

QUIZ APPLICATION



A PROJECT REPORT

Submitted by

DHANVARSHINI T(8115U23EC017)

in partial fulfillment of requirements for the award of the course

EGB1201 – JAVA PROGRAMMING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

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

SAMAYAPURAM – 621 112

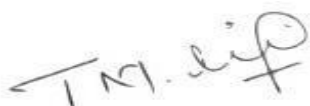
DECEMBER - 2024

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING
(AUTONOMOUS)**

SAMAYAPURAM-621 112

BONAFIDE CERTIFICATE

Certified that this project report “**QUIZ APPLICATION**” is the bonafide work of **DHANVARSHINI T(8115U23EC017)**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.



SIGNATURE

Dr. T. M. NITHYA, M.E., Ph.D.,

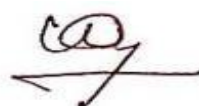
HEAD OF THE DEPARTMENT

ASSOCIATE PROFESSOR

Department of CSE

K.Ramakrishnan College of Engineering
(Autonomous)

Samayapuram-621112.



SIGNATURE

Mr.V.KUMARARAJA, M.E.,(Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Engineering
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 06/12/24



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I jointly declare that the project report on “**QUIZ APPLICATION**” is the result of original work done by 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 fulfillment of the requirement of the award of the course **EGB1201- JAVA PROGRAMMING**

Signature



DHANVARSHINI T

Place: Samayapuram

Date:06/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, “**K.RAMAKRISHNAN COLLEGE OF ENGINEERING (Autonomous)**”, for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided 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 our project and offering an adequate duration to complete it.

I would like to thank **Dr. D. SRINIVASAN, M.E., Ph.D., FIE., MIIW., MISTE., MISAE., C. Engg.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr. T.M.NITHYA., M.E., Ph.D.**, Head of the Department of Computer Science and Engineering, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mr.V.KUMARARAJA., M.E.**, Department of Computer Science and Engineering, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the 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 achieve a prominent position among the top technical institutions.

MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

VISION OF DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting quality education.

MISSION OF DEPARTMENT

M1: To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of Computer Science and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

PROGRAM EDUCATIONAL OBJECTIVES

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.
- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

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 GUI Quiz Application with Login is an interactive Java-based program that combines secure user authentication with a dynamic multiple-choice quiz system. Developed using Java Swing, the application provides functionalities for user registration, login, and participation in quizzes, demonstrating the effective use of object-oriented programming, event handling, and graphical user interface design. Users can securely register and log in, ensuring personalized access to the quiz platform. The quiz interface is visually engaging, featuring dynamic multiple-choice questions, with real-time score calculation and feedback at the end of the quiz. Designed with scalability in mind, the application allows for easy addition of new questions or features, such as timed quizzes, leaderboards, and category selection. The program employs modular design principles, making it maintainable and extendable, and it leverages event listeners for seamless user interactions. This project not only serves as a practical application of Java programming skills but also highlights the integration of theoretical knowledge into real-world software solutions. Its versatility makes it ideal for educational platforms, corporate training programs, or personal learning applications, with future potential for enhancements like password recovery, analytics, and database integration.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGENO.
	ABSTRACT	ix
1	INTRODUCTION	
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming Concepts	2
2	PROJECT METHODOLOGY	
	2.1 Proposed Work	5
	2.2 Block Diagram	7
3	MODULES	
	3.1 User Registration	9
	3.2 User Login	9
	3.3 Display Quiz Interface	9
	3.4 Display Question	10
	3.5 Calculate and Display Results	10
	3.6 Handle User Actions (Login / Register)	10
	3.7 Question Initialization	10
	3.8 Logout Functionality	11
4	CONCLUSION & FUTURE SCOPE	
	4.1 Conclusion	12
	4.2 Future Scope	13
	APPENDIX A (SOURCE CODE)	14
	APPENDIX B (SCREENSHOTS)	20
	REFERNCES	23

CHAPTER 1

INTRODUCTION

12.1 OBJECTIVES

The **GUI Quiz Application with Login** is a Java-based program designed to provide an engaging platform for learning and evaluation through interactive quizzes. Built using Java Swing, it features secure user authentication, allowing users to register, log in, and access a dynamic multiple-choice quiz system. The application calculates scores in real time and displays results at the end of the quiz. Its modular design enables easy customization and future enhancements, such as adding more questions or integrating user progress tracking. This project highlights the practical application of Java in developing user-friendly, event-driven graphical applications for educational and recreational purposes.

