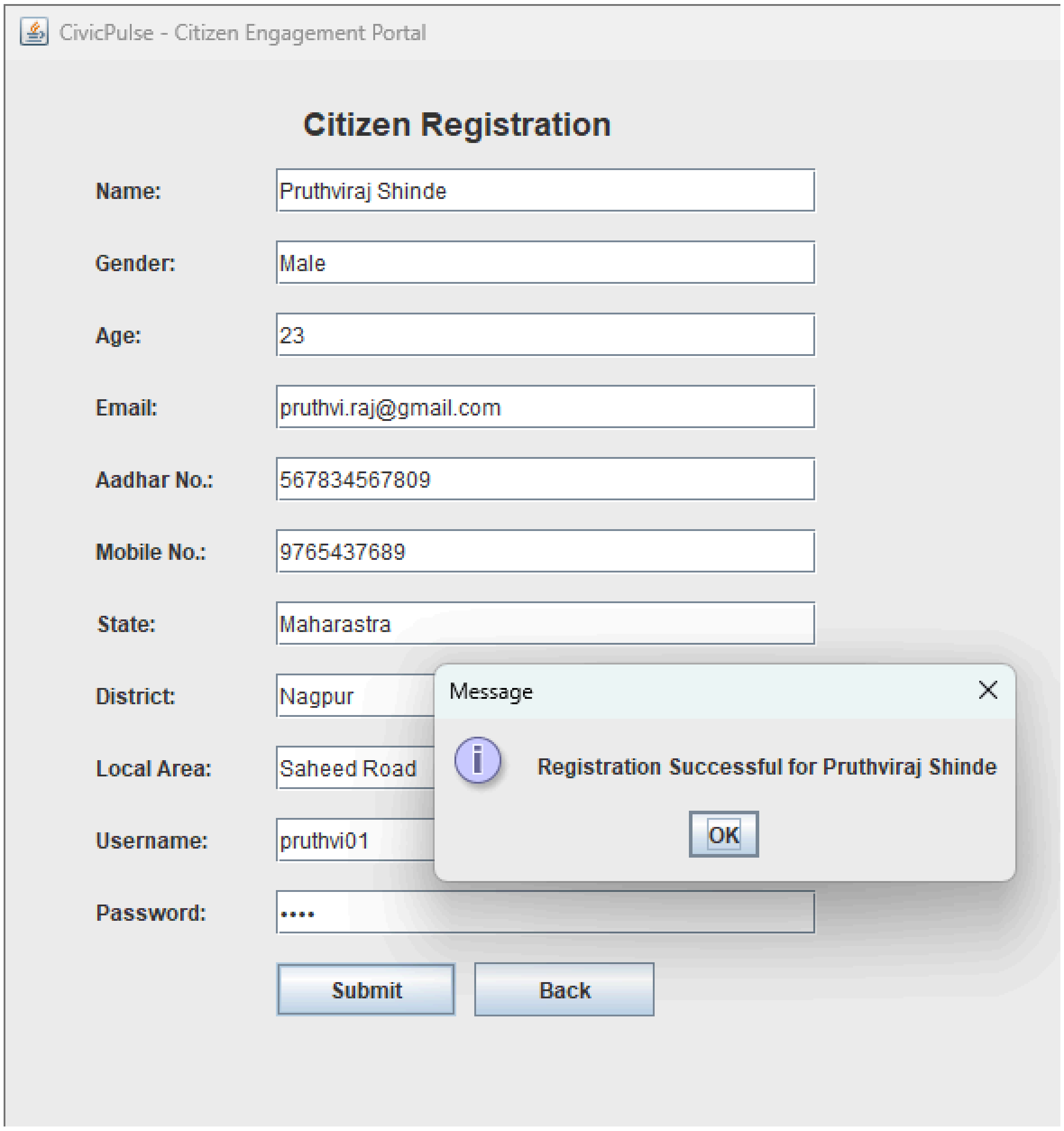
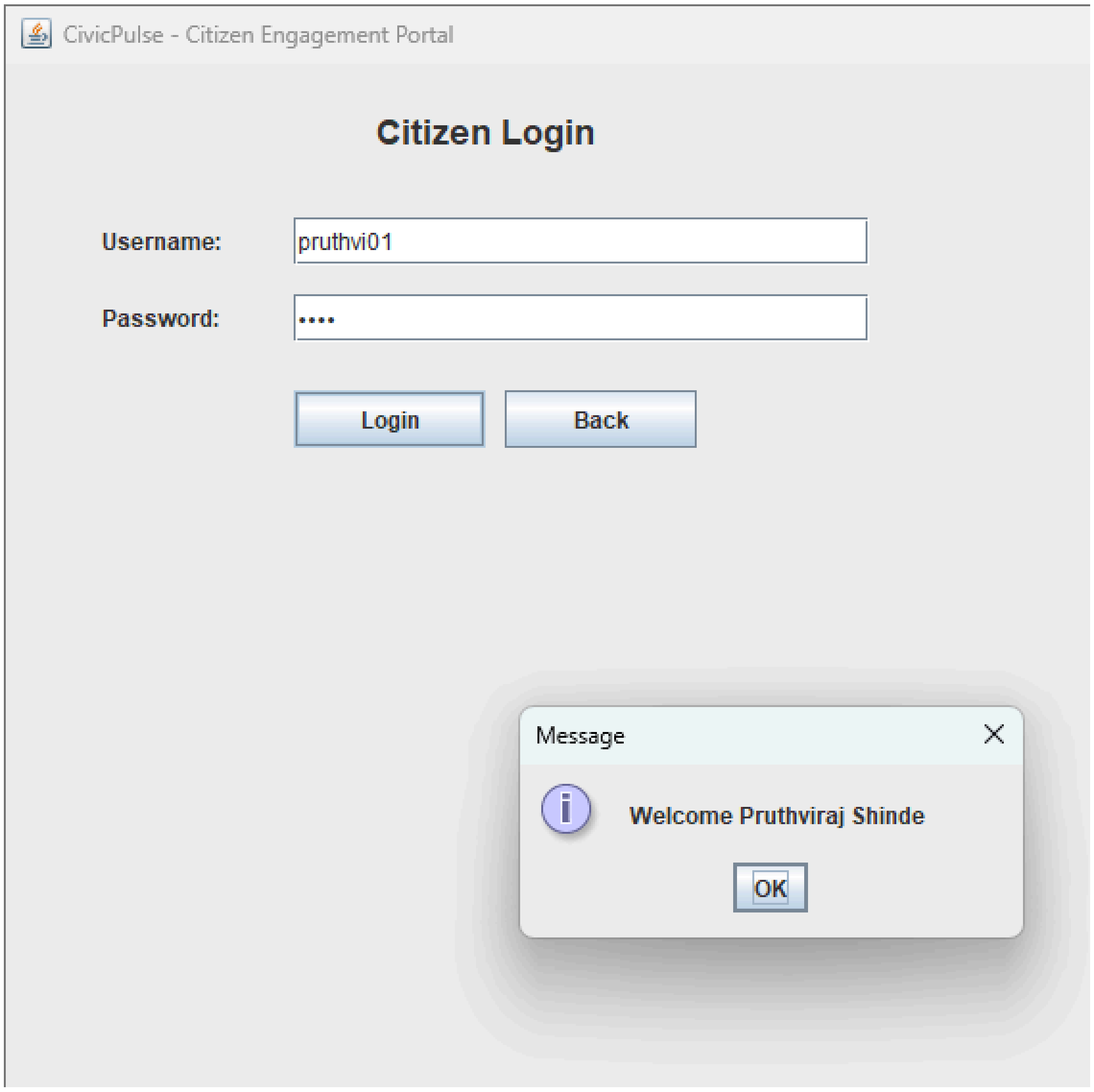

CivicPulse - Citizen Engagement Portal

Citizen Registration

Name:	Pruthviraj Shinde
Gender:	Male
Age:	23
Email:	pruthvi.raj@gmail.com
Aadhar No.:	567834567809
Mobile No.:	9765437689
State:	Maharastra
District:	Nagpur
Local Area:	Saheed Road
Username:	pruthvi01
Password:	••••

