**MTCG – Protocol**

1. **Design Description**

The server is designed to handle incoming requests by utilizing a single TCP socket. This socket acts as a gateway for all incoming requests to the server, allowing the server to divide the workload into separate threads for each request that is received. Each request that is sent to the server is handled by a separate thread, which is created specifically to handle that request. This allows the server to handle multiple requests simultaneously, increasing its overall efficiency and responsiveness. The use of a single TCP socket and thread-per-request model allows the server to effectively handle a large number of requests simultaneously, while also providing the ability to handle individual requests quickly and efficiently.

Additionally, the use of a single TCP socket also allows for easy scaling of the server as more resources can be added to handle an increasing number of requests. This design also simplifies the management of the server, as all incoming requests are funnelled through a single point of entry, making it easier to monitor and troubleshoot any issues that may arise. In summary, the server is designed to handle incoming requests by utilizing a single TCP socket which acts as a gateway for all incoming requests to the server. Each request is then handled by a separate thread, allowing the server to handle multiple requests simultaneously and efficiently. This design allows for easy scaling and simplified management of the server.

1. **Lessons Learned**

- Concurrency with threads: The importance of utilizing threads to handle multiple requests simultaneously was learned. This allows for increased efficiency and responsiveness in the server.

- TCP sockets in a scalable web app: The use of a single TCP socket as a gateway for all incoming requests to the server was implemented. This allows for easy scaling of the server as more resources can be added to handle an increasing number of requests.

- Secure database connection: The importance of securing the connection to the database was learned. This includes implementing secure authentication methods and properly handling sensitive information.

- User authorization through token: The use of tokens for user authorization was implemented. This allows for secure and efficient management of user access to the system.

- General good practices: The use of design patterns, such as the singleton pattern, was implemented to improve the overall design and maintainability of the code.

1. **Unit Testing**

Hard-coded rows in the database are required for the unit tests. These include rows for registerUser(), loginUser() und logoutUser().

1. **Estimated Tracked Time**

Approximately 50-55 hours were spent working on the whole project. This includes:

- Configuring the database and establishing a connection to the program.

- Configuring the server and dividing the workload into separate threads.

- Developing each class inside the manager folder.

- Developing each class in the project folder.

- Writing and running tests for the project.

1. **GitHub Link**