Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Швидкодіючий веб-чат з можливістю універсального розгортування»

Виконав: студент 2 курсу, групи 2КІ-20м
напряму підготовки (спеціальності)
123 — «Комп'ютерна інженерія»
_____ Черненко Д. Ю.

Керівник: проф., д.т.н. ОТ

_____Азаров Д.Ю.

«____» _____ 2021 р.

Опонент: д. т. н., професор, голова секції каф МБІС

_Яремчук Ю.Є.__ (ПІБ рецензента)

«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«____» _____ 2021 р.

Вінниця ВНТУ - 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 — Інформаційні технології
Спеціальність 123 — «Комп'ютерна інженерія»
Освітня програма — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О.Д. Азаров

«___»_____2021 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Черненко Дмитру Юрійовичу

1 Тема роботи «Вебчат підвищенної швидкодії з можливістю універсального розгортування»

керівник роботи Азаров Олексій Дмитрович, д.т.н., професор,

затверджені наказом вищого навчального закладу від 24.12.2021 р. №227

2 Строк подання студентом роботи 15.12.2021 р.

3 Вихідні дані до роботи — засоби програмного забезпечення для розробки сторінок веб-додатку, їх зовнішнього виду та бази даних MongoDB.

4 Засоби — середовище програмування Microsoft Visual Studio Code, мова програмування JavaSript, фреймворк Vue.js, фреймворк Node.js, мова гіпертекстової розмітки HTML, каскадна таблиця стилів CSS, база даних MongoDB інструментарій Docker.

5 Зміст розрахунково-пояснювальної записки: вступ, аналіз технологій які підходять для підвищення швидкодії, розробка клієнтської та серверної частини, результат роботи веб-додатку, висновки, список використаних джерел, додатки.

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3	Черняк О. І., к. т. н., доцент		
4	Яремчук Ю.Є. проф., к.е.н.		

7 Дата видачі завдання 07.09.2021 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2 — Календарний план

№	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	11.07.2021	виконано
2	Аналіз існуючих аналогів	15.07.2021	виконано
3	Аналіз та вибір актуальних технологій	17.07.2021	виконано
4	Програмна реалізація	10.08.2021	виконано
5	Підготовка матеріалів пояснювальної записки	17.08.2021	виконано
6	Перевірка якості оформлення МКР	18.09.2021	виконано
7	Оформлення пояснювальної записки і презентації	23.09.2021	виконано
8	Оформлення економічної частини	10.10.2021	виконано
9	Попередній захист	20.11.2021	виконано

Студент _____ Черненко Д.Ю.
Керівник роботи _____ Азаров О. Д.

АНОТАЦІЯ

УДК 004.9

Черненко Д. Ю. Швидкодіючий вебчат з можливістю універсального розгортування. Магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2021, 102 с.

На укр.мові. Бібліогр.: 54 назв, рис. 25, табл. 6.

В магістерській дипломній роботі створено веб чат підвищеної швидкодії з можливістю універсального розгортування за допомогою мови програмування JavaScript, мови гіпертекстової розмітки документів HTML та каскадної таблиці стилів CSS.

Також використовується фреймворк Vue.js, платформа Node.js, база данних MongoDB та протокол WebSocket. Можливість універсального розгортування досягається використанням інструментарію Docker. Веб-додаток було створено у середовищі розробки Visual Studio Code.

В магістерській роботі були також виконані економічні розрахунки по визначенню доцільності розробки нового програмного продукту.

Графічна частина складається з 8 плакатів із результатами еспрементальних досліджень.

Ключові слова: JavaSript, Vue.js, Node.js, мова гіпертекстової розмітки HTML, каскадна таблиця стилів CSS, база даних MongoDB.

ANNOTATION

Chernenko D. High-speed web chat with the possibility of universal deployment. Master's thesis in specialty 123 - computer engineering, educational program - computer engineering. Vinnytsia: VNTU, 2021, 102 p.

In Ukrainian. Bibliogr .: 54 titles, fig. 25, table. 6.

The master's thesis includes a high-speed web chat with the possibility of universal deployment using the JavaScript programming language, hypertext markup language HTML documents and cascading table of CSS styles.

The Vue.js framework, the Node.js platform, the MongoDB database and the WebSocket protocol are also used. The possibility of universal deployment is achieved by using Docker tools. The web application was created in the Visual Studio Code development environment.

In the master's thesis, economic calculations were also performed to determine the feasibility of developing a new software product.

The graphic part consists of 8 posters with the results of experimental research.

Keywords: JavaSript, Vue.js, Node.js, HTML hypertext markup language, cascading CSS style sheet, MongoDB database.

.

ЗМІСТ

ВСТУП

Інтернет набуває досить широкого поширення в майже усіх сферах людської діяльності і не тільки. Тому на даний час існує безліч, дуже багато різноманітних веб-чатів, соціальних мереж та навіть мобільних додатків для обміну повідомленнями та іншою інформацією. Гіганти в цій ніші —Facebook, ВКонтакте, Telegram, Viber, PHP [22], Python [23], C ++ [24], Java [25], HTML [3-4], CSS [1-2] , Qt [26]. Недоліками ж цих даних гігантів можан назвати складність розробки, обслуговування та також ще додаткова складність нових компонентів, так як більшість технологій, по яких вони створені були, вже застаріли. Часто той ж же самий Facebook «падає», коли розробник оновлює його та додає новий певні нові компонент. Зі складними базами даних, непростими запитами та навіть з тими самими застарілими технологіями розробки обмін повідомленнями із веліканами даних також не дуже то ий швидкий.

Метою даного дипломного проекту є розробка високошвидкісного веб-чату з використанням мови програмування JavaScript, HTML-документа [3-4] мовою розмітки гіпертексту та каскадної таблиці стилів CSS [1-2]. Також використовуються фреймворк Vue.js [7-8], база даних MongoDB [16-17] і протокол WebSocket [14-15]. Універсальне розгортання досягається за допомогою інструментів Docker [18-19]. Веб-додаток створено в середовищі розробки Visual Studio Code [27].

Актуальність дослідження пов’язана із необхідністю вирішення ряду задач обміну повідомленнями в мережі Інтернет. Зокрема, для забезпечення зручності користувацького досвіду завдяки швидкій взаємодії із чатом та зручність розгортування додатку в будь-якому середовищі для розробників.

Об’єктом дослідження процес обміну повідомленнями в мережі Інтернет.

Предметом дослідження є веб чат підвищеної швидкодії з можливістю універсального розгортування.

Мета роботи — підвищеня швидкодії чату для покращення досвіду користування та реалізація універсальності розгортування для зручності розміщення проекту на серверному комп’ютері.

Для досягнення даної мети у дипломній роботі потрібно вирішити такі задачі:

— розробити та стилізувати сторінку;

— розробити програмне забезпечення клієнтської та серверної частин, використовуючи вибрані технології;

— налаштувати Docker для контейнеризації застосунку.

Наукова новизна роботи — вперше запропоновано та реалізовано вебчат на основі технологій Vue.js, Node.js, MongoDB, Docker, що дозволило збільшити швидкодію та забезпечити зручне розгортування додатку на серверному комп’ютері.

Практична цінність роботи полягає в розробці програмного продукту, який відрізняється збільшенною швидкодією та універсальністю розгорування. Сферою застосування результатів роботи є мережа Інтернет.

За результатами дослідження опубліковано тези доповіді: Швидкодіючий вебчат з можливістю універсального розгортування [Текст] Д.Ю. Черненко. Молодь в науці: дослідження, проблеми, перспективи (МН-2021) Тез. доп. — Вінниця, 2021. — Режим доступу https://conferences.vntu.edu.ua/index.php/mn/mn2022/author/submission/14289 [30]

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ СТВОРЕННЯ ВЕБЧАТІВ ТА ПОРІВНЯННЯ ЇХ МІЖ СОБОЮ

В данному розділі розглядаються актуальні на сьогодні технології веб-розробки, які також й підійдуть для створення веб-чату, підвищення його реалізації універсальності розгортування швидкодії, реалізації універсальності розгортування, ну й також можливості простої підтримки та доповнення новими компонентами додатку.

Розглянуто технології веб-розробки, серед яких — Vue.js, Node.js, Express.js, MongoDB, WebSocket, Docker, ну й ще також мова програмування JavaScript, так як вона найактуальніша мова програмування для створення веб-додатків на сьогодні.

1.1 Аналіз розробки на мові програмування JavaScript

JavaScript — динамічна об'єктно-орієнтована мова-прототип для програмування. Також ще реалізація стандарту ECMAScript. Вона є найкращим рішенням для створення сценаріїв веб-сторінки, що дозволяють стороні клієнта почати взаємодіяти із користувачем, або ж керувати браузером, асинхронно спілкуватися із сервером чи ж змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікується в якості прототипу, мови програмування сценаріїв динамічного типу. Окрім ж того самого прототипу, JavaScript також ще й частково підтримує інші різноманітні парадигми програмування та ще деякі пов’язані із ними архітектурні особливості — динамічне та слабке введення, автоматизоване керування пам’яттю, успадкування прототипів та й функціональність в якості першокласного об’єкта. JavaScript застосовується для:

— написання сценаріїв веб-сторінок та придавання їм інтерактивності;

— розробки односторінкових веб-застосунків;

— програмування зі сторін серверів (Node.js);

— мобільних застосунків (React Native, Cordova);

— стаціонарних застосунків (Electron, NW.js);

— всередині PDF-документів тощо;

— сценаріїв в прикладному ПЗ тощо.

Незважаючи на подібність імен, Java і таJavaScript є двома зовсім різними мовами зі схожою функціональністю у стандартній бібліотеці та умовах найменування, але, водночас із різною семантикою. Синтаксис обох мов було успадковано від мови C, проте семантика та дизайн JavaScript можна назвати результатом впливу Self і Scheme.

На даний момент JavaScript можна назвати однією із найпопулярніших мов програмування в мережі Інтернеті. Проте, ще на початку багатьох професійних програмістів думка була скептично до мови, їх цільовою аудиторією були лише програмісти-любителі. Поява AJAX змінила ситуацію і привернула увагу мовної спільноти. Подальші зміни до ES2015 і ES2017 представили багато корисних функцій, яким бракує ефективного програмування. В результаті чого було ще також розроблено та покращено багато варіантів використання JavaScript (включаючи тестування та налагодження), створені бібліотеки та фреймворки, а використання JavaScript вийшло за межі браузера.

JavaScript має доволі багато особливостей об’єктно-орієнтованих мов, проте завдяки концепції прототипів підтримка об’єктів JavaScript відрізняється від інших традиційних мов. Окрім того, JavaScript має багато властивостей, які є унікальними до функціональних мов — діють як елементи перших класів, перенесень, об’єктів в якості списків, анонімних функцій.

JavaScript має синтаксис, доволі подібний з C, проте ще є така принципова відмінність в порівнянні зі тією ж C:

— об'єкти, що можуть мати інтроспекції та динамічну зміни типу через механізм прототипів;

— функції в якості об'єктів перших класів;

— автоматичне збирання сміття;

— автоматичні приведення різних типів;

— обробки серед винятків;

— автоматичне збирання сміття;

— анонімні функції.

JavaScript містить ще й певні вбудовані об’єкти, такі як — Global, Object, Error, Function. Окрім того, JavaScript ще також містить набір певних вбудованих операцій, що кажучи, не обов’язкові можуть бути функціями або методами, а також ще й набором вбудованих операторів, які керують логікою виконання програми. Синтаксис JavaScript практично такий же самий, як йсинтаксис Java, однак він ще й спрощений, рівняно зі синтаксисом JavaScript, для полегшення вивчення мови сценаріїв.

1.2 Мова гіпертекстової розмітки HTML

HTML — мова тегів, що застосовується для створення гіпертекстових документів у меержі Інтернет. Веб-браузер приймає HTML-документ із веб-сервера або ж локальної пам'яті й передає документ на мультимедійну веб-сторінку. HTML семантично описує структуру сторінки та спочатку нараховує сигнал щодо появи документу. Елементи HTML, по суті, є компонентами HTML-сторінок. Структури HTML дозволяють добавляти інші елементи — зображення і інтерактивні форми. HTML надає інструменти для створення структурованих документів, які відображають структуровану семантику тексту — посилання, абзаци, заголовки, списки, цитати тощо. Елементи HTML представлені тегами, що укладені в кутові дужки. Такі мітки, як-от прямий доступ до вмісту всієї сторінки. Інші теги, типу як <img /> <input /> <p>, дають інформацію щодо тексту документів та також можуть ще включати й інші теги — піделементи. Браузер не відображатиме теги HTML, проте застосовує їх для інтерпретації всього вмісту сторінки. HTML може також ще й вбудовувати програми, що написані мовами сценаріїв, типу як JavaScript, щоб буде впливати на їх поведінку й вміст веб-сторінок. Ввімкнення CSS визначає зовнішній вигляд та й макет вмісту. HTML впроваджує засоби для:

— отримання інформації з Всесвітньої мережі за допомогою гіперпосилань;

— створення інтерактивних форм;

— включення зображень, відео, звуку тощо.

1.3 Аналіз каскаднної таблиця стилів CSS

CSS — спеціалізовані сторінки стилів, що використовуються в описі зовнішнього вигляду. Сама ж сторінка написана мовою розмітки, тобто HTML.

CSS найчастіше застосовується для візуального відображення сторінок, написаних на HTML і XHTML, проте ж формати CSS також можна ще застосовувати й до інших типів документів XML.

Дана мова має доволі різні та рівні профілі. Наступний рівень CSS базується безпосередньо на попередньому рівні, додаючи до нього нові або ж при цьому розширюючи наявні функції. Рівні відображаються — CSS1, CSS2 і CSS3. Профіль — набір одного або ж кількох рівних між собою правил CSS, створених для певного типу пристроїв або ж інтерфейсів. CSS замінив табличне, звичне всім, розташування веб-сторінок. Основною перевагою блокових макетів можна назвати можливість відокремити вміст сторінки від її візуальних представлень.

1.4 Розробка на платформі Node.js

Node.js — платформа із відкритим вихідним кодом, що призначена для високопродуктивних мережевих додатків. Якщо раніше використовували JavaScript щоб обробити дані в своєму браузері, то тепер node.js дав можливість запуску сценарію JavaScript на сервері. Платформа Node.js зробила мочву JavaScript спільною мовою із великою спільнотою розробників.

Node.js має наступні властивості:

— неблокуючий ввід або вивід;

— система модулів CommonJS;

— основна сила JavaScript Google V8;

— керування модулями використовує пакетний менеджер npm.

Платформа Node.js була розроблена для запуску високопродуктивних мережевих додатків, що написані мовою програмування JavaScript. Ця платформа використовується не лише для роботи зісценаріями на стороні сервера, а й ще такод для створення клієнтських та серверних додатків. Платформа використовує V8, що розроблений Google.

Node.js обробляє велику кількість одночасних запитів, використовуючи для чого модель виконання асинхронного коду на основі неблокуючої обробки подій та визначення обробників зворотнього виклику. Epoll, kqueue і select підтримуються типу методи мультиплексування з'єднання. Бібліотека libuv застосовується для мультиплексування підключень, а от бібліотека libeio вже використовується щоб створити потоки, а c-ares інтегровано для виконання запитів DNS в неблокуючому режимі. Всі системні виклики, що викликають блокування, будуть здійснювати в потоках та й типу обробники сигналів, що надсилають результати своєї роботи по безіменному каналу.

1.5 Аналіз використання Vue.js

Vue.js — це фреймворк JavaScript, який було використовано для шаблонів MVVM щоб створити інтерфейс користувача на основі моделі даних та за допомогою реактивного зв’язування даних.

Vue використовує синтаксис шаблонів, що робить на основі HTML. Це надає змогу декларативного пов’язування рендерингу DOM із базовими екземпляром даних Vue. Всі шаблони Vue ще є дійсними HTML та їх можна проаналізувати у браузері та HTML-парсері. Внутрішньо Vue компілює шаблон у певну віртуальну функцію візуалізації DOM. Та в поєднанні із реактивною системою Vue може, за нагоди, розумно обчислити кількість компонентів, що необхідно повторно буде відтворити, та й застосовувати при цьому мінімальну кількість операцій DOM, коли стан програми почне змінюється. У Vue можна використовувати синтаксис шаблону або ж ще застосувати JSX вже для безпосереднього написання функцій візуалізації. Для цього просто необхідно замінити шаблон на функцію візуалізації. Функції

візуалізації відкривають можливості для потужних шаблонів, що засвновано на основі компонентів.

Однієї із самих найвиразніших особливостей Vue можна назвати те, що вона ненав'язлива реактивна система. Моделі є просто плоскими JavaScript об'єктами. І це робить керування станами доволі простими та й інтуїтивним. Vue надає оптимізований ре-рендеринг із коробки без потреб робити що-небудь ще при цьому додатково. Кожен із компонентів слідкує за своїми усіма реактивними залежностями під час рендерингу, тому й система знає точно де й коли має відбуватись ре-рендеринг та які при цьому компоненти необхідно ре-рендерити.

Vue дає різноманітні способи застосування для ефектів переходу, при додаванні елементів, що оновлюються або ж видаляються із DOM. Наприклад:

— автоматичне застосування класів;

— інтегрування сторонніх бібліотек для CSS анімацій;

— застосування JavaScript для прямих маніпуляцій;

— інтегрування сторонніх JavaScript бібліотек анімацій.

Це відбувається, коли елемент, обгорнутий компонентом переходу, видаляється або вставляється.

Сам по собі Vue не включає маршрутизацію, але й певний існує пакет vue-router, що вирішує дану проблему. Підтримує зв'язування вкладених шляхів із вкладеними компонентами, щоб був кращий контроль над переходами. Vue дозволяє розробляти програми з допомогою компонентів. Якщо додати до цього vue-router, усе що потрібно було зробити, та ще й пов’язати компоненти вз маршрутами та дозволити vue-router вирішити місця їх відображення.

1.6 Аналіз використання фреймворку React.js

React — відкрита бібліотека JavaScript, що призначена для створення інтерфейсів користувача, вирішення проблеми часткового оновлення вмісту веб-сторінки, який виникає при розробці односторінкового додатка.

