**Banking Application**

Group 3

Software Requirements Specification

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 02/21/2025 | 1.0 | Initial Version | R.H |
| 02/25/2025 | 1.1 | Decided on Initial set of Modules | Group |
| 03/04/2025 | 1.2 | Fixed formatting; Added definitions; Added overview | Sophia |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**1.** **Purpose 4**

1.1. Scope 4

1.2. Definitions, Acronyms, Abbreviations 4

1.3. References 4

1.4. Overview 4

**2.** **Overall Description 5**

2.1. Product Perspective 5

2.2. Product Architecture 5

2.3. Product Functionality/Features 5

2.4. Constraints 5

2.5. Assumptions and Dependencies 5

**3.** **Specific Requirements 6**

3.1. Functional Requirements 6

3.2. External Interface Requirements 6

3.3. Internal Interface Requirements 7

**4.** **Non-Functional Requirements 8**

4.1. Security and Privacy Requirements 8

4.2. Environmental Requirements 8

4.3. Performance Requirements 8

# Purpose

This document outlines the requirements for the Banking Application

## Scope

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

## Definitions, Acronyms, Abbreviations

**Client:** Person who benefits from the banking functions. Including depositing/withdrawing money, owning accounts of any amount.

**Teller:** The employee at a bank who has more permissions than a client. They can open/close accounts, edit accounts, deposit/withdraw money from a client’s account.

**ATM:** The interface where a client can perform transactions and see their account information.

**Deposit** – Adding funds to an account.

**Withdrawal** – Taking funds out of an account.

**Record** – Immutable logs of all transactions, actions performed by clients, and actions performed by employees that modify \_\_\_\_\_\_

**Overdrawing –** Attempting to withdraw more funds than what is present in the account.

**Transaction** – An action that modifies the balance of an account.

**Fraud** – Race conditions; simultaneous access; out of order transactions.

## References

Use Case Specification Document – Step 2 in assignment description

UML Use Case Diagrams Document – Step 3 in assignment description

Class Diagrams – Step 5 in assignment description

Sequence Diagrams – Step 6 in assignment description

## Overview

The Banking Application is designed to provide basic banking services across a large number of physical Bank locations. It operates on an autonomous network, and supports simultaneous server access across all locations.

# Overall Description

## Product Perspective

## Product Architecture

The system will be organized into major modules: the User Interface Module, the ATM module, the Teller Interface module, the Account Module, the Payment Module, the Record Keeping 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

Only Tellers can open, close, and edit accounts.

The application will only be accessible on devices available at the Bank locations.

Tellers have preexisting login credentials; A new user of type “Teller” cannot and will not be instantiated.

## Assumptions and Dependencies

It is assumed that Tellers will always input transactions correctly; there will never be the need to “reverse” a transaction due to accidental circumstances.

It is assumed that Clients will always input ATM transactions correctly; there will never be the need to “reverse” a transaction due to accidental circumstances.

It is assumed that the tellers are logging in from their work environment, device and location.

It is assumed that tellers will always log out after their work shifts.

It is assumed that there is more than one Bank location.

# Specific Requirements

## Functional Requirements

### Common Requirements:

### The system must prompt the user to log in as a Teller or a Client (ATM).

* + - 1. Users can access bank account information at any location of the same bank.
      2. Information is stored after the user signs out of the ATM and after tellers finish account transactions
      3. All banks operate on the same autonomous network.
      4. All information regarding transactions, user data, and teller data are stored on a centralized server

### User Interface Module Requirements:

* + - 1. User friendly- easy navigation
      2. Must be optimized for teller and client devices
      3. Must include a login page
      4. There will be a timeout after a specific time
      5. Will display all the functions for either teller or client

### ATM Module Requirements:

* + - 1. The System must automatically log out of the current open account after a period of inactivity.
      2. ATMs can be used to deposit or withdraw money and look at account balance. – And account history?
      3. There are withdrawal and deposit limits on ATM transactions.
      4. Error message when user's bank account is not found at ATM.
      5. Error when trying to do transactions on an ATM from a bank account from a different bank. – Would this be possible, considering 3.1.3.4?

### Teller Module Requirements:

* + - 1. Tellers (authorized bank employees) are the ones allowed to create or delete bank accounts.
      2. Tellers can perform the same functions as an ATM (but not vice versa).
      3. Tellers/Employees can access someone’s bank account information by using the client’s unique ID.
      4. Tellers can change client’s bank account information (i.e. Address, phone number, username, password, and name)

### Account Module Requirements:

* + - 1. Clients can have any number of saving and checking accounts
      2. Clients can share any number of banking accounts with other users
      3. Overdraft fees if the user attempts to withdraw more money than is available in their account.

### Payment Module Requirements:

* + - 1. Automatically bills the client for their credit card fees

### Record Keeping Module Requirements:

* + - 1. Logs all changes made to a client’s account by the teller
      2. Logs all transactions completed by either teller or client

-- Server module?

## External Interface Requirements

Provide module specific requirements as appropriate.

Example:

3.2.1 The system must provide an interface to the University billing system administered by the Bursar’s office so that students can be automatically billed for the courses in which they have enrolled. The interface is to be in a comma-separated text file containing the following fields: student id, course id, term id, action. Where “action” is whether the student has added or dropped the course. The file will be exported nightly and will contain new transactions only.

## Internal Interface Requirements

Provide module specific requirements as appropriate.

Example:

3.3.1 The system must process a data-feed from the grading system such that student grades are stored along with the historical student course enrolments. Data feed will be in the form of a comma-separated interface file that is exported from the grading system nightly.

3.3.2 The system must process a data-feed from the University billing system that contains new student records. The feed will be in the form of a comma-separated text file and will be exported from the billing system nightly with new student records. The fields included in the file are student name, student id, and student pin number.

# Non-Functional Requirements

## Security and Privacy Requirements

* + 1. The system will not encrypt data being transmitted over the Internet
    2. Bank employees must login with credentials when accessing the Teller interface
    3. Clients must be able to login with credentials when accessing the ATM interface

## Environmental Requirements

Example:

4.2.1 System cannot require that any software other than a web browser be installed on user computers. 

4.2.2 System must make use of the University’s existing Oracle 9i implementation for its database. 

4.2.3 System must be deployed on existing Linux-based server infrastructure.

## Performance Requirements

Example:

4.3.1 System must render all UI pages in no more than 9 seconds for dynamic pages. Static pages (HTML-only) must be rendered in less than 3 seconds.