**Computer Network Course Practice**

**Design and Implementation of** **a Multi-Functional Multi-Threaded Online Chat Platform**

**Report**

**Student ID: 1820259055**

**Name: Zendel Aleksei (阿列克谢)**

**Date: 2025.10.11**

**School of Computer Science and Technology**

**Beijing Institute of Technology**

**October, 2025**

1. **Requirements**

Requirements of the project.

It is required to design and implement a multi-threaded online chat platform with instant messaging, file transfer, collaborative editing and role management.

1. **Analysis**

Understanding of instant messaging, file transfer,collaborative editing, etc.

Instant messaging must support sending text messages from one user to another or from one to multiple, while storing the conversation history and updating chats whenever they have a new message.

Apart from sending text messages, the platform must also support transferring files between users.

On top of that, the application needs to be able to let users share files with others for collaborative editing.

1. **Design**

Please give more details about the whole project.

Server side will listen for and accept incoming client connections.

Client will be able to connect to the server via an internet browser. Once they successfully authorize themselves in the system, they will be given access to their groups and conversation history. They will be able to add other users to their groups or create new groups by adding members to them.

Inside a group, users will have different management roles: a normal user or an administrator. Administrators of a given group will be able to mute users in it or delete them from it. Also, administrators of a group will be able to give administrator rights to any group member or take them back.

In a conversation, users will be given the possibility of sending text, files for downloading and files for collaborative editing with users, for which the file was addressed to.

Collaborative editing lets different users change the contents of a file in real-time and make snapshots of it. The contents of a file will be able to be reverted to any of the snapshots.

1. **Development and Implementation**

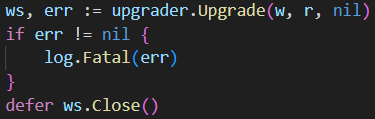
Dev. Tools (OS, languages, tools, database, framework, jar or libraries, etc.), project structure, critical function or class or structure definitions and lines of code, etc.

Client is an HTML page with logic written in JavaScript. Server is written in Go. They can send each other text messages about new events via websockets. The events are transferred in JSON format. The projects uses TLS-encryption.

After a client asks for the authorization page, the server returns it for the client to use.



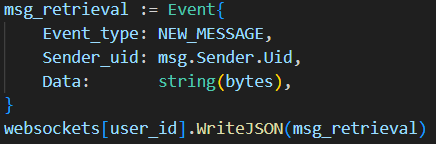
Upon entering their credentials and pressing the “Log in” or “Sign up” button, the client sends a request to the server. If the credentials are correct, the server provides the client with the chat page and a cookie for session maintenance and then upgrades the connection to using websockets.

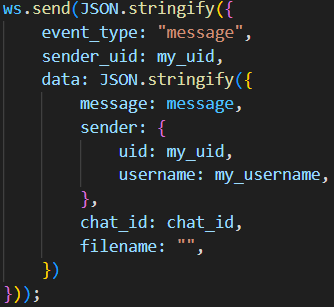


Further communication is dispatched to a worker thread.



Once the new connection is established, all communications between the server and its clients are done by notifying each other of new events by sending these events as JSON strings via the websockets (below are examples of server-to-client and client-to-server communications respectively).





To convert files to strings, base64-encoding is used.



All messages, files and collaborative editing files versions are saved on the server-side so that they can be retrieved by users logging in in the future.

1. **System Deployment, Startup, and Use**

To deploy the application without TLS, it is enough to run the server script with `go run ./server.go`.

To deploy the application with TLS, the deployer must first create certificates, set their names in the config file and change the TLS flag value to `true`.

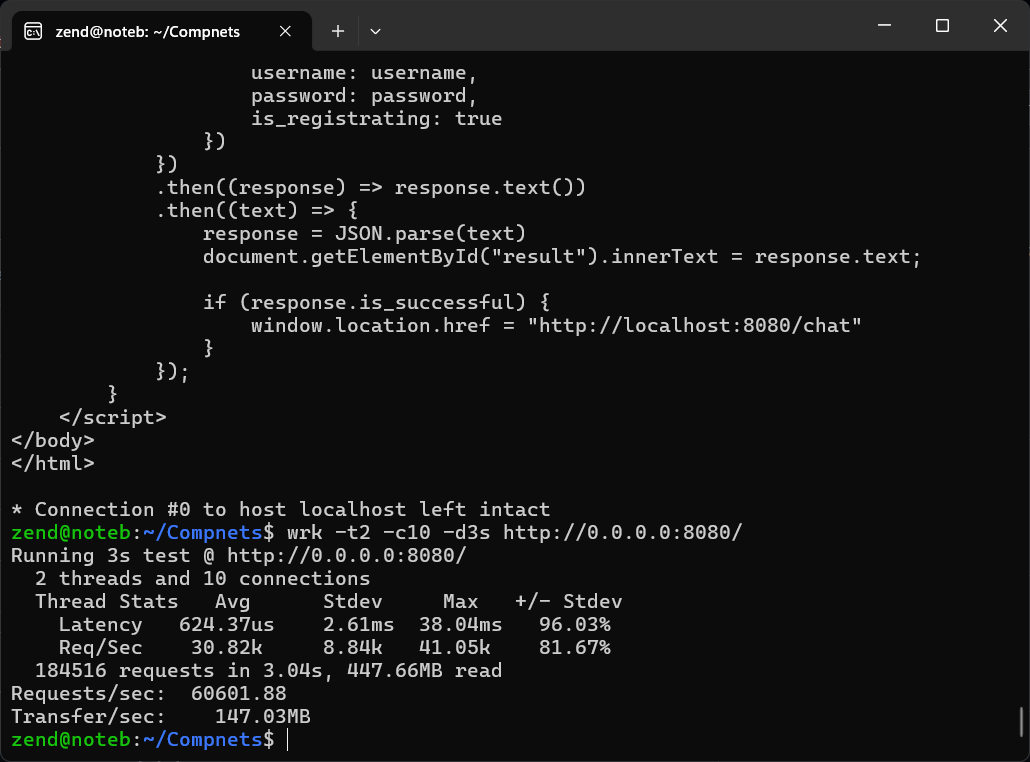
1. **System Test**

Unit test, integrated test, and the result of each test cases with screenshots, etc.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Description | What to press | What to expect |
| 1 | Open the sign up page and enter your credentials |  |  |
| 2 | Get redirected to the chat page |  |  |
| 3 | Reopen the chat tab to make sure that the session is still open |  |  |
| 4 | Reopen the internet browser to log in | - |  |
| 5 | Enter wrong credentials and press “Log in” |  |  |
| 6 | Enter correct credentials and press “Log in” |  |  |
| 7 | Perform steps №1 and №2 in a different browser |  |  |
| 8 | Perform steps №1 and №2 in a different browser |  |  |
| 9 | Create a chat group between user 0 and user 1 |  |  |
| 10 | Add user 2 to the chat group |  |  |
| 11 | Send text messages from all three users |  |  |
| 12 | As the group owner, mute another user |  |  |
| 13 | As a muted user, attempt sending messages. |  |  |
| 14 | As the group owner, grant administrator rights to another user |  |  |
| 15 | As an administrator, kick another user |  |  |
| 16 | As an administrator, attempt to kick/mute/ underpromote the group owner |  |  |
| 17 | As the group owner, revoke the administrator rights from another user |  |  |
| 18 | Send a file into the group chat |  |  |
| 19 | Download the file as a different user |  |  |
| 20 | Create a file for collaborative editing |  |  |
| 21 | Open the file as several users |  |  |
| 22 | Edit the file as one user and see the changes as a different user |  |  |
| 23 | Name this version and save it |  |  |
| 24 | Make more changes and make a save again |  |  |
| 25 | Revert to the previous save |  |  |
| 26 | Revert again to the newest version |  |  |

1. **Performance**

Performance test and result analysis, etc. Show results in data sheet and figures.



1. **Summary or Conclusions**

What you studied and analyzed, what problems to be solved, the technical solutions to solve the problems, what the results have been obtained and how about the system performance. What system features are.

1. **References**

5 References and more.

1. **Comments**

Comments and/or suggestions on the course.