

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:

«Вебзастосунок для пошуку нових знайомств»

Виконав студент групи ІПЗк-23-1
БОНДАРЕНКО Артур Васильович

Керівник роботи:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2025

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 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БОНДАРЕНКО Артур Васильович**

Керівник роботи: **ГОДЛЕВСЬКИЙ Юрій Олександрович**

Тема роботи: **«Вебзастосунок для пошуку нових знайомств»**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: формування архітектури, реалізація функціональних можливостей та розробка інтерфейсу вебзастосунку для пошуку нових знайомств, який забезпечить створення профілів, пошук користувачів за різними параметрами, спілкування у чатах, публікацію дописів, коментування дописів та адміністрування контенту.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Годлевський Ю. О.	27.02.2025р.	27.02.2025р.
2	Годлевський Ю. О.	27.02.2025р.	27.02.2025р.
3	Годлевський Ю. О.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з вебзастосунку для пошуку нових знайомств та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 66 сторінок, 51 ілюстрацію та 26 таблиць. Загальний обсяг 78 сторінок.

Метою роботи є розробка вебзастосунку для пошуку нових знайомств. Застосунок включає модулі реєстрації та авторизації, створення та редагування профілів, пошуку за фільтрами, обміну повідомленнями у чаті, публікації дописів із коментарями, переглядом статистики та керуванням контенту.

У роботі поставлено основні завдання на розробку застосунку, проведено аналіз аналогічних рішень, що дозволило виявити їх сильні та слабкі сторони. На основі отриманих результатів обрано клієнт-серверну архітектуру як найбільш доцільну для реалізації поставлених цілей. Описано ключові сценарії використання системи, визначено логіку взаємодії між основними модулями, побудовано діаграму класів і розроблено алгоритми функціонування основних компонентів. Структуру бази даних спроектовано з використанням MongoDB. На завершальному етапі було проведено тестування системи для перевірки її працездатності та відповідності функціональним вимогам.

КЛЮЧОВІ СЛОВА: ЗНАЙОМСТВА, ВЕБЗАСТОСУНОК, КОРИСТУВАЧ, ЧАТ, ПРОФІЛЬ, ДОПИС.

					ІПЗ.КР.Б – 121 – 25 – ЗПП				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Бондаренко А.В.			Вебзастосунок для пошуку нових знайомств Пояснювальна записка	Літ.	Арк.	Аркуші	
Керівник		Годлевський Ю.О.					3	66	
						Житомирська політехніка група ІПЗк-23-1			
Н. контр.									
Затвердив									

ABSTRACT

The bachelor's qualification work consists of a web application for finding new acquaintances and an explanatory note. The explanatory note to the qualification work contains 66 pages, 51 illustrations and 26 tables. The total volume is 78 pages.

The goal of the work is to develop a web application for finding new acquaintances. The application includes modules for registration and authorization, creating and editing profiles, searching by filters, exchanging messages in the chat, publishing posts with comments, viewing statistics, and managing content.

The work sets the main tasks for the development of the application, analyzes similar solutions, which allows to identify their strengths and weaknesses. Based on the results obtained, the client-server architecture is selected as the most appropriate for the implementation of the set goals. The key scenarios of the system use are described, the logic of interaction between the main modules is determined, a class diagram is built, and algorithms for the functioning of the main components are developed. The database structure is designed using MongoDB. At the final stage, the system is tested to verify its operability and compliance with functional requirements.

KEYWORDS: DATING, WEB APPLICATION, USER, CHAT, PROFILE, MESSAGE.

		Бондаренко А.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПОШУКУ НОВИХ ЗНАЙОМСТВ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	17
Висновки до першого розділу	21
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПОШУКУ НОВИХ ЗНАЙОМСТВ.....	22
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи....	22
2.2 Розробка бази даних	30
2.3 Проєктування та реалізація алгоритмів роботи системи	34
2.4 Реалізація функціоналу вебзастосунку для пошуку нових знайомств	38
Висновки до другого розділу.....	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ	43
3.1 Порядок встановлення та налаштування параметрів застосунку	43
3.2 Структура інтерфейсу та порядок взаємодії із застосунком	45
3.3 Тестування роботи програмного продукту	54
Висновки до третього розділу	62
ВИСНОВКИ	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	64
ДОДАТКИ	67

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

API – Application Program Interface

UI – User Interface

JSON – JavaScript Object Notation

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Люди завжди прагнули до спілкування та нових знайомств, оскільки саме це відкриває можливості для розвитку, отримання підтримки й натхнення. Але у наш час, коли темпи життя прискорюються, а інформаційні технології стають доступнішими люди все частіше використовують вебзастосунки для пошуку співрозмовників, однодумців чи друзів. Саме тому вони є зручними та актуальними інструментами взаємодії і побудови соціальних зв'язків, які дозволяють користувачам знаходити подібних людей за інтересами та цінностями.

Швидкий розвиток технологій однозначно вплинув на способи комунікації між людьми. Функціонал вебзастосунків для пошуку нових знайомств надає не лише швидкий пошук співрозмовників, а й розширює його завдяки поглибленій фільтрації за різними параметрами. Для комунікації між користувачами використовують не тільки обмін повідомленням, а й інші механізми. Завдяки цьому підвищується ефективність та персоналізація встановлення нових зв'язків. Не варто забувати про соціальну адаптацію, адже люди через певні обставини, такі як робота, переїзди чи навчання не завжди мають вільний час для знайомств в реальному житті. Тому застосунки використовуються не лише як інструмент для пошуку нових знайомств, а вже виконують роль соціального механізму, який значно впливає на якість життя користувачів та розширює їхні комунікативні навички.

Метою випускної роботи є проєктування та розробка вебзастосунку для пошуку нових знайомств.

Встановлена мета дослідження обумовила необхідність вирішення низки завдань. З самого початку було проведено аналіз теоретичних та практичних аспектів реалізації вебзастосунків для пошуку нових знайомств, що допомогло сформулювати вимоги до зручності інтерфейсу та функціональності програмного продукту. Також особливу увагу було приділено моделям взаємодії між користувачами. Наступним завданням було визначення оптимальної структури та архітектури застосунку, яка б

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

забезпечувала модульність та масштабованість системи, тому було розглянуто клієнт-серверну архітектуру разом із REST API та способом організації даних у базі даних MongoDB. Під час аналізу аналогів програмного продукту була виявлена низка переваг та недоліків, які були враховані при розробці власного рішення, що дозволило спроектувати більш інтуїтивно зрозумілий інтерфейс, який відповідає всім вимогам сучасних користувачів. При виборі технологій було надано перевагу Node.js, Express.js, React.js, та MongoDB, тому збережено баланс між зручністю розробки та максимальною продуктивністю застосунку.

Після успішного проєктування серверної частини та бази даних почалась безпосередня реалізація ключового функціоналу програмного продукту. Першочерговим завданням була розробка модулів реєстрації та автентифікації користувача, які забезпечили безпечний та зручний доступ до системи.

Наступним завданням стала реалізація сторінок профілю, чатів, пошуку та дописів із всім необхідним функціоналом. Кожна з них була оснащена необхідними інструментами для комфортної взаємодії користувачів. Сторінка профілю дозволяла налаштовувати персональні дані, система чатів забезпечувала швидкий обмін повідомленнями в режимі реального часу, а сторінка пошуку допомагала знаходити співрозмовників за різними критеріями.

Завершальним етапом стала організація комплексного тестування функціоналу на відповідність заявленим вимогам. Перевірка здійснювалась як вручну, так із використанням інструментів для тестування API.

Об'єктом дослідження є реалізація вебзастосунку для пошуку нових знайомств, що забезпечує взаємодію між користувачами.

Предметом дослідження є використання веб-орієнтованих технологій для розробки вебзастосунку для пошуку нових знайомств, зокрема, підходів до побудови клієнтської та серверної частин, організації зберігання даних та забезпечення взаємодії користувачів.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПОШУКУ НОВИХ ЗНАЙОМСТВ

1.1 Постановка задачі

Основною метою кваліфікаційної роботи бакалавра є розробка вебзастосунку для пошуку нових знайомств, який забезпечує зручну, безпечну та ефективну взаємодію між користувачами, а також має стійку та надійну серверну частину. Користування повинно бути інтуїтивно зрозумілим і не викликати труднощів.

Розробка має базуватися на найбільш вдалих та сучасних методах створення вебзастосунків, що дозволить досягти високої якості кінцевого продукту. Усі етапи реалізації будуть виконуватись послідовно.

В першу чергу необхідно створити дизайн користувацького інтерфейсу, що включає в собі компонентний підхід до побудови елементів UI та розроблення загальної візуальної концепції. Основні сторінки застосунку включають: головну, сторінки реєстрації та входу, профіль користувача, пошуку, чатів, дописів та адміністративна панель. На сторінці пошуку необхідно реалізувати фільтрацію користувачів за відповідними параметрами, а сторінка дописів має включати не лише показ усіх дописів, з можливостями сортування, а також показувати останні коментарі, для підвищення взаємодії між користувачами. В той час адміністративна панель матиме декілька основних розділів: перегляд статистики, керування дописами, управління користувачами та забезпечення керування контентом, тобто обробка скарг.

У реалізації серверної частини передбачено побудову таких ключових модулів: користувачів, дописів, коментарів, повідомлень, скарг та адміністративного керування, оскільки кожен модуль системи є важливим для забезпечення повного функціоналу застосунку, то їх буде розглянуто нижче.

Модуль користувачів, який відповідає за керування профілями користувачів, включає захищену реєстрацію та авторизацію з використанням шифрування паролів та токenu автентифікації. Містить функціонал оновлення профілю, розділяє права доступу між користувачами та реалізацію блокування

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

небажаних користувачів. Також зберігає інформацію про активність користувача.

Модуль дописів реалізовує функціонал створення, редагування, оновлення та видалення дописів. Виконує можливості сортування за певними критеріями та відстеження кількості переглядів та коментарів.

Модуль коментарів містить в собі функціонал управління коментарями. Відповідає за створення, оновлення, редагування та видалення коментарів з прив'язкою до конкретного допису.

Модуль повідомлень відповідає за обмін повідомленнями між користувачами. За допомогою підключення до WebSocket виконує свій функціонал в реальному часі [27]. Його функції включають: створення чатів, блокування небажаних користувачів, управління повідомленнями та підрахунок кількості непрочитаних повідомлень.

Адміністративний модуль відповідальний за адміністрування застосунку. Користувач, який є адміністратором має доступ до перегляду статистики застосунку, керує дописами та користувачами, а також виконує обробку скарг на дописи або профілі.

Модуль скарг дає користувачам змогу повідомляти адміністрацію про порушення у вигляді скарг на дописи або профілі. Він реалізує розподіл скарг за типами, надає функціонал відстеження їх статусів з обробкою, а також містить інструменти фільтрації.

Додаткові модулі, що відповідають за зберігання та управління довідковою інформацією про міста та інтереси користувачів, використовуються для заповнення профілів і фільтрацію користувачів під час пошуку.

В ході реалізації програмного продукту є важливим налаштування безпечної системи реєстрацію та авторизації користувача, що включає в себе створення токена автентифікації та збереження у базі даних зашифрованого пароля. Реєстрація передбачає заповнення більшої кількості полів, користувач одразу заповнює свій профіль, що власне і дає змогу після реєстрації

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

користуватися всією функціональністю застосунку. Надалі користувач може в довільний час відредагувати особисту інформацію.

Одразу після розробки користувацького інтерфейсу та всіх ключових модулів необхідно впровадити ефективну та безвідмовну взаємодію серверної частини разом з клієнтським інтерфейсом. Тому, буде реалізовано надійний API зі всіма необхідними ендпоінтами для обробки запитів та взаємодію з базою даних. Також для чатів необхідно реалізувати інтеграцію з WebSocket для обміну повідомленнями між користувачами в реальному часі, що і забезпечить швидкий та безперервний обмін даними в системі спілкування застосунку.

Результатом виконання стане повноцінний вебзастосунок для пошуку нових знайомств, який буде містити сучасний та зручний інтерфейс. Користувачі матимуть можливість не лише ефективно спілкуватись та переглядати персоналізований контент, а й створювати його та знаходити цікавих для себе людей. Адміністративна панель має забезпечити контроль за безпекою та якістю системи. Така реалізація дозволить сформувати комфортне середовище для нових знайомств, що буде сприяти розвитку соціальної взаємодії у цифровому просторі та підвищувати комфорт користувачів.

1.2 Аналіз аналогів програмного продукту

Перш ніж почати розробку програмного продукту необхідно виконати аналіз існуючих рішень на ринку, що дозволить оцінити їх переваги і недоліки. Такий аналіз допомагає уникнути вже існуючих помилок і врахувати кращі практики, які вже довели свою ефективність. Крім того, він допоможе зрозуміти потреби користувачів та розробити стратегію, що зробить застосунок більш ефективним. Тому важливо враховувати сучасні тенденції, щоб створити конкурентоспроможний та зручний продукт.

Під час аналізу ринку застосунків для пошуку нових знайомств було визначено три основні платформи: OkCupid, Bumble та Boo. Нижче наведено детальний розгляд кожної з них, з акцентом на функціональні можливості, переваги та недоліки.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

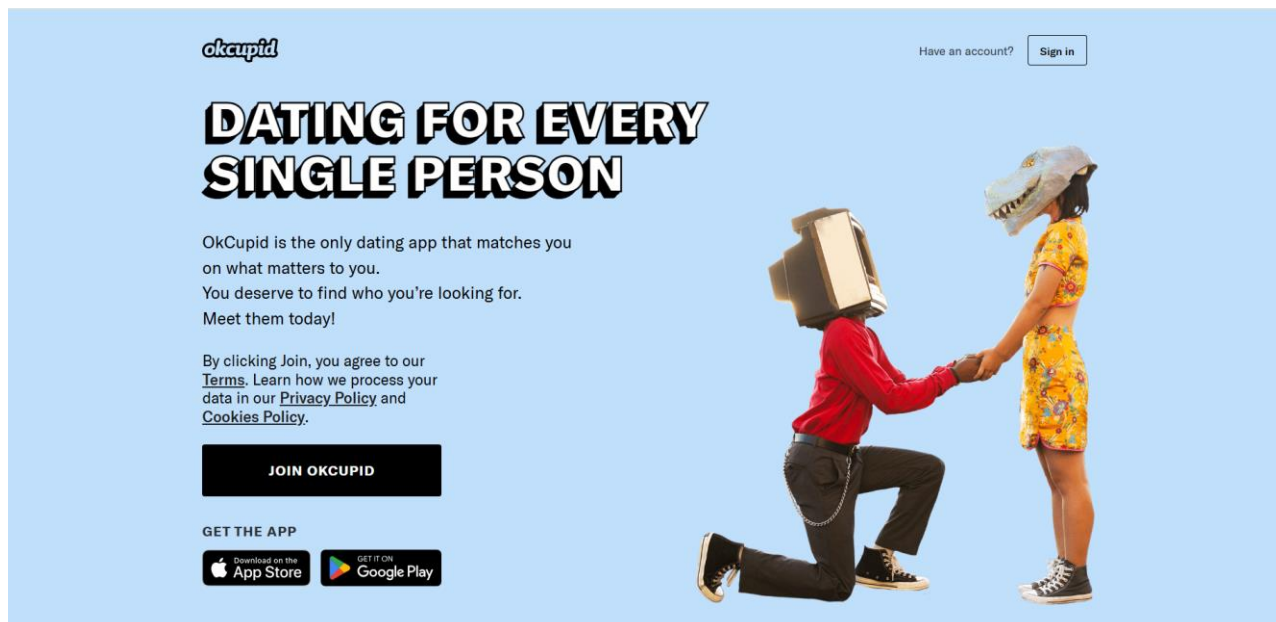


Рис. 1.1 OkCupid

OkCupid – це одна з найбільш популярних платформ для знайомств, яка дозволяє користувачам знаходити співрозмовників та друзів. Вирізняючись своїм глибоким і детальним підходом до створення профілів та пошуку сумісних партнерів [12].

До переваг даного програмного продукту можна віднести те, що, заповнюючи профіль, користувачі відповідають на численні питання, які охоплюють різні аспекти життя, включаючи політичні погляди, життєві цінності та вподобання у стосунках. Переважна більшість основних функцій, такі як деталізований профіль, пошук та взаємодія із користувачами є повністю безкоштовними, що робить цей додаток зручним для використання широким колом користувачів. В той час як інтуїтивний інтерфейс та можливість фільтрувати пошук за різними критеріями підвищують комфорт використання платформи. Крім того, активно підтримується різноманітність, пропонуючи широкий вибір гендерних ідентичностей, що створює безпечний простір для пошуку однодумців і можливість самовираження. Також платформа постійно вдосконалюється, реагуючи на запити користувачів та впроваджуючи зміни.

В той час OkCupid має й певні недоліки, серед яких необхідність відповідати на велику кількість запитань при створенні профілю, що займає

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

деякий час. Також наявна проблема активності користувачів, оскільки переважна більшість не є активною. В той же час неможливо без підписки користуватись додатком ефективно, оскільки будуть недоступні деякі фільтри пошуку та перегляд профілю без попередньої реклами. Крім того, через велику кількість користувачів виділитись майже неможливо, що створює складнощі для новачків або менш активних користувачів. Також зустрічаються випадки наявності ботів, незважаючи на заходи безпеки, що частково знижує довіру аудиторії.



Features ▾The ShopThe app ▾AmbassadorsEventsThe BuzzAboutSign In

WE'RE NOT JUST FOR DATING



Meet someone with Bumble Date

Find someone you actually want to date, then go ahead and Make the First Move. Make genuine connections, vibe over what you both love, and most importantly, have fun.



Find new friends on Bumble For Friends

Meet someone like you. Or unlike you. Make new friends and find new things to do together — whether you're home or away.



Make career moves with Bumble Bizz

Find friends with work benefits. Connect with other business professionals to find your next job, a mentor or even a whole new career.

Рис. 1.2 Bumble

Bumble – це популярний застосунок для знайомств, який містить в собі нестандартну механіку взаємодії між користувачами. На відміну від багатьох інших платформ, де будь-який користувач може почати розмову, застосунок надає жінкам право першими починати діалог у режимі знайомств, відходячи від традиційних моделей взаємодії [13].

До переваг можна віднести високий рівень безпеки, тому що тільки жінки можуть здійснити перший крок до спілкування, що робить досвід взаємодії комфортним та безпечнішим, уникаючи небажаних повідомлень. Також серед користувачів цього застосунку особливо цінується можливість вибору режимів: можна обирати між класичним підходом, пошуку лише друзів або професійним спілкуванням, об'єднуючи знайомства та кар'єрний

розвиток на одній платформі. Інтерфейс є достатньо зрозумілим та яскравим, що сприяє позитивному досвіду використання. Доступна можливість блокування небажаного користувача або контенту.

Серед недоліків необхідно виокремити те, що присутнє обмеження на початок спілкування, яке змушує користувачів швидко реагувати, адже через 24 години анулюється можливість початку діалог. Також через недостатню кількість перевірок користувачів залишається велика кількість фейкових та шахрайських акаунтів, що негативно впливає на досвід використання. Користувачі активно скаржаться на недостатню якість підтримки: затримки з відповідями, шаблонні відповіді, які в повному обсязі не відповідають на поставлене питання. Та активно використовуються маніпулятивні алгоритми, щоб стимулювати купівлю платної підписки.

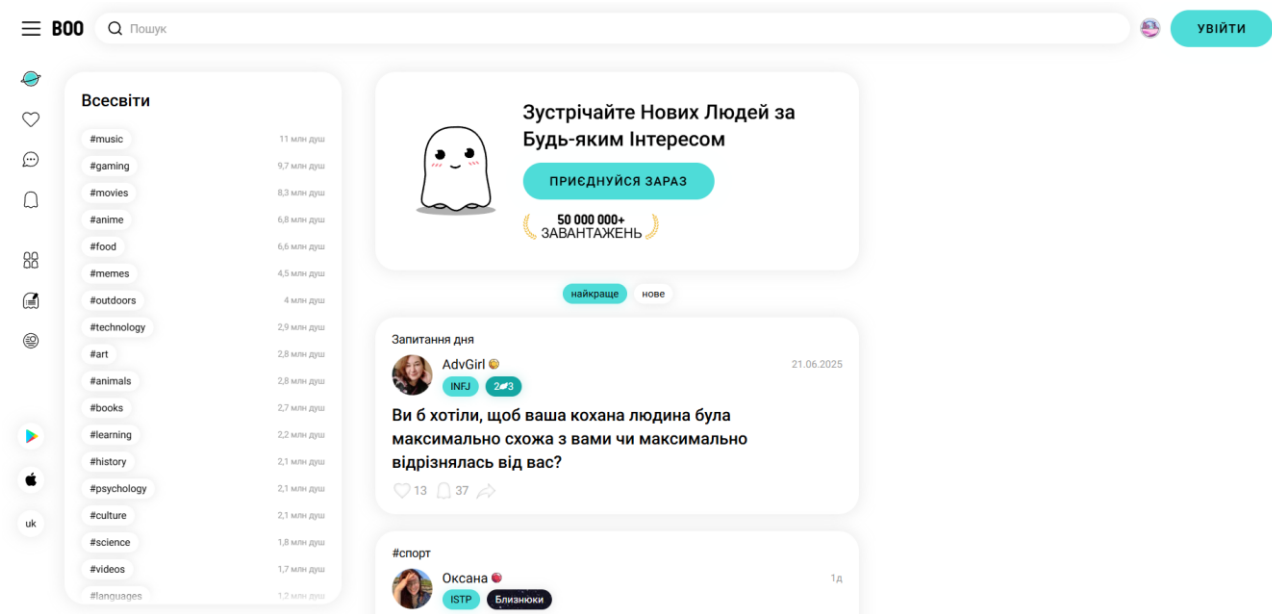


Рис. 1.3 Boo

Boo – це застосунок для пошуку нових знайомств, що орієнтований на створення глибоких зв’язків між користувачами використовуючи сумісність особистостей [14]. Інтегрує мультимедійні функції, серед яких створення 3D аватарів та персоналізовані привітання, що дозволяє виражати користувачам свою індивідуальність. Також в додаток включено елементи соціальних мереж, тобто публікація дописів та взаємодія з іншими через них.

