BANKING SYSTEM

Title of the Project

GROUP MEMBERS

> Gunavanth Reddy

AP23110010505

> Balaji Bhavani Shankar

AP23110010546

TABLE OF CONTENTS

INTRODUCTION

A Banking System is a digital platform that enables Bank User to create account , Deposit ,Withdraw,Balance Enquiry,Transfer efficiently. It serves as a bridge between Bank service providers and Bank User.

Key Features:

- **User Accounts:** Any one can Securely create account in bank.
- **Deposit and Withdraw:** Users can easily Deposit and withdraw their money through this banking system service.
- **Balance Enquiry :** Also Users can easily check the Account Balance.
- **Transfer :** You can Transfer your money to another bank account.

OBJECTIVES

- **User Account Management**:
- Allow users to create personal accounts with a unique username .
- **Deposit :**
- Enable user to easily deposit their money in their account
- **Withdraw :**
- It Enables user to withdraw their money form the account.
- **Balance Enquiry** :
- It allows user to check their bank balance that in their account.
- **Transfer** :
- Any one can easily transfer their money from account to another account by giving the correct information to the system..

OUTCOMES

- **User Account Creation**:
- Users will successfully create accounts with unique usernames.
- The system will prevent duplicate usernames.
- **Deposit** :
- The system will accurately Deposit the money, by providing users with clear pricing information.
- **Withdraw:**
- The system will accurately withdraw the money, by providing users with clear pricing information.
- **Bank Enquiry** :
- The system will show the bank balance of the specific user’s bank account balance.
- **Transfer:**
- The system will Transfer the amount from one account to another account.
- **Exit:**
- The system will stop the program of the banking system service.

CODE:

```
#include <iostream>
#include <vector>
#include <limits>

using namespace std;

class Account {
private:
   int accountNumber;
   string accountHolderName;
   double balance;

public:
   Account(int accNum, string name) : accountNumber(accNum), accountHolderName(name), balance(0.0) {}

   int getAccountNumber() const {
      return accountNumber;
   }

   string getAccountHolderName() const {
      return accountHolderName;
   }
```

```
double getBalance() const {
    return balance;
}

void deposit(double amount) {
    if (amount > 0) {
        balance += amount;
        cout << "Deposited: $" << amount << endl;
    } else {
        cout << "Invalid deposit amount!" << endl;
    }
}

void withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
        balance -= amount;
        cout << "Withdrawn: $" << amount << endl;
    } else {
        cout << "Invalid withdrawal amount!" << endl;
    }
}
```

```
void transfer(Account &receiver, double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            receiver.deposit(amount);
            cout << "Transferred: $" << amount << " to account " << receiver.getAccountNumber() << endl;
        } else {
            cout << "Invalid transfer amount!" << endl;
        }
    }
};

class BankingSystem {
private:
    vector<Account> accounts;

public:
    void createAccount() {
        int accNum = accounts.size() + 1; // Auto-increment account number
        string name;
        cout << "Enter account holder name: ";
        cin.ignore(); // Clear the newline character in the input buffer
        getline(cin, name);
        accounts.emplace_back(accNum, name);
        cout << "Account created successfully! Account Number: " << accNum << endl;
    }
```

```
void deposit() {
    int accNum;
    double amount;
    cout << "Enter account number: ";
    cin >> accNum;
    cout << "Enter amount to deposit: ";
    cin >> amount;

    if (accNum > 0 && accNum <= accounts.size()) {
        accounts[accNum - 1].deposit(amount);
    } else {
        cout << "Invalid account number!" << endl;
    }
}

void withdraw() {
    int accNum;
    double amount;
    cout << "Enter account number: ";
    cin >> accNum;
    cout << "Enter amount to withdraw: ";
    cin >> amount;

    if (accNum > 0 && accNum <= accounts.size()) {
        accounts[accNum - 1].withdraw(amount);
    } else {
        cout << "Invalid account number!" << endl;
    }
}
```

```
void balanceEnquiry() {
    int accNum;
    cout << "Enter account number: ";
    cin >> accNum;

    if (accNum > 0 && accNum <= accounts.size()) {
        cout << "Balance for account " << accNum << ": $" << accounts[accNum - 1].getBalance() << endl;
    } else {
        cout << "Invalid account number!" << endl;
    }
}

void transfer() {
    int senderAccNum, receiverAccNum;
    double amount ;
    cout << "Enter sender account number: ";
    cin >> senderAccNum;
    cout << "Enter receiver account number: ";
    cin >> receiverAccNum;
    cout << "Enter amount to transfer: ";
    cin >> amount;

    if (senderAccNum > 0 && senderAccNum <= accounts.size() && receiverAccNum > 0 && receiverAccNum <= accounts.size()) {
        accounts[senderAccNum - 1].transfer(accounts[receiverAccNum - 1], amount);
    } else {
        cout << "Invalid account numbers!" << endl;
    }
}
```

```
void menu() {
    int choice;
    do {
        cout << "\n--- Banking System Menu ---\n";
        cout << "1. Create Account\n";
        cout << "2. Deposit\n";
        cout << "3. Withdraw\n";
        cout << "4. Balance Enquiry\n";
        cout << "5. Transfer\n";
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                createAccount();
                break;
            case 2:
                deposit();
                break;
            case 3:
                withdraw();
                break;
```

```
            case 4:
                balanceEnquiry();
                break;
            case 5:
                transfer();
                break;
            case 6:
                cout << "Exiting the Banking System. Goodbye!" << endl;
                break;
            default:
                cout << "Invalid choice! Please try again." << endl;
                break;
        }
    } while (choice != 6);
  }
};

int main() {
  BankingSystem bankSystem;
  bankSystem.menu();
  return 0;
}
```

