

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра автоматизованих систем управління



Курсова Робота
з дисципліни
Об'єктно-орієнтоване програмування
на тему:
“Online Chat Room”

Виконав: студент групи ОІ-12
Мамрак Артем

Прийняв: к.т.н., доцент
Зербіно Д. Д.

Львів – 2023

Зміст

Вступ.....	3
1. Постановка задачі.....	4
1.1 Цілі виконання курсової роботи.....	4
1.2 Індивідуальне завдання студента.....	4
2. Огляд літератури.....	5
2.1 Особливості предметної області.....	5
2.2 Порівняльний аналіз.....	6
3. Опис етапу проектування.....	10
3.1 Архітектура роботи серверу.....	10
3.2 Архітектура роботи клієнтної частини додатку.....	11
3.3 Архітектура графічного інтерфейсу додатку.....	13
4. Програмне рішення.....	14
4.1 Розробка графічного інтерфейсу. Python.....	14
4.2 Розробка серверу. Python.....	16
4.3 Розробка Клієнту. Python.....	17
4.4 Розробка вікна логіну. C++.....	18
4.5 Взаємодія різномовних модулів.....	18
4.6 Список нестандартних бібліотек/фреймворків.....	19
5. Опис проведених експериментів.....	20
5.1 Опис дій.....	20
5.2 Проведені експерименти.....	21
Висновки.....	24
Список використаних джерел.....	25
Додатки.....	26
Додаток 1. Повний графічний інтерфейс.....	26
Додаток 2. Програмний модуль server.py.....	29
Додаток 3. Програмний модуль client.py.....	31
Додаток 4. Програмний модуль chat_GUI.py.....	33
Додаток 5. Програмний модуль app.py.....	36
Додаток 6. Програмний модуль CLogin.dll.....	49

Вступ

Мета курсової роботи: ознайомитися із такими технологіями як: багатопоточність (англ. threading) для виконання паралельного опрацювання даних, незалежно від інших потоків, мережеві гнізда (англ. network sockets), які служать кінцевою точкою для надсилання та отримання даних через мережу та графічний інтерфейс (англ. GUI) для відображення даних у зручному для користувача вигляді. Програми, на основі даних методів, є дуже актуальними у сучасному світі, тому їх розвиток ще довго буде підтримуватися. Майбутнє застосування результатів роботи на практиці є однозначним, адже більшість людей, коли користується інтернетом, взаємодіє із цими трьома технологіями.

1. Постановка задачі

1.1 Цілі виконання курсової роботи

- навчитись формулювати задачі для програмних проектів, розкласти їх на підзадачі,
- вибирати методи та алгоритми для їх розв'язання;
- навчитись визначати, порівнювати і вибирати сучасні засоби та технології програмування для розв'язання конкретних прикладних задач;
- навчитись реалізовувати, налагоджувати та розгортати програмні проекти за допомогою мов програмування C++ та Python і технологій розробки з їх використанням;
- навчитись описувати програмні проекти, як кінцеві результати, так і процес виконання.

1.2 Індивідуальне завдання студента

Розробити дві програми: сервер, який приймає нових клієнтів, отримує інформацію від них та відправляє її усім, хто підключений до "кімнати", та додаток-клієнт який виступає у ролі графічного інтерфейсу для користувача та під'єднується до відкритого серверу.

Основні можливості: додаток-клієнт може відправляти повідомлення та файли. Сервер ж, має приймати усі ці дані, обробляти їх та надсилати кожному клієнту.

Вимоги з боку користувача: простий та зрозумілий інтерфейс.

Можливості використання на апаратних та програмних платформах: PC, на операційній системі Windows.

2. Огляд літератури

2.1 Особливості предметної області

Розробка програмного проекту, пов'язаного з онлайн-чат-кімнатами, вимагає детального розуміння предметної області і визначення його місця серед схожих проектів. Основні особливості предметної області, які слід розкрити, включають:

Функціональність: Це можуть бути розмови один-на-один, групові чати, бесіди, канали, у яких можна, наприклад, читати новини. Також популярними функціями є можливість надсилання зображень, відео чи аудіофайлів, налаштування профілю користувача, інтеграція з іншими сервісами тощо.

Безпека: Вона завжди була важливим аспектом у будь-якій сфері, особливо у ІТ, адже компанії отримують тонни інформації, яка включає у себе не тільки, особисті дані користувача, а й іноді навіть дані про його оточення. Необхідно розглянути можливості аутентифікації користувачів, захисту від спаму, шифрування передачі даних та інші механізми, що забезпечують безпеку комунікації. Більш складні методи контролю доступу включають різні форми біометричної аутентифікації. Ці системи безпеки використовують біометричні дані, або унікальні біологічні характеристики, для ідентифікації особи авторизованих користувачів. Відбитки пальців і розпізнавання обличчя є двома прикладами поширених застосувань цієї технології.

Реал-тайм комунікація (англ. RTC): Онлайн-чат-кімнати мають забезпечувати миттєву взаємодію між користувачами. Загалом вони використовують м'яку систему зв'язку (англ. Soft real-time system)[2]. Зазвичай, дані, що передаються через м'яку систему зв'язку в режимі реального часу, не зберігаються на централізованому сервері, і однорангові вузли з'єднані безпосередньо один з одним, а не через сервер, хоча проміжні з'єднувальні вузли між одноранговими вузлами дозволені, якщо прямий зв'язок встановити

неможливо. Для цього можуть використовуватися технології веб-сокетів, що дозволяють обмінюватися повідомленнями в режимі реального часу.

Масштабованість та обмеження: У разі, якщо чат-кімната має багато користувачів, важливо забезпечити "місце" для кожного клієнта. Популярною технологією є використання хмарних сервісів або схожих технологій. Також часто розробники додають певні обмеження для користувачів, щоб полегшити собі роботу та навантаження на загальну систему.

2.2 Порівняльний аналіз

Для порівняльного аналізу можливих варіантів розв'язання поставленої задачі візьмемо для прикладу два відомих додатки: Telegram[1] та Discord[7].

Тип	Telegram	Discord
Функціональність	<p>Налаштування профілів користувачів.</p> <p>Переписка один-на-один.</p> <p>Створення бесід, каналів, групових чатів.</p> <p>Передача зображень, відео чи аудіофайлів.</p> <p>Можливість телефонувати як онлайн, так і за номером телефону.</p> <p>Перегляд старих повідомлень.</p>	<p>Налаштування профілів користувачів.</p> <p>Переписка один-на-один.</p> <p>Створення бесід, каналів, групових чатів.</p> <p>Передача зображень, відео чи аудіофайлів.</p> <p>Можливість телефонувати онлайн.</p> <p>Перегляд старих повідомлень.</p>

Табл. 2.2.1 Порівняння функціональності додатків Telegram та Discord

Можна помітити, що функціонал додатків дуже схожий. Це означає, що більшість онлайн чатів будуть мати ці функції, третій до прикладу WhatsApp. Якщо компанія хоче зробити додаток, який зможе конкурувати на ринку, вона має забезпечити комфорт користувачів. Саме завдяки великому функціоналу створюється велика база користувачів.

Тип	Telegram	Discord
Безпека	Вхід в аккаунт по паролю та QR-коду. Можливість налаштувати подвійну аутентифікацію. Вхід у додаток через пароль або біометрію.	Вхід в аккаунт по паролю. Можливість налаштувати подвійну аутентифікацію, зокрема через Google Authenticator. Вхід після підтвердження повідомлення на пошті або телефоні, через яку/який було створено аккаунт.

Табл. 2.2.2 Порівняння безпеки додатків Telegram та Discord

Як раніше було зазначено, безпека - це одна з найважливіших складових будь-якого продукту ІТ. Кожен користувач бажає, щоб його дані зберігалися як найкраще і ніхто не мав до них доступ. Тому було придумано десятки варіантів безпеки. Найефективніші з них використовують великі компанії, тому їх можна побачити в таблиці. Але знову ж таки, кожна технологія потребує ресурси, у даному випадку бази даних, які зберігатимуть інформацію про клієнта. Доступ до них мають лише одиниці, які наймаються спеціально для роботи з ними. Зазвичай такі робітники підписують договір про нерозголошення. Ось так в ІТ цінується безпека даних користувача.

Тип	Telegram	Discord
Реал-тайм комунікація (англ. RTC)	Для текстових повідомлень: TCP-connection Для дзвінків: UDP-connection	Для текстових повідомлень: TCP-connection Для дзвінків: UDP-connection

Табл. 2.2.3 Порівняння RTC додатків Telegram та Discord

Наразі, кожен додаток використовує такі протоколи передачі даних.

Transmission Control Protocol (TCP) - організовує дані таким чином, щоб їх можна було передавати між сервером і клієнтом. Це гарантує цілісність даних,

які передаються через мережу. Перш ніж передати дані, TCP встановлює з'єднання між джерелом і одержувачем, яке, як він гарантує, залишається активним до початку зв'язку. Потім він розбиває великі обсяги даних на менші пакети, забезпечуючи при цьому цілісність даних протягом усього процесу. [12]

User Datagram Protocol (UDP) - пришвидшує зв'язок, не встановлюючи офіційно з'єднання до передавання даних. Це дозволяє дуже швидко передавати дані, але це також може спричинити втрату пакетів під час передачі - і створити можливості для використання у формі DDoS-атак. [13]

Саме тому для текстових повідомлень використовують TCP - бо важливо, щоб інформація точно дійшла до отримувача, інакше буде втрачено цілісність. Швидкість не так важлива, адже отримувач може прочитати повідомлення навіть через декілька годин після отримання. А ось онлайн дзвінок інша річ. UDP використовується - бо даних, які треба передати, дуже багато, залежить від частоти кадрів камери, тому втративши 1-2 кадри загальна цілісність не зміниться. У даному випадку швидкодія виходить на перше місце, адже кадри ніде не зберігаються, як повідомлення, вони одразу обробляються та показуються отримувачеві.

Тип	Telegram	Discord
Масштабованість	Великі власні серверні. Використання хмарних технологій	Великі власні серверні. Використання хмарних технологій

Табл. 2.2.4 Порівняння масштабованості додатків Telegram та Discord

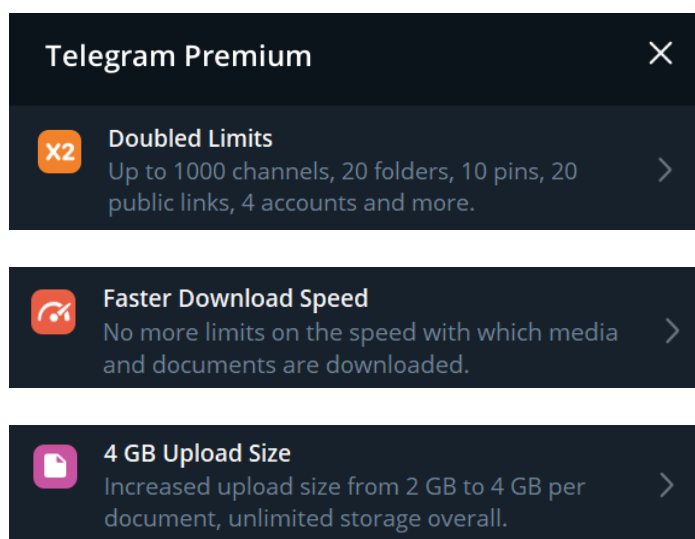
Для збереження інформації, такої як профілі, бесіди, історія повідомлень, потрібні сервери, які можна зробити самим, що буде дорожче, але функціонал повністю в руках компанії, тобто зростає гнучкість та практичність, або орендувати у більших компаній, наприклад Google, що є дешевшим варіантом,

але ефективність може значно впасти. Як висновок, для якості потрібен великий бюджет, який на старті є не у всіх.

Тип	Telegram	Discord
Обмеження	Кількість каналів і супер-груп, учасником яких ви можете бути до 500. Загальна кількість учасників груп до 200 000 і т.д.	Кожен новий сервер Discord має ліміт учасників 250 000 і обмеження на розмір файлу 8 Мб. 5,000 онлайн-учасників для нових серверів і т.д.

Табл. 2.2.5 Порівняння обмежень додатків Telegram та Discord

Практика обмежень вже стала нормою тому, що не кожна компанія має ресурси, щоб обслуговувати усіх користувачів. Плюс обмеження зазвичай можна зняти або полегшити, якщо купити Premium підписку на місяць, що підвищить заробіток компанії.



Pricing and Features		<div>MOST POPULAR</div> <div>NITRO BASIC</div> <div>NITRO</div>	
Bigger file sharing	50 MB	500 MB	
HD streaming	×	Up to 4K and 60fps	
Join up to 200 servers	×	✓	
Longer messages up to 4000 characters	×	✓	

Рис. 2.2.6 Приклад підписок від Telegram та Discord

Після даного аналізу, можна оцінити, як приблизно має виглядати сучасний онлайн-чат та які технології мають бути використано.

3. Опис етапу проектування

3.1 Архітектура роботи серверу

Програмна частина серверу є доволі простою, але потребує знань багатопоточності. Загальна кількість потоків: $n + 1$, де n - кількість клієнтів.

1. Сервер увімкнений завжди, вимкнутись він може лише вручну.
2. Для початку сервер налаштовується, відкриває до себе доступ.
3. Сервер приймає нового користувача.
4. При вдалому підключенні сервер отримує дані клієнта, точніше IP, PORT та username.
5. Отримані дані заносяться у відповідний список, для майбутнього використання.
6. Сервер обов'язково створює новий потік, який буде відповідати за даного клієнта, тобто всі дії саме з цим клієнтом будуть саме на цьому потоці.
7. Основний (перший) потік повертається на прийняття нового клієнта.

Це було описано роботу основного потоку. Нижче наведено роботу допоміжних потоків, які працюють з клієнтами.

1. Сервер отримує потік даних від клієнта.
2. Потік декодується, в результаті отримуємо призначення цих даних та самі дані.
3. Якщо призначення це надіслати просте повідомлення
 - 3.1. Сервер відправляє дані усім клієнтам у списку клієнтів.
 - 3.2. Потік повертається на отримання нових даних.
4. Якщо ж призначенням є надсилання файлу
 - 4.1. Інформація про файл розділяється, оскільки була надіслана одним потоком
 - 4.2. Інформація відправляється усім клієнтам, крім відправника.
 - 4.3. Сервер отримує сам файл від відправника.
 - 4.4. Сервер надсилає файл усім крім відправника.

- 4.5. Потік повертається на отримання нових даних.
- 5. Якщо ж призначення це закриття з'єднання
 - 5.1. Закриваємо підключення.
 - 5.2. Видаляємо клієнта зі списку клієнтів.
 - 5.3. Потік завершує свою роботу.

3.2 Архітектура роботи клієнтної частини додатку

Робота Додатку-Клієнт є іншою, хоч і також потребує багатопоточності. Загальна кількість потоків - 2. Основний (перший) - для графічного інтерфейсу, другий - роботи із сервером. Основний потік:

1. Клієнт робить запит на з'єднання із сервером.
2. Якщо з'єднання не вдалось встановити
 - 2.1. Користувачеві виводиться вікно з повідомленням про помилку.
 - 2.2. Програма завершує свою роботу.
3. Відкривається вікно із запитом отримати username користувача.
4. Серверу відправляються дані клієнта.
5. Створюється потік для подальшої роботи із сервером.
6. Запускається графічний інтерфейс основного вікна чату

Сервер-Клієнт потік:

1. Клієнт отримує потік даних від серверу.
2. Потік декодується, в результаті отримуємо призначення цих даних та самі дані.
3. Якщо призначення це надіслати просте повідомлення
 - 3.1. Клієнт додає дані у спеціальне поле графічного інтерфейсу.
 - 3.2. Потік повертається на отримання нових даних.
4. Якщо ж призначенням є надсилання файлу

- 4.1. Інформація про файл розділяється, оскільки була надіслана одним потоком, і відправляється усім клієнтам, крім відправника.
- 4.2. Користувачеві виводиться вікно, із інформацією про даний файл.
- 4.3. Якщо користувач прийняв скачування файлу
 - 4.3.1. Відкривається вікно з вибором шляху, куди зберегти файл.
 - 4.3.2. Клієнт отримує сам файл.
 - 4.3.3. Потік повертається на отримання нових даних.
- 4.4. Якщо користувач не прийняв скачування файлу
 - 4.4.1. Потік повертається на отримання нових даних.

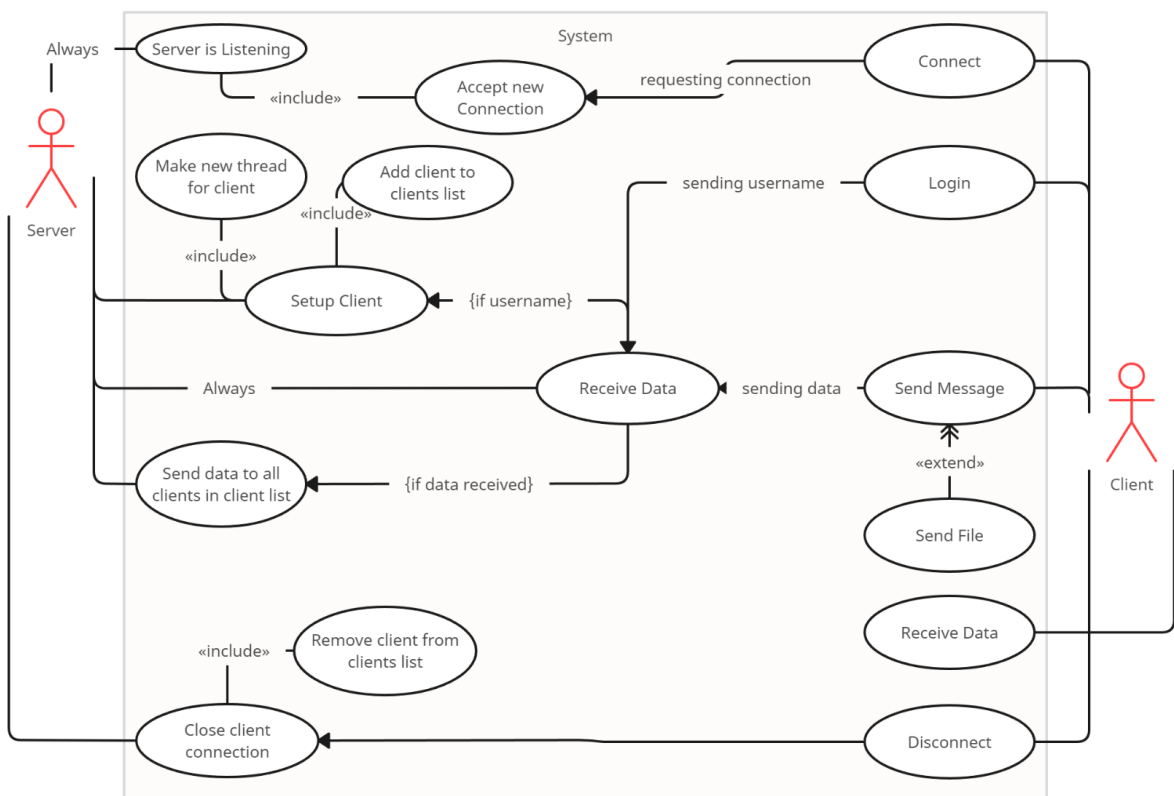
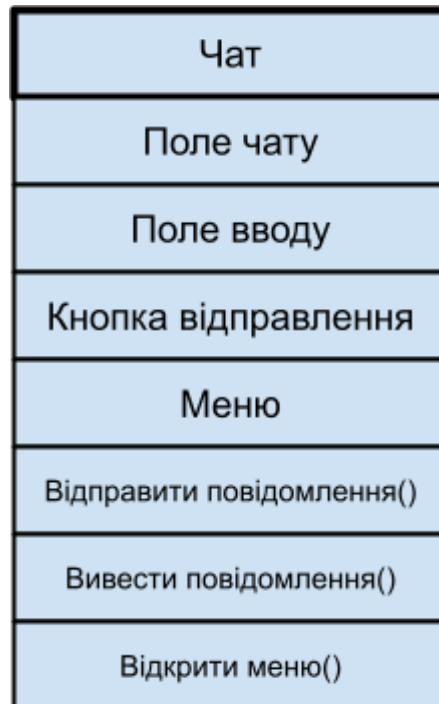


Рис. 3.2.1 Use-case діаграма системи обміну між клієнтом та сервером

3.3 Архітектура графічного інтерфейсу додатку

Із основного, графічний інтерфейс чату має включати:



Діагр. 3.3.1 Клас інтерфейсу чату

Нижче наведено максимально простий вигляд початкового інтерфейсу, із додатковою можливістю надсилання та отримання файлу.

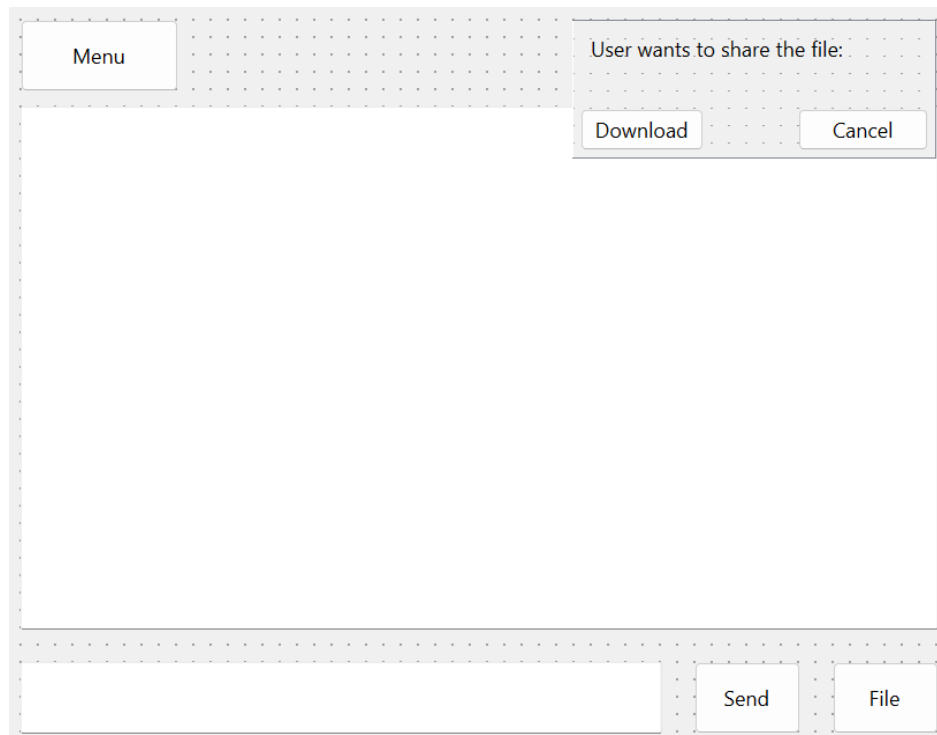


Рис. 3.3.1 Графічний інтерфейс найпростішої чат-кімнати

4. Програмне рішення

4.1 Розробка графічного інтерфейсу. Python

Для розробки графічного інтерфейсу, було вирішено використати мову Python та популярний фреймворк PyQt, а саме його версію PySide6. Дуже важливо було обрати простий і водночас ефективний спосіб створення UI (user interface). При встановленні PyQt програмісти отримують у своє розпорядження спеціальний дизайнер - Qt Designer. На жаль, він не є повноцінним, адже існує платна версія Qt Creator[3], яка має повний функціонал та можливості фреймворку PyQt, але можливостей Qt Designer достатньо, щоб створити повноцінний проект із гарним інтерфейсом.

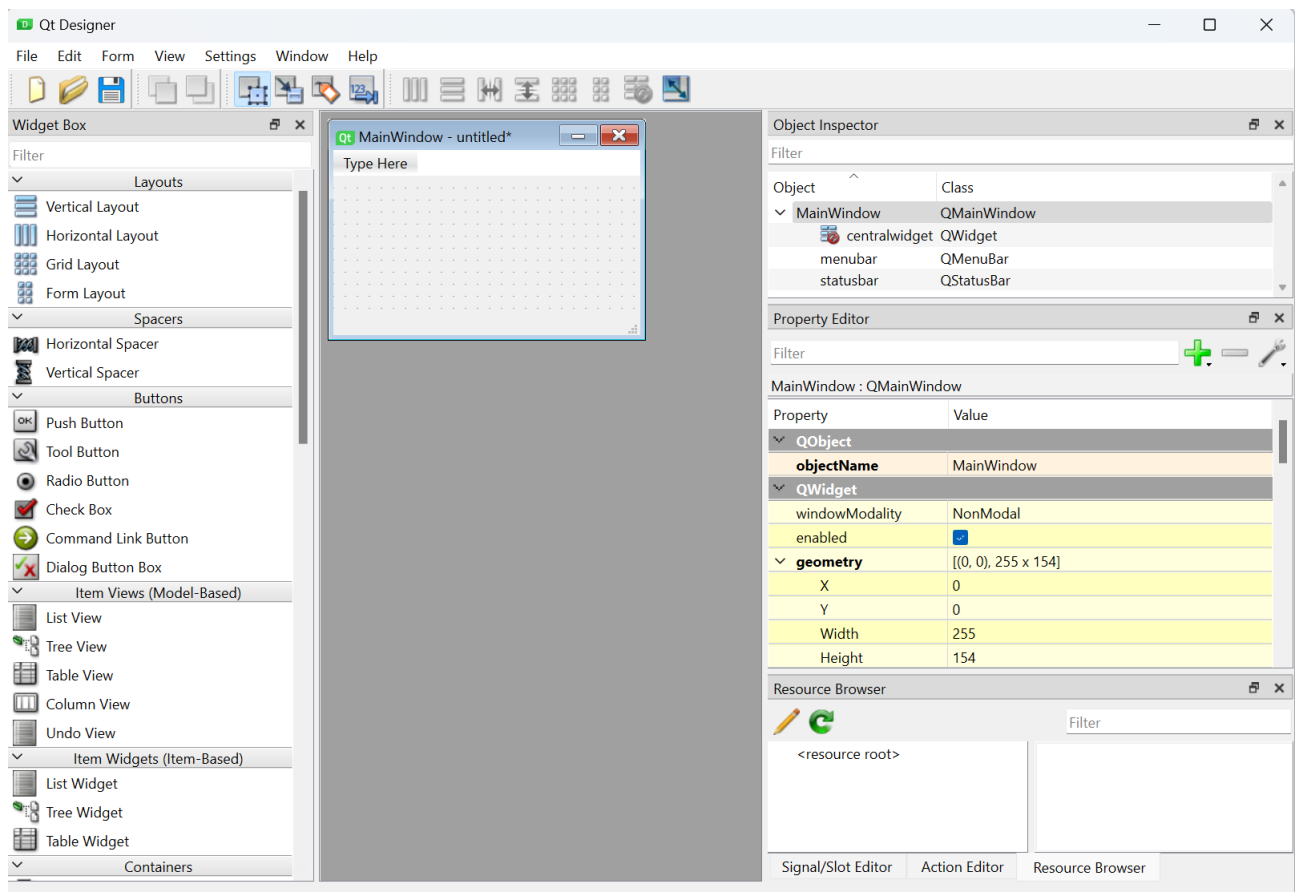


Рис. 4.1.1 Інтерфейс Qt Designer

Як можна побачити з рисунку 3.1.1 дизайнер є дуже простим у використанні. Зліва є Widgets, які є елементами основного вікна. Саме вони задають можливий функціонал інтерфейсу.

Загалом отримання готового інтерфейсу мало 5 стадій:

1. Створення початкового вікна, із мінімальною кількістю елементів.
2. Додавання функціоналу: меню, історія клієнтів, вікно емоджі.
3. Зміна історії клієнтів на клієнтів, які зараз онлайн. Додавання можливості відправки файлу, та вікна отримання файлу.
4. Використання CSS, для покращення вигляду елементів вікна.
5. Отримання фінального варіанту інтерфейсу, із використанням іконок та малюнків.

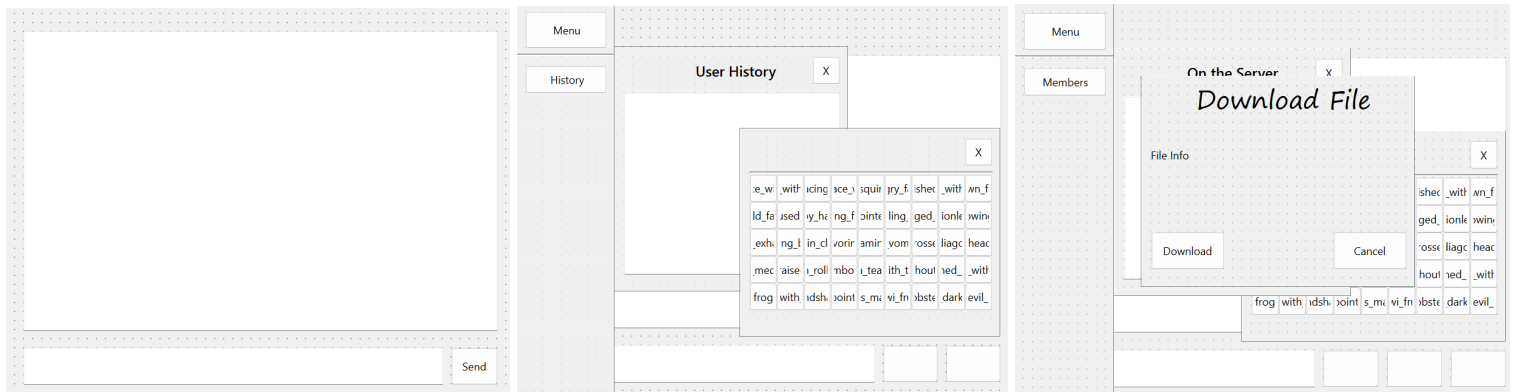


Рис. 4.1.2 Перші три стадії розробки інтерфейсу



Рис. 4.1.3 Дві останні стадії розробки інтерфейсу

4.2 Розробка серверу. Python

Для серверної частини системи, було використано два модулі `threading` та `socket`. Перший дозволяє створювати декілька потоків із одного, що дозволяє працювати із декількома клієнтами одночасно. Другий модуль є основою для зв'язку серверу та додатку-клієнт. Саме його функції дозволяють встановити з'єднання через різні типи підключень, наприклад Internet, Bluetooth, AppleTalk, та протоколи, наприклад TCP та UDP. Також важливо було розробити функцію, яка надсилатиме всім підключеним клієнтам інформацію, адже такої у модулі `socket` немає.

Для початку сервер має отримати запит на підключення від нового клієнта. Далі сервер налаштовує клієнта та надсилає усім, що новий користувач доєднався до кімнати і створює для клієнта потік, який буде приймати дані від цього користувача. У кінці сервер повертається на нову ітерацію циклу. Нижче наведено програмну реалізацію даного методу.

```
def receive():
    while True:
        client, address = server.accept()
        print(f"Server: Connected with {str(address)}")

        nickname = client.recv(1024).decode('utf-8')

        nicknames.append(nickname)
        clients.append(client)

        print(f"Server: Nickname of the client is {nickname}")

        broadcast_to_all(("%%$CHAT" + f"{nickname} joined the chat!").encode('utf-8'))
        broadcast_to_all(("%%$LIST" + str(nicknames)).encode('utf-8'))

        thread = threading.Thread(target=handle, args=(client,))
        thread.start()
```

Можна побачити, що сервер зупинити програмно неможливо, лише якщо вручну закрити програму, або припинити подачу інтернету, що видасть помилку і програма аварійно завершить свою роботу. Для цього у великих компаній є додаткові ресурси живлення та подачі інтернету. Також зазвичай у

серверних кімнатах можна побачити головний комп'ютер, через який можна закрити сервер.

4.3 Розробка Клієнту. Python.

Для клієнтної частини так само були використані модулі socket та threading, також, як вже було зазначено вище, фреймворк PySide6. Однією з особливостей роботи з графічним інтерфейсом PyQt є те, що цикл (mainloop) має бути саме на основному (першому) потоці. Інакше інтерфейс може не коректно працювати та видавати помилки. Тому було вирішено GUI залишити на основному потоці, а роботу із сервером перенести на додатковий потік.

Було розроблено методи встановлення підключення до серверу, отримання даних та декодінгу отриманого потоку даних. Також була розроблений простий спосіб визначення цілі цих даних, наприклад просте повідомлення, файл, список онлайн користувачів.

```
def handle_messages(self):
    stream = self.server.recv(1024).decode('utf-8').split("$$$") #decoding
    for item in stream: # One stream can have multiple data
        msg_status = item[:4]
        message = item[4:]

        if msg_status == "CHAT":
            self.ui.chat_viewer.append(message)

        elif msg_status == "FILE":
            full_file_info = message.split("/") # Information about file
            send_user = full_file_info[0]
            file_name = full_file_info[1]
            file_size = full_file_info[2]
            file_size = self.file_sizes(file_size)

            self.ui.download_file_area.show()
            self.ui.file_info.setText(f"From User: {send_user}\nFile name:
{file_name}\nSize: {file_size}")
            self.to_pass = full_file_info
            self.ready = False # Stop receiving regular data and receive file
            self.disable_all() # Disable GUI

        elif msg_status == "LIST":
            self.ui.online_clients_list.clear() # Clear Old Online Clients List
            for client in message[1:-1].split(", "):
                self.ui.online_clients_list.addItem(client[1:-1])
```

4.4 Розробка вікна логіну. C++

Було використано фреймворк .NET[6], який додає у Microsoft Visual Studio дизайнер, інтерфейс якого дуже схожий на Qt Designer. Особливістю даного методу можна виділити те, що після додавання елемента у головне вікно, програмний код одразу додається у відповідний .h файл. Це спрощує роботу, адже у Qt Designer ми не можемо отримати .py файл. Доступ до полів є таким самим простим як і в PySide6. Розроблене вікно має вигляд:

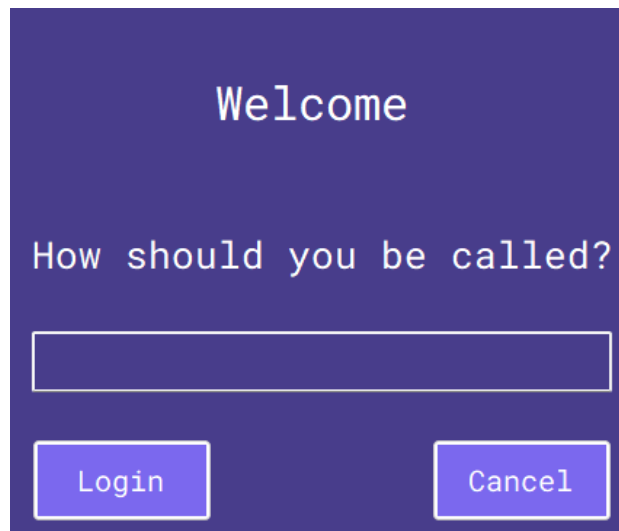


Рис. 4.4.1 Login вікно

4.5 Взаємодія різномовних модулів

Так як основним для проекту я обрав Python, C++ виступав у ролі допоміжної мови програмування. Тому потрібно було налаштувати проект у Microsoft Visual Studio, а саме тип конфігурації змінити з Application (.exe) на Dynamic Library (.dll). Це дало змогу використати функції та класи написані на C++ у середовищі Python. Для того щоб Python міг коректно працювати із цією бібліотекою був підключений модуль ctypes.

Обмін інформацією, а саме логін, виконувався за допомогою запису у файл через C++ та зчитуванні з Python. Оскільки додаток може знаходитись на будь-якому комп'ютері треба було обрати папку, яка є на кожному комп'ютері, щоб туди записувати дані. Такою папкою стала USER папка, яка відповідає за

користувача, який зараз використовує машину. Але така папка у кожного має власну назву, тому у C++ та Python є спеціальні функції для отримання її назви.

`std::getenv("USERPROFILE")` C++

`os.path.expanduser('~')` Python

4.6 Список нестандартних бібліотек/фреймворків

Python

1. PySide6
2. emoji

C++

1. .NET

5. Опис проведених експериментів

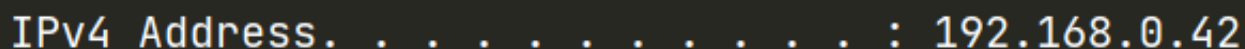
5.1 Опис дій

1. Опис дій з погляду Користувача (Тільки Додаток-Клієнт):

Користувачеві достатньо скачати .exe файл і запустити його. Додаток-клієнт має бути автоматично налаштований тому, що сервер, зазвичай, знаходиться в одному місці, а тому IP не змінює.

2. Опис дій з погляду адміністратора (Сервер та Додаток-Клієнт):

Якщо адміністратор бажає розгорнути власний сервер, йому достатньо запустити .exe файл, він автоматично налаштується, але тільки якщо є доступ в інтернет. А ось у додатку-клієнт треба змінити поле HOST, яке відповідає за IP адресу серверу, до якого буде підключено додаток. Зробити це дуже просто. На машині, яка буде сервером, достатньо подивитися IPv4, за допомогою команди ipconfig у терміналі. Приблизно це може виглядати так:



```
IPv4 Address. . . . . : 192.168.0.42
```

Рис. 5.1.1 Приблизний вигляд потрібного IPv4

Якщо ж сервер підключений до інтернету на пряму по кабелю, а не через роутер, що зазвичай і буває у серверних, IP може відрізнитися, оскільки це буде не local, а public IP address.

У кінці адміністратору достатньо конвертувати client.py із новими налаштуваннями у .exe і поширити кому треба. Зробити це також дуже просто за допомогою модуля auto-py-to-exe, де можна обрати налаштування нового .exe файлу, наприклад назву, іконку і т.д.

3. Опис дій програміста:

Для роботи із серверною частиною, потрібен лише server.py файл та IDE.

Для роботи із додатком потрібні: client.py, chat_GUI.py, app.ui, CLogin.dll, res.qrc та не стандартні бібліотеки/фреймворки PySide6, emoji, .NET. Щоб конвертувати .ui (тип для зберігання gui) та .qrc (тип для зберігання ресурсів даного gui) файли у .py достатньо у терміналі прописати:

```
pyside6-uic app.ui -o app.py
```

```
pyside6-rcc res.qrc -o res_rc.py
```

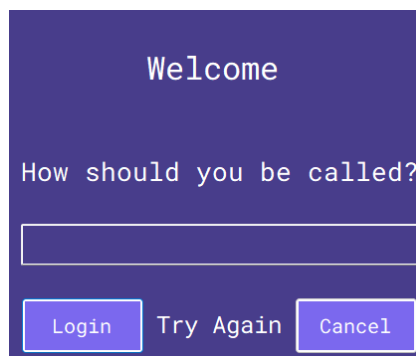
1. Називати так само не обов'язково, просто пам'ятайте, що у client.py ви будете використовувати саме конвертовані .py файли.

2. При конвертуванні .qrc файлу, в кінці назви обов'язково має бути _rc.

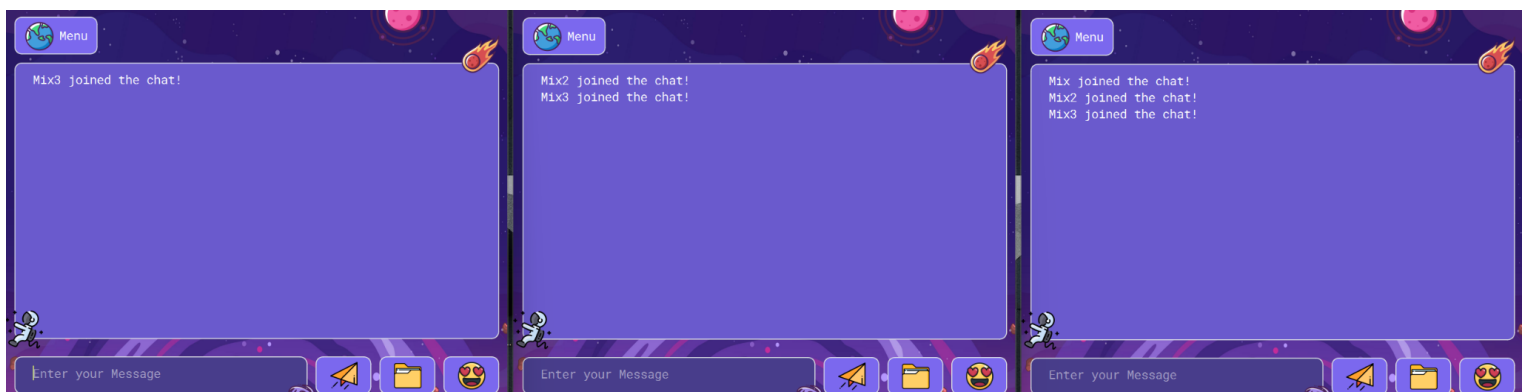
Конвертувати їх треба, щоб оновлювати зміни які були зроблені над GUI в Qt Designer або Qt Creator.

5.2 Проведені експерименти

1. Якщо спробувати зайти без username



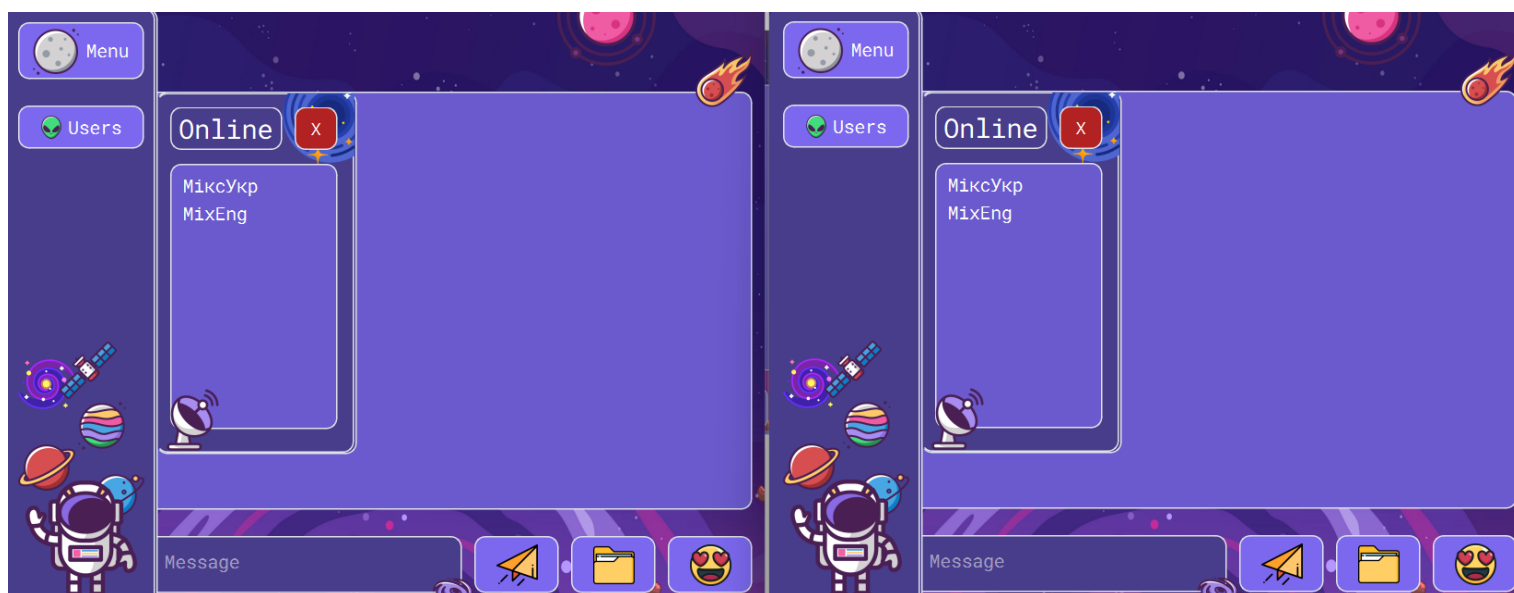
2. Приклад входу трьох клієнтів.



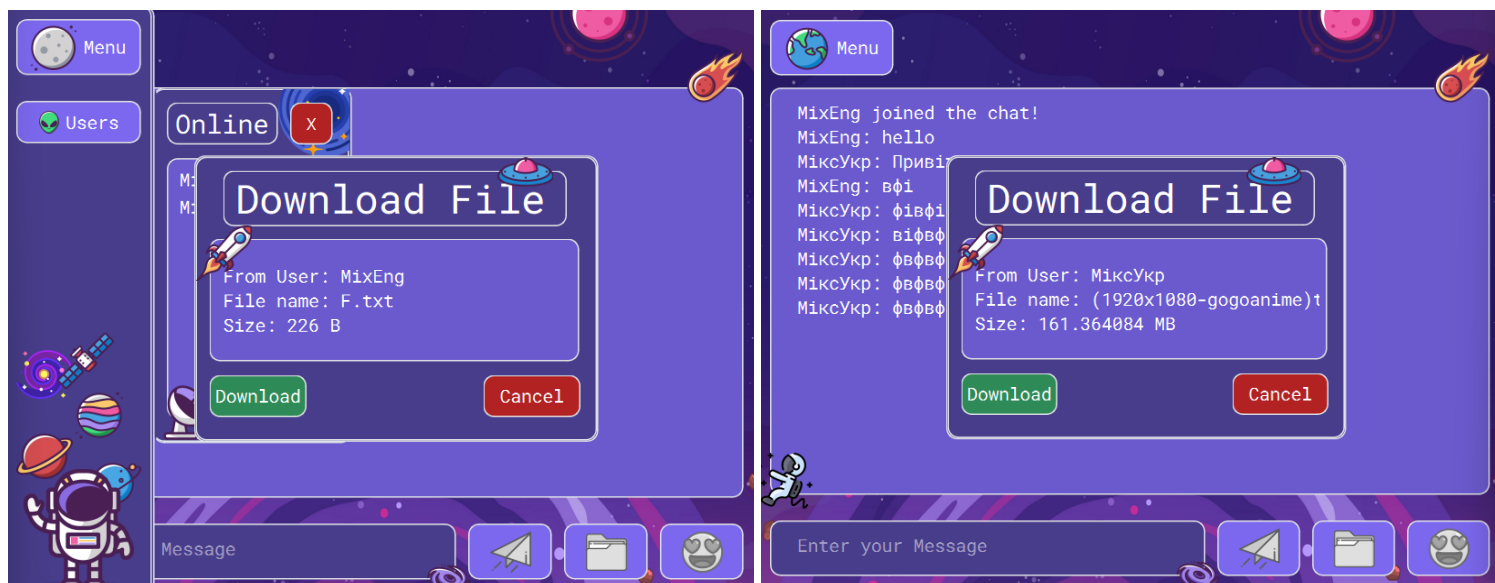
3. Приклад повідомлень (різними мовами) від двох клієнтів



4. Приклад вигляду списку онлайн користувачів



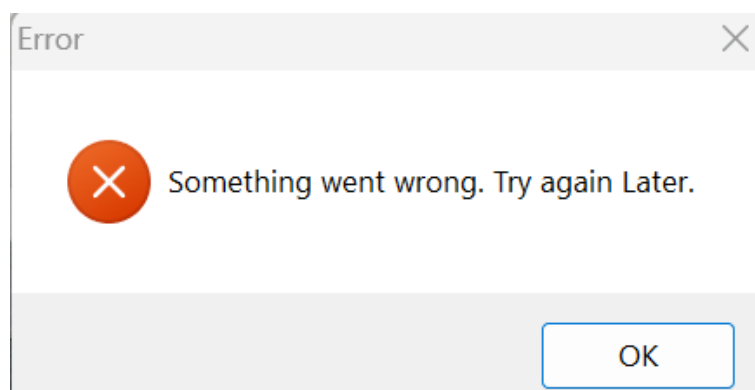
5. Приклад отримання файлу від іншого користувача



6. Приклад використання смайликів



7. Приклад помилки, якщо, наприклад сервер не працює, а клієнт намагається підключитись



Висновки

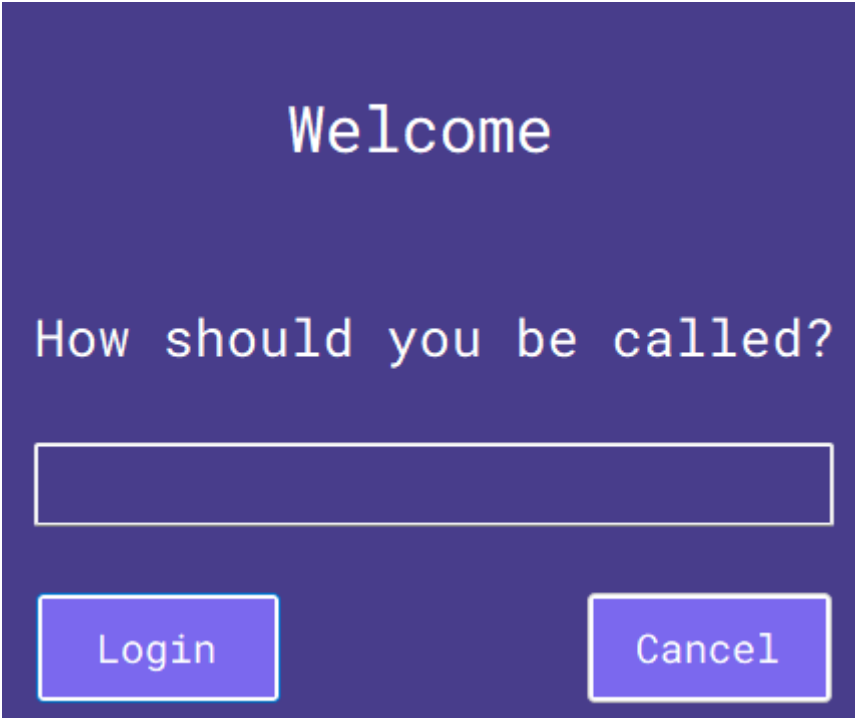
Під час виконання даної курсової роботи я ознайомився з такими технологіями як: багатопоточність (англ. threading), мережеві гнізда (англ. network sockets) та графічний інтерфейс (англ. GUI). Попрацював та розібрався у роботі популярного фреймворку PySide6 для створення графічного інтерфейсу користувача. Розробив програму із використанням цих технологій. Можливими обмеженнями додатку є передача великої кількості даних, адже це ніяк не контролюється. Перспективами розвитку програмної системи є збільшення функціоналу, покращення системи безпеки, створення бази даних користувачів та збільшення кількості серверів.

Список використаних джерел

1. 700 мільйонів користувачів та Telegram Premium. *Telegram*.
URL: <https://telegram.org/blog/700-million-and-premium/uk?ln=a>.
2. Difference between Hard real time and Soft real time system. *GeeksforGeeks*.
URL: <https://www.geeksforgeeks.org/difference-between-hard-real-time-and-soft-real-time-system/>.
3. Embedded software development tools & cross platform IDE | qt creator. *Qt | Tools for Each Stage of Software Development Lifecycle*.
URL: <https://www.qt.io/product/development-tools>.
4. Fitzpatrick M. Create GUI applications with python & qt6 (pyqt6 edition) : ebook. 2022.
5. Low-level networking interface. *Python documentation*.
URL: <https://docs.python.org/3/library/socket.html>.
6. .NET | Build. Test. Deploy. *Microsoft*.
URL: <https://dotnet.microsoft.com/en-us/>.
7. Nitro benefits and features | discord. *Discord*. URL: <https://discord.com/nitro>.
8. Python | os.path.expanduser() method. *GeeksforGeeks*.
URL: <https://www.geeksforgeeks.org/python-os-path-expanduser-method/>.
9. Rhodes B., Goerzen J. Foundations of python network programming. Berkeley, CA : Apress, 2014. URL: <https://doi.org/10.1007/978-1-4302-5855-1>.
10. The Qt Company. Qt documentation | home.
URL: <https://doc.qt.io>.
11. Thread-based parallelism. *Python documentation*.
URL: <https://docs.python.org/3/library/threading.html>.
12. What is TCP/IP?. *cloudflare.com*.
URL: <https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/>.
13. What is UDP?. *cloudflare.com*. URL:
<https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/>

Додатки

Додаток 1. Повний графічний інтерфейс

A dark blue dialog box with a white border. It contains the text "Welcome" in a large, white, sans-serif font. Below it is the text "How should you be called?" in a smaller, white, sans-serif font. Underneath is a white rectangular input field. At the bottom are two white buttons with blue borders: "Login" on the left and "Cancel" on the right.

Welcome

How should you be called?

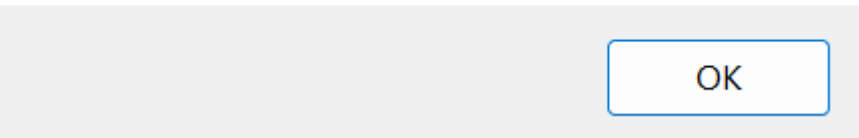
Login Cancel

A light gray horizontal bar with a white border. It contains the text "Error" on the left and a white "X" icon on the right.

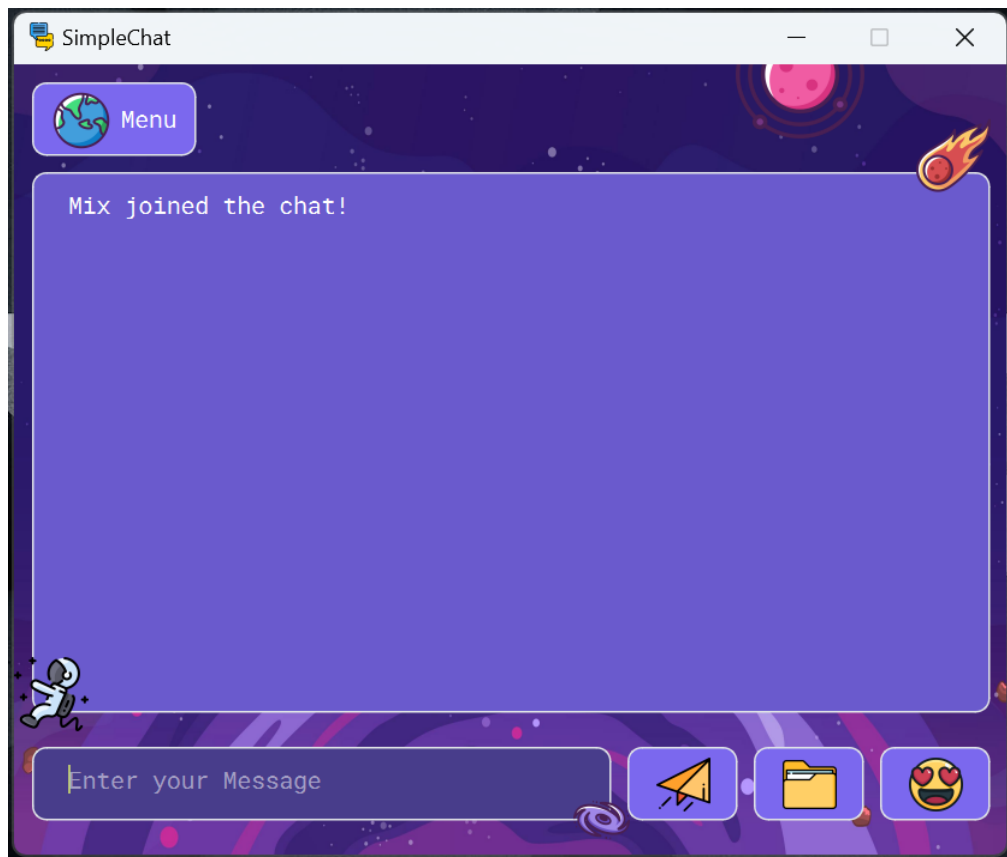
Error X

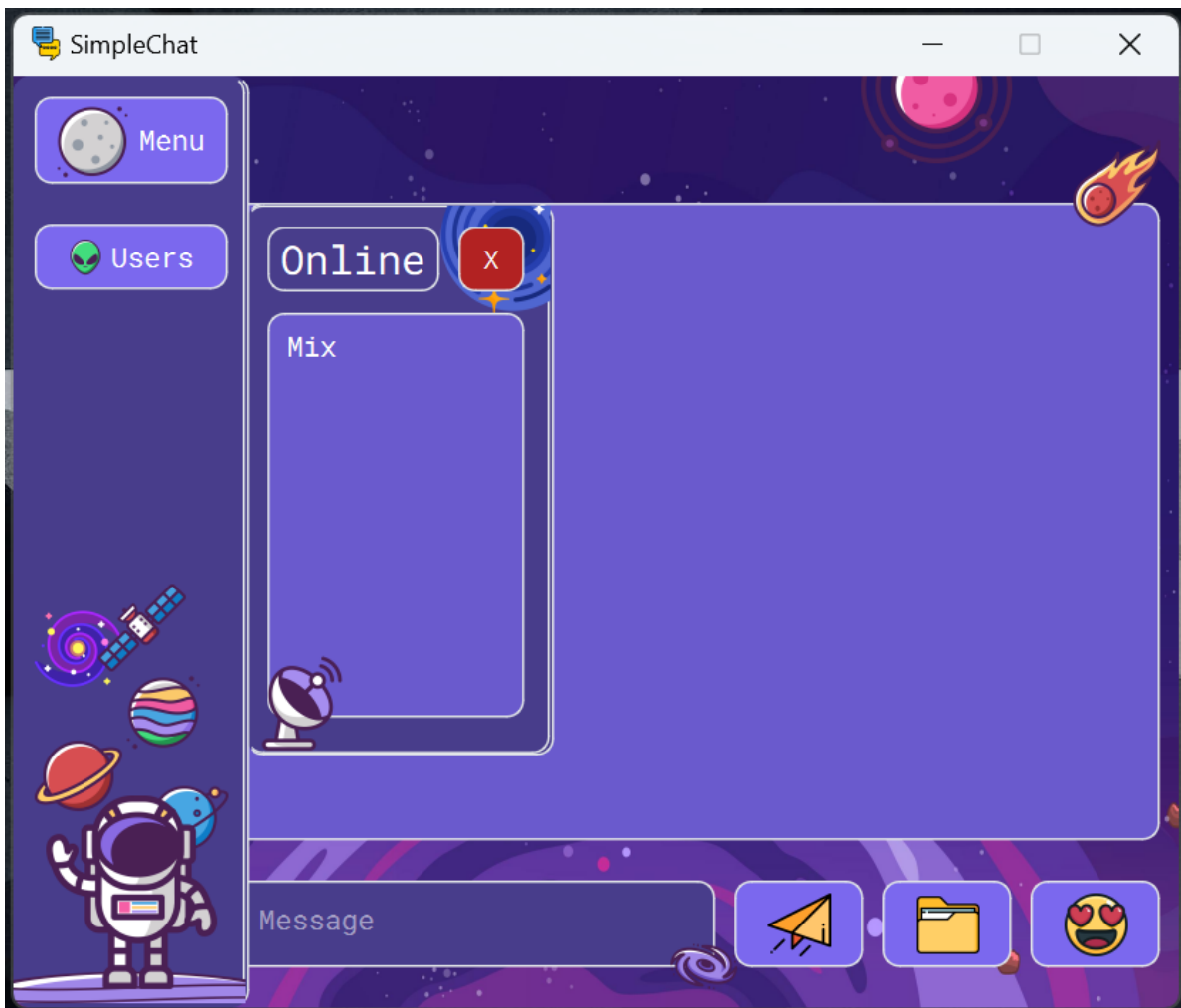
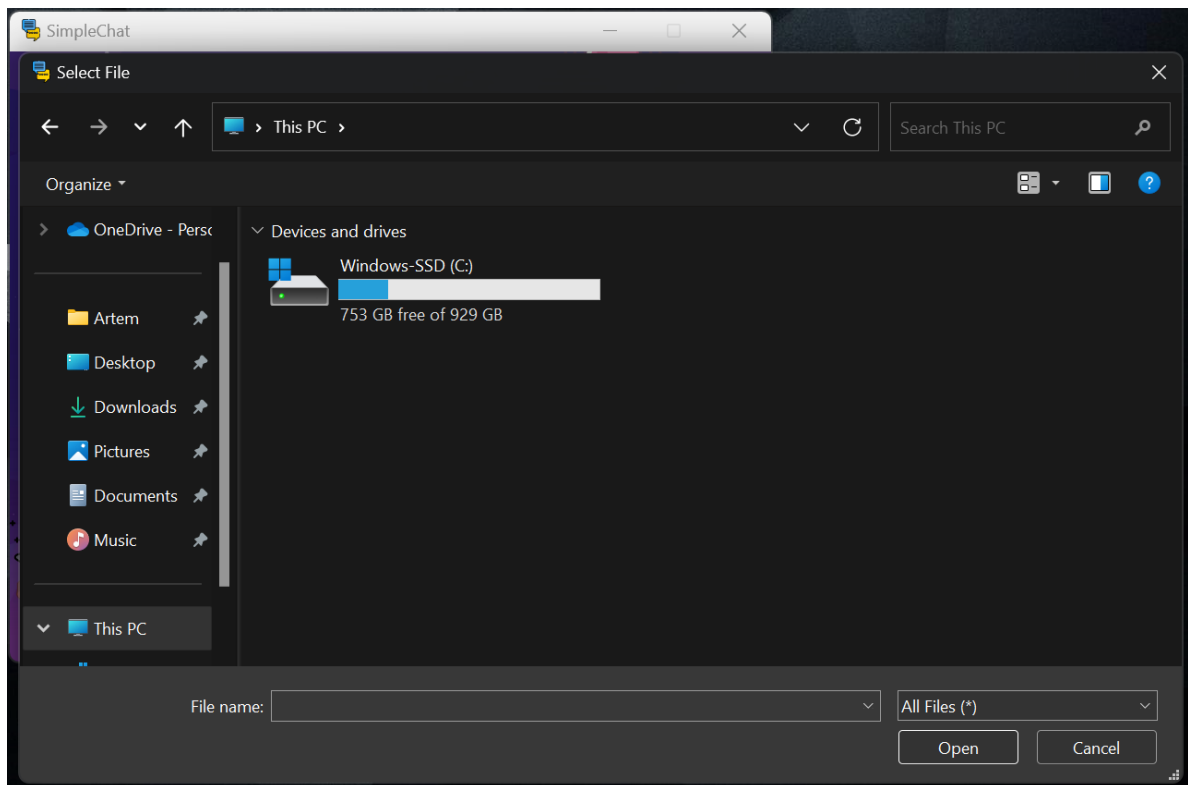


Something went wrong. Try again Later.

A light gray horizontal bar with a white border. It contains a single white button with a blue border and the text "OK" inside.

OK





Додаток 2. Програмний модуль server.py

```
import threading
import socket as sk

HOST = sk.gethostbyname(sk.gethostname())
PORT = 9100

server = sk.socket(sk.AF_INET, sk.SOCK_STREAM)
server.bind((HOST, PORT))
server.listen()

clients = []
nicknames = []

def broadcast_to_all(message):
    for client in clients:
        client.send(message)

def broadcast_with_exception(file_data, exception):
    for client in clients:
        if client != exception:
            client.send(file_data)

def close_connection(client):
    index = clients.index(client)
    nickname = nicknames[index]

    print(f"Server: {nickname} left the server")

    clients.remove(client)
    client.close()
    nicknames.remove(nickname)

    broadcast_to_all(("$$CHAT" + f"{nickname} left the chat!").encode('utf-8'))
    broadcast_to_all(("$$LIST" + str(nicknames)).encode('utf-8'))

def handle(client):
    while True:
        try:
            handle_messages(client)
        except:
            close_connection(client)
            break

def handle_messages(client):
    stream = client.recv(1024).decode('utf-8')
```

```

client_msg_status = stream[:5]
message = stream[5:]

if client_msg_status == "CHAT:":
    broadcast_to_all(("$$CHAT" + message).encode('utf-8'))

elif client_msg_status == "FILE:":
    full_file = message.split("/") # Information about file: name/size
    file_name = full_file[0]
    file_size = full_file[1]

    file_data = client.recv(int(file_size)) # Receiving file
    index = clients.index(client)

    broadcast_with_exception(f"$$FILE{nicknames[index]}/{file_name}/{file_size}".encode('utf-8'),
client)
    broadcast_with_exception(file_data, client)

def receive():
    while True:
        client, address = server.accept()
        print(f"Server: Connected with {str(address)}")

        nickname = client.recv(1024).decode('utf-8')

        nicknames.append(nickname)
        clients.append(client)

        print(f"Server: Nickname of the client is {nickname}")

        broadcast_to_all(("$$CHAT" + f"{nickname} joined the chat!").encode('utf-8'))
        broadcast_to_all(("$$LIST" + str(nicknames)).encode('utf-8'))

        thread = threading.Thread(target=handle, args=(client,)) # Thread that will take care of
client
        thread.start()

# MAIN
print("Server is ready")
receive()

```

Додаток 3. Програмний модуль client.py

```
import os
import sys
from time import sleep

import socket as sk
import threading

from PySide6.QtWidgets import QApplication
from chat_GUI import Chat_GUI

import ctypes

lib = ctypes.CDLL('./CLogin.dll')

HOST = "192.168.0.42"
PORT = 9100

class Client_App(Chat_GUI):
    def __init__(self, host, port):
        self.server = sk.socket(sk.AF_INET, sk.SOCK_STREAM)
        self.server_connection(host, port)
        self.username = self.login() # C++ Login

        super().__init__(self.username, self.server) # Initialize GUI

        self.server.send(self.username.encode('utf-8'))

        receive_thread = threading.Thread(target=self.receive)
        receive_thread.start()

    def server_connection(self, host, port):
        try:
            self.server.connect((host, port))
        except TimeoutError: # If error is that client cannot connect to the server
            ctypes.windll.user32.MessageBoxW\
                (0, "Could not connect to the server. Try again Later.", "Connection Error", 16)
            sys.exit()
        except: # Unknown Error
            ctypes.windll.user32.MessageBoxW\
                (0, "Something went wrong. Try again Later.", "Error", 16)
            sys.exit()

    def receive(self):
        while True:
            if self.ready:
                try:
                    self.handle_messages()
                except UnicodeDecodeError: # When garbage or too much data received, it cannot be
decoded
```

```

        continue
    except:
        self.ui.chat_viewer.append(
            "SERVER: *** Something went wrong. You're disconnected from the server ***")
        self.server.close()
        break
    else:
        sleep(1) # To make less comparisons, less loading

def handle_messages(self):
    stream = self.server.recv(1024).decode('utf-8').split("$$$") # decoding
    for item in stream: # One stream can have multiple data
        msg_status = item[:4]
        message = item[4:]

        if msg_status == "CHAT":
            self.ui.chat_viewer.append(message)

        elif msg_status == "FILE":
            full_file_info = message.split("/") # Information about file:
            client_who_sent/name/size

            send_user = full_file_info[0]
            file_name = full_file_info[1]
            file_size = full_file_info[2]
            file_size = self.file_sizes(file_size)

            self.ui.download_file_area.show()
            self.ui.file_info.setText(f"From User: {send_user}\nFile name: {file_name}\nSize:
{file_size}")

            self.to_pass = full_file_info
            self.ready = False # Stop receiving regular data and receive file data
            self.disable_all() # Disable GUI

        elif msg_status == "LIST":
            self.ui.online_clients_list.clear() # Clear Old Online Clients List
            for client in message[1:-1].split(", "):
                self.ui.online_clients_list.addItem(client[1:-1])

# C++ Login
def login(self):
    file_path = os.path.expanduser('~') # Getting USER folder
    file_path += "\\login_data.txt"

    try:
        lib.login() # C++ function that creates file with username inside USER folder
        file = open(file_path, "r")
        username = file.read()
        file.close()

        os.remove(file_path) # Deleting file

```



```

except:
    sys.exit() # If Login was Canceled
return username

# OTHER
def file_sizes(self, file_size): # Create string with right File Size
    if int(file_size) <= 1000:
        file_size += " B"
    elif 1000 <= int(file_size) <= 1000000:
        file_size = str(int(file_size) / 1000)
        file_size += " KB"
    else:
        file_size = str(int(file_size) / 1000000)
        file_size += " MB"

    return file_size

# MAIN
app = QApplication(sys.argv)
client = Client_App(HOST, PORT)
client.show()
app.exec()

```

Додаток 4. Програмний модуль chat_GUI.py

```

import os
from emoji import emoji

from PySide6.QtWidgets import QMainWindow, QFileDialog
from PySide6.QtCore import Qt
from PySide6.QtGui import QCloseEvent

from app import Ui_MainWindow

class Chat_GUI(QMainWindow):
    def __init__(self, username, server):
        super(Chat_GUI, self).__init__()
        self.setFixedSize(550, 440)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self) # Main Components
        self.set_ui() # Additional Settings

        self.username = username
        self.server = server
        self.to_pass = None # Useful variable to pass data to From Client to GUI
        self.ready = True # Variable to stop receiving when needed

```

```

# GUI METHODS
def send_message(self):
    message = f"CHAT:{self.username}: {self.ui.chat_input.toPlainText()}"
    self.server.send(message.encode('utf-8'))
    self.ui.chat_input.clear()

def send_file(self):
    full_path = QFileDialog.getOpenFileName(self, "Select File", os.getcwd())

    # If user decided to cancel
    try:
        p_file = open(full_path[0], "rb")
    except:
        return

    file_size = os.path.getsize(full_path[0])
    file_path, file_name = os.path.split(full_path[0])

    self.server.send(f"FILE:{file_name}/{file_size}".encode('utf-8'))

    data = p_file.read()
    self.server.sendall(data)

    p_file.close()

def accept_download_file(self):
    file_name = self.to_pass[1]
    file_size = self.to_pass[2]

    full_path = QFileDialog.getSaveFileName(self, "Save File", os.getcwd() + f"/{file_name}") #
Where to save file

    file_data = self.server.recv(int(file_size))

    file = open(full_path[0], "wb")
    file.write(file_data)
    file.close()

    self.ready_to_recv() # Ready to receive new regular data

def add_emoji(self, emoji_text):
    self.ui.chat_input.insertPlainText(emojize(emoji_text))

def close_menu(self):
    self.ui.online_clients_area.hide()
    self.ui.menu_area.hide()

def scroll_to_bottom(self): # Chat Viewer Always at the bottom, to see new messages
self.ui.chat_viewer.verticalScrollBar().setValue(self.ui.chat_viewer.verticalScrollBar().maximum())

def closeEvent(self, event: QCloseEvent): # GUI closed

```

```

self.server.close()

def set_ui(self):
    # AREAS
    self.ui.menu_area.hide()
    self.ui.emoji_area.hide()
    self.ui.online_clients_area.hide()
    self.ui.download_file_area.hide()

    # SCROLLBAR
    self.ui.chat_input.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
    self.ui.chat_viewer.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
    self.ui.chat_viewer.verticalScrollBar().rangeChanged.connect(self.scroll_to_bottom)

    # BUTTONS
    # MENU
    self.ui.menu_open_button.clicked.connect(self.ui.menu_area.show)
    self.ui.menu_close_button.clicked.connect(self.close_menu)

    # HISTORY
    self.ui.online_clients_open_button.clicked.connect(self.ui.online_clients_area.show)
    self.ui.online_clients_close_button.clicked.connect(self.ui.online_clients_area.hide)

    # FILE
    self.ui.download_button.clicked.connect(self.accept_download_file)
    self.ui.cancel_download_button.clicked.connect(self.ready_to_recv)

    # SEND
    self.ui.file_send_button.clicked.connect(self.send_file)
    self.ui.send_button.clicked.connect(self.send_message)

    # EMOJIS
    self.ui.emoji_open_button.clicked.connect(self.ui.emoji_area.show)
    self.ui.emoji_close_button.clicked.connect(self.ui.emoji_area.hide)

    emoji_buttons = \
        [self.ui.emoji_1, self.ui.emoji_2, self.ui.emoji_3, self.ui.emoji_4, self.ui.emoji_5,
self.ui.emoji_6,
        self.ui.emoji_7, self.ui.emoji_8, self.ui.emoji_9,
        self.ui.emoji_10, self.ui.emoji_11, self.ui.emoji_12, self.ui.emoji_13, self.ui.emoji_14,
self.ui.emoji_15,
        self.ui.emoji_16, self.ui.emoji_17, self.ui.emoji_18,
        self.ui.emoji_19, self.ui.emoji_20, self.ui.emoji_21, self.ui.emoji_22, self.ui.emoji_23,
self.ui.emoji_24,
        self.ui.emoji_25, self.ui.emoji_26, self.ui.emoji_27,
        self.ui.emoji_28, self.ui.emoji_29, self.ui.emoji_30, self.ui.emoji_31, self.ui.emoji_32,
self.ui.emoji_33,
        self.ui.emoji_34, self.ui.emoji_35, self.ui.emoji_36,
        self.ui.emoji_37, self.ui.emoji_38, self.ui.emoji_39, self.ui.emoji_40, self.ui.emoji_41,
self.ui.emoji_42,
        self.ui.emoji_43, self.ui.emoji_44, self.ui.emoji_45]

```

```

        for emoji_button in emoji_buttons:
            emoji_button.clicked.connect(lambda x=None, text=emoji_button.text():
self.add_emoji(text))

        for emoji_button in emoji_buttons:
            emoji_button.setText(emojize(emoji_button.text()))

# OTHER
def ready_to_recv(self):
    self.ui.download_file_area.hide()
    self.ready = True
    self.enable_all()

def disable_all(self):
    self.ui.chat_viewer.setEnabled(False)
    self.ui.chat_input.setEnabled(False)
    self.ui.send_button.setEnabled(False)
    self.ui.file_send_button.setEnabled(False)
    self.ui.emoji_open_button.setEnabled(False)

def enable_all(self):
    self.ui.chat_viewer.setEnabled(True)
    self.ui.chat_input.setEnabled(True)
    self.ui.send_button.setEnabled(True)
    self.ui.file_send_button.setEnabled(True)
    self.ui.emoji_open_button.setEnabled(True)

```

Додаток 5. Програмний модуль app.py

```

# -*- coding: utf-8 -*-

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QAbstractScrollArea, QApplication, QLabel, QListWidget,
    QListWidgetItem, QMainWindow, QPushButton, QScrollArea,
    QSizePolicy, QTabWidget, QTextBrowser, QTextEdit,
    QWidget)
import res_rc

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.setObjectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(550, 440)
        icon = QIcon()

```

```

        icon.addFile(u":/icons/Icons/chat.png", QSize(), QIcon.Normal, QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setStyleSheet(u"background-color: #483D8B;\n"
"color: #FFFFFF")
        MainWindow.setTabShape(QTabWidget.Rounded)
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.chat_viewer = QTextBrowser(self.centralwidget)
        self.chat_viewer.setObjectName(u"chat_viewer")
        self.chat_viewer.setGeometry(QRect(10, 60, 531, 301))
        font = QFont()
        font.setFamilies([u"Roboto Mono"])
        font.setPointSize(10)
        self.chat_viewer.setFont(font)
        self.chat_viewer.viewport().setProperty("cursor", QCursor(Qt.PointingHandCursor))
        self.chat_viewer.setFocusPolicy(Qt.NoFocus)
        self.chat_viewer.setLayoutDirection(Qt.LeftToRight)
        self.chat_viewer.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding: 5px 15px;\n"
"color: #FFFFFF;\n"
"background-color: #6A5ACD")
        self.chat_viewer.setVerticalScrollBarPolicy(Qt.ScrollBarAsNeeded)
        self.chat_input = QTextEdit(self.centralwidget)
        self.chat_input.setObjectName(u"chat_input")
        self.chat_input.setGeometry(QRect(10, 380, 321, 41))
        self.chat_input.setFont(font)
        self.chat_input.viewport().setProperty("cursor", QCursor(Qt.IBeamCursor))
        self.chat_input.setLayoutDirection(Qt.LeftToRight)
        self.chat_input.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding-right: 100px;\n"
"padding: 5px 15px;\n"
"color: #FFFFFF")
        self.chat_input.setSizeAdjustPolicy(QAbstractScrollArea.AdjustIgnored)
        self.send_button = QPushButton(self.centralwidget)
        self.send_button.setObjectName(u"send_button")
        self.send_button.setGeometry(QRect(340, 380, 61, 41))
        self.send_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding: 5px 15px;\n"
"color: #FFFFFF;\n"
"background-color: #7B68EE")
        icon1 = QIcon()
        icon1.addFile(u":/icons/Icons/send (1).png", QSize(), QIcon.Normal, QIcon.Off)
        self.send_button.setIcon(icon1)
        self.send_button.setIconSize(QSize(30, 30))
        self.emoji_area = QScrollArea(self.centralwidget)
        self.emoji_area.setObjectName(u"emoji_area")
        self.emoji_area.setGeometry(QRect(250, 140, 291, 231))
        self.emoji_area.setFocusPolicy(Qt.NoFocus)
        self.emoji_area.setContextMenuPolicy(Qt.NoContextMenu)

```

```

        self.emoji_area.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"\n"
"color: #FFFFFF")
        self.emoji_area.setVerticalScrollBarPolicy(Qt.ScrollBarAsNeeded)
        self.emoji_area.setHorizontalScrollBarPolicy(Qt.ScrollBarAsNeeded)
        self.emoji_area.setWidgetResizable(False)
        self.scrollAreaWidgetContents = QWidget()
        self.scrollAreaWidgetContents.setObjectName(u"scrollAreaWidgetContents")
        self.scrollAreaWidgetContents.setGeometry(QRect(0, 0, 289, 229))
        self.emoji_1 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_1.setObjectName(u"emoji_1")
        self.emoji_1.setGeometry(QRect(10, 50, 31, 31))
        self.emoji_close_button = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_close_button.setObjectName(u"emoji_close_button")
        self.emoji_close_button.setGeometry(QRect(250, 10, 31, 31))
        self.emoji_close_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"color: #FFFFFF;\n"
"background-color: #B22222;")
        self.emoji_2 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_2.setObjectName(u"emoji_2")
        self.emoji_2.setGeometry(QRect(40, 50, 31, 31))
        self.emoji_3 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_3.setObjectName(u"emoji_3")
        self.emoji_3.setGeometry(QRect(70, 50, 31, 31))
        self.emoji_4 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_4.setObjectName(u"emoji_4")
        self.emoji_4.setGeometry(QRect(100, 50, 31, 31))
        self.emoji_5 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_5.setObjectName(u"emoji_5")
        self.emoji_5.setGeometry(QRect(130, 50, 31, 31))
        self.emoji_6 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_6.setObjectName(u"emoji_6")
        self.emoji_6.setGeometry(QRect(160, 50, 31, 31))
        self.emoji_7 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_7.setObjectName(u"emoji_7")
        self.emoji_7.setGeometry(QRect(190, 50, 31, 31))
        self.emoji_8 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_8.setObjectName(u"emoji_8")
        self.emoji_8.setGeometry(QRect(220, 50, 31, 31))
        self.emoji_9 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_9.setObjectName(u"emoji_9")
        self.emoji_9.setGeometry(QRect(250, 50, 31, 31))
        self.emoji_14 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_14.setObjectName(u"emoji_14")
        self.emoji_14.setGeometry(QRect(130, 80, 31, 31))
        self.emoji_10 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_10.setObjectName(u"emoji_10")
        self.emoji_10.setGeometry(QRect(10, 80, 31, 31))
        self.emoji_17 = QPushButton(self.scrollAreaWidgetContents)
        self.emoji_17.setObjectName(u"emoji_17")

```

```

self.emoji_17.setGeometry(QRect(220, 80, 31, 31))
self.emoji_13 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_13.setObjectName(u"emoji_13")
self.emoji_13.setGeometry(QRect(100, 80, 31, 31))
self.emoji_18 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_18.setObjectName(u"emoji_18")
self.emoji_18.setGeometry(QRect(250, 80, 31, 31))
self.emoji_11 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_11.setObjectName(u"emoji_11")
self.emoji_11.setGeometry(QRect(40, 80, 31, 31))
self.emoji_15 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_15.setObjectName(u"emoji_15")
self.emoji_15.setGeometry(QRect(160, 80, 31, 31))
self.emoji_12 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_12.setObjectName(u"emoji_12")
self.emoji_12.setGeometry(QRect(70, 80, 31, 31))
self.emoji_16 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_16.setObjectName(u"emoji_16")
self.emoji_16.setGeometry(QRect(190, 80, 31, 31))
self.emoji_23 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_23.setObjectName(u"emoji_23")
self.emoji_23.setGeometry(QRect(130, 110, 31, 31))
self.emoji_19 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_19.setObjectName(u"emoji_19")
self.emoji_19.setGeometry(QRect(10, 110, 31, 31))
self.emoji_26 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_26.setObjectName(u"emoji_26")
self.emoji_26.setGeometry(QRect(220, 110, 31, 31))
self.emoji_22 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_22.setObjectName(u"emoji_22")
self.emoji_22.setGeometry(QRect(100, 110, 31, 31))
self.emoji_27 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_27.setObjectName(u"emoji_27")
self.emoji_27.setGeometry(QRect(250, 110, 31, 31))
self.emoji_20 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_20.setObjectName(u"emoji_20")
self.emoji_20.setGeometry(QRect(40, 110, 31, 31))
self.emoji_24 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_24.setObjectName(u"emoji_24")
self.emoji_24.setGeometry(QRect(160, 110, 31, 31))
self.emoji_21 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_21.setObjectName(u"emoji_21")
self.emoji_21.setGeometry(QRect(70, 110, 31, 31))
self.emoji_25 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_25.setObjectName(u"emoji_25")
self.emoji_25.setGeometry(QRect(190, 110, 31, 31))
self.emoji_32 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_32.setObjectName(u"emoji_32")
self.emoji_32.setGeometry(QRect(130, 140, 31, 31))
self.emoji_28 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_28.setObjectName(u"emoji_28")
self.emoji_28.setGeometry(QRect(10, 140, 31, 31))

```

```

self.emoji_35 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_35.setObjectName(u"emoji_35")
self.emoji_35.setGeometry(QRect(220, 140, 31, 31))
self.emoji_31 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_31.setObjectName(u"emoji_31")
self.emoji_31.setGeometry(QRect(100, 140, 31, 31))
self.emoji_36 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_36.setObjectName(u"emoji_36")
self.emoji_36.setGeometry(QRect(250, 140, 31, 31))
self.emoji_29 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_29.setObjectName(u"emoji_29")
self.emoji_29.setGeometry(QRect(40, 140, 31, 31))
self.emoji_33 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_33.setObjectName(u"emoji_33")
self.emoji_33.setGeometry(QRect(160, 140, 31, 31))
self.emoji_30 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_30.setObjectName(u"emoji_30")
self.emoji_30.setGeometry(QRect(70, 140, 31, 31))
self.emoji_34 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_34.setObjectName(u"emoji_34")
self.emoji_34.setGeometry(QRect(190, 140, 31, 31))
self.emoji_41 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_41.setObjectName(u"emoji_41")
self.emoji_41.setGeometry(QRect(130, 170, 31, 31))
self.emoji_37 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_37.setObjectName(u"emoji_37")
self.emoji_37.setGeometry(QRect(10, 170, 31, 31))
self.emoji_44 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_44.setObjectName(u"emoji_44")
self.emoji_44.setGeometry(QRect(220, 170, 31, 31))
self.emoji_40 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_40.setObjectName(u"emoji_40")
self.emoji_40.setGeometry(QRect(100, 170, 31, 31))
self.emoji_45 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_45.setObjectName(u"emoji_45")
self.emoji_45.setGeometry(QRect(250, 170, 31, 31))
self.emoji_38 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_38.setObjectName(u"emoji_38")
self.emoji_38.setGeometry(QRect(40, 170, 31, 31))
self.emoji_42 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_42.setObjectName(u"emoji_42")
self.emoji_42.setGeometry(QRect(160, 170, 31, 31))
self.emoji_39 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_39.setObjectName(u"emoji_39")
self.emoji_39.setGeometry(QRect(70, 170, 31, 31))
self.emoji_43 = QPushButton(self.scrollAreaWidgetContents)
self.emoji_43.setObjectName(u"emoji_43")
self.emoji_43.setGeometry(QRect(190, 170, 31, 31))
self.base_icon = QLabel(self.scrollAreaWidgetContents)
self.base_icon.setObjectName(u"base_icon")
self.base_icon.setGeometry(QRect(200, 10, 41, 41))
self.base_icon.setStyleSheet(u"background-color: transparent;\n"

```



```

"border: 0px;")
    self.base_icon.setPixmap(QPixmap(u":/icons/Icons/space-station.png"))
    self.base_icon.setScaledContents(True)
    self.scope_icon = QLabel(self.scrollAreaWidgetContents)
    self.scope_icon.setObjectName(u"scope_icon")
    self.scope_icon.setGeometry(QRect(20, 200, 31, 31))
    self.scope_icon.setStyleSheet(u"background-color: transparent;\n"
"border: 0px;")
    self.scope_icon.setPixmap(QPixmap(u":/icons/Icons/telescope.png"))
    self.scope_icon.setScaledContents(True)
    self.emoji_area.setWidget(self.scrollAreaWidgetContents)
    self.emoji_open_button = QPushButton(self.centralwidget)
    self.emoji_open_button.setObjectName(u"emoji_open_button")
    self.emoji_open_button.setGeometry(QRect(480, 380, 61, 41))
    self.emoji_open_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding: 5px 15px;\n"
"color: #FFFFFF;\n"
"background-color: #7B68EE;")
    icon2 = QIcon()
    icon2.addFile(u":/icons/Icons/love.png", QSize(), QIcon.Normal, QIcon.Off)
    self.emoji_open_button.setIcon(icon2)
    self.emoji_open_button.setIconSize(QSize(30, 30))
    self.menu_area = QScrollArea(self.centralwidget)
    self.menu_area.setObjectName(u"menu_area")
    self.menu_area.setGeometry(QRect(0, 0, 111, 440))
    self.menu_area.setStyleSheet(u"border-radius: 8px;\n"
"border-right: 1px solid #e0e4e7;\n"
"color: #FFFFFF")
    self.menu_area.setWidgetResizable(True)
    self.scrollAreaWidgetContents_2 = QWidget()
    self.scrollAreaWidgetContents_2.setObjectName(u"scrollAreaWidgetContents_2")
    self.scrollAreaWidgetContents_2.setGeometry(QRect(0, 0, 109, 438))
    self.menu_close_button = QPushButton(self.scrollAreaWidgetContents_2)
    self.menu_close_button.setObjectName(u"menu_close_button")
    self.menu_close_button.setGeometry(QRect(10, 10, 91, 41))
    self.menu_close_button.setFont(font)
    self.menu_close_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"color: #FFFFFF;\n"
"background-color: #7B68EE;")
    icon3 = QIcon()
    icon3.addFile(u":/icons/Icons/mercury.png", QSize(), QIcon.Normal, QIcon.Off)
    self.menu_close_button.setIcon(icon3)
    self.menu_close_button.setIconSize(QSize(40, 40))
    self.online_clients_open_button = QPushButton(self.scrollAreaWidgetContents_2)
    self.online_clients_open_button.setObjectName(u"online_clients_open_button")
    self.online_clients_open_button.setGeometry(QRect(10, 70, 91, 31))
    self.online_clients_open_button.setFont(font)
    self.online_clients_open_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding: 5px 15px;\n"

```

```

"color: #FFFFFF;\n"
"background-color: #7B68EE;")
    icon4 = QIcon()
    icon4.addFile(u":/icons/Icons/alien.png", QSize(), QIcon.Normal, QIcon.Off)
    self.online_clients_open_button.setIcon(icon4)
    self.online_clients_open_button.setIconSize(QSize(20, 20))
    self.label = QLabel(self.scrollAreaWidgetContents_2)
    self.label.setObjectName(u"label")
    self.label.setGeometry(QRect(10, 340, 91, 91))
    self.label.setStyleSheet(u"background-color: transparent;\n"
"border-right: 0px;")
    self.label.setPixmap(QPixmap(u":/icons/Icons/astronaut.png"))
    self.label.setScaledContents(True)
    self.label_6 = QLabel(self.scrollAreaWidgetContents_2)
    self.label_6.setObjectName(u"label_6")
    self.label_6.setGeometry(QRect(-40, 420, 191, 31))
    self.label_6.setStyleSheet(u"background-color: transparent;\n"
"border-right: 0px;")
    self.label_6.setPixmap(QPixmap(u":/icons/Icons/neptune.png"))
    self.label_6.setScaledContents(True)
    self.label_7 = QLabel(self.scrollAreaWidgetContents_2)
    self.label_7.setObjectName(u"label_7")
    self.label_7.setGeometry(QRect(10, 310, 41, 41))
    self.label_7.setStyleSheet(u"background-color: transparent;\n"
"border-right: 0px;")
    self.label_7.setPixmap(QPixmap(u":/icons/Icons/planet.png"))
    self.label_7.setScaledContents(True)
    self.label_8 = QLabel(self.scrollAreaWidgetContents_2)
    self.label_8.setObjectName(u"label_8")
    self.label_8.setGeometry(QRect(60, 330, 41, 41))
    self.label_8.setStyleSheet(u"background-color: transparent;\n"
"border-right: 0px;")
    self.label_8.setPixmap(QPixmap(u":/icons/Icons/orbit.png"))
    self.label_8.setScaledContents(True)
    self.label_10 = QLabel(self.scrollAreaWidgetContents_2)
    self.label_10.setObjectName(u"label_10")
    self.label_10.setGeometry(QRect(50, 280, 41, 41))
    self.label_10.setStyleSheet(u"background-color: transparent;\n"
"border-right: 0px;")
    self.label_10.setPixmap(QPixmap(u":/icons/Icons/planet (1).png"))
    self.label_10.setScaledContents(True)
    self.label_11 = QLabel(self.scrollAreaWidgetContents_2)
    self.label_11.setObjectName(u"label_11")
    self.label_11.setGeometry(QRect(10, 250, 41, 41))
    self.label_11.setStyleSheet(u"background-color: transparent;\n"
"border-right: 0px;")
    self.label_11.setPixmap(QPixmap(u":/icons/Icons/galaxy.png"))
    self.label_11.setScaledContents(True)
    self.label_12 = QLabel(self.scrollAreaWidgetContents_2)
    self.label_12.setObjectName(u"label_12")
    self.label_12.setGeometry(QRect(40, 240, 41, 41))
    self.label_12.setStyleSheet(u"background-color: transparent;\n"

```

```

"border-right: 0px;")
    self.label_12.setPixmap(QPixmap(u":/icons/Icons/satellite.png"))
    self.label_12.setScaledContents(True)
    self.menu_area.setWidget(self.scrollAreaWidgetContents_2)
    self.label_8.raise_()
    self.menu_close_button.raise_()
    self.online_clients_open_button.raise_()
    self.label_6.raise_()
    self.label.raise_()
    self.label_7.raise_()
    self.label_10.raise_()
    self.label_11.raise_()
    self.label_12.raise_()
    self.menu_open_button = QPushButton(self.centralwidget)
    self.menu_open_button.setObjectName(u"menu_open_button")
    self.menu_open_button.setGeometry(QRect(10, 10, 91, 41))
    self.menu_open_button.setFont(font)
    self.menu_open_button.setCursor(QCursor(Qt.ArrowCursor))
    self.menu_open_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"color: #FFFFFF;\n"
"background-color: #7B68EE;")
    icon5 = QIcon()
    icon5.addFile(u":/icons/Icons/earth.png", QSize(), QIcon.Normal, QIcon.Off)
    self.menu_open_button.setIcon(icon5)
    self.menu_open_button.setIconSize(QSize(40, 40))
    self.online_clients_area = QScrollArea(self.centralwidget)
    self.online_clients_area.setObjectName(u"online_clients_area")
    self.online_clients_area.setGeometry(QRect(110, 60, 145, 261))
    self.online_clients_area.setStyleSheet(u"border-radius: 8px;\n"
"border-right: 1px solid #e0e4e7;\n"
"border-top: 1px solid #e0e4e7;\n"
"border-bottom: 1px solid #e0e4e7;\n"
"color: #FFFFFF")
    self.online_clients_area.setWidgetResizable(True)
    self.scrollAreaWidgetContents_3 = QWidget()
    self.scrollAreaWidgetContents_3.setObjectName(u"scrollAreaWidgetContents_3")
    self.scrollAreaWidgetContents_3.setGeometry(QRect(0, 0, 143, 259))
    self.online_clients_label = QLabel(self.scrollAreaWidgetContents_3)
    self.online_clients_label.setObjectName(u"online_clients_label")
    self.online_clients_label.setGeometry(QRect(10, 10, 81, 31))
    font1 = QFont()
    font1.setFamilies([u"Roboto Mono"])
    font1.setPointSize(14)
    font1.setBold(False)
    font1.setUnderline(False)
    self.online_clients_label.setFont(font1)
    self.online_clients_label.setStyleSheet(u"border: 1px solid #e0e4e7;")
    self.online_clients_label.setTextFormat(Qt.PlainText)
    self.online_clients_label.setScaledContents(False)
    self.online_clients_close_button = QPushButton(self.scrollAreaWidgetContents_3)
    self.online_clients_close_button.setObjectName(u"online_clients_close_button")

```

```

        self.online_clients_close_button.setGeometry(QRect(100, 10, 31, 31))
        self.online_clients_close_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"color: #FFFFFF;\n"
"background-color: #B22222;")
        self.online_clients_list = QListWidget(self.scrollAreaWidgetContents_3)
        self.online_clients_list.setObjectName(u"online_clients_list")
        self.online_clients_list.setGeometry(QRect(10, 51, 121, 191))
        self.online_clients_list.setFont(font)
        self.online_clients_list.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding: 5px 5px;\n"
"color: #FFFFFF;\n"
"background-color: #6A5ACD")
        self.label_14 = QLabel(self.scrollAreaWidgetContents_3)
        self.label_14.setObjectName(u"label_14")
        self.label_14.setGeometry(QRect(0, 210, 51, 51))
        self.label_14.setStyleSheet(u"background-color: transparent;\n"
"border: 0px;")
        self.label_14.setPixmap(QPixmap(u":/icons/Icons/satelite.png"))
        self.label_14.setScaledContents(True)
        self.label_15 = QLabel(self.scrollAreaWidgetContents_3)
        self.label_15.setObjectName(u"label_15")
        self.label_15.setGeometry(QRect(90, -20, 71, 71))
        self.label_15.setStyleSheet(u"background-color: transparent;\n"
"border: 0px;")
        self.label_15.setPixmap(QPixmap(u":/icons/Icons/black-hole (1).png"))
        self.label_15.setScaledContents(True)
        self.online_clients_area.setWidget(self.scrollAreaWidgetContents_3)
        self.label_15.raise_()
        self.online_clients_label.raise_()
        self.online_clients_close_button.raise_()
        self.online_clients_list.raise_()
        self.label_14.raise_()
        self.file_send_button = QPushButton(self.centralwidget)
        self.file_send_button.setObjectName(u"file_send_button")
        self.file_send_button.setGeometry(QRect(410, 380, 61, 41))
        self.file_send_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding: 5px 15px;\n"
"color: #FFFFFF;\n"
"background-color: #7B68EE;")
        icon6 = QIcon()
        icon6.addFile(u":/icons/Icons/folder.png", QSize(), QIcon.Normal, QIcon.Off)
        self.file_send_button.setIcon(icon6)
        self.file_send_button.setIconSize(QSize(30, 30))
        self.download_file_area = QScrollArea(self.centralwidget)
        self.download_file_area.setObjectName(u"download_file_area")
        self.download_file_area.setGeometry(QRect(140, 110, 295, 210))
        self.download_file_area.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"color: #FFFFFF")

```

```

self.download_file_area.setWidgetResizable(True)
self.scrollAreaWidgetContents_4 = QWidget()
self.scrollAreaWidgetContents_4.setObjectName(u"scrollAreaWidgetContents_4")
self.scrollAreaWidgetContents_4.setGeometry(QRect(0, 0, 293, 208))
self.file_info = QLabel(self.scrollAreaWidgetContents_4)
self.file_info.setObjectName(u"file_info")
self.file_info.setGeometry(QRect(10, 60, 270, 90))
self.file_info.setFont(font)
self.file_info.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"padding: 5px 5px;\n"
"color: #FFFFFF;\n"
"background-color: #6A5ACD;")
self.download_button = QPushButton(self.scrollAreaWidgetContents_4)
self.download_button.setObjectName(u"download_button")
self.download_button.setGeometry(QRect(10, 160, 71, 31))
self.download_button.setFont(font)
self.download_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"color: #FFFFFF;\n"
"background-color: #2E8B57;")
self.cancel_download_button = QPushButton(self.scrollAreaWidgetContents_4)
self.cancel_download_button.setObjectName(u"cancel_download_button")
self.cancel_download_button.setGeometry(QRect(210, 160, 71, 31))
self.cancel_download_button.setFont(font)
self.cancel_download_button.setStyleSheet(u"border-radius: 8px;\n"
"border: 1px solid #e0e4e7;\n"
"color: #FFFFFF;\n"
"background-color: #B22222;")
self.download_title = QLabel(self.scrollAreaWidgetContents_4)
self.download_title.setObjectName(u"download_title")
self.download_title.setGeometry(QRect(20, 10, 251, 41))
font2 = QFont()
font2.setFamilies([u"Roboto Mono"])
font2.setPointSize(22)
self.download_title.setFont(font2)
self.download_title.setStyleSheet(u"text-align: center;")
self.rocket_icon = QLabel(self.scrollAreaWidgetContents_4)
self.rocket_icon.setObjectName(u"rocket_icon")
self.rocket_icon.setGeometry(QRect(0, 50, 41, 41))
self.rocket_icon.setStyleSheet(u"background-color: transparent;\n"
"border: 0px;")
self.rocket_icon.setPixmap(QPixmap(u":/icons/Icons/rocket.png"))
self.rocket_icon.setScaledContents(True)
self.ufo_icon = QLabel(self.scrollAreaWidgetContents_4)
self.ufo_icon.setObjectName(u"ufo_icon")
self.ufo_icon.setGeometry(QRect(220, -10, 41, 41))
self.ufo_icon.setStyleSheet(u"background-color: transparent;\n"
"border: 0px;")
self.ufo_icon.setPixmap(QPixmap(u":/icons/Icons/ufo.png"))
self.ufo_icon.setScaledContents(True)
self.download_file_area.setWidget(self.scrollAreaWidgetContents_4)

```

```

self.label_3 = QLabel(self.centralwidget)
self.label_3.setObjectName(u"label_3")
self.label_3.setGeometry(QRect(0, 0, 550, 440))
self.label_3.setPixmap(QPixmap(u":images/Icons/5471985.jpg"))
self.label_3.setScaledContents(True)
self.label_2 = QLabel(self.centralwidget)
self.label_2.setObjectName(u"label_2")
self.label_2.setGeometry(QRect(500, 30, 41, 41))
self.label_2.setStyleSheet(u"background-color: transparent;")
self.label_2.setPixmap(QPixmap(u":icons/Icons/meteor.png"))
self.label_2.setScaledContents(True)
self.label_4 = QLabel(self.centralwidget)
self.label_4.setObjectName(u"label_4")
self.label_4.setGeometry(QRect(310, 400, 31, 41))
self.label_4.setStyleSheet(u"background-color: transparent;")
self.label_4.setPixmap(QPixmap(u":icons/Icons/black-hole.png"))
self.label_4.setScaledContents(True)
self.label_5 = QLabel(self.centralwidget)
self.label_5.setObjectName(u"label_5")
self.label_5.setGeometry(QRect(400, -30, 71, 71))
self.label_5.setStyleSheet(u"background-color: transparent;")
self.label_5.setPixmap(QPixmap(u":icons/Icons/solar-system.png"))
self.label_5.setScaledContents(True)
self.label_9 = QLabel(self.centralwidget)
self.label_9.setObjectName(u"label_9")
self.label_9.setGeometry(QRect(0, 330, 41, 41))
self.label_9.setStyleSheet(u"background-color: transparent;\n"
"border: 0px;")
self.label_9.setPixmap(QPixmap(u":icons/Icons/astronaut (1).png"))
self.label_9.setScaledContents(True)
MainWindow.setCentralWidget(self.centralwidget)
self.label_3.raise_()
self.chat_input.raise_()
self.chat_viewer.raise_()
self.emoji_area.raise_()
self.send_button.raise_()
self.emoji_open_button.raise_()
self.menu_open_button.raise_()
self.file_send_button.raise_()
self.online_clients_area.raise_()
self.download_file_area.raise_()
self.label_2.raise_()
self.label_4.raise_()
self.label_5.raise_()
self.label_9.raise_()
self.menu_area.raise_()

self.retranslateUi(MainWindow)

QMetaObject.connectSlotsByName(MainWindow)
# setupUi

```



```

def retranslateUi(self, MainWindow):
    MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow", u"SimpleChat", None))
    self.chat_input.setHtml(QCoreApplication.translate("MainWindow", u"<!DOCTYPE HTML PUBLIC
\"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><meta charset=\"utf-8\" /><style
type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"hr { height: 1px; border-width: 0; }\n"
"li.unchecked::marker { content: \"\\2610\"; }\n"
"li.checked::marker { content: \"\\2612\"; }\n"
"</style></head><body style=\" font-family:'Roboto Mono'; font-size:10pt; font-weight:400;
font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px; font-family:'Segoe UI'; font-size:9pt;\"><br
/></p></body></html>", None))
    self.chat_input.setPlaceholderText(QCoreApplication.translate("MainWindow", u"Enter your
Message", None))
    self.send_button.setText("")
    self.emoji_1.setText(QCoreApplication.translate("MainWindow",
u":grinning_face_with_big_eyes:", None))
    self.emoji_close_button.setText(QCoreApplication.translate("MainWindow", u"X", None))
    self.emoji_2.setText(QCoreApplication.translate("MainWindow",
u":grinning_face_with_smiling_eyes:", None))
    self.emoji_3.setText(QCoreApplication.translate("MainWindow", u":grimacing_face:", None))
    self.emoji_4.setText(QCoreApplication.translate("MainWindow", u":grinning_face_with_sweat:",
None))
    self.emoji_5.setText(QCoreApplication.translate("MainWindow", u":grinning_squinting_face:",
None))
    self.emoji_6.setText(QCoreApplication.translate("MainWindow", u":angry_face:", None))
    self.emoji_7.setText(QCoreApplication.translate("MainWindow", u":astonished_face:", None))
    self.emoji_8.setText(QCoreApplication.translate("MainWindow",
u":beaming_face_with_smiling_eyes:", None))
    self.emoji_9.setText(QCoreApplication.translate("MainWindow", u":clown_face:", None))
    self.emoji_14.setText(QCoreApplication.translate("MainWindow", u":disappointed_face:", None))
    self.emoji_10.setText(QCoreApplication.translate("MainWindow", u":cold_face:", None))
    self.emoji_17.setText(QCoreApplication.translate("MainWindow", u":expressionless_face:",
None))
    self.emoji_13.setText(QCoreApplication.translate("MainWindow", u":crying_face:", None))
    self.emoji_18.setText(QCoreApplication.translate("MainWindow", u":face_blowing_a_kiss:",
None))
    self.emoji_11.setText(QCoreApplication.translate("MainWindow", u":confused_face:", None))
    self.emoji_15.setText(QCoreApplication.translate("MainWindow", u":drooling_face:", None))
    self.emoji_12.setText(QCoreApplication.translate("MainWindow", u":cowboy_hat_face:", None))
    self.emoji_16.setText(QCoreApplication.translate("MainWindow", u":enraged_face:", None))
    self.emoji_23.setText(QCoreApplication.translate("MainWindow", u":face_screaming_in_fear:",
None))
    self.emoji_19.setText(QCoreApplication.translate("MainWindow", u":face_exhaling:", None))
    self.emoji_26.setText(QCoreApplication.translate("MainWindow", u":face_with_diagonal_mouth:",
None))
    self.emoji_22.setText(QCoreApplication.translate("MainWindow", u":face_savoring_food:", None))
    self.emoji_27.setText(QCoreApplication.translate("MainWindow", u":face_with_head-bandage:",
None))

```

```

self.emoji_20.setText(QCoreApplication.translate("MainWindow", u":face_holding_back_tears:",
None))
self.emoji_24.setText(QCoreApplication.translate("MainWindow", u":face_vomiting:", None))
self.emoji_21.setText(QCoreApplication.translate("MainWindow", u":face_in_clouds:", None))
self.emoji_25.setText(QCoreApplication.translate("MainWindow",
u":face_with_crossed-out_eyes:", None))
self.emoji_32.setText(QCoreApplication.translate("MainWindow", u":face_with_tears_of_joy:",
None))
self.emoji_28.setText(QCoreApplication.translate("MainWindow", u":face_with_medical_mask:",
None))
self.emoji_35.setText(QCoreApplication.translate("MainWindow", u":flushed_face:", None))
self.emoji_31.setText(QCoreApplication.translate("MainWindow",
u":face_with_symbols_on_mouth:", None))
self.emoji_36.setText(QCoreApplication.translate("MainWindow",
u":frowning_face_with_open_mouth:", None))
self.emoji_29.setText(QCoreApplication.translate("MainWindow", u":face_with_raised_eyebrow:",
None))
self.emoji_33.setText(QCoreApplication.translate("MainWindow", u":face_with_tongue:", None))
self.emoji_30.setText(QCoreApplication.translate("MainWindow", u":face_with_rolling_eyes:",
None))
self.emoji_34.setText(QCoreApplication.translate("MainWindow", u":face_without_mouth:", None))
self.emoji_41.setText(QCoreApplication.translate("MainWindow", u":kiss_mark:", None))
self.emoji_37.setText(QCoreApplication.translate("MainWindow", u":frog:", None))
self.emoji_44.setText(QCoreApplication.translate("MainWindow",
u":raised_fist_dark_skin_tone:", None))
self.emoji_40.setText(QCoreApplication.translate("MainWindow", u":index_pointing_up:", None))
self.emoji_45.setText(QCoreApplication.translate("MainWindow", u":see-no-evil_monkey:", None))
self.emoji_38.setText(QCoreApplication.translate("MainWindow",
u":grinning_cat_with_smiling_eyes:", None))
self.emoji_42.setText(QCoreApplication.translate("MainWindow", u":kiwi_fruit:", None))
self.emoji_39.setText(QCoreApplication.translate("MainWindow", u":handshake:", None))
self.emoji_43.setText(QCoreApplication.translate("MainWindow", u":lobster:", None))
self.base_icon.setText("")
self.scope_icon.setText("")
self.emoji_open_button.setText("")
self.menu_close_button.setText(QCoreApplication.translate("MainWindow", u"Menu", None))
self.online_clients_open_button.setText(QCoreApplication.translate("MainWindow", u"Users",
None))
self.label.setText("")
self.label_6.setText("")
self.label_7.setText("")
self.label_8.setText("")
self.label_10.setText("")
self.label_11.setText("")
self.label_12.setText("")
self.menu_open_button.setText(QCoreApplication.translate("MainWindow", u"Menu", None))
self.online_clients_label.setText(QCoreApplication.translate("MainWindow", u"Online", None))
self.online_clients_close_button.setText(QCoreApplication.translate("MainWindow", u"X", None))
self.label_14.setText("")
self.label_15.setText("")
self.file_send_button.setText("")
self.file_info.setText(QCoreApplication.translate("MainWindow", u"File Info", None))

```



```

self.download_button.setText(QCoreApplication.translate("MainWindow", u"Download", None))
self.cancel_download_button.setText(QCoreApplication.translate("MainWindow", u"Cancel", None))
self.download_title.setText(QCoreApplication.translate("MainWindow", u"Download File", None))
self.rocket_icon.setText("")
self.ufo_icon.setText("")
self.label_3.setText("")
self.label_2.setText("")
self.label_4.setText("")
self.label_5.setText("")
self.label_9.setText("")
# retranslateUi

```

Додаток 6. Програмний модуль CLogin.dll Login.h

```

#pragma once
#include <fstream>
#include <cstdlib>
#include <msclr\marshal_cppstd.h>

extern "C" __declspec(dllexport) void login();

namespace CLogin {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for Login
    /// </summary>
    public ref class Login : public System::Windows::Forms::Form
    {
    public:
        Login(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~Login()
        {

```

```

        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::Label^ top_label;
private: System::Windows::Forms::Button^ login_button;
private: System::Windows::Forms::Button^ cancel_button;
protected:

private: System::Windows::Forms::TextBox^ input;
private: System::Windows::Forms::Label^ welcome_label;
private: System::Windows::Forms::Label^ ErrorLabel;
protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        this->top_label = (gcnew System::Windows::Forms::Label());
        this->login_button = (gcnew System::Windows::Forms::Button());
        this->cancel_button = (gcnew System::Windows::Forms::Button());
        this->input = (gcnew System::Windows::Forms::TextBox());
        this->welcome_label = (gcnew System::Windows::Forms::Label());
        this->ErrorLabel = (gcnew System::Windows::Forms::Label());
        this->SuspendLayout();
        //
        // top_label
        //
        this->top_label->AutoSize = true;
        this->top_label->BackColor = System::Drawing::Color::Transparent;
        this->top_label->Font = (gcnew System::Drawing::Font(L"Roboto Mono", 13,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->top_label->ForeColor = System::Drawing::Color::White;
        this->top_label->Location = System::Drawing::Point(10, 152);
        this->top_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
        this->top_label->Name = L"top_label";
        this->top_label->Size = System::Drawing::Size(415, 34);
        this->top_label->TabIndex = 0;
        this->top_label->Text = L"How should you be called?\r\n";
        //
        // login_button

```

```

//
this->login_button->BackColor = System::Drawing::Color::MediumSlateBlue;
this->login_button->Font = (gcnew System::Drawing::Font(L"Roboto Mono", 10,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
    static_cast<System::Byte>(204)));
this->login_button->ForeColor = System::Drawing::Color::White;
this->login_button->Location = System::Drawing::Point(16, 296);
this->login_button->Margin = System::Windows::Forms::Padding(4);
this->login_button->Name = L"login_button";
this->login_button->Size = System::Drawing::Size(124, 57);
this->login_button->TabIndex = 1;
this->login_button->Text = L"Login";
this->login_button->UseVisualStyleBackColor = false;
this->login_button->Click += gcnew System::EventHandler(this,
&Login::login_button_Click);
//
// cancel_button
//
this->cancel_button->BackColor = System::Drawing::Color::MediumSlateBlue;
this->cancel_button->Font = (gcnew System::Drawing::Font(L"Roboto Mono", 10,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
    static_cast<System::Byte>(204)));
this->cancel_button->ForeColor = System::Drawing::Color::White;
this->cancel_button->Location = System::Drawing::Point(291, 296);
this->cancel_button->Margin = System::Windows::Forms::Padding(4);
this->cancel_button->Name = L"cancel_button";
this->cancel_button->Size = System::Drawing::Size(124, 57);
this->cancel_button->TabIndex = 2;
this->cancel_button->Text = L"Cancel";
this->cancel_button->UseVisualStyleBackColor = false;
this->cancel_button->Click += gcnew System::EventHandler(this,
&Login::cancel_button_Click);
//
// input
//
this->input->BackColor = System::Drawing::Color::DarkSlateBlue;
this->input->CausesValidation = false;
this->input->Font = (gcnew System::Drawing::Font(L"Roboto Mono", 13,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
    static_cast<System::Byte>(204)));
this->input->ForeColor = System::Drawing::Color::White;
this->input->Location = System::Drawing::Point(16, 222);
this->input->Margin = System::Windows::Forms::Padding(4);
this->input->MaxLength = 13;
this->input->Name = L"input";
this->input->Size = System::Drawing::Size(399, 42);
this->input->TabIndex = 3;
this->input->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;
//
// welcome_label
//
this->welcome_label->AutoSize = true;

```

```

        this->welcome_label->BackColor = System::Drawing::Color::Transparent;
        this->welcome_label->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
        this->welcome_label->Font = (gcnew System::Drawing::Font(L"Roboto Mono", 16,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->welcome_label->ForeColor = System::Drawing::Color::White;
        this->welcome_label->Location = System::Drawing::Point(134, 43);
        this->welcome_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
        this->welcome_label->Name = L"welcome_label";
        this->welcome_label->Size = System::Drawing::Size(152, 43);
        this->welcome_label->TabIndex = 4;
        this->welcome_label->Text = L"Welcome";
        //
        // ErrorLabel
        //
        this->ErrorLabel->AutoSize = true;
        this->ErrorLabel->Font = (gcnew System::Drawing::Font(L"Roboto Mono", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->ErrorLabel->ForeColor = System::Drawing::Color::White;
        this->ErrorLabel->Location = System::Drawing::Point(146, 307);
        this->ErrorLabel->Name = L"ErrorLabel";
        this->ErrorLabel->Size = System::Drawing::Size(0, 32);
        this->ErrorLabel->TabIndex = 5;
        //
        // Login
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(12, 26);
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
        this->BackColor = System::Drawing::Color::DarkSlateBlue;
        this->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;
        this->ClientSize = System::Drawing::Size(432, 367);
        this->Controls->Add(this->ErrorLabel);
        this->Controls->Add(this->welcome_label);
        this->Controls->Add(this->input);
        this->Controls->Add(this->cancel_button);
        this->Controls->Add(this->login_button);
        this->Controls->Add(this->top_label);
        this->Font = (gcnew System::Drawing::Font(L"Roboto Mono", 10,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::None;
        this->Margin = System::Windows::Forms::Padding(4);
        this->Name = L"Login";
        this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"Login";
        this->ResumeLayout(false);
        this->PerformLayout();

    }

#pragma endregion
    private: System::Void cancel_button_Click(System::Object^ sender, System::EventArgs^ e) {

```

```

        this->Close();
    }
private: System::Void login_button_Click(System::Object^ sender, System::EventArgs^ e) {
    if (this->input->Text == "")
    {
        this->ErrorLabel->Text = "Try Again";
        return;
    }

    std::string username = msclr::interop::marshal_as<std::string>(this->input->Text);

    std::string file_path(std::getenv("USERPROFILE"));
    std::string file_name = "\\login_data.txt";
    file_path += file_name;

    std::ofstream out(file_path);
    out << username;
    out.close();
    this->Close();
}
};
}

```

Login.cpp

```

#include "Login.h"

using namespace System;
using namespace System::Windows::Forms;

extern "C" __declspec(dllexport) void login()
{
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    CLogin::Login loginform;

    loginform.ShowDialog();
}

```