Державний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Програмний модуль «Lit`s Talk» соціальної мережі»

Студента 4 курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення» Крупка Тараса Сергійовича

_____
підпис студента

Гарант освітньої програми кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки Рзаєва Світлана Леонідівна

_____
підпис керівника

Гарант освітньої програми кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки Рзаєва Світлана Леонідівна

_____
підпис гаранта

КИЇВ – 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

Завдання
на випускний кваліфікаційний проєкт студентові

Крупко Тарасу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Програмний модуль «Lit`s Talk» соціальної мережі»

Затверджена наказом ректора від «6» грудня 2022 р. № 3288

2. Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

Мета проєкту розробити програмний модуль соціальної мережі

Об'єкт дослідження процес спілкування в програмному модулю соціальної мережі

Предмет дослідження програмний модуль соціальної мережі

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Загальні положення

1.2. Змістовний опис і аналіз предметної області

1.3. Опис процесу діяльності

1.4 Аналіз успішних онлайн чатів

1.5. Висновок до розділу 1

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Вибір інструментів розробки

2.1.1 Вибір інструментів розробки для написання серверної частини

2.1.2 Вибір інструментів розробки для написання користувацького інтерфейсу

2.2. Моделювання поведінки програмного забезпечення

2.3. Архітектура програмного забезпечення

2.3.1. Опис архітектури програмного забезпечення

2.3.2. Розробка та проектування архітектури бази даних

2.3.3. Розробка та проектування архітектури додатку

2.4. Висновок до розділу 2

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1. Створення інтерфейсу онлайн чата

3.2. Розробка компонентів інтерфейсу

3.3. Опис взаємодії бекенда і фронтенда

3.4. Висновок до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2022*	*21.09.2022*
2.	*Розробка та затвердження завдання на проєкт*	*14.11.2022*	*14.11.2022*
3.	*Вступ та перелік літературних джерел*	*23.12.2022*	*23.12.2022*
4.	*Розділ 1. Аналіз вимог до програмного забезпечення*	*27.01.2023*	*27.01.2023*
5.	*Розділ 2. Моделювання та аналіз програмного забезпечення*	*03.03.2023*	*03.03.2023*
6.	*Розділ 3. Розробка програмного додатку*	*14.04.2023*	*14.04.2023*
7.	*Висновки*	*28.04.2023*	*28.04.2023*
8.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*17.05.2023*	*17.05.2023*
9.	*Підготовка автореферату та презентації доповіді*	*26.05.2023*	*26.05.2023*
10.	*Попередній захист випускного кваліфікаційного проєкту*	*29.05.2023 – 02.06.2023*	
11.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*05.06.2023*	*05.06.2023*
12.	*Здача прошитого випускного кваліфікаційного проєкту на кафедру*	*05.06.2023*	*05.06.2023*
13.	*Публічний захист випускного кваліфікаційного проєкту*		

7. Дата видачі завдання «14» листопада 2022 р.

8. Науковий керівник випускного кваліфікаційного проєкту Рзаєва С.Л.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Рзаєва С.Л.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Крупко Т.С.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента Крупко Т.С

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Рзаєва С.Л.

(прізвище, ініціали, підпис)

Завідувач кафедри _____ Криворучко О. В.

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Випускний кваліфікаційний проєкт на тему: «Програмний модуль «Lit`s Talk» соціальної мережі».

Під час виконання випускного кваліфікаційного проекту, були здобуті теоретичні та практичні навички з проектування клієнт-серверного додатку, був проведений аналіз предметної області, внаслідок чого розроблений програмний модуль соціальної мережі.

Ключові слова: мова програмування C#, прикладний програмний інтерфейс, ASP.NET, IdentityServer, Angular, бекенд, фронтенд, програмний додаток, архітектура, база даних.

ABSTRACT

Graduation qualification project on the topic: "Software module "Lit's Talk" social network".

During the completion of the final qualification project, theoretical and practical skills were acquired in designing a client-server application, an analysis of the subject area was carried out, as a result of which a software application of a social network was developed.

Keywords: C# programming language, application programming interface, ASP.NET, IdentityServer, Angular, back-end, front-end, software application, architecture, database.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

JSON (англ. JavaScript Object Notation) – розширювана мова розмітки

Бекенд – серверна частина додатку

Фронтенд – користувацький інерфейс

IdentityServer - сервер автентифікації, який реалізує стандарти OpenID Connect (OIDC) і OAuth 2.0 для ASP.NET Core

SignalR - бібліотека для розробників ASP.NET, яка спрощує процес додавання веб-функцій реального часу до програм

Веб-сокет - протокол зв'язку поверх TCP-з'єднання, призначений для обміну повідомленнями між браузером та веб-сервером у режимі реального часу

ЗМІСТ

					ДТЕУ 121 06-16.БР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		23.12.22	Програмний модуль «Lit`s Talk» соціальної мережі	*Стадія*	*Арку*	*Аркушів*
Керівник		Рзаєва С.Л.		23.12.22		*З*	*3*	*46*
Гарант		Рзаєва С.Л.		23.12.22		*Факультет інформаційних технологій 4 курс, 6 група*		
Розробив		Крупко Т.С.		23.12.22	*Зміст*			

ВСТУП

Актуальність. Соціальні мережі є важливою складовою сучасної соціальної взаємодії та комунікації. Важливою складовою соціальних мереж є онлайн-чати, які дозволяють легко спілкуватися будь-де і будь-коли та пропонують широкий спектр переваг як для приватних осіб, так і для компаній.

Спілкування в режимі реального часу - одна з ключових переваг онлайн-спілкування. Онлайн-чати дозволяють користувачам миттєво розпочинати розмову, що дає змогу обмінюватися ідеями та думками швидше та ефективніше, ніж за допомогою електронної пошти.

Онлайн-чати також забезпечують зручність, дозволяючи користувачам спілкуватися один з одним з пристроїв таких як комп'ютери, мобільні телефони та планшети, незалежно від відстані, місцезнаходження. Онлайн-чати також зачасту безкоштовні, що робить їх доступним інструментом спілкування як для окремих осіб, так і для корпорацій.

Онлайн-чати дають багато переваг, але вони також мають деякі недоліки, про які повинні знати як користувачі, так і постачальники послуг.

Розуміючи складність вищезазначеного питання, безумовно та об'єктивно його вирішення можна змоделювати на прикладі розробки програмного модуля соціальної мережі.

Мета дослідження: розробити програмний модуль соціальної мережі.

Об'єкт дослідження: процес спілкування в програмному модулю соціальної мережі.

*Предмет дослідження***:** розробка програмного модулю соціальної мережі.

У відповідності з метою дослідження поставлені наступні *завдання*:

- проаналізувати наявні програмні забезпечення;
- розробити технічне завдання;
- побудувати архітектуру програмного модуля;
- розробити програмний модуль соціальної мережі.

Методи дослідження: аналіз існуючих рішень та літератури, вивчення специфікацій та документації відповідних технологій, розробка прототипів системи.

Практичне значення дослідження: створення повнофункціонального онлайн-чату, який можна використовувати в реальних ситуаціях спілкування та співпраці.

РОЗДІЛ 1
АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Загальні положення

Онлайн-чати мають довгу історію та є одним із основних засобів комунікації в інтернеті. На початку розвитку Інтернету, чати були представлені у вигляді текстових інтерфейсів, де користувачі могли обмінюватись повідомленнями в режимі реального часу на основі командного рядка.

З появою графічного інтерфейсу та розвитком веб-технологій, чати стали доступнішими та зручнішими для користувачів. Наприкінці 1990-х і на початку 2000-х років популярність онлайн-чатів суттєво зросла, і з'явилися різні платформи та сервіси, що пропонують функціонал чату.

Проте, з розвитком онлайн-чатів виникли проблеми, які вимагали рішення. Деякі з цих проблем включають:

1. Безпека та конфіденційність: Користувачі зіткнулися з загрозою безпеці, такою як злом облікових записів, витоку особистої інформації та зловживання відомостями про користувачів.
2. Масштабованість: Зростаюча кількість користувачів призводила до проблем масштабованості та продуктивності. Необхідно було розробляти архітектури, здатні ефективно обробляти велику кількість запитів та забезпечувати стабільну роботу навіть за високого навантаження.

3. Модерація та контроль: Онлайн-чати зіткнулися з проблемами неприйнятної поведінки, масового спаму та поширення небажаного контенту. Були розроблені інструменти та алгоритми для модерації та контролю контенту, такі як фільтри та системи виявлення ненормативної лексики.

У сучасних онлайн-чатах проблеми масштабованості, безпеки та контролю постійно вирішуються та вдосконалюються. Розробники активно використовують новітні технології та методи для забезпечення надійного та безпечного функціонування онлайн-чатів. Наприклад, використання хмарних рішень та розподілених систем дозволяє забезпечити високу масштабованість та відмовостійкість.

1.2. Змістовний опис і аналіз предметної області

Соціальна мережа - це мережа людей, які зустрічаються в Інтернеті для спілкування, розміщуючи інформацію та зображення, залишаючи коментарі чи надсилаючи повідомлення. Учасники можуть розширити свої особисті та ділові контакти, зв'язавшись з іншими на веб-сайтах соціальних мереж та в додатках. [1]

Онлайн-чат — це програма, яка дозволяє спілкуватися в режимі реального часу з іншими людьми через Інтернет. До нього можна отримати доступ безпосередньо в браузері або спілкуватися через додаток. [2]

Онлайн чати пропонують ряд переваг та недоліків, які слід враховувати під час їх використання. До плюсів онлайн чатів:

Онлайн чати дозволяють людям обмінюватися повідомленнями в режимі реального часу, що забезпечує швидку та миттєву комунікацію, незалежно від відстані та часових зон.

1. Чати доступні на різних платформах, включаючи комп'ютери, смартфони та планшети, що робить їх зручними для використання у будь-якій ситуації та місці.
2. Онлайн чати дозволяють спілкуватися з людьми з різних країн та культур, що сприяє глобальній комунікації та розширює межі спілкування.

Мінуси онлайн чатів:

1. Онлайн-чати обмежені текстовими повідомленнями, що може ускладнити передачу емоцій, інтонацій та невербальних виразів.
2. В онлайн-чатах існує ризик отримання спаму, фішингових атак, небажаних контактів або потенційного витоку конфіденційної інформації.
3. Залежність від Інтернет-з'єднання: Використання чату онлайн вимагає постійного підключення до Інтернету, що може бути проблематично у разі відсутності стабільного з'єднання або поганого зв'язку.
4. В онлайн-чатах може виникати ризик витоку особистої інформації або приватних діалогів. Несанкціонований доступ або недостатній захист даних можуть призвести до порушення приватності користувачів.

1.3. Опис процесу діяльності

З точки зору користувача, робота з онлайн-чатом включає наступні дії:

1. Реєстрація або вхід до системи: Користувач повинен створити обліковий запис, якщо він новий користувач, або увійти до системи, використовуючи свої облікові дані (логін та пароль), якщо він уже зареєстрований.
2. Пошук та додавання контактів: Користувач може шукати інших користувачів, з якими він хоче спілкуватися, або додавати контакти зі списку рекомендацій або за наявними контактами.
3. Початок чату: Після знаходження бажаного контакту користувач може почати чат, вибравши його зі списку контактів і натиснувши кнопку "Почати чат" або надіславши запрошення на чат.

4. Обмін повідомленнями: Користувач може надсилати текстові повідомлення контакту в режимі реального часу. Він може вводити текст у поле введення повідомлень і натискати кнопку надсилання, щоб його повідомлення було доставлене контакту.
5. Отримання та читання повідомлень: Коли користувач надсилає повідомлення, його контакт отримує повідомлення та може прочитати повідомлення у своєму чаті. Користувач може також отримувати та читати повідомлення, надіслані його контактом.
6. Взаємодія з повідомленнями: Користувач може реагувати на повідомлення, наприклад, відповідати на них, ставити запитання, висловлювати свою думку або надсилати емоційні реакції, використовуючи відповідні функції, що надаються чатом.
7. Закриття чату: Коли користувач закінчив спілкування або тимчасово хоче припинити чат, він може закрити чат або вийти з нього. Чат зберігається, і користувач може будь-коли повернутися до нього, щоб продовжити спілкування.
8. Онлайн-чат надає користувачеві зручний спосіб спілкування з іншими людьми через текстові повідомлення, забезпечуючи швидку та миттєву комунікацію в режимі реального часу.

