QUIZ APPILICATION

A PROJECT REPORT

Submitted by

AKASH N (2303811710421006)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"Quiz Application"** is the bonafide work of **AKASH N (2303811710421006)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Ms.M.Saravanan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
ASSOCIATE PROFESSOR	ASSISTANTP ROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

I declare that the project report on "**QUIZ APPLICATION**" is the result of original work doneby us and best of our knowledge, similar work has not been submitted to "**ANNA UNIVERSITY CHENNAI**" for the requirement of Degree of BACHELOR OF ENGINEERING. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201- JAVA PROGRAMMING.**

.

Signature

AKASH N

Place: Samayapuram
Date: 02.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. M.SARAVANAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Quiz Appilication is an interactive educational tool designed to facilitate effective learning and assessment. The application enables users to create customized quizzes, manage questions with multiple-choice options, and assess their knowledge through an engaging interface. It features core functionalities such as quiz creation, user response handling, scoring, and personalized feedback. The architecture is built with modular components, ensuring scalability and maintainability, with key modules including User Interface, Quiz Management, Question Handling, Scoring and Feedback, and a Persistence Layer for data storage.

The application employs Java Swing for graphical user interface development and uses object serialization for efficient data storage and retrieval. During quiz play, user responses are recorded, and scores are calculated based on correctness, followed by motivational feedback to enhance the user experience. The system ensures robust input validation and error handling, making it reliable for both creators and participants. This Quiz Application serves as a versatile and practical solution for educational purposes, promoting knowledge evaluation and interactive learning.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Quiz Application is an interactive educational tool designed to facilitate effective learning and assessment. The application enables users to create customized quizzes, manage questions with multiple-choice options, and assess their knowledge through an engaging interface. It features core functionalities such as quiz creation, user response handling, scoring, and personalized feedback. The architecture is built with modular components, ensuring scalability and maintainability, with key modules including User Interface, Quiz Management, Question Handling, Scoring and Feedback, and a Persistence Layer for data storage. The application employs Java Swing for graphical user interface development and uses object serialization for efficient data storage and retrieval. During quiz play, user responses are recorded, and scores are calculated based on correctness, followed by motivational feedback to enhance the user experience. The system ensures robust input validation and error handling, making it reliable for both creators and participants. This Quiz Application serves as a versatile and practical solution for educational purposes, promoting knowledge evaluation and interactive learning.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The Quiz Application is a dynamic and interactive platform designed to enhance learning and knowledge evaluation in an engaging manner. In the era of digital transformation, where education is increasingly moving online, this application caters to the need for efficient tools to create, manage, and play quizzes. It offers users the flexibility to create personalized quizzes, answer multiple-choice questions, and receive immediate feedback on their performance.

Developed using Java, the application is built with a modular architecture to ensure scalability, maintainability, and robustness. It incorporates key functionalities such as quiz creation, user response handling, scoring, and motivational feedback. The graphical user interface, developed using Java Swing, provides an intuitive experience, while data persistence ensures quizzes can be saved and reused.

With features that promote active learning and assessment, the Quiz Application is well-suited for educational institutions, training programs, and self-assessment needs. It not only supports knowledge retention but also encourages users to strive for continuous improvement, making it an invaluable tool in modern education.

1.1 Objective:

The primary objective of the **Quiz Application** is to serve as an interactive platform that simplifies the process of creating, managing, and participating in quizzes. It is designed to support educational and self-assessment needs by offering a structured and engaging way to evaluate knowledge. The application seeks to:

1. Facilitate the development of personalized quizzes, enabling users to define questions and multiple-choice options based on their specific requirements.
2. Provide a seamless and user-friendly interface for users to interact with the application, ensuring an intuitive experience for both quiz creators and participants.
3. Deliver accurate and real-time scoring to assess user responses and provide detailed feedback to motivate learners and guide them toward improvement.
4. Leverage data persistence techniques to securely store quizzes, allowing them to be retrieved and reused as needed.

5. Promote active learning and engagement by creating a motivating environment for knowledge testing and retention.

Through these objectives, the Quiz Application aims to bridge the gap between technology and education, offering a reliable and efficient tool to support modern learning methodologies. Its robust design ensures scalability and adaptability, making it a valuable resource for educational institutions, training programs, and individuals seeking self-improvement.