1. User Authentication
2. Dynamic Quiz Management
3. Graphical User Interface
4. Real-Time Score Calculation
5. Modular Architecture

12.2 OVERVIEW

The project is a Java-based Quiz Application designed with a graphical user interface (GUI) to provide an interactive and engaging experience for users. It incorporates a login and registration system to ensure secure access, allowing users to create accounts and log in to participate in quizzes. The

application features multiple-choice questions with real-time score tracking, where users can submit their answers and receive immediate feedback.

The program utilizes Java Swing for GUI development, making use of components such as buttons, labels, and text fields to create an intuitive interface. Hash Maps are used for storing user credentials, enabling quick lookups during login. The application's modular design allows easy extension, enabling future features like user progress tracking, more quizzes, and different question formats. The main objective of the project is to showcase practical applications of Java programming concepts, including object-oriented programming, event handling, data structures, and GUI development, while providing an accessible and enjoyable platform for quiz-based learning.

12.3 JAVA PROGRAMMING CONCEPTS

1.Object-Oriented Programming(OOP):

The program follows Object-Oriented Programming principles, creating classes such as Question and GuiQuizAppWithLogin to organize and encapsulate data and functionality. The Question class holds details about each quiz question, such as the text, options, and correct answer. The GuiQuizAppWithLogin class manages the overall flow of the application, from login to quiz display and results. This structure promotes code reuse, modularity, and scalability.

2. Event-Handling:

Java Swing's event-handling mechanism is crucial to the interactive nature of the application. It listens for user inputs, such as button clicks or selection changes, and responds accordingly. For instance, the ActionListener attached to buttons like "Login," "Submit Answer," and "Register" ensures that specific actions are taken when users interact with the interface. Event handling is central to providing real-time feedback and ensuring the application responds promptly to user actions.

3. Graphical User-Interface(GUI):

The program uses Java Swing for building a responsive GUI. Components like JFrame for window creation, JLabel for displaying text, JRadioButton for quiz options, and JButton for interactive buttons are employed to build a simple yet effective user interface. Swing's flexibility allows for easy customization and organization of components, creating an intuitive environment for the user to interact with the quiz application.

4. HashMap:

A HashMap is used to store user credentials (username and password) for login authentication. This data structure is efficient for searching, as it allows for constant-time lookups. When the user inputs a username and password, the program quickly checks the HashMap to determine if the credentials are valid. This ensures a fast and secure login process, providing a good user experience.

5. Arrays:

Arrays are used to store the quiz questions and their possible answers, while the Question class organizes each question into an object with its properties: the question text, options, and the correct answer.

6. String Handling:

The program uses string methods to capture the username, password, and quiz responses from the user. It compares these inputs with stored values (such as credentials in the HashMap) to authenticate the user and check quiz answers. Proper handling of strings ensures that user input is validated and processed correctly.

7. Exception-Handling:

The application incorporates basic exception handling, especially in areas like login authentication and input validation. If the user enters incorrect credentials or leaves a field blank, error messages are shown using JOptionPane. This prevents the application from crashing and provides the user

with helpful feedback on their actions, improving the overall user experience.

8. Flow-Control:

The program uses control flow structures like if statements and loops to manage the progression of the application. For example, the flow control logic in the login process ensures that only correct usernames and passwords are accepted. Similarly, the quiz display logic uses loops to iterate through the questions and update the interface based on the user's actions, ensuring smooth navigation from one question to the next.

These concepts work together to create a functional, interactive, and user-friendly quiz application that handles login, question navigation, and result display while ensuring a smooth experience for the user.

