CS-401_Group Project Requirements

1 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
10/9/2024	1.6	Final Revision	Rohan Kumar

Table of Contents

1. Purpose

This document outlines the requirements for a Large Banking System.

1.1. 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.

1.1. Definitions, Acronyms, Abbreviations

1.1.1 SU - SuperUser, refers to the class of user with more permissions and functionalities.
1.1.2 GUI - Graphical user interface.
1.1.3 User - A regular account user. Has the ability to draw or add funds from their account.
1.1.4 Joint Account - Multiple users can share the same account.
1.1.5 Client - Module for client code.
1.1.6 Server - Module for server code.
1.1.7 Message - Module for data transfer between client and server.
1.1.8 JVM - Java Virtual Machine.
1.1.9 Package - Encapsulated data.

1.3 References

UseCases.docx or UseCases.txt

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 user, 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 a 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:

a) Transfer funds to other accounts.
b) Manage and see current account balance and information.
c) Make Virtual Deposits.
d) view Transaction history

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

a) Withdraw money.
b) Add, remove or edit access to their accounts.
c) Make a deposit.
d) d close account.

2.3.3. System features include:

a) Secure communication over TCP/IP.
b) Data storage in a secure text file.
c) Real-time applications such as deposit, retrieve money.
d) scalable
e) compatible
f) 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 availability, 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 availability 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.

3.1.2. Operator Module Requirements:

3.1.2.1 Operators will have unique serial IDs created upon operator initialization.
3.1.2.2 Operators 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.

3.1.2.5 Super User (Banker) Module Requirements:
3.1.2.5.1 The superuser will have the ability to create new accounts.
3.1.2.5.2 If a checking account has not been active in the last 6 months (conditions met), the superuser has the ability to deactivate the said account.
3.1.2.5.3 If an account has a negative balance or a history of overdrafts (conditions met), the superuser will have the ability to declare the account closed.
3.1.2.5.4 The superuser can add users to accounts (or accounts to users depending on the design).

3.1.3. Account Module Requirements:

3.1.3.1 The account will have the date of creation.
3.1.3.2 The account will keep track of real time.

3.1.3.3 The account will have balance.
3.1.3.4 Accounts can either be a checking account or saving account.
3.1.3.5 The account will be able to handle depositing funds.
3.1.3.6 The account will be able to handle withdrawing funds.
3.1.3.7 The account will be able to handle destroying itself upon SU’s request.
3.1.3.8 Each account will have a unique serial ID used to identify the account.
3.1.3.9 Serial ID is generated upon account initialization.
3.1.3.10 Serial ID has a pattern for distinction between type (checking and saving).
3.1.3.11 Serial ID will be used for authorization.
3.1.3.9 Serial ID is generated upon account initialization.
3.1.3.10 Serial ID has a pattern for identification.
3.1.3.11 Account balance cannot go into negatives.
3.1.3.12 The removal of the last User in an account will result in its closure.
3.1.3.13 Each registered user gains access to the account through authentication with the account's reference list.
3.1.3.14 Only the SU can remove or add users from accounts.
3.1.3.15 Multiple Users may be registered to the same account.

3.1.3.16 Users can have multiple accounts.
3.1.3.17 The removal of the last User in an account will result in its closure.

3.1.3. Checking Account Sub-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 Sub-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.
3.1.3.3 The savings account will have a 0.10% annual increase rate.

3.1.4 Message Module Requirements:

3.1.4.1 Messages between client and Server Modules through the internet will be encapsulated in a package.
3.1.4.2 Package will contain some data, current address, receiving address.
3.1.4.3 Package will accept different types of data through utilizing adapter design models.
3.1.4.4 Package is able to be decapsulated by message.
3.1.4.5 There will be a processor, a sender, and a receiver on both side of the Connection.
3.1.4.5.1 Processor can encapsulate data into packages or decapsulate packages into data.
3.1.4.5.2 Sender will let the processor create a package from passed data and send the package.
3.1.4.5.3 Receiver will listen on a designated port for packages and send them to the processor to get data.

3.1.5 Client Module Requirements:

3.1.5.1 All operators' interactions with the application will be through client
3.1.5.2 Client will authenticate operator before allow access
3.1.5.3 Client will be able to differentiate Operators access permissions and provide functions accordingly
3.1.5.3.1 Client for User will simulate an ATM interaction
3.1.5.3.2 Client for SuperUser will simulate a bank teller interaction
3.1.5.4 Client will contain GUI
3.1.5.5 Client will receive a read-only version of data from Server for responsive performance.

3.1.6 Server Module Requirements:

3.1.6.1 The data must be stored within the server application so that it has direct access.
3.1.6.2 The server must restart if data is corrupted.
3.1.6.3 Server is able to receive requests from Client through Message module
3.1.6.4 Server will attempt to fulfill client's requests
3.1.6.5 All changes to the application's data must be perform by the server (all write actions)
3.1.6.6 Server will log upon request fulfillment, or when an exception is caught.

3.2. External Interface Requirements

3.2.1 The system must operate over a local network using TCP/IP.
3.2.2 The system must support multiple users simultaneously.
3.2.3 The system must be scalable to any number of users at any time.
3.2.4 The system must store all its data "securely" in a text file.
3.2.5 The system must be able to work in common operating systems such as Windows, Linux and MacOS.

3.3. Internal Interface Requirements

3.3.1 Server and Client must have an Interface to send and receive requests.
3.3.2 The Operator module is connected to a GUI through which all operations take place.
3.3.3 The Superuser Subclass of Operator has a slightly different GUI to accommodate for additional functionalities.
3.3.4 The Server and Client use the Message module for bi-communication.
3.3.5 The Account module must provide an interface for the Operator module to perform account-related operations such as creating, updating, and deleting accounts.
3.3.6 The Message module must provide an interface for encapsulating and decapsulating data.
3.3.7 The Server module must provide an interface for handling authentication requests from the Client module.

3.3.8 The Client module must provide an interface for users to input their credentials and receive authentication status (GUI).
3.3.9 The Server, client and account modules must provide an interface for handling deposits, withdrawals, and transfers.
3.3.10 The account module must provide an interface for keeping data received from the server temporarily to show to the user.
3.3.11 The message module must provide an interface for error handling between the client and server modules.
3.3.12 The account module must provide a way for applying interest rates, account penalties, monthly fees etc.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1. Users 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, any comparisons are done on the server and only a response is returned.
4.1.5 The data will be stored in a text file.

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. //pending more opinions
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 system must be deployable on common operating systems, such as Windows, Linux and MacOS.
4.2.4 The system is dependent on JVM 22 being available.

4.3. Performance Requirements

4.3.1 The system should be fast enough for the user to not experience a delay greater than 3 seconds.
4.3.2 Most operations such as adding or retrieving cash are constant time operations and should be executed in reasonable time. Meaning the algorithm should take O(1) time.
4.3.3 The system should not crash when under load, Example - multiple users try to access it.
4.3.4 The system should be able to handle any unexpected exceptions without crashing the app.