ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ОНЛАЙН-ЧАТУ ДЛЯ СПІЛКУВАННЯ НА ПЛАТФОРМІ ASP.NET МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД– 44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Побережник А.А
(прізвище та ініціали)
Керівник Жебка В.В
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забеспечення

Ступінь вищої освіти «Бакалавр»

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

Негоденко В.В.

“ ___ ” _____ 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

ПОБЕРЕЖНИК АНДРІЙ АНАТОЛІЙОВИЧ

(прізвище, ім‘я, по батькові)

1.Тема роботи: «РОЗРОБКА ОНЛАЙН-ЧАТУ ДЛЯ СПІЛКУВАННЯ НА ПЛАТФОРМІ ASP.NET МОВОЮ C#»

Керівник роботи: Жебка В.В., д.т.н., доц., зав. кафедри ТЦР

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26

2. Строк подання студентом роботи «1» червня 2023 року

3. Вихідні дані до роботи:

3.1 Науково-технічна література з питань, пов‘язаних із застосування веб-додатків;

3.2 Практичній досвід розробки веб-додатків.

3.3 Концепція побудови веб-додатків;

4. Перелік демонстраційних матеріалів

4.1 Тема дипломної роботи

4.2 Мета роботи. Об’єкт дослідження. Предмет дослідження.

4.3 Результат дослідження розробки веб-додатків.

4.4 Результат дослідження фреймворків для веб-розробки.

4.5 Результати дослідження та опис реалізації програми.

4.6 Апробація результатів дослідження

4.7 Висновки

5. Дата видачі завдання 25.02.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02.2023 - 25.02.2023	виконано
2	Аналіз та дослідження існуючих аналогів	26.02.2023 - 10.03.23	виконано
3	Дослідження програмних об’єктів	11.03.2023 - 20.03.23	виконано
4	Моделювання об’єкту проектування	25.02.2023 - 01.03.23	виконано
5	Розробка веб-додатку	02.03.2023 - 20.03.23	виконано
6	Вступ, висновки, реферат	21.03.2023 - 24.04.23	виконано
7	Розробка обов’язкових демонстраційних матеріалів	25.04.2023 - 10.05.23	виконано
8	Попередній захист роботи	26.05.2023	виконано
9	Здача роботи	01.06.2023	виконано

Студент_____ Побережник А.А

(підпис) (прізвище та ініціали)

Керівник роботи_____ Жебка В.В

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Випускна кваліфікаційна робота на тему «Розробка онлайн-чату спілкування мовою C# на платформі ASP.NET» містить 45 сторінок текстового документа, 25 ілюстрацій, 3 таблиці, 7 використаних джерел.

ОНЛАЙН-ЧАТ, КЛІЄНТ-СЕРВЕР, МЕСЕНДЖЕР, БАЗА ДАНИХ, СУБД PgAdmin

Об'єктом дослідження є веб-технології.

Метою даної роботи є підвищення ефективності робочого процесу.

Основні завдання:

- провести аналіз існуючих аналогічних програм;
- розробити алгоритм, базу даних та інтерфейс;
- реалізувати проект із використанням програмних засобів.

В результаті виконання випускної кваліфікаційної роботи було розроблено додаток «Корпоративний месенджер», що дозволяє ефективно взаємодіяти між собою учасникам робочої команди, що дозволяє оптимізувати робочий процес.

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 відео зв’язку, месенджери, соціальні мережі, наприклад: Skype, WhatsApp, Viber.

Однак вищезазначені додатки мають низку 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 компанії. 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.

За даними агентства «Frost&Sullivan», кількість учасників корпоративних соціальних платформ у 2013 році по всьому світу склала 208 млн осіб і, за прогнозами, сягне 1 млрд до 2024 року.

Метою випускної кваліфікаційної роботи є створення корпоративного месенджера.

Для досягнення поставленої мети необхідно вирішити такі завдання:

− пошук та дослідження аналогічних сервісів;

− проектування програми;

− розробка програми.

Об'єкт дослідження - корпоративне спілкування, а предмет дослідження - онлайн-чат для корпоративного спілкування.

Корпоративний онлайн чат на локальній мережі - це інструмент, що змінює спосіб комунікації внутрішньої команди чи організації. Він надає можливість співробітникам знаходитися в постійному зв'язку, незалежно від фізичного розташування та географічних обмежень. Корпоративний онлайн чат надає зручний та надійний інструмент для обміну інформацією, спілкування та співпраці.

Корпоративний онлайн чат на локальній мережі є інструментом, що спрощує та поліпшує процеси комунікації внутрішньої команди або організації. Онлайн чат дозволяє ефективно обмінюватися інформацією, спілкуватися та координувати робочі процеси. Він також забезпечує зручну платформу для групових дискусій, віртуальних зустрічей та спільної роботи над завданнями.

1. ЗАГАЛЬНІ ВІДОМОСТІ

1.1. Теоретичні основи

Останнім часом інтернет значно спростив комунікації між людьми, що так само посприяло все більшому поширенню використання інтернету як одного з інструментів для ведення бізнесу.

Багато компаній використовують на своїх сайтах системи онлайн-

консультування для комунікації з клієнтом. Чат онлайн консультування - це просте та ефективне рішення, що дає можливість миттєво дати відвідувачу допомогу в текстовому вигляді, що зручно - можна чітко і правильно сформулювати питання і отримати на нього не менш чітку відповідь, причому оперативно.

Мета роботи - створення чату онлайн для спілкування мовою C#.

У рамках розробки проекту необхідно вирішити такі завдання:

- Розглянути аналоги програми
- Визначити інструментальні засоби, які будуть використовуватись у розробці;
- Вказати унікальність системи, що розробляється.
- Описати процес системи, що розробляється
- Описати роботу отриманої в результаті системи

Актуальність створення чату онлайн консультування полягатиме в тому, що клієнт отримає гнучке налаштування інтерфейсу користувача при використанні даного чату. Це потрібне, так як багато компаній часом пишуть власні чати для потреб лише через те, що їх не влаштовує графічний інтерфейс, представлений вже існуючими на ринку продуктами. Також буде реалізована можливість гнучкого та докладного налаштування інформації про відвідувача.

Кількість віртуальних чатів, що існують на сьогоднішній день, є колосальною, наводити приклади всіх неможливо, та й не має сенсу. Для розуміння

того, яким має бути розроблюваний нами месенджер, ми вивчили існуючий ринок месенджерів, як платних, так і безкоштовних, проаналізували та вибрали, на наш погляд, найпопулярніші з них.

Під месенджером у сучасному інтернет-просторі розуміють систему миттєвого обміну повідомленнями (англ. Instant messaging, IM), тобто. програми онлайн-консультанти та програми-клієнти для обміну повідомленнями в реальному часі через Інтернет»[3]. Головною особливістю месенджера є спілкування в локальній мережі, що робить додаток безпечним та надійним для корпоративного спілкування.

Опис кожного месенджера передбачає виявлення його технічних характеристик та послуг, які він надає. Розглянемо WhatsApp[4], Viber[5], Telegram[6], Facebook Messenger[7] та Skype[8].

По-перше, безперечною перевагою всіх представлених нижче месенджерів є той факт, що вони можуть бути використані на будь-якому пристрої та з будь-якою платформою: Android, Iphone, комп'ютери Mac та Windows, Windows Phone.

По-друге, вони надають можливість безкоштовно обмінюватися повідомленнями, зокрема. голосовими, створювати груповий чат, здійснювати аудіо та відео дзвінки, пересилати файли (тексти, відео, аудіо, фото), використовувати емодзі (графічна мова ідеограм та смайликів). Набір функцій всіх вивчених месенджерів приблизно однаковий. У Skype додатково можна записувати дзвінки, а також використовувати «автоматичні субтитри, щоб читати слова».

По-третє, розробники обіцяють, що останніх версіях додатків реалізовані різні способи шифрування даних. Так, «наскрізне шифрування в WhatsApp забезпечує захист повідомлень і дзвінків. Тільки учасники спілкування можуть прочитати або прослухати вміст, а ніхто інший навіть WhatsApp». На сайті Телеграм зазначено, що «повідомлення в Telegram зашифровані та мають таймер самознищення».

1.2 Функціональні та нефункціональні вимоги

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

Система має відновлювати своє функціонування при коректному перезапуску апаратних засобів. У разі аварії необхідно забезпечити збереження всіх даних системи. Для цього має бути передбачена можливість організації автоматичного та (або) ручного резервного копіювання даних Системи засобами системного та базового програмного забезпечення (ОС, СУБД), що входить до складу програмнотехнічного комплексу.

Повинна бути передбачена можливість відновлення даних за день збою за допомогою їх повторного введення або імпорту (для даних із зовнішніх систем автоматично).

Для забезпечення захисту від несанкціонованого доступу необхідно передбачити авторизацію користувача під час входу до системи.

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

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

1.3 Аналіз існуючих на ринку додатків онлайн-чату

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

1. Staply. Месенджер із проектним функціоналом. Дозволяє: скласти ієрархію груп; створювати завдання, ставити терміни, важливість та відповідальність у робочих групах; здійснювати контроль за групами.

Рисунок 1.3.1 – Логотип компанії «Staply»

2. Slack. Корпоративний месенджер із підтримкою інтеграції з десятками сторонніх сервісів. Об'єднує в одному вікні обговорення у загальних темах (каналах), приватних групах та особистих повідомленнях. Дозволяє здійснювати відеодзвінки; зберігати повідомлення без доступу до мережі.

Рисунок 1.3.2 – Логотип компанії «Slack»

3. HipChat. Є груповим чатом, відеочатом для команд і компаній з можливостями обміну файлами і демонстрацією екрану.

Рисунок 1.3.3 – Логотип компанії «HipChat»

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 у кількох місцях; працювати у кількох кімнатах одночасно.

Рисунок 1.3.4 – Логотип компанії «Бітрікс24»

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

- інтеграція із сторонніми сервісами;
- створення ієрархії груп;
- розподіл керуючим учасників за ролями;
- можливість контролю керуючим над групами;
- резервне копіювання у кількох місцях;
- робота у кількох кімнатах одночасно;
- оформлення, що налаштовується;
- інструменти;
- оффлайн-повідомлення.

У таблиці 1 представлена порівняльна характеристика перерахованих вище продуктів.

Таблиця 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,MailChimpHeroku та інші.	MS SharePoint,MS Exchange Server,MS Outlook, Продукти Apple та Google.	Дозволяє користувачам відстежувати прогрес у різних проектах за допомогою однієї платформи
Створення ієрархії груп	+	-	-	-	Дозволяє рівномірно розподілити завдання
Розподіл керуючим учасників за ролями		-	-	-	Дозволяє ефективно розподілити завдання
Можливість контролю керуючим проектом		+	+	+	Забезпечує контроль виконання завдань
Резервне копіювання у кількох місцях	+/-	-	-	+	Забезпечує збереження даних

Продовження таблиці 1 – Порівняльна характеристика розглянутих аналогів

Робота в кількох кімнатах одночасно	+	-	-	+	Дозволяє вирішувати кілька завдань одночасно
Настроювання оформлення	-	-	-	-	Дозволяє індивідуалізувати інтерфейс

2. ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Опис технічного завдання

У рамках дипломного проекту ставилося завдання розробити систему оперативного обміну сервісними даними у вигляді текстових файлів, файлів зображень, файлів інших форматів і повідомлень у межах певної групи користувачів – корпоративного месенджера.