React надає змогу розробникам створювати об’ємні веб-додатки, ящо використовують дані, що змінюються зі певним часом, при цьому не перезавантажуючи сторінку. Його мета наступна — бути швидким, простим та масштабованим. React обробляє тільки інтерфейс користувача програми. Та він підтримує ще й представлення шаблонів MVC та може використовуватися у поєднанні й із іншими бібліотеками JavaScript або ж у великих платформах MVC, типу як AngularJS [6]. Також ще й можна застосовувати його із доповненням React для роботи із частинами без інтерфейсу користувача при створенні веб-додатків. Типу як бібліотека інтерфейсу користувача, React доволі часто використовується у поєднанні з іншими бібліотеками.

Цей атрибут передається компоненту візуалізації як атрибут тегу html. Компонент не може безпосередньо змінити передані властивості, але може. Цей механізм називається «зниження атрибутів, зростання події».

React підтримує віртуальний DOM і покладається не тільки на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися з моменту збереження збереженої версії віртуальної DOM, і визначити найкращий спосіб оновлення DOM браузера. Тому програміст керує сторінкою і вважає, що сторінка оновлена, але бібліотека вирішує, який компонент сторінки оновити.

Компоненти React зазвичай пишуться на JSX. Код, що написаний на JSX, компілюється у виклики методів у бібліотеці React. Розробники також можуть писати й на чистому JavaScript. JSX схожий до PHP, ще одної створеної Facebook для розширення XHP.

React використовується не лише й для відтворення HTML у браузері. Наприклад, в Facebook є певна динамічна графіка, яка й відображається за допомогою тегу <canvas>.Netflix і PayPal використовують ізоморфне завантаження для відтворення і одного й того самого HTML на сервері та й клієнті.

Методи життєвого циклу — різні методи, які вбудовані в ReactJS. Що дозволяє розробникам обробляти дані у різні моменти життєвого циклу React. Наприклад:

— shouldcomponentupdate — метод, який повідомляє javascript використовувати логічну змінну для оновлення компонента;

— componentwillmount — метод, який повідомляє javascript налаштувати й певні дані перед монтуванням;

— componentdidmount — це метод життєвого циклу, що подібний до компонента willmount, який й виконується після методу відтворення та може використовуватися й для додавання даних json і визначення властивостей й станів;

— відтворення є найважливішим методом життєвого циклу, який вимагається для будь-якого із компонентів.

Кілька елементів на одному рівні повинні бути обгорнуті тільки в один елемент контейнера, наприклад — , або ж повернути його як масив.

JSX надає ряд атрибутів елемента, призначених для відображення того, що й надається в форматі HTML. Також можна передати користувацькі атрибути компоненту. Усі ці атрибути витягуються компонентом типу як реквізити.

Вирази JavaScript можна використовувати у JSX зі такими фігурними дужками {}.

<h1> {10 + 1} </ h1>

І ось приклад, який наведений вище,та відображатиметься ось так.

<h1>11</h1>

Вирази If-else не можна застосовувати в JSX, проте замість них можна використовувати й інші умовні вирази.

{ i === 1 ? 'true' : 'false' }

Так як рядок 'true', оскільки «i» дорівнює 1.

```
class App extends React.Component {
  render() {
    const i = 1;
    return (
      <div>
        <h1>{ i === 1 ? 'true' : 'false' }</h1>
```

```
      </div>); }
  }
```

Функції та JSX можна й також використовувати у одних умовних виразах.

```
class App extends React.Component {
   const sections = [1, 2, 3];
    <div>{
       sections.length > 0
         ? sections.map(n => <div>Section {n}</div>)
         : null}
    </div>);}}
```

Код, що написаний на JSX, потрібно буде конвертувати зі допомоги інструменту Babel, для того, щоб веб-браузер міг би його зрозуміти. Даний процес зазвичай виконується під час уже процесу складання перед запуском програми.

1.7 Аналіз використання фреймворку Angular.js

AngularJS — це фреймворк JavaScript із відкритим кодом, що буврозроблений Google. Він ж призначений для розробки односторінкового додатка, який складається із однієї сторінки HTML з допомогою CSS та JavaScript. Мета — розширити додаток браузера на основі вже шаблону Model-View-Controller (MVC), для того, щоб спростити тестування та розробку. Фреймворк працює зі сторінками HTML, що містять додаткові атрибути, й пов’язує область введення або ж виведення сторінки із моделлю, яка може бути загальною змінною JavaScript. Значення цих вже змінних можна встановити вручну або ж отримати зі статичних чи динамічних даних JSON.

AngularJS — вхідна частина стеку MEAN, яка складається із бази даних MongoDB, фреймворку розробки веб-додатків Express.js, самого Angular.js і платформи Node.js. AngularJS був заснований на переконанні, що необхідно використовувати декларативне програмування щоб створити інтерфейс користувача і підключення програмних компонентів, тоді ж як імперативное

програмування доволі добре підходить й для визначення бізнес-логіки програми. Фреймворк адаптує та розширює традиційний HTML для відображення динамічного вмісту й через двонаправлені посилання даних та й дозволяє автоматично синхронізувати моделі та представлення. В результаті чого AngularJS зменшує значення явних маніпуляцій DOM та покращує тестування і продуктивність. Конструктивні цілі AngularJS включають в себе відокремлення операцій DOM ужк від логіки програми. Що має значний вплив на те, як будується код, та відокремлює клієнтську частину програми від сервера. Що дозволяє запускати розробку паралельно та й повторно використовувати обидва. Провести розробників по всьому шляху створення додатків — від дизайну інтерфейсу користувача до написання та й тестування бізнес-логіки. AngularJS реалізує шаблони MVC для розділення і представлень, даних й логічних компонентів. Використовуючи реалізація залежностей, Angular вже традиційно надає такі послуги, як контролери, що залежать уже й від перегляду, для веб-додатків клієнт-сервер. Тому навантаження на сервер знижується.

1.8 Технологія WebSocet

WebSocket — це протокол для обміну інформацією у режимі реального часу між браузером й веб-сервером. Він забезпечує двонаправлений повнодуплексний канал від зв'язку через один TCP-сокет. WebSockets розроблено для реалізації в веб-браузерах й веб-серверах, але й також можуть використовуватися й у будь-якому клієнтському серверному додатку. API WebSocket був стандартизований W3C, при цьому протокол WebSocket стандартизований IETF в типу RFC6455. Веб-додаткам рекомендується використовувати протоколи для відображення інформації лише в режимі реального часу за потреби. Альтернативні технології – події, надіслані сервером.

Для того, щоб встановити з’єднання WebSocket, клієнт й надсилає запит на рукостискання. Клієнт також й ще надсилає Sec-WebSocket-Key для

шифрування повідомлення. Відкритий ключ в розділі параметрів запиту HTTP має кодування base64.

GET /ws HTTP/1.1

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Version: 6

Sec-WebSocket-Extensions: deflate-stream

Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==

Коли ж уже з’єднання було встановлено, тоді сервер надсилає відповідь клієнту. Сервер дійсно може й встановити з’єднання WebSocket, правильно ввівши при цьому параметр Sec-WebSocket-Accept. Алгоритм формування — стрічка 258EAFA5-E914-47DA-95CA-C5AB0DC85B11 додається до значення Sec-WebSocket-Key в форматі, й отриманому сервером. Для отриманих стрічок обчислюється хеш SHA1, що закодований в форматі base64.

HTTP/1.1 101 Switching Protocols

Connection: Upgrade

Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=

Специфікація WebSocket визначає й дві нові схеми URI — ws і wss, для незашифрованих і зашифрованих з'єднань відповідно уже. І окрім назви схеми, інші компоненти URI визначаються загальним синтаксисом URI.

Об’єкт WebSocket створюється у конструкторі для встановлення з’єднання для клієнтського сценарію. Даний об’єкт передає параметр WebSocket URI й відповідає функції відповідного виклику при підключенні, отриманні або ж відключенні.

```
<html>
   <head>
        const webSocket = new WebSocket('ws://localhost/echo');
          webSocket.send("Hello Web Socket!");};
        webSocket.onmessage = event => {
          alert('onmessage, ' + event.data);
        webSocket.onclose = event => {
```

```
            alert('onclose');
        </script>
    </head>
    <body>
    </body>
</html>
```

WebSocket підтримується такими браузерами:

— Google Chrome — версія 4.0.249.0;

— Apple Safari — версія 5.0.7533.16;

— Mozilla Firefox — версія 4;

— Opera — версія 10.70 9067.

WebSockets також підтримують:

— мобільну версіяю Safari на ios 4.2.

— браузер os7 Blackberry.

Нова версія WebSockets-07, що виправляє помилки протоколу, була реалізована й ввімкнена за замовчуванням у Firefox 6 та Chrome 14. Існує ще й також параметр команди Google Chrome (-enable-websocket-over-spdy). Експериментальна реалізація WebSocket через SPDY.

1.9 Односторінковий застосунок (SPA - Single Page Application)

Односторінковий додаток, також відомий вже як односторінковий інтерфейс — веб-додаток чи веб-сайт, який поміщається на одній сторінці та й забезпечує зручну роботу для настільних програм.

У односторінковому додатку весь потрібний код (HTML, JavaScript, CSS) завантажується вже разом зі сторінкою або ж зазвичай динамічно за потреби, коли вже користувач взаємодіє із нею. Сторінка не оновлюється або ж користувач не перенаправляється на іншу сторінку під час взаємодії зі сторінкою. Взаємодія односторінкових програм часто передбачає динамічне спілкування з веб-сервером.

1.10 База данних MongoDB

MongoDB — це система управління документами з відкритим вихідним кодом (СУБД), яка вже не потребує опису макета таблиці. MongoDB займає нішу між швидкою, масштабованою системою та яка працює із даними формату ключ/значення, й реляційною базою даних, що є функціональною та легкою для запитів. Код MongoDB написаний на C++ та поширюється під ліцензією AGPLv3.

MongoDB підтримує зберігання документів в форматах JSON, має гнучку мову для генерування запитів, може й індексувати різні атрибути сховища й ефективно надає велике двійкове сховище. Він підтримує протоколювання операцій, що змінюють й додають дані до бази даних, і працює відповідно — відображає/зменшує парадигми та й підтримує конфігурації реплікації та відмов. MongoDB має вбудований інструмент для забезпечення шардингу, який, у поєднанні з реплікацією даних, масштабується й горизонтально без єдиної точки збою. Можна створити кластер зберігання автоматичне відновлення після збою та й передачу навантаження із несправного вузла. Розширення кластера або ж перетворення окремого сервера в кластер можна зробити, та просто додавши нову машину без вимкнення бази даних.

У розробці автори відійшли від необхідності й спеціалізуватися на базах даних, тому їм дуже вдалося відірватися від принципу «вмістити все в один розмір». Мінімізація семантики для маніпулювання транзакціями дозволяє вирішити багато проблем, пов’язаних із недостатньою продуктивністю, та полегшування горизонтального масштабування. Використовувану модель документа зберігання даних (JSON/BSON) доволі легко кодувати та керувати (включаючи використання так званого стилю без схем [4]), а от внутрішнє групування пов’язаних даних ще й більше покращує продуктивність. Реляційний підхід дуже корисний для створення баз даних, де й саме горизонтальне масштабування означає розгортання на кількох машинах. Функції, які й забезпечують найкращу продуктивність, повинні існувати разом зі підтримкою більшої кількості функцій, ніж й дозволяють пари ключ-

значення. Він повинен працювати в будь-якому місці й вз сервера користувача або віртуального. Від машини й до хмарних технологій.

Основні можливості MongoDB:

— документно-орієнтоване сховище;

— досить гнучка мова для створення запитів;

— швидке оновлення «на місці»;

— ефективне зберігання великих обсягів двійкових даних, таких як фотографії та відео;

— журнал операцій, які змінюють дані в базі даних;

— відмовостійкість і підтримка масштабованості — асинхронна реплікація, набори реплікації та шардінг;

— може функціонувати відповідно до парадигми mapreduce.

База даних керує набором документів, що подібний до JSON, який зберігаються у двійковому форматі BSON. MongoDB зберігає та й отримує файли завдяки виклику GridFS. Як й всі інші документно-орієнтовані бази даних, MongoDB не є реляційною базою даних.Як тільки він був випущений, було ж оголошено, що випуск MongoDB 1.0 буде доступний в виробництві як один хост та й у відносинах між провідним й підпорядкованим. Код у цьому випуску дуже стабільний і перевірений у комерційній експлуатації уже протягом 1,5 років. Рекомендуємо використовувати головну/підпорядковану реплікацію для розгортання MongoDB принаймні на двох серверах. Це гарантує доступність відповідних даних в разі відмови однієї з СУБД. MongoDB-Цей продукт настільки молодий, що й вносить помилки, нові можливості тощо. Відрізняється швидкими темпами розвитку. Розробники компанії надають платну підтримку, хостинг та й консультації.

Запити можуть отримувати дані й із вбудованих об’єктів і масивів. Якщо такий об’єкт вставлено в колекцію користувачів:

```
"address" :
  "street" : "123 Main Street",
  "city" : "Springfield",
  "state" : "NY"
```

Можемо запитати цей документ за допомогою.

```
> db.users.find({"address.state" : "NY"})
```

Також можна запитувати елементи масиву.

```
> db.food.insert({"fruit" : ["peach", "plum", "pear"]})
> db.food.find({"fruit" : "pear"})
```

1.11 Express.js

Express.js, або просто Express — програмна платформа для розробки серверної частини й веб-додатка для Node.js й реалізована типу як безкоштовне програмне забезпечення зі відкритим кодом за ліцензією MIT. Він був призначений для створення веб-додатків та API. Фактично — це стандартний фреймворк Node.js. Express є серверною частиною програмного стеку MEAN, а й також базою даних MongoDB і ще й фреймворком AngularJS для інтерфейсу.

1.12 Docker

Docker — це інструмент із відкритим кодом, який й автоматизує розгортання додатків у контейнерах, які й підтримують контейнеризацію. Розробляючи свою програму, повинні знати про всі й необхідні залежності, такі як — бібліотеки, веб-сервери, бази даних тощо. В іншому випадку й програма може працювати на вашому комп’ютері, але не на проміжному сервері чи комп’ютері й іншого розробника чи тестувальника. Цю проблему можна вирішити, відокремивши й програму від системи. Віртуальні машини зазвичай використовуються, для того щоб уникнути несподіваної поведінки програми. Основна проблема полягає в тому й, що віртуальні машини — це додаткові операційні системи, що працюють й на хості, а це додаткові гігабайти проекту.

Часто сервер обслуговує кілька віртуальних машин, кожна із яких займає достатньо місця. Крім того й, відомо, що більшість хмарних платформ платять й за використання додаткового простору. Ще одним істотним недоліком й віртуальних машин є їх повільне завантаження. Docker вирішує всі ці

проблеми шляхом спільного використання ядра ОС між усіма контейнерами, які працюють як окремі процеси ОС хосту.

1.13 Порівняльні характеристики фреймворків

1.13.1 Переваги та недоліки Angular

Переваги Angular:

— angular використовується в поєднанні й із typescript;

— angular-language-service-надає інтелектуальні функції та компоненти html для шаблонів автозаповнення;

— нові функції;

— реалізація залежностей від модулів й компонентів, пов'язаних із загальною модульністю;

— конструкції та архітектури, спеціально розроблені для великих проектів.

Недоліками є:

— продуктивність повільна;

— різноманітні конструкції (ін’єкції, труби, компоненти, модулі тощо) — ускладнює розслідування порівняно з vue, який має лише компоненти.

1.13.2 Переваги та недоліки React

Аналізуючи фреймворк React, визначили його переваги:

— першокласна підтримка прогресивних веб-програм завдяки генератору програм «create-react-app»;

— прив’язка даних є односторонньою, тому й менше непотрібних побічних ефектів;

— redux, найпопулярніша платформа керування додатками react, проста в освоєнні та й використанні;

— програми можна створювати за допомогою typescript або facebook flow, які й мають вбудовану підтримку jsx;

— перемикатися між версіями зазвичай дуже легко — facebook має модуль коду для автоматизації більшої частини процесу.

Недоліками ж є наступне:

— React не підтримує ie8 і нижче браузери;

— якщо веб-сайт не насичений великою кількістю динамічних сторінок, то й доведеться написати багато коду, для того щоб вирішити невелику проблему;

— оманливий синтаксис jsx;

— складність пошукової оптимізації;

— зосередження на інтерфейсі користувача.

1.13.3 Переваги та недоліки Vue

Аналіз фреймворка Vue виявив його такі переваги:

— висока адаптивність — можна здійснити швидкий перехід з інших фреймворків на vue.js;

— швидкість та продуктивність;

— масштабування — vue.js допомагає розробляти великі компоненти для багаторазового використання, які й можна створювати приблизно в той же й час, що й простіші.

Недоліками є:

— ризик надмірної гнучкості — можуть виникнути проблеми, коли vue інтегрується у великий проект, і поки й немає можливого рішення;

— брак ресурсів — vue все ще й має невелику частку ринку порівняно з angular. це означає, що й обмін знаннями в цьому середовищі ще знаходиться на стадії становлення;

— компонентний підхід vue.js не такий гнучкий і очевидний, як react.

1.13.4 Порівняльна таблиця характеристик фреймворків

Результати порівняння фреймворків представлені в окремій таблиці 1.1 ключових характеристик, що відповідають цілям розробки веб-чату.

Таблиця 1.1 — Порівняльна таблиця фреймворків

	Vue	React	Angular
Швидкість та продуктивність	+	—	—
Простота підтримки та розширення	+	+	—
Простота процесу розробки	+	+	—
Безпека розробки	—	—	+

У цьому розділі описуються й аналізуються поточні технології веб-розробки, такі як мови програмування Vue.js, React.js, Angular.js, Node.js, MongoDb, WebSocket та JavaScript. Виконуються характеристики порівняння фреймворків Vue.js, React.js і Angular.js.

