

Project: chatterbox

C# WPF desktop chat application



**April 5th, 2025  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

CPRO 1211 D: Programming .NET, Winter 2025

Instructor: Dr. Mushtaq Raza

Authored by:

Aaryan Garg, 000375834  
Yaksh Sharma, 000375869

Harmanpreet Singh,

Table of Contents

[1. Introduction 2](#_Toc195133815)

[2. Project Description 3](#_Toc195133816)

[3. User Interface (UI) Design and implementation 5](#_Toc195133817)

[4. Testing 14](#_Toc195133818)

[4.1 Testing description 14](#_Toc195133819)

[4.2 Results 14](#_Toc195133820)

[5. Conclusion 27](#_Toc195133821)

[6. References 28](#_Toc195133822)

# Introduction

This project comprises a desktop application developed in Visual Studio Code 2022 using C# WPF. This project implements a basic instant messaging system. The problem addressed by our project is enabling real time communication with simple and user-friendly interface.

ChatterBox: The application provides a platform for user to register an account, log in, manage a list of friends and exchange messages in real time. The app allows new users to register securely and returning users to log in with valid credentials. It also enables users to maintain a personal friends list, including adding new friends and removing existing ones. It provides a chat interface for sending and receiving messages with friends, updating the conversation in real time. It also focuses on prompt feedback for each action like login, register and send messages by giving a success or error message.

This application focuses on connecting SQL Server Database, using stored procedures to securely manage data operations like user authentication, friend relationships, and message retrieval. Overall, we see each aspect in this project starting from features and designs to documenting codes and testing outcomes.

# Project Description

This application is designed as feature-rich application with several integrated functionalities:

* Registration: It allows new users to create an account with a username and password, which are stored in SQL server database via a stored procedure.
* Chat system: Once logged in, users can send and receive messages from any existing chat rooms.
* Join rooms: They can join existing rooms, or they can make new chat rooms, and more people can join the rooms upon logging in from their account.
* Chat in the rooms: The messages are stores in database, and all the past messages are restored for every chat room. This ensures chat history which is loaded each time chat is opened. Sending a message will invoke another command to add the new message.
* Log in: Existing users can log in through the main landing page by entering the username and password with what they made the account. This is validated by the stored procedure in database. Upon successful login, success feedback is given, and the user is taken to the chat room page.
* Friends Management: Users have ability to manage their friends list. This includes adding a friend by username, which then displays the friend’s profile which is taken from the back end using the friends userID and inserts a record into the friendship table. Similarly, removing a friend, deletes a record from friendship table.
* Technologies used: This application is built using Windows presentation foundation (WPF) for the front-end. WPF is chosen for its rich control set and ability to create modern desktop UI with XAML. The database is connected from visual studio using Microsoft.data.sql . The back end is supported by SQL Server Management studio. In the database, all the data is stored including the friends, chats and relationship which is indicated from ***figure 1***.
* Multi page structure: The application uses multiple pages and windows for every feature:
* Main landing page: For login.
* Register page: To register for new login.
* Friends page: to display list of friends.
* Friends Search page: To search for people using username.
* Friends Info page: To show the friends’ profile.
* Main window page: Main page with all the chat rooms.

A computer screen shot of a computer

AI-generated content may be incorrect.

Figure 1: Relational Database design used for chatterbox

# User Interface (UI) Design and implementation

The chatterbox application employs a straightforward and clean UI design for each of its pages, focusing on simple visually appealing clean and clear, easy to use and consistency across the application. All pages share a common design theme and layout, to make it easier for new users to navigate.

***Figure 2*** displays the main login page consisting of two textboxes used for recording the input for username and password. They also have appropriate labels for them with proper designing. The password field is a password box which hides the input for security. This page also consists of log in button which triggers the button click event and the authentication process. Upon successful validation it follows with a message box for success message and follows up by directing to the main window page. If the user is not an existing user, then we have added an text block which directs the user to register page to make a new account.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 2: Login Page GUI

***Figure 3*** Displays the register page which consists of four textboxes to record the information of new user that is its username which is supposed to be unique for every user and if existing username is entered, the application displays a message box indicating of the error message that the username already exists. This page also consists of 2 password columns to enter and verify the password by re entering it. The password conditions consist of basic strong password terms consisting of at least 1 uppercase, 1 lowercase and 1 number. Upon not following the password conditions, the application will catch the error in validation and will show the user another message box displaying the error prompting user to enter at least 1 uppercase, 1 lowercase and 1 number. The last column i.e.- description is optional, its purpose is to write something about yourself which will be visible to other users upon visiting your profile. The page consists of 2 buttons, register and login. Register button will fetch the data from the textboxes and send them for validation. If the validation fails, the application displays the appropriate message box, prompting the user to fix the issues. If the validation succeeds, it triggers the insert stamen in sql to add a new entry in user table using the data fetched. Upon successful run of the sql command, the application will display a message box indicating the successful registration and will automatically take the user to the main login page. The purpose of log in button is to direct the user to the login page in case the user changes its mind or any other variables.

A screenshot of a computer login

AI-generated content may be incorrect.

Figure 3: Register page GUI

Upon successful login, the main window page is opened as shown in ***Figure 4***. This page consists of the button Join new channel on the left top corner, which will display the existing channels from the channel table from database for the user to enter. Below the button is a list box which displays the channels the user is a part of which is derived from the channel table with foreign key as the userID of the user that is logged in. Upon clicking on the channel name, the chat of that channel is loaded on the scrollable listbox on the right. The loaded chat consists of messages in the simple format consisting of the message, user who sent that message and time that message was delivered [1]. The chat of every channel consists of a textbox and a button in the bottom right corner which record the message user wants to send in the chat, and the button triggers the insert statement for chat table which in turn refreshes the chat by displaying the updated data from the chat table consisting of the message that was just delivered. The left bottom corner consists of 2 buttons with their own purpose. On the left, log out button directs the user back to the log in page, which means it will log the user out of the chatterBox. On the right is My friends button, this will direct the user to the friends page consisting of the other users the current user is friends with. This page follows the same color theme and simple clear format for easy navigation.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4: Main Login Page GUI

Upon clicking My Friends button, the friends page is displayed as per ***Figure 5***. The design formal is very similar to the main window page with the friends list on the left side and related information displayed on the right. The colour contrast is a bit different from the pages we discussed so far, this being more on the darker side. Back to the functionality of the components, it consists of 2 buttons on the left corner. The button on the left named back directs the user to the main landing page given that the same user remained logged in. The button on the right named Add Friend directs the user to friends search page for looking for new user. Below the button consists of a listbox titled My Friends consists of the users connected with the current user from friends table. Upon clicking the name of any user in the friends list, the stackpanel on the right side will display 3 textblocks. One indicated the username of the user in larger font than others, rest shows the description of the user and the date the user registered on this application. It also consists of a remove friend named button which triggers the remove statement from friends table for sql and removes the name of the user from the list.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5: Friends page GUI

When Add friend button is clicked, the friend search page is opened as per the ***Figure 6***. The colour theme and layout remain true to the overall application theme. It Consists of a textbox for the user to enter the username of the user it is trying to search. The search button next to the textbox fetch the data from the textbox and trigger a select statement from the users table and displays all the users in the list box below which contains the words anywhere in its username that is entered by the current user. The button next to search button names back, directs the user to the friend’s page. If any user displayed in the listbox below upon searching, the user displayed are clickable and directs the user to friends info page for displaying the profile of the user whose name was clicked. This page also consists of a request button on the right top corner which upon clicked, will direct the user to request page.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6: Friends search page GUI

After clicking on the user from the list of the people searched by the current user, the friend info page opens as shown in ***Figure 7***. The design of this page is made in grid layout and is different from others. The blur button theme remains the same, but the background colour was changed to black to make it more visually appealing as per the dark mode trend nowadays. This page uses textblock to display the information of the user searched. The textblock in stackpanel of this page shows the username, description and the date the user registered their account using select statements from the user table. It consists of 2 buttons at the bottom, The back button as usual directs the user to friend search page. Whereas the 2nd button, send a request to the user for adding the current user as their friend. This triggers an insert statement in the friends table but keep the accepted column at 0 (0=false). If that user accepts the request, the column changes to 1 whereas if rejected, the friend row is deleted from the table itself.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 7: Friends Info page GUI

From the friend search page, if the user clicked the request button, the request page as shown in ***Figure 8*** is opened. This page displays the requests of other users in a list box. It consists of a textblock displaying the username of the person who sent the request. Each request will consist of 2 buttons, accept and decline. The accept button will change the accept column of this field in friend table to 1 which indicates that the request is accepted and will start displaying in their friend list. In case of the decline button, the click event will trigger the remove statement to delete the row in the friend table.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 8: Request page GUI

When the button join channel is clicked from main landing page the channel join page opens as shown in ***Figure 9.*** This page consists of a text box at the top which fetches the name of the channel from the user. This page also consists of a button next to the text box named as search. Upon clicking the button, the search button event handler triggers the select statement from channels table. This statement fetches all the channels which consists of the words entered by the user in the search bar. The loaded channels are then displayed on the listbox below the search bar. Every channel displayed contains a Join button which upon clicked, will trigger the insert statement to insert the user in that channel and will start displaying this channel on their joined channel list.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 9: Join channel page GUI

As shown in the ***Figure 10***, the class SQL Connect is used to connect SQL database with the application. The method Getconn creates new Sqlconnection instance and opens the connection which in turns makes all the procedures and the queries in the database works. The method closeConn closely the connection with the sql with. close function. The try catch function in getconn method makes sure the application doesn’t crash on error or cause issues with the device and shows the error message and closes the connection with the database in case of error. [2]

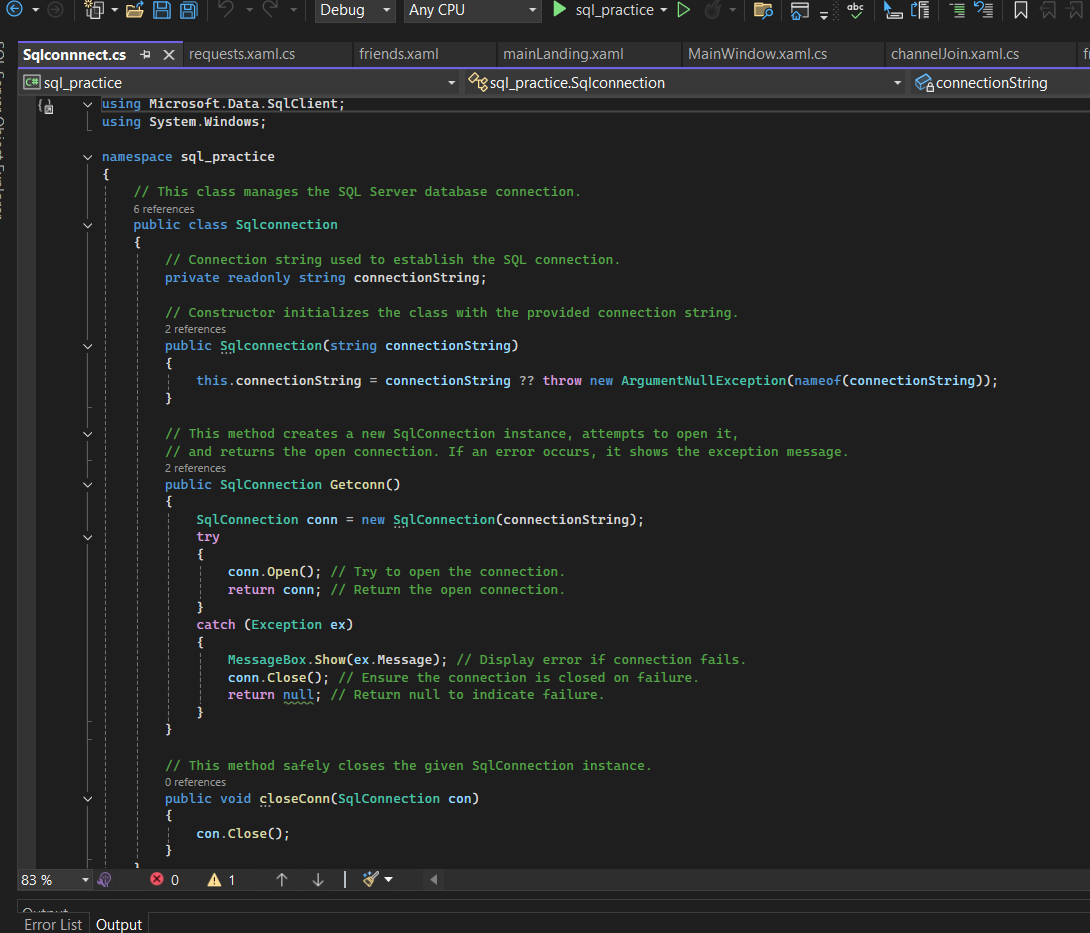


Figure 10: SqlConnection class

# Testing

## Testing description

To verify if the features applied in this application work as intended, the tests are done below. Each test case is outlined with the test description, outcome it delivered and if the result is passed, it shows the outcome is the same as the behaviour expected.

## Results

The test cases are conducted as per the following table along with the expected and actual results. The test was conducted manually and are confirmed by the figures displayed below the case chart:

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Description | Expected Behavior/ Actual Behaviour | Result |
| Test case 1:  Login correctly | Input username and password of already existing user. | Message prompt given to user about successful login.  ***Figure 11*** | Passed |
| Test case 2:  Login with wrong info | Input username and password that doesn’t exist in the database. | Error message displayed to user prompting it to enter correct information as the user doesn’t exist. ***Figure12*** | Passed |
| Test case 3:  Register with correct information | Input all the information in the register page as per the requirements. | Message displayed about successful registration and directing user on login page. ***Figure 13*** | Passed |
| Test case 4:  Register with existing username | Input all correct information on register page except the username. | Error message displayed informing user that username already exists. ***Figure 14*** | Passed |
| Test case 5:  Register with wrong password format. | Input wrong format of password, not following the rule of one number and one uppercase letter at least. | Password validation failed; Error message displayed informing user that password doesn’t match the format. ***Figure 15*** | Passed |
| Test case 6:  Sending text in a chat channel | Sending a message in one of the channels and the chat room is supposed to refresh and display the message along with the name and the time it was sent. | The channel displays the message at the very bottom along with the users username and the current time. ***Figure 16, 17*** | Passed |
| Test case 7:  Joining a channel | Channel named gaming is first searched in the join channel page and after clicking join it will insert a foreign key entry in database and the channel will display on the user’s channel list. | The channel joined successfully message dialog box appear and intended and the channel list is refreshed and the channel name is displayed. ***Figurem18,19,20*** | Passed |
| Test case 8:  Removing a friend | In the friend’s page, the remove friend button is clicked, and friend column is removed from that page along with the entry of that user in friend’s table. | The dialog box appear as intended after removing the friend, and the friend list doesn’t include that friends column anymore. ***Figure 21,22, 23*** | Passed |
| Test Case 9:  Adding a friend | In the friends search page, the friend’s username is searched and all the relevant usernames from the users table are displayed. Upon selecting the user, we want to befriend, it will show their information on different page and the send request will create an entry in friend’s column which upon accepted will make it visible in the friend’s page on both the users. | The name of the friend is shown as intended in the search results, and the request sent successfully dialog box appear after clicking the send request button. ***Figure 24, 25*** | Passed |
| Test case 10:  Sending another request to a user who we already received the friend request. | Upon sending another request to the user who the current user has already sent the request before; the validation process detects the entry in the friends table and notify the current user of the previous made request regardless of if its accepted or not. | A different dialog box saying “Friend request already sent” appears. ***Figure 26*** | Passed |
| Test case 11:  Accepting a request | On the request page, the requests show up with two buttons, The accept button changes the status of accept column in friends table and both users can see each other on their own friends’ page. | The request accepted dialog box appears successfully upon clicking accept button and the user appears in the friends list on friend’s page as intended. ***Figure 27,28,29*** | Passed |
| Test case 12:  Rejecting a request | If the request is rejected, the request is removed from the request page and the entry in removed from the friend’s page. | The rejected dialog box appears as intended and the request is no longer visible on the request page as intended and no update happens to friends list on either of users. ***Figure 30,31*** | Passed |

A screenshot of a computer

AI-generated content may be incorrect.

Figure 11: Test case 1

A screenshot of a computer

AI-generated content may be incorrect.

Figure 12: Test case 2

A screenshot of a computer

AI-generated content may be incorrect.

Figure 13: Test case 3

A screenshot of a computer

AI-generated content may be incorrect.

Figure 14: Test case 4

A screenshot of a computer

AI-generated content may be incorrect.

Figure 15: Test case 5

A screenshot of a chat

AI-generated content may be incorrect.

Figure 16: Test case 6 (Before sending message)

A screenshot of a computer

AI-generated content may be incorrect.

Figure 17: Test case 6(After sending message)

A screenshot of a computer

AI-generated content may be incorrect.

Figure 18: Test case 7: Before joining channel, channel search page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 19: Test case 7: After joining the channel

A screenshot of a computer

AI-generated content may be incorrect.

Figure 20: Test case 7: The updated channel list

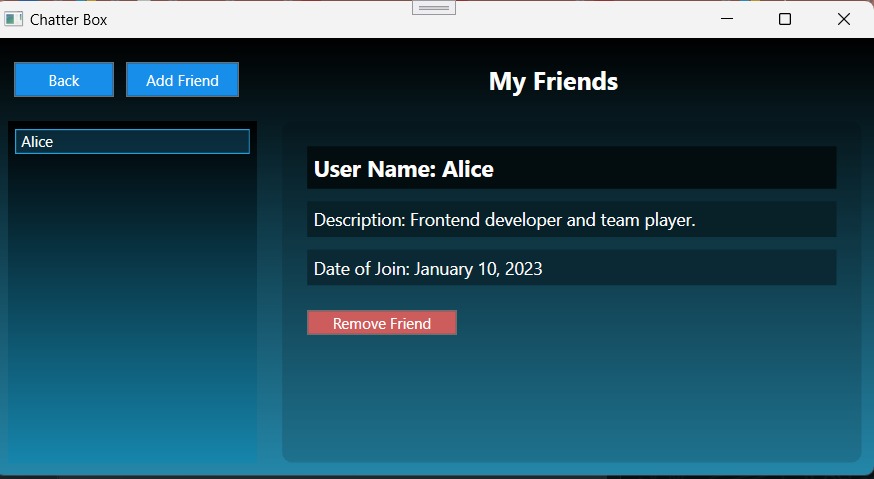


Figure 21: Test case 8: Before clicking remove friend

A screenshot of a computer

AI-generated content may be incorrect.

Figure 22: Test case 8: After clicking remove friend

A screenshot of a computer

AI-generated content may be incorrect.

Figure 23: Test case 8: Updated friend list

A screenshot of a computer

AI-generated content may be incorrect.

Figure 24: Test case 9: friend search page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 25: Test case 9: friend info page: after clicking send request

A screenshot of a computer

AI-generated content may be incorrect.

Figure 26: Test case 10

A screenshot of a computer

AI-generated content may be incorrect.

Figure 27: Test case 11: request page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 28: Test case 11: After accepting the request

A screenshot of a computer

AI-generated content may be incorrect.

Figure 29: Test case 11: Updated friend list

A screenshot of a computer

AI-generated content may be incorrect.

Figure 30: Test case 12

A screenshot of a computer

AI-generated content may be incorrect.

Figure 31: Test case 12: request page updated after rejecting

# Conclusion

In conclusion, the C# ChatterBox Application successfully met its goals and provided valuable experience in WPF XAML. We strengthened our skills in designing user-friendly interfaces, handling events effectively, and validating user input. The project emphasized the importance of clear, robust code, following best practices such as proper naming conventions and separating business logic from the UI, improving readability and maintainability.

The development and testing of this application have resulted in functional basic chat application with a solid foundation. All core objectives were met. The use of WPF enhanced the user interface. The debugging part of the code was equally very tuff, learning, and fun experience. Keeping this project as a foundation, our team is planning to enhance this project based on what we will learn in upcoming future. In no time, ChatterBox will turn into a chat application with no limitations.

Future improvements could enhance chat experience. Using specialized mechanism like SpecialR to provide real time message alert. Features like message deletion, that calls a procedure to delete a entry in the chat table and update the chat. Enhancing password policy including the symbol requirement to enhance security.

Overall, the project effectively demonstrated key programming principles and provided a solid foundation for developing more complex .NET applications.

# References

|  |  |
| --- | --- |
| [1] | w3school, “w3school,” 1999. [Online]. Available: https://www.w3schools.com/cpp/cpp\_date.asp. |
| [2] | “AI Skills fest,” Microsoft , 9 16 2024. [Online]. Available: https://learn.microsoft.com/en-us/sql/connect/ado-net/retrieving-modifying-data?view=sql-server-ver16. |
| [3] | Heinzi, “Software engineering,” 22 January 2018. [Online]. Available: https://softwareengineering.stackexchange.com/questions/364448/how-would-i-design-a-tryparse-method-which-provides-detailed-information-in-case#:~:text=When%20parsing%20user%20input%2C%20it,on%20invalid%20data. |