

# Money Hub

Software Engineering Project

Software Design Documentation

Created by:

John Neis & Sam Dressler

CSCI 463- Software Engineering

4/29/2020

## Table of Contents

### Revision History

#### 1. Introduction

- 1.1. *Document Purpose*
- 1.2. *Product Scope*
- 1.3. *Intended Audience*

#### 2. Design Considerations

- 2.1. *Assumptions and Dependencies*
- 2.2. *General Constraints*
- 2.3. *Goals and Guidelines*
- 2.4. *Development Methods*

#### 3. Architectural Strategies

- 3.1. *Programming Language*
- 3.2. *Component Reuse*
- 3.3. *Further Work*
- 3.4. *User Interface*
- 3.5. *Software Interface*
- 3.6. *Error Detection and Recovery*

#### 4. Software Architecture

- 4.1. *System Architecture*
- 4.2. *System Decomposition*

#### 5. Policies and Tactics

- 5.1. *Coding Guidelines and Conventions*
- 5.2. *Plans for Testing Product*
- 5.3. *Maintaining Product*
- 5.4. *Organization of Source Code*
- 5.5. *Generating System Deliverables*

#### 6. Detailed System Design

- 6.1. *Classification*
- 6.2. *Definition*
- 6.3. *Responsibilities*
- 6.4. *Constraints*
- 6.5. *Compositions*
- 6.6. *Uses and Interactions*
- 6.7. *Resources*
- 6.8. *Processing*
- 6.9. *Detailed Software Design and Interfaces*

## Revision Summary

Version	Name	Description of Change	Date
1.0	Sam Dressler	-Document structure and initial layout -Initial complete draft of section 1. Introduction -Initial complete draft of section 3. Architectural Strategies -initial complete draft of section 5. Policies and Tactics	4/29/2020
2.0	Sam Dressler  John Neis	-added assumption to the design considerations section 2.2 -client considerations for section 6 -Completed sections 2, 4, and server and database components of section 6.	4/30/2020

## 1. Introduction

### 1.1 DOCUMENT PURPOSE

This document contains the detailed design of the Money Hub system. The design will cover the user interface, the backend server, and the SQL database. Furthermore, the document will cover limitations that affect the development of the system. System architecture, policies, developmental strategies, and preliminary talk of testing will also be included. For a general overlook of the document, consult the table of contents found at the top of this report.

### 1.2 PRODUCT SCOPE

Money Hub was launched with the ideal in mind of simplifying how people manage their finances independently. The primary goal of creating Money Hub is to give users access to a simple and easy to use application that enables tracking of expenses and savings. The tools being designed into the product will prepare customers for financial challenges that will no doubt present themselves in life.

Money Hub will provide the user with visibility to a variety of their accounts in one place. Example of the accounts that can be viewed by the user can range from savings and checking to investments and loans.

Where Money Hub goes beyond your typical personal finance application is its access to showing the user their investment portfolios being used in online investment firms. On top of that, one of our goals in creating the system is too be able to see what debts the user has in car, student, and other various loans.

Money Hub will create a comprehensive summary of this information that is visible all in one place. The project team believes that allowing users to see all this is the first step in increasing the financial intelligence of the general American population.

### 1.3 INTENDED AUDIENCE

Software design documentation is primarily written with the purpose of allowing developers, system architects, and software testers a document to consult when they have questions regarding specific areas of the software.

In addition to developers, the design documentation will also serve the purpose of giving the customer a view of how the product is being developed. Referred to as the customer in the previous sentence, Dr. Hassan Reza, will be the primary evaluator of the system and this

document will contain the necessary information to understand how the application will be structured.

## 2. Design Considerations

### 2.1 ASSUMPTIONS AND DEPENDENCIES

At the current phase of development, the Money Hub system considers several assumptions that will affect the system. The first of these, is in the testing of the product. Because the product will rely on displaying information that is taken from external parties, the application will have to set these fields with pre decided information in order to display the desired results in the prototype.

