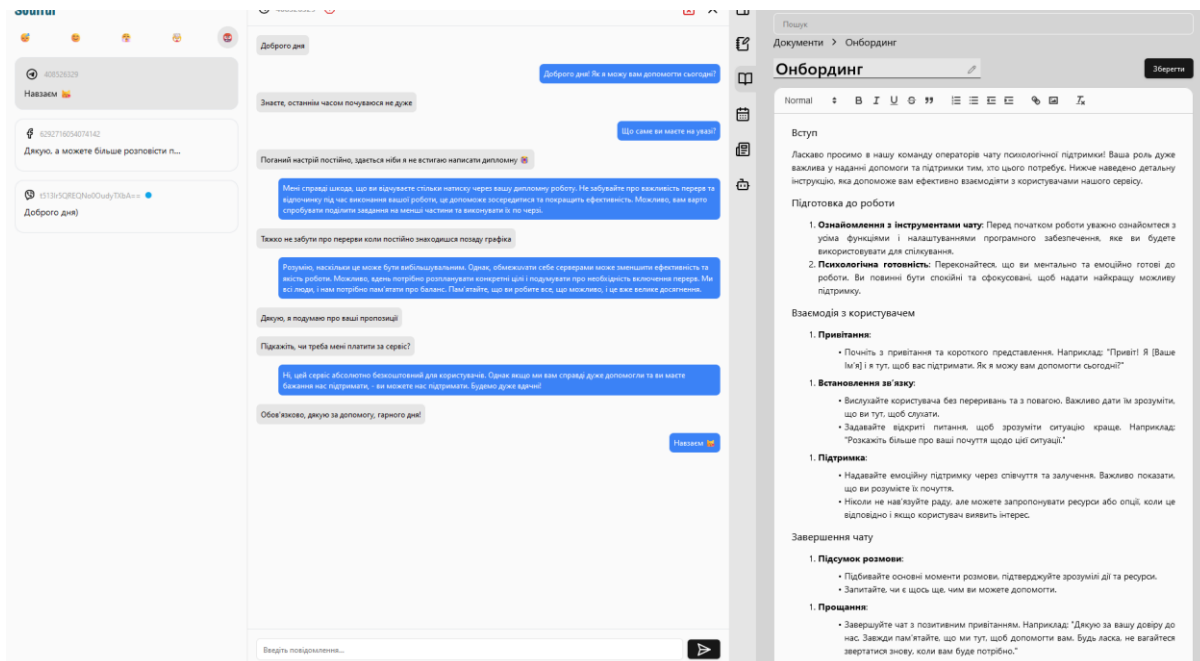


Рисунок 3.30. Редагування документа в базі знань.

Оператор має повний доступ до бази знань, включно з редагуванням. На рисунку 3.30 приведено приклад редагування документа під час спілкування з користувачем.

Отже, описано основний інтерфейс системи адміністрування застосунку психологічної підтримки, включно з сторінками користувачів, дописів, рекомендацій, вправ і питань та відповідей на панелі керування, бази знань, чату психологічної підтримки та реалізованих в рамках роботи інструментів.

3.3 Тестування системи адміністрування застосунку психологічної підтримки

У роботі застосунку критичною є стабільність та відсутність помилок у роботі. Для контролю високої якості створеного програмного забезпечення проведено тестування.

		Каліновський Д.				ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.					103
Змн.	Арк.	№ докум.	Підпис	Дата			