Зроблено висновок, що для прискорення веб-чату його слід використовувати для розробки таких технологій, як Vue.js, Node.js, MongoDb і WebSocket. Щоб реалізувати універсальне розгортання, потрібно використовувати інструменти Docker.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для досягнення мети у дипломній роботі потрібно розробити та стилізувати сторінки, розробити програмне забезпечення клієнтської та серверної частин використовуючи вибрані технології, налаштувати Docker для контейнеризації застосунку.

Розробка програми розпочалася зі створення діаграми архітектури веб-чату (рисунок 2.1).

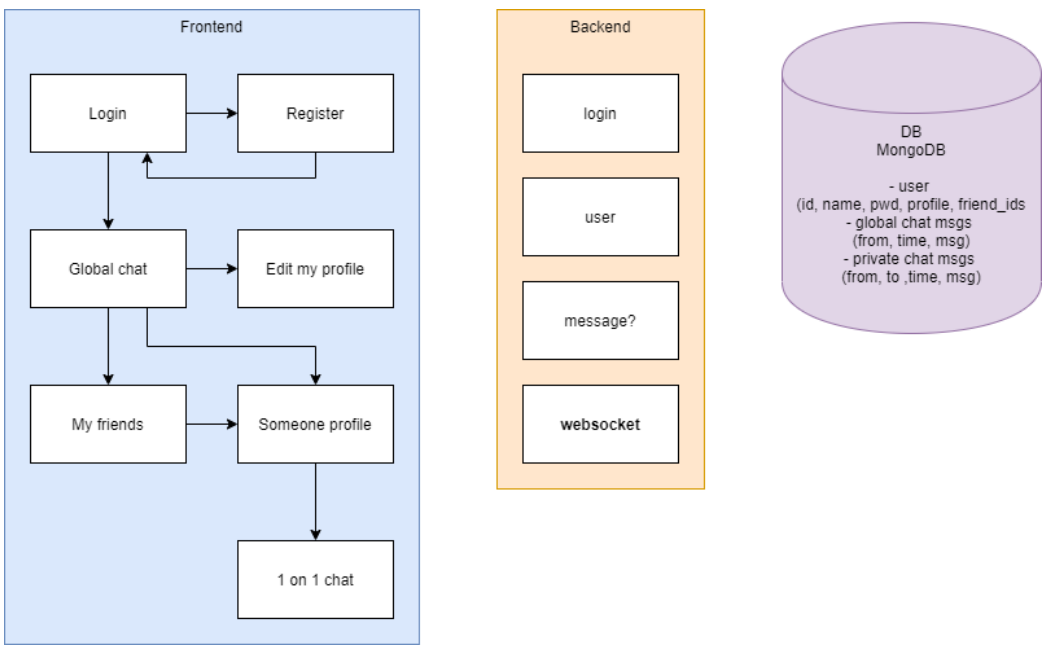


Рисунок 2.1 —Зовнішній вигляд сторінки авторизації

На цьому малюнку також показано схему навігації програми.

2.1 Розробка клієнтської частини

Сторінки були створені та стилізована за допомогою HLML та CSS.

2.1.1 Розробка сторінки авторизації та реєстрації

Сторінка затвердження та реєстрації створена за допомогою тегів блоку , тексту <label> поля вводу <input>, та кнопки <button>.

```
                <div class="container">
                    <label><b>Your name</b></label>
                    <div><input id="username" v-
model="login" placeholder="Enter username" tabindex="1" type="text"></div>
                    <label><b>Your password</b></label>
                    <div><input id="password" v-
model="password" placeholder="Enter password" tabindex="2" type="password">
</div>
                    <div><label><input type="checkbox" checked="checked" name="
remember"> <span class="lgn">Remember me</span></label></div>
                    <div><button v-
        on:click="onRegister()" tabindex="4" class="lgn">Create your account</but
ton></div>
                </div>
              </div>
            </div>
```

Щоб виглядати добре, сторінки стилізовані за допомогою таблиць стилів Cascade CSS. Для властивостей потрібні відступи, кольори, шрифти та тіні. Відступ встановлюється за допомогою властивостей position, left, top, width, height, margin, а властивість box-shadow додає тінь. Ви можете вказати шрифт і фон за допомогою властивості font. Кольори та кольори відповідають за колір фону, а елементи або текст використовуються залежно від тегів, у яких вони використовуються.

```
        body {
          background: url(fine_chat.jpg);
          background-size: 100%;}
        #action_form{
            background-color: white;
            position:absolute;
            width:300px;
            box-shadow: -10px 10px 5px 0px rgba(0, 0, 0, .2)}
```

```
input[id=username], input[id=password] {
    padding: 12px 20px;
    outline:none;
    margin: 8px 0;
    width: 85%;
    border: 1px solid #ccc;
    font: 100% sans-serif;}
input[id=username]:focus{
  box-shadow: 0 0 5px 0px #4CAF50;}
input[id=password]:focus{
  box-shadow: 0 0 5px 0px #4CAF50;}
    color: white;
    padding: 14px 20px;
    margin: 10px 0;
    border: none;
    cursor: pointer;
    width: 100%; }
label
button[class=lgn]:hover {
    opacity: 0.7;}
button[class=lgn]:active{
  background-color: #4CAF50;}
.container
{
    padding: 16px;}
```

В результаті розробки також було розроблено зовнішній вигляд сторінки авторизації та реєстрації (рисунок 2.2).

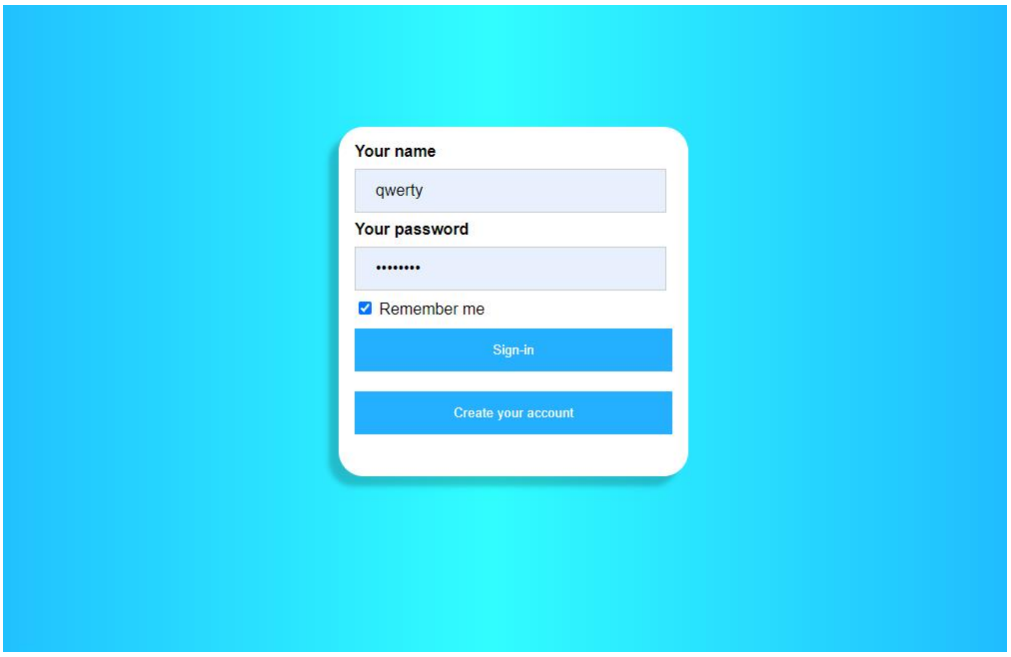


Рисунок 2.2 — Зовнішній вигляд сторінки авторизації

Після схвалення або реєстрації користувач повинен перейти на сторінку чату, далі розвиваючи основний інтерфейс веб-додатка.

2.1.2 Розробка панелі навігації та списку користувачів

Було вирішено створити панель навігації (див. додаток Б), яку можна використовувати звідусіль, й список усіх користувачів, якими можна й користуватися з будь-якого місця. Якщо використовуєте тег блоку для створення блоку навігації, тег <ul> відповідає за список <li>. У списку стилізоване посилання <a> додається до кожного й елемента <li>, і користувач перемикає інтерфейси, коли натискає посилання.

```
<div class="nav-panel">
        <ul class="">
          <li class="item"><a href="#"><span>Global chat</span></a></li>
            <li class="item"><a href="#"><span>My profile</span></a></li>
             <li class="item"><a href="#"><span>My friends</span></a></li>
="button" value="Logout" class="logout-btn">
            </ul>
          </div>
```

Далі панель навігації потрібно було ще й стилізувати.

```
.nav-panel{
    width: 200px;
    background: #59c07e;
    float: left;
-start: 0px;
    text-align: center;}
ul li a{
        padding: 20px;
        color:  black;
#007a258e;
        font-family: 'Roboto', sans-serif;
        }
ul li a:hover{
        background: #10ad4a;
        color: white;
        text-decoration: none;}
```

Тобто, залежно від вибраного інтерфейсу, викликаний блок буде відображатися між блоком й панелі навігації та блоком списку користувачів. Панель навігації та списку користувачів залишаються незмінними незалежно від функції, яку користувач викликає. Список користувачів також показує додаткову мітку (рисунок 2.3).

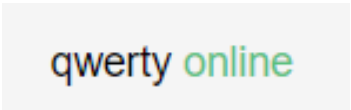


Рисунок 2.3 — Мітка «онлайн»

Якщо інший користувач у мережі, й мітку (рисунок 2.4), якщо вони є у поточному списку друзів.

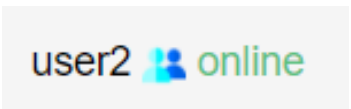


Рисунок 2.4 — Мітка «друг»

Цей список також є й блоком зі списком <li> і текстовим полем . Цей список також стилізовано, щоб добре виглядати.

```
<div class="online-list">
                    <span class="span-online-users">Users:</span>
                    <li class="online-list-item" v-
on:click="onSomeoneClick(item)" v-for="item in userList" :key="item._id">
                    <a class="online-list-a"><span >{{ item.name }}</span></a>
    position: relative;
    left: 20px;
    top: 10px;
}
.online-list-a {
    cursor: pointer;
    color: black;
}
.online-list-a:hover {
    cursor: pointer;
    color: #59c07e;
}
```

2.1.3 Розробка блоків приватного та глобального чатів

Між блоком навігації та списком користувачів потрібно створити блоки глобальний чат, й приватний чат, список друзів, і свій профіль та й чужий профіль. Приватний чат та блокування профілів інших людей використовують глобальний чат і блокування особистого профілю, тому ми розглядатимемо лише чат, списки друзів та розробку вашого власного профілю.

Щоб створити чат, потрібно створити поле введення <input>, куди користувач вводить повідомлення, блок , де й відображаються повідомлення поточного користувача та інших користувачів, й кнопку <button> для відправлення повідомлення.

```
 <div class="chat-list">
```

```
                    <li v-for="item in messages" :key="item._id">
                   <b>{{ item.fromName }}:</b> {{ item.text }}
                   </li>
                </div>
```

Також ще й потрібно було стилізувати даний блок.

```
.chat{
    position: absolute;
    float: left;
.chat-list{
    position: absolute;
    float: left;
    top: 10px;
    bottom: 50px;
    left: 10px;
    right:0px;
}
{
    background: #fff;
    border: 1px solid #cfdae1;
    overflow: hidden;
    float: left;
    }
    .input-area .input-wrapper input{
    height: 30px;
    line-height: 30px;
    border: 0;
    background-color: #4CAF50;
    color: white;
    padding: 8px 17px;
    border: none;
    cursor: pointer;
```

Отримано сторінку із чатом, навігацією та також з списком користувачів (рисунок 2.5).

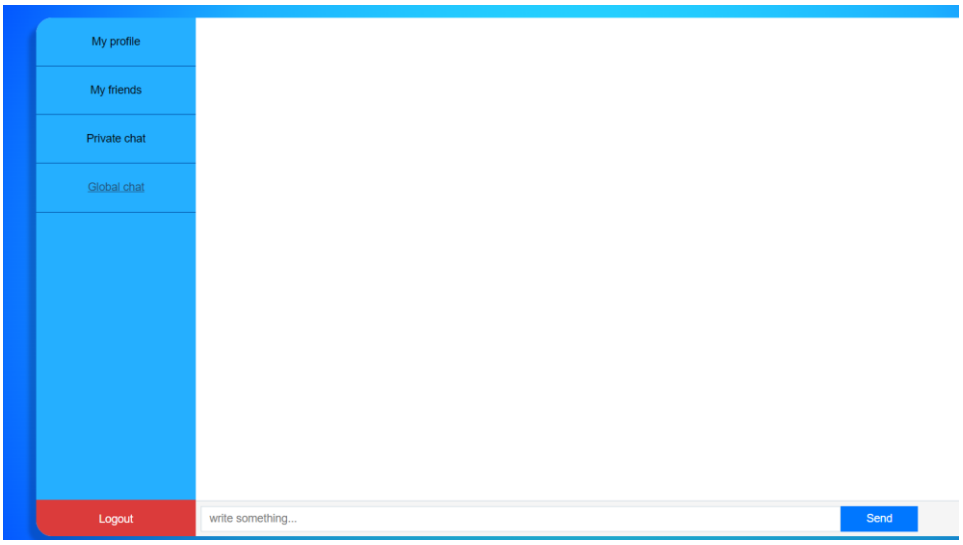


Рисунок 2.5 — Сторінка з чатом, списком користувачів та навігацією

2.1.4 Розробка блоку профілю користувача

Потім й створюється блок власного профілю. Сторінка профілю містить поля введення, які й вимагають від користувачів ввести інформацію про й йсебе, свій статус, ім’я, місто, номер телефон у та адресу електронної пошти. Для збереження змін опитування створено кнопку <button> Save Changes.

Status

<input v-model="user.status" type="textarea" value="" placeholder="write something..."></ div>

span class="span-profile">City

model="user.phone" type="tel" value="" placeholder="write something...">

Email

```
                    <div class="input-wrapper-profile"><input v-
model="user.email" type="email" value="" placeholder="write something..."></di
v>
                    </div>
                    <input type="button" value="Submit" class="submit-btn-
profile" v-on:click="submitUserData()" >
                  </div>
                    </div>
```

Також ще було трішки стилізовано даний блок.

```
.profile-window{
   position: absolute;
   float: left;
   top: 10px;
   bottom: 0;
   left: 10px;
   right:0;
   }
.input-area-profile{
   overflow: hidden;
   float: left;
   }
.input-area-profile .input-wrapper-profile input{
   height: 30px;
   line-height: 30px;
   border: 0;
   margin: 0;
   padding: 0 10px;
   outline: none;
   color: #5D7185;
   min-width: 865px;
   }
```

```
.submit-btn-profile{
    position: relative;
    top: 10px;
    background-color: #4CAF50;
    color: white;
```

Отже, отримали готовий блок для свого профілю. При виклику поточним користувачем він й буде відображатися між блоком навігації та й списком користувачів (рисунок 2.6).

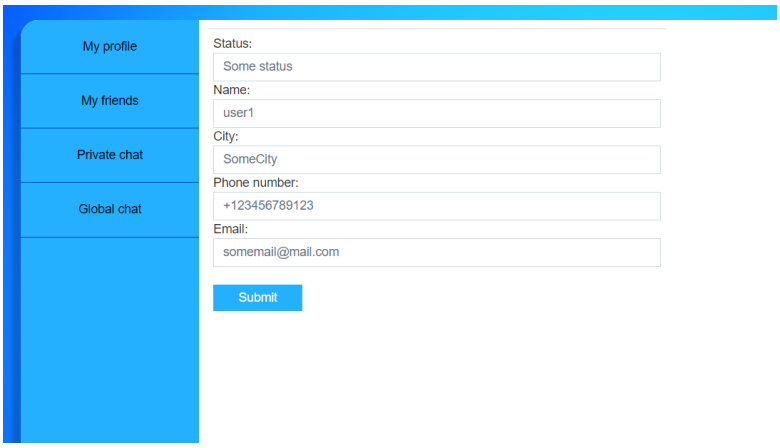


Рисунок 2.6 — Сторінка власного профілю

2.1.5 Розробка блоку списку друзів

Натинувши кнопку «Друзі» на панелі навігації, щоб побачити список друзів. Список <ul> показує імена користувачів, яких й поточний користувач додав до своїх друзів. Імена цих користувачів і є посиланнями <a>, тому їх й можна натиснути. Натисніть на користувача у своєму списку друзів, щоб відкрити й свій профіль.

```
<div class="friend-window">
                    {{user.name}} friends:
                    <ul>
                      <li clas v-for="item in user.friends" :key="item.id">
                        <a v-
on:click="onSomeoneClick(item)" class="friedns-list"
```

Та відбулась стилізація даного блоку.

```
.friend-window{
    position: absolute;
    top: 10px;
    bottom: 0px;
    left: 10px;
```

Ну й було отримано готовий блок списку друзів (рисунок 2.7).

Рисунок 2.7 — Сторінка списку друзів

2.1.6 Розробка переходів між сторінками Vue-router

Далі й створено програму Vue, використовуючи блоки на цих сторінках. Vue дозволяє динамічно змінювати структуру HTML-документа. Ця властивість використовується для створення переходів й (маршрутизації) між сторінками. Далі покажу вам, як й створити маршрутизатор за допомогою Vue.js. Наприклад, ми обговоримо лише один із й переходів від сторінки до сторінки: перехід на сторінку особистого профілю.