Secondly, the system relies on the use of network communication to retrieve information, as well as communication between the server application and the database system. The communication between these components must be reliable to ensure unimpeded use of the system. As of the present time, the project will work on the Windows operating system. Support for other operating systems may, or may not, appear in the production release.

### 2.2 GENERAL CONSTRAINTS

Hardware used to run this product must have a semi-reliable internet connection. Without this, there can be no communication between the client and the server, and thus an end user will never be able to view any relevant information.

To that end, the server must always remain operational. If a user needs to view their account at unusual times, for instance in a time of emergency, the user must be able to access their information. If the server needs to be made inoperable (for servicing/maintenance for example), ample warning should be provided to all users of the system.

Network communication will utilize a reliable protocol (TCP) in order to ensure there is no data loss or corruption when displaying user account information.

For the server, the application must be able to interface with a relational database system. The relational database system will be used to store and maintain user data.

### 2.3 DEVELOPMENT METHODS

Development of the product was conducted via the waterfall method. Small, proof of concept module prototypes was developed alongside the documentation and design of the product following a model more akin to agile development.

## 3 Architectural Strategies

### 3.1 PROGRAMMING LANGUAGES

The Money Hub System Architecture is currently being developed as a desktop application. Production versions of the system will allow for downloads and installation off an external site. The prototype of the system being discussed in this report, will consist of three main components. Each of these components will be implemented using unique programming languages.

The first component is how the users will interact with the system. This part of the architecture will be discussed further in section 3.4 of the system design documentation. The language chosen to implement this will be C#. One of the main reasons C# was used was the ease in which it allows for graphical user interfaces to be developed through Microsoft visual studio. The language also offers the ability to connect to servers that are written in java.

The second component, the server, will act as a conduit for which the user will communicate to the database. Java has options that make it a prime candidate because it can communicate with clients written in other languages.

The final component of the architecture is the database. The Structured Query Language, otherwise known as SQL, was chosen in order to maintain the relative information for the system. Since Money Hub is a query-based system, SQL makes sense for the development team to use for this.

### 3.2 COMPONENT REUSE

The current design for the system does not make use of any existing external components. The system will however offer means for the system to reuse internal data structures and functions.

### 3.3 FURTHER WORK

The System architecture will eventually need to include means in which it can query information from partner banks and firms off the internet. This web API is not currently in development because of the cooperation needed from these third parties. Additionally, the system would eventually be developed for mobile and web use in order to increase the access of the system to clients.

### 3.4 USER INTERFACE

The Money Hub user interface is designed with ease of access of user information in mind. Since the client application is currently being designed for Windows, there is a lot of flexibility with how we organize and size the display. A careful balance of how much information is available on one page and application size is needed in order to make sure

that the user has an enjoyable experience while using the system. The interface will need to adhere to response time requirements outlined in the SRS in order to offer responsive and interactive experience while navigating the client.

### 3.5 SOFTWARE INTERFACE

Software interfaces are how the user client, accessed through the user interface, will communicate and query the database. Requests from the user will be coded and sent via a server connection that contain the request in json or xml format.

### 3.6 ERROR DETECTION AND RECOVERY

From the current point of development, many errors will not be discovered until product testing begins. Testing and error detection will be discussed further in future documents. The current design phase is being completed with initial proof of concept of interaction for the system components as its primary focus.

## 4 Software Architecture

### 4.1 SYSTEM ARCHITECTURE

The product will operate as a query-based system where the users will send requests to a server via the clients User interface. The centralized server will allow each of the unique client applications to request the information needed by the current user logged on that

port.