Message

Registration Successful for Pruthviraj Shinde



CivicPulse - Citizen Engagement Portal

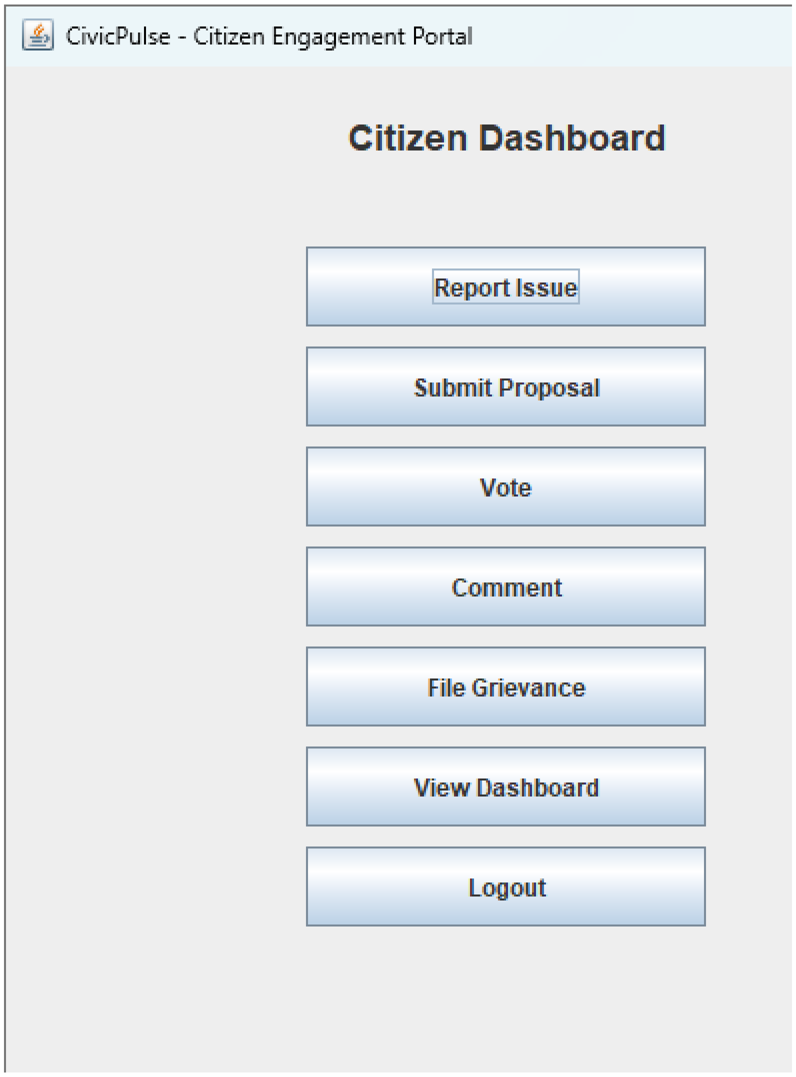
Citizen Login

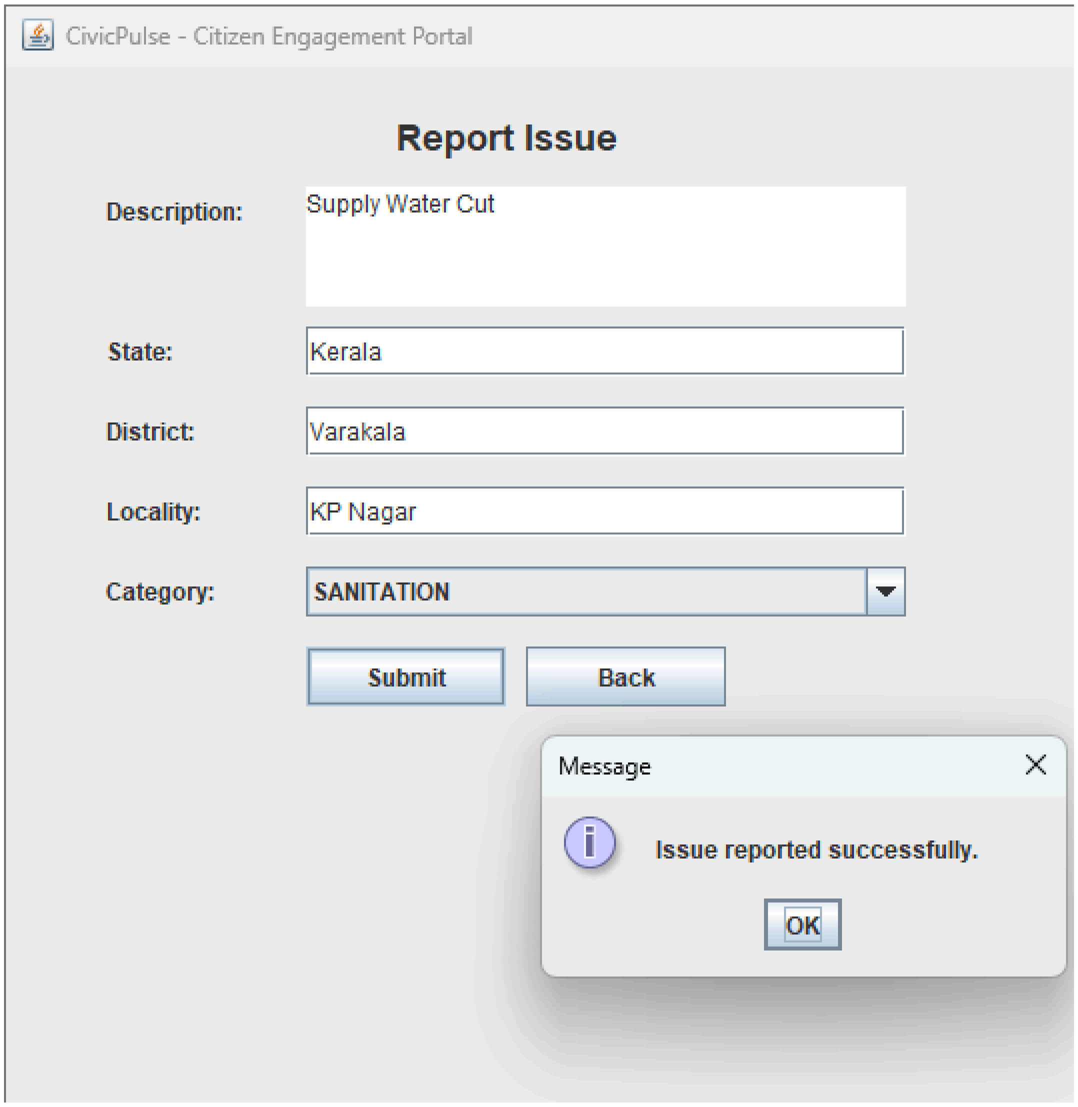
Username: pruthvi01

Password: ••••

Message

Welcome Pruthviraj Shinde


CivicPulse - Citizen Engagement Portal
Citizen Dashboard
Report Issue
Submit Proposal
Vote
Comment
File Grievance
View Dashboard
Logout

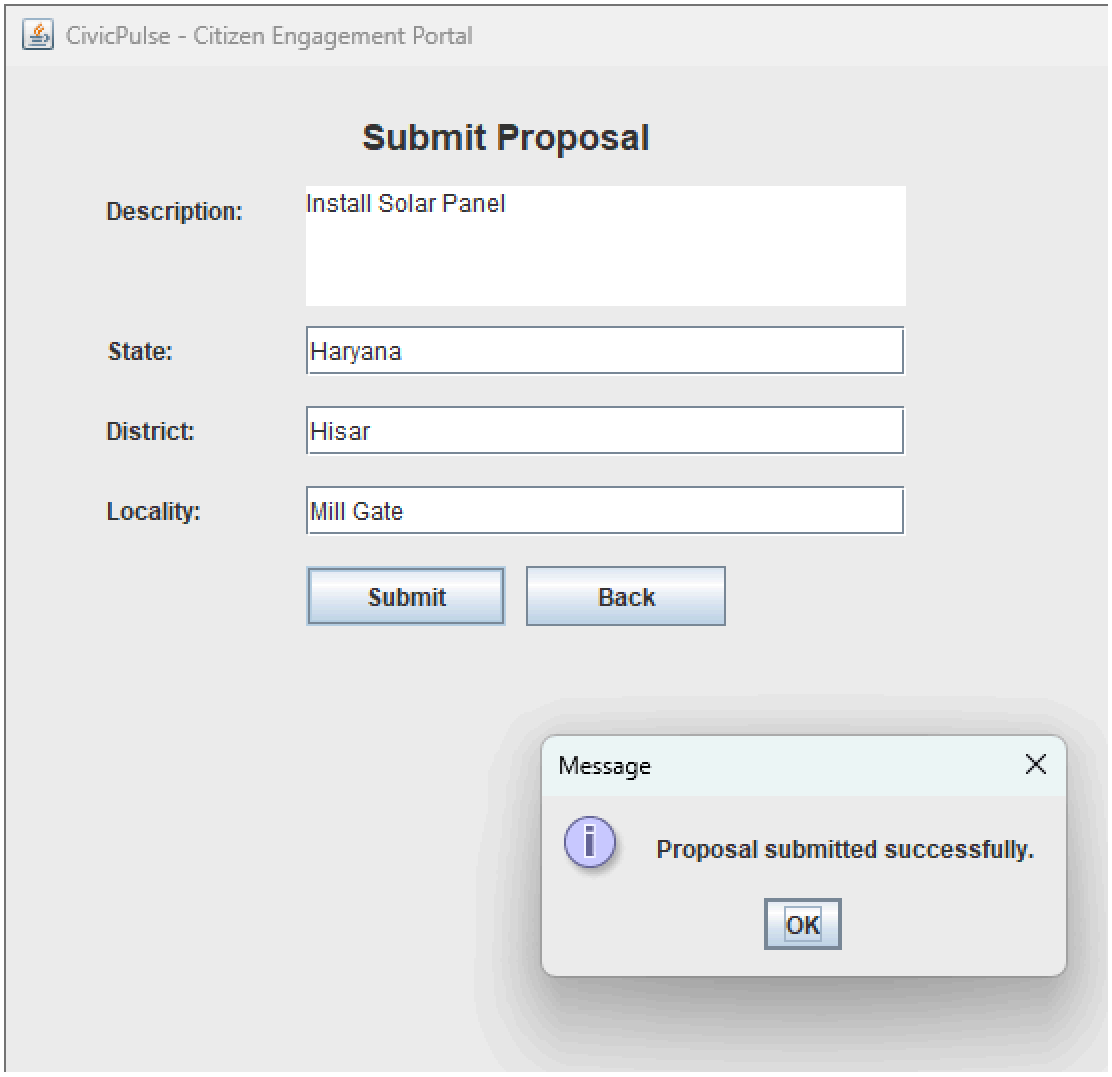

CivicPulse - Citizen Engagement Portal

Report Issue

Description:	Supply Water Cut
State:	Kerala
District:	Varakala
Locality:	KP Nagar
Category:	SANITATION

Message

Issue reported successfully.


CivicPulse - Citizen Engagement Portal

Submit Proposal

Description: Install Solar Panel

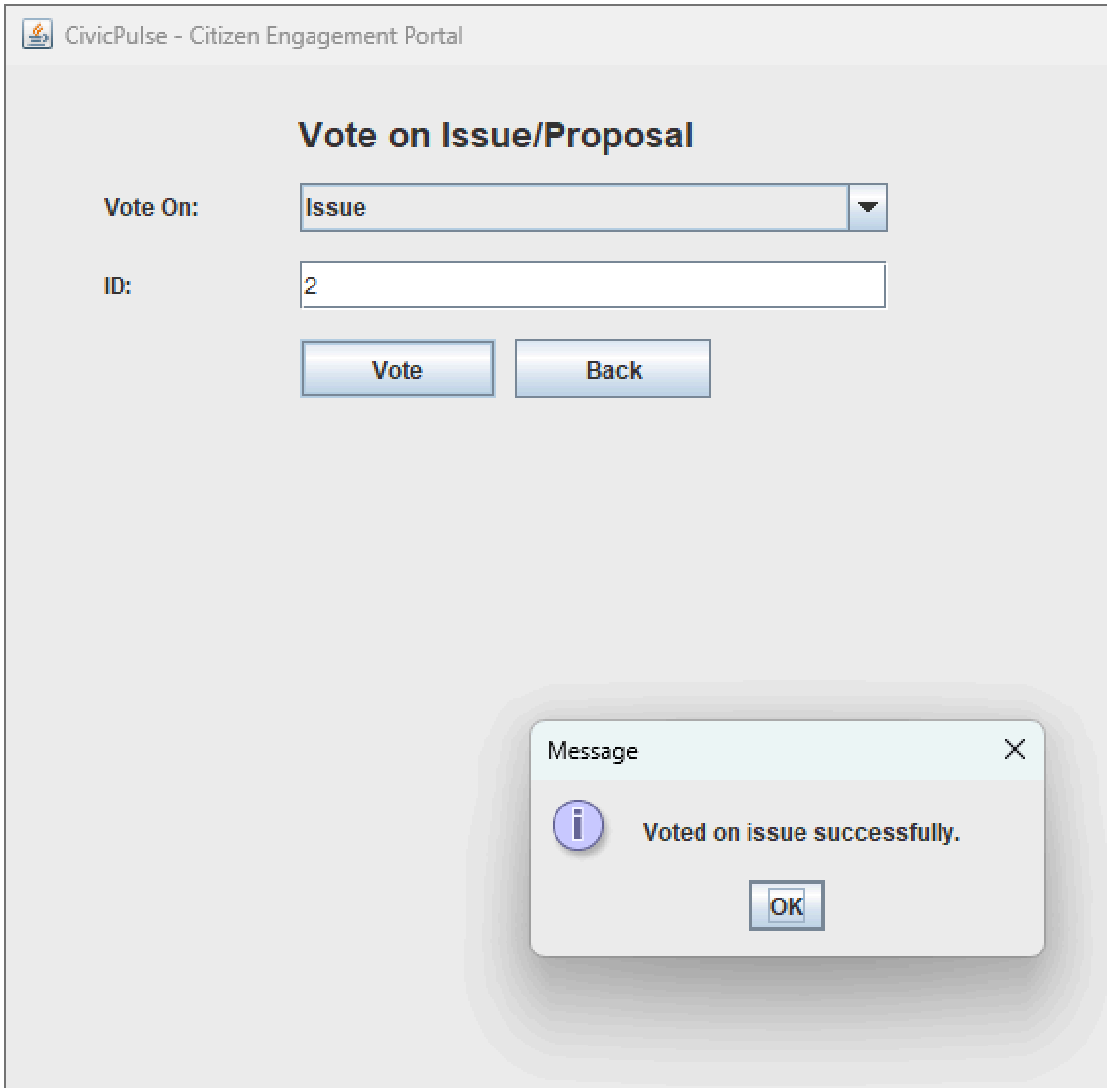
State: Haryana

District: Hisar

Locality: Mill Gate

Message

Proposal submitted successfully.