До переваг Воо можна віднести наявність спільнот за інтересами, де користувачі можуть публікувати власні дописи, обговорювати різні теми та знаходити людей за інтересами, створюючи атмосферу глибокого спілкування. Підвищений рівень безпеки забезпечується за рахунок перевірки профілів, що зменшує кількість фейкових акаунтів. Інтерфейс додатку доволі простий та інтуїтивно зрозумілий, що дозволяє легко орієнтуватись навіть новачкам. Вагомою перевагою є вбудована функція перекладу, яка надає можливість знаходити спілкування навіть між користувачами, які говорять різними мовами. Впроваджено використання тестів на основні теорії типів особистості, що впливає на пошук людей зі схожим світоглядом та пропонуються персональні рекомендації для покращення спілкування.

Серед недоліків варто відзначити недостатню кількість користувачів, що ускладнює пошук потенційних знайомств, особливо в певних регіонах. Частина користувачів скаржиться на відсутність відповідей на їх повідомлення, що свідчить про низьку зацікавленість інших та обмеженість функціоналу без платної підписки, яка ускладнює взаємодію між користувачами. Технічні проблеми також не рідкість, оскільки багато користувачів повідомляють про збої в роботі чату або повільне оновлення контенту у стрічці спільнот та періодичні помилки в ході завантаження профілів і дописів.

В ході аналізу функціональних можливостей провідних застосунків для пошуку нових знайомств було виокремлено низку підходів до реалізації. Всі три застосунки забезпечують базовий функціонал, що містить в собі: реєстрацію, наповненість профілю, реалізовані механізми відповідальності користувачів, можливості перегляду й взаємодії. Але кожна платформа має свої особливості. OkCupid орієнтований на створення детальних профілів з глибокою наповненістю, а Bumble впроваджує унікальну функціональність взаємодії, що тільки жінкам належить ініціатива першого контакту та надає можливість обирати різні режими спілкування. В той же час Воо інтегрує функціональність соціальних мереж, залучаючи користувачів до створення

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

контенту та подальшу взаємодію через нього зі створенням спільнот. Boo підвищує мультимедійність завдяки 3D зображенням профілю і використанням функцій автоматичного перекладу. Водночас було виявлено спільні недоліки, серед яких обмеженість базового функціоналу, недостатня перевірка користувачів та технічні проблеми. Детальне порівняння аналогів зображено на табл. 1.1

Таблиця 1.1

Порівняльна таблиця аналогів

Характеристика	OkCupid	Bumble	Boo	DatApp
Зручність використання	Висока	Середня	Висока	Висока
Фільтрація пошуку	Розширена	За базовими критеріями	За сумісністю	Розширена
Наповненість профілів	На основі питань	Орієнтована на швидкий огляд	На основі тестування	Детальна
Механіка першого кроку	Будь-хто	Жінки	Будь-хто	Будь-хто
Соціальні функції	Обмежені	Обмежені	Публікація дописів	Публікація дописів
Платний функціонал	Частково	Частково	Частково	Відсутній
Система скарг	Присутня	Присутня	Присутня	Присутня

Тому наш програмний продукт має містити базовий функціонал: швидку реєстрацію, наповненість профілю, пошук користувачів за різними параметрами. Додатково реалізація функціоналу дописів для розширення механізму взаємодії між користувачами та спілкування в реальному часу. Однак дуже важливо реалізувати систему скарг та їх обробку, для взаємодії між користувачами та адміністрацією. Такі засоби дозволять підтримувати безпечне та доброзичливе середовище в межах застосунку.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Під час проектування вебзастосунку було обрано клієнт-серверну архітектуру. Обраний підхід забезпечує розподіл між клієнтською та серверною частинами, що оптимізує їх взаємодію. Також розмежовує відповідальність за інтерфейс і обробку даних та підвищує масштабованість, що робить застосунок гнучкішим в наступних оновленнях та простішим у підтримці.

Серед інструментальних засобів для клієнтської частини було обрано React.js разом з Redux, а для стилізації SCSS модулі [10]. React – це фреймворк який дозволяє створювати інтерфейс завдяки компонентному підходу, що здійснює розробку організовано, а застосунок стає більш адаптованим до наступних змін, оскільки кожен компонент відповідає лише за свою частину інтерфейсу [3]. Він працює завдяки віртуальному DOM, що створює легку копію браузерного DOM, уникаючи прямої взаємодії з ним. React при зміні стану застосунку спочатку оновлює власний віртуальний, а вже тільки потім порівнює його з попередньою версією та при потребі вносить зміни до браузерного [20]. Це однозначно впливає на швидкість роботи програмного продукту, підвищуючи продуктивність, особливо при оновленнях складних інтерфейсів. Також забезпечує контрольованість стану застосунку, оскільки реалізує в собі односторонній потік даних, передаючи їх від батьківських до дочірніх компонентів, що значним чином спрощує відстеження помилок та тестування. Його було використано частково з Redux.

Redux – це сучасний інструмент для керування станом застосунку, що містить архітектурний шаблон Flux та забезпечує централізоване управління станом [6]. Необхідні дані зберігаються в одному об'єкті, що спрощує передбачуваність змін. При взаємодії користувача з контентом, створюється подія яка передається до функції обробки стану, в свою чергу вона її обробляє та зберігає новий стан в об'єкт, на який підписані компоненти оновлюють свій інтерфейс відповідно. Для роботи з Redux також було застосовано Redux

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Toolkit, який є офіційним пакетом та призначений для спрощення розробки додатків із керування станом застосунка [7]. Маючи набір своїх інструментів допомагає ефективно керувати даними, забезпечуючи максимальну доступність та синхронізацію між компонентами.

Використання SCSS модулів для модифікацій зовнішнього вигляду чудово доповнює компонентний підхід React. Тому що він є одним із кращих препроцесором CSS, що дозволяє використовувати змінні, міксини, функції та кращу структуру вкладених правил. Коли він використовується як окремий модуль кожного компонента, то забезпечує локальну область видимості стилів, що запобігає конфліктам імен класів.

Для серверної частини було обрано Node.js, платформу з відкритим кодом для виконання високопродуктивних мережевих застосунків мовою JavaScript, яка реалізує це завдяки архітектурі з обробкою подій та асинхронності [1]. Це дозволяє ефективно обслуговувати запити клієнтів.

Для виконання HTTP-запитів між клієнтською та серверною частинами вебзастосунку було обрано бібліотеку Axios [24]. Вона реалізує інтуїтивно зрозумілий та простий API для роботи із запитамі, автоматично трансформуючи дані та надає функціональності для обробки помилок. У програмному додатку Axios забезпечує всі взаємодії REST API сервера, що включає: автентифікацію користувачів, управління постами, коментарями, повідомленнями та функціоналом адміністратора [8]. Додатково використовує перехоплювачі відповідей та запитів, автоматично додаючи токен, що централізує обробку помилок з налаштованим переходом на сторінку авторизації при недійсному токені.

Робота з серверною частиною зумовила використання Express.js – гнучкий та досить мінімалістичний фреймворк для створення та обробки серверних застосунків [2]. Він значно спрощує побудову маршрутизації, обробку HTTP-запитів під час інтеграції з допоміжними функціями та виконує підключення до БД. Сприяє масштабуванню логіки додатку без зайвої

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

складності, зі швидким створенням і підтримкою API, яке і реалізує взаємодію між клієнтом та сервером.

Для зберігання та обробки даних було обрано MongoDB – документно-орієнтовано БД, яка належить до категорії нереляційних систем управління базами даних [4]. Її основна особливість полягає у зберігання даних у вигляді документів формату BSON, тобто двійковому представленні JSON, на відміну від таблиць у представленнях класичний реляційних базах даних. Це дозволяє одразу зберігати складні вкладені структури, з різними типами даних, що включають масиви, об'єкти та інші. Використовуючи MongoDB ми маємо високу адаптивність у структурі збереження даних та можемо наперед не визначати фіксовану схему. Серед інших переваг – підтримка горизонтального масштабування через шардинг та можливість обробляти великі обсяги даних на розподілених системах. А завдяки індексам, додатковому кешу та оптимізованому механізму читання і запису забезпечує високу та стійку швидкість операцій. Надаючи можливості агрегації, сортування та фільтрації даних дозволяє створювати складні запити.

Для інтеграції MongoDB та Node.js було використано Mongoose – об'єктно орієнтований диспетчер, що реалізує зручний функціонал для роботи з БД [5]. Він дозволяє створення чітких схем для документів, визначаючи типи, впроваджуючи валідацію, використовуючи віртуальні поля, а також певні методи та хуки для попередньої обробки даних або після збереження. Тобто Mongoose реалізує схему даних та її логіку на рівні застосунку, полегшуючи подальший контроль та підтримку. Завдяки функціоналу встановлення зв'язків між колекціями даних, підтягуючи пов'язані документи, дає змогу до ефективної взаємодії, як у реляційних базах. Використання MongoDB Atlas дозволяє швидко створити, обслуговувати та масштабувати БД у хмарі з майже відсутніми втратами продуктивності [19].

Функція обробки завантаження зображень у серверній частині забезпечує проміжне програмне забезпечення Multer [11]. Він працює з multipart/form-data – механізм передачі файлів через запити, що дозволить

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

реалізувати функціонал додавання зображень до профілю та дописів користувачів. Зберігання завантажених файлів реалізується локально у окремій директорії, а в полі БД зберігається лише посилання.

Для реалізації функціоналу обміну повідомленнями між користувачами в реальному часі було вирішено використати WebSocket [27]. В той час як використання бібліотеки Socket.io забезпечить двосторонню комунікацію між клієнтською та серверною частиною [28]. Архітектура WebSocket побудована на основі сервера Node.js, який створюється екземпляром Express додатку, а бібліотека вже виконує початкове налаштування поверх HTTP сервера. У клієнтську частину інтегровано Socket.io-client, що є клієнтською версією тієї ж самої бібліотеки [30]. Включаючи налаштовані параметри CORS, що дозволило безпечні підключення з клієнтського домену [18]. Доступ на відправку повідомлень виконаний за допомогою перевірки авторизації, що використовує JWT [15]. Архітектура кімнат використовується для маршрутизації повідомлень, тому після переходу на сторінку чатів користувачем автоматично приєднується до особистої кімнати, яка відповідає його унікальному ідентифікатору.

Архітектурне рішення застосунку є досить оптимізованим з низькими вимогами для обчислювальних можливостей, що робить її легкою у розгортанні на широкому колі типових серверних середовищ. Для забезпечення стабільної роботи рекомендовано сервер із щонайменше 2 ГБ оперативної пам'яті, двоядерним процесором з частотою від 2.0 ГГц та не менше 1 ГБ вільного дискового простору для зберігання зображень профілів та дописів користувачів

Отже, обґрунтований вибір інструментальних засобів та врахування вимог до апаратного забезпечення забезпечило ефективний, зручний у підтримці та готовий до масштабування застосунок, що містить в собі сучасні технології клієнтської та серверної частини. Такий підхід дозволяє не лише стабільну роботу основного функціоналу а й спрощує майбутній розвиток системи та інтеграцію із зовнішніми сервісами за мінімальних витрат ресурсів.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Висновки до першого розділу

Аналіз предметної області дав змогу визначити головні потреби та вимоги користувачів: швидка реєстрація, достатня наповненість профілю, пошук користувачів за різними параметрами, розширення механізмів взаємодії між користувачами за допомогою впровадження системи дописів, забезпечення спілкування між користувачами у реальному часі. Також варто реалізувати ефективну систему скарг, що давала б змогу повідомляти про порушення, яка слугуватиме керуванням контенту, підвищуючи загальну безпеку користуванням застосунку. Провели дослідження існуючих рішень, що дало змогу оцінити функціональні можливості та виокремити переваги з недоліками. Особливу увагу було приділено аналізу типових недоліків конкурентних сервісів, таких як перевантаженість інтерфейсу.

В ході обґрунтування вибору інструментальних засобів було обрано сучасні та зручні інструменти, які розділяють клієнтську та серверну частини, дозволяють масштабувати застосунок і спростять подальшу підтримку. Тому для клієнтської частини було використано React.js із частковим використання Redux, для реалізації компонентного підходу та організації управління станом. А зміни зовнішнього вигляду інтерфейсу виконано за допомогою SCSS модулів, які гарантують ізольованість. Проектування серверу на Node.js із використанням Express.js забезпечить просту маршрутизацію та обробку запитів. Дані було вирішено зберігати у MongoDB та для керування БД використати Mongoose, що підвищить комфорт взаємодії та побудови схем разом з серверною перевіркою. Обраний технологічний стек дозволить забезпечити безпеку даних користувачів завдяки підтримці сучасних методів аутентифікації та авторизації.

Отримано чітко сформоване уявлення про вебзастосунок для пошуку нових знайомств з визначеними ключовими функціональними вимогами, використаними інструментами та технічними характеристиками апаратного забезпечення. Сформовано структуру проєкту, що забезпечить його успішну реалізацію.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПОШУКУ НОВИХ ЗНАЙОМСТВ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Вебзастосунок для пошуку нових знайомств надає інтерфейс, через який користувачі можуть створювати та редагувати профіль, шукати інших користувачів за заданими критеріями, створювати персоналізований контент, взаємодіяти через систему дописів та коментарів, а також спілкуватись в реальному часі. Також впроваджено адміністративну панель яка забезпечить управління дописами та користувачами, надасть можливість переглядати статистику та активність реєстрацій застосунку. Вона містить в собі систему скарг, на які може швидко реагувати адміністратор, забезпечуючи безпеку та комфорт користувачів. Тому користувачів варто розділяти за рівнем функціонального доступу, щоб розмежовувати їх дії у програмному продукті.

Короткий опис акторів представлено в табл. 2.1.

Таблиця 2.1

Виявлення акторів системи

Актор	Короткий опис
Користувач	Людина, яка реєструється у застосунку з метою пошуку нових друзів, партнерів або просто для спілкування. При реєстрації одразу заповнюється профіль, із можливістю подальшого редагування. Має можливість до фільтрованого пошуку користувачів, перегляду та створення дописів, їх редагування, видалення та коментування, а також перегляд профілю інших користувачів, надсилання особистих повідомлень. Може подавати скарги на дописи або профілі.
Адміністратор	Користувач, який має роль адміністратора. Має доступ до адміністративної панелі, де може переглядати статистику, керувати користувачами та дописами, а також опрацьовувати скарги на порушення з боку інших користувачів.

Виявивши акторів системи, побудуємо діаграми варіантів використання для користувача та адміністратора, які ілюструють їх взаємодію із застосунком. Було побудовано діаграму використання для звичайного користувача застосунку (рис. 2.1).



Рис. 2.1 Варіанти використання для користувача

Також реалізовано діаграму використання для адміністратора (рис. 2.2).

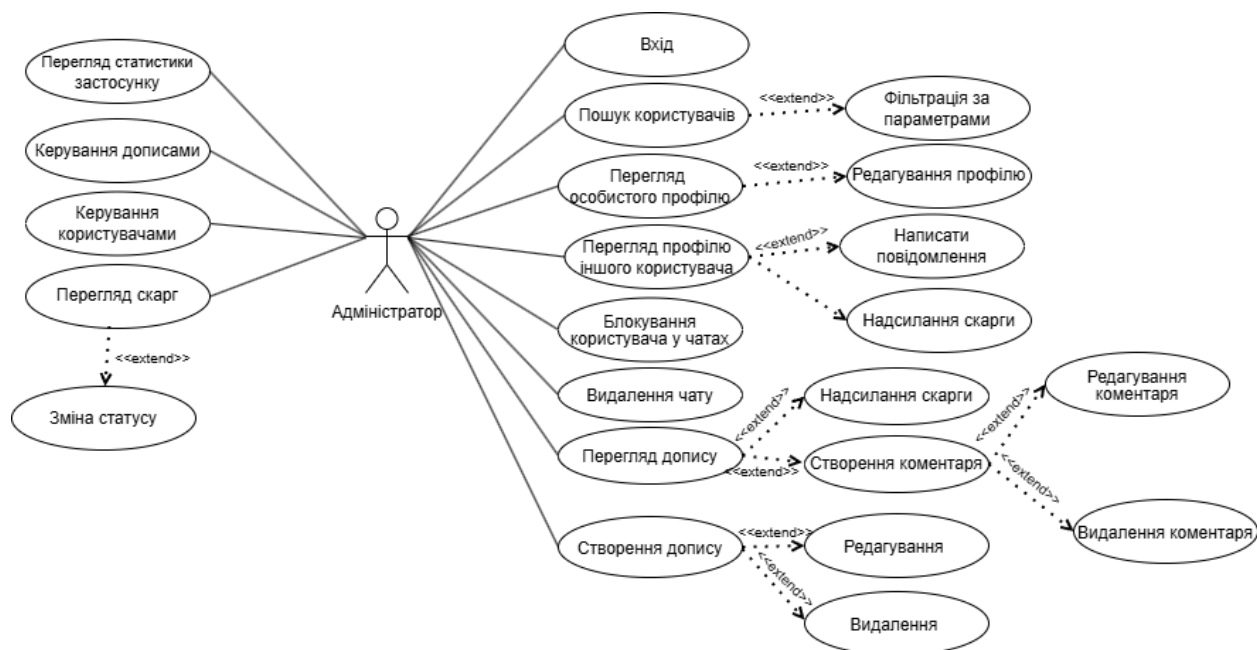


Рис. 2.2 Варіанти використання для адміністратора

Опис діаграм варіантів використання для користувача та адміністратора наведено в табл. 2.2 – 2.14.

Таблиця 2.2

Опис прецеденту «Реєстрація»

Назва варіанту використання	Реєстрація
Основні актори	Користувач
Короткий опис	Процес створення нового облікового запису в системі. Користувач заповнює всі обов'язкові поля, та автоматично створюється заповнений профіль.

Таблиця 2.3

Опис прецеденту «Вхід»

Назва варіанту використання	Вхід
Основні актори	Користувач, адміністратор
Короткий опис	Процес аутентифікації користувача в системі за допомогою облікових даних для отримання повного доступу до функціоналу застосунку.

Таблиця 2.4

Опис прецеденту «Пошук користувачів»

Назва варіанту використання	Пошук користувачів
Основні актори	Користувач, адміністратор
Короткий опис	Користувач здійснює пошук інших користувачів на сторінці пошуку. Також може додатково застосувати фільтри для більш точних результатів.

Таблиця 2.5

Опис прецеденту «Перегляд особистого профілю»

Назва варіанту використання	Перегляд особистого профілю
Основні актори	Користувач, адміністратор
Короткий опис	Користувач має можливість перегляду особистого профілю із можливістю редагування будь-якого поля.

Таблиця 2.6

Опис прецеденту «Перегляд профілю іншого користувача»

Назва варіанту використання	Перегляд профілю іншого користувача
Основні актори	Користувач, адміністратор
Короткий опис	Користувач має змогу переглядати профілі інших користувачів, знайдених через пошук. Він може надіслати повідомлення, що автоматично відкриє чат або подати скаргу на профіль.

Таблиця 2.7

Опис прецеденту «Блокування користувача у чатах»

Назва варіанту використання	Блокування користувача у чатах
Основні актори	Користувач, адміністратор
Короткий опис	Користувач може заблокувати іншого користувача застосунку у чаті, що заборонить йому надсилати повідомлення.

Таблиця 2.8

Опис прецеденту «Видалення чату»

Назва варіанту використання	Видалення чату
Основні актори	Користувач, адміністратор
Короткий опис	Користувач може видалити чат з іншим користувачем, що призведе до видалення всіх повідомлень між ними.

Таблиця 2.9

Опис прецеденту «Перегляд допису»

Назва варіанту використання	Перегляд допису
Основні актори	Користувач, адміністратор
Короткий опис	Користувач переглядає опубліковані дописи, ознайомлюється з вмістом та коментарями. У разі виявлення неприпустимого вмісту він може надіслати скаргу.

Таблиця 2.10

Опис прецеденту «Створення допису»

Назва варіанту використання	Створення допису
Основні актори	Користувач, адміністратор
Короткий опис	Користувач має можливість створювати новий допис, вказуючи заголовок, опис і за бажанням додає зображення та теги.

Таблиця 2.11

Опис прецеденту «Перегляд статистики застосування»

Назва варіанту використання	Перегляд статистики застосування
Основні актори	Адміністратор
Короткий опис	Адміністратор може переглядати статистику, зокрема кількість зареєстрованих і активних користувачів, дописів, повідомлень, коментарів а також графік реєстрацій за останні 30 днів.

Таблиця 2.12

Опис прецеденту «Керування дописами»

Назва варіанту використання	Керування дописами
Основні актори	Адміністратор
Короткий опис	Адміністратор може переглядати дописи користувачів та виявляти ті, що містять неприпустимий вміст, за потреби видаляти їх.

Таблиця 2.13

Опис прецеденту «Керування користувачами»

Назва варіанту використання	Керування користувачами
Основні актори	Адміністратор
Короткий опис	Адміністратор має доступ до перегляду профілів користувачів, змінювати їх ролі чи видаляти облікові записи у разі порушень.

Таблиця 2.14

Опис прецеденту «Перегляд скарг»

Назва варіанту використання	Перегляд скарг
Основні актори	Адміністратор
Короткий опис	Адміністратор переглядає та обробляє скарги, сортуючи їх за типом та статусом.

Комплексний аналіз предметної області, огляд існуючих рішень на ринку та чітка постановка задачі із виявленням акторів дозволили нам сформулювати функціональні та нефункціональні вимоги, які необхідно врахувати під час реалізації вебзастосунку для пошуку нових знайомств. Функціональні вимоги безпосередньо відповідають за взаємодію користувача із застосунком, а нефункціональні охоплюють технічні та якісні характеристики системи.

Зведені вимоги до системи подано у табл. 2.15.

