Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Бакалавр
(назва освітнього ступеня)

на тему: Розробка сервісу обміну текстовими повідомленнями з використанням технологій .NET

Виконав: студент IV курсу, групи СП-43 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Чухарський М.Ю.
(підпис) (прізвище та ініціали)

Керівник _____ Михалик Д.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет _____
(повна назва факультету)

Кафедра _____
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня _____
(назва освітнього ступеня)

за спеціальністю _____
(шифр і назва спеціальності)

студенту _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____

Керівник роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст роботи (перелік питань, які потрібно розробити)

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	**Примітка**

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка сервісу обміну текстовими повідомленнями з використанням технології .NET // Кваліфікаційна робота освітнього рівня «Бакалавр» // Чухарський Максим Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-43 // Тернопіль, 2025 // С.62, рис. – 19, табл. – 0, додат. – 2, бібліо. – 18.

Ключові слова: клієнт-сервер, .NET, месенджер, WPF, WCF.

Кваліфікаційна робота присвячена розробці програми для обміну текстовими повідомленнями на основі технології .Net.

Метою цього проекту є детальне вивчення та практичне застосування технології .NET у розробці додатку з обміну текстовими повідомленнями.

У пешому розділі кваліфікаційної роботи проведено детальний аналіз предметної області месенджер (чат). Здійснено аналіз та вибір найбільш ефективних технологій для реалізації проекту.

У другому розділі кваліфікаційної роботи здійснено проєктування застосунку. Описана її функціональні можливості та визначено вимоги до системи. Створено діаграми прецедентів, сценарії використання та діаграми послідовності для ключових операцій.

У третьому розділі кваліфікаційної роботи було здійснено розробку програмного застосунку. Обґрунтовано вибір відповідних технологій та середовища розробки. Реалізовано основні елементи користувацького інтерфейсу, зокрема розроблено головну сторінку. Окрім цього, було проведено функціональне тестування застосунку з метою перевірки його стабільності, надійності та відповідності заданим вимогам.

ABSTRACT

Development of a text messaging service using .NET technology // Qualification work of the educational level "Bachelor" // Chukharsky Maksym Yuriyovych // Ivan Pulyuy Ternopil National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group SP-43 // Ternopil, 2025 // Pages.62, figures. – 19, tables. – 0, addition. – 2, literature – 18.

Keywords: client-server, .NET, messenger, WPF, WCF.

The qualification work is devoted to the development of a program for exchanging text messages based on .Net technology.

The goal of this project is to study in detail and apply .NET technology in the development of a text messaging application.

In the first part of the qualification work, a detailed analysis of the subject area of instant messaging (chat) is carried out. Analysis and selection of the most effective technologies for project implementation.

In the second section of the qualification work, the application was designed. Its functionality was described and the system requirements were defined. Use case diagrams, usage scenarios, and sequence diagrams for key operations were created.

In the third part of the qualification work, the development of the software was carried out. Selection of appropriate technologies and development environments is justified. The main elements of the copy interface were implemented, in particular, the main page was developed. In addition, functional testing of the application was carried out to verify its stability, reliability, and compliance with the specified requirements.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

.NET Framework – програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків.

WPF – Windows Presentation Foundation графічна підсистема, яка починаючи з NET Framework 3.0 в складі цієї платформи. Має пряме відношення до XAML. WPF разом з .NET Framework 3.0 вбудована в Windows Vista, а також доступна для установки в Windows XP Service Pack 2 і Windows Server 2003.

WCF – Windows Communication Foundation — набір клієнтських бібліотек, що дозволяють застосункам на базі відкритої платформи .NET Core взаємодіяти з сервісами WCF, відправляючи повідомлення між сервісами в асинхронному режимі.

Месенджер – система обміну миттєвими повідомленнями

UML – Unified Modeling Language уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

IDE – Integrated Drive Electronics комплексне програмне рішення для розробки програмного забезпечення.

ВСТУП

В епоху високотехнологічного прогресу, коли інформаційні та комунікаційні технології стають неодмінною частиною нашого повсякденного життя, ефективна комунікація стає ключовим фактором для успіху як в особистому, так і в професійному житті. У цьому контексті, мережеві додатки, зокрема чат-системи, відіграють критичну роль у забезпеченні миттєвого обміну інформацією та зближенні користувачів, незалежно від їх фізичного розташування. Однак розробка таких додатків вимагає потужних та надійних інструментів для розробки, щоб забезпечити швидку та безперебійну передачу даних.

Windows Communication Foundation (WCF) виходить на передній план як важливий інструмент для створення розподілених застосунків у середовищі операційних систем Windows. WCF забезпечує однорідний підхід до створення служб інтерактивного взаємодії, які можуть працювати як локально, так і в розподілених мережах. Ця технологія відкриває широкі можливості для створення надійних, ефективних та масштабованих мережевих додатків.

Метою цього проекту є детальне вивчення та практичне застосування технології .NET, WPF та WCF у розробці сервісу з обміну текстовими повідомленнями. Ми розглянемо основні концепції WCF, вивчимо його архітектурні особливості та оптимальні підходи до реалізації. Особлива увага буде приділена аспектам безпеки та масштабованості чат-додатку, забезпечуючи його ефективну роботу в умовах реального використання.

Подальша частина цього проекту розглядається як спроба поєднати теоретичні аспекти WCF WPF та .NET з практичним застосуванням у розробці сервісу. Результатом буде не лише функціональний продукт, а й глибше розуміння та навички використання сучасних інструментів для побудови надійних та ефективних розподілених систем комунікації.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд вже створених рішень для обміну текстовими повідомленнями

Приклад 1: WhatsApp

WhatsApp — це дуже зручний і простий месенджер, який працює через інтернет. Щоб користуватись ним, потрібно зареєструватись через номер телефону. Програма дозволяє надсилати текстові повідомлення, голосові повідомлення, фото, відео, документи. Також можна робити дзвінки — як звичайні, так і відео. Усе, що ти надсилаєш, захищене шифруванням, тобто ніхто не зможе це прочитати, окрім тебе і того, кому ти пишеш [1]. Інтерфейс зображений на рисунку 1.1.

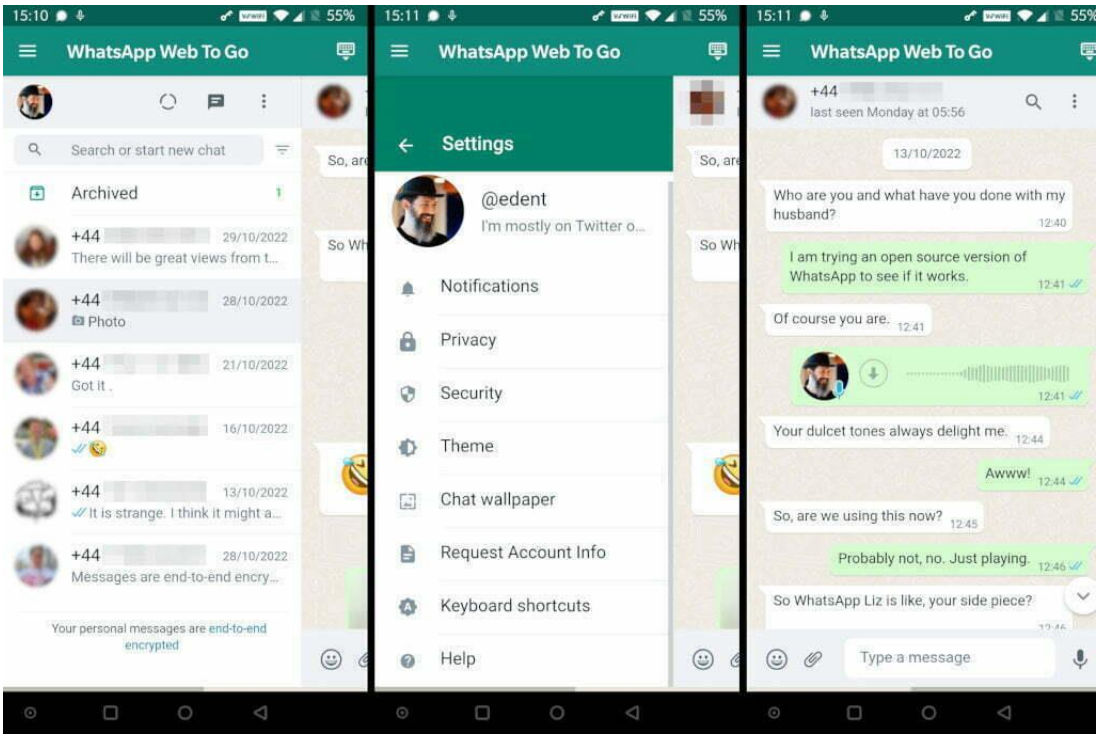

Рисунок 1.1 – Інтерфейс WhatsApp

Основні функції це написання повідомлень, надсилання голосових повідомлень, фото та відео, обмін файлами, створення групових чатів, відеодзвінки та дзвінки, статуси (як сторіс), відправка місцезнаходження, використання на комп’ютері через WhatsApp Web, підтримка бізнес-акаунтів.

Переваги:

- Простий і зрозумілий: легко розібратися навіть людям без досвіду.
- Популярний: більшість знайомих уже користуються ним.
- Надійний захист: усі повідомлення та дзвінки зашифровані.
- Можна дзвонити: якісні аудіо та відеодзвінки, навіть за поганого інтернету.
- Синхронізація з комп’ютером: зручно писати з клавіатури через браузер.

Недоліки:

- Потрібен номер телефону: не можна зареєструватися без SIM-карти.
- Мало додаткових функцій: немає ботів, каналів чи кастомізації, як у Telegram.
- Резервне копіювання не автоматичне: потрібно налаштовувати вручну через Google Drive або iCloud.
- Менше контролю над чатами: складно шукати старі повідомлення чи керувати файлами в чатах [1].

Приклад 2: Telegram

Основні функції це написання повідомлень, надсилання фото, відео та файлів до 2 ГБ, створення груп (до 200 000 людей), публічні та приватні канали, голосові та відеодзвінки, голосові чати (аналог Clubhouse), створення ботів, секретні чати з автознищенням повідомлень, використання на кількох пристроях одночасно, кастомізація тем і фонів [2]. Інтерфейс зображений на рисунку 1.2.

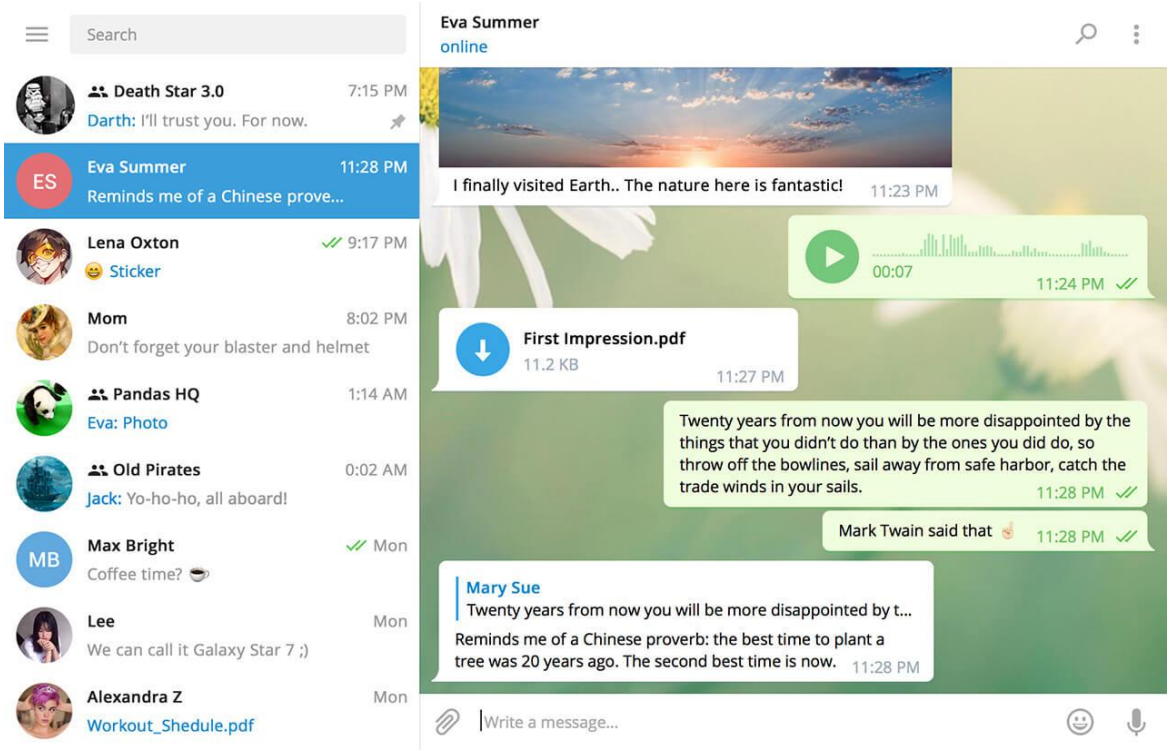

Рисунок 1.2 – Інтерфейс Telegram

Переваги:

- Швидкий і сучасний: працює дуже плавно і не "гальмує".
- Без обмежень: можна надсилати великі файли, створювати великі групи.
- Працює на кількох пристроях: можна користуватись на телефоні, планшеті, комп'ютері одночасно.
- Є канали і боти: зручно читати новини, вчитися, грати, слідкувати за курсами валют тощо.
- Гнучкий інтерфейс: можна налаштувати зовнішній вигляд під себе.

Недоліки:

- Не все шифрується за замовчуванням: звичайні чати зберігаються в хмарі Telegram.
- Може бути занадто насиченим: багато функцій — не кожному це зручно.
- Секретні чати — тільки на одному пристрої: їх не можна переглядати з іншого пристрою [2].

Приклад 3: Viber

Основні функції це написання текстових повідомлень, надсилання фото, відео та файлів, голосові та відеодзвінки, створення групових чатів, спільноти (великі публічні чати), надсилання стікерів і GIF, голосові повідомлення, зникаючі повідомлення, передача місцезнаходження, виклики на звичайні номери (через Viber Out), робота на телефоні й комп’ютері [3].

Інтерфейс зображений на рисунку 1.3.

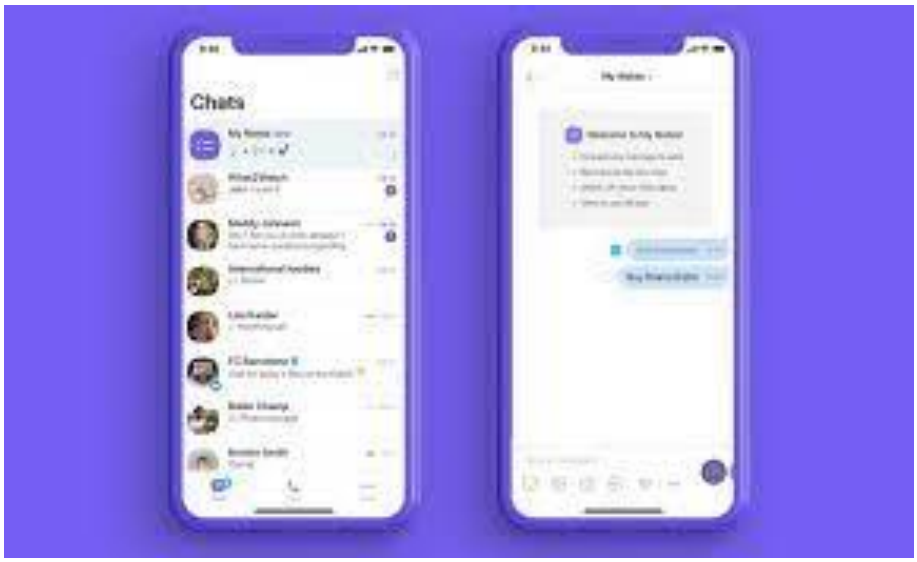

Рисунок 1.3 – Інтерфейс Viber

Переваги:

- Простий у користуванні: інтерфейс зрозумілий, підходить для всіх вікових груп.
- Добре підходить для дзвінків: особливо голосових — гарна якість навіть при слабкому інтернеті.
- Стікери та оформлення: багато безкоштовних і яскравих стікерів, є теми оформлення.
- Підтримка бізнесу: можна спілкуватися з брендами, службами доставки тощо.
- Спільноти: зручно для спілкування великої кількості людей (наприклад, шкільні чи районні чати).

Недоліки:

- Менше функцій, ніж у Telegram: немає ботів, менше гнучкості.
- Може працювати повільніше: на старих пристроях або при великій кількості чатів.
- Менша популярність серед молоді: молодші користувачі частіше обирають Telegram або Instagram.
- Залежність від номера телефону: не можна користуватись без SIM-карти [3].

Сьогодні існує багато популярних додатків для обміну повідомленнями, але найбільш відомі це WhatsApp, Telegram і Viber і їх я описав вище. Усі три мають багато спільного: дозволяють писати повідомлення, надсилати фото, відео, файли, робити дзвінки та створювати групові чати. Проте кожен із них має свої особливості й підходить для різних користувачів. Ось тому я вирішив створити свій сервіс з обміну текстовими повідомленнями.

1.2 Словник предметної області

Користувацький Інтерфейс (UI) - це спосіб взаємодії користувача з програмою чи системою. У випадку мережевого чату, це включає в себе всі графічні та текстові елементи, які користувач може бачити та з якими він може взаємодіяти, такі як вікна чату, кнопки, поле введення тексту, та інші елементи інтерфейсу, що спрощують використання чату.

Інтерфейс Обміну Повідомленнями - це частина чат-додатку, яка дозволяє користувачам взаємодіяти між собою, відправляти та отримувати повідомлення. Це включає в себе можливість відправляти текстові повідомлення, мультимедійні вкладення, створювати групові чати та керувати ними.

Повідомлення про Онлайн Статус - це функціонал, який відображає інформацію про те, чи перебувають інші користувачі в мережі чату. Воно показує, хто з користувачів на даний момент доступний для спілкування та хто відсутній або офлайн.

Керування Правами Користувачів - це система, яка надає адміністраторам чату або користувачам з певними привілеями можливість контролювати рівень доступу інших користувачів. Це може включати в себе встановлення прав на входження в групові чати, надання або обмеження можливостей відправлення повідомлень, а також інші функції, що регулюють взаємодію учасників чату.

1.3 Технологія Розробки

Для розробки сервісу обміну текстовими повідомленнями було обрано середовище Microsoft Visual Studio 2022, яке забезпечує зручні інструменти для створення додатків на основі платформи .NET. Ця платформа є потужною технологічною базою, що дозволяє розробляти масштабовані та безпечні застосунки з використанням великого набору сервісів та інструментів, незалежних від конкретної апаратної архітектури [5].

Ключовою мовою програмування для цього проєкту обрано C#, яка є основною для розробки в екосистемі .NET. Платформа .NET Framework, на якій базується сервіс, включає загальномовне середовище виконання (CLR) та багаті бібліотеки класів, що забезпечують керування пам’яттю, багатопоточність, безпечне виконання коду та інші критично важливі сервіси. Код, що виконується під управлінням CLR, називається керованим (managed code), і саме такий підхід гарантує більшу стабільність і безпеку виконання [4].

У процесі розробки програми для обміну текстовими повідомленнями було використано сучасні технології платформи .NET, зокрема WPF (Windows Presentation Foundation) для створення зручного графічного інтерфейсу

користувача, WCF (Windows Communication Foundation) для реалізації обміну повідомленнями між клієнтом і сервером, а також саму платформу .NET як основу для побудови всієї логіки програми [6].

WPF було обрано як головну технологію для створення інтерфейсу, оскільки вона надає широкі можливості для візуального оформлення, роботи з подіями, анімації та зв’язування даних (data binding). Завдяки WPF інтерфейс програми вийшов інтуїтивно зрозумілим, гнучким і адаптивним. Окремі елементи, такі як список повідомлень, поле введення тексту та кнопки керування, були реалізовані за допомогою стандартних елементів WPF і зв’язані з логікою програми через шаблони MVVM (Model-View-ViewModel), що дозволило чітко розділити інтерфейс і код програми [8].

Для реалізації зв’язку між клієнтами та сервером було застосовано WCF – технологію, яка дозволяє створювати надійні служби обміну даними. У межах цього проєкту WCF-сервіс відповідає за приймання та передачу повідомлень у реальному часі. Серверна частина програми реалізована як WCF-служба, яка отримує повідомлення від одного клієнта та транслює їх іншим. Завдяки цьому вдалося забезпечити швидкий і безпечний обмін повідомленнями між усіма активними користувачами [7].

Вся система побудована на базі платформи .NET, яка забезпечує надійну роботу застосунку, підтримку багатопоточності, управління пам’яттю та безпеку виконання. Платформа дозволяє об'єднати всі компоненти проєкту – інтерфейс, логіку, обробку даних і зв'язок з базою – в єдине узгоджене середовище. Такий підхід зробив програму масштабованою та зручною для подальшого розвитку.

Завдяки поєднанню WPF, WCF та .NET вдалося створити сучасний десктопний застосунок з приємним інтерфейсом та ефективною серверною логікою, що забезпечує обмін текстовими повідомленнями у зручному та зрозумілому вигляді. Таким чином у поєднанні з C# .NET, WCF, WPF, дозволив створити надійний, масштабований і продуктивний сервіс для обміну текстовими повідомленнями, орієнтований на сучасні вимоги до продуктивності та зручності користувача.

1.4 Аналіз вимог до системи

Мережевий чат є програмним продуктом, створеним для полегшення та покращення комунікації між користувачами, які перебувають в різних місцях та використовують розподілені комп'ютерні ресурси.

Виступає важливим інструментом для зближення віддалених користувачів, покращення спілкування та забезпечення швидкого та зручного обміну інформацією, що визначає його значення в різних сферах особистого та професійного життя.