Корпоративний месенджер розроблений для робочих груп, які одночасно виконують кілька проектів, кожен з яких передбачає розподіл ролей між одними і тими ж виконавцями (користувачами). Організація роботи з месенджером передбачає наявність у робочій групі спеціалістів, які виконують конкретні завдання в конкретному проекті. У той же час в іншому проекті той же співробітник може виконувати завдання, відповідне іншій ролі. Продукт орієнтований на організацію віддаленої роботи закритих груп користувачів.

- Проектування архітектури серверної та клієнтської частин програми.
- Створення інформаційної моделі.
- Розробка додатку.

Метою розробки є створення системи, що включає в себе серверну частину, що обробляє запити користувачів системи, що надходять; клієнтську частину, до якої належать інтерфейси користувачів; та шар роботи з даними, що використовує віддалені веб-сервіси для доступу до всієї необхідної інформації.

Розробити корпоративний медіа портал, який відповідає сучасним стандартам продуктивності, безпеки та зручності. Усі особисті дані користувачів, якщо такі є, повинні бути недоступні нікому, крім користувача та адміністратора системи. Паролі та інша чутлива до злому інформація повинні зберігатися у зашифрованому вигляді. Реакція системи на дії користувача не регламентована строго, але не має бути надто великою.

2.2 Вибір інструментальних засобів розробки

Система розроблена на платформі ASP.NET MVC 4 з використанням мови програмування C#. ASP.NET MVC — це структура для створення веб-додатків .NET, яка реалізує шаблон MVC.

ASP.NET MVC — універсальна технологія, яка може використовуватися як для невеликих проектів, так і для великих високонавантажених систем (наприклад, Stackoverflow). На відміну від іншої технології розробки веб-додатків на .NET - ASP.NET WebForms - цей фреймворк не має на меті перенести підхід розробки

віконних додатків Windows до розробки веб-додатків. ASP.NET MVC використовує дуже органічний підхід до веб-розробки, заснований на принципах роботи в Інтернеті та шаблоні MVC [1].

Вихідний код для ASP.NET MVC є загальнодоступним, хоча немає способу змінити його таким чином, щоб зміни ввійшли до нової версії. Повністю розроблено Microsoft.

Для цієї програми Microsoft IIS 7 використовується як веб-сервер. А також .NET Core 4.

IIS і .NET Core входять до складу Windows Server 2008 і не потребують додаткових дій для встановлення та більшої частини конфігурації.

Додаток має власне сховище даних, як реляційна база даних .

Обмін даними з сервісами відбувається по протоколу HTTP. Майже всі дані запиту містяться в URL. Усі дані надходять із сервісів у форматі XML.

ASP.NET MVC, реалізуючи шаблон MVC [2], полегшує керування складними структурами шляхом поділу програми на модель, представлення та контролер. Надає розробникам повний контроль над поведінкою програми. Надає розширену підтримку розробки, керованої тестуванням. Добре підходить для веб-додатків, які обслуговуються великими групами розробників і веб-розробниками, яким потрібен високий рівень контролю над поведінкою додатків.

Платформа ASP.NET MVC надає такі функції:

− відокремлення додатків (логіка введення, бізнес-логіка та логіка інтерфейсу користувача), широке тестування та розробка на основі тестування. Усі основні контракти платформи MVC базуються на інтерфейсі та можуть бути перевірені за допомогою імітацій об’єктів, які імітують поведінку реальних об’єктів програми. Програму можна тестувати без запуску контролерів у процесі ASP.NET, що робить тестування швидшим і гнучкішим. Для тестування можна використовувати будь-який модуль тестування, сумісний із .NET Framework.

− платформа, що розширюється та розширюється. Компоненти платформи ASP.NET MVC можна легко замінити або налаштувати. Розробник може включити власний механізм перегляду, політику маршрутизації URL-адрес, серіалізацію параметрів методу дії та інші компоненти. Платформа ASP.NET MVC також підтримує використання моделей контейнера ін’єкції залежностей (DI) і інверсії керування (IOC). Модель ін’єкції залежностей дозволяє вам вставляти об’єкти в клас, а не чекати, поки клас сам створить об’єкт. Модель інверсії керування вказує, що якщо один об’єкт потребує іншого об’єкта, то перші об’єкти повинні отримати другий об’єкт із зовнішнього джерела (наприклад, із файлу конфігурації). Це полегшує тестування.

− розширена підтримка маршрутизації ASP.NET. Цей потужний компонент відображення URL-адрес дозволяє створювати програми зі зрозумілими URL-адресами, які можна використовувати під час пошуку. URL-адреси не повинні містити розширення імен файлів і розроблені для підтримки шаблонів іменування URL-адрес, які забезпечують оптимізовану для пошукових систем (SEO) адресацію та передачу стану представлення (REST).

− підтримка використання розмітки в існуючих файлах ASP.NET Page (.aspx), Control (ASCX) і Master Page (MASTER) як шаблони перегляду. Ви можете використовувати наявні функції ASP.NET із структурою ASP.NET MVC, наприклад вкладені головні сторінки, вбудовані вирази (<%= %>), декларативні серверні елементи керування, шаблони, зв’язування даних, локалізацію тощо.

− підтримка існуючих функцій ASP.NET. ASP.NET MVC підтримує такі функції, як автентифікація за формами та Windows, автентифікація за URLадресами, членство та ролі, вихідні дані та кешування даних, керування станом сесії та профілю, моніторинг справності, система конфігурації та архітектура провайдера.

ASP.NET MVC надає рішення багатьох проблем, з якими стикаються веброзробники та організації, які хочуть створити інформаційну систему на основі платформи .NET. ASP.NET MVC дозволяє використовувати різні бібліотеки, які доповнюють функціональність платформи або краще реалізують деякі функції. Більшість цих фреймворків і бібліотек можуть працювати незалежно одна від одної, однак вони надають більше функціональних можливостей при спільному використанні.

2.3 Опис середовища та системи

Оскільки C# одна із найпоширеніших мов програмування, йому існує велика кількість інтегрованих середовищ розробки з різним функціоналом. Оскільки висока продуктивність є однією з основних вимог до розробки програми/

Важливим критерієм вибору середовища розробки програми є наявність профільника. Як об'єкти дослідження були обрані інтегровані середовища розробки MS Visual Studio, JetBrains Clion, Qt Creator, NetBeans та Eclipse SDK , як найбільш функціональні та поширені.

MS Visual Studio

Microsoft Visual Studio - це інтегроване середовище, призначене для розробки програмного забезпечення, що має низку додаткових інструментів.

Продукти MS Visual Studio дозволяють розробляти консольні програми, програми з графічним інтерфейсом, веб-сайти, веб-програми та веб-сервіси для платформ.

Рисунок 2.1 –Microsoft Visual Studio

MS Visual Studio містить вбудований інструмент для редагування вихідного коду програми та дозволяє проводити рефакторинг коду. Відладчик, вбудований у

MS Visual Studio, має два рівні роботи: рівень налагодження вихідного коду та

рівень налагодження машинного коду. Додаткові вбудовані інструменти включають форми GUI та зручний редактор для створення програм з інтерфейсами, містять веб-редактор та редактор класів. Це середовище розробки підтримує можливість створення та підключення плагінів (надбудов), що використовуються для розширення функціональності. MS Visual Studio підтримує системи контролю версій вихідного коду (наприклад, Visual SourceSafe або Subversion), містить безліч інструментів, що дозволяють редагувати та проектувати код предметноорієнтованими мовами програмування.

Для роботи з серверною частиною було обрано локальний сервер Hamachi.

Hamachi - це інноваційна програма, яка надає можливість створювати віртуальну приватну мережу (VPN) з легкістю і надійністю. За допомогою Hamachi, ви можете забезпечити безпечне з'єднання між різними комп'ютерами через Інтернет, незалежно від їх географічного розташування.

Однією з ключових особливостей Hamachi є його простота в використанні та налаштуванні. Після встановлення програми, ви можете швидко створити віртуальну мережу, до якої можна підключати комп'ютери з різних місць. Все, що потрібно зробити, - це створити обліковий запис Hamachi та ввести дані для створення мережі.

Hamachi використовує потужну технологію "середників" (mediation), що дозволяє обійти брандмауери, маршрутизатори та інші обмеження мережі для забезпечення надійного з'єднання. Це робить його ідеальним рішенням для різних сценаріїв, включаючи віддалену роботу, спільне використання ресурсів та групові проекти.

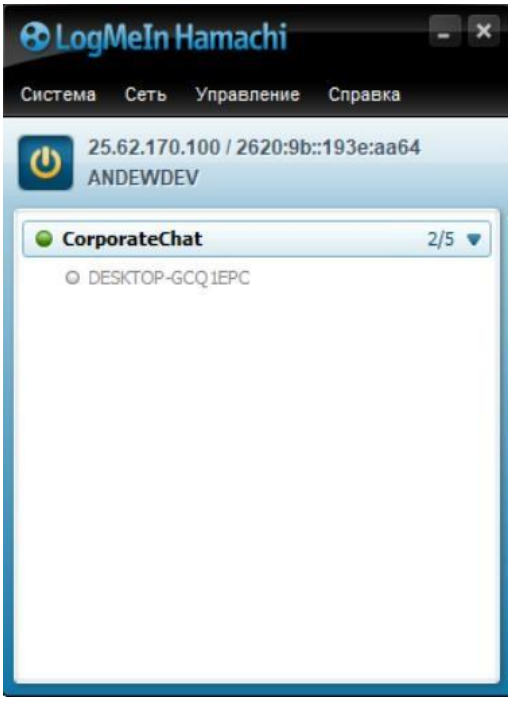


Рисунок 2.2 – Запуск локального серверу

За допомогою Hamachi реалізовано сервер з веб-додатком, яким можна користуватись з різних комп’ютерів.

2.4 Проектування архітектури системи

Діаграма прецедентів (Use Case Diagram) – діаграма, що відображає відносини між акторами та прецедентами і є складовою моделі прецедентів, що

дозволяє описати систему на концептуальному рівні. Ця діаграма призначена для визначення функціональних вимог до програмного продукту.

Базовими 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 малюнку

Користувач може переглядати та редагувати свій профіль, переглядати проекти, повідомлення, організацію (рисунок 2.3).

Рисунок 2.3 – Use case діаграма роботи з профілем користувача

Діаграма роботи адміністратора представлена рисунку 2.4. Адміністратор веде облік існуючих користувачів, може редагувати список посад, користувачів, і навіть може редагувати існуючі проекти, змінювати статус користувачів.

Рисунок 2.4 – Use case діаграма роботи адміністратора

Користувач може створювати, редагувати проекти, додавати учасників, документи, робочі групи проекту, розподіляти завдання робочих груп, створювати, редагувати розмови робочих груп.

Логічні моделі для розробки програмного забезпечення.

За допомогою діаграм прецедентів, діаграм варіантів використання основні користувачі системи та завдання, які має вирішувати система. За допомогою діаграм діяльності описується послідовність дій для кожного прецеденту, яка необхідна для досягнення поставленої мети.

Діаграма зв’язку — це особливий вид діаграми взаємодії, яка зосереджена на обміні даними між різними учасниками взаємодії.

На діаграмі зв’язку не потрібно показувати кожного учасника як рятувальний круг, а також не потрібно показувати послідовність повідомлень по вертикалі як діаграму послідовності. Натомість учасників можна розміщувати за потреби, дозволяючи комунікаціям показувати стосунки між учасниками та використовувати числа для представлення послідовності повідомлень.