Таблиця 2.15

Вимоги до застосунку

Категорія	Підкатегорія	Вимога
Функціональні вимоги	Реєстрація та профіль	Впровадження функціоналу авторизації та реєстрації із заповнення профілю та його редагування.
	Пошук користувачів	Можливість пошуку інших користувачів за фільтрами.
	Система повідомлень	Реалізація обміну повідомленнями між користувачами в реальному часі.
	Скарги та контроль контенту	Можливість залишати скарги на профілі або дописи. Адміністратор має мати механізм перегляду та обробітку цих скарг.
Нефункціональні вимоги	Простота використання	Інтерфейс зрозумілий інтерфейс: Основні функції освоюються користувачами за 2 години, а для досвідчених — за годину.

Категорія	Підкатегорія	Вимога
Нефункціональні вимоги	Продуктивність і масштабованість	Підтримка щонайменше 100 одночасних користувачів, у разі зростання навантаження можливість масштабування до 1000.
	Надійність	Працювати без збоїв не менше 20 робочих днів поспіль. Допустимий час простою — не більше 5% загального часу. Не більше 1 критичної помилки на 5 000 рядків коду.

Аналіз функціональних вимог допоміг сформулювати сутності об'єктно-орієнтованої структури системи, що забезпечать реалізацію вебзастосунку для пошуку нових знайомств. Нижче наведено діаграму класів системи, яка демонструє всі зв'язки між сутностями (рис. 2.3).



Рис. 2.3 Діаграма класів

Далі буде розглянуто кожен клас застосунку та реалізовані в ньому методи відповідно до поставлених вимог.

Клас User є центральним та відповідає за управління користувачами системи. Він виконує функцію реєстрації нового користувача, авторизації у застосунок, отримання інформації про себе. Дозволяє користувачеві редагувати особисті дані, також відповідає за блокування та розблокування

інших користувачів в межах чату. Реалізований функціонал роботи зі списком заблокованих користувачів для відображення статусів у списках чатів, що контролює взаємодію між користувачами.

За впровадження однієї з моделей комунікації між користувачами та управління контентом застосунку відповідає клас Post. Він впроваджує механізми створення, отримання, оновлення та видалення дописів у програмному продукті. Дозволяє отримати всі дописи або отримати один, щоб відкрити повний допис з усім можливим функціоналом. Забезпечує взаємодію користувачів, надаючи можливість ділитися інформацією, ідеями або подіями.

Клас Comment розширює функціонал дописів надаючи можливість їх коментувати. Методи цього класу дозволяють отримувати всі коментарі до окремого допису, видаляти та оновлювати їхній вміст. Крім того відповідає за важливий компонент останніх коментарів на сторінці публікацій, що відображає найактуальнішу інформацію в інтерфейсі застосунка.

Для спілкування користувачів в реальному часі було реалізовано клас Message, який і керує обміном повідомленнями. Методи цього класу надають функціонал для отримання унікальних ідентифікаторів користувачів системи з якими спілкується поточний користувач. Також саме він реалізує створення нових повідомлень, завантаження повідомлень в чат та позначення як прочитаних, їх видалення та видалення чатів.

Клас City відповідає за управління інформацію про локацію користувачів. Його методи передбачають додавання нових, отримання списку доступних, працювати з окремим містом через ідентифікатор, оновлювати дані та видаляти взагалі. Клас використовується для наповнення профілів користувачів, що реалізує майбутній пошук за критеріями.

Керування інтересами користувачів належить класу Interest. Реалізація його методів забезпечує створення нових інтересів, отримання списку всіх або ж роботу з конкретним, оновлення і видалення. Цей клас допомагає формувати профілі користувачів, завдяки додавання до них персоналізованих характеристик.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

За впровадження системи скарг у застосунку відповідає клас Report. Його методи дозволяють створювати нові скарги на користувачів або дописи, отримувати їх з фільтрацією по списку або типу. Для адміністратора його функціонал впроваджує можливості працювати з конкретною скаргою, оновлювати її статус або взагалі видаляти. Сутність важлива для підтримки порядку у програмному продукті.

2.2 Розробка бази даних

При створенні програмного продукту розробка БД є дуже важливим етапом, оскільки саме вона визначає спосіб зберігання та організації інформації. В результаті була створена структура яка повністю задовольняє вимоги застосунку (рис. 2.4).

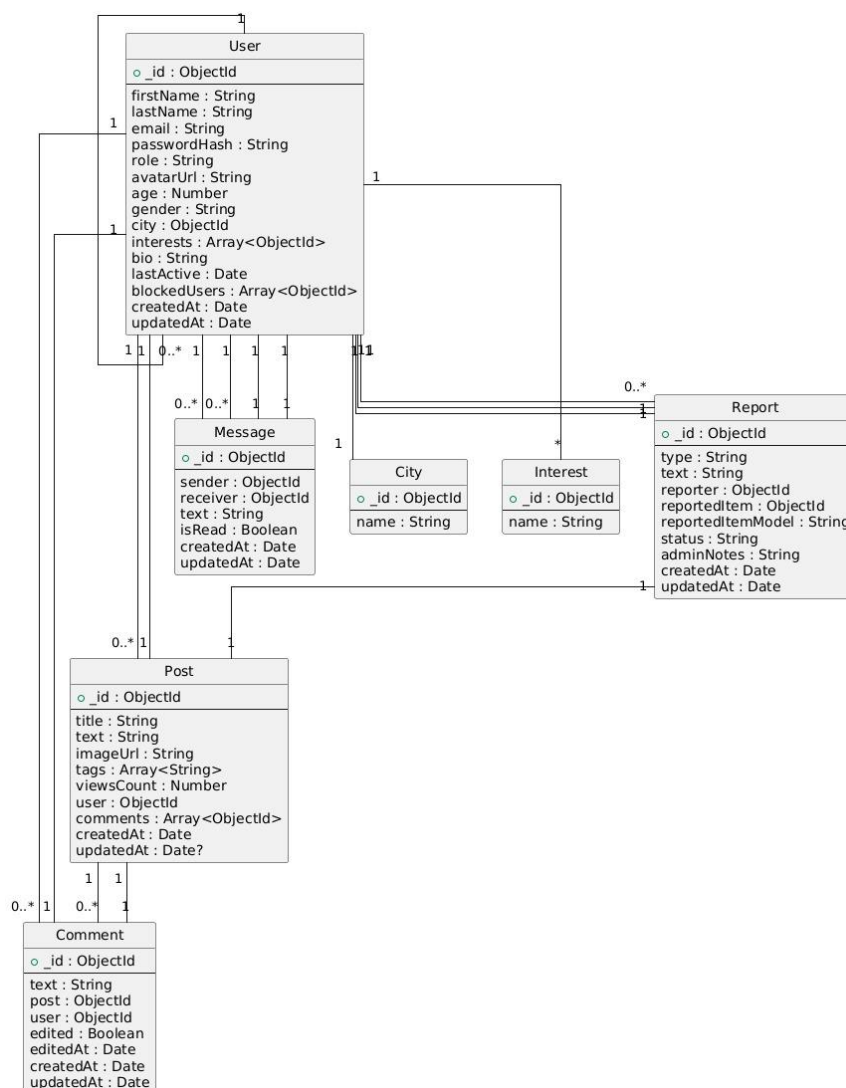


Рис. 2.4 Структура БД

Для зберігання даних було обрано MongoDB – документно-орієнтовану БД, що забезпечує можливість роботи зі складними структурами даних. Для функціонування додатку було побудовано відповідні колекції даних, які відображають основні сутності системи. Зокрема, були створені: User, Message, City, Interest, Report, Post та Comment, кожна з яких відповідає за взаємодію користувача із застосунком.

Структура та опис полів кожної таблиці подані в табл. 2.16 – 2.22.

Таблиця 2.16

Структура таблиці «User»

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор користувача
firstName	String	Ім'я користувача
lastName	String	Прізвище користувача
email	String	Email користувача (унікальний)
passwordHash	String	Хеш пароля
role	String	Роль користувача (user або admin)
avatarUrl	String	Посилання на аватар
age	Number	Вік користувача
gender	String	Стать користувача
city	ObjectId	Посилання на місто
interests	Array<ObjectId>	Список інтересів користувача
bio	String	Біографія користувача
lastActive	Date	Дата останньої активності
blockedUsers	Array<ObjectId>	Список заблокованих користувачів
createdAt	Date	Дата створення
updatedAt	Date	Дата останнього оновлення

Таблиця 2.17

Структура таблиці «Post»

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор допису
title	String	Заголовок допису

Продовження таблиці 2.17

Поле	Тип	Призначення
text	String	Основний текст допису
imageUrl	String	Посилання на зображення
tags	Array<String>	Список тегів
viewsCount	Number	Кількість переглядів
user	ObjectId	Автор допису (посилання на User)
comments	Array<ObjectId>	Список коментарів
createdAt	Date	Дата створення
updatedAt	Date	Дата останнього редагування

Таблиця 2.18

Структура таблиці «Comment»

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор коментаря
text	String	Текст коментаря
post	ObjectId	Посилання на допис
user	ObjectId	Автор коментаря
edited	Boolean	Ознака редагування
editedAt	Date	Дата редагування
createdAt	Date	Дата створення
updatedAt	Date	Дата останнього оновлення

Таблиця 2.19

Структура таблиці «Message»

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор повідомлення
sender	ObjectId	Відправник повідомлення
receiver	ObjectId	Одержувач повідомлення
text	String	Текст повідомлення
isRead	Boolean	Ознака прочитання
createdAt	Date	Дата створення
updatedAt	Date	Дата останнього оновлення

Таблиця 2.20

Структура таблиці «City»

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор міста
name	String	Назва міста

Таблиця 2.21

Структура таблиці «Interest»

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор інтересу
name	String	Назва інтересу

Таблиця 2.22

Структура таблиці «Report»

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор користувача
type	String	Тип скарги (post або profile)
text	String	Текст скарги
reporter	ObjectId	Хто подав скаргу
reportedItem	ObjectId	На що подана скарга
reportedItemModel	String	Модель об'єкта скарги
status	String	Статус розгляду
createdAt	Number	Дата створення
updatedAt	String	Дата останнього оновлення

Використання MongoDB було зумовлено її гнучкість та здатністю ефективно працювати зі складними структурами даних. Для інтеграції з серверною частиною використано Mongoose – бібліотеку, яка спрощує роботу з базою завдяки валідації, зв'язками між колекціями та обробкою даних. Хмарна платформа MongoDB Atlas забезпечила зручне розгортання, масштабування та високу доступність БД. Обрані технології відповідають вимогам до програмного продукту.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

2.3 Проектування та реалізація алгоритмів роботи системи

Проектування алгоритмів системи допомагає завчасно визначити послідовність дій та процесів, яких необхідно дотримуватись для реалізації поставлених функціональних завдань. Тому були створені діаграми активності, що демонструють послідовність дій, процеси прийняття рішень та взаємодію користувачів з системою у процесі роботи.

Нижче діаграма активності (рис. 2.5) показує процес реєстрації та авторизації в застосунку.

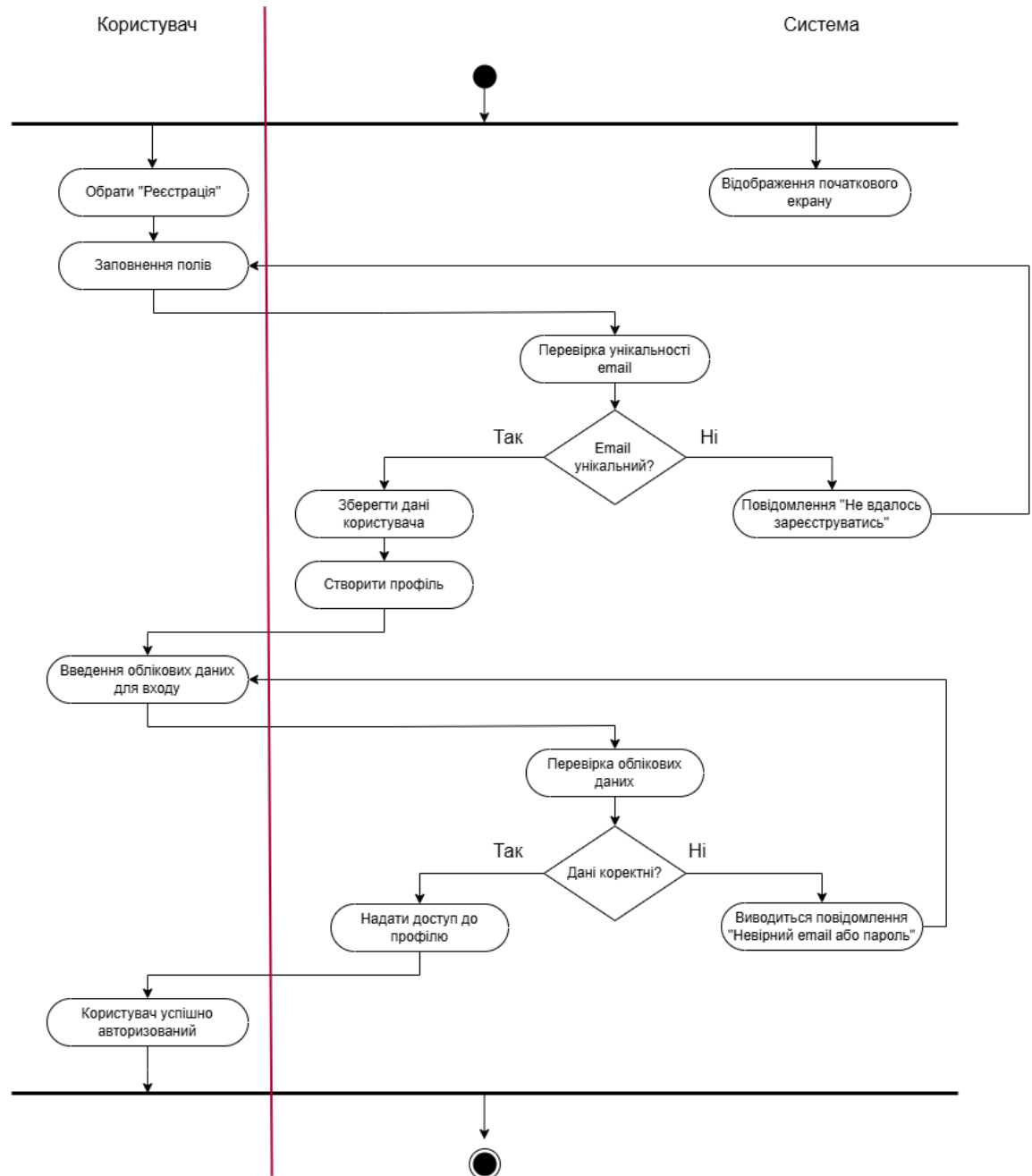


Рис. 2.5 Діаграма активності для авторизації та реєстрації

Процес реєстрації починається з дії користувача, натиснення на кнопку входу та перехід до створення акаунта. Після цього система відображає реєстраційну форму, в якій користувач заповнює поля, серед яких ім'я, email і пароль. Якщо email є унікальним, система зберігає введені дані користувача і створює новий профіль, якщо введений email вже існує, користувач інформується про помилку і процес реєстрації завершується без успіху, у такому випадку має можливість спробувати знову.

Далі розглянемо процес редагування власного профілю (рис. 2.6).

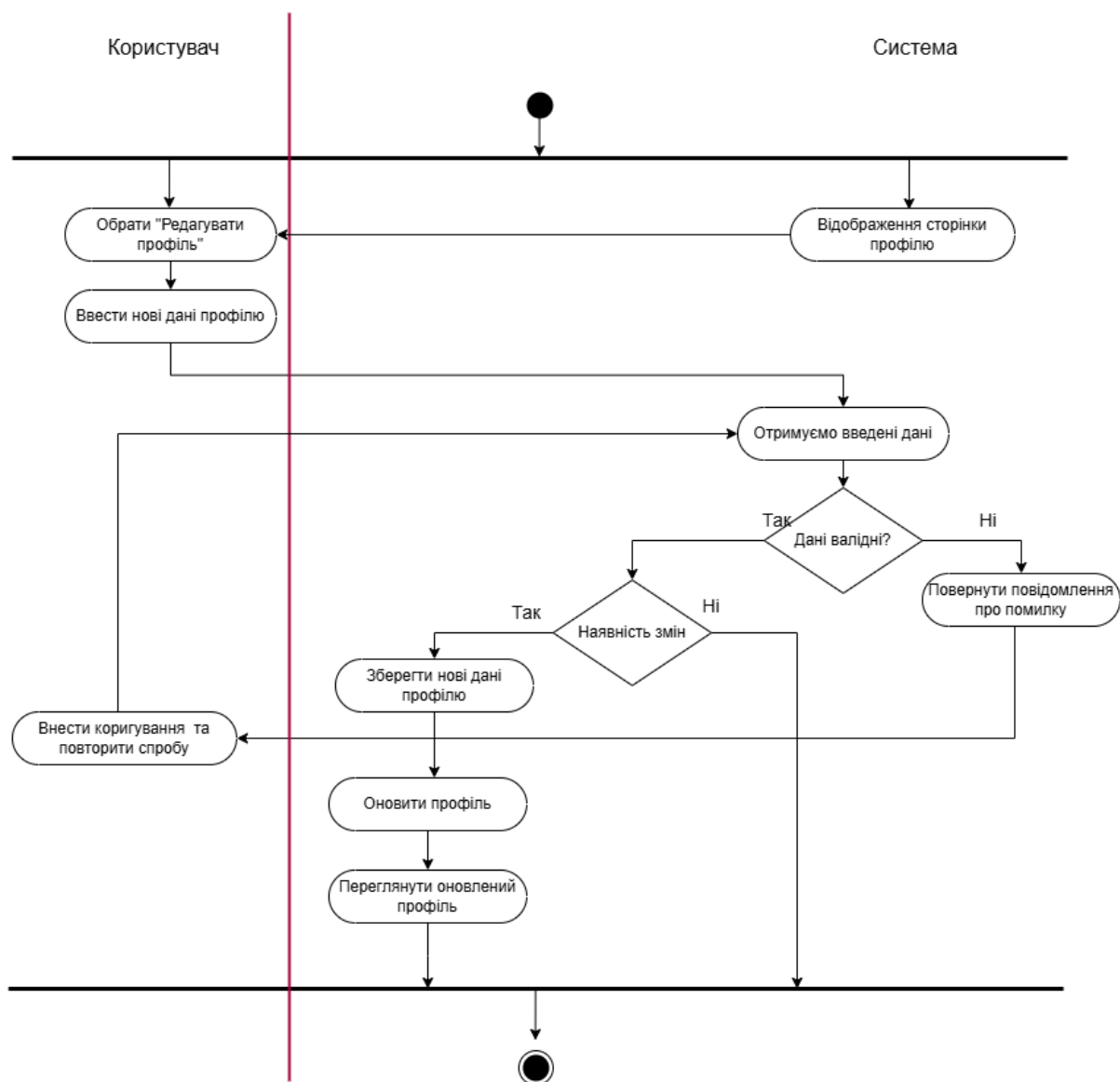


Рис. 2.6 Діаграма активності редагування свого профілю

Редагування профілю починається з натиску на кнопку редагувати, що відкриває заповнену форму поточними даними користувача. Після внесення змін до своїх даних виконується перевірка коректності, у випадку

невідповідності правилам, користувача буде проінформовано із можливістю виправлення. Якщо внесені зміни повністю відповідають вимогам, то виконується порівняння полів із минулою версією, при наявності змін оновлюються відповідні поля у базі. Зміни можна відразу побачити у своєму профілі, такий механізм забезпечує підтримку актуальності даних.

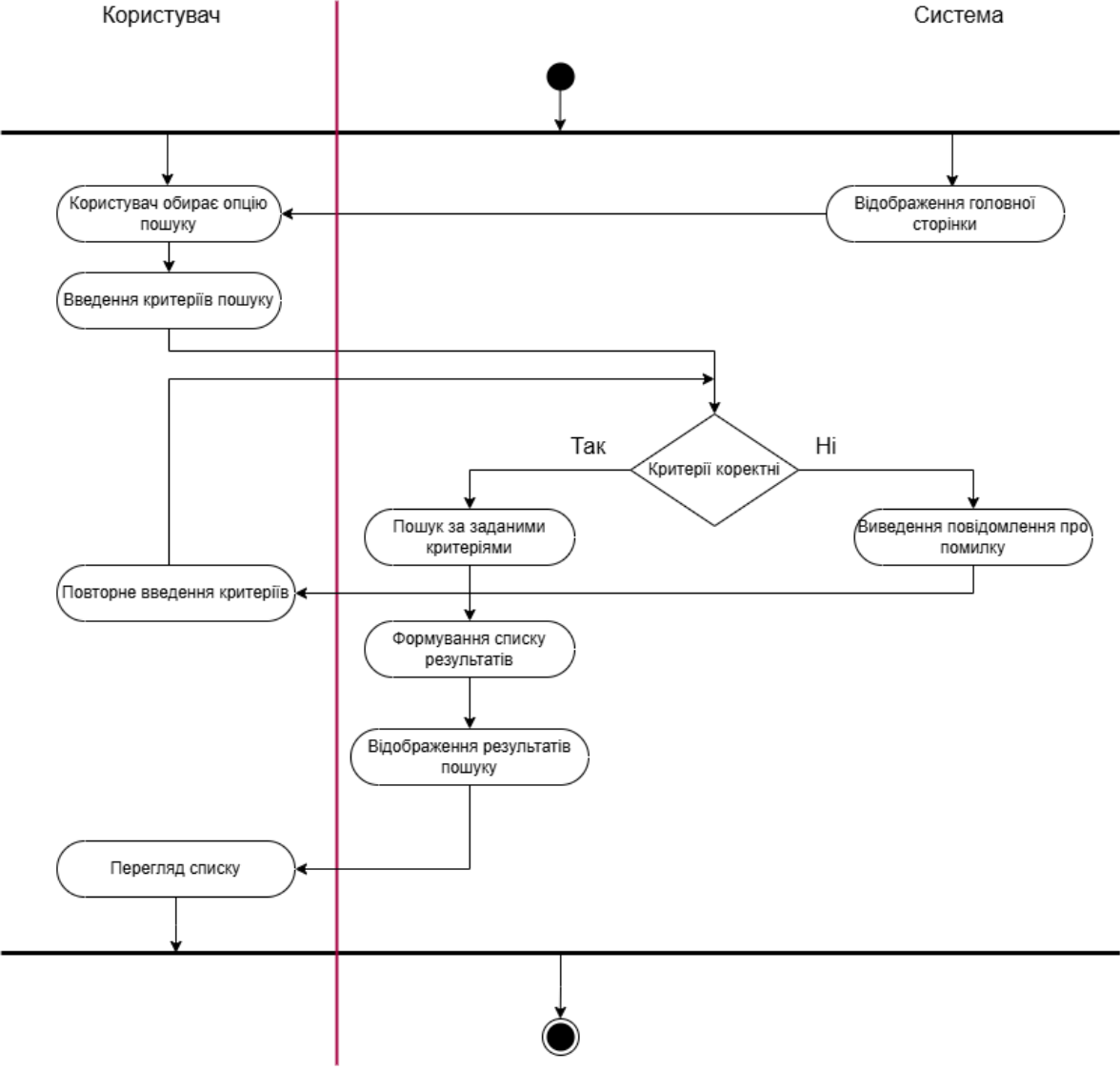


Рис. 2.7 Діаграма активності пошуку користувачів

Діаграма активності пошуку користувачів демонструє взаємодію користувача із застосунком під час пошуку інших (рис. 2.7). На сторінці відображаються всі користувачі системи. За потреби можна налаштувати параметри та повторити пошук, сервер обробляє запит та готує список результатів для перегляду. Цей цикл може повторюватись безліч разів до отримання бажаного результату.