Функціональні вимоги:

Система має передбачати наступні можливості:

- можливість входу в систему;
- можливість відправлення Текстових Повідомлень;
- можливість повідомлення користувачам про системні події.;
- Зміна світлої та темної теми;
- Основний функціонал кнопок вікна самого сервісу;

Нефункціональні вимоги:

1. Безпека:

- Шифрування трафіку: Всі дані, що передаються між клієнтами та сервером, повинні бути зашифровані для забезпечення конфіденційності.
- Механізми Аутентифікації: Застосування надійних методів аутентифікації користувачів перед доступом до системи чату.

2. Витривалість:

- Резервне Копіювання: Забезпечення системи резервного копіювання даних для відновлення в разі втрати інформації.
- Відновлення після Відмов: Швидке відновлення роботи системи в разі відмови чи збою.

3. Ефективність та Масштабованість:

- Оптимізація Ресурсів: Ефективне використання системних ресурсів для запобігання завантаження та збереження високої продуктивності.
- Масштабованість: Здатність системи працювати ефективно при збільшенні кількості одночасних користувачів.

4. Сумісність:

- Крос-платформенність: Забезпечення сумісності із різними операційними системами та пристроями.
- Сумісність Браузерів: Коректна робота чату в різних веб-браузерах.

5. Інтернаціоналізація та Мультимовність:

- Локалізація: Забезпечення можливості локалізації інтерфейсу для різних мов та регіональних налаштувань.
- Мультимовність Повідомлень: Підтримка різних мов для текстових повідомлень.

6. Забезпечення Якості:

- Тестування Продуктивності: Виконання тестів для оцінки продуктивності системи та виявлення можливих буттєвих точок.
- Валідація та Верифікація: Проведення систематичної перевірки на відповідність функціональним та нефункціональним вимогам.

7. Інтерфейс та Взаємодія:

- Відгуки та Зручність Використання: Забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів.
- Час Відповіді: Мінімізація часу відповіді системи на запити користувачів.

8. Безпека Даних:

- Захист від Атак: Забезпечення захисту системи від різних видів атак, таких як SQL-ін'єкції, атаки на отримання несанкціонованого доступу тощо.
- Бекапи та Збереження Даних: Використання механізмів регулярного створення резервних копій та збереження їх в безпечному місці.

9. Система Сповіщень та Журналювання:

- Відслідковування Подій: Ведення системного журналу подій для відслідковування та аналізу важливих подій у системі.
- Системні Сповіщення: Надсилання сповіщень користувачам про певні системні події або оновлення.

Вимоги до інтерфейсу системи:

1. Користувацький Інтерфейс:

- Зручність та Інтуїтивність: Інтерфейс повинен бути легким у використанні та інтуїтивно зрозумілим для користувачів будь-якого рівня.
- Дизайн та Ергономіка: Приділення уваги дизайну, колірній палітрі та компонуванню елементів для комфортного користування.

2. Меню та Навігація:

- Зручне Меню: Впровадження легкодоступного та організованого меню для навігації між різними функціями чату.
- Швидка Навігація: Забезпечення можливості швидкої навігації між основними розділами чату.

3. Вікна та Області Діалогу:

- Зручні Вікна Чату: Чітке та естетичне вікно чату з функціональними елементами для введення тексту, відправлення повідомлень та взаємодії.
- Відображення Групових Чатів: Зручне відображення групових чатів та можливість переключатися між ними.

4. Введення Тексту:

- Зручне Поле Введення: Поле для введення тексту повинно бути достатньо великим та зручним для користувача.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА КОНСТРУЮВАННЯ ЗАСТОСУНКУ

2.1 Опис проектування та технологій системи

Для проектування системи було обрано такі технології як WPF, WCF, .NET та мову програмування C#. Цей підхід дозволяє ефективно взаємодіяти з замовником, швидко реагувати на зміни та надавати користувачам реальні результати на кожному етапі розробки проекту мережевого чату.

1. Проектування Системи:

- Модульна Архітектура: Розробка системи за принципом модульної архітектури для полегшення розширення та тестування окремих компонентів.
- Використання UML: Використання діаграм UML (Unified Modeling Language) для детального моделювання структури та взаємодії компонентів [9].

2. Технології та Засоби Розробки:

- WCF (Windows Communication Foundation): Використання WCF для реалізації мережевого взаємодії та обміну повідомленнями між клієнтами та сервером [6].
- WPF (Windows Presentation Foundation): Використання WPF для створення сучасного графічного інтерфейсу користувача (GUI) з підтримкою шаблонів, стилів та двостороннього зв’язку даних (data binding). Забезпечує зручну реалізацію інтуїтивного UI для десктоп-додатків на платформі Windows [12].
- C# (.NET Framework / .NET Core / .NET 5+): Основна мова програмування проєкту, що використовується для розробки бізнес-логіки, управління подіями, обробки даних та реалізації функціональності програми. Платформа .NET забезпечує підтримку об’єктно-орієнтованого програмування, високий рівень безпеки та продуктивності [4].

2.2 Проектування відношень між акторами і прецедентами

У системі можна виокремити наступні ролі:

- Користувач який має можливість створювати обліковий запис, увійти в систему, встановлювати статус доступності, та відправляти текстові повідомлення іншим користувачам. Він може створювати та управляти груповими чатами, встановлювати фотографії та редагувати профіль.
- Адміністратор який відповідає за управління користувачами, має можливість додавати та видаляти їх з системи, а також керувати правами та ролями. Він моніторить активність користувачів, вирішує технічні питання та вживає заходів для забезпечення безпеки та ефективності системи.

Модератор який відповідає за управління учасниками групового чату, може додавати та видаляти учасників, а також встановлювати права доступу. Він виконує модерацію контенту, моніторить та вилучає порушливий зміст та сприяє активній взаємодії учасників. Нижче показана реалізація варіантів використання.

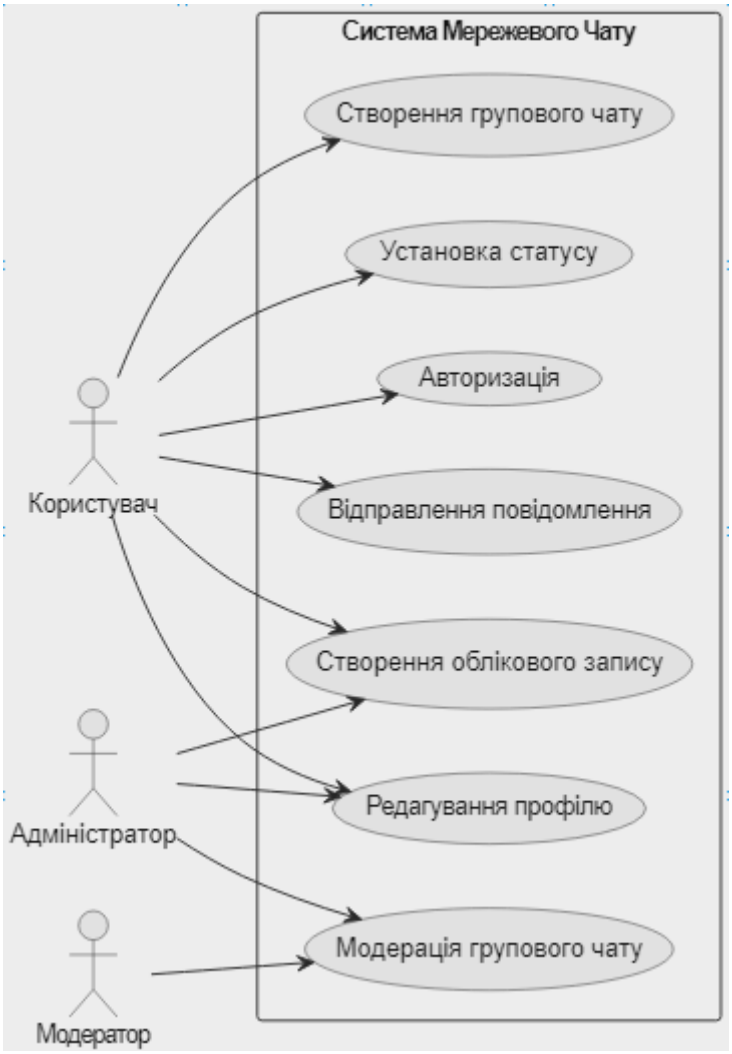

Рисунок 2.1 – Діаграма варіантів використання

2.3 Виявлення класів сутностей

Проаналізувавши основні іменники та опис розроблюваної системи, можемо виділити наступні сутності системи.

Сутність User – ця сутність визначає основного учасника системи мережевого чату. Користувач є ключовим в контексті взаємодії з системою, оскільки від його імені виконуються основні функції. Його обліковий запис служить для ідентифікації та авторизації в системі. Користувач може відправляти повідомлення іншим користувачам, створювати та приєднуватися до групових

чатів, встановлювати статус доступності та редагувати свій профіль. Ця сутність є основною у варіантах використання, які відображають способи взаємодії користувача з системою мережевого чату.

Сутність Message – сутність, що представляє окреме повідомлення в чаті. Зберігає інформацію про текст повідомлення, час відправлення та відправника.

Сутність Administrator – ця сутність визначає адміністратора системи мережевого чату. Адміністратор володіє спеціальними правами управління та контролю над системою, включаючи додавання та видалення користувачів, вирішення технічних питань і контроль за безпекою.

2.4 Опис роботи системи

Діаграми послідовностей (sequence diagrams) — це один із типів діаграм UML (Unified Modeling Language), який використовується для моделювання динаміки системи. Вони показують, як об'єкти взаємодіють між собою у певному сценарії, передаючи повідомлення у визначеному часовому порядку [9].

Користувач відкриває інтерфейс чату, взаємодіє з ним та обирає чат. Система відображає інтерфейс групового чату. Користувач надсилає повідомлення через інтерфейс. Система створює повідомлення та повідомляє користувача про успішну відправку.

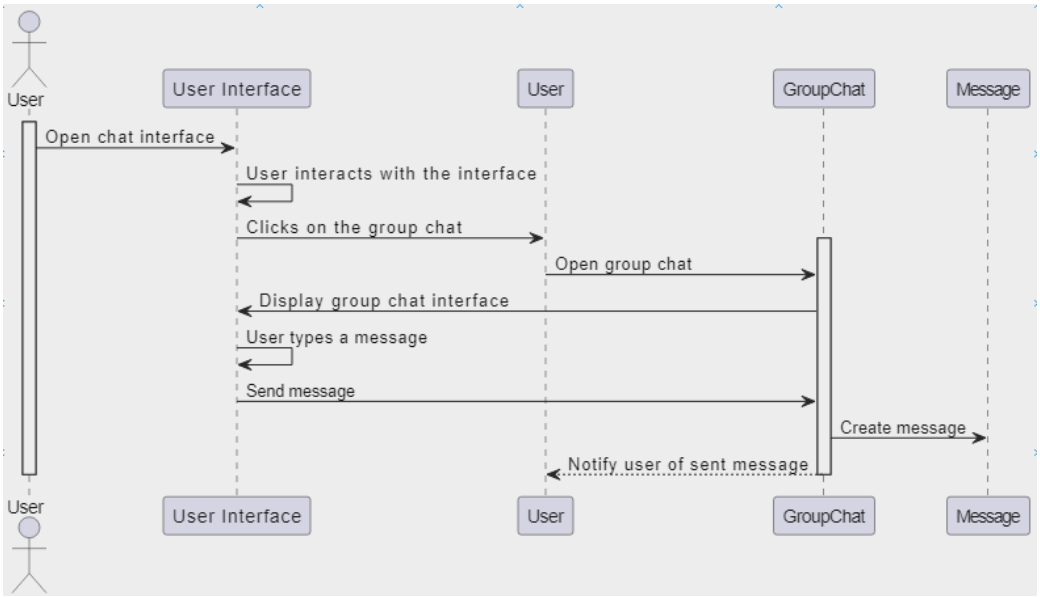

Рисунок 2.2 – Діаграма послідовності

- Користувач взаємодіє з інтерфейсом, обирає груповий чат.
- Система відображає інтерфейс групового чату.
- Користувач взаємодіє з інтерфейсом, надсилає повідомлення.
- Система створює повідомлення Ця діаграма послідовності відображає взаємодію між користувачем та системою під час надсилання повідомлення в груповому чаті.

2.5 Конструювання та опис класів

1.Клас ServerUser, який використовується на стороні сервера в мережевому чаті, побудованому з використанням WCF (Windows Communication Foundation). Кожен екземпляр цього класу представляє одного користувача, який підключився до чату. У нього є три властивості: ID — унікальний ідентифікатор користувача, Name — ім’я або нікнейм користувача, і operationContext — об’єкт, що містить інформацію про поточне з'єднання клієнта з сервером. Цей контекст дозволяє серверу підтримувати зворотній зв’язок із клієнтом, наприклад, надсилати йому

повідомлення. Така структура необхідна для зберігання інформації про активних користувачів і організації обміну повідомленнями між ними в реальному часі.

2.Клас Service1, який реалізує інтерфейс визначений раніше. Усередині класу є список users, який зберігає всі активні підключення у вигляді об'єктів ServerUser, а також змінна nextid, яка відповідає за автоматичне призначення унікального ID кожному новому користувачу.

Метод Connect викликається при підключенні нового користувача. Він створює об’єкт ServerUser, додає його до списку, відправляє повідомлення про приєднання іншим користувачам і повертає клієнту його ID.

Метод Disconnect видаляє користувача з чату за ID та розсилає повідомлення всім про те, що цей користувач покинув чат.

Метод SendMsg відповідає за надсилання повідомлень. Він перебирає всіх користувачів і відправляє їм повідомлення з іменем користувача (який відправив повідомлення) та поточним часом. Повідомлення надсилається через callback-метод MsgCallback, що дозволяє серверу ініціювати передачу даних до клієнтів.

Ця структура забезпечує повноцінне спілкування в чаті між клієнтами в режимі реального часу через єдиний сервер.

3.Клас Program містить точку входу в серверну частину програми чату на базі WCF. Метод Main запускає сервер, який обслуговує підключення клієнтів.

Усередині методу створюється об’єкт ServiceHost, який відповідає за хостинг WCF-сервісу. У цьому випадку в якості сервісу виступає клас Wcf_Chat1.Service1, що реалізує логіку чату. За допомогою методу host.Open() сервіс запускається і стає доступним для клієнтів. Після цього на консоль виводиться повідомлення "Хост розпочав роботу", що означає, що сервер готовий до обслуговування запитів.

Конструкція using гарантує, що ресурси хоста будуть правильно звільнені після завершення роботи (тобто коли користувач натисне Enter). Метод Console.ReadLine() просто утримує програму відкритою, щоб сервер не завершив роботу одразу після запуску.

Цей клас, по суті, є запускним модулем серверної частини чату — він активує та підтримує роботу WCF-сервісу.

РОЗДІЛ 3. РОЗРОБКА ЗАСТОСУНКУ

3.1 Технології для розробки застосунку

У процесі розробки програмного забезпечення для оренди автомобілів було обрано ряд сучасних технологій Microsoft, які забезпечують надійність, масштабованість та зручність у користуванні. Основними технологіями, що використовуються у цьому проєкті, є .NET, Windows Communication Foundation (WCF), Windows Presentation Foundation (WPF) та мова програмування C#. Нижче подано опис кожної технології та обґрунтування її вибору [10].

.NET — це сучасна платформа для розробки різноманітних типів застосунків, яка надає широкий набір інструментів для створення високопродуктивних, безпечних та кросплатформенних рішень. Обрана платформа дозволила ефективно реалізувати як логіку взаємодії з клієнтами, так і зручний інтерфейс користувача.

WCF (Windows Communication Foundation) використовується для реалізації мережевої взаємодії між клієнтом та сервером. Завдяки WCF була забезпечена підтримка дуплексного обміну повідомленнями, що дозволяє обробляти запити в режимі реального часу. Ця технологія дозволила побудувати надійну систему комунікації між елементами застосунку, включаючи реєстрацію, обробку запитів та оновлення даних.

WPF (Windows Presentation Foundation) використовується для створення сучасного та інтерактивного графічного інтерфейсу користувача. Завдяки WPF вдалося реалізувати зручну навігацію, стильне візуальне оформлення, а також адаптивність під різні роздільні здатності екранів. Перевагою WPF є підтримка шаблонів, двостороннього зв'язку з даними (data binding) та гнучке управління UI-компонентами [8].

C# є основною мовою програмування проєкту. Вона використовується як для реалізації логіки на сервері, так і для управління подіями та елементами інтерфейсу на клієнтській стороні. C# забезпечує високу продуктивність, чітку структуру коду,

підтримку об’єктно-орієнтованого підходу та зручність у підтримці та масштабуванні програмного рішення.

Загалом, обрані технології — .NET, WCF, WPF та C# — утворюють надійну технічну основу для створення функціонального, продуктивного та зручного застосунку для оренди автомобілів. Вони дозволяють забезпечити якісний досвід користувача, стабільну роботу системи та можливість подальшого розвитку продукту.

3.2 Середовище розробки

Під час розробки програмного забезпечення для сервісу обміну текстовими повідомленнями було обрано середовище розробки Visual Studio 2022. Це потужне та сучасне інтегроване середовище (IDE), яке забезпечує повноцінну підтримку всіх технологій, використаних у проєкті — зокрема .NET, WCF, WPF та мови програмування C#.

Вибір Visual Studio 2022 був зумовлений її широкими функціональними можливостями, зручністю використання та стабільністю. Середовище дозволяє легко створювати багатокомпонентні застосунки з чіткою архітектурою, підтримує дуплексну взаємодію, що є критично важливою для реалізації чату в режимі реального часу. Завдяки інтелектуальним підказкам коду (IntelliSense), зручному дизайну інтерфейсів у WPF, інтегрованому налагодженню та розгорнутим інструментам аналізу коду, розробка стала максимально ефективною.

Visual Studio 2022 також забезпечила швидкий старт, легке тестування та якісне налагодження логіки передачі повідомлень між клієнтом і сервером. Це середовище оптимально підходить для створення сучасних настільних застосунків із підтримкою мережевого обміну повідомленнями.

Основні переваги використання Visual Studio 2022 для розробки сервісу обміну текстовими повідомленнями:

- Повна інтеграція з .NET, WCF та WPF: середовище має вбудовану підтримку усіх необхідних технологій, що дозволяє розробляти клієнт-серверні застосунки швидко та зручно.
- Потужне налагодження та тестування: Visual Studio 2022 забезпечує гнучке й точне налагодження WCF-сервісів, що дає змогу легко виявляти та виправляти помилки в логіці обміну повідомленнями.
- Зручний дизайнер інтерфейсів WPF: надає візуальні засоби для створення інтуїтивно зрозумілого інтерфейсу користувача, що важливо для чату.
- Інтелектуальні підказки (IntelliSense): допомагають швидше писати код, зменшуючи кількість синтаксичних помилок і пришвидшуючи розробку.
- Велика кількість шаблонів і розширень: дозволяє швидко запускати проєкти, додавати нові компоненти, використовувати сторонні бібліотеки та інтегрувати інструменти аналізу.
- Підтримка сучасних стандартів та оновлень .NET: Visual Studio 2022 постійно оновлюється, підтримує нові версії .NET, що забезпечує актуальність та довговічність рішення.
- Зручна інтеграція з системами контролю версій: підтримка Git, GitHub, Azure DevOps дозволяє керувати змінами в коді, працювати в команді та автоматизувати розгортання.

3.3 Реалізація всіх компонентів застосунку

3.3.1 Створення інтрефейсу для WCF сервісу

Основою для створення реального часу чату, де кожен клієнт може приєднуватись, надсилати повідомлення та отримувати їх від інших користувачів через сервер.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace Wcf_Chat1
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to chang
    [ServiceContract(CallbackContract = typeof(IService1Callback))]
    Ссылок: 1
    public interface IService1
    {
        [OperationContract]
        Ссылок: 1
        int Connect(string name);


        [OperationContract]
        Ссылок: 1
        void Disconnect(int id);


        [OperationContract(IsOneWay = true)]
        Ссылок: 3
        void SendMsg(string msg,int id);


    }

    Ссылок: 2
    public interface IService1Callback
    {
        [OperationContract(IsOneWay = true)]
        Ссылок: 1
        void MsgCallback(string msg);

    }


}
```

Рисунок 3.1 – Код серверної частини

Цей фрагмент коду є частиною серверної частини застосунку для обміну текстовими повідомленнями, реалізованої за допомогою технології WCF (Windows Communication Foundation). Він визначає контракти сервісу — тобто інтерфейси, які описують, які методи може викликати клієнт і як сервер буде відповідати.

Ось що виконує кожна частина коду:

Інтерфейс IService1 — це головний сервісний контракт, що описує функціональність, доступну клієнтам:

Connect(string name) — метод, який викликається при підключенні користувача до чату. Приймає ім’я користувача й повертає унікальний ID.

Disconnect(int id) — метод для відключення користувача від чату за його ID.

SendMsg(string msg, int id) — надсилає повідомлення від клієнта до сервера.

Атрибут [ServiceContract(CallbackContract = typeof(IService1Callback))] вказує, що цей сервіс підтримує дуплексне з’єднання, тобто може не лише приймати виклики від клієнта, а й сам ініціювати виклики назад до клієнта.

Інтерфейс IService1Callback — це зворотній (callback) контракт:

MsgCallback(string msg) — метод, який сервер використовує для надсилання повідомлень усім підключеним клієнтам.

ServerUser — це модель, яка зберігає ідентифікаційні та технічні дані про підключеного користувача, щоб сервер міг коректно надсилати йому повідомлення у рамках дуплексного з’єднання.

```
using System.ServiceModel;

namespace Wcf_Chat1
{
    internal class ServerUser
    {


        public int ID { get; set; }


        public string Name { get; set; }


        public OperationContext operationContext { get; set; }
    }
}
```

Рисунок 3.2 – Код класу серверної частини

Цей клас ServerUser призначений для зберігання інформації про кожного користувача, який підключився до сервера чату.

Ось що виконує кожен елемент класу:

public int ID { get; set; } — унікальний ідентифікатор користувача, який використовується для розпізнавання кожного клієнта у чаті.

public string Name { get; set; } — ім’я користувача, яке буде відображатися під час обміну повідомленнями.

public OperationContext operationContext { get; set; } — контекст поточного підключення користувача до WCF-сервісу. Завдяки цьому об’єкту сервер може ініціювати зворотній виклик (callback) і надіслати повідомлення безпосередньо цьому клієнту.

Клас під назвою ServerUser є допоміжною частиною серверної частини застосунку для обміну текстовими повідомленнями, який реалізовано з використанням технології WCF (Windows Communication Foundation). Його основне призначення — зберігати інформацію про кожного користувача, який приєднується до чату.

У класі визначено три властивості:

ID — це унікальний числовий ідентифікатор, який присвоюється кожному користувачеві при підключенні. Завдяки йому сервер може точно ідентифікувати користувача серед інших підключених клієнтів.

Name — це ім’я користувача або його нікнейм, який буде використовуватись при надсиланні повідомлень у чаті, щоб інші користувачі знали, від кого надійшло повідомлення.

operationContext — це спеціальний об’єкт, який містить технічну інформацію про з’єднання між клієнтом і сервером. Через нього сервер має можливість надсилати дані назад клієнту, тобто реалізовувати функцію зворотного виклику (callback), яка є ключовою для роботи чату в режимі реального часу.

Таким чином, клас ServerUser дозволяє серверу відстежувати, хто підключений до чату, як з ним зв’язатися, та використовувати його дані під час обміну повідомленнями. Це все подано на рисунку 3.3.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace Wcf_Chat1
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to chang
    [ServiceContract(CallbackContract = typeof(IService1Callback))]
    Ссылок: 1
    public interface IService1
    {
        [OperationContract]
        Ссылок: 1
        int Connect(string name);


        [OperationContract]
        Ссылок: 1
        void Disconnect(int id);


        [OperationContract(IsOneWay = true)]
        Ссылок: 3
        void SendMsg(string msg,int id);


    }

    Ссылок: 2
    public interface IService1Callback
    {
        [OperationContract(IsOneWay = true)]
        Ссылок: 1
        void MsgCallback(string msg);

    }


}
```

Рисунок 3.3 – Код серверної частини

3.3.2 – Створення хосту

Код відповідає за запуск і підтримку роботи серверної частини чату, до якої підключаються клієнтські додатки.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ServiceModel;

namespace ChatHost25
{
    internal class Program
    {
        static void Main(string[] args)
        {
            using (var host = new ServiceHost(typeof(Wcf_Chat1.Service1)))
            {
                host.Open();
                Console.WriteLine("Хост розпочав роботу");
                Console.ReadLine();
            }
        }
    }
}
```

Рисунок 3.3 – Код роботи хосту

У цьому коді створено хост (сервер) для чат-сервісу, написаного з використанням технології WCF (Windows Communication Foundation). Це консольна програма, яка запускає сервер, щоб клієнти могли підключатися до нього і обмінюватися повідомленнями.

Всередині методу Main створюється об’єкт ServiceHost, який запускає сервіс Service1, розташований у просторі імен Wcf_Chat1. Саме в цьому класі реалізована логіка чату (наприклад, надсилання та прийом повідомлень).

Далі викликається метод host.Open(), який відкриває сервіс і дозволяє клієнтам підключатися. Після цього виводиться повідомлення "Хост розпочав роботу", що означає успішний запуск сервера.

Програма зупиняється і чекає на натискання Enter, щоб не завершити роботу одразу. Після завершення роботи ServiceHost автоматично закриється завдяки конструкції using.

3.3.3 – Розробка інтерфейсу

XAML (eXtensible Application Markup Language) — це декларативна мова розмітки, яка використовується в середовищі .NET, зокрема у WPF (Windows Presentation Foundation), UWP і Xamarin, для опису інтерфейсу користувача.

XAML — це як HTML для десктопних або мобільних додатків на C#. Він дозволяє описувати зовнішній вигляд вікна, кнопок, текстів, полів вводу та інших елементів інтерфейсу у зручному текстовому форматі. У той час як логіка програми (обробка подій, взаємодія з сервером тощо) пишеться на C#.

Де використовується XAML:

- WPF — для створення сучасних десктопних додатків під Windows.
- UWP/WinUI — для універсальних додатків Windows.
- Xamarin.Forms — для кросплатформових мобільних застосунків.
- MAUI (новіша платформа) — теж використовує XAML.

XAML — це потужний інструмент для створення інтерфейсів у .NET-застосунках. Він дозволяє розділити вигляд і логіку, швидко змінювати стиль інтерфейсу і створювати гнучкий, адаптивний дизайн з мінімумом коду.

Мій код описує вікно графічного інтерфейсу користувача для застосунку чату з назвою MaxChat, створеного у середовищі WPF. Усі елементи інтерфейсу організовані та стилізовані вручну без використання стандартної рамки вікна Windows. Нижче я подав опис основних частин вікна та їх функціонального призначення:

1. Головне вікно має фіксовану початкову висоту та ширину, може змінювати розміри, і не має стандартної рамки Windows (тобто воно без кнопок закриття, мінімізації, максимізації за замовчуванням). Фон вікна задається динамічно і підтримує перемикання тем (світла/темна). Початково використовується темна тема з темно-сірим кольором.

```
<Window x:Class="ChatClient.MainWindow"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ChatClient"
        mc:Ignorable="d"
        Title="MaxChat" Height="452" Width="822"
        ResizeMode="CanResize" WindowStyle="None" AllowsTransparency="True"
        Background="{DynamicResource WindowBackground}">

    <Window.Resources>
        <!-- Темна тема -->
        <SolidColorBrush x:Key="DarkBackground" Color="#FF2C2C2C"/>
        <!-- Світла тема -->
        <SolidColorBrush x:Key="LightBackground" Color="Gray"/>
        <!-- Початково: темна -->
        <SolidColorBrush x:Key="WindowBackground" Color="#FF2C2C2C"/>
    </Window.Resources>
```

Рисунок 3.4 – Код роботи початкового вікна та зміни теми

Для коректної роботи зміни з початкової теми на світлу та темну тему я написав логіку і вона зображена на рисунку 3.5.

```
Ссылок: 1
private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Left)
        this.DragMove();
}

private bool isDark = true;

Ссылок: 1
private void ThemeToggle_Click(object sender, RoutedEventArgs e)
{
    var newColor = isDark ? FindResource("LightBackground") : FindResource("DarkBackground");
    Resources["WindowBackground"] = newColor;
    isDark = !isDark;
}
```

Рисунок 3.5 – Логіка для роботи світлої та темної теми

2. Верхня панель (заголовок) У верхній частині інтерфейсу знаходиться власноруч оформлена панель заголовка висотою 40 пікселів. Вона напівпрозора та реагує на подію перетягування мишкою, щоб переміщувати вікно. На цій панелі розташовано: Кнопка перемикання теми з іконкою у вигляді півмісяця/сонця, яка змінює фонову тему застосунку.

Три кнопки керування вікном:

"—" — згортає вікно.

"▢" — розгортає або відновлює розмір вікна.

"✕" — закриває застосунок.

Кожна кнопка має заокруглену форму, власний колір фону і змінює вигляд відповідно до своєї функції.

```
<!-- Максимізація/відновлення -->
<Button Content="▢" Width="30" Height="30" Margin="2" Click="MaximizeRestore_Click"
        Background="#AA61FBAD" Foreground="White" BorderThickness="0" FontWeight="Bold" Cursor="Hand">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <Border Background="{TemplateBinding Background}" CornerRadius="20">
                <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
            </Border>
        </ControlTemplate>
    </Button.Template>
</Button>

<!-- Закриття -->
<Button Content="✕" Width="30" Height="30" Margin="2" Click="Close_Click"
        Background="#AAFF4C4C" Foreground="White" BorderThickness="0" FontWeight="Bold" Cursor="Hand">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <Border Background="{TemplateBinding Background}" CornerRadius="20">
                <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
            </Border>
        </ControlTemplate>
    </Button.Template>
```

Рисунок 3.6 – Код роботи кнопок на панелі

Також для коректної роботи кнопок на панелі я написав саму логіку їх роботи. Її можна побачити на рисунку 3.7.

```
Ссылок: 1
private void Close_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}

Ссылок: 1
private void Minimize_Click(object sender, RoutedEventArgs e)
{
    this.WindowState = WindowState.Minimized;
}

Ссылок: 1
private void MaximizeRestore_Click(object sender, RoutedEventArgs e)
{
    this.WindowState = this.WindowState == WindowState.Maximized
        ? WindowState.Normal
        : WindowState.Maximized;
}
```

Рисунок 3.7 – Логіка роботи кнопок

3. Основний вміст вікна:

Під верхньою панеллю розміщено головний блок з напівпрозорим фоном і заокругленими кутами, в якому реалізовані всі елементи інтерфейсу для користувача.

У верхній частині цього блоку знаходиться текстове поле з написом #LocalChat:). Воно неактивне, лише виводить назву або статус.

Поле для введення нікнейму

Нижче знаходиться блок для введення нікнейму. Є підпис "Enter nickname", який плаває (анімація зміни позиції під час фокусу).

Нижче — саме поле введення, в якому користувач вводить свій нік. Воно реагує на набір тексту та на події входу й виходу з фокусу.

```
        </TextBlock>

        <!-- Поле вводу -->
        <TextBox x:Name="tbUsername"
Margin="10,0,10,23"
Background="Transparent"
BorderThickness="0"
FontSize="14"
Foreground="White"
VerticalContentAlignment="Bottom"
TextChanged="tbUsername_TextChanged"
GotFocus="tbUsername_GotFocus"
LostFocus="tbUsername_LostFocus"/>
    </Grid>
</Border>


<!-- Кнопка підключення -->
<Button x:Name="bConnDicon" Grid.Row="1" Grid.Column="1"
        Content="Connect" Click="Button_Click"
        Height="38" Margin="10,0,0,10" VerticalAlignment="Top"
        Background="#AA61FBAD" Foreground="Black" FontWeight="Bold"
        BorderThickness="0" Cursor="Hand">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <Border Background="{TemplateBinding Background}" CornerRadius="10">
                <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
            </Border>
```

Рисунок 3.8 – Код XAML роботи основного наповнення вікна

4. Кнопка підключення

Справа від поля нікнейму знаходиться кнопка з написом "Connect". Вона натискається для підключення до сервера чату. Має яскраво-зелений колір, округлу форму та натискання оброблюється в коді програми.

```
<!-- Кнопка підключення -->
<Button x:Name="bConnDicon" Grid.Row="1" Grid.Column="1"
        Content="Connect" Click="Button_Click"
        Height="38" Margin="10,0,0,10" VerticalAlignment="Top"
        Background="#AA61FBAD" Foreground="Black" FontWeight="Bold"
        BorderThickness="0" Cursor="Hand">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <Border Background="{TemplateBinding Background}" CornerRadius="10">
                <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
            </Border>
        </ControlTemplate>
    </Button.Template>
</Button>
```

Рисунок 3.9 – Код XAML кнопки підключення

Також для коректної роботи кнопки я написав логіку яку можна побачити на риунку 3.10.

```
Ссылок: 1
void ConnectUser()
{
    if (!isConnected)
    {
        client = new ServiceClient(new System.ServiceModel.InstanceContext(this));
        ID = client.Connect(tbUsername.Text);
        tbUsername.IsEnabled = false;
        bConnDicon.Content = "Disconnect";
        isConnected = true;
    }
}

Ссылок: 2
void DisconnectUser()
{
    if (isConnected)
    {
        client.Disconnect(ID);
        client = null;
        tbUsername.IsEnabled = true;
        bConnDicon.Content = "Connect";
        isConnected = false;
    }
}
```

Рисунок 3.10 – Код C# для кнопки підключення

Також додав список повідомлень який знаходиться у центрі вікна зліва знаходиться список повідомлень чату (ListBox). Він має прозорий фон, білий текст і дозволяє переглядати всі вхідні/вихідні повідомлення. При виборі повідомлення може викликатися певна логіка.

Поле для введення повідомлення яке знаходиться у нижній частині вікна, по всій ширині, знаходиться велике текстове поле для написання повідомлень. Воно підтримує перенос рядків і реагує на натискання клавіші Enter — найімовірніше, щоб відправити повідомлення. І ось фінальний результат робочого інтерфейсу з правильною логікою та функціоналом. Це можна побачити на рисунку 3.11.

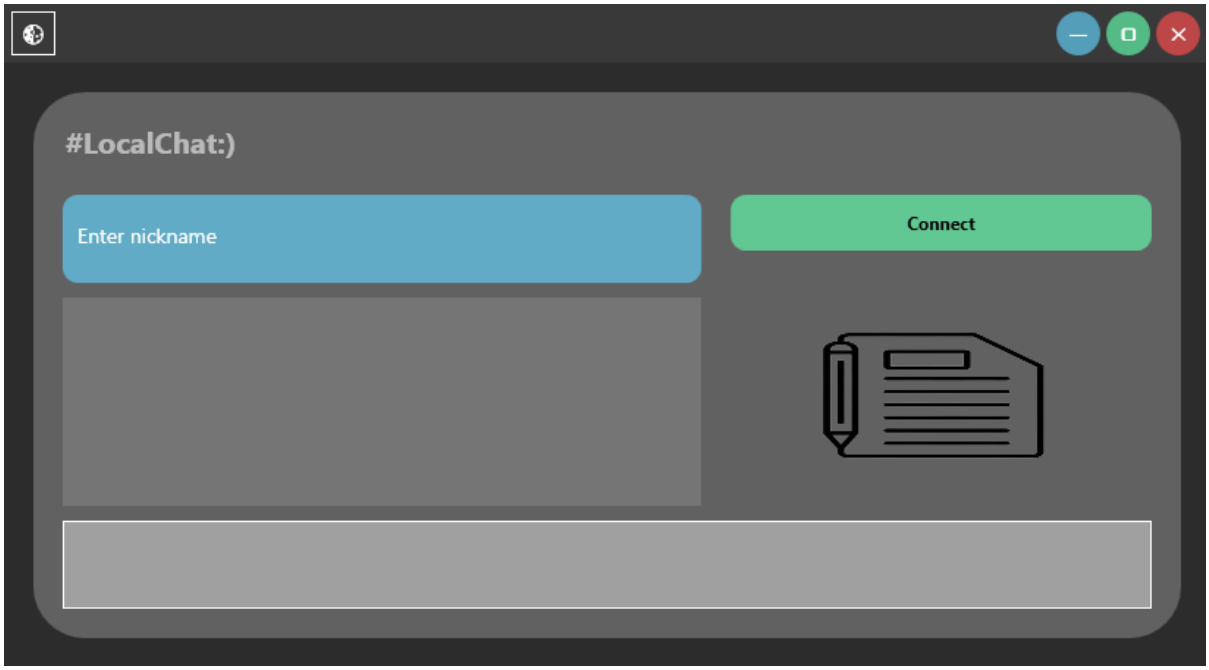

Рисунок 3.11 – Інтерфейс

3.4 Тестування проекту

Тестування є одною складовою процесу створення доволі якісного програмного забезпечення. Його головна мета це гарантувати, що розроблена

система відповідає очікуванням користувачів та вимогам. Це досягається шляхом перевірки функціональності, сумісності, надійності та продуктивності продукту.

Під час тестування здійснюється аналіз взаємодії окремих модулів, перевіряється їхня коректна інтеграція, а також фіксуються відхилення від заданих характеристик. Особливу увагу приділяють безпеці системи, стійкості до помилок і зручності користування.

інтерфейсні рішення представляє собою інтуїтивність, тобто, користувач має легко орієнтуватися у функціоналі та взаємодіяти з системою боз додаткових труднощів.

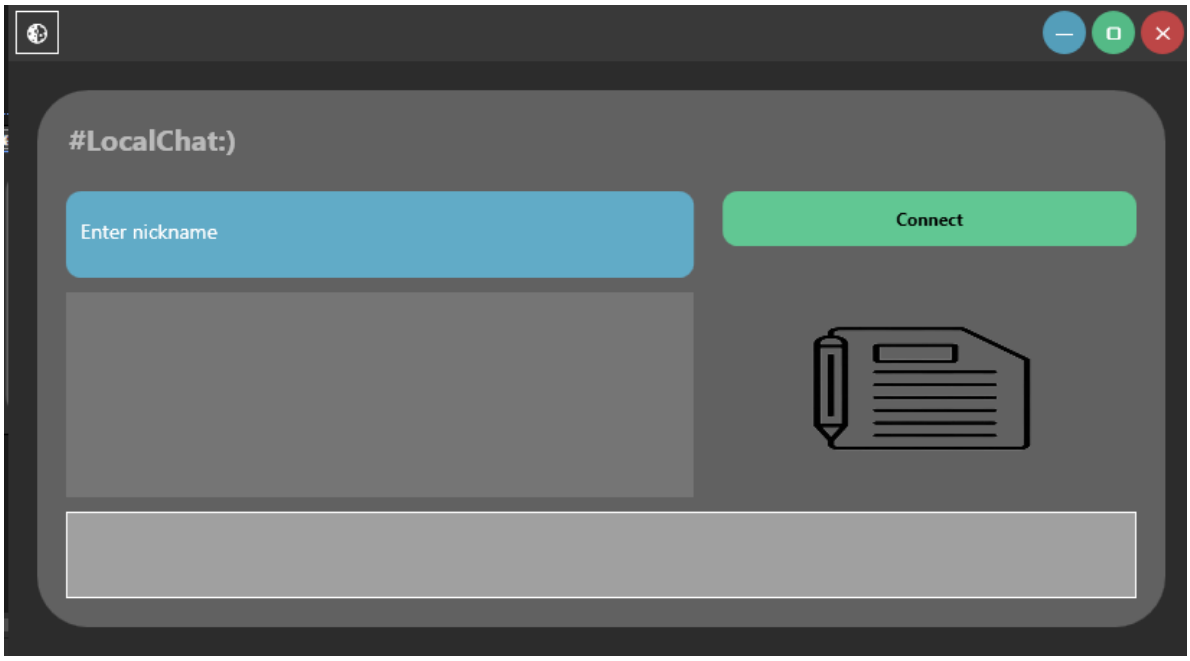

Рисунок 3.11 – Головна сторінка

При запуску програми нас вітає головна сторінка. Якщо користувач не зареєстрований, потрібно ввести свій нікнейм і натиснути кнопку Connect.

Далі я запустив декілька вікон для перевірки працездатності хосту та підключення до хосту. Підключив в систему 4 користувачів та протестував обмін текстовими повідомленнями тим самим перевірив працездатність. Далі провів тест

всіх підключених кнопок та перевірив проект на баги. Успішне тестування можете бачити на рисунках 3.12 та 3.13.

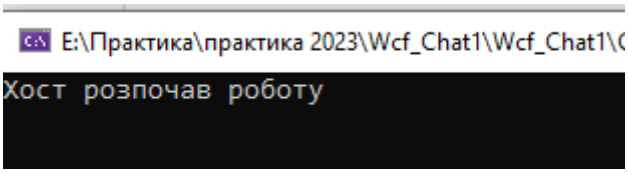

Рисунок 3.12 – Успішна та стабільна робота хосту

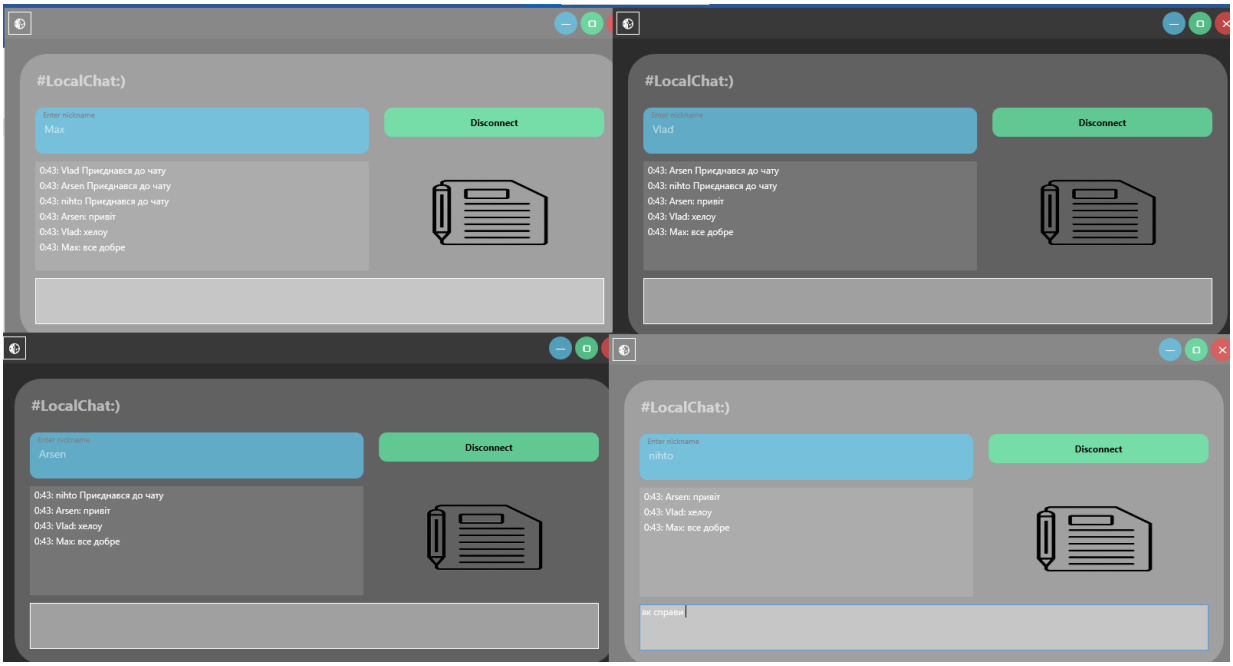

Рисунок 3.13 – Успішний тест всіх функцій на декількох акаунтах

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Менеджмент безпеки.

Під час розробки дипломної роботи на тему «Сервіс з обміну текстовими повідомленнями з використанням технології .NET», питання менеджменту безпеки є критично важливим, оскільки я маю справу з передачею та обробкою потенційно конфіденційної інформації користувачів.

Менеджмент безпеки праці — це система управління, яка охоплює планування, організацію, контроль та вдосконалення процесів, спрямованих на забезпечення безпечних і здорових умов праці. У контексті сучасних підприємств, включаючи сферу IT, цей менеджмент базується на міжнародних стандартах, таких як ISO 45001, та враховує специфіку цифрового середовища.

Дисципліна БЖД, ООП - якісно змінює концепцію та методологію підготовки спеціалістів з важливої проблеми сучасного суспільства - захисту життя та здоров’я людини в побутових умовах, у природному та виробничому середовищах, в умовах надзвичайних ситуацій. Забезпечення безпеки життєдіяльності людини завжди було одним з найважливіших завдань розвитку цивілізації людства [14].

Навчання та інструктажі з охорони праці повинні бути нерозривно зв’язані з пропагандою безпечних методів праці.

В цілях пропаганди охорони праці на підприємствах, відділами, службами охорони праці використовується: радіотрансляційний зв'язок; багатотиражні газети, плакати, різни написи з техніки охорони праці, знаки безпеки охорони праці, а також спеціально обладнані навчальні кабінети з охорони праці, кутки та вітрини в цехах до тематики гігієни і безпеки праці, проведення круглих столів.

Пропаганда охорони праці допомагає швидкому впровадженню у виробництво сучасних засобів техніки безпеки, підвищенню культури виробництва

на всіх участках, створення на кожному робочому місцю умов для високопродуктивної та безпечної праці [15].

Правильна організація праці, навчання та інструктажів працюючих безпечним методам праці – важлива умова зниження травматизму і професійним захворюванням на виробництві.

Особливо важливу роль у пропаганді охорони праці відіграє кабінет з охорони праці. Він створюється у відповідності до „Типового положення про кабінет охорони праці". Кабінет з охорони праці створюється на підприємстві з кількістю працюючих 400 і більше осіб. Коли осіб до 400, то кабінет може бути суміщеним з приміщенням для навчальних занять. Для кабінету виділяється певна площа, кімната, в залежності від кількості працюючих [15].

Одне з загальних положень Закону України Про національну безпеку України говорить: громадська безпека і порядок - захищеність життєво важливих для суспільства та особи інтересів, прав і свобод людини і громадянина, забезпечення яких є пріоритетним завданням діяльності сил безпеки, інших державних органів, органів місцевого самоврядування, їх посадових осіб та громадськості, які здійснюють узгоджені заходи щодо реалізації і захисту національних інтересів від впливу загроз [16].

4.2 Заходи щодо евакуації людей із робочих приміщень.

Евакуація працівників у разі надзвичайної ситуації є важливою складовою системи забезпечення безпеки на підприємстві. Від правильно організованих дій залежить не лише збереження матеріальних цінностей, а насамперед — життя та здоров'я людей. Завчасне планування, інформування персоналу, наявність чітких маршрутів евакуації та проведення регулярних навчань сприяють зниженню ризиків під час надзвичайних подій.

Порядок проведення евакуації у разі загрози виникнення або виникнення надзвичайних ситуацій. Визначає механізм здійснення організованого вивезення (виведення) населення із зон можливого впливу наслідків надзвичайної ситуації або надзвичайної ситуації і розміщення його поза зонами дії вражаючих факторів джерел надзвичайної ситуації у разі виникнення безпосередньої загрози життю та заподіяння шкоди здоров'ю населення, а також заходів з евакуації матеріальних і культурних цінностей, якщо виникає загроза їх пошкодження або знищення.

Обов'язковій евакуації підлягає населення у разі виникнення загрози аварії з викидом радіоактивних і небезпечних хімічних речовин, катастрофічного затоплення місцевості та землетрусів, масових лісових і торф'яних пожеж, зсувів, інших геологічних та гідрогеологічних явищ і процесів, надзвичайних ситуацій на арсеналах, базах (складах) озброєння, ракет, боєприпасів і компонентів ракетного палива, інших вибухопожежо-небезпечних об'єктах Збройних Сил та/або інших військових формувань, утворених відповідно до законів України, а також суб'єктів господарювання, інших юридичних осіб, які використовують у своїй діяльності вибухопожежонебезпечні об'єкти, збройних конфліктів (із районів можливих бойових дій у безпечні райони) [17].

З метою захисту дітей, які перебувають у зоні воєнних дій і збройних конфліктів, в умовах воєнного стану обласні військові адміністрації за погодженням з органами військового командування на відповідній території та Координаційним штабом з проведення евакуаційних заходів та ефективного реагування на масове переміщення населення, утвореним Кабінетом Міністрів України, можуть прийняти рішення про проведення обов'язкової евакуації в примусовий спосіб дітей з їх батьками, особами, які їх замінюють, або іншими законними представниками з окремого населеного пункту в місцевості, де ведуться бойові дії [17].

Евакуація – це організоване виведення чи вивезення з небезпечних зон. Безпосередньо евакуацією займається штаб цивільної оборони, усі організаційні питання вирішують евакуаційні комісії. Евакуація розпочинається після прийняття

рішення начальником цивільної оборони, надзвичайною комісією або органами влади.

Евакуація працюючого населення здійснюється за виробничим принципом, а населення, яке не пов’язане з виробництвом, – за територіальним принципом через домоуправління, ЖЕУ, ЖЕК тощо. Діти евакуюються разом з батьками, але можливе їх вивезення зі школами, дитсадками [18].

Для проведення евакуації використовуються всі види транспорту: залізничний, автомобільний, водний та індивідуальний. Автотранспорт використовується для вивезення на короткі відстані. У деяких випадках частина населення може виводитися пішки колонами по шляхах, котрі не зайняті перевезеннями, або за визначеним маршрутом та колонними шляхами.

Евакуація населення здійснюється через збірні евакуаційні пункти, які розташовують поблизу місць посадки на транспорт або на вихідних пунктах пішого руху, в школах, клубах, кінотеатрах та інших громадських закладах [18].

Про початок та порядок евакуації населення сповіщається по мережі сповіщення. Отримавши повідомлення про початок евакуації, необхідно взяти документ, гроші, речі та продукти і у визначений час прибути на збірний евакуаційних пункт, де населення реєструють, групують та проводять до пункту посадки.

ВИСНОВКИ

В ході виконання дипломної роботи було спроектовано програмний продукт на тему «Розробка сервісу обміну текстовими повідомленнями з використанням технології .NET». Основна мета полягала у створенні зручного, зрозумілого та надійного інструменту для локальної комунікації між користувачами. Програма спроектована таким чином, щоб максимально полегшити процес взаємодії користувача з системою, забезпечуючи при цьому належний рівень функціональності, гнучкості та стабільності.

Під час розробки особливу увагу було приділено зовнішньому вигляду та логіці взаємодії користувача з інтерфейсом. Вікно програми оформлено у сучасному стилі з використанням XAML-розмітки, передбачено перемикання між темною та світлою темами, реалізовано кастомні кнопки управління вікном, зручне поле для введення нікнейму з анімованим підписом, а також чат, що відображає повідомлення в реальному часі. Інтерфейс є візуально привабливим, лаконічним та інтуїтивно зрозумілим навіть для користувачів без технічного досвіду.

Розроблена система демонструє швидку реакцію на дії користувача, швидке завантаження інтерфейсу та коректну роботу при обміні повідомленнями. Забезпечено базову безпеку даних у межах локальної мережі, а сама структура програми дозволяє легко вносити зміни або додавати нові функції без істотного перероблення коду. Одним із важливих завдань було створення такої архітектури, яка дозволила б ефективно керувати всією системою без значних людських затрат, і це завдання було повністю реалізовано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про WhatsApp: https://www.whatsapp.com/about?lang=uk_UA
2. Telegram: що це таке, основні функції: https://gerabot.com/article/telegram_sho_ce_take_osnovni_funkcii
3. Про Viber: https://www.viber.com/ua/about/
4. Офіційна документація Microsoft C#: https://learn.microsoft.com/uk-ua/dotnet/csharp/tour-of-csharp/
5. Pluralsight - C# Fundamentals with Scott Allen: https://www.pluralsight.com/courses/csharp-fundamentals-2
6. Офіційна документація Microsoft WCF: https://www.microsoft.com/en-us/download/details.aspx?id=21459
7. Pluralsight - WPF Fundamentals: https://www.pluralsight.com/courses/wpf-fundamentals
8. Офіційна документація Microsoft WPF: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/
9. Про UML діаграми: https://dou.ua/forums/topic/40575/
10. Офіційна документація .NET learn: https://learn.microsoft.com/en-us/training/dotnet/
11. Microsoft Learn – інтерактивні навчальні курси: https://learn.microsoft.com/en-us/training/
12. WPF Tutorial на WPF-Tutorial.com: https://wpf-tutorial.com/
13. Mastering Windows Presentation Foundation від Sheridan Yuen — сучасний підхід (оновлене до .NET 8+): https://www.bol.com/nl/nl/f/mastering-windows-presentation-foundation/9200000075050568/
14. ТЕМА 1. Актуальність безпеки життєдіяльності людини. Управління безпекою життєдіяльності в Україні. [Електронний ресурс] – Режим доступ до ресурсу: https://dl.tntu.edu.ua/content.php?cid=299140

15. ТЕМА 10. Загальні питання охорони праці. Правові та організаційні основи охорони праці. [Електронний ресурс] – Режим доступ до ресурсу: https://dl.tntu.edu.ua/content.php?cid=289117

16. ЗАКОН УКРАЇНИ Про національну безпеку України. [Електронний ресурс] – Режим доступ до ресурсу: https://zakon.rada.gov.ua/laws/show/2469-19#Text

17. Постанова Кабінету Міністрів України Про затвердження Порядку проведення евакуації у разі загрози виникнення або виникнення надзвичайних ситуацій. [Електронний ресурс] – Режим доступ до ресурсу: https://zakon.rada.gov.ua/laws/show/841-2013-%D0%BF#Text

18. ТЕМА 7. Менеджмент безпеки, правове забезпечення та організаційно-функціонувальна структура захисту населення та адміністративні територіальні одиниці у НС. [Електронний ресурс] – Режим доступ до ресурсу: https://dl.tntu.edu.ua/content.php?cid=299414

ДОДАТКИ

Додаток А – Лістинг програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace Wcf_Chat1
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu
to change the interface name "IService1" in both code and config
file together.
    [ServiceContract(CallbackContract = typeof(IService1Callback))]
    public interface IService1
    {
        [OperationContract]
        int Connect(string name);


        [OperationContract]
        void Disconnect(int id);


        [OperationContract(IsOneWay = true)]
        void SendMsg(string msg,int id);


    }

    public interface IService1Callback
    {
        [OperationContract(IsOneWay = true)]
        void MsgCallback(string msg);

    }



}



using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
```

```
namespace Wcf_Chat1
{
    [ServiceBehavior(InstanceContextMode =
InstanceContextMode.Single)]
    public class Service1 : IService1
    {
        List<ServerUser> users = new List<ServerUser>();
        int nextid = 1;

        public int Connect(string name)
        {
            ServerUser user = new ServerUser() {
                ID = nextid,
                Name = name,
                operationContext = OperationContext.Current

            };
            nextid++;

            SendMsg(": "+user.Name+" Приєднався до чату",0);
            users.Add(user);
            return user.ID;
        }

        public void Disconnect(int id)
        {
            var user = users.FirstOrDefault(i => i.ID == id);
            if (user != null)
            {
                users.Remove(user);
                SendMsg(": "+user.Name + " Залишив чат",0);
            }
        }

        public void SendMsg(string msg, int id)
        {
            foreach (var item in users)
            {
                string answer = DateTime.Now.ToShortTimeString();

                var user = users.FirstOrDefault(i => i.ID == id);
                if (user != null)
                {
                    answer += ": " + user.Name + ": ";
                }

                answer += msg;


item.operationContext.GetCallbackChannel<IService1Callback>().MsgCal
lback(answer);
```

```
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ServiceModel;

namespace ChatHost25
{
    internal class Program
    {
        static void Main(string[] args)
        {
            using (var host = new
ServiceHost(typeof(Wcf_Chat1.Service1)))
            {
                host.Open();
                Console.WriteLine("Хост розпочав роботу");
                Console.ReadLine();
            }
        }
    }
}
```

```
using System.ServiceModel;

namespace Wcf_Chat1
{
    internal class ServerUser
    {


        public int ID { get; set; }

        public string Name { get; set; }

        public OperationContext operationContext { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using ChatClient.ServiceChat;

namespace ChatClient
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window, IService1Callback
    {
        bool isConnected = false;

        Service1Client client;

        int ID;
        public MainWindow()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {

        }

        void ConnectUser()
        {
            if (!isConnected)
            {
                client = new Service1Client(new
System.ServiceModel.InstanceContext(this));
                ID = client.Connect(tbUsername.Text);
                tbUsername.IsEnabled = false;
                bConnDicon.Content = "Disconnect";
                isConnected = true;
            }
        }

        void DisconnectUser()
        {
            if (isConnected)
            {
```

```
                client.Disconnect(ID);
                client = null;
                tbUsername.IsEnabled = true;
                bConnDicon.Content = "Connect";
                isConnected = false;
            }
        }



        private void Button_Click(object sender, RoutedEventArgs e)
        {
            if (isConnected)
            {
                DisconnectUser();
            }
            else
            {
                ConnectUser();
            }
        }

        public void MsgCallback(string msg)
        {
            lbChat.Items.Add(msg);
            lbChat.ScrollIntoView(lbChat.Items[lbChat.Items.Count-
1]);
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void Minimize_Click(object sender, RoutedEventArgs
e)
        {
            this.WindowState = WindowState.Minimized;
        }

        private void MaximizeRestore_Click(object sender,
RoutedEventArgs e)
        {
            this.WindowState = this.WindowState ==
WindowState.Maximized
                ? WindowState.Normal
                : WindowState.Maximized;
        }



        private void tbUsername_GotFocus(object sender,
RoutedEventArgs e)
        {
```

```
            AnimateFloatingLabel(true);
        }

        private void tbUsername_LostFocus(object sender,
RoutedEventArgs e)
        {
            if (string.IsNullOrWhiteSpace(tbUsername.Text))
                AnimateFloatingLabel(false);
        }

        private void tbUsername_TextChanged(object sender,
TextChangedEventArgs e)
        {
            if (!string.IsNullOrWhiteSpace(tbUsername.Text))
                AnimateFloatingLabel(true);
        }

        private void AnimateFloatingLabel(bool up)
        {
            var fontSize = up ? 10.0 : 14.0;
            var margin = up ? new Thickness(10, 2, 0, 0) : new
Thickness(10, 18, 0, 0);

            FloatingLabel.FontSize = fontSize;
            FloatingLabel.Margin = margin;
            FloatingLabel.Foreground = up ? Brushes.Gray :
Brushes.White;
        }




        private void TitleBar_MouseDown(object sender,
MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }

        private bool isDark = true;

        private void ThemeToggle_Click(object sender,
RoutedEventArgs e)
        {
            var newColor = isDark ? FindResource("LightBackground")
: FindResource("DarkBackground");
            Resources["WindowBackground"] = newColor;
            isDark = !isDark;
        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
```

```
        {
            DisconnectUser();
        }

        private void tbMessage_KeyDown(object sender, KeyEventArgs
e)
        {
            if (e.Key == Key.Enter)
            {
                if (client!=null)
                {
                    client.SendMsg(tbMessage.Text, ID);
                    tbMessage.Text = string.Empty;

                }

            }
        }

        private void lbChat_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {

        }



    }
}



<Window x:Class="ChatClient.MainWindow"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:ChatClient"
        mc:Ignorable="d"
        Title="MaxChat" Height="452" Width="822"
        ResizeMode="CanResize" WindowStyle="None"
AllowsTransparency="True"
        Background="{DynamicResource WindowBackground}">

    <Window.Resources>
        <!-- Темна тема -->
        <SolidColorBrush x:Key="DarkBackground" Color="#FF2C2C2C"/>
        <!-- Світла тема -->
        <SolidColorBrush x:Key="LightBackground" Color="Gray"/>
        <!-- Початково: темна -->
```

```
        <SolidColorBrush x:Key="WindowBackground"
Color="#FF2C2C2C"/>
    </Window.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!-- Верхня панель -->
        <Grid Grid.Row="0" Height="40" Background="#10FFFFFF"
MouseDown="TitleBar_MouseDown">
            <!-- Кнопки керування -->
            <StackPanel Orientation="Horizontal"
HorizontalAlignment="Right" VerticalAlignment="Center"
Margin="0,0,5,0">
                <!-- Згортання -->
                <Button Content="—" Width="30" Height="30"
Margin="2" Click="Minimize_Click"
                        Background="#AA61D1FB" Foreground="White"
BorderThickness="0" FontWeight="Bold" Cursor="Hand">
                    <Button.Template>
                        <ControlTemplate TargetType="Button">
                            <Border Background="{TemplateBinding
Background}" CornerRadius="20">
                                <ContentPresenter
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                            </Border>
                        </ControlTemplate>
                    </Button.Template>
                </Button>

                <!-- Максимізація/відновлення -->
                <Button Content="▢" Width="30" Height="30"
Margin="2" Click="MaximizeRestore_Click"
                        Background="#AA61FBAD" Foreground="White"
BorderThickness="0" FontWeight="Bold" Cursor="Hand">
                    <Button.Template>
                        <ControlTemplate TargetType="Button">
                            <Border Background="{TemplateBinding
Background}" CornerRadius="20">
                                <ContentPresenter
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                            </Border>
                        </ControlTemplate>
                    </Button.Template>
                </Button>

                <!-- Закриття -->
                <Button Content="✕" Width="30" Height="30"
Margin="2" Click="Close_Click"
```

```
                        Background="#AAFF4C4C" Foreground="White"
BorderThickness="0" FontWeight="Bold" Cursor="Hand">
                    <Button.Template>
                        <ControlTemplate TargetType="Button">
                            <Border Background="{TemplateBinding
Background}" CornerRadius="20">
                                <ContentPresenter
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                            </Border>
                        </ControlTemplate>
                    </Button.Template>
                </Button>
            </StackPanel>

            <!-- Перемикач теми -->
            <Button Content="◐" Width="30" Height="30" Margin="5"
VerticalAlignment="Center" HorizontalAlignment="Left"
                    Click="ThemeToggle_Click"
Background="Transparent" BorderBrush="White" BorderThickness="1"
Foreground="White"/>
        </Grid>

        <!-- Основний вміст -->
        <Border Grid.Row="1" Background="#40FFFFFF"
CornerRadius="35" Margin="20" Padding="20">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="3*"/>
                    <ColumnDefinition Width="2*"/>
                </Grid.ColumnDefinitions>

                <!-- Назва -->
                <TextBox Text="#LocalChat:)" Grid.Row="0"
Grid.Column="0" Height="40" FontSize="20" FontWeight="Bold"
                         Background="Transparent"
BorderThickness="0" Foreground="White"
                         VerticalAlignment="Top" IsEnabled="False"
Margin="0,0,0,10"/>

                <!-- Введення нікнейму з анімацією заголовка -->
                <Border Grid.Row="1" Grid.Column="0"
Margin="0,0,10,10" Background="#AA61D1FB" CornerRadius="10"
Height="60">
                    <Grid>
                        <!-- Плаваючий надпис -->
                        <TextBlock x:Name="FloatingLabel"
```

```
                    Text="Enter nickname"
                    Foreground="White"
                    FontSize="14"
                    Margin="10,18,0,0"
                    VerticalAlignment="Top"
                    Background="Transparent">
                            <TextBlock.RenderTransform>
                                <TranslateTransform/>
                            </TextBlock.RenderTransform>
                        </TextBlock>

                        <!-- Поле вводу -->
                        <TextBox x:Name="tbUsername"
                 Margin="10,0,10,23"
                 Background="Transparent"
                 BorderThickness="0"
                 FontSize="14"
                 Foreground="White"
                 VerticalContentAlignment="Bottom"
                 TextChanged="tbUsername_TextChanged"
                 GotFocus="tbUsername_GotFocus"
                 LostFocus="tbUsername_LostFocus"/>
                    </Grid>
                </Border>


                <!-- Кнопка підключення -->
                <Button x:Name="bConnDicon" Grid.Row="1"
Grid.Column="1"
                        Content="Connect" Click="Button_Click"
                        Height="38" Margin="10,0,0,10"
VerticalAlignment="Top"
                        Background="#AA61FBAD" Foreground="Black"
FontWeight="Bold"
                        BorderThickness="0" Cursor="Hand">
                    <Button.Template>
                        <ControlTemplate TargetType="Button">
                            <Border Background="{TemplateBinding
Background}" CornerRadius="10">
                                <ContentPresenter
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                            </Border>
                        </ControlTemplate>
                    </Button.Template>
                </Button>

                <!-- Список повідомлень -->
                <ListBox x:Name="lbChat" Grid.Row="2"
Grid.Column="0" Margin="0,0,10,0"
                         Background="#20FFFFFF" BorderThickness="0"
Foreground="White"

SelectionChanged="lbChat_SelectionChanged"/>
```

```
                <!-- Зображення -->
                <Image Grid.Row="2" Grid.Column="1"
Source="/Без_імені-removebg-preview.png" Stretch="Fill"
Margin="0,0,0,0"/>

                <!-- Поле для повідомлення -->
                <TextBox x:Name="tbMessage" Grid.Row="3"
Grid.ColumnSpan="2" Height="60"
                         TextWrapping="Wrap" Background="#65FFFFFF"
Foreground="White"
                         BorderBrush="White" BorderThickness="1"
KeyDown="tbMessage_KeyDown"
                         Margin="0,10,0,0"/>
            </Grid>
        </Border>
    </Grid>
</Window>
```

Додаток Б – Диск з роботою