A Project Report

on

Java GUI-Based Quiz Application – Project Report

by

Ain ul haq Fatima Bcy243060

Samia khalid khan Bcy243060

Ubaid ur rehman Bcy243081

A Project Report submitted to the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

in partial fulfillment of the requirements for the degree of

BACHELORS OF SCIENCE IN Cyber Security

Faculty of Engineering

Capital University of Science & Technology, Islamabad june, 2025

Copyright © 2025 by CUST Student

All rights reserved. Reproduction in whole or in part in any form requires the prior written permission of Ain ul haq Fatima Bcy243060, Samia Khalid Khan Bcy243106, Ubaid ur rehman or designated representative.

DECLARATION

It is declared that this is an original piece of our own work, except where otherwise acknowledged in text and references. This work has not been submitted in any form for another degree or diploma at any university or other institution for tertiary education and shall not be submitted by us in future for obtaining any degree from this or any other University or Institution.

AIN UL HAQ FATIMA
Reg. No. Bcy243060

SAMIA KHALID KHAN
Reg. No. BCY243106

UBAID UR REHMAN
Reg. No. BCY243084

JUNE, 2024

CERTIFICATE OF APPROVAL

It is certified that the project titled Java GUI-Based Quiz Application – Project Report" carried out by samia khalid khan bcy243106, ain ul haq Fatima bcy243060,ubaid ur rehman shah bcy243084 under the supervision of Miss tooba , Capital University of Science & Technology, Islamabad, is fully adequate, in scope and in quality, for the degree of BS cyber scurity.

Supervisor: ----------

Miss tooba

Lecturer

Department of Electrical and Computer Engineering

Faculty of Engineering

Capital University of Science & Technology, Islamabad

HoD: ----------

Dr. Noor Mohammad Khan

Professor

Department of Electrical and Computr Engineering

Faculty of Engineering Capital University of Science & Technology, Islamabad

ACKNOWLEDGMENT

We would like to thank “Miss tooba” whose guidance helped to complete our report in the given time. Last but not the least, this project cannot be completed without the effort and co-operation of group members.

ABSTRACT

This project presents a Java-based GUI Quiz Application developed using Swing and core Java libraries. The application serves as an interactive educational tool, offering users a timed environment to answer multiple-choice questions. It integrates features like automatic question skipping upon timer expiration and high score tracking through file storage. The system demonstrates fundamental object-oriented principles, event-driven programming, GUI design, and file handling in Java. Its modular architecture makes it easily extendable for future improvements, such as subject-based categories or user account integration. The project is suitable for academic purposes and entry-level Java developers exploring GUI applications.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Acronym	Full Form
GUI	Graphical User Interface
OOP	Object-Oriented Programming
UML	Unified Modeling Language
I/O	Input/Output
API	Application Programming Interface
JVM	Java Virtual Machine
IDE	Integrated Development Environment

Java GUI-Based Quiz Application – Project Report

1. Introduction of Project

This project is a **Java-based GUI Quiz Application** developed using **Java Swing**. It allows users to attempt multiple-choice questions within a timed environment. The system auto-skips questions when the time runs out and maintains a **high score** tracker using file handling. The primary objective is to test and enhance users' general knowledge in an interactive and user-friendly manner.

2. Literature Review

Java Swing provides a set of GUI components for building desktop applications. Previous quiz applications have utilized Java Swing for rendering buttons, radio buttons, and labels. Timers and file handling have also been employed in many student-level projects. This application builds upon these components, integrating:

- javax.swing.Timer for countdowns,
- JRadioButton and ButtonGroup for question options,
- File I/O using BufferedReader and BufferedWriter to persist high scores.

This design adheres to basic **Object-Oriented Principles** and GUI best practices to deliver an intuitive experience.

3. Design Phase of Project

a. Architecture Overview

- **Frontend:** Java Swing
- **Backend:** Core Java logic
- **Storage:** Text file for high scores

b. Flow Logic

Start → Load Questions → Display Question & Timer →

User Selects Option → Clicks Next or Timer Ends →

Evaluate Answer → Load Next Question →

Repeat Until Done → Show Score & High Score → End

4. Implementation Phase

a. Core Technologies

- Java Swing for UI
- Java File I/O for persistence
- OOP concepts: Classes, Inheritance, Encapsulation

b. GitHub Repository

🔗 GitHub Repository Link (Example)

https://github.com/Aini005

https://github.com/baidbee/bcy084

https://github.com/Samiakhalidkhan-66

c. LinkedIn Profile

LinkedIn Profile

https://www.linkedin.com/in/ubaid-shah-544126371/

www.linkedin.com/in/aiñyē-khān-b0901a28a

https://www.linkedin.com/in/samiyah-khalidkhan-6a2911316?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app

d. Code (Summarized)

The entire code is packed in a single QuizApp.java file which includes:

- Question inner class for question data
- Main class QuizApp which handles UI rendering, timer logic, and score evaluation
- Countdown timer logic that auto-submits answers
- High score tracking using a local file (highscore.txt)

Refer to Appendix for full code.

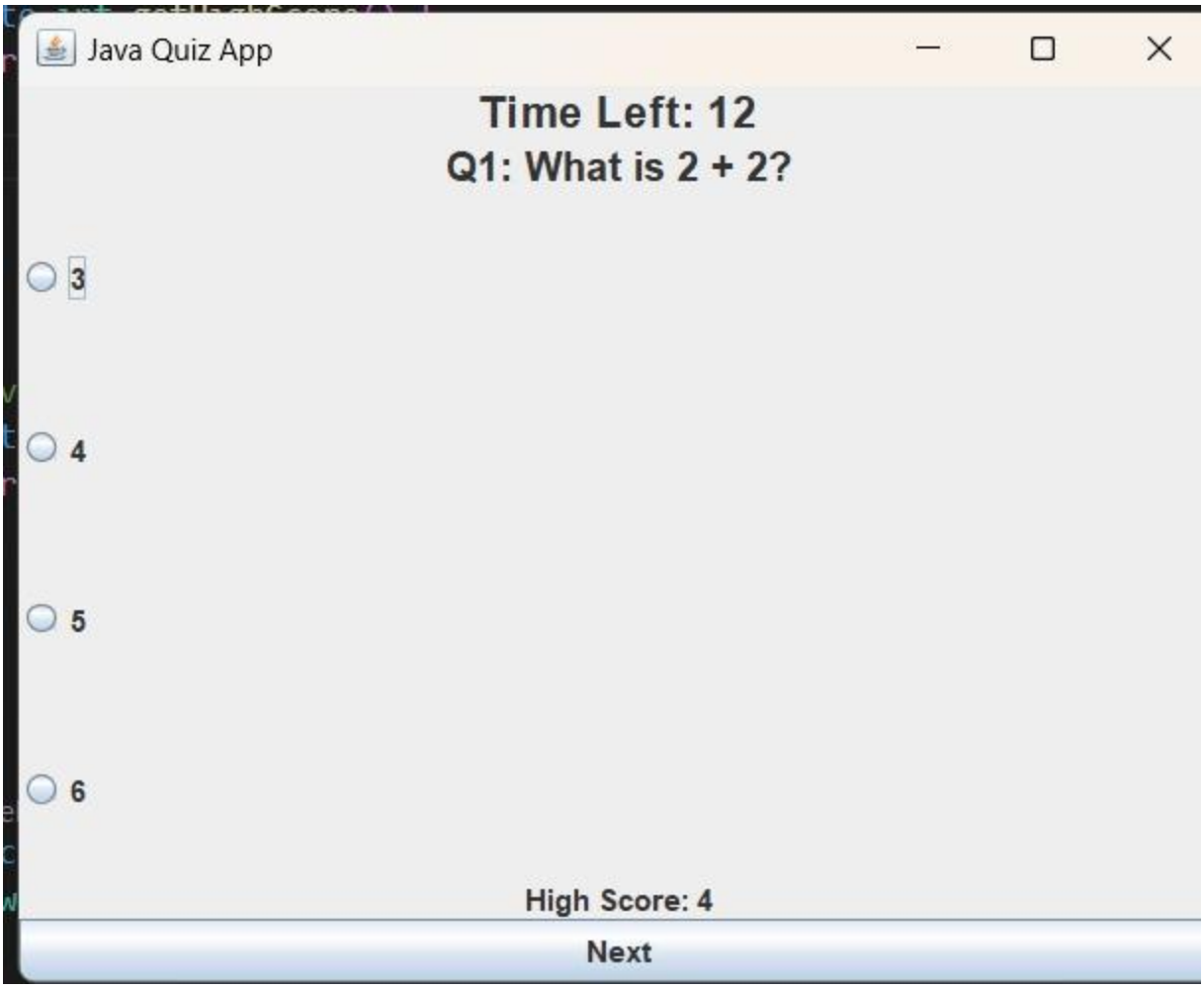
5. Results (Output of Code):

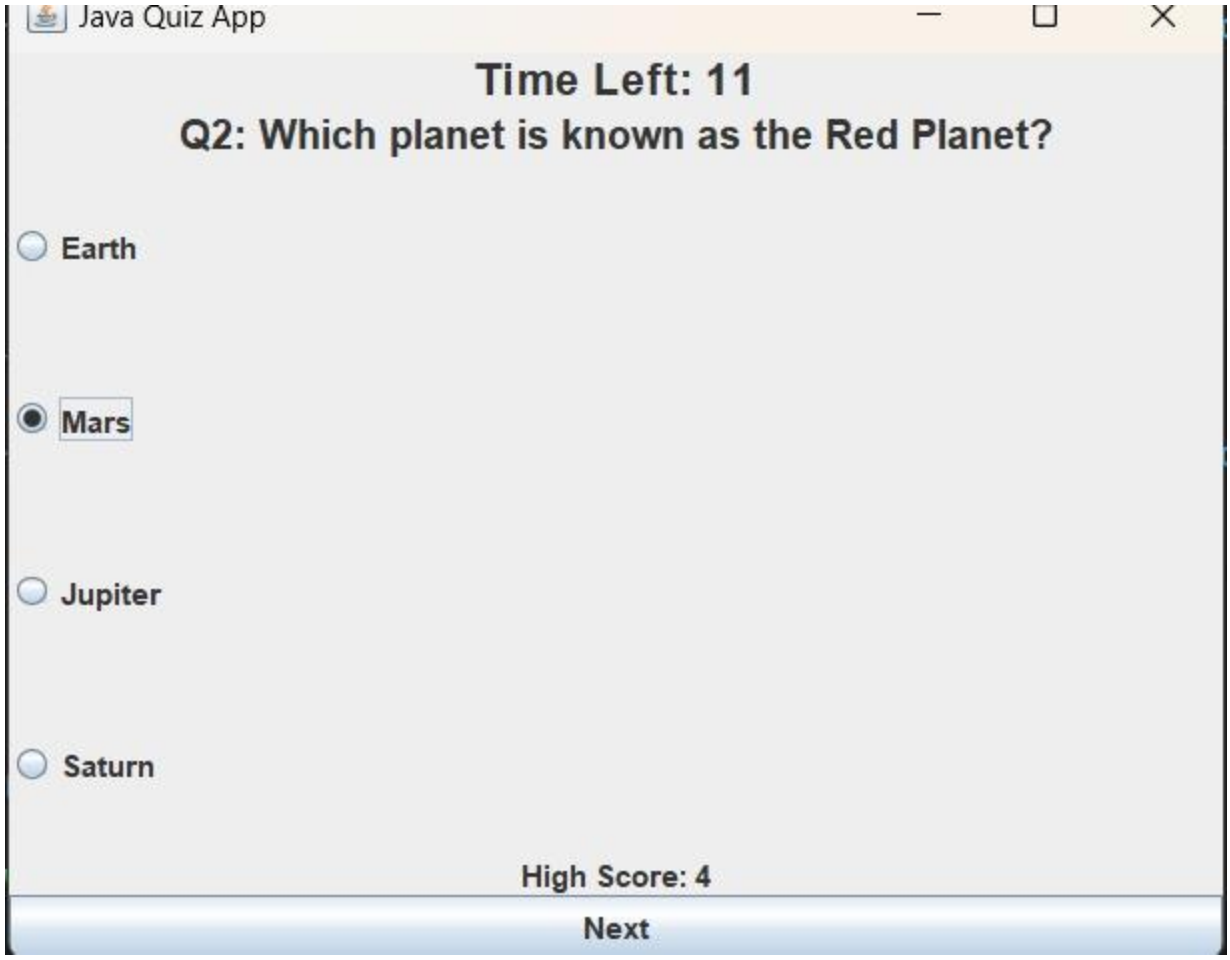
Upon execution:

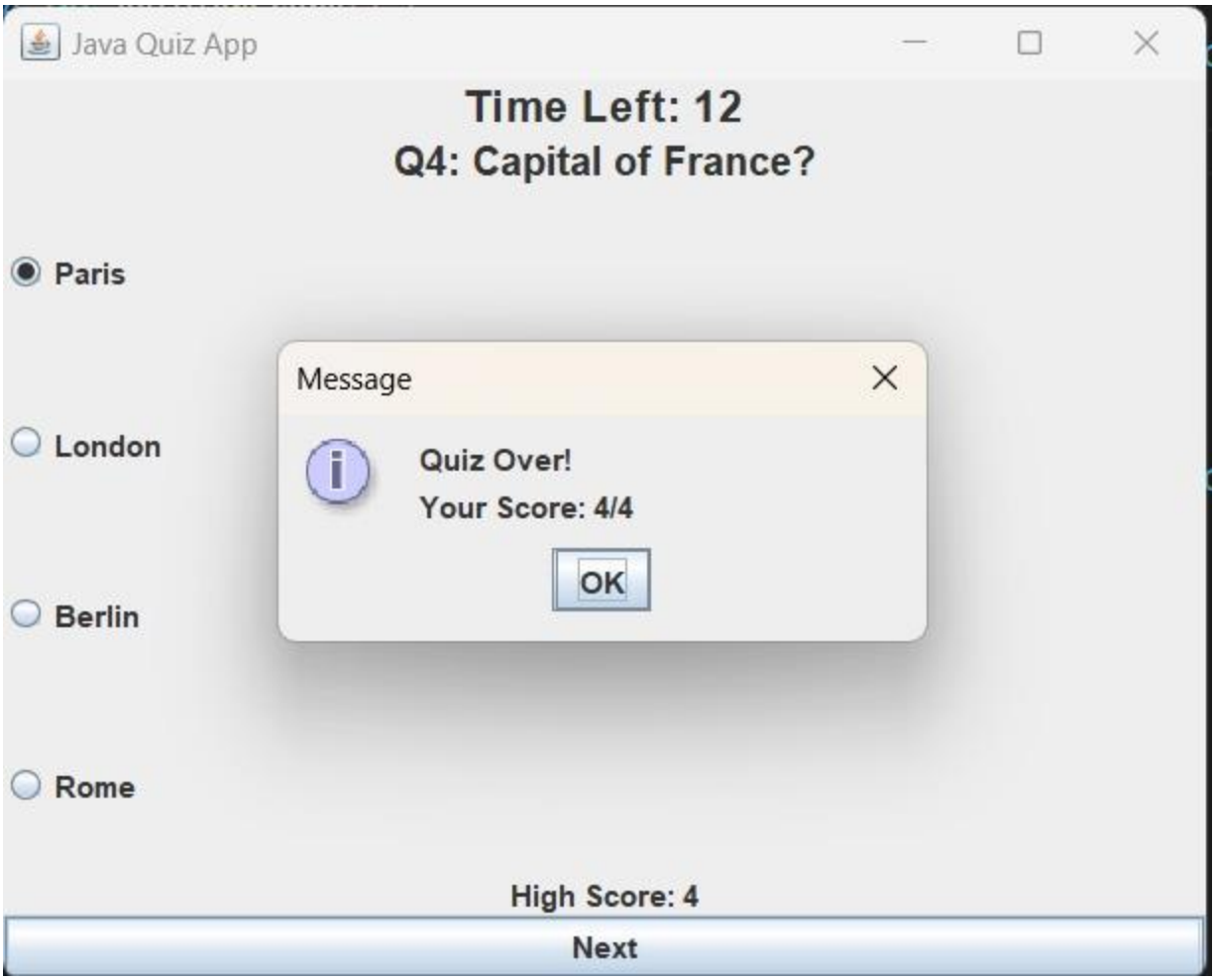
- A window opens with a question, four options, and a timer.
- The user selects the correct answer and clicks "Next".
- If time runs out, the system auto-skips to the next question.
- After the last question, the user's score and the current high score are displayed in a dialog.

Sample Output Screenshots

- Start of Quiz
- Mid-Quiz with Timer Running
- Final Score Message Box
- High Score File Content (highscore.txt)

- 
Java Quiz App
Time Left: 12
Q1: What is 2 + 2?
3
4
5
6
High Score: 4
Next

- 
Java Quiz App
Time Left: 11
Q2: Which planet is known as the Red Planet?
Earth
Mars
Jupiter
Saturn
High Score: 4
Next

- 

6. Conclusion:

The Java GUI Quiz Application successfully demonstrates the use of **Swing**, **event-driven programming**, and **file handling** in Java. The project is modular, user-friendly, and easily extendable for more features like user login, subject categories, or network-based multi-user support. This project reflects a solid understanding of Java development and GUI systems.

📎 Appendix: Full Code

Refer to the Java file QuizApp.java included with this report

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.List;
import java.io.*;

public class QuizApp extends JFrame {
    // Question model
    static class Question {
        String questionText;
        String[] options;
        int correctOptionIndex;

        Question(String q, String[] opts, int correct) {
            this.questionText = q;
            this.options = opts;
            this.correctOptionIndex = correct;
        }
    }
```

```
// Quiz data
private List<Question> questions = new ArrayList<>();
private int currentQuestionIndex = 0;
private int score = 0;
private int timerSeconds = 15;
private Timer countdownTimer;

// GUI Components
private JLabel questionLabel, timerLabel;
private JRadioButton[] optionButtons = new JRadioButton[4];
private ButtonGroup optionGroup = new ButtonGroup();
private JButton nextButton;
private JLabel highScoreLabel;

// File for high score
private final String HIGH_SCORE_FILE = "highscore.txt";

public QuizApp() {
   loadQuestions();
```

```
        initGUI();
        showQuestion();
    }

    // Load sample questions
    private void loadQuestions() {
        questions.add(new Question("What is 2 + 2?", new String[]
        {"3", "4", "5", "6"}, 1));
        questions.add(new Question("Which planet is known as the Red
Planet?", new String[]
        {"Earth", "Mars", "Jupiter", "Saturn"}, 1));
        questions.add(new Question("Who wrote 'Romeo and Juliet'?", new
String[]
        {"Dickens", "Shakespeare", "Hemingway", "Twain"}, 1));
        questions.add(new Question("Capital of France?", new String[]
        {"Paris", "London", "Berlin", "Rome"}, 0));
    }

    // Initialize GUI
    private void initGUI() {
```

```
setTitle("Java Quiz App");
setSize(500, 400);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setLayout(new BorderLayout());

// Top Panel
JPanel topPanel = new JPanel(new GridLayout(2, 1));
timerLabel = new JLabel("Time Left: 15", JLabel.CENTER);
timerLabel.setFont(new Font("Arial", Font.BOLD, 18));
questionLabel = new JLabel("Question", JLabel.CENTER);
questionLabel.setFont(new Font("Arial", Font.BOLD, 16));
topPanel.add(timerLabel);
topPanel.add(questionLabel);
add(topPanel, BorderLayout.NORTH);

// Center Options Panel
JPanel optionsPanel = new JPanel(new GridLayout(4, 1));
for (int i = 0; i < 4; i++) {
   optionButtons[i] = new JRadioButton();
   optionGroup.add(optionButtons[i]);
```

```
            optionsPanel.add(optionButtons[i]);
        }
        add(optionsPanel, BorderLayout.CENTER);

        // Bottom Panel
        JPanel bottomPanel = new JPanel(new BorderLayout());
        nextButton = new JButton("Next");
        highScoreLabel = new JLabel("High Score: " + getHighScore(),
JLabel.CENTER);
        bottomPanel.add(highScoreLabel, BorderLayout.NORTH);
        bottomPanel.add(nextButton, BorderLayout.SOUTH);
        add(bottomPanel, BorderLayout.SOUTH);

        nextButton.addActionListener(e -> handleNext());

        setVisible(true);
    }

    // Show the current question
    private void showQuestion() {
```

```
    if (currentQuestionIndex >= questions.size()) {
        endQuiz();
        return;
    }

    Question q = questions.get(currentQuestionIndex);
    questionLabel.setText("Q" + (currentQuestionIndex + 1) + ": " +
q.questionText);
    for (int i = 0; i < 4; i++) {
        optionButtons[i].setText(q.options[i]);
        optionButtons[i].setSelected(false);
    }

    // Reset and start timer
    if (countdownTimer != null) countdownTimer.stop();
    timerSeconds = 15;
    timerLabel.setText("Time Left: " + timerSeconds);
    countdownTimer = new Timer(1000, e -> {
        timerSeconds--;
        timerLabel.setText("Time Left: " + timerSeconds);
```

```
        if (timerSeconds == 0) {
            ((Timer) e.getSource()).stop();
            handleNext();  // Auto skip
        }
    });
    countdownTimer.start();
}

// Handle Next button or timeout
private void handleNext() {
    countdownTimer.stop();
    Question q = questions.get(currentQuestionIndex);

    for (int i = 0; i < 4; i++) {
        if (optionButtons[i].isSelected()) {
            if (i == q.correctOptionIndex) {
                score++;
            }
            break;
        }
```

```
    }

    currentQuestionIndex++;
    showQuestion();
}

// End the quiz
private void endQuiz() {
    int highScore = getHighScore();
    if (score > highScore) {
        saveHighScore(score);
        highScoreLabel.setText("New High Score: " + score);
    } else {
        highScoreLabel.setText("High Score: " + highScore);
    }

    JOptionPane.showMessageDialog(this, "Quiz Over!\nYour Score: "
    + score + "/" + questions.size());
    System.exit(0);
}
```

```
    // Get high score from file
    private int getHighScore() {
        try (BufferedReader reader = new BufferedReader(new
FileReader(HIGH_SCORE_FILE))) {
            return Integer.parseInt(reader.readLine());
        } catch (IOException | NumberFormatException e) {
            return 0;
        }
    }

    // Save new high score
    private void saveHighScore(int newScore) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(HIGH_SCORE_FILE))) {
            writer.write(String.valueOf(newScore));
        } catch (IOException e) {
            System.err.println("Failed to save high score.");
        }
    }
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new QuizApp());
    }
}
```