**PF-Project Proposal**

**Name: Ali Sufyyan**

**Roll #: L1F23BSDS0003**

## Project Report: Bank Management System

### Table of Contents

1. Introduction
2. Objective
3. Project Scope
4. System Design
5. Implementation
6. Code Explanation
7. Testing and Validation
8. Conclusion

### 1. Introduction

The Bank Management System is a console-based application developed in C++ designed to handle the fundamental operations of a bank. This includes user registration, login, account management, deposit, withdrawal, and balance inquiry. The system is intended to streamline the basic banking processes and maintain secure and efficient handling of user data.

### 2. Objective

The primary objective of this project is to create a simple, user-friendly, and secure banking system. It aims to:

* Provide a secure login mechanism for users.
* Allow users to register and create new accounts.
* Enable users to deposit and withdraw money.
* Provide an interface for admins to view all account holders.
* Implement a mechanism for deleting user accounts.

### 3. Project Scope

This project focuses on implementing core banking functionalities and providing a secure environment for financial transactions. The scope includes:

* User registration and login.
* Admin functionalities.
* Basic operations like deposit, withdrawal, and balance inquiry.
* Persistent storage using text files for account and password information.

### 4. System Design

#### Modules:

1. **Main Module**: Handles the main menu and controls the flow of the program.
2. **User Authentication Module**: Manages user registration, login, and validation.
3. **Account Management Module**: Manages account operations like deposit, withdrawal, and balance inquiry.
4. **Admin Module**: Allows admin users to view all account holders and delete accounts.

#### Data Storage:

* **data.txt**: Stores account holder information including name, username, and balance.
* **password.txt**: Stores user IDs and their corresponding passwords.

### 5. Implementation

**Header Files and Libraries**

The system utilizes several C++ standard libraries:

* iostream for input and output operations.
* fstream for file handling.
* string for string manipulations.
* limits for handling numeric limits, particularly useful when clearing input buffers.

**Constants and Global Variables**

A constant col is defined to represent the number of columns in the password array, which stores usernames and their corresponding passwords.

**Function Definitions**

1. **countlines**:
   * This function counts the number of lines in the data file. It reads the file line by line and increments a counter for each line. This count is used to determine the number of account holders.
2. **readfromfile**:
   * This function reads account holder details from the data file into arrays. It reads the holder names, usernames, and balances from the file and stores them in the respective arrays.
3. **readpasswords**:
   * This function reads usernames and passwords from the password file into a 2D array. This array is used for validating user credentials during login.
4. **showmenu**:
   * This function displays the main menu and ensures that the user makes a valid choice. It repeatedly prompts the user until a valid option (1-4) is entered.
5. **loginmanu**:
   * Handles the login process. It prompts the user for their username and password, validates the credentials against the stored data, and grants access to account operations if the credentials match.
6. **isValidName**:
   * Validates if a given name contains only alphabets and spaces. It returns true if the name is valid, otherwise false.
7. **isValidId**:
   * Validates if a given ID (username or password) contains only digits. It returns true if the ID is valid, otherwise false.
8. **Registration**:
   * Facilitates the registration of new users. It collects user information, validates it, assigns a new user ID, and saves the data into the files.
9. **grow\_arr** and **grow\_arr2**:
   * These functions are used to dynamically increase the size of the arrays holding user data. They create new arrays with increased size, copy existing data into them, and return the new arrays.
10. **sendtofile**:
    * Writes the updated account holder data into the data file. It ensures that all user details are saved for future sessions.
11. **sendtopasswords**:
    * Writes the updated passwords into the password file. This keeps the user credentials up-to-date.
12. **deposit**:
    * Handles the deposit operation. It validates the deposit amount, prompts for admin approval, and updates the user’s balance if approved.
13. **withdrawl**:
    * Handles the withdrawal operation. It validates the withdrawal amount, ensures it does not exceed the available balance, prompts for admin approval, and updates the user’s balance if approved.
14. **showholders**:
    * Displays all account holder details. This function is restricted to admin users and requires admin credentials to access.

### 6. Code Explanation

#### Main Function

The main() function initializes the program by reading existing account and password data from files, displays the main menu, and handles user choices for login, registration, viewing account holders, and exiting the program.

#### User Authentication

The user authentication process involves validating user IDs and passwords during login and registration. Usernames and passwords are stored in separate text files for security.

#### Account Management

The account management module includes functions for depositing and withdrawing money, checking balance status, and deleting accounts. Admin approval is required for deposit and withdrawal operations.

### 7. Testing and Validation

The system was tested for various scenarios including:

* Successful and unsuccessful login attempts.
* Correct and incorrect input validation during registration.
* Deposit and withdrawal operations with both valid and invalid inputs.
* Admin functionalities such as viewing account holders and deleting accounts.

### 8. Conclusion

The Bank Management System provides a comprehensive solution for managing basic banking operations. It ensures data persistence through file handling, supports secure user authentication, and includes admin functionalities for overseeing all accounts. The system is designed to be user-friendly, secure, and efficient, making it a reliable tool for banking management.

**Appendix**

* **Data Files**:
  + data.txt: Stores account holder names, usernames, and balances.
  + password.txt: Stores usernames and corresponding passwords.
* **Admin Credentials**:
  + Username: 968627
  + Password: 968627

This concludes the project report for the Bank Management System. The system's design and functionality have been outlined, along with detailed explanations of each component and function. This report serves as a comprehensive guide to understanding and using the BMS.