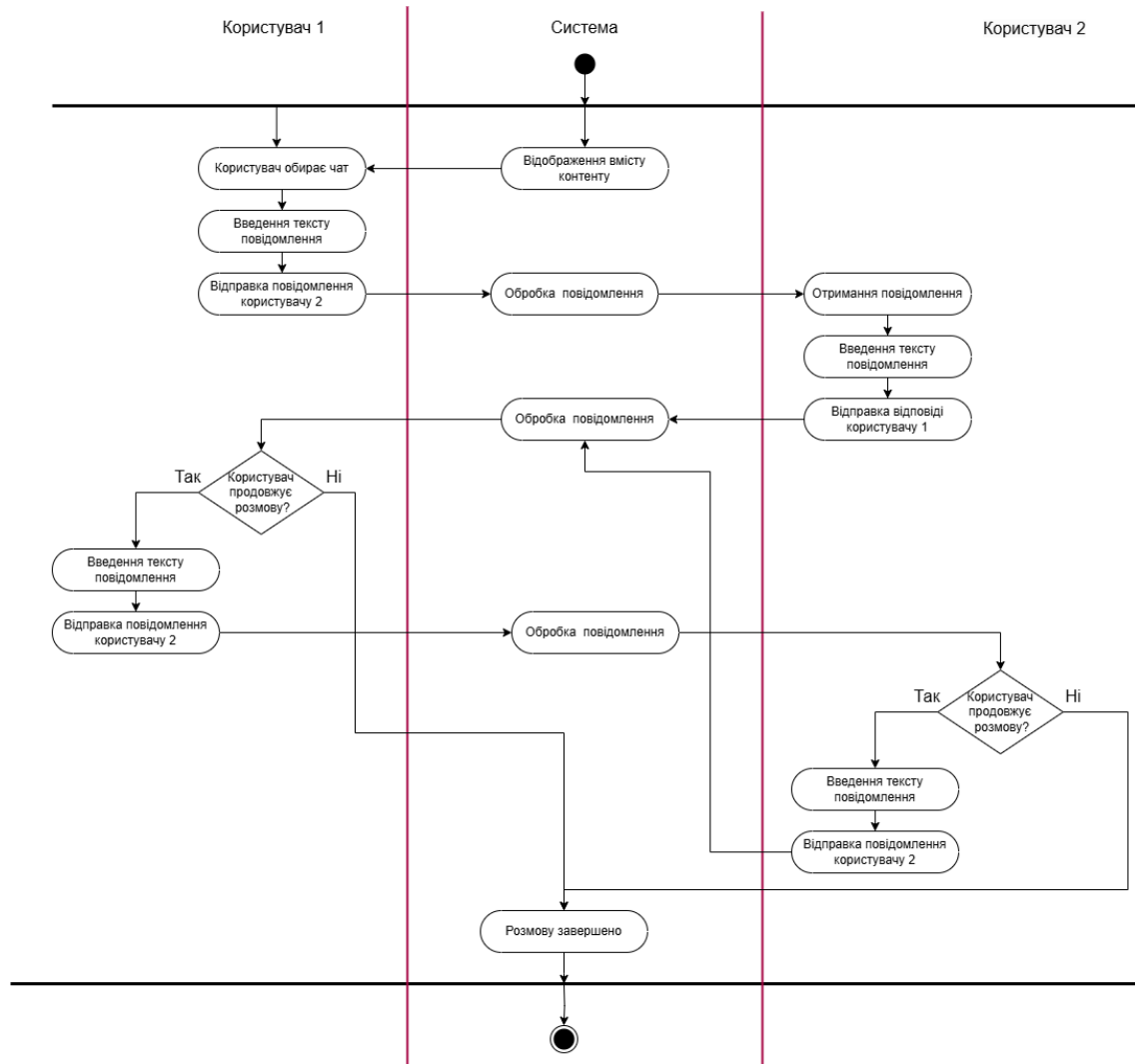


Рис. 2.8 Діаграма активності обміну повідомленнями

Діаграма активності обміну повідомленнями демонструє взаємодію двох користувачів в межах чату (рис. 2.8). Виконання починається з того, що один користувач відкриває вікно чату, вводить текст повідомлення та надсилає його. Система обробляє його із зберіганням до бази та в реальному часі відправляє іншому користувачу, який відразу бачить його на екрані без оновлення сторінки. Якщо співрозмовник вирішує підтримати розмову, то аналогічно набирає текст повідомлення та відправляє. Повідомлення, які були розглянути позначаються прочитаними. Обмін повідомленнями триває допоки обидва користувачі надсилають нові, якщо один із користувачів переходить до іншого чату або сторінки застосунку, всі нові повідомлення будуть позначатись як не прочитані.

2.4 Реалізація функціоналу вебзастосунок для пошуку нових знайомств

Однією з ключових функцій вебзастосунок для пошуку нових знайомств є реєстрація нових користувачів. І цей процес має не лише бути зручним, але й забезпечувати безпеку та надійність, тому що на цьому етапі формується перше враження про застосунок.

Нижче розглянемо реалізацію реєстрації користувача, що відразу забезпечує заповнення профілю (рис.2.9).

```
export const register = async (req, res) => {
  try {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json(errors.array());
    }
    const { email, password, firstName, lastName, age, gender, city, interests, bio } = req.body;
    const salt = await bcrypt.genSalt(10);
    const hash = await bcrypt.hash(password, salt);

    let avatarUrl = null;
    if (req.file) {
      avatarUrl = `/uploads/${req.file.originalname}`;
    }
    const doc = new UserModel({
      email,
      firstName,
      lastName,
      avatarUrl,
      age,
      gender,
      city,
      interests: JSON.parse(interests),
      bio,
      passwordHash: hash,
    });
    const user = await doc.save();
    const token = jwt.sign({ _id: user._id }, JWT_SECRET, { expiresIn: '30d' });
    const userData = user.toObject ? user.toObject() : user._doc;
    const { passwordHash, ...userWithoutHash } = userData;

    res.json({
      ...userWithoutHash,
      token,
    });
  } catch (err) {
    console.error('Помилка реєстрації:', err);
    res.status(500).json({
      message: 'Не вдалось зареєструватись!',
    });
  }
};
```

Рис. 2.9 Метод реєстрації

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Для того щоб користувачі могли завантажувати зображення профілю та доповнювати свої дописи було важливим реалізувати функціонал завантаження файлів, який забезпечить безпечну обробку та збереження даних на сервері (рис.2.10).

```
const storage = multer.diskStorage({
  destination: (_, __, cb) => {
    if (!fs.existsSync('uploads')) {
      fs.mkdirSync('uploads');
    }
    cb(null, 'uploads');
  },
  filename: (_, file, cb) => {
    cb(null, file.originalname);
  },
});

const upload = multer({ storage });
app.use('/uploads', express.static('uploads'));
app.use(express.json());

app.use('/uploads', express.static('uploads'));
app.post('/upload', upload.single('image'), (req, res) => {
  res.json({
    url: `/uploads/${req.file.originalname}`,
  });
});
```

Рис. 2.10 Робота із зображеннями

Валідації полів на серверній під час реєстрації або авторизації було створено окрему допоміжну функцію, яка містить набори правил для перевірки вхідних даних зі встановленими обмеженнями (рис. 2.11).

```
export const registerValidator = [
  body('email', 'Невірний формат ел.пошти').isEmail(),
  body('password', 'Пароль має містити щонайменше 8 символів').isLength({ min: 8 }),
  body('firstName', 'Вкажіть ім'я').isLength({ min: 2 }),
  body('lastName', 'Вкажіть прізвище').isLength({ min: 2 }),
  body('avatarUrl', 'Некоректне посилання на аватар').optional().isURL(),
  body('age', 'Вік обов'язковий').isInt({ min: 18, max: 100 }),
  body('gender', 'Стать обов'язкова').isIn(['male', 'female', 'other']),
  body('city', 'Вкажіть місто').isLength({ min: 2 }),
  body('interests', 'Вкажіть місто').isLength({ min: 2 }),
];

export const loginValidator = [
  body('email', 'Невірний формат ел.пошти').isEmail(),
  body('password', 'Пароль має містити щонайменше 8 символів').isLength({ min: 8 }),
];
```

Рис. 2.11 Обробник помилок реєстрації та авторизації

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Після успішної авторизації користувач перенаправляється на сторінку профілю, де автоматично виконується запит інформації про себе для відображення інформації у профілі (рис. 2.12).

```
export const getMe = async (req, res) => {
  try {
    const user = await UserModel.findById(req.userId).populate('city').populate('interests');

    if (!user) {
      return res.status(404).json({
        message: 'Користувача не знайдено',
      });
    }

    const userData = user.toObject ? user.toObject() : user._doc;
    const { passwordHash, ...userWithoutHash } = userData;

    res.json({
      ...userWithoutHash,
    });
  } catch (err) {
    console.error('Error details:', {
      message: err.message,
      stack: err.stack,
      userId: req.userId,
    });
    res.status(500).json({
      message: 'Немає доступу!',
    });
  }
};
```

Рис. 2.12 Отримання інформації про себе

В той час наявність доступу перевіряє допоміжна функція, яка перевіряє чи виконується запит від користувача з дійсною інформацією. Перевірка забезпечує безпеку системи запобігаючи неправомірний доступ до захищених ендпоінтів (рис. 2.13).

```
export default async (req, res, next) => {
  const token = (req.headers.authorization || '').replace(/Bearer\s?/, '');

  if (token) {
    try {
      const decoded = jwt.verify(token, JWT_SECRET);
      req.userId = decoded._id;

      try {
        await UserModel.findByIdAndUpdate(req.userId, { lastActive: new Date() });
      } catch (err) {
        console.error('Не вдалося оновити lastActive:', err);
      }

      next();
    } catch (e) {
      return res.status(403).json({
        message: 'Немає доступу!',
      });
    }
  } else {
    return res.status(403).json({
      message: 'Немає доступу',
    });
  }
};
```

Рис. 2.13 Перевірка автентифікації

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Для пошуку нових знайомств користувачу необхідно перейти на сторінку пошуку, де він отримує усіх користувачів системи (рис. 2.14).

```
export const getAll = async (req, res) => {
  try {
    const users = await UserModel.find()
      .populate('city')
      .populate('interests')
      .select('-passwordHash');

    res.json(users);
  } catch (err) {
    console.error('Помилка при отриманні користувачів:', err);
    res.status(500).json({
      message: 'Не вдалось отримати список користувачів',
    });
  }
};
```

Рис. 2.14 Перегляд усіх користувачів

За потреби користувач може звужувати пошук налаштовуючи певні критерії і отримуючи більш точні результати (рис. 2.14).

```
const searchUsersSlice = createSlice({
  name: 'searchUsers',
  initialState,
  reducers: {
    setFilters: (state, action) => {
      const newFilters = { ...state.filters, ...action.payload };
      state.filters = newFilters;
      const filtered = state.users.filter((user) => {
        const fullName = `${user.firstName} ${user.lastName}`.toLowerCase();
        const matchesName =
          newFilters.name === '' || fullName.includes(newFilters.name.toLowerCase());
        const matchesGender = newFilters.gender === '' || user.gender === newFilters.gender;
        const matchesCity = newFilters.city === '' || user.city?._id === newFilters.city;
        const matchesAge = user.age >= newFilters.minAge && user.age <= newFilters.maxAge;
        const matchesInterests =
          newFilters.interests.length === 0 ||
          newFilters.interests.every((interest) =>
            user.interests.some((userInterest) => userInterest._id === interest)
          );
        return matchesName && matchesGender && matchesCity && matchesAge && matchesInterests;
      });
    }
  }
});
```

Рис. 2.15 Пошук за параметрами

Як додатковий механізм для взаємодії між користувачами було прийнято рішення реалізувати функціонал дописів. Який допомагає ділитись корисною інформацією або висловлювати власні думки, що дає змогу бути більш активними у застосунку та проявляти себе.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Розглянем функціонал створення допису (рис. 2.16).

```
✓ export const create = async (req, res) => {
  ✓ try {
    const tags = req.body.tags
      ? req.body.tags
        .split(',')
        .map((tag) => tag.trim().toLowerCase())
        .filter(Boolean)
      : [];
    ✓ const doc = new PostModel({
      title: req.body.title,
      text: req.body.text,
      imageUrl: req.body.imageUrl,
      tags,
      user: req.userId,
    });

    const post = await doc.save();

    res.json(post);
  } catch (err) {
    ✓ console.log(err);
    ✓ res.status(500).json({
      message: 'Не можливо створити статтю',
    });
  }
};
```

Рис. 2.16 Функціонал створення допису

Висновки до другого розділу

Під час проєктування вебзастосунку для пошуку нових знайомств було проведено аналіз основних варіантів використання системи відповідно до розподілу прав між ролями користувачів та визначено вимоги щодо функціональності. На основі цього було реалізовано архітектуру застосунку з чітким розподілом серверної та клієнтської частин і їх взаємодією.

Було створено алгоритми роботи застосунку та побудовано діаграми активності основних варіантів використання, що допомогло детально описати функціонал застосунку.

Реалізовано структуру БД, відповідно до потреб застосунку, яка забезпечила ефективне зберігання, пошук та обробку інформації. Особливу увагу було приділено зв'язкам між сутностями для точного відображення взаємодій між користувачами, повідомленнями та іншими елементами.

У функціональній реалізації було продемонстровано частину функціоналу, яка охоплює ключові аспекти взаємодії користувача із застосунком. Зокрема, було реалізовано механізми реєстрації та автентифікації із передбаченою обробкою помилок, функціонал завантаження користувацьких зображень на сервер, отримання інформації про себе та інші елементи, які забезпечують важливу функціональність.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ

3.1 Порядок встановлення та налаштування параметрів застосунку

Встановлення та налаштування параметрів вебзастосунку для пошуку нових знайомств передбачає поетапне встановлення всіх його компонентів із подальшим налаштуванням кожного, а також комплексне налаштування застосунку в цілому для забезпечення його ефективної роботи. Застосунок складається з трьох основних компонентів: клієнтського інтерфейсу, серверної логіки та хмарної БД (рис. 3.1).

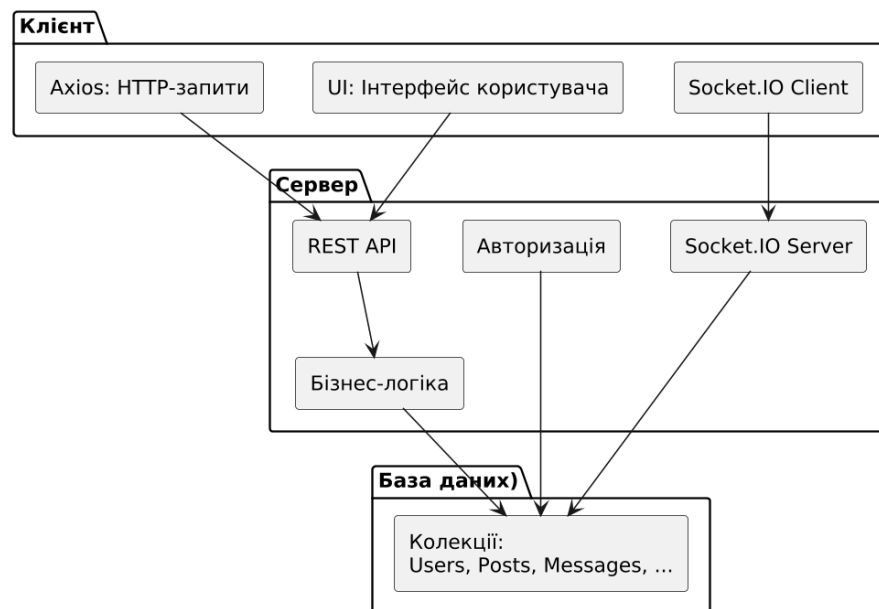


Рис. 3.1 Діаграма компонентів

Оскільки сервер є джерелом даних його налаштування виконується першочергово. Серверна частина реалізована середовищем виконання Node.js у поєднанні з фреймворком Express.js, що забезпечує виконання бізнес логіки та обробку запитів. На цьому рівні реалізовано обробку HTTP-запитів, а також механізми авторизації, взаємодії з БД та повне управління всіма сутностями системи. Оскільки архітектура серверної частини побудована за принципом REST API, у такому випадку кожен ендпоінт відповідає за окрему функції. Для роботи із БД була використана бібліотека Mongoose, що дозволяє визначати структуру документів та встановлює зв'язки між ними [16]. Моделі користувачів створюються у вигляді окремих модулів, які підключаються до основного серверного застосунку. Запуск серверної частини передбачає

створення конфігураційного файлу .env у якому задається пароль адміністратора до БД та секрет формування токенів авторизації.

Після налаштування серверної частини переходимо до встановлення та налаштування клієнтського інтерфейсу. У нашому випадку він реалізований за допомогою React, який відповідає за відображення інтерфейсу, обробку дій користувача та його взаємодію із сервером за допомогою Axios та WebSocket-з'єднання. Для запуску клієнтською частини необхідно встановити всі залежності, після чого вона буде готова до розгортання [23].

Третім етапом є налаштування хмарної БД MongoDB Atlas, яка використовується для інформації про користувачів, дописи, коментарі, повідомлення, інтереси, міст та скарги. Потрібно створити новий кластер, налаштувати права доступу, створивши нового користувача БД, а також сформувати посилання для підключення, пароль до якого зберігається у конфігураційному файлі серверної частини. Структура даних визначена через Mongoose схеми, які автоматично створяться при першому запуску серверу.

Після завершення конфігурації застосунок готовий до взаємодії з користувачами. Для кращого розуміння структури та зв'язків між компонентами було розроблено діаграму розгортання, що демонструє взаємодію клієнта, сервера та БД (рис. 3.2).

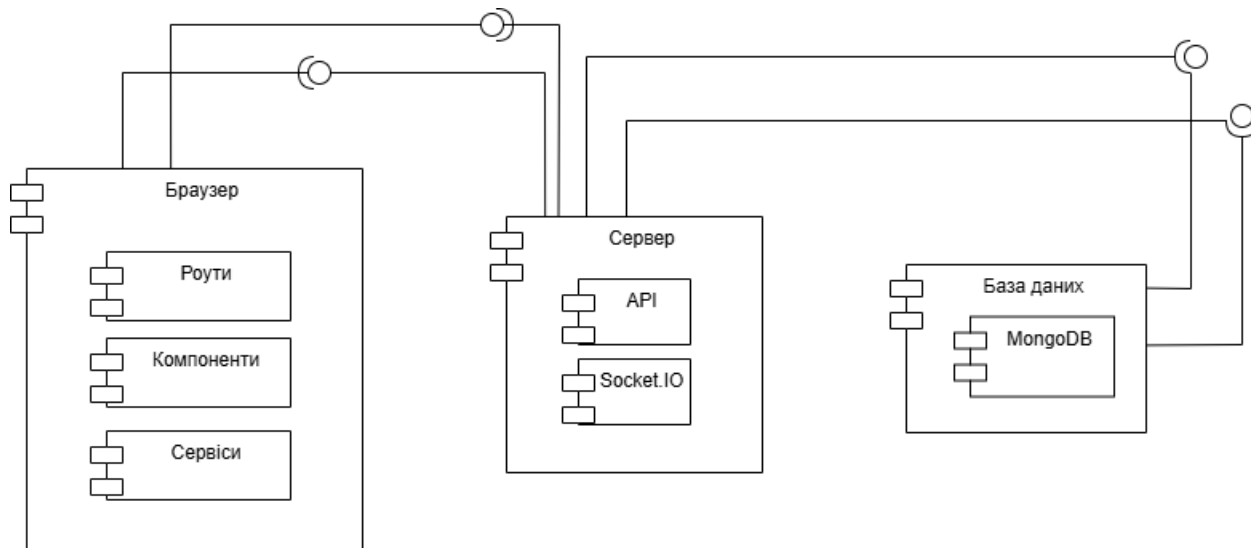


Рис. 3.2 Діаграма розгортання

3.2 Структура інтерфейсу та порядок взаємодії із застосунком

Під час першого візиту користувач натрапляє на головну сторінку, що містить декілька інформаційних блоків та кнопку входу. Інформаційні блоки коротко ознайомлюють із можливостями вебзастосунку (рис. 3.3).

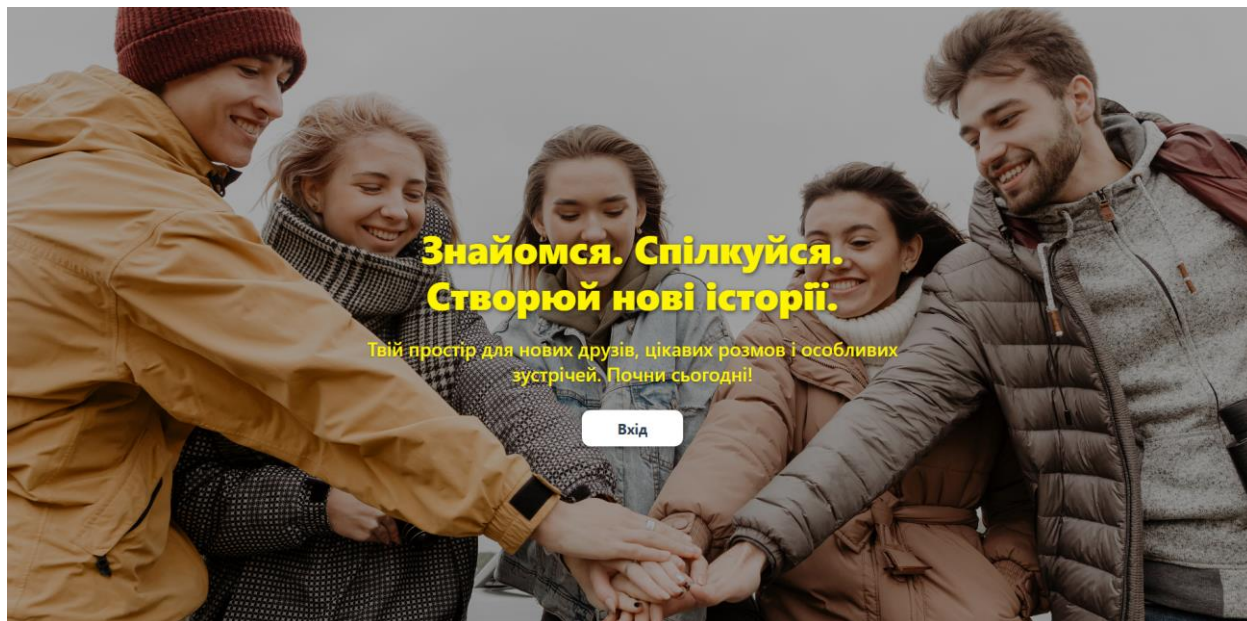


Рис. 3.3 Головна сторінка

Якщо користувач не має облікового запису, він може створити новий, повністю заповнюючи всі поля та опціонально завантажуючи зображення профілю. Усі поля мають вбудовану валідацію, що допомагає уникнути помилок під час реєстрації (рис. 3.4).

Рис. 3.4 Сторінка реєстрації

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

За наявності облікового запису необхідно перейти на сторінку авторизації у застосунок, де необхідно заповнити поля електронної пошти та пароллю для доступу у систему, при введенні некоректних даних зв’яжуться відповідні повідомлення про помилку (рис. 3.5).

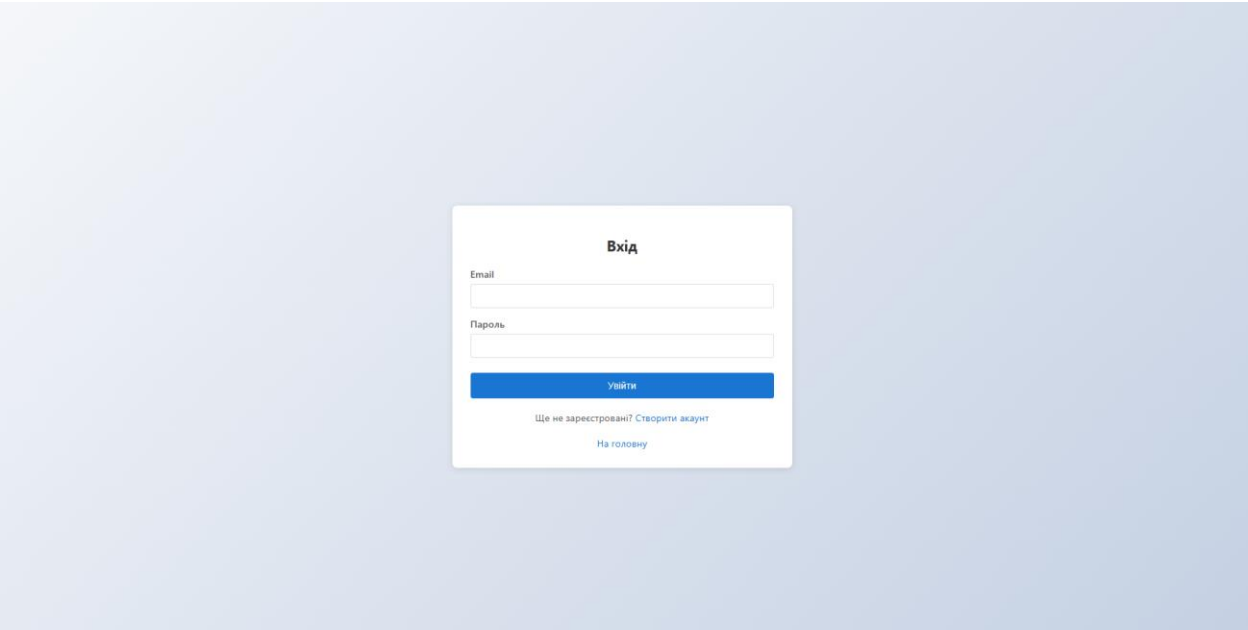


Рис. 3.5 Сторінка авторизації

Після авторизації користувача перенаправляє на сторінку особистого профілю (рис. 3.6). Профіль відображає інформацію заповнену під час реєстрації: ім’я, прізвище, електрону пошту, вік, стать, місто, чим захоплюється користувач та короткий опис про нього.

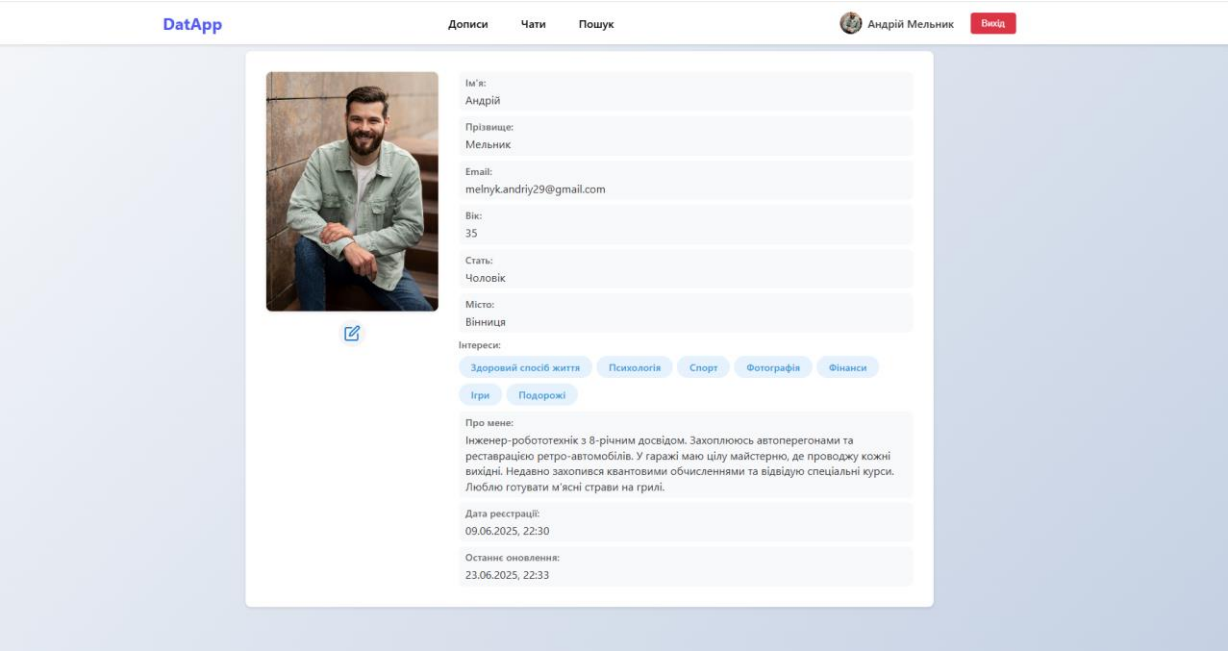


Рис. 3.6 Профіль користувача

Якщо користувач бажає змінити особисту інформацію, він може це зробити, натиснувши кнопку під аватаром, що автоматично відкриє форму для внесення змін із заповненою поточною особистою інформацією (рис. 3.7).

Рис. 3.7 Редагування профілю

Він може змінити будь-яке поле та завантажити новий аватар, внесені зміни відразу побачать інші користувачі. Як тільки користувач забажає знайти інших він переходить на сторінку пошуку (рис. 3.8).

Рис. 3.8 Сторінка пошуку

На сторінці йому буде відображено всіх користувачів застосунку. При необхідності налаштування параметрів пошуку необхідно скористуватись бічною панеллю фільтрації (рис. 3.9).

Рис. 3.9 Панель фільтрації користувачів

Фільтрація може виконуватись за одним полем або відразу за декількома, щоб персоналізувати пошук.

Рис. 3.10 Панель фільтрації користувачів

По закінченню пошуку натиском на картку користувача відкривається профіль іншого користувача (рис. 3.10).

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Годлевський Ю.О.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо було помічено неприпустиму інформацію або поведінку користувача у застосунку є можливість надіслати скаргу (рис. 3.11).

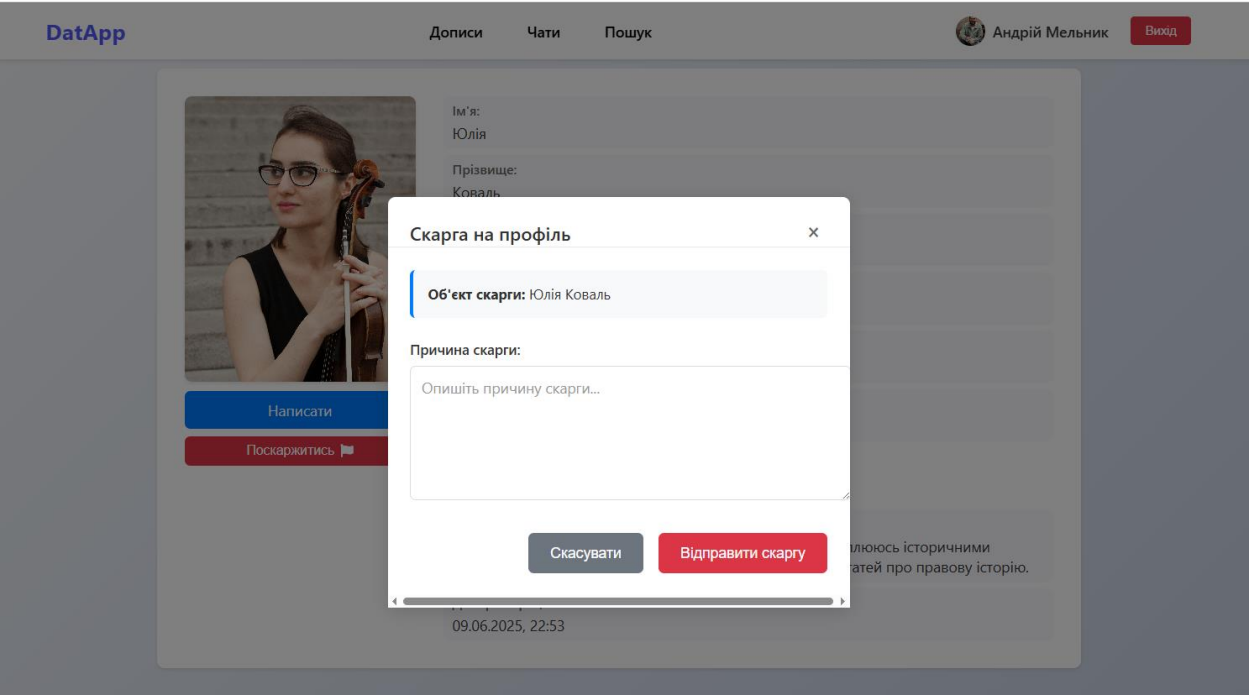


Рис. 3.11 Формування скарги

Буде сформована скарга для подальшого опрацювання адміністратором. Або ж у випадку подальшої взаємодії з користувачем можливо розпочати з ним обмін повідомленнями (рис. 3.12).

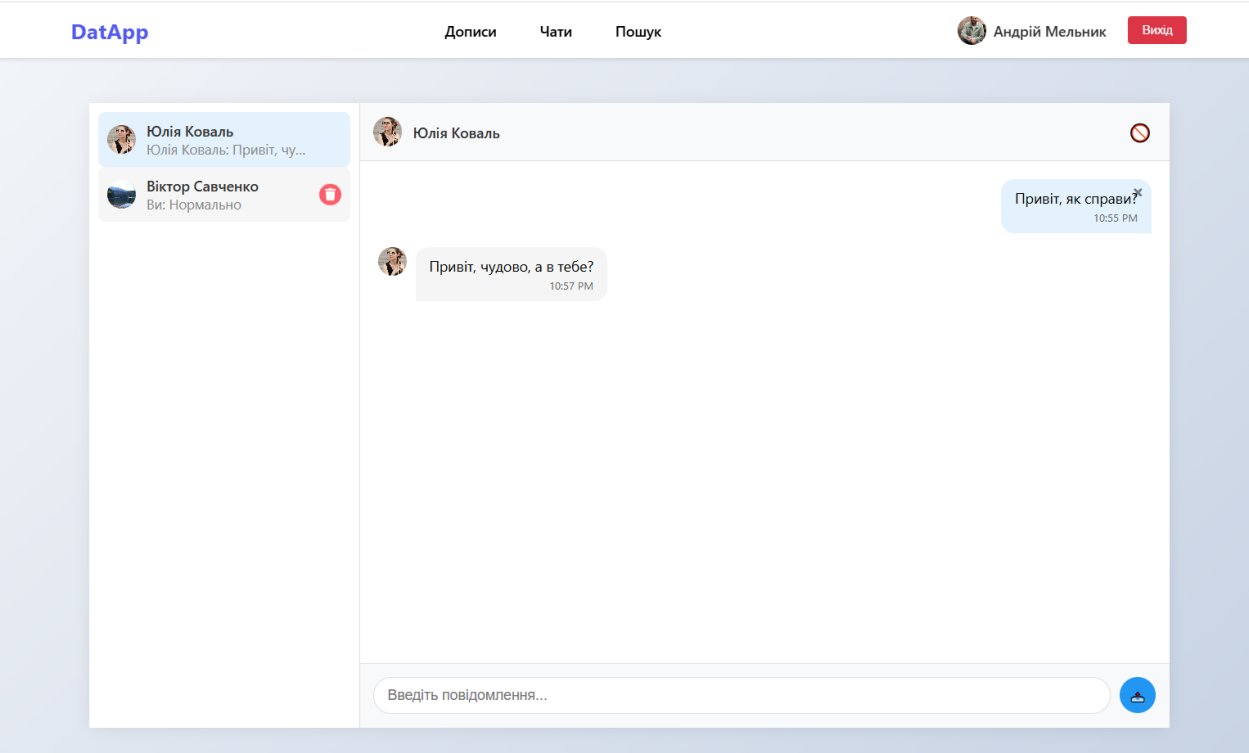


Рис. 3.12 Обмін повідомленнями

При неприпустимій поведінці користувача є можливість його блокування, що йому заборонить надсилати повідомлення (рис. 3.13).

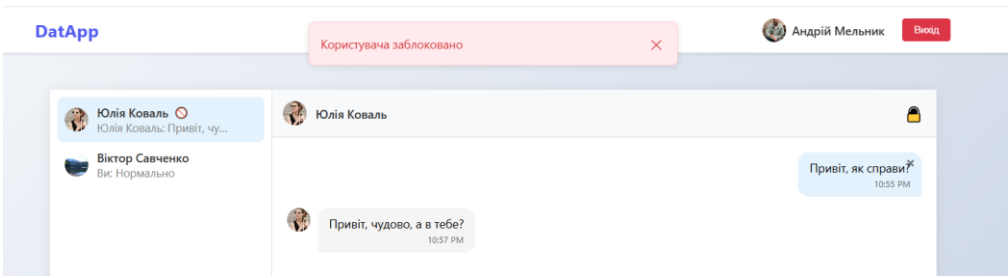


Рис. 3.13 Блокування користувача

Для того щоб прибрати чат доступний функціонал видалення, з видаленням всіх повідомлень між користувачами (рис.3.14).

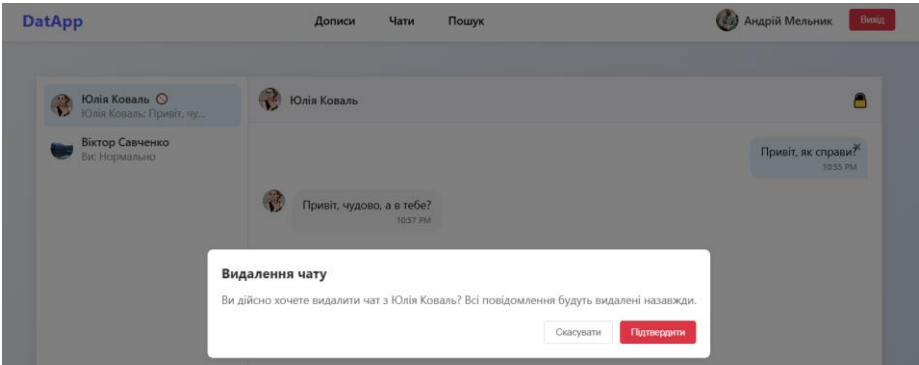


Рис. 3.14 Видалення чату

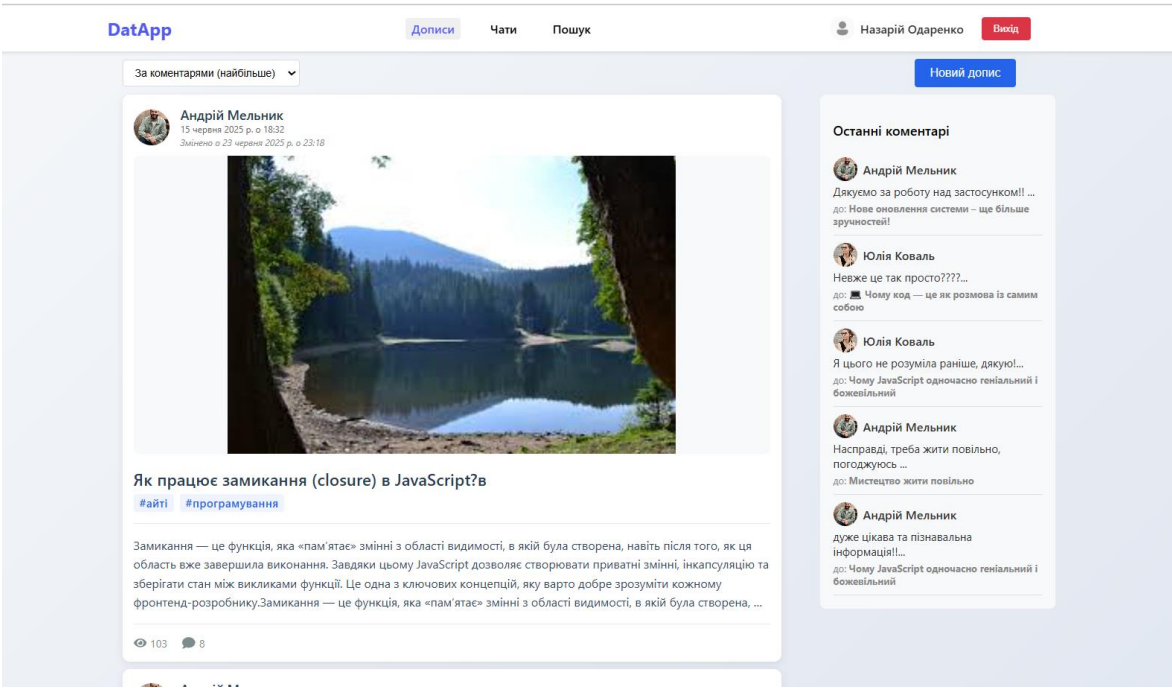


Рис. 3.15 Сторінка дописів

Ще один із механізмів знайомств – сторінка дописів (рис. 3.15) на якій реалізовано функціонал дописів.

На зацікавлений допис можливо перейти по кліку та ознайомитись з його повним описом (рис. 3.16).

