**Phase I: Basic YallaChat (80 pts)**

Client Side:

1. **Graphical User Interface (GUI)**:
   * Design and implement a user-friendly GUI using Python libraries like Tkinter or PyQt.
2. **Connecting to the Server**:
   * Allow users to connect to the server using the server's domain name and port number.
3. **User Authentication**:
   * Implement registration and login functionalities.
   * If a user is not registered, provide a signup option to gather necessary details.
   * For registered users, authenticate them by verifying their username and password.
4. **Displaying Chats**:
   * Upon successful authentication, display a list of chats the user has had with friends or groups.
5. **Adding Friends**:
   * Provide functionality to add friends and display their online/offline status.
6. **Communication with Friends**:
   * Enable users to initiate communication with selected friends and exchange text messages through the server.

Server Side:

1. **Listening on Port**:
   * Implement the server to listen on the specified port number.
2. **User Management**:
   * Allow users to register and login.
   * Maintain a database to store user accounts.
3. **Message Handling**:
   * Handle messages sent by users.
   * If the recipient is offline, store messages in the repository to deliver later.
4. **Handling Multiple Clients**:
   * Implement a multithreaded server to handle multiple clients simultaneously.

**Phase II: Extending YallaChat to support advanced features (20 pts)**

1. **Creating Groups**:
   * Allow users to create groups of friends and communicate with them.
2. **Multimedia Message Exchange**:
   * Enable users to exchange various types of multimedia messages such as text, voice, pictures, and videos.
3. **Hybrid Client/Server and Peer-to-Peer Architecture**:
   * Implement a hybrid architecture where users can communicate with each other in a peer-to-peer manner after obtaining the friend's address from the server.
4. **Group Communication**:
   * Choose and justify one of the options for group communication: server broadcasting messages to all group members or establishing peer-to-peer connections between group members.
5. **Optional Features**:
   * Implement additional features like visiting selected chats, efficient searching based on topics, and any other creative feature of your choice.

**Project Management:**

1. **Group Work Distribution**:
   * Clearly define the workload distribution among group members and document it in the report.
2. **Documentation and Reporting**:
   * Provide comprehensive documentation for the source code.
   * Prepare a 6-page report divided into two main parts detailing each project phase.
   * Include appendices with snapshots of the application, a breakdown of project tasks, and the responsible group member for each task.
3. **Demo and Presentation**:
   * Perform a demo to present the application and all supported features.
   * Be prepared to answer any questions related to the implementation of the project.
4. **Project Submission**:
   * Designate one member to submit the project deliverables (code and report) to Moodle.

**Additional Tips:**

* Plan the project timeline, allocating sufficient time for each phase and task.
* Use version control (e.g., Git) for collaborative development.
* Test the application thoroughly to ensure functionality and stability.
* Document any challenges faced during development and their solutions.