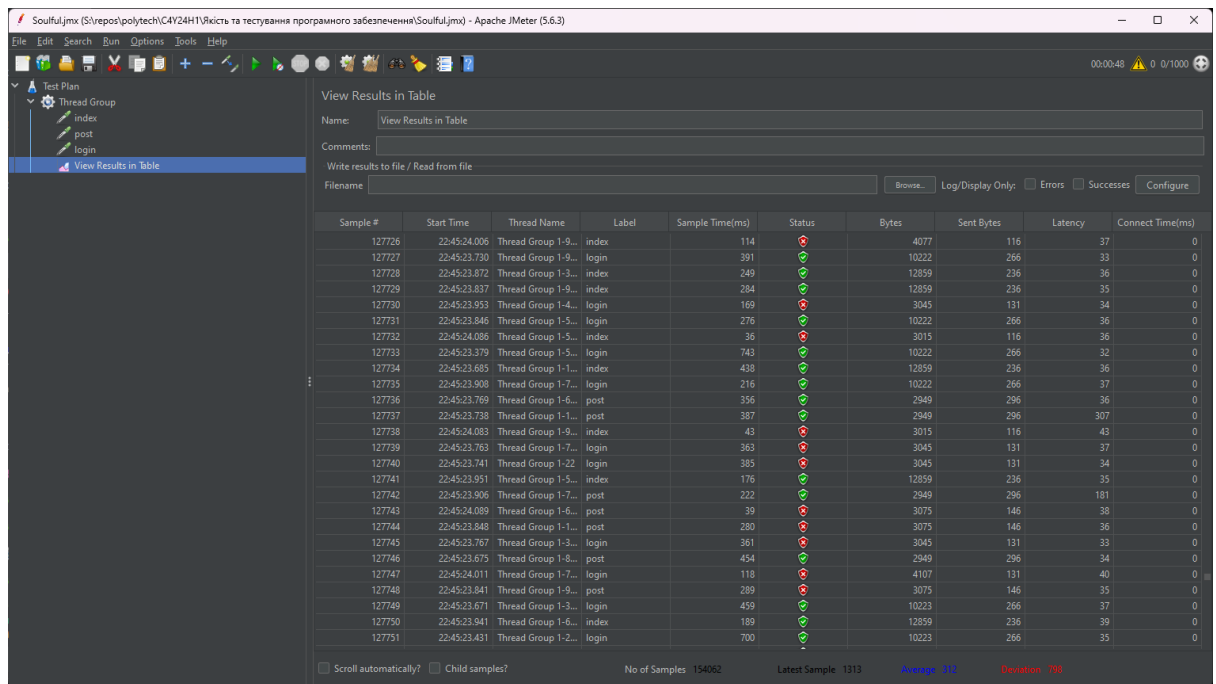


Рисунок 3.31. Стрес-тест застосунку за допомогою JMeter.

В першу чергу протестовано стабільність системи за допомогою тестування навантаженням (рис. 3.31). На рисунку видно, що система витримала десятки тисяч запитів за короткий час (більше ста тисяч запитів за десять секунд). Після довгого навантаження, доступ до застосунку було автоматично обмежено, однак він повернувся до нормальної роботи через кілька секунд після зниження навантаження. Така поведінка зумовлена аномальністю такої кількості запитів до системи. У випадку, якби трафік довгий час збільшувався невеликими інкрементами – система витримала б значно більше навантаження через роботу на інфраструктурі AWS Lambda.

Таблиця 3.1

Тестові випадки

Опис	Кроки	Вхідні дані	Очікуваний результат	Фактичний результат	Результат
Створення облікового запису користувача	1. Відкрити сторінку користувачів на панелі керування 2. Натиснути кнопку «Новий користувач» 3. Заповнити поля вхідними даними. 4. Натиснути «Створити»	Ім'я: Текст довжиною у 100 символів Електронна пошта: ipzk211_kdyu@student.ztu.edu.ua	1. Створення облікового запису 2. Отримання листа на електронний пошті	Успішне створення облікового запису.	Успіх

Продовження таблиці 3.1

Пошук користувача	1. Відкрити сторінку користувачів на панелі керування 2. Ввести запит 3. Зачекати на автоматичне спрацювання пошуку	Текст довжиною у 5 символів	Запис знайдено	Успішне віднаходження облікового запису.	Успіх
Вхід в обліковий запис	1. Відкрити сторінку входу в застосунок 2. Ввести дані 3. Натиснути «увійти»	Електронна пошта: ipzk211_kdyu@student.ztu.edu.ua Пароль: отримано у електронному листі після створення облікового запису	Успішний вхід до облікового запису.	Успішний вхід до облікового запису.	Успіх
Створення та редагування допису	1. Відкрити сторінку дописів на панелі керування 2. Натиснути кнопку «Новий допис» 3. Заповнити поля вхідними даними. 4. Натиснути «Створити» 5. Обрати створений запис, натиснути три крапки напроти нього. 6. У меню обрати «Редагувати». 7. Змінити 1 символ кожного поля на довільний. 8. Натиснути «редагувати»	Текст довжиною у 100 символів Текст довжиною у 1000 символів	Допис успішно створено та відредаговано	Допис успішно створено/відредаговано	Успіх

Продовження таблиці 3.1

Пошук допису за описом та назвою	1. Відкрити сторінку дописів на панелі керування 2. Ввести запит 3. Зачекати на автоматичне спрацювання пошуку	Текст довжиною у 5 символів	Допис знайдено	Успішне віднаходження допису	Успіх
Публікування та приховування допису	1. Обрати довільний допис, натиснути три крапки напроти нього. 2. У меню обрати приховати. 3. Перевірити, що запис приховано. 4. Обрати довільний допис, натиснути три крапки напроти неї. 5. У меню обрати опублікувати.	-	Запис успішно опубліковано та приховано	Запис успішно опубліковано/приховано	Успіх
Створення та редагування вправи	1. Відкрити сторінку користувачів на панелі керування 2. Натиснути кнопку «Нова вправа» 3. Заповнити поля головної сторінки вправи вхідними даними. 4. Заповнити один крок вправи вхідними даними. 5. Натиснути «Створити» 6. Обрати створену вправу, натиснути три крапки напроти неї. 6. У меню обрати «Редагувати». 7. Змінити 1 символ кожного поля на довільний. 8. Натиснути «редагувати»	4 тексти довжиною у 100 символів	Вправу успішно створено та відредаговано	Вправу успішно створено/відредаговано	Успіх

