Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему: СТВОРЕННЯ ПРОГРАМНОЇ ВЕБСИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ З ВІДОБРАЖЕННЯМ АКТИВНОСТІ ТА ГЕОГРАФІЧНИХ КООРДИНАТ КОРИСТУВАЧІВ У РЕАЛЬНОМУ ЧАСІ.
DESIGN AND DEVELOPMENT OF A CLIENT-SERVER WEB MESSAGING APPLICATION WITH USER GEOLOCATION AND REAL-TIME ACTIVITY DISPLAY

Виконав: студент 4 курсу, групи КНТ-212
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Комп'ютерні науки

ПОЙМАН Є.І.
(ПРІЗВИЩЕ та ініціали)
Керівник СТЕПАНЕНКО О.О.
(ПРІЗВИЩЕ та ініціали)
Рецензент ДЬЯЧУК Т.С.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТА

Поймана Євгенія Ігоровича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення програмної вебсистеми обміну повідомленнями з відображенням активності та географічних координат користувачів у реальному часі. Design and Development of a Client-server Web Messaging Application with User Geolocation and Real-time Activity Display
Керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від " 07 " квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 1 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю, ст. викладач		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Євгеній ПОЙМАН
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 81 с., 2 табл., 27 рис., 3 дод., 20 джерел.

ВЕБЗАСТОСУНОК, ЧАТ У РЕАЛЬНОМУ ЧАСІ, WEBSOCKET, SPRING BOOT, STOMP, GEOLOCATION, POSTGRESQL, LEAFLET, DOCKER, RENDER.

Об’єкт дослідження – методи та алгоритми створення програмних вебсистем.

Предмет дослідження – методи та інструменти розроблення для реалізації багатокористувацького вебчату з підтримкою геолокації та відображення активності користувачів.

Мета роботи - розробка програмної вебсистеми обміну повідомленнями у реальному часі з відображенням активності та географічних координат користувачів для підвищення ефективності онлайн-комунікації, оптимізації взаємодії між користувачами, зменшення часу передачі повідомлень та автоматизації процесу визначення географічного розташування користувачів.

Матеріали, методи та технічні засоби: мова програмування Java, фреймворк Spring Boot, технологія WebSocket, протокол STOMP, HTML, CSS, JavaScript, база даних PostgreSQL, система керування залежностями Maven, технологія Docker, платформа Render, бібліотека Leaflet та OpenStreetMap.

Результати, було створено вебзастосунок «Ukraine Regional Chat», який забезпечує обмін повідомленнями між користувачами у реальному часі. Реалізовано систему реєстрації та авторизації користувачів, загальний та регіональний чати, приватні повідомлення, визначення географічного розташ

ування користувач, відображення активності на карті України, збереження історії повідомлень у базі даних.

Висновки. Розроблена система може використовуватися як платформа для онлайн-спілкування користувачів за регіональним принципом, а також як основа для подальшого розвитку сучасних вебсистем комунікації.

Галузь використання - вебсистеми онлайн-комунікації, соціальні платформи, корпоративні системи обміну повідомленнями.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 81 pages, 2 tables, 27 figures, 3 appendixes, 20 sources.

WEB APPLICATION, REAL-TIME CHAT, WEBSOCKET, SPRING BOOT, STOMP, GEOLOCATION, POSTGRESQL, LEAFLET, DOCKER, RENDER.

The object of research is methods and algorithms for creating software web systems.

The subject of research is the development methods and tools for implementing multi-user web chat with geolocation support and displaying user activity.

The purpose of the work is to develop a real-time software web messaging system with displaying user activity and geographic coordinates to increase the efficiency of online communication, optimize interaction between users, reduce message transmission time, and automate the process of determining the geographic location of users.

Materials, methods and technical means: Java programming language, Spring Boot framework, WebSocket technology, STOMP protocol, HTML, CSS, JavaScript, PostgreSQL database, Maven dependency management system, Docker technology, Render platform, Leaflet library and OpenStreetMap.

Results, a web application “Ukraine Regional Chat” was created, which provides real-time messaging between users. A user registration and authorization system, general and regional chats, private messages, determining the geographical location of users, displaying activity on the map of Ukraine, saving message history in the database were implemented.

Conclusions. The developed system can be used as a platform for online communication of users on a regional basis, as well as a basis for the further development of modern web communication systems.

Field of use - online communication web systems, social platforms, corporate messaging systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

CSS – Cascading Style Sheets;

HTML – HyperText Markup Language,

HTTP – HyperText Transfer Protocol;

IDE – Integrated Development Environment;

IP – Internet Protocol;

JSON – JavaScript Object Notation;

ORM – Object-Relational Mapping;

UI – User Interface;

URL – Uniform Resource Locator;

БД – бази даних;

КІ – користувацький інтерфейс.

МЗ – мережеве з’єднання;

ПЗ – програмне забезпечення.

ВСТУП

Сьогодні вебтехнології активно використовуються в усіх сферах життя. Особливо швидко розвиваються сервіси онлайн спілкування, оскільки користувачам важливо отримувати корисну для них інформацію, спілкуватись з близькими без затримок. Люди використовують месенджери для навчання, роботи дехто навіть знаходить свою любов через них. Звісно з кожним роком все більше охочих їх використовують, тем більше зараз коли майже все намагається перейти у дистанційний формат.

Як на мене існує багато невеликих вебчатів, які вже застарілі. Часто це просто сторінка з повідомленнями без всякої взаємодії між користувачами, зараз цього вже буде недостатньо. Ми вже всі звикли бачити онлайн статуси, приватні чати, миттєві повідомленнями, оскільки зараз це все є дійсно невід'ємною частиною життя.

Саме тому такі технології використовуються практично всюди . Застосунки такі як Discord, Telegram, Microsoft Teams працюють за принципом постійного обміну даними між клієнтом і сервером. Повідомлення з'являються без перезавантаження сторінки. Для цього найчастіше використовують WebSocket. Цю технологію ретельно описували Ian Hickson та Michael Carter у специфікації RC6445. Її основна перевага полягає в тому, що вона забезпечує постійне з'єднання між клієнтом та сервером через один порт. Завдяки цьому дані надсилаються миттєво, не чекаючи ніякого часу, тобто на теперешній час він є майже ідеальним варіантів для багатьох додатків.

Окремо варто згадати геолокацію. Багато сучасних платформ уже використовують визначення місцезнаходження користувача. В межах моєї роботи, ця ідея була використана для створення регіонального вебчату по областях України. Мені здалося цікавим не просто зробити обмін повідомленнями, а показати активність користувачів на інтерактивній карті. Через це застосунок виглядає більш живим, а головне безкоштовним.

У роботі було створено вебзастосунок «Ukraine Regional Chat». Система підтримує загальний чат, чат області та приватні повідомлення між користувачами. Також реалізовано авторизацію, систему статусів, збереження історії повідомлень та карту з онлайн-користувачами. Для серверної частини використовувався Spring Boot. Для карти було обрано Leaflet та OpenStreetMap, бо ці рішення є безкоштовними й достатньо зручними для інтеграції.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні системи онлайн спілкування

Інтернет-комунікація за останні роки змінилася дуже сильно. Якщо раніше основним способом взаємодії були форуми або електронна пошта, то зараз більшість користувачів спілкується через системи миттєвого обміну повідомленнями. Люди звикли отримувати інформацію практично одразу. Через це швидкість роботи вебсервісів стала однією з важливіших характеристик сучасних застосунків [1].

Особливо активно розвиток онлайн-комунікації почався після появи смартфонів та мобільного інтернету. Користувач більше не прив’язаний до комп’ютера або конкретного місця. Людина може спілкуватися будь-де. Це вплинуло і на саму структуру сучасних вебсистем. Інтерфейси стали простішими, повідомлення почали оновлюватися автоматично, а більшість сервісів перейшла на принцип роботи в реальному часі [1].

На мою думку, саме зручність і швидкість стали основними причинами популярності сучасних месенджерів. Користувачі майже не помічають, як звикають до миттєвого отримання повідомлень. Якщо сервіс працює повільно або вимагає постійного оновлення сторінки, це відразу створює негативне враження та неохоту користуватись ним [1].

Сьогодні системи онлайн-спілкування використовуються не лише для особистого листування. Вони стали частиною навчання, роботи та організації команд. Наприклад, Microsoft Teams активно використовується в університетах і компаніях для проведення конференцій та обміну файлами, Slack став популярним серед IT-команд, а такий застосунок як Discord використовується не тільки геймерами, але й навчальними спільнотами [1].

В Україні також значно виріс рівень використання месенджерів. Telegram став одним із головних джерел отримання інформації, особливо для локальних спільнот та регіональних каналів. В межах війни люди активно створюють групи для свого міста, університету або навіть окремого району.

Так вони намагаються передати свої почуття, поділитися враженнями, або навіть допомогти комусь у вирішенні складних ситуацій у непростий для країни час. Все це показує, що локальна взаємодія між користувачами стає все більш актуальною та міцнішою [1]-[3].

Під час аналізу предметної області стало помітно, що більшість сучасних систем орієнтується насамперед на глобальне спілкування. Проте локальні сервіси мають свої переваги. Наприклад, користувачам часто цікавіше взаємодіяти з людьми зі свого регіону, аніж із випадковими учасниками інших країн [1]-[3].

Саме ця особливість робить регіональні вебчати перспективним напрямом. Людина бачить знайомий контекст, локальні новини та активність користувачів зі свого регіону.

Окремо варто звернути увагу на зміни в технічній реалізації сучасних чатів. Раніше вебзастосунки працювали переважно через HTTP-запити. Браузер періодично звертався до сервера та перевіряв наявність нових повідомлень. Такий підхід створював затримки, та зайве навантаження на сервер [2].

Після розвитку WebSocket ситуація значно змінилася. Повідомлення почали передаватися через постійне двостороннє з’єднання. Це дозволило реалізувати практично миттєву взаємодію між користувачами [7].

У роботі Ian Hickson та Michael Carter, присвяченій WebSocket Protocol RFC 6455, зазначається, що дана технологія створювалася саме для систем реального часу та інтерактивних вебзастосунків. У сучасних умовах WebSocket використовується майже у всіх популярних сервісах обміну повідомленнями [7].

Ще однією важливою деталю є інтеграція геолокаційних сервісів. Наприклад, Uklon, Google Maps або навіть Telegram використовують визначення координат користувача для персоналізації функцій системи. Геолокація дозволяє створювати локальні групи, рекомендації або інтерактивні карти активності [10]-[11].

Під час аналізу аналогів стало зрозуміло, що у більшості навчальних вебчатів геолокація або повністю відсутня, або реалізована дуже спрощена. Часто система може лише показати країну користувача без будь-якої подальшої взаємодії [1]-[3].

Важливу роль також відіграє адаптивність інтерфейсу. Сучасні вебзастосунки повинні нормально працювати як на комп'ютерах, так і на мобільних пристроях. За даними Statista, понад половина користувачів використовує месенджери саме через смартфони. Через це під час створення системи необхідно враховувати особливості мобільних браузерів та різних розмірів екранів [2].

Як для мене, багато студентських або навчальних проєктів мають одну спільну проблему, вони добре демонструють роботу окремої технології, але не виглядають як повноцінний сервіс.

Саме тому я вирішив створити вебзастосунок, який поєднує обмін повідомленнями у режимі реального часу та систему регіональної взаємодії користувачів по областях України.

1.2 Аналіз інформаційних системи для спілкування

Для своєї роботи я проаналізував інформаційні системи та засоби для швидкого спілкування.

Програмне забезпечення інформаційної системи Mattermost (рис.1.1) активно використовується для організації командного онлайн-спілкування та координації роботи між користувачами. Вона підтримує роботу в режимі реального часу та дозволяє створювати окремі канали для різних груп або проєктів. Mattermost використовується в IT-компаніях, навчальних спільнотах та організаціях, які потребують контрольованого середовища для обміну повідомленнями [2].

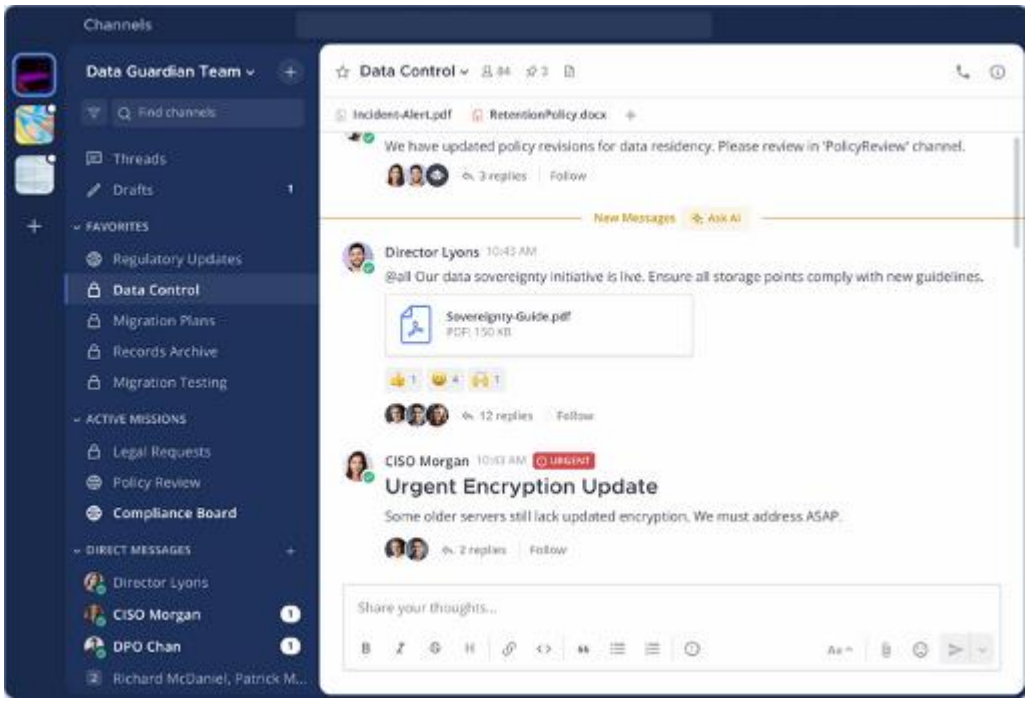

Рисунок 1.1 – Програмне забезпечення інформаційної системи Mattermost

Особливістю інформаційної системи Mattermost є можівість самостійного розгортання на власному сервері. Це дозволяє організаціям контролювати збереження даних та налаштовувати систему відповідно до потреб. Окрім цього платформа підтримує інтеграцію з іншими сервісами для командної роботи та забезпечує швидкий обмін інформацією міжкористувачами [2]-[3].

Програмне забезпечення інформаційної системи Mattermost орієнтоване насамперед на командну взаємодію та структурування спілкування.Система дозволяє розділяти повідомлення між різними каналами, що допомагає зменшити перевантаження загального чату та зробити роботу користувачів більш організованою. Завдяки підтримці постійного обміну повідомленнями Mattermost створений для тих, хто прагне підтримувати ефективний та миттєвий зв’язок без втрати важливої інформації.

Основними особливостями та характеристиками програмного забезпечення інформаційної системи Mattermost є такі функції:

- робота в режимі реального часу: програмне забезпечення інформаційної системи Mattermost підтримує миттєвий обмін повідомленнями між користувачами без необхідності оновлення сторінки;

- підтримка каналів: програмне забезпечення інформаційної системи Mattermost дозволяє створювати окремі канали для різних команд або проєктів;

- система ролей та прав доступу: програмне забезпечення інформаційної системи Mattermost підтримує розмежування прав користувачів та адміністраторів;

- інтеграція з іншими сервісами: програмне забезпечення інформаційної системи Mattermost підтримує інтеграцію з GitHub, Jira, GitLab та іншими платформами для командної роботи;

- передача файлів: програмне забезпечення інформаційної системи Mattermost дозволяє надсилати документи, зображення та інші файли безпосередньо в чаті;

- підтримка групового спілкування. Система дозволяє організовувати командні чати та приватні обговорення між користувачами;

- пошук по історії повідомлень: програмне забезпечення інформаційної системи Mattermost підтримує швидкий пошук інформації в архіві повідомлень;

- підтримка мобільних пристроїв: система доступна на комп’ютерах, планшетах та мобільних телефонах;

- підтримка власного серверного розгортання: програмне забезпечення інформаційної системи Mattermost дозволяє запускати систему у власному мережевому середовищі організації;

- система сповіщень: програмне забезпечення інформаційної системи Mattermost підтримує налаштування повідомлень та сповіщень для користувачів.

Серед недоліків Mattermost можна виділити складніше початкове налаштування у порівнянні зі звичайними месенджерами.

Для повноцінної роботи системи часто необхідно окремо налаштовувати сервер, базу даних та систему адміністрування.

Крім того фнтерфейс платформи може виглядати навантаженим для нових користувачів, оскільки існує багато функцій і налаштувань, та вона зовсім майже не підтримує геолокаційні зв’язки. А деякі функції програми можуть бути доступні лише після придбання платної версії програми, а це може стати фінансовою перешкодою для багатьох людей.

Програмне забезпечення інформаційної системи Rocket.Chat (рис.1.2) створене саме для організації онлайн-спілкування та командної взаємодії між користувачами [1]-[3].

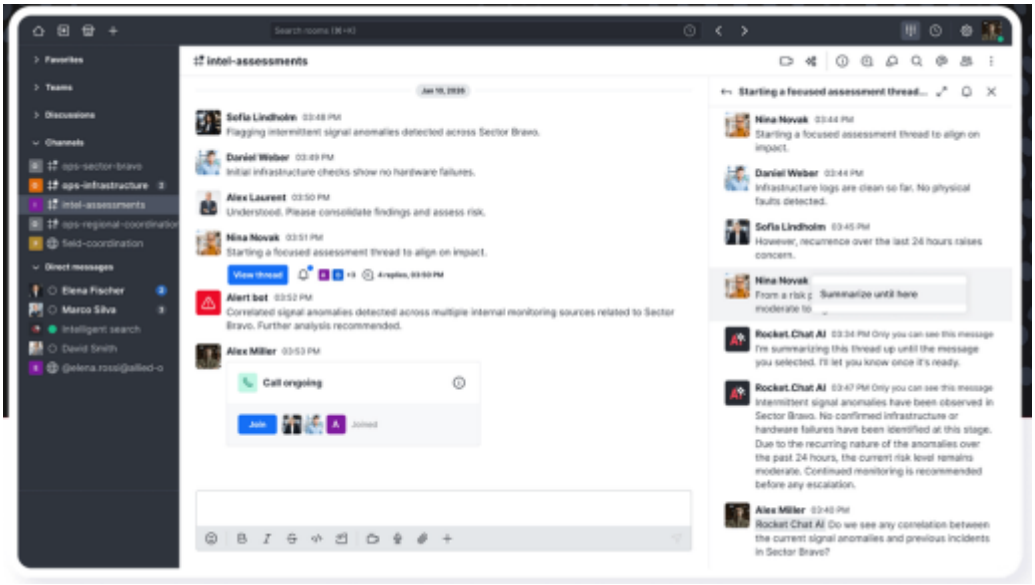

Рисунок 1.2 – Програмне забезпечення інформаційної системи Rocket.Chat

Система підтримує обмін повідомленнями в режиміреального часу, створення окремих кімнат для спілкування та передачу файлівміж учасниками чату.

Rocket.Chat часто використовується компаніями,навчальними установами та організаціями, які хочуть мати власну систему комунікації без використання сторонніх платформ. Однією з головних особливостей Rocket.Chat є можливість самостійного розгортання системи на власному сервері. Завдяки цьому організація може контролювати збереження

повідомлень, керувати доступом користувачів та налаштовувати платформу відповідно до власних потреб.

Система має відкритий вихідний код тому її функціональність можна розширювати або змінювати.

Rocket.Chat підтримує структурування спілкування через окремі кімнати та канали. Це дозволяє розділяти повідомлення між різними групами користувачів та уникати перевантаження одного загального чату великою кількістю повідомлень.

Крім цього, система підтримує миттєве оновленняданих без перезавантаження сторінки, що робить взаємодію між користувачами більш швидкою та зручною.

Під час аналізу системи стало помітно, що Rocket.Chat орієнтується насамперед на командну роботу та корпоративне використання. Платформа підтримує інтеграцію з іншими сервісами, систему ролей та адміністрування користувачів.

Через це Rocket.Chat часто використовують команди розробників, технічні спеціалісти та організації, які потребують власного середовища для внутрішнього спілкування.

Основними особливості та характеристики програмного забезпечення RocketChat:

- підтримка роботи в режимі реального часу: система забезпечує миттєвий обмін повідомленнями між користувачами без необхідності оновлення сторінки;

- підтримка каналів та кімнат: Rocket.Chat дозволяє створювати окремі простори для групового або тематичного спілкування міжкористувачами;

- підтримка відеозв’язку: система підтримує проведення відеоконференцій та дзвінків між учасниками чату;

- система ролей та прав доступу: Rocket.Chat дозволяєналаштовувати рівні доступу для користувачів та адміністраторів системи;

- передача файлів: користувачі можуть надсилати документи,зображення, архіви та інші файли безпосередньо в чаті;

- інтеграція з іншими сервісами: система підтримує підключення GitHub, Jira, GitLab та інших платформ для командної роботи;

- підтримка мобільних пристроїв: Rocket.Chat працює на комп’ютерах, планшетах та смартфонах;

- пошук по історії повідомлень: система дозволяє швидко знаходити необхідні повідомлення або файли серед великої кількості чатів.

Серед недоліків Rocket.Chat можна виділити складніше налаштування системи у порівнянні зі звичайними месенджерами. Для повноцінної роботи необхідно окремо налаштовувати сервер, базу даних та адміністрування користувачів. Окрім його інтерфейс програми дуже схожий на аналоги, тобто не виділяється нічим, як і більшість аналогічних засобів коммунікації. Існують також само і окремі фінансові плани для бізнесів та крупних організацій.

Теж одне з відомих програмних забезпечень Microsoft Teams (рис.1.3), вона є корпоративною програмою яка об’єднує в собі чат, зустрічі, нотатки у робочому середовищі, серед усіх середовищ вище мені здається воно найбільш варте уваги, як дійсно для праці та спілкування у реальному часі [2]-[3].

Система підтримує створення окремих колективів та каналів для різних груп користувачів. Завдяки цьому можна розділити робочі процеси між відділами або окремими завданнями. Також Microsoft Teams підтримує систему рангів та керування доступом користувачів, що дає змогу контролювати доступ до відомостей.

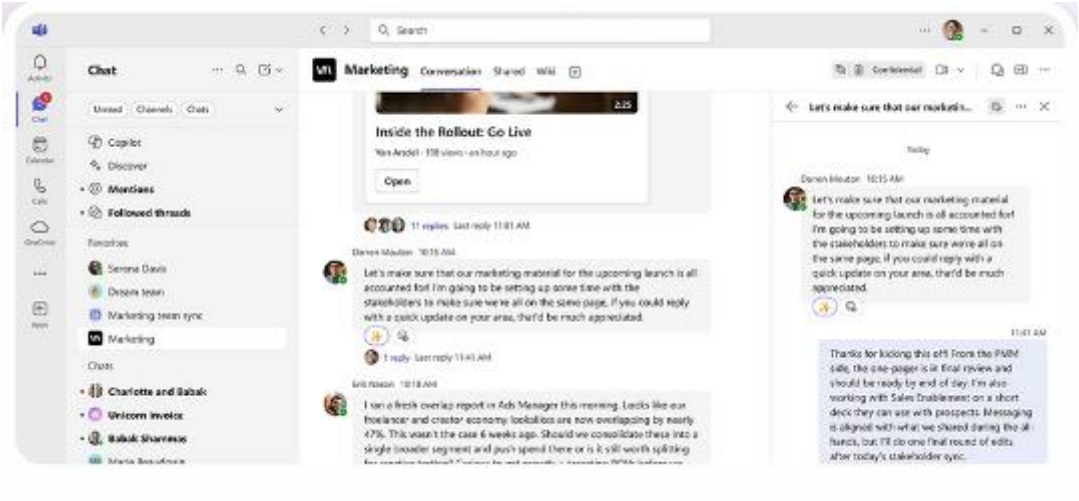

Рисунок 1.3 – Робоче середовище Microsoft Teams

Однією з ключових переваг Microsoft Teams є підтримка сумісної праці над документами. Користувачі можуть одночасно коригувати файли, переглядати зміни та залишати зауваження. Це суттєво полегшує організацію командної роботи та взаємодії між учасниками проєкту.

Водночас, завдяки з’єднаню з Microsoft Word, Excel, PowerPoint платформа дозволяє працювати з файлами просто всередині застосунку.

Не менш важливою перевагою є наявність вебверсії, завдяки чому користувачі можуть користуватись забезчпенням і на мобільних пристроях, і на компьютерах.

Окрім цього, зручна підтримка демонстрації екрана під час відеоконференцій та онлайн-зустрічей, що спрощує проведення презентацій та навчання.

Але суттєвим недоліком Microsoft Teams є те, що вона потребує стабільного зв’язку, особливо під час відеоконференцій або для передачі великих за обсягом файлів. При використанні програми на слабких девайсах, вона може значно «жерти» оперативну пам’ять пристрою та ресурси процесору.

Ще однією особливістю є складний інтерфейс у порівнянні із звичайними застосунками. Через велику кількість вкладок, налаштувань та інструментів новим користувачам інколи буде потрібен час для адаптації до

роботи із системою, але, як і попередні застосунки платформа майже не використовує геолокаційні можливості та не підтримує регіональні механізми.

Усі вищеперераховані застосунки я аналізував оцінював, та навів у таблиці 1.1

Таблиця 1.1 – Порівняння інформаційних систем для спілкування

Критерій порівнння	Mattermost	RocketChat	Microsoft Teams
Зручний інтерфейс	+-	+-	+
Можливість командної роботи	+	+	+
Інеграція з іншими сервісами	+	+	+
Безкоштовна версія	+-	+-	+-
Безека даних	+	+	+
Власне серверне розгортання	+	+	-

За результатами проведеного аналізу можна зробити висновок, що сучасні системи онлайн-спілкування мають досить широкий функціонал та підтримують швидкий обмін повідомленнями між користувачами. Більшість платформ орієнтується на корпоративне використання, командну роботу та організацію внутрішнього спілкування між працівниками або учасниками проєктів.

Разом із цим частина сучасних систем має досить складний інтерфейс, через що новим користувачам потрібен додатковий час для адаптації до роботи із платформою.

Також деякі сервіси потребують окремого налаштування серверної частини або інтеграції з іншими системами, що може створювати складнощі для невеликих організацій або користувачів без технічної підготовки.

Окремо варто звернути увагу на використання ресурсів пристрою. Частина платформ споживає значну кількість оперативної пам’яті та може працювати повільніше на слабших комп’ютерах або мобільних пристроях. Особливо це помітно під час проведення відеоконференцій або роботи з великими обсягами повідомлень і файлів.

Також під час аналізу стало помітно, що більшість сучасних платформ майже не підтримує регіональні механізми взаємодії між користувачами. Системи не використовують автоматичне визначення області користувача, геолокаційні можливості або інтерактивні карти активності, що сьогодні стає все більш актуальним для локальних онлайн-спільнот.

Через це актуальною залишається розробка вебзастосунку «Ukraine Regional Chat», який поєднуватиме швидкий обмін повідомленнями, підтримку роботи в режимі реального часу та регіональну взаємодію між користувачами по областях України.

При розробці програмного забезпечення інформаційної системи Ukraine Regional Chat необхідно забезпечити такі функціональні вимоги як:

- підтримка швидкого обміну повідомленнями між користувачами в режимі реального часу;
- підтримка реєстрації та авторизації у системі;
- підтримка окремих регіональних кімнат для користувачів із різних областей України;
- підтримка приватних повідомлень між користувачами;
- можливість визначення автоматично геолокації користувача;
- створення інтерактивної карти активності користувачів;
- підтримка адаптивного інтерфейсу для мобільних пристроїв;
- підтримка збереження історії повідомлень у базі даних.

1.3 Висновки за розділом 1

Під час виконання першого розділу було проведено аналіз предметноїобласті та сучасних систем онлайн-спілкування. У процесі дослідження сталозрозуміло, що сьогодні сервіси для обміну повідомленнями є важливою частиною не лише особистого спілкування, але й навчання, командної роботита організації взаємодії між користувачами.

Було встановлено, що сучасні системи онлайн-комунікації активно використовують технології роботи в режимі реального часу. Завдяки цьому користувачі можуть практично миттєво отримувати повідомлення та швидко обмінюватися інформацією без необхідності постійного оновлення сторінки.

Також під час аналізу стало помітно, що більшість популярних платформ орієнтується переважно на корпоративне середовище та командну роботу.

У межах розділу було проаналізовано такі інформаційні системи Mattermost, Rocket.Chat та Microsoft Teams. У результаті аналізу були визначені їх основні можливості, переваги та недоліки. Більшість платформ підтримує створення окремих кімнат і каналів, систему ролей, передачу файлів та інтеграцію з іншими сервісами. Разом із цим частина систем має складний інтерфейс, потребує додаткового налаштування серверної частини або використовує значну кількість ресурсів пристрою.

Я проаналізував використання геолокаційних можливостей у сучасних вебзастосунках. Під час аналізу стало зрозуміло, що більшість систем онлайн-спілкування майже не підтримує регіональні механізми взаємодії між користувачами та не використовує інтерактивні карти активності. Саме через це напрям локальних вебчатів залишається актуальним та перспективним.

Мною були сформовані основні функціональні вимоги до вебзастосунку «Ukraine Regional Chat». Система повинна підтримувати обмін повідомленнями в режимі реального часу, реєстрацію та авторизацію користувачів, окремі регіональні кімнати, приватні повідомлення,

автоматичне визначення геолокації користувача, інтерактивну карту активності та збереження історії повідомлень у базі даних.

Усі ці висновки з аналізу предметної області стали основою для подальшого проєктування структури системи та вибору технологій для реалізації вебзастосунку.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Вибір мови програмування

Під час розробки вебзастосунку Ukraine Regional Chat як основну мову програмування було обрано Java. Дана мова програмування є однією з найпоширеніших у сфері створення серверних вебзастосунків та корпоративних інформаційних систем [15]-[16].

Мова програмування Java з’явилася ще в середині 1990-х років, але навіть зараз залишається однією з найвідоміших мов у сфері розробки програмного забезпечення. За цей час вона використовувалась у великій кількості різних проєктів - від невеликих програм до складних серверних систем та вебплатформ [15].

Назва Java пов’язана з кавою. Саме тому символом мови стала чашка гарячого напою. Такий логотип швидко став упізнаваним серед програмістів і залишається одним із найвідоміших у сфері програмування [15].

Цікавим є те, що Java довгий час використовувалась у мобільних телефонах ще до появи сучасних смартфонів. На старих кнопкових телефонах можна було запускати невеликі Java-програми та ігри. Для багатьох користувачів саме Java стала першою технологією, пов’язаною з мобільними застосунками. Та й на зараз вона активно використовується для розробки веб-сервісів, систем онлайн спілкування та інших програмних ріень, які потребують сабільної роботи [15].

Однією з причин вибору Java стала її надійність та стабільність. Мова підтримує об’єктно-орієнтований підхід до програмування, що дозволяє структурувати код та спрощує подальшу підтримку програмного забезпечення.

Крім цього, Java має велику кількість готових бібліотек та фреймворків длястворення вебзастосунків

Архітектура Java побудована таким чином, щоб забезпечити платформонезалежність, стабільність роботи та можливість запуску програм

на різних операційних системах без зміни програмного коду. Основною особливістю Java є використання віртуальної машини Java Virtual Machine (JVM), яка виконує проміжний байт-код та забезпечує роботу програми незалежно від типу операційної системи або апаратного забезпечення [15]-[16].

Під час компіляції Java-програма перетворюється не відразу у машинний код, а у спеціальний байт-код. Після цього JVM виконує даний байт-код та перетворює його у команди, зрозумілі конкретній операційній системі. Завдяки цьому Java-програми можуть працювати на Windows, Linux або macOS без значних змін у структурі проєкту [15].

Ще одним важливим компонентом є Java Runtime Environment (JRE).

Дане середовище відповідає за запуск уже готових Java-програм та містить необхідні бібліотеки й компоненти для роботи системи [19].

Також архітектура Java підтримує велику кількість готових бібліотек, фреймворків та інструментів для розробки вебзастосунків. Наприклад, Spring Boot, Hibernate або Maven. Це значно спрощує створення складних інформаційних систем та прискорює процес розробки програмного забезпечення [4]-[5].

Незважаючи на те, що Java була створена досить давно, мова продовжує активно оновлюватися та використовується у сучасних вебтехнологіях разом і з іншими, дивитись табл. 2.1.

Таблиця 2.1 обгрунтування вибору мови програмування

Критерій порівняння	Java	C#	Python
1	2	3	4
Підтримка веброзробки	+	+	+
Кросплатформеність	+	+-	+
Підтримка WebSocket	+	+	+
Простота вивчення	+-	+	+-
Підтримка серверної архітектури	+	+	+
Робота з великим навантаженням	+	+	+-
Використання у корпоратвних системах	+	+	-

Кінець таблиці 2.1

1	2	3	4
Споживання ресурсів сервера	+-	-	+
Зручність тестування пз	+	+-	+
Підтримка хмарних технологій	+	+	+
Доцільність до систем спілкуванн	+	+-	-

Після проведення порівняння мов програмування було визначено, що Java є найбільш доцільним вибором для розробки вебзастосунку Ukraine Regional Chat. Дана мова програмування забезпечує стабільну роботу серверної частини системи, підтримує сучасні вебтехнології та добре підходить для створення багатокористувацьких вебзастосунків.

Під час аналізу стало помітно, що Java має високий рівень підтримки серверної архітектури та добре працює з WebSocket-з'єднаннями, які є важливою частиною систем онлайн-спілкування. Також мова підтримує роботу з великим навантаженням, що дозволяє одночасно обробляти значну кількість підключень та повідомлень між користувачами [7]-[16].

На відміну від Python, Java краще підходить для створення високонавантажених серверних систем та корпоративних вебзастосунків. Крім цього, Java має велику кількість готових бібліотек і фреймворків для роботи з базами даних, WebSocket та REST API [15]-[19].

У порівнянні з C# мова Java також є більш універсальною для кросплатформеної розробки та часто використовується під час створення серверних вебсистем, які повинні працювати на різних операційних системах.

2.2 Серверна частина вебзастосунку

Під час розробки вебзастосунків саме серверна частина найчастіше відповідає за основне навантаження системи. Вона обробляє повідомлення, запити користувачів, взаємодіє з базою даних та контролює логіку роботи всього вебсервісу. Для систем онлайн-спілкування це особливо важливо,

оскільки сервер повинен стабільно працювати навіть при одночасному підключенні великої кількості користувачів [16].

Для створення серверної частини вебзастосунку «Ukraine Regional Chat» було використано Spring Boot. Даний фреймворк працює на основі Java та досить часто використовується у сучасній веброзробці. Причина його популярності доволі проста. Spring Boot дозволяє швидше створювати серверні застосунки та не потребує великої кількості складних налаштувань на початку роботи [4]-[5].

За час розробки стало помітно, що саме Spring Boot добре підходить для чатів та систем із постійною взаємодією між користувачами. Фреймворк має готову підтримку WebSocket, REST API та роботи з базами даних. Через це не потрібно окремо реалізовувати частину базових механізмів вручну [4],-[5].

Серверна частина «Ukraine Regional Chat» побудована за модульним принципом. Окремі компоненти відповідають за різні функції системи. Це спростило структуру проєкту та зробило код більш зрозумілим. Наприклад, ChatController відповідає за обробку повідомлень та роботу кімнат чату. AuthController використовується для реєстрації та авторизації користувачів. Окремо реалізовано UserController, який працює зі статистикою та даними користувачів [17].

Для взаємодії між клієнтською частиною та сервером у системі використовуються REST-запити та WebSocket-з’єднання. REST API використовується для роботи з авторизацією, статистикою та окремими даними системи. Але для звичайного HTTP-з’єднання є одна проблема. Користувачу потрібно постійно оновлювати сторінку або надсилати нові запити, щоб отримати повідомлення [19].

Саме через це у вебзастосунку використовується WebSocket. Технологія дозволяє підтримувати постійне підключення між клієнтом та сервером. Повідомлення передаються майже миттєво. Без затримок і без

оновлення сторінки. Для чатів це набагато зручніше, ніж класична модель HTTP-запитів [17].

Коли я починав створення системи також використовувався протокол STOMP. Він дозволяє працювати з окремими каналами повідомлень та кімнатами чату. У результаті у вебзастосунку вдалося реалізувати загальний чат, регіональні кімнати та приватні повідомлення між користувачами [6].

Ще однією важливою частиною серверної системи стала робота з базою даних. Для цього використовувалась PostgreSQL. Під час локального тестування застосовувалась H2 Database, оскільки вона простіше налаштовується та добре підходить для тестових запусків [13]-[17].

У серверній частині також було реалізовано систему авторизації користувачів. Паролі не зберігаються у відкритому вигляді. Для їх захисту використовується BCrypt. Подібний підхід зараз використовується у більшості сучасних вебсистем, оскільки звичайне текстове збереження паролів давно вважається небезпечним [17]-[20].

Під час тестування серверна частина працювала стабільно навіть при одночасному підключенні декількох користувачів. Повідомлення відображалися майже без затримки, а система коректно оновлювала статуси, кімнати та список активних користувачів.

У результаті використання Spring Boot, WebSocket та PostgreSQL вдалося створити серверну частину вебзастосунку, яка підтримує роботу системи онлайн-спілкування в режимі реального часу та забезпечує стабільну взаємодію між користувачами з різних областей України [4]-[5], [7]-[13].

2.3 Висновок за розділом 2

У другому розділі було проведено аналіз технологій та програмних засобів, які використовувалися під час розробки вебзастосунку «Ukraine Regional Chat». Під час вибору інструментів основна увага приділялась стабільності роботи системи, підтримці обміну повідомленнями в режимі

реального часу та можливості подальшого розширення функціональності вебзастосунку.

У підсумку порівняння мов програмування було визначено, що Java є доцільним вибором для створення серверної частини системи. Мова добре підходить для багатокористувацьких вебзастосунків, підтримує роботу з високим навантаженням та має велику кількість готових бібліотек і фреймворків для веброзробки. Також важливою перевагою Java стала підтримка багатопотоковості та можливість стабільної роботи серверної частини при одночасному підключенні декількох користувачів.

У ході практичної реалізації сервера використовувався Spring Boot. Даний фреймворк дозволив спростити створення вебзастосунку та реалізувати роботу REST API, WebSocket-з’єднання та взаємодію з базою даних. Крімцього, Spring Boot підтримує модульну структуру проєкту, що зробило кодбільш структурованим та зручним для подальшої підтримки.

Детально проаналізовано технології WebSocket, яка забезпечує миттєвий обмін повідомленнями між користувачами. Використання постійного з’єднання між клієнтом та сервером дозволило реалізувати роботу чату без необхідності оновлення сторінки. Для організації каналів повідомлень та кімнат чату використовувався протокол STOMP.

Проведений аналіз показав, що використані технології добре підходять для створення систем онлайн-спілкування та забезпечують стабільну роботу вебзастосунку в режимі реального часу. Отримані результати стали основою для практичної реалізації вебзастосунку «Ukraine Regional Chat» у наступному розділі.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

3.1 Загальна структура вебзастосунку

В процесі проєктуваня вебзастосунку «Ukraine Regional Chat» одним із головних завдань стало проєктування структури системи. На початковому етапі було важливо не просто реалізувати чат для обміну повідомленнями, а створити таку структуру, яка дозволила б поступово розширювати функціональність без повного переписування проєкту. На практиці це виявилося важливим, оскільки під час розробки функціонал системи декілька разів змінювався. Спочатку застосунок планувався як звичайний вебчат, проте пізніше були додані регіональні кімнати, авторизація користувачів, система статусів та підтримка геолокації.

Для реалізації системи було обрано клієнт-серверний підхід. Подібна архітектура використовується у більшості сучасних вебзастосунків, оскільки дозволяє розділити інтерфейс користувача та логіку обробки даних. У даному випадку браузер виконує роль клієнта, через який користувач взаємодіє із системою, а сервер відповідає за збереження повідомлень, перевірку користувачів та передачу інформації між різними учасниками чату. Подібний підхід використовується у таких платформах як Discord, Mattermost або Rocket.Chat, проте в даному випадку структура була адаптована під потреби локального регіонального спілкування.

Клієнтська частина вебзастосунку реалізована у вигляді односторінкового інтерфейсу. Користувачу не потрібно постійно переходити між сторінками, оскільки всі основні елементи знаходяться в одному середовищі. Після входу в систему відкривається головне вікно чату, де відображаються повідомлення, список активних користувачів та доступні кімнати для спілкування. На мою думку, такий підхід є більш зручним для чатів, оскільки користувач не втрачає контекст спілкування та може швидко переключатися між різними режимами взаємодії.

При розробці структури вебзастосунку важливо було уникнути хаотичного розміщення серверної логіки. Через це система була поділена на декілька незалежних компонентів, які відповідають за окремі частини роботи вебзастосунку. Такий підхід дозволяє легше підтримувати код та спрощує процес внесення змін у майбутньому. Наприклад, реалізація авторизації, обробка повідомлень або робота з геолокацією не залежать одна від одної, тому внесення змін у певний компонент не потребує зміни всієї системи.

Особливістю структури «Ukraine Regional Chat» стала орієнтація саме на регіональне спілкування між користувачами. У більшості вебчатів користувачі взаємодіють у межах одного загального середовища, проте в даному застосунку додатково реалізовано окремі кімнати залежно від області України. Це дозволяє створити більш локальну взаємодію між людьми та зробити систему ближчою до користувачів. Наприклад, людина із Запорізької області може одразу потрапити до відповідного середовища спілкування без необхідності ручного пошуку груп або каналів.

Ще однією важливою частиною структури стала організація обміну повідомленнями в режимі реального часу. Для чатів звичайне оновлення сторінки виглядає незручним і вже практично не використовується у сучасних системах. Саме тому сервер підтримує постійне підключення між користувачем та вебзастосунком. У результаті повідомлення з’являються майже миттєво після надсилання, а список активних користувачів оновлюється автоматично.

За час проєктування структури також враховувалась можливість подальшого масштабування системи. Наприклад, у майбутньому до вебзастосунку можна додати систему реакцій на повідомлення, окремі налаштування профілю, групові дзвінки або додаткові механізми модерації. Структура проєкту дозволяє реалізувати подібні функції без необхідності повної зміни архітектури системи.

Окремо варто звернути увагу на збереження даних користувачів та історії повідомлень. У багатьох простих чатах після перезавантаження

сторінкиповідомлення можуть втрачатися. У даному вебзастосунку історія спілкування зберігається у базі даних, що дозволяє користувачам бачити попередні повідомлення навіть після повторного входу в систему. На практиці це робить чат більш зручним та наближає його до сучасних платформ онлайн спілкування.

Завдяки цьому була сформована структура вебзастосунку, яка поєднує простий інтерфейс, серверну стабільність та підтримку роботи в режимі реального часу. Важливо й те, що система не перевантажена зайвими елементами. Основний акцент зроблено саме на швидкому спілкуванні, регіональній взаємодії між користувачами та зручності використання.

3.2 Реалізація системи обміну повідомленнями

Однією з головних функцій вебзастосунку Ukraine Regional Chat є можливість швидкого обміну повідомленнями між користувачами. Під час розробки основна увага приділялась тому, щоб передача повідомлень відбувалась без затримок, а сам процес спілкування був максимально простим та зрозумілим. На практиці користувачі звикли до того, що повідомлення у чатах відображаються миттєво. Через це використання звичайного оновлення сторінки виглядало б незручним та застарілим.

Для реалізації системи повідомлень використовувалась технологія WebSocket. На відміну від класичних HTTP-запитів, де браузер повинен постійно звертатися до сервера для отримання нової інформації, WebSocket дозволяє підтримувати постійне з’єднання між користувачем та системою. У результаті повідомлення надсилаються та відображаються майже одразу після відправлення.

У вебзастосунку реалізовано декілька режимів взаємодії між користувачами. Першим із них є загальний чат, у якому можуть спілкуватися всі користувачі системи незалежно від регіону. Такий формат дозволяє

швидко обмінюватися інформацією та підтримувати загальну взаємодію між учасниками системи.

Окремо було реалізовано регіональні кімнати. На мою думку, саме ця частина найбільше відрізняє Ukraine Regional Chat від більшості звичайних вебчатів. Після входу до системи користувач автоматично потрапляє до кімнати, яка відповідає його області. Наприклад, користувачі із Запорізької, Львівської або Одеської області можуть спілкуватися в окремих середовищах. Подібний підхід робить взаємодію більш локальною та дозволяє обговорювати новини або питання, які стосуються конкретного регіону.

Ще однією функцією системи стали приватні повідомлення між користувачами. У процесі тестування стало зрозуміло, що загального чату недостатньо, оскільки інколи виникає потреба у персональному спілкуванні. Саме тому було реалізовано можливість надсилання повідомлень конкретному користувачу без відображення тексту для інших учасників чату. (рис. 3.1)

Рисунок 3.1 – Діаграма системи обміну повідомлення в чаті

Під час підключення або виходу користувача система автоматично формує службові повідомлення. Наприклад, у чаті відображається інформація про те, що користувач приєднався до системи або завершив роботу із застосунком. На практиці це дозволяє краще розуміти активність учасників та бачити, хто зараз перебуває онлайн.

Також у системі реалізовано механізм зміни статусів користувачів. Користувач може перебувати у статусі ONLINE, AWAY або BUSY. Під час тестування ця функція виявилась корисною, оскільки дозволяє швидко зрозуміти, чи готова людина до спілкування, або тимчасово недоступна. Для зручності роботи список активних користувачів оновлюється автоматично. Коли новий учасник входить у систему або змінює свій статус, дані одразу оновлюються без необхідності перезавантаження сторінки. Це робить роботу вебзастосунку більш плавною та наближеною до сучасних месенджерів.

У межах розробки системи повідомлень також було передбачено збереження історії чатів. Повідомлення не зникають після закриття браузера, а залишаються у базі даних. Завдяки цьому після повторного входу користувач може переглядати попереднє листування, що робить систему більш зручною для постійного використання.

У результаті реалізована система обміну повідомленнями дозволила забезпечити швидке спілкування між користувачами, підтримку регіональної взаємодії та роботу чату в режимі реального часу. При цьому вебзастосунок залишився простим у використанні та не перевантаженим зайвими функціями.

3.3 Реалізація системи авторизації користувачів

Під час створення вебзастосунку одразу стало зрозуміло, що звичайного входу за ім’ям користувача буде недостатньо. Якщо не реалізувати систему авторизації, будь-який користувач зможе використовувати чуже ім’я або видавати себе за іншу людину. Через це у

вебзастосунку було реалізовано окремий механізм реєстрації та входу в систему.

Користувач починає роботу із застосунком зі створення облікового запису. Під час реєстрації система перевіряє введене ім’я користувача, пароль та додаткову інформацію. Якщо логін уже використовується, сервіс повідомляє про це та не дозволяє створити дубльований акаунт. Подібна перевірка дозволяє уникнути плутанини між користувачами та спрощує подальшу взаємодію в чаті.

На етапі створення авторизації було важливо не лише створити можливість входу в систему, а й забезпечити базовий рівень захисту даних. Через це паролі не зберігаються у відкритому вигляді. Для їх захисту використовується алгоритм BCrypt, який перетворює пароль у спеціальний хеш. Навіть якщо отримати доступ до бази даних, побачити реальний пароль користувача неможливо.

На мою думку, використання BCrypt є виправданим рішенням навіть для відносно невеликого вебзастосунку. У багатьох студентських проєктах можна зустріти збереження паролів у звичайному текстовому форматі, проте такий підхід вже давно вважається небезпечним. Через це було важливо реалізувати хоча б базовий механізм захисту персональних даних.

Після успішної авторизації користувач отримує доступ до основного функціоналу вебзастосунку. Саме на цьому етапі система отримує інформацію про регіон користувача, місто та координати. Надалі ці дані використовуються для роботи регіональних кімнат і формування списку активних користувачів.

Під час реалізації авторизації також було враховано перевірку коректності введених даних. (рис. 3.2) Наприклад, система не дозволяє створитиобліковий запис із порожнім ім’ям користувача або надто коротким паролем.Подібні перевірки хоч і здаються простими, але дозволяють уникнути частини помилок ще на етапі реєстрації.

Рисунок 3.2 – Діаграма прецеденту входу користувачів до чату

У структурі вебзастосунку система авторизації реалізована через окремий контролер, який відповідає за обробку запитів реєстрації та входу. Після отримання запиту сервер перевіряє інформацію, звертається до бази даних та повертає відповідь користувачу. У разі успішного входу система надає доступ до функцій чату, а при помилці виводить відповідне повідомлення.

Окремо варто відзначити, що авторизація зробила вебзастосунок більш

структурованим. Користувачі мають власні акаунти, можуть зберігати свою інформацію та використовувати чат без необхідності щоразу повторно налаштовувати дані профілю.

Кінцевим етапом створення системи авторизації, стало те , шо вона допомогла зробити вебзастосунок більш безпечним та зручним для користувачів. Крім базовогозахисту даних, система забезпечує унікальність користувачів, спрощує ідентифікацію в чаті та створює основу для подальшого розширення функціональності профілю.

3.4 Реалізація геолокації та регіональної взаємодії

Однією з функцій, яка відрізняє вебзастосунок Ukraine Regional Chat від більшості звичайних чатів, стала підтримка геолокації користувачів. Під час розробки виникла ідея зробити спілкування не лише загальним, а й більш локальним. У багатьох сучасних месенджерах користувачі можуть обмінюватися повідомленнями незалежно від місця перебування, проте сама система майже ніколи не враховує регіон або місто людини. Через це було вирішено створити механізм, який дозволяє автоматично визначати місцезнаходження користувача та використовувати цю інформацію для організації взаємодії.

Після відкриття вебзастосунку браузер надсилає запит на доступ до геолокації користувача. Якщо людина погоджується надати дозвіл, система отримує координати місцезнаходження через Geolocation API браузера. Надалі ці координати використовуються для визначення приблизного регіону користувача, відображення активності на карті та організації локального спілкування.

Подібний підхід виявився більш зручним, ніж ручне введення області. Користувачу не потрібно самостійно шукати свій регіон або витрачати час на додаткові налаштування. Після входу система автоматично визначає область та місто, а далі користувач отримує доступ до відповідної регіональної кімнати. Наприклад, людина із Запорізької області може одразу спілкуватися з користувачами зі свого регіону, не переходячи між десятками різних каналів.

У ході формування стало помітно, що браузерна геолокація працює точніше, ніж визначення через IP-адресу. Особливо це помітно у великих містах або при використанні мобільного Інтернету. Разом із цим повністю відмовлятися від визначення через IP недоцільно, оскільки деякі користувачі можуть заборонити доступ до місцезнаходження. У такому випадку система використовує резервний спосіб визначення приблизної області через IP-

адресу, що дозволяє вебзастосунку залишатися працездатним навіть без доступу до точних координат.

Підвищену увагу було зосереджено на нашій інтерактивній карті активності користувачів. Для реалізації даної функції використовувалась бібліотека Leaflet та сервіс OpenStreetMap. Вибір саме цих технологій був пов’язаний не лише з простотою інтеграції, а й із тим, що вони є безкоштовними та не потребують складного налаштування. На відміну від частини комерційних картографічних платформ, OpenStreetMap дозволяє використовувати карти без додаткових витрат, що є важливим для навчального проєкту.

Після успішного визначення координат користувача система автоматично додає відповідний маркер на карту. Це дозволяє побачити, звідки саме підключаються активні учасники чату. Якщо новий користувач входить у систему, карта оновлюється майже миттєво. Аналогічно відбувається й після виходу користувача із вебзастосунку (рис. 3.3)

Рисунок 3.3 – Діаграма визначення регіональної незалежності користувача

На мою думку, саме карта стала однією з найбільш цікавих частин проєкту. У більшості чатів користувачі бачать лише текст повідомлення та

ім’я співрозмовника. У даному випадку можна додатково зрозуміти географію активності учасників та побачити, наскільки різні області України представлені у спілкуванні. Особливо цікаво це виглядає, коли одночасно в чаті перебувають люди із західної, центральної та південної частини країни.

Вагомим рішенням стало створення регіональних кімнат для спілкування. Під час аналізу стало зрозуміло, що один загальний чат може швидко перевантажуватися повідомленнями, особливо якщо кількість користувачів збільшується. Через це було реалізовано окремі кімнати для різних областей України. Подібний підхід дозволяє обговорювати локальні події, знайомитися з людьми зі свого регіону або швидше знаходити користувачів із близького середовища.

Під час практичного використання стало помітно, що локальна взаємодія працює більш природно, ніж очікувалось на початку розробки. Користувачі швидше включаються в обговорення, коли бачать знайомий регіон або місто. У певному сенсі це створює відчуття локальної спільноти, чого часто не вистачає у звичайних глобальних чатах.

Всі ці кроки сприяли стали пощтовхом, зробитити вебзастосунок більш інтерактивним та прив’язаним до реального середовища користувачів. Поєднання браузерної геолокації, резервного визначення через IP, інтерактивної карти та регіональних кімнат стало важливою частиною Ukraine Regional Chat та допомогло зробити систему більш відмінною від традиційних вебчатів.

3.5 Реалізація структури бази даних

На час розробки вебзастосунку Ukraine Regional Chat важливо було забезпечити стабільне збереження інформації. На початкових етапах розробки чат працював без постійного збереження повідомлень, проте досить швидко стало зрозуміло, що такий підхід створює незручності. Після оновлення сторінки або повторного входу користувач втрачав історію

спілкування, а інформація про учасників не зберігалась. Для системи онлайн-комунікації це виглядало недоліком, тому було прийнято рішення реалізувати повноцінну базу даних.

Основною системою керування базами даних у проєкті стала PostgreSQL. Дане рішення було обрано через стабільність роботи, хорошу сумісність із Spring Boot та можливість працювати з великим обсягом інформації. PostgreSQL часто використовується у вебзастосунках та корпоративних системах, тому вибір саме цієї СКБД виглядав логічним.

Разом із цим під час локальної розробки використовувалась H2 Database. Це дозволяло швидше тестувати окремі функції без необхідності щоразу налаштовувати зовнішню базу даних. На практиці H2 виявилась зручною саме для перевірки роботи авторизації, повідомлень та взаємодії між сервером і клієнтом.

Структура бази даних вебзастосунку побудована навколо двох основних сутностей. Перша використовується для роботи з користувачами. У ній зберігається інформація про ім’я користувача, пароль, область, місто, координати та дата створення акаунта. Окрему увагу було приділено захисту паролів. Вони не записуються у відкритому вигляді, а проходять процедуру хешування через BCrypt. Це дозволяє підвищити безпеку та зменшити ризики втрати персональних даних.

Друга сутність відповідає за повідомлення користувачів. У базі даних зберігається текст повідомлення, ім’я автора, тип повідомлення, дата створення, інформація про регіон та додаткові параметри для приватного листування. Подібна структура дозволяє відокремлювати звичайні повідомлення від службових повідомлень про вхід або вихід користувачів.

На мою думку, збереження історії повідомлень стало однією з найбільш важливих частин системи. У багатьох простих навчальних чатах листування зникає після перезапуску сторінки. У даному випадку користувач після повторного входу може переглянути попередню історію

спілкування, що робить вебзастосунок більш наближеним до сучасних месенджерів.

Для взаємодії з базою даних використовувалась технологія Spring Data JPA. Вона дозволила працювати з інформацією через Java-об’єкти без постійного написання SQL-запитів вручну. Наприклад, система може швидко перевірити наявність користувача, отримати повідомлення конкретної кімнати або знайти інформацію про користувачів певного регіону.

Під час тестування структура бази даних працювала стабільно. Повідомлення коректно зберігалися, інформація про користувачів не втрачалася, а система без проблем відновлювала історію після повторного входу до вебзастосунку. Це позитивно вплинуло на загальну стабільність роботи чату.

У результаті використання PostgreSQL та Spring Data JPA вдалося реалізувати надійну структуру збереження даних, яка забезпечує підтримку історії повідомлень, роботу авторизації та стабільне функціонування вебзастосунку в цілому.

3.6 Реалізація користувацького інтерфейсу

В процесі проєктування інтерфейсу основна увага приділялась простоті використання та швидкому доступу до основних функцій. На практиці велика кількість чатів виглядає перевантаженою. Користувачу доводиться переходити між різними сторінками, відкривати додаткові меню або витрачати час на пошук необхідних функцій. Через це під час розробки було важливо зробити інтерфейс максимально зрозумілим навіть для людини, яка вперше відкрила вебзастосунок.

Інтерфейс реалізований за допомогою HTML, CSS та JavaScript. Основна логіка побудована у форматі односторінкового застосунку, де користувач отримує доступ до більшості функцій без переходу між сторінками. Після авторизації відкривається головне вікно чату, яке містить

основне поле повідомлень, список користувачів, кімнати для спілкування та інтерактивну карту активності.

Коли створювався інтерфейс, потрібно було зберегти баланс між простотою та функціональністю. Через це більшість елементів розташована так, щоб користувач міг швидко знайти необхідну функцію. Наприклад, кімнати чату знаходяться у швидкому доступі, список активних користувачів оновлюється автоматично, а повідомлення відображаються у зручному форматі без перевантаження зайвими елементами.

Значний акцент був поставлений на структурі повідомлень. У чаті відображається ім’я користувача, текст повідомлення, час відправлення та додаткова інформація про активність учасників. Службові повідомлення про вхід або вихід із системи також виводяться окремо, що дозволяє краще розуміти, хто зараз перебуває онлайн.

Для підвищення зручності використання було реалізовано систему статусів користувачів. Людина може позначити себе як ONLINE, AWAY або BUSY. Під час тестування це виявилось досить корисним, оскільки дозволяло краще розуміти доступність співрозмовника. Наприклад, користувач у статусі BUSY може тимчасово не відповідати на повідомлення, що зменшує кількість непорозумінь у спілкуванні.

У вебзастосунку також реалізовано темну тему оформлення. На мою думку, це стало корисним рішенням, оскільки значна частина користувачів звикла до темного інтерфейсу в сучасних месенджерах. Крім цього, темний режим виглядає комфортніше при тривалому використанні системи, особливо у вечірній час.

Взагалом інтерфейс вебзастосунку не складний у порівнянні з аналогами, але в ньому також є багато своїх моментів, які можуть зацікавати активного користувача (рис. 3.4)

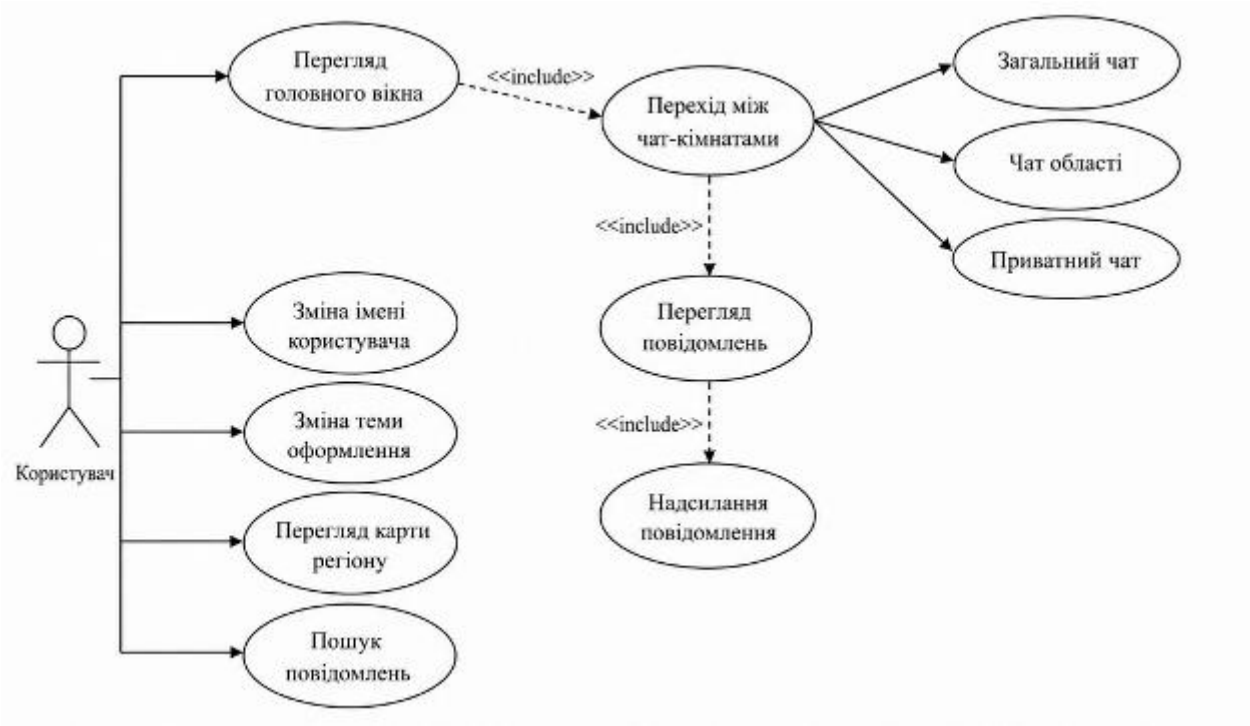

Рисунок 3.4 – Діаграма взаємодії користувача з інтерфейсом вебзастосунку

Ще однією важливою частиною стала адаптивність інтерфейсу. Під час тестування вебзастосунок перевірявся не лише на комп’ютері, а й на смартфонах. Це дозволило зробити інтерфейс більш універсальним. Елементи керування залишаються доступними навіть на менших екранах, а основний функціонал чату працює без помітних обмежень. Користувачу не потрібно довго вивчати структуру застосунку, оскільки більшість функцій зрозуміла інтуїтивно. Саме це, на мою думку, стало однією з сильних сторін вебзастосунку.

У результаті реалізований користувацький інтерфейс забезпечив зручну взаємодію із системою, швидкий доступ до основних функцій та комфортне спілкування між користувачами незалежно від типу пристрою.

3.7 Висновки за розділом 3

Розроблено програмне забезпечення вебзастосунку Ukraine Regional Chat. Під час реалізації встановлено, що для подібних систем особливо

важливими є швидкість передачі повідомлень, стабільність роботи сервера та зручність взаємодії користувачів із функціоналом платформи (рис.3.5).

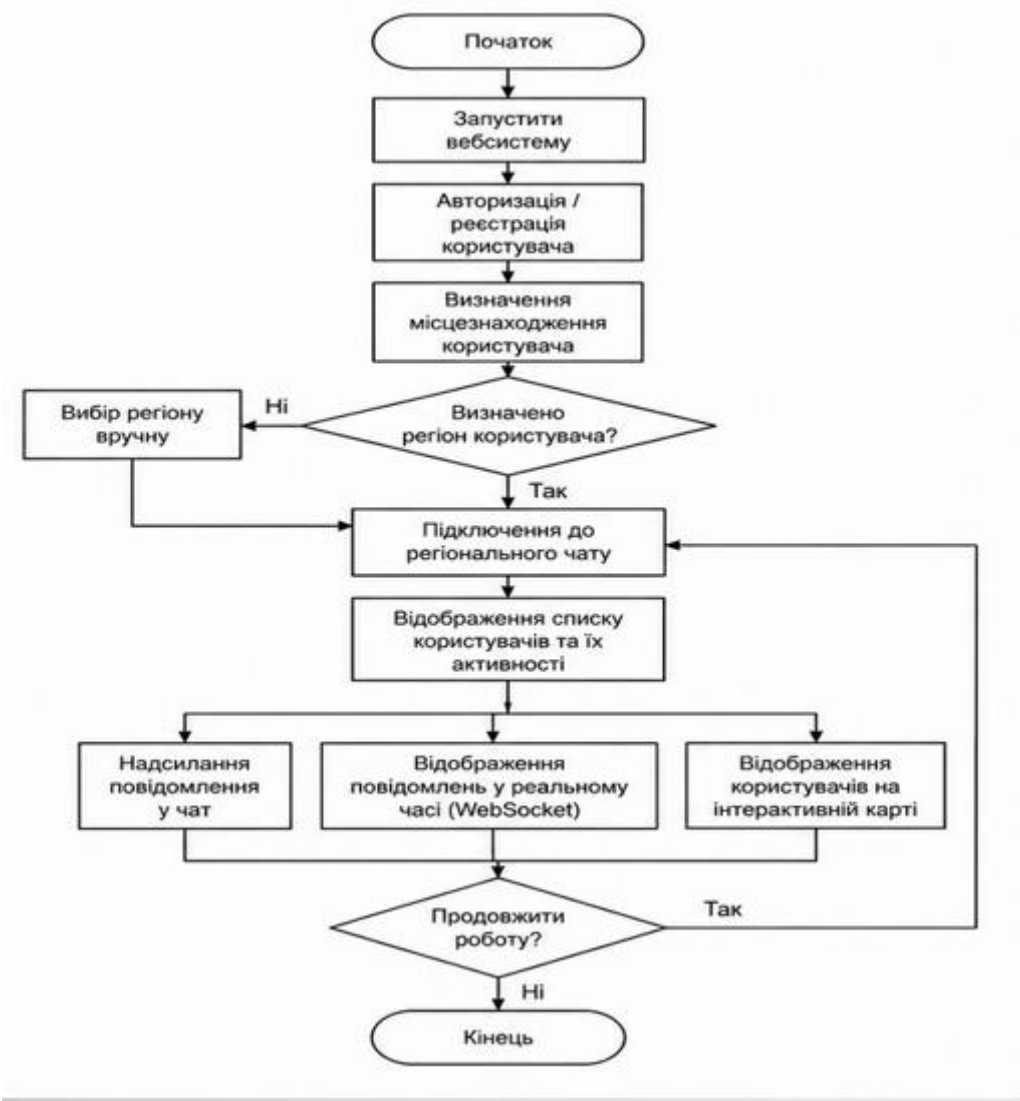

Рисунок 3.5 – Схема функціонування вебзастосунку

Запропоновано структуру програмного забезпечення вебзастосунку, яка базується на клієнт-серверному підході. Серверна частина системи реалізована із використанням Spring Boot та включає модулі обробки повідомлень, взаємодії між користувачами, визначення регіональної належності та забезпечення мережевої комунікації через WebSocket. Використання такого підходу дало змогу забезпечити постійне з’єднання між клієнтом і сервером, що є важливим для роботи чатів без затримок та ручного оновлення сторінки.

Клієнтська частина вебзастосунку включає модулі взаємодії з користувачем, обміну повідомленнями, управління чат-кімнатами та відображення інформації про регіон користувача. Окремо реалізовано інтерактивну карту на базі Leaflet та OpenStreetMap, що дозволило зробити взаємодію із системою більш зрозумілою. Під час тестування було помітно, що візуальне представлення області користувача значно покращує орієнтацію у функціоналі платформи, особливо для нових відвідувачів системи.

Розроблено механізм організації чатів, який передбачає наявність загального каналу спілкування, регіональних кімнат та окремих каналів для персоналізованої взаємодії. Такий підхід дозволяє уникнути перевантаження інформаційного простору та спрощує комунікацію між користувачами.

Описано особливості функціонування програмного забезпечення вебзастосунку та реалізацію його інтерфейсу. Виконано проєктування елементів взаємодії користувача із системою, забезпечено адаптивність інтерфейсу та підтримку обміну повідомленнями у режимі реального часу.

4 ЕКСПЛУАТАЦІЯ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ

4.1 Програмне забезпечення програми

Після запуску системи програмного забезпечення для спілкування користувача вітає автоматично вікно реєстрації, де є можливість зареєструватися, вікно вводу логіну та паролю (рис. 4.1)

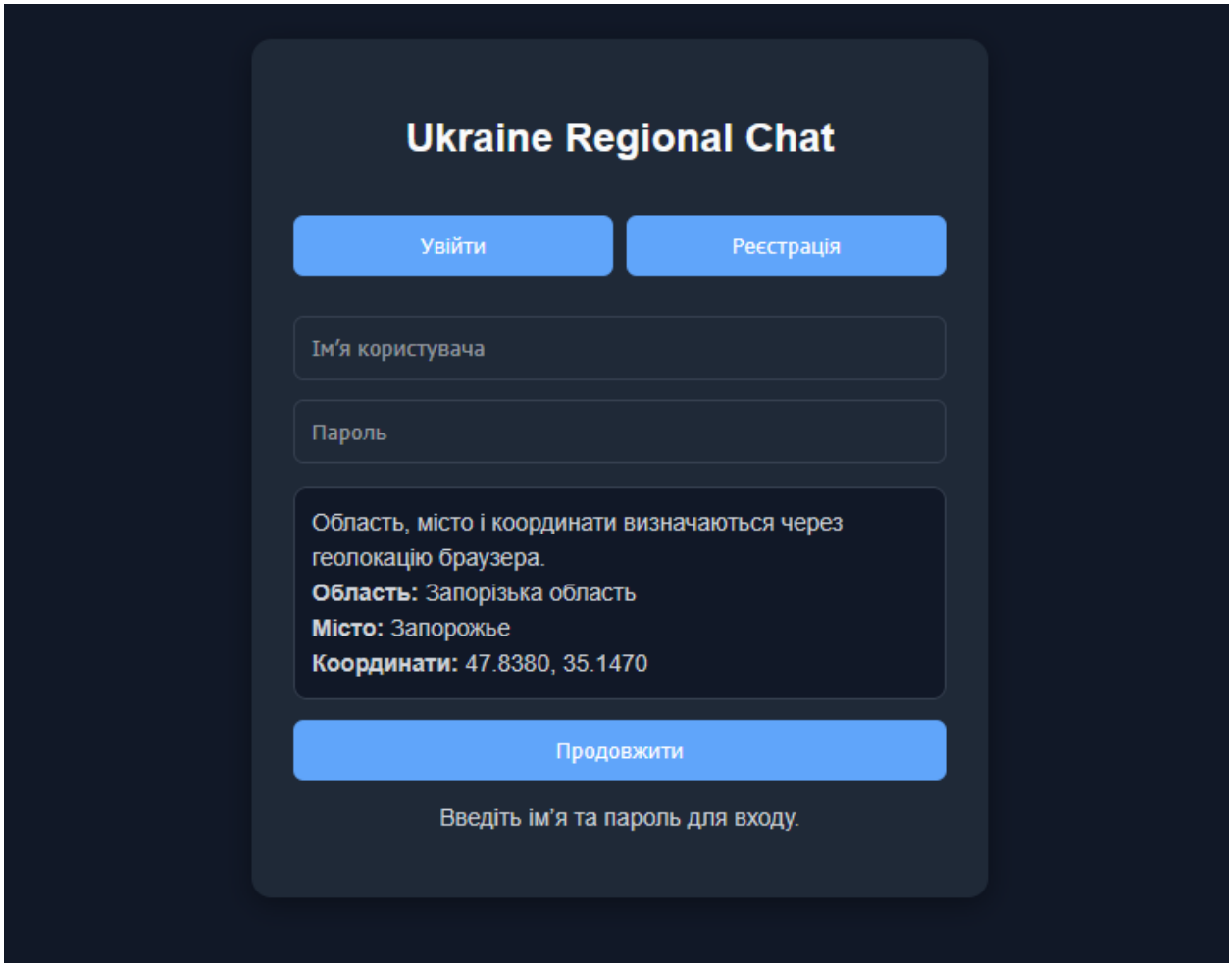

Рисунок 4.1 – Вікно авторизації програмного забезпечення

Після того як користувач успішно авторизувався, або зареєструвався в програмі, його вітає головне вікно чату (рис.4.2), де одразу людина може вибраи цікаву для себе структуру, де йому краще спілкуватись

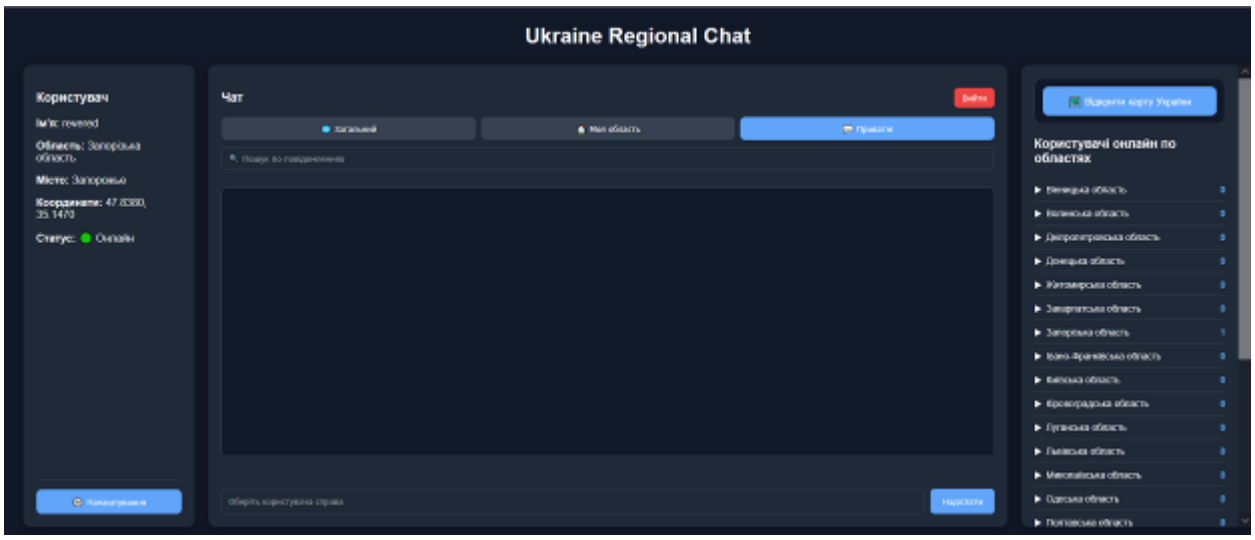

Рисунок 4.2 – Головне вікно програмного забезпечення

При натисканні на блок «Відкрити карту України» (рис.4.3) користувачу автоматично виведеться на екран інтеркативна мапа, з приблизними координатами кожної людини в чаті

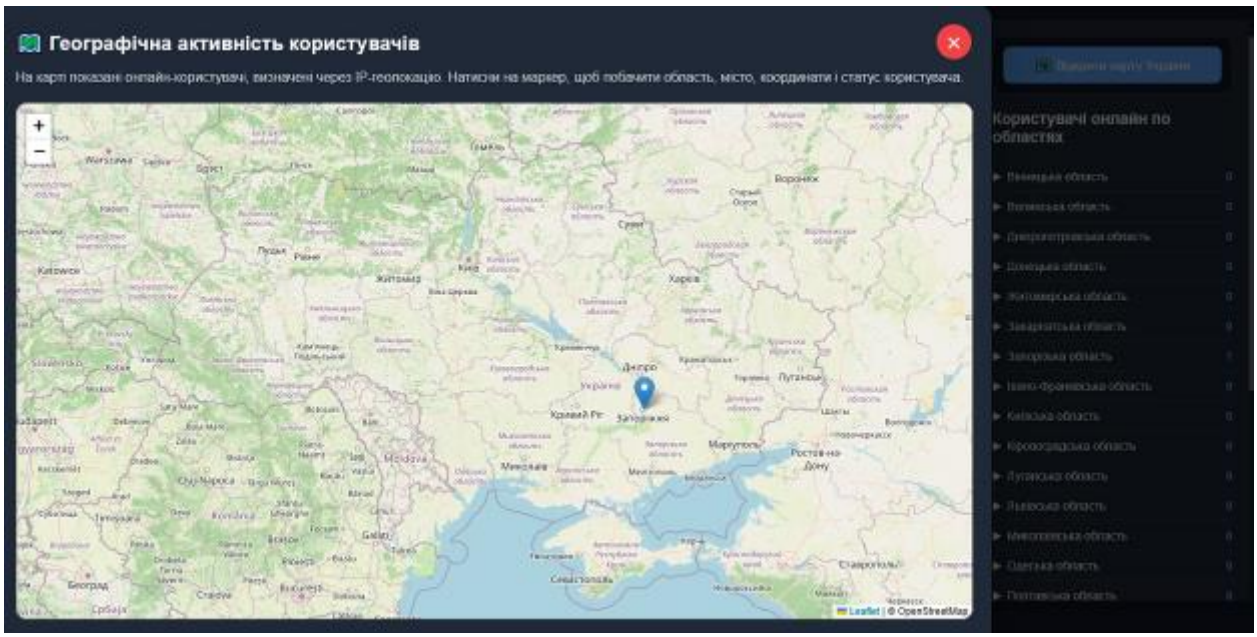

Рисунок 4.3 – Географічна карта активності користувачів

У разі, якщо людині набридла звичайна тема чату, вікн може її перемкнути у налаштуваннях (рис.4.4), там же людина може вимкнути звук

повідомлень й поставити статус активності, та дізнатись актуальні налаштування.

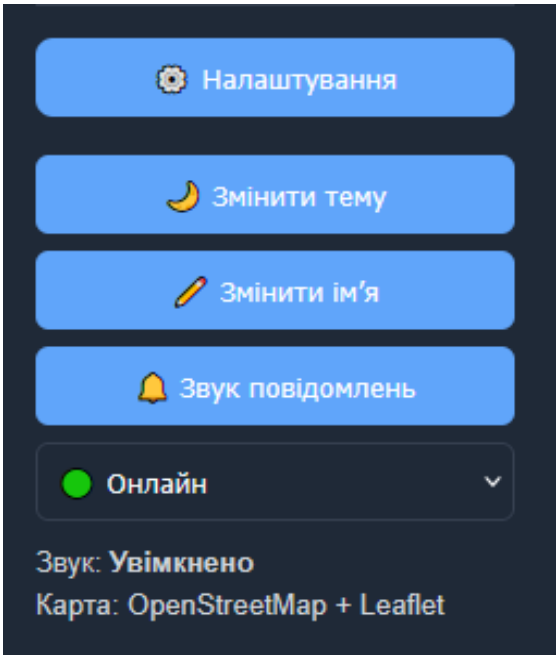

Рисунок 4.4 – Вікно налаштувань програмного забезпечення

Користувача завжди супроводжує в лівому куті інформація стосовно його акаунту (рис.4.5), область, місто та приблизні координати його

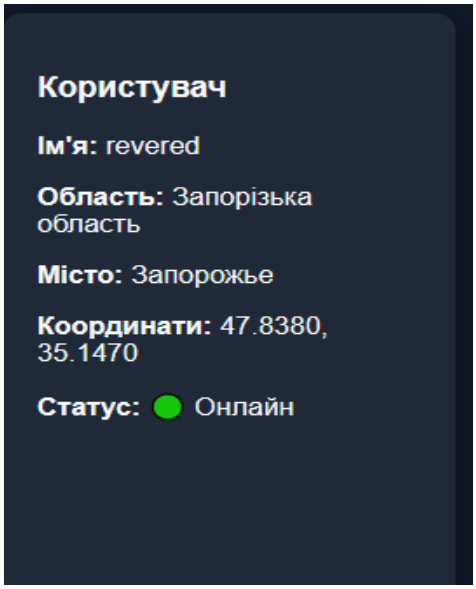

Рисунок 4.5 – Інформація про користувача у програмі

Людина може завжди обрати, де йому краще спілкуватись, в якому з чатів обравши відповідний варіант прямо над головним вікном чату (рис.4.6)

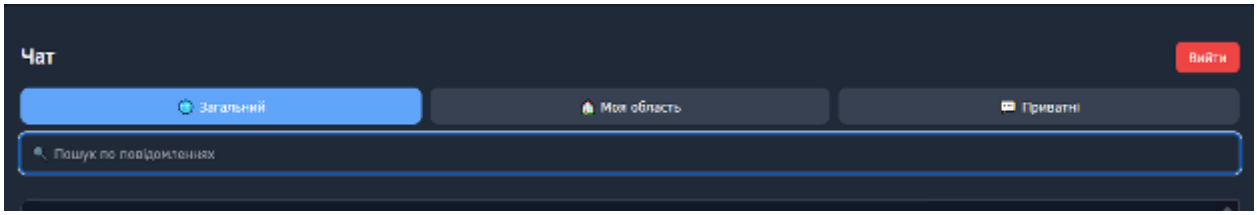

Рисунок 4.6 – Вибір чатів для спілкування у вебзастосунку

4.2 Інструкція по експлуатації програми

4.2.1 Запуск вебзастосунку

Для початку роботи із вебзастосунком Ukraine Regional Chat необхідно запустити серверну частину програмного забезпечення. Під час локального використання запуск здійснюється через середовище розробки Eclipse або IntelliJ IDEA шляхом запуску основного Spring Boot класу проєкту. Після успішної ініціалізації сервера система стає доступною через веб-браузер за локальною адресою http://localhost:8082.

Користувачеві достатньо перейти за вебпосиланням сервісу через будь-який сучасний браузер. Практика тестування показала, що система коректно працює у Google Chrome, Mozilla Firefox та Microsoft Edge без необхідності встановлення додаткового програмного забезпечення.

4.2.2 Підключення користувача до чату

Після відкриття вебзастосунку користувачеві відображається головне вікно підключення до системи. На цьому етапі необхідно ввести ім’я користувача, яке буде використовуватися для ідентифікації в чаті.

Після входу система автоматично визначає приблизний регіон користувача. Це дозволяє отримати доступ до відповідної регіональної

кімнати спілкування без ручного налаштування параметрів. У ході практичного використання встановлено, що автоматизація цього процесу значно спрощує початок роботи із системою та зменшує кількість дій, необхідних для підключення.

Після завершення авторизації користувач автоматично переходить до основного інтерфейсу чату.

4.2.3 Робота користувача із чат-кімнатами

У вебзастосунку реалізовано декілька типів кімнат для спілкування. Користувач отримує доступ до загального чату, де можуть спілкуватися всі учасники системи, а також до регіонального чату, який формується відповідно до області користувача.

Додатково підтримується приватна взаємодія між окремими учасниками. Використання такого підходу дозволяє уникнути перевантаження повідомленнями та зробити процес спілкування більш структурованим. Під час тестування було помітно, що користувачам простіше орієнтуватися в інформаційному просторі, коли повідомлення розподілені за окремими категоріями.

4.2.4 Використання геолокації та карти

Однією з особливостей вебзастосунку є автоматичне визначення приблизного місцезнаходження користувача. Система використовує механізми геолокації браузера та IP-визначення для встановлення регіональної належності користувача.

Для відображення географічної інформації використано інтерактивну карту Leaflet у поєднанні із сервісом OpenStreetMap. Карта дозволяє візуалізувати місцезнаходження користувача та зробити взаємодію із системою більш зрозумілою. Під час експлуатації було встановлено, що

візуальна складова позитивно впливає на сприйняття функціоналу застосунку.

4.3 Перевірка працездатності вебзастосунку

Для оцінки стабільності роботи системи було виконано перевірку основних функціональних можливостей вебзастосунку. Проведено тестування надсилання повідомлень, підключення декількох користувачів, визначення регіону та роботи інтерактивної карти.

Результати перевірки показали стабільну роботу системи як у локальному середовищі, так і після розгортання на платформі Render. Значних затримок під час передачі повідомлень або критичних помилок у процесі використання виявлено не було.

4.4 Висновки за розділом 4

У межах четвертого розділу розглянуто особливості практичного використання вебзастосунку та порядок його експлуатації. Виконано опис запуску системи, процесу підключення користувача, основних можливостей платформи та взаємодії із функціональними елементами застосунку. Практика використання показала, що реалізований механізм роботи не потребує складного налаштування, що спрощує початок роботи із системою навіть для нових користувачів.У процесі перевірки працездатності вебзастосунку протестовано основні функції системи, серед яких підключення користувачів, обмін повідомленнями, робота інтерактивної карти та функціонування чатів.

Результати експлуатації показали стабільну роботу вебзастосунку як у локальному середовищі, так і після розгортання на хмарній платформі Render. Отримані результати дозволяють вважати розроблений застосунок

придатним для організації регіональної онлайн-комунікації між користувачами.

ВИСНОВКИ

У межах виконання дипломної роботи було досліджено особливості проєктування та розробки сучасних веборієнтованих систем комунікації, а також реалізовано власний програмний продукт у вигляді вебзастосунку Ukraine Regional Chat. Актуальність обраної теми пояснюється постійним зростанням потреби у швидкому та доступному обміні інформацією між користувачами, особливо в умовах активного розвитку цифрового середовища. Значна кількість сучасних сервісів комунікації орієнтована на глобальну взаємодію, однак регіональна складова часто залишається недостатньо реалізованою. Саме це стало підґрунтям для створення системи, у якій користувачі можуть взаємодіяти не лише в межах загального інформаційного простору, а й відповідно до своєї області перебування.

При реалізації було покладено в основу аналіз теоретичних аспектів створення вебзастосунків для онлайн-комунікації. Розглянуто принципи клієнт-серверної архітектури, особливості функціонування вебтехнологій та сучасні механізми передачі даних у режимі реального часу. Проведене дослідження показало, що традиційний підхід до оновлення інформації через HTTP-запити не завжди є ефективним для чат-платформ, оскільки створює затримки під час відображення повідомлень та збільшує навантаження на серверну частину. У зв’язку з цим доцільним стало використання технології WebSocket, яка підтримує постійне з’єднання між клієнтом і сервером.

У межах виконання роботи проаналізовано програмні інструменти, що використовуються для побудови веборієнтованих рішень. Для серверної частини обрано Spring Boot, завдяки якому вдалося організувати обробку повідомлень, керування користувачами та взаємодію між окремими компонентами програмного продукту. Застосування цього фреймворку виявилося виправданим через його модульну структуру, підтримку мережевих протоколів та перспективу подальшого розширення функціоналу. Клієнтську частину створено за допомогою HTML, CSS та JavaScript, що

сприяло формуванню інтерфейсу, орієнтованого на швидку та зручну взаємодію користувача із вебсистемою.

Одним із ключових результатів роботи стало створення вебзастосунку для спілкування у режимі реального часу. Впроваджено механізм миттєвої передачі повідомлень, який забезпечує оперативне оновлення інформації без необхідності перезавантаження сторінки. Практичне використання показало, що така модель взаємодії суттєво покращує зручність користування, адже повідомлення надходять із мінімальною затримкою, а процес комунікації сприймається більш природним.

Вагоме значення у роботі мало питання регіональної взаємодії користувачів. У вебсистемі впроваджено автоматичне визначення області користувача, що створює можливість переходу до відповідного чату без потреби ручного налаштування параметрів.

У ході тестування було встановлено, що подібний механізм значно спрощує початок роботи із системою та зменшує кількість дій, необхідних для входу до локального середовища спілкування. Разом із тим виявлено, що точність визначення місцезнаходження може залежати від інтернет-провайдера, VPN-з’єднання або особливостей IP-адресації, що потребує врахування під час подальшого вдосконалення системи.

Додатковою особливістю вебзастосунку стала інтеграція інтерактивної карти на основі Leaflet та OpenStreetMap. Використання картографічних інструментів дозволило зробити регіональну складову більш наочною та зрозумілою для користувача. Практичне тестування показало, що візуальне представлення місцезнаходження позитивно впливає на сприйняття функціоналу застосунку та підвищує загальну зручність користування системою.

За час реалізації практичної частини роботи було отримано низку функціональних можливостей, які підвищують ефективність використання системи. Серед них варто виділити підтримку загального чату, регіональних кімнат спілкування та приватної взаємодії між користувачами. Також

реалізовано адаптивний дизайн, підтримку часових позначок повідомлень, зміну імені користувача, темний режим оформлення та інші допоміжні функції. Наявність таких елементів зробила систему більш сучасною та наближеною до реальних умов використання цифрових платформ комунікації.

У процесі виконання дипломної роботи проведено тестування працездатності програмного забезпечення. Перевірено коректність роботи механізму підключення користувачів, передачі повідомлень, функціонування WebSocket-з'єднання, інтеграції карти та роботи системи після хмарного розгортання на платформі Render. Результати тестування підтвердили стабільність функціонування застосунку як у локальному середовищі, так і при віддаленому доступі через мережу Інтернет. Критичних збоїв у роботі системи виявлено не було, а час доставки повідомлень залишався мінімальним навіть при одночасній взаємодії декількох користувачів.

В процесі побудови роботи стало зрозуміло, що розробка систем онлайн-комунікації потребує комплексного підходу, який поєднує програмну логіку, мережеві технології та принципи зручності користування. Навіть незначні елементи інтерфейсу, наприклад миттєве оновлення повідомлень або візуальна організація чатів, можуть суттєво впливати на загальне сприйняття застосунку користувачами.

Результати виконаної дипломної роботи дозволяють стверджувати, що поставлену мету досягнуто, а визначені завдання виконано у повному обсязі. Розроблений вебзастосунок «Ukraine Regional Chat» успішно реалізує механізми регіональної онлайн-комунікації та може використовуватися як основа для подальшого вдосконалення. Перспективами розвитку системи можуть стати впровадження повноцінної реєстрації користувачів, шифрування повідомлень, push-сповіщень, мобільної версії застосунку та розширення можливостей персоналізації інтерфейсу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Computer-Mediated Communication. URL: https://www.britannica.com/topic/computer-mediated-communication (дата звернення: 02.03.2026).

2. Instant Messaging. URL: https://www.techtarget.com/searchunifiedcommunications/definition/instant-messaging (дата звернення: 04.03.2026).

3. Instant Messaging | ScienceDirect Topics. URL: https://www.sciencedirect.com/topics/computer-science/instant-messaging (дата звернення: 06.03.2026).

4. Spring Boot Documentation. URL: https://spring.io/projects/spring-boot (дата звернення: 09.03.2026).

5. Spring Framework Documentation. URL: https://spring.io/projects/spring-framework (дата звернення: 11.03.2026).

6. Introduction to STOMP Protocol. URL: https://stomp.github.io/ (дата звернення: 13.03.2026).

7. WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket (дата звернення: 16.03.2026).

8. HTML Tutorial. URL: https://www.w3schools.com/html/ (дата звернення: 18.03.2026).

9. CSS Tutorial. URL: https://www.w3schools.com/css/ (дата звернення: 20.03.2026).

10. Leaflet Documentation. URL: https://leafletjs.com/ (дата звернення: 23.03.2026).

11. OpenStreetMap. URL: https://www.openstreetmap.org/ (дата звернення: 25.03.2026).

12. Eclipse IDE Documentation. URL: https://help.eclipse.org/ (дата звернення: 27.03.2026).

13. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 30.03.2026).

14. Render Documentation. URL: https://render.com/docs (дата звернення: 02.04.2026).

15. Java Tutorial. URL: https://www.w3schools.com/java/ (дата звернення: 06.04.2026).

16. Client-Server Architecture. URL: https://www.techtarget.com/searchnetworking/definition/client-server (дата звернення: 09.04.2026).

17. Spring Security Architecture. URL: https://spring.io/projects/spring-security (дата звернення: 13.04.2026).

18. JavaScript Guide. URL: https://javascript.info/ (дата звернення: 16.04.2026).

19. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення: 20.04.2026).

20. Responsive Web Design Basics. URL: https://web.dev/responsive-web-design-basics/ (дата звернення: 23.04.2026).

ДОДАТОК А

Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу онлайн-спілкування між користувачами у режимі реального часу. Вебзастосунок призначений для забезпечення швидкого обміну текстовими повідомленнями між користувачами, а також організації регіональної взаємодії через тематичні чат-кімнати.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Створення програмної вебсистеми обміну повідомленнями з відображенням активності та географічних координат користувачів у реальному часі», затверджене наказом Національного університету «Запорізька політехніка» №139 від 07 квітня 2026 року.

А.2 Призначення розробки

Програмний продукт призначений для підтримки процесу онлайн-комунікації між користувачами через мережу Інтернет.

Вебзастосунок забезпечує можливість швидкого обміну повідомленнями, автоматичного визначення регіональної належності користувача, використання тематичних чат-кімнат та взаємодії між учасниками системи у реальному часі.

А.3 Основні вимоги до програми що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення вебзастосунку повинно забезпечувати виконання таких функцій:

- підтримка можливості швидкого спілкування між користувачами за допомогою текстових повідомлень;

- автоматичне визначення регіону користувача для організації локальної взаємодії;

- підтримка загального чату для спілкування між усіма користувачами системи;

- підтримка регіональних чатів відповідно до області користувача;

- підтримка приватної взаємодії між користувачами;

- підтримка інтерактивної карти для відображення регіональної інформації;

- можливість зміни імені користувача;

- підтримка часових позначок повідомлень;

- забезпечення роботи системи без необхідності оновлення сторінки браузера.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс вебзастосунку повинен бути зрозумілим та зручним для користувачів. Повинна бути забезпечена можливість швидкого переходу між чат-кімнатами, перегляду повідомлень, використання карти та взаємодії із функціональними елементами системи.

Інтерфейс повинен підтримувати роботу у візуальному режимі та коректно відображатися на різних пристроях, включаючи персональні комп’ютери, ноутбуки та мобільні пристрої.

А.3.3 Вимоги до надійності

Програмне забезпечення вебзастосунку повинно забезпечувати стабільне функціонування та підтримку безперервного обміну повідомленнями між користувачами.

У випадку втрати мережевого підключення система повинна повідомляти користувача про помилку з’єднання та забезпечувати можливість повторного підключення.

Система повинна забезпечувати коректну роботу при одночасному використанні декількома користувачами.

А.3.4 Вимоги до експлуатації

Для експлуатації програмного забезпечення необхідна наявність персонального комп’ютера або мобільного пристрою із сучасним веббраузером та доступом до мережі Інтернет.

Для коректної роботи системи рекомендується використання браузерів Google Chrome, Mozilla Firefox або Microsoft Edge із підтримкою JavaScript та WebSocket-з’єднання.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для роботи вебзастосунку необхідні технічні та програмні ресурси, які забезпечують обробку повідомлень у режимі реального часу.

Серверна частина повинна підтримувати роботу Java 17, Spring Boot, WebSocket, STOMP-протоколу та системи управління базами даних H2 або PostgreSQL.

Клієнтська частина повинна підтримувати відображення HTML-інтерфейсу, виконання JavaScript-коду, обробку WebSocket-з’єднання та відображення інтерактивної карти на основі Leaflet і OpenStreetMap.

ДОДАТОК Б

Фрагмент тексту програми

```
function createRegionSelect() {

   let select =
      document.getElementById("regionSelect");

   select.innerHTML = "";

   regions.forEach(region => {

      let option =
         document.createElement("option");

      option.value = region;
      option.innerText = region;

      select.appendChild(option);
   });
}

function getRegionCenter(region) {
   return regionCenters[region] || [0, 0];
}
function updateUserInfoOnPage() {

   document.getElementById("currentRegion").innerText =
      currentRegion;

   document.getElementById("currentCity").innerText =
      currentCity;

   document.getElementById("currentCoords").innerText =
      Number(currentLatitude).toFixed(4)
      + ", "
      + Number(currentLongitude).toFixed(4);
}
function switchAuthMode(mode) {
   authMode = mode;

   document.getElementById("loginTab").classList.remove("active");
   document.getElementById("registerTab").classList.remove("active");

   if (mode === "login") {
      document.getElementById("loginTab").classList.add("active");
      document.getElementById("authMessage").innerText =
         "Введіть ім’я та пароль для входу.";
   } else {
      document.getElementById("registerTab").classList.add("active");
      document.getElementById("authMessage").innerText =
         "Створіть новий акаунт.";
   }
}
```

```
function authSubmit() {
    let username = document.getElementById("loginUsername").value.trim();
    let password = document.getElementById("loginPassword").value.trim();

    if (username === "") {
        document.getElementById("authMessage").innerText =
            "Введіть ім’я користувача.";
        return;
    }

    if (password.length < 4) {
        document.getElementById("authMessage").innerText =
            "Пароль має бути мінімум 4 символи.";
        return;
    }

    let url = authMode === "login"
        ? "/auth/login"
        : "/auth/register";

    fetch(url, {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify({
            username: username,
            password: password,
            region: currentRegion,
            city: currentCity,
            latitude: currentLatitude,
            longitude: currentLongitude
        })
    })
        .then(response => response.json())
        .then(data => {
            if (!data.success) {
                document.getElementById("authMessage").innerText =
                    data.message;
                return;
            }

            currentUsername = data.username;
            currentRegion = normalizeRegion(data.region || currentRegion);
            currentCity = data.city || currentCity;
            currentLatitude = data.latitude || currentLatitude;
            currentLongitude = data.longitude || currentLongitude;

            loginAfterAuth();
        })
        .catch(() => {
```

```
            document.getElementById("authMessage").innerText =
                "Помилка підключення до сервера.";
        });
}

function loginAfterAuth() {
    document.getElementById("currentUsername").innerText =
        currentUsername;

    document.getElementById("currentStatus").innerText =
        getStatusText(currentStatus);

    updateUserInfoOnPage();

    document.getElementById("loginPage").style.display =
        "none";

    document.getElementById("chatPage").style.display =
        "block";

    if (!connected) {
        connect(sendJoinMessage);
    } else {
        sendJoinMessage();
    }
}

function normalizeRegion(region) {

    if (!region) return "Невідома область";

    region = region.toLowerCase();

    if (region.includes("він") || region.includes("vinn")) {
        return "Вінницька область";
    }

    if (region.includes("вол")) {
        return "Волинська область";
    }

    if (region.includes("дніпр") || region.includes("днепр") || region.includes("dnip")) {
        return "Дніпропетровська область";
    }

    if (region.includes("дон")) {
        return "Донецька область";
    }

    if (region.includes("жит")) {
        return "Житомирська область";
    }
```

```
            if (region.includes("закар") || region.includes("uzh")) {
                return "Закарпатська область";
            }

            if (region.includes("запор") || region.includes("zap")) {
                return "Запорізька область";
            }

            if (region.includes("івано") || region.includes("frank")) {
                return "Івано-Франківська область";
            }

            if (region.includes("київ") || region.includes("киев") || region.includes("kyiv") ||
region.includes("kiev")) {
                return "Київська область";
            }

            if (region.includes("кіров") || region.includes("kropyv")) {
                return "Кіровоградська область";
            }

            if (region.includes("луг") || region.includes("luh")) {
                return "Луганська область";
            }

            if (region.includes("льв") || region.includes("lviv")) {
                return "Львівська область";
            }

            if (region.includes("микол") || region.includes("mykol")) {
                return "Миколаївська область";
            }

            if (region.includes("одес") || region.includes("odes")) {
                return "Одеська область";
            }

            if (region.includes("полтав")) {
                return "Полтавська область";
            }

            if (region.includes("рівн") || region.includes("rivne")) {
                return "Рівненська область";
            }

            if (region.includes("сум")) {
                return "Сумська область";
            }

            if (region.includes("терноп")) {
                return "Тернопільська область";
```

```
    }

    if (region.includes("хар") || region.includes("khark")) {
        return "Харківська область";
    }

    if (region.includes("херс")) {
        return "Херсонська область";
    }

    if (region.includes("хмель")) {
        return "Хмельницька область";
    }

    if (region.includes("черкас")) {
        return "Черкаська область";
    }

    if (region.includes("чернів")) {
        return "Чернівецька область";
    }

    if (region.includes("черніг")) {
        return "Чернігівська область";
    }

    return "Невідома область";
}

function detectRegion() {

    if (navigator.geolocation) {

        navigator.geolocation.getCurrentPosition(

            function(position) {

                currentLatitude = position.coords.latitude;
                currentLongitude = position.coords.longitude;

                fetch(
                    "https://nominatim.openstreetmap.org/reverse?format=json"
                    + "&lat=" + currentLatitude
                    + "&lon=" + currentLongitude
                )
                    .then(response => response.json())
                    .then(data => {

                        let address = data.address || {};

                        currentCity =
                            address.city
```

```
                        || address.town
                        || address.village
                        || "Невідоме місто";

                    currentRegion =
                        address.state
                        || address.region
                        || "Невідома область";

                    currentRegion =
                        normalizeRegion(currentRegion);

                    document.getElementById("detectedRegion").innerText =
                        currentRegion;

                    document.getElementById("detectedCity").innerText =
                        currentCity;

                    document.getElementById("detectedCoords").innerText =
                        Number(currentLatitude).toFixed(4)
                        + ", "
                        + Number(currentLongitude).toFixed(4);

                    updateUserInfoOnPage();
                });

            },

            function() {

                detectByIP();

            }

        );

    } else {

        detectByIP();

    }
}

function detectByIP() {

    fetch("/api/geo/region")
        .then(response => response.json())
        .then(data => {

            currentRegion =
                normalizeRegion(data.region);
```

```
            currentCity =
               data.city || "Невідоме місто";

            currentLatitude =
               data.latitude || 0;

            currentLongitude =
               data.longitude || 0;

            document.getElementById("detectedRegion").innerText =
               currentRegion;

            document.getElementById("detectedCity").innerText =
               currentCity;

            document.getElementById("detectedCoords").innerText =
               Number(currentLatitude).toFixed(4)
               + ", "
               + Number(currentLongitude).toFixed(4);

            updateUserInfoOnPage();

         })
         .catch(() => {

            currentRegion = "Невідома область";
            currentCity = "Невідоме місто";

            document.getElementById("detectedRegion").innerText =
               currentRegion;

            document.getElementById("detectedCity").innerText =
               currentCity;
         });
}

function openMapModal() {
   document.getElementById("mapModal").classList.add("show");

   setTimeout(function () {
      initRealMap();
      updateRealMapMarkers(mapStats);

      if (realMap) {
         realMap.invalidateSize();
      }
   }, 100);
}

function closeMapModal() {
   document.getElementById("mapModal").classList.remove("show");
}
```

```
        function initRealMap() {
            if (realMap !== null) return;

            realMap = L.map("realUkraineMap").setView([48.7, 31.2], 6);

            L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
                maxZoom: 18,
                attribution: "&copy; OpenStreetMap"
            }).addTo(realMap);

            realMapMarkersLayer = L.layerGroup().addTo(realMap);
        }

        function updateRealMapMarkers(data) {
            if (realMap === null || realMapMarkersLayer === null) return;

            realMapMarkersLayer.clearLayers();

            let totalUsers = 0;

            regions.forEach(region => {
                let users = data[region] || [];
                totalUsers += users.length;

                users.forEach(user => {
                    let lat = user.latitude || getRegionCenter(region)[0];
                    let lon = user.longitude || getRegionCenter(region)[1];

                    let marker = L.marker([lat, lon]);

                    marker.bindPopup(`
                        <b>${escapeHtml(user.username)}</b><br>
                        Статус: ${getStatusText(user.status)}<br>
                        Область: ${escapeHtml(region)}<br>
                        Місто: ${escapeHtml(user.city || "Невідоме місто")}<br>
                        Координати: ${Number(lat).toFixed(4)}, ${Number(lon).toFixed(4)}<br>
                        <button onclick="startPrivateMessage('${escapeJs(user.username)}');
closeMapModal();">
                            Написати приватно
                        </button>
                    `);

                    marker.addTo(realMapMarkersLayer);
                });
            });

            if (totalUsers === 0) {
                L.marker([currentLatitude, currentLongitude])
                    .bindPopup("Поки що на карті немає інших онлайн-користувачів.")
                    .addTo(realMapMarkersLayer);
            }
```

```
}

function escapeHtml(text) {
    if (!text) return "";

    return String(text)
        .replaceAll("&", "&amp;")
        .replaceAll("<", "&lt;")
        .replaceAll(">", "&gt;")
        .replaceAll("", "&quot;")
        .replaceAll("", "&#039;");
}

function escapeJs(text) {
    if (!text) return "";
    return String(text).replaceAll("", "\\");
}
```

