CPD

Banking 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: | 02.04.2025 | To: | **ABC,** 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 |
| 12/11/2024 | 1.0 | Initial Version |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTS

1. Introduction 1

1.1. Purpose of this Document 1

1.2 PLANNING…………………………………………………………………………1-2

2. GenERAL Description 3

3. Requirements  ****Analysis**** 4

### ****3.1 Functional Requirements****

### ****3.2 Non-Functional Requirements****

4. system design 5

4.1 Architecture

4.2 Database Schema

4.3 Use Case Diagram

5. Development plan 6

5.1 Risks and Mitigation

6. Non-functional attributes 7

7. flow chart 8

# Introduction

The banking industry has evolved significantly with **digital transformation**, requiring secure, efficient, and scalable software solutions. This seminar focuses on the **development of a banking application**, following the **Software Development Life Cycle (SDLC)** stages.

* **Requirement Analysis** – Understanding user needs like account management, transactions, and security.
* **System Design** – Structuring the database, defining workflows, and designing architecture.
* **Development & Implementation** – Writing code for user authentication, transactions, and admin controls.
* **Testing & Debugging** – Ensuring secure and error-free operations.
* **Deployment & Maintenance** – Launching and continuously improving the application.

**1****.1 Purpose of This Document**

This document provides a C-based implementation of a Banking Software System that allows users to manage their bank accounts securely. The purpose of this project is to:

1. Enable Users to:
   * Register and log in securely.
   * Check account details, including balance.
   * Transfer money between accounts.
2. Enable Admin to:
   * Create and delete user accounts.
   * Manage funds (credit/debit).
   * Reset user passwords.
3. Implement Banking Features Securely:
   * User authentication (username & password).
   * Basic transaction handling (fund transfers).
   * Admin-controlled account management.

## ****Planning****

### ****1.2.1 Objective****

To develop a secure and efficient Bank Management System that allows employees to manage customer accounts, check balances, transfer funds, and close accounts.

### ****Key Considerations****

* Security & Compliance (Authentication, Authorization)
* Performance & Scalability
* User Experience
* Data Integrity

### ****Stakeholders****

* Bank Managers
* IT Team (Developers, QA, DevOps)
* End-users (Bank Employees)

**2. General Description**

The Banking Application is a C-based program designed to provide basic banking functionalities for both users and administrators. The system allows users to create accounts, log in securely, check account details, and transfer money between accounts. An admin has the ability to create or delete user accounts, credit or debit balances, and reset user passwords. The program follows a menu-driven approach, enabling easy navigation for users and administrators.

The project is structured with secure authentication mechanisms, ensuring only registered users can access their accounts. It uses arrays to store user data, with unique account numbers assigned to each user. The system follows a modular programming approach, making it easy to extend and integrate with additional features such as file handling for data persistence, encryption for security, and database connectivity for scalability.

## ****3. Requirement Analysis****

### ****3.1 Functional Requirements****

1. **Bank Employee Login** – Employees must log in using unique credentials.
2. **Account Creation** – Ability to create new customer accounts with basic details.
3. **Check Account Balance** – View current account balance.
4. **Money Transfer** – Transfer funds from one account to another.
5. **Deduct Money** – Withdraw or debit money from an account.
6. **Close Account** – Permanently delete an account after verification.

### ****Non-Functional Requirements****

* The system should have a response time of < 3 seconds.
* Multi-factor authentication (MFA) for security.
* Support up to 10,000 concurrent users.

**4. System Design**

**4.1 Architecture:**

* **Frontend:** C-based CLI (Command Line Interface).
* **Backend:** File-based storage (or database if extended).
* **Data Storage:** Structured using **C structs and file handling**.

**4.2 Database Schema:**

| **Table** | **Fields** |
| --- | --- |
| Users | ID, Username, Password, Mobile, Email, Account Number, Balance |
| Transactions | Transaction ID, Sender ID, Receiver ID, Amount, Date |

**4.3 Use Case Diagram:**

* **User**: Login → View Account → Transfer Money
* **Admin**: Login → Create User → Manage Accounts → Logout

**5. Development Plan**

| **Stage** | **Tasks** | **Timeline** |
| --- | --- | --- |
| **Requirement Analysis** | Gather user/admin needs, define scope | **Day 1** |
| **System Design** | Design database schema, flowcharts | **Day 2** |
| **Development** | Implement login, transactions, admin panel | **Day 3-4** |
| **Testing** | Test cases, bug fixes | **Day 5** |
| **Deployment** | Host and optimize code | **Day 6** |
| **Documentation** | Create user/admin guides | **Day 7** |

**5.1 Risks and Mitigation**

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| Security issues | Implement password encryption |
| Data loss | Regular backups for file/database |
| Performance lag | Optimize data structures and transactions |

**6. Non-Functional Requirements:**

* **Security:** Password encryption, authentication validation.
* **Performance:** Fast execution of banking transactions.
* **Scalability:** Handle multiple users and transactions.
* **Usability:** User-friendly interface with clear instructions.

**7. Flow chart**