Продовження таблиці 3.1

Пошук вправи за описом та назвою	1. Відкрити сторінку вправ на панелі керування 2. Ввести запит 3. Зачекати на автоматичне спрацювання пошуку	Текст довжиною у 5 символів	Вправу знайдено	Успішне віднаходження вправи	Успіх
Публікування та приховування вправи	1. Обрати довільну вправу, натиснути три крапки напроти неї. 2. У меню обрати приховати. 3. Перевірити, що запис приховано. 4. Обрати довільну вправу, натиснути три крапки напроти неї. 5. У меню обрати опублікувати.	-	Вправу успішно опубліковано та приховано	Вправу успішно опубліковано/приховано	Успіх
Створення та редагування рекомендації	1. Відкрити сторінку рекомендацій на панелі керування 2. Натиснути кнопку «Нова рекомендація» 3. Заповнити поля вхідними даними. 4. Натиснути «Створити» 5. Обрати створену рекомендацію, натиснути три крапки напроти неї. 6. У меню обрати «Редагувати». 7. Змінити 1 символ кожного поля на довільний. 8. Натиснути «редагувати»	2 тексти довжиною у 100 символів	Рекомендацію успішно створено та відредаговано	Рекомендацію успішно створено/відредаговано	Успіх
Пошук рекомендації за описом та назвою	1. Відкрити сторінку рекомендацій на панелі керування 2. Ввести запит 3. Зачекати на автоматичне спрацювання пошуку	Текст довжиною у 5 символів	Рекомендацію знайдено	Успішне віднаходження рекомендації	Успіх

Продовження таблиці 3.1

Публікування та приховування рекомендації	1. Обрати довільну рекомендацію, натиснути три крапки напроти неї. 2. У меню обрати приховати. 3. Перевірити, що запис приховано. 4. Обрати довільну рекомендацію, натиснути три крапки напроти неї. 5. У меню обрати опублікувати.	-	Рекомендації успішно опубліковано та приховано	Рекомендації успішно опубліковано/приховано	Успіх
Створення та редагування питання та відповіді	1. Відкрити сторінку питань та відповідей на панелі керування 2. Натиснути кнопку «Нове питання» 3. Заповнити поля вхідними даними. 4. Натиснути «Створити» 5. Обрати створений запис, натиснути три крапки напроти нього. 6. У меню обрати «Редагувати». 7. Змінити 1 символ кожного поля на довільний. 8. Натиснути «редагувати»	2 тексти довжиною у 100 символів	Питання та відповідь успішно створено та відредаговано	Питання та відповідь успішно створено/відредаговано	Успіх
Пошук питання та відповіді за наповненням	1. Відкрити сторінку питань та відповідей на панелі керування 2. Ввести запит 3. Зачекати на автоматичне спрацювання пошуку	Текст довжиною у 5 символів	Питання та відповідь знайдено	Успішне віднаходження питання та відповіді	Успіх

Продовження таблиці 3.1

Публікування та приховування питання та відповіді	1. Обрати довільне питання, натиснути три крапки напроти неї. 2. У меню обрати приховати. 3. Перевірити, що запис приховано. 4. Обрати довільне питання, натиснути три крапки напроти неї. 5. У меню обрати опублікувати.	-	Питання та відповідь успішно опубліковано та приховано	Питання та відповідь успішно опубліковано/приховано	Успіх
Створення, видалення файлу та папки	1. Відкрити сторінку бази знань 2. Натиснути праву кнопку 3. Обрати «створити папку» 4. Ввести вхідні дані у вікно 5. Натиснути «створити» 6. Повторити кроки 2-5 для файлу 7. Натиснути праву кнопку на щойно створену папку 8. Обрати «Видалити» 9. Повторити кроки 7-8 для файлу	Текст довжиною у 15 символів	Файл та папку успішно створено та видалено	Файл та папку успішно створено/видалено	Успіх
Пошук файлів, папок	1. Відкрити сторінку бази знань 2. Ввести запит 3. Зачекати на автоматичне спрацювання пошуку	Текст довжиною у 5 символів	Рекомендацію знайдено	Успішне віднаходження рекомендації	Успіх
Редагування файлу	1. Обрати довільний файл, натиснути на нього правою кнопкою 2. Обрати «перейменувати» 3. Натиснути «Enter»	Текст довжиною у 1000 символів	Файл успішно відредаговано	Файл успішно відредаговано	Успіх
Відправка повідомлень зі сторони користувача, є оператори онлайн	1. Підключитися до чату під записом оператора 2. Увійти до підтримуваного месенджера 3. Обрати чат-бот застосунку 4. Написати та відправити повідомлення	Текст довжиною у 100 символів	Повідомлення отримано оператором	Повідомлення отримано оператором	Успіх

Продовження таблиці 3.1

Відправка повідомлень зі сторони користувача, немає операторів онлайн	1. Увійти до підтримуваного месенджера 2. Обрати чат-бот застосунку 3. Написати та відправити повідомлення	Текст довжиною у 100 символів	Повідомлення отримано. При наступному підключенні оператор отримав цей чат	Повідомлення отримано. При наступному підключенні оператор може отримати цей чат	Успіх
Відправка повідомлень зі сторони оператора	1. Підключитися до чату під записом оператора 2. Обрати один з тестових чатів 4. Написати та відправити повідомлення	Текст довжиною у 100 символів	Повідомлення отримано користувачем	Повідомлення отримано користувачем	Успіх
Архівація чату	1. Підключитися до чату під записом оператора 2. Обрати тестовий чат 3. Натиснути кнопку архівації	-	Чат заархівовано	Чат заархівовано	Успіх
Відправка повідомлень користувачем після архівації чату.	1. Підключитися до чату під записом оператора 2. Увійти до підтримуваного месенджера 3. Обрати чат-бот застосунку 4. Написати та відправити повідомлення	Текст довжиною у 100 символів	Чат розархівовано, оператор бачить повну історію повідомлень	Чат розархівовано, оператор бачить повну історію повідомлень	Успіх

Додатково проведено повне тестування системи користувачами, в ході якого виявлено 7 невеликих недоліків.