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed work aims to develop a **GUI-based Quiz Application with Login** that provides a secure and interactive platform for users to test their knowledge. The methodology focuses on combining user authentication with a structured quiz system to ensure a seamless experience. Below is the proposed approach:

1. User Authentication

Implement a login and registration system using **HashMap** to store and validate user credentials securely. Ensure password protection and prevent unauthorized access to the quiz.

2. Quiz Interface

Create a user-friendly graphical interface using **Java Swing** components like JFrame, JLabel, JRadioButton, and JButton. Organize the layout for better navigation and display using layout managers such as Flow Layout and Grid Layout.

3. Question Bank

Store quiz questions and options as objects in an array. Use a Question class to encapsulate details like the question text, answer options, and the correct answer.

4. Quiz Flow

Display questions one by one, allowing users to select answers using radio buttons. Navigate through questions and calculate the score based on user responses.

5. Result Evaluation

Show the final score after the quiz is completed. Provide options to restart the

quiz or log out, enabling flexibility in usage.

6.Error Handling

Include validations for login, registration, and quiz interactions to handle incorrect inputs gracefully. Ensure the application does not crash during invalid operations.

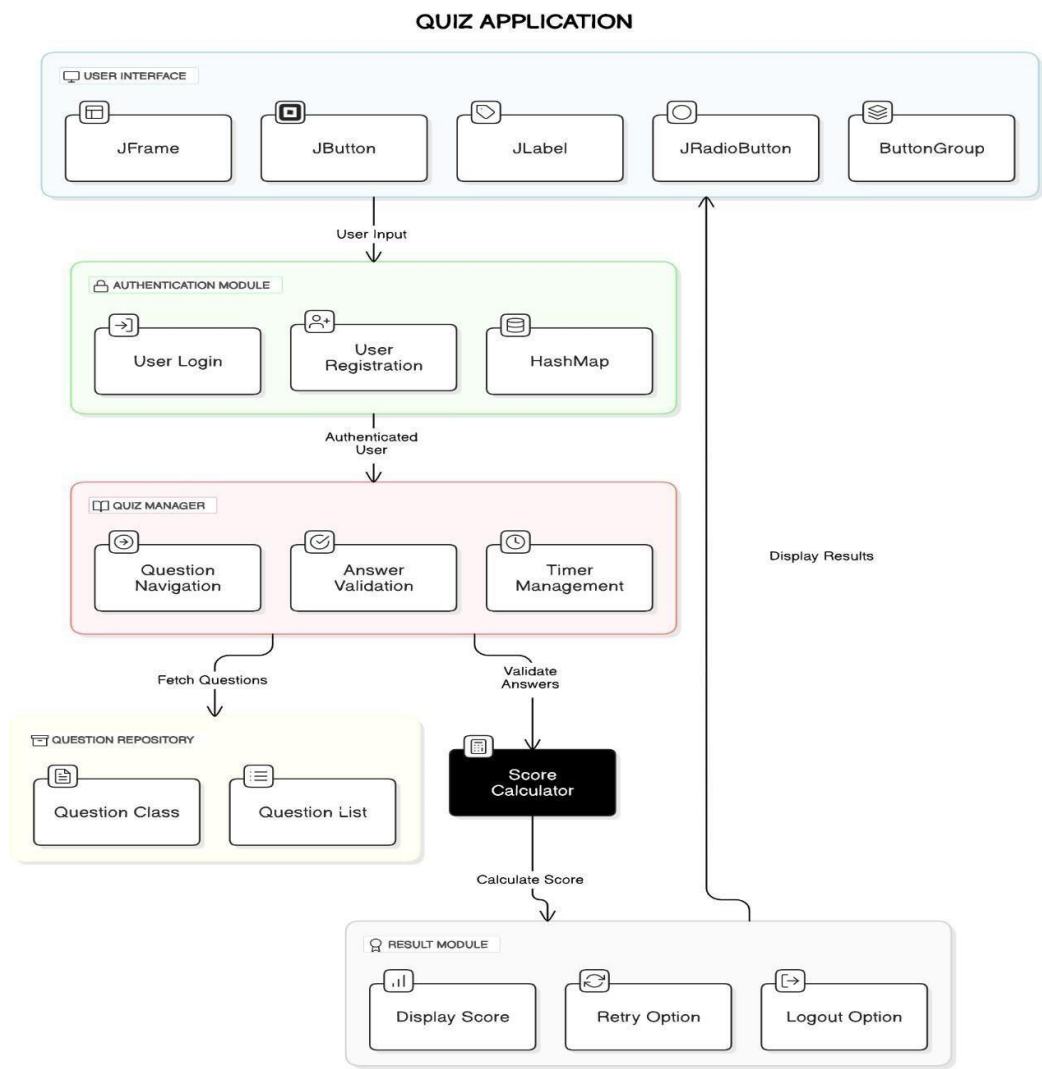
7.Logout and Session Management

Allow users to log out securely and clear their session data to ensure privacy.

This methodology ensures the development of a robust, efficient, and interactive quiz application that meets the objectives of user engagement and secure access.

2.2 BLOCK DIAGRAM

The architecture diagram provides a visual representation of the design and functionality of the Java-based Quiz Application. It highlights the core components, modules, and their interactions to illustrate the application's workflow. The system is divided into several modules, each responsible for specific tasks, ensuring modularity, scalability, and clarity in the design.



The Quiz Application Architecture consists of the following key modules:

1. User Interface Module

This module is built using Java Swing components such as JFrame, JButton, JLabel, JRadioButton, and ButtonGroup. It handles user input, displays quiz questions, and provides navigation options for the quiz.

2. Authentication Module

Responsible for managing user login and registration. A HashMap is used for efficient storage and retrieval of user credentials.

3. Quiz Manager Module

Question Navigation: Allows users to move through questions.