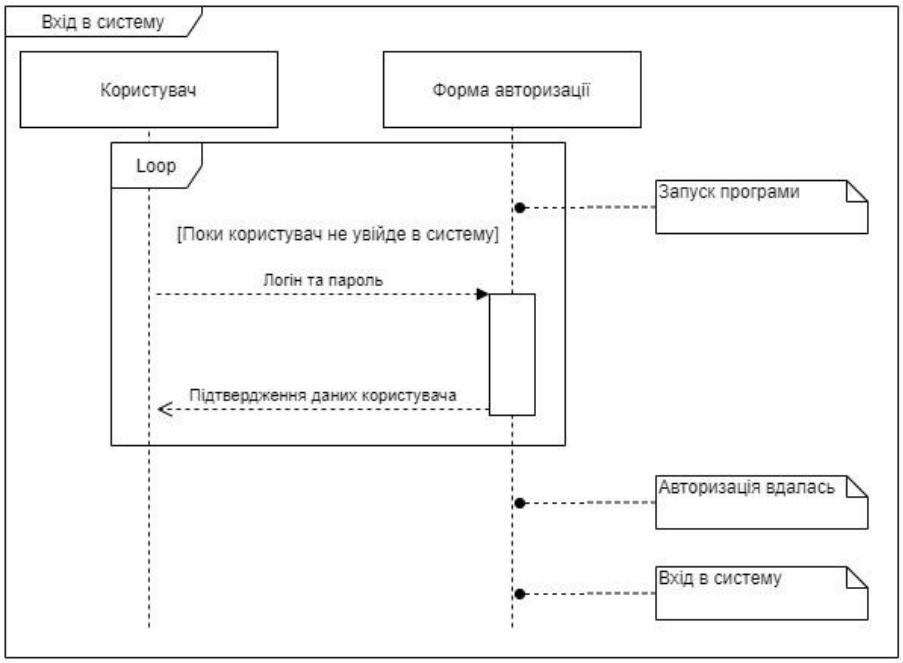


Рисунок 2.5 – Діаграма послідовності запису

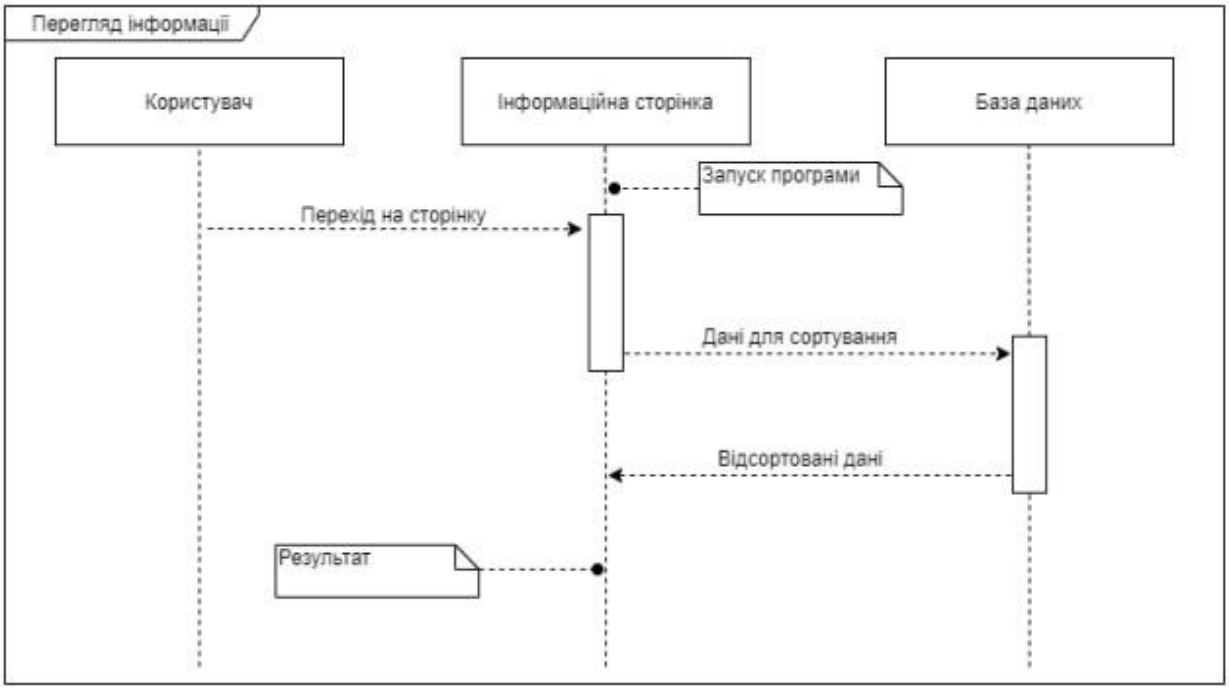


Рисунок 2.6 – Діаграма послідовності для перегляду інформації

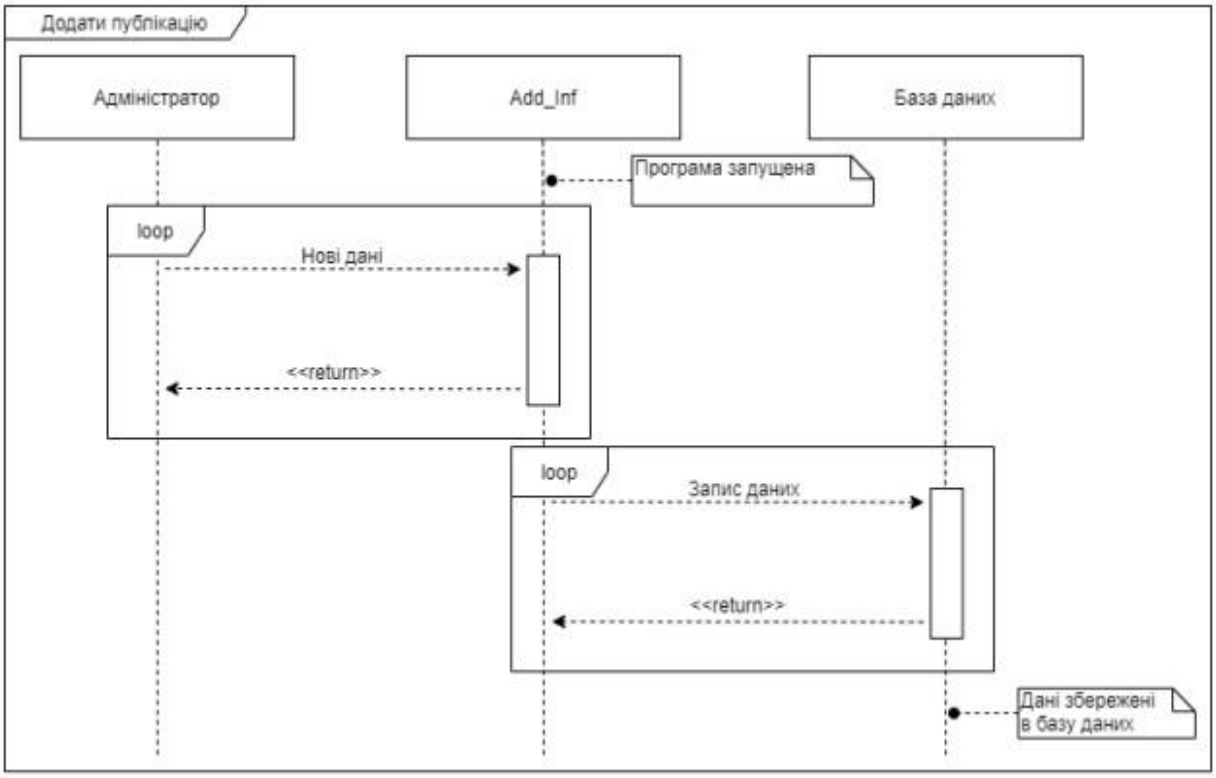


Рисунок 2.7 – Діаграма послідовності додавання інформації

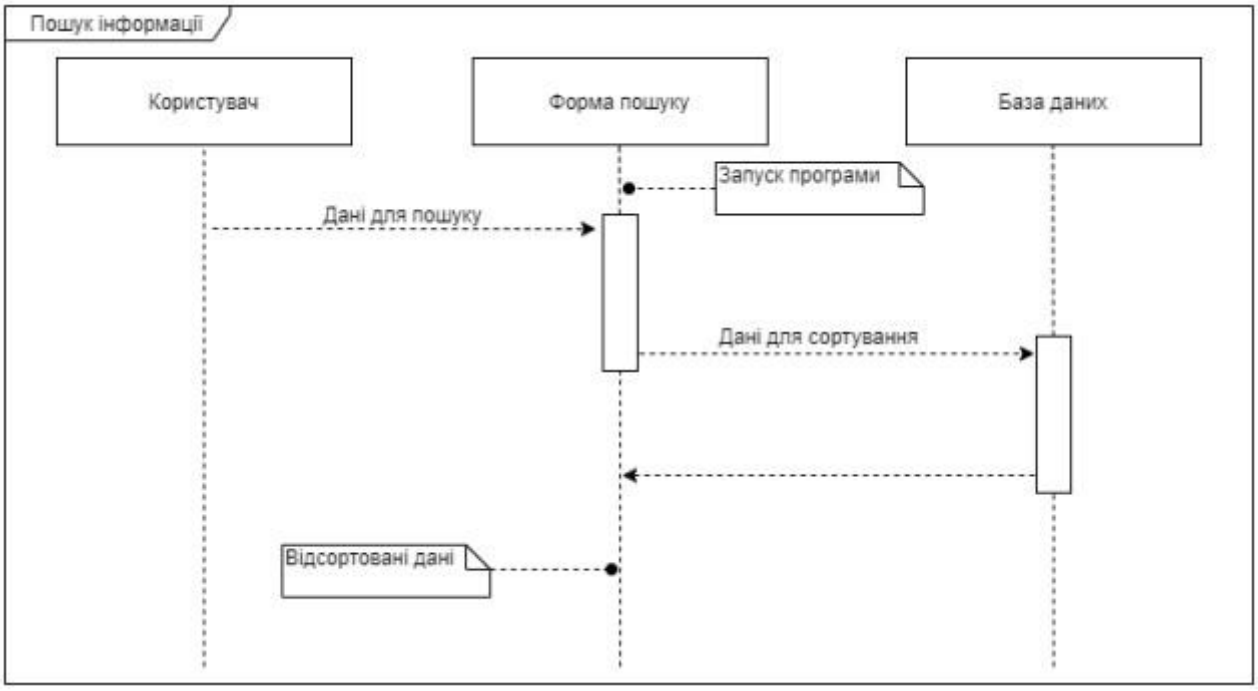


Рисунок 2.8 – Діаграма послідовності пошуку інформації

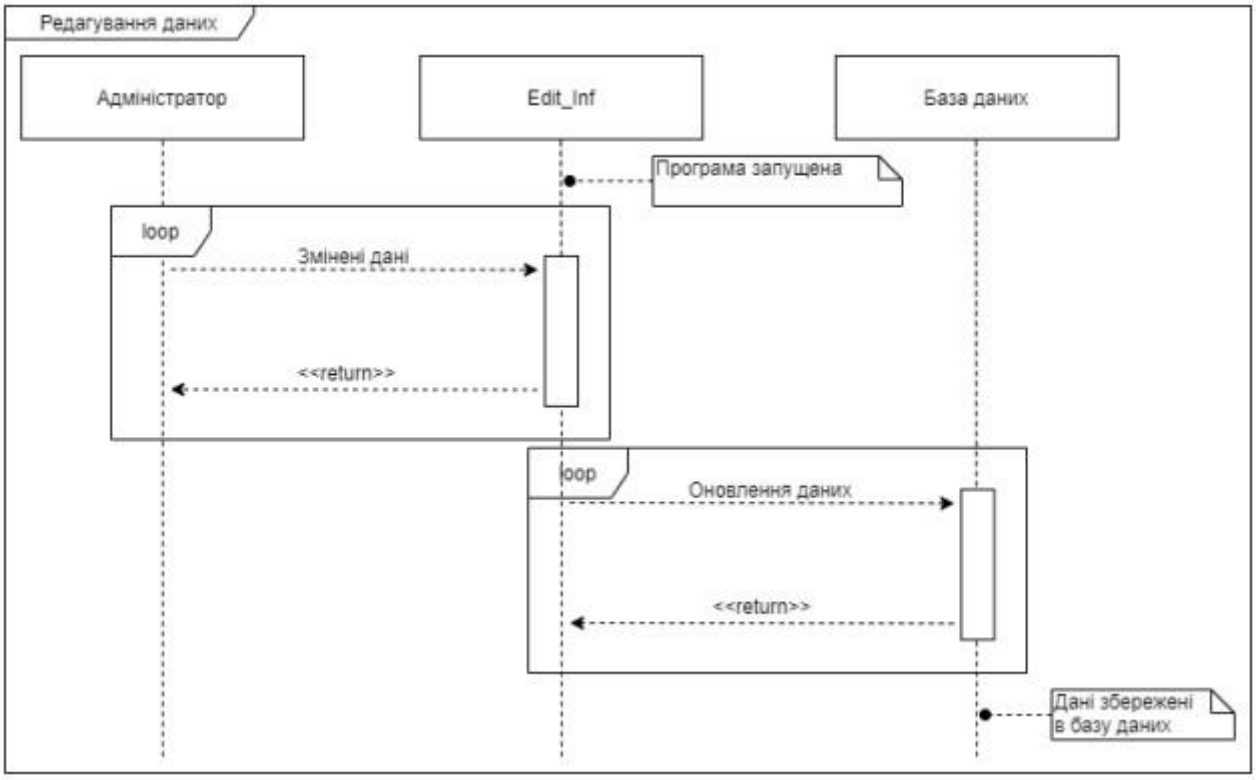


Рисунок 2.9 – Діаграма послідовності редагування інформації

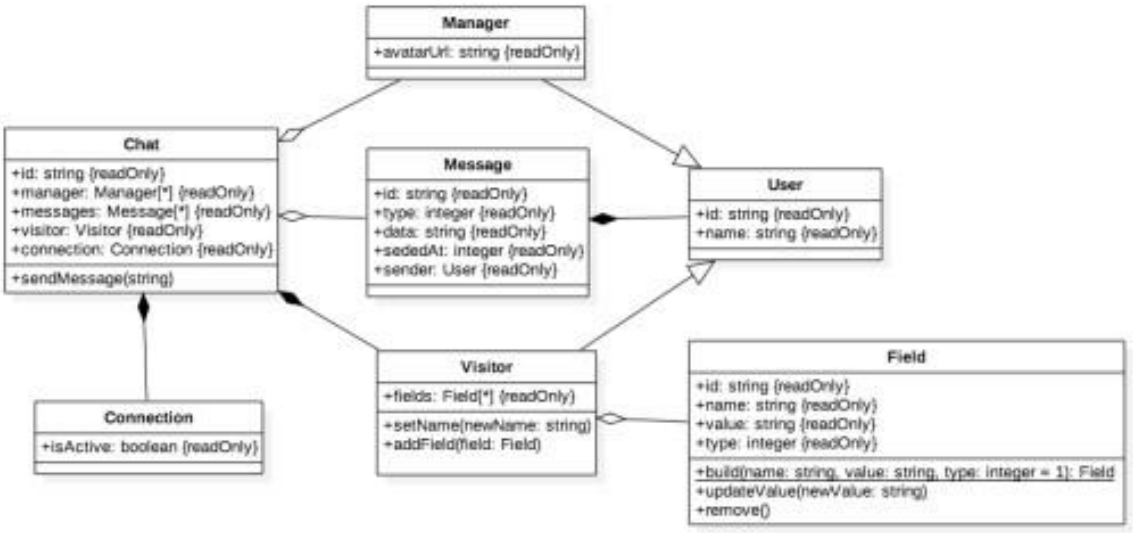


Рисунок 2.10 - Початкова діаграма класів

На малюнку представлені основні класи, що використовуються під час розробки.

Нижче описано, навіщо потрібен кожен клас:

- Chat-клас чату.
- User-Універсальний клас для учасників чату.
- Visitor- клас відвідувача чату, успадковується від User.
- Manager - клас співробітника, що відповідає у чаті, успадковується від User.
- Massager – клас повідомлення в чаті.
- Field – клас додаткових властивостей відвідувача чату.
- Connection – клас з'єднання з сервером за протоколом WebSoket.

2.5 Вимоги до системи

2.5.1. Функціональні вимоги

До програмного забезпечення висуваються наступні вимоги:

- Можливість спілкуватися в онлайн-чату;
- Реалізація алгоритму з’єднання з користувачами;

- Реалізація метод1у підключення до бази даних, пошук інформації, додавання користувачів;
- Візуалізація результатів роботи програми у вигляді онлайн-чату.

2.5.2. Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до апаратного забезпечення комп'ютера або мобільного пристрою для коректної роботи розробленого програмного продукту:

- Процесор - 2 x IntelXeon3 ГГц;
- Обсяг оперативної пам'яті – 16 ГБ;
- Дискова підсистема – 4 x 146 ГБ;
- Дисковод компакт-дисків (DVD-ROM);
- Мережевий адаптер - 100 Мбіт/с. для ПК користувача:
- Процесор - Intel Pentium 1,5 ГГц;
- Обсяг оперативної пам'яті – 256 МБ;
- Дискова пам'ять – 40 ГБ;
- Мережевий адаптер - 100 Мбіт/с.

Розроблене програмне забезпечення може виконуватись на операційних системах Windows та Linux за умови, що встановлене програмне забезпечення, необхідне для роботи продукту.

2.5.3. Вимоги до вхідних та вихідних даних

Вхідними даними для розробленого програмного продукту є база даних з інформацією про чати та користувачів.

Вихідними даними є онлайн-чат з можливістю реєстрації та спілкування онлайн.

2.5.4. Вимоги до інтерфейсу

Інтерфейс програмного продукту повинен бути декларативним. Користувач повинен мати можливість реєструватися в системі, переглядати свій профіль, спілкуватись онлайн.

Взаємодія із застосунком здійснюється через додаток.

