|  |  |
| --- | --- |
| OOPS LAB MANUAL  Online Examination System | GRoup#7   1. Mahad ur Rahman (38) 2. Sohaib Ali (39) 3. Minhaj Haider (41)   Object-Oriented Programming |
|  |  |

Object-Oriented Programming Project Manual

# Project Title: Online Examination System

# Table of Contents

1. Introduction

2. Tools and Technologies

3. System Design

3.1 Class Diagram

3.2 Class Descriptions

3.3 Relationships

4. Implementation Details

4.1 Core Functionalities

4.2 Sample Code Snippets

5. File Handling

6. Testing and Output

7. Results

8. Limitations

9. Future Enhancements

10. Conclusion

11. References

12. Appendices

# 1. Introduction

Objective: To design and implement a console-based Online Examination System using C++ that demonstrates object-oriented programming concepts like inheritance, polymorphism, operator overloading, and file handling.

Scope: This system allows two types of users – Admin and Student. Admins can view results, while students can take a multiple-choice exam and have their results saved to a file.

# 2. Tools and Technologies

Language: C++

Compiler: g++ / Visual Studio Code with C++ extension

Concepts Used: Inheritance, Polymorphism, Operator Overloading, File Handling, Abstract Classes

# 3. System Design

## 3.1 Class Diagram

## 3.2 Class Descriptions

- User (Abstract Class): Contains a pure virtual login function.  
- Admin: Inherits from User. Can login and view stored student results.  
- Student: Inherits from User. Inputs name and attempts the quiz.  
- Question (Abstract Class): Base for MCQ with virtual functions.  
- MCQ: Inherits from Question, contains question text and correct answer.  
- Answer: A simple class with overloaded == operator to compare answers.

## 3.3 Relationships

Inheritance: User → Admin, Student  
Polymorphism: Question → MCQ via virtual functions  
Operator Overloading: Answer class compares user answer with correct answer

# 4. Implementation Details

## 4.1 Core Functionalities

- Admin login with hardcoded credentials  
- Student login with name input  
- Quiz system with multiple MCQs  
- Result calculation and display  
- Result saved to a text file

## 4.2 Sample Code Snippet

Answer(char a) : ans(toupper(a)) {}  
bool operator==(const Answer &other) {  
 return ans == other.ans;  
}

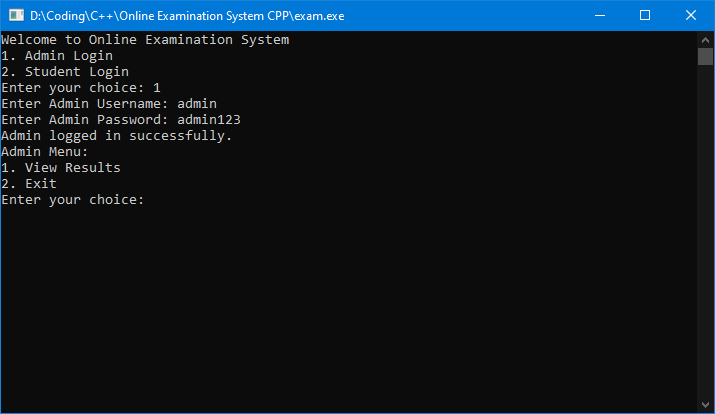
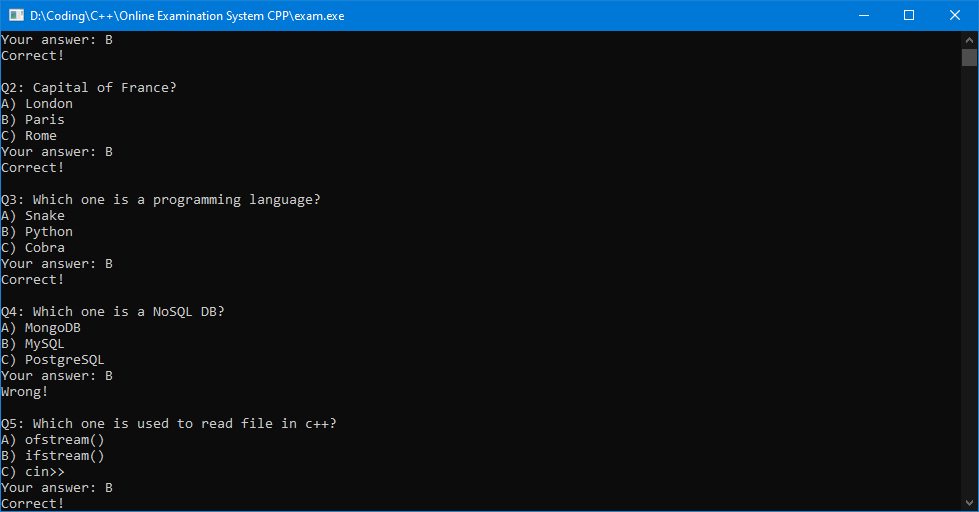
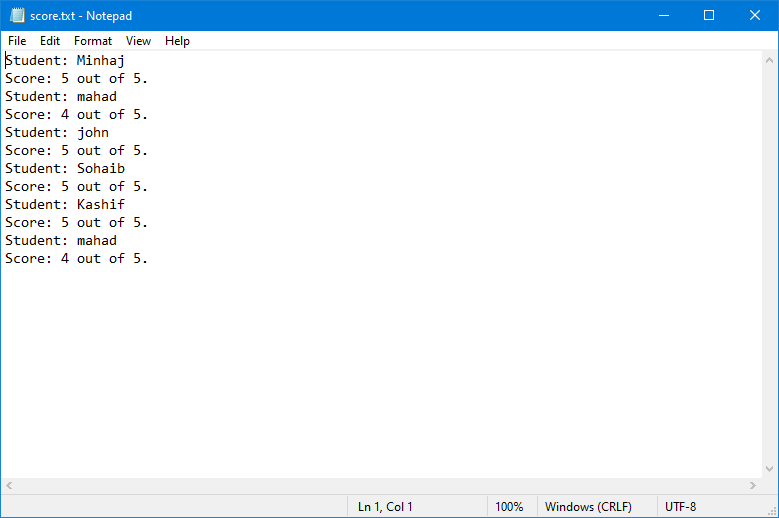
# 5. File Handling

The file '*data/score.txt*' is used to append student results.  
Results are written in the following format:  
Student: [Name]  
Score: X out of Y.

# 6. Testing and Output

- Sample Inputs:  
 Admin login: admin/admin123  
 Student Name: Ali Raza

- Sample Outputs and Screenshots:

# 7. Results

- Student name is accepted properly using getline  
- Quiz results are calculated correctly  
- Results are saved in file and viewable by admin  
- Code follows OOP concepts successfully

# 8. Limitations

- Console-based UI  
- Hardcoded admin login  
- No persistent student data storage  
- Text file-based result management  
- No retry attempts or question randomization

# 9. Future Enhancements

- Add GUI using Qt or C++/CLI  
- Store questions and user data in SQLite or MySQL  
- Add authentication for each student  
- Implement question shuffling or multiple exam sets

# 10. Conclusion

This project helped reinforce object-oriented programming principles like inheritance, polymorphism, and operator overloading. File handling allowed for basic persistent storage. The Online Examination System demonstrates the ability to design and implement modular, scalable applications in C++.

# 11. References

- https://www.geeksforgeeks.org/  
- C++ Object-Oriented Programming notes  
- Our repo: https://github.com/dopescripts/Online-Examination-System-CPP

# 12. Appendices

Source Code: <https://github.com/dopescripts/Online-Examination-System-CPP>

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <cctype> // for toupper()

using namespace std;

// Abstract Base Class

class User

{

public:

    virtual void login() = 0; // pure virtual function

};

// Admin Class

class Admin : public User

{

public:

    void login() override

    {

        string username, password;

        cout << "Enter Admin Username: ";

        cin >> username;

        cout << "Enter Admin Password: ";

        cin >> password;

        if (username == "admin" && password == "admin123")

        {

            cout << "Admin logged in successfully." << endl;

        }

        else

        {

            cout << "Invalid username or password" << endl;

            return; // Exit program if login fails

        }

    }

    void viewResults()

    {

        ifstream inFile("data/score.txt");

        if (!inFile)

        {

            cerr << "Error opening file for reading." << endl;

            return;

        }

        string line;

        cout << "Results:\n";

        while (getline(inFile, line, '.'))

