Bank Management System in C++

Title:

Design and Implementation of a Console-Based Bank Management System Using Object-Oriented Programming in C++

Abstract

This project presents the development of a **console-based Bank Management System (BMS)** using the **C++ programming language**. The system is built upon fundamental principles of **Object-Oriented Programming (OOP)** such as encapsulation, abstraction, and modularity. The application manages **customer and admin interactions**, including secure login, account registration, deposits, withdrawals, account statement generation, and transaction history tracking. Data persistence is achieved through **file handling**, ensuring that customer and transaction data is retained between sessions. The goal of this project is to simulate real-world banking operations in a simple, educational environment for students and beginner developers.

Chapter 1: Introduction

1.1 Project Motivation

The banking sector is increasingly reliant on digital systems to handle customer records, transactions, and reporting. To understand how such systems operate at a fundamental level, this project simulates a simplified version of a banking application using **C++**, designed with a modular and object-oriented approach.

1.2 Project Objective

The primary objective of this project is to:

- Build a functional, role-based bank system with C++
- Apply key OOP concepts
- Use text files to simulate database storage
- Provide separate user experiences for **Admin** and **Customers**

Chapter 2: System Requirements

2.1 Software Requirements

- OS: Windows/Linux (Console-based)
- Language: C++
- Compiler: g++, MinGW, or any standard C++ compiler
- IDE (Optional): Visual Studio Code, Code::Blocks

2.2 Hardware Requirements

- RAM: 2GB or more
- Processor: Intel Pentium or above
- Storage: 10MB for source files and data

Chapter 3: System Design

3.1 Architecture Overview

The system is divided into several layers:

- **UI Layer** (main.cpp): Provides menus and collects input
- **Logic Layer** (Admin & Customer modules): Handles business logic
- **Model Layer** (Account & Transaction): Stores user data structures
- **Persistence Layer** (FileManager): Handles file I/O operations
- **Validation Layer**: Ensures correct user input

Chapter 4: Modules Description

4.1 main.cpp

- Acts as the starting point of the program
- Displays the main menu (Admin / Customer)
- Navigates to respective dashboards

4.2 Admin Module

- Login with username/password
- View all active accounts
- Edit or delete customer records
- Search accounts
- Show professional account statements
- View transaction logs
- Change admin password

4.3 Customer Module

- Register account (must be 18+)
- Login with account number and 4-digit PIN
- Deposit or withdraw funds
- View balance and transaction history
- Change name, phone, PIN, or password
- Delete their own account
- Show full account statement

4.4 Models

4.4.1 Account

- Contains fields like name, CNIC, DOB, phone, balance, PIN, etc.
- Used by both admin and customers

4.4.2 Transaction

- Tracks each deposit, withdrawal, account update or deletion
- Stored in `transactions.txt`

4.5 Utils

- `Validation.h`: Ensures input formatting and age restriction
- `FileManager.h`: Manages reading/writing data from/to `.txt` files

Chapter 5: Data Files

5.1 accounts.txt

Stores account details in comma-separated format:

```
1001,Ali Khan,12345-6789012-3,03001234567,2000-01-01,Savings,myPass,1234,10000,1
```

5.2 transactions.txt

Logs all transactions with timestamps:

```
1001,2025-06-16,Deposit,5000,15000
1001,2025-06-17,Withdraw,3000,12000
```

5.3 logins.txt *(optional)*

May be used in future for multiple admin/customer logins.

Chapter 6: Output Screenshots (Text Simulation)

```
====== Bank Management System ======
1. Admin Panel
2. Customer Panel
0. Exit
```

Customer Menu:

```
--- Customer Dashboard ---
1. View Balance
```

```
2. Deposit Money
3. Withdraw Money
4. Transaction History
...
```

Admin Menu:

```
--- Admin Panel ---
1. View All Accounts
2. Search Account
3. Delete Account
...
```

Account Statement (Professional Look):

```
=========== ACCOUNT STATEMENT ===========
Account #: 1001
Name     : Ali Khan
Phone    : 03001234567
Type     : Savings
Balance  : Rs. 12000.00

------ TRANSACTION HISTORY ------
Date        Type       Amount    Balance
2025-06-16  Deposit     5000      15000
2025-06-17  Withdraw    3000      12000
=========================================
```

Chapter 7: OOP Concepts Applied

Concept	Application in Project
Encapsulation	Account data secured with private members and methods
Inheritance	Shared model used in Admin/Customer modules
Abstraction	Simplified user interface with clear roles
Modularity	Code divided across folders for clarity
File Handling	All data stored in `txt` files for persistence

Chapter 8: Conclusion

The Bank Management System successfully demonstrates the application of **C++ OOP concepts** in a real-world scenario. It provides two distinct user roles with proper access control and supports operations such as

deposits, withdrawals, data editing, and transaction tracking. The system is **easily extendable** to add GUI or database support in future versions.

Chapter 9: Future Enhancements

- Integrate with a database (MySQL, SQLite)
- Add GUI with Qt or WinForms
- Implement real-time date and time stamps
- Add email/SMS notification simulation
- Encrypt PIN/passwords for stronger security

Chapter 10: Author & Acknowledgements

Author: Muhammad Ikram **Field:** Software Engineering **Institution:** University of Swat **Email:** (add your email if required) **GitHub:** https://github.com/ikram-3

> Special thanks to my teachers, classmates .

Appendices

File Structure Tree

```
BankManagementSystem/
├── main.cpp
├── README.md
├── Admin/
│   ├── Admin.h
│   └── Admin.cpp
├── Customer/
│   ├── Customer.h
│   └── Customer.cpp
├── Models/
│   ├── Account.h
│   ├── Account.cpp
│   └── Transaction.h
├── Utils/
│   ├── FileManager.h
│   └── Validation.h
└── data/
    ├── accounts.txt
    ├── transactions.txt
    └── logins.txt
```