1.4. Аналіз успішних онлайн чатів

Аналіз успішних онлайн чатів дозволяє виділити деякі ключові фактори, що сприяють їхньому успіху. Далі наведені приклади успішних онлайн чатів та особливості, які роблять їх популярними:

1. WhatsApp - один із найпопулярніших месенджерів у світі, який пропонує миттєве спілкування через текстові повідомлення, аудіо- та відеодзвінки. Він отримав широке визнання завдяки своїй простоті використання, широкій платформній підтримці та високому рівню конфіденційності.

2. Facebook Messenger – популярний месенджер, інтегрований із соціальною мережею Facebook. Він пропонує обмін повідомленнями, голосові та відеодзвінки, а також додаткові функції, такі як ігри, стікери та можливість платежів.
3. Slack - комунікаційна платформа, призначена для командної роботи та обміну повідомленнями у робочому середовищі. Він надає можливість створення каналів, групових чатів та прямих повідомлень, а також інтеграцію з різними інструментами та сервісами. Slack дозволяє впорядкувати комунікацію всередині команди, забезпечує швидку та ефективну взаємодію між співробітниками та підвищує продуктивність роботи.
4. Telegram – месенджер, який славиться своєю високою швидкістю доставки повідомлень та широким набором функцій. Він підтримує надсилання текстових повідомлень, фотографій, відео, аудіо та інших типів файлів. Telegram також пропонує функцію повідомлень, що самознищуються, можливість створення публічних каналів і групових чатів. Він забезпечує високий рівень безпеки та шифрування даних.
5. Viber – популярний месенджер, який пропонує миттєве спілкування через текстові повідомлення, аудіо- та відеодзвінки. Він забезпечує шифрування кінцевого пристрою, щоб забезпечити безпеку повідомлень та приватність користувачів.

Таблиця 1.1 Ключові особливості успішних онлайн чатів

Месенджер	Текстові повідомлення	Голосові повідомлення	Відеовиклики	Передача файлів	Створення каналів	Публічні канали	Інтеграції
WhatsApp	✓	✓	✓	✓	✓	✗	✗
Facebook Messenger	✓	✓	✓	✓	✓	✗	✓
Slack	✓	✓	✗	✓	✓	✗	✓
Telegram	✓	✓	✓	✓	✓	✓	✗
Viber	✓	✓	✓	✓	✓	✗	✗

Дивлячись на таблицю 1.1, можна виділити ключові особливості успішних онлайн чатів: пропонують широкий спектр функцій, забезпечують швидку та надійну доставку повідомлень, мають інтуїтивно зрозумілий інтерфейс та високий рівень безпеки. Вони надають користувачам зручність, ефективність та можливість спілкуватися та взаємодіяти з іншими людьми в режимі реального часу.

1.5 Розробка технічного завдання

1.5.1 Розробка функціональних вимог

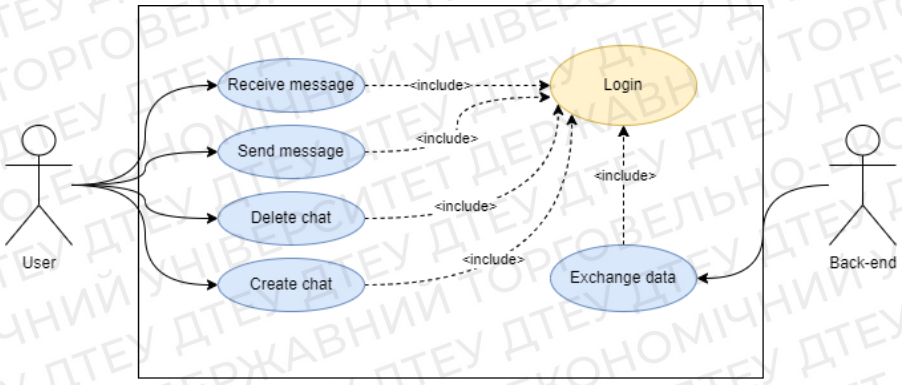


Рис. 1.1 – Use-Case діаграма

Джерело: побудовано автором за допомогою сервісу [3]

Наведена діаграма складається зі сценарію використання додатка. Згідно діаграми, користувач має чотири загальні дії:

1. Створити чат
2. Видалити чат
3. Написати повідомлення
4. Отримати повідомлення

Всі ці дії можуть бути виконані, якщо користувач пройшов авторизацію. Якщо користувач авторизований, то діаграма може виконувати додаткову дію – обмін даними на бекенд сервері.

Таблиця 1.2 Функціональні вимоги та їх пріорітети.

Функціональна вимога	**Пріоритет**
Відправка текстових повідомлень	Високий
Передача файлів	Середній
Створення групових чатів	Середній
Створення публічних каналів	Середній
Інтеграція з іншими сервісами	Низький
Підтримка платежів	Низький

Вище наведена таблиця функціональних вимог до онлайн-чату та їх пріоритети. Функціональні вимоги оцінені за пріоритетом, де "Високий" означає найважливіші вимоги, "Середній" - середню важливість, а "Низький" - менш критичні вимоги. Це дозволяє визначити основні функції, які мають бути реалізовані в першу чергу, щоб задовольнити потреби користувачів.

1.5.2 Розробка нефункціональних вимог

Нижче наведено список нефункціональних вимог до онлайн-чату:

1. Безпека: Забезпечення захисту даних та конфіденційності користувачів, включаючи шифрування повідомлень та захист від несанкціонованого доступу.
2. Надійність: Забезпечення стабільної та безперервної роботи чату, мінімізація можливих збоїв та аварійних ситуацій.
3. Продуктивність: Забезпечення високої швидкості передачі повідомлень та мінімальних затримок під час використання чату.
4. Доступність: Гарантована доступність сервісу для користувачів у будь-який час та на різних платформах.
5. Зручність використання: Інтуїтивно зрозумілий інтерфейс, простота навігації та можливість легко освоїти та використовувати чат навіть для нових користувачів.
6. Сумісність: Сумісність чату з різними операційними системами, браузерами та пристроями, щоб користувачі могли отримати доступ до чату з будь-якого пристрою.
7. Адаптивний дизайн: Адаптація інтерфейсу чату під різні розміри екранів та пристроїв, включаючи мобільні телефони, планшети та комп'ютери.
8. Підтримка багатоплатформенності: Можливість використання чату на різних платформах, включаючи операційні системи Windows, MacOS, Linux, а також мобільні платформи iOS та Android.
9. Розширюваність та наявність API: Надання відкритого API для розробників, щоб вони могли розширювати функціональність чату, створювати інтеграції з іншими системами та розробляти додаткові модулі та плагіни.

10. Збереження історії повідомлень: Автоматичне збереження історії повідомлень, щоб переглянути та шукати попередні листування, а також архівувати та скопіювати дані чату.

Ці нефункціональні вимоги відіграють важливу роль у створенні зручного, безпечного та гнучкого онлайн чату, який відповідає потребам користувачів та надає надійне та задовільне враження.

1.5.3 Розробка технічного завдання

Метою розробки онлайн-чату є створення сучасного та функціонального комунікаційного інструменту, який дозволить користувачам обмінюватися повідомленнями в режимі реального часу. Головна мета полягає у наданні зручного, надійного та безпечного засобу зв'язку для користувачів, який задовольнятиме їх потреби у комунікації.

Онлайн-чат дозволить користувачам спілкуватися з іншими людьми, включаючи індивідуальні бесіди та групові обговорення. Чат забезпечить простий та інтуїтивно зрозумілий інтерфейс для обміну текстовими повідомленнями, а також можливість надсилання медіа-файлів, таких як фотографії та відео.

Мета розробки онлайн чату полягає у створенні інструменту, який буде легко доступний та використовуватися як настільними комп'ютерами, так і мобільними пристроями. Користувачі зможуть спілкуватися з людьми по всьому світу та насолоджуватися швидкою та надійною передачею повідомлень. Для досягнення цієї мети необхідно вирішити наступні задачі:

1. Обрати мову програмування для написання серверної частини
2. Обрати мову програмування для написання користувацького інтерфейсу
3. Обрати середовище розробки для бекенд та фронтенд частин
4. Обрати базу даних для збереження даних користувачів
5. Обрати інструменти для побудови зв’язку між клієнтом і сервером

6. Створити систему обробки користувацьких запитів
7. Створити користувацький інтерфейс для взаємодії з сервером
8. Додати можливість користувачу зареєструватися та авторизуватися
9. Додати в користувацький інтерфейс можливості по створенню чату та спілкуванні в ньому
10. Розробити адаптивний дизайн
11. Створити базу даних для зберігання даних користувачів.

1.6. Висновки до розділу 1

В результаті аналізу було розглянуто актуальність та значущість теми, проведено аналіз вимог та позначено ключові особливості онлайн чатів. Були опрацьовано успішні приклади існуючих онлайн чатів та їх функціональність.

Також було визначено, що онлайн-чати відіграють важливу роль у сучасній комунікації, забезпечуючи людей можливість швидкого та зручного обміну повідомленнями. Вони мають ряд переваг, таких як миттєвість передачі повідомлень, гнучкість у використанні та можливість встановлення зв'язку в режимі реального часу. Онлайн чати стають все більш популярними серед користувачів, перетворюючись на важливий інструмент як особистої, так і в діловій сфері.

Розробка онлайн чату є важливим завданням, що вимагає використання сучасних технологій і гарного розуміння вимог користувачів. Технічне завдання визначило необхідну функціональність та основні аспекти розробки, такі як середовище розробки, використовувані технології, архітектура застосування, тестування та документація.

В результаті розробки онлайн чату очікується створення зручного та надійного інструменту комунікації, здатного задовольнити потреби користувачів у швидкій та безпечній передачі повідомлень. Розробка такого

чату має потенціал залучити широку аудиторію користувачів та принести значну користь як в особистій, так і професійній сферах.

Таким чином, розробка онлайн чату є перспективним та цікавим завданням, яке дозволить створити сучасний інструмент комунікації, що задовольняє високі вимоги користувачів.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір інструментів розробки

2.1.1 Вибір інструментів для написання серверної частини

При розробці зручного інтерфейсу для онлайн-чату є кілька поширених мов програмування, які забезпечують ефективність і гнучкість у створенні веб-додатків. Ці мови включають в себе:

1. Java - є однією з найпоширеніших мов програмування для розробки веб-застосунків. Він має широке співтовариство розробників, багатий набір інструментів і фреймворків, таких як Spring і JavaEE, які полегшують створення бекенда з високою продуктивністю і масштабованістю. [4]
2. Python також користується популярністю у сфері веб-розробки завдяки своїй простоті та читання коду. Він має велику бібліотеку фреймворків, таких як Django та Flask, які спрощують створення бекенду з мінімальними зусиллями. [5]
3. Ruby, особливо з використанням фреймворку Ruby on Rails, є популярним вибором для розробки веб-додатків. Він забезпечує швидку та елегантну розробку за допомогою своєї простоти та конвенцій.
4. Node.js дозволяє використовувати JavaScript як мову серверної розробки. Він має високу продуктивність і неблокуючу архітектуру, що робить його ідеальним вибором для створення масштабованих і реактивних веб-додатків.

Платформа ASP.NET надає середовище для створення веб-застосунків мовою C#. Вона включає потужний набір інструментів і фреймворків, таких як ASP.NET MVC (Model-View-Controller) і ASP.NET Core, які дозволяють розробляти високопродуктивні і масштабовані бекенд-системи.[6]

Більше того, розробка мовою C# та платформою ASP.NET нерозривно пов'язана з використанням інтегрованого середовища розробки (IDE) Visual Studio. Visual Studio надає розробникам потужні інструменти для створення, налагодження та розгортання програм. Загалом використання Visual Studio у поєднанні з мовою C# і платформою ASP.NET дозволяє розробникам створювати високоякісні, надійні та масштабовані системи.

