

EXAM MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

SAHANA P (2303811724322093)

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

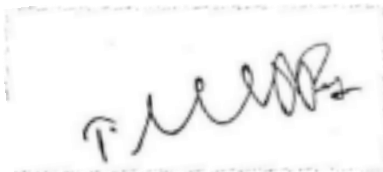
DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “ **EXAM MANAGEMENT SYSTEM**” is the Bonafide work of **SAHANA P (2303811724322093)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



Signature

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

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,
K. Ramakrishnan College of Engineering,
Samayapuram, Trichy -621 112.



Signature

Mrs. S. GEETHA M.E.,

SUPERVISOR,

Department of Artificial Intelligence,
K. Ramakrishnan College of Engineering,
Samayapuram, Trichy -621 112.

S

Submitted for the viva-voce examination held on 3.12.24



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**EXAM MANAGEMENT SYSTEM**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.



SAHANA P

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”**, for providing us with the opportunity to do this project.

I extend our sincere acknowledgement and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

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 modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The Exam Management System is a Java-based software designed to simplify and enhance the process of creating, scheduling, administering, and grading exams. By leveraging a secure and user-friendly platform, the system facilitates seamless interactions for instructors and students while ensuring academic integrity. The comprehensive and scalable design enables institutions to manage examinations efficiently, reducing manual errors and improving administrative workflows. Key features include automated exam scheduling, question paper management, real-time grading, and robust security measures to prevent unauthorized access. This system serves as a versatile solution for modern educational needs, fostering transparency and efficiency in the examination process.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	1
2	PROJECT METHODOLOGY	2
	2.1 PROPOSED WORK	2
	2.2 BLOCK DIAGRAM	3
3	JAVA PROGRAMMING CONCEPTS	4
	3.1 OBJECT ORIENTED PROGRAMMING	4
	3.2 EXCEPTION HANDLING	5
4	MODULE DESCRIPTION	6
	4.1 EXAM MODULE	6
	4.2 STUDENT MODULE	6
	4.3 LOGIN MODULE	6
	4.4 DRIVER MODULE	7
5	CONCLUSION	8
	REFERENCES	9
	APPENDICES	10
	Appendix A – Source code	10
	Appendix B – Screen shots	20

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Examinations are critical to assessing students' academic performance, yet traditional methods often involve time-intensive processes prone to errors. The Exam Management System aims to address these challenges by offering a digital platform that automates exam-related tasks. Developed using Java, the system integrates functionalities such as question paper creation, scheduling, secure administration, and automated grading. It provides a role-based interface, ensuring instructors and students have access to relevant tools while maintaining data security and academic integrity. This project is tailored to meet the needs of educational institutions seeking to improve their examination workflows with a scalable and user-centric solution.

1.2 OBJECTIVE

The objective of the Exam Management System is to create a scalable and efficient platform that streamlines the entire lifecycle of examinations, including creation, scheduling, administration, and grading. The system aims to reduce manual effort by automating key processes, thereby enhancing productivity and minimizing errors. By offering a secure and intuitive interface, it caters to the needs of both instructors and students, ensuring seamless access to relevant functionalities. Additionally, the system emphasizes maintaining academic integrity through robust security measures such as role-based access control and encryption. Designed with scalability in mind, the platform can adapt to evolving institutional requirements, providing a reliable and transparent solution for managing examinations in educational settings.

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The development of the Exam Management System involves the following phases:

1. Requirement Gathering:

- Identify key functionalities needed by stakeholders, such as exam creation, scheduling, and grading.
- Define roles (Admin, Instructor, Student) and their permissions.

2. System Design:

- Create UML diagrams (use case, class, and sequence diagrams) to define system architecture.
- Design a database schema for managing users, exams, and results.

3. Development:

- Build the system using Java for the backend, employing frameworks like JavaFX or Swing for the GUI.
- Implement database integration using JDBC with a relational database such as MySQL.

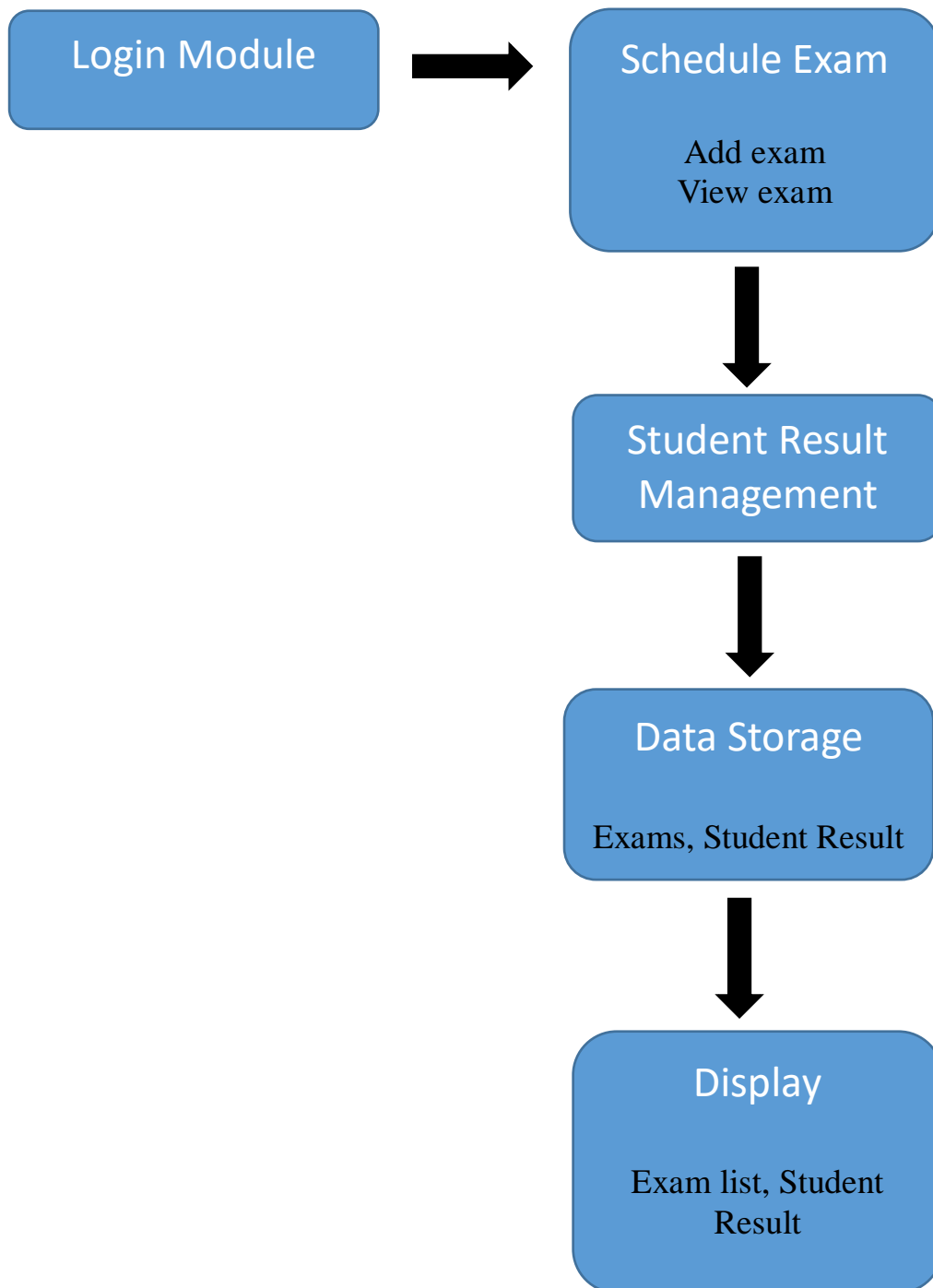
4. Testing:

- Conduct unit, integration, and system testing to ensure the functionality and reliability of the platform.
- Perform security testing to safeguard data and prevent unauthorized access.

5. Deployment and Maintenance:

- Deploy the system in an educational institution and provide training for instructors and administrators.
- Maintain and update the system as per user feedback and evolving needs.

2.2 BLOCK DIAGRAM



CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1 Object-Oriented Programming (OOP):

- **Encapsulation:** Data related to exams, users, and results are encapsulated into classes like Exam, User, and Result, with private attributes and public getter/setter methods.
- **Inheritance:** Role-specific functionality is implemented using inheritance. For example, Admin, Instructor, and Student classes can inherit from a base User class.
- **Polymorphism:** Polymorphic methods enable customized actions, such as generating reports differently for students and instructors.
- **Abstraction:** Interfaces and abstract classes define core functionalities (e.g., GradingInterface for grading logic), separating implementation details from the system design.

Collection Framework:

- **ArrayList, HashMap and HashSet:** Collections are used for managing dynamic data like lists of users, exams, and questions efficiently.
- **Iterators:** Iterators allow seamless traversal of lists for processing exams, results, or users.

3.2 Exception Handling:

- **Try-Catch Blocks:** Exception handling ensures the system gracefully manages errors, such as invalid inputs or database connection failures.
- **Custom Exceptions:** Custom exceptions like `InvalidLoginException` or `ExamNotFoundException` provide meaningful error handling for specific scenarios.

File Handling and Serialization:

- **File I/O:** Used for exporting results, question papers, or logs to external files.
- **Serialization:** Enables saving and retrieving Java objects (e.g., `Exam` or `User` objects) to/from files or databases.

CHAPTER 4

MODULE DESCRIPTION

4.1 EXAM MODULE:

This module is responsible for managing the exams in the system, including the creation and viewing of exam details.

Core Functions:

Add Exam: Allows users to create a new exam by specifying the exam name and date.

View Exams: Displays the list of all exams currently stored in the system.

4.2 STUDENT MODULE:

This module is responsible for managing students and their results, allowing users to add student information and their exam scores.

Core Functions:

Add Student Result: Allows users to enter a student's name, the exam they are associated with, and the score they received.

View Student Results: Displays the results of students who have taken a specific exam, including their scores.

4.3 LOGIN MODULE:

This module handles the authentication process, ensuring that only authorized users

(in this case, an "admin") can access the system.

Core Functions:

Login: Prompts the user to enter a username and password. If the credentials are valid (e.g., username: admin, password: admin), access to the system is granted.

Authentication: Verifies the entered username and password against hardcoded values or other authentication methods.

4.4 DRIVER MODULE:

This module is responsible for the graphical user interface (GUI), handling how users interact with the system through buttons, text fields, and dialogs.

Core Functions:

Display Exam List: Shows a list of all exams available in the system.

Add New Exam: Provides a form for adding a new exam.

Add Student Result: Provides a form to enter a student's result for a specific exam.

View Student Results: Displays the results for students who have taken a particular exam.

CHAPTER 5

CONCLUSION

In conclusion, the Exam Management System provides an efficient, user-friendly, and scalable solution for managing the lifecycle of exams in an educational setting. By incorporating core Java programming concepts such as object-oriented programming, collections, and exception handling, the system offers a robust platform for administrators, instructors, and students to interact with. The system allows for the creation and scheduling of exams, the addition of questions, and provides students with a straightforward interface to attempt exams and view their results.

This project automates various time-consuming processes, such as scoring and result management, thereby improving operational efficiency and reducing the chances of human error. Additionally, the modular architecture of the system makes it easy to extend with new features, such as database integration, role-based access control, and support for different question types. The scalability and flexibility of the system ensure that it can be adapted to meet the evolving needs of educational institutions.

Ultimately, the Exam Management System aims to streamline the examination process, maintain academic integrity, and enhance the overall educational experience for both students and administrators. By providing a secure and organized platform for managing exams, the system supports effective exam administration and enables real-time feedback for students.

REFERENCES:

1.Oracle Java Documentation

2.Official Java language reference and API documentation.

<https://docs.oracle.com/en/java/>

GeeksforGeeks Java Tutorials

3.Comprehensive guides and problem-solving examples.

<https://www.geeksforgeeks.org/java/>

W3Schools Java Tutorial

4.Simple and beginner-friendly tutorials with examples.

<https://www.w3schools.com/java/>

TutorialsPoint Java Guide

5.Detailed Java tutorials and coding examples.

<https://www.tutorialspoint.com/java/index.htm>

JavaTpoint Java Tutorials

6.Covers basic to advanced Java concepts. <https://www.javatpoint.com/java-tutorial>

APPENDICES

APPENDIX A – SOURCE CODE

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

// Main EMS Class
public class EMS {
    private JFrame frame;
    private JTextArea textArea;
    private ArrayList<Exam> exams;
    private ArrayList<Student> students;

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                EMS window = new EMS();
                window.frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    public EMS() {
        exams = new ArrayList<>();
        students = new ArrayList<>();
    }
}
```

```

        initialize();
    }
    private void initialize() {
        // Frame setup
        frame = new JFrame();
        frame.setTitle("Exam Management System");
        frame.setBounds(100, 100, 700, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Set background color
        frame.getContentPane().setBackground(new Color(245, 245, 245));

        // Panel for buttons
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(1, 4, 10, 10));
        panel.setBackground(new Color(40, 40, 40));
        frame.getContentPane().add(panel, BorderLayout.NORTH);

        // Buttons with styling
        JButton btnAddExam = createStyledButton("Add Exam", "Enter exam details
to add.");
        btnAddExam.addActionListener(e -> showAddExamDialog());
        panel.add(btnAddExam);

        JButton btnViewExams = createStyledButton("View Exams", "View list of all
exams.");
        btnViewExams.addActionListener(e -> viewExams());
        panel.add(btnViewExams);

        JButton btnAddStudent = createStyledButton("Add Student Result", "Add
student score for an exam.");

```

```

        btnAddStudent.addActionListener(e -> showAddStudentResultDialog());
        panel.add(btnAddStudent);

        JButton btnViewStudentResults = createStyledButton("View Student Results",
"View student results for a specific exam.");

        btnViewStudentResults.addActionListener(e ->
viewStudentResultsForExam());

        panel.add(btnViewStudentResults);
        // Text area to display results
        textArea = new JTextArea();
        textArea.setEditable(false);
        textArea.setBackground(new Color(255, 255, 255));
        textArea.setFont(new Font("Arial", Font.PLAIN, 14));
        frame.getContentPane().add(new JScrollPane(textArea),
BorderLayout.CENTER);

        // Login to proceed to the exam management screen
        login();
    }

    // Create a custom-styled button
    private JButton createStyledButton(String text, String tooltip) {
        JButton button = new JButton(text);
        button.setFont(new Font("Arial", Font.BOLD, 14));
        button.setForeground(Color.WHITE);
        button.setBackground(new Color(100, 149, 237)); // Cornflower blue
        button.setFocusPainted(false);
        button.setToolTipText(tooltip);
        button.setBorder(BorderFactory.createLineBorder(new Color(70, 130, 180),
2));    button.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        button.setPreferredSize(new Dimension(160, 40));
    }

```

```

        return button;
    }

// Modified login method to take username and password in a single dialog
private void login() {
    // Create a JPanel for the username and password input
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(2, 2, 10, 10));

    // Create text fields for username and password
    JTextField tfUsername = new JTextField();
    JPasswordField pfPassword = new JPasswordField();

    panel.add(new JLabel("Username:"));
    panel.add(tfUsername);
    panel.add(new JLabel("Password:"));
    panel.add(pfPassword);

    // Show the dialog with the panel
    int option = JOptionPane.showConfirmDialog(frame, panel, "Login",
        JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

    // Check if user pressed OK
    if (option == JOptionPane.OK_OPTION) {
        String username = tfUsername.getText();
        String password = new String(pfPassword.getPassword());

        // Validate the login credentials
        if (username.equals("admin") && password.equals("admin")) {

```

```

        showCustomMessage("Login Successful", "You have successfully logged
in!", JOptionPane.INFORMATION_MESSAGE);
    } else {
        showCustomMessage("Invalid Credentials", "Please check your username
and password.", JOptionPane.ERROR_MESSAGE);
        System.exit(0); // Exit the program if invalid credentials
    }
} else {
    System.exit(0); // Exit if user cancels
}
}

```

// Show dialog for adding an exam

```

private void showAddExamDialog() {
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(2, 2, 10, 10));

    JTextField tfExamName = new JTextField();
    JTextField tfExamDate = new JTextField();

    panel.add(new JLabel("Exam Name:"));
    panel.add(tfExamName);
    panel.add(new JLabel("Exam Date (yyyy-mm-dd):"));
    panel.add(tfExamDate);

    int option = JOptionPane.showConfirmDialog(frame, panel, "Add Exam",
JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

    if (option == JOptionPane.OK_OPTION) {

```

```

String examName = tfExamName.getText();
String examDate = tfExamDate.getText();

if (!examName.isEmpty() && !examDate.isEmpty()) {
    exams.add(new Exam(examName, examDate));
    showCustomMessage("Exam Added Successfully", "The exam has been
successfully added to the system.", JOptionPane.INFORMATION_MESSAGE);
} else {
    showCustomMessage("Invalid Input", "Please provide both exam name
and date.", JOptionPane.ERROR_MESSAGE);
}
}
}

```

// Show dialog for adding student result

```

private void showAddStudentResultDialog() {
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(3, 2, 10, 10));

    JTextField tfStudentName = new JTextField();
    JTextField tfExamName = new JTextField();
    JTextField tfScore = new JTextField();

    panel.add(new JLabel("Student Name:"));
    panel.add(tfStudentName);
    panel.add(new JLabel("Exam Name:"));
    panel.add(tfExamName);
    panel.add(new JLabel("Score:"));
    panel.add(tfScore);
}

```



```

int option = JOptionPane.showConfirmDialog(frame, panel, "Add Student
Result", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

if (option == JOptionPane.OK_OPTION) {
    String studentName = tfStudentName.getText();
    String examName = tfExamName.getText();
    String scoreStr = tfScore.getText();

    try {
        int score = Integer.parseInt(scoreStr);

        Exam exam = getExamByName(examName);
        if (exam != null) {
            students.add(new Student(studentName, exam, score));
            showCustomMessage("Student Result Added", "The student result has
been successfully recorded.", JOptionPane.INFORMATION_MESSAGE);
        } else {
            showCustomMessage("Exam Not Found", "The exam name provided
does not exist.", JOptionPane.ERROR_MESSAGE);
        }
    } catch (NumberFormatException e) {
        showCustomMessage("Invalid Input", "Please enter a valid score.",
JOptionPane.ERROR_MESSAGE);
    }
}

// Helper method to fetch an exam by its name

```

```

private Exam getExamByName(String name) {
    for (Exam exam : exams) {
        if (exam.getName().equalsIgnoreCase(name)) {
            return exam;
        }
    }
    return null; // Return null if the exam is not found
}

```

// Method to view all exams

```

private void viewExams() {
    textArea.setText("Exams:\n");
    for (Exam exam : exams) {
        textArea.append(exam + "\n");
    }
}

```

// Method to view student results for a specific exam

```

private void viewStudentResultsForExam() {
    String examName = JOptionPane.showInputDialog(frame, "Enter Exam Name
to View Results:");

```

```

    Exam exam = getExamByName(examName);
    if (exam != null) {
        // Display all students who took this exam and their scores
        textArea.setText("Student Results for Exam: " + exam.getName() + "\n");
        boolean found = false;
        for (Student student : students) {
            if (student.getExam().getName().equalsIgnoreCase(exam.getName())) {

```

```

        textArea.append(student + "\n");
        found = true;
    }
}

if (!found) {
    textArea.append("No students have taken this exam yet.\n");
}
} else {
    showCustomMessage("Exam Not Found", "The exam name provided does
not exist.", JOptionPane.ERROR_MESSAGE);
}
}

// Show custom message with icon and style
private void showCustomMessage(String title, String message, int messageType)
{
    JOptionPane.showMessageDialog(frame, message, title, messageType);
}

// Inner Exam class to represent an exam
public static class Exam {
    private String name;
    private String date;

    public Exam(String name, String date) {
        this.name = name;
        this.date = date;
    }
}

```

```

    public String getName() {
        return name;
    }

    public String getDate() {
        return date;
    }

    @Override
    public String toString() {
        return "Exam: " + name + " | Date: " + date;
    }
}

// Inner Student class to represent a student
public static class Student {
    private String name;
    private Exam exam;
    private int score;

    public Student(String name, Exam exam, int score) {
        this.name = name;
        this.exam = exam;
        this.score = score;
    }

    public String getName() {
        return name;
    }

```

```

    }

    public Exam getExam() {
        return exam;
    }

    public int getScore() {
        return score;
    }

    @Override
    public String toString() {
        return "Student: " + name + " | Exam: " + exam.getName() + " | Score: " +
score;
    }
}
}

```

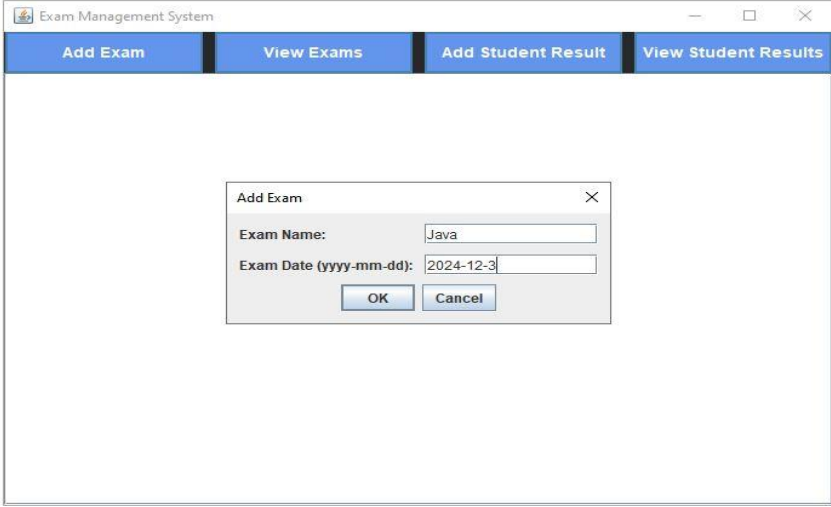
APPENDIX B - SCREENSHOTS

LOGIN PAGE:

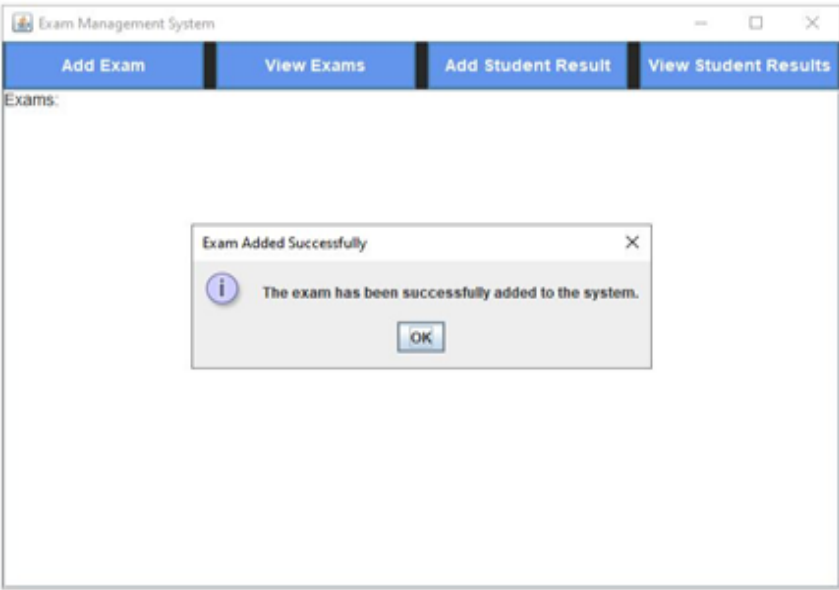


A small dialog box titled "Login" with a close button (X) in the top right corner. It contains two input fields: "Username:" and "Password:". Below the input fields are two buttons: "OK" and "Cancel".

ADD EXAM:

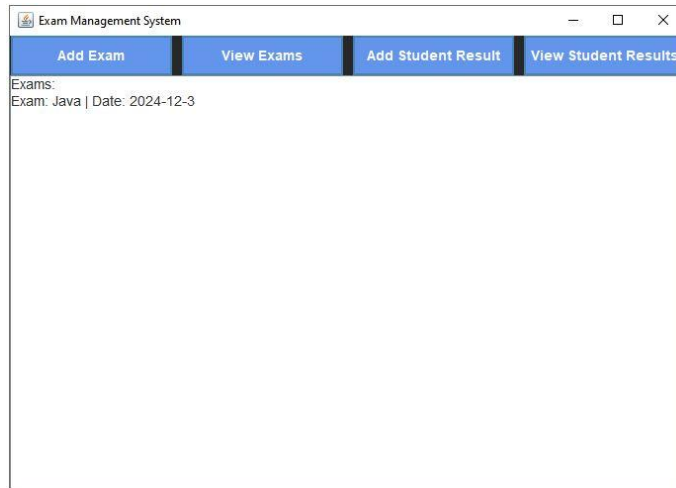


The main window of the "Exam Management System" is shown. It has a title bar with the system name and standard window controls. Below the title bar is a blue navigation bar with four tabs: "Add Exam", "View Exams", "Add Student Result", and "View Student Results". The "Add Exam" tab is selected. In the center of the window, an "Add Exam" dialog box is open. This dialog box has a close button (X) and contains two input fields: "Exam Name:" with the text "Java" and "Exam Date (yyyy-mm-dd):" with the text "2024-12-3". Below these fields are "OK" and "Cancel" buttons.

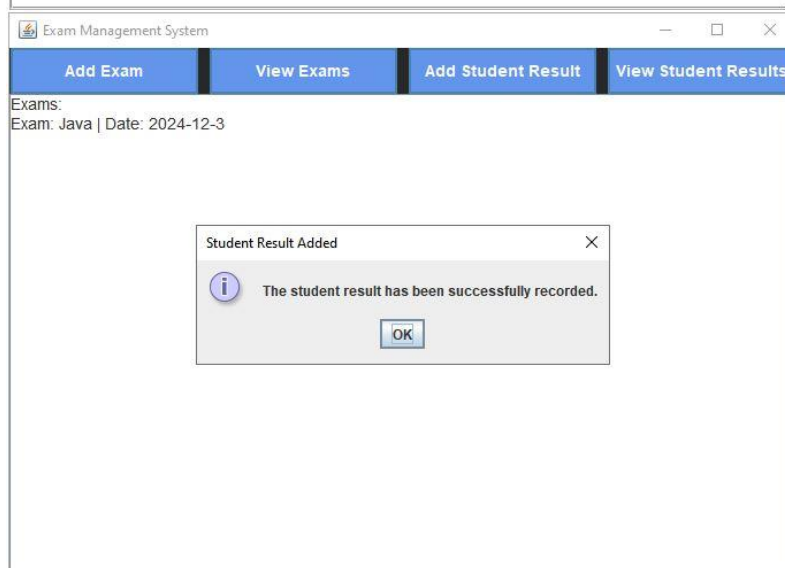
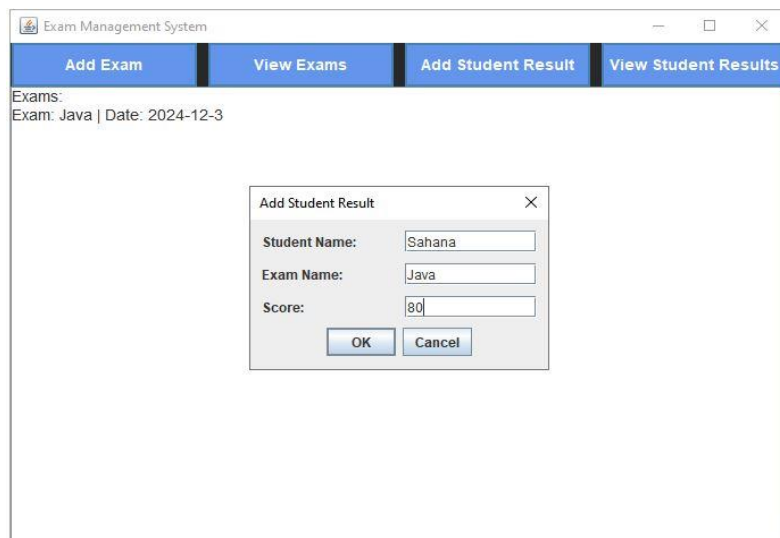


The main window of the "Exam Management System" is shown. It has a title bar with the system name and standard window controls. Below the title bar is a blue navigation bar with four tabs: "Add Exam", "View Exams", "Add Student Result", and "View Student Results". The "Add Exam" tab is selected. Below the navigation bar, the text "Exams:" is visible. In the center of the window, a message box titled "Exam Added Successfully" is displayed. The message box contains an information icon (i) and the text "The exam has been successfully added to the system." Below the message is an "OK" button.

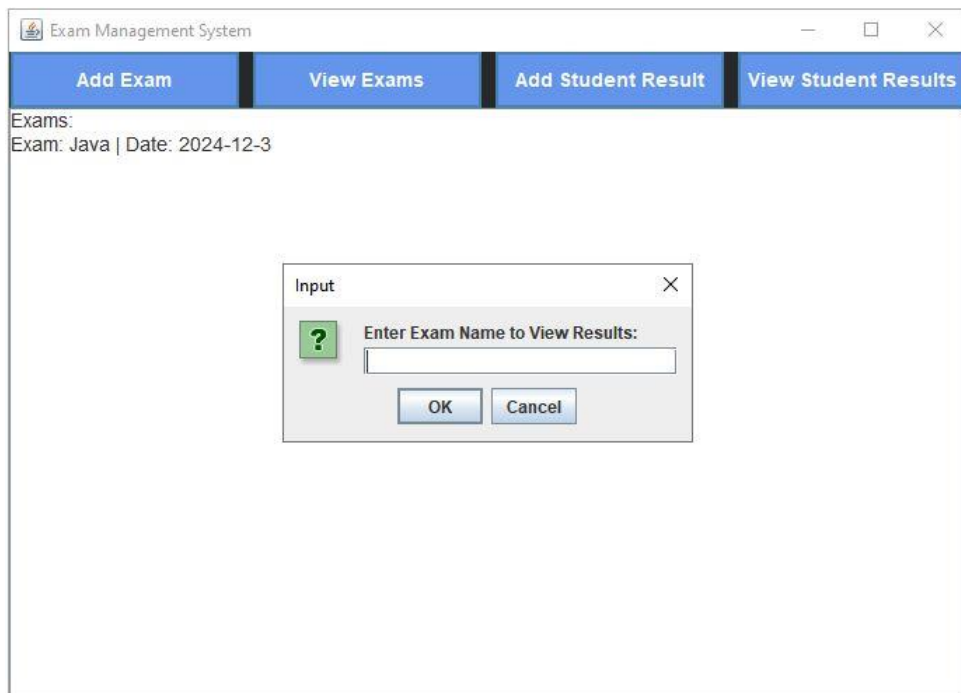
VIEW EXAMS:



ADD STUDENT RESULT:



VIEW STUDENT RESULT:



RESULT:

