[QUIZ APPLICATION]

Project Submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfilment of the requirements to award the degree of

Bachelor of Technology

In

Computer science and Engineering

School of Engineering and Sciences

Submitted by

AP23110010277 – CH.PAVAN KALYAN RAM

AP23110010274 – G TEJESH KUMAR

AP23110010311 – G.PRANAY

AP23110010313 – CH.SANTOSH

Under the Guidance of

(MS.KAVITHA RANI)

SRM University - AP

NeeruKonda, Mangalagiri, Guntur, Andhra Pradesh - 522240

[NOVEMBER 2024]

Certificate

Date: 20-NOV 2024

This is to certify that the work present in this Project entitled **"QUIZ APPLICATION"** has been carried out By **GROUP 18** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in School of Engineering and Sciences.

Supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Co-supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to everyone who has contributed to the successful completion of this project report on "Quiz Application in C++ Using Object-Oriented Programming."

First and foremost, I would like to thank MS. Kavitha rani mam for their invaluable guidance, support, and encouragement throughout this project. Their insightful feedback and expertise in the field of Object-Oriented Programming have been instrumental in refining my understanding and approach to developing this application. I extend my appreciation to the faculty members of the School of Engineering and sciences at SRM AP for providing a conducive environment for learning and research, which has greatly inspired my academic pursuits.

I am also grateful to my peers and classmates who offered their support and shared ideas, making teamwork and discussions both enlightening and enjoyable. The collaborative spirit greatly motivated me to enhance the functionality and design of the application.

Finally, I would like to thank my friends for their constant encouragement and understanding during the time invested in this project.

TABLE OF CONTENTS

ABSTRACT

This project is about building a Quiz Application using C++ and Object-Oriented Programming (OOP). The main purpose of the application is to create a fun and easy-to-use quiz game that highlights the use of OOP concepts like classes, inheritance, and encapsulation.

The application allows users to choose different quiz topics and difficulty levels. Questions are presented in a random order to make the quiz more engaging. The program uses classes to manage questions, quiz functions, and scores, making the code organized and easy to update or expand in the future. The project also includes features for error handling to ensure the application runs smoothly and handles user input correctly. Overall, this project demonstrates how C++ and OOP can be used to build practical and efficient software solutions.

Introduction

In the ever-evolving landscape of technology, learning through interactive applications has become one of the most effective methods for both education and entertainment. Quizzes, in particular, are an excellent way to test knowledge, engage users, and make learning more enjoyable. This project focuses on developing a simple yet efficient Quiz Application using C++ programming, showcasing the application of fundamental programming concepts such as arrays, loops, and conditional statements, without the use of advanced object-oriented principles. The primary aim of this project is to create an interactive quiz system where users can test their knowledge on various topics. The application allows users to choose answers to multiple-choice questions and provides immediate feedback on whether their answer is correct. By using C++, this project demonstrates how programming languages can be utilized to develop real-world applications that are not only functional but also engaging for users.

The quiz application operates through a straightforward structure, with questions and multiple-choice options stored in arrays. The user interacts with the system via a console-based interface, where they are prompted to choose the correct answer by selecting the corresponding option number. After answering each question, the program checks if the user's input matches the correct answer stored in the array and updates the user's score accordingly. At the end of the quiz, the user is provided with a total score that reflects their performance throughout the quiz.

The design of this application is simple yet effective, with a primary goal of enhancing the user's learning experience. The program was developed using a straightforward approach, avoiding complex object-oriented programming techniques or advanced features, making it accessible to new programmers. The interface, though basic, ensures clear communication of the quiz questions, options, and the results, providing a user-friendly experience. Additionally, the program ensures that any invalid input is properly handled, thus maintaining smooth execution and reducing the risk of errors. A key feature of the application is its adaptability. While this version of the quiz includes hardcoded questions and answers, the design is flexible enough to allow for future expansion. For instance, the application can easily be modified to load questions from external files, add different categories of quizzes, or even track high scores over multiple

sessions. This demonstrates the power of C++ as a language that is capable of scaling from simple projects to more complex systems.

