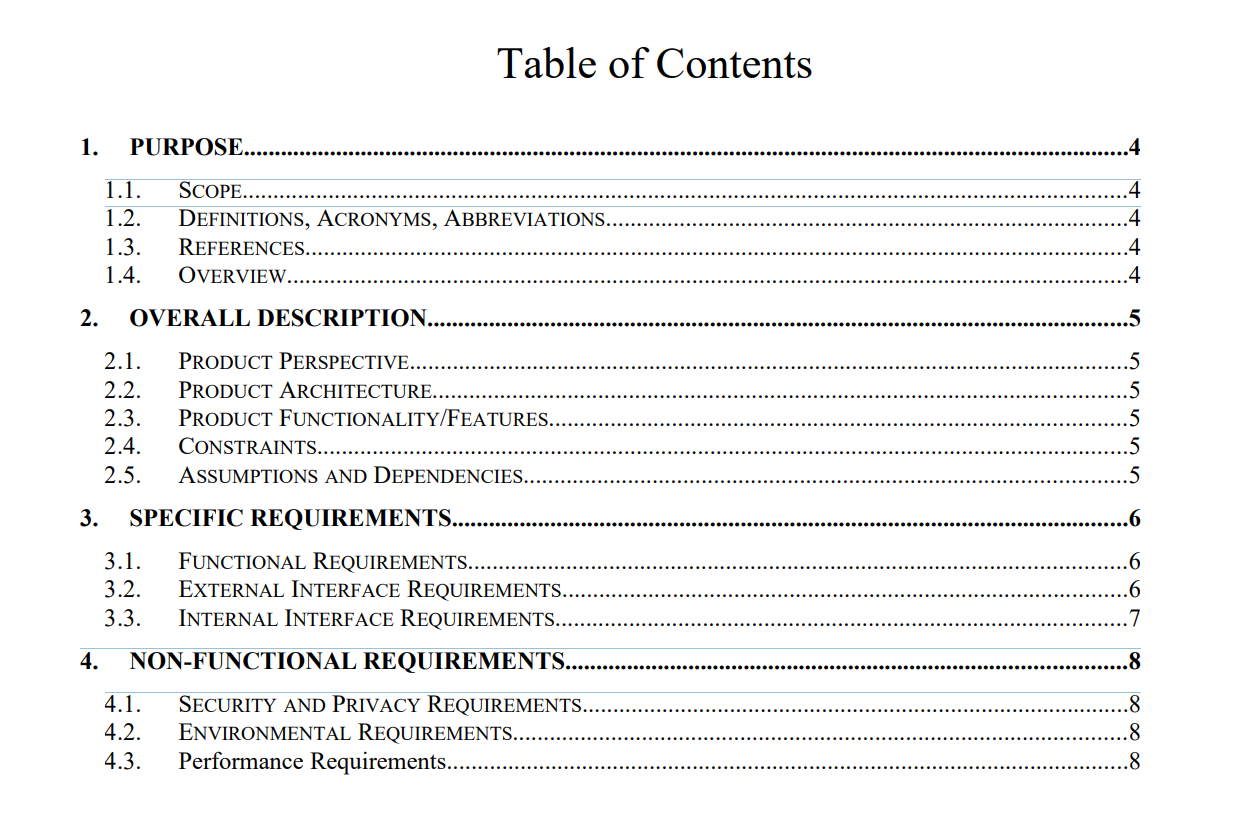
CS-401\_Group Project Requirements

# Software Requirements Specification

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Description | Author |
| mm/dd/yyyy | 1.0 | Initial Version | Your Name |
| 9/21/2024 | 1.1 | Saving and checkings Account Module | Rohan Kumar |
| 9/24/24 | 1.2 | SuperUser (Admin) Module | Alexis Rojas |
| 9/17/24 | 1.2.1 | 3.1.1.1-2 common requirement module (accounts) | Phuong Nguyen |
| 9/24/2024 | 1.3 | External Requirements | Rohan Kumar |
| 9/25/2024 | 1.4 | Revise modules to new modules type per client discussion: Operator, Account, Client, Host, Message | Phuong Nguyen |
| 9/26/2024 | 1.5 | Added part 2 and 4 | Rhenjiro Gunawan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



1. Purpose

This document outlines the requirements for a Large Banking System.