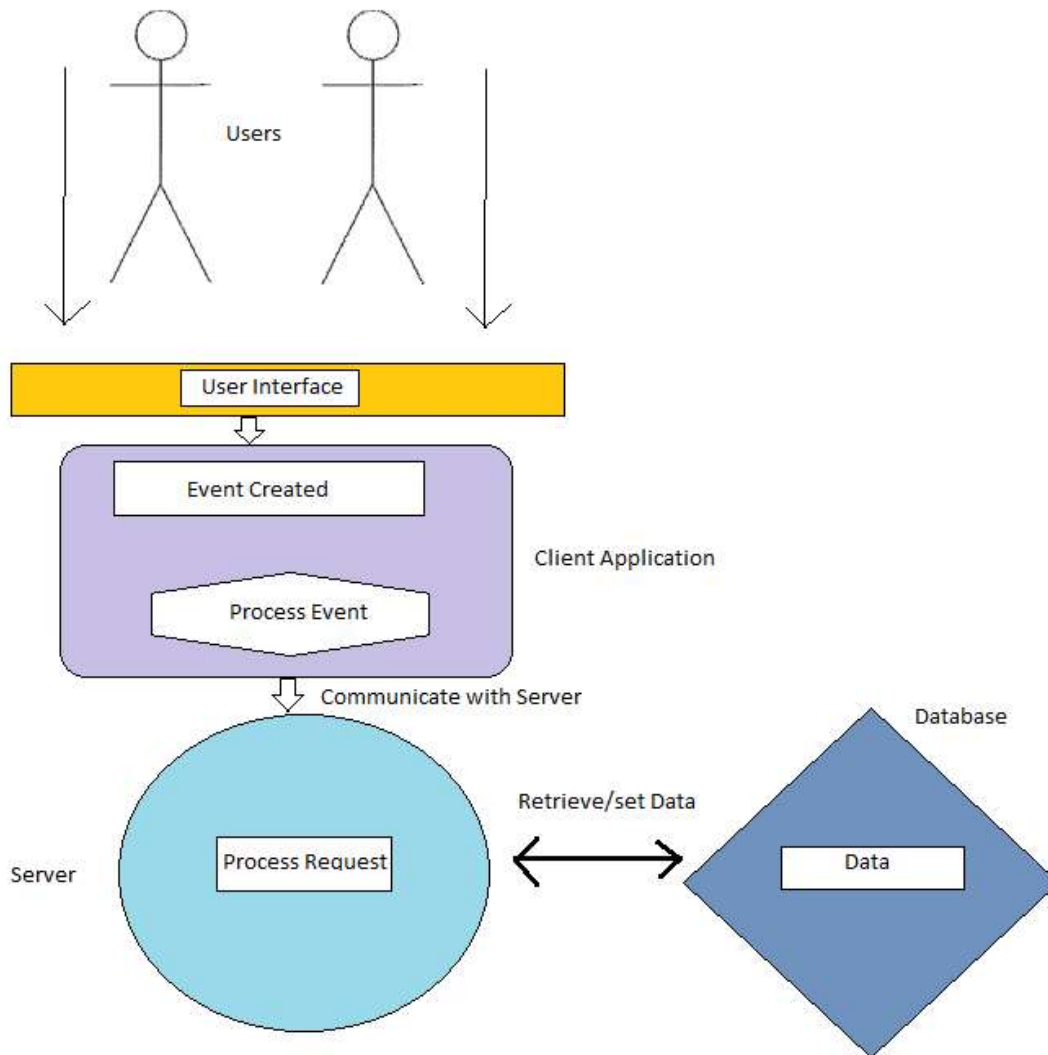


Figure 1 System Architecture Diagram

## 4.2 SYSTEM DECOMPOSITION

**Client:** The client module will request the user provide account login information. In the final release of the product, the client will encrypt the login information before sending it to the server. Once the server responds, the client will display either the information requested by the user, or a failed login message, prompting the user to resubmit their information.

**Server:** The server will remain in a state of standby, listening for login requests from clients. The server will parse the information received from the client, and if the login information is valid, will build a string of data attributed to the user and send it back to the client. In the final release, this information will also be encrypted.



Database: The database will also remain in a state of standby, as its action will be paired with the server's. The server will maintain the validity of the database.

