# Group Project – Banking System

Software Requirements Specification

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 09/24/2025 | 1.0 | Initial Version | Colin Mogilefsky |
| 09/24/2025 | 1.1 | Purpose overview added | Monica Rico-Antonio |
| 09/25/2025 | 1.2 | Functional Requirements | Colin, Christian, Monica |
| 09/25/2025 | 1.3 | Non-Functional Requirements | Colin, Christian, Monica |
| 09/25/2025 | 1.4 | UML Use Case Rough Draft | Colin, Christian, Monica |
| 09/25/2025 | 1.5 | UML Class Diagram Rough Draft | Colin, Christian, Monica |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Purpose [4](#__RefHeading___Toc19440719)

1.1. Scope [4](#__RefHeading___Toc19440720)

1.2. Definitions, Acronyms, Abbreviations [4](#__RefHeading___Toc19440721)

1.3. References [4](#__RefHeading___Toc19440722)

1.4. Overview [4](#__RefHeading___Toc19440723)

2. Overall Description [5](#__RefHeading___Toc19440724)

2.1. Product Perspective [5](#__RefHeading___Toc19440725)

2.2. Product Architecture [5](#__RefHeading___Toc19440726)

2.3. Product Functionality/Features [5](#__RefHeading___Toc19440727)

2.4. Constraints [5](#__RefHeading___Toc19440728)

2.5. Assumptions and Dependencies [5](#__RefHeading___Toc19440729)

3. Specific Requirements [6](#__RefHeading___Toc19440730)

3.1. Functional Requirements [6](#__RefHeading___Toc19440731)

3.2. External Interface Requirements [6](#__RefHeading___Toc19440736)

3.3. Internal Interface Requirements [7](#__RefHeading___Toc19440737)

4. Non-Functional Requirements [8](#__RefHeading___Toc19440738)

4.1. Security and Privacy Requirements [8](#__RefHeading___Toc19440739)

4.2. Environmental Requirements [8](#__RefHeading___Toc19440740)

4.3. Performance Requirements [8](#__RefHeading___Toc19440741)

# Purpose

This document outlines the requirements for the Banking System.

## Scope

This document will catalog the user, system, and hardware requirements for the Banking system. It will not, however, document how these requirements will be implemented.

## Definitions, Acronyms, Abbreviations

ATM – Automated Teller Machine

LOC – Line of Credit

## References

4. UML Use Case Diagram – In this document

5. UML Class Diagram – In this document

## Overview

The Banking System is designed to perform ATM services and provide a separate interface for authorized bank employees to assist customers. Fraud should be minimized, and basic banking services should be provided.

# Overall Description

## Product Perspective

## Product Architecture

The system will be organized into 2 major modules: the Client module and the Server module.

## 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):

## Constraints

No databases, libraries, or frameworks will be used.

## Assumptions and Dependencies

It is assumed that employees will properly verify the identities of anyone they are assisting.

It is assumed that employees will create a separate customer account if they wish to do business with the bank.

It is assumed that Customers will not need to change their password.

# Specific Requirements

## Functional Requirements

### Common Requirements:

2.1.1.1 There are three types of accounts available:

* Checking accounts will have no withdrawal limits and do not gain any interest.
* Savings accounts will have a daily withdrawal limit and a monthly interest rate set when opening the account. The daily withdrawal limit can be modified later, but the interest rate cannot.
* Line Of Credit (LOC) accounts have a credit limit, minimum monthly payment, and interest rate, all set when opening the account. None of these attributes can be changed at a later date.

2.1.1.2 Checking and Savings accounts must have at least 1 owner, and can have up to 5 joint owners. LOC accounts can only have 1 individual owner.

### Client Module Requirements:

2.1.2.1 Customers and Employees can log in to the client interface using a username and password.

2.1.2.2 Once logged in, Customers can access an ATM interface that allows them to check their account balances and deposit or withdraw funds from their existing accounts. No other features are available to customers.

2.1.2.3 Once logged in, Employees are able to open and close accounts, view transaction history of accounts, add or remove users to and from accounts, deposit or withdraw funds from accounts, and transfer funds between accounts.

2.1.2.4 Employees are able to view a list of accounts that a Customer has ownership of.

### Server Module Requirements:

* + - 1. There must be a method to sign up as a new Customer.
      2. Each Customer will have a unique id.

2.1.3.3 The server will keep a log of transactions for each account, which can be sent to the Client application to be viewed by Employees.

2.1.3.4 Users can only be logged in via one connection at a time.

2.1.3.5 The server will verify that a transaction is valid before completing it. (ex. Withdrawing more than account balance is not allowed, exceeding daily withdrawal limit is not allowed)

## Internal Interface Requirements

2.2.1 The system will read a comma-separated text file that includes usernames and passwords of Employees.

2.2.2 The system will read a comma-separated text file containing the list of Customers. The fields included are id, username, password, legal name, social security number, and a list of account ids the user has ownership of.

2.2.3 The system will be able to read a comma-separated text file containing the current status and transaction history of each account. The first line will include the fields account id, account type, status (open/closed), users with ownership of the account, current balance, and any other attributes specific to the type of account. The following lines contain the transaction history of the account, with each line containing the fields transaction id, amount, date, the transaction type, and name of the user performing the transaction (if applicable).

# Non-Functional Requirements

## Security and Privacy Requirements

3.1.1

## Environmental Requirements

3.2.1 The Client program should not require any external software other than Java.

## Performance Requirements

3.3.1 Transactions should be verified and completed by the server and sent back to the Client application without delay.

3.3.2 The system should not require any regular downtime to operate.

3.3.3 The system should be able to scale to a large number of users as long as it is provided sufficient hardware resources.

# UML Use Case Rough Draft

**A diagram of a bank system

AI-generated content may be incorrect.**

# UML Class Diagrams Rough Draft

A diagram of a computer

AI-generated content may be incorrect.