Можна виділити кілька переваг, які роблять їх привабливими порівняно з мовами, що конкурують, і платформами:

1. Інтеграція з екосистемою Microsoft: C# та ASP.NET розроблені Microsoft, що означає глибоку інтеграцію з іншими продуктами та інструментами, такими як Visual Studio та Azure.
2. Багатий набір інструментів та фреймворків: C# та ASP.NET мають широкий вибір фреймворків та бібліотек, які полегшують розробку веб-додатків.
3. Висока продуктивність: C# є компілюваною мовою, що забезпечує її високу продуктивність. Завдяки цьому веб-програми, розроблені на базі C# і ASP.NET, можуть обробляти великі обсяги даних і витримувати високі навантаження.
4. Безпека: C# має сильну типізацію та сувору систему безпеки типів, що допомагає запобігти безлічі помилок і вразливостей, пов'язаних з типами даних.
5. Підтримка спільноти розробників: C# і ASP.NET мають активну та велику спільноту розробників. Це означає наявність великої кількості

документації, навчальних матеріалів, форумів та блогів, де можна отримати підтримку та поради від досвідчених розробників.

6. Інструменти для тестування: C# та ASP.NET надають багатий набір інструментів для тестування програм.
7. Екосистема .NET: C# та ASP.NET базуються на платформі .NET, яка пропонує широкий набір інструментів та бібліотек для розробки різноманітних програм.

В якості місця зберігання інформації існує безліч СУБД (систем управління базами даних) з різними особливостями та функціональністю. Нижче наведено короткі описи деяких популярних СУБД:

1. MySQL: MySQL є однією з найбільш широко використовуваних реляційних СУБД. Вона відрізняється високою продуктивністю, надійністю та простотою у використанні.
2. Oracle: Oracle Database є потужною реляційною СУБД, що широко застосовується в підприємствах. Вона пропонує високу продуктивність, розширені можливості безпеки, управління транзакціями та масштабованість. Однак, Oracle зазвичай потребує більш високих витрат на ліцензування та підтримку.
3. Microsoft SQL Server: SQL Server є реляційною СУБД, розробленою Microsoft. Вона має широкі можливості для розробки та управління базами даних, включаючи інтеграцію з іншими продуктами Microsoft. SQL Server має гарну масштабованість та інструменти для бізнес-аналітики.
4. PostgreSQL: PostgreSQL є потужною та гнучкою реляційною СУБД з відкритим вихідним кодом. Вона має широкий спектр функцій, включаючи підтримку тригерів, процедур, що зберігаються, реплікації та географічних даних. PostgreSQL також славиться своєю надійністю, розширюваністю та сумісністю з ANSI SQL.

Після аналізу перерахованих вище СУБД і з урахуванням вимог для онлайн чату, рекомендується вибрати PostgreSQL. Крім того, PostgreSQL також є привабливим вибором з точки зору вартості. PostgreSQL є системою з відкритим вихідним кодом та розповсюджується за ліцензією PostgreSQL, яка дозволяє використовувати її безкоштовно та без обмежень. [7]

Загалом, вибір мови C#, платформи ASP.NET та СУБД PostgreSQL для розробки онлайн чату забезпечує безліч переваг, включаючи високу продуктивність, безпеку, масштабованість та підтримку спільноти розробників. Це дозволяє створити надійний, потужний і гнучкий додаток, здатний задовольнити потреби користувачів в ефективної комунікаційній платформі чату.

2.1.2 Вибір інструментів для написання користувацького інтерфейсу

Існує кілька популярних фреймворків для розробки фронтенду, що надають зручність та потужні інструменти. Один з таких фреймворків – Angular. Angular - це платформа і фреймворк для створення веб-додатків, що масштабуються. Він заснований мовою TypeScript і має модульну архітектуру, компонентний підхід, вбудовану маршрутизацію та інші можливості, які роблять розробку фронтенду більш ефективною.

Angular пропонує широкий набір інструментів і функціональності, які допоможуть створити потужні та інтуїтивно зрозумілі інтерфейси користувача. Використовуючи Angularє можливість легко керувати станом програми, створювати компоненти, що перевикористовуються, забезпечувати навігацію між сторінками та інші важливі функції. Крім того, Angular активно підтримується спільнотою розробників, що забезпечує стабільність та надійність фреймворку.

Для зручної розробки на Angular рекомендується використовувати IDE Visual Studio Code. Visual Studio Code - це легковагове та потужне інтегроване середовище розробки, яке надає широкий набір інструментів для створення та налагодження коду. Вона володіє зручним інтерфейсом користувача, підсвічуванням синтаксису, автодоповненням коду, інтеграцією з Git і багатьма іншими корисними функціями. Завдяки своїй гнучкості та можливості розширення за допомогою плагінів, Visual Studio Code стає ідеальним інструментом для розробки на Angular.

2.2 Моделювання поведінки програмного забезпечення

Діаграма стану - це інструмент для моделювання та візуалізації різних станів, переходів та поведінки системи. У контексті розробки онлайн-чату, діаграма архітектури станів допомагає описати різні стани, в яких може перебувати користувач під час взаємодії з чатом. Нижче наведена ця діаграма та опис до неї.

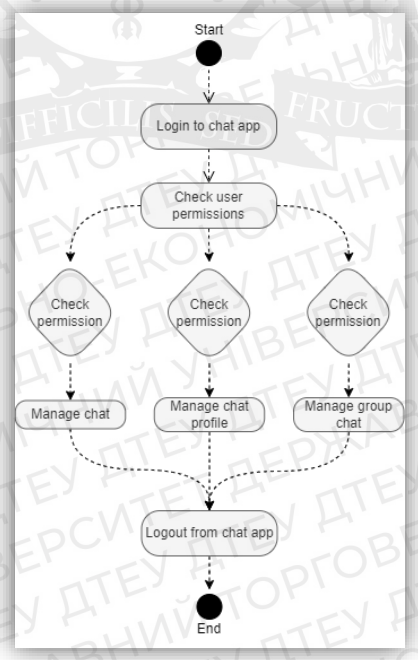


Рис. 2.1. Діаграма стану

Джерело: побудовано автором за допомогою сервісу [3]

Згідно з діаграмою наведеною на рис. 2.1, відбуваються наступні дії:

1. Користувач проходить процес авторизації в додатку.
2. Проводиться перевірка даних користувача.
3. На кожному етапі система перевіряє доступ користувача до виконання певних дій.
4. Користувач може вийти зі свого облікового запису.
5. Процес завершується.

Діаграма послідовності використання - це візуальне представлення порядку взаємодії між різними об'єктами системи в конкретному варіанті використання. Вона показує порядок операцій і зв'язок між об'єктами в рамках певного сценарію.

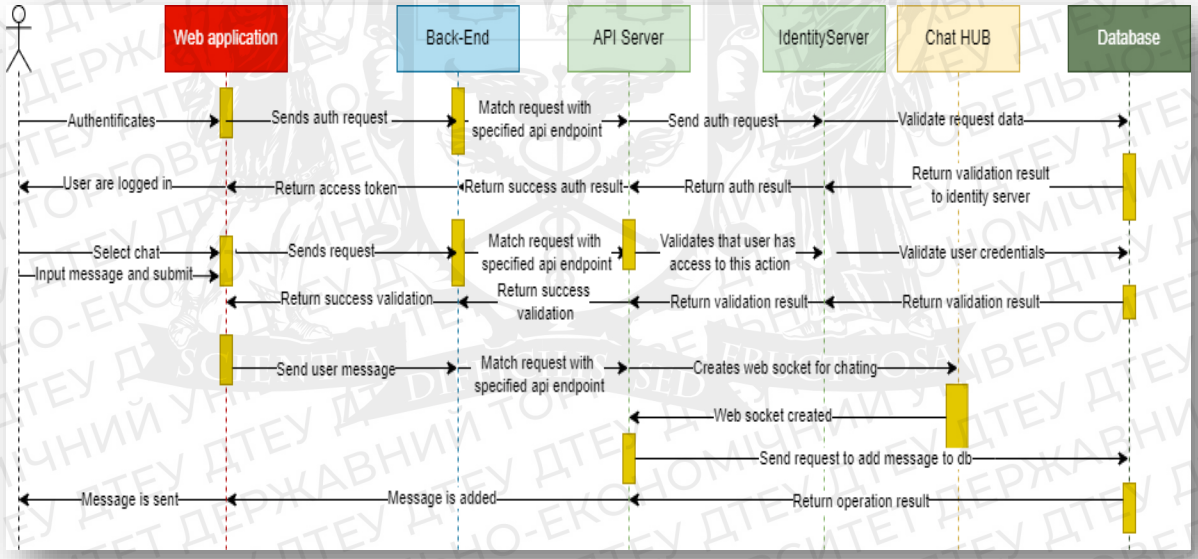


Рис. 2.2. Діаграма послідовності використання

Джерело: побудовано автором за допомогою сервісу [3]

На рис. 2.2 показано сценарій використання програми та послідовність дій. Відповідно до цієї послідовності описані наступні кроки:

1. Користувач заходить на сайт і авторизується.

2. Веб-додаток надсилає запит до бек-енду, який зв'язується з сервером автентифікації IdentityServer для перевірки введених даних і повертає результат автентифікації.
3. Після успішної автентифікації користувач може вибрати кімнату чату і відправити повідомлення.
4. Як тільки користувач входить до чату, на веб-сокеті створюється сеанс, що дозволяє миттєву передачу даних.
5. Коли користувач надсилає повідомлення, воно спочатку надсилається в чат-хаб, а потім додається до бази даних.
6. Як тільки повідомлення додано, поточний користувач і всі учасники чату можуть його побачити.

2.3 Архітектура програмного забезпечення

Для розробки онлайн-чату слід використовувати патерн MVC (Model-View-Controller)[8]. Цей патерн є одним з найбільш поширених і широко застосовуваних веб-розробок. MVC поділяє додаток на три основні компоненти:

1. Модель (Model): Відповідає за обробку даних та бізнес-логіку програми. Тут визначено об'єкти та методи, які обробляють дані, взаємодіють з базою даних та виконують операції, пов'язані з бізнес-процесами програми. Модель є основним джерелом даних для відображення в інтерфейсі користувача.
2. Відображення (View): Відповідає за відображення даних користувачеві. Подання визначає структуру і зовнішній вигляд інтерфейсу користувача, включаючи елементи управління, макети, стилі та інші аспекти візуалізації. Воно отримує дані від моделі та представляє їх користувачу у зрозумілій та зручній формі.

3. Контролер (Controller): Відповідає за керування потоком даних та взаємодію між моделлю та поданням. Контролер обробляє дії користувача, такі як натискання кнопок, відправлення форм та інші події. Він отримує дані від подання, обробляє їх та взаємодіє з моделлю для виконання відповідних операцій.

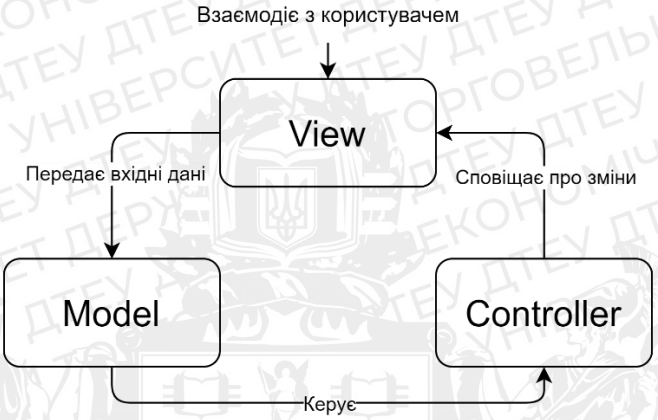


Рис. 2.3 – Модель роботи патерну MVC

Джерело: побудовано автором за допомогою сервісу [3]

На рис. 2.3 наедений паттерн MVC забезпечує чіткий поділ відповідальності між компонентами програми, що сприяє більш ефективній розробці, підтримці та масштабуванню коду. Він також дозволяє легко змінювати зовнішній вигляд та функціональність програми, не торкаючись його бізнес-логіки.

Використання паттерна MVC при розробці онлайн чату допомагає створити структурований код, що легко підтримується, полегшує розширення функціональності і покращує загальну продуктивність програми.

Для написання додатку було вирішено використовувати мову програмування C# на платформі ASP.NET в поєднанні за базою даних PostgreSQL для написання серверної частини, а для створення користувацького інтерфейсу – Angular.

2.3.1 Опис архітектури програмного забезпечення

Загалом програмне забезпечення складається з трьох основних частин: Model, View, Controller. Де, View – візуальна частина, а Model та Controller – серверна частина. Серверна частина в свою чергу розподілена на два основних компоненти: сервер авторизації (Identity Server) та сервер для обробки і зберігання даних, що надходять від користувацького інтерфейсу (API Server). Кожна дія зроблена користувачем спочатку валідується сервером авторизації, і тільки після цього надає дозвіл на виконання відповідних дій основному серверу. Нижче на рис. 2.4 наведена загальна архітектура додатку.

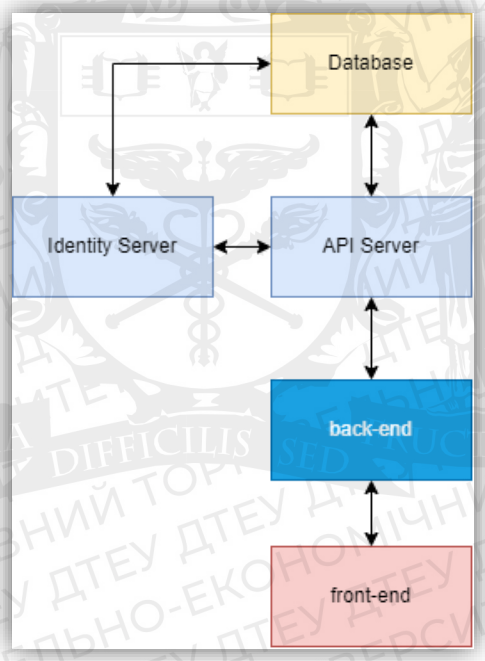


Рис. 2.4: Загальна архітектура додатку

Джерело: побудовано автором за допомогою сервісу [3]

2.3.2 Розробка та проектування архітектури бази даних

Розробка та проектування бази даних є важливим етапом у створенні програмного забезпечення. Процес розробки має в собі створення концептуальної, логічної і фізичної моделей бази даних.

Першим етапом розробки бази даних було проектування концептуальної моделі бази даних. Концептуальна модель - це високорівневе представлення даних та їхніх зв'язків, яке не залежить від конкретної реалізації. [9]

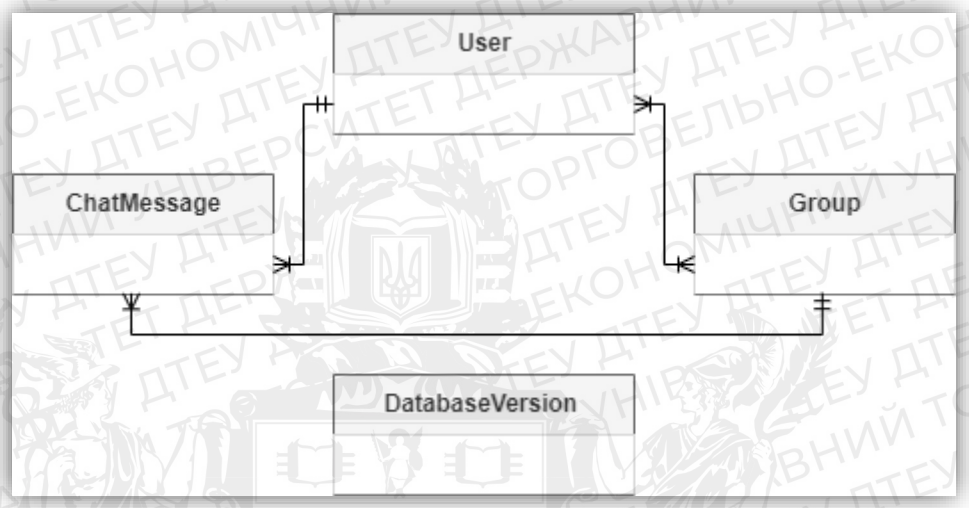


Рис. 2.5: Концептуальна модель бази даних

Джерело: побудовано автором за допомогою сервісу [3]

На рис. 2.5 наведена концептуальна модель бази даних, її ідея полягає в тому, що один User(користувач) або Group(чат) можуть мати багато ChatMessage(повідомлення), також один User(користувач) може мати багато Group(чат) і навпаки. Також важливим елементом є таблиця DatabaseVersion, що буде використовуватись для контролю версій бази даних.

Наступним етапом розробки бази даних було проектування логічної моделі. Логічна модель – це розширення концептуальної моделі бази даних. Вона має в собі в себе всі сутності, атрибути, ключі та взаємозв'язки. Логічна модель є абстрактним представленням даних і незалежить від конкретної реалізації у фізичній базі даних. [9]

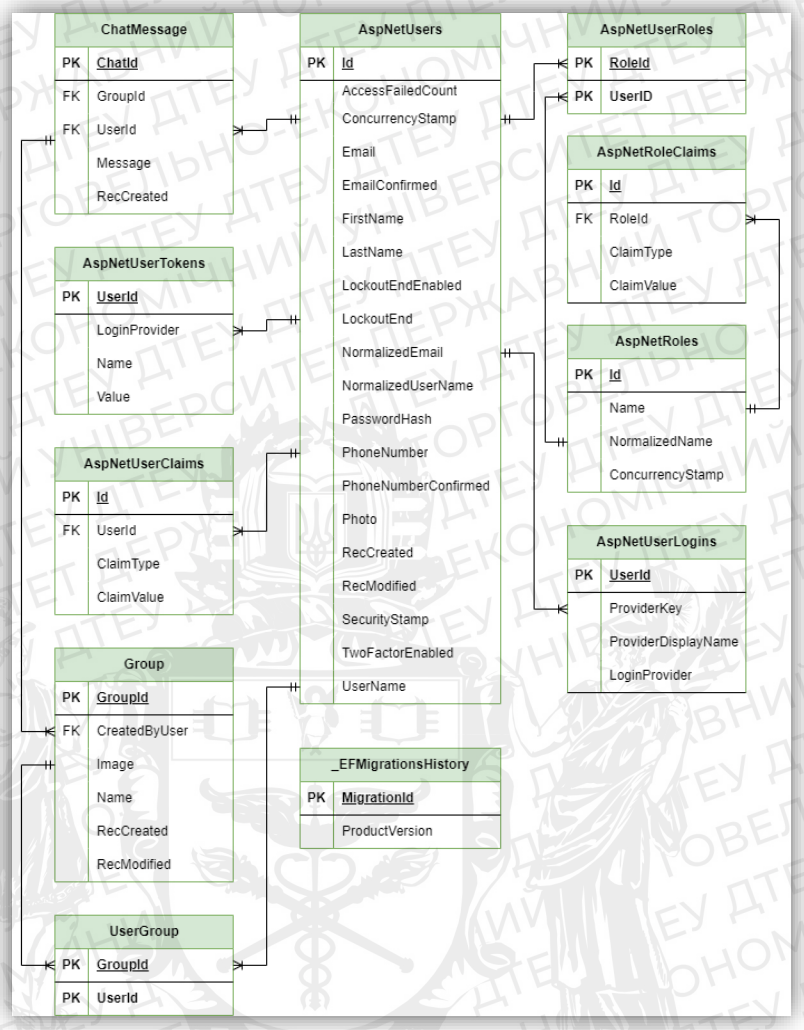


Рис. 2.6: Логічна модель бази даних

Джерело: побудовано автором за допомогою сервісу [3]

На рис. 2.6 наведена логічна модель бази даних, яка розширила концептуальну модель не тільки за кількістю таблиць, а і надала представлення того, які дані будуть зберігатись в таблицях. Таблиці, що мають приставку «AspNet» надані ORM системою Entity Framework Core, але для реалізації чату вони були розширені додатковими полями.

Після завершення логічного проектування важливим кроком стало фізичне проектування бази даних. На цьому етапі логічна модель

трансформувалася у фізичну модель, яка визначала атрибути та типи даних. Також були враховані аспекти продуктивності, такі як створення індексів для прискорення виконання запитів.

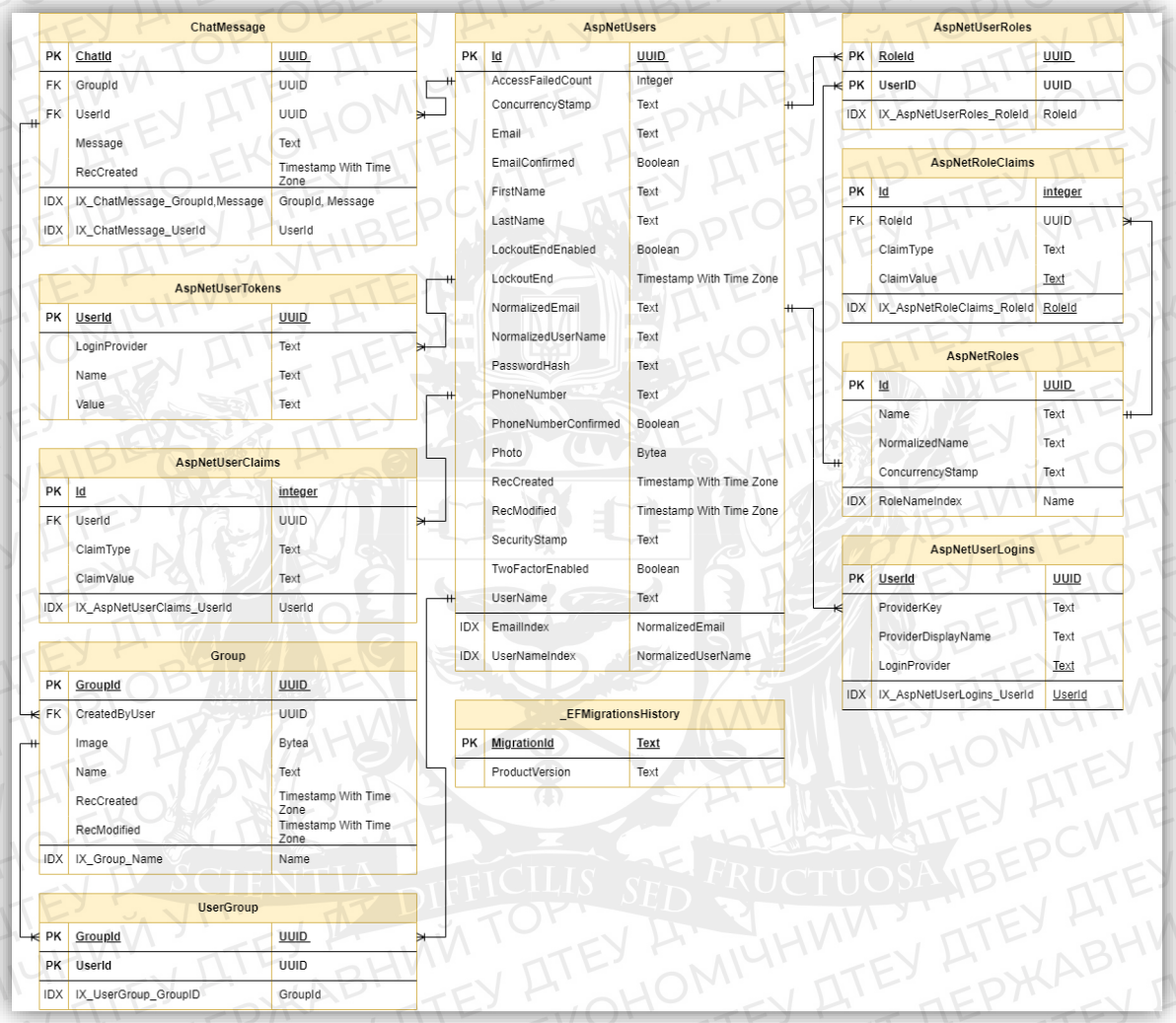


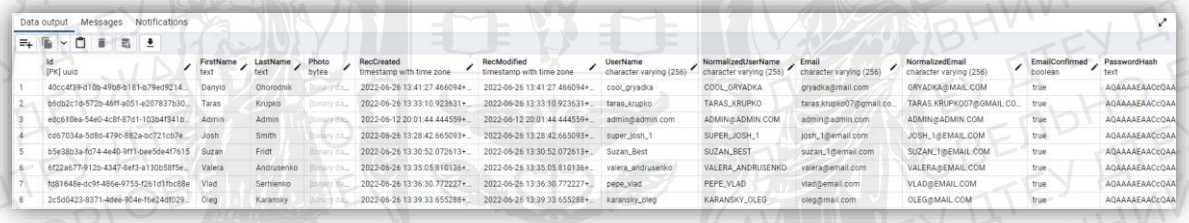
Рисунок 2.7: Фізична модель бази даних

Джерело: побудовано автором за допомогою сервісу [3]

На рис. 2.7 наведена фізична модель бази даних, яка є важливим компонентом системи, оскільки дозволяє ефективно зберігати та організовувати дані, виконувати різні операції з базою даних, а також забезпечує надійність та безпеку даних.

База даних складається з 11 таблиць, що містять такі сутності, як AspNetUsers, AspNetRoles, AspNetUserRoles, AspNetRoleClaims, AspNetUserClaims, AspNetUserLogins, ChatMessage AspNetUserTokens, Group, UserGroup, _EFMigrationsHistory та інші сутності.

AspNetUsers - це загальна модель, що містить дані про користувачів і є основним компонентом, який пов'язаний з іншими таблицями; вона зберігає важливу інформацію про користувачів, таку як ідентифікатор, ім'я, адреса електронної пошти та пароль. Ця таблиця використовується для встановлення зв'язків з іншими таблицями та забезпечення належного функціонування системи. Приклад заповнення наведений нище на рис. 2.8.



	Id [PK] uuid	FirstName text	LastName text	Photo bytea	RecCreated timestamp with time zone	RecModified timestamp with time zone	UserName character varying (256)	NormalizedUserName character varying (256)	Email character varying (256)	NormalizedEmail character varying (256)	EmailConfirmed boolean	PasswordHash text
1	40cc4f39-d10b-49b8-b181-b79ed9214...	Danylo	Ohorodnik	[binary da...	2022-06-26 13:41:27.466094+...	2022-06-26 13:41:27.466094+...	cool_gryadka	COOL_GRYADKA	gryadka@mail.com	GRYADKA@MAIL.COM	true	AQAAAAEAACcQAA
2	b6db2c1d-572b-46ff-a051-e207837b30...	Taras	Krupko	[binary da...	2022-06-26 13:33:10.923631+...	2022-06-26 13:33:10.923631+...	taras_krupko	TARAS_KRUPKO	taras.krupko07@gmail.co...	TARAS.KRUPKO07@GMAIL.CO...	true	AQAAAAEAACcQAA
3	edc610ea-54e0-4c8f-8701-103b4f341b...	Admin	Admin	[binary da...	2022-06-12 20:01:44.444559+...	2022-06-12 20:01:44.444559+...	admin@admin.com	ADMIN@ADMIN.COM	admin@admin.com	ADMIN@ADMIN.COM	true	AQAAAAEAACcQAA
4	cb67034a-5d8d-479c-882a-bc721cb7e...	Josh	Smith	[binary da...	2022-06-26 13:28:42.665093+...	2022-06-26 13:28:42.665093+...	super_josh_1	SUPER_JOSH_1	josh_1@email.com	JOSH_1@EMAIL.COM	true	AQAAAAEAACcQAA
5	b5e3b3a-fd74-4e40-9ff1-bee5de4f7615	Suzan	Frist	[binary da...	2022-06-26 13:30:52.072613+...	2022-06-26 13:30:52.072613+...	Suzan_Best	SUZAN_BEST	suzan_1@email.com	SUZAN_1@EMAIL.COM	true	AQAAAAEAACcQAA
6	6f22a577-912b-4347-8ef3-a130b58f5e...	Valera	Andrusenko	[binary da...	2022-06-26 13:35:05.810136+...	2022-06-26 13:35:05.810136+...	valera_andrusenko	VALERA_ANDRUSENKO	valera@email.com	VALERA@EMAIL.COM	true	AQAAAAEAACcQAA
7	fd81648e-dc9f-486e-9755-f261d1fbc88e	Vlad	Serhienko	[binary da...	2022-06-26 13:36:30.772227+...	2022-06-26 13:36:30.772227+...	pepe_vlad	PEPE_VLAD	vlad@email.com	VLAD@EMAIL.COM	true	AQAAAAEAACcQAA
8	2c5d0423-8371-4dee-904e-f6e24df029...	Oleg	Karansky	[binary da...	2022-06-26 13:39:33.655288+...	2022-06-26 13:39:33.655288+...	karansky_oleg	KARANSKY_OLEG	oleg@mail.com	OLEG@MAIL.COM	true	AQAAAAEAACcQAA

Рис. 2.8. Заповнення таблиці AspNetUsers

Джерело: знімок екрану

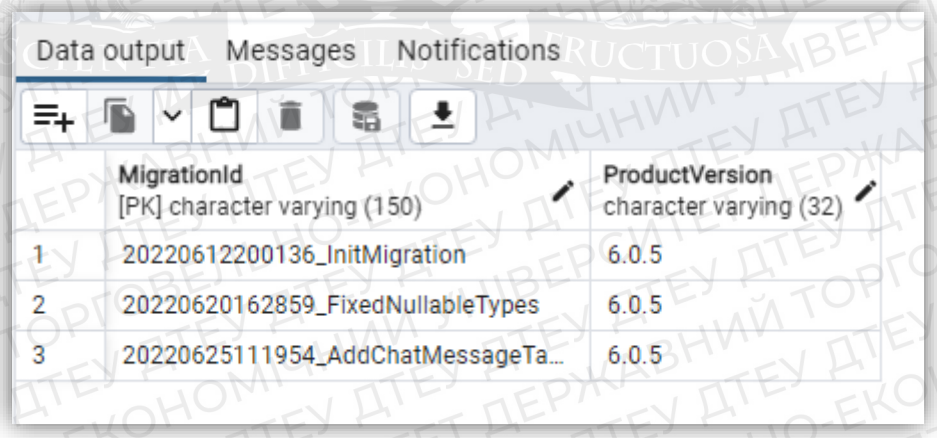
Таблиця AspNetUsers має наступні зв’язки:

1. Відношення "багато-до-багатьох" з таблицею AspNetRoles через зв’язуючу таблицю AspNetUserRoles. Кожен AspNetUsers може мати багато AspNetRoles, і навпаки.
2. Відношення "багато-до-багатьох" з таблицею Groups через зв’язуючу таблицю UserGroups. Кожен AspNetUsers може бути частиною багатьох Groups, і навпаки.
3. Відношення "один-до-багатьох" з таблицею AspNetUserLogins. Кожен AspNetUsers може мати багато AspNetUserLogins.

4. Відношення "один-до-багатьох" з таблицею ChatMessage. Кожен AspNetUsers може мати багато ChatMessage.
5. Відношення "один-до-багатьох" з таблицею AspNetUserTokens. Кожен AspNetUsers може мати багато AspNetUserTokens.
6. Відношення "багато-до-багатьох" з таблицею AspNetUserClaims через зв’язуючу таблицю AspNetUserRoles. Кожен AspNetUsers може мати багато AspNetUserClaims, і навпаки.

Також до основних таблиць можна віднести Group, ChatMessage, AspNetUserTokens. В таблиці Group зберігається інформація по чатах, а саме по користувачах групи. Таблиця ChatMessage зберігає інформацію по повідомленням, що надсилають користувачі. В таблиці AspNetUserTokens зберігаються короткотривалі токени для авторизації в системі.

Також присутня таблиця _EFMigrationsHistory, яка використовується суто для контролю варсій бази даних, та не має зв’язків з іншими таблицями. Ця таблиця є автозгенерованою ORM системою EntityFramework Core. Приклад заповнення таблиці наведений нище на рис. 2.9.


Data output Messages Notifications

	MigrationId [PK] character varying (150)	ProductVersion character varying (32)
1	20220612200136_InitMigration	6.0.5
2	20220620162859_FixedNullableTypes	6.0.5
3	20220625111954_AddChatMessageTa...	6.0.5

Рис 2.9. Заповнення таблиці _EFMigrationsHistory

Джерело: знімок екрану

2.3.3 Розробка та проектування архітектури додатку

Одним із ключових інструментів при проектуванні архітектури додатку є діаграма класів.

Діаграма класів є графічним уявленням структури та взаємодії класів у системі. Вона відображає класи, їх атрибути та методи, а також зв'язки між класами. Ця діаграма дозволяє візуалізувати архітектуру програми, ідентифікувати основні класи та їх взаємозв'язки, а також зрозуміти загальну структуру проекту.

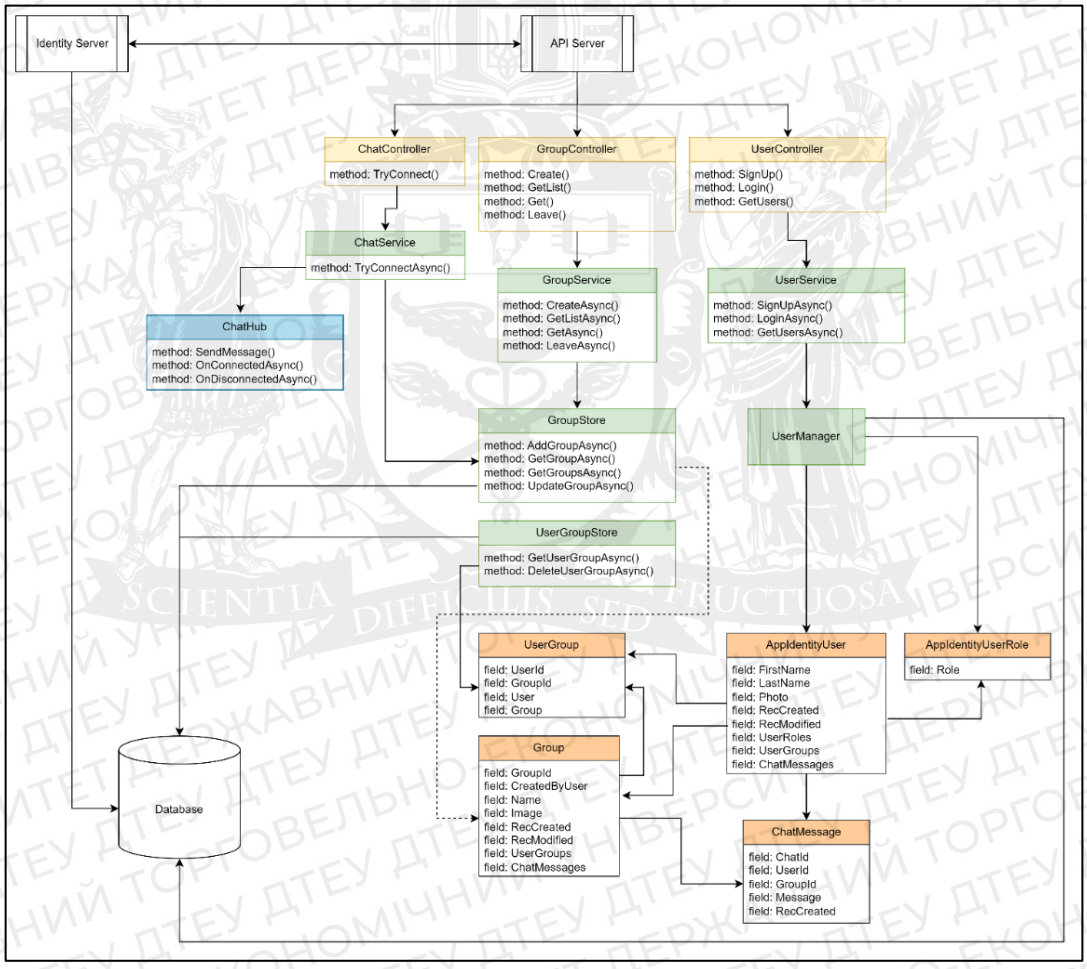


Рис 2.10: Діаграма класів серверної частини додатку

Джерело: побудовано автором за допомогою сервісу [3]

На рис. 2.10 зображена діаграма класів програмного додатку, подальше роз'яснення окремих компонентів наведене в таблиці 2.1.

Таблиця 2.1 Компоненти серверної частини додатку

Компонент	Клас	Призначення
Controller	ChatController	Обробка запитів, пов'язаних з обміном повідомленнями у чаті.
	GroupController	Обробка запитів, пов'язаних із керуванням та взаємодією з груповими сутностями, такими як створення та приєднання до групи, управління учасниками та інше.
	UserController	Обробка запитів, пов’язаних із даними користувачів. Його метою є обробка запитів, пов'язаних з обліковими записами користувачів.
Model	ChatService	Відповідає за логіку та функціональність пов'язану з обміном повідомленнями в онлайн чаті
	GroupService	Відповідає за логіку та функціональність пов'язану з керуванням груповими чатами.
	UserService	Відповідає за керування даними користувача та операції, пов'язані з обліковими записами користувачів.
	ChatHub	Компонент, який використовує технологію веб-сокетів та фреймворк SignalR для забезпечення взаємодії в режимі реального часу між бекендом та фронтендом в онлайн чаті.
	UserManager	Клас, що надається фреймворком ASP.NET Identity, який надає API для управління користувачами у базоі даних.
	GroupStore	Клас, що відповідає за управління групами у додатку та взаємодію з базою даних. Він надає API для управління групами.
	UserGroupStore	Клас, що відповідає за управління зв'язком між користувачами та групами у додатку та взаємодію з базою даних.
	UserGroup	Модель, яка встановлює зв'язок між користувачами та групами в системі.
	Group	Модель, що ідентифікує групи у системі.
	AppIdentityUser	Модель, що представляє сутність користувача у системі. Вона є розширенням стандартної моделі користувача Identity у ASP.NET.
	AppIdentityUserRole	Модель, що є сутністю ролей та привілеїв користувачів у додатку на основі ASP.NET Identity.
	ChatMessage	Модель, що представляє суть повідомлення в чаті.

2.4. Висновки до розділу 2

В ході моделювання та аналізу програмного забезпечення були вивчені та проаналізовані різні аспекти, пов'язані з розробкою клієнт-серверних додатків. Більшість уваги було приділено вибору мови програмування, бази даних та архітектури додатка, а також моделюванню внутрішньої структури системи.

При виборі мови для написання серверної частини додатка були розглянуті різні варіанти, в результаті чого було обрано C# та його платформу ASP.NET. Цей вибір був обґрунтований їх потужними можливостями, підтримкою об'єктно-орієнтованого програмування та широкою спільнотою розробників. В якості бази даних було обрано PostgreSQL через її надійне зберігання та управління даними.

Для написання клієнтської частини програми розглядалися різні мови програмування, але було обрано Angular. Цей фреймворк надає потужні інструменти для розробки користувацьких інтерфейсів, забезпечуючи гнучкість та ефективність розробки.

Важливою частиною моделювання був опис поведінки чату, для чого були створені діаграми станів і послідовностей. Ці діаграми дозволяють візуалізувати різні сценарії взаємодії користувача з чатом і допомагають розробникам краще зрозуміти його функціональність і логіку.

Архітектура додатку була визначена на основі патерну Model-View-Controller (MVC). Цей патерн допомагає забезпечити розділення даних, представлень та логіки додатку, а також покращити зручність обслуговування та масштабованість.

База даних була важливою частиною процесу розробки та проектування. Були розроблені концептуальні, логічні та фізичні моделі для опису структур даних, взаємозв'язків та методів зберігання. Для подальшого

аналізу та проектування була розроблена діаграма класів додатку на стороні сервера для представлення основних класів та їх взаємозв'язків.

Проведений аналіз програмного забезпечення та моделювання архітектури додатку, дозволили розробити ефективну модель роботи онлайн чату. Проведена робота є важливим етапом розробки програмного забезпечення, оскільки дає чітке уявлення про систему, допомагає визначити основні компоненти та взаємодію між ними, а також формує основу для подальшої роботи над проектом.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1. Створення інтерфейсу онлайн чата

Після визначення вимог до системи, настав етап розробки користувацького інтерфейсу сайту. Створення сайту – це важлива ступінь, що буде надавати користувачеві можливість взаємодіяти з функціоналом чату через зручний та інтуїтивно зрозумілий інтерфейс.

Розробка дизайну проводилась в середовищі Figma. Дизайн був створений з використанням спокійних кольорів та мінімалістиного дизайну. Вигляд сторінок та опис функціоналу наведені нижче.

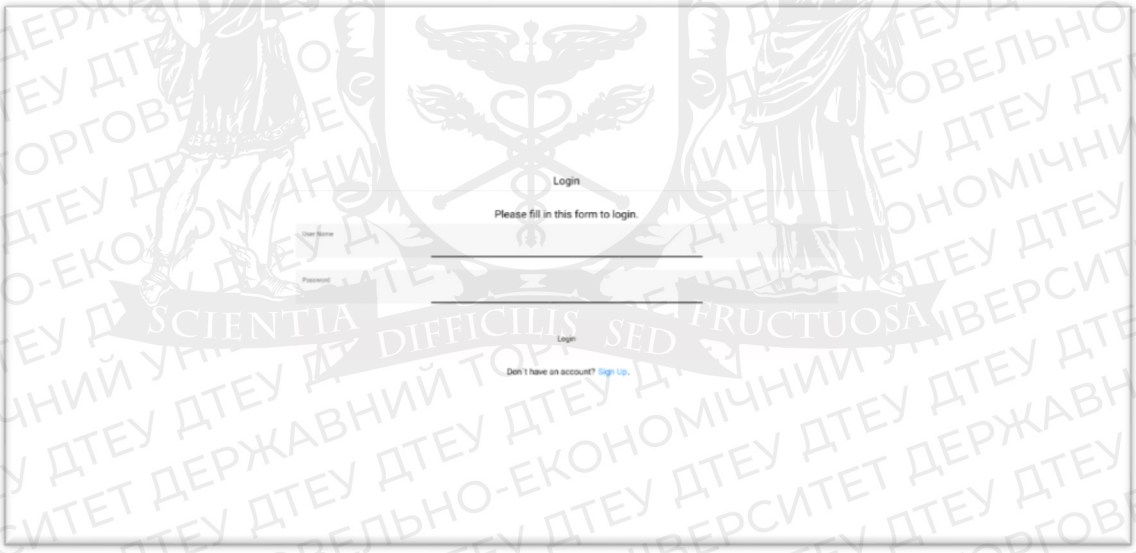


Рис. 3.1. Вигляд сторінки авторизації
Джерело: знімок екрану

Перше з чим зіткнеться користувач при відкритті сайту – це сторінка авторизації (рис. 3.1).

Важливість сторінки авторизації полягає у забезпеченні захисту даних користувачів та контролю доступу до функціоналу чату. Вона є першим етапом автентифікації користувачів і гарантує, що лише зареєстровані користувачі можуть отримати доступ до своїх облікових записів і використовувати функції чату.

На сторінці авторизації користувач може ввести свій логін і пароль для входу в систему. Після успішної аутентифікації він буде перенаправлений на головну сторінку чату, де зможе взаємодіяти з іншими користувачами.

Крім того, на сторінці авторизації є кнопка «Sign Up», що дозволяє користувачеві перейти на сторінку створення нового облікового запису.

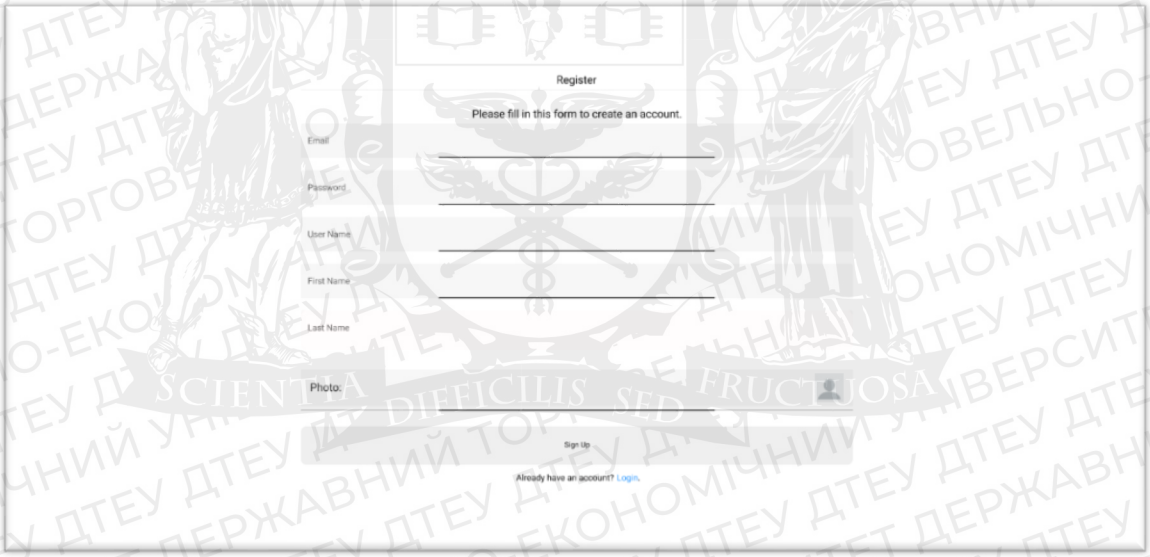


Рис 3.2. Вигляд сторінки реєстрації
Джерело: знімок екрану

Сторінка реєстрації (рис. 3.2) є важливим компонентом онлайн чату, оскільки вона надає новим користувачам можливість створити обліковий запис та отримати доступ до функцій чату. На цій сторінці користувач може вказати дані, необхідні для реєстрації: пошта, пароль, логін, ім’я, прізвище та фото.

Після заповнення всіх необхідних полів та натискання кнопки «Sign Up», дані користувача пройдуть процес валідації та збережуться в базі даних. Потім користувач автоматично перенаправляється на сторінку авторизації, де може увійти в систему, використовуючи свій новостворений обліковий запис.

