OOP PROJECT REPORT

ON

BANKING MANAGEMENT SYSTEM

School of Engineering & Technology

MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES, Faridabad
(NAAC Accredited 'A' Grade)

3rd Semester (Academic Year 2024-25)

COURSE NAME: Object Oriented Programming Lab
COURSE CODE: BCS-DS-352A

Submitted By:-
Ayushi Bindal (1/23/SET/BCS/341)
Diya Sharma (1/23/SET/BCS/347)
Branch: BTech CSE
Section: F2

Submitted To:-
Faculty Name:
Mr. Ashok Madaan
Designation: A.P.
Department: CSE

Hardware and Software Requirements

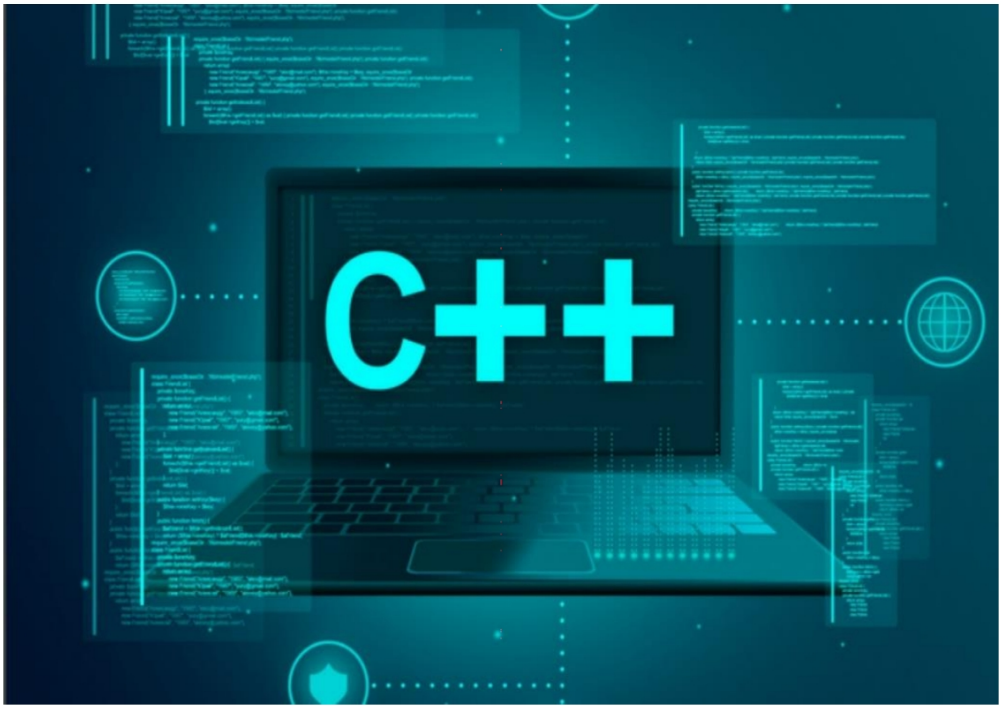
Hardware Requirements:

- Minimum CPU: Intel i3 Processor or above required.
- 4 GB RAM or above necessary.
- System should be 10th generation or above.
- Monitors and keyboard are implied necessary tools

Software Requirements:

- Operating System: Windows 10 or higher/ Linux/ macOS required
- Programming Language: C++
- Development Tools: Visual Studio Code, Turbo C++, Online C++ compiler

INTRODUCTION TO C++ LANGUAGE IN OOP



C++ is a versatile and powerful programming language widely known for its performance, flexibility, and the combination of low-level control with high-level functionality. Developed by Bjarne Stroustrup in the early 1980s as an enhancement to C, C++ introduced object-oriented programming (OOP) principles into the C programming framework, making it one of the first languages to popularize OOP.

OOP in C++ helps in creating modular, reusable, and maintainable code, making it widely used for applications requiring efficient performance, such as operating systems, games, real-time systems, and more. The primary features of OOP in C++ include encapsulation, inheritance, polymorphism, and abstraction.

Key Concepts of Object-Oriented Programming in C++ are :-

Classes and Objects: In C++, a class is a blueprint for creating objects. It defines attributes (data members) and behaviours (member functions) that the objects created from it will have.

An object is an instance of a class. It represents a real-world entity with characteristics (attributes) and actions (methods).

