

creation

CPD

Software Requirements Specification –

bank management system AppLication

**VERSION 1.0**

APPROVAL SIGNATURES

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

| **PROJECT MANAGER** | | | | |
| --- | --- | --- | --- | --- |
| Date: | 13.11.2024 | To: | **Manoj k r,** 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 |  |
| Reviewed By |  |
| Approved By |  |

| **DOCUMENT REVISION HISTORY** | | | |
| --- | --- | --- | --- |
| Date | Document Version | Revision Description | Author |
| 13/11/2024 | 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

2.1. PLANNING 3

2.2. FEASABILITY STUDY 3

2.3. Design 3

2.4. Development (Implementation) 4

2.5. References 4

3. ROLES AND RESPONSIBILITIES 4

3.1. Project Executive Steering Committee 5

3.1.1. ROLE 5

3.1.2. RESPONSIBILITY 5

3.2. Project Sponsors 5

3.2.1. Project Executive Sponsor 5

3.2.1.1 ROLE 5

3.2.1.2 RESPONSIBILITY 5

3.2.2. Project Business Sponsor 5

3.2.2.1 ROLE 5

3.2.2.2 RESPONSIBILITY 5

3.3. Project Director 5

3.3.1. ROLE 5

3.3.2. RESPONSIBILITY 6

3.4. Project Manager 6

3.4.1. State Project Manager 6

3.4.1.1 ROLE 6

3.4.1.2 RESPONSIBILITY 6

3.4.2. Vendor Project Manager 6

3.4.2.1 ROLE 6

3.4.2.2 RESPONSIBILITY 6

3.5. Requirements Managers 6

3.5.1. State Requirements Manager 6

3.5.1.1 ROLE 6

3.5.1.2 RESPONSIBILITY 7

3.5.2. Vendor Requirements Manager 7

3.5.2.1 ROLE 7

3.5.2.2 RESPONSIBILITY 7

4. REQUIREMENTS MANAGEMENT PROCESS 7

4.1. The Process 8

5. Establish Controlled Requirements Repository 10

5.1. Requirements Repository 11

5.2. Create Requirements Baseline 11

5.2.1. Load Requirements Baseline 12

5.2.2. Identify and Tag Requirement 12

5.2.3. Set Requirement Attributes 12

6. Perform Requirements Traceability 12

7. Managing Requirements Change 13

7.1. Requirements Change Impact Assessment 13

7.2. Updating Baseline Requirements 18

7.2.1. Modification Update 18

7.2.2. Addition Update 19

7.2.3. Deletion Update 19

7.3. Reviewing Traceability 19

8. Requirement Metrics and Status Reporting 20

APPENDIX A. Overview of Requirements Traceability A-1

APPENDIX B. Example Requirements Tagging, Attributes, and Relationships B-1

APPENDIX C. Traceability Metrics and Reporting C-1

APPENDIX D. COTS, MOTS, AND CUSTOM CONSIDERATIONS D-1

# 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. It will also act as a reference point throughout the project's lifecycle to ensure that the delivered product aligns with the defined goals. This document defines the functional and non-functional requirements for developing a Bank Management System. The system will serve bank employees, facilitating account management, transaction processing, and secure customer data handling.

## 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 value this document provides to the customer includes a clear framework for understanding the project's goals, deliverables, and the expected outcomes. The document outlines requirements for essential features, such as account creation, transactions (money transfer, balance checking, deduction), and account closure. It also addresses security, performance, and regulatory compliance requirements.

## Overview

The "Bank Management System" software application is a user-friendly platform designed to help individuals store, manage, and organize their contacts efficiently. Users can add, delete, and search for contacts. The Bank Management System is designed as a secure, user-friendly application enabling employees to manage customer accounts, perform transactions, and ensure customer data security. The overall goal of the "Bank Management system" application is to provide a reliable, seamless solution for managing personal or professional contact information in an organized and efficient manner.

# GenERAL Description

The **"Bank Management System"** application is a comprehensive yet simple-to-use software solution designed to help users efficiently store, manage, and organize their contact information. Whether for personal or professional use, the application provides a central location where individuals can maintain detailed records of accounts, including names, phone numbers and email addresses.

The primary functions of the application include adding, searching and deleting accounts. With a focus on user-friendliness, the application ensures that individuals of all technical skill levels can manage their account without encountering steep learning curves or complex procedures.

Key features of the **Bank Management System** include:

* **Account Management**: Users can easily create new Account entries, search existing information, and remove outdated or unnecessary accounts. Each account record can store multiple fields of information such as name, phone number and email address.
* **Search and Filter**: A powerful search function enables users to quickly locate specific accounts by entering account number, names, or other attributes.
* **User Interface (UI)**: The application will feature an intuitive, clean, and command prompt interface that simplifies navigation. The design will prioritize ease of use, ensuring that both tech-savvy and non-technical users can efficiently perform tasks with minimal effort.
* **Lightweight and Responsive**: Designed to be fast and responsive, the application will provide a smooth user experience, even when managing a large number of accounts. The software will be optimized for performance, ensuring that users can access and modify their data quickly and with minimal delay.

The overall goal of the **"Bank Management System"** application is to provide a reliable, efficient, and accessible solution for managing contact information. 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 personal or professional accounts. Whether it’s for managing business accounts, personal relationships, or a mix of both, the Bank Management System aims to be an indispensable tool for everyday Bank management.

# FUNCTional Requirements

## Bank Management System

* The application must allow users to create new account by entering details such as:
  + Account number
  + Name
  + Phone Number
  + Email Address
* The system should validate the entered data for required fields ( account number ,name, phone number & email) and ensure data formats are correct (e.g., email format, phone number format).
* The validated data should be saved into a file on windows file system.

### ACCOUNT Search

* The application must allow users to search for accounts by entering keywords, including:
  + Account number
  + Costumer’s name (partial or full)
  + Phone number
  + Email address
* The system should read the file where the data is saved and retrieve the matching accounts.
* The search results should display all matching accounts in real-time with serial numbers.
* The searched results should give an option to delete account pointing the serial number - “Enter serial number/numbers to delete the accounts or 0 to exit search to main menu”

### closing an account

* The system must allow users to delete account from their list after searching.
* To delete the prompt should ask the user to select the searched serial number and confirm deletion.
* Deleting an account must prompt the user for a confirmation to prevent accidental deletion.
* The deletion action should be saved to the file on windows file system.

### Data Import/Export

#### Importing accounts

* The application should allow users to import accounts from other file formats (e.g., CSV, txt).
* The system must ensure the correct mapping of imported fields (application number, name, phone number, email.) to the contact data structure.

#### Exporting accounts

* Users must be able to export their accounts to a variety of formats (e.g., CSV, txt) for backup, sharing, or transferring to another system.

# Interface Requirements

## User Interface (UI) Requirements

### General UI Layout

* The application should have a clean, simple, and intuitive layout with easy navigation.
* The main screen should display the account list, allowing users to perform actions such as adding, searching and deleting accounts.
* The interface should be designed to minimize the number of steps required to perform common tasks.

## External Interface Requirements

### Import/Export Interfaces

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

## System Interfaces

### Database/Storing Interface

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

* Support CRUD (Create, Read, Update, Delete) operations for account 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 application must return search results for any query (e.g., searching by account number, name, phone number, or email) within 1 **second** under normal conditions.
* For searches with more than **1,000 accounts**, the search results must still be displayed within **5 seconds**.

### Adding and Deleting accounts

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

### Viewing account Details

* Viewing the details of an account (including all fields such as account number, name, phone numbers, and emails) must take no longer than **1 second** after the user searches the contact in the list.

## Throughput

### Data Import/Export

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

## System Resource Utilization

### CPU and Memory Usage

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

## 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 application should ensure that no data is lost in case of a failure during synchronization or data entry.

## Load Testing

### Stress Testing

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

### System Failover and Redundancy

* The application 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 application is designed and runs only on Windows PC with Windows 10 and greater.

### Device Specifications

* The application 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 colour 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 application 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 contacts**, the search results must still be displayed within 2 **seconds**.

### Adding and Deleting Contacts

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

### Viewing Contact Details

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

### Data Import/Export

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

### CPU and Memory Usage

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

### 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 application 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 contact information and personal data, must be **encrypted**.

### Access Control

* The app should prevent unauthorized access to sensitive data.

### Compliance

* The application 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 application 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 application 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 application should support keyboard navigation for desktop users.

### Localization

* The application 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 application 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 users and contacts** while maintaining performance.