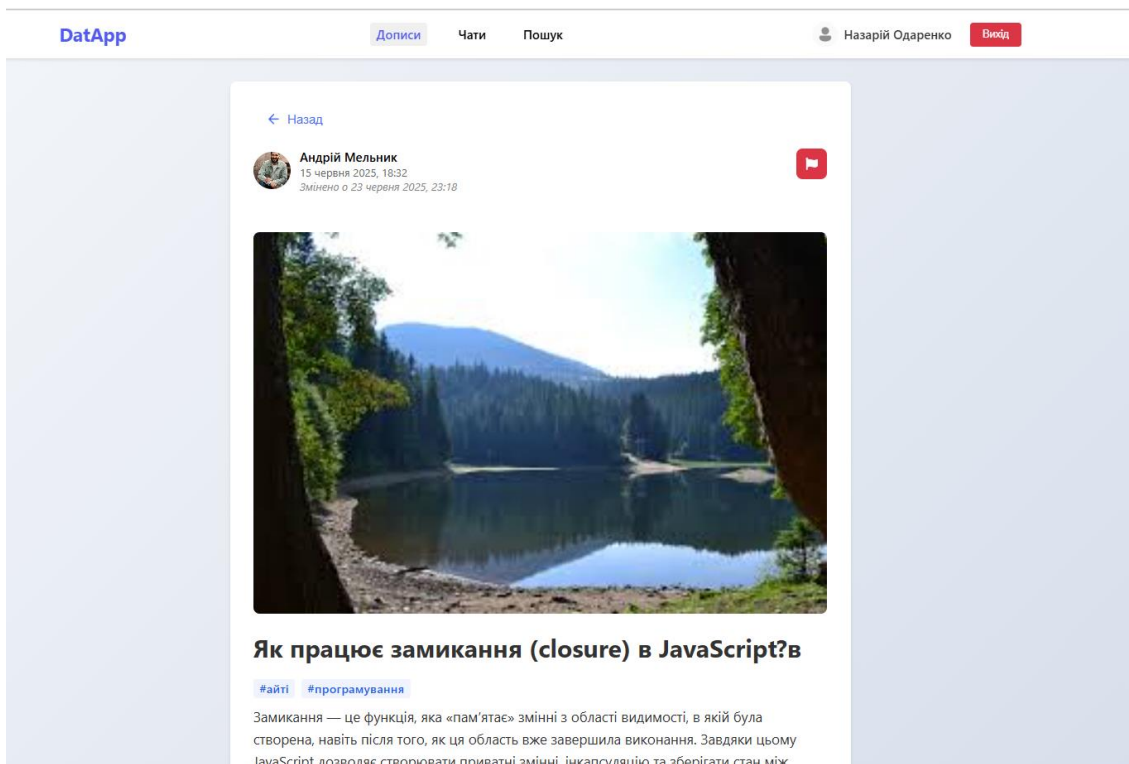


Рис. 3.16 Сторінка окремого допису

Тут користувач може переглядати дописи інших людей, приєднуватись до обговорень у коментарях з іншими користувачами, а також додавати свої для розповсюдження власних думок та захоплень (рис. 3.17).

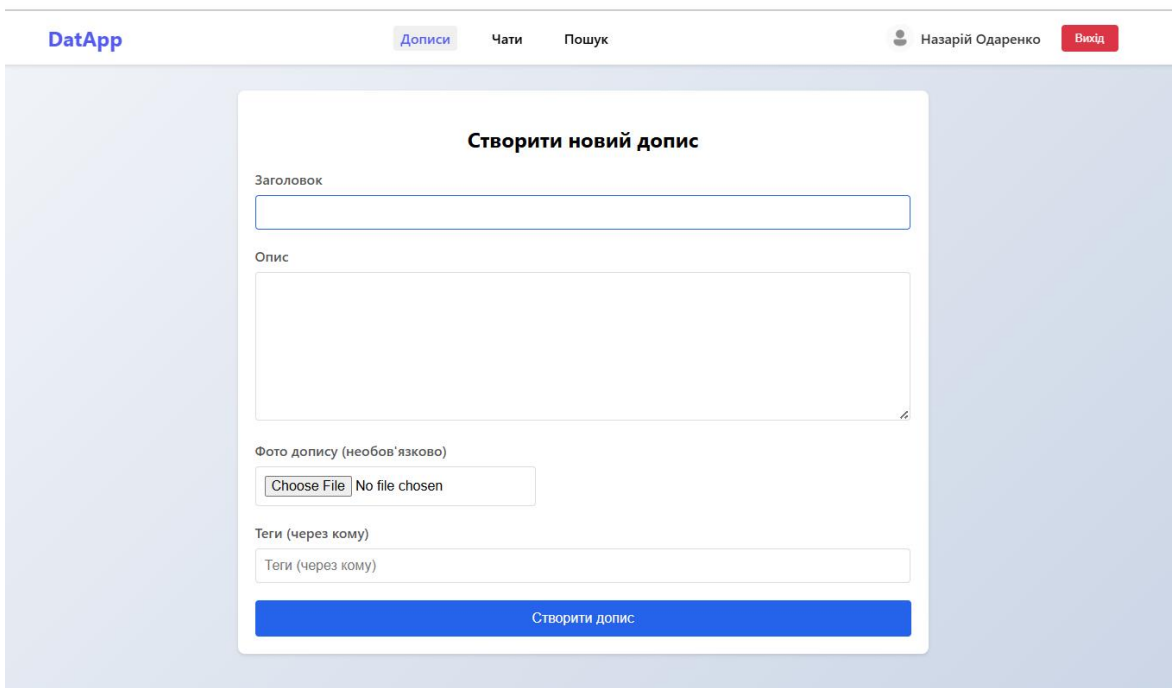


Рис. 3.17 Створення допису

Процес додавання коментарів до допису через елемент інтерфейсу на сторінці повного допису (рис. 3.18).

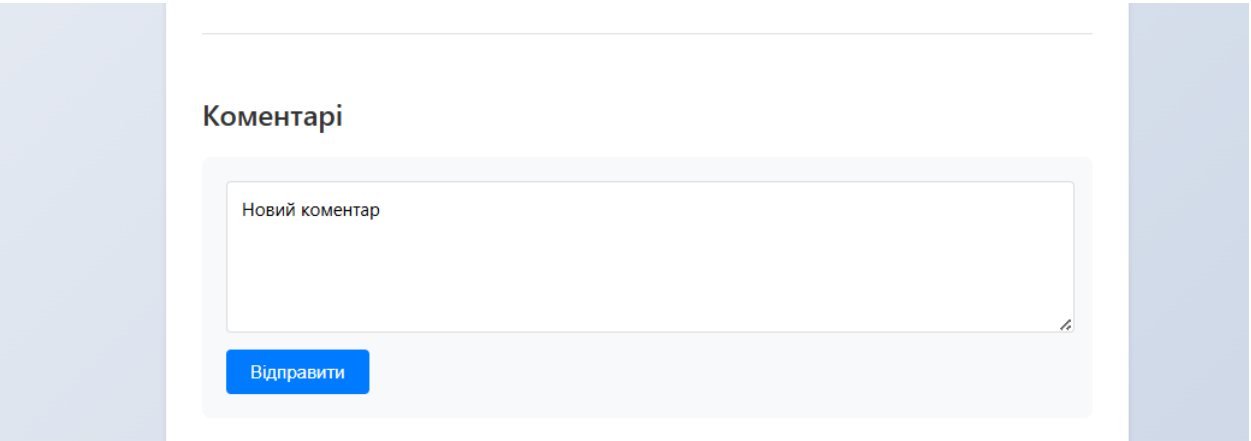


Рис. 3.18 Додавання коментарю

Далі розглянемо додатковий інтерфейс користувача який має права адміністратора. Користувач з правами адміністратора додатково має доступ до адміністративної панелі. Першочергово їй відображається статистика застосунку, що містить в собі: відображення кількості користувачів, активних користувачів, дописів, коментарів та повідомлень. Також може відслідковувати активність реєстрацій нових користувачів, яка реалізована за допомогою графіку (рис. 3.19).

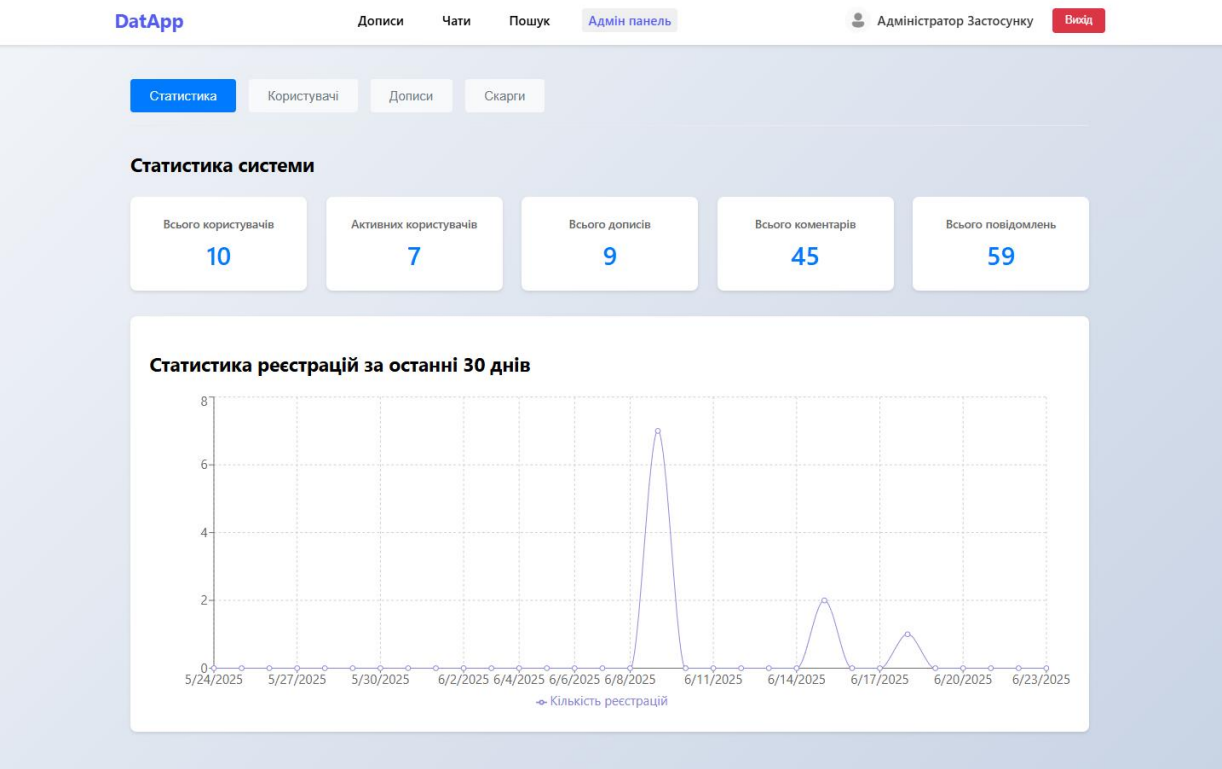


Рис. 3.19 Перегляд статистики

Наступне вікно, яке доступне адміністратору, це перелік користувачів який йому доступний для керування (рис. 3.20).

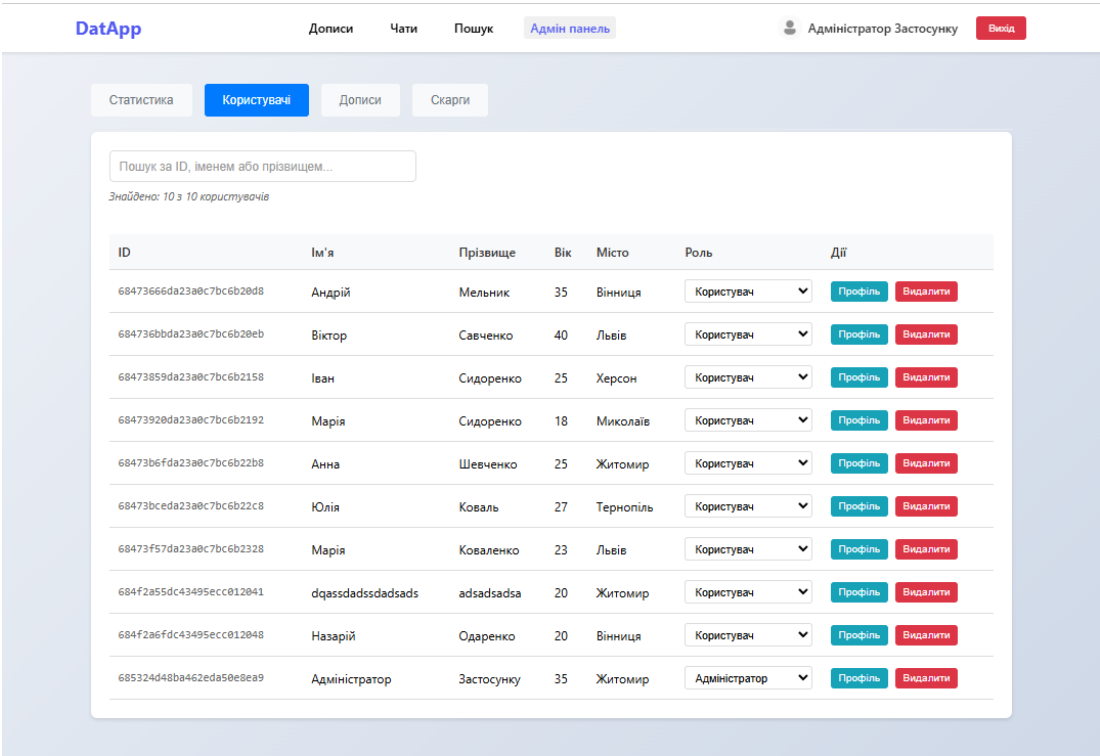


Рис. 3.20 Керування користувачами

Аналогічний функціонал для керування дописами (рис. 3.21).

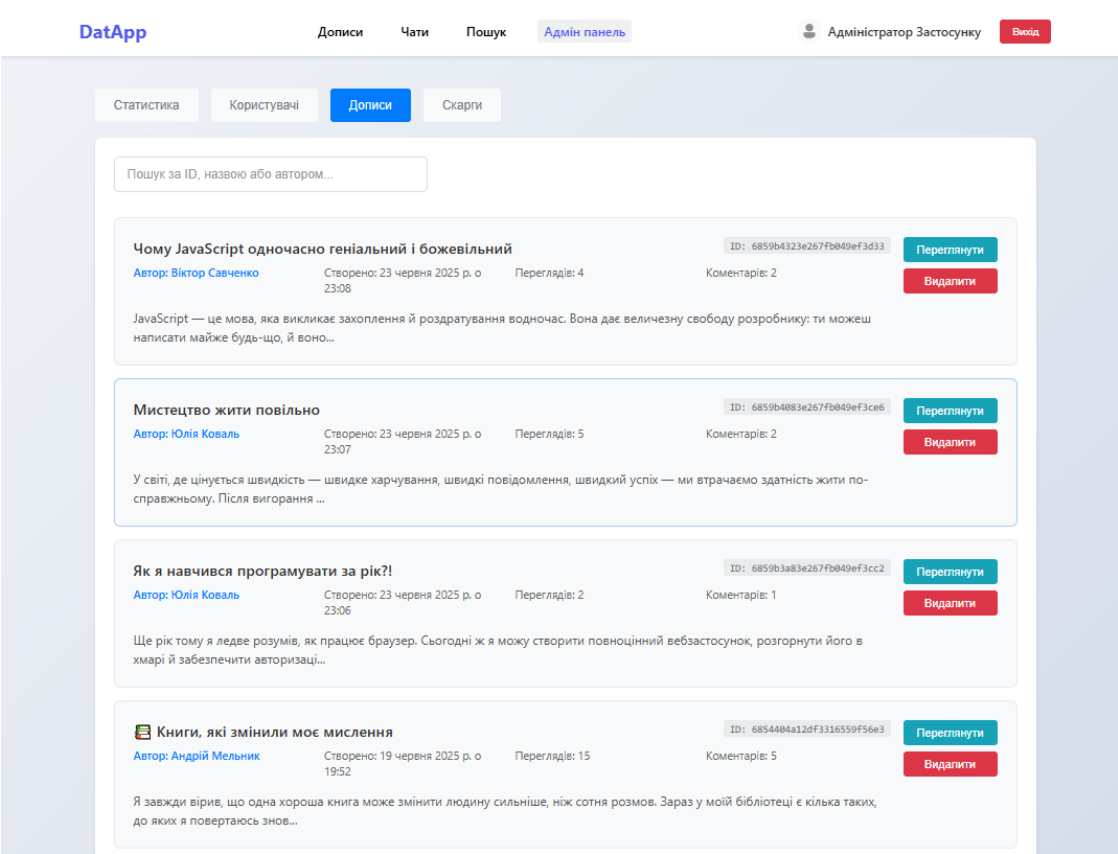


Рис. 3.21 Керування дописами

ДатApp

ДописиЧатиПошукАдмін панель

АдміністраторЗастосункуВихід

СтатистикаКористувачіДописиСкарги

Всі статуси

Всі типи

ID: 6859bf9d3e267fb049ef459fПрофільВирішено23.06.2025, 23:57:01

Від: Анна Шевченко (ID: 68473b6fda23a0c7bc6b22b8)

На: Марія Сидоренко (ID: 68473920da23a0c7bc6b2192)

Текст скарги:

Агресивне спілкування

👁️🗑️

ID: 6859bf833e267fb049ef456bПрофільВідхилено23.06.2025, 23:56:35

Від: Анна Шевченко (ID: 68473b6fda23a0c7bc6b22b8)

На: Назарій Одаренко (ID: 684f2a6fdca43495ecc012048)

Текст скарги:

Фейковий обліковий запис

👁️🗑️

ID: 6859bf653e267fb049ef4529ПрофільВирішено23.06.2025, 23:56:05

Від: Андрій Мельник (ID: 68473666da23a0c7bc6b20d8)

3.3 Тестування роботи програмного продукту

В ході розробки програмного продукту виконувались тестування як окремих компонентів так і взаємодій між ними для унеможливлення великої кількості помилок, оскільки кожен продукт містить дефекти, які потребують усунення в наступних ітераціях.

		Бондаренко А.В.			ПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

їх перевірку реалізуємо за допомогою мануального тестування. Такий, на першу думку, базовий метод контролю якості насправді допомагає чітко відстежити поведінку інтерфейсу, валідацію полів і власне реакцію застосунку на різні випадки використання. Це дозволило переконатись у надійності реалізованої логіки валідації та обробки помилок. Усі дії здійснювались безпосередньо через інтерфейс користувача з фіксацією результатів кожної спроби входу чи реєстрації.

Ретельне проходження цих сценаріїв дозволить на ранньому етапі виявити потенційні помилки в роботі та гарантувати стабільну взаємодію користувача із застосунком.

Таблиця 3.1

Тестування функціоналу реєстрації

№	Опис	Очікуваний результат	Фактичний результат	Статус
1	Всі поля порожні	Виводяться повідомлення про необхідність заповнення всіх полів.	Всі обов'язкові повідомлення про помилки з'явилися.	Пройдено
2	Некоректний email	Повідомлення «Невірний формат ел.пошти».	Повідомлення з'явилося.	Пройдено
3	Пароль менше 8 символів	Повідомлення «Пароль має містити хоча б 8 символів».	Повідомлення з'явилося.	Пройдено
4	Ім'я або прізвище менше 2 символів	Виводяться повідомлення про обов'язкову довжину.	Вивід помилки для кожного поля.	Пройдено
5	Вік менше 18 або більше 100	Повідомлення «Вік обов'язковий».	Не приймається некоректний вік, вивід помилки.	Пройдено
6	Обрано статтю поза списком	Повідомлення «Стаття обов'язкова».	Блокується некоректне значення, вивід помилки.	Пройдено
7	Невірне посилання на аватар	Повідомлення «Некоректне посилання на аватар».	Виведено повідомлення, аватар не зберігається.	Пройдено

Продовження таблиці 3.1

№	Опис	Очікуваний результат	Фактичний результат	Статус
8	Поля місто та інтереси менше 2 символів	Вивід «Вкажіть місто/інтерес»	Вивід помилки	Пройдено
9	Всі поля заповнено правильно	Успішна реєстрація, додавання користувача до БД	Реєстрація проходить, додано до БД, отримано токен	Пройдено

Результати тестування процесу реєстрації мануальним підходом підтвердили правильну обробку всіх можливих випадків.

Таблиця 3.2

Тестування функціоналу авторизації

№	Опис	Очікуваний результат	Фактичний результат	Статус
1	Надсилання форми з порожніми полями	Форма не відправляється, вивід повідомлень про обов'язковість заповнення.	Форма не відправляється, виводяться повідомлення про обов'язковість.	Пройдено
2	Некоректний формат email	З'являється повідомлення «Невірний формат електронної пошти».	Повідомлення з'являється.	Пройдено
3	Введення некоректного логіну або пароля	Виведення помилки про помилку входу.	Помилка з'являється.	Пройдено
4	Введення правильного логіну та паролю	Авторизація проходить успішно.	Користувач авторизований та спрямований на профіль.	Пройдено

Результат мануального тестування функціоналу авторизації підтвердив, що додаток правильно реагує на всі передбачувані сценарії взаємодії. Перевірка полів працює відповідно до заданих умов, повідомлення про

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

помилки виводяться вірно, а за випадку введення правильних логіну та паролю відбувається успішна авторизація користувача з подальшим переходом на сторінку профілю.

Наступним етапом стало тестування серверної частини, а саме API. Для імітації реальної взаємодії з сервером було використано інструмент Postman, що дозволяє вручну надсилати запити та аналізувати відповіді [17]. Тестування проводилось за методологією чорної скриньки, де кожен ендпоінт перевірявся на відповідність очікуваній поведінці. Було створено колекцію тестових запитів, що охопили всю основну функціональність системи. Нижче буде наведено огляд головних запитів, що були протестовані.

Реєстрація користувача вимагає обов'язковості заповнення всіх полів згідно з валідаційними правилами. Кожне поле проходить на перевірку встановленим критеріям: електрона пошта має мати правильний формат, пароль має містити щонайменше 8 символів, ім'я та прізвище не можуть бути порожніми, вік – це числове значення від 18 до 100, стать має відповідати одному з допустимих варіантів, місто та інтереси повинні містити більше двох символів. Система валідації реалізована за допомогою бібліотеки `express-validator`, що автоматично перевіряє вхідні дані запитів.

Відтворимо процес реєстрації нового користувача (рис. 3.23).

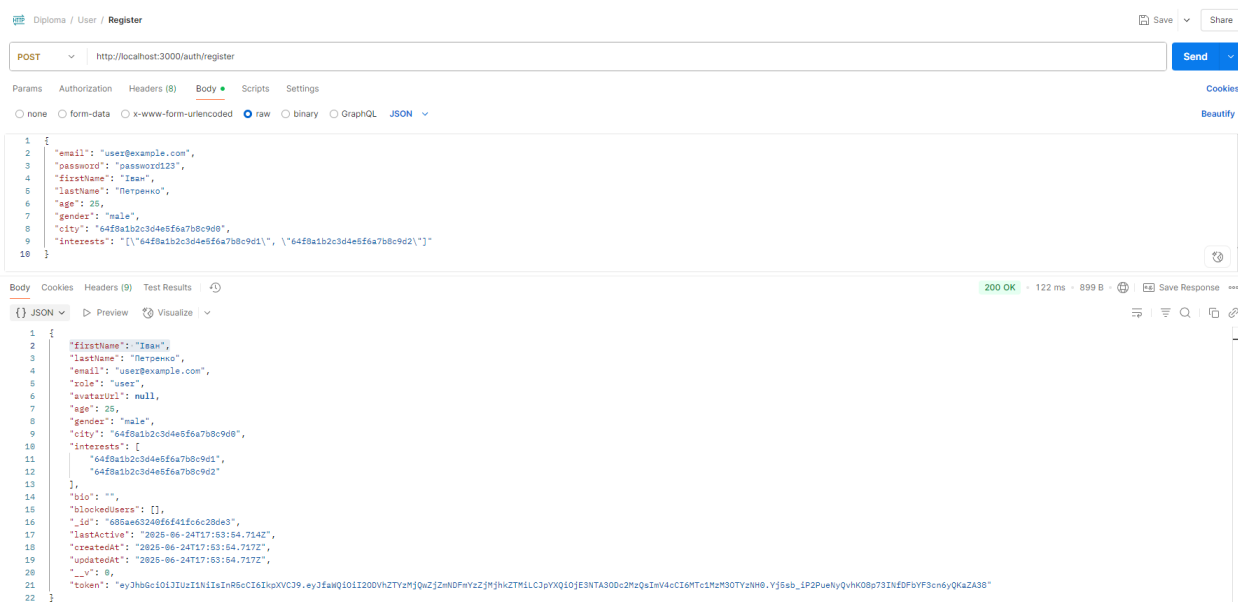


Рис. 3.23 Реєстрація користувача

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Годлевський Ю.О.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

У випадку некоректного заповнення хоча б одного з полів сервер повертає відповідь у форматі JSON з деталями помилки та відповідним кодом стану (рис. 3.24).

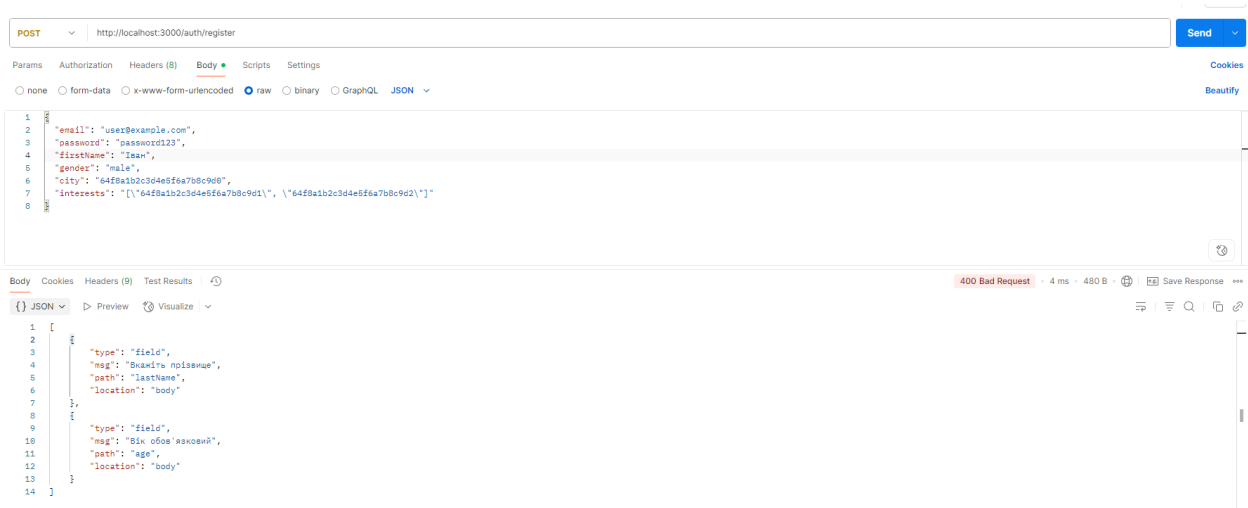


Рис. 3.24 Помилка валідації при реєстрації

Для авторизації в застосунку користувач має вести коректний логін та пароль. Якщо дані вірні, користувач авторизується та отримує доступ до повного функціоналу застосунку. У відповідь повертається об’єкт користувача з унікальним токеном авторизації, але без хешованого паролю (рис. 3.25).

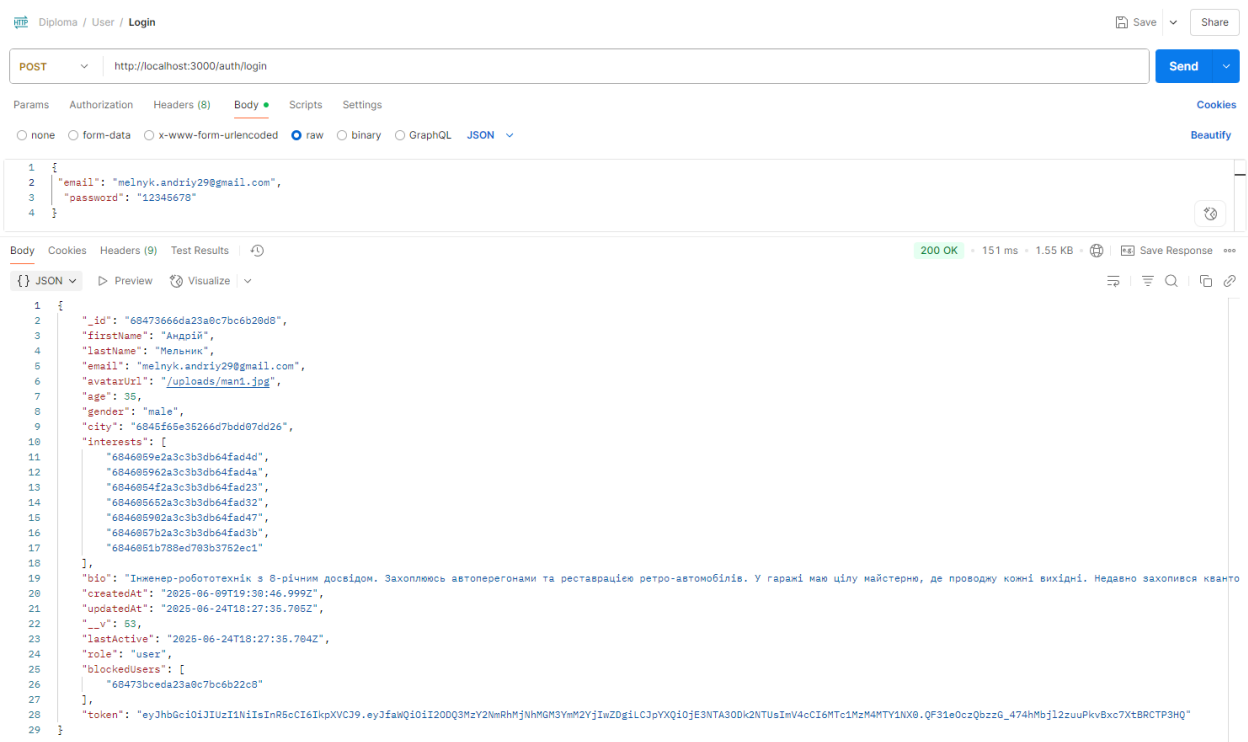


Рис. 3.25 Авторизація користувача

Після успішної авторизації користувач перенаправляється на сторінку власного профілю. Для відображення персональної інформації необхідно виконати запит до відповідного ендпоінту, що поверне дані про поточного користувача, із токеном авторизації (рис. 3.26).

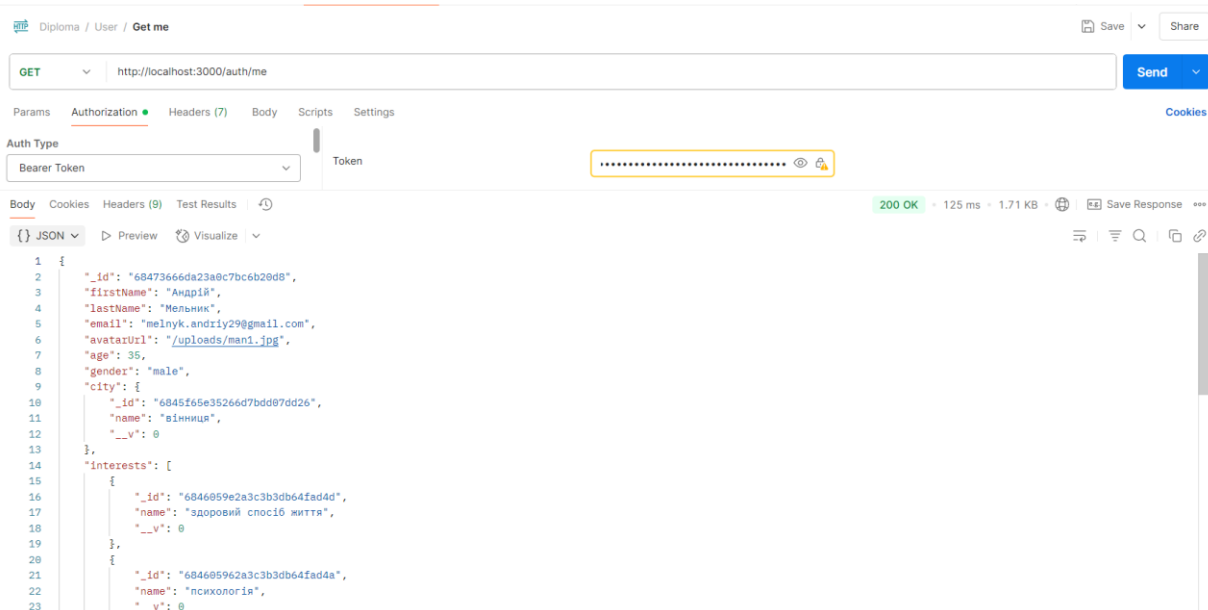


Рис. 3.26 Авторизація користувача

Для пошуку нових користувачів необхідно перейти на сторінку пошуку, де автоматично виконується запит до серверу на отримання всіх користувачів застосунку (рис. 3.27).

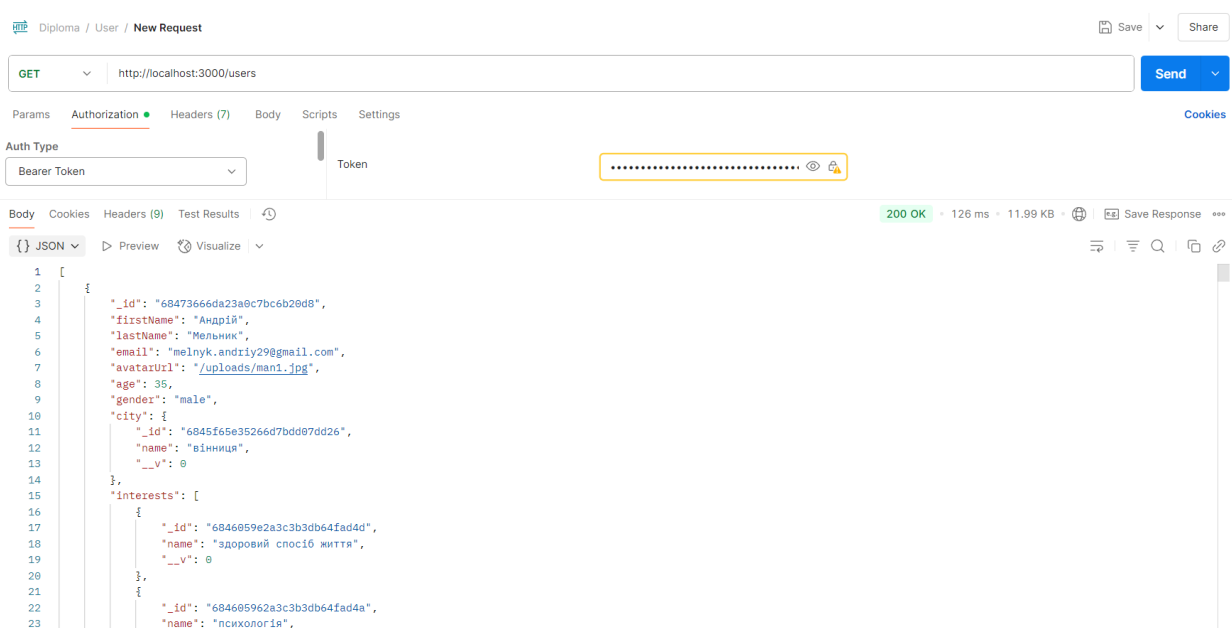


Рис. 3.27 Отримання всіх користувачів

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Також для перегляду профілю іншого користувача необхідно виконати запит до серверу за його унікальним ідентифікатором (рис. 3.28).

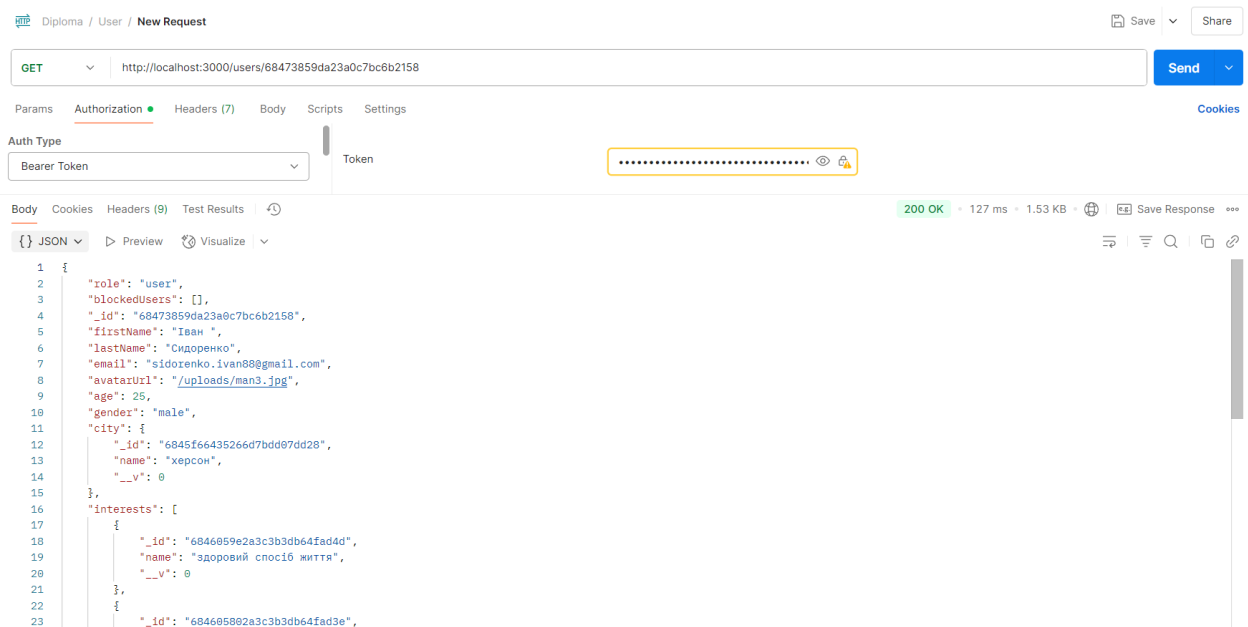


Рис. 3.28 Отримання одного користувача

При переході на сторінку дописів виконується запит на отримання всіх, що повертає масив об’єктів, кожен з яких містить інформацію про автора, вміст допису , дату створення та кількості переглядів і коментарів (рис. 3.29).

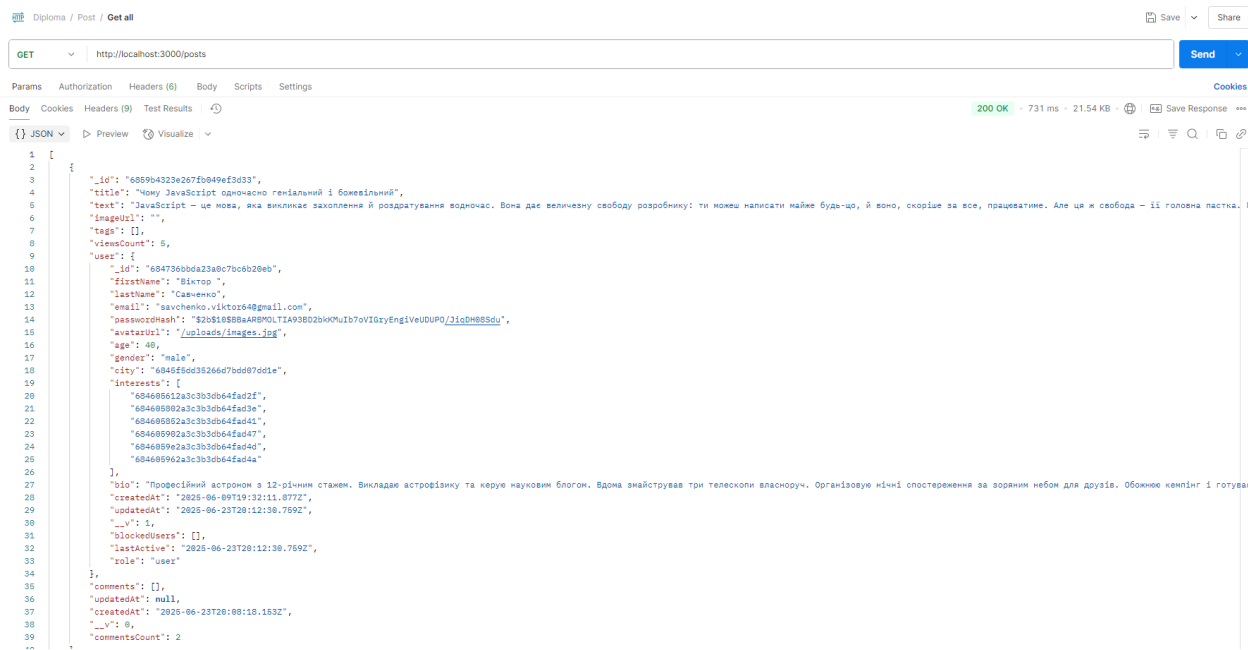


Рис. 3.29 Отримання всіх дописів

Повне відкриття допису за його унікальним ідентифікатором надає користувачеві можливість перегляду повного вмісту та долучити до обговорень, залишаючи коментарі (рис. 3.30).

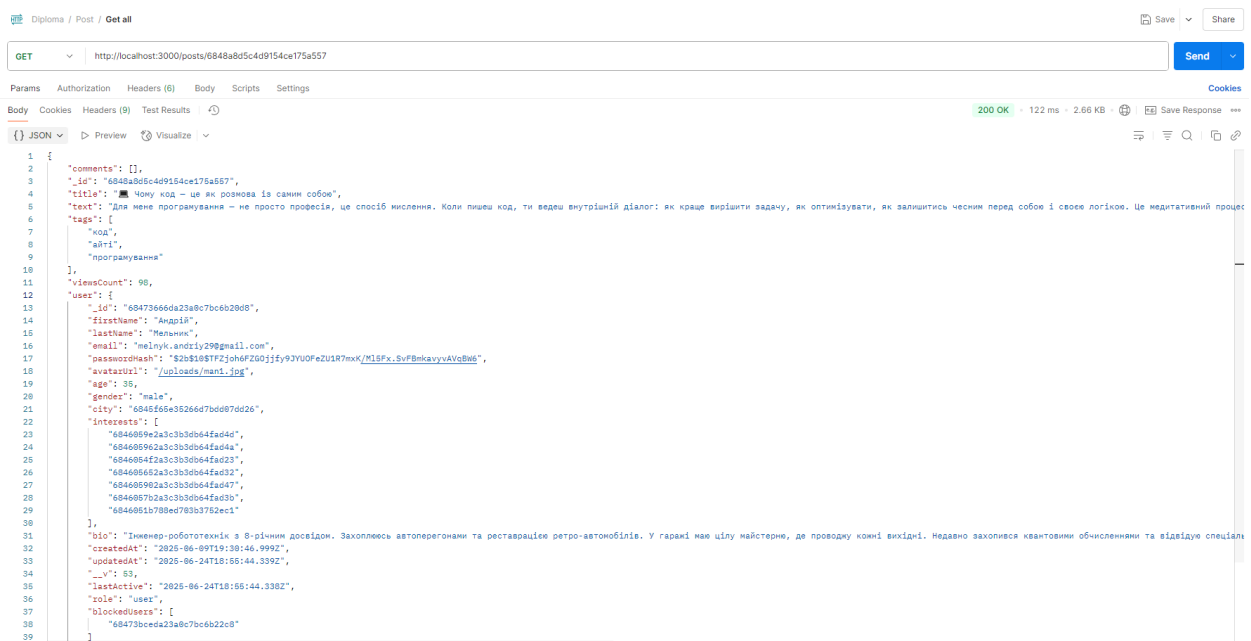


Рис. 3.30 Отримання всіх дописів

При відкритті повного допису користувач може залишити свій коментар, прив'язуючи його за унікальним ідентифікатором до конкретного допису (рис. 3.31). Для цього у інтерфейсі реалізовано окремий функціонал введення тексту, після чого дані надсилаються на сервер, де зберігаються у базі даних.

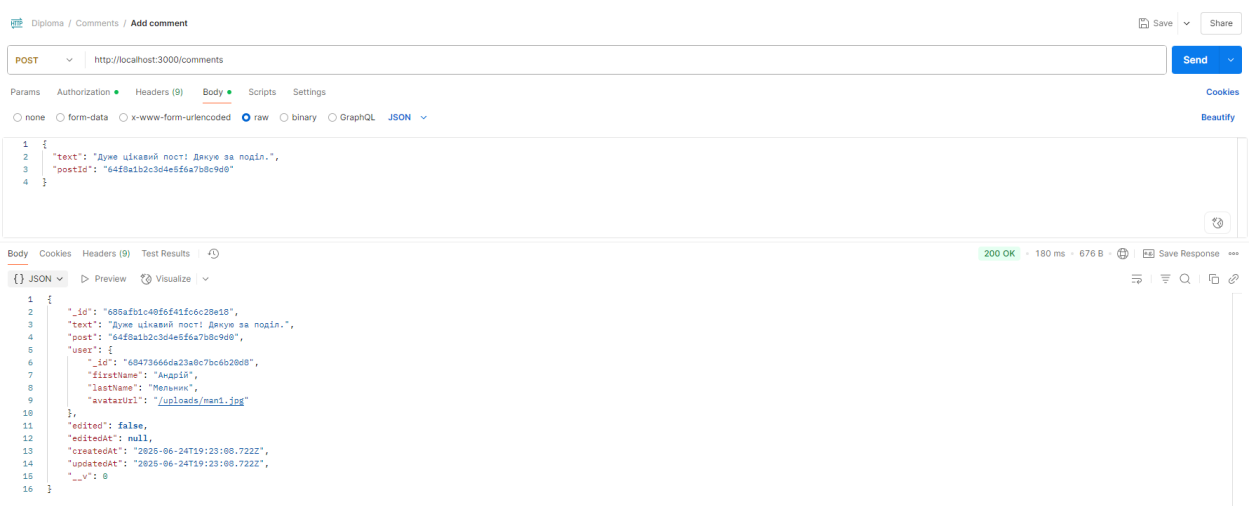


Рис. 3.31 Створення коментаря

Для отримання списку повідомлень між поточним та іншим користувачем передбачено ендпоінт, який повертає історію повідомлень між ними(рис. 3.32)

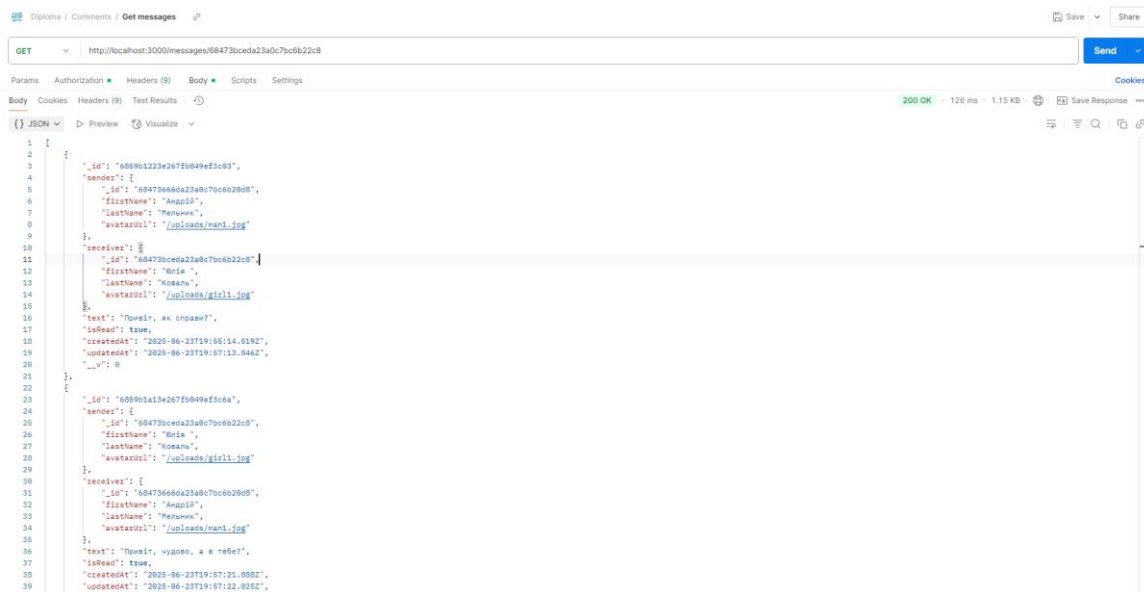


Рис. 3.32 Отримання повідомлень між користувачами

Висновки до третього розділу

В даному розділі було детально розглянуто процес налаштування та запуску всіх частин вебзастосунку для пошуку нових знайомств, включаючи серверну частину, клієнтський інтерфейс та БД. Було визначено порядок встановлення компонентів, зокрема особливості налаштування Node.js, фреймворку Express.js та бібліотеки Mongoose, що використовуємо для роботи з документно-орієнтованими даними.

Також описано структуру інтерфейсу користувача та порядок взаємодії з основними сторінками застосунку. Окремо розглянуто можливості адміністративної панелі для користувача із підвищеними правами доступу.

З метою перевірки коректної роботи основних функціональних можливостей застосунку було проведено тестування як і клієнтської, так і серверної частини. У процесі ручного тестування за допомогою Postman перевірено відповідність ендпоінтів очікуваним сценаріям поведінки.

Результати тестування підтвердили стабільну роботу модулів застосунку та правильність обробки вхідних даних.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано вебзастосунок для пошуку нових знайомств, а також проведено аналіз предметної області та порівняльне дослідження аналогічних рішень. У результаті цього було розроблено архітектуру системи та визначено основні вимоги до її побудови.

Проектування застосунку включало аналіз варіантів використання програмного продукту та проектування структури, включаючи моделювання взаємодії між складовими частинами. Було визначено ключові модулі застосунку, реалізовано структуру бази даних із урахуванням зв'язків між сутностями.

Описано алгоритми роботи з додатком та визначено послідовність дій для основних сценаріїв його використання. Розглянуто основний функціонал, який включає створення нового користувача, автентифікацію, завантаження зображень на сервер, перегляд усіх користувачів, а також пошук за заданими параметрами.

Також була надана детальна схема розгортання вебзастосунку, описано взаємодію структурних елементів на діаграмі компонентів та наведено опис інтерфейсу розробленого програмного продукту із демонстрацією основних сценаріїв взаємодії користувача з системою.

Результатом кваліфікаційної роботи є готовий до експлуатації вебзастосунок для пошуку нових знайомств, що відповідає сучасним вимогам користувачів і забезпечує зручний та інтуїтивно зрозумілий інтерфейс. Реалізовано основний функціонал, який включає реєстрацію, вхід у систему, редагування профілю, пошук користувачів за параметрами, обмін повідомленнями в реальному часі, створення дописів і коментарів. Проведено тестування функціональності як вручну, так і за допомогою інструментів, що дозволило перевірити коректність взаємодію між клієнтською та серверною частинами. Отримані результати підтвердили стабільність роботи системи в межах сценаріїв використання.

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Node.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://nodejs.org/docs/latest/api/>
2. Express.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://expressjs.com/en/5x/api.html>
3. React.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/learn>
4. MongoDB [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.mongodb.com/docs/guides/>
5. Mongoose [Електронний ресурс] – Режим доступу до ресурсу:
<https://mongoosejs.com/docs/>
6. Redux [Електронний ресурс] – Режим доступу до ресурсу:
<https://redux.js.org/>
7. Redux Toolkit [Електронний ресурс] – Режим доступу до ресурсу:
<https://redux-toolkit.js.org/>
8. Axios [Електронний ресурс] – Режим доступу до ресурсу:
<https://axios-http.com/docs/intro>
9. Socket.IO [Електронний ресурс] – Режим доступу до ресурсу:
<https://socket.io/docs/v4/server-api/>
10. SCSS/SASS [Електронний ресурс] – Режим доступу до ресурсу:
<https://sass-lang.com/documentation/>
11. Multer [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.npmjs.com/package/multer>
12. OkCupid [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.okcupid.com/>
13. Bumble [Електронний ресурс] – Режим доступу до ресурсу:
<https://bumble.com/en-us/>
14. Boo [Електронний ресурс] – Режим доступу до ресурсу:
<https://boo.world/uk>

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

15. JWT [Электронный ресурс] – Режим доступа до ресурсу: <https://jwt.io/introduction>
16. REST API Design Guidelines [Электронный ресурс] – Режим доступа до ресурсу: <https://restfulapi.net/>
17. Postman [Электронный ресурс] – Режим доступа до ресурсу: <https://learning.postman.com/docs/introduction/overview/>
18. CORS у Express.js [Электронный ресурс] – Режим доступа до ресурсу: <https://expressjs.com/en/resources/middleware/cors.html>
19. MongoDB Atlas [Электронный ресурс] – Режим доступа до ресурсу: <https://www.mongodb.com/products/platform/atlas-database>
20. React Router [Электронный ресурс] – Режим доступа до ресурсу: <https://reactrouter.com/>
21. Express-validator [Электронный ресурс] – Режим доступа до ресурсу: <https://express-validator.github.io/docs/>
22. JavaScript [Электронный ресурс] – Режим доступа до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
23. NPM [Электронный ресурс] – Режим доступа до ресурсу: <https://docs.npmjs.com/>
24. HTTP/HTTPS [Электронный ресурс] – Режим доступа до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTTP>
25. JSON [Электронный ресурс] – Режим доступа до ресурсу: <https://www.json.org/json-en.html>
26. Bcrypt.js [Электронный ресурс] – Режим доступа до ресурсу: <https://www.npmjs.com/package/bcryptjs>
27. WebSocket API (WebSockets) [Электронный ресурс] – Режим доступа до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
28. Socket.IO Rooms [Электронный ресурс] – Режим доступа до ресурсу: <https://socket.io/docs/v4/rooms/>

29. React Router DOM [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/6.28.0/start/overview>

30. Socket.IO Client [Електронний ресурс] – Режим доступу до ресурсу: <https://socket.io/docs/v4/client-api/>

		Бондаренко А.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

ДОДАТКИ

Лістинг файлу «PostController.js»

```
import PostModel from '../models/Post.js';
import CommentModel from '../models/Comment.js';

export const create = async (req, res) => {
  try {
    const tags = req.body.tags
      ? req.body.tags
        .split(',')
        .map((tag) => tag.trim().toLowerCase())
        .filter(Boolean)
      : [];
    const doc = new PostModel({
      title: req.body.title,
      text: req.body.text,
      imageUrl: req.body.imageUrl,
      tags,
      user: req.userId,
    });

    const post = await doc.save();

    res.json(post);
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: 'Не можливо створити статтю',
    });
  }
};

export const getAll = async (req, res) => {
  try {
    let sort = {};
    if (req.query.sort === 'views') {
      sort = { viewsCount: -1 };
    } else if (req.query.sort === 'views-asc') {
      sort = { viewsCount: 1 };
    } else if (req.query.sort === 'comments') {
      sort = { commentsCount: -1 };
    } else if (req.query.sort === 'comments-asc') {
      sort = { commentsCount: 1 };
    } else if (req.query.sort === 'old') {
      sort = { createdAt: 1 };
    } else {
      sort = { createdAt: -1 };
    }
    const posts = await PostModel.find().populate('user').sort(sort).exec();
  }
```

```

const validPosts = posts.filter((post) => post.user);

const postsWithComments = await Promise.all(
  validPosts.map(async (post) => {
    const commentsCount = await CommentModel.countDocuments({ post: post._id
  });
    return { ...post._doc, commentsCount };
  })
);

if (req.query.sort === 'comments' || req.query.sort === 'comments-asc') {
  postsWithComments.sort((a, b) => {
    if (req.query.sort === 'comments') {
      return b.commentsCount - a.commentsCount;
    } else {
      return a.commentsCount - b.commentsCount;
    }
  });
}

res.json(postsWithComments);
} catch (err) {
  console.log(err);
  res.status(500).json({
    message: 'Не вдалось отримати всі статті',
  });
}
};

export const getOne = async (req, res) => {
  try {
    const post = await PostModel.findOneAndUpdate(
      { _id: req.params.id },
      { $inc: { viewsCount: 1 } },
      {
        new: true,
        populate: {
          path: 'user',
          select: 'firstName lastName avatarUrl',
        },
      },
    ).exec();

    if (!post) {
      return res.status(404).json({ message: 'Пост не знайдено' });
    }
    if (!post.user) {
      return res.status(404).json({ message: 'Пост не знайдено' });
    }

    const commentsCount = await CommentModel.countDocuments({ post: post._id });

```

```

    res.json({ ...post._doc, commentsCount });
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: 'Не вдалося отримати пост',
    });
  }
};

export const getById = async (req, res) => {
  try {
    const post = await PostModel.findById(req.params.id).populate('user').exec();

    if (!post) {
      return res.status(404).json({ message: 'Пост не знайдено' });
    }

    if (!post.user) {
      return res.status(404).json({ message: 'Пост не знайдено' });
    }

    const commentsCount = await CommentModel.countDocuments({ post: post._id });
    res.json({ ...post._doc, commentsCount });
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: 'Не вдалося отримати пост',
    });
  }
};

export const remove = async (req, res) => {
  try {
    const postId = req.params.id;
    const doc = await PostModel.findOneAndDelete({ _id: postId });
    if (!doc) {
      return res.status(404).json({
        message: 'Стаття не знайдена',
      });
    }
    res.json({
      success: true,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: 'Не вдалось видалити статтю',
    });
  }
};

```

```

export const update = async (req, res) => {
  try {
    const postId = req.params.id;
    const tags = req.body.tags
      ? req.body.tags
        .split(',')
        .map((tag) => tag.trim().toLowerCase())
        .filter(Boolean)
      : [];
    const updatedPost = await PostModel.findOneAndUpdate(
      {
        _id: postId,
        user: req.userId,
      },
      {
        title: req.body.title,
        text: req.body.text,
        imageUrl: req.body.imageUrl,
        tags,
        updatedAt: new Date(),
      },
      { new: true }
    );
    if (!updatedPost) {
      return res.status(403).json({
        success: false,
        message: 'Немає доступу',
      });
    }
    res.json({
      success: true,
      post: updatedPost,
    });
  } catch (err) {
    console.error(err);
    res.status(500).json({
      success: false,
      message: 'Помилка оновлення',
    });
  }
};

```

Лістинг файлу «AdminController.js»

```
import UserModel from '../models/User.js';
import PostModel from '../models/Post.js';
import MessageModel from '../models/Message.js';
import CommentModel from '../models/Comment.js';

export const getStats = async (req, res) => {
  try {
    const totalUsers = await UserModel.countDocuments();
    const activeUsers = await UserModel.countDocuments({
      lastActive: { $gte: new Date(Date.now() - 24 * 60 * 60 * 1000) },
    });

    const postsWithUsers = await PostModel.find().populate('user');
    const validPosts = postsWithUsers.filter((post) => post.user);
    const totalPosts = validPosts.length;

    const messagesWithUsers = await
    MessageModel.find().populate('sender').populate('receiver');
    const validMessages = messagesWithUsers.filter((message) => message.sender &&
    message.receiver);
    const totalMessages = validMessages.length;

    const commentsWithUsers = await CommentModel.find().populate('user');
    const validComments = commentsWithUsers.filter((comment) => comment.user);
    const totalComments = validComments.length;

    const thirtyDaysAgo = new Date();
    thirtyDaysAgo.setDate(thirtyDaysAgo.getDate() - 30);

    const chartStats = await UserModel.aggregate([
      {
        $match: {
          createdAt: { $gte: thirtyDaysAgo },
        },
      },
      {
        $group: {
          _id: { $dateToString: { format: '%Y-%m-%d', date: '$createdAt' } },
          count: { $sum: 1 },
        },
      },
      {
        $sort: { _id: 1 },
      },
    ]);

    const registrationData = [];
    const currentDate = new Date(thirtyDaysAgo);
```

```

while (currentDate <= new Date()) {
  const dateStr = currentDate.toISOString().split('T')[0];
  const dayStats = chartStats.find((s) => s._id === dateStr);

  registrationData.push({
    date: dateStr,
    count: dayStats ? dayStats.count : 0,
  });

  currentDate.setDate(currentDate.getDate() + 1);
}

res.json({
  generalStats: {
    totalUsers,
    activeUsers,
    totalPosts,
    totalMessages,
    totalComments,
  },
  chartStats: registrationData,
});
} catch (error) {
  console.error('Помилка отримання статистики:', error);
  res.status(500).json({
    message: 'Не вдалось отримати статистику',
  });
}
};

```

Лістинг файлу «createAdmin.js»

```
import mongoose from 'mongoose';
import 'dotenv/config';
import bcrypt from 'bcrypt';
import UserModel from '../models/User.js';
import { DB_PASSWORD } from '../config.js';

const createAdmin = async () => {
  try {
    await mongoose.connect(
      `mongodb+srv://artur:${DB_PASSWORD}@diploma.n7w7vau.mongodb.net/diploma?ret
ryWrites=true&w=majority&appName=Diploma`
    );

    const adminData = {
      firstName: 'Admin',
      lastName: 'User',
      email: 'admin@example.com',
      passwordHash: await bcrypt.hash('admin123', 10),
      role: 'admin',
      age: 30,
      gender: 'male',
      city: '6845f3b7505d7c41fc484ba7',
      interests: [],
      bio: 'Системний адміністратор',
    };

    const adminExists = await UserModel.findOne({ email: adminData.email });

    if (adminExists) {
      console.log('Адміністратор вже існує');
      process.exit(0);
    }

    const admin = await UserModel.create(adminData);
    console.log('Адміністратор успішно створений:', admin);

    process.exit(0);
  } catch (error) {
    console.error('Помилка створення адміністратора:', error);
    process.exit(1);
  }
};

createAdmin();
```

Лістинг файлу «checkAdmin.js»

```
import UserModel from '../models/User.js';

export default async (req, res, next) => {
  try {
    const user = await UserModel.findById(req.userId);

    if (!user) {
      return res.status(404).json({
        message: 'Користувача не знайдено',
      });
    }

    if (user.role !== 'admin') {
      return res.status(403).json({
        message: 'Немає прав адміністратора',
      });
    }

    next();
  } catch (err) {
    console.error('Помилка перевірки прав адміністратора:', err);
    res.status(500).json({
      message: 'Помилка перевірки прав доступу',
    });
  }
};
```

Лістинг файлу «ProfileCard.jsx»

```

import React, { useState } from 'react';
import styles from './ProfileCard.module.scss';
import Avatar from '../Avatar/Avatar';
import capitalizeFirstLetter from '../utils/capitalizeFirstLetter';
import { Link } from 'react-router-dom';
import { FiEdit } from 'react-icons/fi';
import { FaFlag } from 'react-icons/fa';
import ReportModal from '../ReportModal/ReportModal';

const ProfileCard = ({ user, profileId }) => {
  const [isReportModalOpen, setIsReportModalOpen] = useState(false);

  if (!user) {
    return <div className={styles.loading}>Завантаження...</div>;
  }

  const formatDate = (dateString) => {
    if (!dateString) return 'Не вказано';
    const date = new Date(dateString);
    return date.toLocaleString('uk-UA', {
      day: '2-digit',
      month: '2-digit',
      year: 'numeric',
      hour: '2-digit',
      minute: '2-digit',
    });
  };

  const currentUserId = localStorage.getItem('userId');
  const isOwnProfile = String(profileId) === String(currentUserId);

  const handleReport = () => {
    setIsReportModalOpen(true);
  };

  return (
    <div className={styles.profileCard}>
      <div className={styles.profileContent}>
        <div className={styles.left}>
          <div className={styles.avatarContainer}>
            <Avatar user={user} size="large" />
          </div>
          {!isOwnProfile && (
            <div className={styles.actionButtons}>
              <Link to={`/chat/${user._id}`} className={styles.writeButton}>
                Написати
              </Link>
            </div>
          )}
        </div>
      </div>
    </div>
  );
};

```

```

        <button
            onClick={handleReport}
            className={styles.reportButton}
            title="Скарга на користувача"
        >
            {' '}
            Поскаржитись
            <FaFlag />
        </button>
    </div>
)}
{isOwnProfile && (
    <Link
        to={` /profile/edit/${localStorage.getItem('userId')}`}
        className={styles.editProfileButton}
        title="Редагувати профіль"
    >
        <FiEdit size={24} />
    </Link>
)}
</div>
<div className={styles.userInfo}>
    <div className={styles.infoItem}>
        <span className={styles.label}>Ім'я:</span>
        <span className={styles.value}>{user?.firstName || 'Не
вказано'}</span>
    </div>
    <div className={styles.infoItem}>
        <span className={styles.label}>Прізвище:</span>
        <span className={styles.value}>{user?.lastName || 'Не
вказано'}</span>
    </div>
    <div className={styles.infoItem}>
        <span className={styles.label}>Email:</span>
        <span className={styles.value}>{user?.email || 'Не вказано'}</span>
    </div>
    <div className={styles.infoItem}>
        <span className={styles.label}>Вік:</span>
        <span className={styles.value}>{user?.age || 'Не вказано'}</span>
    </div>
    <div className={styles.infoItem}>
        <span className={styles.label}>Стать:</span>
        <span className={styles.value}>
            {user?.gender === 'male'
                ? 'Чоловік'
                : user?.gender === 'female'
                ? 'Жінка'
                : 'Не вказано'}
        </span>
    </div>
</div>
<div className={styles.infoItem}>

```

```

        <span className={styles.label}>Місто:</span>
        <span className={styles.value}>
            {user?.city?.name ? capitalizeFirstLetter(user.city.name) : 'Не
вказано'}
        </span>
    </div>
    <div className={styles.infoItem}>
        <span className={styles.label}>Інтереси:</span>
        <div className={styles.interestsList}>
            {user?.interests && user.interests.length > 0 ? (
                user.interests.map((interest) => (
                    <span key={interest._id} className={styles.interestTag}>
                        {capitalizeFirstLetter(interest.name)}
                    </span>
                ))
            ) : (
                <span className={styles.value}>Не вказано</span>
            )}
        </div>
    </div>
    <div>
        {user?.bio && (
            <div className={styles.infoItem}>
                <span className={styles.label}>Про мене:</span>
                <span className={styles.value}>{user.bio}</span>
            </div>
        )}
        <div className={styles.infoItem}>
            <span className={styles.label}>Дата реєстрації:</span>
            <span className={styles.value}>{formatDate(user?.createdAt)}</span>
        </div>
        {user?.updatedAt && user?.createdAt && user.updatedAt !==
user.createdAt && (
            <div className={styles.infoItem}>
                <span className={styles.label}>Останнє оновлення:</span>
                <span className={styles.value}>{formatDate(user.updatedAt)}</span>
            </div>
        )}
    </div>
</div>
<ReportModal
    isOpen={isReportModalOpen}
    onClose={() => setIsReportModalOpen(false)}
    type="profile"
    reportedItemId={user._id}
    reportedItemTitle={` ${user.firstName} ${user.lastName}`}
/>
</div>
);
};

export default ProfileCard;

```