The foundational concepts of OOP in C++ are encapsulation, inheritance, polymorphism, and abstraction. Together, these principles allow developers to create modular, reusable, and maintainable code:

1. **Encapsulation:** Encapsulation in C++ involves bundling data (variables) and functions (methods) that operate on the data within a single unit called a "class." This concept ensures that the internal representation of an object is hidden from the outside world, safeguarding data integrity and reducing complexity. Access specifiers (public, protected, and private) control visibility, making certain elements accessible only within the class, to derived classes, or publicly.
2. **Inheritance:** Inheritance allows a new class (derived class) to acquire properties and behaviours from an existing class (base class), fostering code reuse. This hierarchical relationship allows for streamlined enhancements, as modifications in a base class can cascade down to derived classes, creating more maintainable code.
3. **Polymorphism:** Polymorphism enables one interface to be used for different data types, fostering flexibility and scalability in code. In C++, polymorphism is achieved through function overloading (compile-time) and virtual functions (runtime). Virtual functions allow derived classes to override methods of the base class, achieving runtime polymorphism.
4. **Abstraction:** Abstraction simplifies complex reality by modelling classes appropriate to the problem, focusing only on essential details. C++ achieves abstraction through classes and access modifiers, providing a clear interface while hiding internal implementation details.

CODE EXPLANATION

Our program is a simple banking management system in C++. It allows users to perform basic operations such as creating an account, depositing and withdrawing money, viewing account details, listing all accounts, and exiting the system. Now explaining the main subparts of the program one by one.

1. Account Class

The Account class models a single bank account with the following features:

- Attributes: It has attributes for accountNumber, name (account holder), and balance.
- Constructor: Initializes an account with an account number, holder name, and an optional initial balance.
- Methods:
    - deposit(double amount): Adds a specified amount to the balance.
    - withdraw(double amount): Deducts a specified amount from the balance if sufficient funds are available.
    - displayAccount(): Displays account details.
    - getAccountNumber(): Returns the account number.
    - getBalance(): Returns the balance.

2. BankingSystem Class

The BankingSystem class manages a collection of accounts:

- Attributes: It has a vector accounts to store multiple Account objects.

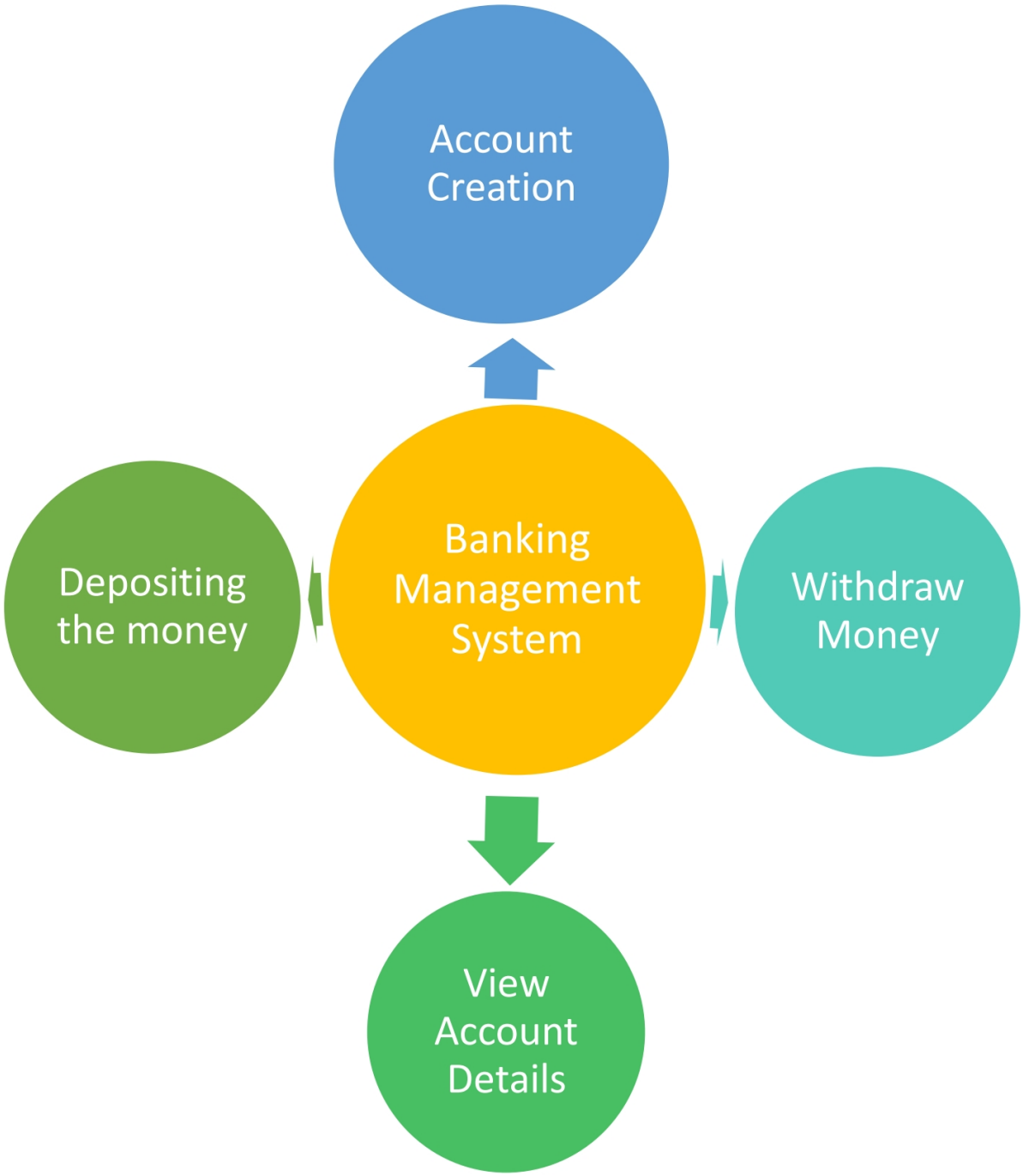
- Methods:
  - createAccount(string name, double initialBalance): Generates a new account number, creates an account, and adds it to the accounts vector.
  - deposit(int accountNumber, double amount): Finds the account by its number and deposits the specified amount.
  - withdraw(int accountNumber, double amount): Finds the account by its number and withdraws the specified amount if funds are sufficient.
  - displayAccountDetails(int accountNumber): Displays details of the specified account.
  - listAccounts(): Lists all accounts with their details.