## 5 Policies and Tactics

### 5.1 CODING GUIDELINES AND CONVENTIONS

Strict coding guidelines will be used in the implementation of the Money Hub System. The reason being it allows for more rapid and effective development. For the development of the C# and Java components of the system, several guidelines will be followed and discussed in this section.

The coding convention and their benefits are as follows:

- Naming Conventions- create a consistent look in the code. Allows developers to focus on the content and not the layout.
- Layout Conventions – Enables better copying, changing, and maintaining of code.
- Commenting Conventions – Enables testers and developers to better understand how the code works. Enables rapid bug fixing as well as enables non-developers to understand how the code is working

Security is also a primary concern for the Money Hub Application so a few security guidelines will also be followed:

- Secure Resource Access – Use of a database to hold passwords and user account information rather than storing in a file that the client has direct access to.
- Security-Neutral Code – Means the program runs with whatever permission is allowed for that user when they register their account. No one can change their *own* permission level once they create their account.

### 5.2 PLANS FOR TESTING PRODUCT

Various testing strategies will be used at different stages of implementation in order to ensure the components are working as designed. Unit testing will be the primary method to test various cases. Since the implementation is not yet complete at this point the testing will be saved for its own report.

### 5.3 MAINTAINING PRODUCT

Once out of the prototyping phase with a proven application, product maintenance will be essential to keeping users' level of contentedness high. Additionally, any bugs in the system will need to be periodically fixed once they are found. The system will have means by which the users can communicate any errors in the system that they encounter.

## 5.4 ORGANIZATION OF CODE

Organization of the source code shall be in a project files directory. Within that directory, there will be additional directories for the user interface as well as one for both the server and database file. This organization strategy keeps the components and dependencies for the system all in one place which allows for quicker development.

## 5.5 GENERATING SYSTEM DELIVERABLES

In order to ensure the application and each of its components are compiled correctly, a batch script seems like a valid option to set up the various dependencies in the system. A detailed user manual will be developed prior to the final delivery to ensure the evaluator can correctly use the system.

# 6 Detailed System Design

## 6.1 CLASSIFICATION

### 6.1.1 – Client Component

Files associated with a visual studio project will be included, i.e. resource files, .config files, and properties for the project. Additionally, the files that are part of the client will contain all the code that runs and displays the GUI.

The following are the classes and types of the project so far:

- accountCreator *form*
- Navigator *form*
- AccountTypeEnum enumeration
- AdminAccount class
- Login *form*
- MoneyHubClient *form*
- UserAccount class

### 6.1.2 – Server Component

- ServerDriver class
- MySQL JDBC library

### 6.1.3 – Database Component

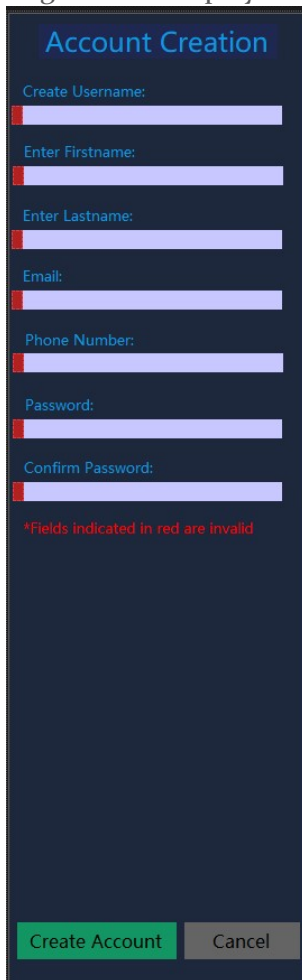
- MySQL server

## 6.2 DEFINITION

### 6.2.1 – Client Component

The client contains many classes and forms that handle the users actions while interacting the system.

-accountCreator: This is the form that is displayed while a new class is being created. If a field is invalid, it will be indicated with a red marker and the overall error message will be displayed.