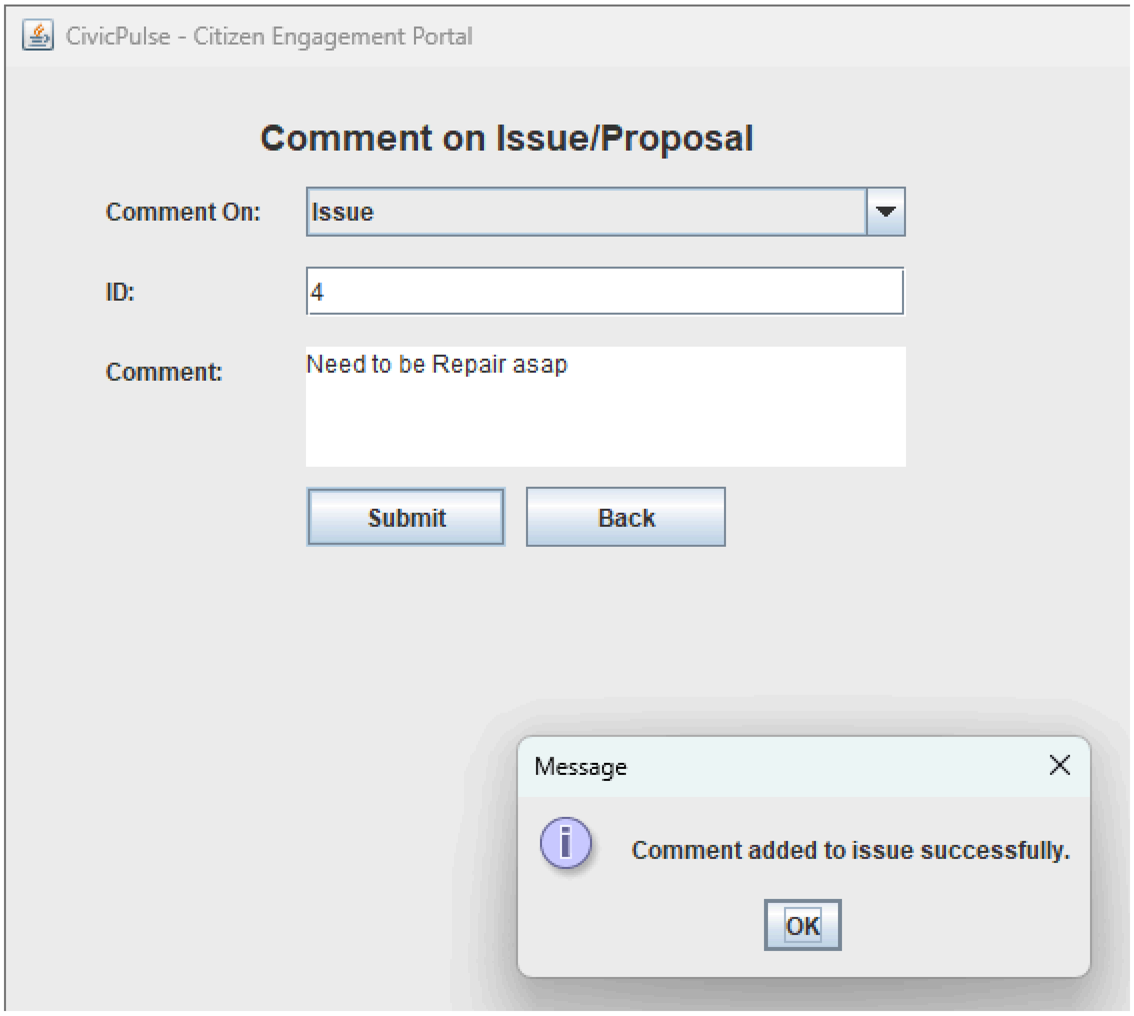

CivicPulse - Citizen Engagement Portal

Vote on Issue/Proposal

Vote On:	Issue
ID:	2

Message

Voted on issue successfully.

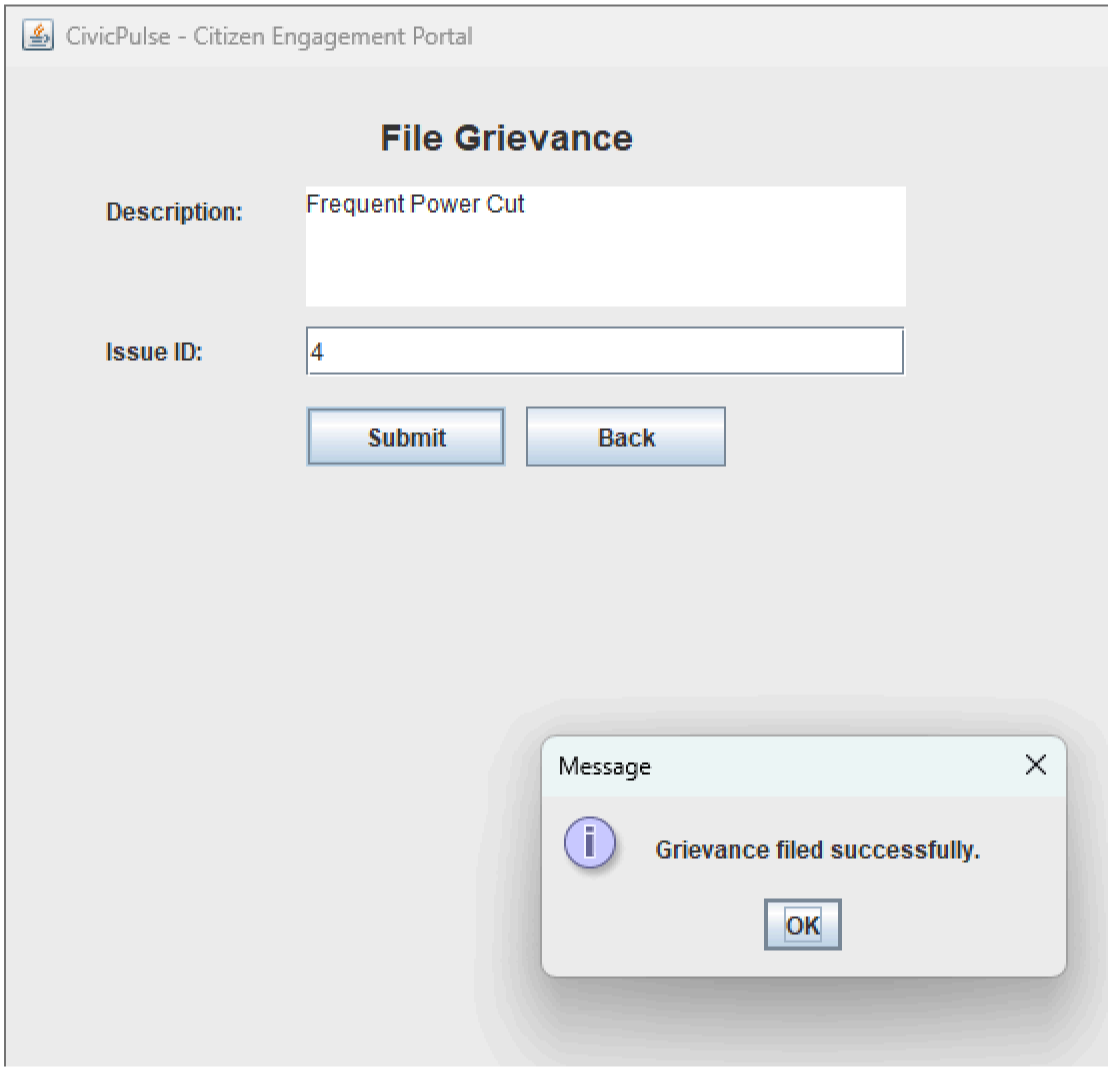

CivicPulse - Citizen Engagement Portal

Comment on Issue/Proposal

Comment On:	Issue
ID:	4
Comment:	Need to be Repair asap

Message

Comment added to issue successfully.



CivicPulse - Citizen Engagement Portal

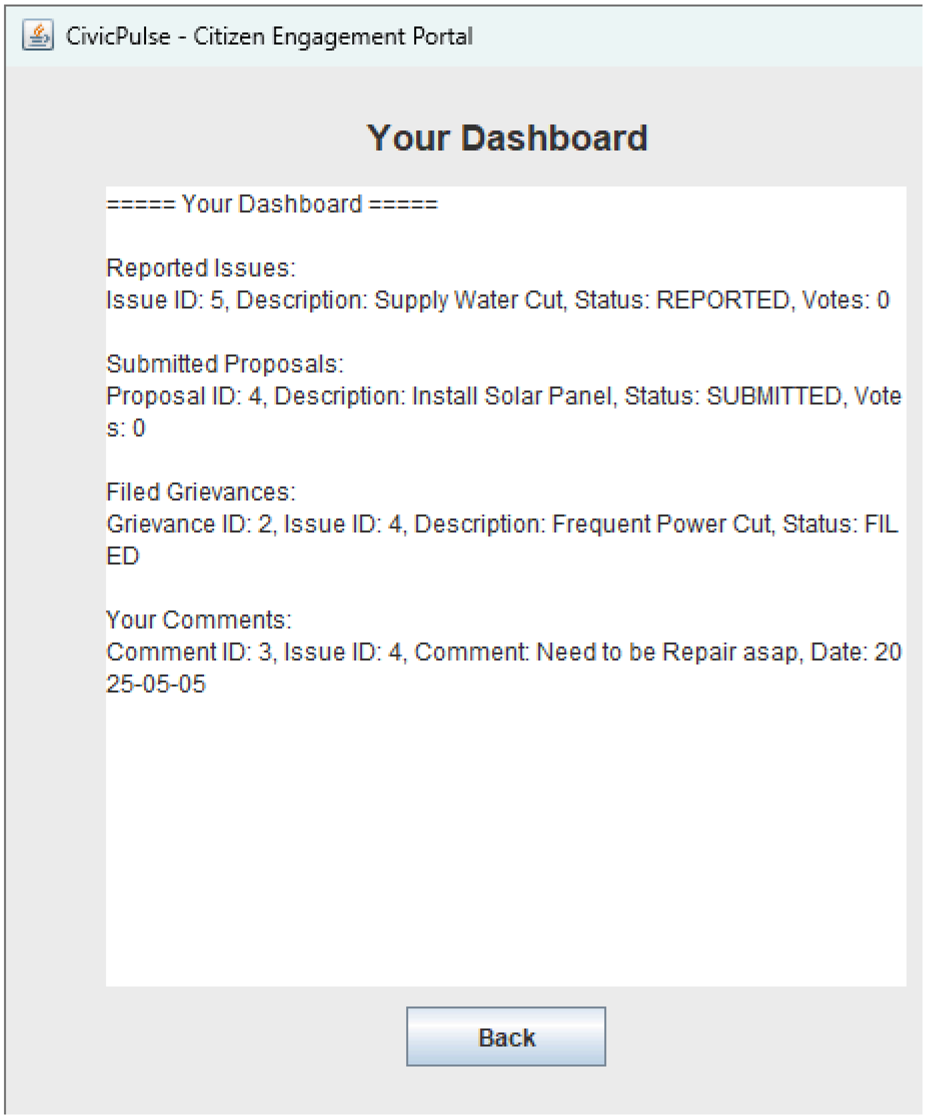
File Grievance

Description: Frequent Power Cut

Issue ID: 4

Message

Grievance filed successfully.