Функція була оголошена, й ключ dataUri створено. Спочатку ключ був встановлений на "global_chat", оскільки було й вирішено, що головною сторінкою буде глобальний чат користувача. Значенням, яке приймає ключ, є блок, який відображається у відповідь на дію користувача.

```
export default {
    data: function () {
        return {
            dataUri: 'global_chat',
        };
    },
```

Директива v- вказує браузеру що використовується саме Vue, а обробник подій on:click означаєй, що потрібно й зробити якусь дію по натисканню правої кнопки миші. Та й ключу dataUri присвоїти значення ‘my_profile’.

```
<li class="item" v-on:click="dataUri = 'my_profile'">
<a href="#"><span>My profile</span></a>
</li>
```

Далі потрібно вказати у Vue, що малювати одним й натисканням кнопки. Для визначення умов використовуються умовні директиви v-if. Якщо ж ключ dataUri — «my_profile», відобразіть блок й, що містить особистий профіль. У цьому блоці створено поля опитування користувачів. Також й створена кнопка збереження. При натисканні кнопки викликається функція submitUserData, а й тіло функції POST пропонує внести зміни й до бази даних.

```
<div v-if="dataUri === 'my_profile'">
                        <div class="profile-window">
                         <div class="input-area-profile">
                            <div><span class="span-profile">Status</span></div>
                               <div class="input-status">
                                  <div class="input-wrapper-profile"><input v-
model="user.status" type="textarea" value="" placeholder="write something..."></
div>
                               </div>
                                  <div><span class="span-
profile">Name</span></div>
                            <div class="input-name">
                               <div class="input-wrapper-profile"><input v-
model="user.name" type="textarea" value="" placeholder="write something..." ><
/div>
                            </div>
                                <div><span class="span-
profile">City</span></div>
                            <div class="input-city">
```

```
                    <div class="input-wrapper-profile"><input v-
model="user.city" type="textarea" value="" placeholder="write something..."></di
v>
                    <div class="input-wrapper-profile"><input v-
model="user.email" type="email" value="" placeholder="write something..."></di
v>
                </div>
                <input type="button" value="Submit" class="submit-
btn-profile" v-on:click="submitUserData()" >
            </div>
        </div>
    submitUserData() {
        axios.post(`${backendUrl}/users/${this.userId}`, this.user);
```

Нижче описується вихід бази даних користувача. Наприклад, буде й описано виведення інформації профілю іншого користувача. Блоки профілів інших користувачів подібні й до поточних блоків профілю, із деякими відмінностями. Не можнай редагувати профілі інших користувачів. Лише й для відображення додано кнопку [Додати до друга]. Якщо жвибраний користувач вже є другом поточного користувача, кнопка буде динамічно замінена. За допомогою кнопки «Видалити з друзів». Для виведення даних користувача із бази даних використовувалася директива компонента v-model. Значення some.name витягується із масиву й відображається у формі.

```
<div><span class="span-someone-profile">Name</span></div>
                    <div class="input-name">
                        <div class="input-wrapper-someone-
profile"><input v-
model="someone.name" type="textarea" readonly value="" ></div>
```

Кнопки «Додати до друзів» і «Видалити з друзів» працюють таким чином. Створено кнопку та використовувано умовну директиву v-if, щоб встановити умову. Якщо ж вибраний профіль є другом поточного користувача isFriend = true, видаліть його із кнопки друзів й встановіть v-, щоб додати.

Клацніть директиву обробника подій. Події — це функція видалення друзів. Ця функція використовує фільтр масиву для видалення вибраного користувача та й замінює значення isFriend з = true на = false. Тіло й використовує існуючу функцію submitUserData. Це й змушує запит POST час від часу оновлювати користувачів сервера та бази даних й видаляти їх зі списку друзів. Таким же чином, вибраний користувач більше не є другом поточного користувача, і замість кнопки Видалити з друга відображається кнопка Додати до друга. Це й повна протилежність.

```
<input type="button" v-if="!someone.isFriend" v-
on:click="addFriend()" value="+ Add to friends" class="add-to-friends-btn">
                                <input type="button" v-if="someone.isFriend" v-
on:click="removeFriend()" value="Remove friend" class="add-to-friends-btn">
    addFriend()
{
            this.user.friends.push({
               id: this.someone.id,
            let zz = this.someone;
            this.someone = {};
            this.someone = zz;
removeFriend()
{
            this.user.friends = this.user.friends.filter(e => e.id !== this.someon
e.id);
            this.submitUserData();
            this.someone.isFriend = false;
            let zz = this.someone;
```

2.1.7 Розробка функціоналу кешування данних з використанням Cookie

Нижче описано, як файли cookie працюють на стороні клієнта. Комп’ютерний термін cookie — це термін, який використовується для опису інформації у вигляді текстових або двійкових даних, отриманої з веб-сайту на

веб-сервері, що й зберігається клієнтом або браузером на тому самому сайті при повторенні. Вона й буде надіслана, й отримана. У цьому випадку ідентифікатор користувача та й маркер повинні зберігатися в файлі cookie. За допомогою файлів cookie користувачам не потрібно повторно авторизуватися. Якщо користувач уже відвідує сайт, браузер повідомляє серверу, що й користувач уже схвалений. Якщо користувач не схвалений, або якщо у нього немає власного профілю, маркер буде нульовим, тому якщо токен дорівнює нулю, то й є умова відправити користувача на сторінку затвердження.

```
this.userId = $cookies.get('userId');
this.usertToken = $cookies.get('userToken');
        if (this.usertToken === null)
{
            window.location.href = '/loginForm.html';
```

Кнопка деавтентифікації також й працює із логікою cookies. Якщо ж користувач нажимає на вихід, то й видаляється його токен, як це описано вище, якщо ж токен рівний нулю, то й користувач потрапляє на форму авторизації.

```
logout()
{
            $cookies.remove('userToken');
            window.location.href = '/loginForm.html';
```

2.1.8 Розробка валідації

Також й є ще один HTML-документ у вигляді затвердження та реєстрації користувача. Дані перевіряються, й коли користувач вводить логін і пароль. Наприклад, логін має містити щонайменше 4 символи, а пароль — не менше 6 символів. Це й лише одна із кількох перевірок.

```
validateLogin(login, password) {
            if(login.length <= 0) {
                alert('Password and Login fields cannot be empty');
                return false;
            }
```

```
                if(password.length <= 0) {
                   alert('Password and Login fields cannot be empty');
                   return false;
                }
                   {alert('Password and Login fields cannot be empty'); return fa
lse;}
                if(!charMatch.test(login))
                   {alert('Login must contain only latin letters'); return false;}
                if(login.length < 4 || login.length > 20)
                   { alert('Login must be between 4 and 20 characters'); return fa
lse;}
                if(parseInt(login.substr(0, 1)))
                   {alert('Login must begin with a letter'); return false;}
                if(password.length <= 0)
                   {alert('Password and Login fields cannot be empty'); return fa
lse;}
                if(!charMatch.test(password))
                   {alert('Password must contain only latin letters'); return false}
                if(password.length < 6)
                   { alert('Password must be more than 6 characters'); return fals
```

Якщо ж користувач хоче зареєструватися, натисніть кнопку «Створити обліковий запис». На сервер надходить запит POST, логін й пароль користувача надсилаються на сервер, й також призначаються маркер та ідентифікатор cookie. Якщо ж користувач з таким логіном вже існує, отримаєте помилку «Не вдається створити користувача». Для затвердження ви побачите помилку «Недійсний логін та/або пароль».

```
onLogin() {
                if (this.validateLogin(this.login, this.password) === true) {
                   axios.post('http://localhost:3000/login/', {
                      login: this.login,
                      password: this.password,
```

```
                    })
                    .then(response => {
                        $cookies.set('userId', response.data.id);
                        $cookies.set('userToken', response.data.token);
                        window.location.href = '/';
                if (this.validateRegister(this.login, this.password) === true) {
                    axios.post('http://localhost:3000/register/', {
                        login: this.login,
                        $cookies.set('userId', response.data.id);
                        $cookies.set('userToken', response.data.token);
                        window.location.href = '/';})
                    .catch(error => {
                        alert('Can`t create this user');});}},
```

2.2 Розробка серверної частини

Основне завдання при створенні чату — реалізувати відправку повідомлення. У цьому чаті повідомлення надсилається як приватно (переконайтеся, що спілкуються лише два користувачі), так і глобально (видиме для всіх користувачів).

2.2.1 Розробка функціоналу відправлення та отримання повідомленнь

Клієнт повинен отримати повідомлення від сервера і показати його користувачеві.

```
onMessageReceived(data)
{
            if (data.type === 'public') {
                console.log('New public message from:', data.from, ', data:', data
);
                this.messages.push(data);
                // TODO: DO we need this?
```

```
                if (this.messages.length > 10) {
                    this.messages = this.messages.slice(-10);
                }
            } else if (data.type === 'private') {
                let friend = data.from === this.userId ? data.to : data.from;
                if (!this.private_messages[friend]) {
                    this.private_messages[friend] = [];
```

Далі потрібно відобразити повідомлення. Використовувати директиву v-model, щоб призначити значення тексту повідомлення. Користувачеві видно лише й текст повідомлення та відправника.

```
<div class="input-wrapper"><input v-model="privateChatText" type="textarea" value="" placeholder="write something..."></div>
<div class="input-wrapper"><input v-model="globalChatText" type="textarea" value="" placeholder="write something..."></div>
```

Далі й потрібно надіслати повідомлення. Повідомлення надсилається через відкритий веб-сокет. Й Під час надсилання загальнодоступного повідомлення надсилаються лише такі дані, як ідентифікатор користувача, тип повідомлення, текст тощо. Для приватних повідомлень додається ключ до. Він отримує й значення вибраного користувача та розуміє, куди надіслати повідомлення.

2.2.2 Розробка з’єднання через WebSocket

Для забезпечення швидкого з’єднання між клієнтом та сервером розробленно з’єднання через WebSocket, що також дозволило не навантажувати сервер великою кількістю повторних запитів, щоб отримувати актуальну інформацію на стороні клієнту.

```
app.ws('/chat/:id', (ws, req) => {
    ws.uuid = uuidv4();
    ws.userId = req.params.id;
```

```
sessions.push(ws);
console.log('Connected:', ws.uuid, ', userId:', ws.userId);
sessions.forEach(w => w.send(JSON.stringify({
  type: 'online',
})));
ws.on('message', async (msg) => {
  let data = JSON.parse(msg);
  console.log(data);
  var result = await db.collection('users').find().toArray();
  var userNames = {};
  result.forEach(u => userNames[u._id] = u.name);
  if (data.type === 'public') {
    var message = {
      type: data.type,
      from: data.from,
      fromName: userNames[data.from],
      text: data.text,
    };
    await db.collection('messages').insertOne(message);
    sessions.forEach(w => w.send(JSON.stringify(message)));
  } else if (data.type === 'private') {
    var message = {
      type: data.type,
      from: data.from,
      fromName: userNames[data.from],
      to: data.to,
      toName: userNames[data.to],
      text: data.text,
    };
    await db.collection('private-messages').insertOne(message);
    // Send to recipient and sender only
```

```
        sessions.filter(w => w.userId === data.from || w.userId === data.to)
          .forEach(w => w.send(JSON.stringify(message)));
      }
    });
```

2.2.3 Розробка зв’язку з базою данних MongoDB

Щоб відкрити з’єднання з базою данних було реалізовано наступний функціонал

```
mongo.MongoClient.connect(dbUrl, (err, client) => {
  if (err) throw err;
  const db = client.db(dbName);
  console.log('Connected to MongoDB: ' + dbUrl + '/' + dbName);
  app.use(function(req, res, next) {
    res.header('Access-Control-Allow-Origin', frontendUrl);
    res.header('Access-Control-Allow-Credentials', true);
    res.header('Access-Control-Allow-
Methods', 'HEAD, OPTIONS, GET, POST, PUT, DELETE');
    res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-
With, Content-Type, Accept, Authorization');
    next();
  });
```

2.2.4 Розробка функціоналу реєстрації користувача

У цьому розділі описано реєстрацію користувача. Існує запит на публікацію, й користувачеві призначається маркер cookie, зашифрований у формі uuidv4. Потім й тіло описує екземпляр користувача з порожніми даними. Це й тому, що він новий користувач і ще не увійшов у свій профіль. Потім, коли ж ви оновлюєте колекцію в базі даних, користувачу буде призначено ідентифікатор й маркер у базі даних.

```
  app.post('/register', async (req, res) => {
    var user = await db.collection('users').findOne({
      login: req.body.login,
```

```
  });
  if (user !== null) {
    res.status(500).json({message:"already exist"});
  } else {
    newToken = uuidv4();
    var newUser = {
      login: req.body.login,
      password: req.body.password,
      token: newToken,
      name: req.body.login,
      active: true,
      status: '',
      city: '',
      phone: '',
      email: '',
      friends: []
    };
```

Повний список користувачів отримано для відображення клієнту. Користувач шукає в базі даних і відображається його ідентифікатор та ім’я.

```
app.get('/users', async (req, res) => {
  var result = await db.collection('users').find({ active: true }).toArray();
  var users = result.map(u => {
    return {
      id: u._id,
      name: u.name
    };
  });
  res.json(users);
});
```

2.2.5 Розробка відправлення інформації про користувача з серверу

Щоб побачити дані про обраного користувача на клієнті, також потрібно прочитати їх із бази даних на сервері.

```
app.get('/users/:id', async (req, res) => {
   var result = await db.collection('users').find().toArray();
   var userNames = {};
   result.forEach(u => userNames[u._id] = u.name);
   var userId = new mongo.ObjectID(req.params.id);
   var result = await db.collection('users').findOne({ '_id': userId });
   var user = {
     id: result._id,
     login: result.login,
        id: u.id, name: userNames[u.id]};
```

Коли користувач змінює дані, йому ж потрібно натиснути кнопку підтвердження, щоб й оновити дані в базі даних. Для пошуку потрібного користувача викликається запит POST.

```
app.post('/users/:id', async (req, res) => {
   var userId = new mongo.ObjectID(req.params.id);
   var userToken = req.cookies.userToken;
   var user = {
     name: req.body.name,
     active: req.body.active,
     status: req.body.status,
     city: req.body.city,
     phone: req.body.phone,
     email: req.body.email,
     friends: req.body.friends,
   res.json(result);});
```

2.2.6 Розробка відправлення інформації про повідомлення на клієнт

Щоб побачити старі повідомлення, потрібно знайти їх у базі даних, відсортувати та й показати користувачам.

```
app.get('/messages', async (req, res) => {
  var users = await db.collection('users').find().toArray();
  var result = await db.collection('messages').find().sort({$natural:-
1}).limit(5).toArray();
  result.reverse();
  result.forEach((m) =>  {
    var fromUser = users.find(u => u._id == m.from);
    m.fromName = fromUser == null ? 'Unknown' : fromUser.name;
  });
  res.json(result);
});
```

Відправлені повідомлення також й мають бути записані в базу даних.

```
app.post('/messages/:id', async (req, res) => {
  var message = {
    from: req.params.id,
    text: req.body.text,
  };
  var result = await db.collection('messages').insertOne(message);
  res.json(result.ops[0]);
});
```

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ДОДАТКУ

Робота веб-чату буде продемонстрована відповідно до пунктів, описаних у другому розділі дисертації.

3.1 Дослідження роботи форми авторизації та реєстрації

Форма затвердження та створення нового облікового запису (рисунок 3.1).

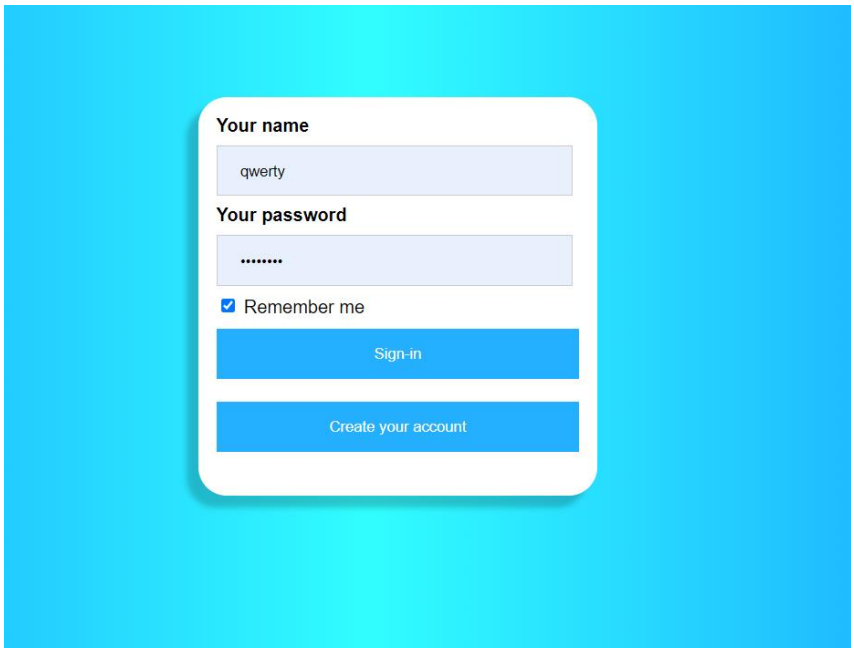


Рисунок 3.1 — Форма авторизації та створення аккаунту

Ця форма дозволяє користувачеві ввести логін і пароль. Якщо у вас немає облікового запису, вам потрібно буде натиснути кнопку «Створити обліковий запис», а потім підтвердити та переконатися, що користувач з таким логіном зареєстрований. Буде створений новий користувач. Дані, які він ввів.

3.2 Дослідження роботи форми глобального чату

Для відправлення повідомлень які буде видно всім користувачам потрібно використати форму глобального чату (рисунок 3.2).

Ця форма дозволяє користувачам надсилати повідомлення до Global Chat та отримувати повідомлення від інших користувачів у Global Chat, відображаючи список усіх користувачів. Користувачі, які зараз користуються чатом, будуть виділені як «онлайн» у списку.

