МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему «Розробка корпоративного сервісу обміну повідомленнями»
«Corporate messaging service development»

Виконав: студент денної форми навчання
спеціальності 123 – Комп'ютерна інженерія.

(шифр і назва напряму підготовки, спеціальності)

Автєньєв Дмитро Вячеславович

(прізвище, ім'я, по-батькові)

Керівник Трубіна Н.Ф. _____

(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент к.т.н., доцент Пенко В.Г._____

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2022 р.

Завідувач кафедри

_____ Євгеній МАЛАХОВ

(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___

протокол № __ від «___» ____ 2022 р.

Оцінка _____/_____/____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ Надія КАЗАКОВА_

(підпис) (ім'я, прізвище)

Одеса - 2022

АНОТАЦІЯ

Дипломна робота є складовою частиною проекту по розробці корпоративного клієнт-серверного месенджера.

Метою роботи є підвищення ефективності робочого процесу.

Основні завдання:

а) провести аналіз існуючих аналогічних програм;

б) розробити алгоритм, базу даних та інтерфейс;

в) реалізувати проект із використанням програмних засобів.

В результаті виконання випускної кваліфікаційної роботи було розроблено web-додаток «Корпоративний месенджер», що дозволяє ефективно взаємодіяти між собою учасникам системи.

ABSTRACT

The graduation work is an integral part of the project to develop a corporate client-server messenger.

The purpose of the work is to increase the efficiency of the work process.

Main tasks:

a) analyze existing similar programs;

b) develop an algorithm, database and interface;

c) implement the project using software.

Based on the results of the final qualifying work, a web application "Corporate Messenger" was developed, which allows participants to interact effectively with each other.

ЗМІСТ

ВСТУП

Практично всі компанії нарівні з електронною поштою та стільниковим зв'язком використовують альтернативні канали зв'язку для вирішення миттєвих робочих питань. Це можуть бути програми для відеозв'язку, месенджери, соціальні мережі, наприклад: Skype, WhatsApp, Viber.

Однак вищезазначені програми мають ряд недоліків:

– співробітники часто користуються різними програмами для обміну інформацією;

– у контакт-листі месенджерів присутні контакти, які не мають відношення до роботи, наприклад, родичі та друзі.

Для зручності комунікацій та роботи всередині компанії, а також для компенсації перерахованих вище недоліків використовуються корпоративні месенджери.

Переваги корпоративних месенджерів:

– підвищення швидкості комунікації у вирішенні робочих питань та поділ потоків спілкування. Спеціалізовані месенджери можуть акцентувати їхню увагу на вирішенні виключно робочих питань, дозволяючи не відволікатися та не засмічувати корпоративну пошту зайвою інформацією;

– командна робота та можливість зведення докупи кількох інформаційних потоків;

– економія внутрішніх технічних ресурсів підприємства. Готові месенджери, як правило, працюють за принципом «хмарних сховищ» даних, і дозволяють організації заощадити місце на серверах та знизити витрати на створення та обслуговування власного сервісу комунікації.

Крім безпеки, важливу роль відіграє слабка орієнтованість таких систем під потреби корпоративного сегменту.

Потрібно просто скористатися спеціалізованим продуктом, необхідним для використання на підприємстві.

Таких програм чимало і всі вони мають мінімальний набір необхідних функцій: авторизацію користувачів, структурування за підрозділами підприємства, зберігання історії повідомлень, організацію можливості обміну файлами тощо.

При виборі конкретного програмного продукту основні критерії – широкий набір можливостей, невисока ціна та зручність у розгортанні, налаштуванні та використанні.

За даними агентства «Frost&Su1livan» [1], кількість учасників корпоративних соціальних платформ у 2013 році по всьому світу становила 208 млн осіб і, за прогнозами, досягне 535 млн до 2022 року.

Метою даного дипломного проекту є створення системи, що забезпечує зручну, швидкісну та захищену електронну комунікацію серед користувачів системи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– пошук та дослідження аналогічних сервісів;
– проектування програми;
– розробка програми;

1 СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ ТА ТЕХНОЛОГІЇ ЇХ СТВОРЕННЯ

1.1. Огляд існуючих додатків

Було проведено пошук та аналіз існуючих чат-сервісів для корпоративних користувачів. На ринку існує багато програмних продуктів для комунікацій усередині робочої команди. Нижче розглянуті деякі з них:

1) Staply**.** Месенджер із проектним функціоналом. Дозволяє: скласти ієрархію груп; створювати завдання, важливість та відповідальність у робочих групах; здійснювати контроль за групами [2];
2) Slack**.** Корпоративний месенджер із підтримкою інтеграції з десятками сторонніх сервісів. Об'єднує в одному вікні обговорення у спільних темах (каналах), приватних групах та особистих повідомлень. Дозволяє здійснювати відеодзвінки; зберігати повідомлення без доступу до мережі [3];
3) HipChat**.** Є груповим чатом, відеочатом для команд і компаній з можливостями обміну файлами і демонстрацією екрану [4];
4) Бітрікс24**.** Система управління внутрішнім інформаційним ресурсом компанії для колективної роботи над завданнями, проектами та документами, для ефективних внутрішніх комунікацій. Дозволяє контролювати, делегувати, оцінювати завдання; створювати резервні копії в кількох місцях; працювати у кількох кімнатах одночасно [5].

На основі функцій аналогів та поставлених завдань можна виділити наступні критерії для порівняння:

– інтеграція зі сторонніми сервісами;
– створіння ієрархії груп;
– розподіл керуючим учасників по ролям;
– робота в кількох кімнатах одночасно.

У таблиці 1.1 представлена порівняльна характеристика перерахованих вище продуктів.

