**OBJECT ORIENTED PROGRAMMING**

**PROJECT REPORT**

## https://lh7-us.googleusercontent.com/KJsp5Ceua-DErzIlweN5hOL3s_AZ2exVIMtX2TBfHG2VaRZDt7Mvv6zqiSjZSP4CDr6NMeamN6TtU32krC4Qt3vQCBKIDz6s7dkLOEZVAWKoczkG2iINcCaY3RSuuDhDGR9_0OnJmFXYAHxl9ZNSHw

## *Submitted To*

*Mr. Nadeem Yousaf*

***Submitted By***

***Group Members:***

* *Aiman Rehan (230201084)*
* *Zimal Babar (230201074)*

**Department of Computer Science,**

Institute Of Space Technology, Islamabad.

DATE [4th June 2024]

**PROJECT TITLE: BANK MANAGEMENT SYSTEM**

**1. Introduction**

The banking system project simulates basic banking operations, enabling users to create accounts, deposit and withdraw money, transfer funds between accounts, and view account details and transaction history. This system is implemented using C++ and object-oriented programming principles.

**2. System Overview**

The system consists of several classes that represent different types of accounts and their associated operations. It uses text files to store and retrieve account and transaction data, ensuring data persistence between sessions.

**3. Class Structure**

**Base Class: Account**

The Account class serves as the base class for different types of accounts. It contains attributes like account number, account holder name, PIN, and balance. The class provides methods for:

* Depositing money
* Withdrawing money
* Viewing account details
* Saving account details to a file
* Retrieving account details from a file

**Derived Classes**

**1. SavingsAccount**

This class inherits from the Account class and includes an interest rate. It provides an additional method to add interest to the balance.

**2. StudentAccount**

This class inherits from the Account class and includes an overdraft limit. It overrides the withdrawal method to account for the overdraft facility.

**3. ChildSavingsAccount**

This class inherits from the Account class and requires a guardian's name. It overrides the withdrawal method to restrict withdrawals.

**Friend Class**

**Transactions Class**

This class handles the operations related to transactions between accounts, such as transferring money, viewing transaction history, and generating account statements.

**4. Features**

**Account Management**

* Create Account: Users can create different types of accounts (Current, Savings, Student, and Child Savings).
* View Account Details: Users can view their account details, including the balance and account holder information.
* Delete Account: Users can delete their accounts after verification.

**Banking Operations**

* Deposit Money: Users can deposit money into their accounts after entering their PIN.
* Withdraw Money: Users can withdraw money from their accounts, with restrictions based on the account type.
* Transfer Money: Users can transfer money between accounts securely.

**Transaction Management**

* View Transaction History: Users can view the history of their deposits, withdrawals, and transfers.
* Generate Account Statement: Users can generate a detailed account statement, including all transactions.

**5. Implementation Details**

**Account Class**

The Account class includes methods to deposit and withdraw money, view account details, and save and retrieve account information from a file. It uses file I/O operations to maintain account data.

**Transactions Class**

The Transactions class includes methods to handle transactions between accounts, view transaction history, and generate account statements. It logs transaction details in a file to maintain a record of all transactions.

**Derived Classes**

The derived classes (SavingsAccount, StudentAccount, and ChildSavingsAccount) extend the functionality of the Account class to cater to specific account types. They override methods as needed to provide specialized behavior.

**File Management**

The system uses text files (account.txt, deposit.txt, withdrawal.txt, transaction.txt) to store account details and transaction history. This ensures data persistence and allows the system to retrieve and update information as needed.

**6. Drawbacks**

**Limited Error Handling:**

The project may lack comprehensive error handling, such as checking for invalid inputs (e.g., negative amounts for deposits/withdrawals) and handling exceptions (e.g., insufficient funds).

**User Interface:**

The project might not have a user-friendly interface. While the CLI (command-line interface) is useful for development and testing, a real-world application would benefit from a graphical user interface (GUI) or a web-based interface.

**8. Conclusion**

The banking system project successfully implements a basic banking application with essential features. It demonstrates the use of object-oriented programming principles and file handling in C++. The system ensures secure access to accounts and maintains a persistent record of transactions.