

HELPDESK MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

VAITHYANADHAN S G 2303811724321119

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

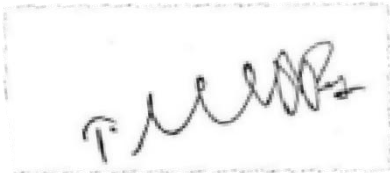
DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**HELPDESK MANAGEMENT SYSTEM**” is the bonafide work of **VAITHYANADHAN S G 2303811724321119** 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 Technology,

Samayapuram, Trichy -621 112.



Signature

Mrs. S. GEETHA M.E.,

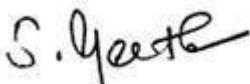
SUPERVISOR,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24 .



INTERNAL EXAMINER

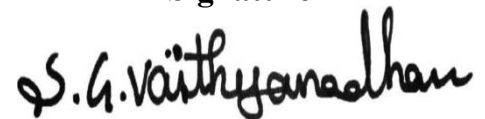


EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**HELPDESK 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**.

Signature


VAITHYANADHAN S G

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 **Helpdesk Management System** simplifies support ticket management by offering a seamless platform for interaction between **users** and **administrators**. Users can register, manage their profiles, submit support tickets, and track their statuses. Administrators can manage and prioritize all tickets, ensuring efficient resolution of issues. This project is built using Java programming, demonstrating key Object-Oriented Programming (OOP) concepts such as encapsulation and modularity. Data is dynamically managed using Java's Collections Framework, including **Hash Map** and **ArrayList**, for efficient storage and retrieval. The system features a menu-driven interface, ensuring an interactive yet simple user experience. With a focus on **modularity** and **scalability**, the design ensures future adaptability, such as the potential for integrating graphical interfaces or databases. This project enhances the efficiency of support ticket handling, offering an automated and user-friendly solution.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	viii
1	INTRODUCTION	2
	1.1 INTRODUCTION	2
	1.2 OBJECTIVE	2
2	PROJECT METHODOLOGY	4
	2.1 PROPOSED WORK	4
	2.2 BLOCK DIAGRAM	6
3	JAVA PROGRAMMING CONCEPTS	7
	3.1 OBJECT ORIENTED PROGRAMMING LANGUAGE	6
	3.2 GUI COMPONENTS	6
4	MODULE DESCRIPTION	8
	4.1 MAIN APPLICATION MODULE	8
	4.2 USER MODULE	9
	4.3 ADMIN MODULE	10
	4.4 UTILITY MODULE	10
5	CONCLUSION	12
	REFERENCES	14
	APPENDICES	16
	Appendix A – Source code	16
	Appendix B – Screen shots	22

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Helpdesk Management System is a Java-based project designed to streamline the process of handling customer support tickets, offering a seamless interface for both users and administrators. The application allows users to register, submit support tickets, and view the status of their requests, while administrators can manage and prioritize tickets efficiently. Built with Java Swing, the system provides an intuitive graphical interface that ensures a user-friendly experience. Administrators can view and resolve all support issues, ensuring timely responses to user queries. The platform improves the support process by enabling effective communication and management of support tickets, making it easier to track and resolve issues in a structured manner.

1.2 OBJECTIVE

The primary objective of the Helpdesk Management System is to create an efficient, user-friendly platform that enhances the management of customer support tickets. The system aims to:

1. **Automate ticket management:** Reducing manual tracking of issues while streamlining the support process for users and administrators.
2. **Enhance user convenience:** Offering an intuitive interface for tasks such as ticket submission, issue tracking, and administration.
3. **Demonstrate the power of Java programming:** Utilizing advanced features such as Object-Oriented Programming (OOP), data encapsulation, and Java collections for effective ticket handling and data management.
4. **Address traditional inefficiencies:** Replacing outdated support methods with a modern, digital solution for efficient issue resolution.
5. **Improve response times:** Enabling administrators to prioritize and

resolve tickets based on severity, ensuring timely responses to user issues.

6. **Ensure data security:** Protecting sensitive user and ticket data with secure handling practices.

7. **Improve accessibility:** Designing the system to be simple and accessible for users with varying levels of technical proficiency..

8. **Enable smooth interaction:** Facilitating efficient communication between users and administrators for quick issue resolution

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The project aims to develop a user-friendly and efficient Helpdesk Management System. The core objective is to create a Java-based application that allows seamless interaction between users and administrators for managing customer support tickets. The methodology focuses on:

1. Understanding Requirements:

- Identifying needs for user registration, secure login, ticket submission, and management.
- Analyzing existing systems to understand user pain points and improve ticket handling.
- modular Java code using Object-Oriented Programming principles.

2. Employing data structures like HashMap for datastorage:

- Architecting the system for functionality, scalability, and user-friendliness.

3. Implementation:

- Developing a menu-driven graphical interface for user interaction.
- Implementation for user and admin modules for ticket handling and management.

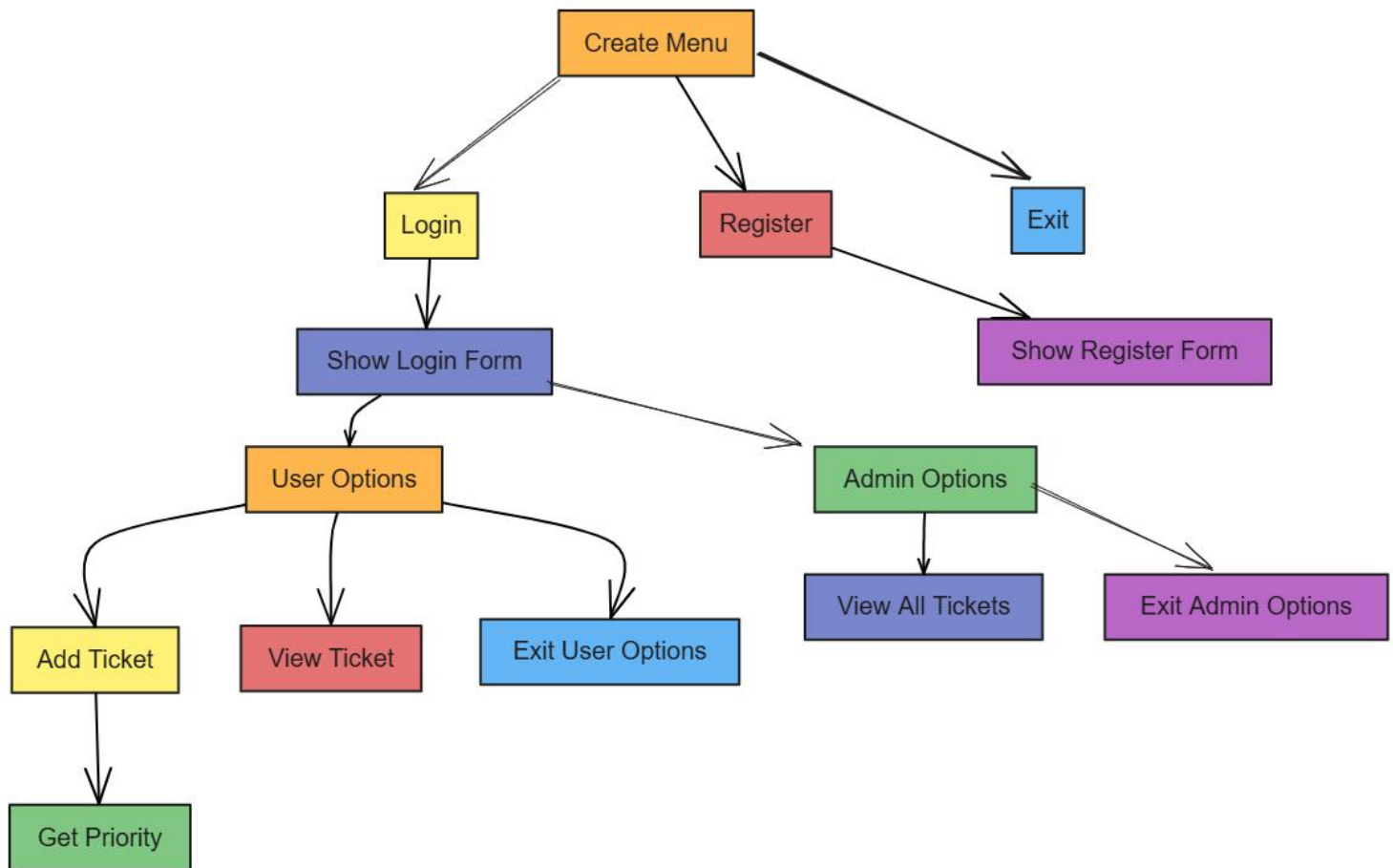
4. Testing and Validation:

- Testing each module independently for functionality and integration

5. Deployment and Feedback:

- Deploying a fully functional application for end users.

2.2 BLOCK DIAGRAM



CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1 OBJECT-ORIENTED PROGRAMMING (OOP):

- Classes and Objects: Represent users, admins, and tickets with separate classes to manage system functionalities.
- Encapsulation: Protect user data using private fields and public getter/setter methods for secure access.
- Inheritance: Future-proof the system by allowing easy extension of classes for additional features.

2. Java Collections Framework:

- HashMap: Stores user credentials and ticket details efficiently for quick lookups.
- ArrayList: Manages dynamic data such as tickets.

3. Control Flow Mechanisms:

- Loops and Conditional Statements: Help navigate through menus and perform actions based on user choices.
- Switch Case: Organizes user inputs and directs actions in the menu system.

4. Input Handling with Scanner:

Scanner: Captures user input for actions like registration, login, and ticket submission.

3.2 GUI COMPONENTS

1. Frame: The main window displaying menus and user interactions.
2. Label: Displays text like "Username" and "Password" for user guidance
3. Button: Initiates actions like login, registration, and viewing tickets.
4. TextField: Accepts input for user credentials and ticket information.

CHAPTER 4

MODULE DESCRIPTION

4.1 MAIN APPLICATION MODULE

4.1.1 Objective

The Main Application Module serves as the entry point for the HelpDesk Management System. It provides a simple and intuitive interface for users to navigate between the User and Admin roles. This module ensures smooth navigation and helps the user select the appropriate functionality.

4.1.2 Design

- Graphical Components Used:
 - **Frame:** The main container that holds the entire graphical user interface (GUI).
 - **Button:** Interactive buttons such as "Login," "Register," and "Exit" for user actions.
 - **Label:** Displays informative or instruction text to the user.
- Navigation Workflow:
 1. User opens the application.
 2. Main menu appears with options for login, registration, and exit.
 3. Based on the user's choice, either the user or admin module is activated.

4.1.3 Usage

- This module serves as the first interaction point in the system, guiding users to appropriate actions like logging in or registering. It ensures a clear and user-friendly start to the program.

4.2 USER MODULE

4.2.1 Objective

The User Module allows users to register, log in, manage their profiles, and interact with tickets. This module ensures users can create new tickets and track their progress easily.

4.2.2 Submodules

1. Registration Submodule:

- **Objective:** Collects personal details for user registration.
- **Features:**
 - Captures user name, email, and password.
 - Validates input (ensures valid email format and strong password).
- **Workflow:**
 1. User enters required details into the form.
 2. The system validates the data and stores it securely.

2. Ticket Management Submodule:

- **Objective:** Allows users to create and view tickets.
- **Feature:**
 - Users can enter issues, submit tickets, and view their status.
- **Workflow:**
 1. User submits an issue with description.
 2. The ticket is assigned a priority and tracked in the system.

4.3 ADMIN MODULE

4.3.1 Objective

The Admin Module allows administrators to log in securely, view and manage all user tickets, and track ticket statuses. Admins can handle all support issues within the system.

4.3.2 Submodules

1. Admin login submodule:

- Objective: Authenticates the admin using username and password.
- Workflow:
 - Admin enters login credentials.
 - If validated, the admin gains access to ticket management features.

2. Ticket Management Submodule:

- Objective: Allows admins to view, update, and manage all tickets.
- Features: Admins can review all tickets with their status and priority.
- Workflow:
 1. Admin accesses the list of tickets.
 2. Admin can close, update, or escalate tickets as necessary.

4.4 UTILITY MODULE

4.4.1 Objective

The System Module handles backend operations such as data storage, error handling, and user interaction messages. It ensures proper functioning and storage of user and ticket data.

4.4.2 Features

1. Data Storage: Uses **HashMap** to store user credentials (name, password) and ticket data (issue, priority).
2. Ticket Prioritization: Automatically assigns priority levels to tickets based on predefined issue types.
3. Message Handling: Uses dialog boxes to display error or success messages, keeping the user informed of their actions.

CHAPTER 5

CONCLUSION

SUMMARY OF ACHIEVEMENTS

The HelpDesk Management System effectively meets the needs of both users and admins, providing a streamlined and interactive solution for ticket management and support.

1. For User:

- Simple registration and login processes.
- Easy ticket creation, submission, and tracking.

2. For Admins:

- Secure login and authentication system.
- Comprehensive ticket management features for reviewing and updating ticket statuses.

3. Overall System:

- A well-organized, modular design ensuring easy navigation.
- Efficient data handling through Java's collections framework (HashMap and ArrayList).

LIMITATIONS

1. No persistent storage (the current implementation relies on in-memory data storage).
2. Basic GUI design limited by AWT's capabilities.

FUTURE SCOPE

1. **Database Integration:** Use SQL or NoSQL databases for better scalability.
2. **Enhanced Matching Algorithm:** Implement advanced features like auto-ticket assignment and AI-driven ticket prioritization.
3. **Reporting and Analytics:** Add detailed reporting features for admins to analyze ticket trends, resolution times, and user feedback.

4. **Mobile Compatibility:** Extend the application for Android and iOS platforms using JavaFX or Kotlin.
5. **Analytics Dashboard:** Add analytics for employers to track job postings and applicant statistics.

REFERENCES

The development of this project involved the use of various resources to understand Java programming principles and implement best practices. The references include:

Books

1. *Core Java Volume I: Fundamentals* by Cay S. Horstmann and Gary Cornell
 - Comprehensive guide to Java fundamentals, including OOP concepts and collections.
2. *Effective Java* by Joshua Bloch
 - Focus on practical programming techniques and Java best practices.

Online Tutorials

1. **W3Schools Java Tutorials:**
<https://www.w3schools.com/java/>
 - For understanding basic to advanced Java programming concepts.
2. **GeeksforGeeks Java Programming:**
<https://www.geeksforgeeks.org/java/>
 - Detailed explanations of Java collections, object-oriented principles, and practical examples.

Documentation

1. **Oracle Java Documentation:**
<https://docs.oracle.com/javase/tutorial/>
 - Official documentation for Java programming and the Collections Framework.
2. **Java SE 17 API Documentation:**
<https://docs.oracle.com/en/java/javase/17/docs/api/>
 - Detailed reference for Java APIs used in the project.

Videos

1. **Programming with Mosh – Java Full Course (YouTube):**
<https://www.youtube.com/watch?v=grEKMHGYYns>
 - Detailed video tutorials covering Java basics to advanced topics.

2. **CodeWithHarry – Java for Beginners** (YouTube):

<https://www.youtube.com/CodeWithHarry>

- Simplified explanation of Java programming concepts.

APPENDICES

APPENDIX A – SOURCE CODE

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

class HelpdeskSystem {
    // Shared Data Storage
    private static HashMap<String, String> users = new HashMap<>();
    private static HashMap<String, Integer> tickets = new HashMap<>();

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

    // **System Module**
    private static void createMenu() {
        JFrame frame = new JFrame("Helpdesk Management System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        JMenuBar menuBar = new JMenuBar();
        JMenu menu = new JMenu("Menu");
        JMenuItem loginItem = new JMenuItem("Login");
        JMenuItem registerItem = new JMenuItem("Register");
        JMenuItem exitItem = new JMenuItem("Exit");

        menu.add(loginItem);
        menu.add(registerItem);
        menu.addSeparator();
        menu.add(exitItem);
        menuBar.add(menu);
        loginItem.addActionListener(e -> showLoginForm(frame));
        registerItem.addActionListener(e -> showRegisterForm(frame));
        exitItem.addActionListener(e -> System.exit(0));

        frame.setJMenuBar(menuBar);
        frame.setVisible(true);
    }
}
```



```

}

private static void showLoginForm(JFrame frame) {
    JFrame loginFrame = new JFrame("Login");
    loginFrame.setSize(300, 200);
    loginFrame.setLayout(new GridLayout(4, 2));

    JLabel nameLabel = new JLabel("Name: ");
    JTextField nameField = new JTextField();
    JLabel passwordLabel = new JLabel("Password: ");
    JPasswordField passwordField = new JPasswordField();
    JLabel roleLabel = new JLabel("Role (user/admin): ");
    JTextField roleField = new JTextField();
    JButton loginButton = new JButton("Login");

    loginFrame.add(nameLabel);
    loginFrame.add(nameField);
    loginFrame.add(passwordLabel);
    loginFrame.add(passwordField);
    loginFrame.add(roleLabel);
    loginFrame.add(roleField);
    loginFrame.add(loginButton);

    loginButton.addActionListener(e -> {
        String name = nameField.getText();
        String password = new String(passwordField.getPassword());
        String role = roleField.getText();

        if (users.containsKey(name) && users.get(name).equals(password)) {
            if ("admin".equalsIgnoreCase(role)) {
                AdminModule.showAdminOptions(frame);
            } else {
                UserModule.showUserOptions(frame);
            }
            loginFrame.dispose();
        } else {
            JOptionPane.showMessageDialog(frame, "Invalid login.");
        }
    });
    loginFrame.setVisible(true);
}

private static void showRegisterForm(JFrame frame) {
    JFrame registerFrame = new JFrame("Register");

```

```

registerFrame.setSize(300, 200);
registerFrame.setLayout(new GridLayout(3, 2));

JLabel nameLabel = new JLabel("Name: ");
JTextField nameField = new JTextField();
JLabel passwordLabel = new JLabel("Password: ");
JPasswordField passwordField = new JPasswordField();
JButton registerButton = new JButton("Register");

registerFrame.add(nameLabel);
registerFrame.add(nameField);
registerFrame.add(passwordLabel);
registerFrame.add(passwordField);
registerFrame.add(registerButton);

registerButton.addActionListener(e -> {
    String name = nameField.getText();
    String password = new String(passwordField.getPassword());

    if (!users.containsKey(name)) {
        users.put(name, password);
        JOptionPane.showMessageDialog(frame, "Registration successful.");
        registerFrame.dispose();
    } else {
        JOptionPane.showMessageDialog(frame, "User already exists.");
    }
});

registerFrame.setVisible(true);
}

// **User Module**
static class UserModule {
    static void showUserOptions(JFrame frame) {
        JFrame userFrame = new JFrame("User Options");
        userFrame.setSize(300, 200);
        userFrame.setLayout(new FlowLayout());

        JButton addTicketButton = new JButton("Add Ticket");
        JButton viewTicketButton = new JButton("View Tickets");
        JButton exitButton = new JButton("Exit");

        userFrame.add(addTicketButton);

```

```

userFrame.add(viewTicketButton);
userFrame.add(exitButton);

addTicketButton.addActionListener(e -> addTicket(frame));
viewTicketButton.addActionListener(e -> viewTickets(frame));
exitButton.addActionListener(e -> userFrame.dispose());

userFrame.setVisible(true);
}

private static void addTicket(JFrame frame) {
    JFrame ticketFrame = new JFrame("Add Ticket");
    ticketFrame.setSize(300, 200);
    ticketFrame.setLayout(new GridLayout(2, 2));

    JLabel ticketLabel = new JLabel("Ticket Issue: ");
    JTextField ticketField = new JTextField();
    JButton submitButton = new JButton("Submit");

    ticketFrame.add(ticketLabel);
    ticketFrame.add(ticketField);
    ticketFrame.add(submitButton);

    submitButton.addActionListener(e -> {
        String issue = ticketField.getText();
        int priority = SystemModule.getPriority(issue);
        tickets.put(issue, priority);
        JOptionPane.showMessageDialog(frame, "Ticket added with priority: " + priority);
        ticketFrame.dispose();
    });

    ticketFrame.setVisible(true);
}

private static void viewTickets(JFrame frame) {
    JFrame ticketViewFrame = new JFrame("View Tickets");
    ticketViewFrame.setSize(300, 200);

    StringBuilder ticketDetails = new StringBuilder();
    tickets.forEach((issue, priority) -> ticketDetails.append("Issue: ")
        .append(issue).append(", Priority: ").append(priority).append("\n"));

    JTextArea ticketArea = new JTextArea(ticketDetails.toString());
    ticketArea.setEditable(false);
}

```

```

        ticketViewFrame.add(new JScrollPane(ticketArea));
        ticketViewFrame.setVisible(true);
    }
}

// **Admin Module**
static class AdminModule {
    static void showAdminOptions(JFrame frame) {
        JFrame adminFrame = new JFrame("Admin Options");
        adminFrame.setSize(300, 200);
        adminFrame.setLayout(new FlowLayout());

        JButton viewTicketsButton = new JButton("View All Tickets");
        JButton exitButton = new JButton("Exit");

        adminFrame.add(viewTicketsButton);
        adminFrame.add(exitButton);

        viewTicketsButton.addActionListener(e -> viewAllTickets(frame));
        exitButton.addActionListener(e -> adminFrame.dispose());

        adminFrame.setVisible(true);
    }

    private static void viewAllTickets(JFrame frame) {
        JFrame ticketViewFrame = new JFrame("View All Tickets");
        ticketViewFrame.setSize(300, 200);

        StringBuilder ticketDetails = new StringBuilder();
        tickets.forEach((issue, priority) -> ticketDetails.append("Issue:
").append(issue).append(", Priority: ").append(priority).append("\n"));

        JTextArea ticketArea = new JTextArea(ticketDetails.toString());
        ticketArea.setEditable(false);

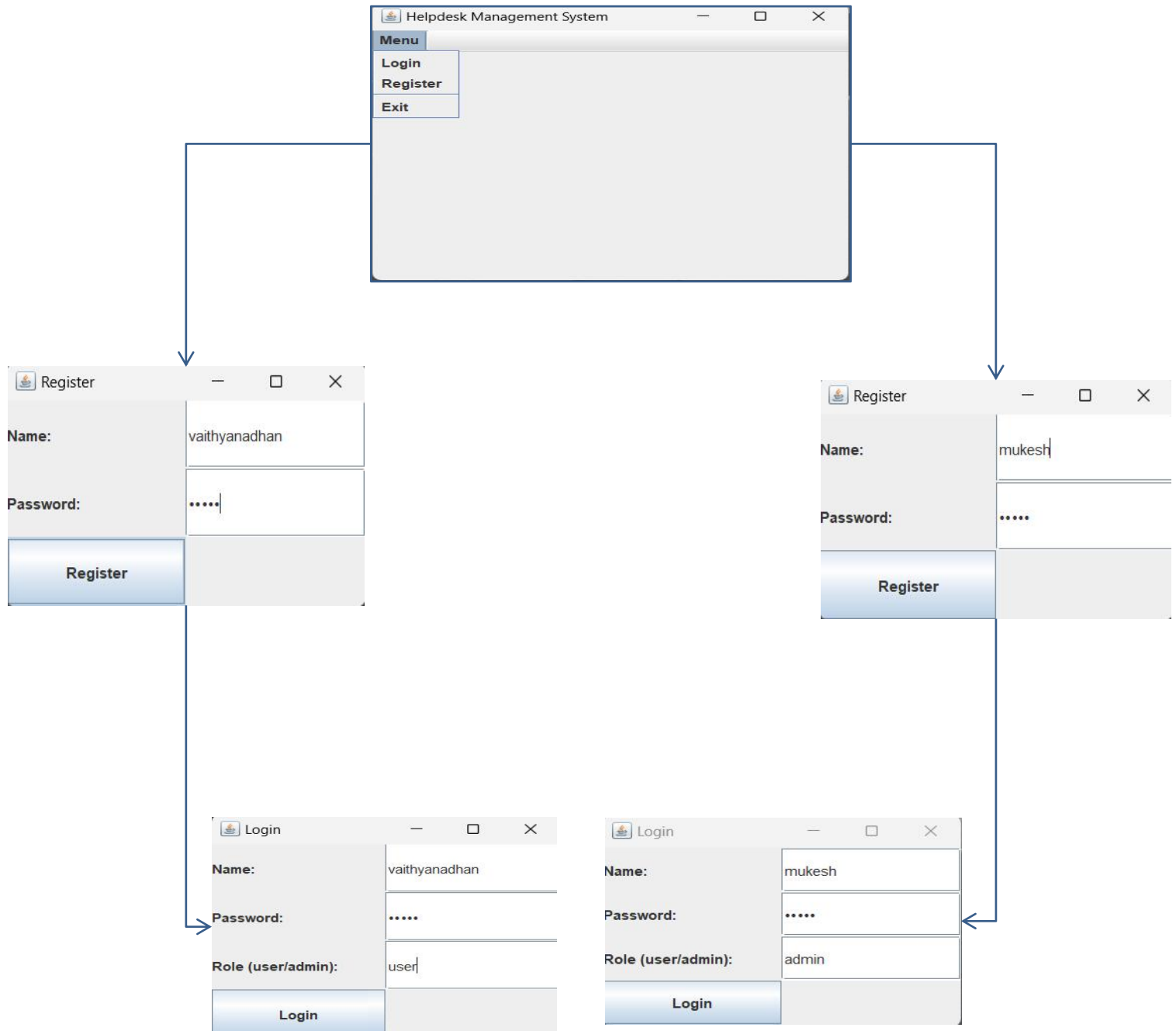
        ticketViewFrame.add(new JScrollPane(ticketArea));
        ticketViewFrame.setVisible(true);
    }
}

// **System Utility Methods**
static class SystemModule {
    static int getPriority(String issue) {

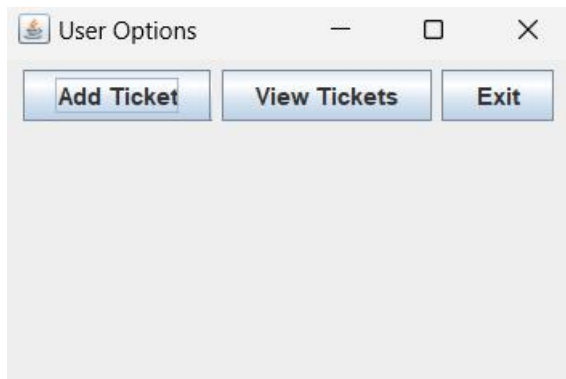
```

```
switch (issue.toLowerCase()) {  
  case "login issues":  
    return 1;  
  case "system issues":  
    return 2;  
  case "mail correction":  
    return 3;  
  default:  
    return 4; // Default priority for other issues  
}  
}  
}
```

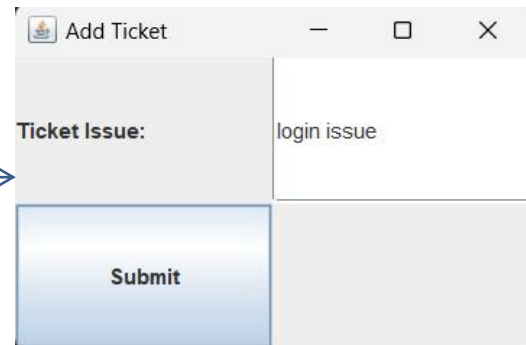
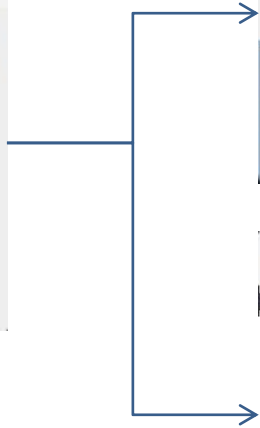
APPENDIX B - SCREENSHOTS



User Login:



A window titled "User Options" with a standard Windows-style title bar (minimize, maximize, close buttons). It contains three buttons: "Add Ticket", "View Tickets", and "Exit".

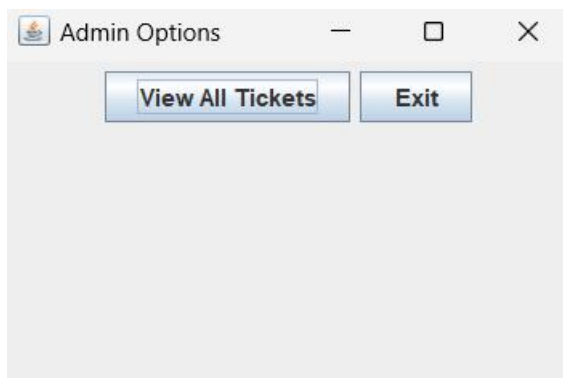


A window titled "Add Ticket" with a standard Windows-style title bar. It contains a label "Ticket Issue:" followed by a text input field containing "login issue". Below the input field is a large blue button labeled "Submit".

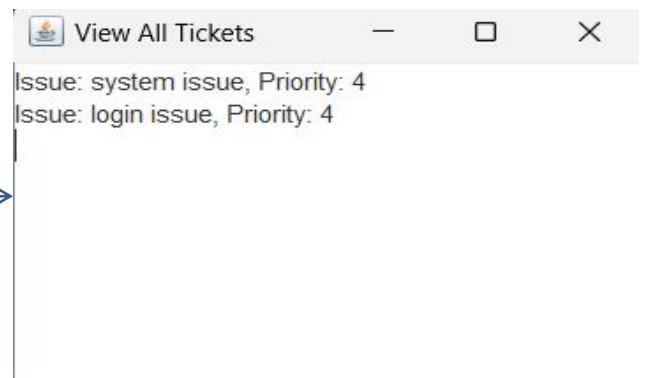


A window titled "View Tickets" with a standard Windows-style title bar. It displays the text "Issue: login issue, Priority: 4".

Admin Login:



A window titled "Admin Options" with a standard Windows-style title bar. It contains two buttons: "View All Tickets" and "Exit".



A window titled "View All Tickets" with a standard Windows-style title bar. It displays a list of tickets: "Issue: system issue, Priority: 4" and "Issue: login issue, Priority: 4".