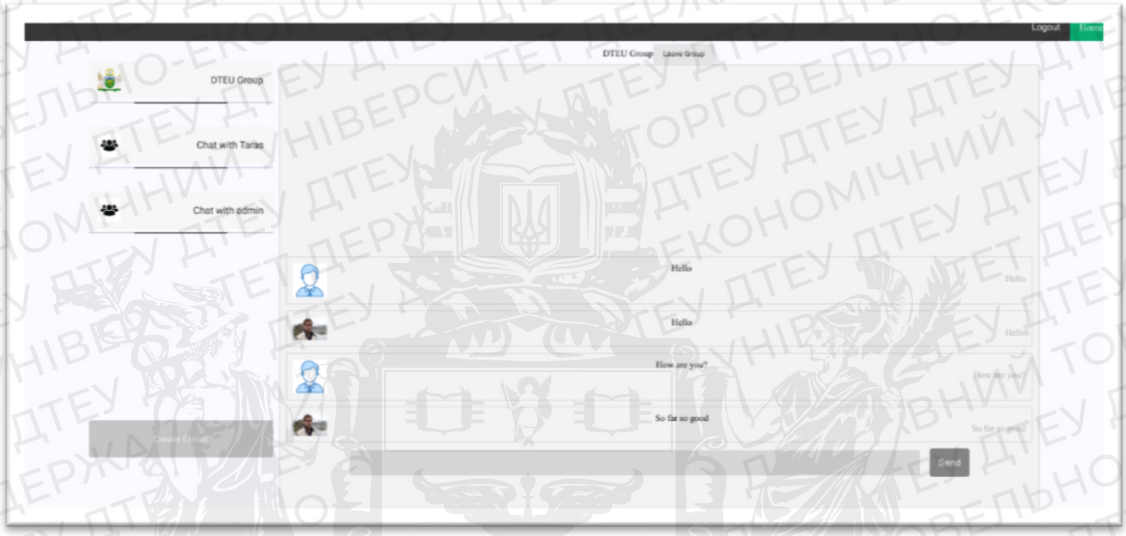


Рисунок 3.3. Вигляд сторінки з чатом
Джерело: знімок екрану

Сторінка з чатами(рис. 3.3) є важливою частиною онлайн чату, оскільки вона надає користувачеві зручний спосіб керування своїми чатами та взаємодії з іншими користувачами. Сторінка з чатами забезпечує зручну навігацію та доступ до основних функцій онлайн чату. Користувач може швидко переглядати та вибирати існуючі чати, створювати нові та керувати своїм обліковим записом. Цей функціонал забезпечує зручність використання сайту та забезпечує більш ефективну взаємодію користувачів у чатах. Загалом ця сторінка має наступний функціонал: відобразити список існуючих чатів, відкрити існуючий чат, вийти з облікового запису та створити новий чат.

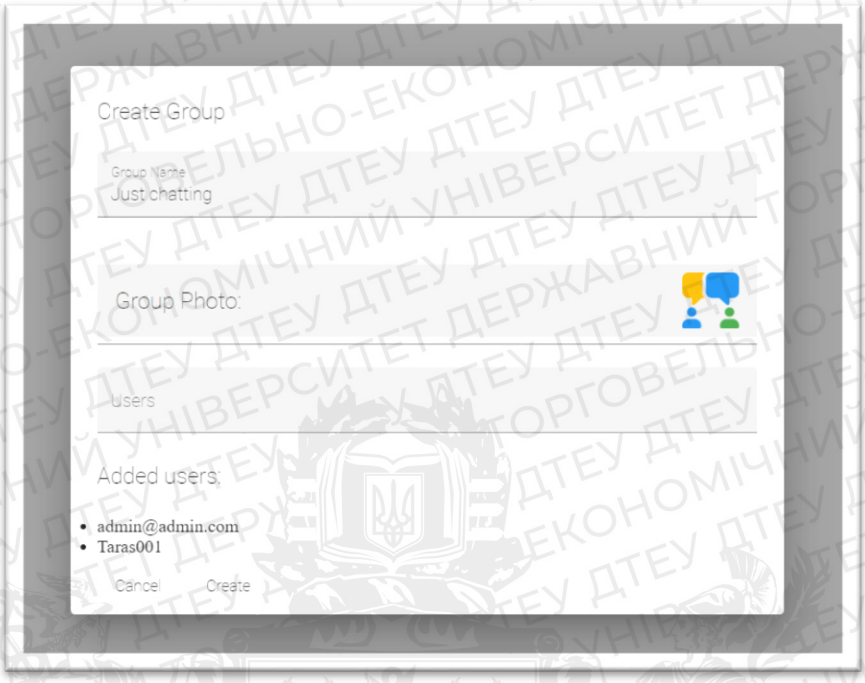


Рисунок 3.4. Вигляд модального вікна зі створенням нового чату
Джерело: знімок екрану

Для створення чату користувачу необхідно натиснути кнопку «Create Group» на основній сторінці. Після цього відкривається модальне вікно , як дозволяє визначити назву та зображення чату, а також запросити інших користувачів для участі у ньому. Це сприяє гнучкості та персоналізації чатів, а також дозволяє користувачам створювати різноманітні групи для спілкування та співпраці.

3.2. Розробка компонентів інтерфейсу

Розробка інтерфейсу була розподілена на окремі системні компоненти, використовуючи фреймворк Angular та мову TypeScript. Вимоги до системи передбачали розробку декількох сторінок, які забезпечують функціонал для аутентифікації, реєстрації, управління чатом, надсилання повідомлень всередині групи та виходу з групи.

Для створення сторінки авторизації та реєстрації був створений компонент на Angular, що містить необхідні елементи інтерфейсу, такі як поля для введення логіну та паролю, а також кнопку для відправки даних і переходу до реєстрації. Також була додана валідація для перевірки правильності введених даних. Натискання кнопки «Login» або «Sign Up» виконує відповідну дію, наприклад, надсилає запит на сервер для перевірки облікових даних або створює нового користувача. Програмний код сторінок наведений в Додатках В та Г.

Для сторінки з усіма чатами створено окремий компонент, який відображає список чатів. Цей компонент отримує дані про доступні чати з сервера і відображає їх у вигляді елементів списку. Користувач може вибрати конкретний чат або створити новий за допомогою відповідного елемента інтерфейсу.

Для реалізації можливості створення груп на сторінку чату було додано компонент, який дозволяє користувачам створювати нові групи чатів. Користувач вводить необхідні дані, що містить назву групи, і натискає кнопку "Create", яка надсилає запит на сервер. Якщо група успішно створена, інформація про неї оновлюється на сторінці чату.

Для створення функціоналу спілкування всередині групи створено компонент чату, який містить поле для введення тексту та кнопку відправки. Коли користувач вводить текст повідомлення і натискає на кнопку «Send» або клавішу «Enter», компонент відправляє запит з даними повідомлення на сервер, який додає його до відповідної групи чату. Якщо повідомлення успішно відправлено, воно відображається в інтерфейсі чату всіх учасників групи.

Для реалізації можливості покинути чат, на сторінку з поточною групою було додано відповідний елемент інтерфейсу. Користувач обирає групу, після цього відбувається оновлення інтерфейсу та з’являється кнопка «Leave

Group», що дозволяє покинути вибрану групу. Після підтвердження дії, на сервер надсилається запит на видалення користувача з групи, а в інтерфейсі відображається інформація про оновлення статусу групи.

3.3. Опис взаємодії бекенда і фронтенда

Робота між фронтендом і бекендом чату включає кілька важливих компонентів. Для забезпечення безпечного та авторизованого зв'язку використовується система авторизації IdentityServer. При вході в систему користувачі проходять процес аутентифікації та одержують авторизаційний токен.

Рисунок 3.5. Авторизаційний токен
Джерело: знімок екрану

Коли користувач виконує дії на фронтенді, такі як надсилання повідомлення або приєднання до чату, ці запити надсилаються на бекенд для обробки. Перед відправкою запиту на бекенд фронтенд перевіряє валідність авторизаційного токена (рис. 3.5), використовуючи IdentityServer. Таким чином, відбувається перевірка автентичності користувача та його авторизаційних даних.

Для забезпечення миттєвого зворотного зв'язку між користувачами та миттєвої передачі повідомлень використовуються хаби SignalR [Додаток А]. SignalR надає бібліотеку та протокол для двостороннього обміну повідомленнями між клієнтом (фронтендом) та сервером (бекендом) з використанням веб-сокетів. Це дозволяє реалізувати оновлення чату в реальному часі, де повідомлення та події передаються негайно та без необхідності перезавантаження сторінки. Реалізація хабу на сервері наведена в Додатку А, використання хабу на стороні сайту в Додатку Б.

Крім використання хабів SignalR для миттєвої передачі повідомлень, між фронтендом і бекендом також відбувається обмін HTTP запитами.

Наприклад, при створенні нового чату або надсиланні повідомлення фронтенд формує HTTP запит з відповідними даними і відправляє його на бекенд. Бекенд, в свою чергою, приймає цей запит, обробляє і виконує необхідні дії, наприклад, створює новий чат, зберігає повідомлення у базі даних тощо. Після завершення обробки запиту бекенд повертає відповідь фронтенду, що містить результат виконання операції або необхідні дані.

Таким чином, звичайні або не належать до передачі реального часу повідомлень.

Таким чином, фронтенд та бекенд чату взаємодіють за допомогою авторизаційних токенів, що перевіряються IdentityServer, хабів SignalR для забезпечення миттєвої передачі повідомлень між користувачами та HTTP запитами, що використовуються для взаємодії між фронтендом та бекендом у випадках, коли потрібне виконання операцій, які не потребують миттєвого оновлення. Це створює плавний та безпечний досвід використання онлайн чату для користувачів.

3.4. Висновки до розділу 3

В результаті розробки веб-сайту та його тісної інтеграції з бекендом, IdentityServer для авторизації та хабами SignalR для забезпечення спілкування в чаті в режимі реального часу, було створено повноцінний веб додаток. Веб-сайт надає користувачам зручний інтерфейс, де вони мають змогу взаємодіяти один з одним, обмінюватися повідомленнями та створювати нові чати.

Одним із ключових аспектів розробки була інтеграція з IdentityServer для забезпечення безпеки та автентифікації користувачів. IdentityServer служить як центральний сервер для перевірки та валідації авторизаційних токенів, що забезпечує безпеку доступу до функціоналу сайту та захист особистої інформації користувачів.

Для забезпечення спілкування в режимі реального часу було використано хаби SignalR. Ці хаби забезпечують більш ефективну та надійну передачу повідомлень між клієнтами та сервером, дозволяючи користувачам спілкуватись у режимі реального часу без необхідності оновлення сторінки. Це створює більш плавну і безперервну взаємодію в чаті, покращуючи користувальницький досвід.

У результаті, завдяки розробці користувацького інтерфейсу сайту, його інтеграції з серверною частиною, був створений потужний та безпечний онлайн-чат додаток, який забезпечує зручність використання, захист даних та можливість спілкування в режимі реального часу.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

З проведених досліджень можна зробити наступні висновки:

1. Проведено аналіз різних аспектів, пов'язаних з розробкою програмного забезпечення та моделюванням модуля соціальної мережі. У процесі розробки було розглянуто та проаналізовано різні мови програмування та бази даних для реалізації програмного додатку.
2. В якості операційної системи було обрано Windows 11 через її багаті можливості та зручне середовище розробки. Однак слід зазначити, що додаток є багатоплатформним і не залежить від конкретної операційної системи. Це означає, що його можна запускати і використовувати на різних платформах, таких як Windows, macOS і Linux. Мультиплатформенність досяглась завдяки використанні мови програмування C# та фреймворк ASP.NET, що мають можливості для створення крос-платформних додатків, які можуть працювати на різних операційних системах.
3. Проведений аналіз дозволив обрати мову програмування C# з платформою ASP.NET та базу даних PostgreSQL, оскільки вони мають необхідні можливості та надійність для створення серверної частини чату.
4. Також було проаналізовано мову програмування для клієнтської частини додатку, в результаті чого було обрано Angular, яка має потужні інструменти для створення інтерактивних інтерфейсів та підтримує компонентний підхід до розробки веб-додатків. Для моделювання та аналізу поведінки чату були використані діаграми стану та послідовностей, які візуалізують та описують взаємодію між