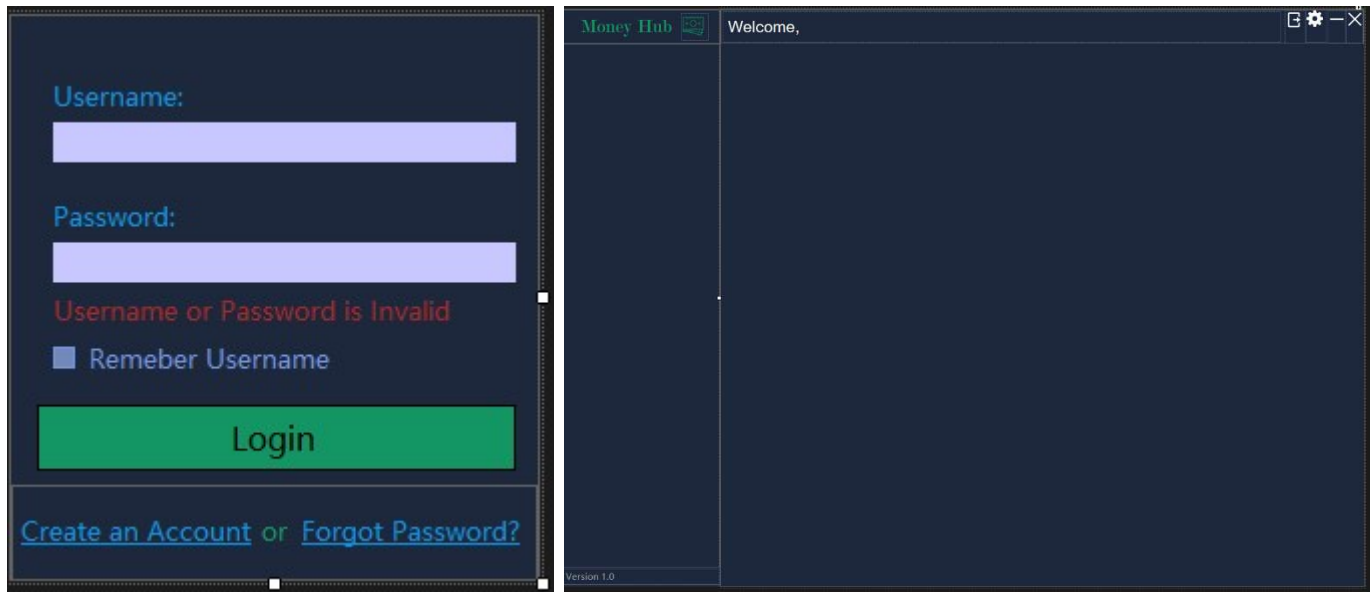
The 'Account Creation' form is a vertical layout with a dark blue background. It features several input fields, each with a red vertical bar on the left side, indicating they are required or invalid. The fields are labeled: 'Create Username:', 'Enter Firstname:', 'Enter Lastname:', 'Email:', 'Phone Number:', 'Password:', and 'Confirm Password:'. At the bottom, there are two buttons: 'Create Account' (green) and 'Cancel' (grey). A red error message at the bottom reads: '\*Fields indicated in red are invalid'.



-Navigator (above right) – this is the preliminary design of the menu that the users will utilize to navigate throughout the application

- AccountTypeEnum- this enumeration will help classify accounts so that the forms layout can be adjusted accordingly

