

Momentous

omentous

78m

CPD

Software Requirements Specification – bank management sysytem

**VERSION 1.0**

APPROVAL SIGNATURES

| CONTRACTOR | | DATE |
| --- | --- | --- |
| <<Deliverable Owner>>  <<John Doe, Manager>> | <<Signature>> |  |
| <<Vendor Project Manager>> | <<Signature>> |  |
| <<Vendor Managing Director>> | <<Signature>> |  |

| **PROJECT MANAGER** | | | | |
| --- | --- | --- | --- | --- |
| Date: | 12.04.2025 | To: | **Kishore,** Project Manager | |
|  | I approve this deliverable and have no further questions or comments. | | | |
|  | I approve this deliverable conditionally, contingent on the review and approval of the following corrections (see comments). | | | |
|  | I reject this deliverable for the following reasons identified (see comments). | | | |
|  | | | | |
| **<<SIGNATURE>>** | | | | **<<DATE>>** |
| Comments | | | | |
|  | | | | |

DOCUMENT HISTORY

| **DOCUMENT APPROVAL HISTORY** | |
| --- | --- |
| Prepared By | Kishore |
| Reviewed By | Srikanth |
| Approved By |  |

| **DOCUMENT REVISION HISTORY** | | | |
| --- | --- | --- | --- |
| Date | Document Version | Revision Description | Author |
| 12/11/2024 | 1.0 | Initial Version |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTS

1. Introduction 1

1.1. Purpose of this Document 1

1.2. Scope of this Document 1

1.3. Overview 1

2. GenERAL Description 2

3. FUNCTional Requirements 4

3.1. account Management 4

3.1.1. Bank employee login criteria 4

3.1.2. Cration of account 4

3.1.3. checking amount in the account 5

3.1.4 TRANSFER MONEY TO THE ACCOUNT…………………………………………………...5

3.1.5 DEDUCTING MONEY FROM THE ACCOUNT…………………………………………..5

4. Interface Requirements 5

4.1. User Interface (UI) Requirements 5

4.1.1. General UI Layout 6

4.2. External Interface Requirements 6

4.2.1. Import/Export Interfaces 6

4.3. System Interfaces 6

4.3.1. Database/Storing Interface 6

5. Performance Requirements 6

5.1. Response time 6

5.1.1. ACCOUNT Search 6

5.1.2. Adding and Deleting CUSTOMER 7

5.1.3. Viewing CUSTOMER Details 7

5.2. Throughput 7

5.2.1. Data Import/Export 7

5.3. System Resource Utilization 7

5.3.1. CPU and Memory Usage 7

5.4. Availability and Reliability 7

5.4.1. System Uptime 7

5.4.2. Error Handling and Recovery 7

5.5. Load Testing 8

5.5.1. Stress Testing 8

5.5.2. System Failover and Redundancy 8

6. Design Constraints 8

6.1. Platform and Device Compatibility 8

6.1.1. Cross-Platform Support 8

6.1.2. Device Specifications 8

6.2. Visual Design Constraints 8

6.3. Data storage and management 8

6.4. Performance Constraints 9

6.4.1. Latency and load time 9

6.4.2. Adding and Deleting Contacts 9

6.4.3. Viewing Contact Details 9

6.4.4. Data Import/Export 9

6.4.5. CPU and Memory Usage 9

6.4.6. System Uptime 9

6.4.7. Error Handling and Recovery 9

7. Non-functional attributes 10

7.1. Security 10

7.1.1. Data Protection 10

7.1.2. Access Control 10

7.1.3. Compliance 10

7.1.4. Incident Response and Recovery 10

7.2. Reliability 10

7.2.1. Availability 10

7.2.2. Fault Tolerance 11

7.2.3. Data Integrity 11

7.3. Usability 11

7.3.1. User Experience (UX) 11

7.3.2. Accessibility 11

7.3.3. Localization 11

7.4. Maintainability 11

7.4.1. Modularity and Extensibility 11

7.4.2. Code Quality 11

7.4.3. Automated Testing and Continuous Integration 12

7.5. Scalability 12

7.5.1. Horizontal Scalability 12

# Introduction

## Purpose of this Document

The purpose of this Requirements Specification document is to clearly define and outline the functional and non-functional requirements for the development of the "Bank management system" software application. This document serves as a foundation for both the development and the validation processes, ensuring that all stakeholders, including developers, testers, and end-users, have a shared understanding of the project's objectives. By detailing the specific requirements, expectations, and constraints of the system, this document aims to minimize any ambiguity and set a clear roadmap for the design, development, and implementation of the application. It will also act as a reference point throughout the project's lifecycle to ensure that the delivered product aligns with the defined goals.

## Scope of this Document

This document encompasses all the essential details needed to develop the "Bank management system" software application. It covers the full range of functionalities, including user interface requirements, system architecture, and integration with other platforms or services. The scope also includes system performance and security requirements, data management strategies, and user accessibility features. Additionally, the document highlights the expected development timeline, resource allocation, and associated costs to deliver the product to completion. By thoroughly addressing both technical and business considerations, this document will ensure that the project stays within budget, meets deadlines, and satisfies the end-users' needs. The value this document provides to the customer includes a clear framework for understanding the project's goals, deliverables, and the expected outcomes.

## Overview

The **Bank Management System** project is a software application designed to streamline and secure essential banking operations for bank employees, enabling efficient management of customer accounts and transactions. This system provides functionalities like secure employee login, account creation, balance inquiries, money transfers, withdrawals, and account closures. The primary aim is to enhance the accuracy, security, and accessibility of banking operations, replacing traditional manual processes with an automated system.

# GenERAL Description

The **Bank Management System**  is a comprehensive software solution developed to simplify, secure, and automate essential banking operations. This system enables bank employees to perform day-to-day activities—such as creating accounts, checking balances, transferring funds, making withdrawals, and closing accounts—through a centralized and secure platform. By replacing traditional manual processes with digital workflows, the Bank management system enhances accuracy, minimizes errors, and improves efficiency, ensuring that customers' financial transactions are handled smoothly and securely.

Key features of the **Bank management system** include:

1. Employee Login :

* Secure login functionality for bank employees with role-based access control.
* Multi-factor authentication (optional) to ensure only authorized personnel can access sensitive banking data and functions.
* Logging and monitoring of login attempts for security auditing.

2. Account Creation

* Enables bank employees to create new customer accounts by entering essential customer details (e.g., name, address, ID).
* Assigns unique account numbers and securely stores account information in the database.
* Allows for updating and managing customer details as needed.

3. Check account balance

* Provides the ability to check account balances for customers in real-time.
* Allows employees to view transaction summaries and account history for better customer support.

4. Transfer the money

* Facilitates the transfer of money between accounts within the bank, ensuring secure and seamless transactions.
* Tracks and logs each transfer for auditing and accountability purposes.
* Includes transaction limits and security checks to prevent fraud.

5. Withdrawals and Deposits

* Allows employees to process customer withdrawals and deposits with real-time balance updates.
* Ensures accurate record-keeping for each transaction and flags any suspicious activities.
* Provides receipt generation for both deposits and withdrawals.

6. Account Closure

* Enables employees to close customer accounts securely, following necessary verification steps.
* Prevents further transactions on closed accounts and flags them as inactive in the database.
* Provides options for final account balance settlement.

The overall goal of the **"Bank management system"** application is to provide a reliable, efficient, and accessible solution for managing bank account. By offering essential features like easy data entry and quick retrieval, the application will cater to the diverse needs of users who require an organized, up-to-date record of their customers. Whether it’s for managing business transactions, personal transactions, or a mix of both, the Bank management system aims to be an indispensable tool for everyday communication management.

# FUNCTional Requirements

## account management

### bank employee login criteria

* The bank employee can be login entering details such as:
  + User name or Employee ID
  + Password
* The employee enters their unique username and password, which are issued by the bank’s IT or HR department.
* The system verifies that both fields are completed before proceeding to validate the credentials.

### creation of account

* The Bank employee enters essential customer information into the account creation form, including:
  + Customer full name
  + Date of birth
  + Address and Contact information
  + Email address and Account creation date

Then the bank employee selects the type of account the customer wants to open, such as:

1)Savings Account

2)Checking Account

3)Business Account

* The Bank management system automatically generates a **unique account number** for the new account.
* This account number is linked to the customer’s details and stored in the database to ensure that it remains unique and traceable.

### checking amount in the account

* The employee enters the **account number** or **customer ID** to locate the specific account.
* The system verifies that the entered account number or ID exists and is valid.
* Once the account is confirmed, the system retrieves the **current balance** from the database.
* Additional information like **available balance** (minus any holds or pending transactions) may also be retrieved.

### transfer money to the account

 The bank employee logs into the Bank management system and selects the  **Transfer** option from the main menu .

 This section is restricted to employees with the necessary permissions to perform transfers.

* The bank employee inputs the **source account number** (from which money will be deducted) and the **destination account number** (to which money will be credited).
* The system validates both account numbers to ensure they exist, are active, and are eligible for transfers.
* The Bank employee enters the amount to be transferred.
* The system checks the **available balance** in the source account to ensure that it is sufficient for the transfer, including any minimum balance requirements.

### Deducting money from the account

* The bank employee logs into the Bank management system and selects the **Deduct Money** from the main menu .
* The employee enters the **account number** from which the money will be deducted.
* The system verifies the account number to ensure it is valid, active, and eligible for deductions.

# Interface Requirements

## User Interface (UI) Requirements

### General UI Layout

* The **Bank Management System**  interface requires a secure login for employee access and a clear dashboard displaying key functions like account management, balance inquiry, and transaction handling.
* Each section should include easy navigation, validation, and real-time data updates, allowing employees to efficiently create, close, or manage accounts.
* Transaction interfaces (for transfers and deductions) need confirmation steps and receipt generation, with optional notifications to customers.
* An audit log section provides searchable history for tracking all actions and compliance. The interface should be responsive for mobile or tablet use, with role-based access to ensure data security.

## External Interface Requirements

### Import/Export Interfaces

* The Bank employee should provide an interface for importing and exporting bank account in common formats (e.g., CSV, txt).
* For importing, employee should be able to upload a file.
* For exporting, employee should be able to download a customer list in CSV or txt format, which can be opened in other applications.

## System Interfaces

### Database/Storing Interface

The Bank employee should use a backend file (e.g., txt or similar) to store customer information. The file interface should:

* Support CRUD (Create, Read, Update, Delete) operations for customer and associated data.
* Ensure that the data is securely stored and encrypted to protect user privacy.
* Maintain file integrity, preventing data loss or corruption during app usage.

# Performance Requirements

## Response time

### account Search

* The Bank employee must return search results for any query (e.g., searching by name, phone number, or email) within 1 **second** under normal conditions.
* For searches with more than **1,000 customers**, the search results must still be displayed within **5 seconds**.

### Adding and Deleting Customers

* When adding or deleting a customers, the system must reflect the change and display the updated information within **1 second** after the user submits the action.
* The bank emplyoee must immediately update the displayed customer list, ensuring no delay in the interface when users perform these actions.

### Viewing Customers Details

* Viewing the details of a customers must take no longer than **1 second** after the Bank employee searches the customer in the list.

## Throughput

### Data Import/Export

* The bank employee should be capable of importing customer from external files (CSV, txt) at a rate of 100 customer per second.
* The export process should allow users to export their entire customer list (up to 5,000 customer) in less than 30 seconds.

## System Resource Utilization

### CPU and Memory Usage

* The bank employee should operate efficiently, ensuring that CPU usage does not exceed 2**5%** during typical operations, including loading customer, performing searches, or deleting customer details.
* Memory consumption must be kept within **100 MB** for a customer list of up to **5,000** customer.

## Availability and Reliability

### System Uptime

* The system should have a **99.9% uptime** over a rolling 30-day period, ensuring the application is consistently available to users without significant downtime.
* Maintenance windows or outages for updates and patches should be scheduled during off-peak hours and should not exceed **1 hour per month**.

### Error Handling and Recovery

* The system must be resilient and able to recover gracefully from common errors (e.g., database failures).
* If the system encounters an error, it should display a clear error message to the user.
* The Bank employee should ensure that no data is lost in case of a failure during synchronization or data entry.

## Load Testing

### Stress Testing

* The Bank employee should undergo load and stress testing to verify that it can handle sudden increases in user activity.

### System Failover and Redundancy

* The Bank employee must include system failover mechanisms, such as database clustering and load balancing, to ensure that performance is not impacted by system failures or high load.