5. компонентами додатку. Це дозволило створити чітку та структуровану модель функціональності чату. Архітектурним шаблоном для розробки чату було обрано патерн Model-View-Controller (MVC), який забезпечує розділення логіки додатку, представлення даних та користувацького інтерфейсу. Це дозволяє створювати гнучкі та легко розширювані додатки, полегшує обслуговування та розробку нових функцій. Крім того, була розроблена і спроектована база даних чату, що включало концептуальну, логічну та фізичну моделі. Також було створено діаграму класів для серверної частини програми, яка описує структуру та взаємодію класів у додатку. Це полегшило розуміння та управління компонентами серверної частини додатку.
6. В процесі виконання проєкту розроблено потужний та масштабований бекенд, написаний на мові C# з використанням фреймворку ASP.NET. Бекенд забезпечує обробку запитів, бізнес-логіку та взаємодію з базою даних; фронтенд, створений за допомогою Angular, надає користувацький інтерфейс для взаємодії користувачів з функціоналом додатку. PostgreSQL база даних використовується для зберігання та управління даними додатку, забезпечуючи ефективне зберігання, доступ та маніпулювання даними. Всі ці компоненти працюють разом, щоб забезпечити повну функціональність додатку та його здатність реагувати на потреби користувачів.
7. Розроблена система повністю задовольняє поставленим вимогам і цілям. Було проведено аналіз існуючих мов та баз даних, та обрано найбільш підходящі для реалізації завдань. Було розроблено та спроектовано базу даних з використанням концептуальної, логічної та фізичної моделей. Були створені сторінки авторизації та реєстрації, сторінка з чатами, функціонал створення групи, надсилання повідомлень та можливість покинути групу. Були розроблені як бекенд з базою даних, так і

фронтенд компоненти, щоб забезпечити повноцінне функціонування докладання. Таким чином, система повною мірою відповідає поставленим вимогам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ЩО ТАКЕ СОЦІАЛЬНІ МЕРЕЖІ? [Електронний ресурс] – Режим доступу до ресурсу: https://futurenow.com.ua/shho-take-sotsialni-merezhi-vydy-klasyfikatsiya-bezpeka/.
2. What is online chat? [Електронний ресурс] – Режим доступу до ресурсу: https://www.liveagent.com/customer-support-glossary/online-chat/.
3. Draw IO [Електронний ресурс] – Режим доступу до ресурсу: https://draw.io/.
4. What is Java? [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/java/.
5. What is Python? [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/python-programming-language/.
6. ASP.NET overview [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/overview.
7. Вступ до PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.education-wiki.com/5154595-what-is-postgresql.
8. MVC Framework - Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm.
9. Моделювання даних: огляд [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/articles/556790/.

ДОДАТКИ

ДОДАТОК А

Код класу ChatHub

```
using Microsoft.AspNetCore.SignalR;

namespace Chat.Api.Hubs
{
   public class ChatHub : Hub
   {
      private const string SendMessageName = "SendMessage";

      private Guid GroupId
      {
         get
         {
            var segments = this.Context.GetHttpContext().Request.Path.ToString().Split("/").ToList();

            if (segments.Count != 3 || !Guid.TryParse(segments[2], out var groupId))
            {
               throw new ArgumentException("Incorect groupId");
            }

            return groupId;
         }
      }

      public async Task SendMessage(object user, string message)
      {
         await Clients.Group(GroupId.ToString()).SendAsync(SendMessageName, user, message);
      }

      /// <summary>
      /// Called when a new connection is established with the hub.
      /// </summary>
      /// <returns>A System.Threading.Tasks.Task that represents the asynchronous connect.</returns>
      public override async Task OnConnectedAsync()
      {
         await this.Groups.AddToGroupAsync(this.Context.ConnectionId, this.GroupId.ToString());
         await base.OnConnectedAsync();
      }

      /// <summary>
      /// Called when [disconnected asynchronous].
      /// </summary>
      /// <param name="ex">The ex.</param>
      /// <returns>A System.Threading.Tasks.Task that represents the asynchronous disconnect.</returns>
      public override async Task OnDisconnectedAsync(Exception ex)
      {
         await this.Groups.RemoveFromGroupAsync(this.Context.ConnectionId, this.GroupId.ToString());
         await base.OnDisconnectedAsync(ex);
      }
   }
}
```

ДОДАТОК Б

Підключення та використання хабу SignalR зі сторони сайту

```
public async CreateConnection()
  {
    var url = environment.hubConnectionURL.replace("{{groupId}}", this.groupId as
string);
    this.hubConnection = new signalR.HubConnectionBuilder()
      .withUrl(environment.hubConnectionURL.replace("{{groupId}}", this.groupId
as string))
      .configureLogging(signalR.LogLevel.Information)
      .build();

    this.hubConnection.on("SendMessage", (userName: User, message:string) =>
this.mapReceivedMessage(userName, message));

    this.start();
  }

  public async SendMessage(message:string | undefined)
  {
     await this.hubConnection?.send("SendMessage", this.User, message as string);
  }

  // Start the connection
  public async start()
  {
    try
    {
      await this.hubConnection?.start();
      console.log("connected");
    } catch (err)
    {
      console.log(err);
      setTimeout(() => this.start(), 5000);
    }
  }

  public async disconnect()
  {
    this.messages.splice(0);
    await this.hubConnection?.stop();
  }
```

Програмний код сторінки авторизації

```
<div class="login">
  <h2 mat-dialog-title>Login</h2>
  <hr>
  <p mat-dialog-title>Please fill in this form to login.</p>
  <mat-dialog-content>
    <mat-form-field appearance="fill">
      <mat-label>User Name</mat-label>
      <input matInput [(ngModel)]="LoginRequest.UserName" placeholder="Enter
email" type="email">
    </mat-form-field>
    <mat-form-field appearance="fill">
      <mat-label>Password</mat-label>
      <input matInput [(ngModel)]="LoginRequest.Password" placeholder="Enter
password" type="password">
    </mat-form-field>
  </mat-dialog-content>
  <mat-dialog-actions>
    <button mat-button class="loginButton" (click)="login($event)">Login</button>
    <nav>
      <p mat-dialog-title style="font-size:medium;">Don`t have an account? <a
class="signup" routerLink="/signup"a>Sign Up</a>.</p>
    </nav>
  </mat-dialog-actions>
</div>


/* Set a grey background color and center the text of the "sign in" section */
.login {
  text-align: center;
  position: absolute;
    top: 50%;
    left: 50%;
    margin-right: -50%;
    transform: translate(-50%, -50%);
  width: 50%;
}

/* Overwrite default styles of hr */
hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Add a blue text color to links */
a {
```

```
  color: dodgerblue;
}

.mat-form-field {
  width: 100%;
}

.myfilebrowser mat-toolbar
{
  float: inline-start;
}

.myfilebrowser mat-toolbar mat-label
{
  float: inline-start;
}

#fileUpload {
  position: absolute;
  z-index: 9;
  opacity: 0;
  height: 100%;
  width: 100%;
  left: 0px;
  top: 0px;
  cursor: pointer;
}

/* Set a style for the submit/register button */
.loginButton {
  padding: 16px 20px;
  margin: 8px 0;
  border: none;
  width: 100%;
  opacity: 0.9;
  border-radius: 10px;
}
```

```
import { LoginRequest } from '../../Models/Users/LoginRequest';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AutorizeService } from 'src/app/services/Autorize.service';

@Component({
  selector: 'app-authorization',
  templateUrl: './authorization.component.html',
  styleUrls: ['./authorization.component.css']
})
export class AuthorizationComponent implements OnInit {
```

```
  public LoginRequest: LoginRequest = new LoginRequest();

  constructor(private _authService: AutorizeService,
              private router: Router)
  {
  }

  public async login(evt:any): Promise<any>
  {
    try
    {
      var result = await this._authService.Login(this.LoginRequest);

      if (result === false)
      {
        throw new Error("cant login");
      }

      this.router.navigateByUrl('/chat');

    } catch (error)
    {
      console.log(error);
      this.router.navigateByUrl('/error');
    }
  }

  ngOnInit(): void {
  }

}
```

Програмний код сторінки реєстрації

```
<div class="signin">
  <h2 mat-dialog-title>Register</h2>
  <hr>
  <p mat-dialog-title>Please fill in this form to create an account.</p>
  <mat-dialog-content>
    <mat-form-field appearance="fill">
      <mat-label>Email</mat-label>
      <input matInput [(ngModel)]="User.Email" placeholder="Enter email"
type="email">
    </mat-form-field>
    <mat-form-field appearance="fill">
      <mat-label>Password</mat-label>
      <input matInput [(ngModel)]="User.Password" placeholder="Enter password"
type="password">
    </mat-form-field>
    <mat-form-field appearance="fill">
      <mat-label>User Name</mat-label>
      <input matInput [(ngModel)]="User.UserName" placeholder="Enter user name"
type="text">
    </mat-form-field>
    <mat-form-field appearance="fill">
      <mat-label>First Name</mat-label>
      <input matInput [(ngModel)]="User.FirstName" placeholder="Enter First Name"
type="text">
    </mat-form-field>
    <mat-form-field appearance="fill">
      <mat-label>Last Name</mat-label>
      <input matInput [(ngModel)]="User.LastName" placeholder="Enter Last Name"
type="text">
    </mat-form-field>
    <mat-form-field>
      <div class="myfilebrowser">
        <mat-toolbar>
          <mat-label>Photo: </mat-label>
          <input matInput readonly/>
          <img mat-image src="{{User.Photo}}"
style="height: 50px;">
        </mat-toolbar>
        <!-- Fetch selected filed on change -->
        <input type="file" #UploadFileInput id="fileUpload"
(change)="onUploadChange($event)" name="fileUpload" accept="image/*" />
      </div>
    </mat-form-field>
  </mat-dialog-content>
  <mat-dialog-actions>
    <button mat-button class="registerButton" (click)="SignUp($event)">Sign
Up</button>
```

```
    <nav>
      <p mat-dialog-title style="font-size:medium;">Already have an account? <a
class="login" routerLink="/login"a>Login</a>.</p>
    </nav>
  </mat-dialog-actions>
</div>

/* Set a grey background color and center the text of the "sign in" section */
.signin {
  text-align: center;
  position: absolute;
    top: 50%;
    left: 50%;
    margin-right: -50%;
    transform: translate(-50%, -50%);
  width: 50%;
}

/* Overwrite default styles of hr */
hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Add a blue text color to links */
a {
  color: dodgerblue;
}

.mat-form-field {
  width: 100%;
}

.myfilebrowser mat-toolbar
{
  float: inline-start;
}

.myfilebrowser mat-toolbar mat-label
{
  float: inline-start;
}

#fileUpload {
  position: absolute;
  z-index: 9;
  opacity: 0;
  height: 100%;
```

```
  width: 100%;
  left: 0px;
  top: 0px;
  cursor: pointer;
}

/* Set a style for the submit/register button */
.registerButton {
  padding: 16px 20px;
  margin: 8px 0;
  border: none;
  width: 100%;
  opacity: 0.9;
  border-radius: 10px;
}
```

```
import { AutorizeService } from 'src/app/services/Autorize.service';
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { CreateUser } from 'src/app/Models/Users/CreateUser';
import { User } from 'src/app/Models/Users/User';

@Component({
  selector: 'app-sign-up',
  templateUrl: './sign-up.component.html',
  styleUrls: ['./sign-up.component.css']
})
export class SignUpComponent implements OnInit
{
  public User:CreateUser = new CreateUser();

  constructor(private _authService:AutorizeService,
              private router: Router)
  {
  };

  public async SignUp(evt:any): Promise<any>
  {
    try
    {
      var result = await this._authService.SignUp(this.User);

      if (result === null)
      {
        throw new Error("cant create user");
```

```
      }

        this.router.navigateByUrl('/login');

        let jsonResult = JSON.stringify(result);
        console.log(`User created: ${jsonResult}`);

      } catch (error)
      {
        console.log(error);
        this.router.navigateByUrl('/error');
      }
    }

    onUploadChange(evt: any) {
      const file = evt.target.files[0];

      if (file) {
        const reader = new FileReader();

        reader.onload = this.handleReaderLoaded.bind(this);
        reader.readAsBinaryString(file);
      }
    }

    handleReaderLoaded(e: any)
    {
      this.User.Photo = btoa(e.target.result);
    }

    async ngOnInit(): Promise<any>
    {
      var response = await fetch('/assets/default_user_image.jpg');
      const reader = new FileReader();
      reader.onload = this.handleReaderLoaded.bind(this);
      reader.readAsBinaryString(await response.blob());
    }
  }
```