The project also highlights the importance of efficient error handling. Simple input validation ensures that the program does not crash when a user enters invalid data, such as a number outside the available options. Proper error handling is a crucial part of any software development process, and even in this basic application, it ensures that the user experience remains smooth and reliable.

Furthermore, the development of this quiz application serves as a practical exercise for understanding the core concepts of C++ programming, such as the use of arrays, loops, and conditionals. These concepts are the foundation of many more advanced techniques in C++ and other programming languages, making this project a valuable stepping stone for anyone looking to deepen their programming knowledge.

METHODOLOGY:

The development of the Quiz Application follows a straightforward approach using basic C++ concepts, including arrays, loops, and conditionals, without the need for complex functions or classes. The methodology consists of the following key steps:

1. Requirement Analysis

Initially, the requirements for the quiz application were gathered. The goal was to create a simple, interactive quiz program that allows users to answer multiple-choice questions and receive immediate feedback on their performance.

2. System Design

A basic design was chosen, focusing on simplicity and ease of implementation. The system was broken down into three main parts:

- **Questions:** An array was created to store the quiz questions.
- **Options:** A 2D array was used to store the multiple-choice options for each question.
- **Correct Answers:** An array was used to store the index of the correct answer for each question.

The design ensures easy modification of questions and options while maintaining a clear and concise structure.

3. Implementation

- **Data Representation:**

  Arrays were used to store the questions, options, and correct answers. This approach is both memory-efficient and easy to manage for small-scale projects like this.

- **User Interaction:**

  A simple console-based interface was used to display questions and options to the user. The user is prompted to enter their answer by choosing the number corresponding to their chosen option.

- **Score Calculation:**

  A loop was used to go through each question, take the user's answer, check if it matches the correct answer, and update the score accordingly. The logic was kept simple, ensuring that the application functions smoothly while providing accurate feedback to the user.

4. Testing and Debugging

After implementation, testing was performed to verify the correctness of the quiz functionality. Different test cases were created to check for edge cases, such as invalid input (e.g., entering a number outside the available options). Debugging was carried out to ensure that the program handled all possible scenarios correctly.

5. Evaluation and Optimization

The final step involved evaluating the overall functionality and performance of the application. As the program was small in scale, no major performance issues were encountered. The code was optimized for clarity and simplicity, ensuring that future modifications, such as adding more questions, could be done easily by adjusting the arrays.

PSEUDO-CODE REPRESNTAION:

```
Start
    Display welcome message
    Initialize score to 0
    Initialize array of questions and options
    Initialize array of correct answers
    Shuffle the questions for randomness
    While user wants to play again:
        Initialize score to 0
        Shuffle the questions randomly
        For each question:
            Display the question and options
            Set a timer for 10 seconds
            While the user hasn't answered and time is remaining:
                Prompt user for answer (1-4)
                Validate the answer:
                    If valid answer is provided:
                        Check if answer is correct:
                            If correct, increase score
                            If incorrect, display correct answer
                    If invalid answer, prompt again
                    If timer runs out, display time's up and correct answer
        Display the total score
        Ask user if they want to play again (y/n)
End
```

The pseudo-code represents the logical flow of the quiz application. It starts by displaying a welcome message and initializing the score to zero. The program then enters a loop where it presents each question along with four options to the user. After the user inputs their answer, the input is validated to ensure it's a valid option (1–4). If the answer is correct, the score is incremented; if it's incorrect, the correct answer is displayed. After all questions have been answered, the program displays the total score and prompts the user to decide whether to play again. The loop continues until the user opts to stop playing.

Overview of the Code

The Quiz Application developed in C++ is a simple console-based program that allows users to answer multiple-choice questions. The application utilizes basic C++ constructs such as arrays, loops, and conditionals to create a functional and interactive quiz experience. The code is designed to be straightforward and easy to understand, with the goal of introducing users to fundamental programming concepts.