Проведено тестування навантаженням на розгорнуту систему, а також мануальне тестування критичного функціоналу. Виявлені недоліки виправлено у вигляді фінального оновлення обох застосунків системи. Проведено повторне регрес-тестування та не виявлено жодних додаткових помилок, відповідно тестування завершено успішно.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				110
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до третього розділу

Проаналізувавши процеси розробки, налаштування та розгортання системи адміністрування застосунку психологічної підтримки, були детально розглянуті ключові аспекти, важливі для забезпечення стабільності й ефективності роботи системи. Локальний режим розробки та повноцінний режим роботи дозволяють оптимізувати процеси впровадження та подальшої експлуатації застосунку. Використання таких інструментів як Docker і AWS Lambda сприяє забезпеченню масштабованості та безпеки, що є критично важливим для підтримки стабільної роботи в реальному часі.

Додатково, у розділі було розглянуто важливість зручної адміністративної панелі, яка дозволяє легко керувати користувачами та контентом без потреби залучення розробників. Функціональність системи, що забезпечує детальний пошук та можливість налаштування прав доступу, покращує управління ресурсами і сприяє збільшенню продуктивності операторів та адміністраторів. Такий підхід не лише забезпечує ефективність роботи застосунку, але й підвищує загальне задоволення користувачів.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				111
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В ході дослідження процесів адміністрування вебзастосунків психологічної підтримки було визначено необхідність створення рішень, що відповідають актуальним вимогам у цій галузі. Розроблений застосунок стане ефективним інструментом для підвищення якості психологічної підтримки, забезпечуючи користувачам ергономічний інтерфейс і інтеграцію різноманітних платформ комунікації. Реалізація автоматизованого чату та системи управління сесіями, реєстрації користувачів та публікації наповнення відіграють ключову роль у забезпеченні інтуїтивно зрозумілої взаємодії та високого рівня особистісного підходу у наданні психологічної підтримки. Завдяки адаптивності та масштабованості системи, забезпечується можливість швидкого підлаштування до змінюваних вимог користувачів та удосконалення сервісу відповідно до нових тенденцій та потреб цільової аудиторії.

Підкреслено значимість проєкту в контексті зростаючого попиту на психологічні послуги та важливість інтеграції новітніх інформаційних технологій для оптимізації взаємодії між користувачами та фахівцями.

Також реалізація системи адміністрування забезпечує високу ефективність обробки даних, зменшення часу відповіді на запити користувачів, та підвищує загальну доступність сервісу. Це сприяє покращенню ментального здоров'я користувачів за допомогою забезпечення надійного і постійного доступу до кваліфікованих психологічних ресурсів.

Впровадження розробленої системи сприяє посиленню ролі сучасних та безпечних технологій в галузі психологічної допомоги та створенню стандартів для майбутніх розробок в цій області. Це, в свою чергу, слугує стимулом для подальшого розвитку проєктів та підвищення загальної ефективності послуг у цій галузі на національному рівні. Завдяки аналізу поточних та потенційних потреб користувачів, проєкт не лише відповідає існуючим викликам, а й пропонує стратегічний підхід до їх вирішення, сприяючи більшій інтегрованості та систематизації психологічної підтримки.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				112
Змн.	Арк.	№ докум.	Підпис	Дата		

Завершення кваліфікаційної роботи відзначається досягненням поставлених цілей, а також відкриває перспективи для подальших досліджень та розвитку. Під час виконання роботи на стороні тиловиці реалізовано ролі користувачів, що об'єднують дозволи, таким чином дозволяючи адміністрації налаштовувати доступ цілих груп користувачів. Для наповнення передбачено впровадження тегів – коротких характеристик сутностей, невелику кількість яких можна прив'язати до допису, вправи чи іншої сутності, покращивши зрозумілість цілей та короткого змісту відповідного об'єкта у системі. Передбачено систему аудиту, що створює перелік дій кожного користувача у системі, підвищуючи безпеку та відкриваючи горизонти звітності, доступні лише у найкращих застосунках серед доступних.

Такі кроки дозволять зробити систему ще більш комплексною зсередини, але зрозумілішою, ергономічнішою та надійнішою ззовні, адаптованою до різноманітних потреб користувачів. Згадане забезпечить зростання якості та доступності психологічної допомоги.