Answer Validation: Verifies user-selected answers.

Timer Management: Ensures time constraints are applied where necessary.

4. Question Repository

Stores quiz data, including questions and their options, using a Question class and Question List. Provides dynamic question retrieval for the quiz.

5. Score Calculator

Validates answers submitted by users. Calculates and displays the final score based on correct responses.

6. Result Module

Displays quiz results to users. Offers options to retry the quiz or logout from the application.

This modular architecture ensures smooth interaction between the user and the system while providing a flexible design for future enhancements.

CHAPTER 3

MODULE DESCRIPTION

3.1 USER REGISTRATION

- **Function Name:** 'showRegisterScreen()'
- **Description:** Displays the registration interface for new users. Prompts the user to input a unique username and password. Validates the input to ensure fields are not empty and the username is not already taken. Updates the user Credentials HashMap to store the new user's login information securely and confirms successful registration.

3.2 USER LOGIN

- **Function name:** 'showLoginScreen()'
- **Description:** Displays the login interface for existing users. Prompts the user to input their username and password. Validates the credentials against the user Credentials HashMap. If authentication succeeds, grants access to the quiz; otherwise, shows an error message.

3.3 DISPLAY QUIZ INTERFACE

- **Function name:** 'displayQuiz()'
- **Description:-** Initializes and displays the quiz interface. Sets up the JFrame for showing questions and their corresponding options. Prepares for dynamic updates of questions and user interactions during the quiz.

3.4 DISPLAY QUESTION

- **Function name:** 'displayQuestion(JFrame frame)'
- **Description:** Displays the current question and its options using JLabels and JRadioButtons. Handles user input by allowing selection of one option. On submission, validates the answer and updates the score if correct. Advances to the next question or ends the quiz when all questions are completed.

3.5 CALCULATE AND DISPLAY RESULTS

- **Function Name:** 'showResults(JFrame frame)'
- **Description:** Displays the final score and feedback upon quiz completion. Provides the option to log out and return to the login/register interface. Ensures the application resets the state for subsequent logins.

3.6 HANDLE USER ACTIONS(LOGIN/REGISTER)

- **Function Name:** 'showLoginOrRegisterScreen()'
- **Description:** Displays the main menu for user authentication. Offers two buttons to either log in as an existing user or register as a new user. Directs the user to the appropriate interface based on their choice.