* 1. **Scope**

This document provides requirements for a Large Banking System. This software will support countless people and interactions. The primary objective being, providing a GUI for the bank employees and users to interact with their accounts. The bank has a server that stores all the data and is retrieved through the GUI. The system includes features such as Data storage, Account management, transaction operations, scalability. It operates over TCP/IP and is compatible with operating systems with a JVM.

* 1. **Definitions, Acronyms, Abbreviations** 
     1. SU - SuperUser, refers to the class of user with more permissions and functionalities.
     2. GUI - Graphical user interface.
     3. User - A regular account user. Has the ability to draw or add funds from their account.
     4. Joint Account - Multiple users can share the same account.
     5. Client - Module for client code.
     6. Server - Module for server code.
     7. Message - Module for data transfer between client and server.
     8. JVM - Java Virtual Machine.
     9. Package - Encapsulated data.

**1.3. References**

**1.4. Overview**

The large banking system will provide users with an interface for deposits, and withdrawals from their checking or savings accounts. The SU can do the same for the use, additionally, the SU can also add or remove accounts, add people to an account, or take off people from an account. It has functionalities such as account management, transaction operations, user authentication, data storage, data retrieval, scalability, security. It operates over local TCP/IP network.

2. Overall Description

**2.1. Product Perspective**

The Large Banking System is an application that should be designed to operate within a local network among multiple devices. It should provide Server and Client applications. One server should support multiple client applications. The system has several interconnected modules such as The Operator Module, Account Module, Server Module, Message Module and the Client Module. The system should be designed to be scalable, usable with various operating systems through JVM and operational over TCP/IP.

**2.2. Product Architecture**

The system will be organized into 5 major modules: the Operator module, the Account module, the Server module, the Message, and the Client module. Note: System architecture should follow standard OO design practices.

**2.3. Product Functionality/Features**

The high-level features of the system are as follows (see section 3 of this document for more

detailed requirements that address these features):

2.3.1. Users would be able to login to the system, and conduct actions such as:

* 1. Transfer funds to other accounts.
  2. Manage and see current account balance and information.
  3. Make Virtual Deposits.
  4. view Transaction history

2.3.2. Users are able to, with the help of a teller, do the following actions:

* 1. Withdraw money.
  2. Add, remove or edit access to their accounts.
  3. Make a deposit.
  4. d close account.

2.3.3. System features include:

* 1. Secure communication over TCP/IP.
  2. Data storage in a secure text file.
  3. Real-time applications such as deposit, retrieve money.
  4. scalable
  5. compatible
  6. light-weight

**2.4. Constraints**

2.4.1. The software will be made in Java; hence the software will need to keep in mind the amount of objects and keep it to a minimum at all times.

2.4.2. The software will be made for home use as well, hence we will need to keep in mind that internet connection may be unstable/interrupted during use, and proper measures will need to be taken care of.

2.4.3. The system must ensure data integrity and available, it must not get corrupted.

2.4.4 The system must not let users in if their credentials are invalid.

2.4.5 The system GUI's must be simple and user friendly.

2.4.6 The system must have protections against attempted fraud.

**2.5. Assumptions and Dependencies**

2.5.1. It is assumed that for client-teller interactions, both parties are physically together, and that the customers can provide credentials and proof of identities to the tellers.

2.5.2. It is assumed that in-person deposits are monitored and assisted by a teller such that any increase in bank balance is already verified to be correct. 2.5.3. It is assumed that there are no transfer fees for any transfer action.

3. Specific Requirements

**3.1. Functional Requirements**

**3.1.1. Common Requirements:**

3.1.1.1 There should be a standard way of storing information in the txt files.

3.1.1.2 The system must ensure data integrity and availablility with server/client pattern.

3.1.1.3 The system must be scalable.

3.1.1.4 The system must handle exceptions without conflicts with GUI.

3.1.1.5 The system will utilize multithreading programming to support multiple operators.

3.1.1.2 User may be registered to an account.

3.1.1.2.1 Each registered user gains access to the account through with the account’s reference list.

3.1.1.2.2 Adding and Removing users to account requires the SU.

3.1.1.2.3 Multiple Users may be registered to the same account.

3.1.1.2.4 The removal of the last User in an account will result in its closure.

**3.1.2. User Module Requirements:**

3.1.2.1 Operator will have unique serial IDs created upon operator initialization.

3.1.2.2 Operator must be able to authenticate using ID and passcode.

3.1.2.3 Serial ID has a pattern for distinction between type (user and superuser).

3.1.2.4 User Sub-Module Requirements:

3.1.2.4.1 Users must be able to view balance.

3.1.2.4.2 Users must be able to transfer funds.

3.1.2.4.3 Users must be able to deposit cash.

3.1.2.4.4 Users must be able to withdraw cash.

3.1.2.4.5 Users must be able to view their transaction history.

3.1.2.4.6 User will retain a list of authorized accounts IDs, these will be the account this user is registered.

**3.1.3. Checking Account Module Requirements:**

3.1.3.1 The checking account has no limit on the number of withdrawals.

3.1.3.2 The checking account has a $5 maintenance fee per month.

**3.1.3. Saving Account Module Requirements:**

3.1.3.1 Savings account has a withdrawal limit of 6 monthly withdrawals without a fee.

3.1.3.2 If the user withdraws more than 6 times, they will be charged a $5 fee for every withdrawal.

3.1.5.2 The savings account will have a 0.10% annual increase rate.

**3.1.4 Message Module Requirements:**

3.1.4.1 Message between client and host through the internet will be encapsulated.

3.1.4.1.1 Encapsulated data will be referred to a package.

3.1.4.1.2 Package will contain data, current address, receiving address, and type of data.

3.1.4.2 The encapsulated method will accept different type of information.

3.1.4.3 The encapsulated data is able to be decapsulated by the recipient.

3.1.4.4 There will be a processor, a sender, and a receiver on both side of the connection.

3.1.4.4.1 Processor can encapsulate data into package or decapsulate package into data.

3.1.4.4.2 Sender will let processor create package from passed data and send package.

3.1.4.4.3 Receiver will listen on a designated port for packages and send them to processor to get data.

3.1.4.4.4 Processor, Sender, and Receiver can handle multiple packages synchronously.

**3.1.5 Client Module Requirements:**

**3.1.6 Host Module Requirements:**

**3.2. External Interface Requirements**

**3.3. Internal Interface Requirements**

4. Non-Functional Requirements

**4.1. Security and Privacy Requirements**

4.1.1. User will need to be authenticated before getting access to any sensitive information such as bank balance, account information and account actions.

4.1.2. A sequential ordering of server request and response need to be created to prevent inappropriate sequence of actions that affects the balance.

4.1.3. Proper authentication methods need to be in place to prevent users from being able to access credentials of other users in the system.

4.1.4. Passwords and other authentication information needs to be processed server side, that is that any comparisons are done in the server and only a response is returned.

**4.2. Environmental Requirements**

4.2.1. The software needs to be “adaptive” and light weight as the software will be used by customers in a variety of environments.

4.2.2. The software needs to be internet friendly, which means that information sent to the server should be checked and verified to be complete and whole before being sent.

4.2.3. The software would need to be

**4.3. Performance Requirements**

4.3.1. Actions within the software should not take any more than 1 minute to complete.