A REPORT

ON

BANK MANAGEMENT SYSTEM

(TASK 1)

Submitted By:
Sakshi Mittal

INTERN AT VERVEBRIDGE
POSITION- C++ Developer

BANK MANAGEMENT SYSTEM

INTRODUCTION

This report covers the design and functionality of a Bank Management System implemented in C++. The program provides basic banking operations such as account creation, authentication, deposit and withdrawal of money, balance inquiry, money transfers, and transaction history review. Additionally, the system offers a user-friendly interface to perform these actions and ensures data persistence using a structure to store account information.

SYSTEM DESIGN OVERVIEW

The Bank Management System is structured into several distinct functions, each responsible for a specific part of the functionality. The design follows a menu-driven approach, with a series of nested menus guiding users through available operations.

Key Components:

1. ACCOUNT STRUCTURE
   - Contains account number, account holder's name, password, current balance, and a transaction history.
   - The structure Account is typedef'ed as ''acc'' for simpler usage.

Main Features:

1. Main Menu

- Displays options to create an account, log in, view customer information, and exit.

2. Bank Menu

- Allows logged-in users to deposit, withdraw, inquire about their balance, view transaction history, transfer money, or log out.

3. Authentication

- Ensures that users can log in only with a valid account number and password.
- Users are allowed to change their password if forgotten, after confirming their identity.

Functional Details:

1. Account Creation

When a new account is created, the system

- Automatically assigns an account number.

- Prompts the user for their name and password.
- Initializes the balance to 0 and stores the account information in a vector of structures.

2. Authentication

Upon selecting the login option:

- The user is asked for their account number and password
- If authenticated, they can access the bank menu to perform various transactions.
- If the password is incorrect, the user is given an option to change the password.

3. Deposit

In the deposit functionality:

- The user enters the amount to deposit.
- The system ensures that the deposit amount is positive.
- The balance is updated, and a transaction record is added to the account's history.

4. Withdraw

For withdrawing money:

- The user inputs the amount to withdraw.
- The system checks if the account balance is sufficient for the withdrawal.
- If the balance is sufficient, it deducts the amount and updates the account's transaction history.

5. Balance Inquiry

When the user selects this option:

- The account balance, along with account number and holder's name, is displayed.

6. Account Statement

- Displays a detailed statement of the account, including all transactions recorded in the history.

7. Transfer Money

- Allows transferring funds from one account to another.
- The sender's balance is reduced, and the receiver's balance is increased by the transfer amount, with both accounts recording the transaction.

ERROR HANDLING & EDGE CASES

The system handles common user errors and edge cases:

- **Authentication Errors:** If the account number is not found, or the password is incorrect, the system provides appropriate error messages and options.
- **Negative and Zero Deposits/Withdrawals:** The system checks for valid input when depositing or withdrawing money.
- **Insufficient Funds:** When withdrawing or transferring funds, the system ensures that the account has sufficient balance before proceeding with the transaction.

CODE ANALYSIS

The code is divided into small, modular functions which improve readability and reusability. Each function has a single responsibility, and the separation of concerns is well-maintained.

- **Vectors:** A dynamic array (vector) is used to store account details, allowing the system to handle a flexible number of accounts.
- **Transaction History:** The transaction history is stored as a vector of strings within each account structure, making it easy to maintain records of all transactions.

CONCLUSION

This Bank Management System effectively handles a wide range of banking operations with a clean, menu-driven interface. It is a useful tool for small-scale simulations of banking processes. By adding features like file storage and advanced error handling, the system could be further enhanced to serve more complex needs.