1. Project Overview

The team is developing a multi-user, TCP-based chat application with its own client, server, and SQLite-based database. The goal of the project is to create a stable, multi-client communication system with user management, secure authentication, chat history, and GUI support.

2. System Architecture

The system consists of three main components:

2.1 Client

Responsible for connecting to the server, sending commands, receiving messages, and providing user interaction.

Functions:

- TCP client connection
- Request username at startup
    - if empty: "ano"
- Interpret user commands
- Background message listener task
- Send messages

Commands:

Command	Function
/register <user>	registration – sends:

<pass>	REGISTER : user : pass
/login <user>	login – sends:
<pass>	LOGIN : user : pass
/quit	close client
anything else	chat message

2.2 Server (ChatServer)

The central controller that accepts client connections, performs authentication, broadcasts messages, and logs operations.

Main functions:

- Handle multiple clients using ConcurrentDictionary
- Broadcast to all connected clients
- Server-side logging (all operations logged)

Command processing:

- USERNAME : – set username
- REGISTER : – save new user to DB
- LOGIN : – login + chat history (20 messages)
- chat message → save to DB + broadcast

2.3 Database

Tables:

Users

- Id
- Username
- PasswordHash

Messages

- Id
- Sender
- Content
- Timestamp

Operations:

- AddUser(username, passwordHash)
- ValidateUser(username, passwordHash)
- AddMessage(sender, content)
- GetLastMessages(count)

2.4 Security

- No plaintext password storage
- Validation during login

3. Program Operation

Client startup:

1. Request username a. If empty, "ano"
2. TCP connection to server
3. Listener task starts in background
4. Command or chat message can be sent

Registration /register <user> <pass>

- Server checks if user exists
- If yes → error
- If not → new record created in DB

Login /login <user> <pass>

- SHA256 hash validation
- On success → last 20 messages with timestamp sent

Chat

- Every chat message:
  - saved to DB
  - broadcast to all clients

Exit /quit → client window closes Server removes connection

4. GUI Status and Current Issues

GUI is in the *gui* branch, inside the *Forms* folder.

Current issues:

- DLL not accepted → new separate class created
- Connection drops after registration/login
  - likely async/socket handling issue
  - or multiple threads writing to same stream
- Could not push from terminal → one large commit
- GUI template exists → functionality to be added later

5. Testing Documentation (N.N)

5.1 Testing Goals

- Verify stable client-server communication
- Check correct operation of functional commands
- Test database operations
- Identify and document faulty behavior
- Test GUI once functions are implemented

5.2 Testing Environment

- Visual Studio 2022
- Windows 10/11
- Custom TCP-based server + clients
- Multiple clients (2–3) launched for testing

5.3 Testing Process

1. Build + start server
2. Build + start client
3. Test username input
4. Test registration commands
5. Verify login process
6. Check chat broadcast with multiple clients
7. Verify message timestamps
8. Check correctness of chat history
9. Test disconnect handling
10. Test error handling and edge cases

5.4 Test Cases

5.4.1 Registration

TC ID	Description	Input	Expected Result
TC_REG_01	Register new user	/register test 123	Successful registration
TC_REG_02	Existing user	/register test 123	Server error message
TC_REG_03	Empty name	/register "" 123	Error message

5.4.2 Login

TC ID	Description	Input	Expected Result
TC_LOG_01	Correct login	/login test 123	Login + 20 messages

TC_LOG_02	Wrong password	/login test wrong	Error message
TC_LOG_03	Nonexistent user	/login someone 111	Error message

5.4.3 Chat Functions

TC ID	Description	Input	Expected Result
TC_MSG_01	Send message	Hi	Broadcast to all clients
TC_MSG_02	Save message to DB	send message	Appears in DB with timestamp
TC_MSG_03	Multiple clients	3 clients chatting	All messages appear everywhere

5.4.4 Disconnect

TC ID	Description	Input	Expected Result
TC_DISC_01	Quit command	/quit	Client closes → server removes
TC_DISC_02	Unexpected close	X icon	Server logs disconnect

5.4.5 GUI (current state)

Test	Expectation	Current State
DLL loading	GUI starts	error, custom class needed
Login/registration	Stable connection	connection drops
Chat box	Messages appear	under development
Multiple windows	Stable operation	not ready yet

6. Known Issues

- GUI → connection drops after registration/login
- In some cases client does not reconnect
- GUI branch push is one large commit (Forms folder)
- DLL compatibility issue

7. Development Suggestions

- Separate NetworkService layer for GUI
- Consistent use of async/await
- Shared communication module for GUI and CLI
- Retry logic for connection drops
- Add unit tests for database and protocol

8. Summary

The project currently has stable CLI-based chat functions:

- registration
- login
- chat
- broadcast
- DB integration
- 20-message history
- stable server and client connection (CLI)

GUI development is ongoing, with technical obstacles, but stable foundations exist.