Таблиця 1.1 – Порівняльна характеристика розглянутих аналогів

Критерій	Аналоги				Примітка
	Staply	Slack	HipChat	Бітрікс24	
Інтеграція зі сторонніми сервісами	-	Dropbox, Google Drive, GitHub, Google Docs, Google Hangouts, Twitter та інші.	GitHub, MailChimp Heroku та інші.	MS SharePoint, MS Exchange Server, MS Outlook, Продукти Apple та Google.	Дозволяє користувачам відстежувати прогресс в різних проектах з допомогою однієї платформи
Створіння ієрархії групп	+	-	-	-	Дозволяє рівномірно розподілити завдання
Розподіл керуючим учасників по ролям	-	-	-	-	Дозволяє ефективно розподілити завдання
Можливість контроля керуючим над проектом	+	+	+	+	Забезпечує контроль виконання завдань
Резервне копіювання у кількох місцях	+/−	-	-	+	Забезпечує безпека даних
Робота в кількох кімнатах одночасно	+	-	-	+	Дозволяє вирішувати кілька завдань одночасно

Продовження таблиці 1.1

Критерій	Аналоги				Примітка
	Staply	Slack	HipChat	Бітрікс24	
Інструменти	Блокнот, Завдання			Щоденник, планувальник подій, календар, графік збори та планерки	Дозволяють оперативно виконувати завдання
Оффлайн-повідомлення	+	+	+	–	Дозволяє працювати з інформацією без мережі

1.2. Огляд технологій

1.2.1. Web-додаток

Веб-програма – клієнт-серверна програма, в якій клієнт взаємодіє із сервером за допомогою браузера, а за сервер відповідає веб-сервер. Логіка веб-додатка розподілена між сервером і клієнтом, зберігання даних здійснюється переважно на сервері, обмін інформацією відбувається за мережі. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-програми є міжплатформними службами[6].

1.2.2. Архітектура «клієнт- сервер»

«Клієнт-сервер» – обчислювальна або мережева архітектура, в якої завдання або мережеве навантаження розподілені між постачальниками послуг, званими серверами, та замовниками послуг, які називаються клієнтами.

Веб-додаток складається з клієнтської та серверної частин, тим самим реалізуючи технологію «клієнт-сервер».

Клієнтська частина реалізує інтерфейс користувача, формує запити до сервера і обробляє відповіді від нього. Код клієнтської частини написаний з використанням HTML, CSS та JavaScript [7].

Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує веб-сторінку та надсилає її клієнту через мережу з використанням протоколу HTTP. Код серверної частини може бути написаний на будь-якій кількості мов програмування – приклади популярних мов серверної частини включають себе PHP, Python та C#.

1.2.3. Концепція MVC

Проект розроблено на Javascript з використанням платформи Node.js. Ця платформа заснована на використанні концепції MVC (Model-View-Controller).

Основна мета застосування цієї концепції полягає у розподілі бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого розподілу підвищується можливість повторного використання. Найбільш корисно застосування даної концепції в тих випадках, коли користувач повинен бачити ті самі дані одночасно в різних контекстах і/або з різних точок зору. Зокрема, виконуються такі завдання:

– до однієї моделі можна приєднати кілька видів, у своїй не торкаючись реалізацію моделі. Наприклад, деякі дані можуть бути представлені у вигляді електронної таблиці, гістограми і кругової діаграми;
– не зачіпаючи реалізацію видів, можна змінити реакцію дії користувача (натискання мишею на кнопці, введення даних), цього досить використовувати інший контролер;
– ряд розробників спеціалізується тільки в однієї з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку.

Тому можна домогтися того, що програмісти, що займаються розробкою бізнес-логіки (моделі), взагалі не будуть обізнані про те, яке уявлення використовуватиметься.

Концепція MVC дозволяє розділити дані, представлення та обробку дій користувача на три окремі компоненти:

– модель (англ. Model). Модель надає знання: дані та методи роботи з цими даними реагує на запити, змінюючи свій стан. Не містить інформації, як ці знання можна, можливо візуалізувати;
– подання, вид (англ. View). Відповідає відображення інформації (візуалізацію). Часто як уявлення виступає форма (вікно) із графічними елементами;
– контролер (англ. Controller). Забезпечує зв'язок між користувачем та системою: контролює введення даних користувачем та використовує модель та уявлення для реалізації необхідної реакції;

Як уявлення, і контролер залежить від моделі. Проте модель залежить ні від уявлення, ні від контролера. Тим самим досягається призначення такого поділу: воно дозволяє будувати модель незалежно від візуального уявлення, і навіть створювати кілька різних уявлень однієї моделі [8].

1.3. Огляд засобів розробки

1.3.1. Мова розмітки HTML

HTML – стандартна мова розмітки документів у Всесвітньому павутинні. Більшість веб-сторінок створюються при допомоги мови HTML (або XHTML). Мова HTML інтерпретується браузерами та відображається у вигляді документа у зручній для людини формі.

HTML є додатком («приватним випадком») SGML (Стандартного узагальненого мови розмітки) і відповідає міжнародному стандарту ISO 8879. XHTML ж є додатком XML.

Мова HTML була розроблений британським ученим Тімом Бернерсом-Лі приблизно у 1986-1991 роках у стінах Європейської ради з ядерних дослідженням у Женеві (Швейцарія). HTML створювався як мова для обміну науковою та технічною документацією, придатна для використання людьми, що не є спеціалістами в області верстки. HTML успішно справлявся з проблемою складності SGML шляхом визначення невеликого набору структурних та семантичних елементів дескрипторів. Дескриптори також часто називають "тегами". З допомогою HTML можна легко створити відносно простий, але красиво оформлений документ. Крім спрощення структури документа, HTML внесена підтримка гіпертексту. Мультимедійні можливості були додано пізніше.

Текстові документи, що містять розмітку мовою HTML (такі документи зазвичай мають розширення .html або .htm), обробляються спеціальними програмами, які відображають документ у його форматованому вигляді. Браузери надають користувачеві зручний інтерфейс для запиту веб-сторінок, їх перегляду (і виведення інших зовнішніх пристроїв) і, за необхідності, відправки введених користувачем даних на сервер [9].