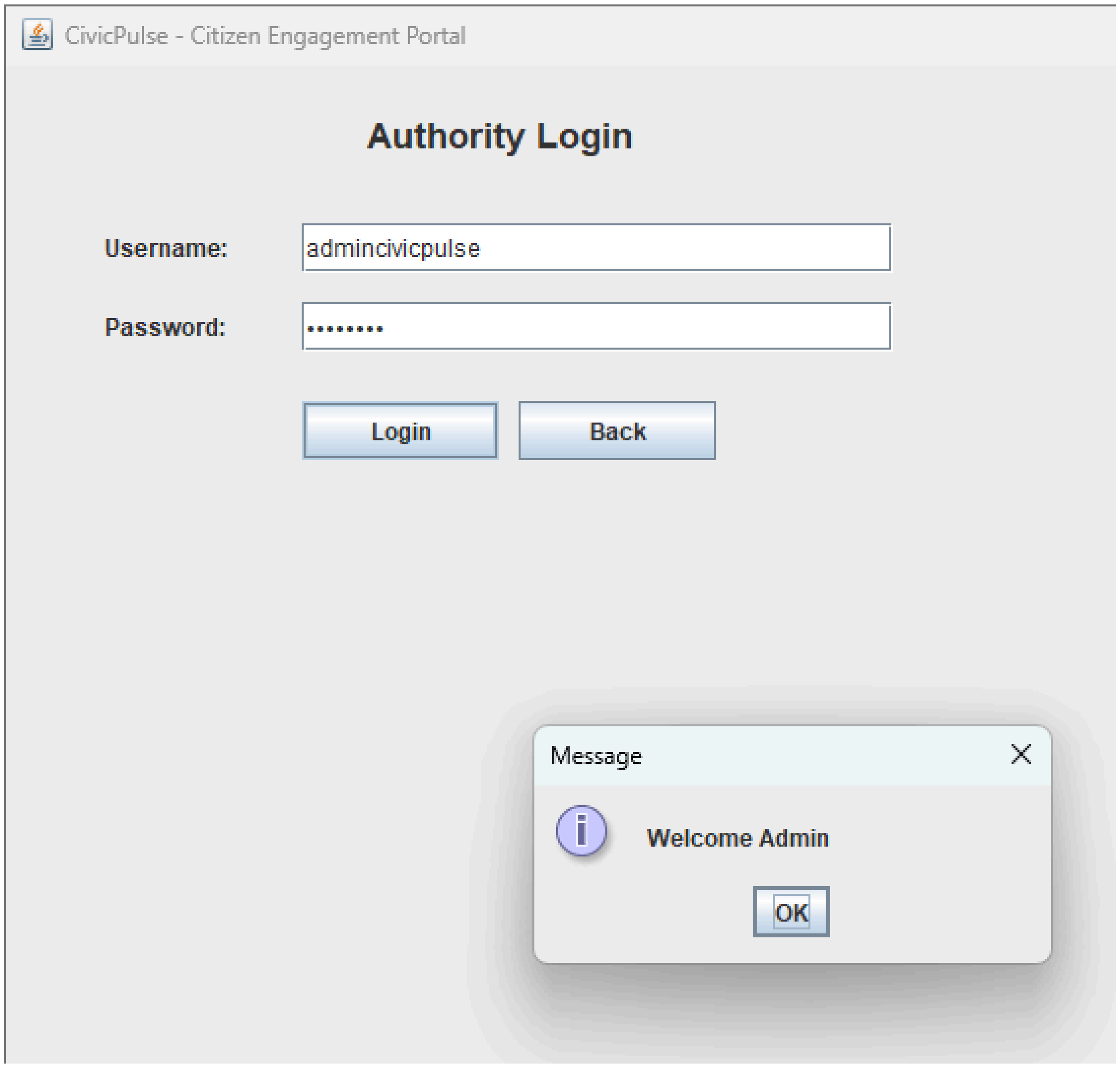
```
===== Your Dashboard =====

Reported Issues:
Issue ID: 5, Description: Supply Water Cut, Status: REPORTED, Votes: 0

Submitted Proposals:
Proposal ID: 4, Description: Install Solar Panel, Status: SUBMITTED, Vote
s: 0

Filed Grievances:
Grievance ID: 2, Issue ID: 4, Description: Frequent Power Cut, Status: FIL
ED

Your Comments:
Comment ID: 3, Issue ID: 4, Comment: Need to be Repair asap, Date: 20
25-05-05
```



CivicPulse - Citizen Engagement Portal

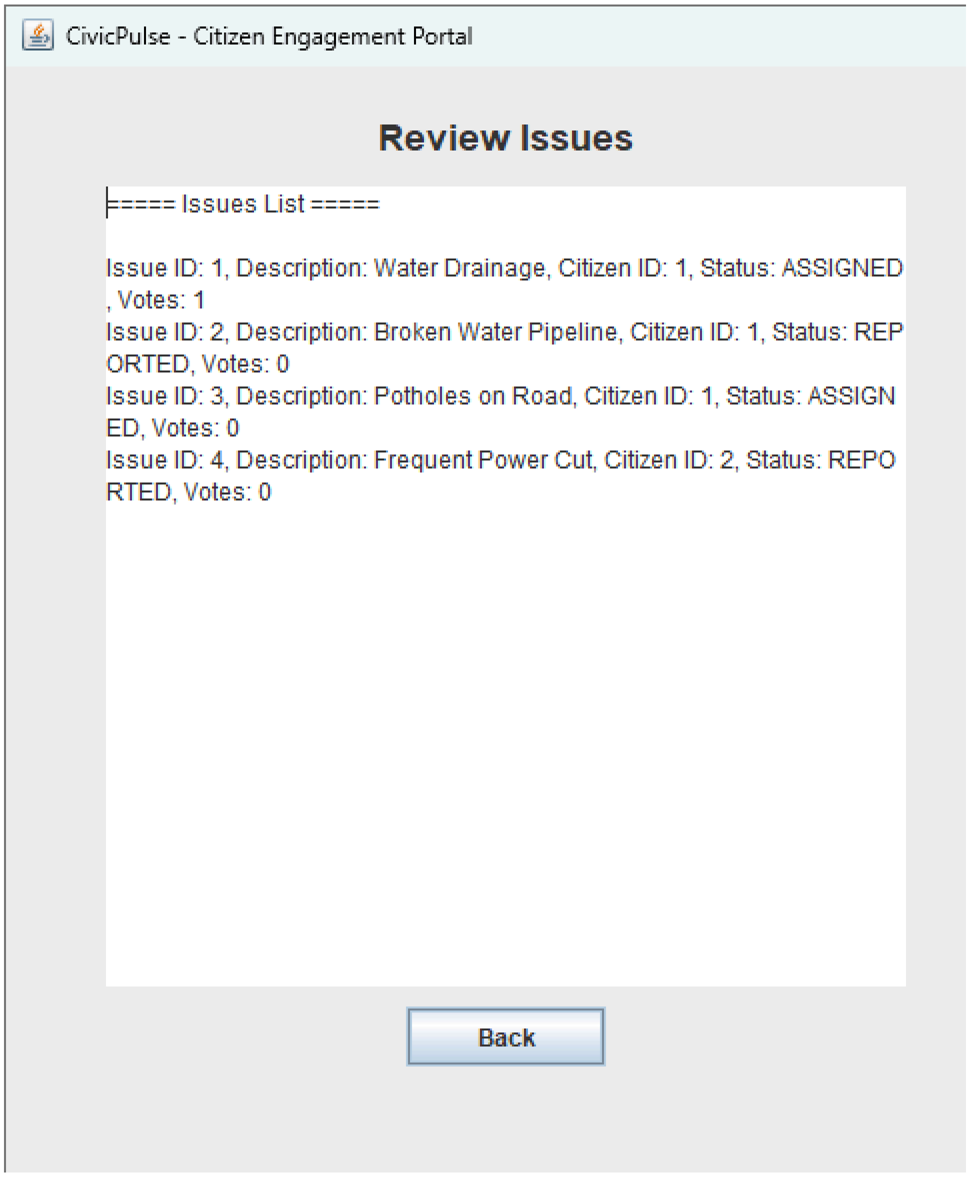
Authority Login

Username: admincivicpulse

Password: ••••••••

Message

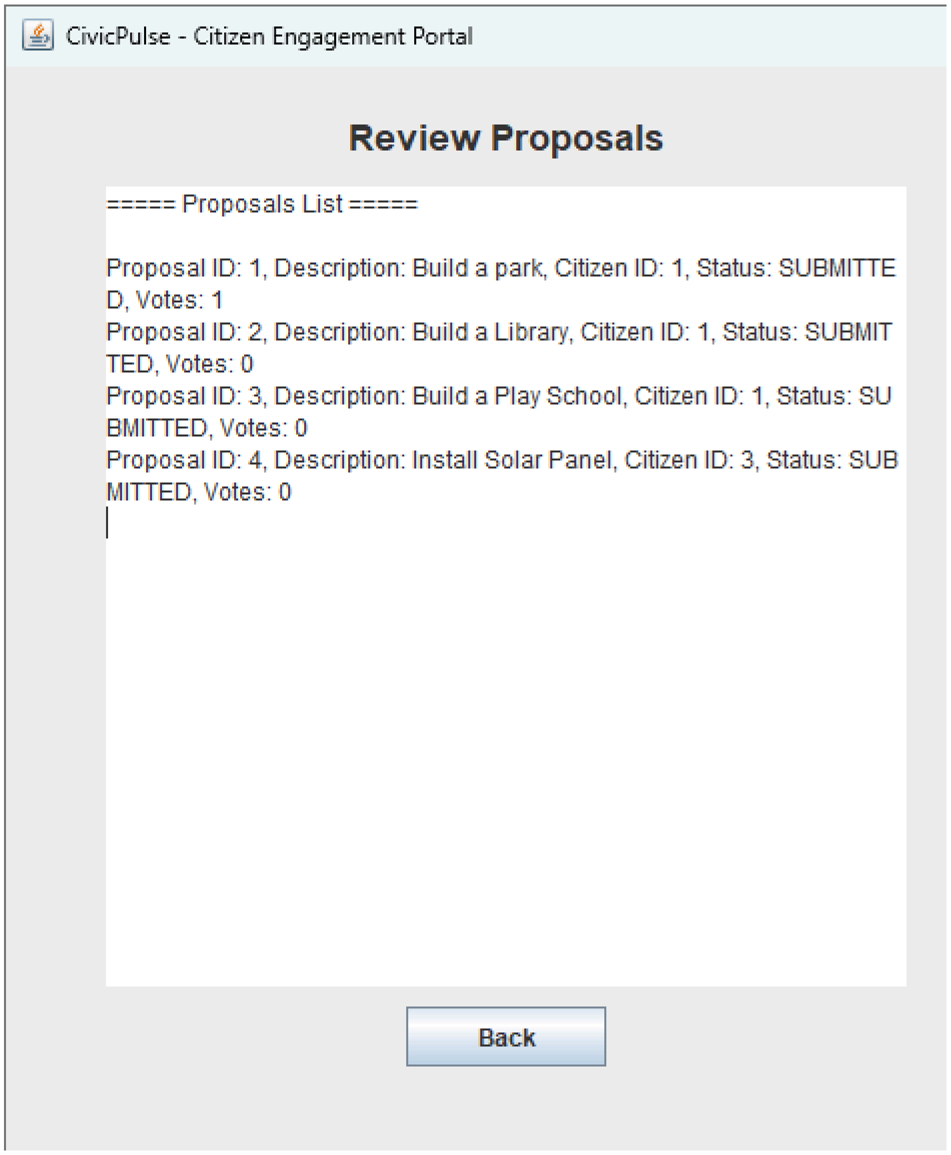
Welcome Admin



Review Issues

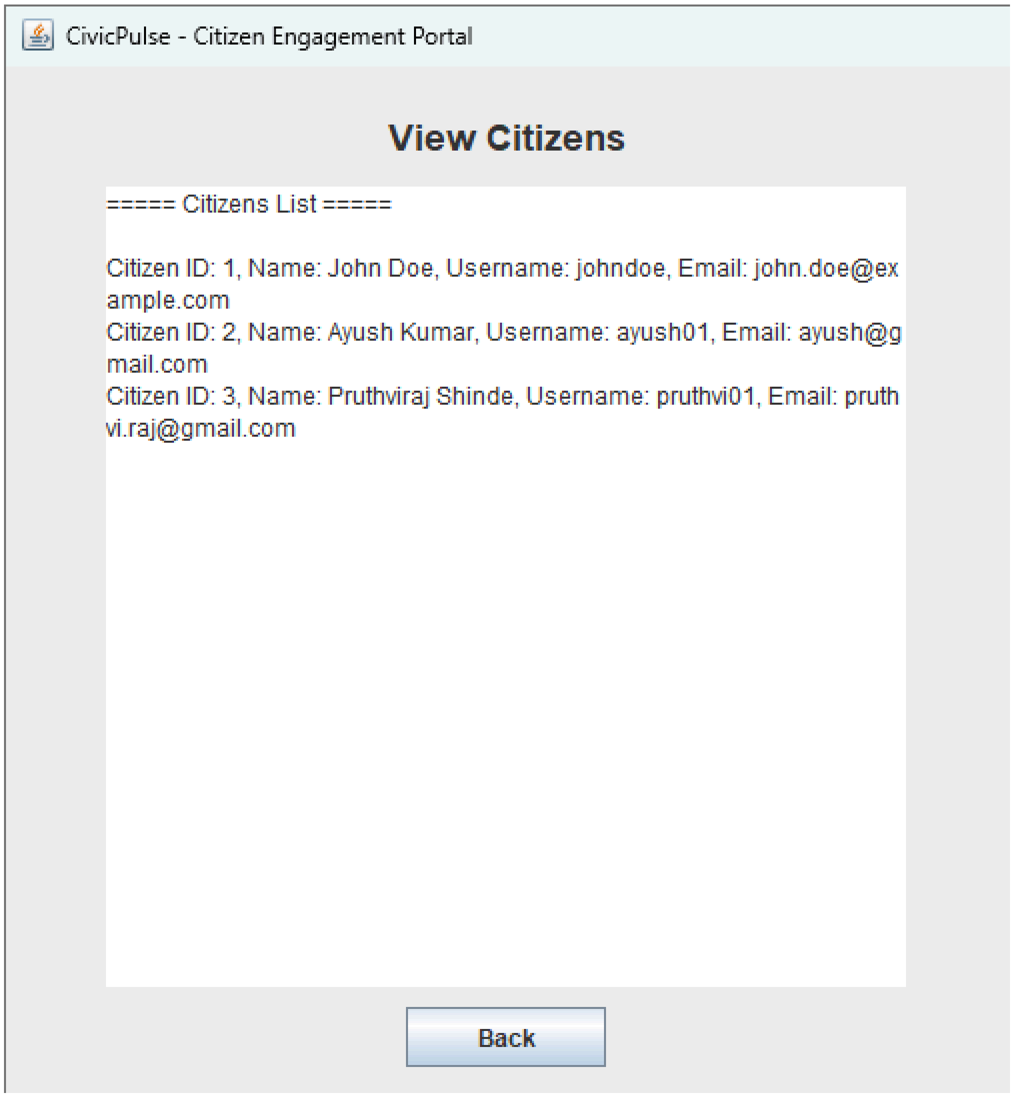
```
===== Issues List =====

Issue ID: 1, Description: Water Drainage, Citizen ID: 1, Status: ASSIGNED
, Votes: 1
Issue ID: 2, Description: Broken Water Pipeline, Citizen ID: 1, Status: REP
ORTED, Votes: 0
Issue ID: 3, Description: Potholes on Road, Citizen ID: 1, Status: ASSIGN
ED, Votes: 0
Issue ID: 4, Description: Frequent Power Cut, Citizen ID: 2, Status: REPO
RTED, Votes: 0
```



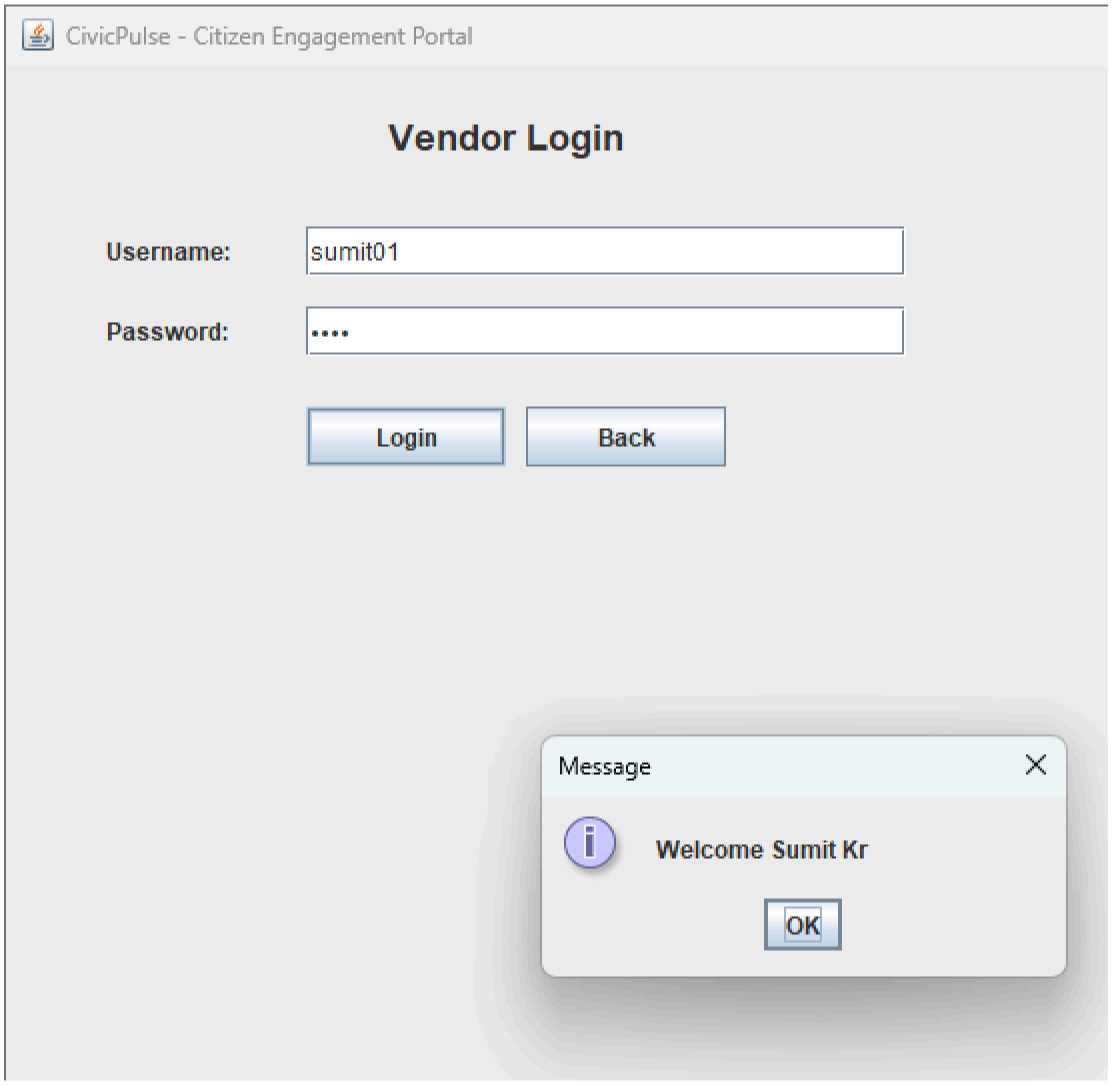
```
===== Proposals List =====

Proposal ID: 1, Description: Build a park, Citizen ID: 1, Status: SUBMITTE
D, Votes: 1
Proposal ID: 2, Description: Build a Library, Citizen ID: 1, Status: SUBMIT
TED, Votes: 0
Proposal ID: 3, Description: Build a Play School, Citizen ID: 1, Status: SU
BMITTED, Votes: 0
Proposal ID: 4, Description: Install Solar Panel, Citizen ID: 3, Status: SUB
MITTED, Votes: 0
```



```
===== Citizens List =====

Citizen ID: 1, Name: John Doe, Username: johndoe, Email: john.doe@ex
ample.com
Citizen ID: 2, Name: Ayush Kumar, Username: ayush01, Email: ayush@g
mail.com
Citizen ID: 3, Name: Pruthviraj Shinde, Username: pruthvi01, Email: pruth
vi.raj@gmail.com
```


