RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS23333-Object Oriented Programming using Java

SCHEMES FINDER

Submitted by

TEJASWINI K 231001228

YASASWINI D 231001252

Of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

(AUTONOMOUS)

THANDALAM,CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

Certified that this project report **"SCHEMES FINDER"**is the Bonafide Work of **"TEJASWINI K (231001228), YASASWINI D (231001252)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Dr.P.Valarmathie

Professor and Head

Information Technology,

Rajalakshmi Engineering College

(Autonomous),

Thandalam, Chennai - 602 105.

SIGNATURE

Mr.K.E.Narayana

Assistant Professor,

Information Technology,

Rajalakshmi Engineering College

(Autonomous),

Thandalam, Chennai - 602 105.

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

ABSTRACT

Schemes Finder is a platform facilitating easy access to a diverse range ofschemes offeredby various organizations and government bodies. Users canexplore schemes through intuitive search and browsing features, findingdetails such as eligibility criteria, benefits, andapplication processes. Schemecategorization streamlines navigation, while user feedbackfosters communityengagement. Notifications keep users updated on new opportunities anddeadlines. Admins manage the platform, ensuring scheme accuracy andsystem integrity.With a user-centric design, the platform aims to empowerindividuals and organizations byproviding transparent and accessibleinformation for informed decision-making.

1.INTRODUCTION

1.1 INTRODUCTION

The Scheme Finder project aims to create an efficient system for discovering and managing various government and private schemes available to individuals and organizations. This system will provide a centralized platform where users can easily search for and access detailed information about different schemes, eligibility criteria, application processes, and benefits. The primary goal is to enhance accessibility and awareness of available schemes, ensuring that potential beneficiaries can take full advantage of the support offered.

1.2 OBJECTIVES

1.CENTRALIZED INFORMATION: To compile a comprehensive database of schemes from various sectors.

2.USER-FRIENDLY INTERFACE: To develop an intuitive interface that allows users to search and filter schemes based on specific criteria.

3.UP-TO-DATE DATA: To maintain accurate and current information regarding the schemes.

4.ELIGIBILITY CHECK: To provide tools for users to check their eligibility for different schemes.

5.APPLICATION GUIDANCE: To offer step-by-step guidance on how to apply for the schemes.

1.3 MODULES

1.USER AUTHENTICATION: Module for user login and registration.

2.SCHEME DATABASE MANAGEMENT: Module for updating, and deleting scheme information.

3.SEARCH AND FILTER: Module for searching and filtering schemes based on various parameters.

4.ELIGIBILITY CHECKER:Tool for users to check their eligibility for scheme.

5.APPLICATION GUIDANCE: Step-by-step instructions for applying to schemes.

6.NOTIFICATIONS: Module for sending updates and reminders to users about scheme deadlines and updates.

2.SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The Scheme Finder project leverages modern web technologies and frameworks to build a robust and scalable application. The backend is developed using Java with a SQL database connectivity for managing scheme data. The frontend is designed using Java Swing and AWT tool kit to ensure a responsive and user-friendly interface.

2.2 LANGUAGES

2.1 SQL

The Scheme Finder project leverages modern web technologies and frameworks to build a robust and scalable application. The backend is developed using Java with a SQL database connectivity for managing scheme data. The frontend is designed using Java Swing and AWT tool kit to ensure a responsive and user-friendly interface.

2.2 JDBC CONNECTIVITY

JDBC (Java Database Connectivity) is a Java-based API that enables communication between Java applications and relational databases. It provides a standardized method for connecting to databases, executing SQL queries, and retrieving results. JDBC is part of the Java Standard Edition (Java SE) and supports various relational databases, making it versatile for different data sources. It uses a driver-based architecture, ensuring seamless database interaction, and is essential for building data-driven Java applications.

3.REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

1) FUNCTIONAL REQUIREMENTS:

- User registration and login system.
- Database for storing scheme information.
- Search and filter functionality.
- Eligibility checking tool.
- Application guidance and steps.
- Notification system.

2)NON-FUNCTIONAL REQUIREMENTS:

- Scalability to handle a growing number of users and schemes.
- Security measures to protect user data.
- High availability and reliability.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

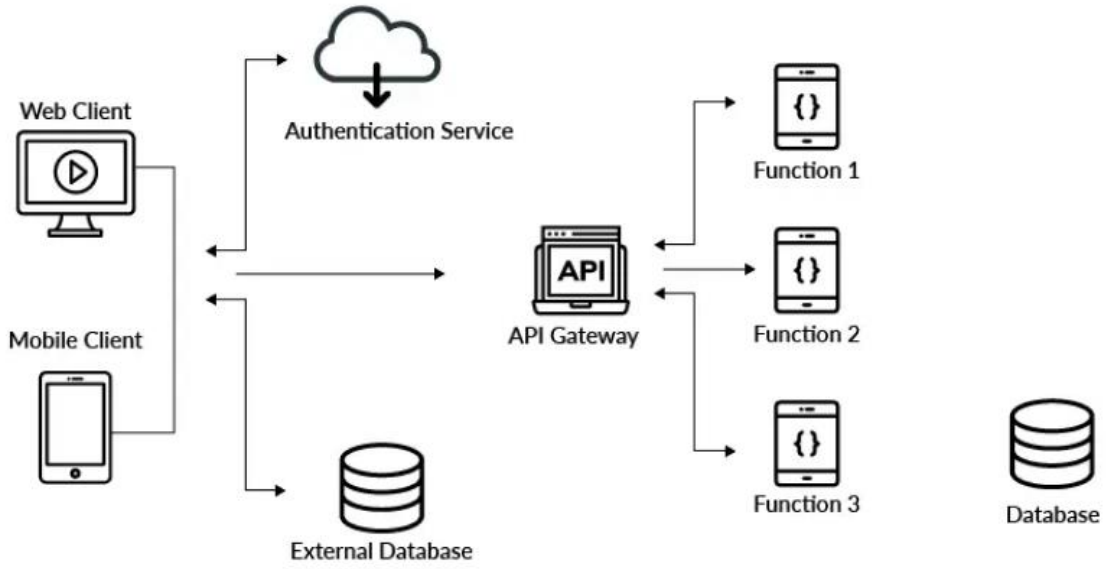
1)HARDWARE:

- Server with sufficient CPU and RAM to handle multiple requests.
- Storage for database and backups.