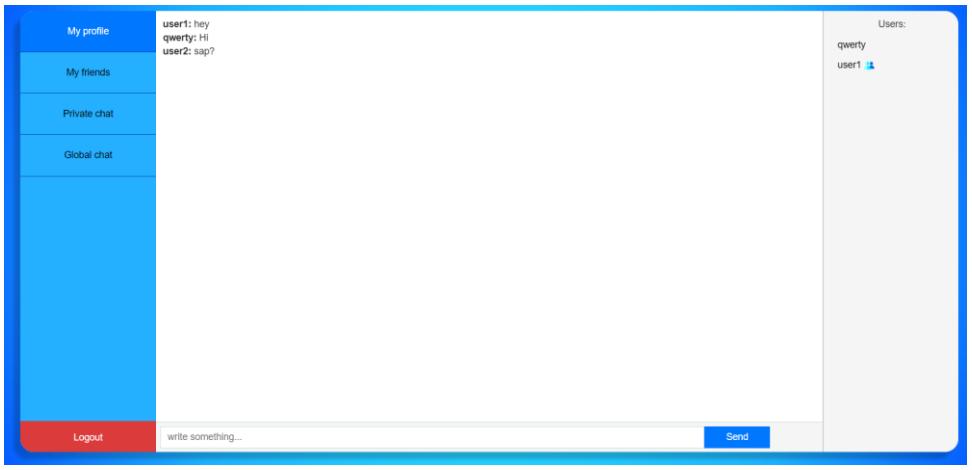


Рисунок 3.2 – Форма глобального чату

3.3 Дослідження роботи панелі навігації

Ліворуч також є панель навігації (рисунок 3.3). Навігація та списки користувачів доступні в будь-якому форматі. Будуть змінені лише блоки з вибраною користувачем функцією.

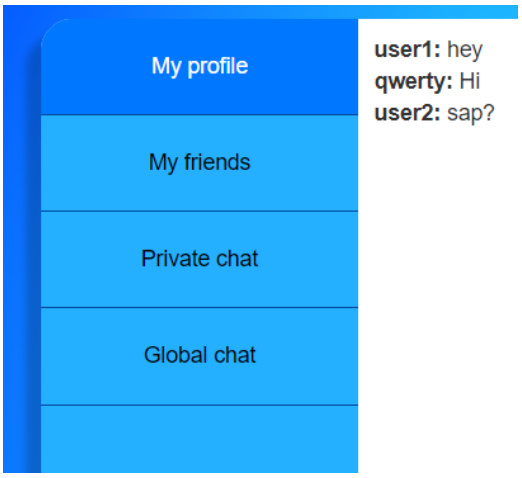


Рисунок 3.3 — Блок панелі навігації

3.4 Дослідження блоку профілю користувача

Коли поточний користувач натискає на користувача зі списку всіх користувачів, відображається форма профілю для вибраного користувача (рисунок 3.4). Цю форму можна лише переглянути. Тобто редагування поля заборонено. Якщо вибраний користувач не є другом поточного користувача, відображається кнопка «Додати до друга». В іншому випадку, якщо вибраний користувач уже є у вашому поточному списку друзів, ви побачите кнопку «Видалити з друзів» (рисунок 3.5).

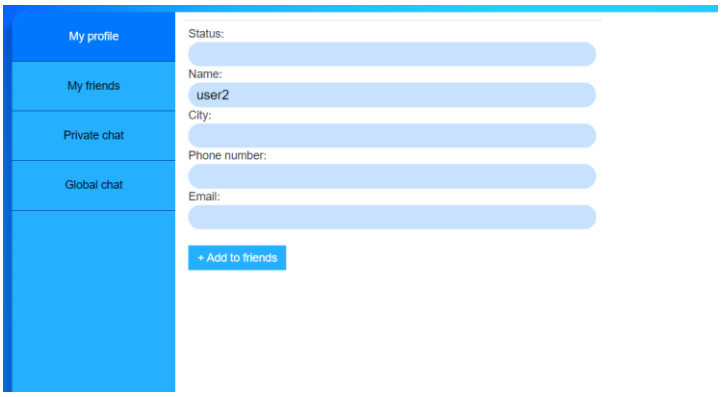


Рисунок 3.4 — Форма профілю вибраного користувача, який не є другом

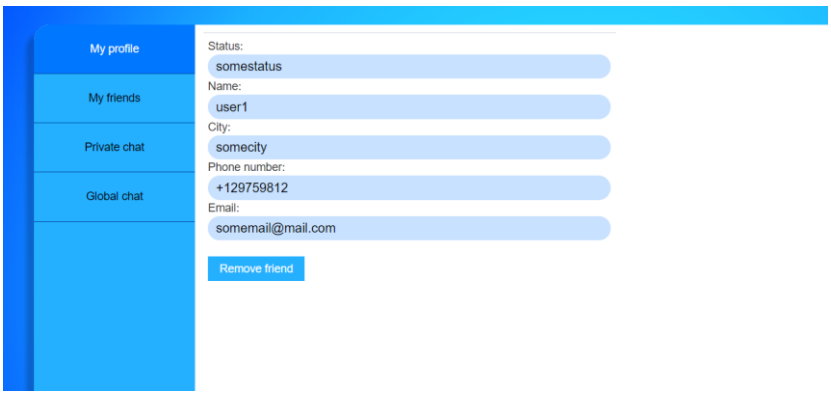


Рисунок 3.5 — Форма профілю вибраного користувача, який є в списку друзів

Форма особистого профілю містить поля для введення персональних даних. Ваш статус, ім’я, місто, номер телефону, адреса електронної пошти. Ви можете змінити інформацію, але ви повинні натиснути кнопку «Зберегти зміни», щоб зберегти її (рисунок 3.6).

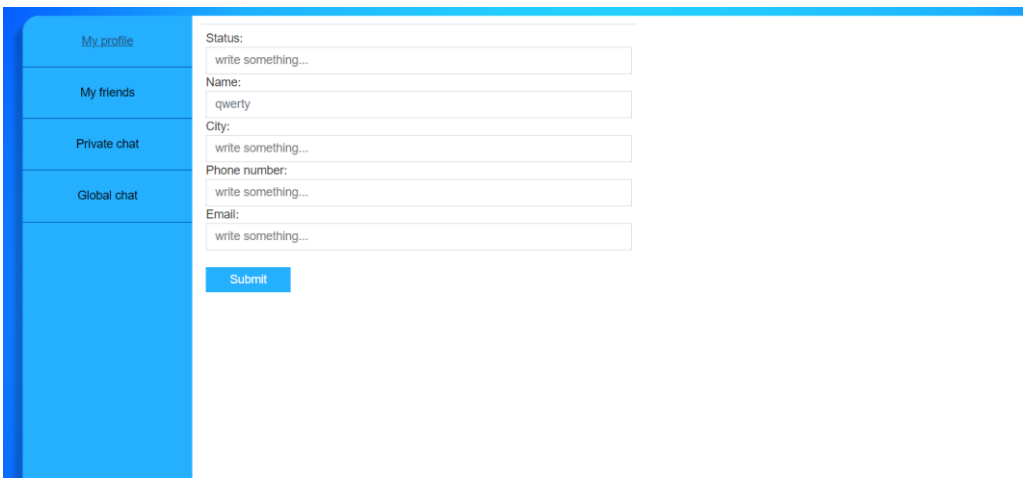


Рисунок 3.6 — Форма власного профілю зі змогою його редагування

3.5 Дослідження форми списку друзів

Користувач повинен бути в списку друзів, щоб надіслати приватне повідомлення. Наприклад, розглянемо нового користувача, у якого ще немає друзів. Щоб надіслати повідомлення, потрібно вибрати користувача зі списку та натиснути «Додати до друга». Вибраний користувач з’явиться у вашому списку друзів. Відкрийте форму списку друзів (рисунок 3.7).

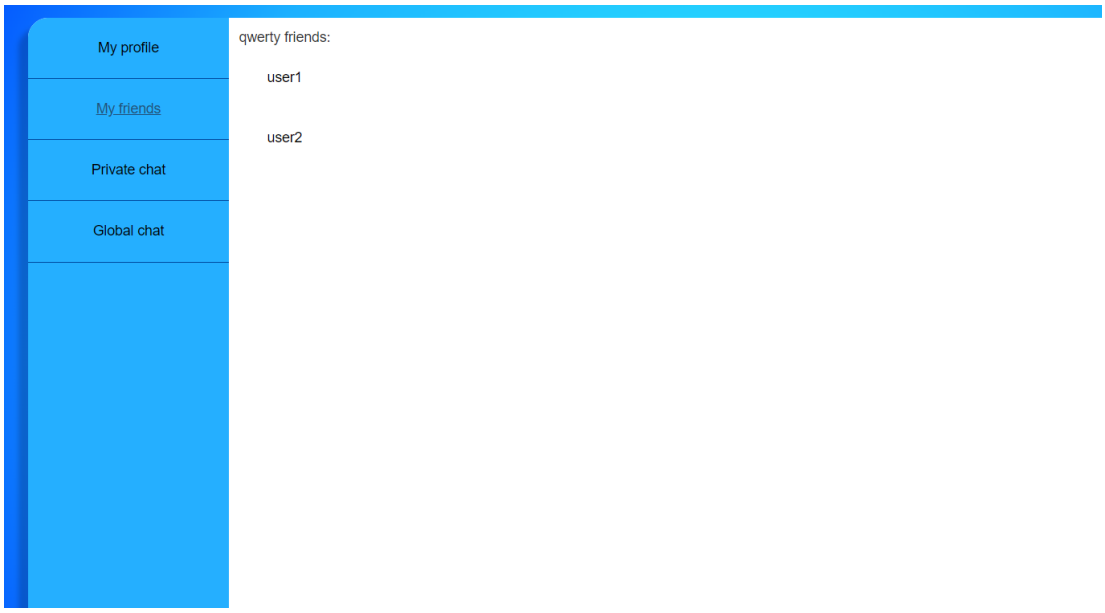


Рисунок 3.7 — Форма списку друзів

Поточний користувач може вибрати іншого користувача зі списку. Клацнувши користувача зі списку, поточний користувач перейде до профілю вибраного користувача. Існує схожий список для відправки повідомлень. Вам потрібно вибрати Приватний чат на панелі навігації. Відобразиться список користувачів, які є друзями (рисунок. 3.8).

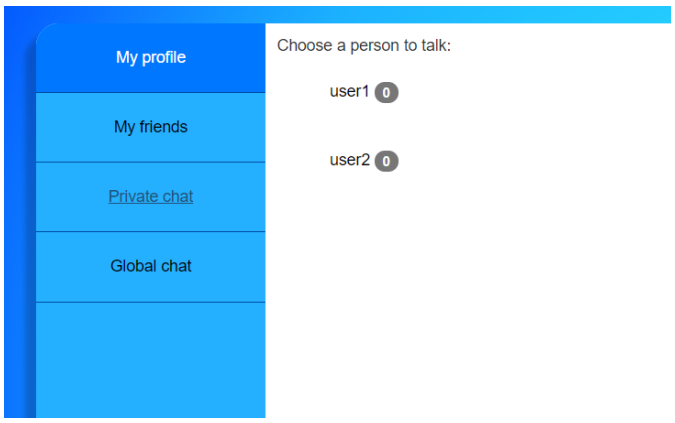


Рисунок 3.8 — Форма списку користувачів для приватного чату

Вибір користувача з цього списку призведе поточного користувача до форми приватного чату, де відображатимуться лише повідомлення поточного відомого користувача (рисунок 3.9).

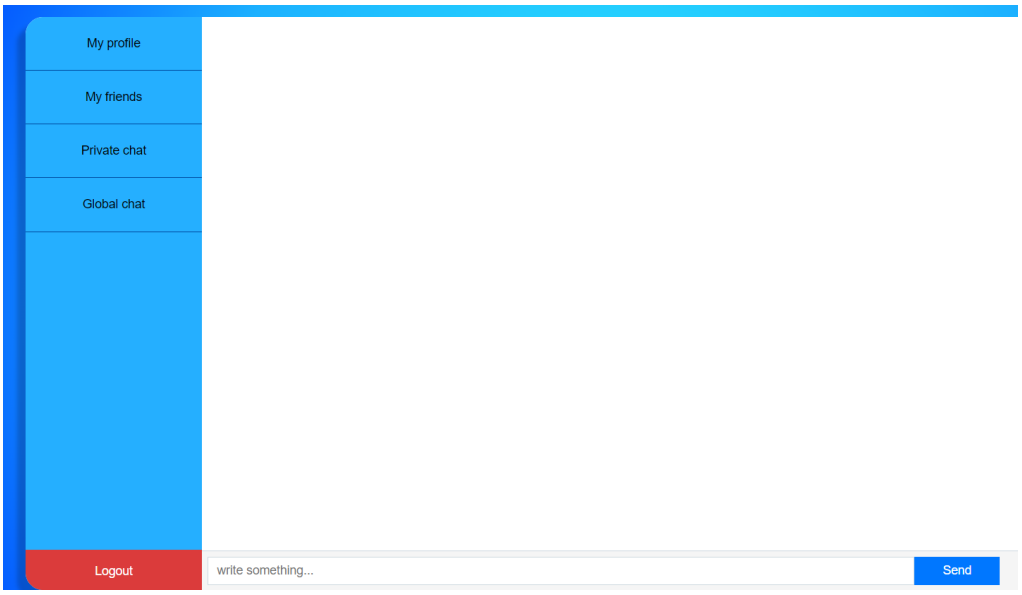


Рисунок 3.8 – Форма приватного чату з повідомленнями

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ СТВОРЕННЯ ВЕБЧАТУ ПІДВИЩЕННОЇ ШВИДКОДІЇ З МОЖЛИВІСТЮ УНІВЕРСАЛЬНОГО РОЗГОРТУВАННЯ

Дослідження завжди дорогі. Ці витрати на виробництво та реалізацію товарів необхідно постійно зменшувати, оскільки це прогрес будь-якого виробництва. На основі економічних розрахунків можна продемонструвати рентабельність та ефективність впровадження результатів досліджень у виробництво, тобто комерціалізація наукових досліджень. Дана магістерська кваліфікаційна робота відноситься до розряду прикладної науково-технічної роботи. Прогнозується виведення науково-технічної розробки на ринок із залученням потенційним інвестора. Дана послідовність приведена на рисунку 4.1.

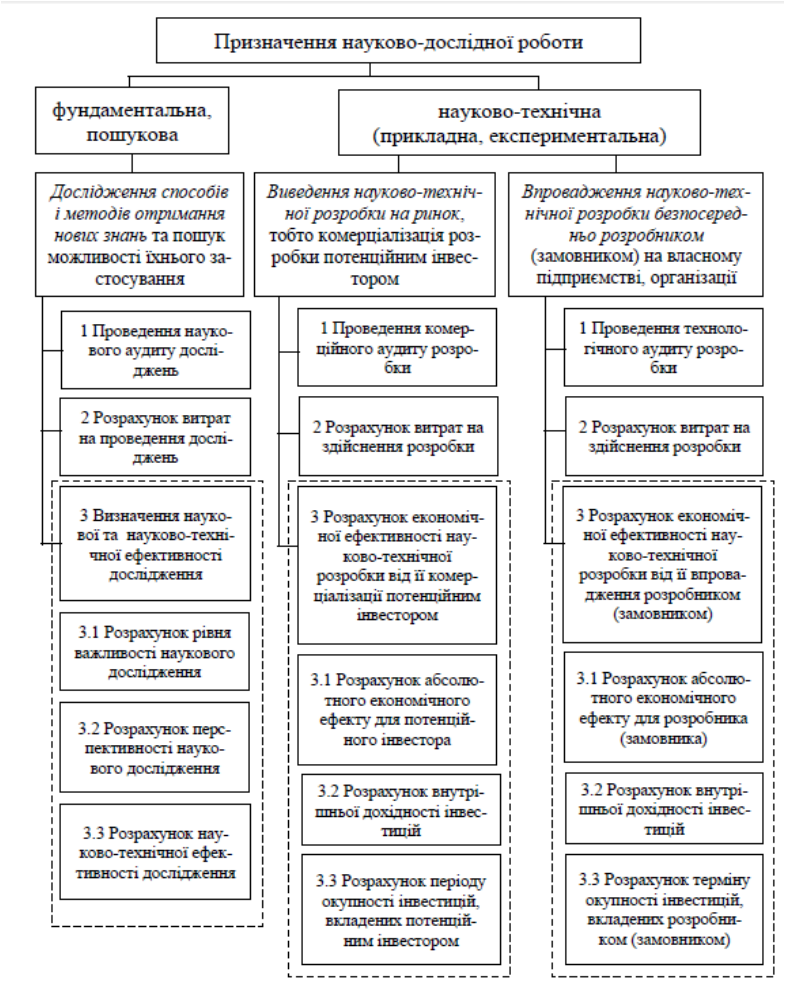


Рисунок 4.1 — Складові економічної частини магістерської кваліфікаційної роботи

Економічна частина цієї магістерської роботи буде поділена на такі елементи. Усі подальші економічні розрахунки будуть розглянуті у згаданих розділах економічної частини.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою оцінки потенціалу комерційного розвитку є оцінка потенціалу комерційного розвитку, що випливає з науково-технічних досліджень. За результатами оцінки робляться висновки про напрямки (особливості) організації в майбутньому її впровадження з урахуванням встановленої оцінки.

Комерційний потенціал інвестицій буде оцінюватись відповідно до дванадцяти критеріїв, наведених у таблиці 4.1.

Таблиця 4.1 — Оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Багато аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
Практична здійсненність					
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси.Джерела фінансування є	Потрібні незначніфінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років.Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років.Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності інвестицій менше 3-х років

Закінчення таблиці 4.1

12	Необхідно регламентні документи та велика кількість дозвільних документів на виробництво продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

На основі таблиці різні експерти, у нашому випадку керівник магістерської роботи та інші незалежні експерти з різних профілів програмування, визначають різні результати. Результати цієї оцінки комерційного потенціалу узагальнено у таблиці 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	1 Олексіюк А.П., Технічний директор ElifTech Vinnytsia	2 Крупельницький Л.В., к.т.н., доц. кафедри ОТ	3 Черняк О.І., к.т.н., доц. кафедри ОТ
	Бали:		
1.Технічна здійсненність концепції	3	2	3
2.Ринкові переваги (наявність аналогів)	3	2	3
3. Ринкові переваги (ціна продукту)	2	3	3
4.Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	3	3
6. Ринкові перспективи (розмір ринку)	2	3	3
7. Ринкові перспективи (конкуренція)	2	3	3
8. Практична здійсненність (наявність фахівців)	3	2	2
9. Практична здійсненність (наявність фінансів)	2	2	3

Продовження таблиці 4.2

10.Практична здійсненність (необхідність нових матеріалів)	1	2	2
11. Практична здійсненність (термін реалізації)	1	2	3
12. Практична здійсненність (розробка документів)	2	3	3
Сума балів	$СБ_1 = 26$	$СБ_2 = 30$	$СБ_3 = 34$
Середньо-арифметична сума балів $СБ_С$	$СБ_С = \frac{\sum_1^3 СБ_i}{3} = \frac{26 + 30 + 34}{3} = 30$		

Відповідно до таблиці 4.2, а також відповідно до рекомендацій, наведених у таблиці 4.3, можна зробити висновок про рівень потенціалу комерційного розвитку.

