JAVA MINI PROJECT

n

**Online Quiz Management System**

submitted in partial fulfilment of the requirements for the award of

degree of

**BACHELOR OF TECHNOLOGY**  
in  
**COMPUTER SCIENCE & ENGINEERING**  
**(Artificial Intelligence & Machine Learning)**

By

**23WH1A6615 Akshitha.N**

**23WH1A6621 Vyshnavi.K**

**23WH1A6623 Sowmya.B**

**under the esteemed guidance of  
Dr. B. Lakshmi Praveena  
HoD & Professor CSE (AI & ML)**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**  
**(Artificial Intelligence & Machine Learning)**  
**BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN  
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)  
Accredited by NAAC with A Grade  
Bachupally, Hyderabad – 500090**

**BVRIT HYDERABAD  
COLLEGE OF ENGINEERING FOR WOMEN**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)  
Accredited by NAAC with A Grade  
Bachupally, Hyderabad – 500090

**Department of Computer Science & Engineering  
(Artificial Intelligence & Machine Learning)**



**CERTIFICATE**

This is to certify that the Java mini project entitled “BRICK BREAKER GAME” is a bonafide work carried out **by Ms. Akshitha.N (23WH1A6615), Ms. B.Sowmya (23WH1A6623), Ms. Vyshnavi.K (23WH1A6621)** in partial fulfilment for the award of B. Tech degree **in Computer Science & Engineering (AI & ML), BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad,** affiliated to Jawaharlal Nehru Technological University Hyderabad, under my guidance and supervision. The results embodied in the project work have not been submitted to any other**.**

**Internal Guide Head of the Department**

**Dr. B Lakshmi Praveena Dr. B Lakshmi Praveena**

**HOD & Professor HOD & Professor**

**Dept of CSE(AI&ML) Dept of CSE(AI&ML)**

**Project Aim**

The Online Quiz Management System aims to create a robust and scalable Java-based program for managing online quizzes, assessments, and mock tests. The goal is to replace paper-based quizzes with a digital platform that is automated and easy to use.  
2. Provide real-time quiz management for students, professionals, and institutions.  
3. Use automated scoring and secure login mechanisms to ensure the security, integrity, and accuracy of the tests.  
4. Provide administrators with a comprehensive platform for managing quizzes, analysing performance, and tracking progress effectively.  
5. Design an intuitive user interface that allows participants to navigate and interact seamlessly.  
  
The system's primary goal is to make it easier to create, organize, and evaluate quizzes, hence promoting effective learning, assessment, and competition.

**3. Project Description**

The Online Quiz Management System, a Java-based platform, enables online quizzes and exams through role-based capabilities. Students can log in, take quizzes, track their progress, and evaluate the results. Teachers create and manage quizzes, keep question banks, and manage user accounts. Organizations can use mock tests or evaluations to analyse performance data. The technology protects data, interacts with LMS, and provides a configurable interface. For the backend, the technical stack includes Java, relational databases, and frameworks such as Spring, while the frontend includes HTML/CSS, JavaScript, and React/Angular. Web servers such as Apache Tomcat are used for deployment, with optional cloud services for scalability. This system offers an effective, scalable solution for academic and competitive testing requirements.

**4. Process / Methodology**

**Step 1:**

**System Architecture of the Online Quiz Management System:  
Frontend: Creates a graphical user interface using Java Swing or JavaFX.  
Backend: The core logic in Java controls quiz flow, question randomization, timer functionality, and outcome computations.  
Database: A relational database, such as MySQL or SQLite, holds all system data, assuring consistency and scalability.  
Modular Architecture: Frontend, backend, and database activities are clearly separated, allowing for easier maintenance and scaling.  
This architecture offers a reliable and scalable approach for efficiently organizing online quizzes.**

**Step 2:**

. Phases of Development:  
Identify the functional and non-functional requirements through requirement analysis.  
Determine the most important features for administrators (teachers) and users (students).  
2. System Design: - Make thorough UML diagrams for each component of the system.  
Create data flow diagrams to show how modules interact with one another.  
3. Implementation: Create backend logic to calculate results, display questions, and authenticate users.  
Make a user interface that is responsive to provide easy navigation.  
Integrate a database to handle information.  
4. Testing: - Examine each module separately using unit testing.  
To guarantee smooth component communication, conduct integration testing.  
To test security and performance, use real-world scenarios.  
5. Deployment: - Set up the system on cloud platforms or local servers.  
Give administrators training and a user manual.

