

QuizApp: An Interactive GUI-Based Quiz Application

IA-3

Subject: Object-Oriented Programming (DSE 2123)

Mini Project

Student Name	Prajnaa Ray Choudhury
Reg No	230968348
Assignment No.	IA-3
Subject Code	DSE 2123
Subject	OOP with Java
Marks	10M

Contents

1.	Problem Statement	Page 3
2.	Methodology	Page 4
3.	Results and Screenshot	Page 7
4.	Program Code	Page 14
5.	Files used for file handling	Page 21
6.	References	Page 22

Problem Statement 5: Develop a GUI-Based Quiz Application

Requirements:

1. Quiz Setup: Create a GUI for administrators to add, edit, and delete quiz questions with multiple choice answers.
2. Quiz Interface: Users should be able to take quizzes, selecting answers from multiple-choice questions.
3. Timer Functionality: Implement a timer for each quiz, with the option to start and stop the quiz automatically based on the timer.
4. Result Display: At the end of the quiz, display the user's score and provide feedback (correct/incorrect answers).

The goal of QuizApp is to create an engaging, user-friendly GUI based quiz application that allows users to answer multiple-choice questions. The primary aim of this application is to make learning more engaging and accessible by incorporating real-time feedback, score tracking, time management and a user-friendly interface.

Methodology

1. Requirements Gathering

- **Objective:** To develop a GUI-based quiz application QuizApp, that caters to two types of users: **Admins** and **Users**. The requirements were gathered through brainstorming sessions focused on providing an interactive quiz experience with features like question management, score tracking, time management, and feedback.
- **Admin Requirements:** Admins can add, edit, and delete questions.
- **User Requirements:** Users experience a simple interface to answer questions, receive instant feedback, option to pause the timer and view their scores at the end of each quiz.
- **Non-functional Requirements:** Ensures user-friendliness, responsiveness, and fast performance for an optimal quiz experience.

2. Design Phase

- **System Architecture:** A modular, object-oriented structure was planned to separate different aspect of the application, like question management and other UI elements.
- **Class Structure**
 - **QuizApp Class:** Manages the overall quiz flow, with a login panel, an admin panel where the admin can manage the questions and a quiz panel where the user attempts the timed multiple-choice questions.
 - **Question Class:** Represents each question with its text, options and the correct answer, making it easier to add or modify questions.
- **File Handling with `users.txt` and `questions.txt`**
 - **`users.txt`:** Stores user login information, such as usernames and passwords.
 - **`questions.txt`:** Contains quiz questions, options and correct answers. By using this file, admins can update the question bank externally without modifying the core code, and the app can dynamically load questions each time a quiz is launched.
- **Swing Libraries for User Interface(UI)**
 - **JFrame:** Used as the main window for the app, serving as the container for all UI elements, including the quiz display, question navigation, and score summary.
 - **JLabel:** Displays static text on the screen, such as instructions, question prompts, and feedback messages.
 - **JButton:** Provides clickable buttons for user actions

- **TextField and JPasswordField:** Used for user input fields in the login and registration screens, with JPasswordField providing hidden text for secure password entry.
- **JOptionPane:** Used for displaying pop-up dialogs to show alerts, confirmations, and score summaries.
- **JRadioButton:** Used for multiple-choice answer options, allowing users to select a single answer per question.
- **JPanel:** Serves as a container to organize other UI components.
- **JList and JScrollPane:** Together, they allow for a scrollable list view, used in admin sections for managing or selecting questions in a list format.
- **User Interface (UI) Design:** A straightforward, user-friendly interface is created to guide users seamlessly through the quiz, displaying questions, feedback and scores clearly.

3. Development Phase

- **Programming Language:** Java, used due to its flexibility. Object-oriented structure, and cross-platform compatibility.
- **Core Functionality**
 - **Quiz Logic:** Implemented in Java using Swing components for question navigation, answer selection, and score tracking.
 - **Question Management and File Handling:** Used questions.txt to load questions at runtime and update data without altering the code.
 - **User Interface:** Swing components create an engaging and organised quiz interface.
 - **Score Calculation:** Score updates occur in real time as each question is answered, giving users instant feedback on their progress.

4. Implementation of Key Features

- **Question Management:** Created an admin interface to manage questions in questions.txt. Admins can edit the file to update questions without modifying the code.
- **Score Tracking and Feedback:** Final score summaries with feedbacks are implemented.
- **User Authentication:** User login and registration, interacting with users.txt for secure access.
- **Timed Questions:** A timer is added to each question for a challenge, which can be paused anytime by the user.

5. Testing and Debugging

- **Unit Testing:** Tested each component and feature individually, including file handling for users.txt and questions.txt.

- **UI Testing:** Ensured all Swing components worked as intended, testing for ease of use, accessibility, and smooth interaction.
- **Performance Testing:** Assessed response times and file handling to maintain an efficient, fast experience.

6. Git link

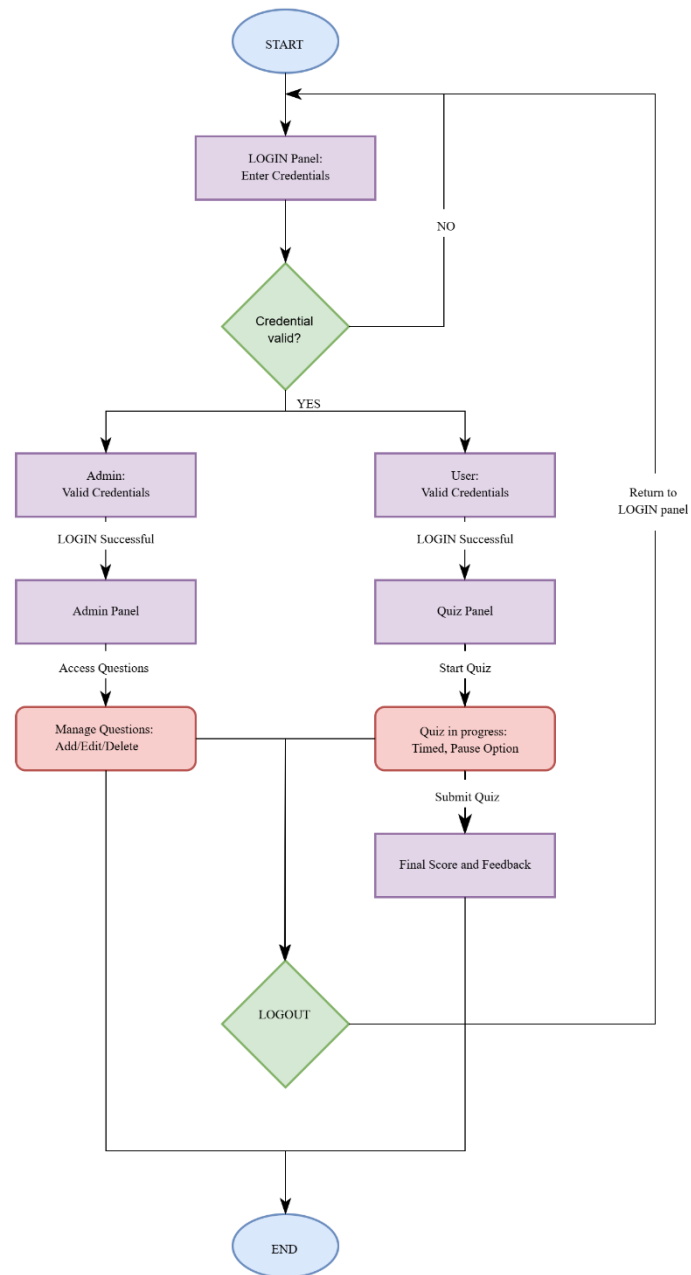
- <https://github.com/ankshirc/GUI-Based-Quiz-Application.git>

Results and Screenshot

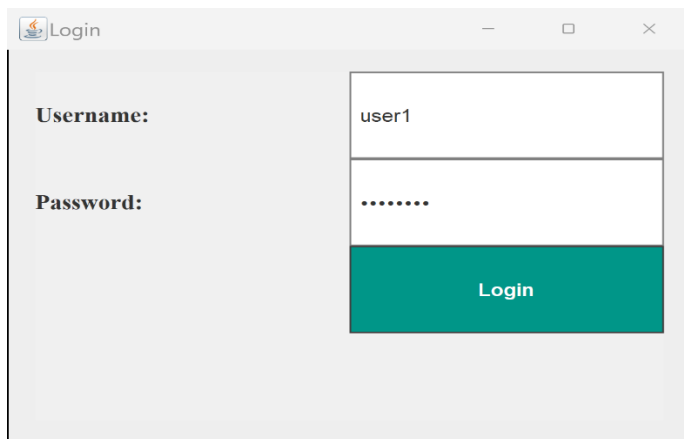
Events and Actions

1. Login panel appears. Enter credentials.
2. Entered as user.
 - a. If valid credentials, then login successful. Quiz panel appears. Timed quiz with multiple choice questions. Timer can be paused as well as per user's desire. Display final score and feedback after quiz. Program terminates.
 - b. If invalid credentials, then login unsuccessful. Login panel reappears to try again.
 - c. Once entered as user, panel also contains a logout button to logout of quiz and login panel reappears.
3. Entered as admin.
 - a. If valid credentials, then login successful. Admin panel appears. Admin can add, edit or delete questions, as per their desire.
 - b. If invalid credentials, then login unsuccessful. Login panel reappears to try again.
 - c. Once entered as admin, panel also contains a logout button to logout of admin panel and login panel reappears.

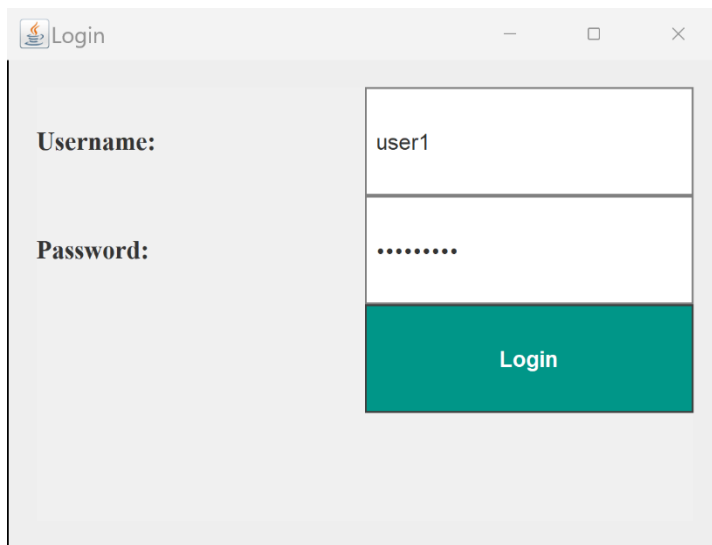
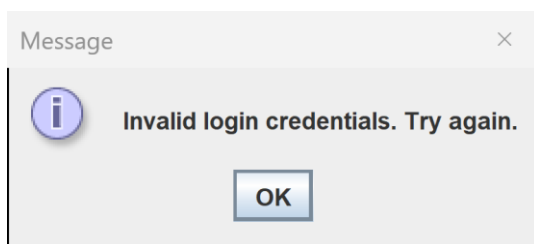
Flowchart



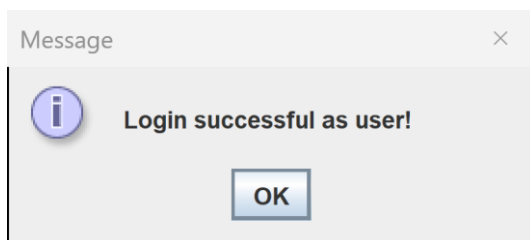
Logging in as user

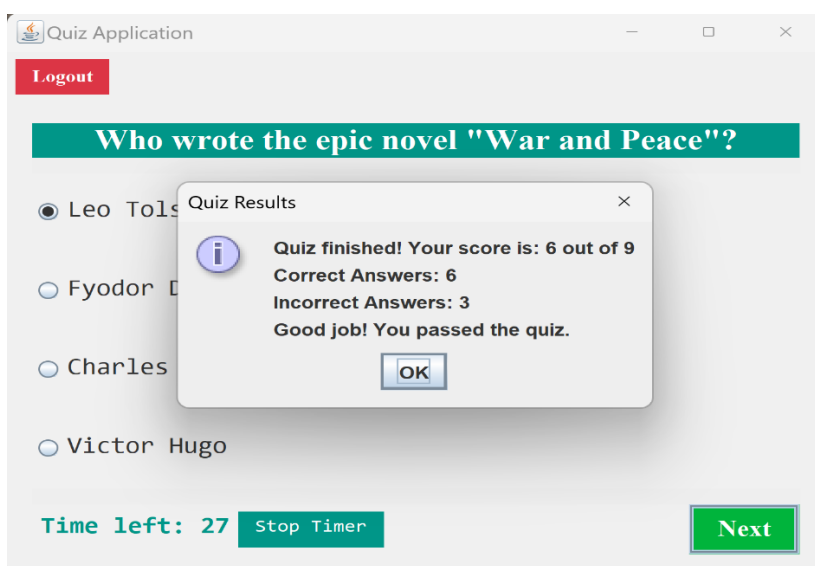
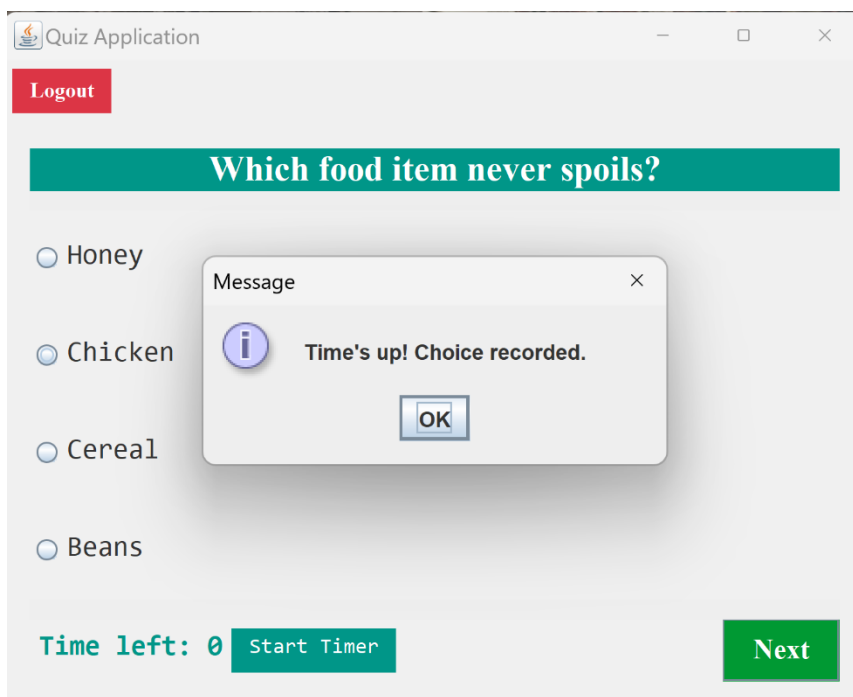
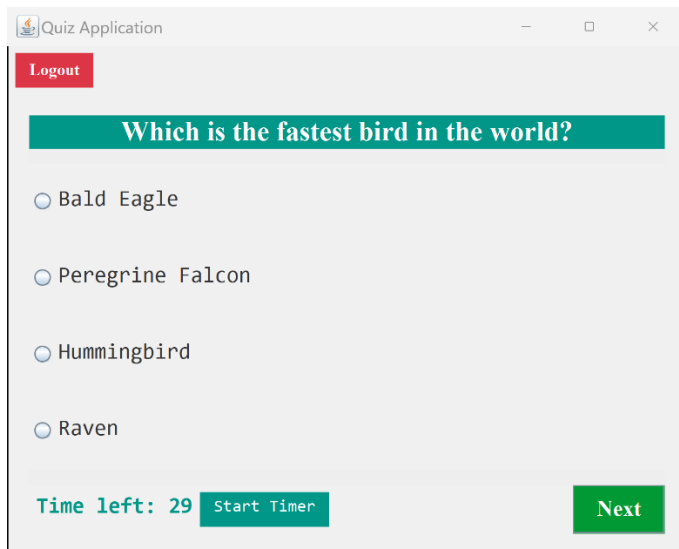


A screenshot of a web application's login window. The window has a title bar with a small icon and the text "Login". Inside, there are two labels: "Username:" and "Password:". To the right of "Username:" is a text input field containing the text "user1". To the right of "Password:" is a password input field filled with seven dots. Below these fields is a large green button with the text "Login" in white.

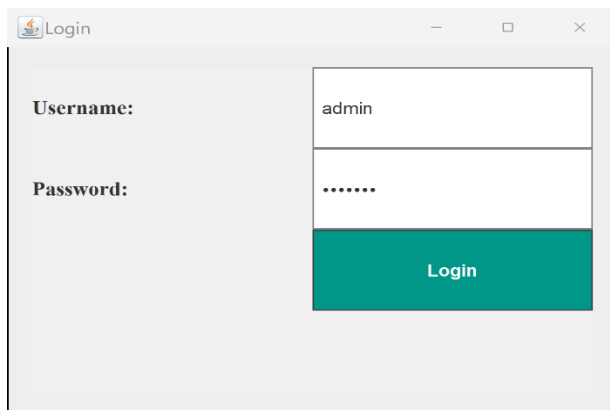


A screenshot of the same login window as before. The "Username:" field contains "user1" and the "Password:" field contains seven dots. The green "Login" button is visible at the bottom right.

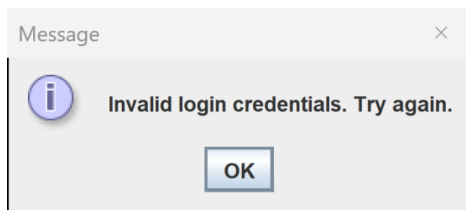




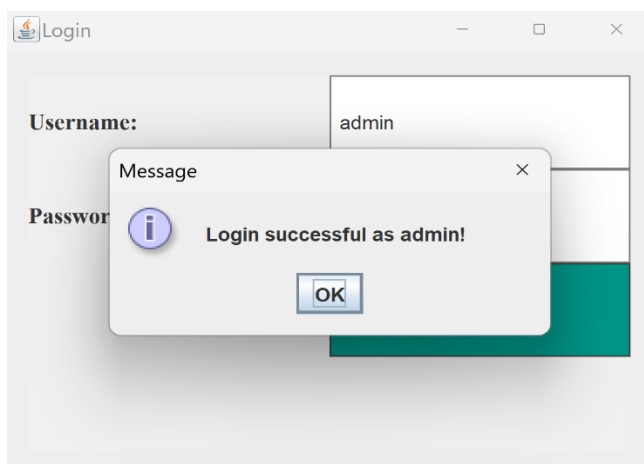
Logging in as admin



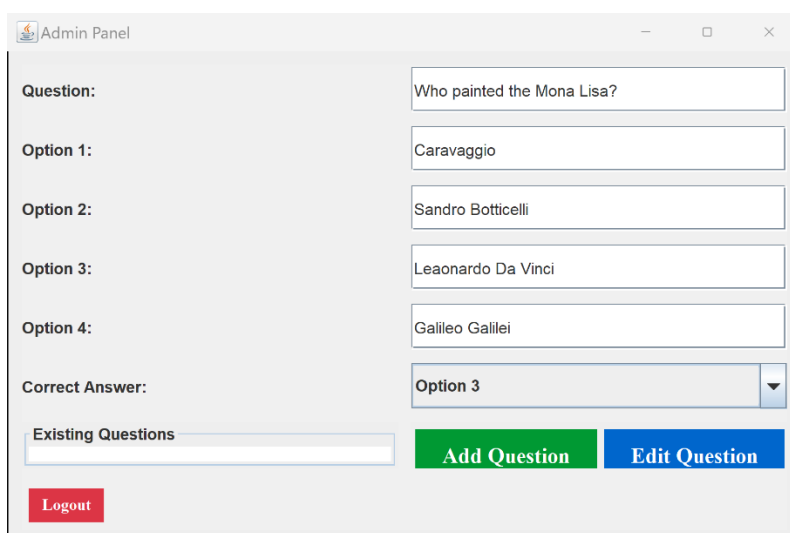
A screenshot of a web application window titled "Login". It contains two input fields: "Username:" with the text "admin" and "Password:" with masked characters ".....". Below the password field is a green "Login" button.



A screenshot of a "Message" dialog box. It features an information icon (i) and the text "Invalid login credentials. Try again." Below the text is an "OK" button.



A screenshot of the "Login" window with the "Username:" field containing "admin". A "Message" dialog box is overlaid on top, displaying an information icon (i) and the text "Login successful as admin!". The dialog has an "OK" button.



A screenshot of the "Admin Panel" window. It contains several input fields for creating a question: "Question:" (text: "Who painted the Mona Lisa?"), "Option 1:" (text: "Caravaggio"), "Option 2:" (text: "Sandro Botticelli"), "Option 3:" (text: "Leonardo Da Vinci"), and "Option 4:" (text: "Galileo Galilei"). Below these is a "Correct Answer:" dropdown menu with "Option 3" selected. At the bottom left is an "Existing Questions" input field. At the bottom right are two buttons: "Add Question" (green) and "Edit Question" (blue). A red "Logout" button is located at the bottom left.

Admin Panel

Question: Who painted the Mona Lisa?

Option 1: Caravaggio

Option 2:

Option 3:

Option 4:

Correct Answer: Option 3

Existing Questions

Add Question Edit Question

Logout

Message

Question Added!

OK

Admin Panel

Question: Which food item never spoils?

Option 1: Honey

Option 2: Chicken

Option 3: Cereal

Option 4: Beans

Correct Answer: Option 1

Existing Questions

QuizApp.Question@615bcbc2

QuizApp.Question@69d30b8

QuizApp.Question@3ebdd9b1

QuizApp.Question@33b44f5

Logout

Add Question Edit Question Delete Question

Admin Panel

Question: Which food item never spoils?

Option 1: Honey

Option 2: Chicken

Option 3:

Option 4: Fruit

Correct Answer: Option 1

Existing Questions

QuizApp.Question@615bcbc2

QuizApp.Question@69d30b8

QuizApp.Question@686ef93c

QuizApp.Question@33b44f5

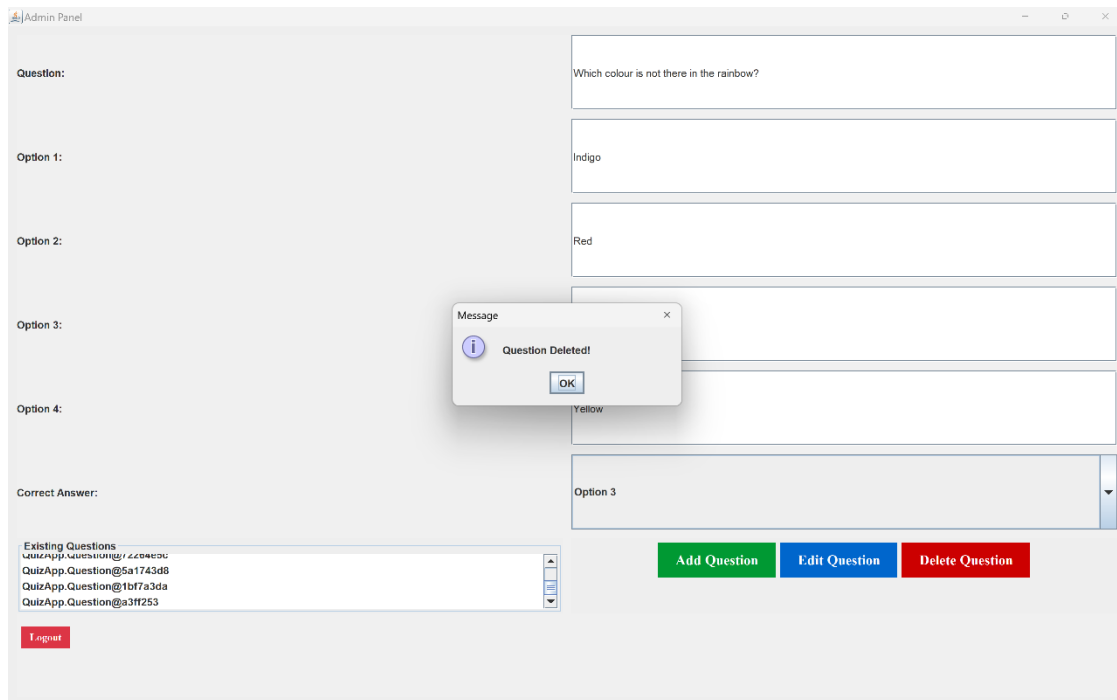
Logout

Add Question Edit Question Delete Question

Message

Question Edited!

OK



Logout button works in Quiz panel as well as the Admin panel.

Program code

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
1 package QuizApp;
2
3 import javax.swing.*;
4
5
6
7
8
9
10
11 class Question {
12     private String questionText;
13     private String[] options;
14     private int correctAnswerIndex;
15
16     public Question(String questionText, String[] options, int correctAnswerIndex) {
17         this.questionText = questionText;
18         this.options = options;
19         this.correctAnswerIndex = correctAnswerIndex;
20     }
21
22     public String getQuestionText() {
23         return questionText;
24     }
25
26     public String[] getOptions() {
27         return options;
28     }
29
30     public int getCorrectAnswerIndex() {
31         return correctAnswerIndex;
32     }
33
34     // Convert question to a file string
35     public String toFileString() {
36         return questionText + ";" + String.join(";", options) + ";" + correctAnswerIndex;
37     }
38
39     // Parse question from a file string
40     public static Question fromFileString(String line) {
41         String[] parts = line.split(";");
42         String[] options = parts[1].split(",");
43         return new Question(parts[0], options, Integer.parseInt(parts[2]));
44     }
45 }
46
47 public class QuizApp {
48     private JFrame frame;
49     private JLabel questionLabel;
50     private JRadioButton[] optionButtons;
51     private ButtonGroup optionGroup;
52     private JButton nextButton;
53     private JLabel timerLabel;
54
55     private List<Question> questions;
56     private int currentQuestionIndex = 0;
57     private int score = 0;
```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
58 private int timeLeft = 30; // Time left for the quiz
59 private boolean timerRunning = false; // timer state
60 private Timer timer; // timer object
61 private JButton startStopButton;
62 private String currentUserRole = ""; // To store if current user is admin or user
63
64
65 // Path to store user credentials
66 private final String USER_FILE_PATH = "users.txt"; // file to store login details
67 private final String QUESTION_FILE_PATH = "questions.txt"; // File to store questions
68
69 public QuizApp() {
70     // Start with the login screen
71     createLoginPanel();
72 }
73
74 private void createLoginPanel() {
75     frame = new JFrame("Login");
76     frame.setSize(400, 300);
77     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
78     frame.setLayout(new GridLayout(4, 2));
79     frame.setLocationRelativeTo(null); // Center the frame
80     frame.getContentPane().setBackground(new Color(240, 240, 240)); // Light gray background
81     frame.getRootPane().setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15)); // Frame padding
82
83     JLabel userLabel = new JLabel("Username:");
84     userLabel.setFont(new Font("Times New Roman", Font.BOLD, 14));
85     JLabel passLabel = new JLabel("Password:");
86     passLabel.setFont(new Font("Times New Roman", Font.BOLD, 14));
87
88     JPasswordField userField = new JPasswordField();
89     userField.setBorder(BorderFactory.createCompoundBorder(
90         BorderFactory.createLineBorder(Color.GRAY, 1),
91         BorderFactory.createEmptyBorder(5, 5, 5, 5)
92     ));
93
94     JPasswordField passField = new JPasswordField();
95     passField.setBorder(BorderFactory.createCompoundBorder(
96         BorderFactory.createLineBorder(Color.GRAY, 1),
97         BorderFactory.createEmptyBorder(5, 5, 5, 5)
98     ));
99
100     JButton loginButton = new JButton("Login");
101     frame.getContentPane().setBackground(new Color(240, 240, 240)); // Light gray background
102     loginButton.setBackground(new Color(5, 150, 130)); // teal colour
103     loginButton.setForeground(Color.WHITE);
104     loginButton.setFocusPainted(false);
105     loginButton.setBorder(BorderFactory.createLineBorder(Color.DARK_GRAY, 1)); // Border for button
106     userField.setBorder(BorderFactory.createCompoundBorder(BorderFactory.createLineBorder(Color.GRAY, 1), BorderFactory.createEmptyBorder(5, 5, 5, 5)));
107     passField.setBorder(BorderFactory.createCompoundBorder(BorderFactory.createLineBorder(Color.GRAY, 1), BorderFactory.createEmptyBorder(5, 5, 5, 5)));
108 }
```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
109 // Action on login button click
110 loginButton.addActionListener(new ActionListener() {
111     @Override
112     public void actionPerformed(ActionEvent e) {
113         String username = userField.getText();
114         String password = new String(passField.getPassword());
115
116         if (authenticateUser(username, password)) {
117             JOptionPane.showMessageDialog(frame, "Login successful as " + currentUserRole + "!");
118             frame.dispose(); // Close login frame
119             if (currentUserRole.equals("admin")) {
120                 createAdminPanel();
121             } else {
122                 createQuizPanel();
123             }
124         } else {
125             JOptionPane.showMessageDialog(frame, "Invalid login credentials. Try again.");
126         }
127     }
128 });
129
130 frame.add(userLabel);
131 frame.add(userField);
132 frame.add(passLabel);
133 frame.add(passField);
134 frame.add(new JLabel("")); // Spacer
135 frame.add(loginButton);
136
137 frame.setVisible(true);
138 }
139
140 // Authenticate user by checking username and password from file
141 private boolean authenticateUser(String username, String password) {
142     try (BufferedReader reader = new BufferedReader(new FileReader(USER_FILE_PATH))) {
143         String line;
144         while ((line = reader.readLine()) != null) {
145             String[] userDetails = line.split(";");
146             if (userDetails[0].equals(username) && userDetails[1].equals(password)) {
147                 currentUserRole = userDetails[2]; // "admin" or "user"
148                 return true;
149             }
150         }
151     } catch (IOException e) {
152         System.out.println("Error reading users file: " + e.getMessage());
153     }
154 }
155
156 // Create the Quiz Panel for regular users
157 private void createQuizPanel() {
158     JFrame frame = new JFrame("Quiz Application");
159     frame.setSize(500, 400);
160     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
161     frame.setLayout(new BorderLayout());
162     frame.setLocationRelativeTo(null); // Center the frame
163
164     // Add top panel with logout button
165     frame.add(createTopPanelWithLogout(), BorderLayout.NORTH);
166
167     // Set background color
168     JPanel mainPanel = new JPanel();
169     mainPanel.setBackground(new Color(240, 240, 240)); // Light gray background
170     mainPanel.setLayout(new BorderLayout());
171     mainPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15)); // Main padding
172
173     // Timer Label
174     JLabel timerLabel = new JLabel("Time left: " + timeLeft);
175     timerLabel.setFont(new Font("Consolas", Font.BOLD, 16));
176     timerLabel.setForeground(new Color(0, 150, 136)); // Teal text
177
178     // Timer Button
179     JButton startStopButton = new JButton("Start Timer");
180     startStopButton.setFont(new Font("Consolas", Font.PLAIN, 12));
181     startStopButton.setBackground(new Color(0, 150, 136)); // Teal background
182     startStopButton.setForeground(Color.WHITE); // White Text Color
183     startStopButton.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10)); //padding for smaller button
184
185     startStopButton.addActionListener(new ActionListener() {
186         @Override
187         public void actionPerformed(ActionEvent e) {
188             if (timerRunning) {
189                 stopTimer();
190             } else {
191                 startTimer();
192             }
193         }
194     });
195 }
196
```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
153 }
154 return false;
155 }
156
157 // Create the Quiz Panel for regular users
158 private void createQuizPanel() {
159     JFrame frame = new JFrame("Quiz Application");
160     frame.setSize(500, 400);
161     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
162     frame.setLayout(new BorderLayout());
163     frame.setLocationRelativeTo(null); // Center the frame
164
165     // Add top panel with logout button
166     frame.add(createTopPanelWithLogout(), BorderLayout.NORTH);
167
168     // Set background color
169     JPanel mainPanel = new JPanel();
170     mainPanel.setBackground(new Color(240, 240, 240)); // Light gray background
171     mainPanel.setLayout(new BorderLayout());
172     mainPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15)); // Main padding
173
174     // Timer Label
175     JLabel timerLabel = new JLabel("Time left: " + timeLeft);
176     timerLabel.setFont(new Font("Consolas", Font.BOLD, 16));
177     timerLabel.setForeground(new Color(0, 150, 136)); // Teal text
178
179     // Timer Button
180     JButton startStopButton = new JButton("Start Timer");
181     startStopButton.setFont(new Font("Consolas", Font.PLAIN, 12));
182     startStopButton.setBackground(new Color(0, 150, 136)); // Teal background
183     startStopButton.setForeground(Color.WHITE); // White Text Color
184     startStopButton.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10)); //padding for smaller button
185
186     startStopButton.addActionListener(new ActionListener() {
187         @Override
188         public void actionPerformed(ActionEvent e) {
189             if (timerRunning) {
190                 stopTimer();
191             } else {
192                 startTimer();
193             }
194         }
195     });
196 }
197
```

```

eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
197 // Timer that updates every second
198 timer = new Timer(1000, new ActionListener() {
199     @Override
200     public void actionPerformed(ActionEvent e) {
201         if (timeLeft > 0) {
202             timeLeft--;
203             updateTimeLabel();
204         } else {
205             timer.stop();
206             timerRunning = false;
207             startStopButton.setText("Start Timer");
208             JOptionPane.showMessageDialog(frame, "Time's up! Choice recorded.");
209             checkAnswer(); // Automatically check the answer when time runs out
210             currentQuestionIndex++;
211             if (currentQuestionIndex < questions.size()) {
212                 displayQuestion(currentQuestionIndex); // Display the next question
213             } else {
214                 endQuiz(); // End the quiz if there are no more questions
215             }
216         }
217     }
218 });
219
220 // Create a panel for the timer and Start/Stop button
221 JPanel timerPanel = new JPanel();
222 timerPanel.setLayout(new FlowLayout()); // Flow layout for horizontal alignment
223 timerPanel.setBackground(new Color(240, 240, 240)); // Match panel color
224 timerPanel.add(timerLabel);
225 timerPanel.add(startStopButton); // Add the start/stop button to the timer panel
226
227 // Question Label
228 questionLabel = new JLabel("Question will appear here");
229 questionLabel.setHorizontalAlignment(SwingConstants.CENTER); // Center the question
230 questionLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
231 questionLabel.setBackground(new Color(0, 150, 135)); // Teal background
232 questionLabel.setForeground(Color.WHITE); // White text
233 questionLabel.setOpaque(true); // Make background visible
234 mainPanel.add(questionLabel, BorderLayout.NORTH);
235
236 // Option Buttons
237 JPanel optionPanel = new JPanel();
238 optionPanel.setLayout(new GridLayout(4, 1));
239 optionPanel.setBorder(BorderFactory.createEmptyBorder(10, 0, 10, 0)); // Option padding
240 optionButtons = new JRadioButton[4];

```

```

eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
241 optionGroup = new ButtonGroup();
242 for (int i = 0; i < 4; i++) {
243     optionButtons[i] = new JRadioButton();
244     optionButtons[i].setFont(new Font("Consolas", Font.PLAIN, 16));
245     optionButtons[i].setBackground(new Color(240, 240, 240)); // Match panel color
246     optionGroup.add(optionButtons[i]);
247     optionPanel.add(optionButtons[i]);
248 }
249 mainPanel.add(optionPanel, BorderLayout.CENTER);
250
251 // Next Button
252 nextButton = new JButton("Next");
253 nextButton.setFont(new Font("Times New Roman", Font.BOLD, 16));
254 nextButton.setBackground(new Color(0, 153, 51)); // Green background
255 nextButton.setForeground(Color.WHITE); // White text
256
257 nextButton.addMouseListener(new java.awt.event.MouseAdapter() {
258     public void mouseEntered(java.awt.event.MouseEvent evt) {
259         nextButton.setBackground(new Color(0, 180, 60)); // Lighter green on hover
260     }
261
262     public void mouseExited(java.awt.event.MouseEvent evt) {
263         nextButton.setBackground(new Color(0, 153, 51)); // Original green
264     }
265 });
266
267 nextButton.addActionListener(new ActionListener() {
268     @Override
269     public void actionPerformed(ActionEvent e) {
270         checkAnswer();
271         currentQuestionIndex++;
272         if (currentQuestionIndex < questions.size()) {
273             displayQuestion(currentQuestionIndex);
274         } else {
275             endQuiz();
276         }
277     }
278 });
279
280 // Create a bottom panel to hold the timer panel and next button
281 JPanel bottomPanel = new JPanel();
282 bottomPanel.setLayout(new BorderLayout());
283 bottomPanel.setBackground(new Color(240, 240, 240)); // Match panel color
284 bottomPanel.add(timerPanel, BorderLayout.WEST); // Timer and button on the left

```



```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
285 bottomPanel.add(nextButton, BorderLayout.EAST); // Next button on the right
286
287 mainPanel.add(bottomPanel, BorderLayout.SOUTH); // Add the bottom panel to the south
288
289 //Load Questions from file
290 loadQuestions();
291
292 //display the first question
293 displayQuestion(currentQuestionIndex);
294
295 frame.add(mainPanel, BorderLayout.CENTER);
296 frame.setVisible(true);
297
298 // Method to stop the timer
299 private void stopTimer() {
300     timerRunning = false;
301     startStopButton.setText("Start Timer");
302     timer.stop();
303 }
304
305 // Method to start the timer
306 private void startTimer() {
307     timerRunning = true;
308     startStopButton.setText("Stop Timer");
309     timer.start();
310 }
311
312 // Method to update the timer label
313 private void updateTimerLabel() {
314     timerLabel.setText("Time left: " + timeLeft);
315 }
316
317 // Create the Admin Panel for managing questions
318 private void createAdminPanel() {
319     frame = new JFrame("Admin Panel");
320     frame.setSize(600, 400);
321     frame.setLayout(new GridLayout(0,2,10,10)); // 0 rows, 2 columns
322     frame.getContentPane().setBackground(new Color(240, 240, 240)); // Light gray background
323     frame.setLocationRelativeTo(null); // Center the frame
324     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Close application on exit
325     frame.getRootPane().setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); // Add padding
326
327 // Initialize questions if null
328
```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
329 if (questions == null) {
330     questions = new ArrayList<>();
331     loadQuestions();
332 }
333
334 JTextField questionText = new JTextField();
335 JTextField option1 = new JTextField();
336 JTextField option2 = new JTextField();
337 JTextField option3 = new JTextField();
338 JTextField option4 = new JTextField();
339 JComboBox<String> correctAnswerDropdown = new JComboBox<>(new String[]{"Option 1", "Option 2", "Option 3", "Option 4"});
340
341 // create buttons
342 JButton addButton = createButton("Add Question", new Color(0, 153, 51));
343 JButton editButton = createButton("Edit Question", new Color(0, 102, 204));
344 JButton deleteButton = createButton("Delete Question", new Color(204, 0, 0));
345
346 // List to display existing questions
347 DefaultListModel<Question> listModel = new DefaultListModel<>();
348 JList<Question> questionList = new JList<>(listModel);
349 JScrollPane scrollPane = new JScrollPane(questionList);
350 scrollPane.setBorder(BorderFactory.createTitledBorder("Existing Questions"));
351
352 // Adding components to frame
353 frame.add(new JLabel("Question:"));
354 frame.add(questionText);
355 frame.add(new JLabel("Option 1:"));
356 frame.add(option1);
357 frame.add(new JLabel("Option 2:"));
358 frame.add(option2);
359 frame.add(new JLabel("Option 3:"));
360 frame.add(option3);
361 frame.add(new JLabel("Option 4:"));
362 frame.add(option4);
363 frame.add(new JLabel("Correct Answer:"));
364 frame.add(correctAnswerDropdown);
365
366 //adding scroll pane for the question list
367 frame.add(scrollPane);
368
369 //Create a panel for buttons
370 JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
371 buttonPanel.add(addButton);
372 buttonPanel.add(editButton);
```

```

eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
QuizApp.java
373 buttonPanel.add(deleteButton);
374
375 //add button panel to the frame
376 frame.add(buttonPanel);
377
378 // Action listener for adding questions
379 addButton.addActionListener(new ActionListener() {
380     @Override
381     public void actionPerformed(ActionEvent e) {
382         String[] options = {option1.getText(), option2.getText(), option3.getText(), option4.getText()};
383         int correctIndex = correctAnswerDropdown.getSelectedIndex();
384         questions.add(new Question(questionText.getText(), options, correctIndex));
385         saveQuestions(); // Save updated questions to the file
386         JOptionPane.showMessageDialog(frame, "Question Added!");
387
388         // Clear input fields after adding
389         clearInputFields(questionText, option1, option2, option3, option4, correctAnswerDropdown);
390         loadQuestionsIntoListModel(listModel); // Refresh the list
391     }
392 });
393
394 //Action listener for editing questions
395 editButton.addActionListener(new ActionListener() {
396     @Override
397     public void actionPerformed(ActionEvent e) {
398         int selectedIndex = questionList.getSelectedIndex();
399         if (selectedIndex >= 0) {
400             // Update selected question
401             String[] options = {option1.getText(), option2.getText(), option3.getText(), option4.getText()};
402             int correctIndex = correctAnswerDropdown.getSelectedIndex();
403             Question updatedQuestion = new Question(questionText.getText(), options, correctIndex);
404             questions.set(selectedIndex, updatedQuestion); // Update the list
405             listModel.set(selectedIndex, updatedQuestion); // Update the list model
406             saveQuestions(); // Save updated questions to the file
407             JOptionPane.showMessageDialog(frame, "Question Edited!");
408             clearInputFields(questionText, option1, option2, option3, option4, correctAnswerDropdown);
409         } else {
410             JOptionPane.showMessageDialog(frame, "Please select a question to edit.");
411         }
412     }
413 });
414
415 // Action listener for deleting questions
416 deleteButton.addActionListener(new ActionListener() {

```

```

eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
QuizApp.java
417 @Override
418 public void actionPerformed(ActionEvent e) {
419     int selectedIndex = questionList.getSelectedIndex();
420     if (selectedIndex >= 0) {
421         questions.remove(selectedIndex); // Remove from the list
422         listModel.remove(selectedIndex); // Remove from the list model
423         saveQuestions(); // Save updated questions to the file
424         JOptionPane.showMessageDialog(frame, "Question Deleted!");
425         clearInputFields(questionText, option1, option2, option3, option4, correctAnswerDropdown);
426     } else {
427         JOptionPane.showMessageDialog(frame, "Please select a question to delete.");
428     }
429 }
430 });
431
432 // add mouse listener to the question list to populate fields when a question is selected
433 questionList.addListSelectionListener(e -> {
434     if (!e.getValueIsAdjusting()) {
435         Question selectedQuestion = questionList.getSelectedValue();
436         if (selectedQuestion != null) {
437             questionText.setText(selectedQuestion.getQuestionText());
438             String[] options = selectedQuestion.getOptions();
439             option1.setText(options[0]);
440             option2.setText(options[1]);
441             option3.setText(options[2]);
442             option4.setText(options[3]);
443             correctAnswerDropdown.setSelectedIndex(selectedQuestion.getCorrectAnswerIndex());
444         }
445     }
446 });
447
448 // Add the top panel with logout button
449 frame.add(createTopPanelWithLogout(), BorderLayout.SOUTH);
450 frame.setVisible(true);
451 }
452
453 // Helper method to create buttons with common styles
454 private JButton createButton(String text, Color background) {
455     JButton button = new JButton(text);
456     button.setBackground(background); // Set button background
457     button.setForeground(Color.WHITE); // White text
458     button.setFont(new Font("Times New Roman", Font.BOLD, 16));
459     button.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20)); // Add padding
460     return button;

```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
461 }
462
463 private JPanel createTopPanelWithLogout() {
464     // Create the top panel for the logout button
465     JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
466     topPanel.setBackground(new Color(240, 240, 240)); // Light gray background to match main panels
467
468     // Logout button with non-intrusive styling
469     JButton logoutButton = new JButton("Logout");
470     logoutButton.setFont(new Font("Times New Roman", Font.BOLD, 12));
471     logoutButton.setBackground(new Color(220, 53, 69)); // Red color to signify logout
472     logoutButton.setForeground(Color.WHITE); // White text color
473     logoutButton.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10)); // Padding for smaller button
474     logoutButton.setFocusPainted(false); // Remove focus border on click
475     logoutButton.addActionListener(e -> logout());
476
477     // Add the logout button to the top panel
478     topPanel.add(logoutButton);
479
480     return topPanel;
481 }
482
483 // Load existing questions into the list model
484 private void loadQuestionsIntoListModel(DefaultListModel<Question> listModel) {
485     listModel.clear();
486     for (Question question : questions) {
487         listModel.addElement(question);
488     }
489 }
490
491 // Clear input fields
492 private void clearInputFields(JTextField questionText, JTextField option1, JTextField option2, JTextField option3, JTextField option4, JComboBox<String> correctAnswerDropdown) {
493     questionText.setText("");
494     option1.setText("");
495     option2.setText("");
496     option3.setText("");
497     option4.setText("");
498     correctAnswerDropdown.setSelectedIndex(0);
499 }
500
501 // Logout method to handle switching between modes
502 private void logout() {
503     frame.dispose(); // Close current frame
504     createLoginPanel(); // Redirect to login panel
505 }
506
507 // Load questions from the file
508 private void loadQuestions() {
509     questions = new ArrayList<>();
510     File questionFile = new File(QUESTION_FILE_PATH);
511
512     if (!questionFile.exists()) {
513         JOptionPane.showMessageDialog(frame, "Questions file not found. Please check the path.", "Error", JOptionPane.ERROR_MESSAGE);
514         return;
515     }
516
517     try (BufferedReader reader = new BufferedReader(new FileReader(QUESTION_FILE_PATH))) {
518         String line;
519         while ((line = reader.readLine()) != null) {
520             questions.add(Question.fromFileString(line));
521         }
522     } catch (IOException e) {
523         JOptionPane.showMessageDialog(frame, "Error loading questions from file: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
524     }
525 }
526
527 // Save questions to the file, appending instead of overwriting
528 private void saveQuestions() {
529     // Save only the last added question to avoid duplicates
530     if (!questions.isEmpty()) {
531         Question lastQuestion = questions.get(questions.size() - 1);
532         String lastQuestionString = lastQuestion.toFileString();
533
534         // Check if the question already exists in the file
535         try (BufferedReader reader = new BufferedReader(new FileReader(QUESTION_FILE_PATH))) {
536             String line;
537             boolean exists = false;
538             while ((line = reader.readLine()) != null) {
539                 if (line.equals(lastQuestionString)) {
540                     exists = true; // Question already exists
541                     break;
542                 }
543             }
544
545             // Append only if it doesn't exist
546             if (!exists) {
547                 // Append the question to the file
548                 try (FileWriter writer = new FileWriter(QUESTION_FILE_PATH, true)) {
549                     writer.write(lastQuestionString + "\n");
550                 }
551             }
552         }
553     }
554 }
```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
505 }
506
507 // Load questions from the file
508 private void loadQuestions() {
509     questions = new ArrayList<>();
510     File questionFile = new File(QUESTION_FILE_PATH);
511
512     if (!questionFile.exists()) {
513         JOptionPane.showMessageDialog(frame, "Questions file not found. Please check the path.", "Error", JOptionPane.ERROR_MESSAGE);
514         return;
515     }
516
517     try (BufferedReader reader = new BufferedReader(new FileReader(QUESTION_FILE_PATH))) {
518         String line;
519         while ((line = reader.readLine()) != null) {
520             questions.add(Question.fromFileString(line));
521         }
522     } catch (IOException e) {
523         JOptionPane.showMessageDialog(frame, "Error loading questions from file: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
524     }
525 }
526
527 // Save questions to the file, appending instead of overwriting
528 private void saveQuestions() {
529     // Save only the last added question to avoid duplicates
530     if (!questions.isEmpty()) {
531         Question lastQuestion = questions.get(questions.size() - 1);
532         String lastQuestionString = lastQuestion.toFileString();
533
534         // Check if the question already exists in the file
535         try (BufferedReader reader = new BufferedReader(new FileReader(QUESTION_FILE_PATH))) {
536             String line;
537             boolean exists = false;
538             while ((line = reader.readLine()) != null) {
539                 if (line.equals(lastQuestionString)) {
540                     exists = true; // Question already exists
541                     break;
542                 }
543             }
544
545             // Append only if it doesn't exist
546             if (!exists) {
547                 // Append the question to the file
548                 try (FileWriter writer = new FileWriter(QUESTION_FILE_PATH, true)) {
549                     writer.write(lastQuestionString + "\n");
550                 }
551             }
552         }
553     }
554 }
```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

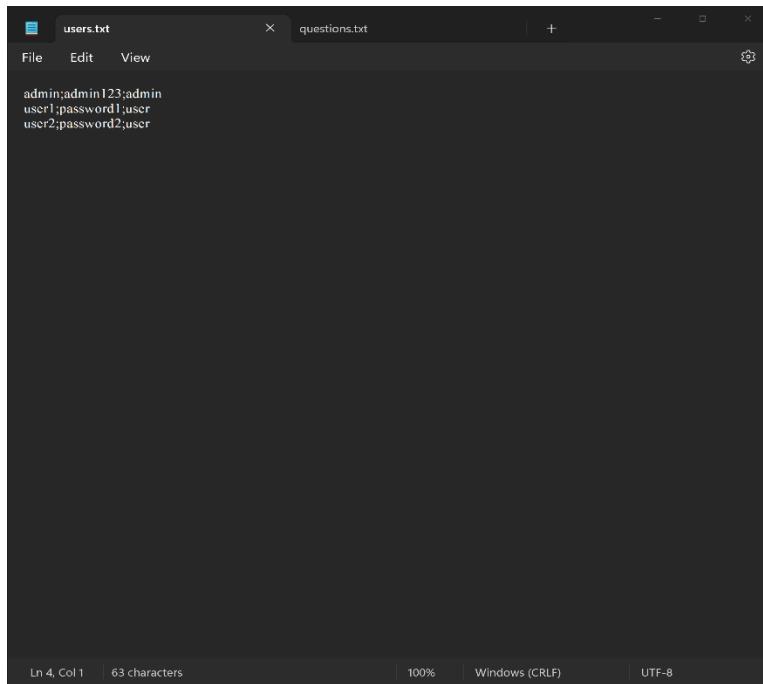
QuizApp.java
549 try (BufferedWriter writer = new BufferedWriter(new FileWriter(QUESTION_FILE_PATH, true))) {
550     writer.write(lastQuestionString);
551     writer.newLine();
552 }
553 } catch (IOException e) {
554     System.out.println("Error saving questions to file: " + e.getMessage());
555 }
556 }
557 }
558 }
559
560 // Display the question on the quiz panel
561 private void displayQuestion(int questionIndex) {
562     Question question = questions.get(questionIndex);
563     questionLabel.setText(question.getQuestionText());
564
565     String[] options = question.getOptions();
566     for (int i = 0; i < options.length; i++) {
567         optionButtons[i].setText(options[i]);
568     }
569
570     optionGroup.clearSelection(); // Clear previous selection
571     timeLeft = 30; // Reset the timer for the new question
572     updateTimerLabel(); // update the timer display
573     startTimer(); //start the timer for the new question
574 }
575
576 // Check the answer for the current question
577 private void checkAnswer() {
578     Question question = questions.get(currentQuestionIndex);
579     int correctAnswerIndex = question.getCorrectAnswerIndex();
580
581     // Check which radio button is selected
582     for (int i = 0; i < optionButtons.length; i++) {
583         if (optionButtons[i].isSelected() && i == correctAnswerIndex) {
584             score++;
585             break;
586         }
587     }
588 }
589
590 // End the quiz and show the final score
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```
eclipse-workspace - swingers/src/QuizApp/QuizApp.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

QuizApp.java
582 // Check which radio button is selected
583 for (int i = 0; i < optionButtons.length; i++) {
584     if (optionButtons[i].isSelected() && i == correctAnswerIndex) {
585         score++;
586         break;
587     }
588 }
589
590 // End the quiz and show the final score
591
592 private void endQuiz() {
593     timer.stop();
594
595     // Calculate the number of correct and incorrect answers
596     int totalQuestions = questions.size();
597     int correctAnswers = score;
598     int incorrectAnswers = totalQuestions - correctAnswers;
599
600     // Create feedback message
601     String feedbackMessage = "Quiz finished! Your score is: " + score + " out of " + totalQuestions + "\n" +
602         "Correct Answers: " + correctAnswers + "\n" +
603         "Incorrect Answers: " + incorrectAnswers + "\n";
604
605     // Provide comments based on performance
606     if (correctAnswers == totalQuestions) {
607         feedbackMessage += "Excellent work! You got all the answers correct!";
608     } else if (correctAnswers >= totalQuestions / 2) {
609         feedbackMessage += "Good job! You passed the quiz.";
610     } else {
611         feedbackMessage += "Don't worry! Review the material and try again.";
612     }
613
614     // Show the feedback message in a dialog
615     JOptionPane.showMessageDialog(frame, feedbackMessage, "Quiz Results", JOptionPane.INFORMATION_MESSAGE);
616     frame.dispose(); // Close the quiz frame
617
618
619
620
621 public static void main(String[] args) {
622     new QuizApp();
623 }
624 }
625 //end of program
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Files used for file handling

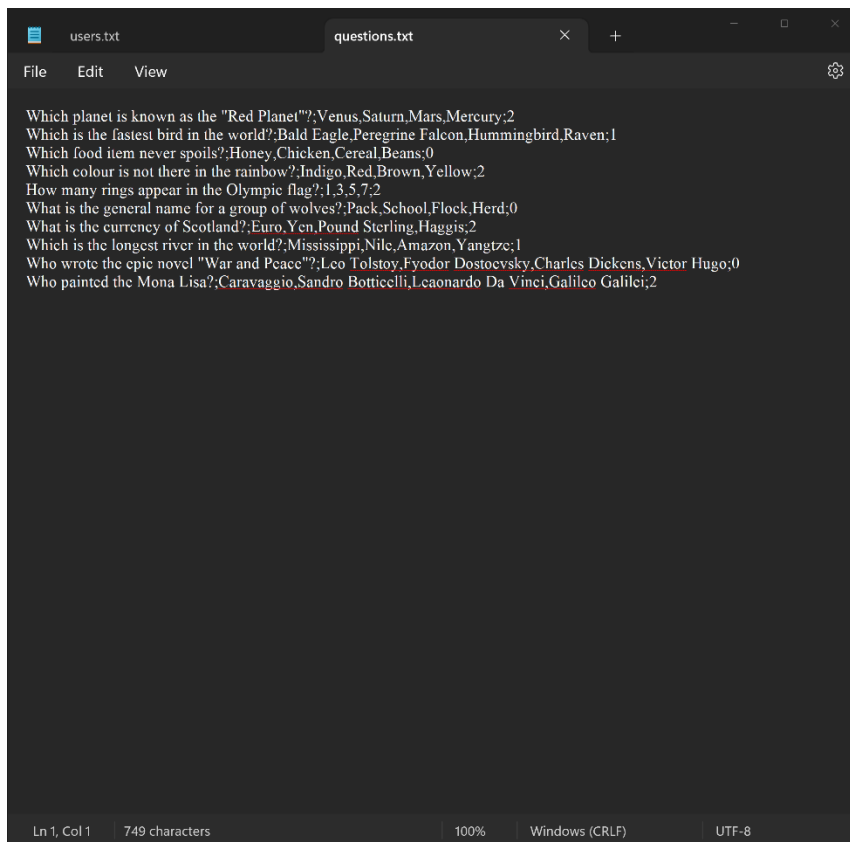
users.txt



A screenshot of a text editor window with two tabs: 'users.txt' and 'questions.txt'. The 'users.txt' tab is active, showing three lines of text: 'admin;admin123;admin', 'user1;password1;user', and 'user2;password2;user'. The editor has a dark theme, a menu bar with 'File', 'Edit', and 'View', and a status bar at the bottom indicating 'Ln 4, Col 1', '63 characters', '100%', 'Windows (CRLF)', and 'UTF-8'.

```
admin;admin123;admin
user1;password1;user
user2;password2;user
```

questions.txt



A screenshot of a text editor window with two tabs: 'users.txt' and 'questions.txt'. The 'questions.txt' tab is active, showing ten lines of text, each a question followed by a semicolon and a list of options and a number. The text is: 'Which planet is known as the "Red Planet"?;Venus,Saturn,Mars,Mercury;2', 'Which is the fastest bird in the world?;Bald Eagle,Peregrine Falcon,Hummingbird,Raven;1', 'Which food item never spoils?;Honey,Chicken,Cereal,Beans;0', 'Which colour is not there in the rainbow?;Indigo,Red,Brown,Yellow;2', 'How many rings appear in the Olympic flag?;1,3,5,7;2', 'What is the general name for a group of wolves?;Pack,School,Flock,Herd;0', 'What is the currency of Scotland?;Euro,Yen,Pound Sterling,Haggis;2', 'Which is the longest river in the world?;Mississippi,Nile,Amazon,Yangtze;1', 'Who wrote the epic novel "War and Peace"?;Leo Tolstoy,Fyodor Dostoevsky,Charles Dickens,Victor Hugo;0', and 'Who painted the Mona Lisa?;Caravaggio,Sandro Botticelli,Leaonardo Da Vinci,Galileo Galilei;2'. The editor has a dark theme, a menu bar with 'File', 'Edit', and 'View', and a status bar at the bottom indicating 'Ln 1, Col 1', '749 characters', '100%', 'Windows (CRLF)', and 'UTF-8'.

```
Which planet is known as the "Red Planet"?;Venus,Saturn,Mars,Mercury;2
Which is the fastest bird in the world?;Bald Eagle,Peregrine Falcon,Hummingbird,Raven;1
Which food item never spoils?;Honey,Chicken,Cereal,Beans;0
Which colour is not there in the rainbow?;Indigo,Red,Brown,Yellow;2
How many rings appear in the Olympic flag?;1,3,5,7;2
What is the general name for a group of wolves?;Pack,School,Flock,Herd;0
What is the currency of Scotland?;Euro,Yen,Pound Sterling,Haggis;2
Which is the longest river in the world?;Mississippi,Nile,Amazon,Yangtze;1
Who wrote the epic novel "War and Peace"?;Leo Tolstoy,Fyodor Dostoevsky,Charles Dickens,Victor Hugo;0
Who painted the Mona Lisa?;Caravaggio,Sandro Botticelli,Leaonardo Da Vinci,Galileo Galilei;2
```

References

- <https://github.com/ankshirc/GUI-Based-Quiz-Application.git>
- <https://www.coursera.org/learn/writing-java-code-for-applications/home/info>
- <https://www.coursera.org/projects/build-java-gui-apps>
- [Lesson: Using Swing Components \(The Java™ Tutorials > Creating a GUI With Swing\)](#)
- [Java Swing Tutorial - javatpoint](#)

THANK YOU