HTML – тегова мова розмітки документів. Будь-який документ мовою HTML є набір елементів, причому початок і кінець кожного елемента позначається спеціальними позначками- тегами. Елементи можуть бути порожніми, тобто не містять жодного тексту та інших даних (наприклад, тег перекладу рядка
). У цьому випадку зазвичай не вказується закриває тег. Крім того, елементи можуть мати атрибути, які визначають будь-які їх властивості (наприклад, розмір шрифту для елемента font). Атрибути вказуються в відкриває тег.

1.3.2. Мова стилів CSS

CSS – формальний мова опису зовнішнього виду документа, написаного за допомогою мови розмітки.

Переважно використовується як засіб опису, оформлення зовнішнього виду веб-сторінок, написаних з за допомогою мов розмітки HTML та XHTML, але може також застосовуватися до будь-яких XML-документів, наприклад, до SVG або XUL.

CSS використовується творцями веб-сторінок для завдання квітів, шрифтів, розташування окремих блоків і інших аспектів подання зовнішнього вигляду цих веб-сторінок. Основною метою розробки CSS було розділення опису логічної структури веб-сторінки (яке виробляється за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер виробляється за допомогою формальної мови CSS). Такий поділ може збільшити доступність документа, надати більшу гнучкість та можливість управління його поданням, а також зменшити складність та повторюваність у структурному вмісті [10].

1.3.3. Бібліотека React.js

React – це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду – “компонентів”.

Особливості:

– одностороння передача даних, властивості передаються в рендерер компоненту, як властивості html тегу. Компонент не може напряму змінювати властивості, що йому передані, але може їх змінювати через callback функції;

– віртуальний DOM, React підтримує віртуальний DOM, а не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити;

– JSX, компоненти React зазвичай написані на JSX. Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники можуть так само писати на чистому JavaScript;

– атрибути, JSX надає ряд атрибутів елементів, призначених для відображення тих, що надаються у форматі HTML. Користувацькі атрибути також можуть бути передані компоненту. Всі атрибути будуть отримані компонентом як реквізит [11].

1.3.4. Платформа Node.js

Node.js – програмна платформа, заснована на движку V8 (компілюючий JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення.

Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виводу через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані різними мовами, забезпечуючи виклики до них із JavaScript-коду.

Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js та десктопні віконні програми [12].

1.3.5. СУБД MongoDB

Система управління базами даних (СУБД) – сукупність програмних та лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням та використанням баз даних.

MongoDB — система управління базами даних, яка працює з документоорієнтованою моделлю даних. На відміну від реляційних СУБД, MongoDB не потребує таблиці, схеми або окремої мови запитів. Інформація зберігається як документів чи колекцій [13].

Особливості:

– кросплатформність. СУБД розроблена мовою програмування C++, тому легко інтегрується під будь-яку операційну систему (Windows, Linux, MacOS та інших);

– формат даних. MongoDB використовує власний формат зберігання інформації – Binary JavaScript Object Notation (BSON), який побудований на основі JavaScript;

– документ. Якщо реляційні БД використовують рядки, MongoDB - документи, які зберігають значення і ключі;

– замість таблиць MongoDB використовує колекцію. Вони містять різні типи наборів даних;

– реплікація. Система зберігання інформації у СУБД представлена вузлами. Існує один головний та безліч вторинних. Дані реплікуються між крапками. Якщо один первинний вузол виходить із ладу, то вторинний стає головним;

– індексація. Технологія застосовується до будь-якого поля у документі на розсуд користувача. Проіндексована інформація обробляється швидше;

– для збереження великого розміру даних MongoDB використовує власну технологію GridFS, що складається з двох колекцій. У першій (files)

містяться імена файлів та метадані за ними. Друга (chunks) зберігає сегменти інформації, розмір яких не перевищує 256 Кб;

– СУБД здійснює пошук за спеціальними запитами. Наприклад, користувач може створити діапазонний запит та миттєво отримати відповідь.

1.3.6. Бібліотека Socket.IO

Socket.IO – це бібліотека JavaScript для веб-застосунків реального часу. Він забезпечує двосторонній зв'язок у реальному часі між веб-клієнтами та серверами. Він складається з двох частин: клієнтської бібліотеки, яка запускається у браузері, та серверної бібліотеки для node.js. Обидва компоненти мають ідентичний API.

Програма реального часу (RTA) – це програма, яка функціонує протягом періоду, який користувач сприймає як негайний або поточний.

Деякі приклади програм у реальному часі:

– месенджери – програми для чату, такі як Whatsapp, Facebook Messenger і т. д. Вам не потрібно оновлювати свою програму / веб-сайт для отримання нових повідомлень;

– push-сповіщення – коли хтось позначає вас на фотографії у Facebook, ви одразу отримуєте повідомлення;

– програми для спільної роботи – програми, такі як Google Docs, які дозволяють декільком людям одночасно оновлювати одні й ті самі документи та вносити зміни до всіх екземплярів людей [14].

2 ПРОЕКТУВАННЯ СИСТЕМИ ОРГАНІЗАЦІЇ КОРПОРАТИВНОГО СПІЛКУВАННЯ

У рамках виконання дипломного проекту поставлено завдання розробити систему оперативного обміну даними у вигляді текстових файлів, файлів зображень, файлів інших форматів та повідомлень у межах користувачів корпоративного месенджера.

Корпоративний месенджер розробляється в організацію віддаленої роботи користувачів однієї організації чи установи.

2.1. Функціональні характеристики

Додаток, що розробляється, дозволить здійснювати обмін даними між зареєстрованими користувачами, обмінюватися повідомленнями та файлами, об'єднувати користувачів у робочі групи та адмініструвати роботу системи.