1.2 Overview

The **Quiz Application** is a modular and interactive tool developed to streamline the creation, management, and execution of quizzes for educational and self-assessment purposes. Designed using **Java**, the application incorporates multiple functionalities to enhance the user experience while ensuring flexibility and reliability.

At its core, the application allows users to create quizzes with customizable questions and multiple-choice answers. It stores quizzes securely using object serialization, enabling users to retrieve and reuse them whenever needed. During the quiz, users can answer questions in a structured format, and their responses are evaluated in real-time. The application calculates scores based on correctness and provides motivational feedback to encourage continuous learning and improvement.

The system architecture is designed with distinct modules, including User Interface, Quiz Management, Question Handling, Scoring and Feedback, and Persistence Layer. Each module is responsible for specific tasks, ensuring a clear separation of responsibilities and ease of maintenance. The graphical user interface, developed with **Java Swing**, ensures an intuitive and engaging experience for users, whether they are quiz creators or participants.

By offering a robust, user-friendly platform for knowledge testing, the Quiz Application is a valuable resource for educators, trainers, and learners, catering to the growing demand for interactive and effective learning tools.

1.3 Java Programming Concepts

The development of the **Quiz Application** leverages several fundamental and advanced Java programming concepts to ensure its functionality, maintainability, and user interactivity. These concepts include:

1. **Object-Oriented Programming (OOP):**
    - **Classes and Objects:** The application is structured around core classes like Question and Quiz to encapsulate data and behaviors.
    - **Encapsulation:** Data is securely stored within classes, ensuring controlled access through getter and setter methods.
    - **Inheritance:** Extensibility is achieved by structuring related classes hierarchically where applicable.
2. **Java Swing for GUI Development:**
    - The user interface is designed using Swing components such as JFrame, JPanel, JButton, JLabel, and JOptionPane. These provide an intuitive and interactive experience for users.
3. **Event Handling:**
    - User interactions, such as button clicks and option selections, are managed through event listeners like ActionListener to enable dynamic functionality.
4. **File Handling and Object Serialization:**
    - Persistent storage of quizzes is achieved by serializing objects to files, allowing quizzes to be saved and retrieved across sessions.
5. **Java Collections Framework:**
    - Data structures like ArrayList are used to store and manage questions and their options, while HashMap manages multiple quizzes efficiently.
6. **Exception Handling:**
    - Robust exception handling mechanisms ensure the application can gracefully handle errors, such as invalid inputs or file read/write issues.
7. **Control Structures:**
    - Iterative constructs like for and while loops and decision-making constructs like if-else are used extensively for flow control within the application.
8. **Polymorphism (Runtime and Compile-time):**
    - Polymorphism simplifies method implementation and enhances code reusability, especially during interaction with user-defined objects.

These concepts collectively ensure the application is feature-rich, reliable, and user-friendly, meeting its intended purpose for educational and assessment use cases.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The **proposed work** for this project involves the development of a **Quiz Application** designed to cater to educational needs by providing a platform for quiz creation, taking quizzes, scoring, and generating feedback. The main goal is to design a user-friendly application that can handle multiple types of quizzes and provide users with real-time feedback based on their performance. The project will be implemented using **Java programming** to ensure portability and a smooth user experience. Below is a detailed breakdown of the proposed work:

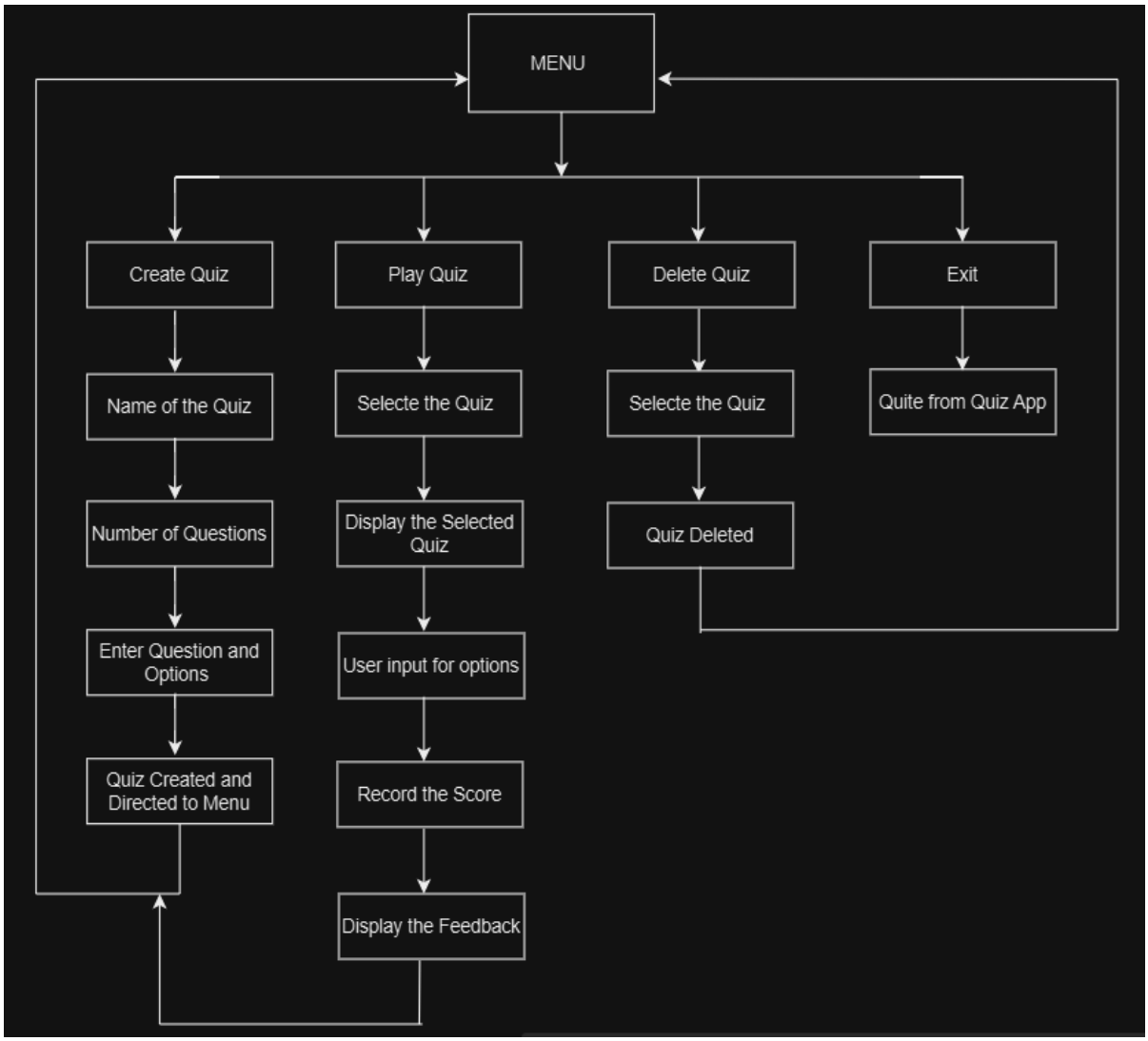
1. **Quiz Creation**:
   - Users will have the ability to create quizzes by entering a set of questions and multiple-choice answers. Each question can have four options, and the user will specify the correct answer for each question.
   - A user-friendly interface will be provided for quiz creation, allowing easy addition of questions and options.
2. **Quiz Playing**:
   - Once a quiz is created, users will be able to play it by selecting one of the available quizzes.
   - During the quiz, users will be presented with one question at a time, with the ability to select an answer. After answering each question, the user will be allowed to proceed to the next question.
   - The interface will ensure that users cannot skip questions without selecting an answer.
3. **Scoring and Feedback**:
   - After completing the quiz, users will receive their score based on the number of correct answers.
   - Motivational feedback will be provided to the user depending on their performance. For example, a perfect score might result in a congratulatory message, while lower scores might prompt users to try again and improve.
4. **Persistence and Data Storage**:
   - All quizzes, including questions and answers, will be stored persistently using **object serialization**. This will ensure that quizzes can be saved and accessed later, even after the application is closed.
   - Users can save multiple quizzes and continue where they left off.
5. **User Interface**:
   - The user interface (UI) will be designed using **Java Swing**, ensuring a responsive and intuitive layout. Key components will include text fields for quiz creation, buttons for user interaction, and panels to display questions and options.

- The UI will be simple yet effective, focusing on usability and smooth navigation.

6. **Error Handling**:
   - The system will include error handling to deal with invalid inputs, such as incomplete quiz creation or incorrect answers, providing useful prompts to guide the user.
7. **Testing and Debugging**:
   - Thorough testing will be conducted throughout the development cycle. This includes **unit testing** for individual components (e.g., scoring, question handling), as well as **integration testing** to ensure all modules work together seamlessly.
8. **Deployment**:
   - The final product will be compiled into an executable JAR file for easy use. Documentation will be provided, including a user guide to explain how to create and play quizzes.

This structured approach will ensure the development of a fully functional and user-friendly Quiz Application that meets the educational needs of students and teachers alike.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Quiz Creation Module

Objective: The purpose of this module is to enable the user to create quizzes by entering a quiz name, adding questions, providing multiple-choice options, and specifying the correct answer.

Description:

- **Input**: The user starts by entering the quiz name. The module then prompts for the number of questions they want to add. For each question, the user provides:
    1. The question text.
    2. Four possible options.
    3. The number corresponding to the correct option.
- **Functionality**:
    - The user can add multiple questions. Each question includes the question text and four options.
    - After providing the options, the user selects the correct option by specifying its number.
    - All question data, including the question, options, and the correct answer, is stored in an object (using the Question class).
    - The quiz is saved in a Quiz object which contains a collection of questions.
- **Validation**:
    - The module ensures that all required fields (question text, options, and correct answer) are filled.
    - It also verifies that the correct answer option is one of the four available options (1-4).
- **Data Handling**:
    - The quiz data is serialized and saved into a file (e.g., "quizzes.ser"), ensuring that the quiz data can be retrieved and played later.

Technologies Used:

- **Java Swing**: For creating the graphical user interface to input quiz data.
- **Java Serialization**: To save and load the quizzes in a serialized file format for persistence.

3.2 Quiz Taking Module

Objective: To enable the user to take the quiz by displaying the questions with multiple-choice options and capturing user responses.

Description:

- **Input**: The user selects a quiz from the available list of created quizzes. Each quiz consists of multiple questions, and the user answers each question by selecting one of the four options.

- **Functionality**:
    - The module displays each question with its options in a user-friendly format.
    - The user selects an answer for each question.
    - Once the user has answered all the questions, the module calculates the score by comparing the user's answers with the correct answers stored in the quiz data.
- **Scoring System**:
    - The score is incremented for every correct answer. After all the questions are answered, the score is calculated as the number of correct answers.
    - The percentage score is also computed, and feedback is provided based on the score.
- **Feedback**:
    - After completing the quiz, the module provides motivational feedback based on the score, such as:
        - **100%**: Perfect score! You're a genius!
        - **80% - 99%**: Great job! Keep it up!
        - **50% - 79%**: Good effort! You can do better with practice!
        - **Below 50%**: Don't worry, keep trying, and you'll improve!

Technologies Used:

- **Java Swing**: For presenting the questions and options to the user.
- **Java Collections (ArrayList)**: For storing questions and answers dynamically.

3.3 Quiz Management Module

Objective: To allow users to manage quizzes by viewing, deleting, or modifying them.

Description:

- **Input**: The user is given options to manage the quizzes after the application has been started.
- **Functionality**:
    - **View Quizzes**: The user can view all available quizzes, including the default one and any user-created quizzes.
    - **Delete Quizzes**: The user can delete quizzes that are no longer needed. However, the default quiz cannot be deleted.
    - **Modify Quizzes** (optional feature for future expansion): Although modification is not implemented in this version, this module can be expanded in future versions to allow editing of existing quizzes.
- **Validation**:
    - The module ensures that only non-default quizzes can be deleted.
- **Data Handling**:
    - Any changes made (like deleting a quiz) are immediately reflected in the serialized file after saving.

Technologies Used:

- **Java Swing**: For creating buttons and list selections to manage quizzes.
- **File I/O (Serialization)**: To update the list of quizzes in the persistent storage.

3.4 Feedback and Scoring Module

Objective: To provide feedback to the user based on their performance after taking the quiz.
Description:

- **Input**: The user's score (number of correct answers) is used to determine the feedback.
- **Functionality**:
  - After the quiz is completed, the module calculates the score percentage and provides feedback.
  - Feedback messages are designed to motivate the user and encourage further participation. The messages vary depending on the score range:
    - **100%**: "Perfect score! You're a genius! "
    - **80% - 99%**: "Great job! You scored really well. Keep it up! "
    - **50% - 79%**: "Good effort! You can do even better with some more practice! "
    - **Below 50%**: "Don't worry, keep trying, and you'll improve! "
- **Technologies Used**:
  - **Java Swing**: For displaying the feedback message in a dialog box after the quiz is completed.
  - **Mathematical Calculations**: To compute the percentage score from the total number of correct answers.