2)SOFTWARE:

- Operating System: Linux/Windows
- Database: MySQL/PostgreSQL
- Backend: JDBC Connectivity
- Frontend: Java Swing
- Version Control: Git

3.3 ARCHITECTURE DIAGRAM



4.PROGRAM CODE

JDBC CONNECTIVITY CODE:

DATABASE CONNECTION UTILITY (DBConnection.java)

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBConnection {

    private static final String URL = "jdbc:mysql://localhost:3306/mydb";

    private static final String USER = "root";

    private static final String PASSWORD = "Aswini^2";

    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(URL, USER, PASSWORD);
```

}}

HOME SERVLET (HomeServlet.java)

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.IOException;
public class HomeServlet extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
      String imagePath = request.getServletContext().getRealPath("static/Image/bg.jpg");
      request.setAttribute("image", imagePath);
      RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
      dispatcher.forward(request, response);
   }
}
```

JAVA SWING CODE:

MySchemeUI.JAVA

```
import javax.swing.*;
import java.awt.*;
public class MySchemeUI extends JFrame {
   public MySchemeUI() {
      setTitle("My Scheme - Home");
      setSize(1500, 1000);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLayout(null);
      JPanel headerPanel = new JPanel();
      headerPanel.setBounds(0, 0, 1500, 90);
headerPanel.setBackground(new Color(35, 192, 35))
      headerPanel.setLayout(null);
```

```
add(headerPanel);
        JLabel logo1 = new JLabel(new ImageIcon("path/to/emblem-black.png"));
        logo1.setBounds(40, 20, 50, 50);
        headerPanel.add(logo1);
        JLabel logo2 = new JLabel(new ImageIcon("path/to/myscheme-logo.png"));
        logo2.setBounds(90, 30, 150, 50);
        headerPanel.add(logo2);
        JLabel logo3 = new JLabel(new ImageIcon("path/to/digital-india.png"));
        logo3.setBounds(330, 20, 50, 50);
        headerPanel.add(logo3);
        JTextField searchField = new JTextField("Search...");
        searchField.setBounds(550, 20, 300, 30);
    headerPanel.add(searchField);
        JButton searchButton = new JButton(new ImageIcon("path/to/search-icon.png"));
        searchButton.setBounds(860, 20, 30, 30);
        headerPanel.add(searchButton);
        JButton signInButton = new JButton("Sign In");
        signInButton.setBounds(1100, 20, 100, 30);
        signInButton.setBackground(new Color(18, 122, 83));
        signInButton.setForeground(Color.WHITE);
        signInButton.setBorder(BorderFactory.createEmptyBorder());
        headerPanel.add(signInButton);
        JLabel banner = new JLabel(new ImageIcon("path/to/bg.jpg"));
        banner.setBounds(0, 90, 1500, 700);
        add(banner);
        JButton totalSchemesButton = createInfoBox("2000+", "Total Schemes", 350, 800);
        add(totalSchemesButton);       JButton centralSchemesButton = createInfoBox("510+",
"Central Schemes", 650, 800);
```

```
    add(centralSchemesButton);
        JButton stateSchemesButton = createInfoBox("1650+", "State Schemes", 950, 800);
        add(stateSchemesButton);
        JPanel footerPanel = new JPanel();
        footerPanel.setBounds(0, 900, 1500, 100);
        footerPanel.setBackground(new Color(51, 76, 166));
        footerPanel.setLayout(new GridLayout(1, 3));
        add(footerPanel);
        JLabel contactLabel = new JLabel("<html><center>Contact Us<br>Email: support-
myscheme@gov.in<br>Phone: (011) 24303714</center></html>", JLabel.CENTER);
        contactLabel.setForeground(Color.WHITE);
 footerPanel.add(contactLabel);
        JLabel linksLabel = new JLabel("<html><center>Quick Links<br>Home | About | Services
| Contact</center></html>", JLabel.CENTER);
        linksLabel.setForeground(Color.WHITE);
   footerPanel.add(linksLabel);
        JLabel socialLabel = new JLabel("<html><center>Follow Us<br>Facebook | Twitter |
Instagram</center></html>", JLabel.CENTER);
        socialLabel.setForeground(Color.WHITE);
  footerPanel.add(socialLabel);
        setVisible(true);
    }
    private JButton createInfoBox(String number, String text, int x, int y) {
        JButton button = new JButton("<html><center><span style='font-size:20px;'>" + number
+ "</span><br>" + text + "</center></html>");
        button.setBounds(x, y, 180, 75);
        button.setBackground(new Color(35, 192, 35));
        button.setForeground(Color.WHITE);
        button.setBorder(BorderFactory.createEmptyBorder());
        return button;
```

```
}
public static void main(String[] args) {
        new MySchemeUI();
    }
}
```

LoginSignupUI.JAVA

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class LoginSignupUI extends JFrame {
    CardLayout cardLayout;
    JPanel mainPanel;
    public LoginSignupUI() {
        setTitle("Login/Sign Up");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 800);
        setLayout(new BorderLayout());
        cardLayout = new CardLayout();
        mainPanel = new JPanel(cardLayout);
        mainPanel.add(createLoginPanel(), "Login");
        mainPanel.add(createSignupPanel(), "SignUp");
        add(mainPanel, BorderLayout.CENTER);
        cardLayout.show(mainPanel, "Login");
        setVisible(true);
    }
    private JPanel createLoginPanel() {
        JPanel loginPanel = new JPanel(null);
```