Користувач має можливість переглядати свій профіль.

Окрім цього, у користувача повинна бути можливість розширювати функціональність розробленого програмного продукту, маючи можливість здійснювати онлайн спілкування.

2.5.5. Вимоги до тестування програмного забезпечення

Для тестування програмного забезпечення необхідно виконати наступні дії:

- Зареєструватися в системі;
- Зробити вхід в систему;
- Почати спілкування;

При виконанні вище перерахованих дії для тестування роботи програмного забезпечення, користувач буде зареєстрований у системі, його данні з’являться у базі даних, та матиме можливість спілкуватись онлайн.

2.5.6 Вимоги до програмної документації

Програмне забезпечення постачається разом із супроводжувальною документацією, до складу якої входить:

- Технічне завдання.
- Опис та обґрунтування обраної архітектури.
- Функціональна специфікація.
- Технічна специфікація.
- Опис програми.
- Програма та методика випробувань.

3. РОЗРОБКА ДОДАТКУ ОНЛАЙН-ЧАТУ ДЛЯ СПІЛКУВАННЯ

3.1 Структура додатку

На етапі логічного проектування описується організація елементів, що становлять програмне рішення. Модель, отримана на стадії логічного проектування, повинна забезпечувати:

- незалежність від засобів розробки;
- простота моделі;
- відображення структури програмного продукту, що розробляється.

У логічній моделі створюються алгоритми чи інші логічні елементи, з яких здійснюється рішення прикладної задачі. На рис. 3.1. показана діаграма потоків даних, що показує взаємодію користувача та програми.

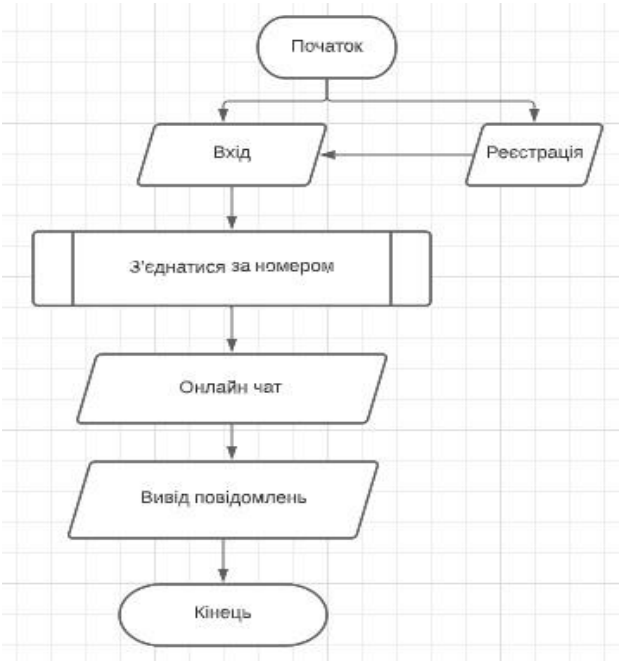


Рисунок 3.1 – Блок-схема роботи програми

3.2 Проектування та розробка БД

Для реалізації веб-додатка була обрана стандартна клієнт-серверна архітектура, що має три чітко розділені рівні: рівень подання даних (інтерфейс користувача), серверна частина, що забезпечує логічну обробку подій, і база даних як рівень управління даними. Ця архітектура була обрана з урахуванням не лише вимог до швидкості роботи програми на пристроях користувачів, але й з урахуванням забезпечення розподілу рівнів доступу до інформації. Також варто відзначити, що було обрано клієнт-серверну архітектуру з товстим сервером і тонким клієнтом. Такий розподіл логіки обробки інформації обумовлений вимогою підтримки сайту будь-яким браузером і з будь-якого пристрою, тому мінімізується використання ресурсів пристроїв користувачів. Крім того, ця архітектура забезпечує віддалене зберігання даних, незалежно від сеансу користувача або роботи сервера. Схематично архітектура програми представлена рис. 3.2.