# Design Constraints

## Platform and Device Compatibility

### Cross-Platform Support

* There is no cross-platform support, the Bank employee is designed and runs only on Windows PC with Windows 10 and greater.

### Device Specifications

* The Bank employee must work on any Intel and AMD processors.

## Visual Design Constraints

* The app's design should minimize the use of high-resolution images or heavy graphics to reduce loading times and conserve system resources.
* The color palette should be simple and neutral.

## Data storage and management

* The local data storage is to be performed using windows file system with a txt file.

## Performance Constraints

### Latency and load time

* The Bank employee must return search results for any query (e.g., searching by name, phone number, ) within 1 **second** under normal conditions.
* For searches with more than **1,000 customer**, the search results must still be displayed within 2 **seconds**.

### Adding and Deleting Customer

* When adding or deleting a customer, the system must reflect the change and display the updated information within **1 second** after the user submits the action.
* The Bank employee must immediately update the displayed customer list, ensuring no delay in the interface when users perform these actions.

### Viewing Customer Details

* Viewing the details of a customer (including all fields such as name, phone numbers, ) must take no longer than **1 second** after the user searches the customer in the list.

### Data Import/Export

* The Bank employee should be capable of importing customer from external files (CSV, txt) at a rate of 100 customer per second.
* The export process should allow users to export their entire customer list (up to 5,000 customer) in less than 30 seconds.

### CPU and Memory Usage

* The Bank employee should operate efficiently, ensuring that CPU usage does not exceed 2**5%** during typical operations, including loading customer, performing searches, or deleting customer details.
* Memory consumption must be kept within **100 MB** for a customer list of up to **5,000** customer.

### System Uptime

* The system should have a **99.9% uptime** over a rolling 30-day period, ensuring the Bank employee is consistently available to customer without significant downtime.
* Maintenance windows or outages for updates and patches should be scheduled during off-peak hours and should not exceed **1 hour per month**.

### Error Handling and Recovery

* The system must be resilient and able to recover gracefully from common errors (e.g., database failures).
* If the system encounters an error, it should display a clear error message to the user.
* The Bank employee should ensure that no data is lost in case of a failure during synchronization or data entry.

# Non-functional attributes

## Security

### Data Protection

* All user data, including customer information and personal transaction, must be **encrypted**.

### Access Control

* The app should prevent unauthorized access to sensitive data.

### Compliance

* The Bank employee must comply with relevant privacy laws and regulations such as **GDPR** (General Data Protection Regulation), **CCPA** (California Consumer Privacy Act), and **HIPAA** (if applicable), ensuring user data is handled securely and users have control over their data.
* Users must be able to delete their data upon request, and they should be notified about data collection practices during initial use.

### Incident Response and Recovery

* The system must include logging and monitoring to detect potential security incidents (e.g., unauthorized access attempts or data breaches).
* In case of a security breach, the system should notify the affected users within **72 hours** and follow proper incident response protocols, including data recovery procedures.

## Reliability

### Availability

* The system must have a **99.9% uptime** for both the mobile and web applications. Scheduled maintenance should not exceed **1 hour per month**.
* The Bank employee must include automatic failover mechanisms to ensure continued operation in case of hardware or software failures.

### Fault Tolerance

* The system should be designed to recover gracefully from errors and failures, without losing user data. For example, in case of synchronization failure, the app should retry the operation automatically.
* **Backup systems** should be implemented to protect user data, and the system should allow users to restore their contacts from a backup if data loss occurs.

### Data Integrity

* The system should ensure that user data is always consistent and not corrupted, especially during synchronization or database updates.

## Usability

### User Experience (UX)

* The Bank employee should have an intuitive, easy-to-use interface, making it accessible to users with varying levels of technical expertise.
* Key actions, such as adding a new contact, searching for a contact, and viewing contact details, should require **no more than 3 steps** to perform.

### Accessibility

* The Bank employee should support keyboard navigation for desktop users.

### Localization

* The Bank employee should support **multiple languages** (e.g., English, Spanish, French, and German) to cater to users in different regions.
* The user interface (UI) should automatically adapt to the language and regional preferences of the user, including date formats, currency symbols, and number formats.