2.2. Функціональні вимоги

Проектована система є клієнт-серверним комунікаційним комплексом для корпоративної мережі. Функціональність повинна включати передачу/ прийом текстових повідомлень, файлів, розсилки широкомовних повідомлень, адресну книгу та багато іншого. Всі повідомлення, що передаються, необхідно зберігати в базі даних.

Для забезпечення захисту від несанкціонованого доступу необхідно передбачити авторизацію користувача при вході в систему.

Реєстрація користувача можлива лише поштою @stud.onu.edu.ua, а також вона має бути внесена до whitelist у базі даних.

Адміністратор при резервному копіюванні та архівуванні інформації бази даних повинен захистити її службовим паролем.

У системі необхідно передбачити можливість розмежування повноважень користувачів. Необхідно, щоб адміністратор міг розподіляти рівні доступу до інформації кожного користувача та надавати їм права доступу до системи.

2.3. Діаграма прецедентів

Діаграма прецедентів (Use Case Diagram) діаграма, що відображає відносини між акторами та прецедентами та складова модель прецедентів, що дозволяє описати систему на концептуальному рівні. Ця діаграма призначена визначення функціональних вимог до програмного продукту.

Суть діаграми прецедентів полягає в тому, що проєктована система подається у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання.

Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором.

Основними елементами діаграми прецедентів є актори та прецеденти. Актор – роль, яку відіграють зовнішні сутності. Прецедент – послідовності дій, які система чи інша сутність можуть виконувати у процесі взаємодії з акторами. Були виділені актори: адміністратор та користувач.

Діаграма роботи з профілем користувача представлена на рисунку 2.1. Користувач може переглядати та редагувати свій профіль, переглядати новини, повідомлення.

Рисунок 2.1 – Use case діаграма роботи з профілем користувача

Діаграма роботи адміністратора представлена рисунку 2.2. Адміністратор веде облік існуючих користувачів, може редагувати список посад, користувачів, і навіть може редагувати існуючі проекти, змінювати статус користувачів.

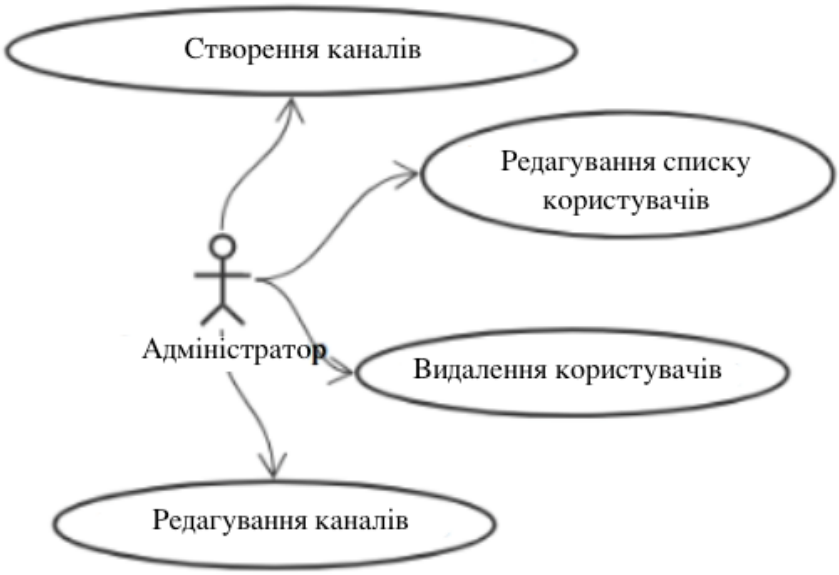


Рисунок 2.2 – Use case діаграма роботи адміністратора

2.4. Розробка бази даних

Для роботи програми була створено база даних, яка містить 5 колекції. Таблиця 2.1 містить опис колекцій бази даних.

Таблиця 2.1 – Опис колекцій бази даних

Назва колекції	Поля	Опис
Users	_id	ID користувача
	profilePicture	Фотографія користувача
	followers	Список передплатників користувача
	following	Список на кого підписаний користувач
	username	ПІБ користувача
	email	Пошта користувача
	password	Пароль користувача
	department	Кафедра користувача
	position	Посада користувача
Posts	_id	ID поста
	likes	Лайки
	userID	ID творця поста
	description	Опис
	img	Зображення
Messages	_id	ID повідомлення
	sender	ID відправника
	text	Текст повідомлення
	conversationID	ID бесіди
Conversations	_id	ID бесіди
	members	Список учасників
Whitelist	_id	ID пошти
	email	Пошта яка дозволена

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ТА ЇЇ ФУНКЦІЮНУВАННЯ

3.1. Реалізація серверної частини

Першим етапом становлення сервера є його написання та запуск на локальному хості на виділеному порту. Спочатку необхідно ініціалізувати npm, через який встановлюються необхідні пакети. При ініціалізації створюється файл package.json, в якому описано версію, пакети та інші параметри.

```
{
  "name": "mes",
  "version": "0.0.1",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.18.2",
    "express": "^4.16.3",
    "mongoose": "^5.0.14",
    "socket.io": "^2.1.0"
  }
}
```

Рисунок 3.1 – Зміст файлу package.json

Для запуску сервера потрібна наявність пакета Express. Це гнучкий веб-фреймворк, який надає широкий набір функцій як для веб-додатків, так і для мобільних. У своєму розпорядженні він має безліч методів HTTP та обробників, за допомогою яких можна зібрати надійний API (application programming interface).

Методи HTTP, які використовуються при реалізації:

– get. Використовується для запиту інформації із сервера;

– post. Використовується для передачі даних з сервера;

Нижче показано як реалізується цей етап.

```
var app — reqиire('express) )
var bodyPars — require('body-parser)
var to — require('socket.to)()
app.use{bodyPars.json())
var api — require('./api)
app.use{'/api', api)
app.listen(3000, function{){
console.log{”server 3s started!”)
```

По даному лістингу видно, що сервер запускається локальному хості на 3000 порту.

Також можна побачити використання пакету Bodyparser. Він перетворює вхідні дані у необхідний йому тип подальшого використання.

Наступним етапом необхідно створити канал підключення до MongoDB. Для цього знадобиться пакет mongoose. Він створений для роботи у системі управління базами даних. У нього вбудовані інструменти для написання запитів до бази даних, створення колекцій та документів. Нижче показана реалізація підключення до MongoDB:

```
mongoose.connect(’mongodb://user:123@ds241869.mlab.com:41869/
mes’, err => {
     if(err) {
       console.log(err)
     }
     else {
        console.log(“Connection complete!“)
     }
})
```

Метод connect приймає 2 аргументи – рядок підключення і callback функцію. Рядок підключення містить ім'я користувача, його пароль, порт підключення, назву бази даних.

Callback функція виводить на консоль помилку, за умови, що вона є, у протилежному випадку повідомляє, що з'єднання виконано.

Також за допомогою mongoose створюються дві колекції – User і Message. Це показано нижче.

```
var mongoose = require('mongoose') var Schema = mongoose.Schema
var userSchema = new Schema({
  phone : Number, name : String, password : String,
})
module.exports = mongoose.model('user', userSchema, 'users')
var mongoose = require('mongoose') var Schema = mongoose.Schema
var message = new Schema({
  date: Date,
  text: String, username: String
})
module.exports = mongoose.model('message', message, 'messages')
```

Колекція User містить три поля: номер телефону (phone), ім'я користувача (name), пароль (password).

Колекція Message містить також три поля: дата відправлення повідомлення (date), текст повідомлення (text) та ім'я відправника (username).

Наприкінці опису схеми йде додавання її до бази методом model. Він приймає три параметри: назву фізичного файлу, в якому описано схему, ім'я змінної схеми, назву колекції в базі.

Далі необхідно формування http запитів реєстрації та авторизації, вони відображені нижче.

```
router.post('/register', (req, res) =>{
     let userData = req.body
     let user = new User(userData)
     user.save((err, registeredUser)=>{
          if(err){
               console.log(err)
          }else{
               res.status(200).send(registeredUser)
          }
     })
})
```

Реєстрація користувача – запис даних користувача з метою обліку на сайті, мобільному додатку, інформаційній системі.

Було створено новий користувач за описаною схемою User та збережено методом save на основі введених даних.

```
router.post('/login', (req, res)=>{
     User.findOne({name:req.body.name}, (err, user)=>{
          if(err){
               console.log(err)
          }else{
          if(!user || user.password !== req.body.password{
               res.status(401).send("Incorrect name or
password!")
          }else{
               res.status(200).send(user)
          }
     }
}))})
```

Авторизація – надання зареєстрованому користувачеві прав на будь-які дії в системі, а також процес перевірки даних прав при спробі виконання дій.

Авторизація відбувається таким чином: методом findOne за номером телефону знаходиться користувач, потім порівнюються введені ним дані для авторизації з даними, що зберігаються в базі даних і при успішному виконанні він пропускається в систему та отримує автентифікатор.

Наведені вище функції підключення до бази даних, реєстрації та авторизації описані в модулі API, який є маршрутизатором між головним модулем сервера та іншими модулями програми.

Після того, як користувач зареєструється та авторизується в системі, у нього з'являється можливість надсилати повідомлення іншим користувачам за умови, що він знає номер телефону.

Модуль надсилання повідомлень побудований таким чином, що користувач з бажанням поспілкуватися з кимось вибирає зі списку або за допомогою введення номера телефону за умови, що той зареєстрований у базі даних.

Далі відбувається створення або підключення до кімнати (діалогу). Після цього відбувається завантаження повідомлень, якщо вони вже існували. Користувач вводить текст повідомлення та надсилає співрозмовнику. При надсиланні повідомлення з параметрами зберігається у базі даних.

Для підключення та надсилання повідомлень використовується інтегрований сервіс Socket.io, описаний у розділі 1.

Після написання модуля надсилання повідомлень закінчується розробка серверної частини.

3.2. Реалізація клієнтської частини

Кожне вікно програми складається з 2 частин виконуваного файлу з розширенням .jsx і файлу, що описує стилі об'єктів, з розширенням .css.

Під час створення клієнтської частини використовувався компонентний підхід та основні інструменти, що надаються React.

Вікна авторизації (рис. 3.2) та реєстрації (рис. 3.3) містять поля на введення та кнопки. У виконуваних файлах описані перевірки на порожні поля та збіги. Також описаний перехід на інше вікно.

Після невдалої спроби введення даних у полях вікон, невірно заповнені поля підсвічуються, і поруч із ним виринає пояснювальна фраза. Також у даному проекті дозволено реєстрацію лише з поштою університету (@stud.onu.edu.ua). Якщо ж користувач вірно ввів дані, його перенаправляє на головне вікно.

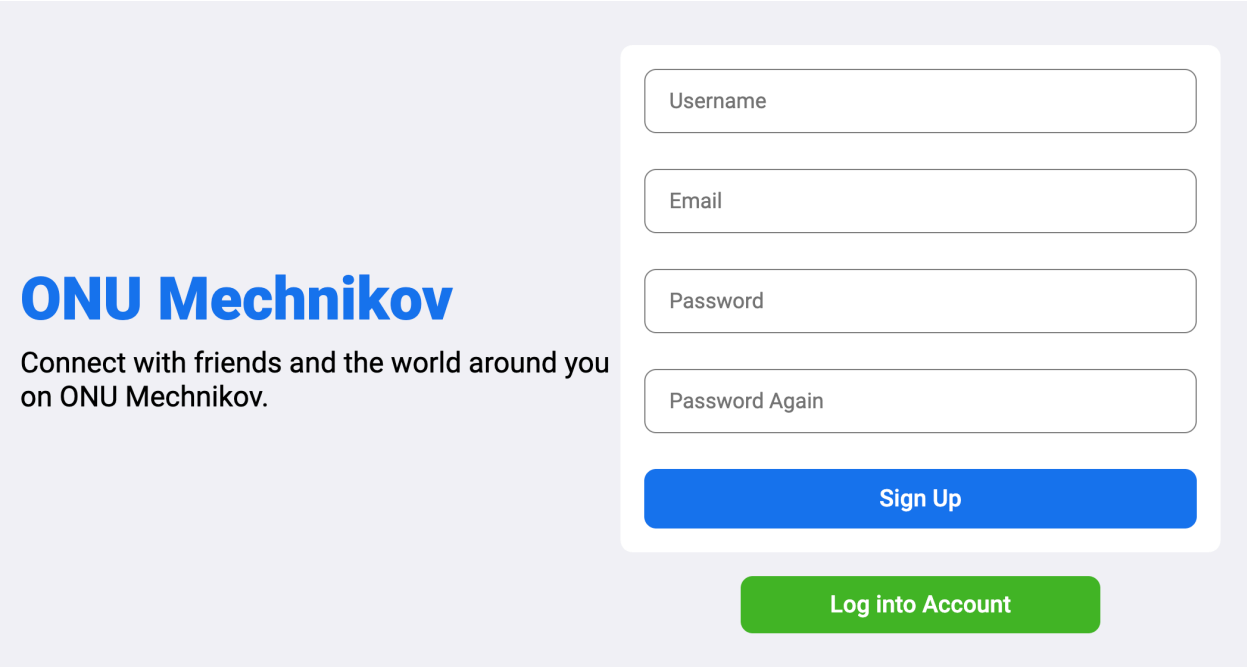


Рисунок 3.2 – Вікно реєстрації

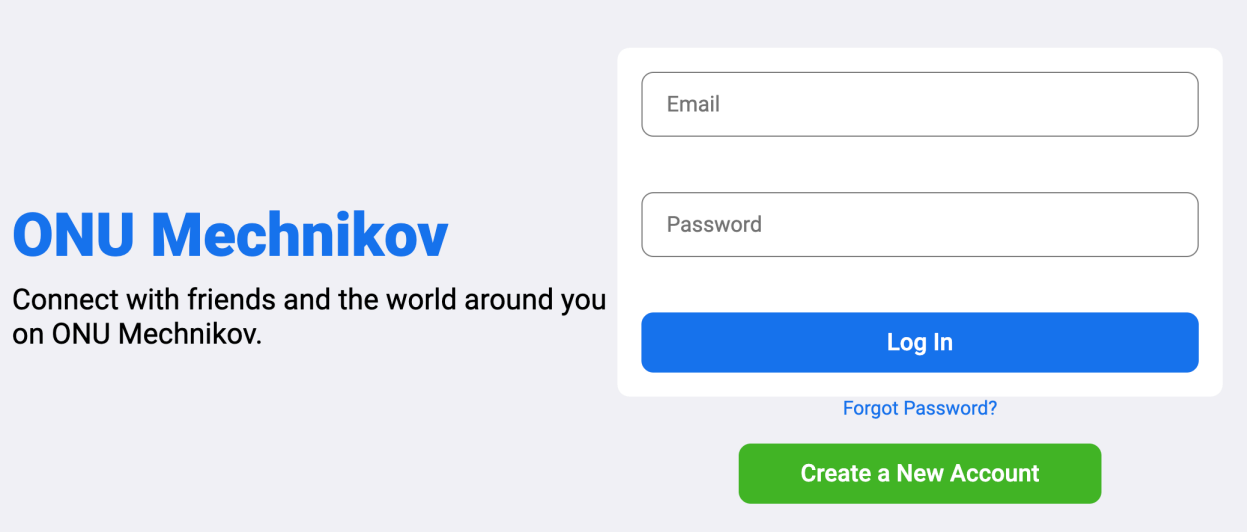


Рисунок 3.3 – Вікно авторизації

На головній сторінці користувача (рис. 3.4) є можливість переглядати новини каналів та користувачів на яких він підписаний, перейти на свою сторінку, перейти в месенджер та інше.

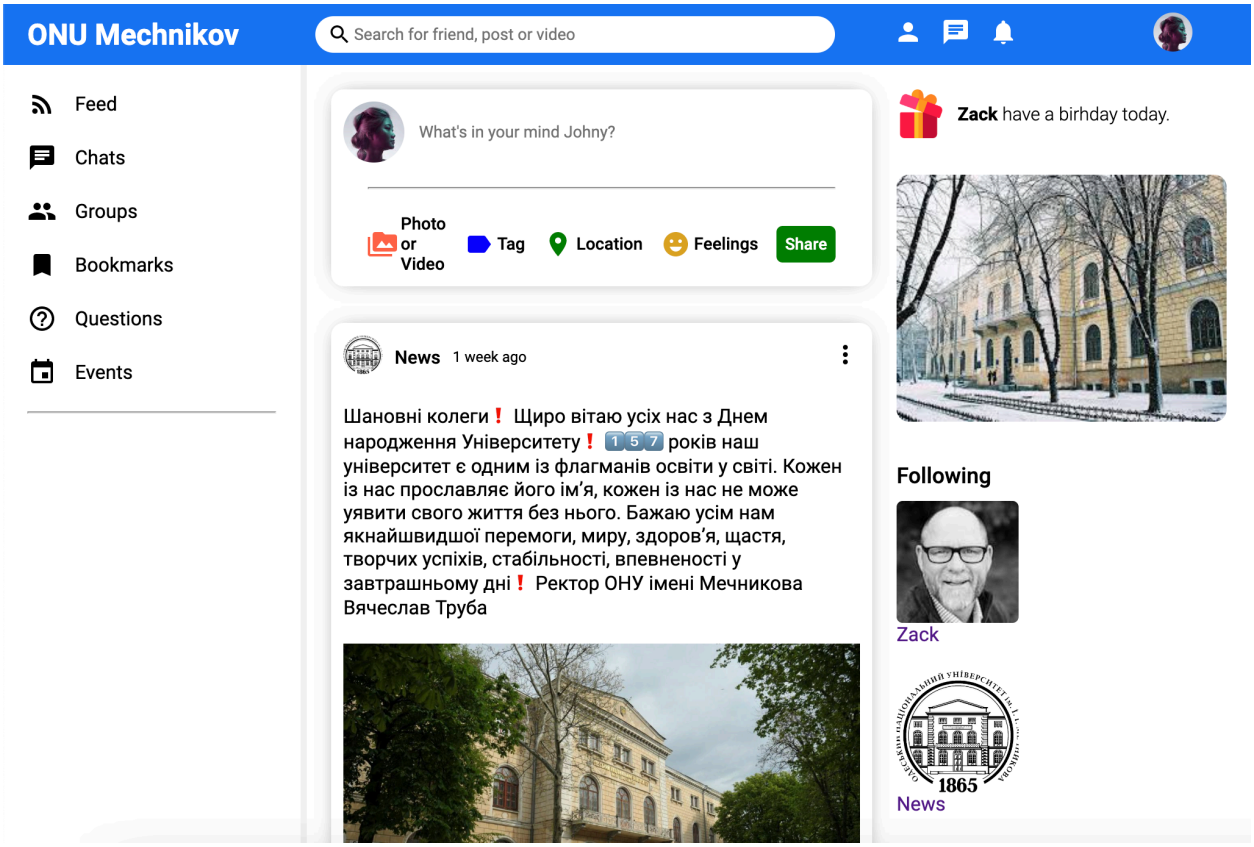


Рисунок 3.4 – Головне вікно

Після того, як користувач вибере співрозмовника, йому відкриється вікно з діалогом (рис. 3.5), в якому є текстове поля для введення та кнопка відправки. Частина вікна займатимуть повідомлення, які вже колись були написані, або знову написані. Також є список онлайн людей на яких підписаний користувач.

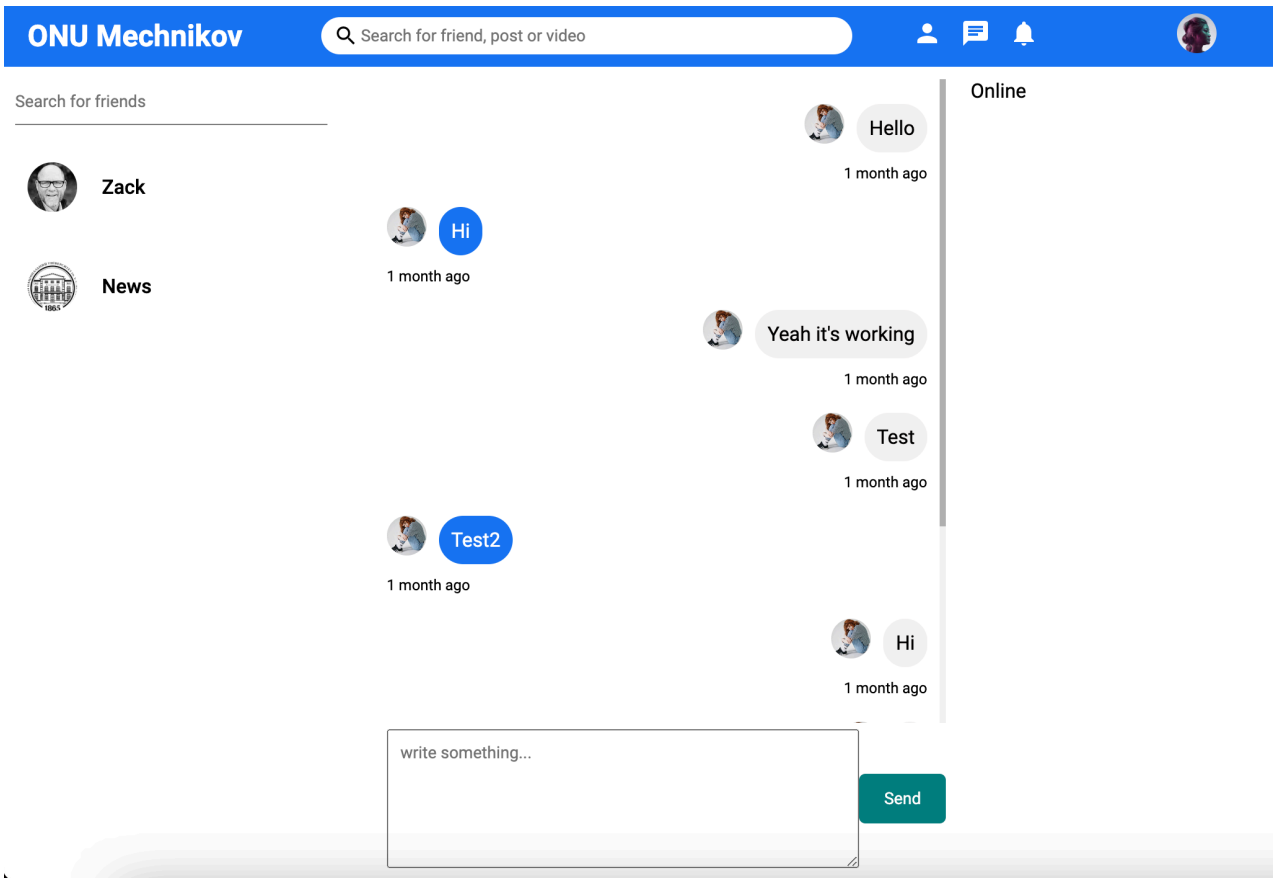


Рисунок 3.5 – Вікно месенджера

Також важливою деталлю клієнтської частини є шифрування.

Шифрування – реверсує перетворення інформації з метою приховування її від неавторизованих осіб, а також з наданням авторизованим користувачам доступу до даних. Головною метою шифрування є дотримання конфіденційності інформації, що передається.