CivicPulse - Citizen Engagement Portal

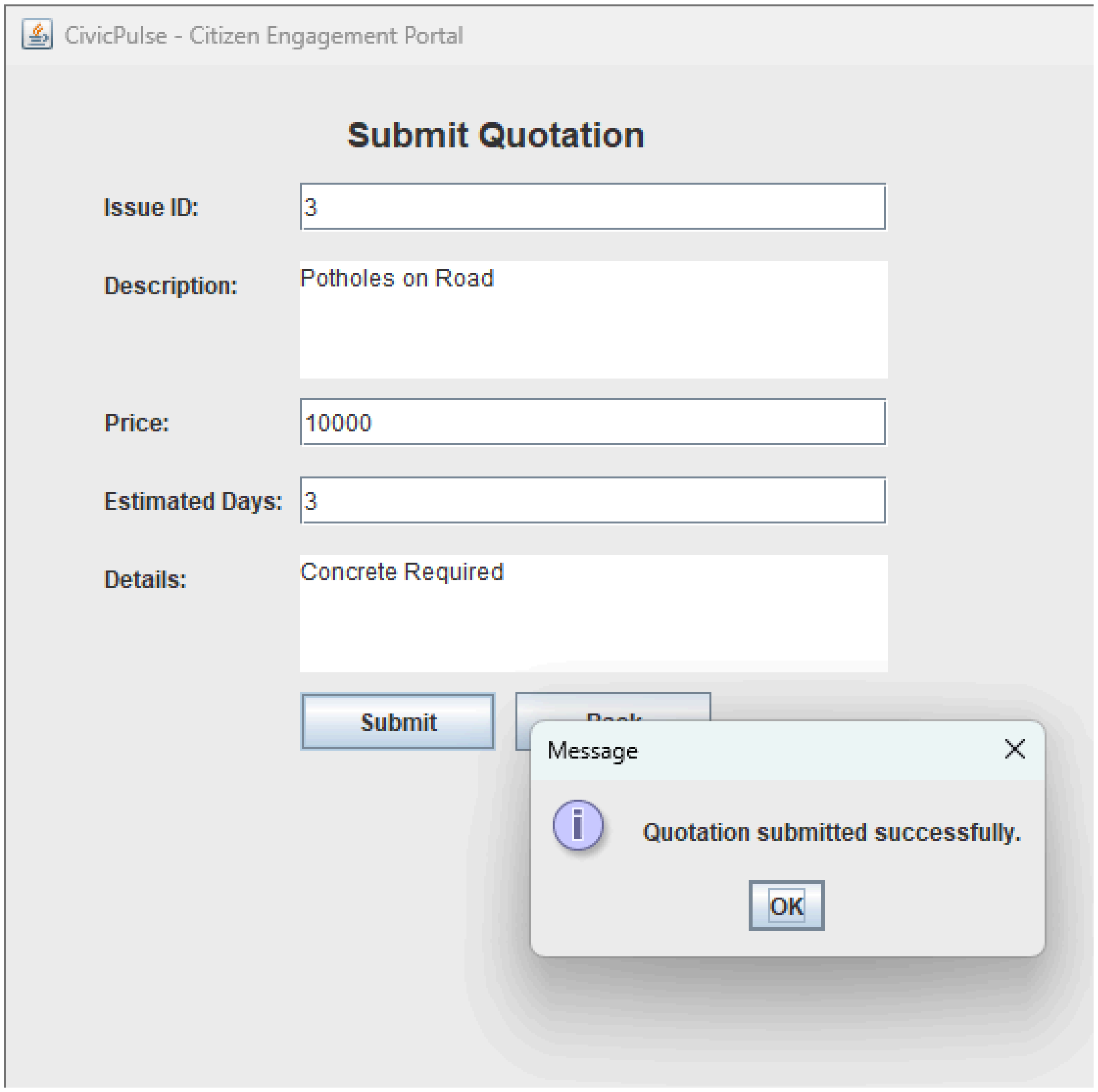
Vendor Login

Username: sumit01

Password: ••••

Message

Welcome Sumit Kr


CivicPulse - Citizen Engagement Portal

Submit Quotation

Issue ID:	3
Description:	Potholes on Road
Price:	10000
Estimated Days:	3
Details:	Concrete Required

Quotation submitted successfully.

```
CivicPulseApp
├── src
│   └── com
│       └── civicpulse
│           ├── core
│           │   ├── Category.java
│           │   └── Role.java
│           ├── user
│           │   └── model
│           │       ├── User.java
│           │       ├── Citizen.java
│           │       ├── Authority.java
│           │       └── VendorUser.java
│           ├── issue
│           │   └── model
│           │       ├── Issue.java
│           │       └── Comment.java
│           ├── proposal
│           │   └── model
│           │       ├── Proposal.java
│           ├── vendor
│           │   └── model
│           │       ├── Vendor.java
│           │       ├── Assignment.java
│           │       └── Quotation.java
│           ├── grievance
│           │   └── model
│           │       ├── Grievance.java
│           └── system
│               ├── CivicPulse.java
│               └── CivicPulseGUI.java
├── bin
│   └── com
│       └── civicpulse
│           ├── core
│           │   ├── Category.class
│           │   └── Role.class
│           ├── user
│           │   └── model
│           │       ├── User.class
│           │       ├── Citizen.class
│           │       ├── Authority.class
│           │       └── VendorUser.class
│           ├── issue
│         │   └── model
│         │       ├── Issue.class
│         │       └── Comment.class
│         ├── proposal
│         │   └── model
│         │       ├── Proposal.class
│         ├── vendor
│         │   └── model
│         │       ├── Vendor.class
│         │       ├── Assignment.class
│         │       └── Quotation.class
│         ├── grievance
│         │   └── model
│         │       ├── Grievance.class
│         └── system
│             ├── CivicPulse.class
│             └── CivicPulseGUI.class
├── lib
│   └── (empty or optional JDBC driver, e.g., mysql-connector-java.jar if using MySQL)
├── resources
│   ├── database
│   │   └── schema.sql
│   └── config
│       └── dbconfig.properties
├── docs
│   ├── design.md
│   └── user_manual.md
├── scripts
│   ├── compile.sh
│   └── run.sh
├── README.md

└── LICENSE
```

Detailed Breakdown of Directories and Files

1. **src Directory**
    - **Purpose**: Contains all Java source files, organized by package.
    - **Structure**: Mirrors the package hierarchy (com.civicpulse).
    - **Files**:
        - src/com/civicpulse/core/Category.java: Enum for categories (e.g., WATER, ELECTRICITY).

        - src/com/civicpulse/core/Role.java: Enum for user roles (e.g., CITIZEN, AUTHORITY, VENDOR).
        - src/com/civicpulse/user/model/User.java: Abstract base class for users.
        - src/com/civicpulse/user/model/Citizen.java: Model for citizen users.
        - src/com/civicpulse/user/model/Authority.java: Model for authority users.
        - src/com/civicpulse/user/model/VendorUser.java: Model for vendor users.
        - src/com/civicpulse/issue/model/Issue.java: Model for issues.
        - src/com/civicpulse/issue/model/Comment.java: Model for comments on issues or proposals.
        - src/com/civicpulse/proposal/model/Proposal.java: Model for proposals.
        - src/com/civicpulse/vendor/model/Vendor.java: Model for vendors.
        - src/com/civicpulse/vendor/model/Assignment.java: Model for vendor assignments.
        - src/com/civicpulse/vendor/model/Quotation.java: Model for quotations.
        - src/com/civicpulse/grievance/model/Grievance.java: Model for grievances.
        - src/com/civicpulse/system/CivicPulse.java: Main application logic (e.g., database connection).
        - src/com/civicpulse/system/CivicPulseGUI.java: GUI implementation (already provided).
2. **bin Directory**
    - **Purpose**: Contains compiled .class files, generated after compiling the source files.
    - **Structure**: Mirrors the src directory structure.
    - **Files**: Each .java file in src has a corresponding .class file in bin after compilation (e.g., Category.class, CivicPulseGUI.class).
3. **lib Directory**
    - **Purpose**: Stores external libraries (e.g., JDBC drivers).
    - **Files**:
        - (Optional) mysql-connector-java.jar: If using MySQL, this would be the JDBC driver. For this project, it’s assumed to use a database like MySQL, but the driver isn’t strictly required if using an in-memory database like H2 for testing.
4. **resources Directory**
    - **Purpose**: Contains non-code resources like database scripts and configuration files.
    - **Subdirectories**:
        - resources/database/
            - schema.sql: SQL script to create the database tables (citizens, authorities, vendors, issues, proposals, grievances, assignments, quotations, comments).
        - resources/config/
            - dbconfig.properties: Configuration file for database connection (e.g., URL, username, password).

5. **docs Directory**
    - **Purpose**: Contains documentation for the project.
    - **Files**:
        - design.md: High-level design document outlining the architecture.
        - user_manual.md: Instructions for using the application.
6. **scripts Directory**
    - **Purpose**: Contains shell scripts for compiling and running the project (assuming a simple setup without build tools).
    - **Files**:
        - compile.sh: Script to compile all Java files.
        - run.sh: Script to run the application.
7. **README.md**
    - **Purpose**: Provides an overview of the project, setup instructions, and usage.
8. **LICENSE**
    - **Purpose**: Specifies the licensing terms (e.g., MIT License).

