K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

A Project Report

on

EXAM MANAGEMENT SYSTEM

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Ms. Hema R., M.E.,

Assistant Professor / Information Technology

Submitted By

GAYATHRI K K (2303811710622026)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(Autonomous)

TRICHY - 621112

DECEMBER 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University, Chennai)

TRICHY - 621112

BONAFIDE CERTIFICATE

Certified that this project report on **"EXAM MANAGEMENT SYSTEM"** is the Bonafide work of **GAYATHRI K K (2303811710622026)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature

Ms. HEMA R., M.E.,

SUPERVISOR,

Department of Information Technology,

K. Ramakrishnan College of Technology,

Trichy - 621112

Signature

Dr. SYEDAKBAR S., M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of ECE,

K. Ramakrishnan College of Technology,

Trichy – 621112

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To create innovative and socially responsible Electronics and Communication Engineers with design skills and research focus to meet Societal and Industrial needs.

MISSION OF THE DEPARTMENT

- M1: To provide high quality education and professional ethics to students through enhanced learning environment
- M2: To impart a creative environment towards centre of excellence in department with design skill and exposure for research.
- M3: To nurture required employable skills of students to satisfy the industry and social needs with ethical and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

- PEO1: Core Knowledge Development: Graduates will have enhanced engineering skills in the field of electronics, communication and interdisciplinary areas to serve the society with global standards.

- PEO2: Professional development: Graduates will apply the technical knowledge for continuous up gradation of their professional skills to become an inimitable employee, researcher or entrepreneur.
- PEO3: Analytical Thinking: Graduates will have analytic and thinking skills to provide the innovative solutions for industry and societal requirements.

PROGRAM OUTCOMES

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 modelling 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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- PSO1: To analyse, design and develop solutions by applying foundational concepts of electronics and communication engineering.
- PSO2: To apply design principles and best practices for developing quality products for scientific and business applications.

ABSTRACT

The ExamManagementSystem is a Java Swing-based application designed to facilitate the management of online exams. The application is structured with a CardLayout that enables smooth transitions between different stages: user login, exam creation, and taking the exam. After logging in, the user can create an exam by entering a title and a series of questions, which are separated by semicolons. The system then presents the questions one by one, with a timer counting down from a set duration of 2 hours. The user can navigate through the questions using a "Next Question" button. The remaining time is displayed on the interface and updated every second. Once the timer reaches zero, the exam automatically ends, and the user is notified that the time is up. This application provides a straightforward way to create and manage online exams, with built-in time tracking and question navigation features.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	Pos MAPPED	PSOs MAPPED
Understand and apply object-oriented programming concepts like encapsulation, abstraction, and polymorphism.	**1**	**1**
Develop modular software solutions to address real-world problems using Java.	**2**	**2**
Analyze and implement efficient algorithms for managing structured data in software systems	**5**	**6**

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

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The Exam Management System is a Java Swing application for managing online exams. It allows users to log in, create an exam with multiple questions, and take the exam within a set time limit. The system features a timer that counts down from 2 hours, displaying one question at a time. Users can navigate through the questions using a "Next Question" button. Once the timer expires, the exam automatically ends. The application simplifies exam creation, management, and time tracking.

1.1 Objective

The Exam Management System code is designed to create a basic Java-based graphical user interface (GUI) for managing online exams. The system begins with a login panel where the user can input their username and password. Upon successful login, the user can proceed to the exam creation panel, where they can enter the exam title and a series of questions (separated by semicolons). These questions are stored and later displayed during the exam. The system enables the creation of custom exams, allowing for flexibility in the number and type of questions.

Once the exam is created, the system transitions to the exam interface where the user can view and navigate through the questions. Each question is displayed in a non-editable text area, and the user can move to the next question using a "Next Question" button. Additionally, the system includes a countdown timer that tracks the remaining exam time in hours, minutes, and seconds. The timer updates every second, and once it reaches zero, or when all questions are answered, the exam ends with a notification that the time has expired. This application serves as a foundational framework for creating and managing timed online exams.

1.2 Overview

The Exam Management System is a Java-based GUI application for managing online exams. It allows users to log in, create custom exams with multiple questions, and navigate through them during the exam. A countdown timer tracks the remaining time for the exam, updating every second. When time runs out or all questions are answered, the exam ends with a notification.

1.3 Java Programming Concepts

- **Object-Oriented Programming (OOP)**: The Exam ManagementSystem follows object-oriented principles, with the main class encapsulating all the functionality for the exam. Methods and variables are used to hide the internal workings of the system, supporting encapsulation. Event-driven programming handles user interactions with the GUI components, ensuring responsiveness.
- **Control Structures**: Control structures like if statements manage user actions, such as checking if the last question is reached or if the timer has expired. Loops are used to process and store questions entered by the user. These structures ensure correct program behavior under different conditions.
- **GUI Programming (Swing)**: The application uses Swing components like JPanel, JButton, and JTextArea to create interactive exam panels. CardLayout helps switch between different screens, while layout managers like GridLayout and BorderLayout organize the components effectively within the window.
- **Timer, Multithreading, and Collections**: A Timer runs in a separate thread to update the countdown every second, ensuring the UI remains responsive. An ArrayList stores the questions dynamically. Multithreading enables the timer and question navigation to function concurrently, improving performance and responsiveness.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work in the Exam Management System involves developing a Java-based application that manages online exams through a graphical user interface (GUI). The system starts with a login screen where the user can input their credentials. Upon successful login, the user is taken to the exam creation panel, where they can input the exam title and list questions, which are then stored in an ArrayList.

The questions are separated by semicolons and displayed sequentially during the exam. The system uses a CardLayout to switch between the login, exam creation, and exam question panels. The questions panel displays each question in a JTextArea, which is updated as the user navigates between them using a "Next Question" button.

The exam is timed using a Timer object that tracks the remaining time. The time is displayed in a JLabel in the format HH:MM:SS, and the timer updates every second. When the time runs out or the last question is reached, a message appears notifying the user that the exam is over.

This solution integrates key Java concepts such as object-oriented programming, event-driven programming, GUI development with Swing, and multithreading for the countdown timer. It ensures an interactive and smooth user experience for creating and taking exams, providing both functionality and usability. The proposed system also allows for future expansion, such as adding features like answer submission or result tracking.

2.2 Block Diagram

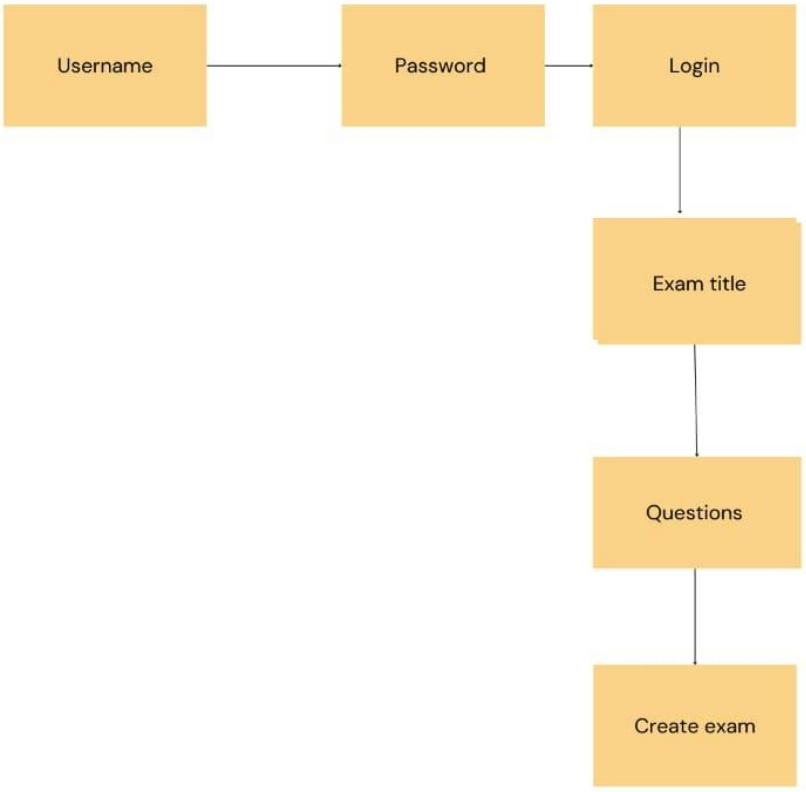

Fig2.2.1 Block Diagram

CHAPTER 3

MODULE DESCRIPTION

3.1 Login Module:

The Login Module is responsible for authenticating users before they can proceed with the exam creation or participation. It consists of a username field, a password field, and a login button. When the user enters valid credentials and clicks the login button, the system navigates to the exam creation panel. Currently, the login logic is not implemented, and the system directly proceeds to the exam creation panel upon clicking the login button. This module serves as the entry point to the system, ensuring that only authorized users can access further functionalities.

3.2 Exam Creation Module :

The Exam Creation Module allows the user to create a new exam by entering a title and a set of questions. The title of the exam is entered into a text field, while the questions are entered in a multi-line text area, separated by semicolons. The "Create Exam" button processes the input, splits the questions by the semicolons, and stores them in a list. Upon clicking the button, the system sets the current question index to zero and starts the timer for the exam (set to 2 hours by default). After this, the system navigates to the questions panel, where the user can begin answering the questions.

3.3 Question Panel Module :

The Question Panel Module is the core part of the exam-taking process. It displays the current question in a non-editable text area and includes a timer that continuously updates to show the remaining time in the exam. The user can navigate through the questions using the "Next Question" button, which increments the question index to display the subsequent question. If the user reaches the end of the question list, the system displays a message indicating that all questions have been answered. This module serves as the main interface for answering the questions and managing the exam's progress.

3.4 Timer Module :

The Timer Module controls the countdown timer for the exam duration. The remaining time is shown in a label formatted as HH:MM:SS. The timer starts when the exam begins and updates every second, decrementing the time remaining. The timer is managed by a Timer object, which schedules a task to update the time at regular intervals (every second). Once the remaining time reaches zero, the timer stops, and the system notifies the user that the exam has ended. This module ensures that the exam is timed, providing both the user and the system with real-time feedback on the time left to complete the exam.

3.5 Navigation and User Interface (UI) Module :

The **Navigation and UI Module** manages the layout and flow of the application, using CardLayout to switch between different panels (login, exam creation, and question panels). It ensures a smooth transition between these views and uses layout managers like GridLayout and BorderLayout for user-friendly arrangement of components. This module handles the visibility and interaction between the panels, making the system intuitive and easy to navigate for the user.

CHAPTER 4

RESULTS AND DISCUSSION

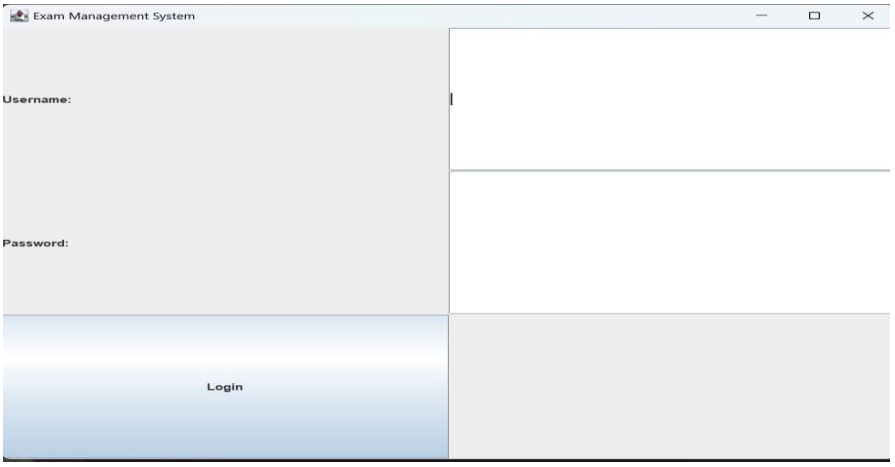

Fig:4.1 Admin Panel

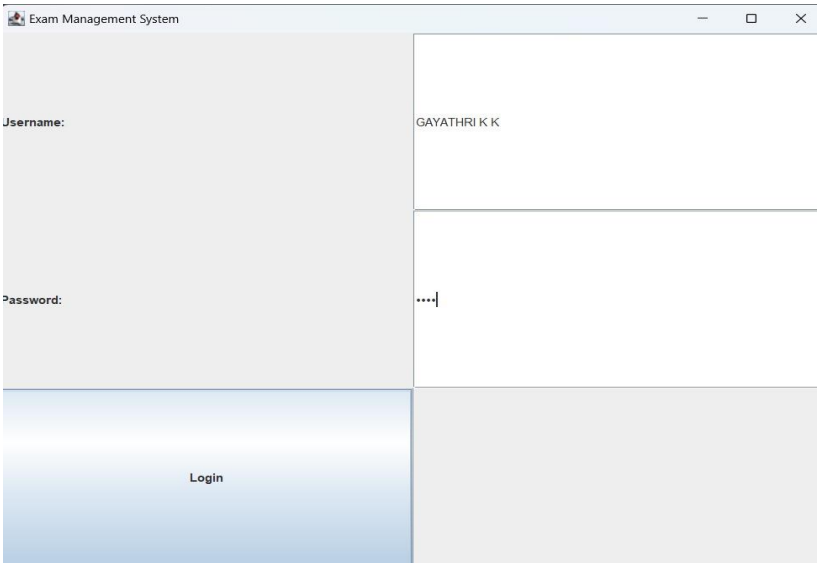

Fig:4.2 Creating user name and password to login

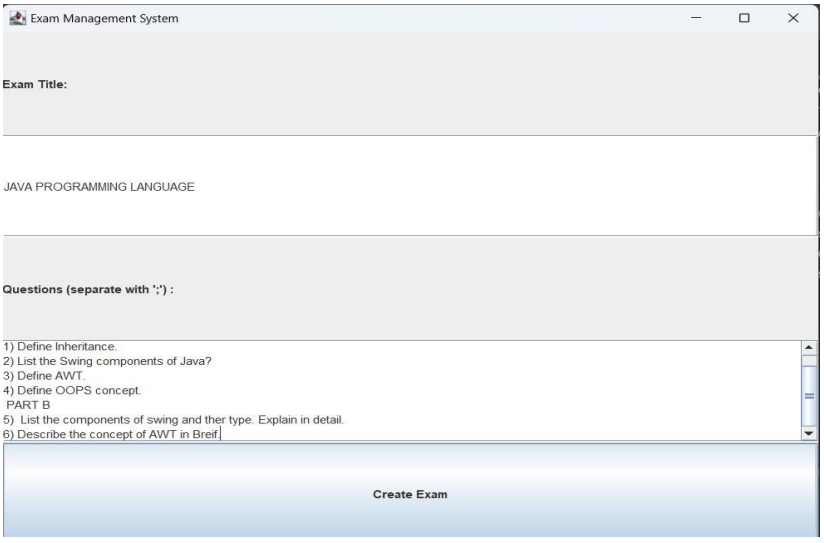

Fig 4.3 Adding Questions and create exam

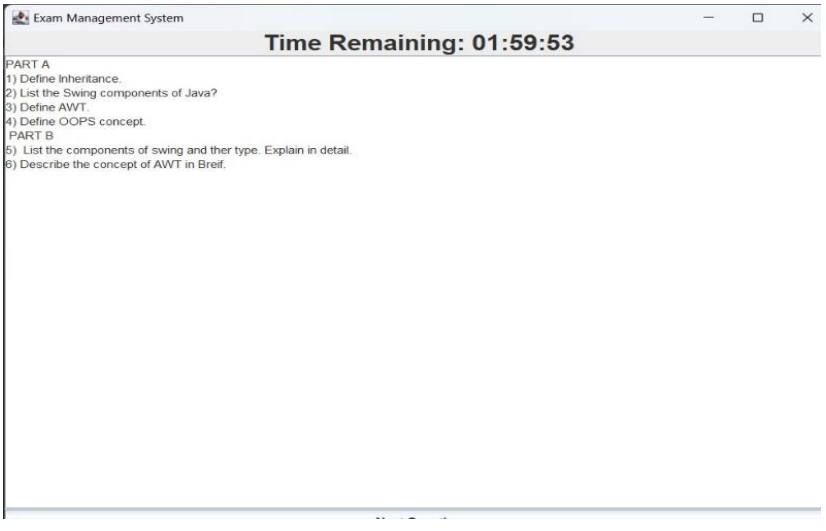

Fig 4.4 Exam created viewing question with timer

CHAPTER 5

CONCLUSION

In conclusion, the **Exam Management System** provides a functional foundation for managing timed exams. It allows users to log in, create an exam by entering questions, and navigate through them while a countdown timer tracks the remaining time. The system is implemented using Java Swing, with CardLayout to manage transitions between panels, ensuring a smooth user experience.

However, the system has several limitations. The login process lacks proper authentication, and the exam creation only supports plain text questions without options for multiple-choice or other question types. Additionally, there is no scoring or feedback mechanism, meaning the system doesn't evaluate answers after the exam ends. It also lacks the ability to save progress, so users would lose their data if they close the application.

For further development, the system could be improved by adding user authentication, supporting different question formats, and implementing an automatic grading system. Error handling and input validation would also be beneficial to prevent issues. Adding data persistence would allow users to save their progress and resume the exam later.

Overall, the system is a good starting point, and with these enhancements, it could evolve into a fully-fledged exam management platform, suitable for real-world use.

REFERENCES

Java Swing Basics:

Orcale's Official Swing Tutorial:https://docs.orcale.com/javase/tutorial/uiseing..Covers how to build GUIs using Swing components.

Layouts in Swing:

Orcale's Layout Manager Tutorial:https://docs.orcale.com/javase/tutorial/uiswing/layout/index.html. Detialed explanation of layout managers like CardLayout,and BorderLayout.

Event Handling:

Oracle's Event Listeners Tutorial:https://docs.oracle.com/javase/tutorial/uiswing/events/index.html.Learn about handling user input and interaction.

Java Timer and TimerTask:

Documentation:https://docs.orcale.com/javase/8/docs/api/java/util/Timer.html.Provides details on how to use timers jfor scheduling tasks.

Using Lardlayout: https://www.baeldung.com/java-cardlayout.Understand how to switch between panels dyanamically.

APPENDIX (Coding)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

public class ExamManagementSystem extends JFrame {
    private CardLayout cardLayout;
    private List<String> questions; // List to hold questions
    private int currentQuestionIndex; // Index of the current question
    private Timer examTimer; // Timer for the exam
    private int remainingTime; // Remaining time in seconds
    private JLabel timerLabel; // Label to display the timer

    public ExamManagementSystem() {
        setTitle("Exam Management System");
        setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
cardLayout = new CardLayout();
        setLayout(cardLayout);

        // Create login panel
        JPanel loginPanel = createLoginPanel();
        add(loginPanel, "Login");

        // Create exam creation panel
        JPanel examCreationPanel =
createExamCreationPanel();
        add(examCreationPanel, "Create Exam");

        // Create questions panel
        JPanel questionsPanel =
createQuestionsPanel();
        add(questionsPanel, "Questions");

        // Show login panel initially
        cardLayout.show(getContentPane(),
"Login");
    }

    private JPanel createLoginPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 2));

        JTextField usernameField = new
JTextField();
        JPasswordField passwordField = new
JPasswordField();
        JButton loginButton = new JButton("Login");
```

```
 panel.add(new JLabel("Username:"));
        panel.add(usernameField);
        panel.add(new JLabel("Password:"));
        panel.add(passwordField);
        panel.add(loginButton);

        loginButton.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
e) {
                // Handle login logic here
                // If successful, show the exam creation
panel
                cardLayout.show(getContentPane(),
"Create Exam");
            }
        });

        return panel;
    }

    private JPanel createExamCreationPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(5, 2));

        JTextField examTitleField = new
JTextField();
        JTextArea questionField = new JTextArea();
        JButton createExamButton = new
```

```
JButton("Create Exam");
  panel.add(new JLabel("Exam Title:"));
      panel.add(examTitleField);
      panel.add(new  JLabel("Questions  (separate
with ';') :"));
      panel.add(new JScrollPane(questionField));
      panel.add(createExamButton);

      createExamButton.addActionListener(new
ActionListener() {
          @Override
          public void actionPerformed(ActionEvent
e) {
              // Handle exam creation logic here
              String questionsInput =
questionField.getText();
              questions = new ArrayList<>();
              for (String question :
questionsInput.split(";")) {
                 questions.add(question.trim());
              }

              // Show the questions panel with the
created questions
              currentQuestionIndex = 0; // Reset to the
first question
              updateQuestionDisplay((JTextArea)
((JScrollPane) ((JPanel)
getContentPane().getComponent(2)).getCompone
nt(0)).getViewport().getView());
              startTimer(120); // Start timer for 2
```

```
hours (120 minutes)
cardLayout.show(getContentPane(),

"Questions");
            }
        });

        return panel;
    }

    private JPanel createQuestionsPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());

        JTextArea questionsArea = new JTextArea();
        questionsArea.setEditable(false);
        panel.add(new    JScrollPane(questionsArea),
BorderLayout.CENTER);

        timerLabel = new JLabel("Time Remaining:
02:00:00", SwingConstants.CENTER);
        timerLabel.setFont(new            Font("Arial",
Font.BOLD, 24));
        panel.add(timerLabel,
BorderLayout.NORTH);

        JButton nextButton = new JButton("Next
Question");
        nextButton.addActionListener(new
ActionListener() {
```

```
 @Override
            public void actionPerformed(ActionEvent
e) {

if (currentQuestionIndex <
questions.size() - 1) {
                    currentQuestionIndex++;

updateQuestionDisplay(questionsArea);
                } else {

JOptionPane.showMessageDialog(panel,      "You
have reached the end of the questions.");
                }
            }
        });
        panel.add(nextButton,
BorderLayout.SOUTH);

        return panel;
    }

    private void startTimer(int duration) {
        remainingTime = duration * 60; // Convert to
seconds
        updateTimerLabel();

        examTimer = new Timer();
        examTimer.scheduleAtFixedRate(new
TimerTask() {
            @Override
            public void run() {
```

```
if (remainingTime > 0) {
                remainingTime--;
 updateTimerLabel();
            } else {
                examTimer.cancel();
JOptionPane.showMessageDialog(getContentPan
e(), "Time is up! The exam has ended.");
                // Optionally, you can navigate back
to the login or main menu
            }
          }
       }, 0, 1000); // Update every second
    }

    private void updateTimerLabel() {
       int hours = remainingTime / 3600;
       int minutes = (remainingTime % 3600) / 60;
       int seconds = remainingTime % 60;
       timerLabel.setText(String.format("Time
Remaining: %02d:%02d:%02d", hours, minutes,
seconds));
    }

    private void updateQuestionDisplay(JTextArea
questionsArea) {
       if (currentQuestionIndex < questions.size()) {

questionsArea.setText(questions.get(currentQuest
ionIndex));
```

```
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            ExamManagementSystem app = new
ExamManagementSystem();




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