Assignment 2

Socket Programming: Developing a Chat Application Using Python and AI-Assisted Development

Aim

This is a group assignment, and the grouping will be the same as that of assignment 1. The assignment exposes students to socket programming in Python and explores how AI tools (like ChatGPT) can be leveraged to generate and refine software solutions. Students will develop a client-server chat application, analyze AI-assisted outputs, and critically assess their limitations and benefits. Additionally, students will produce test cases and architectural documentation (UML) to demonstrate comprehensive understanding and design clarity.

Requirements

1. Application Development

1. **Client-Server Chat Application**:
    - User authentication with unique usernames.
    - Broadcasting messages with sender identification.
    - Essential commands:
        - **@quit** Disconnects a client from the server and informs others.
        - **@names** Lists all connected usernames.
        - **@username <message>** Sends a private message to a specific user (e.g., @Stella Are you free next week?).
    - **Group management**:
        - **@group set ggg xxx, yyy, zzz** Creates a group ggg with specified members xxx, yyy, zzz.
        - **@group send ggg <message>** Sends a message to all members of group ggg.
        - **@group leave ggg** Removes the user from group ggg.
        - **@group delete ggg** Deletes the group ggg.
    - **Error Handling**: Ensure the program gracefully handles errors such as invalid usernames, duplicate group names, and non-existent groups, among others.
    - **Develop the application to support command line interface**. No extra marks will be awarded for graphical user interface.

2. **AI-Driven Development Process**:
    - Use ChatGPT (or another generative AI tool) to assist in generating and refining code.
    - Document the AI interaction:
        - **Prompts** used and **responses** received.
        - **Changes** made to improve AI-generated code.
    - **Critique** the AI-generated code, focusing on:
        - Strengths, weaknesses, and potential pitfalls.
        - Code quality, maintainability, and suggested improvements.
3. **Enhancements**:
    - Propose and implement at least **one additional feature** (e.g., message encryption, chat history, or customizable user statuses).
    - Justify why you chose that feature(s) in your report.
4. **Testing & Architecture Documentation**:
    - **Test Plan & Test Cases**
        - Create a **comprehensive test plan** detailing how you tested each mandatory feature and your additional enhancement(s).
        - Include **normal flows** (e.g., successful login) and **edge cases** (e.g., duplicate username, invalid commands, etc.).
        - Demonstrate **AI involvement**:
            - Generate an initial set of test cases using ChatGPT.
            - Critique and expand on those AI-suggested tests to ensure thorough coverage.
        - Provide for each test case: purpose, steps, expected result, and actual result.
    - **Architectural Design**
        - Provide **UML and architectural diagrams** (e.g., class diagrams or module diagrams, sequence diagrams) illustrating how the client(s) and server interact, including the chat application message format, protocol, message sequencing, etc.
        - Show how messages are handled, how group membership is stored, and how various commands are processed.

        - In the report, discuss **how/why** you arrived at (or evolved from) ChatGPT's initial design suggestions.

5. **Critical Reflection**:
    - Reflect on the **AI-assisted development process**:
        - How did ChatGPT influence your coding style or approach?
        - What were the key benefits and challenges of using AI tools in your workflow?

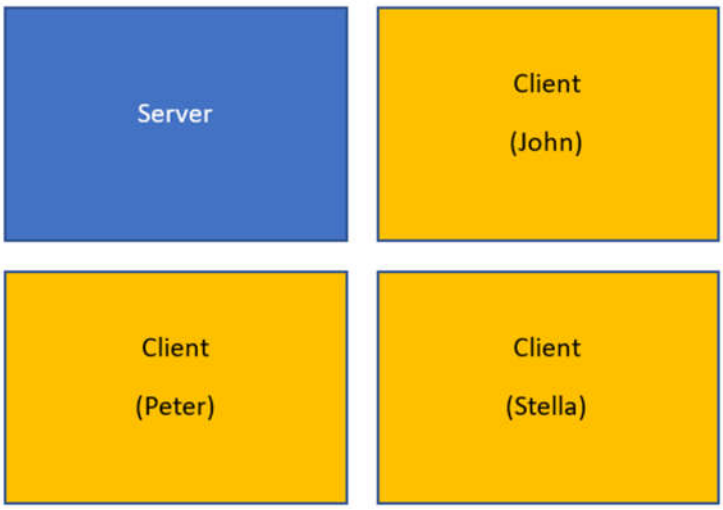
Deliverables

1. **Submission**:
    - Prepare a **zip file** structured as follows:

```
GroupName.zip/
├── codes/
│   ├── server.py
│   ├── client.py
│   └── README.md (explains instructions, and group details)
├── video/
│   └── video_url.txt (contains URL to a YouTube video. Do not
attach the raw video file)
├── report/
    └── report.pdf
```

    - **README.md**:
        1. Features implemented.
        2. Instructions to run the application.
        3. Group members (name, student ID, email).
    - **Name the zip file as GroupName.zip** (e.g., G1.zip for Group 1).
2. **Report** (1500–2500 words, not inclusive of appendix)
    - **Introduction**: Overview of the project, objectives, and scope.
    - **Development Process**:
        - Prompts and responses from ChatGPT (summaries or excerpts).
        - Changes made to the AI-generated code.
    - **Code Analysis**:
        - Strengths and weaknesses of AI-generated code.

        - Improvements implemented (and reasons).
    - **Testing & Architecture**:
        - **Test Plan** with coverage of all mandatory features and enhancements.
        - Explanation of **test-case generation** (both AI-suggested and manually refined).
        - **UML diagrams** (class diagrams, sequence diagrams, etc.) and explanation of system design.
    - **Reflection**:
        - Evaluation of the AI-assisted process.
        - Lessons learned and overall experience.
    - **Conclusion**: Summary of key findings and final thoughts.
    - **Note**: The above report's structure is just a suggested structure, and ***it is by no means complete. You are free to structure in whatever ways*** that will enhance the quality of the report.
3. **Video Demonstration** (5–8 minutes)
    - **Screen Setup**:
        - Four consoles in view:
            - **Top-left**: Server console.
            - **Top-right**: John's client.
            - **Bottom-left**: Peter's client.
            - **Bottom-right**: Stella's client.



- **Demo Flow**:
    1. Start the server and connect clients (John, Peter, and Stella).
    2. Attempt to connect with a **duplicate username** (e.g., another "John").
    3. Show:
        - Broadcasting messages.
        - Private messages (@username).
        - @names command listing all users.
        - Group creation, messaging, leaving, deletion (@group commands).
        - **Test your additional feature** (if relevant).
    4. Disconnect clients using @quit.
    5. You are free to exercise creativity in presenting your work. The above sequence of presentations/demo is only a suggested guideline.
    6. Video should be uploaded to Youtube. DO NOT submit the video file to the LMS (see submission instructions).

Assessment Criteria

1. **Functionality (30%)**
    - Implementation of core features (client/server, commands, group management).
    - Successful integration of at least one additional feature.
2. **AI Utilization & Critique (30%)**
    - Effectiveness of ChatGPT in your workflow.
    - Depth and clarity of AI-generated code critique.
3. **Report Quality (25%)**
    - Completeness and clarity of all required sections (including test plan and UML diagrams).
    - Logical flow and adherence to word count.
4. **Video Quality (15%)**
    - Clarity of demonstration (set up, narration, and completeness).

Notes

- Cite ChatGPT (or other AI tools) as a resource where relevant.
- Late submissions incur penalties as per SIT policy.
- Ensure your **test plan** and **UML diagrams** meaningfully illustrate your application's design and confirm that all features (and edge cases) function as intended.