        {

            cout << line << endl;

            cout << "-------------------" << endl;

        }

    }

};

// Student Class

class Student : public User

{

public:

    string name;

    Student() {}

    Student(string n) : name(n) {}

    void login() override

    {

        cout << "Student " << name << " logged in.\n";

    }

    void inputAndLogin()

    {

        cout << "Enter your name (single word): ";

        cin.ignore();

        getline(cin, name); // Use getline to allow spaces

        login();

    }

};

// Abstract Base Question Class

class Question

{

public:

    virtual void ask() = 0;

    virtual char getCorrectAnswer() = 0;

    virtual ~Question() {} // Virtual destructor

};

// MCQ Class, Derived from Question

class MCQ : public Question

{

    string text;

    char correctOption;

public:

    MCQ(string t, char c) : text(t), correctOption(toupper(c)) {} // constructor

    void ask() override

    {

        cout << text << endl;

    }

    char getCorrectAnswer() override

    {

        return correctOption;

    }

};

// Answer Class with Operator Overloading

class Answer

{

    char ans;

public:

    Answer(char a) : ans(toupper(a)) {}

    bool operator==(const Answer &other)

    {

        return ans == other.ans;

    }

};

int main()

{

    Student s;

    int user;

    cout << "Welcome to Online Examination System" << endl;

    cout << "1. Admin Login\n2. Student Login\nEnter your choice: ";

    cin >> user;

    if (user == 1)

    {

        Admin admin;

        admin.login();

        cout << "Admin Menu:\n";

        cout << "1. View Results\n2. Exit\nEnter your choice: ";

        int choice;

        cin >> choice;

        if (choice == 1)

        {

            admin.viewResults();

            system("pause");

        }

        else

        {

            cout << "Exiting...\n";

            return 0;

        }

        return 0;

    }

    else if (user == 2)

    {

        // Student Login

        s.inputAndLogin();

    }

    else

    {

        cout << "Invalid choice!" << endl;

    }

    // Creating the quiz (MCQs only)

    vector<Question \*> quiz;

    // push\_back for Output

    quiz.push\_back(new MCQ("Q1: What is 2 + 2?\nA) 3\nB) 4\nC) 5", 'B'));

    quiz.push\_back(new MCQ("Q2: Capital of France?\nA) London\nB) Paris\nC) Rome", 'B'));

    quiz.push\_back(new MCQ("Q3: Which one is a programming language?\nA) Snake\nB) Python\nC) Cobra", 'B'));

    quiz.push\_back(new MCQ("Q4: Which one is a NoSQL DB?\nA) MongoDB\nB) MySQL\nC) PostgreSQL", 'A'));

    quiz.push\_back(new MCQ("Q5: Which one is used to read file in c++?\nA) ofstream()\nB) ifstream()\nC) cin>>", 'B'));

    // Start the quiz

    cout << "Starting the quiz...\n";

    cout << "Please answer with A, B or C.\nReady?";

    system("pause");

    char userInput;

    int score = 0;

    for (auto q : quiz)

    {

        q->ask();

        cout << "Your answer: ";

        cin >> userInput;

        Answer userAns(userInput);

        Answer correctAns(q->getCorrectAnswer());

        if (userAns == correctAns)

        {

            cout << "Correct!\n\n";

            score++;

        }

        else

        {

            cout << "Wrong!\n\n";

        }

    }

    // Save result to file

    ofstream outFile("data/score.txt", ios::app);

    if (!outFile)

    {

        cerr << "Error opening file for writing." << endl;

        return 1;

    }

    outFile << "Student: " << s.name << endl;

    outFile << "Score: " << score << " out of " << quiz.size() << "." << endl;

    outFile.close();

    cout << "Your score: " << score << " out of " << quiz.size() << endl;

    cout << "Your score has been saved! Thank you\n";

    // Free memory because we use new

    for (auto q : quiz)

    {

        delete q;

    }

    system("pause");

    cout << "Exiting...\n";

    return 0;

}

Screenshots:

