



# ONLINE SURVEY SYSTEM



## A PROJECT REPORT

*Submitted by*

**SWAATHI S (2303811710422164)**

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

**CGB1201 - JAVA PROGRAMMING**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

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

**SAMAYAPURAM – 621 112**

**NOVEMBER- 2024**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY  
(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “**ONLINE SURVEY SYSTEM**” is the bonafide work of **SWAATHI S(2303811710422154)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING  
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,  
HEAD OF THE DEPARTMENT  
PROFESSOR

CGB1201-JAVA PROGRAMMING  
Mr.MALARMANNAN A, M.E.,  
ASSISTANT PROFESSOR

**SIGNATURE**

**SIGNATURE**

Mrs.A.Delphin Carolina Rani, M.E.,Ph.D., Mr. A. Malarmannan, M.E.,

**HEAD OF THE DEPARTMENT**

**SUPERVISOR**

PROFESSOR

ASSISTANT PROFESSOR

Department of CSE

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram–621112.

Samayapuram–621112.

Submitted for the viva-voce examination held on 06/12/2024.

CGB1201-JAVA PROGRAMMING  
Mr. M. SIVANANDAN, M.E.,  
INTERNAL EXAMINER  
ASSISTANT PROFESSOR

**INTERNAL EXAMINER**

CGB1201-JAVA PROGRAMMING  
Mr.R. KANAKHIK, M.E.,  
EXTERNAL EXAMINER  
ASSISTANT PROFESSOR  
8138-SCE, TRICHY.

**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**ONLINE SURVEY SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

**Signature**



---

SWAATHI S

Place: Samayapuram

Date: 06/12/2024

## ACKNOWLEDGEMENT

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

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for 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 to our project and offering adequate duration in completing our project.

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

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

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

## **VISION OF DEPARTMENT**

To be a center of eminence in creating competent software professionals with research and innovative skills.

## **MISSION OF DEPARTMENT**

**M1: Industry Specific:** To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

**M2: Research:** To prepare students for research-oriented activities.

**M3: Society:** To empower students with the required skills to solve complex technological problems of society.

## **PROGRAM EDUCATIONAL OBJECTIVES**

### **1. PEO1: Domain Knowledge**

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

### **2. PEO2: Employability Skills and Research**

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

### **3. PEO3: Ethics and Values**

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

#### **PSO 1: Domain Knowledge**

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

#### **PSO 2: Quality Software**

To apply software engineering principles and practices for developing quality software for scientific and business applications.

#### **PSO 3: Innovation Ideas**

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

### **PROGRAM OUTCOMES (POs)**

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.

## ABSTRACT

The **Role-Based User Authentication System** is an interactive and user-friendly platform designed to manage user accounts and enforce role-based access control, implemented in Java using Swing for the graphical interface. The system facilitates the registration and login of users, with distinct roles such as ADMIN and USER, providing role-specific functionalities to enhance user experience and operational efficiency.

The core features include secure user registration with username, password, and role assignment, alongside a robust login mechanism. Upon successful login, users are directed to perform actions based on their assigned roles. ADMIN users can create and view survey results, while USER roles can respond to surveys and view existing results. These functionalities ensure a tailored experience for each role, promoting operational clarity and user satisfaction.

The graphical user interface, built using Swing, enables seamless interaction through intuitive screens for registration, login, and role-specific operations. A clean and modular design ensures scalability and maintainability, allowing for future enhancements. The system employs a dynamic structure, storing user data and surveys in collections, thus enabling flexibility in handling varying loads of user and survey data.

By integrating role-based access control with interactive GUI elements, the system not only demonstrates the practical application of Java programming concepts like classes, collections, and Swing, but also serves as a valuable educational resource. The **Role-Based User Authentication System** provides a scalable, efficient, and user-centric solution for managing user roles and operations in a secure and intuitive environment.



## ABSTRACT WITH POs AND PSOs MAPPING

### CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Role-Based User Authentication System is a Java application with a Swing-based graphical interface that manages user accounts and enforces role-based access control. It allows secure registration and login, assigning users roles such as ADMIN or USER, with tailored functionalities. ADMIN users can create and view survey results, while USER roles can respond to surveys and view results. Featuring an intuitive GUI and dynamic data handling through collections, the system ensures scalability, flexibility, and a user-friendly experience. It demonstrates core Java concepts, providing a secure, efficient, and educational solution for role-based operations.	<b>PO1 -3</b> <b>PO2 -3</b> <b>PO3 -3</b> <b>PO4 -3</b> <b>PO5 -3</b> <b>PO6 -3</b> <b>PO7 -3</b> <b>PO8 -3</b> <b>PO9 -3</b> <b>PO10 -3</b> <b>PO11-3</b> <b>PO12 -3</b>	<b>PSO1 -3</b> <b>PSO2 -3</b> <b>PSO3 -3</b>

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

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	x
1	<b>INTRODUCTION</b>	
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	1
2	<b>PROJECT METHODOLOGY</b>	
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	<b>MODULE DESCRIPTION</b>	
	3.1 User Module	4
	3.2 UserManager Module	4
	3.3 GUI Module	4
	3.4 Role-Based Access Module	4
	3.5 Data Handling Module	5
4	<b>CONCLUSION &amp; FUTURE SCOPE</b>	
	4.1 Conclusion	6
	4.2 Future Scope	6
	<b>APPENDIX A (SOURCE CODE)</b>	7
	<b>APPENDIX B (SCREENSHOTS)</b>	15
	<b>REFERENCES</b>	19

# CHAPTER 1

## INTRODUCTION

### 1.1 Objective

The primary objective of the **Role-Based User Authentication System** is to develop a scalable and interactive application that manages user accounts and enforces role-based access control. The system aims to provide:

1. A secure and user-friendly platform for managing user registration and login.
2. Role-specific functionalities that allow ADMIN users to manage surveys and USER roles to respond to surveys.
3. An intuitive graphical user interface (GUI) for seamless interaction with the system.
4. A practical application to demonstrate the principles of object-oriented programming, graphical programming using Swing, and dynamic data handling.

### 1.2 Overview

The Role-Based User Authentication System is an application designed to manage users based on roles and provide functionalities accordingly. The system includes the following features:

1. User Registration: Allows new users to register by providing a username, password, and role (ADMIN or USER).
2. User Login: Authenticates users based on their credentials and grants access to role-specific functionalities.
3. Role-Specific Actions:
  - ADMIN: Can create surveys and view survey results.
  - USER: Can respond to surveys and view results of existing surveys.
4. Graphical User Interface (GUI): Built using Swing, it offers a series of intuitive forms and menus for user interaction.
5. Dynamic Data Management: Uses Java collections to manage users and surveys dynamically without predefined limits.

### 1.3 Java Programming Concepts

The Role-Based User Authentication System utilizes several Java programming concepts, including:

1. Object-Oriented Programming (OOP):
  - Classes and Objects:

- The User class represents individual users with attributes like username, password, and role.
  - The UserManager class manages user registration, login, and survey data.
- Encapsulation: Attributes of the User class are private and accessed through getter methods.
  - Abstraction: Role-specific actions are abstracted within modular methods.
2. Collections Framework:
    - The system uses collections like ArrayList to manage user and survey data dynamically.
    - The HashMap is used for mapping surveys to their responses, providing efficient retrieval and management of data.
  3. Graphical Programming with Swing:
    - Swing components like JFrame, JPanel, JButton, JLabel, JTextField, and JPasswordField are used to design the GUI.
    - Event handling is implemented to respond to user actions such as button clicks.
  4. Input Validation:
    - User inputs are validated to ensure correct data entry, such as valid roles (ADMIN or USER) and non-empty fields.
  5. Control Structures:
    - Conditional statements (if-else) determine role-based access and enforce correct user behavior.
  6. Dynamic Data Handling:
    - The system dynamically handles data, allowing for addition and deletion of users and surveys without a predefined size limit.
  7. Modularity and Scalability:
    - The application is modular, with separate methods and classes for distinct functionalities, making it easy to maintain and extend.

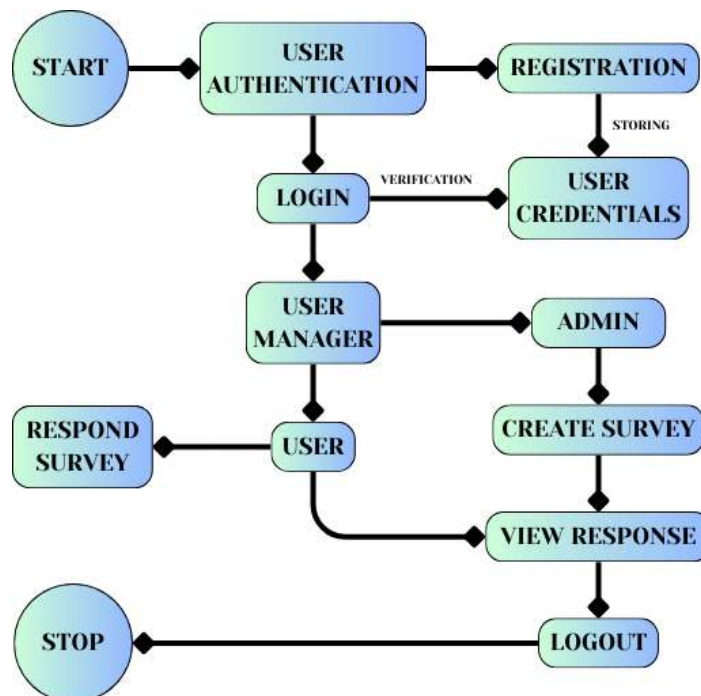
## CHAPTER 2

### PROJECT METHODOLOGY

#### 2.1 Proposed Work

The proposed work for the **Role-Based User Authentication System** focuses on developing a secure and scalable platform for managing user accounts and role-specific functionalities. The system enables users to register with unique credentials, log in securely, and perform actions based on their assigned roles, such as creating surveys, responding to surveys, or viewing results. It incorporates role-based access control (RBAC) to ensure that ADMIN users have privileges to manage surveys, while USER roles can interact with the surveys accordingly. A dynamic data management approach using Java collections allows flexible handling of users and surveys without predefined limits, ensuring scalability. The system features an intuitive graphical user interface (GUI) built with Java Swing, providing a seamless and interactive experience for registration, login, and role-specific operations. With a modular design, it supports easy maintenance and future extensions, such as database integration, advanced survey management, and enhanced security features. This work aims to deliver a robust, user-friendly application that effectively demonstrates the principles of secure authentication, RBAC, and dynamic data handling in Java.

#### 2.2 Block Diagram



## **CHAPTER 3**

### **MODULE DESCRIPTION**

#### **3.1 User Module**

The User module is the foundation of the system, representing individual users with attributes like username, password, and role (ADMIN/USER). This module encapsulates user-related data and provides methods to access these details securely through getters. A constructor initializes the user attributes upon creation. By separating user information into its own module, the system ensures secure storage and structured management of user data, promoting reusability and modularity.

#### **3.2 UserManager Module**

The UserManager module handles all user-related operations, including registration, authentication, and survey management. It allows new users to register by adding them to the system, verifies login credentials to grant secure access, and provides functionalities to manage surveys dynamically. ADMIN users can create and view survey results, while USER roles can respond to surveys. This module serves as the core of the system, coordinating user actions and ensuring the proper execution of role-specific operations.

#### **3.3 GUI Module**

The GUI module provides an interactive interface for users to engage with the system. Built using Java Swing, it includes separate panels for login, registration, and role-based dashboards. The login panel facilitates secure access, while the registration panel allows new users to join the system. Role-specific dashboards offer ADMIN users options to create surveys and view results, while USER roles can respond to surveys and view results. This module ensures a seamless and intuitive user experience, improving the system's usability and accessibility.

#### **3.4 Role-Based Access Module**

The Role-Based Access module enforces the system's access control policies. It uses role checks during the login process to redirect users to role-specific dashboards. ADMIN users can perform privileged actions like survey creation, while USER roles are limited to responding to surveys and viewing results. This module ensures that actions are restricted to authorized users, enhancing security and maintaining system integrity.

### **3.5 Data Handling Module**

The Data Handling module is responsible for managing user and survey data efficiently. It uses dynamic data structures like ArrayList to store user information and HashMap to organize surveys and responses. These structures allow for flexible growth without predefined size limits, ensuring scalability as the system grows. This module guarantees efficient storage, retrieval, and modification of data, forming the backbone of the system's dynamic functionality.

## CHAPTER 4

### 4.1 CONCLUSION

The development of the **Role-Based User Authentication System** with a graphical user interface (GUI) has proven to be a successful implementation, demonstrating the effective use of object-oriented programming principles and dynamic data management in Java. By utilizing Java Swing for the GUI, the system provides a user-friendly interface that ensures seamless interaction for both ADMIN and USER roles. The integration of role-based access control (RBAC) enables secure and efficient management of user privileges, allowing distinct functionalities such as survey creation, response submission, and result viewing.

The system's modular architecture, combined with dynamic data structures like Array List and HashMap, ensures scalability and efficient handling of users and surveys. Robust error handling mechanisms enhance system stability, providing clear feedback to users and preventing unauthorized actions. This project not only demonstrates technical expertise in Java programming and GUI development but also highlights the importance of secure authentication, scalability, and user-centric design. Overall, the system offers a reliable, adaptable, and intuitive solution for managing role-based user actions, laying a solid foundation for future enhancements.

### 4.2 FUTURE SCOPE

The Role-Based User Authentication System has significant potential for future enhancements to meet evolving user and technological demands. Future iterations could include integrating advanced security measures, such as password hashing and multi-factor authentication, to ensure robust user protection. Expanding survey functionality to support advanced question types, multimedia integration, and analytics would increase its versatility and appeal. Integration with external databases and cloud services could enhance scalability and enable seamless multi-device synchronization. Adding role customization would allow organizations to define roles beyond ADMIN and USER, catering to diverse application needs. Incorporating real-time notifications and reminders, as well as multi-language support for localization, would improve user engagement and accessibility. The system could also benefit from the implementation of machine learning algorithms to analyze survey data, providing insightful recommendations and trends. Finally, a mobile application version of the system would extend its reach, offering users convenience and accessibility across platforms. These enhancements would position the system as a comprehensive, secure, and adaptive solution for managing role-based functionalities.



## APPENDIX A (SOURCE CODE)

### User Class

```
package project;

public class User {
    private String username;
    private String password;
    private String role;
    public User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }
    public String getUsername() {
        return username;
    }
    public String getPassword() {
        return password;
    }
    public String getRole() {
        return role;
    }
}
```

### UserManager Class

```
package project;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class UserManager {
    private List<User> users = new ArrayList<>();
    private Map<String, List<String>> surveys = new HashMap<>();

    public void registerUser(String username, String password, String role) {
```

```

        users.add(new User(username, password, role));
    }
    public User loginUser(String username, String password) {
        for (User user : users) {
            if (user.getUsername().equals(username) && user.getPassword().equals(password))
            {
                return user;
            }
        }
        return null;
    }
    public void createSurvey(String title) {
        surveys.put(title, new ArrayList<>());
    }
    public void respondToSurvey(String title, String response) {
        if (surveys.containsKey(title)) {
            surveys.get(title).add(response);
        }
    }
    public String viewSurveyResults(String title) {
        if (surveys.containsKey(title)) {
            return String.join("\n", surveys.get(title));
        }
        return "No results found for this survey.";
    }
    public List<String> getSurveyTitles() {
        return new ArrayList<>(surveys.keySet());
    }
}

```

### **RoleBasedAuthSystemSwing Class**

```

package project;
import javax.swing.*;
import java.awt.*;
import java.util.List;

```

```

public class RoleBasedAuthSystemSwing {
    private UserManager userManager = new UserManager();
    public static void main(String[] args) {
        SwingUtilities.invokeLater(RoleBasedAuthSystemSwing::new);
    }
    public RoleBasedAuthSystemSwing() {
        showMainMenu();
    }
    private void showMainMenu() {
        JFrame frame = new JFrame("Main Menu");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 1, 10, 10));
        JButton registerButton = new JButton("Register");
        JButton loginButton = new JButton("Login");
        JButton exitButton = new JButton("Exit");
        registerButton.addActionListener(e -> {
            frame.dispose();
            showRegisterScreen();
        });
        loginButton.addActionListener(e -> {
            frame.dispose();
            showLoginScreen();
        });
        exitButton.addActionListener(e -> System.exit(0));
        panel.add(registerButton);
        panel.add(loginButton);
        panel.add(exitButton);
        frame.add(panel);
        frame.setVisible(true);
    }
    private void showRegisterScreen() {
        JFrame frame = new JFrame("Register");
    }

```

```

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(400, 300);
JPanel panel = new JPanel(new GridLayout(4, 2, 10, 10));
JLabel usernameLabel = new JLabel("Username:");
JTextField usernameField = new JTextField();
JLabel passwordLabel = new JLabel("Password:");
JPasswordField passwordField = new JPasswordField();
JLabel roleLabel = new JLabel("Role (ADMIN/USER):");
JTextField roleField = new JTextField();
JButton registerButton = new JButton("Register");
JButton backButton = new JButton("Back");
registerButton.addActionListener(e -> {
    String username = usernameField.getText();
    String password = new String(passwordField.getPassword());
    String role = roleField.getText().toUpperCase();
    if (username.isEmpty() || password.isEmpty() || (!role.equals("ADMIN")      &&
    !role.equals("USER")))) {
        JOptionPane.showMessageDialog(frame, "Invalid input! Please try    again.");
    } else {
        userManager.registerUser(username, password, role);
        JOptionPane.showMessageDialog(frame, "User registered
            successfully!");
        frame.dispose();
        showMainMenu();
    }
});
backButton.addActionListener(e -> {
    frame.dispose();
    showMainMenu();
});
panel.add(usernameLabel);
panel.add(usernameField);
panel.add(passwordLabel);
panel.add(passwordField);

```

```

        panel.add(roleLabel);
        panel.add(roleField);
        panel.add(registerButton);
        panel.add(backButton);
        frame.add(panel);
        frame.setVisible(true);
    }

    private void showLoginScreen() {
        JFrame frame = new JFrame("Login");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        JPanel panel = new JPanel(new GridLayout(3, 2, 10, 10));
        JLabel usernameLabel = new JLabel("Username:");
        JTextField usernameField = new JTextField();
        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordField = new JPasswordField();
        JButton loginButton = new JButton("Login");
        JButton backButton = new JButton("Back");
        loginButton.addActionListener(e -> {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());
            User user = userManager.loginUser(username, password);
            if (user != null) {
                frame.dispose();
                showRoleBasedActions(user);
            } else {
                JOptionPane.showMessageDialog(frame, "Invalid credentials! Please try again.");
            }
        });
        backButton.addActionListener(e -> {
            frame.dispose();
            showMainMenu();
        });
    }

```

```

panel.add(usernameLabel);
panel.add(usernameField);
panel.add(passwordLabel);
panel.add(passwordField);
panel.add(loginButton);
panel.add(backButton);
frame.add(panel);
frame.setVisible(true);
}

private void showRoleBasedActions(User user) {
    JFrame frame = new JFrame("Role-Based Actions");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 300);
    JPanel panel = new JPanel(new GridLayout(4, 1, 10, 10));
    if (user.getRole().equals("ADMIN")) {
        JButton createSurveyButton = new JButton("Create Survey");
        JButton viewResultsButton = new JButton("View Survey Results");
        createSurveyButton.addActionListener(e -> {
            String title = JOptionPane.showInputDialog(frame, "Enter survey title:");
            if (title != null && !title.isEmpty()) {
                userManager.createSurvey(title);
                JOptionPane.showMessageDialog(frame, "Survey created successfully!");
            }
        });
        viewResultsButton.addActionListener(e -> {
            String title = JOptionPane.showInputDialog(frame, "Enter survey title to view results:");
            if (title != null && !title.isEmpty()) {
                String results = userManager.viewSurveyResults(title);
                JOptionPane.showMessageDialog(frame, results);
            }
        });
        panel.add(createSurveyButton);
        panel.add(viewResultsButton);
    }
}

```

```

} else if (user.getRole().equals("USER")) {
    JButton respondButton = new JButton("Respond to Survey");
    JButton viewResultsButton = new JButton("View Survey Results");
    respondButton.addActionListener(e -> {
        List<String> surveys = userManager.getSurveyTitles();
        if (surveys.isEmpty()) {
            JOptionPane.showMessageDialog(frame, "No surveys available!");
            return;
        }
        String title = (String) JOptionPane.showInputDialog(
            frame, "Select survey:", "Surveys",
            JOptionPane.QUESTION_MESSAGE, null,
            surveys.toArray(), null
        );
        if (title != null) {
            String response = JOptionPane.showInputDialog(frame, "Enter your
response:");
            if (response != null && !response.isEmpty()) {
                userManager.respondToSurvey(title, response);
                JOptionPane.showMessageDialog(frame, "Response submitted
successfully!");
            }
        }
    });
    viewResultsButton.addActionListener(e -> {
        String title = JOptionPane.showInputDialog(frame, "Enter survey title to view
results:");
        if (title != null && !title.isEmpty()) {
            String results = userManager.viewSurveyResults(title);
            JOptionPane.showMessageDialog(frame, results);
        }
    });
    panel.add(respondButton);
    panel.add(viewResultsButton);

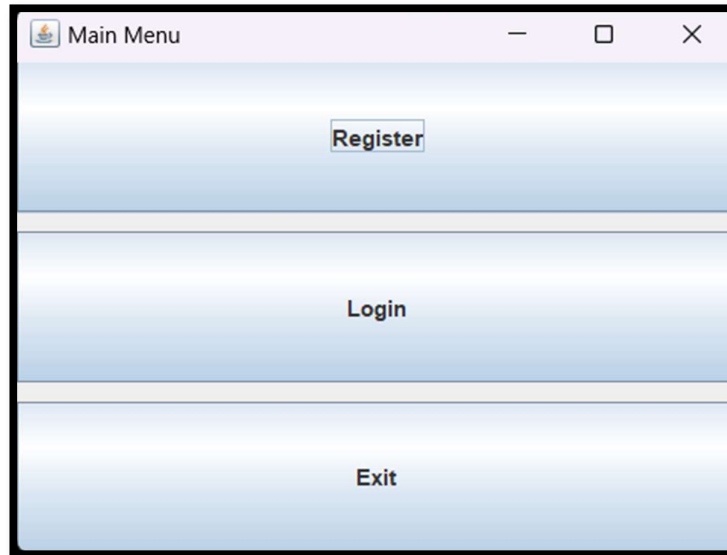
```

```
    }  
    JButton logoutButton = new JButton("Logout");  
    logoutButton.addActionListener(e -> {  
        frame.dispose();  
        showMainMenu();  
    });  
    panel.add(logoutButton);  
    frame.add(panel);  
    frame.setVisible(true);  
}  
}
```



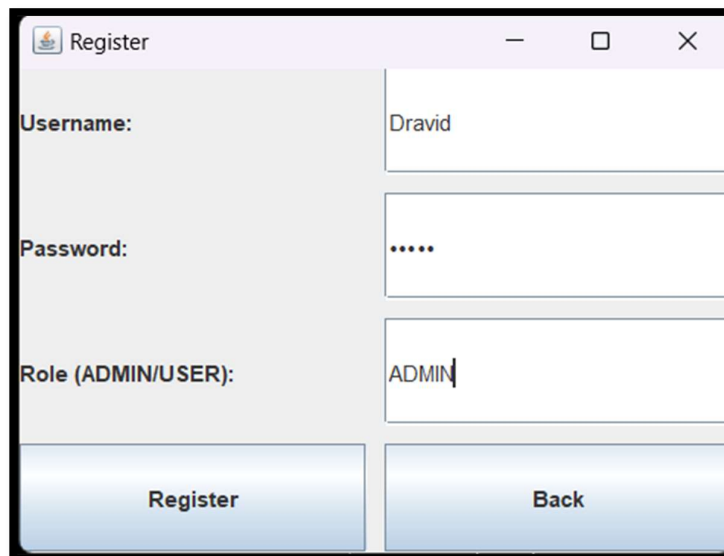
## APPENDIX B (SCREENSHOTS)

### 1. USER INTERFACE



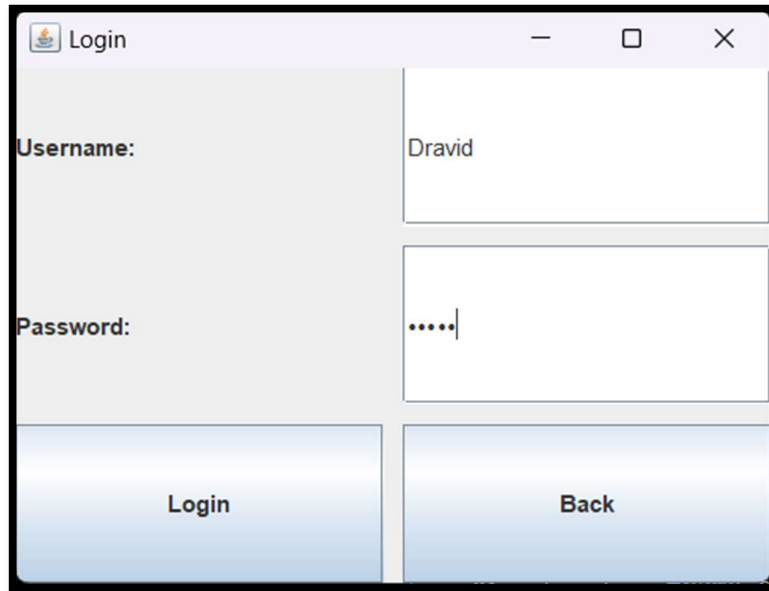
This is the main menu or main page where user can register or login or exit according to their requirements.

### 2. USER REGISTRATION

A screenshot of a software window titled "Register". The window has a light blue gradient background and a white title bar with standard minimize, maximize, and close buttons. The main content area is divided into two columns. The left column contains labels for "Username:", "Password:", and "Role (ADMIN/USER):". The right column contains three text input fields. The first field contains the text "Dravid". The second field contains five dots, indicating a password. The third field contains the text "ADMIN". At the bottom of the window, there are two buttons: "Register" on the left and "Back" on the right.

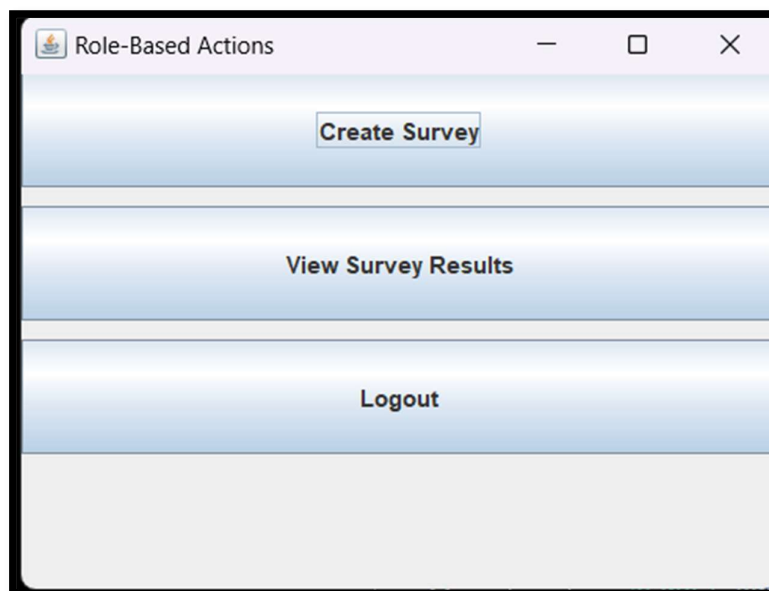
The registration panel allows users to create an account by providing a username, password, and role (ADMIN or USER). The system validates inputs and displays success or error messages accordingly. This ensures secure and structured user registration.

### 3. LOGIN PAGE

A screenshot of a web application window titled "Login". The window has a light purple header bar with standard window controls (minimize, maximize, close). The main content area is divided into two columns. The left column is light gray and contains labels "Username:" and "Password:". The right column is white and contains two text input fields. The first field contains the text "Dravid". The second field contains five dots, indicating a password. Below the input fields are two blue buttons with white text: "Login" on the left and "Back" on the right.

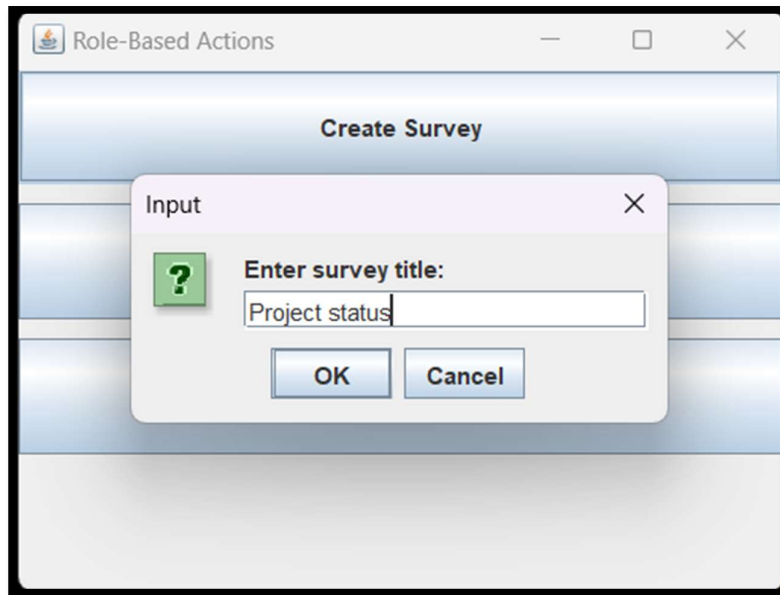
The login panel requires users to enter their credentials. Successful authentication redirects users to their respective role-specific dashboards, while invalid credentials display an error message. This module demonstrates secure authentication.

### 4. ADMIN DASHBOARD

A screenshot of a web application window titled "Role-Based Actions". The window has a light purple header bar with standard window controls (minimize, maximize, close). The main content area is a light gray rectangle. Overlaid on this are three blue buttons with white text, stacked vertically. The top button is labeled "Create Survey". The middle button is labeled "View Survey Results". The bottom button is labeled "Logout".

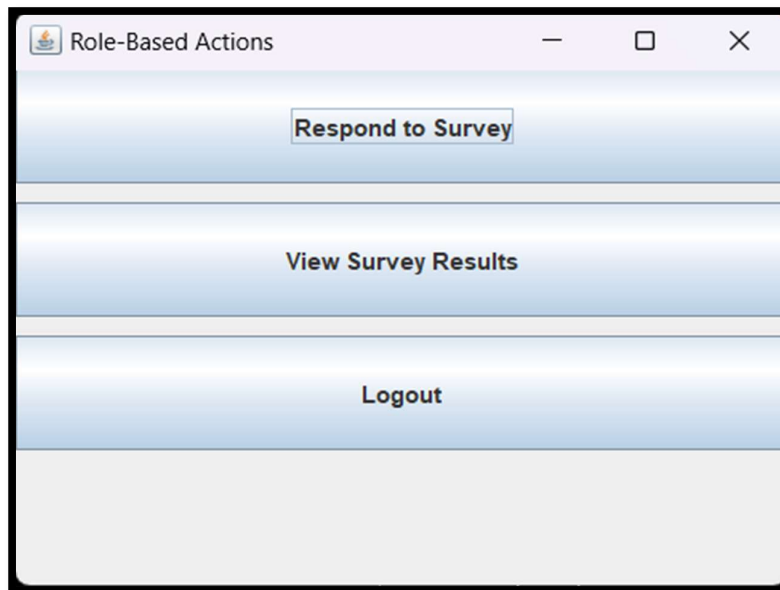
Upon login, ADMIN users are presented with options to create surveys and view survey results. This role-specific access ensures that only authorized users can manage surveys.

## 5. SURVEY CREATION



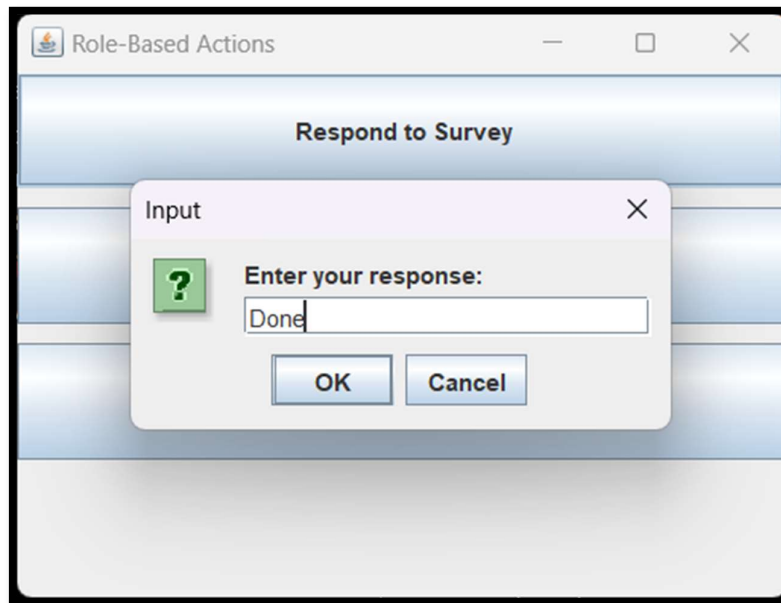
ADMIN users can create surveys by entering a title and question. The created survey is stored dynamically, demonstrating the system's ability to handle data efficiently.

## 6. USER DASHBOARD



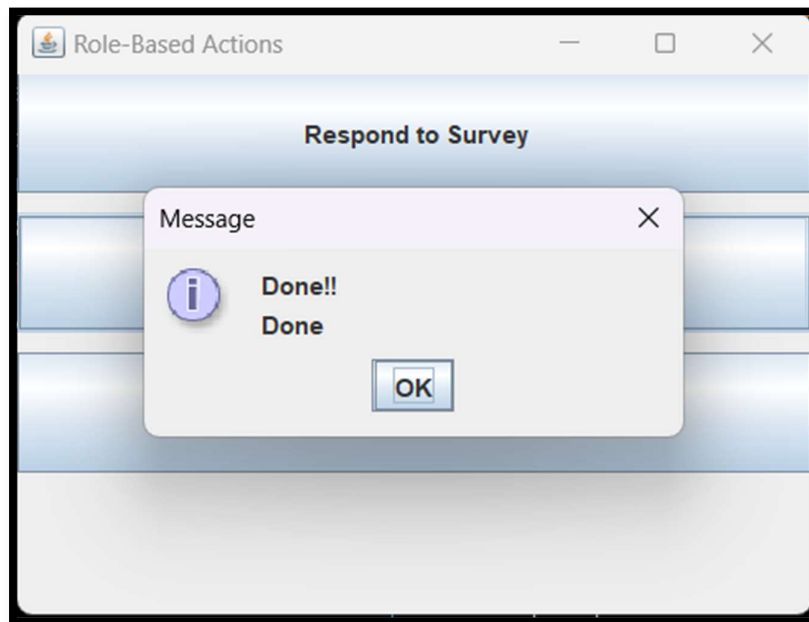
USER roles have access to respond to surveys and view existing results. The options are restricted to their role, highlighting the role-based access control implemented in the system.

## 7. SURVEY RESPONDING



USER roles can view available surveys and submit their responses. This feature ensures seamless interaction with the surveys.

## 8. VIEWING RESULTS



Both ADMIN and USER roles can view survey results. This feature displays responses in a clear format, ensuring transparency and user satisfaction.

## REFERENCES

1. Herbert Schildt (2018) Java: The Complete Reference, McGraw-Hill Education, 11th Edition, pp. 1-1150.
2. Cay S. Horstmann (2018) Core Java Volume I – Fundamentals, Pearson Education, 11th Edition, pp. 1-1040.
3. Bruce Eckel (2006) Thinking in Java, Prentice Hall, 4th Edition, pp. 1-1156.
4. Programming with Mosh: <https://www.youtube.com/c/programmingwithmosh>
5. Telusko: <https://www.youtube.com/c/Telusko>
6. CodeWithHarry: <https://www.youtube.com/c/CodeWithHarry>