Шифрування ділиться на 2 види – симетричне та асиметричне. При першому використовується один ключ, за допомогою якого зашифровується та розшифровується інформація. При другому вигляді існує 2 – ключі відкритий, він використовується для зашифровки і передається незахищеним каналу, і секретний, який використовується для розшифровки та його значення знає лише дешифратор.

Для шифрування запитів використовувався алгоритм RSA. Він є криптографічним алгоритмом з відкритим ключем, що ґрунтується на

обчислювальній складності завдання факторизації великих цілих чисел. Криптосистема стала першою придатною і для шифрування, і цифрового підпису. Цей алгоритм використовується у великій кількості криптографічних додатків.

Для реалізації цього способу шифрування використовувалися стандартні бібліотеки, які містяться у фреймворку React – crypto і security. Перша шифрує та дешифрує дані за вибраним алгоритмом. Друга бібліотека генерує ключі.

Через створений клас з доступом до шифрування та дешифрування проходять дані, що надсилаються серверу, а також надходять від нього. І в зашифрованому дані зберігаються в базі.

3.3. З'єднання клієнтської та серверної частин

Підключення клієнта до сервера будується на HTTP запитах. На клієнтській частині використовується від React бібліотека Axios, яка дозволяє відправляти HTTP запити на сервер.

Нижче в лістингу буде наведено приклад функції з використанням даної бібліотеки.

```
const getConversations = async () => {
      try {
        const res = await axios.get("/conversations/" +
user._id);
        setConversations(res.data);
      } catch (err) {
        console.log(err);
      }
    };
```

На стороні сервера використовується від платформи Node.js бібліотека Router, вона дозволяє приймати HTTP запити і повертати відповідь

Нижче в лістингу буде наведено приклад функції з використанням даної бібліотеки.

```
router.get("/:conversationId", async (req, res) => {
  try {
    const messages = await Message.find({
      conversationId: req.params.conversationId,
    });
    res.status(200).json(messages);
  } catch (err) {
    res.status(500).json(err);
  }
});
```

Організувавши клієнт-серверне з'єднання, можна вважати розробку програми завершеною.

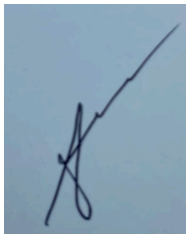
ВИСНОВКИ

У процесі виконання випускної кваліфікаційної роботи було проаналізовано вже існуючі аналоги та обрано інструментальні засоби. Після цього було розроблено месенджер на WEB технологіях.

Як основні інструментальні засоби були обрані NodeJS, React і JavaScript, оскільки вони є перспективними, зручними та широко використовуваними технологіями. NodeJS — досить проста і легка для розуміння, яка добре себе зарекомендувала на ринку розробки мобільних та веб-додатків. React — молодий фреймворк, що швидко розвивається, для розробки WEB-додатків. JavaScript є класикою для розробки WEB додатків.

Реалізований програмний продукт підтримує будь-яку операційну систему, у якої є сучасний браузер.

Програма має можливість модифікацій та додавань нового функціоналу.



ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Frost & Sullivan: Количество опреснительных установок [Электронный ресурс]. Режим доступа: http://www.proatom.ru/modules.php?name=News&file=article&sid=6348
2. Staply – Рабочее пространство Мобильного предприятия [Електронний ресурс] – Режим доступу: https://staply.co
3. Slack: что это за мессенджер и как им пользоваться [Електронний ресурс] – Режим доступу: https://blog.skillfactory.ru/glossary/slack/
4. HipChat — обзор сервиса [Електронний ресурс] – Режим доступу: https://startpack.ru/application/hipchat
5. Что такое Битрикс24? [Електронний ресурс] – Режим доступу: https://www.bitrix24.ru/whatisthis/
6. Веб-приложение [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Веб-приложение
7. Клиент-серверная архитектура [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/495698/
8. Концепция MVC [Електронний ресурс] – Режим доступу: https://ruseller.com/lessons.php?id=666
9. Д. Гудман JavaScript и DHTML. Сборник рецептов. Для профессионалов – СПб.: Питер, 2004, 528с.
10. Основные языки программирования и разметки, применяющиеся в web-разработке [Электронный ресурс] // HTML, CSS, PHP, JavaScript, SQL. Режим доступа: httд://www.codeharmony.ru/materials/125
11. Документация по ReactJS [Электронный ресурс]. Режим доступа: https://reactjs.org/docs/getting-started.html
12. Документация по NodeJS [Электронный ресурс]. Режим доступа: https://nodejs.org/en/docs/

13. MongoDB: что это, для чего используется, особенности [Електронний ресурс] – Режим доступу: https://itglobal.com/ru-ru/company/glossary/mongodb/

14. Что такое веб-сокеты и socket.io [Електронний ресурс] – Режим доступу: https://brander.ua/ru/technologies/socketio

15. СТО 4.2-07-2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности — Введ. 09.01.2014. — Красноярск: ИПК СФУ, 2014. — 60 с.

16. Современный учебник JavaScript [Электронный ресурс]. // Электронная документация. — Режим доступа: httds://learn.lavascridt.ru

17. Назначение, стандарты и преимущества языка SQL [Электронный ресурс] // Лекции.Орг. — Режим доступа: httd://lektsii.orq/5- 77336.html

18. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство: 2-е издание / Б. Маклафлин — Санкт-Петербург: Питер, 2014. — 544 с.

19. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript / Р. Никсон — Санкт-Петербург: Питер, 2013. — 496 с.

20. Диго, С. М. Базы данных: проектирование и использование: учебник / С.М.Диго. — Москва: Финансы и статистика, 2005. — 592 с.

21. Дэвид, Ф. А. JavaScript: учебник / Ф.А. Дэвид. — Москва: АСВ, 2002. — 169 с.