Таблиця 4.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 — 10	Низький
11 — 20	Нижче середнього
21 — 30	Середній
31 — 40	Вище середнього
41 — 48	Високий

З урахуванням середніх арифметичних балів $СБ_С$ = 30, які були визначені експертами, можна зробити висновок, що рівень комерційного потенціалу цієї розробки буде середній.

Програмний ресурс Viber було використано для порівняння властивостей. Це програмне забезпечення має більш широкий спектр застосування та дещо вищий відсоток злагодженої роботи.

Нова розробка, навпаки, є вузькоспрямованою і тому має більшу швидкість. Окрім того, інтерфейс не перевантажений зайвою інформацією, що значно полегшує роботу, а також має велику швидкодію і легке оновлення функціоналу. Також, ще один аналог розробленого програмного забезпечення

— Telegram, який призначений для обмінну повідомленнями в мережі інтернет.

Порівння розробки з її аналогами приведено в таблиці 4.4.

Таблиця 4.4 — Порівняння характеристик розробки із аналогом

Показники	Розробка	Аналог1	Аналог2
Функціонал	7	9	9
Швидкодія	9	7	9
Надійність	9	8	8
Метод розповсюдження	7	7	6
Інтерфейс, простота використання	8	8	7

Продукт буде просуватися за допомогою реклами в соціальних мережах, пошукових системах та багатьох інших джерелах Інтернету. Використовуючи аналітику цих сервісів, можна буде націлити рекламу на цільову групу захисників інформації. Продукт може бути використаний в мережі інтернет для швидкого і зручного обміну повідомленнями.

Новизна дослідження полягає в тому, що вперше буде запропоновано та впроваджено вебчат підвищенної швидкодії з можливістю універсального розгортування. Виходячи з результатів цього порівняння, можна з упевненістю сказати, що новий дизайн є конкурентоспроможним, оскільки в деяких аспектах в ньому переважає один з найкращих аналогів на ринку.

Даний рівень було досягнуто за рахунок покращення та/або розширення функціональних можливостей нової науково-технічної розробки порівняно з аналогічними розробками, існуючими в цей час на ринку.

4.3 Розрахунок витрат на здійснення науково-дослідної роботи

У магістерській роботі розглядається програмне забезпечення для зручного та швидкого листування в мережі інтернет, тому значну частину витрат складають витрати на розробку, а не на виробництво та відтворення. Відповідно, є певна специфіка розрахунків.

4.3.1 Витрати на оплату праці

Основна заробітна плата розробників, що працюють над проектом, визначена у формулі (4.1):

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

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

$М_{ni}$ — місячний посадовий оклад конкретного дослідника, грн;

$Т_р$ — середня кількість робочих днів в місяці, $Т_р$ — 22 дні;

t_i— кількість днів роботи конкретного дослідника, днів.

Над створенням розробки працював менеджер проекту та інженер програмного забезпечення, тому ми виконаємо для них усі необхідні розрахунки, і вносимо їх до таблиці 4.5.

$$Зо.к. = \frac{25000 \cdot 10}{22} = 11363{,}63(грн).$$

$$Зо.в. = \frac{20000 \cdot 44}{22} = 40000\ (грн).$$

Таблиця 4.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	25000	1136,36	10	11363,6
Старший інженер-програміст	20000	909,09	44	39999,96
Всього				51363,56

Витрати на основну заробітну плату робітників за відповідними найменуваннями робіт відсутні, тобто $З_р = 0$. Додаткова винагорода ($З_{дод.}$) усіх розробників та працівників, які брали участь у цьому етапі роботи,

обчислюється як 10 ... 12% від суми основної заробітної плати дослідників та робітників за формулою (4.2):

$$З_{дод.} = (З_о + З_р) \cdot \frac{Н_{дод.}}{100\%}, \quad (4.2)$$

де $Н_{дод.}$ — норма нарахування додаткової заробітної плати.

$$З_{дод.к.} = \frac{10 \cdot 11363,6}{100} = 1136,36 \text{ (грн)},$$

$$З_{дод.в.} = \frac{10 \cdot 39\,999,96}{100} = 3999,99 \text{ (грн)},$$

$$З_{дод} = З_{дод.к.} + З_{дод.в.} = 5136,35 \text{(грн)}.$$

4.3.2 Відрахування на соціальні заходи

Заробітна плата робітників відсутня, тому $З_р = 0$. Нарахування на заробітну плату дослідників та нарахування на заробітну плату працівників, які брали участь у цьому етапі роботи, розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою (4.3)

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.3)$$

де $Н_{зп}$ — норма нарахування на заробітну плату.

$$З_н = (51\,363,56 + 0 + 5\,136,35) \cdot \frac{22\%}{100\%} = 12495,98 \text{ (грн.)}$$

4.3.3 Сировина та матеріали

Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою (4.4):

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.4)$$

$Н_j$ — кількість матеріалу j-го виду, шт.;

n — кількість видів матеріалу.

$Ц_j$ — ціна матеріалу j-го виду, грн;

$К_j$ — коефіцієнт транспортних витрат, $К_j$ = (1,1…1,15), обираємо $К_j$ 1,15;

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.

Результати розрахунків занесено до таблиці 4.6.

Таблиця 4.6 — Витрати на матеріали

Найменування комплектуючих	Ціна за 1 штуку, грн	Кількість матеріалу, штук	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн.
Ручка	30,00	1	0,06	1,80	29,89
Карта пам’яті	850,00	1	0	0,00	850,00
Пачка офісного папіру	115,00	1	0,5	57,50	86,25
Всього (з урахуванням транспортних витрат)					1336,06

4.3.4 Розрахунок витрат на комплектуючі

Оскільки кінцевий продукт, який ми створюємо — це програмний інструмент, це не спричиняє жодних витрат на компоненти та $К_в = 0$.

4.3.5 Спецустаткування для наукових (експериментальних) робіт

Спецустаткування для проведення експериментальних робіт по створенню програмного продукту вебчату підвищенної швидкодії з можливістю універсального розгортування не має потреби залучати.

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Програмне забезпечення для створення програмного продукту вебчату підвищенної швидкодії з можливістю універсального розгортування

використовується таке, що є у вільному розповсюдженні, тому витрати на придбання такого забезпечення відсутні. Це мова програмування JavaScript та програмні продукти та бібліотеки із відкритим кодом Vue.js, Node.JS, MongoDB, Dockerм.

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою (4.5):

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12,} \quad (4.5)$$

Таблиця 4.7 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців.	Амортизаційні відрахування, грн
ЕОМ	15000	2	2	1250
Приміщення	190000	20	2	1583,32
Всього				2833,32

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) розраховують за формулою (4.6):

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.6)$$

W_{yi} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), $Ц_е = 4{,}62$ [28];

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі}$ <1; обираємо $К_{впі}$= 0,7;

η_i — коефіцієнт корисної дії обладнання, η_i<1.

$$В_е = \sum_{i=1}^{1} \frac{0{,}6 \cdot 352 \cdot 4{,}62 \cdot 0{,}7}{0{,}8} = 853{,}77 \text{ (грн.)},$$

Проведені розрахунки необхідно звести до таблиці 4.8.

Таблиця 4.8 — Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ЕОМ	0,6	352	853,77
Всього			853,77

4.3.9 Службові відрядження

Під час розробки програмного забезпечення відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, магістрів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень, не плануються.

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

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» не плануються, так як у цьому не має потреби.

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою (4.7):

$$I_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} \quad (4.7)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$I_{в.к.} = 11363{,}6 \cdot \frac{50\%}{100\%} = 5681{,}8(грн.),$$

$$I_{в.в.} = 39999{,}96 \cdot \frac{50\%}{100\%} = 19999{,}98\ (грн.),$$

$$I_в = I_{в.к.} + I_{в.в.} = 25681{,}78\ (грн.).$$

4.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників [29] та робітників за формулою (4.8):

$$В_{нзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} \quad (4.8)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Беремо норму нарахування 100%.

$$В_{нзв.к.} = 11363{,}6 \cdot \frac{100\%}{100\%} = 11363{,}6(грн.),$$

$$В_{нзв.в.} = 39999{,}96 \cdot \frac{100\%}{100\%} = 39999{,}96(грн.),$$

$$В_{нзв} = В_{нзв.к.} + В_{нзв.в.} = 51363{,}56\ (грн.).$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою (4.9):

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.9)$$

У нашому випадку:

$З_р = 0, К_в = 0, В_{спец} = 0, В_{прг} = 0, В_{св} = 0, В_{сп} = 0$, тому отримаємо:

$$В_{заг} = З_о + З_{дод} + З_н + М + А_{обл} + В_е + І_в + В_{нзв}$$
$$= 51\,363{,}56 + 5\,136{,}35 + 12\,495{,}98 + 2\,836{,}06 + 2\,833{,}32 + 853{,}77$$
$$+ 25\,681{,}78 + 51\,363{,}56 = 152564{,}38 (грн).$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою (4.10):

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.10)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, обираємо його 0,5.

$$ЗВ = \frac{152564{,}38}{0{,}5} = 305128{,}76 \text{ (грн)}.$$

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

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

Для всіх наведених випадків можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_і$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою (4.11):

$$\Delta П_і = (\pm \Delta Ц_о \cdot N \cdot Ц_о \cdot \Delta N)_і \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \quad (4.11)$$

$\pm \Delta Ц_о$— зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

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

$Ц_о$ — основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_о = Ц_б \pm \Delta Ц_о$;

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

ΔN — зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

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

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту (

п ϑ — ставка податку на прибуток, який має сплачувати потенційний і
нвестор, у 2021 році ϑ=18%.

с В результаті впровадження результатів наукових розробок п
оліпшується якість програмного забезпечення, що дозволяє подорожчати за й
ого впровадження, а кількість потенційних користувачів ресурсу збільшиться г
— у перший рік — на 200 одиниць, на другий рік — ще на 350 одиниць, на т
ретій рік — ще 800 штук.

) Ми прогнозуємо щорічний приріст чистого прибутку компанії від впровадження результатів наукових розробок щодо вихідного стану.

Збільшення чистого прибутку підприємства $\Delta П_і$ за перший рік складе:

Р
е $$\Delta П_1 = [1800 \cdot 1 + (3500 + 1800) \cdot 200] \cdot 0,8333 \cdot 0,26 \cdot \left(1 - \frac{18\%}{100\%}\right) = 188571 \text{ (грн)}.$$

к З

б
і
л
ь
ш
е

Збільшення чистого прибутку підприємства $\Delta П_i$ на третій рік складе:

$$\Delta П_3 = [1800 \cdot 1 + (3500 + 1800) \cdot (200 + 350 + 800)] \cdot 0,8333 \cdot 0,26 \cdot \left(1 - \frac{18\%}{100\%}\right) = 1271473,93 \text{ (грн)}.$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки (формула 4.12).

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.12)$$

де $\Delta П_i$— збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15, обираємо τ 0,1;

t— період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{188571}{(1+0,1)^1} + \frac{518197,40}{(1+0,1)^2} + \frac{1271473,93}{(1+0,1)^3} = 171428,18 + 428262,31 + 955277,18 = 1554967,67 \text{ (грн.)}$$

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу (4.13):

$$PV = k_{інв} \cdot ЗВ, \quad (4.13)$$

де $k_{інв}$— коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, обираємо даний коефіцієнт 2;

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

$$PV = 2 \cdot 305128{,}76 = 610257{,}52 \text{ (грн.)}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохіддля потенційного інвестора від можливого впрова-дження та комерціалізації науково-технічної розробки становитиме (формула 4.14):

$$E_{абс} = ПП - PV \qquad (4.14)$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV — теперішня вартість початкових інвестицій, грн.

$$E_{абс} = 1554967{,}67 - 610257{,}52 = 944710{,}15 \text{ (грн.)}$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою (4.15):

$$E_в = \sqrt[т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.15)$$

$E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$Т_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[з]{1 + \frac{944710{,}15}{610257{,}52}} - 1 = 0{,}366$$

Далі визначають бар’єрну ставку дисконтування τ_{min}, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впрова-

дження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$визначається за формулою (4.16):

$$\tau_{мін} = d + f, \quad (4.16)$$

d— середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d=0,9...0,12, обираємо d 0,1;

f — показник, що характеризує ризикованість вкладення інвестицій, зазвичай величина f=0,05...0,5, але може бути і значно вищою, обираємо 0,19.

$$\tau_{мін} = d + f = 0{,}1 + 0{,}19 = 0{,}29\%$$

Величина $E_в > \tau_{мін}$, отже інвестор може бути зацікавлений у фінансуванні цього дослідження.

Далі розраховуємо період окупності інвестицій $T_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки (формула 4.17):

$$T_{ок} = \frac{1}{E_в}, \quad (4.17)$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0{,}366} = 2{,}73 \text{ року}$$

Оскільки $T_{ок}$= 2,73 року тоді розвиток доречний.

ВИСНОВКИ

В даній магістерській роботі було створено швидкий чат. Продуктивність досягається за допомогою Vue.js, Websocket і швидкої бази даних MongoDB, а універсальність розгортання досягається завдяки контейнеризації додатків за допомогою інструментів Docker.

Основною роботою при розробці цього проекту було початкове створення проекту в Visual Studio Code. Для розробки цього програмного забезпечення була розроблена повноцінна архітектура проекту. JavaScript був успішною мовою програмування завдяки великій кількості відкритих бібліотек, які пропонують широкий спектр можливостей у сфері веб-розробки.

Усі бур’яни, з якими потрібно взаємодіяти користувачам, були створені, стилізовані, а серверна та клієнтська частини були розроблені. В результаті ви можете зареєструватися та відправляти повідомлення як у глобальний чат, так і особисто обраним користувачам, вводити інформацію про себе у свій профіль, переглядати список усіх користувачів та їхні профілі Реалізовано веб-чат. Ви також можете розгорнути програму в будь-якому середовищі.

Отже, мета була досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. CSS. URL [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/CSS/.

2. CSS документація [Електронний ресурс]. Режим доступу: https://www.w3schools.com/css/.

3. HTML [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/HTML.

4. HTML документація [Електронний ресурс]. Режим доступу: https://www.w3schools.com/html/default.asp.

5. JavaScript [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/JavaScript.

6. JavaScript документація [Електронний ресурс]. Режим доступу: https://javascript.info/.

7. Vue.js. [Електронний ресурс]. Режим доступу: https://vuejs.org/.

8. Vue.js документація [Електронний ресурс]. Режим доступу: https://vuejs.org/.

9. React.js. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/React.

10. Angular.js. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/AngularJS.

11. SPA. [Електронний ресурс]. Режим доступу: https://amp.uk.media-inform.com/1672950/1/odnostorinkoviy-zastosunok.html.

12. Node.js. [Електронний ресурс]. Режим доступу: https://nodejs.org/uk/.

13. Node.js документація. [Електронний ресурс]. Режим доступу: https://nodejs.org/uk/.

14. Websocet. [Електронний ресурс]. Режим доступу:: https://uk.wikipedia.org/wiki/WebSocket.

15. Websocet документація. [Електронний ресурс]. Режим доступу:: https://learn.javascript.ru/websockets.

16. MongoDB. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/MongoDB.

17. MongoDB документація. [Електронний ресурс]. Режим доступу:: https://www.mongodb.com/.

18. Docker. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Docker.

19. Docker документація. [Електронний ресурс]. Режим доступу:: https://www.docker.com/.

20. Cookies. [Електронний ресурс]. Режим доступу: https://ssl.com.ua/blog/what-are-cookies/.

21. Express.js [Електронний ресурс]. Режим доступу: https://expressjs.com/ru/.

22. PHP. [Електронний ресурс]. Режим доступу: https://www.php.net/manual/ru/intro-whatis.php.

23. Python. [Електронний ресурс]. Режим доступу: https://www.python.org/.

24. C++. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/C%2B%2B.

25. Java. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Java.

26. Qt. [Електронний ресурс]. Режим доступу: https://www.qt.io/.

27. Visual Studio Code [Електронний ресурс]. Режим доступу: https://code.visualstudio.com/.

28. Тарифи на електроенергію [Електронний ресурс]. Режим доступу: http://index.minfin.com.ua/tarif/electric.php. Дата звернення: Листопад 30, 2021.

29. Кавецький В. В. Економічне обґрунтування інноваційних рішень: Практикум /В.В.Кавецький, В.О.Козловський, І.В.Причепа. — ВНТУ, 2013. — 110 с.

30. Черненко Д.Ю. Швидкодіючий веб-чат з можливістю універсального розгортування [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/author/submission/14289

ДОДАТОК А

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
_____проф., д.т.н. О. Д. Азаров
«___» _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Вебчат підвищенної швидкодії з можливістю універсального розгортування»
08-23.МКР.027.00.000 ТЗ

Науковий керівник: д.т.н., професор
_____ Азаров О.Д
Магістрант групи 2КІ-20м
_____ Черненко Д.Ю

Вінниця 2021

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність даного дослідження пов’язана з необхідністю вирішення ряду задач обміну повідомленнями в мережі Інтернет. Зокрема, забезпечення зручності користувацького досвіду завдяки швидкій взаємодії з чатом та зручність розгортування додатку в будь-якому середовищі для розробників.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета магістерської роботи полягає в підвищенні швидкодії чату для покращення досвіду користування та реалізація універсальності розгортування для зручності розміщення проекту на серверному комп’ютері.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані для виконання МКР

В даній кваліфікаційній роботі переслідується вирішення задачі створення веб-додатку який дозволить отримувати зручніший досвід користування, за рахунок підвищеня швидкодії та підвищить зручність розробки за рахунок змоги розгорнути додаток в будь-якій операційній системі.