3.5 Data Storage and Retrieval Module

Objective: To manage the saving and loading of quizzes for persistence across application sessions.
Description:

- **Input**: When a quiz is created or updated, the data is serialized and saved into a file.
- **Functionality**:
  - **Saving Quizzes**: The module serializes the quiz data into a file (e.g., "quizzes.ser") for later use. This allows users to access their saved quizzes even after restarting the application.
  - **Loading Quizzes**: On starting the application, the saved quizzes are deserialized and loaded into memory, making them available for viewing, taking, or deleting.
- **Data Integrity**:
  - The module ensures that all quizzes are stored and retrieved correctly, maintaining data consistency across sessions.
- **Error Handling**:
  - The module handles errors in file operations, such as file not found or serialization errors, and notifies the user if there is an issue loading or saving data.

Technologies Used:

- **Java I/O (File Handling)**: For reading and writing quiz data to and from files.
- **Java Serialization**: For storing and retrieving quiz objects in a persistent file format.

3.6 User Interface (UI) Module

Objective: The purpose of this module is to provide a user-friendly interface for interacting with the quiz application.

Description:

- **Input**: The user interacts with the graphical interface for all operations such as quiz creation, taking quizzes, managing quizzes, and viewing feedback.
- **Functionality**:
    - **Main Screen**: Displays buttons for creating, playing, and managing quizzes, as well as for exiting the application.
    - **Dialogs and Forms**: Provides input fields for quiz creation, question entries, and answer selections.
    - **Question Display**: Displays each question and its options in a clear format, with the ability to select an option and submit the answer.

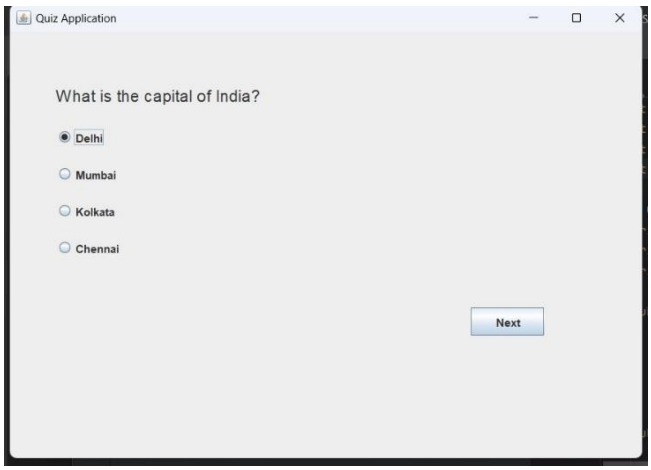
Technologies Used:

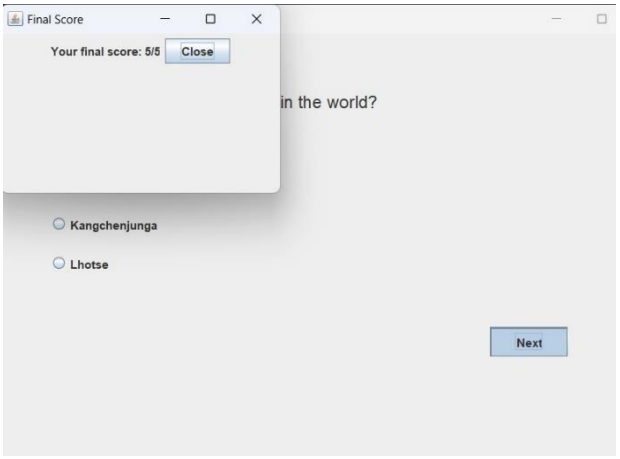
- **Java Swing**: The entire user interface is built using Java Swing components like JFrame, JButton, JTextField, JRadioButton, JOptionPane, etc., to create an interactive GUI.

