# **Project Design Document**

## **Overview**

The project involves the development of a TCP-based client-server system for handling user accounts, financial transactions, and authentication processes.

Components

1. **Client**
   * **Responsibility**: Engages in money operations, account administration, user registration, and login interactions with the server.
   * **Features**: Registration, Login, Lodgement, User Listing, Password Update, Transaction Viewing, Logout.
2. **Server**
   * **Responsibility**: Handles incoming client requests, manages user accounts, and maintains transaction logs.
   * **Features**: User Registration, Login Authentication, Lodgement, Password Update, User Listing, Transaction Processing.
3. **Account**
   * **Purpose**: Represents a user account, storing user details and balance.
   * **Attributes**: User ID, User Details (name, email, password, address), Current Balance.
   * **Features**: Getter and Setter methods for user details and balance management.
4. **Transaction**
   * **Purpose**: Defines a transaction structure, recording date, amount, transaction type, sender, and recipient.
   * **Attributes**: Date, Amount, Type (Lodge/Transfer), Sender ID, Recipient ID.
5. **User**
   * **Purpose**: Holds user information independently.
   * **Attributes**: Name, ID, Email, Password, Address, Balance.
   * **Features**: Getter methods for user details.

**Client Implementation**

* **User Interface**: Offers a command-line interface for user interactions.
* **Login & Registration**: Facilitates user authentication and registration using provided credentials.
* **Account Operations**: Allows users to lodge money, view registered users, transfer funds, view transactions, and update passwords.

**Server Implementation**

* **Socket Handling**: Listens for incoming connections and creates a separate thread for each client.
* **Account Management**: Manages user accounts, verifies logins, processes financial operations, and maintains transaction logs.
* **Data Persistence**: Stores user account information in a text-based database file.

**Communication Protocol**

* **Client-Server Interaction**: Communication occurs via object streams using predefined protocol messages.
* **Protocol Commands**: REGISTER, LOGIN, LODGE, TRANSFER, UPDATE\_PASSWORD, RETRIEVE\_USERS, VIEW\_TRANSACTIONS, LOGOUT.

**Error Handling**

* **Client Side**: Handles invalid input, login failures, and unsuccessful transactions.
* **Server Side**: Manages duplicate IDs/emails, insufficient balances, password mismatches, and recipient not found scenarios.

**Enhancements and Future Work**

* **Encryption**: Implement encryption for secure data transmission.
* **Improved User Experience**: Provide a graphical user interface for the application running on the client side.
* **Optimization**: Improve server performance to manage several connections at once.

**Conclusion**

Secure connection between clients and the server, account management, and financial transactions are all made possible by the powerful functionality of the TCP-based client-server system that has been put in place.

This structure should aid in the proper organisation of the design document, explaining various parts of your project and how the code components contribute to its operation. You can elaborate on each item by going over specifics, options for implementation, and things to think about down the road.

A screenshot of a computer

Description automatically generated