ДОДАТОК В

Слайди презентації

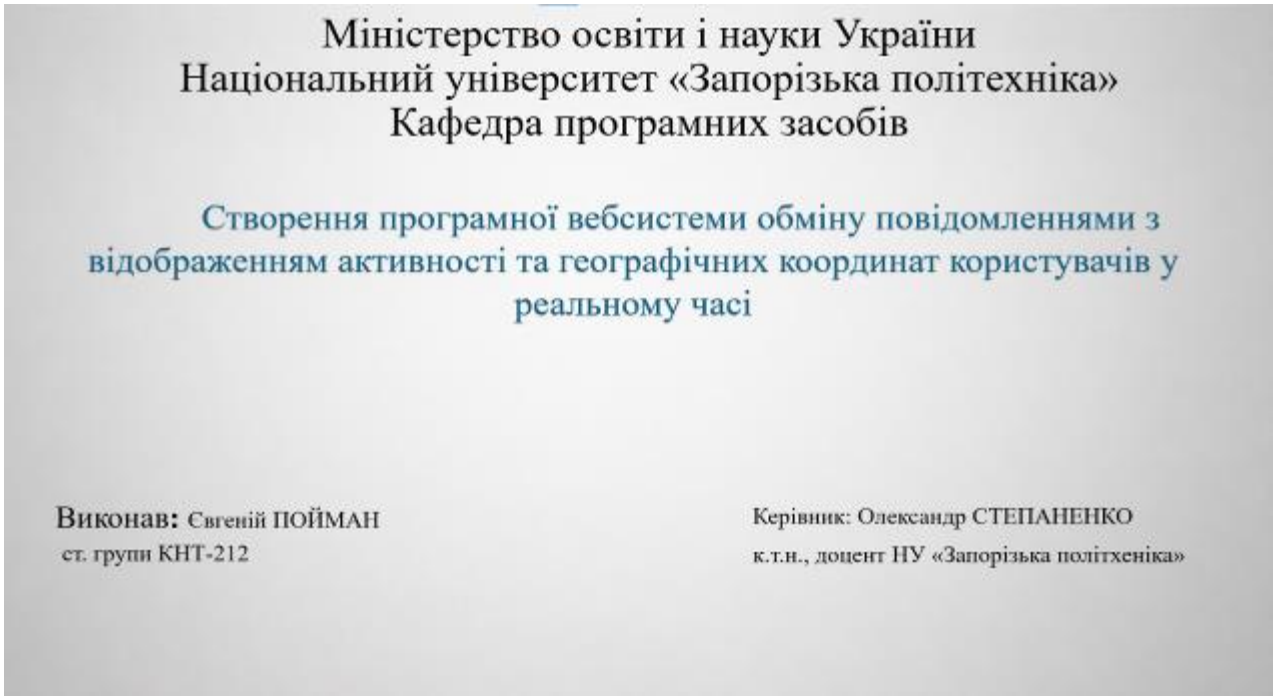

Рисунок В.1 – Слайд 1

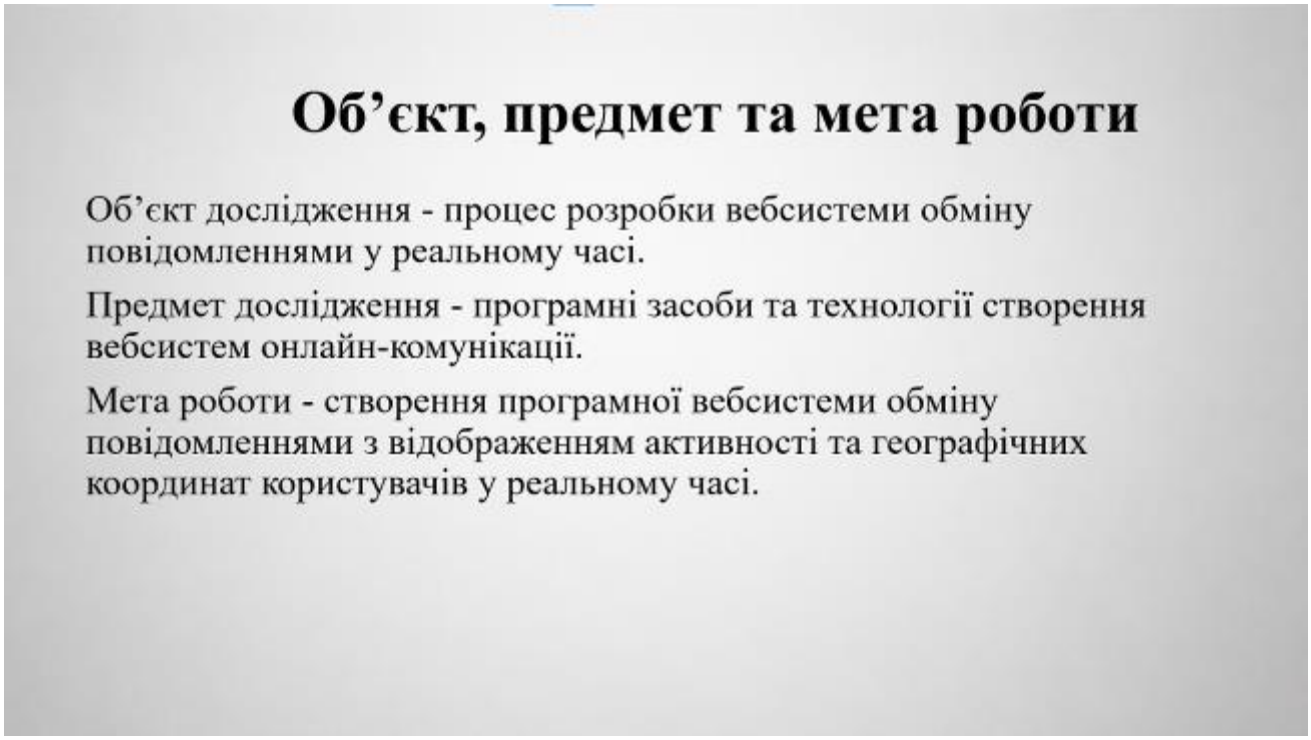

Рисунок В.2 – Слайд 2

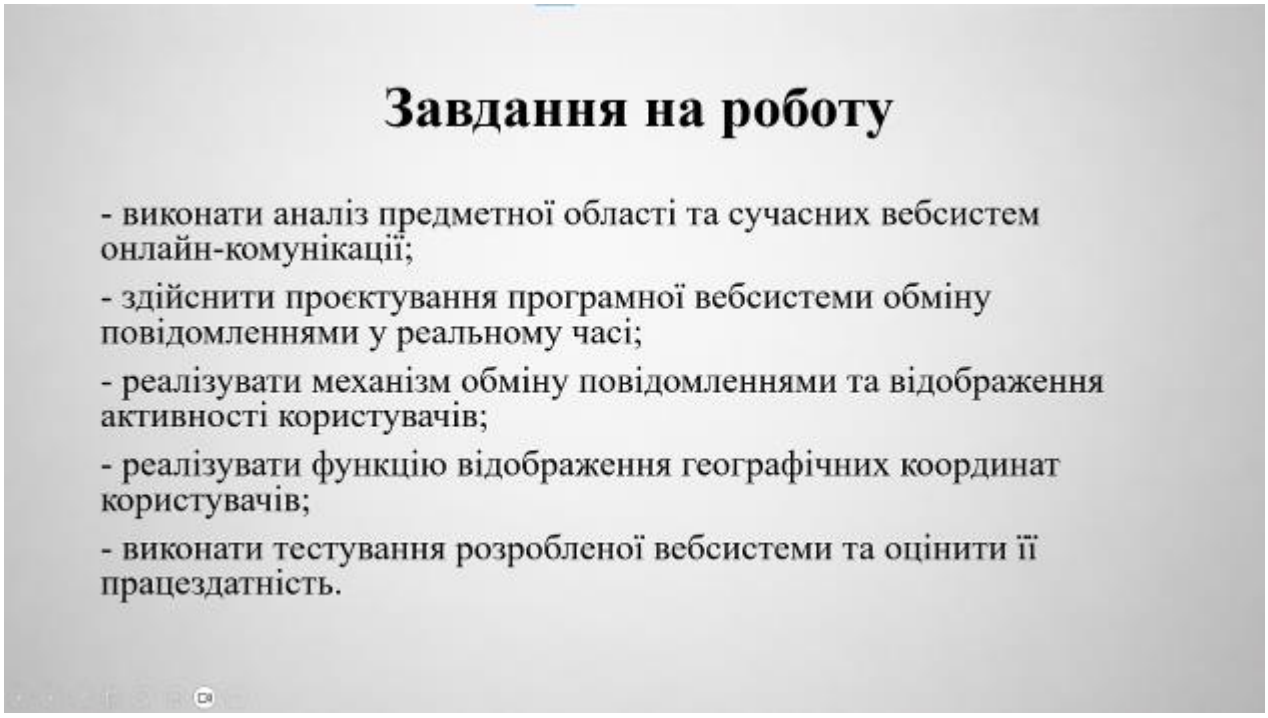

Рисунок В.3 – Слайд 3

Рисунок В.4 – Слайд 4

Рисунок В.5 – Слайд 5

Критерій порівняння	Java	C#	Python
Підтримка веброзробки	+	+	+
Кросплатформеність	+	+-	+
Підтримка WebSocket	+	+	+
Простота вивчення	+-	+	+-
Підтримка серверної архітектури	+	+	+
Робота з великим навантаженням	+	+	+-
Використання у корпоратвних системах	+	+	-
Споживання ресурсів сервера	+-	-	+
Зручність тестування пз	+	+-	+
Підтримка хмарних технологій	+	+	+
Доцільність до систем спілкувани	+	+-	-

Рисунок В.6 – Слайд 6

Рисунок В.7 – Слайд 7

Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

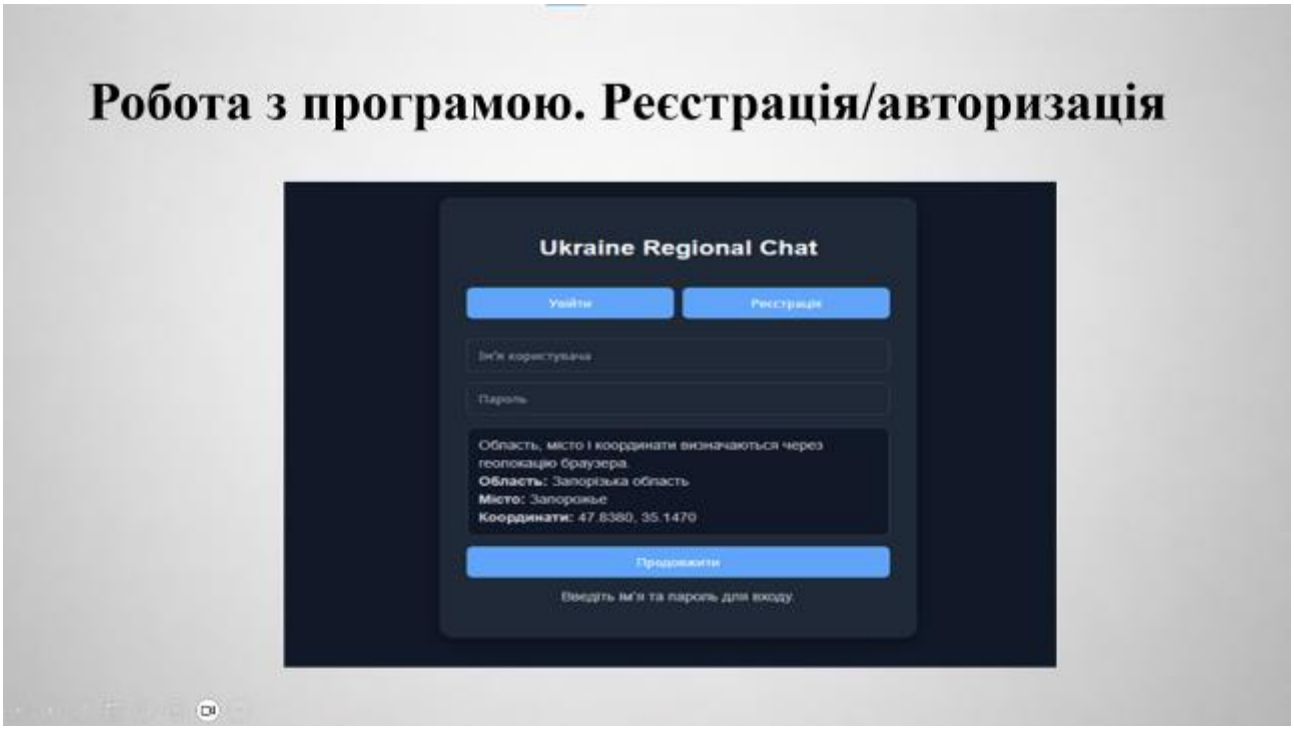

Рисунок В.10 – Слайд 10

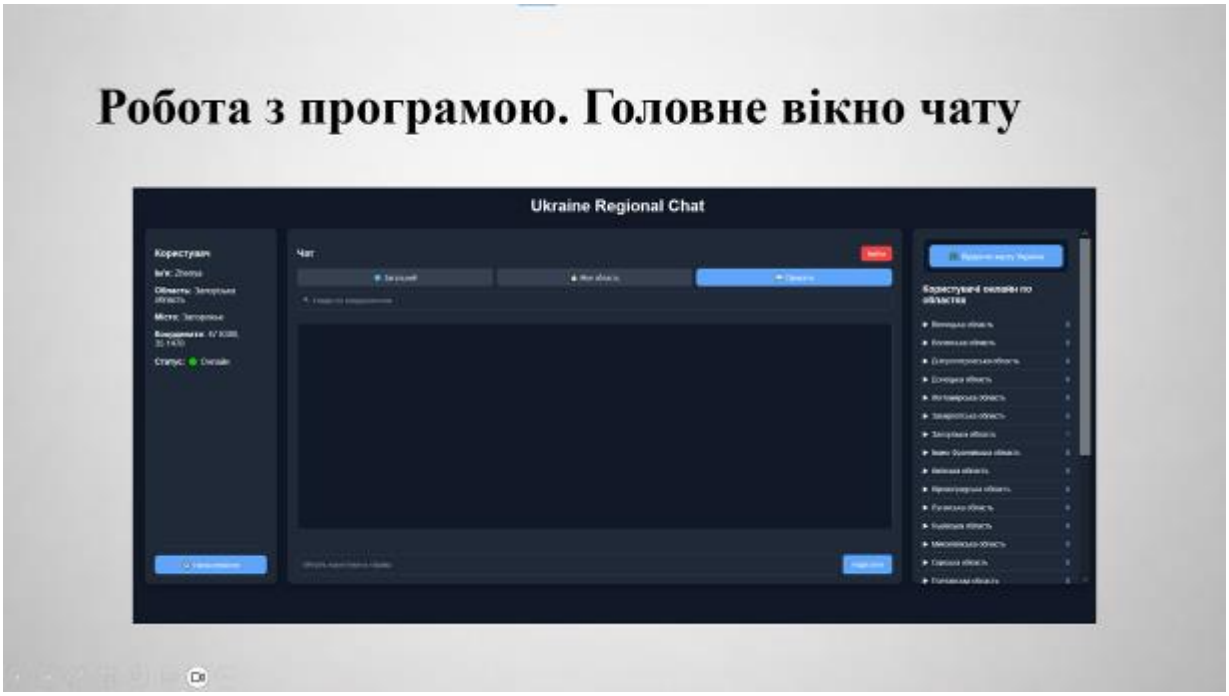

Рисунок В.11 – Слайд 11

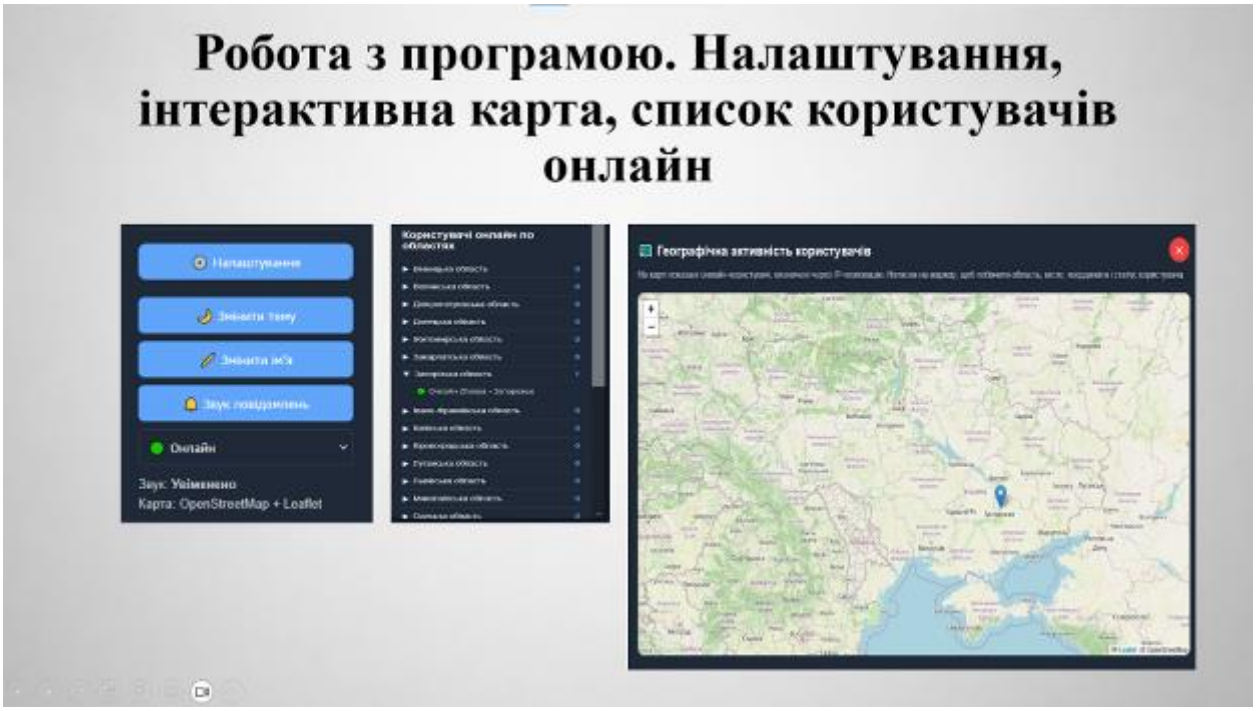

Рисунок В.12 – Слайд 12

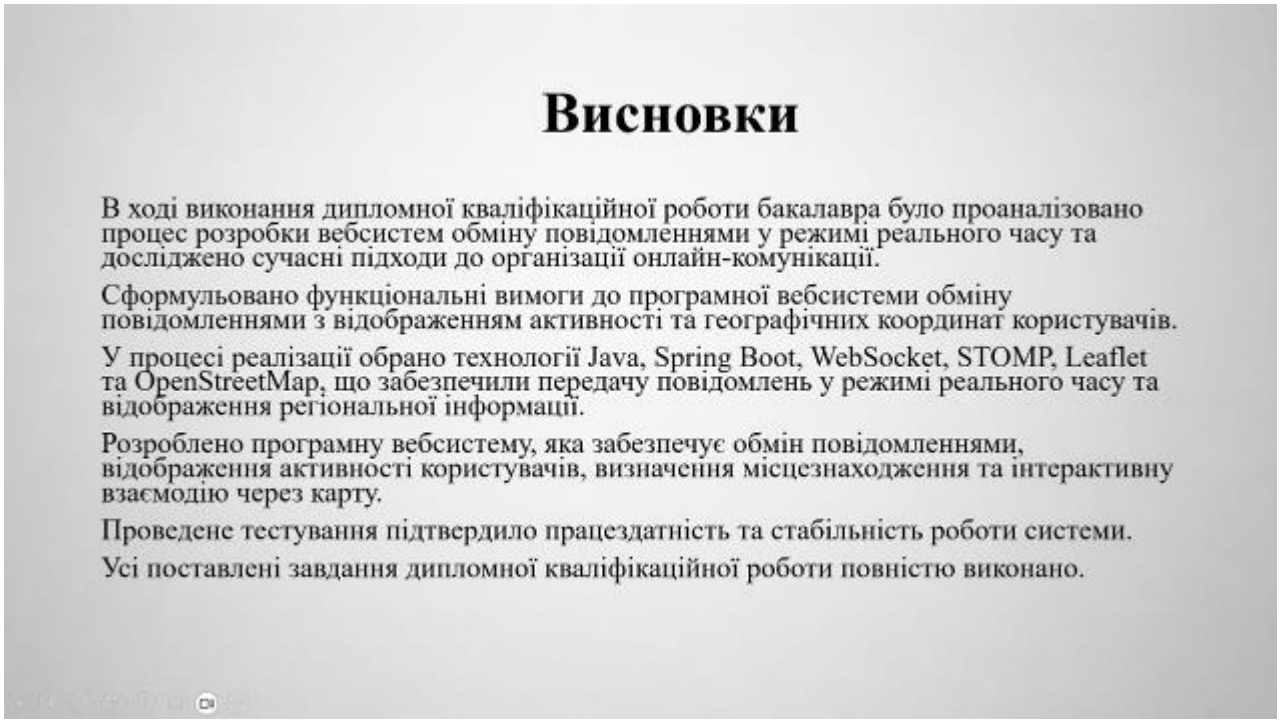

Рисунок В.13 – Слайд 13