```
    loginPanel.setBackground(new Color(240, 240, 240
        JLabel bgImage = new JLabel(new ImageIcon("background-image.jpg"));
        bgImage.setBounds(0, 0, 600, 800);
        loginPanel.add(bgImage);
        JPanel loginBox = new JPanel(null);
        loginBox.setBounds(100, 200, 400, 300);
        loginBox.setBackground(new Color(255, 255, 255, 230));
 loginBox.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, 1));
        JLabel title = new JLabel("MyScheme");
        title.setFont(new Font("Arial", Font.BOLD, 36));
        title.setForeground(new Color(76, 175, 80));
        title.setHorizontalAlignment(SwingConstants.CENTER);
        title.setBounds(50, 20, 300, 40);
        loginBox.add(title);
        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setBounds(30, 80, 100, 25);
        loginBox.add(usernameLabel);
JTextField usernameField = new JTextField();
        usernameField.setBounds(150, 80, 200, 30);
        loginBox.add(usernameField);
        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(30, 130, 100, 25);
        loginBox.add(passwordLabel);
        JPasswordField passwordField = new JPasswordField();
        passwordField.setBounds(150, 130, 200, 30);
        loginBox.add(passwordField);
        JButton loginButton = new JButton("Login");
        loginButton.setBounds(150, 180, 100, 30);
```

```
        loginButton.setBackground(new Color(76, 175, 80));
        loginButton.setForeground(Color.WHITE);
        loginButton.setBorder(BorderFactory.createEmptyBorder());
        loginButton.addActionListener(e -> JOptionPane.showMessageDialog(this, "Logged
in!"));
        loginBox.add(loginButton);
        JButton signupRedirect = new JButton("Sign Up");
        signupRedirect.setBounds(150, 220, 100, 30);
        signupRedirect.setBackground(new Color(63, 81, 181));
        signupRedirect.setForeground(Color.WHITE);
        signupRedirect.setBorder(BorderFactory.createEmptyBorder());
        signupRedirect.addActionListener(e -> cardLayout.show(mainPanel, "SignUp"));
  loginBox.add(signupRedirect);
        loginPanel.add(loginBox);
        return loginPanel;
    }
    private JPanel createSignupPanel() {
        JPanel signupPanel = new JPanel(null);
        signupPanel.setBackground(new Color(240, 240, 240));
        JPanel signupBox = new JPanel(null);
        signupBox.setBounds(100, 200, 400, 400);
        signupBox.setBackground(new Color(255, 255, 255, 230));
signupBox.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, 1));
        JLabel title = new JLabel("Sign Up");
        title.setFont(new Font("Arial", Font.BOLD, 30));
        title.setHorizontalAlignment(SwingConstants.CENTER);
        title.setBounds(50, 20, 300, 40);
        signupBox.add(title);
        JLabel emailLabel = new JLabel("Email:");
```

```
        emailLabel.setBounds(30, 80, 100, 25);
      signupBox.add(emailLabel);
        JTextField emailField = new JTextField();
        emailField.setBounds(150, 80, 200, 30);
        signupBox.add(emailField);
        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(30, 130, 100, 25);
    signupBox.add(passwordLabel);
        JPasswordField passwordField = new JPasswordField();
        passwordField.setBounds(150, 130, 200, 30);
        signupBox.add(passwordField);
        JLabel confirmPasswordLabel = new JLabel("Confirm Password:");
        confirmPasswordLabel.setBounds(30, 180, 120, 25);
        signupBox.add(confirmPasswordLabel);
        JPasswordField confirmPasswordField = new JPasswordField();
        confirmPasswordField.setBounds(150, 180, 200, 30);
        signupBox.add(confirmPasswordField);
        JButton signupButton = new JButton("Sign Up");
        signupButton.setBounds(150, 230, 100, 30);
        signupButton.setBackground(new Color(76, 175, 80));
        signupButton.setForeground(Color.WHITE);
        signupButton.setBorder(BorderFactory.createEmptyBorder());
        signupButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (String.valueOf(passwordField.getPassword())
                        .equals(String.valueOf(confirmPasswordField.getPassword()))) {
                    JOptionPane.showMessageDialog(signupPanel, "Sign-up successful!");
                } else {
```

```
                JOptionPane.showMessageDialog(signupPanel, "Passwords do not match.",
                        JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        signupBox.add(signupButton);
        JButton loginRedirect = new JButton("Login");
        loginRedirect.setBounds(150, 280, 100, 30);
        loginRedirect.setBackground(new Color(63, 81, 181));
        loginRedirect.setForeground(Color.WHITE);
        loginRedirect.setBorder(BorderFactory.createEmptyBorder());
        loginRedirect.addActionListener(e -> cardLayout.show(mainPanel, "Login"));
        signupBox.add(loginRedirect);
        signupPanel.add(signupBox);
        return signupPanel;
    }
    public static void main(String[] args) {
        new LoginSignupUI();
    }
}
```

MySchemeApp.JAVA

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class MySchemeApp extends JFrame {
    public MySchemeApp() {
```