Потрібно реалізувати веб-додаток який буде швидко відправляти повідомлення, швидко реагувати на дії користувача та розгортуватись в будь-якій операційній системі.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— можливість авторизації та реєстрації;
— обмін повідомленнь між користувачами;
— перегляд інформації про користувачів;

— перегляд списку користувачів;

— виведення тільки тих данних, які потрібно бачити користувачу, хешування паролів.

5 Етапи МКР та очікувані результати таблиця А.1

Таблиця А.1 — Етапи виконання роботи

№	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ			Вступ
2	Інформаційний пошук та огляд літературних джерел			Розділ 1
3	Розробка програмного засобу			Розділ 2
4	Опис результатів роботи ПЗ			Розділ 3
7	Оформлення пояснювальної записки			ПЗ, презентація

6 Матеріали, що подаються до захисту МКР

Пояснювальна записка, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів розрахункової та графічної документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

Вимоги викладені в ДСТУ 3008:2015 та положенні ВНТУ про МКР-2021.

Технічне завдання до виконання отримав _____ Черненко Д. Ю.

ДОДАТОК Б

Лістинг програми

server.js

```
const express = require('express');
const mongo = require('mongodb');
const cookieParser = require('cookie-parser');
const { v4: uuidv4 } = require('uuid');
const app = express();
const expressWs = require('express-ws')(app);



const port = 3000;
const dbUrl = 'mongodb://zz28:123456z@ds233238.mlab.com:33238/fine-chat'
const dbName = 'fine-chat';
const frontendUrl = 'http://localhost:8080';
let sessions = [];

app.use(express.json());
app.use(cookieParser());



mongo.MongoClient.connect(dbUrl, (err, client) => {
  if (err) throw err;

  const db = client.db(dbName);
  console.log('Connected to MongoDB: ' + dbUrl + '/' + dbName);

  app.use(function(req, res, next) {
    res.header('Access-Control-Allow-Origin', frontendUrl);
    res.header('Access-Control-Allow-Credentials', true);
```

```
  res.header('Access-Control-Allow-Methods', 'HEAD, OPTIONS, GET, POST, PUT, DELETE');
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');
  next();
 });

 app.ws('/chat/:id', (ws, req) => {
  ws.uuid = uuidv4();
  ws.userId = req.params.id;
  sessions.push(ws);
  console.log('Connected:', ws.uuid, ', userId:', ws.userId);

  ws.on('message', async (msg) => {
   let data = JSON.parse(msg);
   console.log(data);

   var result = await db.collection('users').find().toArray();
   var userNames = {};
   result.forEach(u => userNames[u._id] = u.name);

   if (data.type === 'public') {
    // Save to DB
    var message = {
     type: data.type,
     from: data.from,
     fromName: userNames[data.from],
     text: data.text,
    };
    await db.collection('messages').insertOne(message);
    // Send to everyone, including the sender
```

```
      sessions.forEach(w => w.send(JSON.stringify(message)));

    } else if (data.type === 'private') {
      // Save to DB
      var message = {
        type: data.type,
        from: data.from,
        fromName: userNames[data.from],
        to: data.to,
        toName: userNames[data.to],
        text: data.text,
      };
      await db.collection('private-messages').insertOne(message);
      // Send to recipient and sender only
      sessions.filter(w => w.userId === data.from || w.userId === data.to)
        .forEach(w => w.send(JSON.stringify(message)));
    }
  });

  ws.on('close', () => {
    sessions = sessions.filter(e => e !== ws);
    console.log('Disconnected', ws.uuid);
  });
});

// Register http://localhost:3000/register/
app.post('/register', async (req, res) => {
  var user = await db.collection('users').findOne({
    login: req.body.login,
  });
```

```
  if (user !== null) {
    res.status(500).json({message:"already exist"});
  } else {
    newToken = uuidv4();

    var newUser = {
      login: req.body.login,
      password: req.body.password,
      token: newToken,
      name: req.body.login,
      active: true,
      status: '',
      city: '',
      phone: '',
      email: '',
      friends: []
    };

    var result = await db.collection('users').insertOne(newUser);
    var user = result.ops[0];
    res.json({
      id: user._id,
      token: newToken
    });
  }
});

// Login http://localhost:3000/login/
app.post('/login', async (req, res) => {
  var user = await db.collection('users').findOne({
    login: req.body.login,
```

```
      password: req.body.password,
    });
console.log("login", user);

    if (user === null) {
      res.status(500).json({});
    } else {
      newToken = uuidv4();
      user.token = newToken;

      await db.collection('users').replaceOne(
        { _id: user._id },
        user
      );
      res.json({
        id: user._id,
        token: newToken
      });
    }
  });

  // Get all users: http://localhost:3000/users/
  app.get('/users', async (req, res) => {
    var result = await db.collection('users').find({ active: true }).toArray();
    var users = result.map(u => {
      return {
        id: u._id,
        name: u.name
      };
    });
    res.json(users);
```

```
});

// Get one user by ID: http://localhost:3000/users/5eca6d7ac4af8618503b323e
app.get('/users/:id', async (req, res) => {
  var result = await db.collection('users').find().toArray();
  var userNames = {};
  result.forEach(u => userNames[u._id] = u.name);

  var userId = new mongo.ObjectID(req.params.id);
  var result = await db.collection('users').findOne({ '_id': userId });
  var user = {
    id: result._id,
    login: result.login,
    name: result.name,
    active: result.active,
    status: result.status,
    city: result.city,
    phone: result.phone,
    email: result.email,
    friends: result.friends.map(u => {
      return {
        id: u.id,
        name: userNames[u.id]
      };
    }),
  };
  res.json(user);
});

// Update user by ID: http://localhost:3000/users/5eca6d7ac4af8618503b323e
app.post('/users/:id', async (req, res) => {
```

```
  var userId = new mongo.ObjectID(req.params.id);
  var userToken = req.cookies.userToken;
  var user = {
    name: req.body.name,
    active: req.body.active,
    status: req.body.status,
    city: req.body.city,
    phone: req.body.phone,
    email: req.body.email,
    friends: req.body.friends,
  };

  var result = await db.collection('users').findOneAndUpdate(
    { _id: userId, token: userToken },
    { $set: user },
    { upsert: true });
  res.json(result);
});

// Get all global messages
app.get('/messages', async (req, res) => {
  var users = await db.collection('users').find().toArray();
  var result = await db.collection('messages').find().sort({$natural:-
1}).limit(5).toArray();
  result.reverse();

  result.forEach((m) => {
    var fromUser = users.find(u => u._id == m.from);
    m.fromName = fromUser == null ? 'Unknown' : fromUser.name;
  });
  res.json(result);
```

```
  });

  // Get messages from User 1 to User 2 and vise-versa
  app.get('/messages/:id1/:id2', async (req, res) => {
    var users = await db.collection('users').find().toArray();
    var result = await db.collection('private-messages').find(
      { '$or': [{from: req.params.id1, to: req.params.id2}, {from: req.params.id2, to:
req.params.id1} ]}
    ).toArray();

    result.forEach((message) => {
      var fromUser = users.find(u => u._id == message.from);
      message.fromName = fromUser == null ? 'Unknown' : fromUser.name;
    });

    res.json(result);
  });

  // Post a new message to global chat
  app.post('/messages/:id', async (req, res) => {
    var message = {
      from: req.params.id,
      text: req.body.text,
    };
    var result = await db.collection('messages').insertOne(message);
    res.json(result.ops[0]);
  });

  // Post a new private message
  app.post('/messages/:id1/:id2', async (req, res) => {
    var message = {
```

```
    from: req.params.id1,
    to: req.params.id2,
    text: req.body.text,
  };
  var result = await db.collection('private-messages').insertOne(message);
  res.json(result.ops[0]);
 });


 app.listen(port, () => {
  console.log('Listening on http://localhost:' + port + '/');
 });
});
```

App.vue

```
<template>
  <div id="app">
    <div id="chatForm">
      <form action="" method="post">
        <div class="chat-window">

          <div class="nav-panel">
            <ul class="">
              <!-- TODO: add counter to unread msgs -->
              <li class="item" v-on:click="dataUri = 'global_chat'"><a href="#"><span>Global chat</span></a></li>
              <li class="item" v-on:click="dataUri = 'my_profile'"><a href="#"><span>My profile</span></a></li>
              <li class="item" v-on:click="dataUri = 'my_friends'"><a href="#"><span>My friends</span></a></li>
```

```
                    <li class="item" v-on:click="dataUri = 'private_chat'"><a href="#"><span>Private chat</span></a></li>
                    <input type="button" value="Logout" class="logout-btn" v-on:click="logout()">
                </ul>
            </div>
            <div class="chat">
                <div v-if="dataUri === 'private_chat'">
                    <div class="friend-window">
                    Choose a person to talk:
                    <ul>
                      <li clas v-for="item in user.friends" :key="item.id">
                        <a v-on:click="openPrivateChat(item)" class="friedns-list" href="#"><span>{{ item.name }}</span></a>
                      </li>
                    </ul>
                </div>
            </div>
            <div v-if="dataUri === 'private_chat_messages'">
                <!-- TODO: ADD PRIVATE CHAT -->
                <div class="chat-list">
                    <li v-for="item in private_messages[someone.id]" :key="item.id">
                        <b>{{ item.fromName }}:</b> {{ item.text }}
                    </li>
                </div>
                <div class="input-area">
                <div class="input-wrapper"><input v-model="privateChatText" type="textarea" value="" placeholder="write something..."></div>
                <input v-on:click="sendPrivateMessage(privateChatText)" type="button" value="Send" class="send-btn">
```

```
                </div>
            </div>
            <div v-if="dataUri === 'global_chat'">
                <div class="chat-list">
                    <li v-for="item in messages" :key="item._id">
                    <b>{{ item.fromName }}:</b> {{ item.text }}
                    </li>
                </div>
                <div class="input-area">
                <div class="input-wrapper"><input v-model="globalChatText" type="textarea" value="" placeholder="write something..."></div>
                <input v-on:click="sendMessage(globalChatText)" type="button" value="Send" class="send-btn">
                </div>
            </div>
            <div v-if="dataUri === 'my_profile'">
              <div class="profile-window">
                <div class="input-area-profile">
                    <div><span class="span-profile">Status</span></div>
                        <div class="input-status">
                            <div class="input-wrapper-profile"><input v-model="user.status" type="textarea" value="" placeholder="write something..."></div>
                        </div>
                            <div><span class="span-profile">Name</span></div>
                    <div class="input-name">
                            <div class="input-wrapper-profile"><input v-model="user.name" type="textarea" value="" placeholder="write something..." ></div>
                    </div>
                            <div><span class="span-profile">City</span></div>
```

```
                    <div class="input-city">
                        <div class="input-wrapper-profile"><input v-model="user.city" type="textarea" value="" placeholder="write something..."></div>
                    </div>
                        <div><span class="span-profile">Phone number</span></div>
                    <div class="input-phone-number">
                        <div class="input-wrapper-profile"><input v-model="user.phone" type="tel" value="" placeholder="write something..."></div>
                    </div>
                        <div><span class="span-profile">Email</span></div>
                    <div class="input-email">
                        <div class="input-wrapper-profile"><input v-model="user.email" type="email" value="" placeholder="write something..."></div>
                    </div>
                    <input type="button" value="Submit" class="submit-btn-profile" v-on:click="submitUserData()" >
                </div>
            </div>
            </div>

            <div v-if="dataUri === 'my_friends'">
                <div class="friend-window">
                 {{user.name}} friends:
                  <ul>
                    <li clas v-for="item in user.friends" :key="item.id">
                       <a  v-on:click="onSomeoneClick(item)"  class="friedns-list" href="#"><span>{{ item.name }}</span></a>
                    </li>
```

```
                </ul>
              </div>
            </div>
          </div>

          <div class="online-list">
            <span class="span-online-users">Users:</span>
            <li class="online-list-item" v-on:click="onSomeoneClick(item)" v-for="item in userList" :key="item._id">
            <a class="online-list-a"><span >{{ item.name }}</span></a>
            <span v-if="friendSign(item)" class="friend-sign"> online</span>
            <!--TODO: add sign to online users -->
            </li>
          </div>
          <div v-if="dataUri === 'someone_profile'">
            <div class="someone-profile-window">
              <div class="input-area-someone-profile">
                <div><span class="span-someone-profile">Status</span></div>
                  <div class="input-status">
                    <div class="input-wrapper-someone-profile"><input v-model="someone.status" type="textarea" readonly value="" ></div>
                  </div>
                    <div><span class="span-someone-profile">Name</span></div>
                  <div class="input-name">
                    <div class="input-wrapper-someone-profile"><input v-model="someone.name" type="textarea" readonly value="" ></div>
                  </div>
                    <div><span class="span-someone-profile">City</span></div>
```

```
                    <div class="input-city">
                        <div class="input-wrapper-someone-profile"><input v-model="someone.city" type="textarea" readonly value="" ></div>
                    </div>
                        <div><span class="span-someone-profile">Phone number</span></div>
                    <div class="input-phone-number">
                        <div class="input-wrapper-someone-profile"><input v-model="someone.phone" type="tel" readonly value="" ></div>
                    </div>
                        <div><span class="span-someone-profile">Email</span></div>
                    <div class="input-email">
                        <div class="input-wrapper-someone-profile"><input v-model="someone.email" type="email" readonly value="" ></div>
                    </div>
                    <input type="button" v-if="!someone.isFriend" v-on:click="addFriend()" value="+ Add to friends" class="add-to-friends-btn">
                    <input type="button" v-if="someone.isFriend" v-on:click="removeFriend()" value="Remove friend" class="add-to-friends-btn">
                </div>
            </div>
        </div>

    </div>
  </form>
 </div>
</div>
</template>

<script>
```

```
const axios = require('axios');
const backendUrl = 'http://localhost:3000';
const backendWS = 'ws://localhost:3000';

export default {
    data: function () {
        return {
            connection: null,
            dataUri: 'global_chat',

            userId: null,
            usertToken: null,
            user: null,
            someone: {},
            friend: {},
            globalChatText: '',
            privateChatText: '',

            userList: [],
            messages: [],
            private_messages: {},
        };
    },

    created() {
        this.userId = $cookies.get('userId');
        this.usertToken = $cookies.get('userToken');

        if (this.usertToken === null) {
            window.location.href = '/loginForm.html';
        } else {
```

```
            this.connection = new WebSocket(`${backendWS}/chat/${this.userId}`);
            this.connection.onmessage = (event) => {
                this.onMessageReceived(JSON.parse(event.data));
            };
        }
    },
    mounted() {
        axios.defaults.withCredentials = true;
        axios.get(`${backendUrl}/users/`)
            .then(response => { this.userList = response.data });
        axios.get(`${backendUrl}/messages/`)
            .then(response => { this.messages = response.data });
        axios.get(`${backendUrl}/users/${this.userId}`)
            .then(response => { this.user = response.data });
        //
axios.get(`${backendUrl}/messages/5eca6d7ac4af8618503b323e/5eca7240b50de6
06c084494b`)
        //     .then(response => {this.private_messages = response.data});
    },
    methods: {

        logout() {
            $cookies.remove('userToken');
            window.location.href = '/loginForm.html';
        },
        submitUserData() {
            axios.post(`${backendUrl}/users/${this.userId}`, this.user);
        },
        onSomeoneClick(someone) {
            axios.get(`http://localhost:3000/users/${someone.id}`)
                .then(response => {
```

```
                this.someone = response.data;
                this.someone.isFriend = this.user.friends.some(e => e.id === someone.id);
                this.dataUri = 'someone_profile';
            });
        },
        addFriend(){
            this.user.friends.push({
                id: this.someone.id,
                name: this.someone.name
            });
            this.submitUserData();
            this.someone.isFriend = true;
            let zz = this.someone;
            this.someone = {};
            this.someone = zz;
        },
        removeFriend(){
            this.user.friends = this.user.friends.filter(e => e.id !== this.someone.id);
            this.submitUserData();
            this.someone.isFriend = false;
            let zz = this.someone;
            this.someone = {};
            this.someone = zz;
        },
        friendSign(testUser){
            return this.user.friends.some(e => e.id === testUser.id);
        },
        openPrivateChat(testUser) {
            this.someone = testUser;
            this.dataUri = 'private_chat_messages';
```

```
},
sendMessage(text) {
    this.connection.send(JSON.stringify({
        from: this.userId,
        type: 'public',
        text: text,
    }));
},
sendPrivateMessage(text) {
    this.connection.send(JSON.stringify({
        from: this.userId,
        to: this.someone.id,
        type: 'private',
        text: text,
    }));
},
onMessageReceived(data) {
    if (data.type === 'public') {
        console.log('New public message from:', data.from, ', data:', data);
        this.messages.push(data);
        // TODO: DO we need this?
        if (this.messages.length > 10) {
            this.messages = this.messages.slice(-10);
        }
    } else if (data.type === 'private') {
        let friend = data.from === this.userId ? data.to : data.from;
        if (!this.private_messages[friend]) {
            this.private_messages[friend] = [];
        }
        console.log('New private message from:', friend, ', data:', data);
        this.private_messages[friend].push(data);
```

```
                let zz = this.private_messages;
                this.private_messages = {};
                this.private_messages = zz;
            }
        },
    },
};
</script>
<style>
    @import './App.css';
</style>
```

loginForm.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Login</title>
    <link href="style.css" rel="stylesheet">
    <script src="https://unpkg.com/vue"></script>
    <script src="https://unpkg.com/vue-cookies@1.7.0/vue-cookies.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>
<body>
    <div id="app">
        <div id="action_form">
            <div class="container">
                <label><b>Your name</b></label>
                <div><input id="username" v-model="login" placeholder="Enter username" tabindex="1" type="text"></div>
```

```
            <label><b>Your password</b></label>
            <div><input id="password" v-model="password" placeholder="Enter password" tabindex="2" type="password"></div>
            <div><label><input type="checkbox" checked="checked" name="remember"> <span class="lgn">Remember me</span></label></div>
            <div><button v-on:click="onLogin()" tabindex="3" class="lgn">Sign-in</button></div>
            <div><button v-on:click="onRegister()" tabindex="4" class="lgn">Create your account</button></div>
        </div>
    </div>
</div>
<script>
    const charMatch = new RegExp('^[a-zA-Z_0-9]*$');
    // Vue.use(VueCookies)
    var app = new Vue({
        el: '#app',
        data: {
            login: "",
            password: "",
        },
        methods: {
            onLogin() {
                if (this.validateLogin(this.login, this.password) === true) {
                    axios.post('http://localhost:3000/login/', {
                        login: this.login,
                        password: this.password,
                    })
                    .then(response => {
                        $cookies.set('userId', response.data.id);
                        $cookies.set('userToken', response.data.token);
```

```
            window.location.href = '/';
        })
        .catch(error => {
            alert('Invalid login or password');
        });
    }
},
onRegister() {
    if (this.validateRegister(this.login, this.password) === true) {
        axios.post('http://localhost:3000/register/', {
            login: this.login,
            password: this.password,
        })
        .then(response => {
            $cookies.set('userId', response.data.id);
            $cookies.set('userToken', response.data.token);
            window.location.href = '/';
        })
        .catch(error => {
            alert('Can`t create this user');
        });
    }
},
validateLogin(login, password) {
    if(login.length <= 0) {
        alert('Password and Login fields cannot be empty');
        return false;
    }
    if(password.length <= 0) {
        alert('Password and Login fields cannot be empty');
        return false;
```

```
            }
            return true;
        },
        validateRegister(login, password) {
            if(login.length <= 0)
                {alert('Password and Login fields cannot be empty'); return false;}
            if(!charMatch.test(login))
                {alert('Login must contain only latin letters'); return false;}
            if(login.length < 4 || login.length > 20)
                { alert('Login must be between 4 and 20 characters'); return false;}
            if(parseInt(login.substr(0, 1)))
                {alert('Login must begin with a letter'); return false;}
            if(password.length <= 0)
                {alert('Password and Login fields cannot be empty'); return false;}
            if(!charMatch.test(password))
                {alert('Password must contain only latin letters'); return false;}
            if(password.length < 6)
                { alert('Password must be more than 6 characters'); return false;}
            return true;
        },
    },
  })
 </script>
</body>
</html>
```

ДОДАТОК В

Презентація

Веб-чат підвищеної швидкодії з можливістю універсального розгортування

Розробив:
Студент групи 2КІ-20м
Черненко Дмитро Юрійович

Керівник: д. т. н., професор
Азаров Олексій Дмитрович

Консультант: к.т.н., доц. каф. ОТ
Черняк Олександр Іванович

- **Метою** даної дипломної роботи є підвищеня швидкодії чату для покращення досвіду користування та реалізація універсальності розгортування для зручності розміщення проекту на серверному комп'ютері. .
- **Об'єктом дослідження** процес обміну повідомленнями в мережі Інтернет.
- **Предметом дослідження** є веб чат підвищеної швидкодії з можливістю універсального розгортування.
- **Наукова новизна** запропонованого рішення полягає у тому, вперше запропоновано та реалізовано вебчат на основі технологій Vue.js, Node.js, MongoDB, Docker, що дозволило збільшити швидкодію та забезпечити зручне розгортування додатку на серверному комп'ютері.
- **Практична цінність роботи** полягає в розробці програмного продукту, який відрізняється збільшенною швидкодією та універсальністю розгорування. Сферою застосування результатів роботи є мережа Інтернет.

Для досягнення даної мети у дипломній роботі потрібно вирішити наступні **задачі:**

- Розробити та стилізувати сторінку.
- Розробити програмне забезпечення клієнтської та серверної частин, використовуючи вибрані технології.
- Налаштувати Docker для контейнеризації застосунку

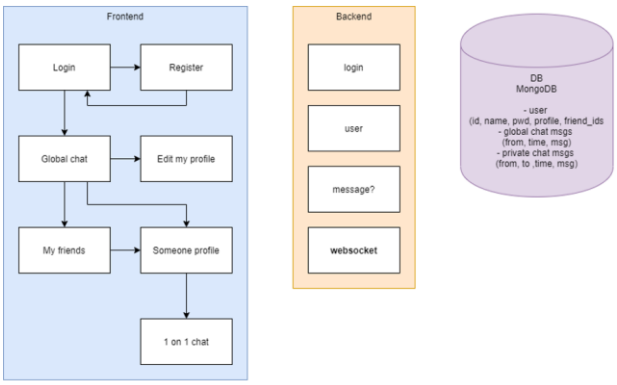
Порівняльна таблиця фреймворків

	Vue.js	React	ANGULARJS
Швидкість та продуктивність	+	—	—
Простота підтримки та розширення	+	+	—
Простота процесу розробки	+	+	—
Безпека розробки	—	—	+

Список використаних технологій

- node
- Vue.js
- CSS HTML
- Websocket
- mongoDB
- docker

Діаграма роботи чату



Створення форми для рєстрації/авторизації

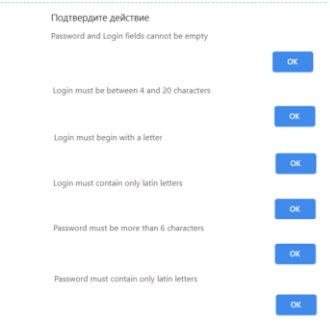
```
<div id="app">
    <div id="action_form">
        <div class="container">
            <label><b>Your name</b></label>
            <div><input id="username" v-model="login" placeholder="Enter username" tabindex="1" type="text"></div>
            <label><b>Your password</b></label>
            <div><input id="password" v-model="password" placeholder="Enter password" tabindex="2" type="password"></div>
            <div><label><input type="checkbox" checked="checked" name="remember"> <span class="lgn">Remember me</span></label>
            <div><button v-on:click="onLogin()" tabindex="3" class="lgn">Sign-in</button></div>
            <div><button v-on:click="onRegister()" tabindex="4" class="lgn">Create your account</button></div>
        </div>
    </div>
</div>
```

```
onLogin() {
    if (this.validateLogin(this.login, this.password) === true) {
        axios.post('http://localhost:3000/login/', {
            login: this.login,
            password: this.password,
        })
        .then(response => {
            $cookies.set('userId', response.data.id);
            $cookies.set('userToken', response.data.token);
            window.location.href = '/';
        })
        .catch(error => {
            alert('Invalid login or password');
        });
    }
},
```

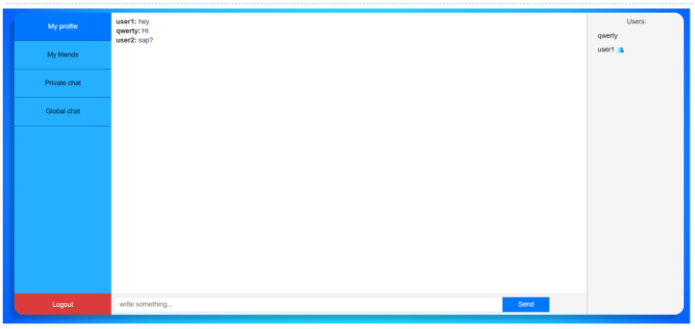
```
onRegister() {
    if (this.validateRegister(this.login, this.password) === true) {
        axios.post('http://localhost:3000/register/', {
            login: this.login,
            password: this.password,
        })
        .then(response => {
            $cookies.set('userId', response.data.id);
            $cookies.set('userToken', response.data.token);
            window.location.href = '/';
        })
        .catch(error => {
            alert('Can`t create this user');
        });
    }
},
```

Валідація форми для реєстрації/авторизації

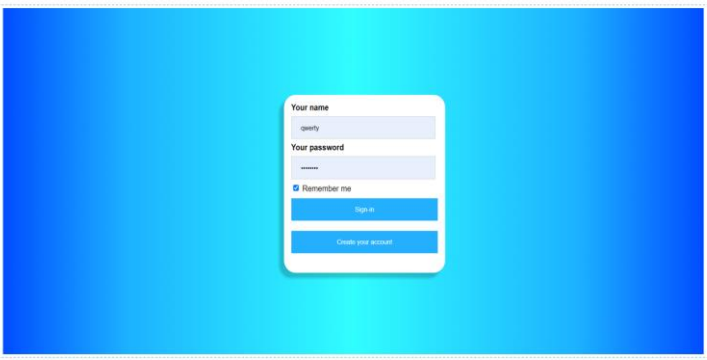
```
validateLogin(login, password) {
    if(login.length <= 0) {
        alert('Password and Login fields cannot be empty');
        return false;
    }
    if(password.length <= 0) {
        alert('Password and Login fields cannot be empty');
        return false;
    }
    return true;
},

validateRegister(login, password) {
    if(login.length <= 0)
        {alert('Password and Login fields cannot be empty'); return false;}
    if(!charMatch.test(login))
        {alert('Login must contain only latin letters'); return false;}
    if(login.length < 4 || login.length > 20)
        { alert('Login must be between 4 and 20 characters'); return false;}
    if(parseInt(login.substr(0, 1)))
        {alert('Login must begin with a letter'); return false;}
    if(password.length <= 0)
        {alert('Password and Login fields cannot be empty'); return false;}
    if(!charMatch.test(password))
        {alert('Password must contain only latin letters'); return false;}
    if(password.length < 6)
        { alert('Password must be more than 6 characters'); return false;}
    return true;
},
```



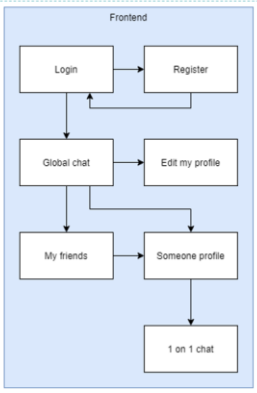
Вигляд форми глобального чату



Вигляд форми для рєстрації або авторизації



Розробка панелі навігації, блоку чату та роутеру.
Використання особливостей фреймворка Vue.js.




```
<li class="item" v-on:click="dataUri = 'global_chat'"><a href="#"><span>Global chat</span></a></li>
<li class="item" v-on:click="dataUri = 'my_profile'"><a href="#"><span>My profile</span></a></li>
<li class="item" v-on:click="dataUri = 'my_friends'"><a href="#"><span>My friends</span></a></li>
<li class="item" v-on:click="dataUri = 'private_chat'"><a href="#"><span>Private chat</span></a></li>
<input type="button" value="Logout" class="logout-btn" v-on:click="logout()">
```

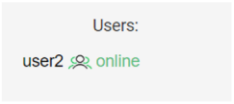
```
<div v-if="dataUri === 'global_chat'">
    <div class="chat-list">
        <li v-for="item in messages" :key="item._id">
        <b>({{ item.fromName }}:</b> {{ item.text }}
        </li>
    </div>
    <div class="input-area">
    <div class="input-wrapper"><input v-model="globalChatText" v-on:keyup.enter="sendMessage(globalChatText)"
                                      type="textarea" value="" placeholder="Write something..."></div>
    <input v-on:click="sendMessage(globalChatText)" type="button" value="Send" class="send-btn">
    </div>
</div>
```

Розробка списку користувачів.
Використання особливостей фреймворка Vue.js.

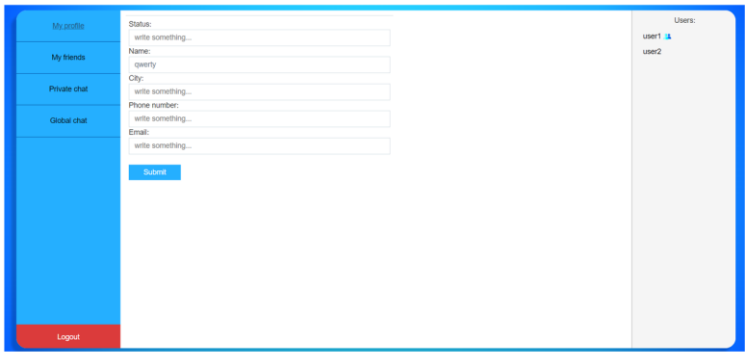
```
<div class="online-list">
    <span class="span-online-users">Users:</span>
    <li class="online-list-item" v-on:click="onSomeoneClick(item)" v-for="item in userList" :key="item._id">
    <a class="online-list-a"><span >{{ item.name }}</span></a>
    <span v-if="friendSign(item)" class="friend-sign"> <img :src="require('./friend_icon.png')"
                                                            width="20px"
                                                            height="15px"/></span>
    <span v-if="item.online" class="online-sign"> online</span>
    </li>
</div>
```

```
onSomeoneClick(someone) {
    axios.get(`http://localhost:3000/users/${someone.id}`)
        .then(response => {
            this.someone = response.data;
            this.someone.isFriend = this.user.friends.some(e => e.id === someone.id);
            this.dataUri = 'someone_profile';
        });
},
```

```
friendSign(testUser){
    return this.user.friends.some(e => e.id === testUser.id);
},
```



Вигляд особистого профілю



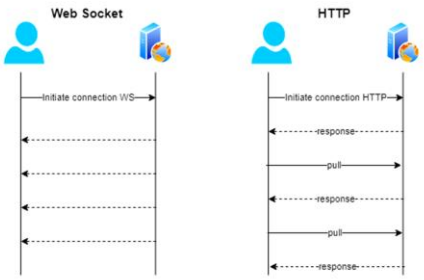
Підключення бази даних MongoDB.

```
mongo.MongoClient.connect(dbUrl, (err, client) => {
  if (err) throw err;

  const db = client.db(dbName);
  console.log('Connected to MongoDB: ' + dbUrl + '/' + dbName);

  app.use(function(req, res, next) {
    res.header('Access-Control-Allow-Origin', frontendUrl);
    res.header('Access-Control-Allow-Credentials', true);
    res.header('Access-Control-Allow-Methods', 'HEAD, OPTIONS, GET, POST, PUT, DELETE');
    res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');
    next();
  });
```

```
{
    "_id": {
        "$oid": "5ee8967a260f013a0048d6ff"
    },
    "login": "user3",
    "password": "$2b$10$SsNO7l4.rSWxJsjCP4rHTeBQrFWywQk3cMjXmfXJLw6go4/RtWVKK",
    "token": "a6f5ed45-5587-4db3-9281-885d90495e49",
    "name": "user3",
    "active": true,
    "status": "SOme status",
    "city": "City1",
    "phone": "1111111111",
    "email": "email@mail.com",
    "friends": []
}
```

Екземпляр користувача

З'єднання через Websocket



Створення WebSocket

Створення

```
app.ws('/chat/:id', (ws, req) => {
  ws.uuid = uuidv4();
  ws.userId = req.params.id;
  sessions.push(ws);
  console.log('Connected:', ws.uuid, ', userId:', ws.userId);
```

```
this.userId = $cookies.get('userId');
this.userToken = $cookies.get('userToken');

if (this.userToken === null) {
    window.location.href = '/loginform.html';
} else {
    this.connection = new WebSocket(`${backendWS}/chat/${this.userId}`);
    this.connection.onmessage = (event) => {
        this.onMessageReceived(JSON.parse(event.data));
    };
}
```

Використання

```
ws.on('message', async (msg) => {
  let data = JSON.parse(msg);
  console.log(data);

  var result = await db.collection('users').find().toArray();
  var userNames = {};
  result.forEach(u => userNames[u._id] = u.name);

  if (data.type === 'public') {
    // Save to DB
    var message = {
      type: data.type,
      from: data.from,
      fromName: userNames[data.from],
      text: data.text,
    };
    await db.collection('messages').insertOne(message);
    // Send to everyone, including the sender
    sessions.forEach(w => w.send(JSON.stringify(message)));
```

Контейнеризація Docker

Вигляд контейнерів в Docker UI

Docker-compose backend

```
services:

  nodejs-server:
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    depends_on:
      - mongodb
    container_name: node-api
    restart: unless-stopped
    volumes:
      - ./backend:/usr/src/app/api
      - node_modules:/home/node/app/node_modules
    networks:
      - app-network
```

Docker-compose frontend

```
vue-ui:
  build:
    context: ./frontend
    dockerfile: Dockerfile
  ports:
    - "8080:8080"
  depends_on:
    - nodejs-server
  container_name: vue-ui
  volumes:
    - ./frontend:/usr/src/app/my-app
    - /usr/src/app/my-app/node_modules
```

Docker-compose MongoDB

```
mongodb:
  command: mongod --noauth --port 27017
  container_name: mongodb
  environment:
    - MONGO_INITDB_DATABASE=fine-chat
  image: mongo:4.4.5-bionic
  restart: always
  ports:
    - 27017:27017
  volumes:
    - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
  networks:
    - app-network
```

- В даному проекті створено чат підвищеної швидкодії. Швидкодія досягається за рахунок використання технологій Vue.js, Websocket та швидкої бази даних MongoDB. Універсальність розгортування досягається за рахунок використання інструментарію Docker.

- Основною роботою в розробці даного проекту було початкове створення проекту в середовищі Visual Studio Code. Для розробки даного програмного забезпечення було продумано повноцінну архітектуру проекту. Було вдало обрано мову програмування – JavaScript, так як вона багата на відкриті бібліотеки, що дають широкий спектр можливостей в сфері веб розробки.

- Було створено та стилізовано всі сорінки з якими повинен взаємодіяти користувач, розроблено сервену та клієнтську частини. В результаті отримано веб-чат підвищенної швидкодії з можливістю реєструватись та надсилати повідомлення як в глобальний чат, так і персонально вибраному користувачу, заповнювати власний профіль інформацією про себе, переглядати список всіх користувачів та їх профілі. Також даний веб-чат може розгортуватись в будь-якій системі завдяки інструментарію Docker

Дякую за увагу!

ДОДАТОК Г

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Швидкодіючий вебчат з можливістю універсального розгортання

Тип роботи:_____магістерська кваліфікаційна робота_____

(кваліфікаційна роботи, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____Кафедра обчислювальної техніки_____

(кафедра, факультет (інститут), навчальна група)

Науковий керівник ___Азаров О. Д., професор_____

(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	86,9
Тривога/Білі знаки	/	Схожість	13,1

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор ___ __Черненко Д.Ю.___

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

____Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

_____

_____

_____

Особа, відповідальна за перевірку_____ ___Захарченко С.М___

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали)