Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних
та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету(відділення))

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)

(рівень вищої освіти)

на тему: «Розроблення корпоративного IT месенджера для компанії YoloLogic»

Виконав студент 4 курсу, групи ІСТ-41
спеціальності:
126 „Інформаційні системи та
технології ”
(шифр і назва напрямку підготовки спеціальності)

Капій Микола Володимирович
(прізвище, ініціали)

Керівник: ст.викл. Нечепуренко А.В.
(прізвище, ініціали)

Рецензент: Крошний І.М.
(прізвище, ініціали)

Львів-2023

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних систем та комп'ютерного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 " Інформаційні системи та технології"
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри ІСКМ
Сторожук О.Л.
"21" 11 2022 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Капій Микола Володимирович
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення корпоративного IT месенджера для компанії YoloLogic»*
керівник роботи *с т . в и к л . Нечепуренко А.В.*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від *"21"11 2022 року № С-521*

2. Термін подання студентом роботи *10.06.2023р.*

3. Вихідні дані до роботи: *Розробити програмне забезпечення корпоративного IT месенджера використовуючи .Net Core. У програмному забезпечені передбачити функціонал кращий месенджерів сьогодення. Також врахувати вимоги користувачів в IT компанії.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

1) Стан проблемної області;

2) Інформаційне та математичне забезпечення;

3) Програмне та технічне забезпечення;

4)Висновки.

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

Підготовка матеріалу до доповіді .

6. Консультанти розділів проекту (роботи)

7. Дата видачі завдання 23 листопада 2022р.

КАЛЕНДАРНИЙ ПЛАН

№, з/п	Етапи бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	23.11-10.12	викон.
2.	Постановка задачі і її формалізація	11.12-26.12	викон.
3.	Початок розроблення серверної частини	03.01-01.02	викон.
4.	Реалізація головних класів проекту	02.02-28.02	викон.
5.	Розроблення інтерфейсу програми	03.03.-10.03.	викон.
6.	Виконання етапу тестування бета-версії.	11.03.-16.04.	викон.
7.	Оформлення записки до дипломного проекту.	20.05.-31.05.	викон.
8.	Здача пояснювальної записки для проходження процедури рецензування.	05.06.-09.06.	викон.
9.	Створення презентації дипломної роботи	11.06.-20.06.	викон.

Студент Капій Микола Володимирович

(підпис) (прізвище та ініціали)

Керівник роботи _____ ст.викл. Нечепуренко А.В.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 76 сторінок пояснювальної записки, 80 рисунків, 15 використаних джерел та 1 додаток.

Загалом цей проект спрямований на створення корпоративного IT-месенджера, який буде ефективним інструментом спілкування для співробітників компанії, а також розділить систему на дві основні частини: клієнтську та серверну.

Для реалізації цього проекту використовувався .Net Framework 4.8, який дозволяє розробити потужний і надійний месенджер з багатьма функціями і можливостями для забезпечення безпеки і комфорту користувачів.

Клієнтська частина месенджера відповідає за взаємодію з користувачами та надає їм зручний доступ до всіх функцій месенджера.

Серверна частина месенджера відповідає за зберігання даних і обробку повідомлень.

Ключові слова: .Net Framework 4.8., програмне забезпечення, база даних, інтерфейс, хостинг.

ABSTRACT

The thesis contains 76 pages of explanatory notes, 80 pictures, 15 sources used and 1 appendix.

In general, this project is aimed at creating a corporate IT messenger, which will become an effective communication tool for company employees, and will also divide the system into two main parts: client and server.

To implement this project, .Net Framework 4.8 was used, which allows you to develop a powerful and reliable messenger with many functions and opportunities to ensure the safety and comfort of users.

The client part of the messenger is responsible for interaction with users and provides them with convenient access to all functions of the messenger.

The server part of the messenger is responsible for data storage and message processing.

Keywords: .Net Framework 4.8., software, database, interface, hosting.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити корпоративний месенджер для зручної та безпечної комунікації між співробітниками в межах підприємства. Месенджер повинен мати клієнтську та серверну частини, які взаємодіють між собою.

Клієнтська сторона:

- Використання Windows Forms для розробки клієнтської сторони.
- Використання .NET Framework 4.8.
- Реєстрація та авторизація користувачів з використанням ідентифікаційних даних (логін, пароль).
- Створення, перегляд та керування різними групами для обміну повідомленнями.
- Відправлення текстових, голосових, файлових повідомлень між користувачами.
- Запис та відтворення голосових повідомлень з використанням бібліотеки NAudio та AxWMPLib.
- Відправлення запитів до API ChatGPT з використанням бібліотеки RestSharp для отримання відповідей від бота.

Серверна сторона:

- Використання бази даних MySql для збереження користувачів, повідомлень та інших даних.
- Реалізація сервісу розсилки повідомлень за допомогою Windows Communication Foundation (WCF).

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ

AxWMPLib – ActiveX Windows Media Player Library, бібліотека для відтворення мультимедіа у Windows-додатках;

NET DB – DataBase, база даних;

GUI – Graphical User Interface, графічний інтерфейс користувача;

HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту;

JSON – JavaScript Object Notation, формат обміну даними;

REST – Representational State Transfer, архітектурний стиль взаємодії веб-сервісів;

SMTP – Simple Mail Transfer Protocol, протокол передачі електронної пошти;

SQL – Structured Query Language, мова структурованих запитів до баз даних;

WCF – Windows Communication Foundation, технологія створення розподілених сервісів у .NET.

ВСТУП

У сучасному світі швидкого розвитку технологій, ефективний внутрішній обмін інформацією є ключовим фактором для успішної роботи підприємств. Корпоративні месенджери стали невід'ємною складовою частиною організаційного середовища, забезпечуючи швидкий обмін повідомленнями між працівниками компанії. Враховуючи це, метою даної дипломної роботи є розробка корпоративного IT месенджера, який надасть зручність та ефективність внутрішнього комунікаційного процесу в компанії.

Об’єктом дослідження створення корпоративного месенджера за допомогою .Net Framework 4.8.

Метою роботи. Розробка корпоративного IT мессенджера.

Предметом дослідження є реалізація корпоративного месенджера за допомогою .Net Framework 4.8.

Практичне значення Розробка програми для обміну (текстових, голосових файлових) повідомленнями та пошуку працівників за критеріями. А також можливість створення груп користувачів.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Огляд проблемної області.

Корпоративні месенджери є інструментом внутрішнього комунікаційного процесу в організаціях. Вони дозволяють працівникам обмінюватись повідомленнями, файлами та іншою інформацією в зручний та ефективний спосіб. Основна мета корпоративних месенджерів полягає в полегшенні спілкування та спільної роботи між працівниками в межах організації.

1.2 Елементи корпоративних месенджерів

Основні елементи, які зазвичай містяться в корпоративних месенджерах, включають:

- Текстові повідомлення: Проста форма обміну інформацією, яка дозволяє працівникам обговорювати проекти, задачі та інші питання.
- Файли: Можливість передавати та отримувати файли у месенджері спрощує спільну роботу над документами та спільний доступ до необхідної інформації.
- Голосові повідомлення: Використання голосу для обміну інформацією може бути зручним у випадках, коли важко або неефективно писати текстові повідомлення.
- Групи користувачів: Можливість створювати групи працівників за проектами, відділами або іншими параметрами дозволяє зручно організовувати спільну роботу та обмін інформацією всередині цих груп.

1.3 Типи корпоративних месенджерів

На ринку існує багато різних типів корпоративних месенджерів, які мають свої особливості та переваги. Ось деякі з них:

Веб-базовані месенджери: Ці месенджери працюють через веб-браузер та не вимагають додаткового програмного забезпечення для встановлення. Вони доступні з будь-якого комп'ютера з Інтернетом, що робить їх зручними для користувачів, які працюють з різних пристроїв.

Мобільні месенджери: Ці додатки розроблені для використання на мобільних пристроях, таких як смартфони та планшети. Вони дозволяють працівникам обмінюватись повідомленнями в режимі реального часу, незалежно від їх місця перебування.

Локальні месенджери: Ці месенджери встановлюються на внутрішніх серверах компанії та забезпечують контроль над обміном повідомленнями. Вони можуть бути більш безпечними з точки зору конфіденційності даних, оскільки всі повідомлення зберігаються всередині мережі організації.

Хмарні месенджери: Ці месенджери зберігають дані на хмарних серверах, що дозволяє доступатись до них з будь-якого місця з Інтернетом. Вони можуть мати більш широкі можливості та інтеграцію з іншими сервісами.

1.4 Завдання корпоративного месенджера

Корпоративний месенджер має виконувати низку завдань, спрямованих на поліпшення комунікації та спільної роботи серед працівників організації. Основні завдання, які ставляться перед корпоративним месенджером, включають:

1. Забезпечення швидкого та зручного обміну повідомленнями: Головною метою корпоративного месенджера є забезпечення простого та швидкого способу обміну повідомленнями між працівниками. Це дозволяє зменшити затримки в комунікації та сприяє швидкому розв'язанню питань.

2. Підтримка різноманітних типів повідомлень: Корпоративний месенджер повинен забезпечувати можливість відправки текстових повідомлень, передачу файлів та голосових повідомлень. Це дозволяє працівникам обмінюватись різними типами інформації в зручний для них спосіб.

3. Створення груп користувачів: Корпоративний месенджер повинен підтримувати можливість створення груп користувачів для спільної роботи над проектами або завданнями. Це дозволяє працівникам знаходити та спілкуватись зі співробітниками, які мають спеціалізацію або інтереси, спільні з їхніми потребами.

4. Пошук працівників за спеціальністю: Одним з важливих завдань корпоративного месенджера є можливість швидкого пошуку працівників з певною спеціалізацією. Наприклад, працівникам може знадобитись швидкий доступ до Backend розробників або Frontend.

1.5. Переваги використання корпоративного месенджера

Використання корпоративного месенджера має ряд переваг порівняно зі звичайними засобами комунікації. Ось деякі з них:

1. Швидкість та ефективність комунікації: Корпоративний месенджер надає миттєвий обмін повідомленнями, що дозволяє працівникам швидко та ефективно обговорювати питання, вирішувати проблеми та спілкуватися один з одним. Він зменшує затримки в комунікації та сприяє оперативному прийняттю рішень.

2. Централізована комунікація: Корпоративний месенджер забезпечує централізовану платформу для комунікації всередині організації. Це означає, що всі повідомлення та обговорення зберігаються на одній платформі, що спрощує пошук історії спілкування та забезпечує зручний доступ до попередніх повідомлень.

3. Підвищена продуктивність: Корпоративний месенджер допомагає збільшити продуктивність працівників шляхом зниження необхідності в електронній пошті, телефонних дзвінках або зустрічах вживу для комунікації. Він сприяє швидкій передачі інформації та спрощує спільну роботу над проектами.

4. Зручний пошук інформації та співробітників: Корпоративний месенджер часто має функцію пошуку, що дозволяє швидко знайти необхідну інформацію або співробітників з певною спеціалізацією. Це спрощує спілкування та співпрацю між різними відділами та командами.

Ці переваги демонструють, як корпоративний месенджер може покращити комунікацію та спільну роботу всередині організації, сприяючи зростанню ефективності та продуктивності.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. .NET

У даному розділі буде розглянуто технологічне забезпечення, використане для розробки корпоративного месенджера. Зокрема, будуть описані наступні компоненти технологій:

1. .Net Framework 4.8: .Net Framework є програмним забезпеченням, розробленим компанією Microsoft, яке надає зручність інструментів та середовище виконання для розробки та запуску програм на мові програмування C#. Використання .Net Framework 4.8 дозволило створити програму, яка працює на платформі Windows та використовує різноманітні функціональні можливості, доступні у фреймворку.

2. WinForm: WinForm є одним з найпоширеніших інструментів для розробки десктопних додатків на платформі Windows. Використання WinForm дозволило створити графічний інтерфейс корпоративного месенджера з допомогою елементів керування, таких як кнопки, тексти, списки тощо. Це дозволило забезпечити зручну та інтуїтивно зрозумілу взаємодію користувача з програмою.

3. Windows Communication Foundation (WCF): WCF є технологією, розробленою Microsoft, яка дозволяє створювати розподілені додатки з використанням веб-служб та інших протоколів комунікації. Використання WCF дозволило реалізувати сервіс розсилки повідомлень, який забезпечує передачу повідомлень між користувачами через мережу.

4. NAudio: NAudio є бібліотекою для роботи з аудіофайлами та аудіо-потоками в додатках на платформі .NET. Використання NAudio дозволило здійснювати запис голосових повідомлень у корпоративному месенджері, що надає можливість користувачам обмінюватися голосовими повідомленнями.

5. AxWMPLib: AxWMPLib є ActiveX-контролем, який надає функціональність відтворення аудіо- та відеофайлів у додатках на платформі Windows. Використання AxWMPLib дозволило реалізувати відтворення голосових повідомлень у корпоративному месенджері.

6. RestSharp є потужним інструментом для здійснення HTTP-запитів та взаємодії з веб-сервісами у .NET Framework. Вона надає зручний та простий у використанні інтерфейс для створення, відправлення та обробки HTTP-запитів. Використання бібліотеки RestSharp дозволило здійснення HTTP-запитів до API ChatGPT.

7. MySQL база даних: MySQL є однією з найпоширеніших систем управління базами даних (СУБД), яка використовується для зберігання та управління даними. Використання MySQL дозволило створити базу даних для зберігання повідомлень, контактів та інших даних, необхідних для роботи корпоративного месенджера.

Ці технології та компоненти були використані для розробки функціональності месенджера, такої як розсилка повідомлень, запис та відтворення голосових повідомлень, робота з базою даних та графічний інтерфейс користувача. Використання цих технологій допомогло забезпечити потрібні функції та можливості для корпоративного месенджера.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Розробка серверної частини.

Почнемо зі створення бази даних де буде зберігатись вся необхідна інформація.

В базі даних міститься 5 таблиць

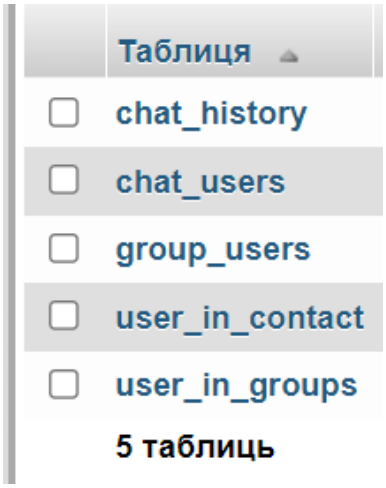


Рис. 3.1. Таблиці бази даних

Детальніше про таблиці:

1. chat_history – таблиця зберігаї історію переписки користувачів.
2. chat_users – таблиця містить всіх користувачів які зареєстровані в нашому додатку.
3. group_users – в цій таблиці ми зберігаємо групи які створили користувачі.
4. user_in_contact – ця таблиця містить зв'язок багато до багатьох, саме такий зв'язок забеспечує поєднання даних які містяться у інших таблиця. Таблиця зберігає дані про те які контакти істують в певного користувача.
5. user_in_groups – це таблиця як і попередня реалізовує зв'язок багато до багатьох. Зберігає інформацію про те в яких групах знаходяться користувачі.

Використання MySQL дозволило створити базу даних для зберігання повідомлень, контактів та інших даних, необхідних для роботи корпоративного месенджера.

Для зв’язку з базою даних було використано MySQL.Data.MySqlClient він є частиною .NET-платформи і представляє собою набір класів, які дозволяють взаємодіяти з MySQL базами даних з додатків, розроблених на платформі .NET. Цей набір класів надає різноманітні функції та методи для підключення до MySQL сервера, виконання запитів SQL, отримання та збереження даних у базі даних.

Для виконання запитів був реалізований статичний клас Info та наступні методи в ньому:

SelectFromDB – реалізує вибірку з бази даних.

```
14 references
public static DataTable SelectFromDB(string commandString)
{
    DataTable dataTable = new DataTable();

    using (MySqlConnection connection = new MySqlConnection(Info.ConStr))
    {
        connection.Open();

        using (MySqlCommand command = new MySqlCommand(commandString, connection))
        {
            try
            {
                using (MySqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.HasRows)
                    {
                        dataTable.Load(reader);
                    }
                }
            }
            catch
            {
                MessageBox.Show("Неможливо з'єднатися з SQL-сервером!",
                    "Помилка з'єднання", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    return dataTable;
}
```

Рис. 3.2. Метод вибірки з бази даних

InsertToDB – реалізує вставку в базу даних.

```
13 references
public static int InsertToDB(string commandString)
{
    int id = 0;

    using (MySqlConnection connection = new MySqlConnection(Info.ConStr))
    {
        connection.Open();

        using (MySqlCommand command = new MySqlCommand(commandString, connection))
        {
            try
            {
                id = Convert.ToInt32(command.ExecuteScalar());
            }
            catch
            {
                MessageBox.Show("Неможливо з'єднатися з SQL-сервером!",
                    "Помилка з'єднання", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    return id;
}
```

Рис. 3.3. Метод вставки з бази даних

DeleteFromDB – видалення з бази даних.

```
2 references
public static void DeleteFromDB(string commandString)
{
    using (MySqlConnection connection = new MySqlConnection(Info.ConStr))
    {
        connection.Open();

        using (MySqlCommand command = new MySqlCommand(commandString, connection))
        {
            try
            {
                command.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Неможливо з'єднатися з SQL-сервером!",
                    "Помилка з'єднання", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```

Рис. 3.4. Метод видалення з бази даних

UploadFileToDB – завантаження файлів в базу даних.

```
4 references
public static void UploadFileToDB(string path, string commandString)
{
    if (!string.IsNullOrWhiteSpace(path))
    {
        byte[] rawData;

        using (FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read))
        {
            int fileSize = (int)fs.Length;
            rawData = new byte[fileSize];
            fs.Read(rawData, 0, fileSize);
        }

        using (MySqlConnection connection = new MySqlConnection(Info.ConStr))
        {
            connection.Open();

            using (MySqlCommand command = new MySqlCommand(commandString, connection))
            {
                try
                {
                    command.Parameters.AddWithValue("@File", rawData);
                    command.Parameters.AddWithValue("@Filename", Path.GetFileNameWithoutExtension(path));
                    command.Parameters.AddWithValue("@Extension", Path.GetExtension(path));

                    command.ExecuteNonQuery();
                }
                catch
                {
                    MessageBox.Show("Неможливо з'єднатися з SQL-сервером!",
                        "Помилка з'єднання", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
}
```

Рис. 3.5. Метод завантаження файлів в базу даних

DownloadFileFromDB – скачування файлів з бази даних.

```
3 references
public static byte[] DownloadFileFromDB(string commandString)
{
    byte[] data = null;

    using (MySqlConnection connection = new MySqlConnection(Info.ConStr))
    {
        MySqlCommand command = new MySqlCommand(commandString, connection);

        connection.Open();
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            try
            {
                if (reader.Read())
                {
                    if (!reader.IsDBNull(0))
                    {
                        data = (byte[])reader.GetValue(0);
                    }
                }
            }
            catch
            {
                MessageBox.Show("Неможливо з'єднатися з SQL-сервером!",
                    "Помилка з'єднання", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    return data;
}
```

Рис. 3.6. Метод скачування файлів з бази даних

Маючи реалізовані методи для роботи з базою даних можна перейти до розробки сервісу по обміну повідомленнями.

Для цього використаємо Windows Communication Foundation (WCF). Створюємо окремий проект wcf_service в якому створюємо інтерфейс та класи. Перший інтерфейс містить методи

Підключення(Connect), Відключення(Disconnect), та

Відправку повідомлення (SendMsg).

```
[ServiceContract(CallbackContract = typeof(IServerCallback))]
1 reference
public interface IServiceAniChat
{
    [OperationContract]
    1 reference
    int Connect(string name, string chatname);

    [OperationContract]
    1 reference
    void Disconnect(int id, string chatname);

    [OperationContract(IsOneWay = true)]
    1 reference
    void SendMsg(string msg, int id , string chatname);
}
```

Рис. 3.7. Інтерфейс IServiceAniChat

Наступний містить метод MsgCallBack який заберпечує саму розсилку.

```
2 references
public interface IServerCallback
{
    [OperationContract(IsOneWay = true)]
    1 reference
    void MsgCallback(string msg);
}
```

Рис. 3.8. Інтерфейс IServerCallBack

Клас ServerUser використовується, для представлення даних користувача на серверній стороні WCF-сервісу. Кожен об'єкт ServerUser містить інформацію про окремого користувача, таку як його ідентифікатор, ім'я та інші властивості. Також він має доступ до OperationContext, що дозволяє отримати доступ до контексту поточної операції, наприклад, для отримання доступу до каналу зв'язку чи інших сервісних функцій.

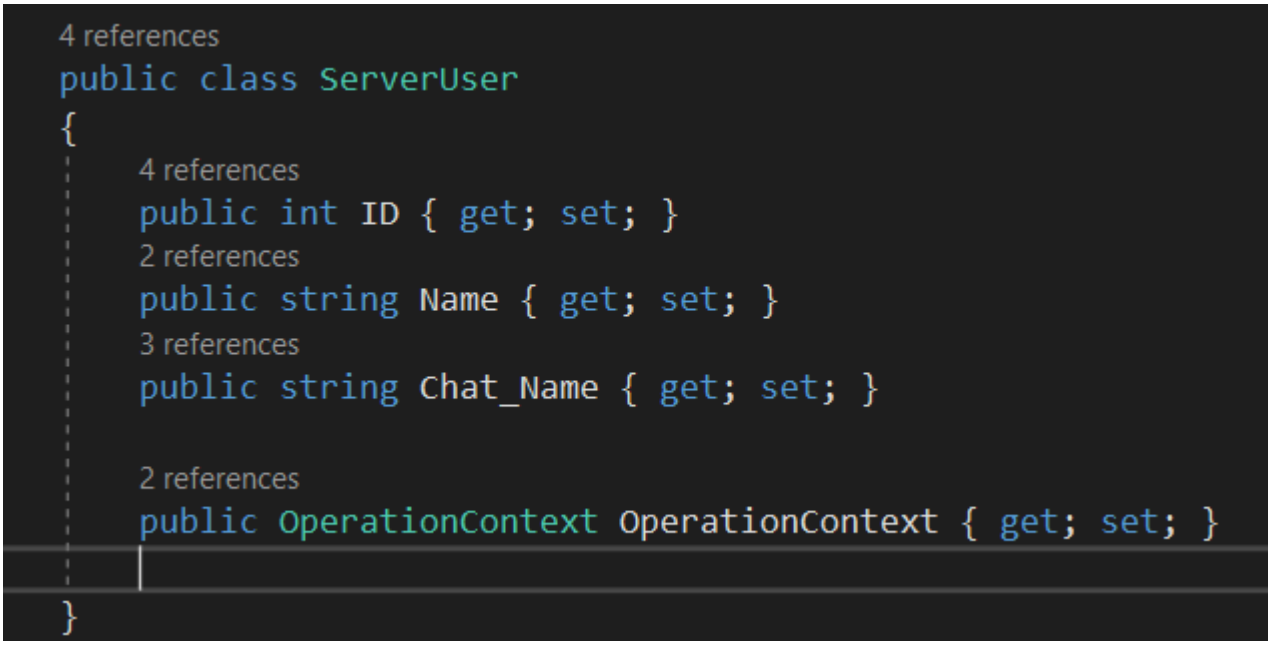


Рис. 3.9. Клас ServerUser

Клас ServiceAniChat, який також використовується для реалізації WCF-сервісу. Цей клас реалізує інтерфейс IServiceAniChat, і він встановлює режим контексту екземпляра сервісу як InstanceContextMode.Single. Це означає, що для кожного клієнта буде створено тільки один екземпляр сервісу.

В класі ServiceAniChat визначено ряд полів та змінних:

List<ServerUser> users - список, який містить об'єкти ServerUser. Цей список представляє користувачів, які підключені до сервісу.

int id - цілочисельна змінна, яка використовується для присвоєння унікального ідентифікатора кожному користувачеві при підключенні.

Метод Connect(string name, string chatname) використовується для підключення нового користувача до сервісу. Він створює новий об'єкт ServerUser з вказаними параметрами (ідентифікатор, ім'я користувача, назва чату) і додає його до списку користувачів (users). Потім повертається ідентифікатор, присвоєний користувачу.

```
1 reference
public int Connect(string name, string chatname)
{
    ServerUser user = new ServerUser()
    {
        ID = id,
        Name = name,
        Chat_Name = chatname,
        OperationContext = OperationContext.Current
    };
    id++;

    //SendMsg(". "+user.Name +" : "+ " Connect !",0,chatname);
    users.Add(user);
    return user.ID;
}
```

Рис. 3.10. Метод Connect

Метод Disconnect(int id, string chatname) використовується для відключення користувача від сервісу. Він знаходить користувача за заданим ідентифікатором у списку users і видаляє його зі списку.

```
1 reference
public void Disconnect(int id, string chatname)
{
    var user = users.FirstOrDefault(i => i.ID == id);
    if (user != null)
    {
        users.Remove(user);
        //SendMsg(". " + user.Name + " : " + " Disconnect !",0, chatname);
    }
}
```

Рис. 3.11. Метод Disconnect

Метод SendMsg(string msg, int id, string chatname) використовується для відправки повідомлення всім користувачам, які належать до певного чату (chatname). Він знаходить користувачів, які належать до вказаного чату або чату "notification" і викликає метод MsgCallback для кожного користувача, щоб

надіслати повідомлення. Повідомлення містить інформацію про чат, відправника та текст повідомлення.

```
1 reference
public void SendMsg(string msg, int id, string chatname)
{
    var group_user =
    from users in users
    where users.Chat_Name == chatname || users.Chat_Name == "notification"
    select users;

    foreach (var item in group_user)
    {
        string answer = String.Empty;// DateTime.Now.ToShortTimeString();

        var user = users.FirstOrDefault(i => i.ID == id);

        if (user != null)
        {
            answer += "{'chatname':'" + chatname + "' ,'username':'" + user.Name + "',";
        }

        answer += msg + " }";

        item.OperationContext.GetCallbackChannel<IServerCallback>().MsgCallback(answer);
    }
}
```

Рис. 3.12. Метод SendMsg

Після створення нашого сервісу потрібно створити ще один проект AniHost для хостінгу нашого сервісу.

Він містить класс Host який містить метод Main, є точкою входу програми.

У методі Main створюється екземпляр ServiceHost, який служить для хостингу (запуску) WCF-сервісу. Клас ServiceHost отримує тип сервісу, який він буде хостити, в даному випадку - wcf_service.ServiceAniChat.

wcf_service є назвою простору імен, в якому розташований клас ServiceAniChat.

За допомогою методу Open() відбувається відкриття хоста і початок прослуховування вхідних з'єднань.

Console.ReadLine() використовується для того, щоб програма не завершувалась і продовжувала прослуховування вхідних з'єднань до тих пір, поки користувач не натисне клавішу Enter.

Далі нам необхідно відкрити файл App.config та зробити наступні налаштування

```
<system.serviceModel>
    <behaviors>
        <serviceBehaviors>
            <behavior name="mexBeh">
                <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />
                <serviceDebug includeExceptionDetailInFaults="false" />
            </behavior>
        </serviceBehaviors>
    </behaviors>
    <services>
        <service name="wcf_service.ServiceAniChat" behaviorConfiguration="mexBeh">
            <endpoint address="" binding="netTcpBinding" contract="wcf_service.IServiceAniChat">
            </endpoint>
            <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
            <host>
                <baseAddresses>
                    <add baseAddress="http://localhost:8301" />
                    <add baseAddress="net.tcp://localhost:8302" />
                </baseAddresses>
            </host>
        </service>
    </services>
</system.serviceModel>
```

Рис. 3.13. Файл конфігуратцій ServiceHost

3.2. Розробка інтерфейсу програми

Розроблення інтерфейсу за допомогою Windows Forms.

Основні особливості WinForms:

1. Простота використанні: WinForms надає простий та зрозумілий API для створення графічного інтерфейсу. Розробка відбувається за допомогою перетягування та розміщення елементів керування (кнопок, текстових полів, списків тощо) на формі і задання їх властивостей через візуальний дизайнер.
2. Багатофункціональність: WinForms надає велику кількість вбудованих елементів керування (кнопки, поле введення тексту, таблиці, дерева, вкладки тощо), які можна використовувати для створення різноманітних інтерфейсів користувача.
3. Підтримка подій: WinForms підтримує події і дозволяє реагувати на взаємодію користувача з елементами керування. Наприклад, можна

створити обробники подій для натискання кнопок, введення тексту або вибору елементів списку.

4. Розширені можливості налаштування: WinForms надає широкі можливості налаштування властивостей елементів керування, таких як розміщення, розмір, кольори, шрифти, стилі, межі тощо. Це дозволяє створювати індивідуальний та привабливий інтерфейс.
5. Інтеграція з .NET Framework: WinForms повністю інтегрований з .NET Framework, що дає доступ до багатьох інших функцій і можливостей цього фреймворку, таких як робота з базами даних, мережеве взаємодія, робота з файлами та інше.

WinForms є широко використовуваним інструментом для створення десктопних програм на платформі Windows. Він дозволяє розробникам швидко створювати графічні інтерфейси з високою функціональністю та зручним використанням.

3.2.1 Авторизація в месенджер

Для початку реалізуємо інтефейс для входу в додаток.

Для цього створимо форму LogIn яка має наступний вигляд

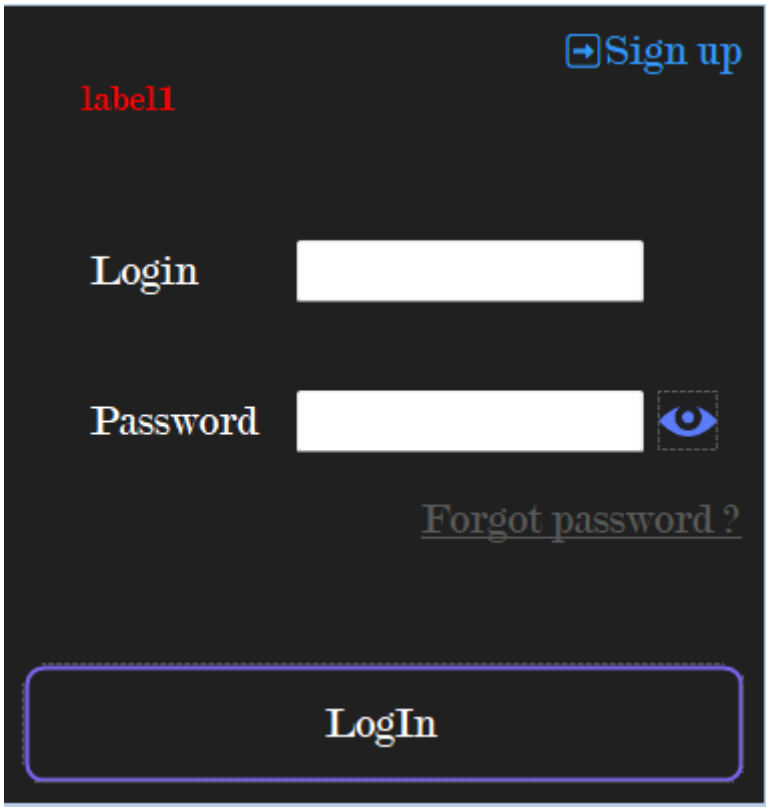


Рис. 3.14. Форма авторизації

Код форми містить наступні методи:

LogIn_Load: Це подія, яка відбувається при завантаженні форми LogIn. В ній встановлюється рядок підключення до бази даних, виконується асинхронна операція Load_User(), ініціалізуються різні елементи інтерфейсу (наприклад, встановлюється видимість певних елементів і фокусується на полі введення логіна).

```
1 reference
private async void LogIn_Load(object sender, EventArgs e)
{
    Info.ConStr = "server = " + IPSQLserver +
        "; characterset = cp1251;" +
        "user = " + LogSQLserver + "; database = " +
        DBserver + "; password = " + PasSQLserver;

    await Task.Run(() => Load_User());

    retrieve_PassLb.Visible = false;
    validErr_lb.Visible = false;

    txtBoxPass.UseSystemPasswordChar = true;
    txtBoxLogin.Focus();
}
```

Рис. 14. Метод LogIn_Load

Load_User: Цей метод виконує запит до бази даних для отримання інформації про користувачів. Результати запиту зберігаються у змінній dt (об'єкт DataTable), і дані переносяться до двовимірного масиву matrix.

```
3 references
public void Load_User()
{
    dt = Info.SelectFromDB("Select * from Chat_Users");
    int count = dt.Rows.Count;

    matrix = new string[count, 7];
    for (int i = 0; i < count; i++)
    {
        matrix[i, 0] = dt.Rows[i].Field<int>("idChat_Users").
        matrix[i, 1] = dt.Rows[i].Field<string>("Email");
        matrix[i, 2] = dt.Rows[i].Field<string>("Login");
        matrix[i, 3] = dt.Rows[i].Field<string>("Password");
        matrix[i, 4] = dt.Rows[i].Field<string>("Rank");
        matrix[i, 5] = dt.Rows[i].Field<string>("Gitlink");
        matrix[i, 6] = dt.Rows[i].Field<string>("Inlink");
    }
}
```

Рис. 3.15. Метод Load_User

Avtorization: Цей метод виконує процес авторизації користувача. Він перевіряє, чи введений логін користувача і пароль збігаються з даними, отриманими з бази даних. Якщо збігаються, виконується відповідна обробка (установка значень змінних Info, очищення полів введення і перехід до головної форми MainForm). У противному випадку виводиться повідомлення про помилку.

```
3 references
private void Avtorization()
{
    bool flUser = false;
    for (int i = 0; i < matrix.GetLength(0); i++)
    {
        if (String.Equals(txtBoxLogin.Text.ToUpper(), matrix[i, 2].ToUpper()))
        {
            flUser = true;

            Info.idUser = int.Parse(matrix[i, 0]);
            email = matrix[i, 1];
            Info.nameUser = matrix[i, 2];

            if (String.Equals(Info.EncriptedPassword(txtBoxPass.Text), matrix[i, 3]))
            {
                Info.Rank = matrix[i, 4];
                Info.GitLink = matrix[i, 5];
                Info.InLink = matrix[i, 6];
                txtBoxLogin.Text = string.Empty;
                txtBoxPass.Text = string.Empty;

                this.Hide();

                MainForm mf = new MainForm();
                mf.ShowDialog();
                dt.Dispose();
            }
            else
            {
                ShowErr("Невірний password !");
                txtBoxPass.Text = String.Empty;
                txtBoxPass.Focus();

                retrieve_PassLb.Visible = true;
            }
        }
    }
    if (!flUser)
    {
        ShowErr("Невірний login !");
```

Рис. 3.16. Метод Avtorization

label_reg_Click і retrieve_PassLb_Click: Ці події відбуваються при натисканні на відповідні мітки на формі. Вони відкривають нові форми (AddNewUser і RetrievePassForm) для реєстрації нового користувача або відновлення пароля.

```
1 reference
private void label_reg_Click(object sender, EventArgs e)
{
    this.Hide();
    AddNewUser new_user = new AddNewUser(this);
    new_user.ShowDialog();
}
```

Рис. 3.17. Подія label_reg_Click

```
1 reference
private void retrieve_PassLb_Click(object sender, EventArgs e
{
    this.Hide();
    RetrievePassForm new_user = new RetrievePassForm(this, em
    new_user.ShowDialog();
}
```

Рис. 3.18. Подія retrieve_PassLb_Click

3.2.2 Реєстрація в месенджері

Далі створемо самі форми для Реєстрації та Відновлення паролю.

Форма Реєстрації має такий вигляд:

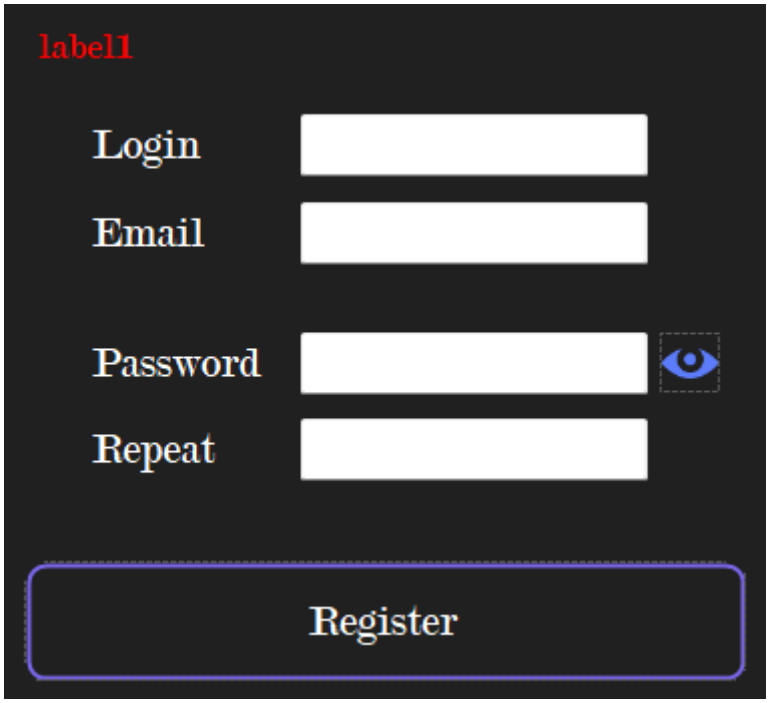


Рис. 3.19. Форма Реєстрації

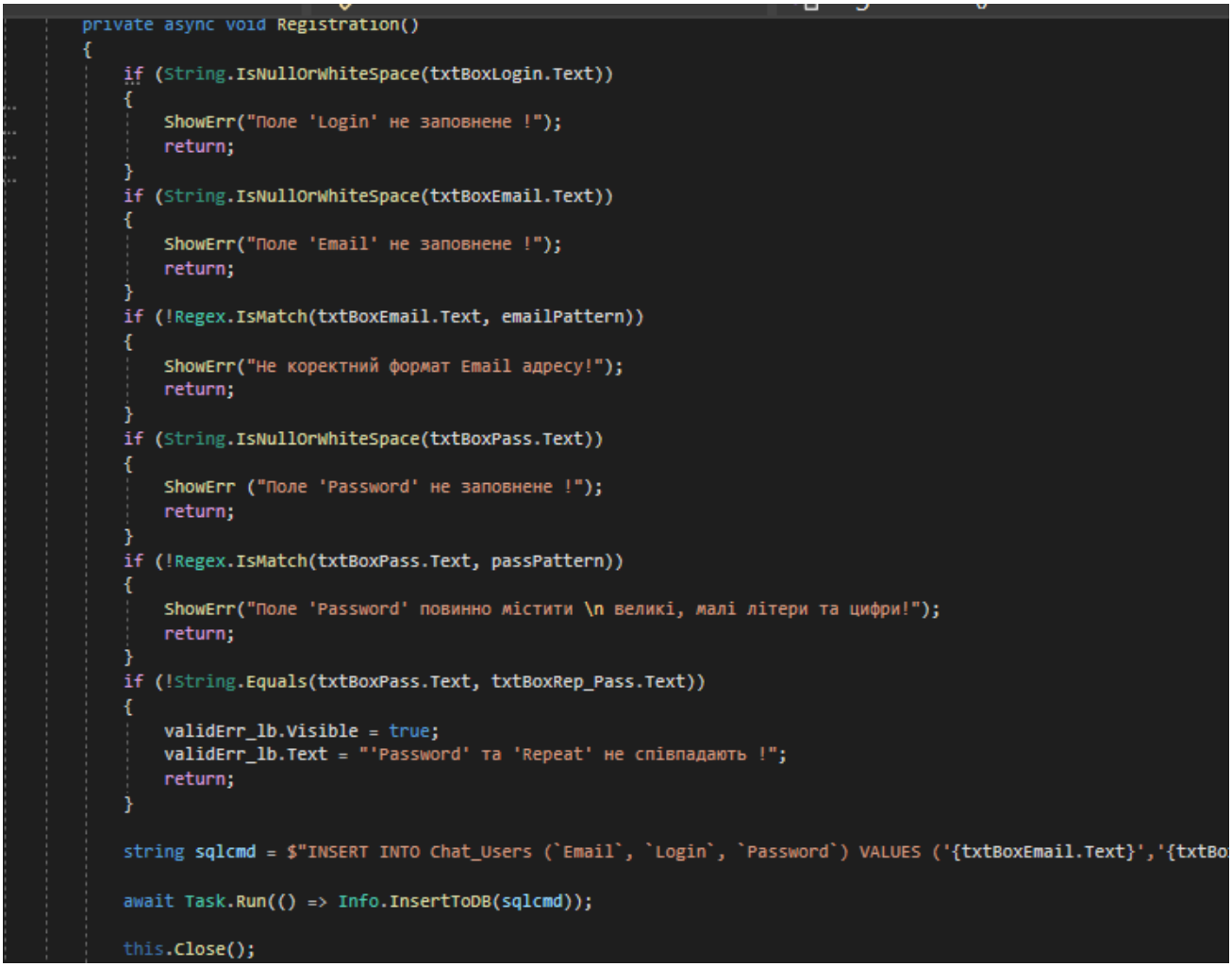
Код форми містить наступні методи:

btnRegister_Click: Ця подія відбувається при натисканні кнопки "Зареєструватися". Викликається метод Registration для реєстрації нового користувача.

```
1 reference
private void btnRegister_Click(object sender, EventArgs e)
{
    Registration();
}
```

Рис. 3.20. Подія btnRegister_Click

Registration: Цей метод виконує процес реєстрації нового користувача. Перевіряється введена інформація (логін, електронна пошта, пароль) на наявність помилок (наприклад, порожні поля або неправильний формат електронної пошти). Якщо дані коректні, виконується вставка запису про нового користувача у базу даних.



```
private async void Registration()
{
    if (String.IsNullOrWhiteSpace(txtBoxLogin.Text))
    {
        ShowErr("Поле 'Login' не заповнене !");
        return;
    }
    if (String.IsNullOrWhiteSpace(txtBoxEmail.Text))
    {
        ShowErr("Поле 'Email' не заповнене !");
        return;
    }
    if (!Regex.IsMatch(txtBoxEmail.Text, emailPattern))
    {
        ShowErr("Не коректний формат Email адресу!");
        return;
    }
    if (String.IsNullOrWhiteSpace(txtBoxPass.Text))
    {
        ShowErr ("Поле 'Password' не заповнене !");
        return;
    }
    if (!Regex.IsMatch(txtBoxPass.Text, passPattern))
    {
        ShowErr("Поле 'Password' повинно містити \n великі, малі літери та цифри!");
        return;
    }
    if (!String.Equals(txtBoxPass.Text, txtBoxRep_Pass.Text))
    {
        validErr_lb.Visible = true;
        validErr_lb.Text = "'Password' та 'Repeat' не співпадають !";
        return;
    }

    string sqlcmd = $"INSERT INTO Chat_Users (`Email`, `Login`, `Password`) VALUES ('{txtBoxEmail.Text}','{txtBo

    await Task.Run(() => Info.InsertToDB(sqlcmd));

    this.Close();
```

Рис. 3.21. Метод Registration

3.2.3 Відновлення втраченого паролю

Для відновлення паролю буде використовуватись пошта яку вказав користувавач при реєстрації. А семе буде генеруватись 4-х значний код та відравлятись на пошту. Після того як користувач введе цей код у відповідне поле форми йому стане доступна зміна паролю.

Для початку створимо класс SmtpEmail, який відповідає за надсилання електронних листів за допомогою протоколу SMTP.

Рис. 22. Клас SmtpEmail

Send: Цей публічний метод приймає параметри toEmail (адреса електронної пошти одержувача), subject (тема листа) і text (тіло листа). Він виконує надсилання електронного листа.

Створюється об'єкт MailMessage, у якому вказується відправник, одержувач, тема та тіло повідомлення. Тіло повідомлення в даному випадку формується у форматі HTML.

Створюється об'єкт SmtpClient для відправки повідомлення. Вказується SMTP-сервер (у цьому випадку smtp.gmail.com для Gmail) та інші налаштування, такі як порт, облікові дані для автентифікації і включення SSL.

Викликається метод Send об'єкта SmtpClient, який надсилає електронне повідомлення.

Після того як був реалізований метод відправки повідомлення на пошту, створемо форму для відновлення.

Форма відновлення паролю має такий вигляд:

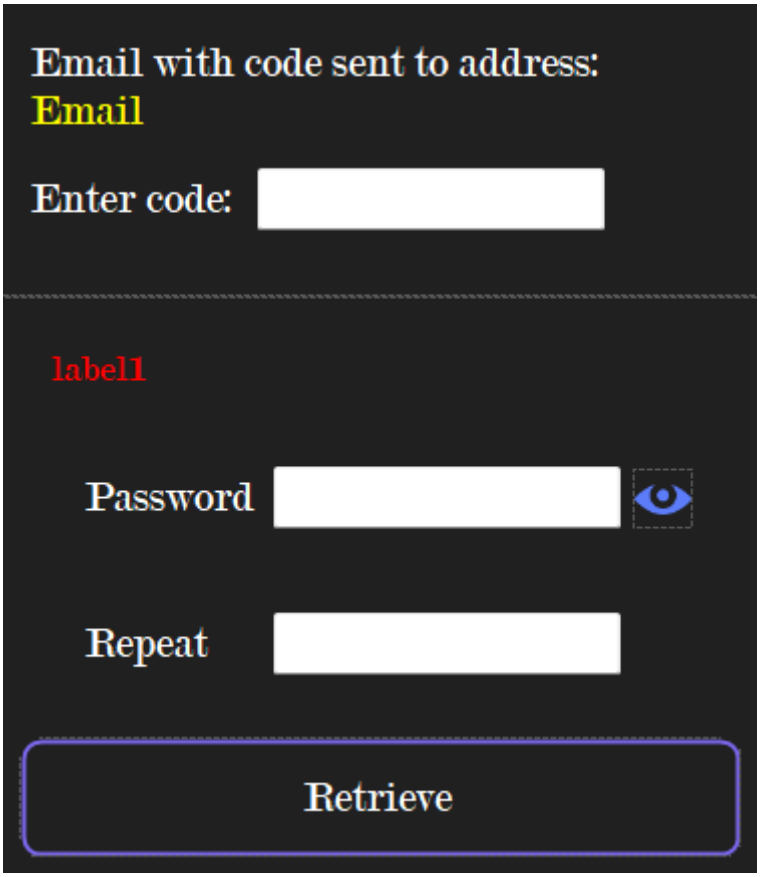


Рис. 3.23. Форма відновлення паролю.

Код форми містить наступні методи:

RetrievePassForm_Load: Цей метод викликається при завантаженні форми для відновлення пароля. Він встановлює адресу електронної пошти у відповідному елементі керування, приховує елементи для введення коду та нового пароля, і асинхронно відправляє код відновлення на вказану електронну пошту.

```
1 reference
private async void RetrievePassForm_Load(object sender, EventArgs e)
{
    Email_lb.Text = email;
    retrievePass_pl.Visible = false;
    this.Height = 200;
    await Task.Run(() => SendCodetoEmail());
}
```

Рис. 3.24. Подія RetrievePassForm_Load

SendCodetoEmail: Цей метод відправляє код відновлення на адресу електронної пошти користувача. Він використовує клас SmtpEmail для надсилання електронного листа з кодом.

- Генерується випадковий код відновлення за допомогою класу Random.
- Створюється текстове повідомлення у форматі HTML, яке містить код відновлення.
- Створюється об'єкт SmtpEmail і викликається його метод Send, передаючи адресу електронної пошти користувача, тему листа та текст повідомлення.

Рис. 3.25. Метод SendCodetoEmail

Retrieve_btn_Click: Цей метод викликається при натисканні кнопки "Retrieve" для збереження нового пароля. Він перевіряє, чи введений пароль відповідає вимогам (містить великі і малі літери та цифри) та чи співпадають пароль і повторення пароля. Якщо перевірка пройшла успішно, пароль оновлюється у базі даних, виводиться повідомлення про успішне зміну пароля, і форма закривається.