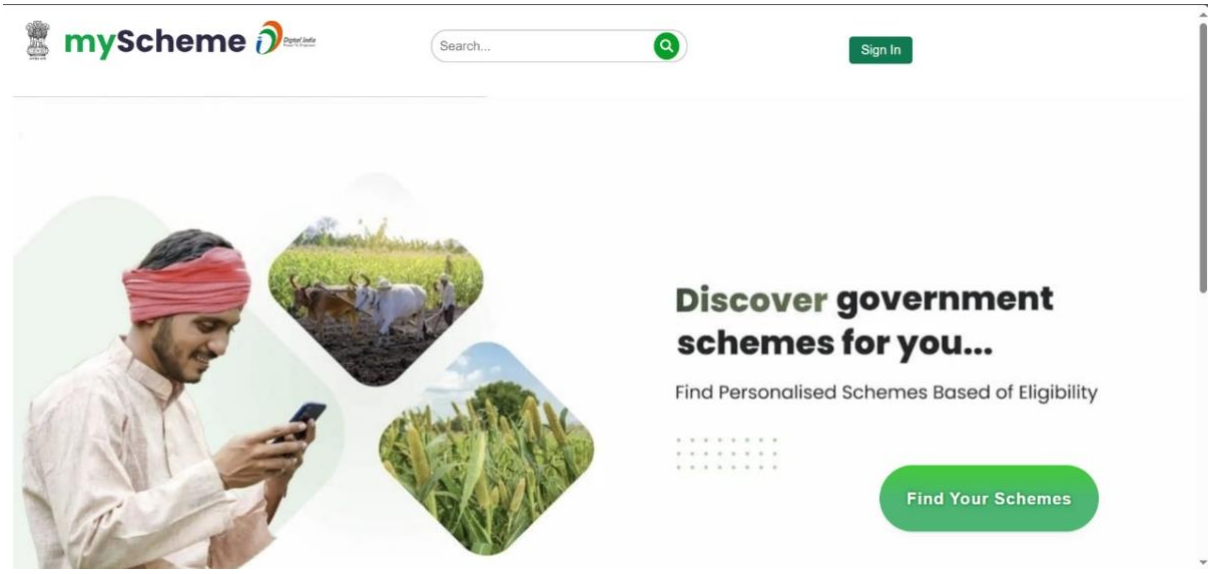
```
        setTitle("My Scheme - Home");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(1200, 800);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
        JPanel header = new JPanel(new FlowLayout(FlowLayout.LEFT));
        header.setBackground(new Color(255, 255, 255));
        JLabel logo1 = new JLabel(new ImageIcon("path_to_logo1"));
        JLabel logo2 = new JLabel(new ImageIcon("path_to_logo2"));
        JLabel logo3 = new JLabel(new ImageIcon("path_to_logo3"));
        header.add(logo1);
        header.add(logo2);
        header.add(logo3);
        JPanel userOptions = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        JTextField searchField = new JTextField(20);
        JButton searchButton = new JButton("Search");
        JButton signInButton = new JButton("Sign In");
        userOptions.add(searchField);
        userOptions.add(searchButton);
 userOptions.add(signInButton);
        header.add(userOptions);
        add(header, BorderLayout.NORTH);
        JPanel content = new JPanel();
content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));
content.setBackground(new Color(240, 240, 240));
        content.setBorder(new EmptyBorder(20, 20, 20, 20));
        for (int i = 0; i < 3; i++) {
            JPanel schemePanel = new JPanel();
```

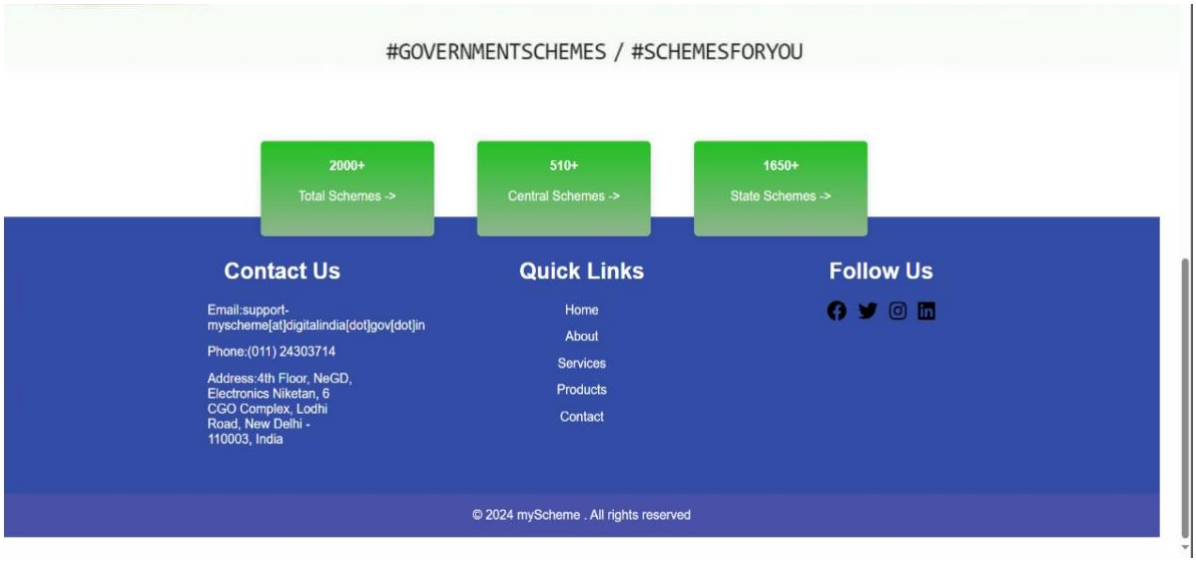
```
            schemePanel.setLayout(new BoxLayout(schemePanel, BoxLayout.Y_AXIS));
            schemePanel.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));
            JLabel schemeName = new JLabel("Scheme Name: Scheme " + (i + 1));
            JLabel provider = new JLabel("Provided by: Provider " + (i + 1));
            JLabel benefits = new JLabel("Benefits: Lorem ipsum...");
            JButton linkButton = new JButton("Visit Site");
            linkButton.addActionListener(e -> JOptionPane.showMessageDialog(this, "Redirecting to website..."));
            schemePanel.add(schemeName);
            schemePanel.add(provider);
            schemePanel.add(benefits);
            schemePanel.add(linkButton);
            content.add(Box.createVerticalStrut(10));
            content.add(schemePanel);
        }
        JScrollPane scrollPane = new JScrollPane(content);
        add(scrollPane, BorderLayout.CENTER);
        JPanel footer = new JPanel(new GridLayout(3, 1));
        footer.setBackground(new Color(51, 76, 166));
        JLabel contact = new JLabel("Contact Us: support-myscheme@digitalindia.gov.in | (011) 24303714");
        JLabel address = new JLabel("Address: 4th Floor, NeGD, Electronics Niketan, New Delhi");
        JLabel copyright = new JLabel("© 2024 MyScheme. All rights reserved.", SwingConstants.CENTER);
        contact.setForeground(Color.WHITE);
        address.setForeground(Color.WHITE);
copyright.setForeground(Color.WHITE);
        footer.add(contact);
        footer.add(address);
```

```
        footer.add(copyright);
        add(footer, BorderLayout.SOUTH);
        setVisible(true);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(MySchemeApp::new);
    }
}
```