3.7 QUESTION INITIALIZATION

- **Function Name:** 'Question(constructor)'
- **Description:** Initializes a Question object with a question text, four options, and the correct answer index. Encapsulates the data for individual questions to ensure modularity and ease of access.

3.8 LOGOUT FUNCTIONALITY

- **Function Name:** 'logout()'
- **Description:** Allows the user to log out after completing the quiz. Resets the application state, including the score and question index, and redirects the user to the login/register screen for reauthentication.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The GUI Quiz Application with Login successfully demonstrates a practical integration of user authentication and interactive quiz functionalities, leveraging Java Swing for a dynamic and user-friendly interface. Through the modular design, users can easily register, log in, and participate in a quiz, with scores tracked in real-time. The program ensures secure user authentication by validating credentials and offers a smooth experience for both new and returning users. It effectively utilizes object-oriented programming principles, event handling, and GUI design, offering an excellent foundation for further extensions, such as adding more quiz questions, enhancing user profiles, or integrating database storage. Overall, the project showcases the effective use of Java in creating interactive, user-driven applications.

Additionally, the program's design allows for easy scalability and customization, enabling the addition of more features such as time limits, different types of questions, or user feedback mechanisms. The separation of concerns in the code—such as handling user authentication, quiz logic, and display—makes the application modular and easy to maintain. The successful implementation of the quiz functionality, coupled with a secure login system, makes this project a solid foundation for more advanced applications, including the potential to add features like storing user progress, generating reports, or even creating a multi-user environment. Overall, it effectively highlights key Java programming concepts, making it a valuable learning tool for both developers and students.

4.2 FUTURE SCOPE

The future scope of the GUI Quiz Application with Login is vast, offering numerous opportunities for enhancement and expansion. One key area for improvement is the integration of a database system, such as MySQL or MongoDB, to securely store user credentials, quiz scores, and user activity logs. This would enable features like password recovery, multiple user profiles, and persistent score tracking across sessions. Additionally, the application can be extended to support more complex question types, such as true/false, fill-in-the-blank, or image-based questions, providing a more engaging and varied user experience. Implementing a timer for each question or a countdown for the entire quiz could add an element of challenge and urgency to the application, making it more dynamic. Another possible enhancement is the addition of a leaderboard to display top scorers, fostering competition among users. Furthermore, the system could be expanded to allow users to create and share their own quizzes, enhancing the application's community engagement. Incorporating analytics to track user performance over time, along with personalized feedback, could also be valuable for educational purposes. On the technical side, transitioning the application to a web-based platform using technologies like HTML, CSS, and JavaScript (along with a backend framework like Spring Boot or Node.js) would make it accessible across different devices, increasing its reach and usability. Lastly, incorporating mobile app development (e.g., using Java for Android or Flutter) could extend the application's scope to the growing mobile user base. These advancements would significantly enhance the app's functionality and user experience, making it a comprehensive and versatile quiz platform.