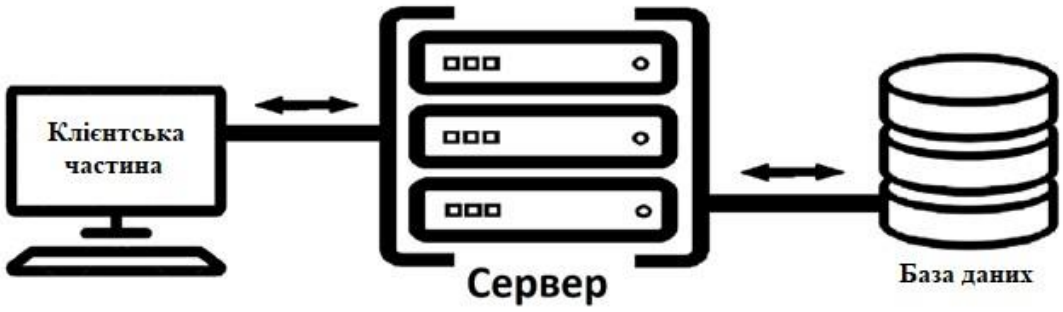


Рисунок 3.2 – Архітектура програми

У системі керування контентом застосовується СУБД PgAdmin. База даних складається із таблиці.

Усі дані зберігаються на локальному сервері в окремих таблицях.

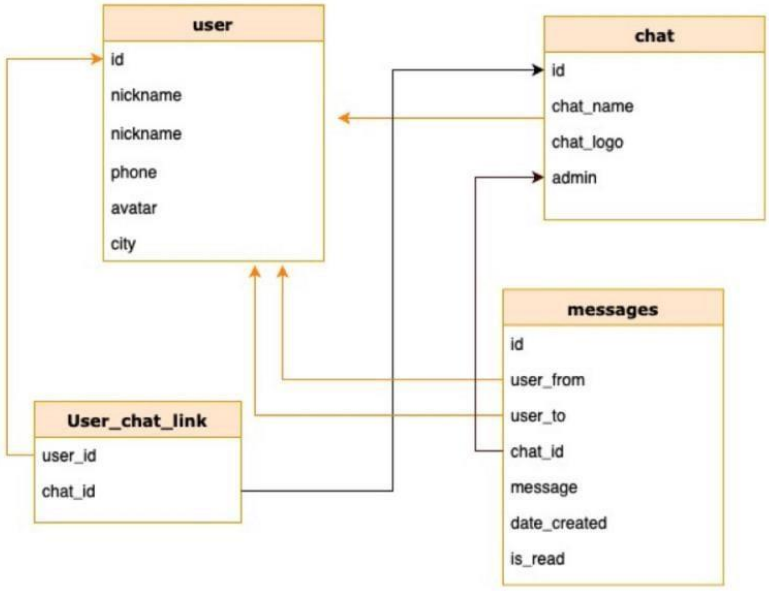


Рисунок 3.3 - Структура БД

Для підключення до бази даних та налаштування таблиць було використано наступний код програми.

```
public class AppDbContext : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<ContactUs> ContactUsMessages { get; set; }
public DbSet<Role> Roles { get; set; }        public
DbSet<ChatModel> Chats { get; set; }        public DbSet<UserChat>
UserChat { get; set; }        public DbSet<Message> Messages { get;
set; }


        public      AppDbContext(DbContextOptions<AppDbContext>
    options) : base(options)
        {
            Database.EnsureCreated();
        }
        protected override void OnModelCreating(ModelBuilder
    modelBuilder)        {
            var adminRole = new Role()
            {
                Id = 1,
```

```
            Name = "admin"
        };

        var userRole = new Role()
        {
            Id = 2,
            Name = "user"
        };

        var admin = new User()
        {
            UserId = 1,
            FirstName = "Bill",
            LastName = "Tomson",
            Email = "Example@gmail.com",
            Age = 33,
            Password = "12345Qwerty".GetHash(),
            RoleId = adminRole.Id
        };

        modelBuilder.Entity<Role>()
            .HasData(new Role[] { adminRole, userRole });

        modelBuilder.Entity<User>()
            .HasData(admin);
    }

}
```

Для додавання в базу даних інформації було використано наступну частину коду:

```
if (await _db.Users.AnyAsync(s => s.Email == model.Email))
return "There is already a user with this email";
```

```
                var user = new User()
                {
                    Email = model.Email,
                    FirstName = model.LastName,
                    LastName = model.LastName,
                    Age = model.Age,
                    Password = model.Password.GetHash(),
                    FK_Role_Id = await
    _db.Roles.FirstOrDefaultAsync(r => r.Name == "user")
                };

                await _db.Users.AddAsync(user); await
_db.SaveChangesAsync(); await Authenticate(user);

                return null;
```

3.3 Опис основних форм інтерфейсу онлайн-чату

Для входу в систему було створено форму із двох полів для логіна та паролю, та двох кнопок входу (рис. 3.4) та реєстрація(рис. 3.5)

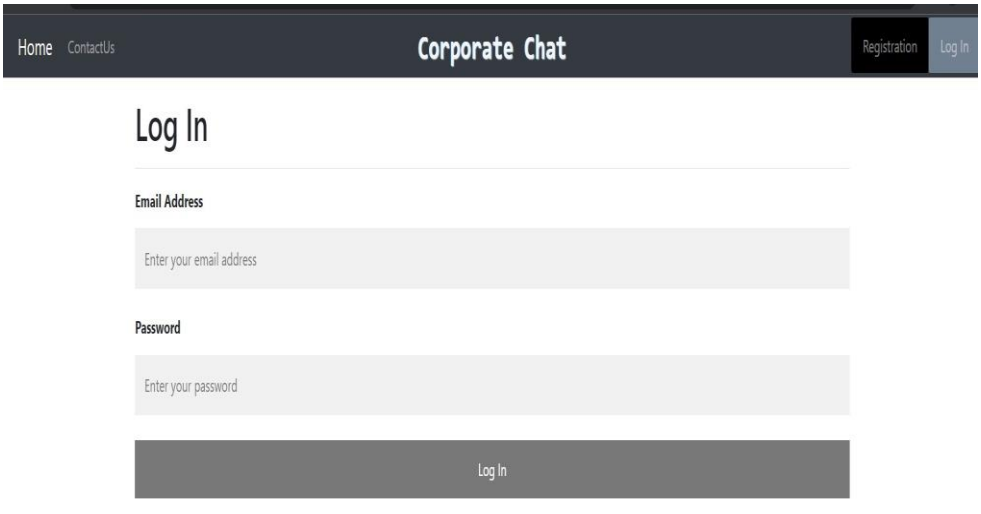


Рисунок 3.4 – Форма входу

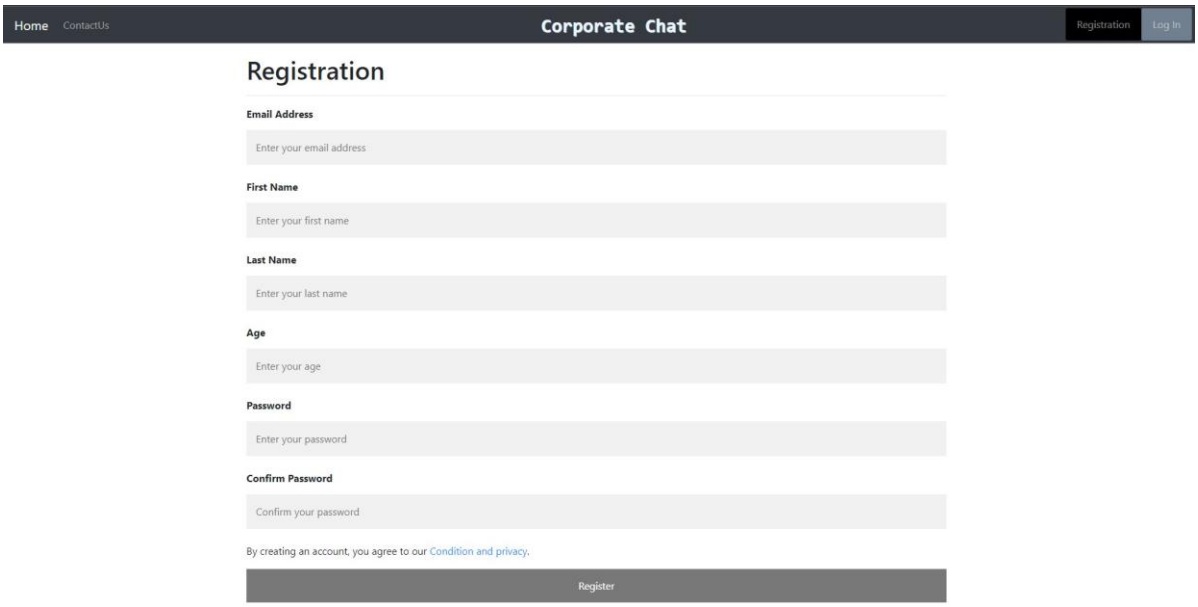


Рисунок 3.5 – Форма реєстрації

Для з’єднання з користувачем, було зроблено дві кнопки, одна з’єднує з рандомним користувачем, інша з вказаним Для входу в окремі чати, було створено сторінку, де відображаються списки чатів доступні користувачу та функція створення нового чату (рис. 3.6).

Рисунок 3.6 – список з чатами, які доступні користовачу

Також було створено поле для онлайн спілкування. В ньому відображаються перелік користувачів, обрання користувача з ким зв’язатися, поле для відображення діалогу та полем для введення нового повідомлення (рис. 3.7).

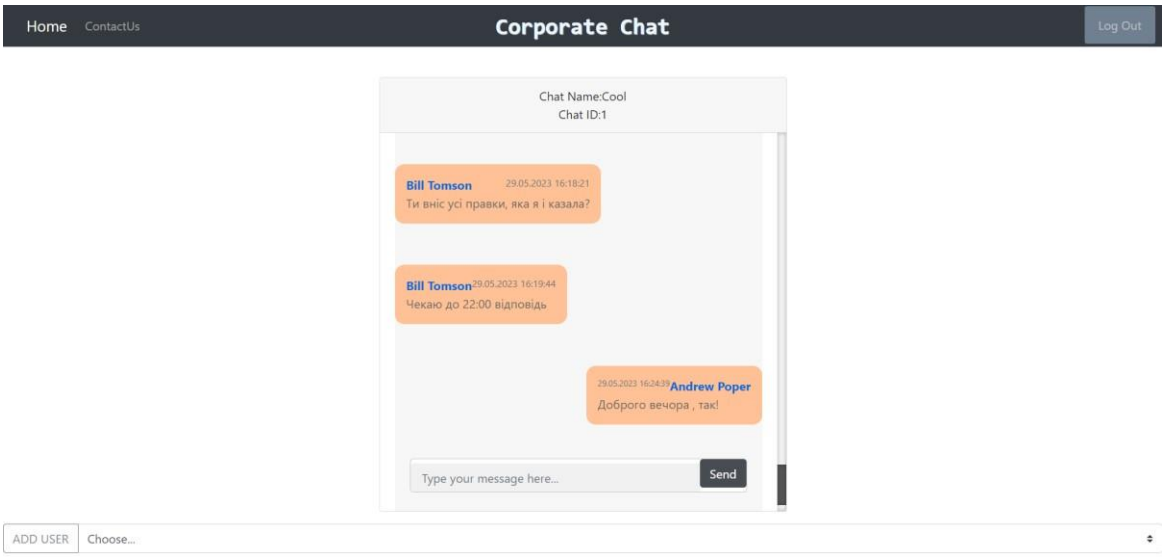


Рисунок 3.7 – форма онлайн-чату

3.4 Опис функціоналу додатка

Перед запуском програми потрібно запустити локальний сервер. У нашому випадку це Hamachi (рис. 3.8).