## Maintainability

### Modularity and Extensibility

* The Bank employee should be built in a modular fashion, with each feature or component of the system isolated in separate modules for easier updates, bug fixes, and scalability.
* The architecture should support extensibility, allowing for easy addition of new features without requiring major changes to the core system.

### Code Quality

* The codebase must adhere to **clean code principles**, ensuring it is well-documented, readable, and easy to understand.
* It should follow coding standards and best practices, including version control (e.g., **Git**) and regular code reviews to ensure high-quality, maintainable code.

### Automated Testing and Continuous Integration

* The system should include automated tests (unit, integration, and functional tests) to ensure that all components work correctly and that new changes do not break existing functionality.
* Continuous integration (CI) and continuous deployment (CD) pipelines should be used to automate the build, test, and deployment processes, ensuring that updates and bug fixes can be released quickly and safely.

## Scalability

### Horizontal Scalability

* The system should support horizontal scalability, meaning it can handle increasing numbers of users and contacts.
* The database must be capable of scaling to support **millions of customers** while maintaining performance.

# High Level Design

The **Bank management system** application will be implemented in **C** and designed to run natively on **Windows OS**. The application will provide a command-line interface (CLI) to allow users to perform bank management operations , as well as import/export customer in standard formats like CSV and vCard. This high-level design outlines the architecture, key components, and overall system structure for implementing the **Bank management system** in C on Windows.

## Architecture Overview

The application will follow a **Modular Architecture** with separate layers for:

* **User Interface (CLI)**: Command-line input/output handling.
* **Core Logic**: bank management and business logic.
* **Data Storage**: File-based storage for customers.
* **File I/O**: For import/export operations.
* **Utilities**: Common utility functions like input validation, search, etc.

The system will be entirely local, meaning no cloud synchronization or networking components are involved in this version of the **Bank management system**. All customer data will be stored in files (e.g., CSV files) on the local filesystem.

## Key Components

### User Interface (CLI)

The **CLI** will provide the interface through which bank employee use the application. The main interface components include:

* **Main Menu**: Display options for various operations like employee login,creation of account, check amount in the account, transferring money, deducting money, ,importing/exporting customer, and quitting the account.
* **Bank employee login** :
* Secure login functionality for bank employees with role-based access control.
* Multi-factor authentication (optional) to ensure only authorized personnel can access sensitive banking data and functions.
* Logging and monitoring of login attempts for security auditing.
* **Account creation:**
* Enables bank employees to create new customer accounts by entering essential customer details (e.g., name, address, ID).
* Assigns unique account numbers and securely stores account information in the database.
* Allows for updating and managing customer details as needed.
* **Check account balance:**
* Provides the ability to check account balances for customers in real-time.
* Allows employees to view transaction summaries and account history for better customer support.
* **Transfer the money:**
* Facilitates the transfer of money between accounts within the bank, ensuring secure and seamless transactions.
* Tracks and logs each transfer for auditing and accountability purposes.
* Includes transaction limits and security checks to prevent fraud.
* **Withdrawals and Deposits:**
* Allows employees to process customer withdrawals and deposits with real-time balance updates.
* Ensures accurate record-keeping for each transaction and flags any suspicious activities.
* Provides receipt generation for both deposits and withdrawals.
* **Account Closure:**
* Enables employees to close customer accounts securely, following necessary verification steps.
* Prevents further transactions on closed accounts and flags them as inactive in the database.
* Provides options for final account balance settlement.
* **Import/Export customer**: Ability to import customer from CSV files and export them into CSV format.

Main Menu:

1. Bank employee login
2. Account creation
3. Check account balance
4. Transfer money
5. Withdrawals and deposits
6. Account closure
7. Import/Export customer

The **CLI** will display appropriate prompts, read user input, validate commands, and show results based on the banl employee’s actions.

### Core Logic

The core logic handles the processing behind each user action. This will include managing the customer data, performing searches, and interacting with the file system for storage.

* **Employee login**: The employee enters their unique username and password, which are issued by the bank’s IT or HR department.
* **Creation of account**: The Bank employee enters essential customer information into the account creation such as customer full name,date of birth,address and contact list,email address and account creating date
* **Transferring money to the account**: The bank employee logs into the Bank management system and selects the  **Transfer** option from the main menu and this section is restricted to employees with the necessary permissions to perform transfers.
* **Deducting money from the account**: The bank employee logs into the Bank management system and selects the **Deduct Money** from the main menu .The employee enters the **account number** from which the money will be deducted.

### Data Storage

Since the bank management system will be implemented in C, customer data will be stored in local files (e.g., CSV files), which is a simple and efficient way to persist data on Windows.

* **Data File Format (CSV)**:
  + Customer will be stored in a **CSV (Comma Separated Values)** format where each customer record is a line in the file, and fields are separated by commas.
* Each record will include basic fields such as customer details,account details and transaction details
  + Example of the CSV format for customer details:
* Mahammad ibrahim, 21/09/2003, Bengaluru, 9108149720, [mahammadmahammadibrahim@gmail.com](mailto:mahammadmahammadibrahim@gmail.com)
* Example of the CSV format for account details:
* 123567899, Savings bank, 50, active

### File I/O for Import/Export

* **Import Customer**:
  + Bank employee will be able to import customer from a CSV file. The program will parse the file line by line, extract each customer details, and add them to the system.
  + The CSV file must follow the same format as the application’s internal storage format, i.e., Name, Phone Number, Email.
* **Export Customer**:
  + Bank employee can export the current list of customer to a CSV file. The bank employee will write and check the customer details to the output file in the same format used for internal storage.

These import/export operations will be implemented using file handling functions in C.

### Utilities and Helper Functions

The bank management system will use several utility functions to help with tasks like input validation, error handling, and searching:

* **Input Validation**:
  + Ensure that customer details and account details are in valid formats before storing them.
  + Ensure that required fields are not empty when adding a new customer.
* **Search Functionality**:
  + Implement a function that allows bank employee to search for customer details .
  + This could involve iterating over the loaded customer and comparing the search term with each field.
* **File Management**:
  + Utility functions will handle reading from and writing to the CSV files. These functions will be responsible for ensuring the data is correctly formatted and properly persisted.

## Data Flow

The data flow will follow these steps:

1. **Initialization**:
   * When the program starts, it will check if the customer data file exists.
   * If it exists, the program will load the customer from the file into data structures in C.
   * If the file does not exist, the application will initialize an empty customer list.
2. **User Interaction**:
   * The user interacts with the CLI, selecting options from the menu.
   * The system will process employee input, invoking the appropriate core logic function .

**4. Import/Export Operations**:

* + The bank employee can import customer from a CSV file, which will be parsed and checked the customer account in data structure.
  + The bank employee can export the customer in data structure to a CSV file.

## Technology and Tools

* **Programming Language**: C
  + C will be used for its efficiency, control over system resources, and simplicity for this small-scale, file-based application.
* **Windows OS**:
  + The program will be designed to run on a Windows OS with standard libraries for file operations, memory management, and user input handling.
* **Libraries/Tools**:
  + Standard C libraries (e.g., stdio.h, stdlib.h, string.h) will be used for basic operations like reading from files, handling user input, and string manipulation.

## Error Handling

* The application should handle errors gracefully, such as:
  + Invalid input from the user (e.g., empty fields or invalid phone numbers).
  + File errors (e.g., file not found, permissions issues).
  + Memory allocation errors (e.g., failure to allocate memory for storing contact data).

Error messages should be displayed to the user with clear instructions on how to resolve the issue.

# Low Level Design

Bank management system

Account exist

Display the customer account balance

Input 2

Employee can enter the account number

Account number will be generated

Customer details valid

Display options

1. Enter 1 for creation
2. Enter 2 for checking
3. Enter 3 for transfer
4. Enter 4 for deducting

Display main menu

1.Creation of account

2. Check Amount in the Account

3. Transfer Money to the Account

4. Deduct Money from the Account

5.Close account

Employee login

search

Employee Can enter the customer details

Input 1

Login screen

start

No

yes

Input 5

Input 4

Input 3

Employee can enter the account number

Employee can enter the source and target account

No

No

No

No

No

valid

Account exist

yes

yes

Check bank balance of source account

yes

yes

Check balance of acc

yes

Employee can click close button

Bank employee can enters the amount to transfer the money

yes

Deduct money from account

End