Database :-

```
-- Create the civicpulse_db database
CREATE DATABASE IF NOT EXISTS civicpulse_db;
USE civicpulse_db;

-- Drop existing tables to ensure a clean setup (optional, comment out if you want to preserve
data)


-- Table for citizens
CREATE TABLE citizens (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   gender VARCHAR(10),
   age INT,
   email VARCHAR(100),
   aadhar_no VARCHAR(12),
   mobile_no VARCHAR(15),
   state VARCHAR(100),
   district VARCHAR(100),
   local_area VARCHAR(100),
   username VARCHAR(50) NOT NULL UNIQUE,
   password VARCHAR(50) NOT NULL
);

-- Table for authorities
```

```
CREATE TABLE authorities (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(50) NOT NULL
);

-- Table for vendors
CREATE TABLE vendors (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100),
    mobile_no VARCHAR(15),
    expertise VARCHAR(50),
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(50) NOT NULL
);

-- Table for issues
CREATE TABLE issues (
    issue_id INT AUTO_INCREMENT PRIMARY KEY,
    description TEXT NOT NULL,
    state VARCHAR(100) NOT NULL,
    district VARCHAR(100) NOT NULL,
    locality VARCHAR(100),
    citizen_id INT NOT NULL,
    category VARCHAR(50) NOT NULL,
    status VARCHAR(50) NOT NULL,
    votes INT DEFAULT 0,
    reported_date DATE NOT NULL,
    FOREIGN KEY (citizen_id) REFERENCES citizens(id)
);

-- Table for proposals
CREATE TABLE proposals (
    proposal_id INT AUTO_INCREMENT PRIMARY KEY,
    description TEXT NOT NULL,
    state VARCHAR(100) NOT NULL,
    district VARCHAR(100) NOT NULL,
    locality VARCHAR(100),
    citizen_id INT NOT NULL,
    status VARCHAR(50) NOT NULL,
    votes INT DEFAULT 0,
    submitted_date DATE NOT NULL,
    FOREIGN KEY (citizen_id) REFERENCES citizens(id)
);

-- Table for grievances
CREATE TABLE grievances (
    grievance_id INT AUTO_INCREMENT PRIMARY KEY,
    description TEXT NOT NULL,
    issue_id INT NOT NULL,
    citizen_id INT NOT NULL,
    filed_date DATE NOT NULL,
    status VARCHAR(50) NOT NULL,
    FOREIGN KEY (issue_id) REFERENCES issues(issue_id),
    FOREIGN KEY (citizen_id) REFERENCES citizens(id)
);

-- Table for assignments
CREATE TABLE assignments (
    assignment_id VARCHAR(50) PRIMARY KEY,
    issue_id INT NOT NULL,
    vendor_id INT NOT NULL,
    assigned_date DATE NOT NULL,
    status VARCHAR(50) NOT NULL,
    FOREIGN KEY (issue_id) REFERENCES issues(issue_id),
    FOREIGN KEY (vendor_id) REFERENCES vendors(id)
);

-- Table for quotations
CREATE TABLE quotations (
    quotation_id INT AUTO_INCREMENT PRIMARY KEY,
    issue_id INT NOT NULL,
    vendor_id INT NOT NULL,
    description TEXT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    estimated_days INT NOT NULL,
    details TEXT NOT NULL,
    submission_date DATE NOT NULL,
    status VARCHAR(50) NOT NULL,
    FOREIGN KEY (issue_id) REFERENCES issues(issue_id),
    FOREIGN KEY (vendor_id) REFERENCES vendors(id)
);

-- Table for comments
CREATE TABLE comments (
    comment_id INT AUTO_INCREMENT PRIMARY KEY,
```

```sql
    citizen_id INT NOT NULL,
    issue_id INT,
    proposal_id INT,
    comment_text TEXT NOT NULL,
    comment_date DATE NOT NULL,
    FOREIGN KEY (citizen_id) REFERENCES citizens(id),
    FOREIGN KEY (issue_id) REFERENCES issues(issue_id),
    FOREIGN KEY (proposal_id) REFERENCES proposals(proposal_id),
    CONSTRAINT chk_comment_target CHECK (
        (issue_id IS NOT NULL AND proposal_id IS NULL) OR
        (issue_id IS NULL AND proposal_id IS NOT NULL)
    )
);

-- Insert a default authority for testing
INSERT INTO authorities (name, username, password) VALUES ('Admin', 'admincivicpulse',
'admin123');

-- Insert a default citizen for testing
INSERT INTO citizens (name, gender, age, email, aadhar_no, mobile_no, state, district,
local_area, username, password)
VALUES ('John Doe', 'Male', 30, 'john.doe@example.com', '123456789012', '9876543210',
'Rajasthan', 'Kota', 'Downtown', 'johndoe', 'pass123');

-- Insert a default vendor for testing
INSERT INTO vendors (name, email, mobile_no, expertise, username, password)
VALUES ('Vendor One', 'vendor1@example.com', '9123456789', 'CONSTRUCTION', 'vendor1',
'vendorpass');
USE civicpulse_db;

SELECT * FROM citizens;
SELECT * FROM authorities;
SELECT * FROM vendors;
SELECT * FROM issues;
SELECT * FROM proposals;
SELECT * FROM comments;
SELECT * FROM assignments;
SELECT * FROM quotations;  -- Fixed typo: quotationsvendors → quotations
SELECT * FROM grievances;

-- Exit MySQL–
```

```
package com.civicpulse.system;

import com.civicpulse.core.Category;
import com.civicpulse.core.Role;
import com.civicpulse.user.model.*;
import com.civicpulse.issue.model.Issue;
import com.civicpulse.proposal.model.Proposal;
import com.civicpulse.vendor.model.Vendor;
import com.civicpulse.assignment.model.Assignment;
import com.civicpulse.quotation.model.Quotation;
import com.civicpulse.grievance.model.Grievance;
import javax.swing.*;
import java.awt.*;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class CivicPulseGUI {
    private JFrame frame;
    private CardLayout cardLayout;
    private JPanel mainPanel;
    private CivicPulse app;
    private User currentUser;

    // Panels for CardLayout
    private static final String MAIN_MENU = "MainMenu";
    private static final String CITIZEN_LOGIN = "CitizenLogin";
    private static final String CITIZEN_REGISTER = "CitizenRegister";
    private static final String CITIZEN_LOGIN_FORM = "CitizenLoginForm";
    private static final String CITIZEN_DASHBOARD = "CitizenDashboard";
    private static final String AUTHORITY_LOGIN = "AuthorityLogin";
    private static final String AUTHORITY_DASHBOARD = "AuthorityDashboard";
    private static final String VENDOR_LOGIN = "VendorLogin";
    private static final String VENDOR_REGISTER = "VendorRegister";
    private static final String VENDOR_LOGIN_FORM = "VendorLoginForm";
    private static final String VENDOR_DASHBOARD = "VendorDashboard";
    private static final String REPORT_ISSUE = "ReportIssue";
    private static final String SUBMIT_PROPOSAL = "SubmitProposal";
    private static final String FILE_GRIEVANCE = "FileGrievance";
    private static final String ASSIGN_VENDOR = "AssignVendor";
    private static final String SUBMIT_QUOTATION = "SubmitQuotation";
    private static final String VOTE_PANEL = "VotePanel";
    private static final String COMMENT_PANEL = "CommentPanel";
    private static final String VIEW_DASHBOARD = "ViewDashboard";
    private static final String REVIEW_ISSUES = "ReviewIssues";
    private static final String REVIEW_PROPOSALS = "ReviewProposals";
    private static final String VIEW_CITIZENS = "ViewCitizens";
    private static final String VIEW_QUOTATIONS = "ViewQuotations";
    private static final String VIEW_ASSIGNMENTS = "ViewAssignments";

    public CivicPulseGUI(CivicPulse app) {
        this.app = app;
        initialize();
    }

    private void initialize() {
        frame = new JFrame("CivicPulse - Citizen Engagement Portal");
        frame.setSize(900, 700);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        cardLayout = new CardLayout();
        mainPanel = new JPanel(cardLayout);

        // Add all panels to the CardLayout
        mainPanel.add(createMainMenuPanel(), MAIN_MENU);
        mainPanel.add(createCitizenLoginPanel(), CITIZEN_LOGIN);
        mainPanel.add(createCitizenRegisterPanel(), CITIZEN_REGISTER);
        mainPanel.add(createCitizenLoginFormPanel(), CITIZEN_LOGIN_FORM);
        mainPanel.add(createCitizenDashboardPanel(), CITIZEN_DASHBOARD);
        mainPanel.add(createAuthorityLoginPanel(), AUTHORITY_LOGIN);
        mainPanel.add(createAuthorityDashboardPanel(), AUTHORITY_DASHBOARD);
        mainPanel.add(createVendorLoginPanel(), VENDOR_LOGIN);
        mainPanel.add(createVendorRegisterPanel(), VENDOR_REGISTER);
        mainPanel.add(createVendorLoginFormPanel(), VENDOR_LOGIN_FORM);
        mainPanel.add(createVendorDashboardPanel(), VENDOR_DASHBOARD);
        mainPanel.add(createReportIssuePanel(), REPORT_ISSUE);
        mainPanel.add(createSubmitProposalPanel(), SUBMIT_PROPOSAL);
        mainPanel.add(createFileGrievancePanel(), FILE_GRIEVANCE);
        mainPanel.add(createAssignVendorPanel(), ASSIGN_VENDOR);
        mainPanel.add(createSubmitQuotationPanel(), SUBMIT_QUOTATION);
        mainPanel.add(createVotePanel(), VOTE_PANEL);
        mainPanel.add(createCommentPanel(), COMMENT_PANEL);
        mainPanel.add(createViewDashboardPanel(), VIEW_DASHBOARD);
        mainPanel.add(createReviewIssuesPanel(), REVIEW_ISSUES);
        mainPanel.add(createReviewProposalsPanel(), REVIEW_PROPOSALS);
        mainPanel.add(createViewCitizensPanel(), VIEW_CITIZENS);
        mainPanel.add(createViewQuotationsPanel(), VIEW_QUOTATIONS);
        mainPanel.add(createViewAssignmentsPanel(), VIEW_ASSIGNMENTS);
```

```
        frame.add(mainPanel);
        frame.setVisible(true);
        cardLayout.show(mainPanel, MAIN_MENU);
    }

    private JPanel createMainMenuPanel() {
        JPanel panel = new JPanel(null);
        JLabel welcomeLabel = new JLabel("Welcome to CIVIC PULSE",
SwingConstants.CENTER);
        welcomeLabel.setBounds(50, 20, 800, 50);
        welcomeLabel.setFont(welcomeLabel.getFont().deriveFont(24.0f));
        panel.add(welcomeLabel);

        JLabel subtitleLabel = new JLabel("Citizen Voice, Govt. Action",
SwingConstants.CENTER);
        subtitleLabel.setBounds(50, 80, 800, 30);
        subtitleLabel.setFont(subtitleLabel.getFont().deriveFont(16.0f));
        panel.add(subtitleLabel);

        JLabel selectLabel = new JLabel("Please select the User type:",
SwingConstants.CENTER);
        selectLabel.setBounds(50, 120, 800, 30);
        selectLabel.setFont(selectLabel.getFont().deriveFont(14.0f));
        panel.add(selectLabel);

        JButton citizenButton = new JButton("Citizen");
        citizenButton.setBounds(350, 160, 150, 40);
        citizenButton.setFont(citizenButton.getFont().deriveFont(14.0f));
        citizenButton.addActionListener(e -> cardLayout.show(mainPanel, CITIZEN_LOGIN));
        panel.add(citizenButton);

        JButton authorityButton = new JButton("Authority");
        authorityButton.setBounds(350, 210, 150, 40);
        authorityButton.setFont(authorityButton.getFont().deriveFont(14.0f));
        authorityButton.addActionListener(e -> cardLayout.show(mainPanel,
AUTHORITY_LOGIN));
        panel.add(authorityButton);

        JButton vendorButton = new JButton("Vendor");
        vendorButton.setBounds(350, 260, 150, 40);
        vendorButton.setFont(vendorButton.getFont().deriveFont(14.0f));
        vendorButton.addActionListener(e -> cardLayout.show(mainPanel, VENDOR_LOGIN));
        panel.add(vendorButton);
        JButton exitButton = new JButton("Exit");
        exitButton.setBounds(350, 310, 150, 40);
        exitButton.setFont(exitButton.getFont().deriveFont(14.0f));
        exitButton.addActionListener(e -> {
            app.closeDatabaseConnection();
            frame.dispose();
        });
        panel.add(exitButton);

        return panel;
    }

    private JPanel createCitizenLoginPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Citizen Portal", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel subtitleLabel = new JLabel("Please register or login to access the portal",
SwingConstants.CENTER);
        subtitleLabel.setBounds(50, 80, 800, 30);
        subtitleLabel.setFont(subtitleLabel.getFont().deriveFont(16.0f));
        panel.add(subtitleLabel);

        JButton registerButton = new JButton("Register");
        registerButton.setBounds(350, 120, 150, 40);
        registerButton.setFont(registerButton.getFont().deriveFont(14.0f));