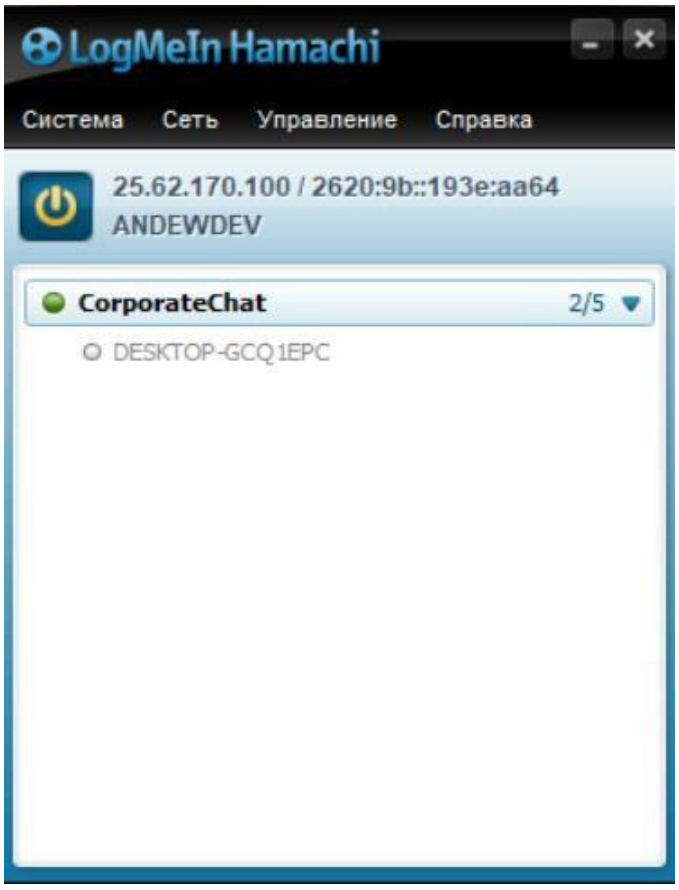


Рисунок 3.8 – запуск локального сервера

Запустивши програму, перед вами постане наступне вікно. Тут ви зможете авторизуватись, та користуватися чатом (рис. 3.9)

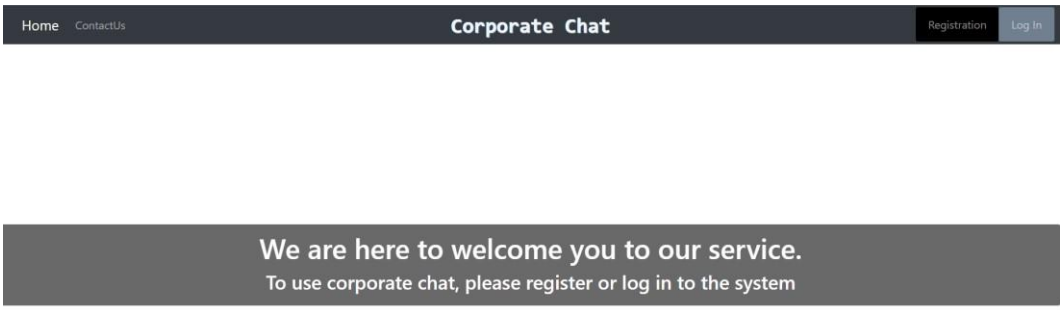


Рисунок 3.9 – стартове вікно програми

Далі ввівши логін «Example@gmail.com» та пароль «12345Qwerty» користувач ввійде як адміністратор, або інший логін чи пароль користувач увійде як звичайний користувач (рис. 3.10).

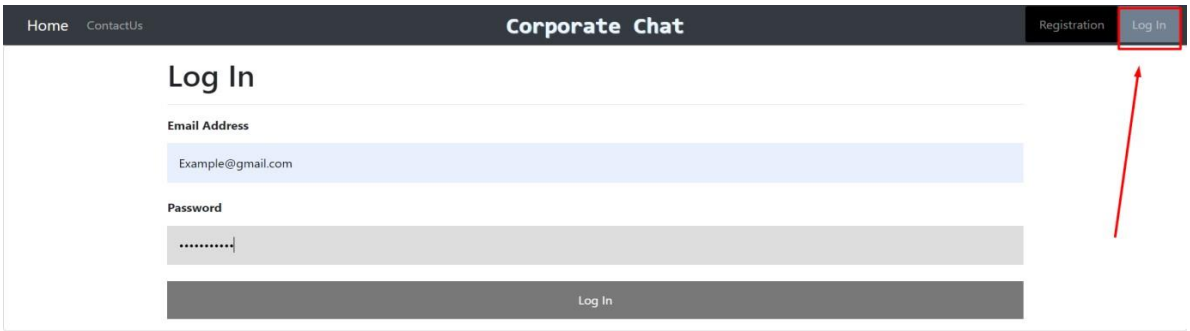


Рисунок 3.10 – Вхід

Після входу, користувач може переглянути свої чати або створити новий (рис. 3.11)

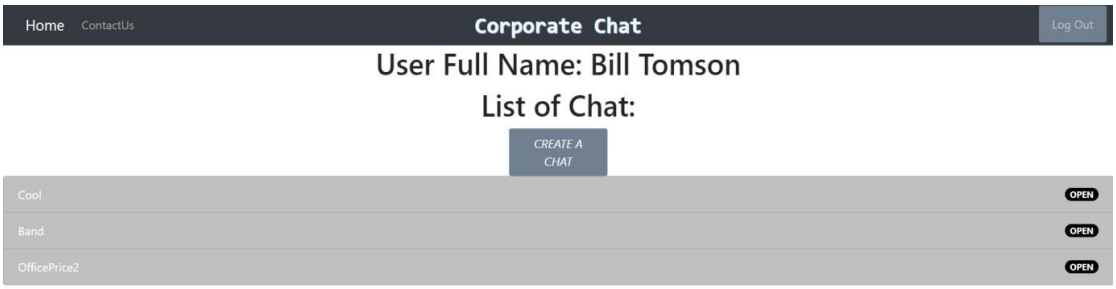


Рисунок 3.11 – Список чатів

Ввійшовши до системи користувач може обрати з ким почати спілкуватися, та почати писати повідомлення.

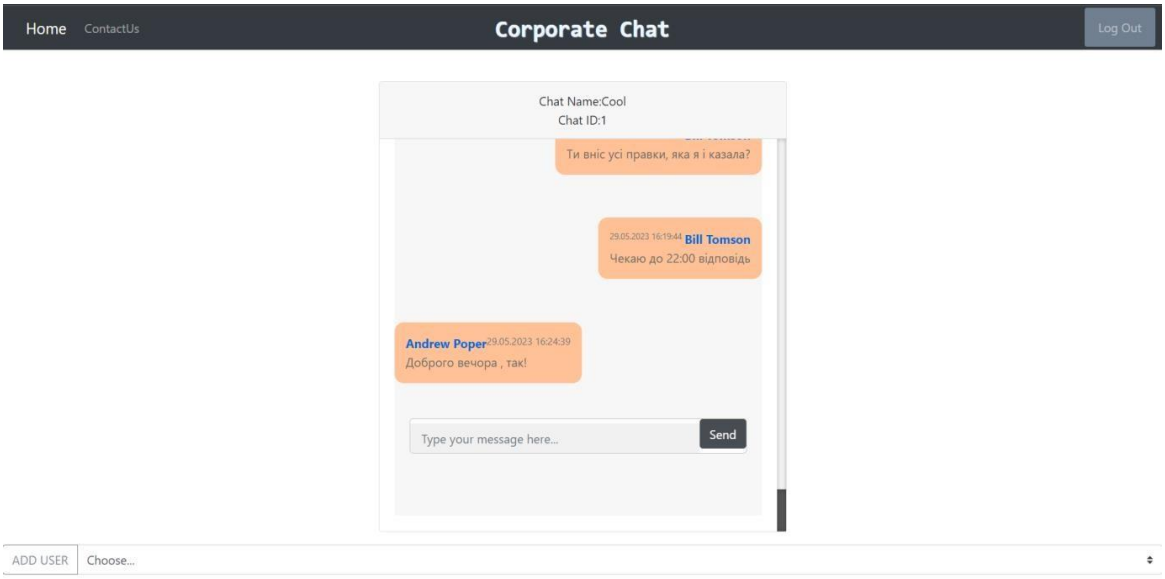


Рисунок 3.12 – можливість писати повідомлення

3.5 Тестування розробленого додатку

Під час тестування усі виникаючі помилки одразу ж виправлялись. Нижче наведено декілька прикладів тестування роботи програми.

Починалось тестування з реєстрації користувача (рис. 3.13).

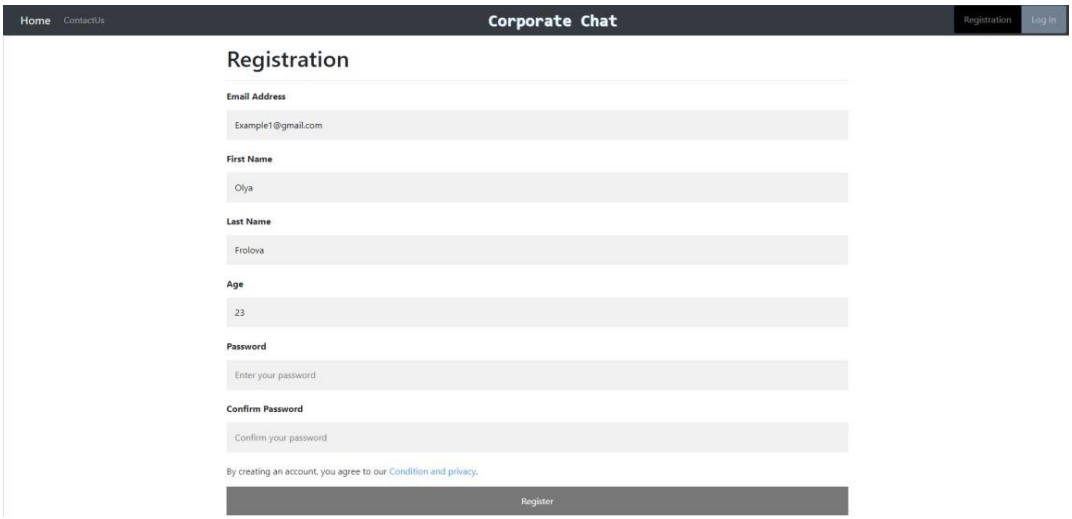


Рисунок 3.13 – тестування «додавання користувачів»

Наступним кроком було тестування додавання повідомлення (рис. 3.14)

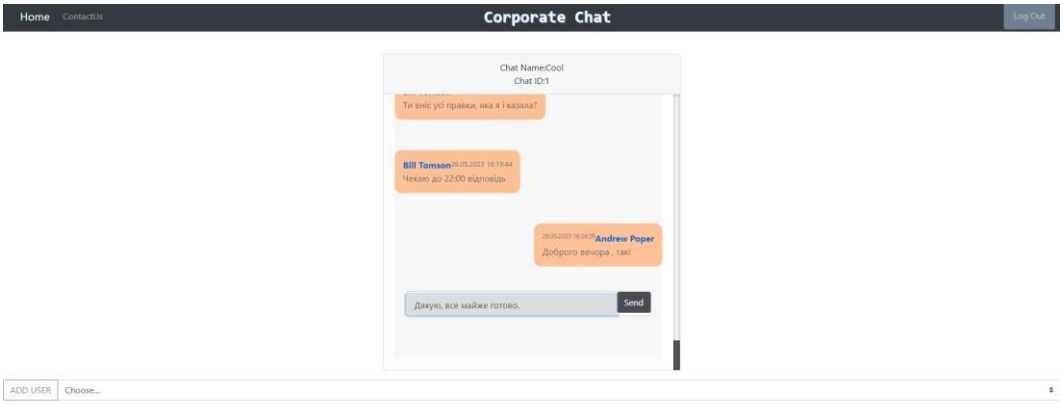


Рисунок 3.14 – тестування «додавання повідомлення»

Після проведення тестування всі помилки в моєму онлайн чаті були виправлені, що значно покращило його функціональність та надійність. Тепер користувачі можуть відправляти повідомлення без перебоїв та забезпечувати ефективне корпоративне спілкування. Виправлення помилок дозволило забезпечити стабільну роботу чату і покращити загальне враження від його використання. Результати тестування дозволили забезпечити оптимальну функціональність та зручний інтерфейс для користувачів.