```
1 reference
private void Retrieve_btn_Click(object sender, EventArgs e)
{
    validErr_lb.ForeColor = Color.Red;

    if (String.IsNullOrWhiteSpace(pass_tb.Text))
    {
        ShowErr("Поле 'Password' не заповнене !");
        return;
    }
    if (!Regex.IsMatch(pass_tb.Text, pattern))
    {
        ShowErr("Поле 'Password' повинно містити \n великі, малі літери та цифри!");
        return;
    }
    if (!String.Equals(pass_tb.Text, repeatpass_tb.Text))
    {
        validErr_lb.Text = "'Password' та 'Repeat' не співпадають !";
        return;
    }

    string sqlcmd = $"UPDATE Chat_Users SET Password = '{Info.EncriptedPassword(pass_tb.Text)}' WHERE idChat_Users = '{Info.idUser}' ;";
    Info.InsertToDB(sqlcmd);

    MessageBox.Show("Your password has been changed !");

    this.Close();
}
```

Рис. 3.26. Подія Retrieve_btn_Click

3.2.4 Основний інтерфейс месенджера

Після того як був розроблений вхід, реєстрація та відновлення паролю, ми можемо перейти до основного інтерфейсу корпоративного чату.

Для цього створемо mainForm форму яка буде мати наступний вигляд.

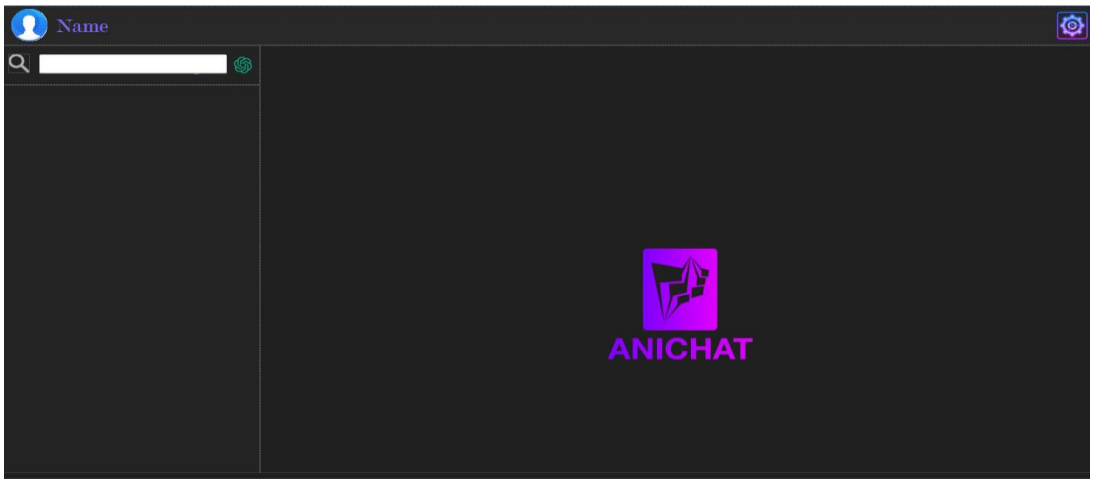


Рис. 3.27.

Код форми містить такі методи:

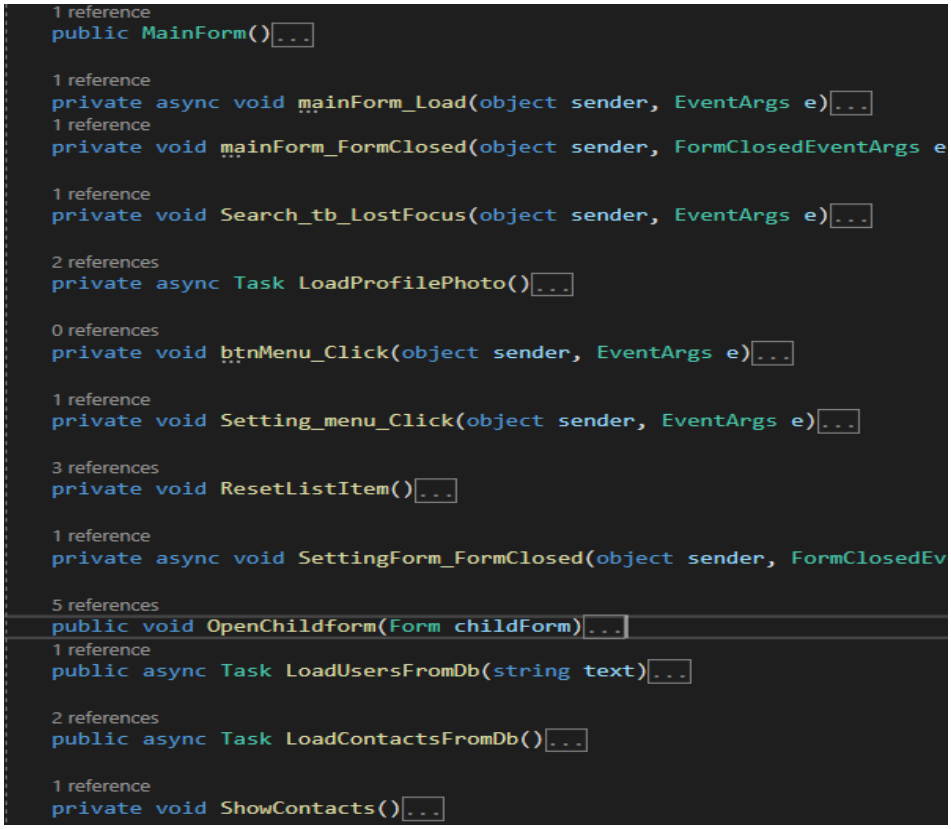


Рис. 3.28. Методи класу MainForm

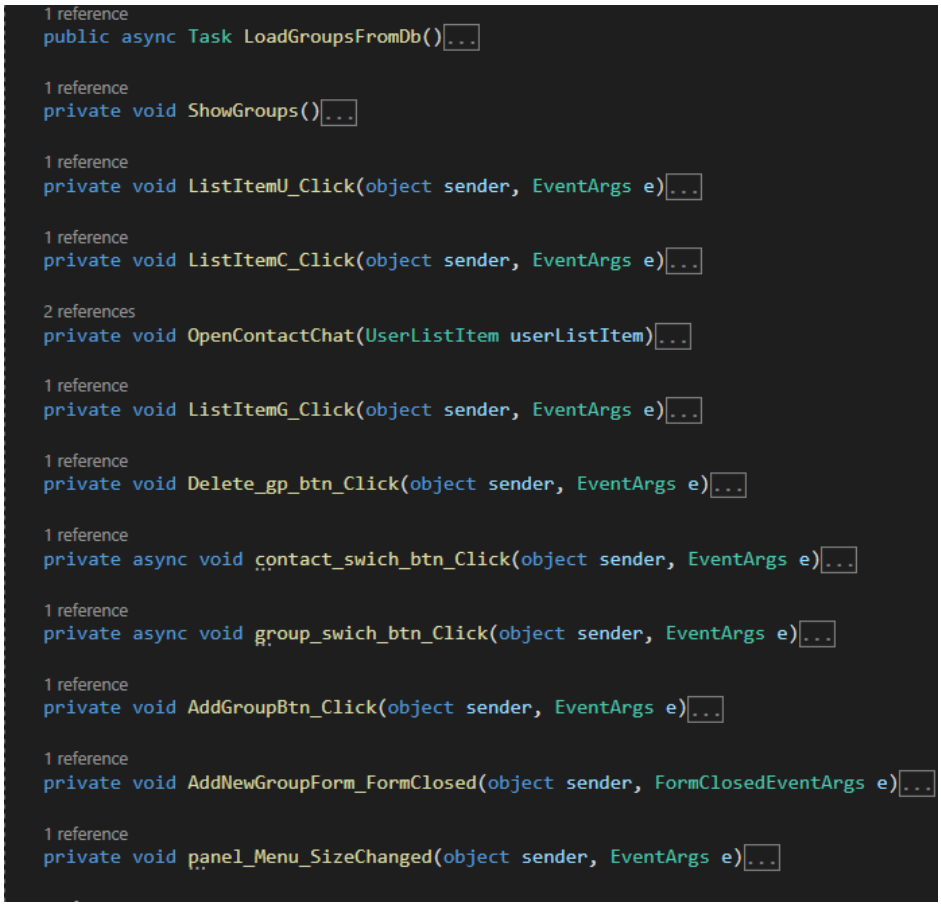


Рис. 3.29. Методи класу MainForm

```
1 reference
private async void mainForm_SizeChanged(object sender, EventArgs e)...

1 reference
private void MenuPanelAnimation_hide()...

1 reference
private void MenuPanelAnimation_show()...

1 reference
private async void Search_tb_KeyDown(object sender, KeyEventArgs e)...

1 reference
public void MsgCallback(string msg)...
1 reference
public void Alert(string msg)...

1 reference
public void ConnectUser()...

2 references
public void DisconnectUser()...

1 reference
private void pictureBox1_Click(object sender, EventArgs e)...

1 reference
private void GPTBot_btn_Click(object sender, EventArgs e)...
}
```

Рис. 3.30. Методи класу MainForm

Короткий опис методів mainFrom:

Метод mainForm_Load: Цей метод викликається після завантаження форми і встановлює різні параметри та викликає інші методи, такі як завантаження фото профілю та контактів з бази даних. Також, в залежності від стану підключення користувача (isUserConn), відбувається з'єднання або відключення користувача до сервісу.

Метод mainForm_FormClosed: Цей метод викликається при закритті головної форми і він відповідає за закриття активної дочірньої форми, відключення користувача від сервісу та закриття програми.

Методи LoadProfilePhoto, LoadContactsFromDb, LoadUsersFromDb та LoadGroupsFromDb: Ці методи відповідають за завантаження профільного фото користувача, контактів з бази даних та користувачів та груп з бази даних. Вони використовують асинхронні операції для отримання даних з бази даних або зовнішніх джерел.

Методи OpenChildform, OpenContactChat, ListItemU_Click, ListItemC_Click, ListItemG_Click, Delete_gp_btn_Click: Ці методи відповідають за відкриття нових форм або перемикання на існуючі форми при натисканні відповідних елементів у головному вікні. Наприклад, OpenChildform відкриває нову дочірню форму, OpenContactChat відкриває форму чату з контактом, а ListItemU_Click, ListItemC_Click, ListItemG_Click та Delete_gp_btn_Click реагують на події натискання на елементи у списку контактів або груп та виконують відповідні дії.

contact_swich_btn_Click: Цей метод викликається при натисканні кнопки переключення на контакти. Він змінює відображення активного розділювача між контактами та групами, очищає панель контактів і асинхронно завантажує контакти з бази даних.

group_swich_btn_Click: Цей метод викликається при натисканні кнопки переключення на групи. Він змінює відображення активного розділювача між контактами та групами, очищає панель контактів і асинхронно завантажує групи з бази даних.

AddGroupBtn_Click: Цей метод викликається при натисканні кнопки "Додати групу". Він створює нову форму для додавання нової групи, додає обробник події FormClosed для нової форми і викликає метод OpenChildform для відкриття нової форми. Також викликається метод ResetListItem.

AddNewGroupForm_FormClosed: Цей метод викликається при закритті форми для додавання нової групи. Він викликає метод PerformClick для кнопки group_swich_btn, що викличе подію переключення на групи.

panel_Menu_SizeChanged: Цей метод викликається при зміні розміру панелі меню. Він встановлює максимальне значення прокрутки панелі в горизонтальному напрямку та оновлює панель.

mainForm_SizeChanged: Цей метод викликається при зміні розміру головної форми. Він перевіряє ширину форми і в залежності від значення викликує анімаційні методи MenuPanelAnimation_hide або MenuPanelAnimation_show.

MenuPanelAnimation_hide та MenuPanelAnimation_show: Ці методи відповідають за анімацію панелі меню, яка зникає або з'являється в залежності від її поточного стану.

Search_tb_KeyDown: Цей метод викликається при натисканні клавіші Enter у полі пошуку. Він перевіряє, чи не є поле пошуку порожнім і асинхронно завантажує користувачів з бази даних за заданим критерієм пошуку.

MsgCallback: Цей метод викликається ззовні і приймає повідомлення msg. Він аналізує повідомлення у форматі JSON, отримує ім'я чату, перевіряє, чи активна форма відповідає цьому чату і встановлює кількість нових повідомлень для відповідного елемента контакту чи групи. Якщо головна форма згорнута або не відображена, він викликає метод Alert для відображення сповіщення.

Alert: Цей метод відображає сповіщення, створюючи і відкриваючи нову форму FormAlert.

ConnectUser та DisconnectUser: Ці методи відповідають за підключення і відключення користувача до сервера. ConnectUser намагається встановити з'єднання з сервером за допомогою ідентифікатора ID. Якщо з'єднання успішно, змінна isUserConn встановлюється на true. У разі невдачі виводиться повідомлення про помилку. DisconnectUser відключає користувача від сервера шляхом виклику методу Disconnect з ідентифікатором ID і встановлює isUserConn на false.

GPTBot_btn_Click: Цей метод викликається при натисканні кнопки "GPTBot". Він створює нову форму для взаємодії з ботом "OpenAI", викликає метод OpenChildform для відкриття нової форми та викликає ResetListItem.

3.2.5 Налаштування персональних даних

Для того щоб користувач зміг прикріпити посилання на Github, LinkedIn або мав змогу вказати посаду в компанії, чи змінити фото профілю потрібна форма налаштувань SettingForm.

Інтерфейс SettingForm:

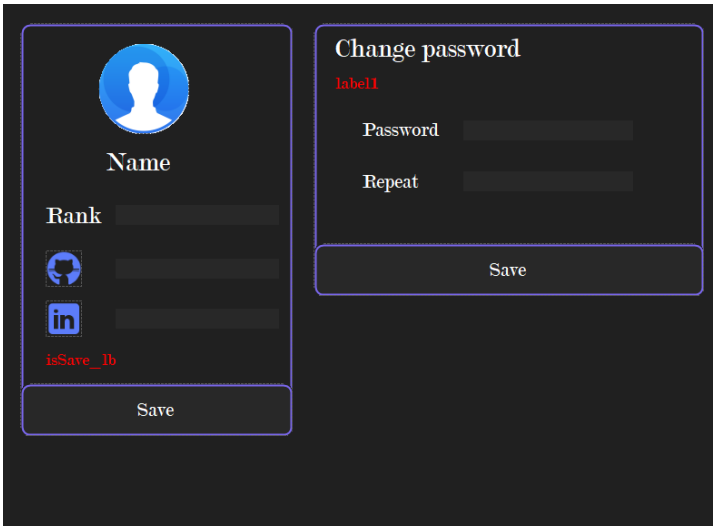


Рис. 3.31.

Код форми наступний:

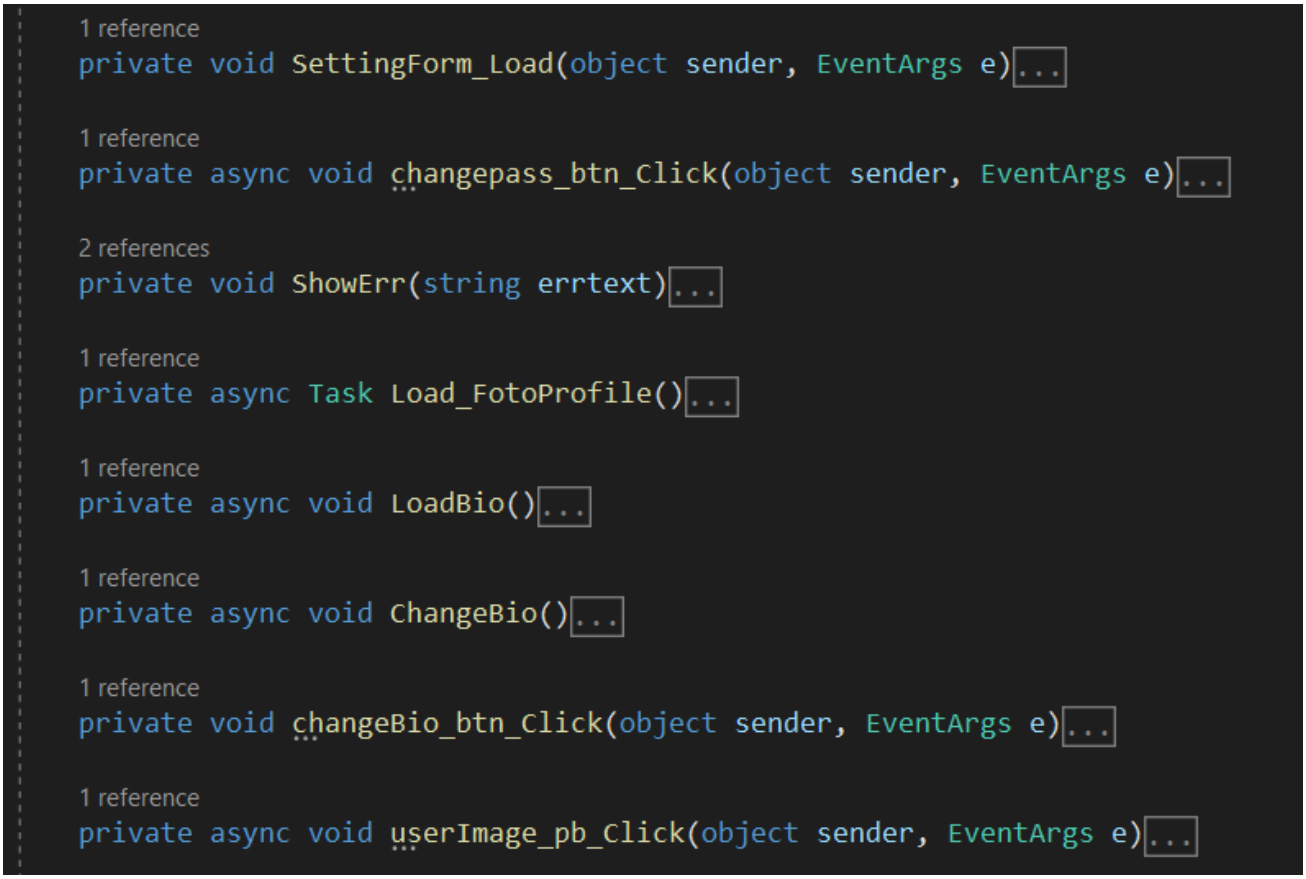


Рис. 3.32.

Опис методів SettingForm:

Метод SettingForm_Load викликається при завантаженні форми і приховує відображення деяких повідомлень та завантажує біографічні дані користувача.

Метод changepass_btn_Click виконує перевірку та зміну пароля користувача в базі даних, засновану на введеному новому паролі та перевірці його відповідності вимогам.

Метод ShowErr відображає помилки, пов'язані з введеними даними користувача.

Метод Load_FotoProfile завантажує фото профілю користувача з бази даних та відображає його на формі.

Метод LoadBio викликає метод Load_FotoProfile та заповнює решту біографічних даних користувача на формі.

Метод ChangeBio оновлює біографічні дані користувача в базі даних на основі введених даних на формі.

Метод changeBio_btn_Click викликає метод ChangeBio та відображає повідомлення про успішне оновлення біографічних даних.

Метод userImage_pb_Click відкриває діалогове вікно для вибору файлу зображення, відображає вибране зображення на формі, оновлює шлях до зображення в змінній Info.pathToFoto та завантажує зображення до бази даних.

3.2.6 Відображення та створення повідомлень

Для відображення повідомлень, включаючи текстові повідомлення, аудіофайли та файли було створено класс MessageBoxUser_ctrl який є підкласом UserControl.

```
17 references
public partial class MessageBoxUser_ctrl : UserControl
{
    PictureBox loadSignpicBox = new PictureBox();
    string pathtofile = (Application.StartupPath).Substring(0, (Application.StartupPath).IndexOf("AniChat"));
    2 references
    public Color BgColor { get; set; }
    6 references
    public MessageBoxUser_ctrl(string username, string time, DockStyle dock, Color bgcolor)...

    9 references
    public int MessID { get; set; }

    5 references
    public string FileName { get; set; }

    3 references
    public void Message(string text)...

    2 references
    private void AutoSizeTextMessBox(string txt, Font font)...

    // Event raised from RichTextBox when user clicks on a link:
    1 reference
    private void richTextBox_LinkClicked(object sender, LinkClickedEventArgs e)
    {
        LaunchWeblink(e.LinkText);
    }
```

Рис. 3.33. Методи класу MessageBoxUser_ctrl

```
private void LaunchWeblink(string url)
{
    if (IsHttpURL(url)) Process.Start(url);
}

// Simple check to make sure link is valid,
// can be modified to check for other protocols:
1 reference
private bool IsHttpURL(string url)...

3 references
public async void VoiceMessage(string filename)...

3 references
public void FileMessage(string filename)...

1 reference
private string GetFileSizeValue()...

1 reference
private string GetFileIconImage(string filename)...

1 reference
private async void LoadSignpicBox_MouseClick(object sender, MouseEventArgs e)...

2 references
private void SaveFile(string filePath, string directoryPath)...
```

Рис. 3.34. MessageBoxUser_ctrl

```
1 reference
private void loadSignpicBox_MouseEnter(object sender, EventArgs e)...

1 reference
private void loadSignpicBox_MouseLeave(object sender, EventArgs e)...
}
```

Рис. 3.35. MessageBoxUser_ctrl

Опис методів MessageBoxUser_ctrl:

Метод Message(string text):

- Створює екземпляр RichTextBox.
- Встановлює параметри тексту повідомлення.
- Додає RichTextBox до контейнера panel1.

Метод VoiceMessage(string filename):

- Завантажує аудіофайл із вказаного шляху.
- Створює екземпляр VoiceMessage для відтворення аудіо.
- Додає VoiceMessage до контейнера panel1.

Метод FileMessage(string filename):

- Створює елементи (зображення, розмір файлу, текст) для відображення файлу.
- Додає елементи до контейнера panel1.

Метод AutoSizeTextMessBox(string txt, Font font):

- Автоматично визначає розміри контейнера panel1 на основі розмірів тексту повідомлення.

Обробники подій:

- richTextBox_LinkClicked: виконує перехід за посиланням, якщо воно є в тексті повідомлення.
- LaunchWeblink(string url): відкриває посилання у веб-браузері.
- IsHttpURL(string url): перевіряє, чи є посилання валідним HTTP-посиланням.
- LoadSignpicBox_MouseClick: обробляє подію кліку на зображенні знаку завантаження.
- loadSignpicBox_MouseEnter та loadSignpicBox_MouseLeave: обробляють події наведення та зняття курсора з зображення знаку завантаження.

Окремої уваги потребує метод VoiceMessage.

```
3 references
public async void VoiceMessage(string filename)
{
    FileName = filename;//"test.mp3"

    string ext = Path.GetExtension(this.FileName);//".mp3"
    string finalpath = pathtofile + "temp\\" + Path.GetFileNameWithoutExtension(filename) + "_" + MessID.ToString();

    if (!File.Exists(finalpath + ext)) //"..\\test_MessID.mp3"
    {
        await Task.Run(() => SaveFile(finalpath, Path.GetDirectoryName(finalpath)));
    }

    VoiceMessage voiceMessage = new VoiceMessage(finalpath + ext);
    voiceMessage.Dock = DockStyle.Fill;
    voiceMessage.BackColor = BgColor;
    panel1.Width = 350;
    Height = 150;
    panel1.Controls.Add(voiceMessage);
    voiceMessage.BringToFront();
```

Рис. 3.36.

Метод VoiceMessage використовується для відтворення голосових повідомлень.

Параметр filename - це назва аудіофайлу, який потрібно відтворити.

Спочатку метод встановлює значення поля FileName вказаної назви файлу.

Далі отримує розширення файлу за допомогою функції Path.GetExtension, що повертає рядок, що містить розширення файлу (наприклад, ".mp3").

Далі проводиться перевірка, чи файл із таким шляхом і розширенням вже існує. Якщо файл не існує, запускається асинхронна задача (Task.Run) для збереження файлу на вказаний шлях. Для збереження файлу використовується метод SaveFile, який приймає шлях до файлу (finalpath) і шлях до папки, де файл буде збережений.

VoiceMessage є підкласом UserControl який реалізує інтерфейс та методи відтворення голосових повідомлень.

Інтерфейс VoiceMessage:

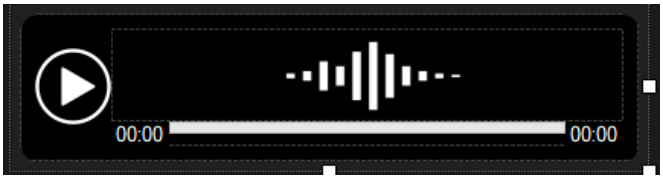


Рис. 3.37. Вигляд VoiceMessage

Код VoiceMessage:

```
using AxWMPLib;

namespace AniChat
{
    9 references
    public partial class VoiceMessage : UserControl
    {
        string pathtofileIcon = (Application.StartupPath).Substring(0, (Application.StartupPath).IndexOf("AniChat")) + "Images\\";
        AxWindowsMediaPlayer player = null;
        3 references
        public VoiceMessage(string path)
        {
            Path = path;
            InitializeComponent();
        }

        4 references
        public string Path { get; set; }

        1 reference
        private void timer1_Tick(object sender, EventArgs e)...

        1 reference
        private void playPause_btn_Click(object sender, EventArgs e)...

        1 reference
        private void Player_PlayStateChange(object sender, _WMPOCXEvents_PlayStateChangeEvent e)...

        1 reference
        private void timeline_pB_MouseDown(object sender, MouseEventArgs e)...
    }
```

Рис. 3.38. Методи класу VoiceMessage

Цей код реалізує логіку відтворення аудіофайлів за допомогою AxWindowsMediaPlayer та відображення прогресу відтворення на полосі прогресу. Також здійснюється керування анімацією звуку та кнопками відтворення/паузи.

3.2.7 Форма з вибраним контактом

Створимо форму чату ChatForm для прийому та відправки повідомленнь між контактами.

Вона буде мати наступний вигляд:

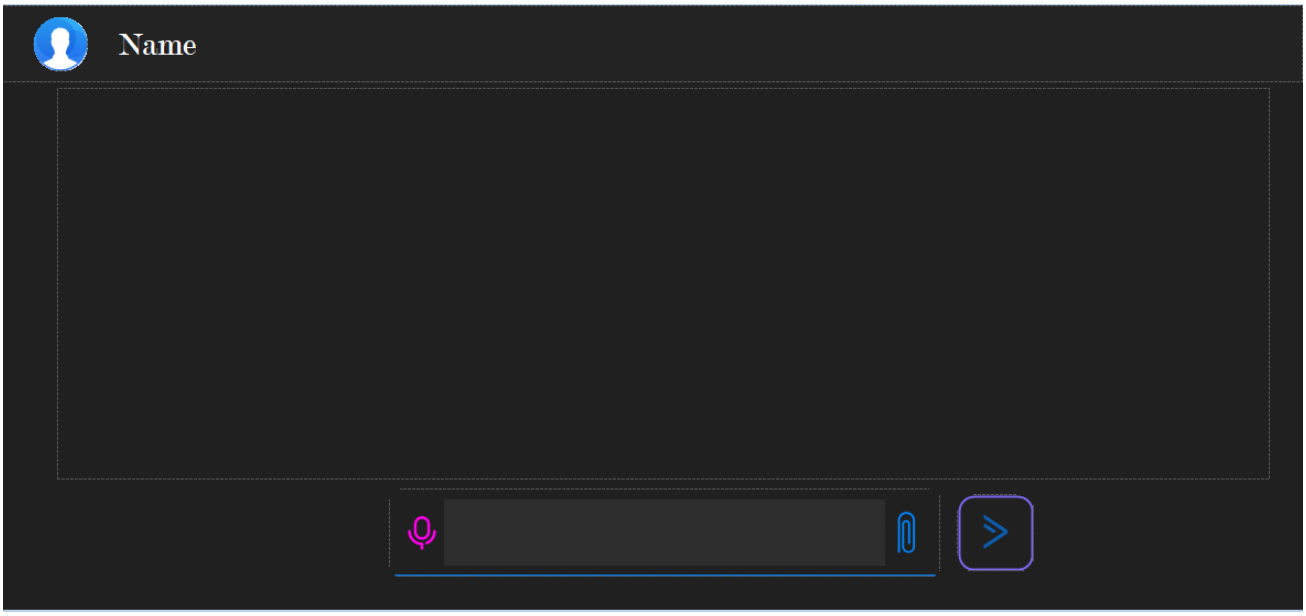


Рис. 3.39. Вигляд вікна чату

Ця форма містить поле для відправки повідомлень, а саме текстових, голосових та відправка файлів. Також на формі передбачена подія DragAndDrop – перетягування фалів в вікно чату за домомогою мишки.

Код ChatForm форми має наступний вигляд:

```
2 references
public ChatForm(Image image, string name, int contact_id)...

1 reference
private void Msgbox_panel_DragDrop(object sender, DragEventArgs e)...

1 reference
private void Msgbox_panel_DragOver(object sender, DragEventArgs e)...

2 references
private void Msgbox_panel_DragLeave(object sender, EventArgs e)...

1 reference
private void Msgbox_panel_DragEnter(object sender, DragEventArgs e)...

1 reference
private async void ChatForm1_Load(object sender, EventArgs e)...
1 reference
public void MsgCallback(string msg)...

2 references
private void ScrolltoLastItem()...

2 references
private void AddMess(int messID ,string time, string username, string textmess, string filename)...

2 references
private void CtrMessProp(string textmess, string filename, MessageBoxUser_ctrl msgBox)...

1 reference
public void ConnectUser()...
```

Рис. 3.40. Методи ChatForm

```
2 references
public void DisconnectUser()...

1 reference
private void MsgTextbox_KeyDown(object sender, KeyEventArgs e)...

1 reference
private void btn_Send_Click(object sender, EventArgs e)...

1 reference
private void btnSendFile_Click(object sender, EventArgs e)...

3 references
private void SendFile(string path)...

1 reference
private string OpenFile()...

1 reference
private void ChatForm1_FormClosed(object sender, FormClosedEventArgs e)...

2 references
private int SetHistory(string text,string pathToFile, string time)...

1 reference
private async Task GetHistory()...

1 reference
private async void Msgbox_panel_SizeChanged(object sender, EventArgs e)...

1 reference
private void microphone_btn_Click(object sender, EventArgs e)...

1 reference
private void stop_btn_Click(object sender, EventArgs e)...

1 reference
private void Cansel_btn_Click(object sender, EventArgs e)...
```

Рис. 3.41. Методи ChatForm

```
2 references
private void Set_default_msgpanel_view()...

1 reference
private void timer1_Tick(object sender, EventArgs e)...
```

Рис. 3.42. Методи ChatForm

Короткий опис методів ChatFrom:

ChatForm: Конструктор класу, який ініціалізує компоненти форми, встановлює зображення профілю користувача, ім'я контакту та розміри панелі повідомлень. Також додає обробники подій для перетягування файлів у вікно чату.

Методи Msgbox_panel_DragDrop, Msgbox_panel_DragOver, Msgbox_panel_DragLeave, Msgbox_panel_DragEnter: Ці методи відповідають за обробку подій перетягування файлів до вікна чату. Вони дозволяють користувачу перетягувати файли у вікно чату, отримувати шляхи до цих файлів і викликати метод SendFile для кожного файлу.

ChatForm1_Load: Цей метод викликається при завантаженні форми чату. Він встановлює деякі властивості елементів форми, підключає користувача до сервера, завантажує історію повідомлень (якщо користувач ще не підключений) і відображає необхідні елементи форми.

MsgCallback: Цей метод викликається ззовні і отримує повідомлення msg. Він аналізує повідомлення у форматі JSON і викликає метод AddMess для додавання нового повідомлення до панелі повідомлень. Після цього викликається метод ScrolltoLastItem для прокручування до останнього повідомлення.

ScrolltoLastItem: Цей метод прокручує панель повідомлень до останнього елемента.

AddMess: Цей метод додає нове повідомлення до панелі повідомлень. Він створює екземпляр MessageBoxUser_ctrl залежно від власника повідомлення і встановлює необхідні властивості, такі як текст повідомлення, час, власник, тип повідомлення (текстове, голосове або файлове) та інше.

ConnectUser і DisconnectUser: Ці методи відповідають за підключення та відключення користувача до/від сервера. Метод ConnectUser викликає метод Connect клієнта з ім'ям користувача та назвою чату, які передаються серверу. Метод DisconnectUser викликає метод Disconnect клієнта з ідентифікатором підключення та назвою чату для відключення користувача від сервера.

MsgTextbox_KeyDown: Цей метод обробляє подію натискання клавіші в полі введення повідомлення. Якщо натиснуті клавіші Shift + Enter, то викликається подія кнопки btn_Send_Click.

btn_Send_Click: Цей метод викликається при натисканні кнопки "Відправити". Якщо встановлено прапорець isVoiseMsg, то викликається метод SendFile для відправлення голосового повідомлення. В іншому випадку, перевіряється, чи введено текстове повідомлення, і якщо так, то створюється JSON-рядок повідомлення і викликається метод SendMsg сервера для відправлення повідомлення. Після цього очищається поле введення повідомлення.

btnSendFile_Click: Цей метод викликається при натисканні кнопки "Відправити файл". Викликається метод OpenFile, який відкриває діалогове вікно вибору файлу. Після вибору файлу викликається метод SendFile для відправлення файлу.

SendFile: Цей метод відправляє файл на сервер. Викликається метод SetHistory для збереження історії повідомлень і створення JSON-рядка повідомлення. Потім викликається метод SendMsg сервера для відправлення повідомлення.

OpenFile: Цей метод відкриває діалогове вікно вибору файлу і повертає шлях до обраного файлу.

ChatForm1_FormClosed: Цей метод викликається при закритті форми чату. Він оновлює кількість нових повідомлень для контакту в базі даних, відключає користувача від сервера і оновлює прапорець isUserConn.

SetHistory: Цей метод зберігає історію повідомлень в базі даних. Викликається SQL-запит для вставки запису у таблицю chat_history і оновлення кількості нових повідомлень для контакту. Якщо вказано шлях до файлу, викликається метод UploadFileToDB для завантаження файлу до бази даних.

GetHistory: Цей метод отримує історію повідомлень з бази даних. Викликається SQL-запит для отримання даних з таблиці chat_history і додається кожне повідомлення до панелі повідомлень у формі.

Msgbox_panel_SizeChanged: Цей метод обробляє подію зміни розміру панелі повідомлень (Msgbox_panel). Виконується асинхронна операція, яка використовує Task.Run та Parallel.ForEach, для зміни ширини кожного контролера MessageBoxUser_ctrl всередині Msgbox_panel. Після зміни ширини викликається метод PerformLayout для оновлення розміщення контролерів.

microphone_btn_Click: Цей метод викликається при натисканні кнопки "Мікрофон". Він видаляє кнопку "Відправити файл" і кнопку "Мікрофон" з message_pl, створює нові елементи для запису голосового повідомлення, розпочинає запис голосу та запускає таймери для відображення часу запису.

stop_btn_Click: Цей метод викликається при натисканні кнопки "Стоп" під час запису голосового повідомлення. Зупиняє запис голосу, зупиняє таймери, видаляє кнопку "Стоп" та додає кнопку "Скасувати".

Cansel_btn_Click: Цей метод викликається при натисканні кнопки "Скасувати". Він повертає форму до початкового стану, очищає контролери та відновлює можливість введення тексту.

Set_default_msgpanel_view: Цей метод встановлює початковий вигляд панелі повідомлень. Він очищає контролери, вимикає режим голосового повідомлення та відновлює кнопки для відправки повідомлень та файлів.

timer1_Tick: Цей метод викликається при кожному тіку таймера timer1. Оновлює відображення поточного часу запису голосу на екрані.

3.2.8 Форма створення групи користувачів

Форма AddNewGroupForm відповідає за додавання нової групи в програмі. Основна функціональність полягає в завантаженні контактів з бази даних, виборі учасників групи, виборі зображення для групи та створенні нової групи з відповідними учасниками.

Інтерфейс форми AddNewGroupForm:

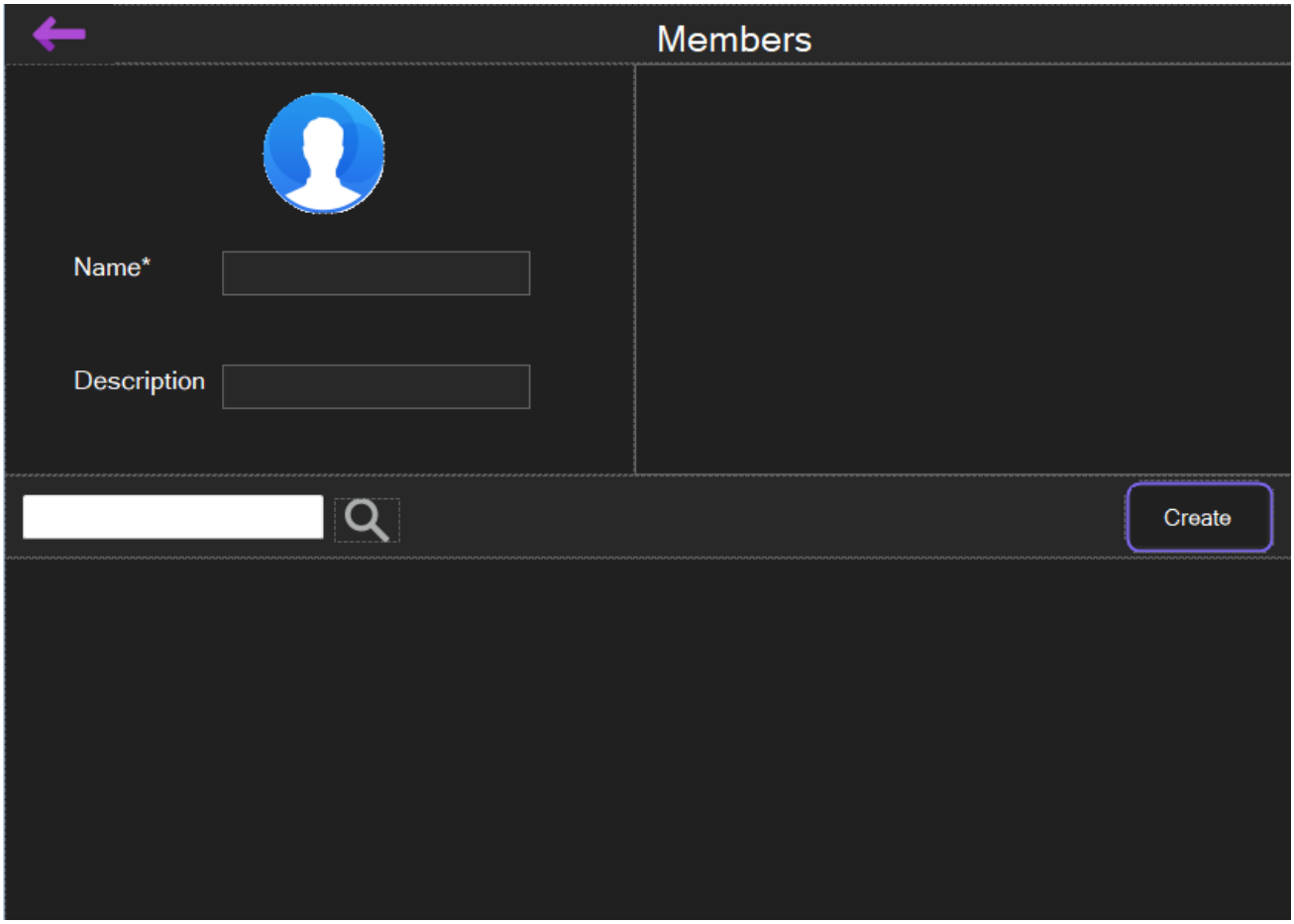


Рис. 3.43. Вигляд форми створення групи

Код форми AddNewGroupForm:

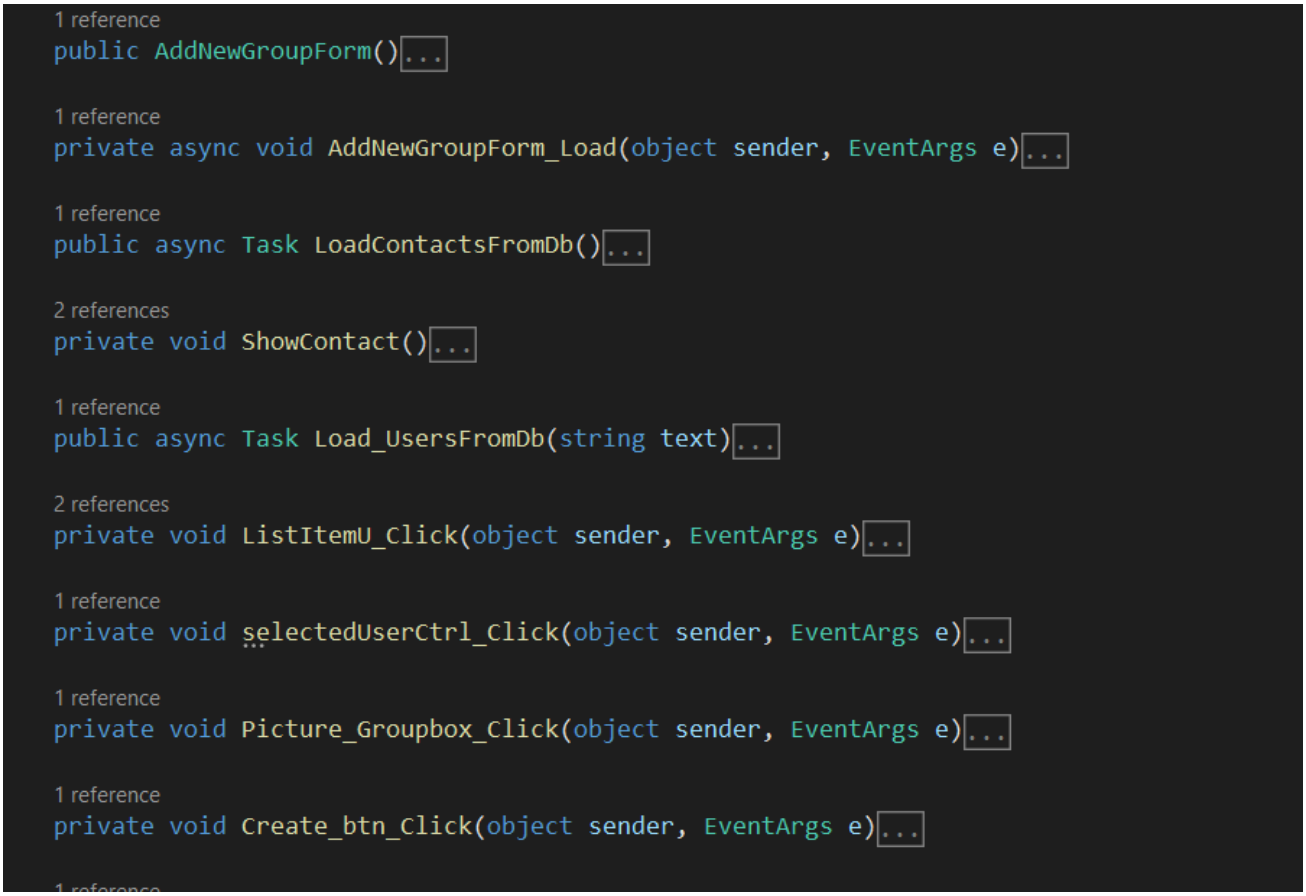


Рис. 3.44. Методи класу AddNewGroupForm

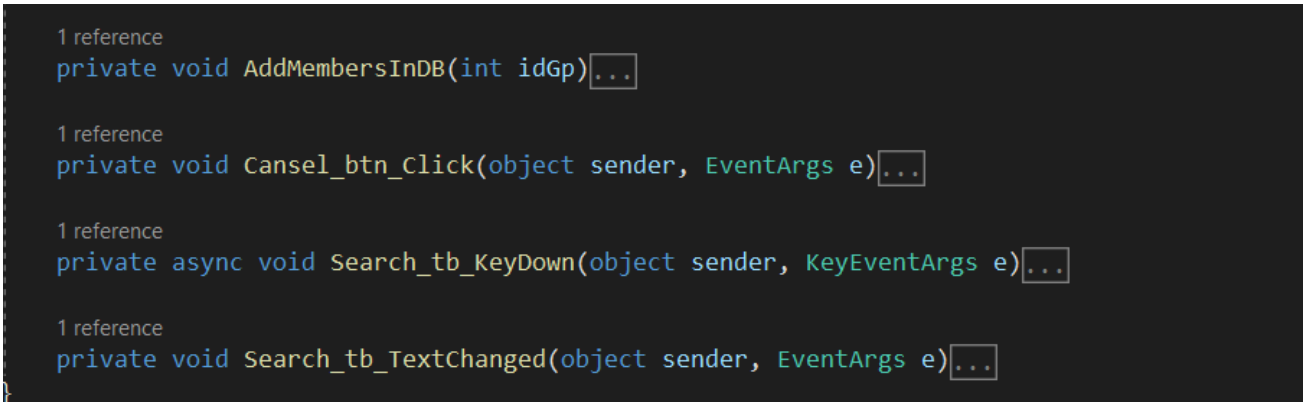


Рис. 3.45. Методи класу AddNewGroupForm

Опис методів AddNewGroupForm:

Метод AddNewGroupForm_Load викликається при завантаженні форми та викликає метод LoadContactsFromDb для завантаження контактів користувача з бази даних.

Метод LoadContactsFromDb завантажує контакти користувача з бази даних та створює об'єкти UserListItem для кожного контакту. Далі, вони додаються до списку contacts та відображаються на формі.

Метод ShowContact відображає контакти на формі в Members_tLPanel.

Метод Load_UsersFromDb завантажує користувачів з бази даних на основі пошукового тексту та відображає їх на формі.

Метод ListItemU_Click викликається при кліку на елемент UserListItem (контакт) та додає обраних учасників до списку вибраних учасників AddedUser_fLPanel.

Метод selectedUserCtrl_Click викликається при кліку на кнопку видалення обраного учасника зі списку AddedUser_fLPanel.

Метод Picture_Groupbox_Click викликається при кліку на Picture_Groupbox (область зображення групи) та відкриває діалогове вікно для вибору зображення для групи.

Метод Create_btn_Click викликається при натисканні кнопки "Створити" та створює нову групу в базі даних з вибраними учасниками та зображенням.

Метод AddMembersInDB додає учасників групи до бази даних на основі списку вибраних учасників з AddedUser_fLPanel.

Метод Cansel_btn_Click викликається при натисканні кнопки "Відміна" та закриває форму.

Метод Search_tb_KeyDown викликається при натисканні клавіші Enter в полі пошуку та завантажує користувачів з бази даних на основі введеного тексту.

Метод Search_tb_TextChanged викликається при зміні тексту в полі пошуку та відображає всі контакти, якщо поле пошуку порожнє.

Також створимо форму ChatGroupForm для реалізації прийому та відправки повідомлень в групах. В загальному форма ChatGroupForm реалізує всі основні методи відправки та прийому повідомлень які ми могли бачити в формі ChatForm але реалізація цих методів відрізняється, а також додані нові методи, пов'язані з відображенням та керуванням учасниками групи чату.

```
1 reference
private void btnMenu_Click(object sender, EventArgs e)...

3 references
public List<SelectedUserCtrl> LoadMember(string strSql)...

1 reference
public void ShowMembers()...

1 reference
private void add_swich_lb_Click(object sender, EventArgs e)...

1 reference
private void AddMbrsBtn_Click(object sender, EventArgs e)...

1 reference
private void delete_swich_lb_Click(object sender, EventArgs e)...

1 reference
private void RemMbrsBtn_Click(object sender, EventArgs e)...

1 reference
private void delete_gp_btn_Click(object sender, EventArgs e)...
```

Рис. 3.46. Методи класу ChatGroupForm

Короткий опис даних методів:

btnMenu_Click: Цей метод викликається при натисканні кнопки меню групи чату. Він перевіряє стан видимості панелі меню (GroupMenupl) та адмінської панелі (AdminPanel) і відповідно встановлює їх видимість. Також завантажує учасників групи за допомогою методу LoadMember і відображає їх за допомогою методу ShowMembers.

LoadMember: Цей метод завантажує учасників групи з бази даних на основі SQL-запиту, створює об'єкти SelectedUserCtrl для кожного учасника та повертає список цих об'єктів.

ShowMembers: Цей метод очищує панель MemberTLPanel і додає об'єкти SelectedUserCtrl до панелі для відображення учасників групи.

add_swich_lb_Click: Цей метод викликається при натисканні лівої вкладки "Додати" в меню групи. Він встановлює видимість потрібних панелей (underscore_Add_pl, underscore_Delete_pl) і завантажує учасників, які не входять до поточної групи, за допомогою методу LoadMember. Далі відображає цих учасників на панелі MemberTLPanel, додаючи кнопку "Видалити" для кожного учасника.

delete_swich_lb_Click: Цей метод викликається при натисканні правої вкладки "Видалити" в меню групи. Він встановлює видимість потрібних панелей (underscore_Add_pl, underscore_Delete_pl) і завантажує учасників, які входять до поточної групи, за допомогою методу LoadMember. Далі відображає цих учасників на панелі MemberTLPanel, додаючи кнопку "Видалити" для кожного учасника.

RemMbrsBtn_Click: Цей метод викликається при натисканні кнопки "Видалити" для учасника, який входить до поточної групи. Він виконує SQL-запит для видалення учасника з поточної групи та видаляє його з панелі MemberTLPanel.

delete_gp_btn_Click: Цей метод викликається при натисканні кнопки "Видалити групу" або "Вийти з групи". Він перевіряє, чи користувач є адміністратором групи і виконує відповідний SQL-запит для видалення групи з бази даних або виходу користувача з групи. Після цього закриває форму чату.

3.2.9 Бот ChatGPT

Оскільки месенджер розроблений для IT компаній то було розроблено форму BotsForm для підключення ботів та переписки з ними. Дана форма була розроблена для підключення ChatGPT від OpenAI. Основна ідея полягала у зручному та швидкому доступі до штучного інтелекту для працівників IT фірм які використовують AniChat, як корпоративний месенджер фірми.

Для того щоб підключити ChatGPT створемо клас ChatGPTClient, який є клієнтом для взаємодії з API ChatGPT.

```
internal class ChatGPTClient
{

    private readonly string _apiKey;
    private readonly RestClient _client;

    // Constructor that takes the API key as a parameter
    1 reference
    public ChatGPTClient(string apiKey)
    {
        _apiKey = apiKey;
        // Initialize the RestClient with the ChatGPT API endpoint
        _client = new RestClient("https://api.openai.com/v1/engines/text-davinci-003/completions");
    }

    // Method to send a message to the ChatGPT API and return the response
    1 reference
    public string SendMessage(string message)
    {
        try
        {
            // Create a new POST request
            var request = new RestRequest("", Method.Post);
            // Set the Content-Type header
            request.AddHeader("Content-Type", "application/json");
            // Set the Authorization header with the API key
            request.AddHeader("Authorization", $"Bearer {_apiKey}");

            // Create the request body with the message and other parameters
            var requestBody = new
            {
                prompt = message,
                max_tokens = 100,
                n = 1,
                stop = (string)null,
                temperature = 0.7,
            };

            // Add the JSON body to the request
            request.AddJsonBody(JsonConvert.SerializeObject(requestBody));

            // Execute the request and receive the response
            var response = _client.Execute(request);

            // Deserialize the response JSON content
            var jsonResponse = JsonConvert.DeserializeObject<dynamic>(response.Content ?? string.Empty);

            // Extract and return the chatbot's response text
            return jsonResponse?.choices[0]?.text?.ToString()?.Trim() ?? string.Empty;
        }
        catch (Exception ex)
        {
            // Handle any exceptions that may occur during the API request
```

Рис. 3.47. Методи класу ChatGPTClient

Опис методів класу ChatGPTClient:

RestClient _client: Поле для створення екземпляру RestClient, який використовується для відправки HTTP-запитів до API.

ChatGPTClient(string apiKey): Конструктор класу, який приймає API-ключ як параметр і ініціалізує поля _apiKey і _client.

string SendMessage(string message): Метод для надсилання повідомлення до API ChatGPT і отримання відповіді. Метод приймає повідомлення як параметр і повертає рядок з відповіддю чатбота.

У методі виконуються наступні кроки:

- Створюється новий POST-запит.
- Встановлюються заголовки Content-Type і Authorization.
- Створюється тіло запиту з повідомленням та іншими параметрами.
- Додається JSON-тіло до запиту.
- Виконується запит до API і отримується відповідь.
- Розпаковується JSON-відповідь і отримується текст відповіді чатбота.
- Повертається текст відповіді чатбота.

У разі виникнення помилки під час виконання запиту, метод перехоплює виключення, обробляє його і повертає повідомлення про помилку разом зі стандартним повідомленням про помилку.

3.2.10 Спливаючі повідомлення

Якщо основна форма згорнута або на основній формі прихована панель контактів та груп, користувачу потрібно надавати інформацію про нове повідомлення іншим методом, а саме створення вікна сповіщення яке появляється в нижньому правому куті екрану.

Інтерфейс FormAlert:

Рис. 3.48. Вигляд спливаючого повідомлення

Код форми FormAlert:

```
7 references
public partial class FormAlert : Form
{
    1 reference
    public FormAlert()
    {
        InitializeComponent();
    }

    8 references
    public enum EnmAction
    {
        wait,
        start,
        close
    }

    private int x, y;
    private FormAlert.EnmAction action;

    1 reference
    public void ShowAlert(string msg)...

    1 reference
    private void close_btn_Click(object sender, EventArgs e)...

    1 reference
    private void timer1_Tick(object sender, EventArgs e)...
}
```

Рис. 3.49.

Опис методів FormAlert:

- Перерахунок EnmAction визначає дії, які можуть виконуватися з формою спливаючого повідомлення.
- Змінні x і y відповідають за координати форми на екрані.
- Метод ShowAlert відображає спливаюче повідомлення з переданим текстом. Використовуючи цикл, він знаходить доступну назву для форми і встановлює її положення на екрані. Після цього встановлюються текст повідомлення і форма показується на екрані.
- Метод close_btn_Click викликається при натисканні кнопки "Закрити" і встановлює дію EnmAction.close, щоб почати процес закриття форми.

- Метод timer1_Tick викликається під час інтервалу таймера і виконує різні дії залежно від значення action. В залежності від дії, форма змінює прозорість, положення та закривається.

Цей код дозволяє створити спливаючі повідомлення з анімацією для відображення користувачеві важливих повідомлень чи сповіщень.

3.3. Тестування проекту

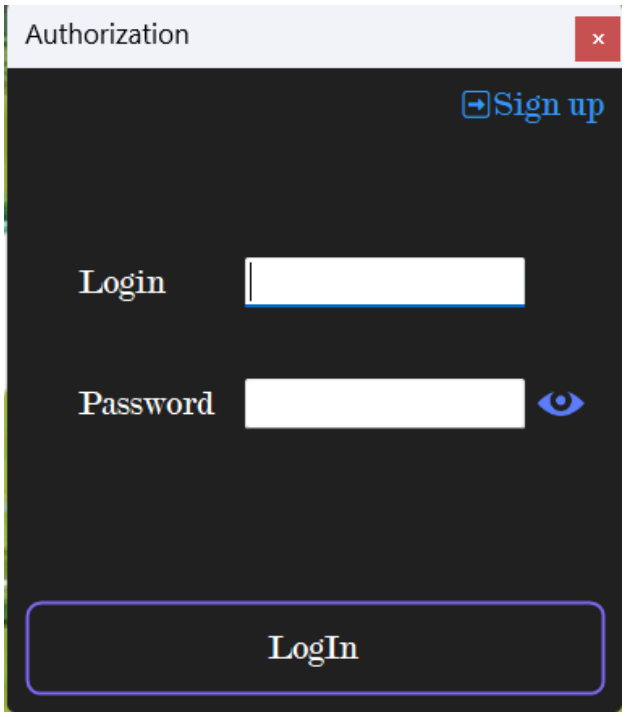


Рис. 3.50. Форма «Authorization» для входу

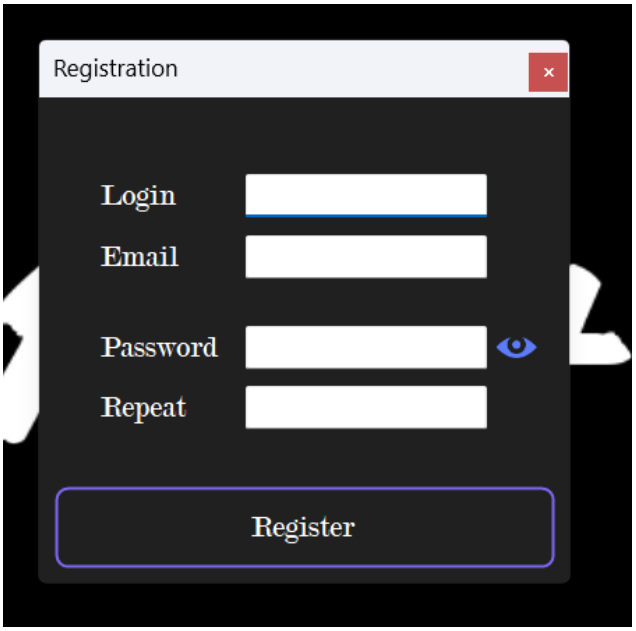


Рис. 3.51. Форма «Registration» для реєстрації

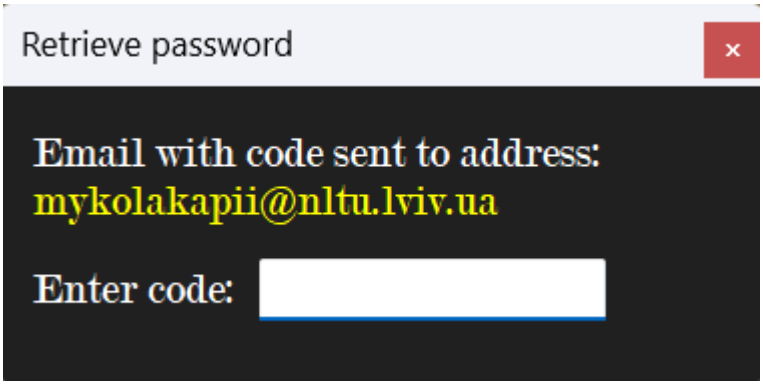


Рис. 3.52. Форма «Retrieve password» для введення коду відновлення

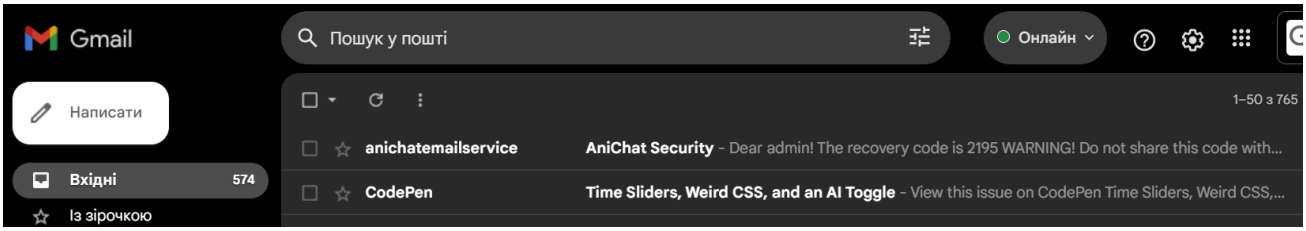


Рис. 3.53. Пошта на яку прийшов лист від AniChat Security

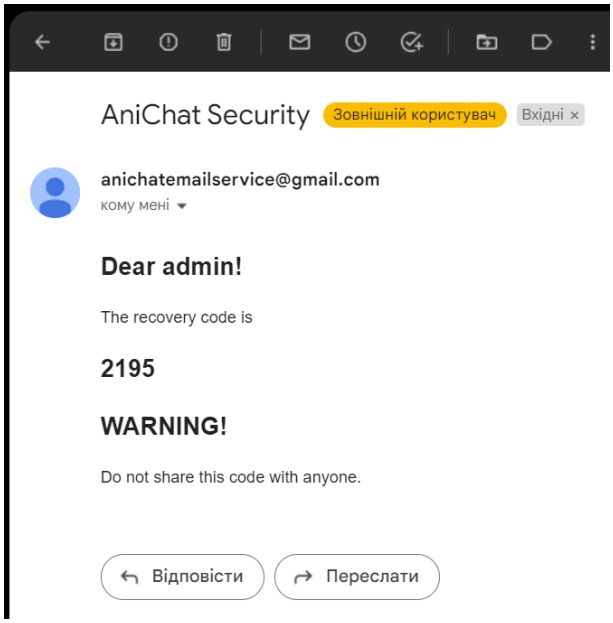


Рис. 3.54. Вигляд електронного листа з кодом

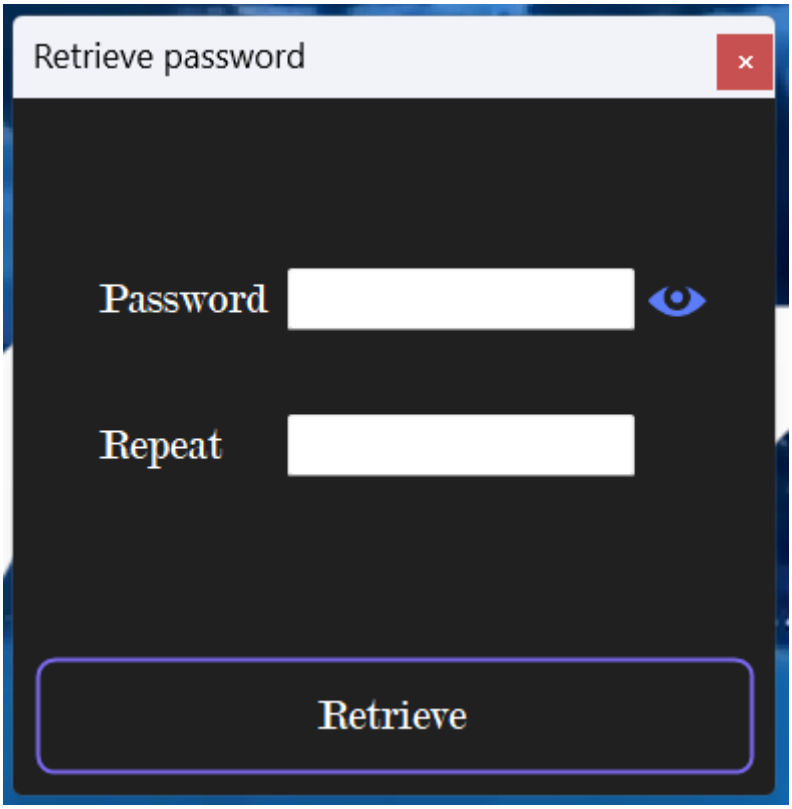


Рис. 3.55. Форма «Retrieve password» після введення коду з пошти

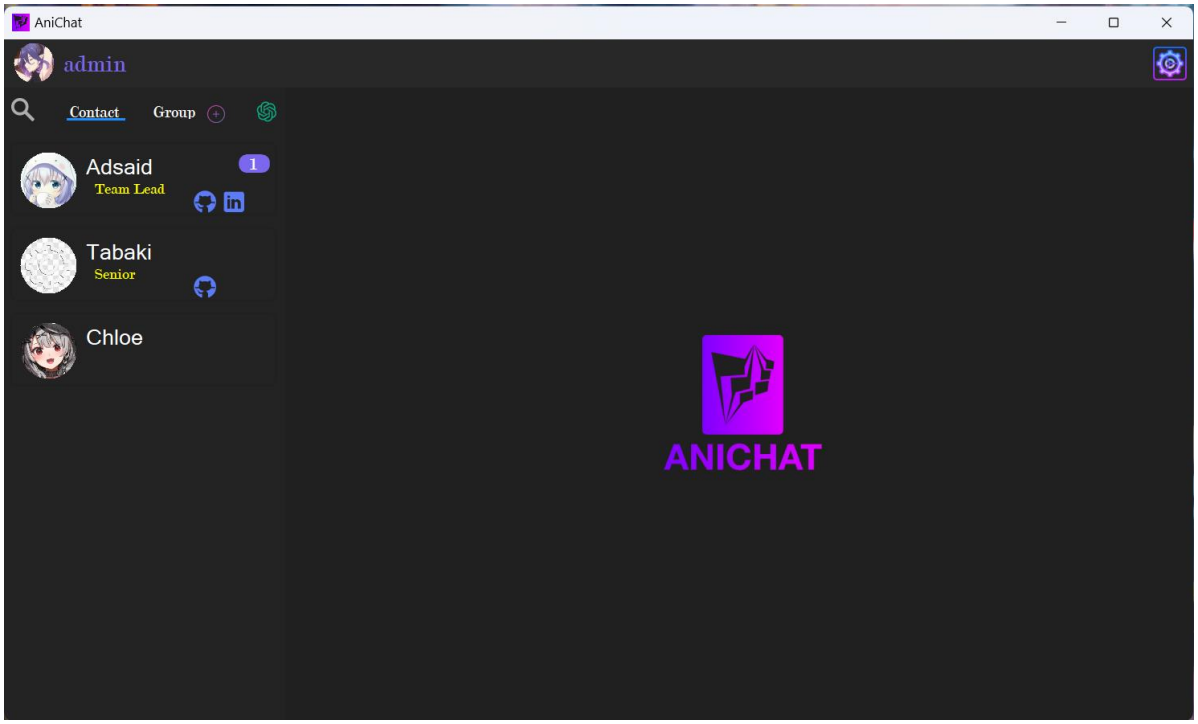


Рис. 56. Форма «AniChat» на якій відображено контакти користувача admin

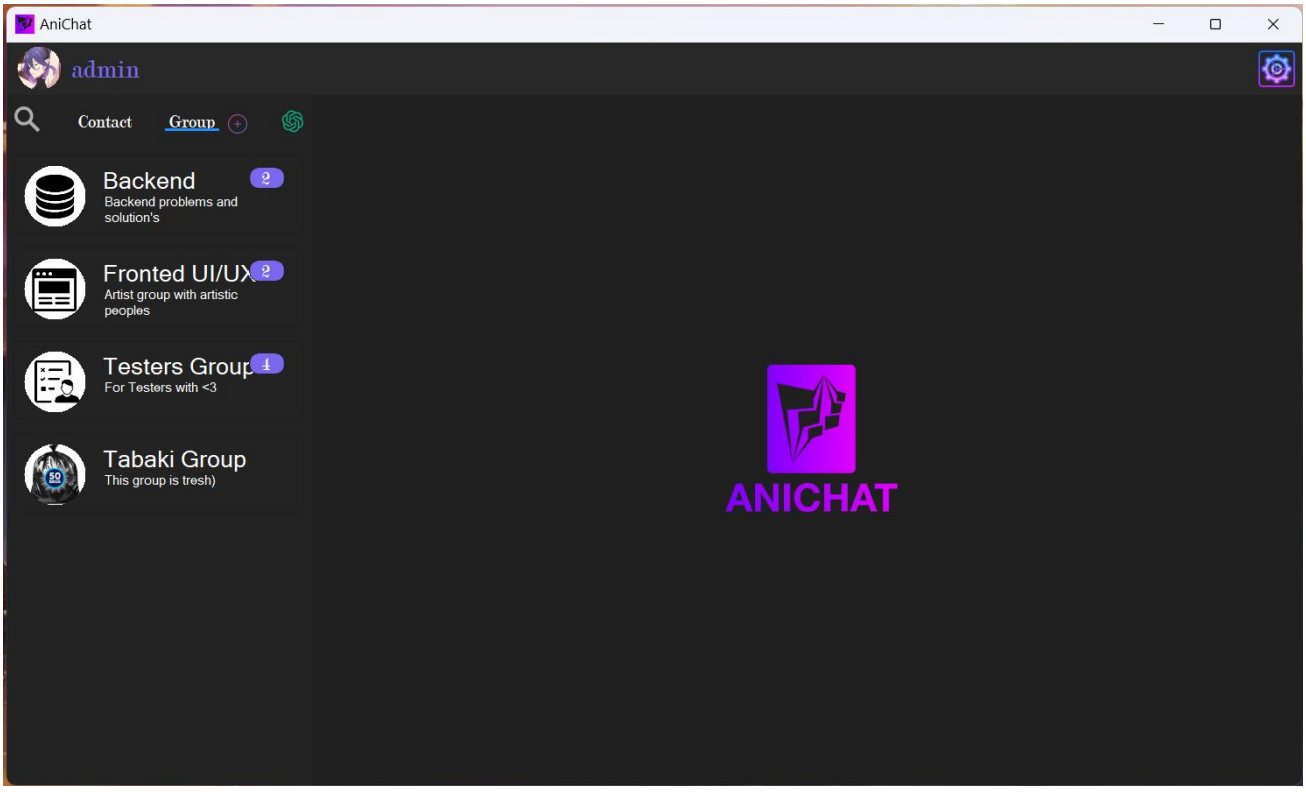


Рис. 57. Форма «AniChat» на якій відображено групи в яких перебуває admin

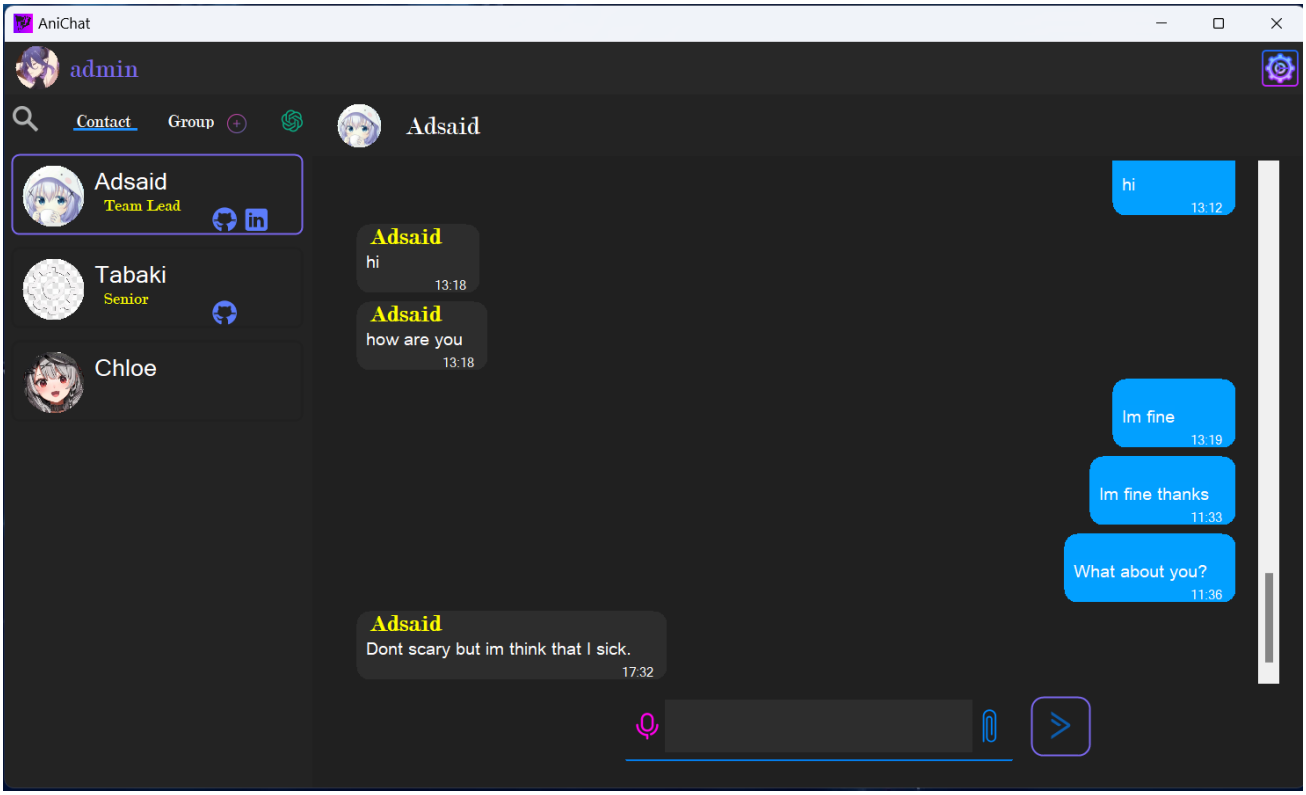


Рис. 3.58. Форма «AniChat» відкритий чат з користувачем Adsaid

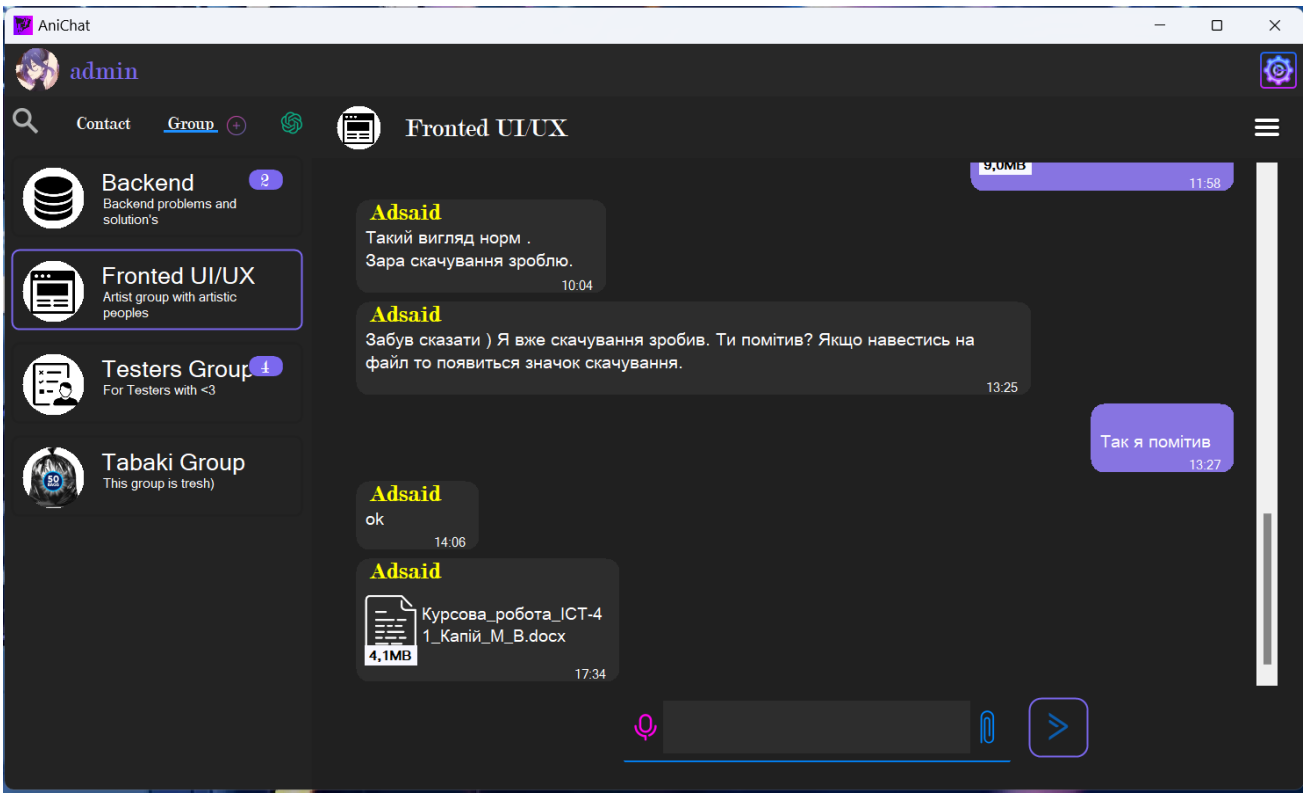


Рис. 3.59. Форма «AniChat» з відкритим чатом групи Frontend UI/UX

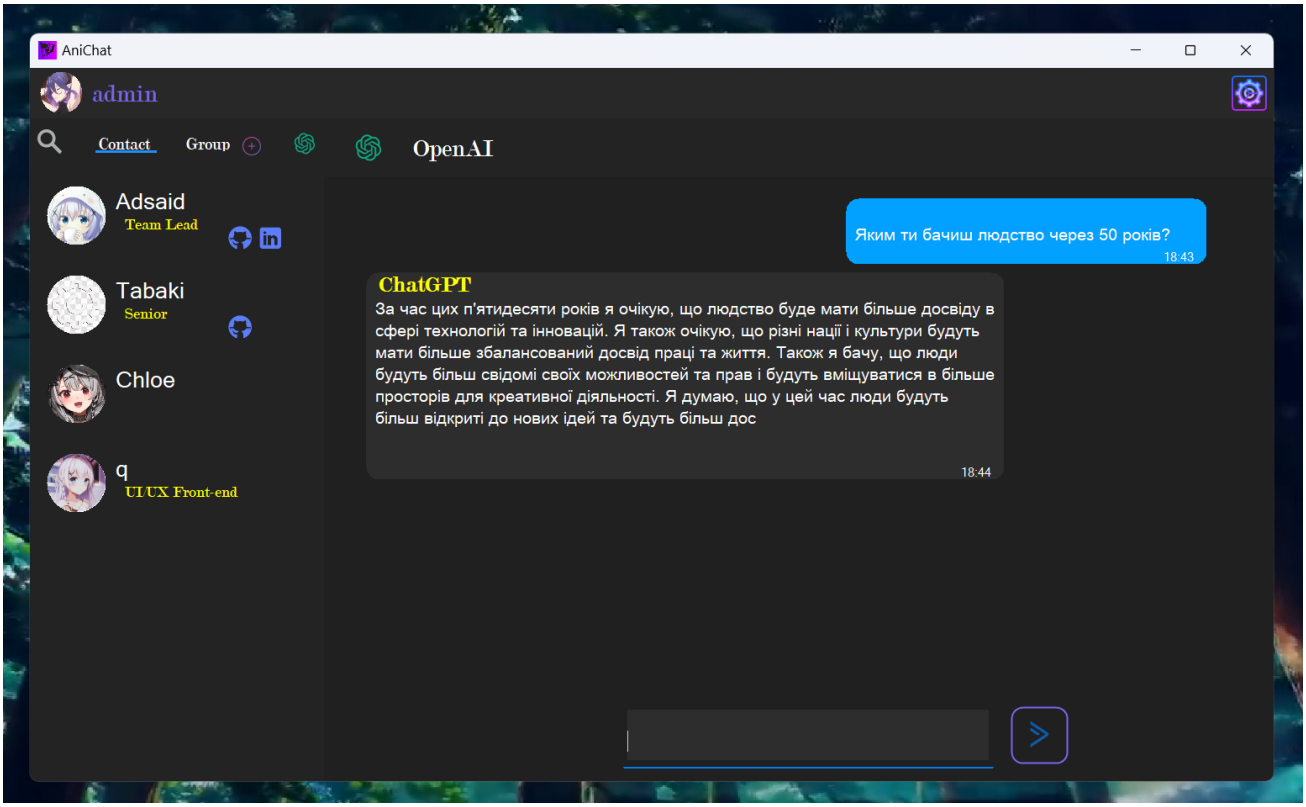


Рис. 3.60. Форма «AniChat» з відкритим чатБотом ChatGPT

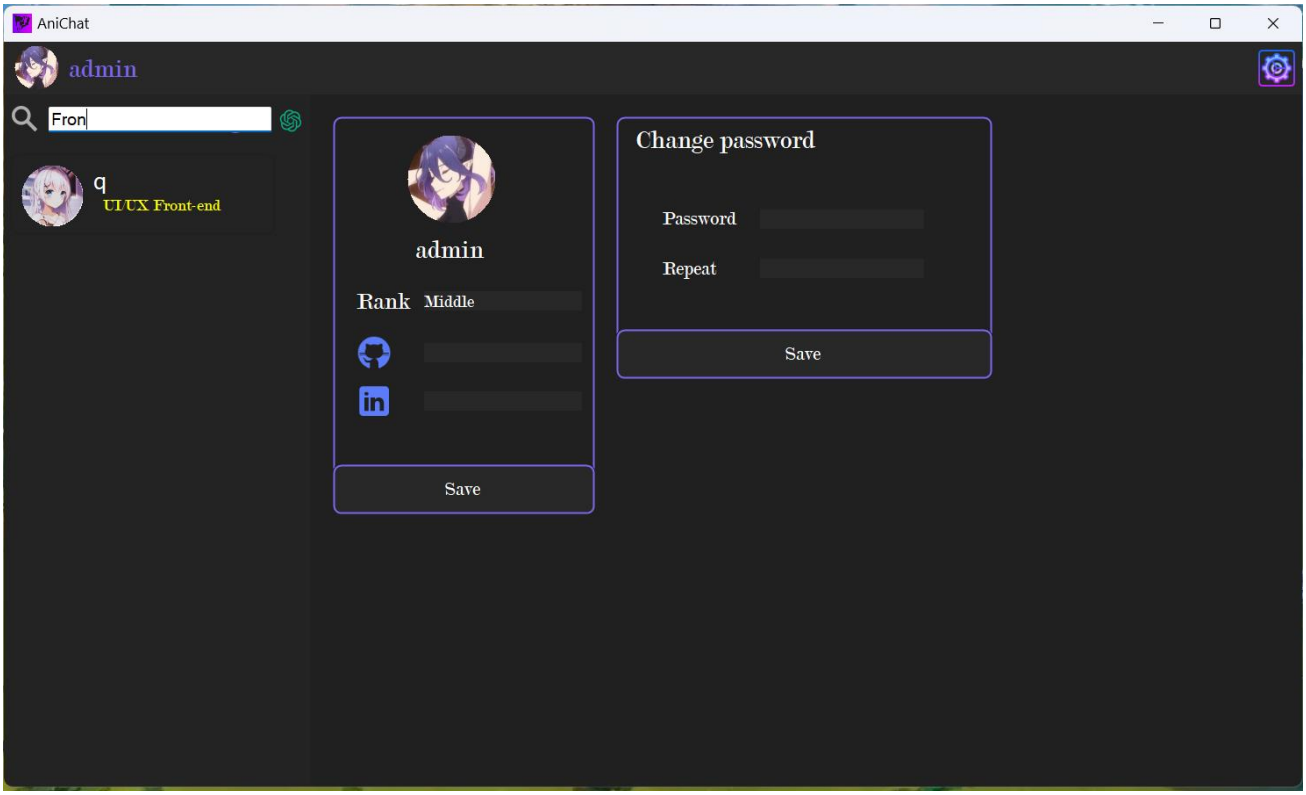


Рис. 3.61. Форма «AniChat» з відкритими налаштуваннями користувача admin та пошук контактів.

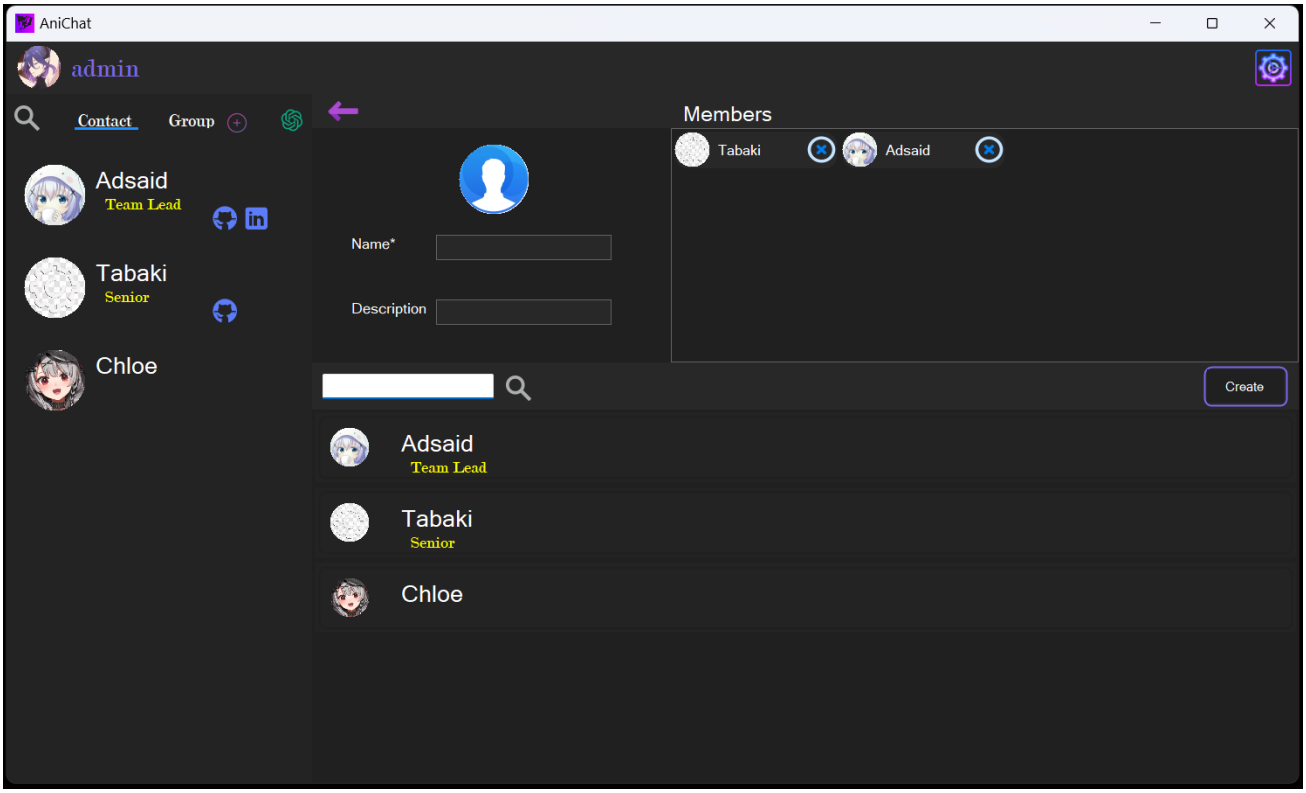


Рис. 3.62. Форма «AniChat» створення нової групи користувача admin

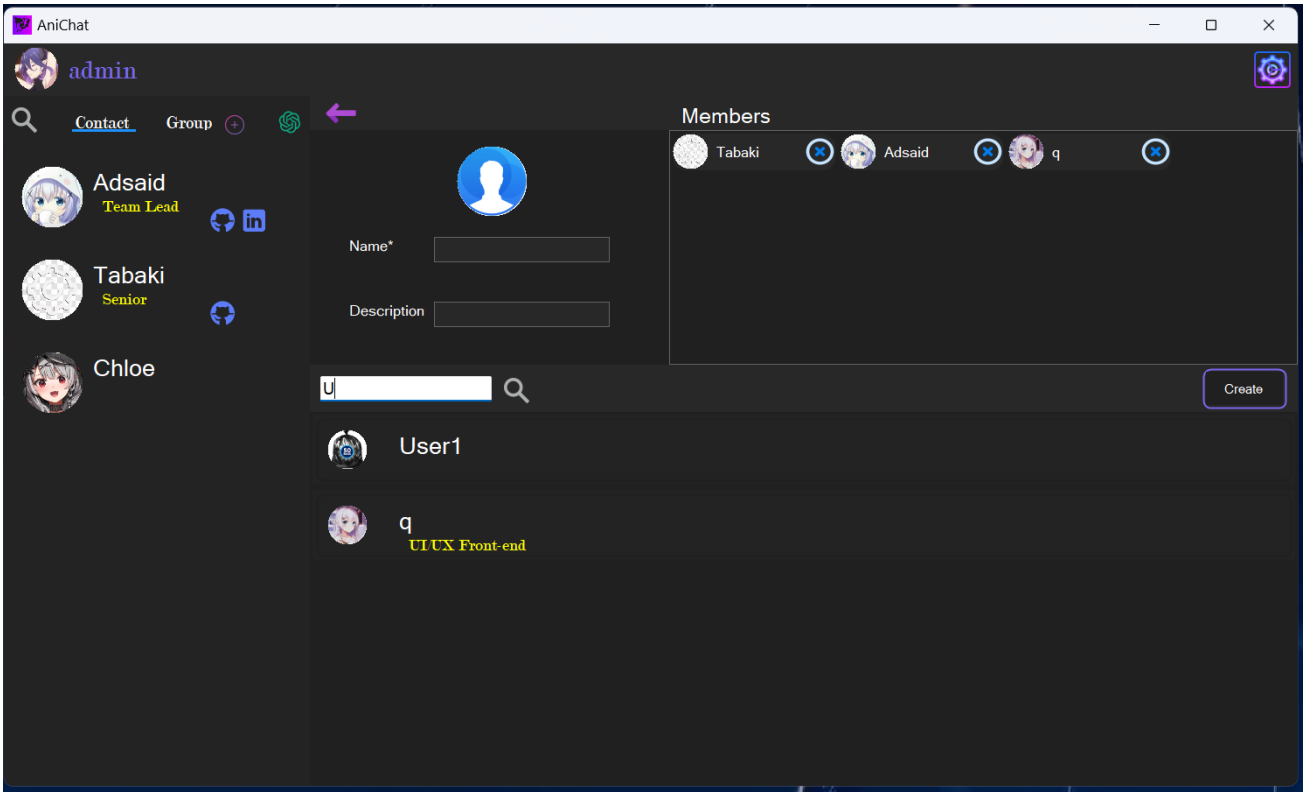


Рис. 3.63. Форма «AniChat» пошук користувачів для додавання їх у групу.

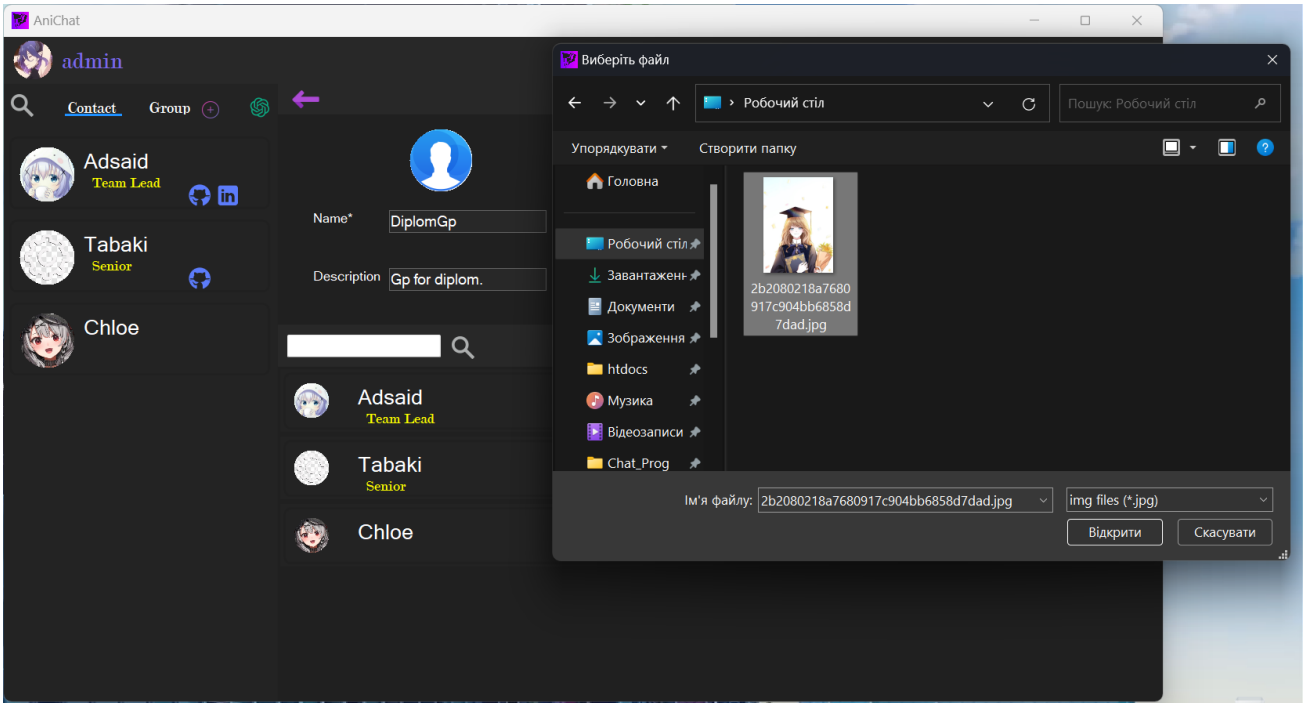


Рис. 3.64. Форма «AniChat» вибір зображення для чату та заповннення поля Імя(Name – обовязкове поле) та Опис(Description)

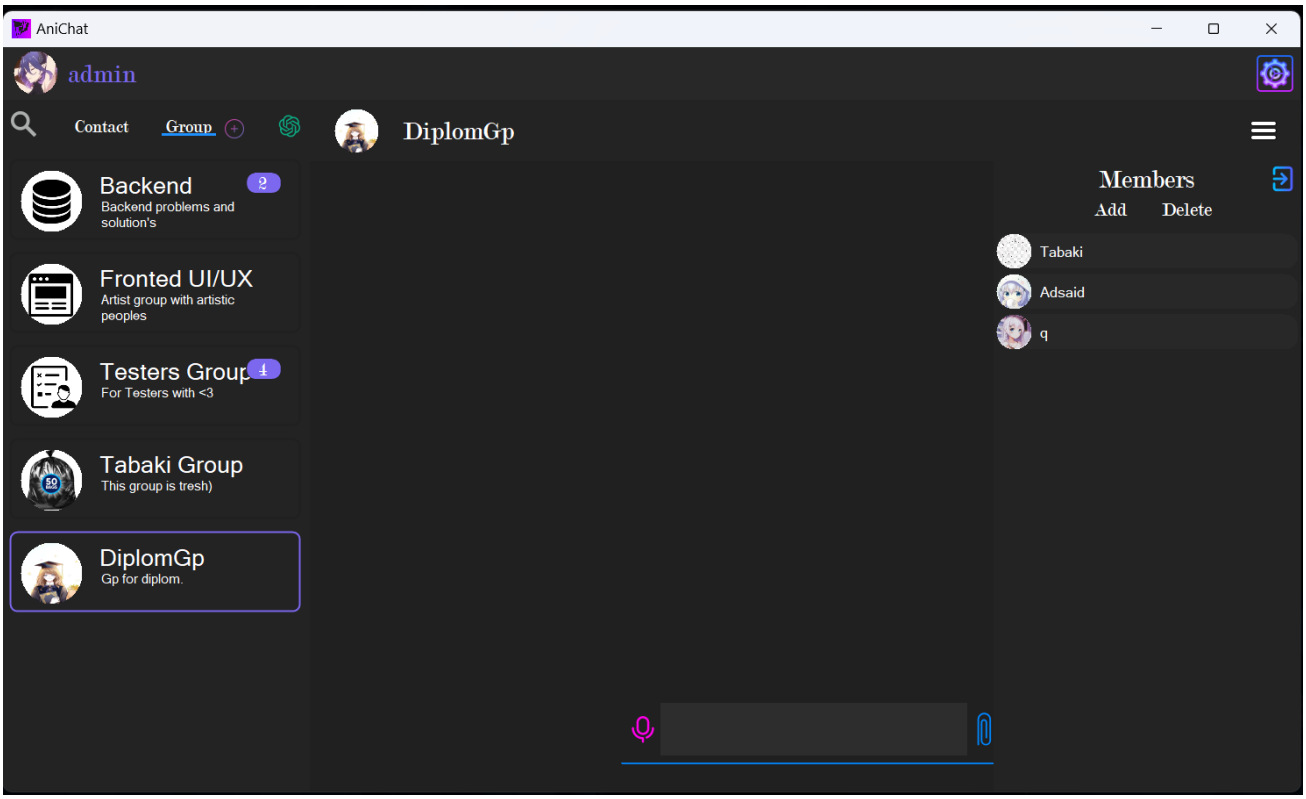


Рис. 3.65. Форма «AniChat» чат попередньо створеної групи відкрита панель для налаштувань та перегляду учасників.

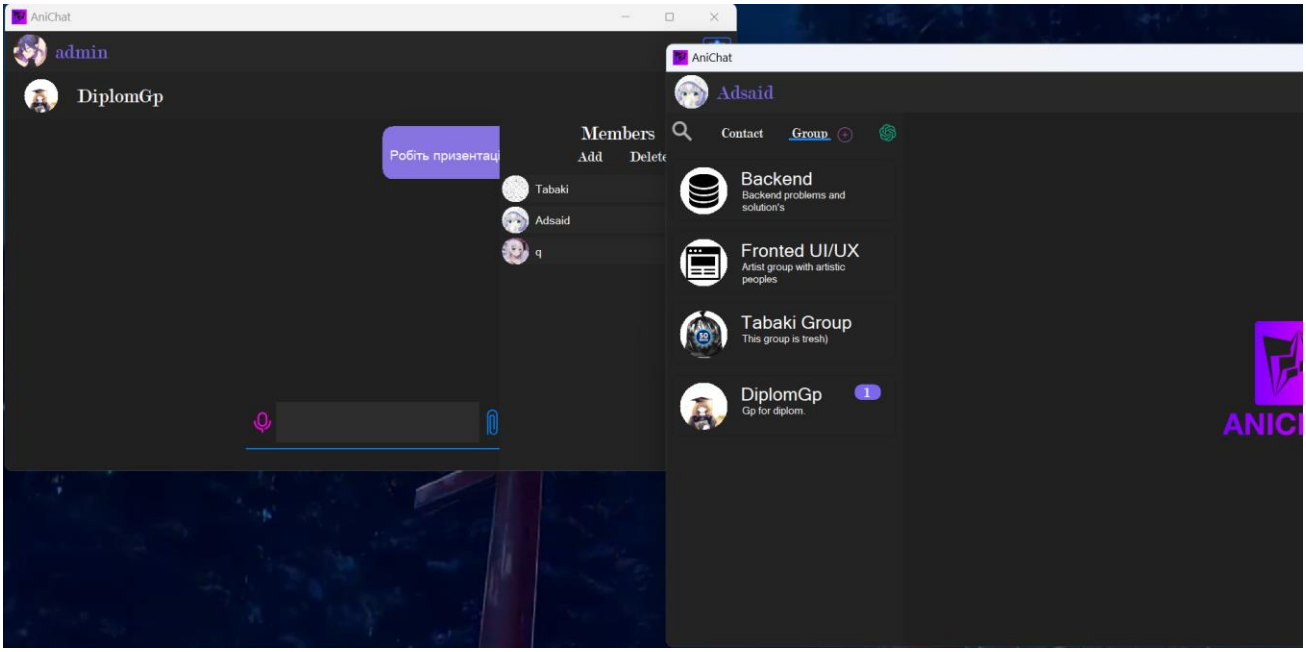


Рис. 3.66. Запущено другий мессенджер для демонстрації що користувачі яких ми додали мають доступ до групи.

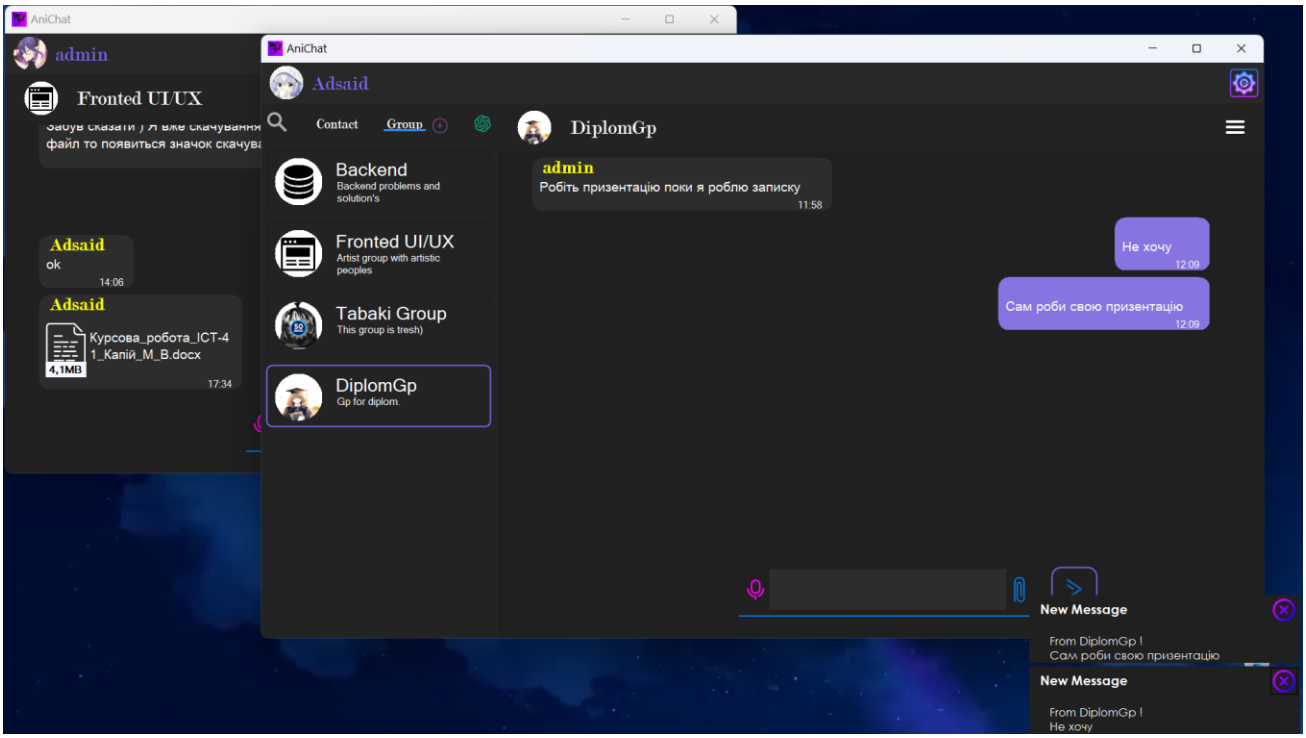


Рис. 3.67. Демострація спливаючий вікон.

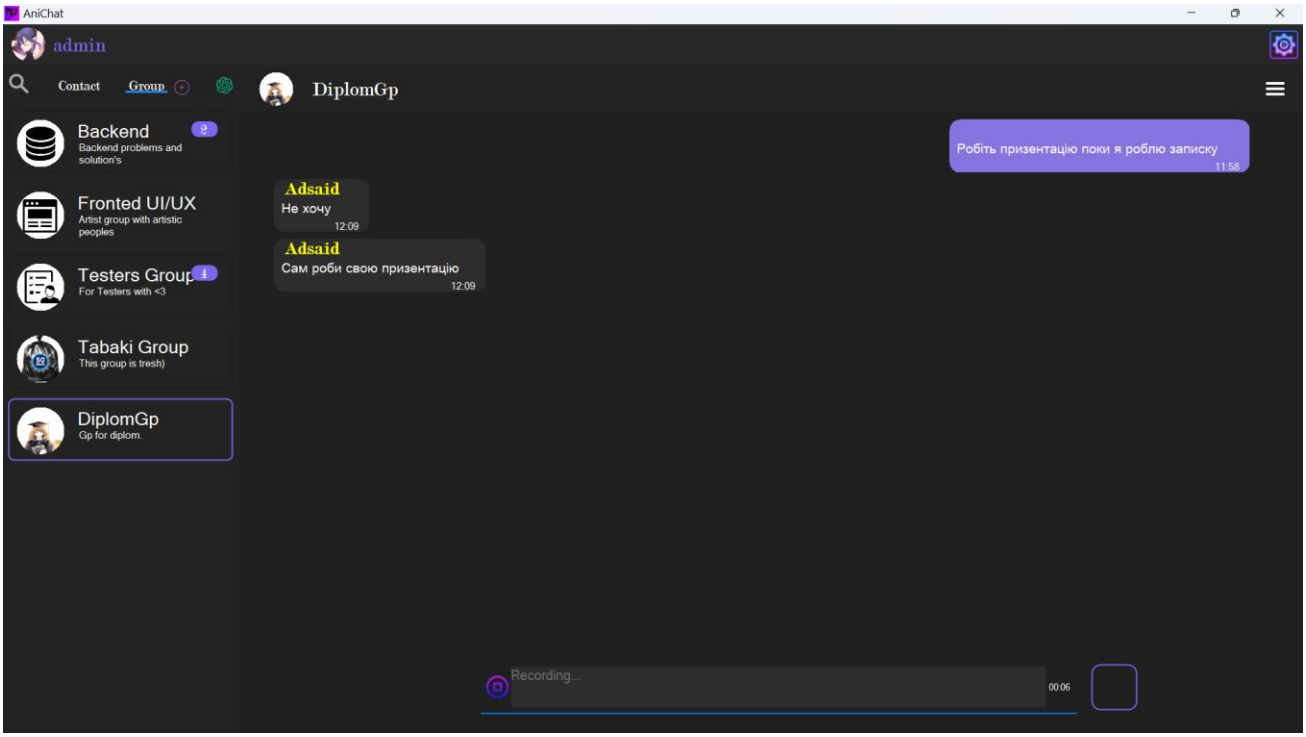


Рис. 3.68. Запис голосового повідомлення

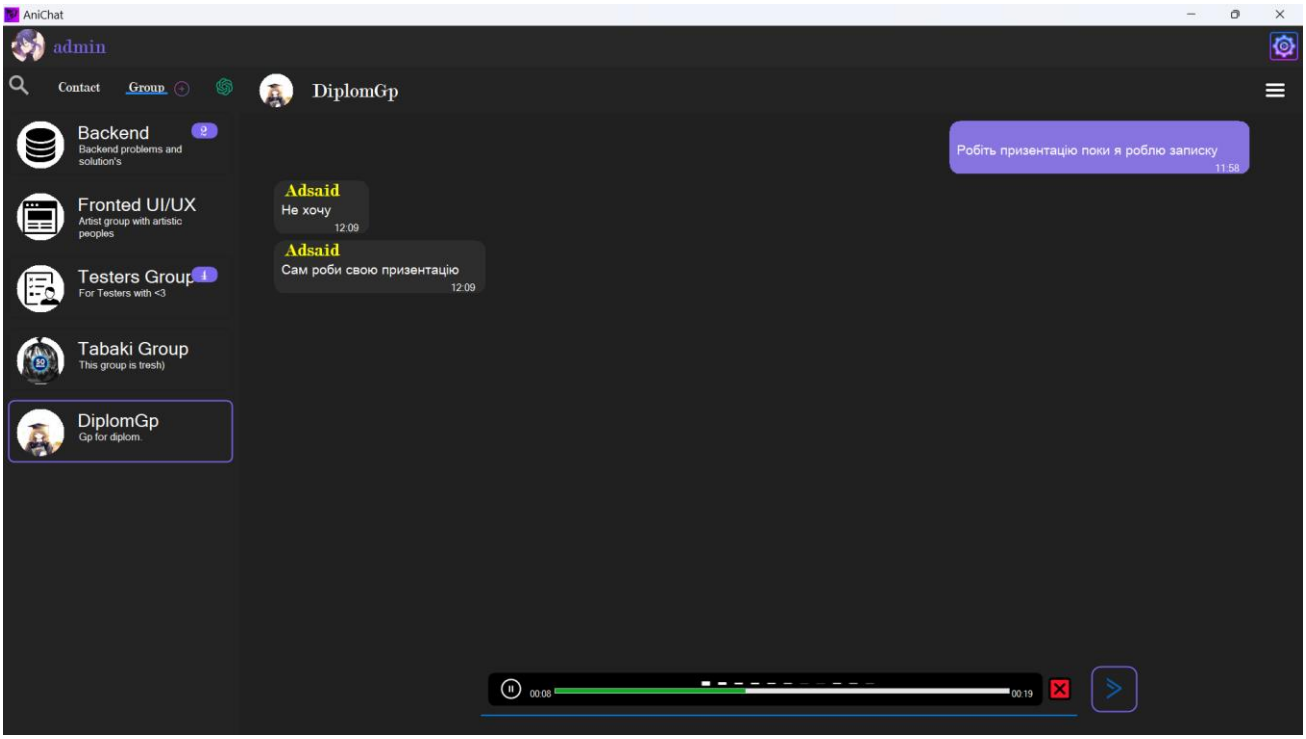


Рис.69. Прослуховування записаного голосового повідомлення.

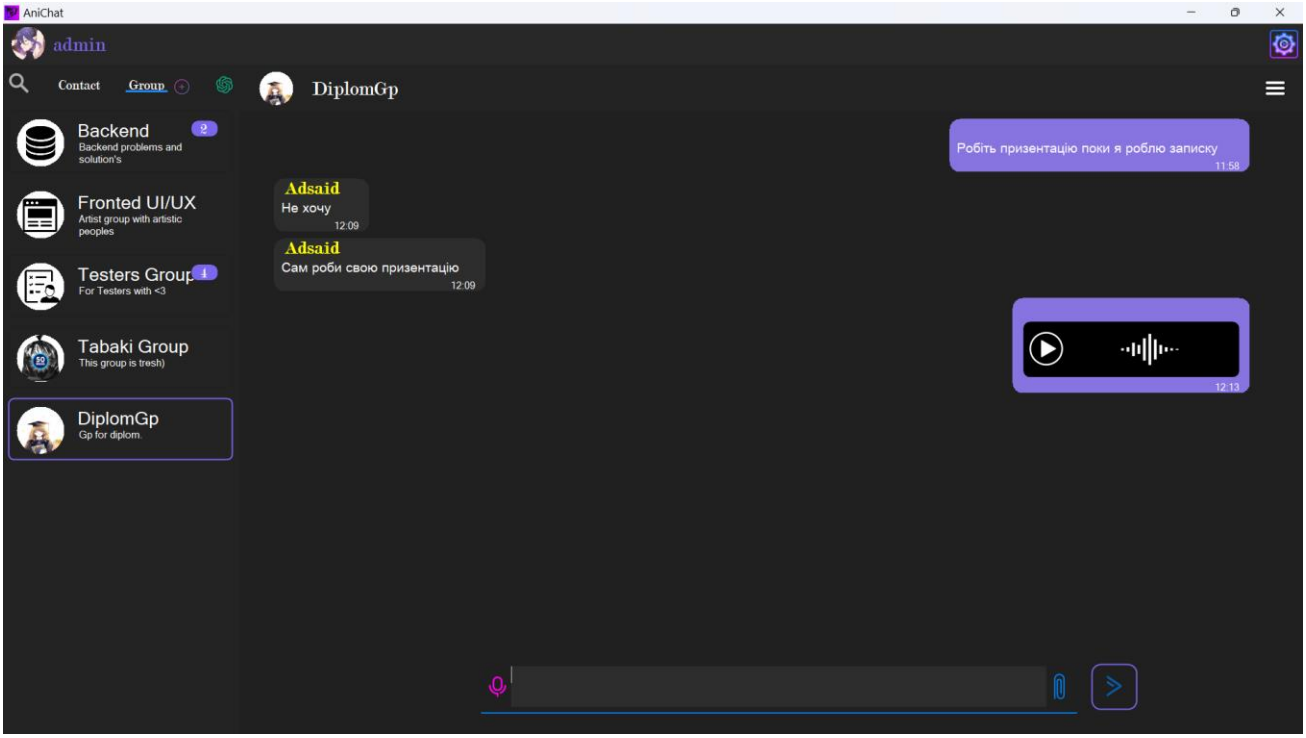


Рис. 3.70. Вигляд відправленого голосового повідомлення

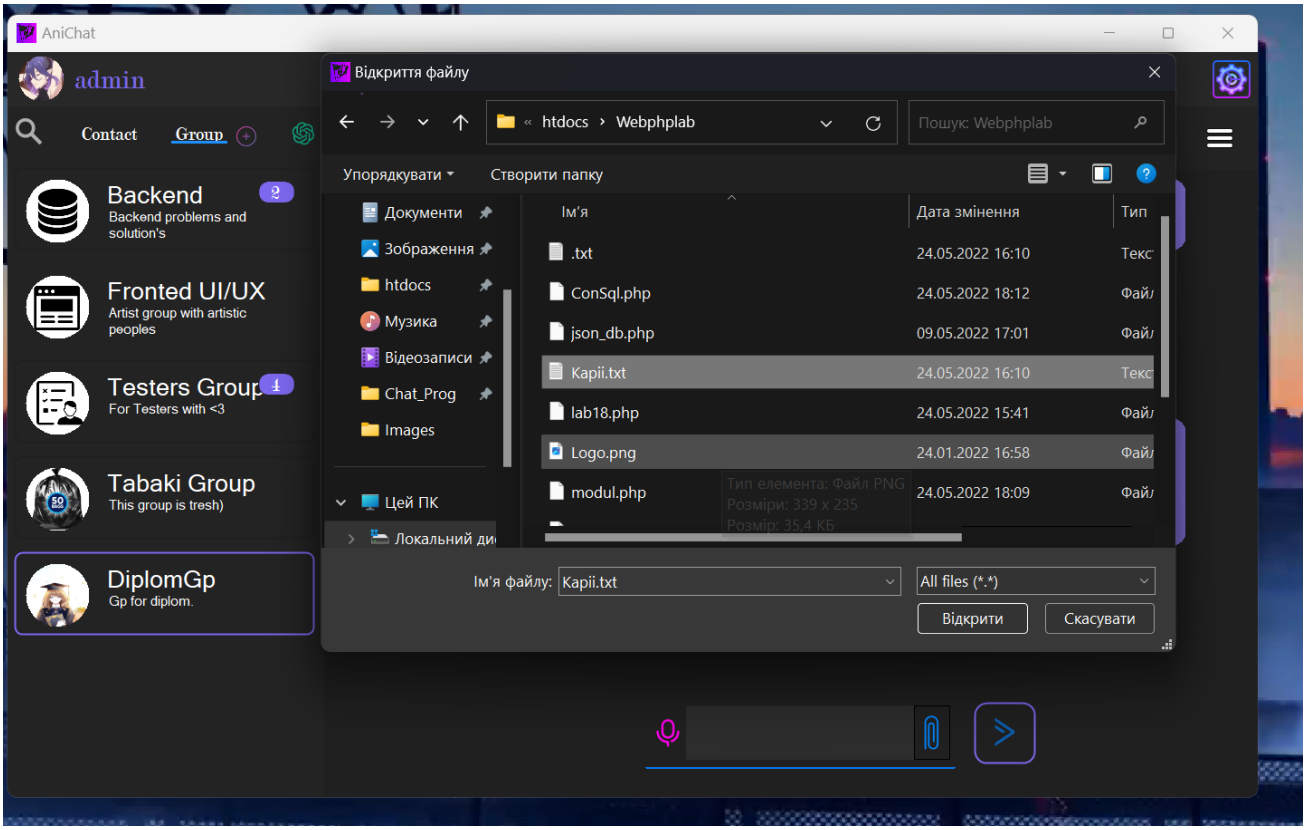


Рис. 3.71. Відправка файла

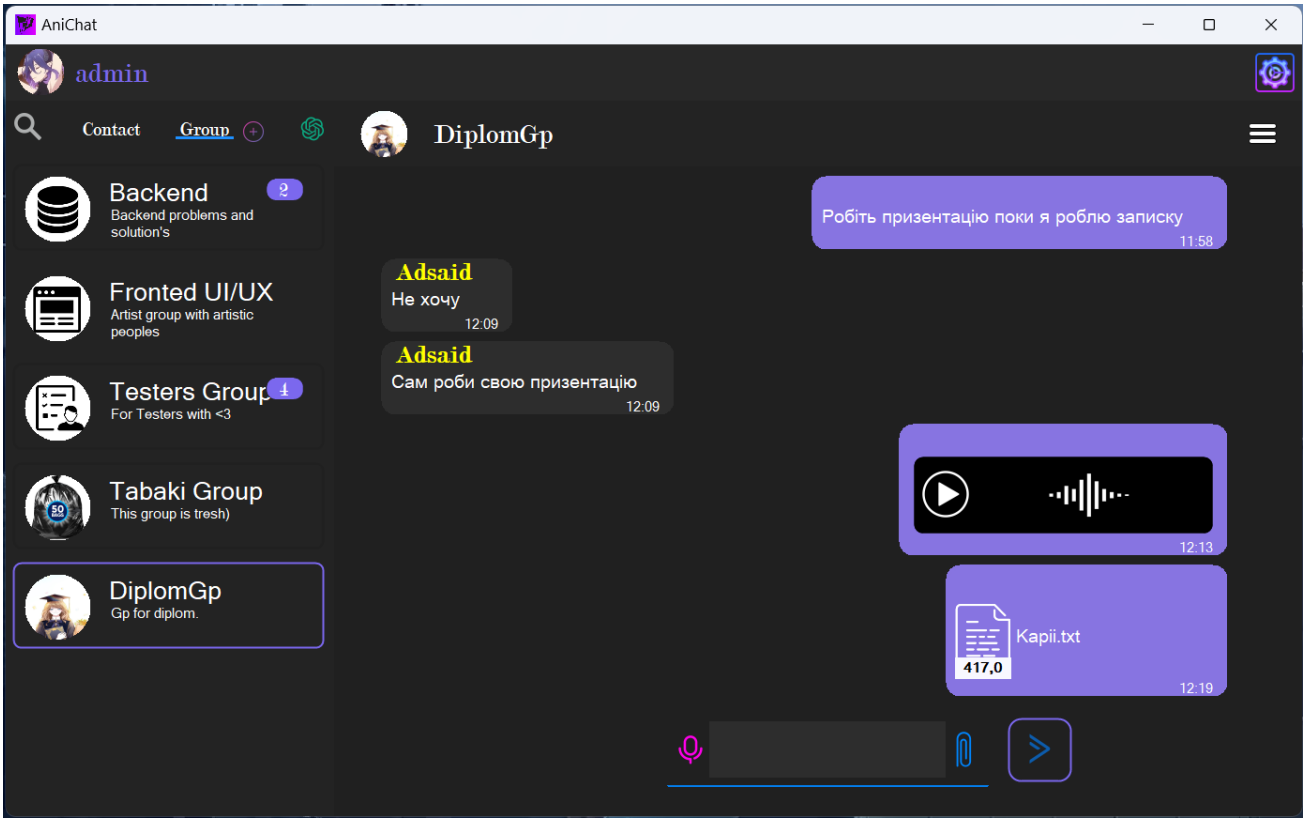


Рис. 3.72. Вигляд відправленого файла в чаті.

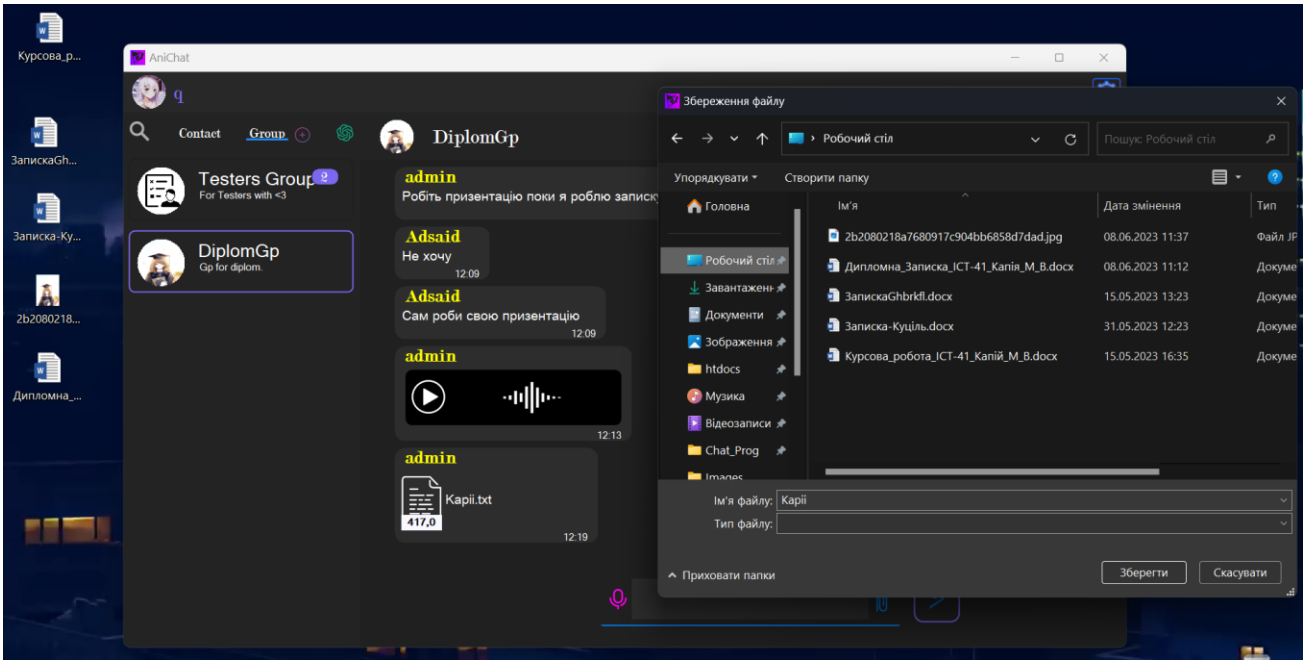


Рис. 3.73. Скачування файла відправленого користувачем admin в групу DiplomaGP

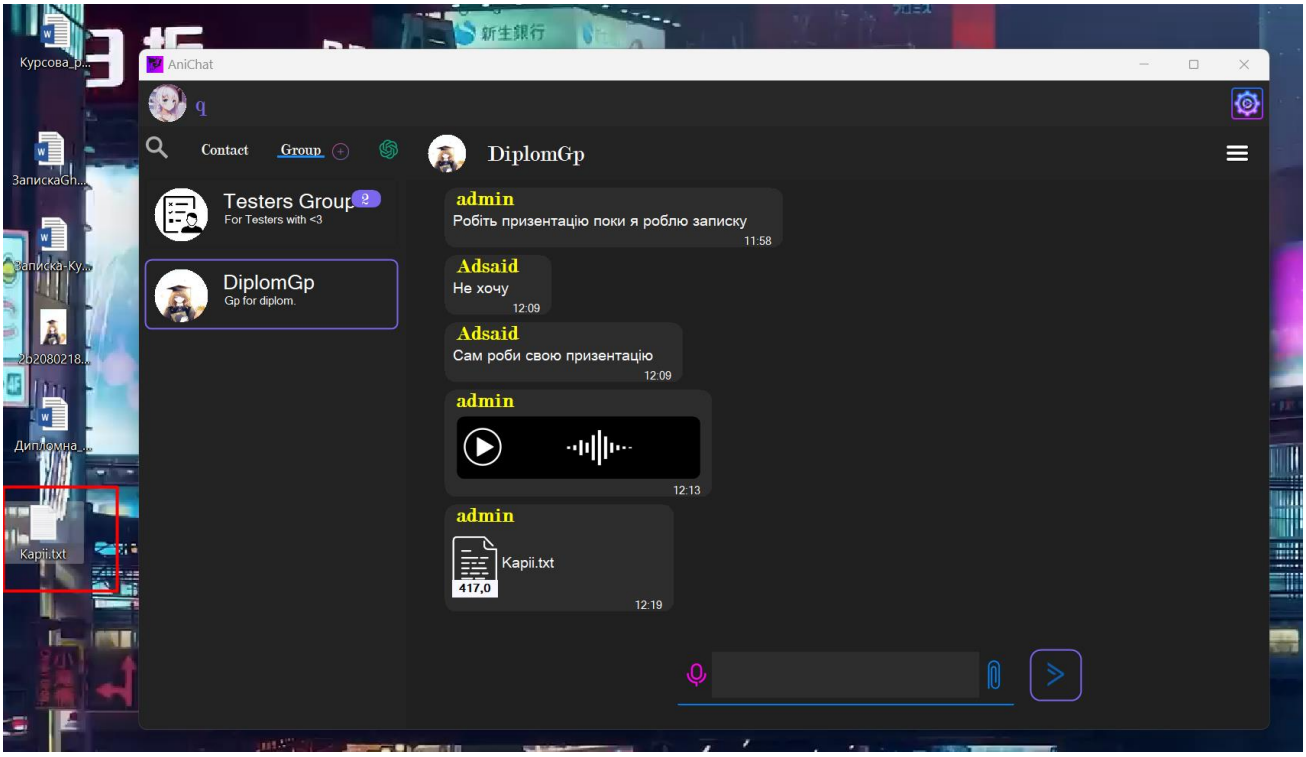


Рис. 3.74. Демонстрація, що файл скачався до вибраної нами раніше директорії

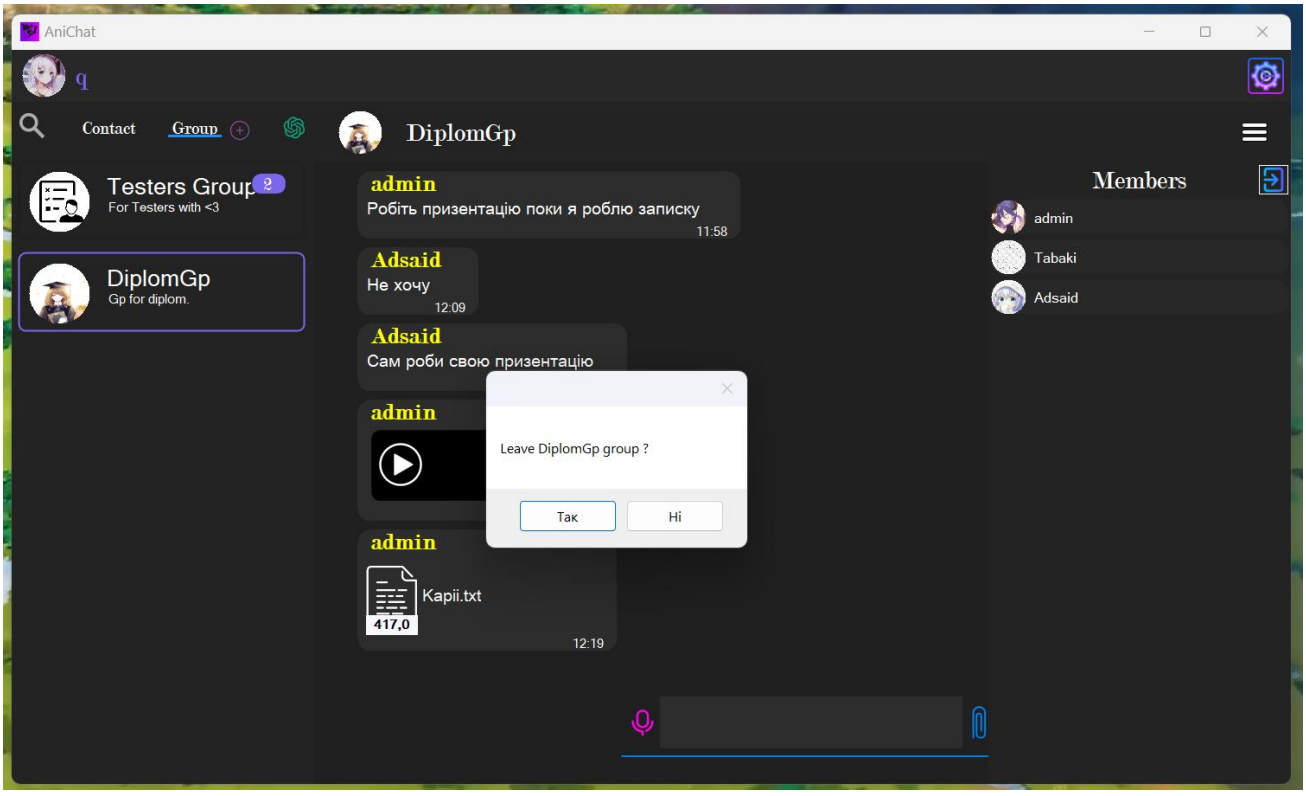


Рис. 3.76. Покидання групи DiplomaGP

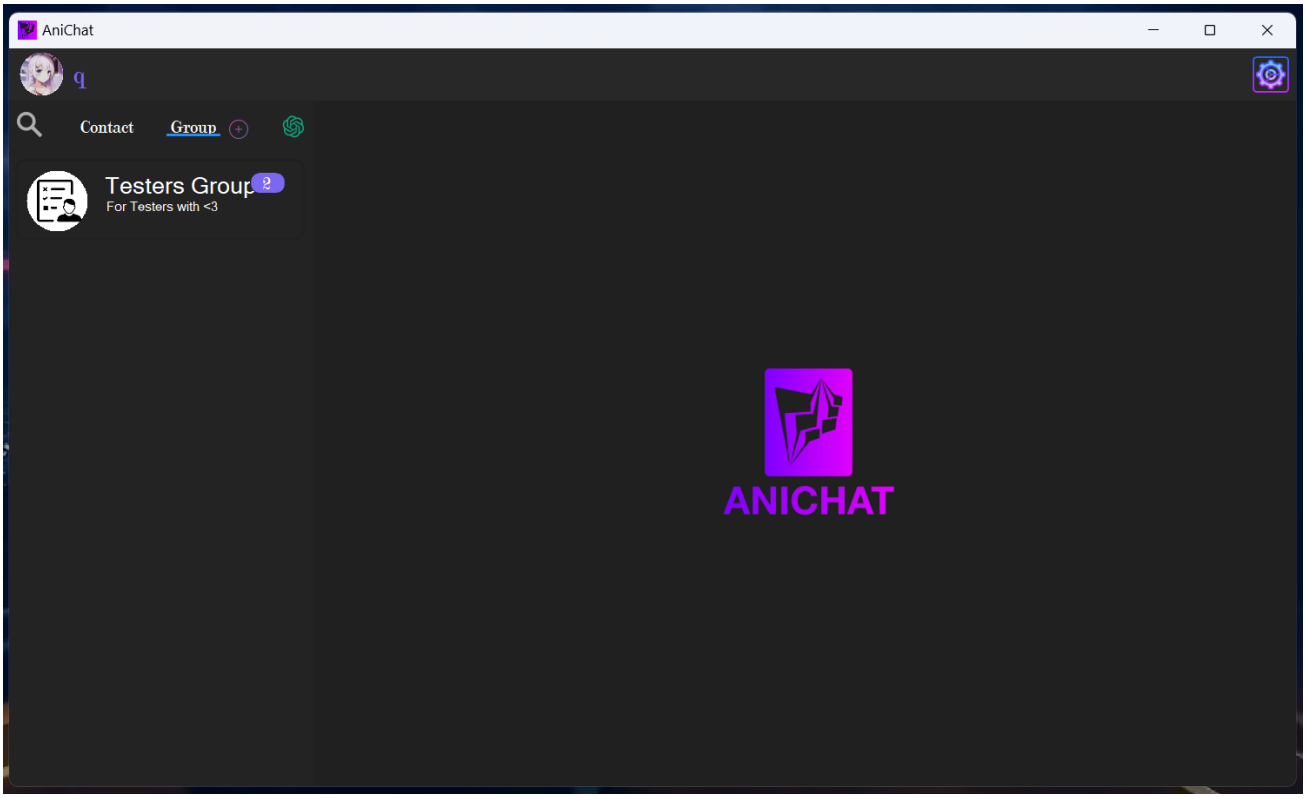


Рис. 3.77. Група більше не доступна у списку груп.

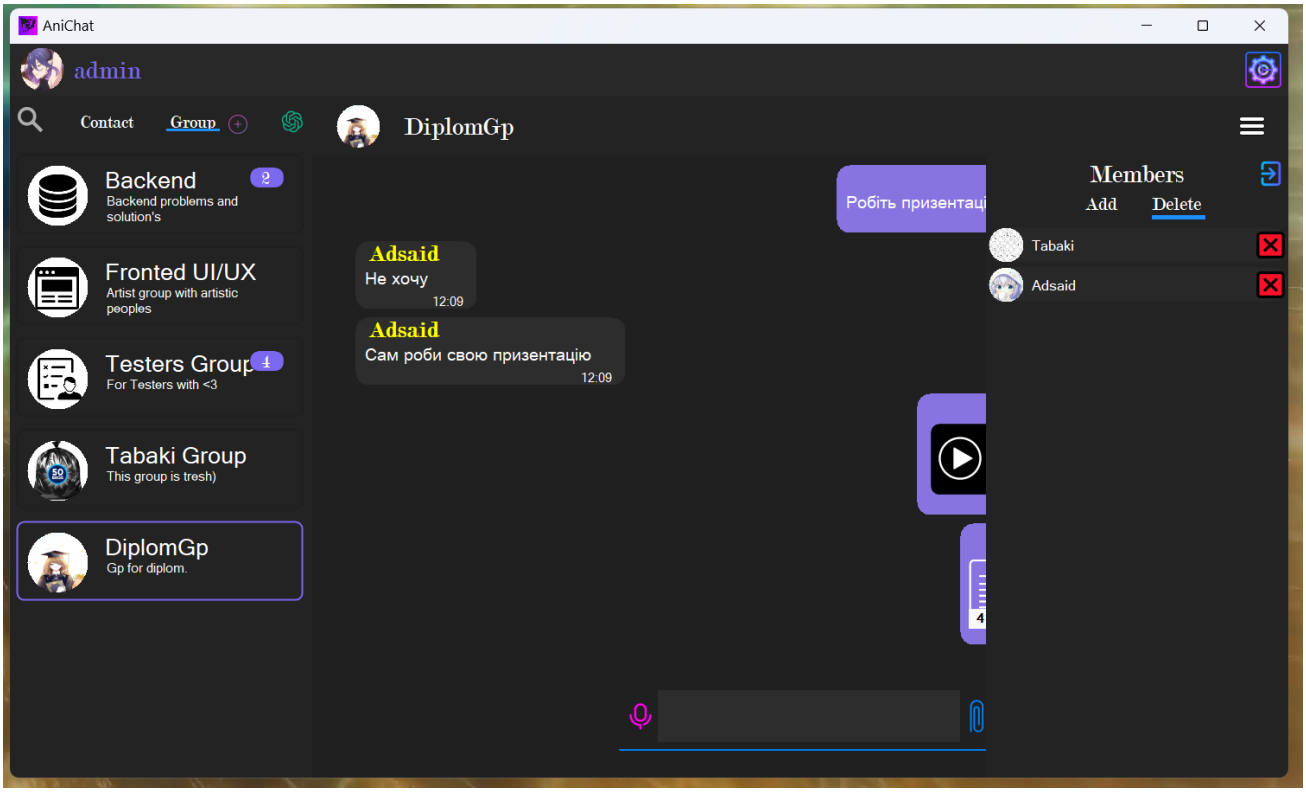


Рис. 3.78. Відкрите меню групи (тільки для адміністратора групи) для видалення користувачів з групи.

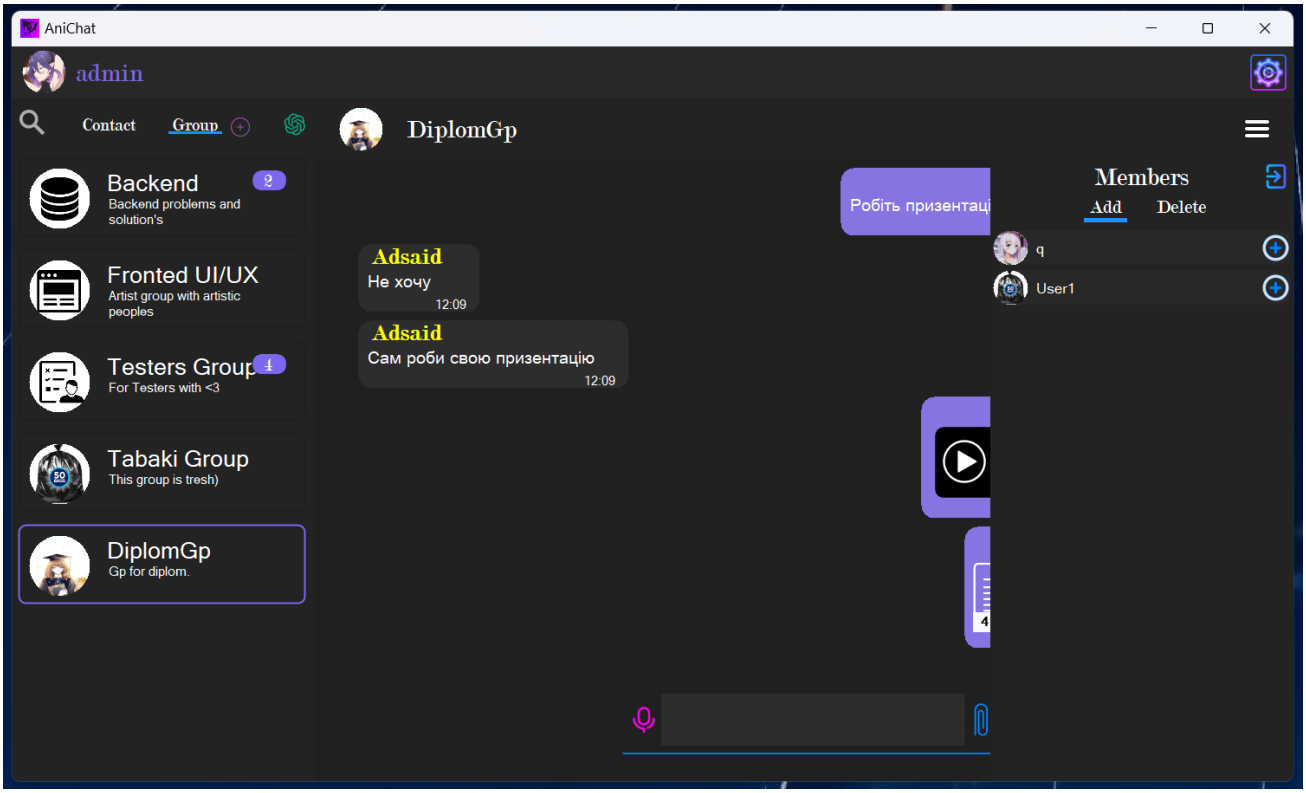


Рис. 3.79. Відкрите меню групи (тільки для адміністратора групи) для додавання користувачів у групу.

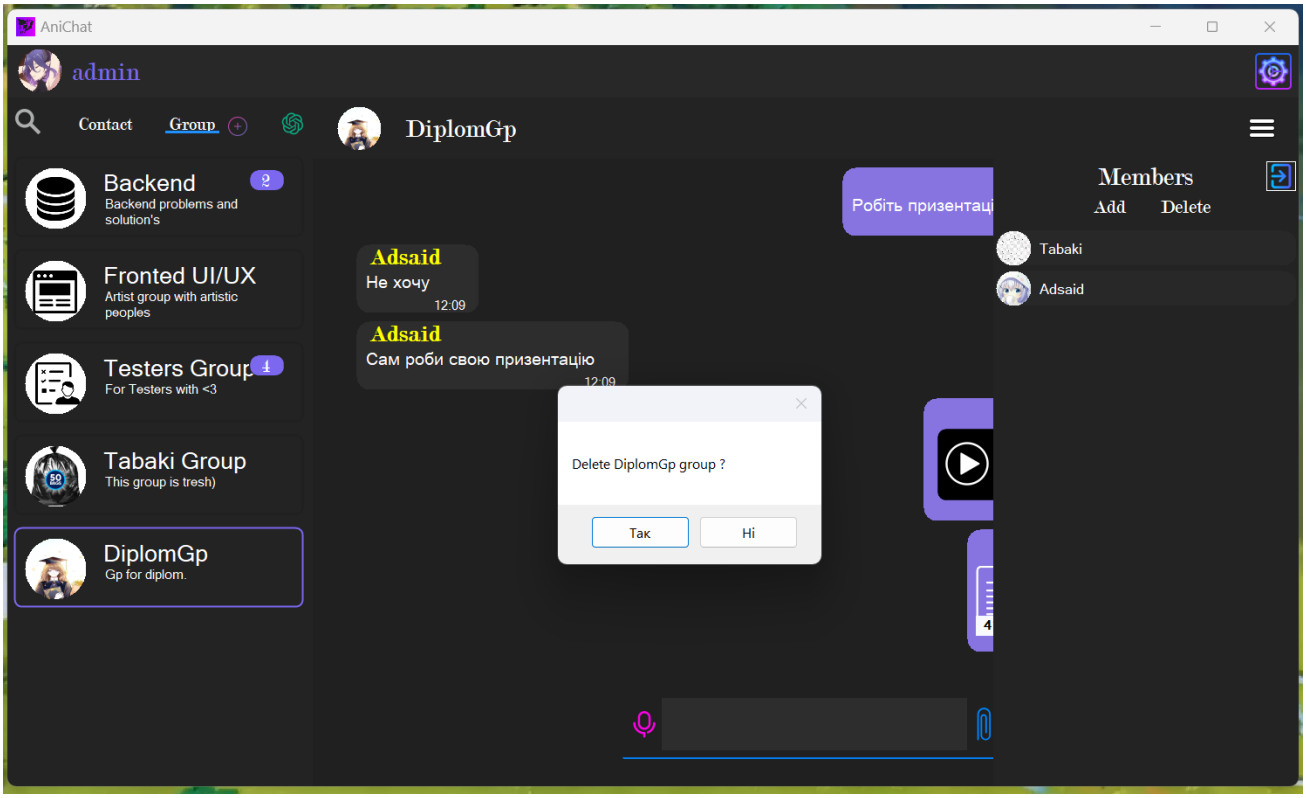


Рис. 3.80. Видалення групи.

ВИСНОВКИ

Розроблено корпоративний месенджер для зручної та безпечної комунікації між співробітниками в межах підприємства. Месенджер має клієнтську та серверну частини, які взаємодіють між собою.

Клієнтська сторона:

- Використано Windows Forms для розробки клієнтської сторони.
- Використано .NET Framework 4.8.
- Реєстрація та авторизація користувачів з використанням ідентифікаційних даних (логін, пароль).
- Створено, перегляд та керування різними групами для обміну повідомленнями.
- Реалізовано відправлення текстових, голосових, файлових повідомлень між користувачами.
- Для запису та відтворення голосових повідомлень було використано бібліотеки NAudio та AxWMPLib.
- Для відправлення запитів та отримання відповідей від бота до API ChatGPT використано бібліотеку RestSharp.

Серверна сторона:

- Використано бази даних MySql для збереження користувачів, повідомлень та інших даних.
- Реалізований сервіс розсилки повідомлень за допомогою Windows Communication Foundation (WCF).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Skeet J. C# in Depth. – Manning Publications, 2019. – 528 p.
2. Troelsen A., Japikse P. Pro C# 9 with .NET 5. – Apress, 2021. – 1616 p.
3. Wagner B. Effective C#: 50 Specific Ways to Improve Your C#. – Boston: Addison-Wesley, 2020. – 328 p.
4. Griffiths I. Programming C# 8.0. – O'Reilly Media, 2019. – 816 p.
5. Albahari J., Albahari B. C# 9.0 in a Nutshell: The Definitive Reference. – O'Reilly Media, 2021. – 1088 p.
6. Hocking J. Unity in Action: Multiplatform Game Development in C#. – Manning Publications, 2019.
7. Hocking J. Unity in Action: Multiplatform Game Development in C#. – Manning Publications, 2019. – 350 p.
8. Freeman A. Pro ASP.NET Core 3. – Apress, 2019. – 1000 p.
9. Price M. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development. – Packt Publishing, 2019. – 820 p.
10. Troelsen A., Japikse P. Pro C# 8 with .NET Core 3. Apress, 2019. – 1600 p.
11. Esposito D. Modern Web Development with ASP.NET Core 3. – Microsoft Press, 2020. – 300 p.
12. Lock A. ASP.NET Core in Action. – Manning Publications, 2020. – 500 p.
13. Nagel C. Professional C# 8 and .NET Core 3.0. – Wrox, 2020. – 1100 p.
14. Richards M., Ford N. Fundamentals of Software Architecture. – O'Reilly Media, 2021. – 450 p.
15. Fowler M. Refactoring: Improving the Design of Existing Code. – Addison-Wesley, 2021. – 450 p.

ДОДАТКИ

Клас Host:

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

namespace AniHost
{
   class Host
   {
      static void Main(string[] args)
      {
         using (var host = new ServiceHost(typeof(wcf_service.ServiceAniChat)))
         {
            host.Open();
            Console.WriteLine("Host alredy starting !!!");
            Console.ReadLine();
            Console.ReadLine();
         }
      }
   }
}
```

Клас UserListItem :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AniChat
{
   public partial class UserListItem : UserControl
   {
      int msgNum = 0;

      public int ID { get; set; }
      public string Chatname { get; set; }
      public string GitLink { get; set; }
```

```
public string InLink { get; set; }

public UserListItem(Image image, string name, string rank)
{
    InitializeComponent();
    if (image != null)
    {
        UserPictureBox.Image = image;
    }
    UserNamelb.Text = name;
    Rank_lb.Text = rank;
}

public void SetnewMsgNum(int num)
{
    if (num <= 0)
    {
        customPanel1.Visible = false;
        msgNum = 0;
        return;
    }

    msgNum += num;
    newMsgNum_lb.Text = msgNum.ToString();
    customPanel1.Visible = true;
}
private void Item_cPl_MouseEnter(object sender, EventArgs e)
{
    Item_cPl.BackColor = Color.FromArgb(65, 65, 65);
    //Item_cPl.BorderColor = Color.MediumSlateBlue;
}
private void Item_cPl_MouseLeave(object sender, EventArgs e)
{
    Item_cPl.BackColor = Color.FromArgb(35, 35, 35);
    //Item_cPl.BorderColor = Color.FromArgb(35, 35, 35);
}

private void gitLink_pb_Click(object sender, EventArgs e)
{
    LaunchWeblink(GitLink);
}

private void InLink_pb_Click(object sender, EventArgs e)
{
    LaunchWeblink(InLink);
}

// Performs the actual browser launch to follow link:
private void LaunchWeblink(string url)
{
    if (IsHttpURL(url)) Process.Start(url);
```

```
        }

        // Simple check to make sure link is valid,
        // can be modified to check for other protocols:
        private bool IsHttpURL(string url)
        {
            return
                ((!string.IsNullOrWhiteSpace(url)) &&
                (url.ToLower().StartsWith("http")));
        }

        private void UserListItem_Load(object sender, EventArgs e)
        {
            if (String.IsNullOrEmpty(GitLink))
            {
                gitLink_pb.Visible = false;
            }
            if (String.IsNullOrEmpty(InLink))
            {
                InLink_pb.Visible = false;
            }

        }
    }
}
```

Клас GroupListItem :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AniChat
{
    public partial class GroupListItem : UserControl
    {
        int msgNum = 0;
        public GroupListItem(int id, Image image, string name, string description, string admin)
        {
            InitializeComponent();
            GroupId = id;
            GroupAdmin = admin;
            if (image != null)
            {
```

```
            UserPictureBox.Image = image;
        }
        UserNamelb.Text = name;
        Descr_lb.Text = description;
    }
    public int GroupId { get; set; }

    public string GroupAdmin { get; set; }

    public void SetnewMsgNum(int num)
    {
        if (num <= 0)
        {
            customPanel1.Visible = false;
            msgNum = 0;
            return;
        }

        msgNum += num;
        newMsgNum_lb.Text = msgNum.ToString();
        customPanel1.Visible = true;
    }
    private void Item_cPl_MouseEnter(object sender, EventArgs e)
    {
        Item_cPl.BackColor = Color.FromArgb(65, 65, 65);
        //Item_cPl.BorderColor = Color.MediumSlateBlue;
    }

    private void Item_cPl_MouseLeave(object sender, EventArgs e)
    {
        Item_cPl.BackColor = Color.FromArgb(35, 35, 35);
        //Item_cPl.BorderColor = Color.FromArgb(35, 35, 35);
    }

  }
}
```

Клас Recorder:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using NAudio.Wave;

namespace AniChat
{
  class Recorder
```

```
    {
        private WaveInEvent waveIn;
        private WaveFileWriter writer;

        public void StartRecording(string filePath)
        {
            waveIn = new WaveInEvent();
            waveIn.WaveFormat = new WaveFormat(48000, 1); // 48kHz mono
            waveIn.DataAvailable += WaveIn_DataAvailable;

            writer = new WaveFileWriter(filePath, waveIn.WaveFormat);

            waveIn.StartRecording();
        }

        public void StopRecording()
        {
            waveIn.StopRecording();
            writer.Close();
            writer?.Dispose();
            writer = null;
            waveIn?.Dispose();
            waveIn = null;
        }

        private void WaveIn_DataAvailable(object sender, WaveInEventArgs e)
        {
            writer.Write(e.Buffer, 0, e.BytesRecorded);
        }
    }
}
```