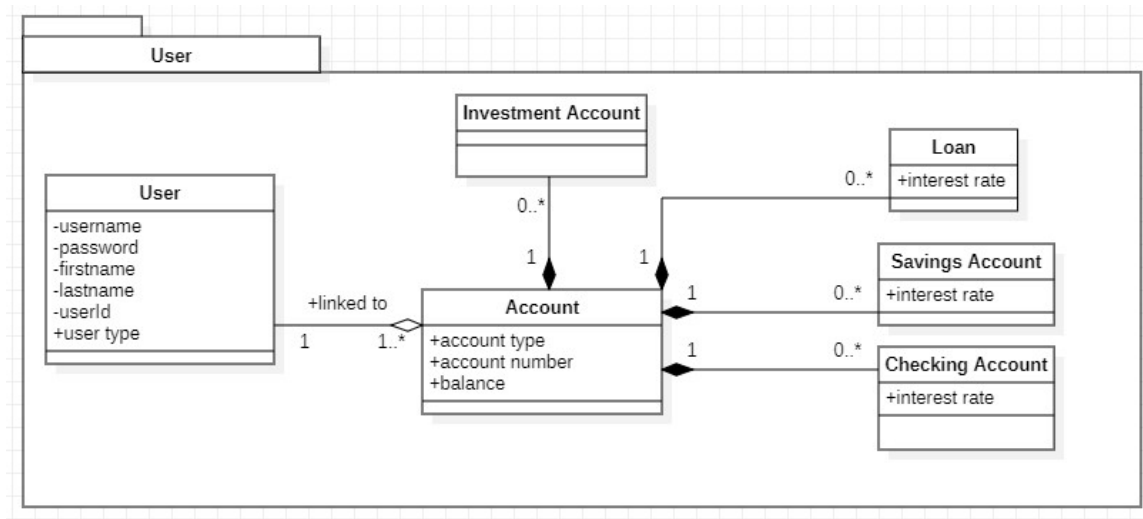
-Login (bottom left) – The first panel the users will see when they open our application. This is where the user will enter their username or password before continuing into the application. The user will also have the option to select “remember username”, “create an account” or “forgot password”



-MoneyHubClient (above right) – this is the base form of the GUI. It will hold the panels for the login as well as the navigation panel and account creator when those options are selected. Options for the settings, minimization, and exiting are in the top right of the panel. Additionally, the log out button will be visible once users successfully are logged into the system.

-UserAccount – this class will be used to store the information regarding a user’s accounts as well as other information.

The UML class diagram below will show how the user and their accounts are related:



### 6.2.2 – Server Component

- ServerDriver: The main application of the server. This is what will wait and listen for incoming requests from the client.
- MySQL JDBC library: A collection of classes that allow java applications to interface with MySQL database systems.

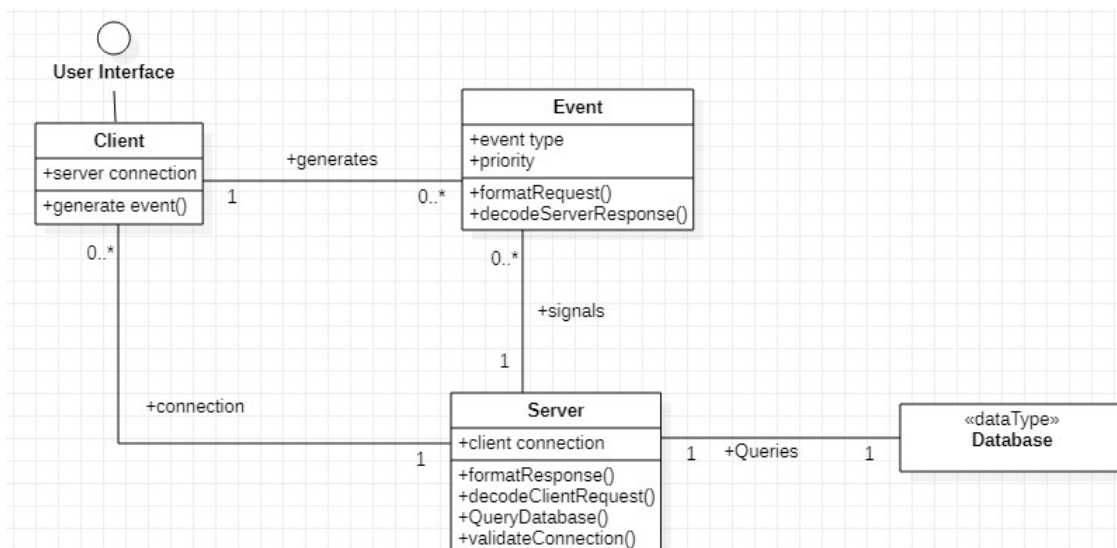
### 6.2.3 – Database Component

- MySQL server: The database system that is used to implement and maintain user account information

## 6.3 RESPONSIBILITIES

### 6.3.1 – Client Component

The client has the overall responsibility of connection interpreting the user's actions into events that the server and database will understand, and then be able to display back the results for that user.



### 6.3.2 – Server Component

- ServerDriver: Listen for request from clients, process their requests, and return either a fail state report, or the data requested by the user.
- MySQL JDBC library: Allow interfacing functionality between the server and the MySQL server.

### 6.3.3 – Database Component

- MySQL server: Store user information.

## 6.4 CONSTRAINTS

### 6.3.1 – Client Component

-Login – usernames and passwords must be retrieved and validated from the database before they can be logged into the system. This will require a connection to view the user's information. Furthermore, this will require more time to be spent validating the credentials

### 6.3.2 – Server Component

- ServerDriver: Must allow for concurrent access to the database.

## 6.5 COMPOSITIONS

### 6.3.1 – Client Component

The client is composed of various forms that allow for the display of various windows while using the product. These components were discussed in section 6.2 and will be added to as needed as development progresses. An example of a component that has not yet been implemented will be the overview window. This form will be where the summary of the user's data will be presented.

## 6.6 USES AND INTERACTIONS

### 6.3.1 – Client Component

-This component will interact with the server and database components via data sent over a TCP connection.

### 6.3.2 – Server Component

- Interacts only with the client and the database server

### 6.3.3 – Database Component

- Interacts only with the server application

## 6.7 RESOURCES

### 6.3.1 – Client Component

-GUI will require the most memory since the forms will utilize various images held in the resource folder of the program.

### 6.3.2 – Server Component

- ServerDriver: Minimal memory usage. Each concurrent access must have access to the database server. This is a possible race condition and must be mitigated by locking the database connection to a specific thread.

## 6.8 PROCESSING

### 6.3.1 – Client Component

-Logins are validated by sending the username and password entered in the fields of the login form to the server. The server will then query the database. If the username exists and the entered password matches that which is held in the database, the login will be accepted.

-Creating an account will require all the fields in the account creator form to not be empty as well as meet the requirements laid out in the requirements document.

-Once a user's account is created, the data will be entered into an object that is then formatted and sent to the server where it is parsed and stored in the database. The same process but in reverse is used when the user is logged in and the data in the overview form is loaded.

### 6.3.2 – Server Component

- ServerDriver: Will spawn a new thread for each concurrent access to the database. This will be limited the number of threads that can be executed by the system's processor.

## 6.9 DETAILED SUBSYSTEM DESIGN AND INTERFACES

### 6.3.1 – Client Component

The client has the main service of retrieving, setting, and displaying data from the database. Each user will maintain a data structure that contains all their account information. Once the user logs in successfully, since efficiency is not an enormous concern for the prototype phase, the data for each of the user's accounts is loaded into a data structure. From there, the client will parse the data into the various forms that the user can view.

The major exceptions that can affect the usage of the system are outlined in the table below.

Exception Type	Description
Login exception	User credentials are invalid
Server Connection exception	The connection between the client and the server is not valid
Data Format Exception	The data being returned from the client or from the server does not have the correct format and is being rejected.

The client will contain subroutines that create events, format data, and unpack data to be displayed in the GUI. The parameters for the format data and create event subroutines will depend on the type of subroutine being performed. The unpack data subroutine will be called in response to an event that requires data returned from the server.

### 6.3.2 – Server Component

- Terminal Interface only. Minimal human interaction possible. All interaction will be carried out through software.