**Step 3:**

**. Functional Workflow:**

**1. User Workflow: Log in with secure credentials.  
- To load the quiz, enter its unique key.  
- View the test details and click the "START" button to begin the quiz.  
- Move through the questions, answering them within the allocated time.  
- Review the answers and submit the test.  
- See the results if quick feedback is enabled.  
2. Admin Workflow: Log in using administrative credentials.  
- Create quizzes by uploading questions and specifying criteria (such as time restrictions and question count).  
- Manage user accounts and track performance with thorough reporting.**

**Detailed Features and Modules**:

1. Login and Authentication: -

Provide secure login for both users and admins.  
- Password encryption to safeguard user credentials.

2. Quiz Module:

- Question Management:  
- Create, modify, and remove questions.  
- Organize questions by subject and difficulty.  
  
- Quiz Navigation: Display questions sequentially or randomly.  
- Navigation controls to move between questions.  
  
- Timer Integration: - A countdown timer appears during the test.  
- Automatic submission after the timer expires.

3. Result Processing:

- Instant calculation and comprehensive score breakdown.  
- Analyse the correct, wrong, and skipped questions.  
  
4. Administrative Dashboard:  
- An overview of all current and completed quizzes.  
- User performance analytics.  
- Tools for modifying quiz settings or questions.  
  
5. Database Maintenance:  
- Organized tables for users, quizzes, questions, and outcomes.  
- A relational model for effective data retrieval.

6.Tech Stack:  
- Programming Language: Java (backend and UI development).  
- Database: MySQL or SQLite (data storage).  
- Tools: include Java IDEs (Eclipse, IntelliJ IDEA) and database management tools (phpMyAdmin).

7. Advantages of the System:

1. Efficiency: Automating quiz processes decreases administrative overhead.  
2. Accessibility: Participants can take quizzes from anywhere at any time.  
3. Scalability: Can accommodate an increasing number of users and quizzes.  
4. Flexibility: Administrators can quickly create quizzes for a variety of purposes.  
5. Security: Encrypted data protects user and quiz integrity.

8. Possible improvements include:

Integration of mobile apps for increased accessibility.  
AI-powered question suggestions that take user performance into account.  
For smooth educational operations, integration with learning management systems (LMS) is essential.  
Support for multiple languages for a wide range of users.

Key Features:

User Authentication:

A login system validates users before accessing quizzes, ensuring data security.

Test participants and administrators have separate access privileges.

Test Management:

Questions are categorized by subjects, topics, and difficulty levels.

A unique test key system links users to specific quizzes.

User Interface:

A clear and interactive interface that adapts based on user roles (student or admin).

Real-time feedback on time remaining during the test.

Post-Test Analysis:

Users can review their answers before submitting.

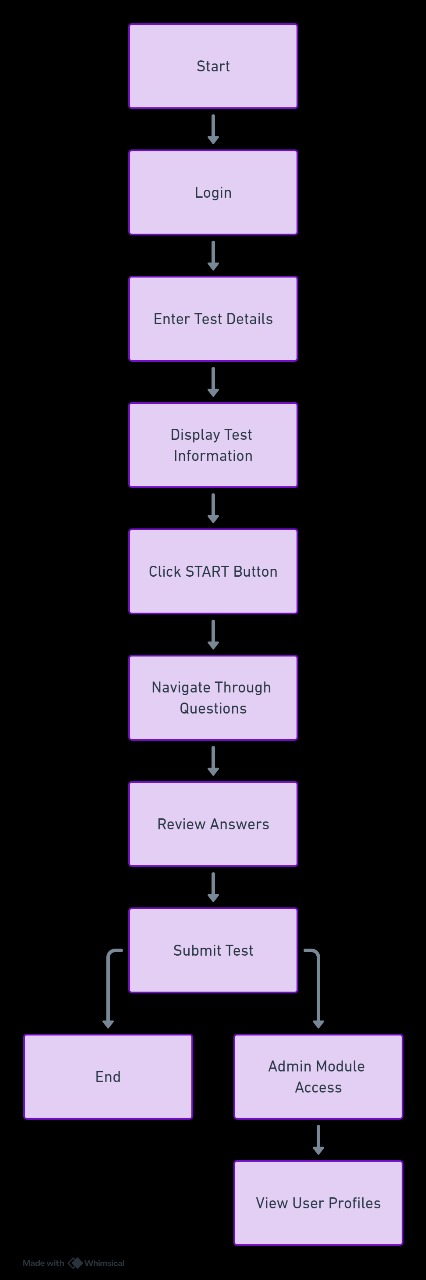
Administrators gain insights into test performance and trends.

FLOW CHART

This Java-based project is an online application that serves as a kind of platform for practice tests and competitions. The login button, which the user must click in order to begin the exam, is displayed in the user interface. It displays the number of questions, the time permitted, and a "START" button to start the exam when you enter the test's details (which might include a unique key). You are asked to review your answers after completing the test and click "SUBMIT" to send them in. The admin section allows you to access the user's profile. Here's a flowchart description:

1. **Start**: The user opens the application.
2. **Login**: The user clicks the login button and enters their credentials.
3. **Enter Test Details**: The user enters test details, which might include a unique key.
4. **Test Setup**:
   * Display the number of questions.
   * Display the time permitted.
   * Show the "START" button.
5. **Start Exam**: The user clicks the "START" button to begin the exam.
6. **Taking Exam**:
   * User answers the questions.
   * Timer runs concurrently.
7. **Review Answers**: Upon completion, the user reviews their answers.
8. **Submit Answers**: The user clicks the "SUBMIT" button to send in their answers.
9. **Admin Section**: Administrators access the user's profile to view results or manage user data.
10. **End**: The exam process is complete.

This flowchart outlines the major steps involved from the user's login to the exam submission and admin profile access



**Project Code:**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**public class ColorfulQuizApp {**

**private static JFrame frame;**

**private static JPanel panel;**

**private static CardLayout cardLayout;**

**private static JTextField usernameField;**

**private static JPasswordField passwordField;**

**private static ButtonGroup optionGroup;**

**private static int score = 0;**

**private static int currentQuestionIndex = 0;**

**private static Timer timer;**

**private static int timeLeft = 10;  // 10 seconds per question**

**private static JLabel timerLabel;**

**private static String[] questions = {**

**"What is the capital of France?",**

**"Which planet is known as the Red Planet?",**

**"What is 2 + 2?"**

**};**

**private static String[][] options = {**

**{"Paris", "London", "Rome", "Berlin"},**

**{"Mars", "Venus", "Earth", "Jupiter"},**

**{"3", "4", "5", "6"}**

**};**

**private static int[] correctAnswers = {0, 0, 1}; // Correct answers (index starting from 0)**

**public static void main(String[] args) {**

**// Initialize frame and layout**

**frame = new JFrame("Colorful Quiz App");**

**frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**cardLayout = new CardLayout();**

**panel = new JPanel(cardLayout);**

**// Login Panel**

**JPanel loginPanel = new JPanel();**

**loginPanel.setLayout(new GridLayout(3, 2));**

**loginPanel.setBackground(Color.CYAN);**

**JLabel usernameLabel = new JLabel("Username:");**

**usernameField = new JTextField();**

**JLabel passwordLabel = new JLabel("Password:");**

**passwordField = new JPasswordField();**

**JButton loginButton = new JButton("Login");**

**loginPanel.add(usernameLabel);**

**loginPanel.add(usernameField);**

**loginPanel.add(passwordLabel);**

**loginPanel.add(passwordField);**

**loginPanel.add(new JLabel());  // Empty label for alignment**

**loginPanel.add(loginButton);**

**// Quiz Panel**

**JPanel quizPanel = new JPanel();**

**quizPanel.setLayout(new BorderLayout());**

**quizPanel.setBackground(Color.LIGHT\_GRAY);**

**JPanel questionPanel = new JPanel();**

**questionPanel.setLayout(new GridLayout(7, 1));**

**questionPanel.setBackground(Color.LIGHT\_GRAY);**

**JLabel questionLabel = new JLabel("Question will appear here", JLabel.CENTER);**

**questionLabel.setFont(new Font("Arial", Font.BOLD, 14));**

**questionLabel.setForeground(Color.BLACK);**

**questionPanel.add(questionLabel);**

**optionGroup = new ButtonGroup();**

**JRadioButton option1 = new JRadioButton();**

**JRadioButton option2 = new JRadioButton();**

**JRadioButton option3 = new JRadioButton();**

**JRadioButton option4 = new JRadioButton();**

**optionGroup.add(option1);**

**optionGroup.add(option2);**

**optionGroup.add(option3);**

**optionGroup.add(option4);**

**option1.setBackground(Color.LIGHT\_GRAY);**

**option2.setBackground(Color.LIGHT\_GRAY);**

**option3.setBackground(Color.LIGHT\_GRAY);**

**option4.setBackground(Color.LIGHT\_GRAY);**

**questionPanel.add(option1);**

**questionPanel.add(option2);**

**questionPanel.add(option3);**

**questionPanel.add(option4);**

**timerLabel = new JLabel("Time: " + timeLeft + "s", JLabel.CENTER);**

**timerLabel.setFont(new Font("Arial", Font.BOLD, 14));**

**timerLabel.setForeground(Color.RED);**

**questionPanel.add(timerLabel);**

**JButton nextButton = new JButton("Next");**

**nextButton.setBackground(Color.GREEN);**

**nextButton.setForeground(Color.WHITE);**

**questionPanel.add(nextButton);**

**quizPanel.add(questionPanel, BorderLayout.CENTER);**

**// Result Panel**

**JPanel resultPanel = new JPanel();**

**resultPanel.setLayout(new BorderLayout());**

**resultPanel.setBackground(Color.YELLOW);**

**JLabel resultLabel = new JLabel("Your score: 0", JLabel.CENTER);**

**resultLabel.setFont(new Font("Arial", Font.BOLD, 20));**

**resultLabel.setForeground(Color.BLUE);**

**resultPanel.add(resultLabel, BorderLayout.CENTER);**

**JButton restartButton = new JButton("Restart");**

**restartButton.setBackground(Color.ORANGE);**

**restartButton.setForeground(Color.WHITE);**

**resultPanel.add(restartButton, BorderLayout.SOUTH);**

**// Add panels to the cardLayout**

**panel.add(loginPanel, "Login");**

**panel.add(quizPanel, "Quiz");**

**panel.add(resultPanel, "Result");**

**frame.setContentPane(panel);**

**frame.setSize(400, 350);**

**frame.setLocationRelativeTo(null); // Center the frame**

**frame.setVisible(true);**

**// Action Listeners**

**loginButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**String username = usernameField.getText();**

**String password = new String(passwordField.getPassword());**

**if (username.equals("user1") && password.equals("123")) {**

**cardLayout.show(panel, "Quiz");**

**loadNextQuestion();**

**startTimer();**

**} else {**

**JOptionPane.showMessageDialog(frame, "Invalid username or password.");**

**}**

**}**

**});**

**nextButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**if (isAnswerCorrect()) {**

**score++;**

**}**

**currentQuestionIndex++;**

**if (currentQuestionIndex < questions.length) {**

**loadNextQuestion();**

**resetTimer();**

**} else {**

**showResult();**

**}**

**}**

**});**

**restartButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**score = 0;**

**currentQuestionIndex = 0;**

**loadNextQuestion();**

**resetTimer();**

**cardLayout.show(panel, "Quiz");**

**}**

**});**

**}**

**// Method to load the next question**

**private static void loadNextQuestion() {**

**if (currentQuestionIndex < questions.length) {**

**JPanel questionPanel = (JPanel) ((JPanel) panel.getComponent(1)).getComponent(0);**

**JLabel questionLabel = (JLabel) questionPanel.getComponent(0);**

**JRadioButton option1 = (JRadioButton) questionPanel.getComponent(1);**

**JRadioButton option2 = (JRadioButton) questionPanel.getComponent(2);**

**JRadioButton option3 = (JRadioButton) questionPanel.getComponent(3);**

**JRadioButton option4 = (JRadioButton) questionPanel.getComponent(4);**

**questionLabel.setText(questions[currentQuestionIndex]);**

**option1.setText(options[currentQuestionIndex][0]);**

**option2.setText(options[currentQuestionIndex][1]);**

**option3.setText(options[currentQuestionIndex][2]);**

**option4.setText(options[currentQuestionIndex][3]);**

**optionGroup.clearSelection();**

**}**

**}**

**// Method to check if the selected answer is correct**

**private static boolean isAnswerCorrect() {**

**JPanel questionPanel = (JPanel) ((JPanel) panel.getComponent(1)).getComponent(0);**

**JRadioButton option1 = (JRadioButton) questionPanel.getComponent(1);**

**JRadioButton option2 = (JRadioButton) questionPanel.getComponent(2);**

**JRadioButton option3 = (JRadioButton) questionPanel.getComponent(3);**

**JRadioButton option4 = (JRadioButton) questionPanel.getComponent(4);**

**if (option1.isSelected() && correctAnswers[currentQuestionIndex] == 0) {**

**return true;**

**} else if (option2.isSelected() && correctAnswers[currentQuestionIndex] == 1) {**

**return true;**

**} else if (option3.isSelected() && correctAnswers[currentQuestionIndex] == 2) {**

**return true;**

**} else if (option4.isSelected() && correctAnswers[currentQuestionIndex] == 3) {**

**return true;**

**}**

**return false;**

**}**

**// Method to start the timer for each question**

**private static void startTimer() {**

**timeLeft = 10; // Reset to 10 seconds per question**

**JPanel questionPanel = (JPanel) ((JPanel) panel.getComponent(1)).getComponent(0);**

**timerLabel = (JLabel) questionPanel.getComponent(6);**

**timer = new Timer(1000, new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**timeLeft--;**

**timerLabel.setText("Time: " + timeLeft + "s");**

**if (timeLeft == 0) {**

**timer.stop();**

**nextQuestion();**

**}**

**}**

**});**

**timer.start();**

**}**

**// Method to reset the timer for the next question**

**private static void resetTimer() {**

**if (timer != null && timer.isRunning()) {**

**timer.stop();**

**}**

**startTimer();**

**}**

**// Method to go to the next question when the timer runs out**

**private static void nextQuestion() {**

**if (isAnswerCorrect()) {**

**score++;**

**}**

**currentQuestionIndex++;**

**if (currentQuestionIndex < questions.length) {**

**loadNextQuestion();**

**resetTimer();**

**} else {**

**showResult();**

**}**

**}**

**// Method to display the result after the quiz is finished**

**private static void showResult() {**

**JPanel resultPanel = (JPanel) panel.getComponent(2);**

**JLabel resultLabel = (JLabel) resultPanel.getComponent(0);**

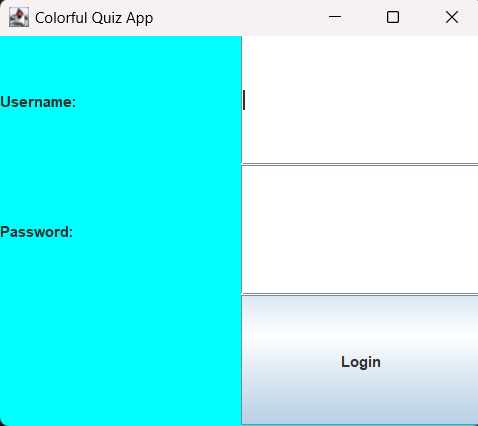
**resultLabel.setText("Your score: " + score);**

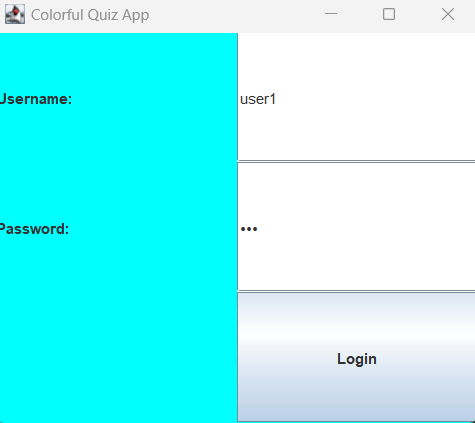
**cardLayout.show(panel, "Result");**

**}**

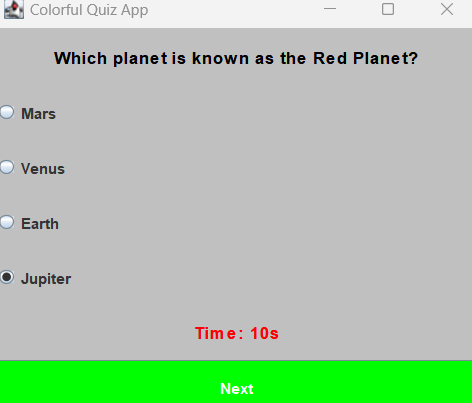
**}**

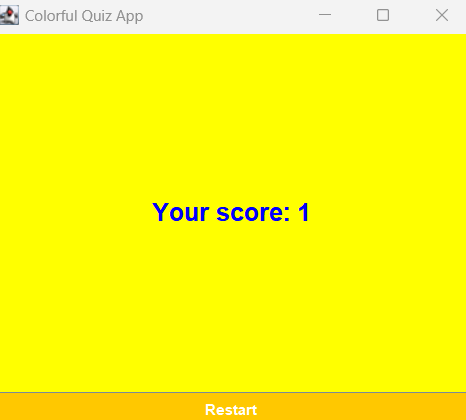
**OUTPUT SCREENS:**

****

****

****

****

****

**CONCLUSION:**

**The Online Quiz Management System is a cutting-edge Java-based platform created to streamline and modernize the administration of online tests and quizzes. It effectively satisfies the needs of students, educators, and organizations because to its strong architecture, intuitive user interface, and safe environment. The system is a priceless resource for both professional and educational contexts since it facilitates efficient quiz administration, effective learning, and perceptive performance analysis. Its adaptability and scalability guarantee that it can support a range of institutions, encouraging ongoing improvement and effective management. All things considered, it transforms conventional testing techniques and improves the learning process for all users.**