APPENDICES

APPENDIX A-SOURCE CODE

```
package quiz;
import javax.swing.*.*;
public class GuiQuizAppWithLogin{
    private static int score = 0;
    private static int currentQuestionIndex = 0;
    private static final HashMap<String, String> userCredentials = new
    HashMap<>();
    private static final Question[] questions = {
        new Question("What is 2+2?", "3", "4", "5", "6", 2),
        new Question("Capital of France?", "Berlin", "Madrid", "Paris",
"London", 3)
    };
    private static String loggedInUser = "";
    public static void main (String[] args) {
        userCredentials.put("user", "pass");
        showLoginOrRegisterScreen();
    }
    private static void showLoginOrRegisterScreen() {
        JFrame loginFrame = new JFrame("Login or Register");
        loginFrame.setDefaultCloseOperation
(JFrame.EXIT_ON_CLOSE);
        loginFrame.setSize(300, 200);
        loginFrame.setLayout (new FlowLayout());
        JButton loginButton = new JButton("Login");
        JButton registerButton = new JButton("Register");
        loginButton.addActionListener(e -> {
            loginFrame.dispose();
            showLoginScreen();
        });
        registerButton.addActionListener(e -> {
            loginFrame.dispose();
            showRegisterScreen();
        });
        loginFrame.add(new JLabel("Welcome to the Quiz App!"));
        loginFrame.add(loginButton);
        loginFrame.add(registerButton);

        loginFrame.setVisible(true);
    }
}
```



```

private static void showLoginScreen() {
    JFrame loginFrame = new JFrame("Login");
    loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    loginFrame.setSize(300, 200);
    loginFrame.setLayout(new FlowLayout());
    JLabel usernameLabel = new JLabel("Username:");
    JTextField usernameField = new JTextField(15);
    JLabel passwordLabel = new JLabel("Password:");
    JPasswordField passwordField = new JPasswordField(15);
    JButton loginButton = new JButton("Login");
    loginButton.addActionListener(e -> {
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());

        if (userCredentials.containsKey(username) &&
            userCredentials.get(username).equals(password)) {
            loggedInUser = username;
            loginFrame.dispose();
            displayQuiz();
        } else {
            JOptionPane.showMessageDialog(loginFrame, "Invalid
            username or password", "Login Failed",
            JOptionPane.ERROR_MESSAGE);
        }
    });
    loginFrame.add(usernameLabel);
    loginFrame.add(usernameField);
    loginFrame.add(passwordLabel);
    loginFrame.add(passwordField);
    loginFrame.add(loginButton);

    loginFrame.setVisible(true);
}

private static void showRegisterScreen() {
    JFrame registerFrame = new JFrame("Register");
    registerFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    registerFrame.setSize(300, 200);
    registerFrame.setLayout(new FlowLayout());

    JLabel usernameLabel = new JLabel("Choose a Username:");
    JTextField usernameField = new JTextField(15);
    JLabel passwordLabel = new JLabel("Choose a Password:");
    JPasswordField passwordField = new JPasswordField(15);
    JButton registerButton = new JButton("Register");

```

```

registerButton.addActionListener(e -> {
    String username = usernameField.getText();
    String password = new String(passwordField.getPassword());

    if (username.isEmpty() || password.isEmpty()) {
        JOptionPane.showMessageDialog(registerFrame, "Username and
password cannot be empty.", "Registration Failed",
JOptionPane.ERROR_MESSAGE);
    } else if (userCredentials.containsKey(username)) {
        JOptionPane.showMessageDialog(registerFrame, "Username
already exists.", "Registration Failed",
JOptionPane.ERROR_MESSAGE);
    } else {
        userCredentials.put(username, password);
        JOptionPane.showMessageDialog(registerFrame, "Registration
successful! Please login.", "Success",
JOptionPane.INFORMATION_MESSAGE);
        registerFrame.dispose();
        showLoginScreen();
    }
});

registerFrame.add(usernameLabel);
registerFrame.add(usernameField);
registerFrame.add(passwordLabel);
registerFrame.add(passwordField);
registerFrame.add(registerButton);

registerFrame.setVisible(true);
}

private static void displayQuiz() {
    JFrame frame = new JFrame("Quiz Application");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(500, 400);
    frame.setLayout(new BorderLayout());

    displayQuestion(frame);

    frame.setVisible(true);
}

private static void displayQuestion(JFrame frame) {
    if (currentQuestionIndex < questions.length) {
        Question question = questions[currentQuestionIndex];

```

```

frame.getContentPane().removeAll();

JLabel questionLabel = new JLabel("Question: " +
question.getText());
questionLabel.setFont(new Font("Arial", Font.BOLD, 14));
frame.add(questionLabel, BorderLayout.NORTH);

JRadioButton option1Btn = new
JRadioButton(question.getOption1());
JRadioButton option2Btn = new
JRadioButton(question.getOption2());
JRadioButton option3Btn = new
JRadioButton(question.getOption3());
JRadioButton option4Btn = new
JRadioButton(question.getOption4());

ButtonGroup optionsGroup = new ButtonGroup();
optionsGroup.add(option1Btn);
optionsGroup.add(option2Btn);
optionsGroup.add(option3Btn);
optionsGroup.add(option4Btn);

JPanel optionsPanel = new JPanel(new GridLayout(4, 1));
optionsPanel.add(option1Btn);
optionsPanel.add(option2Btn);
optionsPanel.add(option3Btn);
optionsPanel.add(option4Btn);
frame.add(optionsPanel, BorderLayout.CENTER);

JButton submitButton = new JButton("Submit Answer");
submitButton.addActionListener(e -> {
    if ((question.getCorrectAnswer() == 1 &&
option1Btn.isSelected()) ||
        (question.getCorrectAnswer() == 2 &&
option2Btn.isSelected()) ||
        (question.getCorrectAnswer() == 3 &&
option3Btn.isSelected()) ||
        (question.getCorrectAnswer() == 4 &&
option4Btn.isSelected())) {
        score++;
    }
    currentQuestionIndex++;
    displayQuestion(frame);
});

```

```

        frame.add(submitButton, BorderLayout.SOUTH);

        frame.revalidate();
        frame.repaint();
    } else {
        showResults(frame);
    }
}

private static void showResults(JFrame frame) {
    frame.getContentPane().removeAll();

    JLabel resultLabel = new JLabel("Quiz Completed! Your Score: " +
        score + "/" + questions.length);
    resultLabel.setFont(new Font("Arial", Font.BOLD, 16));
    resultLabel.setHorizontalAlignment(SwingConstants.CENTER);
    frame.add(resultLabel, BorderLayout.CENTER);

    JButton logoutButton = new JButton("Logout");
    logoutButton.addActionListener(e -> {
        int confirm = JOptionPane.showConfirmDialog(frame, "Are you
            sure you want to logout?", "Logout",
            JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            score = 0;
            currentQuestionIndex = 0;
            loggedInUser = "";
            frame.dispose();
            showLoginOrRegisterScreen();
        }
    });
    frame.add(logoutButton, BorderLayout.SOUTH);

    frame.revalidate();
    frame.repaint();
}

class Question {
    private final String text;
    private final String option1;
    private final String option2;
    private final String option3;
    private final String option4;
    private final int correctAnswer;
}

```

```

public Question(String text, String option1, String option2, String
    option3, String option4, int correctAnswer) {
    this.text = text;
    this.option1 = option1;
    this.option2 = option2;
    this.option3 = option3;
    this.option4 = option4;
    this.correctAnswer = correctAnswer;
}

public String getText() {
    return text;
}
public String getOption1() {
    return option1;
}
public String getOption2() {
    return option2;
}
public String getOption3() {
    return option3;
}
public String getOption4() {
    return option4;
}
public int getCorrectAnswer() {
    return correctAnswer;
}
}

```

APPENDIX B (SCREENSHOTS)