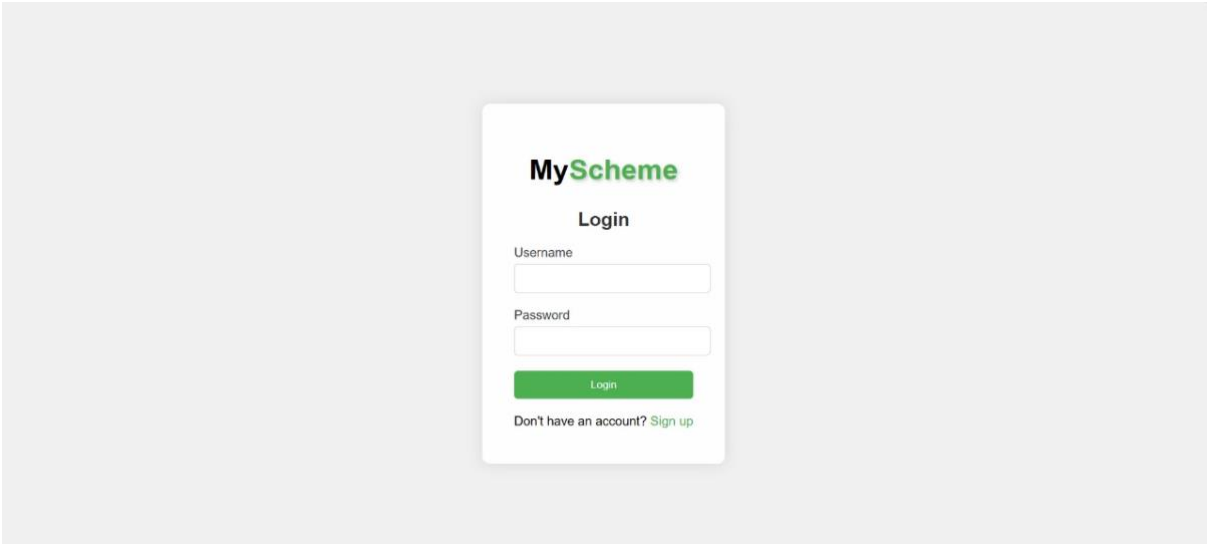
5.RESULTS AND DISCUSSION

Step 1:Home page for Schemes Finder Website

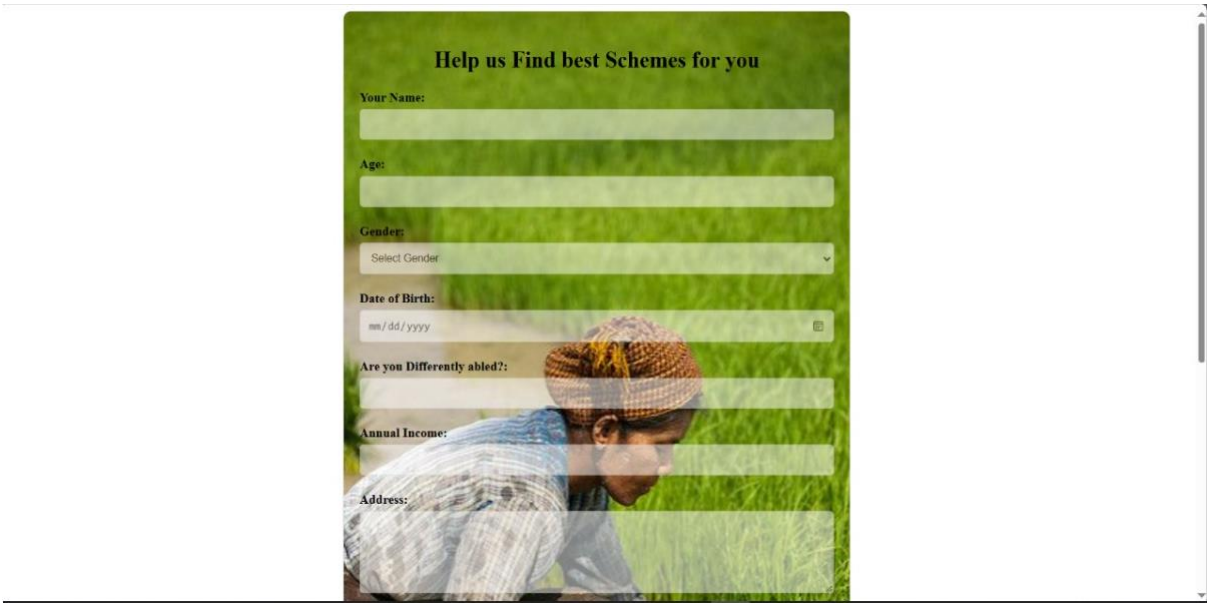


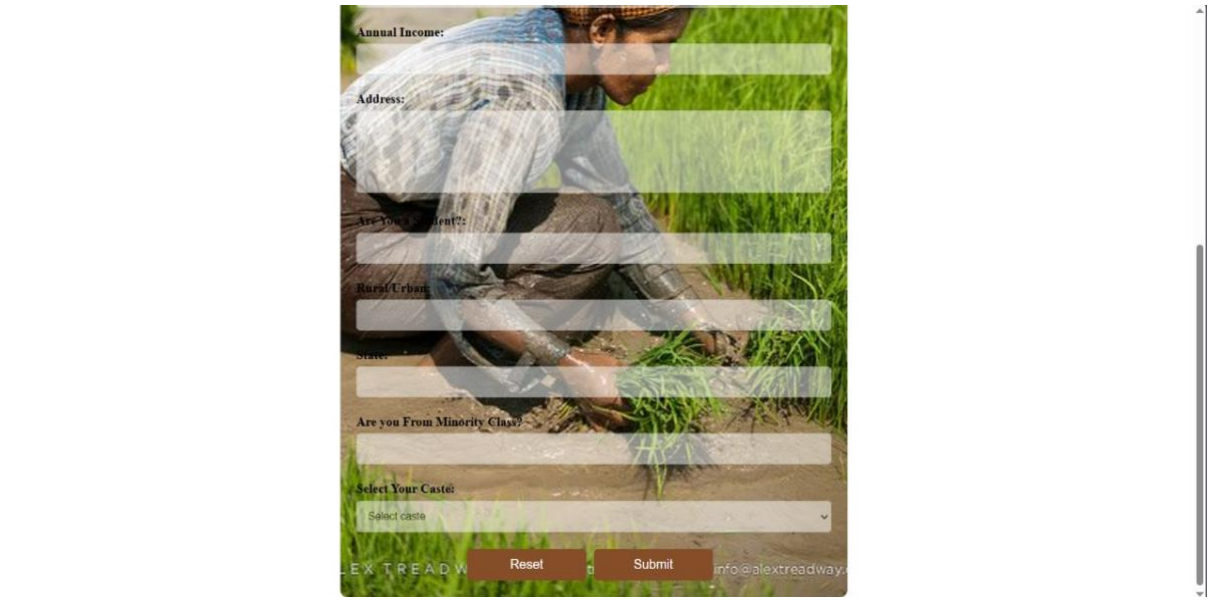


Step 2: Login form of the web page

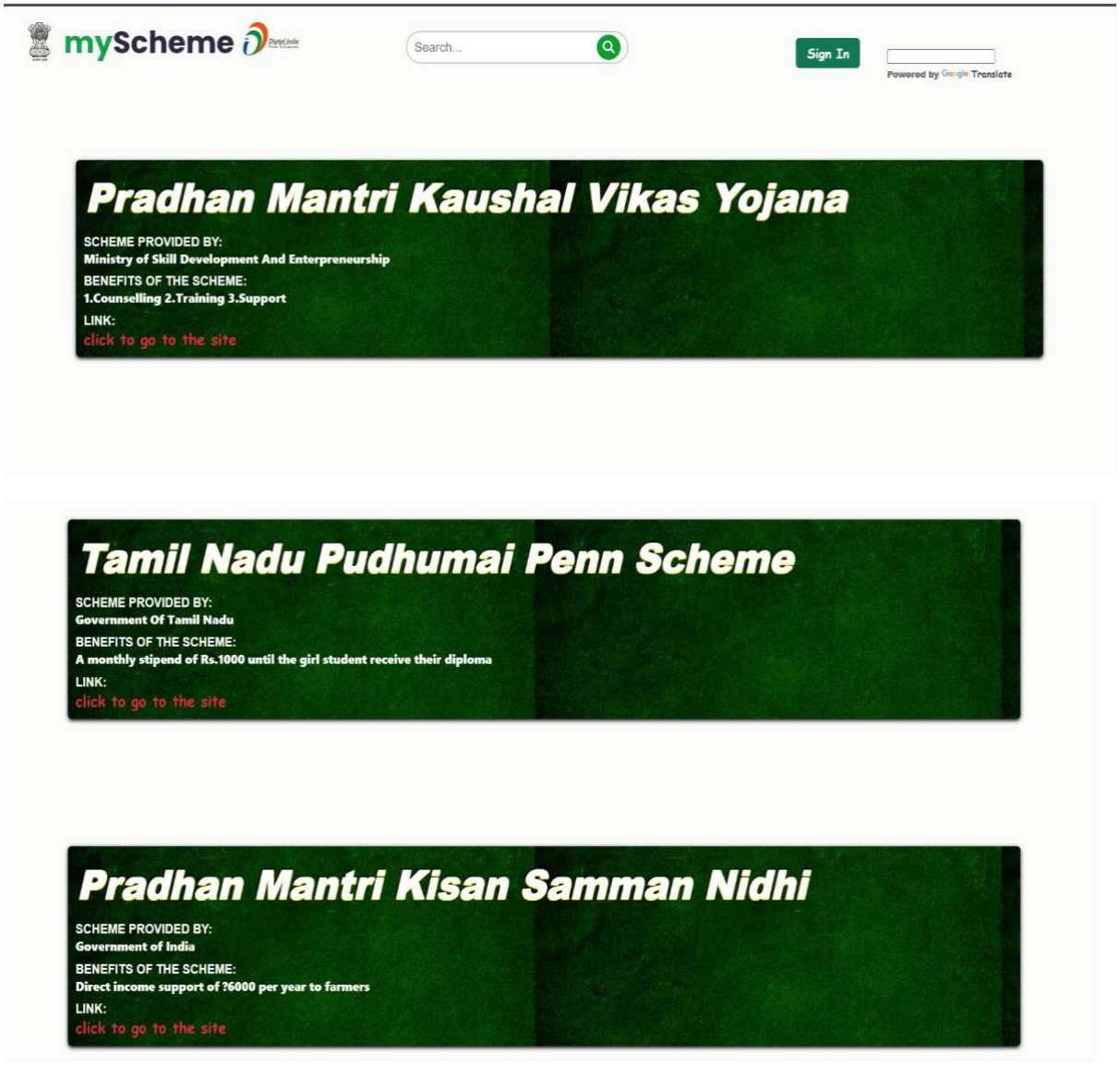


Step 3: Form to check eligible schemes

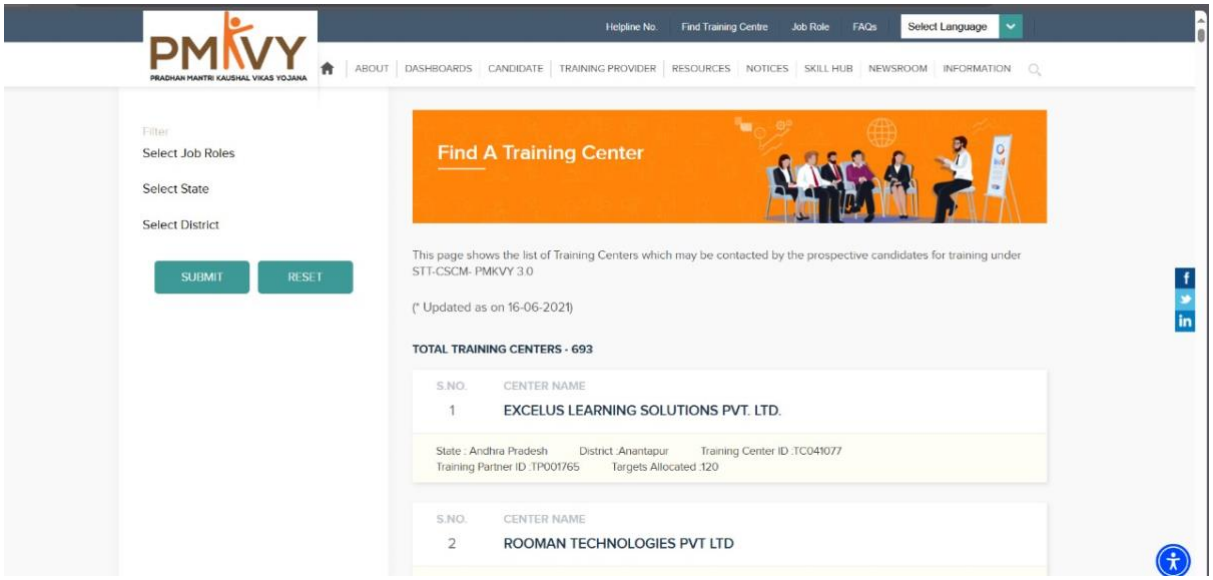




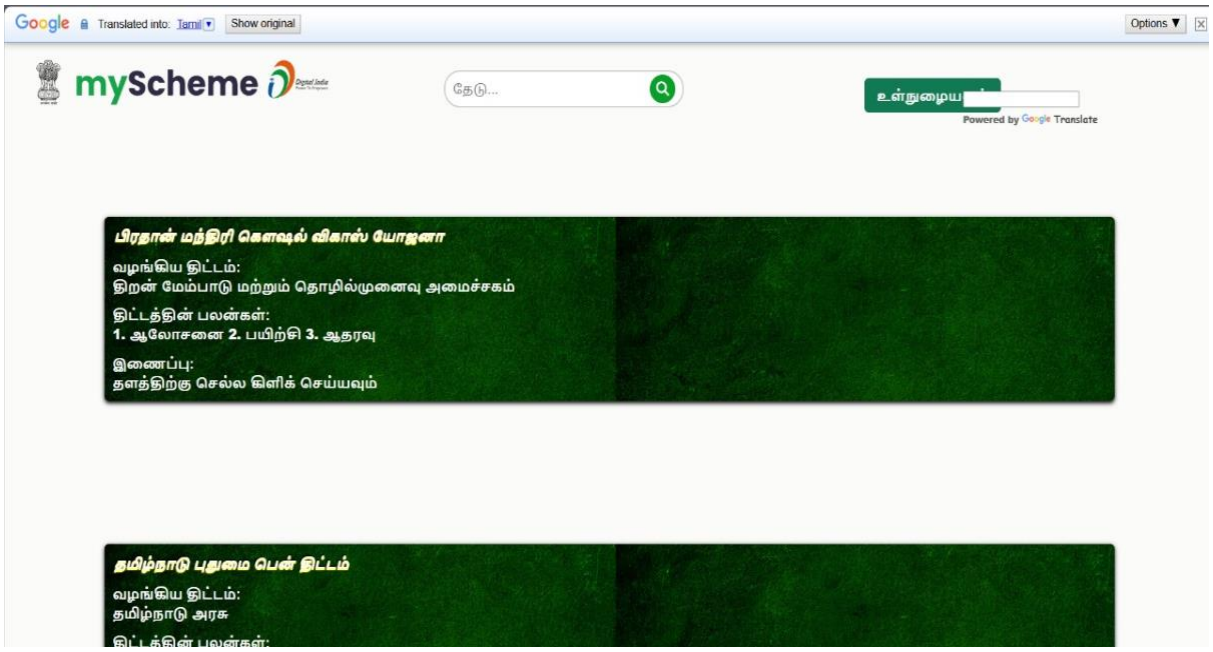
Step 4: Displaying the schemes that are eligible

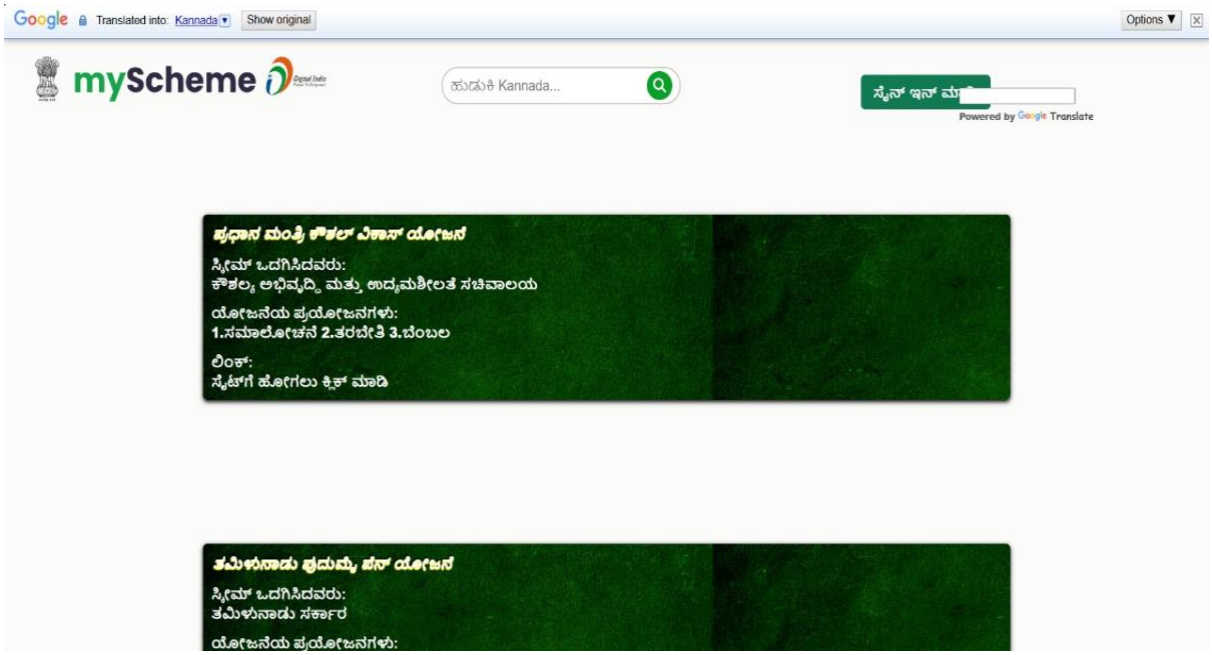


Step 5: Redirecting to the link to know more details about schemes and applying for it

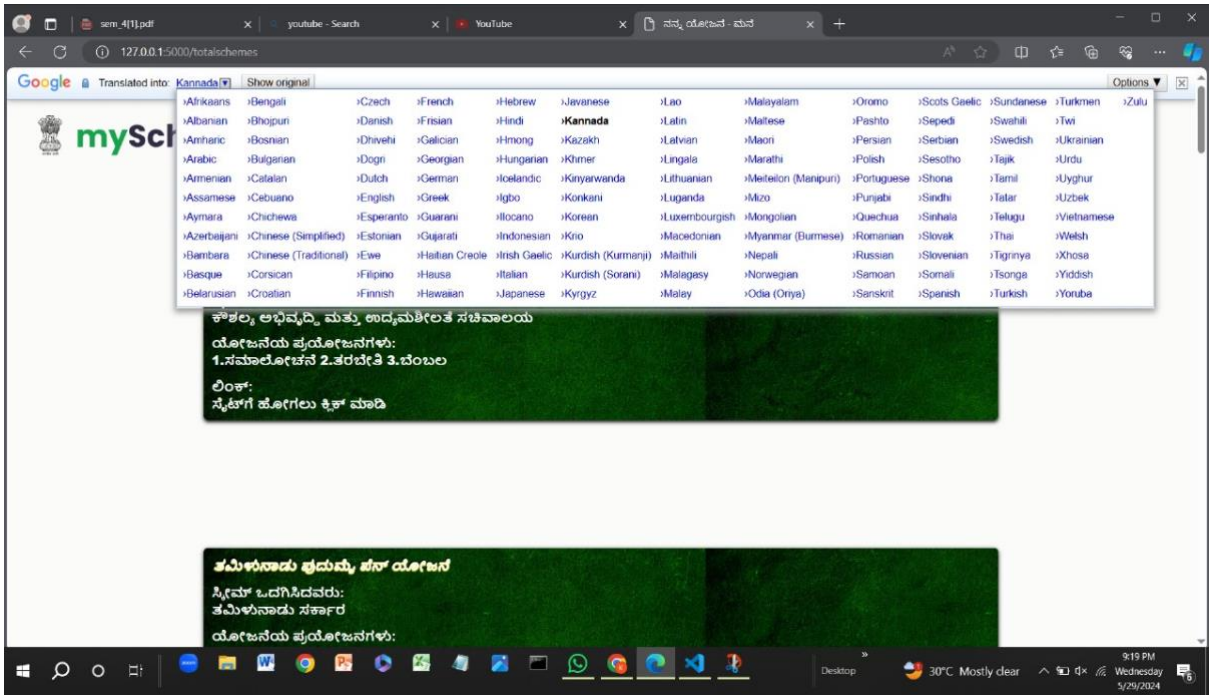


Step 6: Translating to other languages





Step 7: Languages that are available



6.CONCLUSION

The Scheme Finder project has effectively met its goals of providing a centralized platform for accessing information on various schemes. By simplifying the process of discovering, checking eligibility for, and applying to schemes, the application has the potential to significantly enhance users' ability to benefit from available support.

KEY ACHIEVEMENTS

- Developed a user-friendly, efficient interface.

- Established a comprehensive and up-to-date database of schemes.

- Implemented effective search, filter, and eligibility checking functionalities.

- Provided clear application guidance and timely notifications.

FUTURE IMPROVEMENTS

- Expand Scheme Database: Continuously add new schemes and update existing ones to ensure comprehensive coverage.

- User Experience Enhancements: Based on ongoing user feedback, further

refine the interface and add new features to improve usability.

- Mobile Application: Develop a mobile version of the application to increase accessibility and convenience for users on-the-go.

- AI Integration: Implement AI-driven recommendations to suggest schemes to users based on their profiles and search history.

In conclusion, the Scheme Finder project has laid a strong foundation for a valuable resource that can significantly aid individuals and organizations in accessing various support schemes. Continued development and refinement will further enhance its impact and usability.

7.REFERENCES

1. "Development of Agricultural Information Systems for Farmer Assistance"

Authors: R. Sharma, V. Gupta

This paper highlights the design of an online platform to help farmers access government schemes, using data-driven approaches to ensure personalized scheme recommendations.

2. "Application of Big Data in Farmer Welfare Management"

Authors: S. Patel, N. Rao

This study discusses leveraging big data technologies to build efficient systems that match farmers with appropriate subsidy schemes based on real-time data and predictive analytics.

3. "Design and Implementation of a Web-based Agricultural Support System"

Authors: A. Khan, M. Verma

Focused on creating an interactive website for farmers, this paper emphasizes usability and integration of mobile-friendly tools for easy access to government resources and schemes.

4. "Integration of IoT and AI for Enhancing Farmer Support Systems"

Authors: P. Singh, K. Das

This research introduces IoT-enabled sensors and AI algorithms to collect farm-specific data, which is then used to recommend suitable government programs and optimize resource allocation.

5. "Smart Farming Management Systems for Scheme Allocation"

Authors: L. Chen, J. Wong

This paper proposes a cloud-based solution for managing farmer-related data and automating scheme discovery, ensuring transparency and accessibility for marginalized communities.