1. Data Representation

The quiz questions, options, and correct answers are stored using arrays in the program. Specifically:

- questions []: A string array is used to store the text for each quiz question. This array holds the questions in a sequential manner, with each question corresponding to an index.
- options []: A 2D array of strings is used to store the four multiple-choice options for each question. Each row in this array corresponds to a question, and each column represents one of the four possible options (A, B, C, D).
- correct Answers []: An integer array is used to store the index of the correct option for each question. This array ensures that the program can easily verify if the user's answer matches the correct one.

2. User Interaction

The application allows for simple user interaction through a console interface:

- The user is presented with a quiz question and a list of four options.
- The user is prompted to enter a number (1–4) corresponding to the option they believe is correct.
- The program reads the user's input, checks if it matches the correct answer, and updates the score accordingly.

3. Question Display and Answer Validation

- For each question, the program displays the question text followed by the four multiple-choice options.
- After displaying the options, the user is prompted to enter their answer, which is read as an integer input.
- The program then compares the user's input with the correct answer, which is stored in the correct Answers [] array.
- If the answer is correct, the program prints "Correct!" and increments the score. If the answer is wrong, the program informs the user of the correct answer.

4. Score Calculation

The score is tracked throughout the quiz. It is initialized to zero at the beginning and is incremented by one every time the user answers a question correctly. At the end of the quiz, the user's total score is displayed, showing how many questions were answered correctly out of the total number of questions.

5. Error Handling

The current version of the application does not implement advanced error handling for invalid inputs (e.g., entering a number outside the 1–4 range). However, in its current form, it assumes the user will provide valid input. Future versions of the program could implement input validation to handle such cases, ensuring that the user enters only valid answers.

6. End of Quiz

Once all the questions have been answered, the program displays a message showing the total score out of the total number of questions. This gives the user clear feedback on their performance, which enhances the interactive experience.

7. Code Efficiency

The program is written efficiently, with minimal lines of code while maintaining clarity. The use of arrays to store questions, options, and answers keeps the code organized. The for loops simplify the task of displaying the questions and options as well as checking the answers. This design ensures that the program is easy to modify, such as adding new questions or changing existing ones.

8. Future Improvements

- Input Validation: The code could be extended to handle invalid input. For example, if a user enters a number outside the valid range of 1–4, the program could prompt them again to enter a valid choice.
- Dynamic Question Loading: Instead of hardcoding the questions, a future version of the application could load questions from an external file or database, allowing for easier updates and expansions.
- Timer Feature: A timer could be added to limit the time allowed for answering each question, adding an extra level of challenge.
- Graphical User Interface (GUI): While the current version uses a console interface, a GUI could be developed in the future for a more interactive and visually appealing experience.
- Multiple Quizzes and Categories: The program could be extended to allow the user to select different categories of quizzes or even set difficulty levels (easy, medium, hard).

SOURCE CODE:

QUIZ APPLICATION SOURCE CODE.cpp

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <unistd.h>
using namespace std;

void shuffleQuestions(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        int randIndex = rand() % size;
        int temp = arr[i];
        arr[i] = arr[randIndex];
        arr[randIndex] = temp;
    }
}

int main()
{
    string questions[] =
    {
        "What is the capital of Telangana?",
        "What is the national song of INDIA?",
        "Who is the Prime Minister of India?",
        "What is the capital of India?",
        "Which city is known as the Silicon Valley of India?",
        "What is the national flower of India?"
    };

    string options[][4] =
    {
        {"Madhyapradesh", "UP", "AP", "Hyderabad"},
        {"Vande Mataram", "Sare Jahan se Accha", "National Anthem", "Chak de India"},
        {"Donald Trump", "Narendra Modi", "Joe Biden", "K.A.Paul"},
        {"New Delhi", "Mumbai", "Chennai", "Kolkata"},
        {"Bangalore", "Hyderabad", "Chennai", "Pune"},
        {"Lotus", "Rose", "Sunflower", "Tulip"}
```

```
};

int correctAnswers[] = {4, 1, 2, 1, 1, 1};
int score = 0;
int userAnswer;
char playAgain = 'y';

srand(time(0));

cout << "Welcome to the Quiz!" << endl;
cout << "Answer the following questions by entering the option number (1-4)." << endl;

while (playAgain == 'y' || playAgain == 'Y')
    {
    score = 0;
    int questionOrder[] = {0, 1, 2, 3, 4, 5};
    shuffleQuestions(questionOrder, 6);

    for (int i = 0; i < 6; i++)
        {
        cout << "\nQuestion " << i + 1 << ": " << questions[questionOrder[i]] << endl;
        for (int j = 0; j < 4; j++)
        {
            cout << j + 1 << ". " << options[questionOrder[i]][j] << endl;
        }

        bool validInput = false;
        int timer = 10;
        while (!validInput && timer > 0)
            {
            cout << "Your answer (1-4): ";
            cin >> userAnswer;
            if (userAnswer >= 1 && userAnswer <= 4)
             {
                validInput = true;
            }
        else
```

```
            {
                cout << "Invalid input! Please choose a number between 1 and 4." << endl;
            }
            sleep(1);
            if (timer > 0)
            {
                timer--;
            }
        }

        if (validInput && userAnswer == correctAnswers[questionOrder[i]])
            {
            cout << "Correct!" << endl;
            score++;
        }
        else if (!validInput)
            {
            cout << "Time's up! The correct answer was option " << correctAnswers[questionOrder[i]] << ": "
                 << options[questionOrder[i]][correctAnswers[questionOrder[i]] - 1] << "." << endl;
        }
        else
            {
            cout << "Wrong answer. The correct answer was option " << correctAnswers[questionOrder[i]] << ": "
                 << options[questionOrder[i]][correctAnswers[questionOrder[i]] - 1] << "." << endl;
        }
    }

    cout << "\nQuiz completed!" << endl;
    cout << "Your total score: " << score << " out of 6" << endl;
    cout << "Would you like to play again? (y/n): ";
    cin >> playAgain;
}

cout << "Thank you for playing the quiz! Goodbye!" << endl;

return 0;
}
```

SAMPLE OUTPUT:

```
Welcome to the Quiz!
Answer the following questions by entering the option number (1-4).

Question 1: What is the capital of Telangana?
1. Madhyapradesh
2. UP
3. AP
4. Hyderabad
Your answer (1-4): 4
Correct!

Question 2: What is the national song of INDIA?
1. Vande Mataram
2. Sare Jahan se Accha
3. National Anthem
4. Chak de India
Your answer (1-4): 1
Correct!
```

```
Question 3: Who is the Prime Minister of India?
1. Donald Trump
2. Narendra Modi
3. Joe Biden
4. K.A.Paul
Your answer (1-4): 2
Correct!

Question 4: What is the capital of India?
1. New Delhi
2. Mumbai
3. Chennai
4. Kolkata
Your answer (1-4): 1
Correct!
```

```
Question 5: Which city is known as the Silicon Valley of India?
1. Bangalore
2. Hyderabad
3. Chennai
4. Pune
Your answer (1-4): 1
Correct!

Question 6: What is the national flower of India?
1. Lotus
2. Rose
3. Sunflower
4. Tulip
Your answer (1-4): 1
Correct!

Quiz completed!
Your total score: 6 out of 6
Would you like to play again? (y/n): n
Thank you for playing the quiz! Goodbye!
```

CONCLUSION:

The quiz program successfully demonstrates the integration of basic C++ concepts, such as arrays, loops, conditionals, and randomization, into an interactive application. By incorporating features like shuffling the order of questions, setting a timer for user input, and providing feedback based on the user's performance, the program enhances the overall user experience. The inclusion of a score system and the option to replay the quiz adds interactivity and encourages engagement. This project provides a solid foundation for further exploration and development of more complex applications, incorporating additional features such as scoring history, multiple difficulty levels, and data storage for user performance tracking. Overall, the project serves as an excellent exercise in C++ programming and offers a fun and educational experience for users.