Programiz C++ Online Compiler

main.cpp

Share Run

```
150                 break;
151             case 3:
152                 withdraw();
153                 break;
154             case 4:
155                 balanceEnquiry();
156                 break;
157             case 5:
158                 transfer();
159                 break;
160             case 6:
161                 cout << "Exiting the Banking System.
                        Goodbye!" << endl;
162                 break;
163             default:
164                 cout << "Invalid choice! Please try again
                        ." << endl;
165                 break;
166         }
167     } while (choice != 6);
168 }
169 };
170
171
```

Output Clear

```
--- Banking System Menu ---
1. Create Account
2. Deposit
3. Withdraw
4. Balance Enquiry
5. Transfer
6. Exit
Enter your choice: 1
Enter account holder name: Guna
Account created successfully! Account Number: 1

--- Banking System Menu ---
1. Create Account
2. Deposit
3. Withdraw
4. Balance Enquiry
5. Transfer
6. Exit
Enter your choice: 2
Enter account number: 1
Enter amount to deposit: 1000
Deposited: $1000
```

main.cpp

Share Run

```
150                 break;
151             case 3:
152                 withdraw();
153                 break;
154             case 4:
155                 balanceEnquiry();
156                 break;
157             case 5:
158                 transfer();
159                 break;
160             case 6:
161                 cout << "Exiting the Banking System.
                        Goodbye!" << endl;
162                 break;
163             default:
164                 cout << "Invalid choice! Please try again
                        ." << endl;
165                 break;
166         }
167     } while (choice != 6);
168 }
169 };
170
171
```

Output Clear

```
--- Banking System Menu ---
1. Create Account
2. Deposit
3. Withdraw
4. Balance Enquiry
5. Transfer
6. Exit
Enter your choice: 2
Enter account number: 1
Enter amount to deposit: 1000
Deposited: $1000

--- Banking System Menu ---
1. Create Account
2. Deposit
3. Withdraw
4. Balance Enquiry
5. Transfer
6. Exit
Enter your choice: 3
Enter account number: 1
Enter amount to withdraw: 500
Withdrawn: $500
```

Programiz C++ Online Compiler

Programiz PRO >

main.cpp

Share Run

```
150                 break;
151             case 3:
152                 withdraw();
153                 break;
154             case 4:
155                 balanceEnquiry();
156                 break;
157             case 5:
158                 transfer();
159                 break;
160             case 6:
161                 cout << "Exiting the Banking System.
                        Goodbye!" << endl;
162                 break;
163             default:
164                 cout << "Invalid choice! Please try again
                        ." << endl;
165                 break;
166         }
167     } while (choice != 6);
168 }
169 };
170
171
```

Output

Clear

```
--- Banking System Menu ---
1. Create Account
2. Deposit
3. Withdraw
4. Balance Enquiry
5. Transfer
6. Exit
Enter your choice: 4
Enter account number: 1
Balance for account 1: $500

--- Banking System Menu ---
1. Create Account
2. Deposit
3. Withdraw
4. Balance Enquiry
5. Transfer
6. Exit
Enter your choice: 6
Exiting the Banking System. Goodbye!


=== Code Execution Successful ===
```

CONCLUSION

The Banking System developed in the provided code effectively addresses fundamental functionalities essential for managing train few banking sector services. Through its structured design utilizing classes for users, the system enables a seamless experience for users by allowing them to create accounts , Deposit, book withdraw, Bank enquiry and Transfer as needed.

THANK YOU