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				113
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Каліновський Д.Ю., Чернюк А. М., Савіцький Р.С. ІНТЕГРАЦІЯ МУЛЬТИПЛАТФОРМНИХ МЕСЕНДЖЕРІВ З ВИКОРИСТАННЯМ FASTAPI ТА WEBSOCKETS ДЛЯ СТВОРЕННЯ УНІФІКОВАНОЇ КОМУНІКАЦІЙНОЇ ПЛАТФОРМИ. Тези XIV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології – 2024», 28-29 березня 2024 р. Житомир: ДУ «Житомирська Політехніка», 2024. С. 13.
2. Чернюк А. М., Каліновський Д.Ю., Савіцький Р.С. ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В СИСТЕМАХ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ. Тези XV міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології», 25 – 26 квітня 2024. Київ: «НАУ», 2024. С. 169.
3. Savitskyi R., Kukharonok S., Kalinovskiy D. UNVEILING AI's ILLUSION OF KNOWLEDGE. Current Trends in Young Scientists' Research: XI All Ukrainian Scientific and Practical Conference, April 25, 2024. Zhytomyr: Zhytomyr Polytechnic State University, 2024. P. 153.
4. Незалежний портал безкоштовних послуг Help24 [Електронний ресурс] – Режим доступу до ресурсу: <https://help24.org.ua/uk/>
5. Розкажи мені. Ваша підтримка поруч [Електронний ресурс] – Режим доступу до ресурсу: <https://tellme.com.ua/>
6. Тепло – психолог онлайн [Електронний ресурс] – Режим доступу до ресурсу: <https://teplo.app/>
7. UA Mental Help – Безкоштовна психологічна допомога [Електронний ресурс] – Режим доступу до ресурсу: <https://uamentalhelp.org/mental-help/>
8. What Is tRPC and Why Should You Use It? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.makeuseof.com/trpc-what-why-use/>
9. Microservices vs. monolithic architecture [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith>

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				114
Змн.	Арк.	№ докум.	Підпис	Дата		

10. FastAPI Websockets [Електронний ресурс] – Режим доступу до ресурсу:
<https://fastapi.tiangolo.com/advanced/websockets/>
11. How to Build a Fullstack App with Next.js, Prisma, and Vercel Postgres [Електронний ресурс] – Режим доступу до ресурсу:
<https://vercel.com/guides/nextjs-prisma-postgres>
12. Create T3 App template documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://create.t3.gg/en/introduction>
13. JavaScript Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/javascript>
14. TypeScript Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>
15. ESLint Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://eslint.org/docs/latest/>
16. Prettier Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://prettier.io/docs/en/>
17. Husky GitHub Repository [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/typicode/husky>
18. Node.js documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>
19. Bun documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://bun.sh/docs>
20. Documentation for the npm registry, website, and command-line interface [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.npmjs.com/>
21. Zod Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://zod.dev/>
22. Zod-i18n GitHub Repository [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/aiji42/zod-i18n>
23. Validator.js GitHub Repository [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/validatorjs/validator.js/>

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				115
Змн.	Арк.	№ докум.	Підпис	Дата		

- 24.React Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/reference/react>
- 25.Next.JS Documantation [Електронний ресурс] – Режим доступу до ресурсу:
<https://nextjs.org/docs>
- 26.Tailwindcss Documentation [Електронний ресурс] – Режим доступу до
ресурсу: <https://tailwindcss.com/docs/>
- 27.Radix UI Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.radix-ui.com/themes/docs/overview/getting-started>
- 28.Shadcn/UI Documentation [Електронний ресурс] – Режим доступу до
ресурсу: <https://ui.shadcn.com/docs>
- 29.React Charts Documentation [Електронний ресурс] – Режим доступу до
ресурсу: <https://react-charts.tanstack.com/docs/overview>
- 30.Date-fns Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://date-fns.org/docs/Getting-Started>
- 31.Use-sound GitHub Repository [Електронний ресурс] – Режим доступу до
ресурсу: <https://github.com/joshwcomeau/use-sound>
- 32.Prisma Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.prisma.io/docs>
- 33.NextAuth.js Documentation [Електронний ресурс] – Режим доступу до
ресурсу: <https://next-auth.js.org/getting-started/introduction>
- 34.tRPC Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://trpc.io/docs>
- 35.AWS SDK for JavaScript v3 Documentation [Електронний ресурс] – Режим
доступу до ресурсу:
<https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/>
- 36.AWS CLI v2 Documentation [Електронний ресурс] – Режим доступу до
ресурсу: <https://docs.aws.amazon.com/cli/>
- 37.Python Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.python.org/3/>

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				116
Змн.	Арк.	№ докум.	Підпис	Дата		

- 38.PyPI User Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.pypi.org/>
- 39.Poetry Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://python-poetry.org/docs/>
- 40.FastAPI Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://fastapi.tiangolo.com/>
- 41.Pydantic Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.pydantic.dev/2.0/migration/>
- 42.SQLAlchemy Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.sqlalchemy.org/en/20/>
- 43.AWS SDK for Python (Boto3) documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>
- 44.PostgreSQL 16.3 Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/16/index.html>
- 45.Jetbrains WebStorm Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/help/webstorm/meet-webstorm.html>
- 46.Jetbrains PyCharm Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/help/pycharm/getting-started.html>
- 47.GitLab Runner Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.gitlab.com/runner/>
- 48.GitLab CI/CD pipelines Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.gitlab.com/ee/ci/pipelines/>
- 49.Docker Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/>
- 50.Why PostgreSQL Is the Bedrock for the Future of Data [Електронний ресурс] – Режим доступу до ресурсу: <https://www.timescale.com/blog/postgres-for-everything/>
- 51.PgAdmin Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.pgadmin.org/docs/>

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савицький Р.С.				117
Змн.	Арк.	№ докум.	Підпис	Дата		

- 52.Many-to-many relations in Prisma [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/prisma-schema/data-model/rerelations/many-to-many-relations>
- 53.CUID Github Repository [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/paralleldrive/cuid>
- 54.Differences Between Role and Permission-based Access Control [Електронний ресурс] – Режим доступу до ресурсу: <https://www.baeldung.com/cs/role-vs-permission-based-access-control>
- 55.How Figma’s databases team lived to tell the scale [Електронний ресурс] – Режим доступу до ресурсу: <https://www.figma.com/blog/how-figmas-databases-team-lived-to-tell-the-scale/>
- 56.OpenAI API Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://platform.openai.com/docs/overview>

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				118
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				119
Змн.	Арк.	№ докум.	Підпис	Дата		

User Router

Файл src/server/api/routers/user.ts:

```

import {
  createTRPCRouter,
  multilevelPermissionProcedure,
  permissionProcedure,
  protectedProcedure,
  publicMultilevelPermissionProcedure,
} from '~/server/api/trpc';
import {
  CreateUserSchema,
  SearchUsersSchema,
  SetNotesSchema,
  SetBooleanSchema,
  StringIdSchema,
  UpdateUserSchema,
  ChangePasswordSchema,
  SelfUpdateUserSchema,
} from '~/utils/schemas';
import { archiveChat } from '~/server/api/routers/common';
import { randomUUID } from 'crypto';
import bcrypt from 'bcrypt';
import { env } from '~/env';
import {
  sendChangeEmailEmail,
  sendPasswordUpdateEmail,
  sendRegEmail,
} from '~/utils/email/templates';
import { AccessType, SearchableUserFields } from '~/utils/types';
import { getAccessTypeFromContext } from '~/utils/auth';

export const userRouter = createTRPCRouter({
  list: multilevelPermissionProcedure
    .input(SearchUsersSchema)
    .query(
      async ({
        input: { page, limit, query, orderBy, order,
permissions },
        ctx,
      }) => {
        const p = ['*:*', '*:*:*'];
        for (const permission of permissions || []) {
          p.push(permission);
          p.push(`${permission}:*`);
        }

        let where: Record<string, object> | undefined =
permissions && {
          OR: [
            {

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        permissions: {
            some: {
                title: {
                    in: p,
                },
            },
        },
    },
    {
        roles: {
            some: {
                permissions: {
                    some: {
                        title: {
                            in: p,
                        },
                    },
                },
            },
        },
    },
],
};

const contains = {
    contains: query,
    mode: 'insensitive',
};

const containsQuery = {
    OR:
Object.values(SearchableUserFields).map((field) => ({
    [field]: contains,
}))),
};

if (query) {
    where
        ? (where = {
            AND: [where, containsQuery],
        })
        : (where = containsQuery);
}

const [count, values] = await
ctx.db.$transaction([
    ctx.db.user.count({ where }),
    ctx.db.user.findMany({
        where,
        select: getProjection(ctx.isFullAccess),
        orderBy: {
            [orderBy ? orderBy : 'createdAt']:
order

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				121
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ? order
        : 'desc',
    },
    skip: (page - 1) * limit,
    take: limit,
  )),
]);

return {
  count,
  values,
};
},
),

get: publicMultilevelPermissionProcedure
  .input(StringIdSchema.optional())
  .query(async ({ input, ctx }) => {
    const accessType = getAccessTypeFromContext(ctx);
    if (
      input !== ctx.session?.user?.id &&
      accessType === AccessType.NONE
    ) {
      throw new Error('Access denied');
    }

    const id = input || ctx.session?.user?.id;

    if (!id) {
      throw new Error('User not found');
    }

    return ctx.db.user.findUnique({
      where: {
        id,
      },
      select: getProjection(accessType ===
AccessType.ALL),
    });
  )),

  getAccessToken: protectedProcedure.query(async ({ ctx }) => {
    return (
      await ctx.db.session.findFirst({
        where: {
          userId: ctx.session.user.id,
          expires: {
            gte: new Date(),
          },
        },
        select: {
          sessionToken: true,
        },
      })
    );
  })
);

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				122
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        })
    )?.sessionToken;
  }},

  create: permissionProcedure
    .input(CreateUserSchema)
    .mutation(async ({ ctx, input: data }) => {
      const password = randomUUID();

      const user = await ctx.db.user.create({
        data: {
          ...data,
          password: await bcrypt.hash(password,
env.SALT_ROUNDS),
          isOAuth: false,
        },
      });

      await sendRegEmail(data.email, data.name, password,
ctx.host);

      return user;
    }),

  notes: permissionProcedure
    .input(SetNotesSchema)
    .mutation(async ({ ctx, input: { id, notes } }) => {
      return ctx.db.user.update({
        where: {
          id,
        },
        data: {
          notes,
        },
        select: {
          notes: true,
        },
      });
    }),

  selfUpdate: protectedProcedure
    .input(SelfUpdateUserSchema)
    .mutation(async ({ ctx, input: data }) => {
      return ctx.db.user.update({
        where: {
          id: ctx.session.user.id,
        },
        data,
      });
    }),

  update: permissionProcedure
    .input(UpdateUserSchema)

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				123
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .mutation(async ({ ctx, input: { id, ...data } }) => {
            const user = await ctx.db.user.findUnique({
                where: {
                    id,
                },
                select: {
                    isOAuth: true,
                    name: true,
                    email: true,
                },
            });

            if (!user) {
                throw new Error('Користувач не знайдений');
            }

            if (data.email && user.email && data.email !==
user.email) {
                if (user.isOAuth) {
                    throw new Error(
пошти для OAuth користувача',
                    );
                }

                await sendChangeEmailEmail(
                    user.name ?? 'Анонім',
                    user.email,
                    data.email,
                );
            }

            return ctx.db.user.update({
                where: {
                    id,
                },
                data,
            });
        }),

resetPassword: permissionProcedure
    .input(StringIdSchema)
    .mutation(async ({ ctx, input: id }) => {
        const checkUser = await ctx.db.user.findUnique({
            where: {
                id,
            },
            select: {
                password: true,
            },
        });

        if (!checkUser || !checkUser.password) {

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				124
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        throw new Error('Користувач не використовує пароль
для входу');
    }

    const password = randomUUID();

    const user = await ctx.db.user.update({
        where: {
            id,
        },
        data: {
            password: await bcrypt.hash(password,
env.SALT_ROUNDS),
        },
    });

    if (!user || !user.email) {
        throw new Error('Користувач не має електронної
пошти');
    }

    await sendPasswordUpdateEmail(
        user.email,
        user.name ?? 'Анонім',
        password,
        ctx.host,
    );

    return true;
}),

changePassword: protectedProcedure
    .input(ChangePasswordSchema)
    .mutation(async ({ ctx, input: { oldPassword, newPassword
} }) => {
        const checkUser = await ctx.db.user.findUnique({
            where: {
                id: ctx.session.user.id,
                isOAuth: false,
            },
            select: {
                password: true,
                email: true,
            },
        });

        if (!checkUser || !checkUser.password ||
!checkUser.email) {
            throw new Error('Користувач не використовує пароль
для входу');
        }
    }

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				125
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!(await bcrypt.compare(oldPassword,
checkUser.password))) {
            throw new Error('Неправильний старий пароль');
        }

        const user = await ctx.db.user.update({
            where: {
                id: ctx.session.user.id,
                isOAuth: false,
            },
            data: {
                password: await bcrypt.hash(newPassword,
env.SALT_ROUNDS),
            },
        });

        await sendPasswordUpdateEmail(
            user.email!, // checked above
            user.name ?? 'Анонім',
            newPassword,
            ctx.host,
        );
    })),

suspend: permissionProcedure
    .input(SetBooleanSchema)
    .mutation(async ({ ctx, input: { id, value } }) => {
        if (ctx.session.user.id === id) {
            throw new Error('Неможливо змінити статус власного
запису');
        }

        const chatIds = await ctx.db.chat.findMany({
            where: {
                OR: [
                    {
                        personnelId: id,
                    },
                    {
                        userId: id,
                    },
                ],
            },
            select: {
                id: true,
            },
        });

        const promises: Promise<boolean>[] = [];
        for (const chat of chatIds) {
            promises.push.archiveChat(chat.id, ctx));
        }
    }

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				126
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        await Promise.all(promises);

        return ctx.db.user.update({
            where: {
                id,
            },
            data: {
                suspended: value,
            },
        });
    })),

    delete: permissionProcedure
        .input(StringIdSchema)
        .mutation(async ({ ctx, input: id }) => {
            if (ctx.session.user.id === id) {
                throw new Error('Неможливо видалити власний
запис');
            }

            return ctx.db.user.delete({
                where: {
                    id,
                },
            });
        })),
    });

function getProjection(isFullAccess: boolean) {
    return {
        id: true,
        email: true,
        name: true,
        image: true,
        description: true,
        isOAuth: true,
        notes: isFullAccess,
        createdAt: true,
        updatedAt: true,
        reportCount: true,
        suspended: isFullAccess,
        permissions: true,
    };
}

```

ДОДАТОК Б

Клас Event та наслідувач Telegram

Файл src/event.py:

```

from pydantic import BaseModel
from typing import Union, List

```

		Каліновський Д.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Савіцький Р.С.				127
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from src.util import AbcNoPublicConstructor
from src.attachment import Attachment

from abc import abstractmethod
from fastapi import Request
from os import environ

class Event(metaclass=AbcNoPublicConstructor):
    """
    Event class for all messengers with a private constructor.
    It stores text, chat_id and original request body
    """
    original: BaseModel
    __attachments: List[Attachment] | None = None

    def __init__(self, original: BaseModel) -> None:
        if not isinstance(original, BaseModel):
            raise ValueError("original must be a pydantic
BaseModel subclass")

        self.original = original

    @property
    def user_unique_id(self) -> str:
        """
        A method that returns a unique id for user, who initiated
the event
        :return: a unique id
        """
        return f'{self.platform_name}_{self.chat_id}'

    @classmethod
    def create(cls, data: BaseModel) -> "Event":
        """
        A class method that creates an event from respective
pydantic model
        :param data: an incoming request body, already checked for
validity and in pydantic model format
        :return: an event object
        """
        if cls is Event:
            raise NotImplementedError("create should never be
called on Event directly")
        return cls._create(data)

    @property
    @abstractmethod
    def platform_name(self) -> str:
        """
        A property that returns the name of the platform
        :return: platform name

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				128
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        """
        raise NotImplementedError("platform_name is a subclass-
implemented property")

    @property
    @abstractmethod
    def chat_id(self) -> int | str:
        """
        A property that returns a chat id
        :return: a chat id
        """
        raise NotImplementedError("chat_id is a subclass-
implemented property")

    @property
    @abstractmethod
    def text(self) -> str:
        """
        A property that returns message text
        :return: a text
        """
        raise NotImplementedError("text is a subclass-implemented
property")

    @property
    @abstractmethod
    def attachments(self) -> List[Attachment]:
        """
        A property that returns message attachments
        :return: List[Attachment]
        """
        if self.__attachments is None:
            self.__attachments = self._get_attachments()

        return self.__attachments

    @abstractmethod
    def _get_attachments(self) -> List[Attachment]:
        """
        A method that gathers message attachments
        :return: List[Attachment]
        """
        raise NotImplementedError("text is a subclass-implemented
property")

    def download_attachments(self, save: bool = True) -> List[str
| bytes | None]:
        """
        A method that downloads message attachments
        :param save: if True, saves attachments to local storage
and returns paths
        :return: List[str | bytes | None]
        """

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				129
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return [attachment.download(save) for attachment in
self.attachments]

    @classmethod
    async def create_if_valid(cls, request: Request) ->
Union["Event", None]:
        """
        A class method that creates an event from json if it is
valid
:param request: an incoming request object
:return: an event object or None if request is invalid
        """
        data = await cls.is_request_valid(request)
        if data:
            return cls.create(data)
        return None

    @classmethod
    async def is_request_valid(cls, request: Request) -> BaseModel
| None:
        """
        A static method that checks if json is valid for this
event
:param request: an incoming request object
:return: True if request is valid, False otherwise
        """
        return (await cls.is_request_authentic(request)) and
cls.is_json_valid(await request.json())

    @staticmethod
    @abstractmethod
    async def is_request_authentic(request: Request) -> bool:
        """
        A static method that checks if request comes from a valid
place
:param request: an incoming request object
:return: True if request is valid, False otherwise
        """
        raise NotImplementedError("is_request_authentic is a
subclass-implemented method")

    @staticmethod
    @abstractmethod
    def is_json_valid(json_data: dict) -> BaseModel | None:
        """
        A static method that checks if json is valid for this
event
:param json_data: an incoming request body in json format
:return: True if json is valid, False otherwise
        """
        raise NotImplementedError("is_json_valid is a subclass-
implemented method")

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				130
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    @abstractmethod
    def send_message(self, text: str) -> None:
        """
        A method that sends given message to the chat in the
        messenger the event was received from
        :param text: a message to send
        :return: None
        """
        raise NotImplementedError("send_message is a subclass-
        implemented method")

class EventFactory:
    @staticmethod
    async def create_event(request: Request,
    is_message_required=True) -> Event:
        """
        A static method that decides exact class for an event and
        creates it from json
        :param request: an incoming request object
        :param is_message_required: if True, raises ValueError if
        message is not present in the request, even if event is valid
        :return an event object
        """
        for messenger in Event.__subclasses__():
            evt = await messenger.create_if_valid(request)
            if evt is None:
                continue
            if is_message_required and not evt.text:
                raise ValueError("Message is required")
            return evt

        raise ValueError("Unknown request origin")

```

Файл src/platforms/telegram/event.py:

```

import requests
from pydantic import BaseModel
from starlette.requests import Request

from src.attachment import Attachment, AttachmentType
from src.platforms.telegram import api
from src.platforms.telegram.model import Model
from src.event import Event

from os import environ, path

telegram_verification_token =
environ["TELEGRAM_VERIFICATION_TOKEN"]
telegram_token = environ["TELEGRAM_TOKEN"]

class TelegramEvent(Event):

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				131
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    original: Model # this is needed to tell pydantic that
    original is a Model

    @property
    def platform_name(self) -> str:
        return 'telegram'

    @property
    def chat_id(self) -> int | str:
        return self.original.message.chat.id

    @property
    def text(self) -> str:
        return self.original.message.text

    def _get_attachments(self) -> list:
        attachments = []

        if self.original.message.photo is None:
            return attachments

        file_id = self.original.message.photo[-1].file_id
        file_url =
f"https://api.telegram.org/bot{telegram_token}/getFile?file_id={fi
le_id}"
        response_json = requests.get(file_url).json()
        if not response_json["ok"]:
            raise ValueError('Error getting file path')
        file_path = response_json["result"]["file_path"]
        file_name = path.splitext(file_path.split('/')[-1])
        attachments.append(
            Attachment(
                name=file_name[0],
                extension=file_name[-1],
                type_=AttachmentType.Image,

url=f"https://api.telegram.org/file/bot{telegram_token}/{file_path
}"
            )
        )
        return attachments

    @staticmethod
    async def is_request_authentic(request: Request) -> bool:
        try:
            verification_token = request.headers.get("X-Telegram-
Bot-API-Secret-Token")
            if telegram_verification_token == verification_token:
                return True
        except:
            return False
        return False

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				132
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@staticmethod
def is_json_valid(json_data: dict) -> BaseModel | None:
    try:
        return Model(**json_data)
    except:
        return None

def send_message(self, text) -> None:
    api.send_message(self.chat_id, text)

```

		Каліновський Д.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Савіцький Р.С.				133
Змн.	Арк.	№ докум.	Підпис	Дата		