3. main Function

The main function provides a menu-driven interface for interacting with the banking system. It performs the following actions based on user input:

- Option 1: Create a new account by prompting for a name and an initial deposit.
- Option 2: Deposit a specified amount into an account.
- Option 3: Withdraw a specified amount from an account.
- Option 4: Display details of a specified account.
- Option 5: List all accounts with details.
- Option 6: Exit the program.

PROJECT DIAGRAM



Source Code

```
#include<iostream>
#include<fstream>
#include<vector>
using namespace std;

class Account{
private:
   int accountNumber;
   string name;
   double balance;
public:
   Account(int accNum, string accName, double initialBalance = 0.0)
      : accountNumber(accNum), name(accName), balance(initialBalance) {}

   void deposit(double amount) {
      balance += amount;
   }
   void withdraw(double amount) {
      if (amount <= balance) {
         balance -= amount;
      } else {
         cout << "Insufficient funds!" << endl;
      }
   }
   void displayAccount() const {
      cout << "Account Number : " << accountNumber
           << "\nAccount Holder : " << name
           << "\nBalance : $" << balance << "\n";
   }
   int getAccountNumber() const { return accountNumber; }
   double getBalance() const { return balance; }
};

class BankingSystem{
private:
   vector<Account> accounts;
   int generateAccountNumber() {
      return accounts.size() + 1;
   }
public:
   void createAccount(string name, double initialBalance = 0.0) {
      int accNum = generateAccountNumber();
      accounts.push_back(Account(accNum, name, initialBalance));
      cout << "Account created with account number : " << accNum << endl;
```

```
    }
    void deposit(int accountNumber, double amount) {
        for (auto &acc : accounts) {
            if (acc.getAccountNumber() == accountNumber) {
                acc.deposit(amount);
                cout << "Deposited $" << amount << " to account number " << accountNumber <<
endl;
                return;
            }
        }
        cout << "Account not found." << endl;
    }
    void withdraw(int accountNumber, double amount) {
        for (auto &acc : accounts) {
            if (acc.getAccountNumber() == accountNumber) {
                acc.withdraw(amount);
                cout << "Withdrew $" << amount << " from account number " << accountNumber <<
endl;
                return;
            }
        }
        cout << "Account not found." << endl;
    }
    void displayAccountDetails(int accountNumber) const {
        for (const auto &acc : accounts) {
            if (acc.getAccountNumber() == accountNumber) {
                acc.displayAccount();
                return;
            }
        }
        cout << "Account not found." << endl;
    }
    void listAccounts() const {
        for (const auto &acc : accounts) {
            acc.displayAccount();
            cout << "----------------------\n";
        }
    }
};

int main() {
    BankingSystem bank;
    int choice;
    string name;
    double amount;
    int accountNumber;
```

```
    while (true) {
        cout << "\nBanking Management System\n1. Create Account\n2. Deposit\n3.
Withdraw\n4. Account Details\n5. List All Accounts\n6. Exit\nChoose an option : ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter name for new account : ";
                cin >> name;
                cout << "Enter initial deposit : ";
                cin >> amount;
                bank.createAccount(name, amount);
                break;
            case 2:
                cout << "Enter account number : ";
                cin >> accountNumber;
                cout << "Enter amount to deposit : ";
                cin >> amount;
                bank.deposit(accountNumber, amount);
                break;
            case 3:
                cout << "Enter account number : ";
                cin >> accountNumber;
                cout << "Enter amount to withdraw : ";
                cin >> amount;
                bank.withdraw(accountNumber, amount);
                break;
            case 4:
                cout << "Enter account number : ";
                cin >> accountNumber;
                bank.displayAccountDetails(accountNumber);
                break;
            case 5:
                bank.listAccounts();
                break;
            case 6:
                cout<<"-------THANKYOU-------";
                return 0;
            default:
                cout << "Invalid option! Try again." << endl;
        }
    }
}
```

Output Screenshots

Output

```
Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 1
Enter name for new account : Jiya
Enter initial deposit : 6700
Account created with account number : 1
```

```
Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 1
Enter name for new account : Sarthak
Enter initial deposit : 26900
Account created with account number : 2

Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 1
Enter name for new account : Dhanwan
Enter initial deposit : 86539
Account created with account number : 3
```

```
Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 5
Account Number : 1
Account Holder : Jiya
Balance : $6700
---------------------
Account Number : 2
Account Holder : Sarthak
Balance : $26900
---------------------
Account Number : 3
Account Holder : Dhanwan
Balance : $86539
---------------------
```

```
Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 2
Enter account number : 2
Enter amount to deposit : 5000
Deposited $5000 to account number 2

Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 3
Enter account number : 4
Enter amount to withdraw : 2899
Account not found.
```

```
Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 3
Enter account number : 3
Enter amount to withdraw : 2700
Withdrew $2700 from account number 3

Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 4
Enter account number : 1
Account Number : 1
Account Holder : Jiya
Balance : $6700
```

```
Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 5
Account Number : 1
Account Holder : Jiya
Balance : $6700
----------------------
Account Number : 2
Account Holder : Sarthak
Balance : $31900
----------------------
Account Number : 3
Account Holder : Dhanwan
Balance : $83839
----------------------
```

```
Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 7
Invalid option! Try again.

Banking Management System
1. Create Account
2. Deposit
3. Withdraw
4. Account Details
5. List All Accounts
6. Exit
Choose an option : 6
-------THANKYOU-------

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

LIMITATIONS

- Lack of Persistent Data Storage: This code stores account information in memory (std::vector<Account>), which means all account data is lost once the program is closed whereas, in real-world applications, a database or file storage system is used to store account details persistently. Adding file operations or database integration would allow the system to retain data after it is closed.

- No Security or Access Control: The program does not implement any security measures, such as user authentication or password protection. In real banking systems, access to accounts is restricted with login credentials, PINs, or multi-factor authentication, ensuring that only authorized users can access or modify account information.

- Limited Transaction Capabilities : This system lacks a transaction history feature, so users cannot view past deposits, withdrawals, or changes made to the account. Real banking systems maintain detailed transaction logs that allow users and auditors to trace each action on an account.

- No Graphical User Interface (GUI) : This project uses a command-line interface, which limits its usability. Real banking systems often have user-friendly graphical interfaces or even web-based or mobile interfaces to provide easy and intuitive access for users.

- The above program also does not have a feature to delete an account or to review transaction history.

REFERENCES

- https://stackoverflow.com/ for fixing errors.
- https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/
- https://iq.opengenus.org/bank-management-system-in-cpp/
- We have used Youtube and Google for source code.
- Some content from direct Google searches for collecting different information.