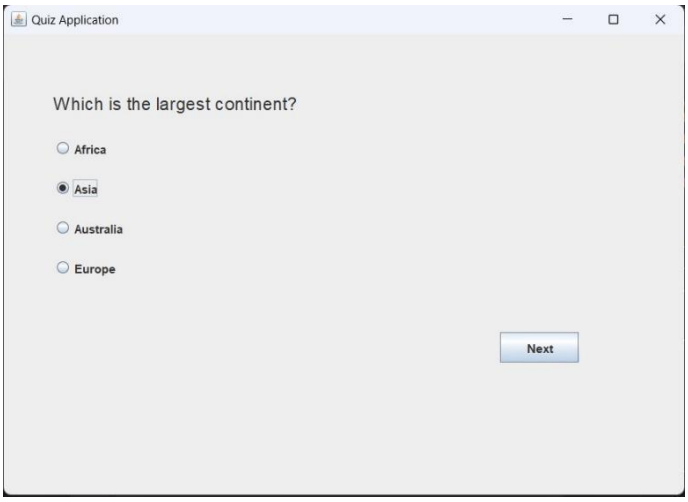
These modules work together to provide a complete and functional quiz application. Each module has specific responsibilities, from quiz creation to management and taking the quiz, providing a user-friendly and interactive experience. The modular design ensures that each component can be independently modified or expanded in the future.

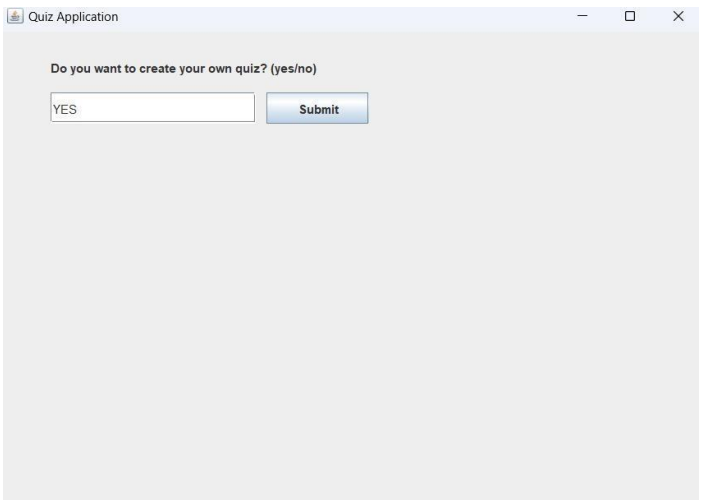
CHAPTER 4

RESULTS AND DISCUSSION









CHAPTER 5

CONCLUSION

The Quiz Application successfully addresses the need for an interactive and educational platform to create, manage, and take quizzes efficiently. By implementing robust Java programming concepts and adhering to modular architecture, the application ensures flexibility, user-friendliness, and scalability. Its key features, such as customizable quiz creation, instant feedback, and scoring, provide a comprehensive learning experience. This project not only demonstrates the practical application of programming and design principles but also serves as a foundation for further enhancements, such as integrating advanced analytics or multi-user capabilities. Overall, the Quiz Application achieves its objectives and highlights the potential of software in enhancing education.

REFERENCES

1. Herbert Schildt, *Java: The Complete Reference*, McGraw-Hill Education, 11th Edition, 2019.
2. Cay S. Horstmann, *Core Java Volume I – Fundamentals*, Pearson Education, 11th Edition, 2018.
3. Bruce Eckel, *Thinking in Java*, Prentice Hall, 4th Edition, 2006.
4. Oracle Java Documentation - https://docs.oracle.com/javase/
5. Official Swing Tutorials - https://docs.oracle.com/javase/tutorial/uiswing/
6. Oracle Serialization Guide - https://docs.oracle.com/javase/tutorial/jndi/

These references provide detailed insights into Java programming, user interface design with Swing, and serialization concepts used in the project.