1. Register / Login Page

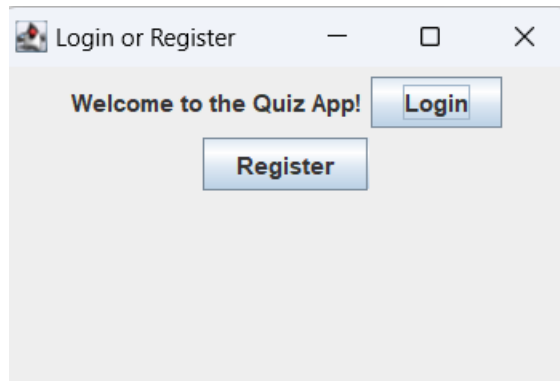


Figure 1: Register / Login the app

2. Registration

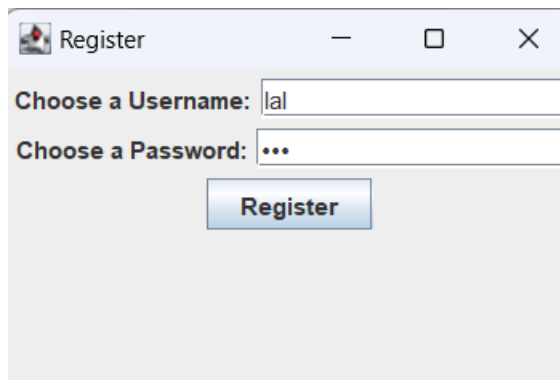


Figure 2: Registration Page

3. Registration Successful

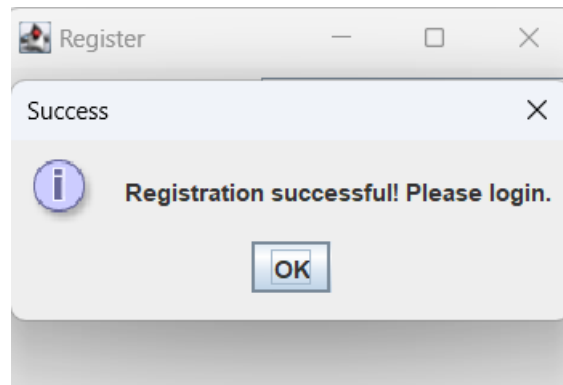


Figure 3: Registration Successfully Completed

4. Login Page

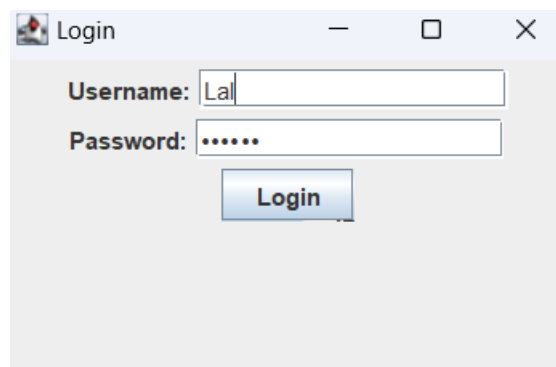


Figure 4: Login Page

5. Question Display

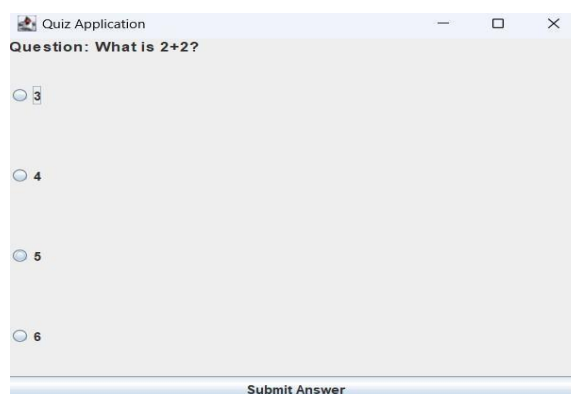


Figure 5: Question Page

6. Result

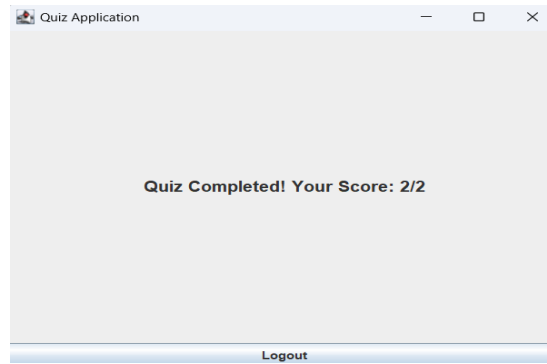


Figure 6: Result for the Quiz

REFERENCES

1. Java The Complete Reference, 13E Paperback-21 March 2024 by Herbert Schildt
2. Core Java Volume I – Fundamentals was published in 2018 by Cay S. Horstmann and Gary Cornell
3. Head First Java was published in 2005 by Kathy Sierra and Bert Bates
4. Java Programming: From Problem Analysis to Program Design was published in 2011 by D.S. Malik
5. Java Swing was published in 2002 by Marc Loy, Robert Eckstein, Dave Wood, James Elliott, and Brian Cole
6. Effective Java was published in 2018 by Joshua Bloch
7. Java Programming and Data Structures was published in 2021 by Y. Daniel Liang
8. <https://github.com/topics/quiz-application?l=java>
9. Head First Java by Kathy Sierra and Bert Bates
10. Programming with Java: A Primer by E. Balagurusamy