ВИСНОВКИ

Результатом дипломної роботи було створення онлайн-чату для спілкування мовою C# на платформі ASP.NET.

На ринку програмного забезпечення існує велика кількість подібних програмних продуктів, які також допомагають вирішувати порушені мною проблеми. Однак подібні програми найчастіше вимагають від користувача максимум знань ПК, і споживають велику кількість ресурсів комп'ютера, тому, власне, і було вирішено написати цю програму.

Цей програмний продукт дає можливість, маючи мінімальні знання для роботи на комп'ютері, вільно користуватися програмою і робити будь-які підстроювання.

Цей програмний продукт може бути використаний як у домашніх умовах, так і в умовах виробництва.

Реалізовано програмний продукт, що дозволяє надсилати повідомлення по мережі Інтернет.

Тестування програмного продукту показало, що програма відповідає вимогам замовника та користувачів. Програмний продукт відповідає основним чинникам якості, таким як зрозумілість, стислість, супроводжуваність, ефективність.

ПЕРЕЛІК ПОСИЛАНЬ

1. PostgreSQL. [Електронний ресурс] – Режим доступу: https://www.postgresql.org/ (дата звернення 10.05.23р)
2. Шілдт Г. "Самоучитель C#". – Київ: Освіта, 2017. – 670 с.
3. Мєшков А., Тихомиров Ю. "Visual C++ і MFC". – Харків: Місто, 2018. – 1017 с.
4. Культін Н. "C# в задачах та прикладах". – Одеса: МКТ, 2012. – 288 с.
5. Петров Б., Алексєєв Т. "C#". – Запоріжжя, 2021. – 370 с.: іл.
6. Брагін І. "Побудова мереж". – Київ: Київ, 2020. – 480 с.: іл.
7. Entity Framework Core. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/ef/core/ (дата звернення 10.05.23р)
8. Hamachi. [Електронний ресурс] – Режим доступу: https://www.vpn.net/ (дата звернення 10.05.23р)
9. ASP.NET Core. [Електронний ресурс] – Режим доступу: https://dotnet.microsoft.com/apps/aspnet (дата звернення 10.05.23р)
10. Відеоуроки по C# на сайті Learn Visual Studio. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/csharp/ (дата звернення 10.05.23р)
11. Офіційний сайт Microsoft для завантаження Visual Studio. [Електронний ресурс] – Режим доступу: https://visualstudio.microsoft.com/ (дата звернення 10.05.23р)
12. Офіційна документація з EF Core для PostgreSQL. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/ef/core/providers/npgsql/ (дата звернення 10.05.23р)
13. Офіційна документація HTML and CSS. [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення 10.05.23р)
14. Офіційна документація Bootstrap. [Електронний ресурс] – Режим доступу: https://getbootstrap.com/ (дата звернення 10.05.23р)

ДОДАТКИ

Додаток А (HomeController.cs)

```
using NStack; using Terminal.Gui;

namespace VectorChat.Client_Console
{
 internal class MainWindow : Window
 {
     public MainWindow(ustring title = null) :
base(title) { }

  public MainWindow(Rect frame, ustring title = null) :
base(frame, title) { }

  public MainWindow(ustring title = null, int padding =
0) : base(title, padding) { }

    public MainWindow(Rect frame, ustring title = null,
int padding = 0) : base(frame, title, padding) { }

     public MainWindow() : base() { }
 }
} public class HomeController : Controller
    {
        private AppDbContext _context;         private
readonly HttpContext _httpContext;         private
IChatService _chatService;
```

```
        public   HomeController(AppDbContext   context,
IHttpContextAccessor httpContextAccessor, IChatService
chatService)
        {
            _context = context;
            _httpContext                                =
httpContextAccessor.HttpContext;
            _chatService = chatService;
        }
        [HttpGet]         [Route("/")]
        public IActionResult HomePage()
        {
           if (User.Identity.IsAuthenticated)
return   RedirectToAction("ShowChat",
"Home");             return View();
        }

        [HttpGet]         [Route("/about")]
        public IActionResult ContactUs()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        [Route("/about")]
        public        async        Task<IActionResult>
ContactUs(ContactUs data)
        {
```

```
            if (!ModelState.IsValid)
return View();

            await
_context.ContactUsMessages.AddAsync(data);
await _context.SaveChangesAsync();

            return RedirectToAction("HomePage",
"Home");
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
[Route("/show/message")]
        public async Task<IActionResult> ShowMessage()
        {
            return                              View(await
_context.ContactUsMessages.ToListAsync());
        }

        [HttpGet]
        [Authorize]         [Route("/chat")]
        public IActionResult ShowChat()
        {

            var chatIds = _context.UserChat
                .Where(uc      =>      uc.UserId      ==
_httpContext.User.GetUserId())
                .Select(uc => uc.ChatId)
                .ToList();
```

```
            var userChats = _context.Chats
                .Where(c => chatIds.Contains(c.ChatId))
                .ToList();

            var currentUserID=
_httpContext.User.GetUserId();
            ViewBag.currentUser =
_context.Users.FirstOrDefault(u => u.UserId ==
currentUserID);

            return View(userChats);
        }


        [HttpGet]
        [Authorize]         [Route("/newchat")]
        public IActionResult CreateChat()
        {
            return View();
        }

        [HttpPost]
        [Authorize]
        [Route("/newchat")]
        public    IActionResult    CreateChat(ChatModel
chatModel)         {
            var                result                =
_chatService.CreateChat(chatModel);

            if (result == null)
```

```
                return    RedirectToAction("ShowChat",
"Home");
            return View();
        }

        /*[HttpPost]
        [Route("/chat/window")]
        public IActionResult ChatWindow(Message
message)
        {
            if (!ModelState.IsValid)
return View(message);

            var message = new Message()
            {
                ChatId = _context.Users;
            }
            return View();
        }*/
    }
```

Додаток Б (program.cs)

```
using Chat.dyplom.Web.Domain; using
Chat.dyplom.Web.Services;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Builder; using
Microsoft.AspNetCore.Hosting; using
Microsoft.AspNetCore.Http; using
Microsoft.EntityFrameworkCore; using
Microsoft.Extensions.Configuration; using
Microsoft.Extensions.DependencyInjection; using
Microsoft.Extensions.Hosting;
```

```
namespace Chat.dyplom
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }         public IConfiguration Configuration {
get;    }                                   public   void
ConfigureServices(IServiceCollection services)        {
            string connection =
Configuration.GetConnectionString("DefaultConnection");
services.AddDbContext<AppDbContext>(options =>
options.UseNpgsql(connection));

            services.AddScoped<IAuthService,
AuthService>();
            services.AddScoped<IChatService,
ChatService>();
            services.AddScoped<IMessageService,
MessageService>();

            services.AddHttpClient();
services.AddHttpContextAccessor();


services.AddAuthentication(CookieAuthenticationDefaults
.Aut henticationScheme)
                .AddCookie(options =>
                {
                    options.LoginPath = new
PathString("/login");
                });

            services.AddAuthorization();
services.AddMvc();
        }

        // This method gets called by the runtime. Use
this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app,
IWebHostEnvironment env)
        {
```

```
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }            else            {
                app.UseExceptionHandler("/Home/Error");
// The default HSTS value is 30 days. You may want to
change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
app.UseHsts();
            }
            app.UseHttpsRedirection();
app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthentication();
app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
name: "default",                    pattern:
"{controller=Home}/{action=HomePage}");
            });
        }
```

Додаток В (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка онлайн-чату для спілкування на платформі ASP.NET мовою C#

Виконав студент 4 курсу
групи ПД-44
Побережник Андрій Анатолійович
Керівник роботи
д.т.н., доц., зав. кафедри ТЦР Жебка Вікторія Вікторівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спрощення процесу корпоративного спілкування за рахунок використання онлайн-чату, написаного мовою програмування C#.
- **Об’єкт дослідження** – корпоративне спілкування.
- **Предмет дослідження** – онлайн-чат для корпоративного спілкування.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз існуючих аналогічних програм.
2. Розробити вимоги до програмного забезпечення, враховуючи недоліки наявних рішень.
3. Провести аналіз робочих процесів онлайн-чатів та виявити процеси, що негативно впливають на їх ефективність.
4. Проаналізувати необхідні засоби для створення онлайн-чату.
5. Спроектувати схему бази даних для зберігання інформації користувачів.
6. Реалізувати проект із використанням програмних засобів.
7. Протестувати розроблений онлайн-чат.

АНАЛІЗ АНАЛОГІВ

Назва	Staply	Slack	HipChat	**Розроблений додаток**
Інтеграція зі сторонніми сервісами	-	+	+	**-**
Створення ієрархії груп	+	-	-	**-**
Розподіл керуючим учасників за ролями	-	-	-	**+**
Настроювання оформлення	+	+	+	**+**
Робота в локальній мережі	-	-	-	**+**

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

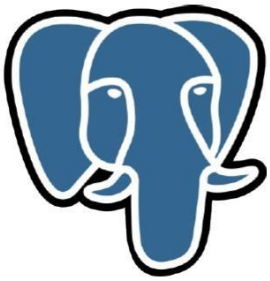
1. Користувач може зареєструватися, ввівши обов'язкову інформацію, таку як ім'я, електронну пошту та пароль.
2. Користувач може авторизуватися в системі, використовуючи введені при реєстрації облікові дані.
3. Після авторизації користувач може переглядати список наявних чатів.
4. Користувач може створити новий чат, вказавши його назву.
5. Користувач може вибрати існуючий чат для спілкування.
6. В чаті користувач може додавати інших учасників, вказуючи їх імена або електронні адреси.
7. Користувач може відправляти повідомлення в обраний чат.
8. Адміністратор має можливість входу в систему зі спеціальними привілеями.
9. Адміністратор може видаляти будь-який чат зі списку наявних.
10. Адміністратор може блокувати користувача, що призводить до обмеження його доступу до чатів та повідомлень.
11. Чат працює в локальній мережі.

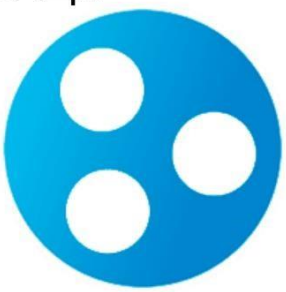
Нефункціональні вимоги:

1. Система має бути безпечною і захищеною від несанкціонованого доступу до користувацьких даних.
2. Чат повинен бути надійним і стабільним, з мінімальною ймовірністю виникнення помилок або збоїв.
3. Всі дані, введені користувачем, повинні бути достовірними та зберігатися у системі.
4. Інтерфейс користувача повинен мати привабливий та зручний дизайн.
5. Система має бути швидкою та ефективною у використанні.
6. Вимоги до продуктивності системи повинні бути задоволені, навіть при великій кількості користувачів та чатів.
7. Чат має забезпечувати конфіденційність та приватність повідомлень між користувачами.
8. Система повинна бути легко розширюваною, щоб додавання нових функцій та можливостей було простим процесом.
9. Вся комунікація між клієнтом і сервером повинна бути зашифрованою для забезпечення безпеки даних.
10. Система повинна бути сумісною з різними операційними системами та браузерами для забезпечення доступності для широкого кола користувачів.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

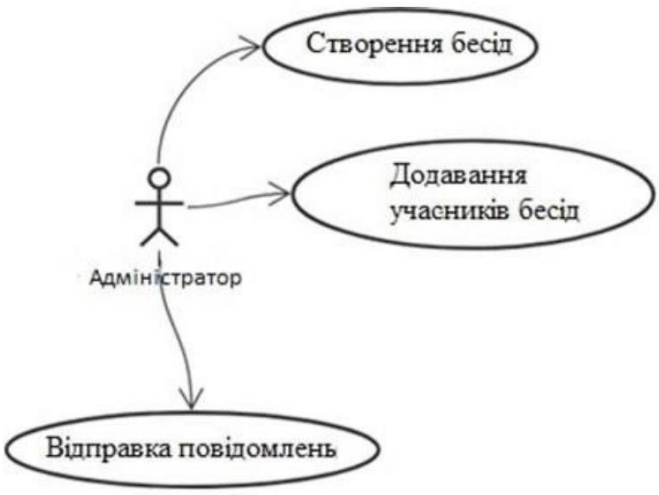




ASP.NET Core

6

ДІАГРАМА ПРЕЦЕДЕНТІВ

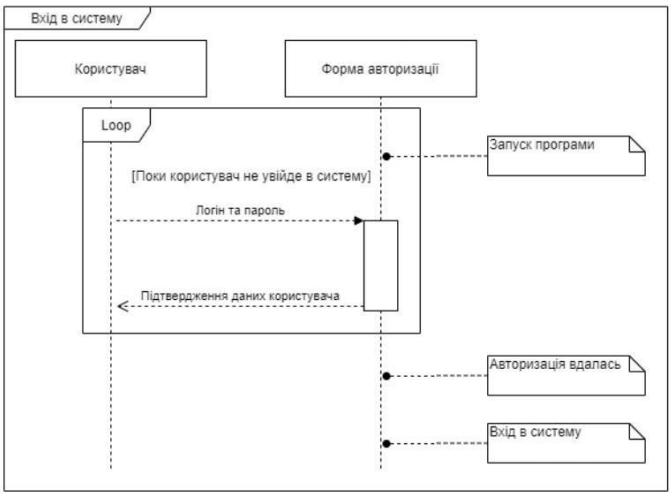


7

ДІАГРАМА ПРЕЦЕДЕНТІВ

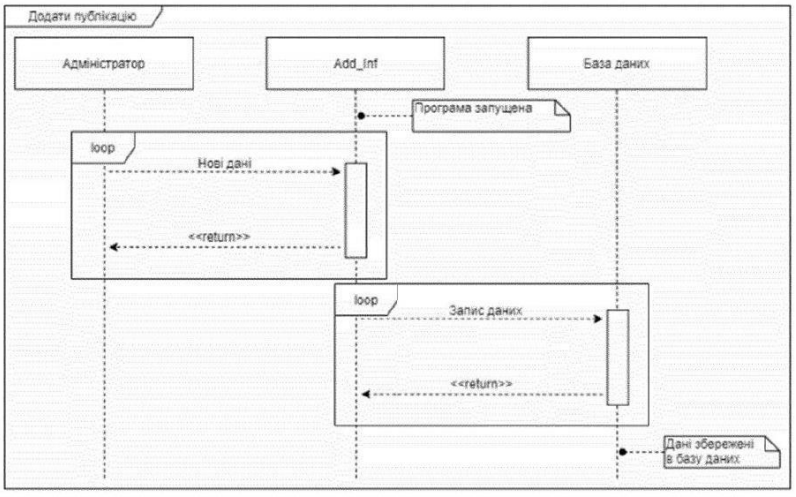
8

ДІАГРАМА ПОСЛІДОВНОСТІ



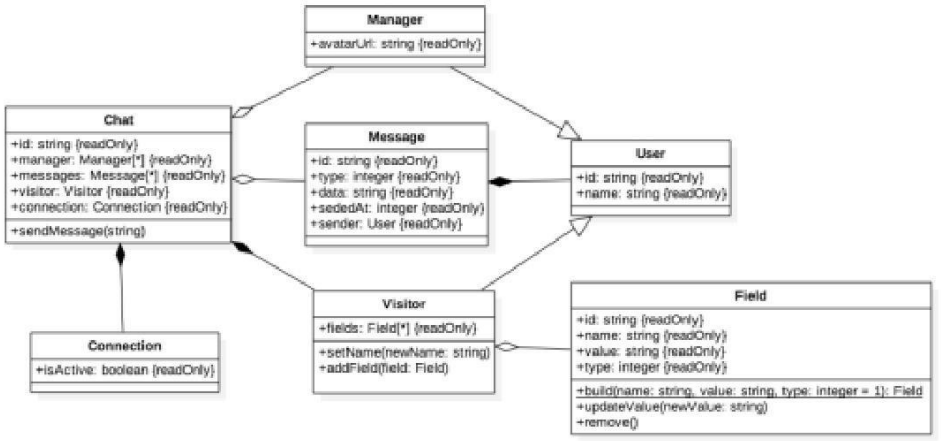
9

ДІАГРАМА ПОСЛІДОВНОСТІ



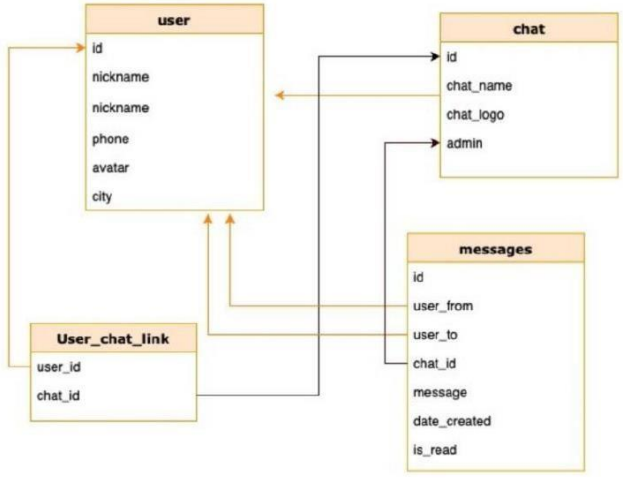
10

ДІАГРАМА КЛАСІВ



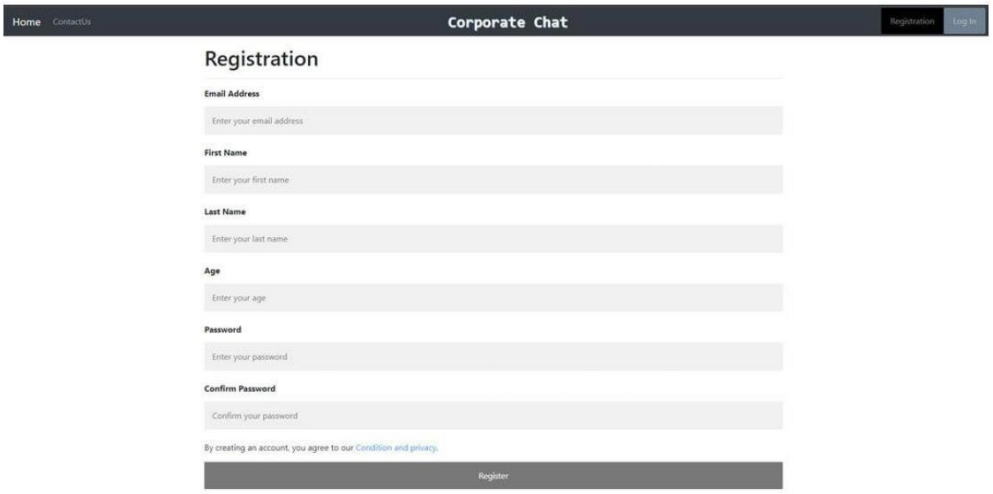
11

СТРУКТУРА ТАБЛИЦЬ БАЗИ ДАНИХ



12

ЕКРАННА ФОРМА



Форма реєстрації

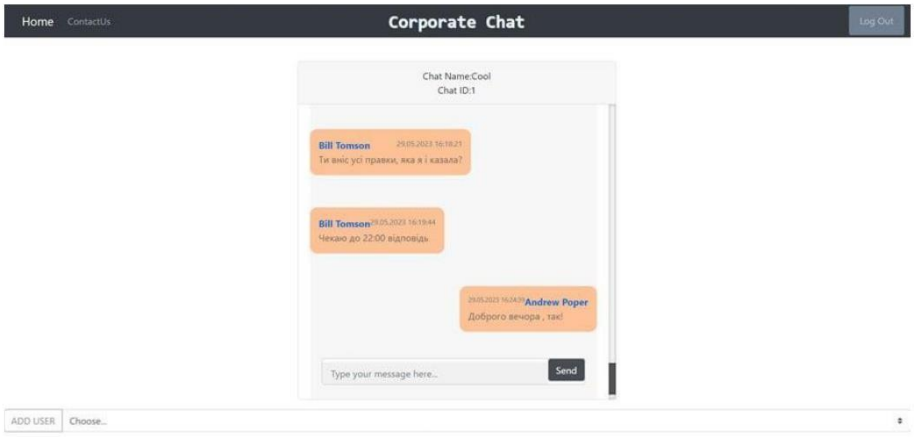
ЕКРАННА ФОРМА

Форма входу

ЕКРАННА ФОРМА

Сторінка перегляду доступних чатів

ЕКРАННА ФОРМА



Форма чату

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Побережник А.А. Розробка онлайн-чату для спілкування мовою C# на платформі ASP.NET/ А.А. Побережник // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії: Матеріали науково-практичної конференції. Збірник тез, 01.06.23, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 31 – 32.
- Побережник А.А. Опис технології Hamachi/ А.А. Побережник // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії: Матеріали науково-практичної конференції. Збірник тез, 01.06.23, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 43 – 44.

ВИСНОВКИ

1. Дипломна робота успішно виконала мету створення онлайн-чату на платформі ASP.NET з використанням мови C#.
2. Виявлено, що існуючі програмні продукти для спілкування мають високі вимоги до знань користувача та ресурсовитратність, що спонукало до створення даної програми з метою забезпечити простоту використання і низький рівень вимог до комп'ютера користувача.
3. Розроблений програмний продукт забезпечує можливість використання без потреби високих комп'ютерних навичок, а також дозволяє здійснювати різноманітні налаштування.
4. Програмний продукт може бути успішно застосований як у домашніх умовах, так і у виробничому середовищі.
5. Було реалізовано функціонал надсилання повідомлень по мережі Інтернет.
6. Під час тестування програмний продукт відповідав вимогам замовника та користувачів, а також демонстрував зрозумілість, стислість, супроводжуваність та ефективність.