APPENDIX (Source Code)

```
import java.util.ArrayList;
import java.util.Scanner;

class Question {
    private String questionText;
    private ArrayList<String> options;
    private int correctOption;

    public Question(String questionText, ArrayList<String> options, int correctOption) {
        this.questionText = questionText;
        this.options = options;
        this.correctOption = correctOption;
    }

    public void displayQuestion() {
        System.out.println("\n" + questionText);
        for (int i = 0; i < options.size(); i++) {
            System.out.println((i + 1) + ". " + options.get(i));
        }
    }

    public boolean isCorrect(int userAnswer) {
        return userAnswer == correctOption;
    }

    // Getter method to access options
    public ArrayList<String> getOptions() {
        return options;
    }
}

class Quiz {
    private ArrayList<Question> questions;
    private int score;

    public Quiz() {
        questions = new ArrayList<>();
        score = 0;
    }

    public void addQuestion(Question question) {
        questions.add(question);
```

```
    }

    public void start() {
        Scanner scanner = new Scanner(System.in);
        for (Question question : questions) {
            question.displayQuestion();
            int userAnswer = 0;
            boolean validInput = false;

            // Loop until a valid input is received
            while (!validInput) {
                System.out.print("Your answer (Enter the option number): ");
                userAnswer = scanner.nextInt();
                if (userAnswer < 1 || userAnswer > question.getOptions().size()) {
                    System.out.println("Warning: Please enter a valid option number.");
                } else {
                    validInput = true;
                }
            }

            if (question.isCorrect(userAnswer)) {
                System.out.println("Correct!");
                score++;
            } else {
                System.out.println("Incorrect.");
            }
        }
        System.out.println("\nQuiz Over!");
        System.out.println("Your final score: " + score + "/" + questions.size());
        generateFeedback(score, questions.size());
    }

    // Method to generate feedback based on the score
    public void generateFeedback(int score, int totalQuestions) {
        double percentage = ((double) score / totalQuestions) * 100;
        System.out.println("\n--- Feedback ---");
        if (percentage == 100) {
            System.out.println("Outstanding! You got a perfect score. Keep up the amazing
work!");
        } else if (percentage >= 80) {
            System.out.println("Great job! You're doing really well. Keep striving for
excellence!");
        } else if (percentage >= 50) {
            System.out.println("Good effort! You're on the right track. With a bit more practice,
```

```
you'll get even better!");
        } else {
            System.out.println("Don't worry! Every step counts. Keep learning and improving,
you're doing great!");
        }
    }

    public void loadDefaultGKQuestions() {
        ArrayList<String> options1 = new ArrayList<>();
        options1.add("Delhi");
        options1.add("Mumbai");
        options1.add("Kolkata");
        options1.add("Chennai");
        addQuestion(new Question("What is the capital of India?", options1, 1));

        ArrayList<String> options2 = new ArrayList<>();
        options2.add("Mars");
        options2.add("Venus");
        options2.add("Jupiter");
        options2.add("Earth");
        addQuestion(new Question("Which planet is known as the 'Red Planet'?", options2, 1));

        ArrayList<String> options3 = new ArrayList<>();
        options3.add("Africa");
        options3.add("Asia");
        options3.add("Australia");
        options3.add("Europe");
        addQuestion(new Question("Which is the largest continent?", options3, 2));

        ArrayList<String> options4 = new ArrayList<>();
        options4.add("Nile");
        options4.add("Amazon");
        options4.add("Ganga");
        options4.add("Mississippi");
        addQuestion(new Question("What is the longest river in the world?", options4, 1));

        ArrayList<String> options5 = new ArrayList<>();
        options5.add("Mount Everest");
        options5.add("K2");
        options5.add("Kangchenjunga");
        options5.add("Lhotse");
        addQuestion(new Question("What is the highest mountain in the world?", options5, 1));
    }
}
```

```
public class QuizApplication {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Quiz quiz = new Quiz();

        System.out.println("Do you want to create your own quiz? (yes/no)");
        String createQuiz = scanner.nextLine();

        if (createQuiz.equalsIgnoreCase("yes")) {
            System.out.println("Create a Quiz");
            System.out.print("Enter the number of questions: ");
            int numQuestions = scanner.nextInt();
            scanner.nextLine();  // Consume newline

            for (int i = 0; i < numQuestions; i++) {
                System.out.println("\nEnter details for Question " + (i + 1) + ":");
                System.out.print("Question: ");
                String questionText = scanner.nextLine();

                ArrayList<String> options = new ArrayList<>();
                System.out.print("How many options? ");
                int numOptions = scanner.nextInt();
                scanner.nextLine();  // Consume newline

                for (int j = 0; j < numOptions; j++) {
                    System.out.print("Option " + (j + 1) + ": ");
                    String option = scanner.nextLine();
                    options.add(option);
                }

                System.out.print("Enter the correct option number: ");
                int correctOption = scanner.nextInt();
                scanner.nextLine();  // Consume newline

                quiz.addQuestion(new Question(questionText, options, correctOption));
            }

        } else {
            System.out.println("\nLoading the default General Knowledge Quiz...");
            quiz.loadDefaultGKQuestions();
        }

        System.out.println("\nStarting the Quiz...");
```

```
        quiz.start();
    }
}
```