        registerButton.addActionListener(e -> cardLayout.show(mainPanel,
CITIZEN_REGISTER));
        panel.add(registerButton);

        JButton loginButton = new JButton("Login");
        loginButton.setBounds(350, 170, 150, 40);
        loginButton.setFont(loginButton.getFont().deriveFont(14.0f));
        loginButton.addActionListener(e -> cardLayout.show(mainPanel,
CITIZEN_LOGIN_FORM));
        panel.add(loginButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 220, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, MAIN_MENU));
        panel.add(backButton);

        return panel;
    }

    private JPanel createCitizenRegisterPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Citizen Registration", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel nameLabel = new JLabel("Name:");
        nameLabel.setBounds(50, 80, 150, 30);
        JTextField nameField = new JTextField();
        nameField.setBounds(200, 80, 600, 30);
        panel.add(nameLabel);
        panel.add(nameField);

        JLabel genderLabel = new JLabel("Gender:");
        genderLabel.setBounds(50, 120, 150, 30);
        JTextField genderField = new JTextField();
        genderField.setBounds(200, 120, 600, 30);
        panel.add(genderLabel);
        panel.add(genderField);

        JLabel ageLabel = new JLabel("Age:");
        ageLabel.setBounds(50, 160, 150, 30);
        JTextField ageField = new JTextField();
        ageField.setBounds(200, 160, 600, 30);
        panel.add(ageLabel);
        panel.add(ageField);

        JLabel emailLabel = new JLabel("Email:");
        emailLabel.setBounds(50, 200, 150, 30);
        JTextField emailField = new JTextField();
        emailField.setBounds(200, 200, 600, 30);
        panel.add(emailLabel);
        panel.add(emailField);

        JLabel aadharLabel = new JLabel("Aadhar No.:");
        aadharLabel.setBounds(50, 240, 150, 30);
        JTextField aadharField = new JTextField();
        aadharField.setBounds(200, 240, 600, 30);
```

```
panel.add(aadharLabel);
panel.add(aadharField);

JLabel mobileLabel = new JLabel("Mobile No.:");
mobileLabel.setBounds(50, 280, 150, 30);
JTextField mobileField = new JTextField();
mobileField.setBounds(200, 280, 600, 30);
panel.add(mobileLabel);
panel.add(mobileField);

JLabel stateLabel = new JLabel("State:");
stateLabel.setBounds(50, 320, 150, 30);
JTextField stateField = new JTextField();
stateField.setBounds(200, 320, 600, 30);
panel.add(stateLabel);
panel.add(stateField);

JLabel districtLabel = new JLabel("District:");
districtLabel.setBounds(50, 360, 150, 30);
JTextField districtField = new JTextField();
districtField.setBounds(200, 360, 600, 30);
panel.add(districtLabel);
panel.add(districtField);

JLabel localAreaLabel = new JLabel("Local Area:");
localAreaLabel.setBounds(50, 400, 150, 30);
JTextField localAreaField = new JTextField();
localAreaField.setBounds(200, 400, 600, 30);
panel.add(localAreaLabel);
panel.add(localAreaField);

JLabel usernameLabel = new JLabel("Username:");
usernameLabel.setBounds(50, 440, 150, 30);
JTextField usernameField = new JTextField();
usernameField.setBounds(200, 440, 600, 30);
panel.add(usernameLabel);
panel.add(usernameField);

JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setBounds(50, 480, 150, 30);
JPasswordField passwordField = new JPasswordField();
passwordField.setBounds(200, 480, 600, 30);
panel.add(passwordLabel);
panel.add(passwordField);
```

```
        JButton submitButton = new JButton("Submit");
        submitButton.setBounds(300, 520, 150, 40);
        submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
        submitButton.addActionListener(e -> {
            String name = nameField.getText().trim();
            String gender = genderField.getText().trim();
            String ageText = ageField.getText().trim();
            String email = emailField.getText().trim();
            String aadharNo = aadharField.getText().trim();
            String mobileNo = mobileField.getText().trim();
            String state = stateField.getText().trim();
            String district = districtField.getText().trim();
            String localArea = localAreaField.getText().trim();
            String username = usernameField.getText().trim();
            String password = new String(passwordField.getPassword()).trim();

            if (name.isEmpty() || username.isEmpty() || password.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Name, Username, and Password are
required.");
                return;
            }

            int age;
            try {
                age = Integer.parseInt(ageText);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Age must be a valid number.");
                return;
            }

            registerCitizen(name, gender, age, email, aadharNo, mobileNo, state, district, localArea,
username, password);
            cardLayout.show(mainPanel, CITIZEN_LOGIN);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 520, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, CITIZEN_LOGIN));
        panel.add(backButton);

        return panel;
```

```
    }

    private void registerCitizen(String name, String gender, int age, String email, String
aadharNo, String mobileNo,
                                String state, String district, String localArea, String username, String
password) {
        String query = "INSERT INTO citizens (name, gender, age, email, aadhar_no, mobile_no,
state, district, local_area, username, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query,
PreparedStatement.RETURN_GENERATED_KEYS)) {
            stmt.setString(1, name);
            stmt.setString(2, gender);
            stmt.setInt(3, age);
            stmt.setString(4, email);
            stmt.setString(5, aadharNo);
            stmt.setString(6, mobileNo);
            stmt.setString(7, state);
            stmt.setString(8, district);
            stmt.setString(9, localArea);
            stmt.setString(10, username);
            stmt.setString(11, password);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into citizens: " + rowsAffected);

            ResultSet rs = stmt.getGeneratedKeys();
            int id = -1;
            if (rs.next()) {
                id = rs.getInt(1);
            }
            currentUser = new Citizen(id, name, Role.CITIZEN, username, password, gender, age,
email, aadharNo, mobileNo, state, district, localArea);
            JOptionPane.showMessageDialog(frame, "Registration Successful for " + name);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error registering citizen: " + e.getMessage());
        }
    }

    private JPanel createCitizenLoginFormPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Citizen Login", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);
        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setBounds(50, 80, 150, 30);
        JTextField usernameField = new JTextField();
        usernameField.setBounds(200, 80, 600, 30);
        panel.add(usernameLabel);
        panel.add(usernameField);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(50, 120, 150, 30);
        JPasswordField passwordField = new JPasswordField();
        passwordField.setBounds(200, 120, 600, 30);
        panel.add(passwordLabel);
        panel.add(passwordField);

        JButton loginButton = new JButton("Login");
        loginButton.setBounds(300, 160, 150, 40);
        loginButton.setFont(loginButton.getFont().deriveFont(14.0f));
        loginButton.addActionListener(e -> {
            String username = usernameField.getText().trim();
            String password = new String(passwordField.getPassword()).trim();
            if (username.isEmpty() || password.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Username and Password are required.");
                return;
            }
            loginCitizen(username, password);
        });
        panel.add(loginButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 160, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, CITIZEN_LOGIN));
        panel.add(backButton);

        return panel;
    }

    private void loginCitizen(String username, String password) {
        String query = "SELECT * FROM citizens WHERE username = ? AND password = ?";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setString(1, username);
            stmt.setString(2, password);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
```

```
                int id = rs.getInt("id");
                String name = rs.getString("name");
                String gender = rs.getString("gender");
                int age = rs.getInt("age");
                String email = rs.getString("email");
                String aadharNo = rs.getString("aadhar_no");
                String mobileNo = rs.getString("mobile_no");
                String state = rs.getString("state");
                String district = rs.getString("district");
                String localArea = rs.getString("local_area");
                currentUser = new Citizen(id, name, Role.CITIZEN, username, password, gender,
age, email, aadharNo, mobileNo, state, district, localArea);
                JOptionPane.showMessageDialog(frame, "Welcome " + name);
                cardLayout.show(mainPanel, CITIZEN_DASHBOARD);
            } else {
                JOptionPane.showMessageDialog(frame, "Authentication failed.");
            }
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(frame, "Error during login: " + e.getMessage());
    }
}

private JPanel createCitizenDashboardPanel() {
    JPanel panel = new JPanel(null);
    JLabel titleLabel = new JLabel("Citizen Dashboard", SwingConstants.CENTER);
    titleLabel.setBounds(50, 20, 800, 50);
    titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
    panel.add(titleLabel);

    JLabel welcomeLabel = new JLabel("", SwingConstants.CENTER);
    welcomeLabel.setBounds(50, 80, 800, 30);
    if (currentUser != null) {
        welcomeLabel.setText("Welcome " + currentUser.getName());
    }
    panel.add(welcomeLabel);

    JButton reportIssueButton = new JButton("Report Issue");
    reportIssueButton.setBounds(350, 120, 150, 40);
    reportIssueButton.setFont(reportIssueButton.getFont().deriveFont(14.0f));
    reportIssueButton.addActionListener(e -> cardLayout.show(mainPanel,
REPORT_ISSUE));
    panel.add(reportIssueButton);
```

```
        JButton submitProposalButton = new JButton("Submit Proposal");
        submitProposalButton.setBounds(350, 170, 150, 40);
        submitProposalButton.setFont(submitProposalButton.getFont().deriveFont(14.0f));
        submitProposalButton.addActionListener(e -> cardLayout.show(mainPanel,
SUBMIT_PROPOSAL));
        panel.add(submitProposalButton);

        JButton voteButton = new JButton("Vote");
        voteButton.setBounds(350, 220, 150, 40);
        voteButton.setFont(voteButton.getFont().deriveFont(14.0f));
        voteButton.addActionListener(e -> cardLayout.show(mainPanel, VOTE_PANEL));
        panel.add(voteButton);

        JButton commentButton = new JButton("Comment");
        commentButton.setBounds(350, 270, 150, 40);
        commentButton.setFont(commentButton.getFont().deriveFont(14.0f));
        commentButton.addActionListener(e -> cardLayout.show(mainPanel,
COMMENT_PANEL));
        panel.add(commentButton);

        JButton fileGrievanceButton = new JButton("File Grievance");
        fileGrievanceButton.setBounds(350, 320, 150, 40);
        fileGrievanceButton.setFont(fileGrievanceButton.getFont().deriveFont(14.0f));
        fileGrievanceButton.addActionListener(e -> cardLayout.show(mainPanel,
FILE_GRIEVANCE));
        panel.add(fileGrievanceButton);

        JButton viewDashboardButton = new JButton("View Dashboard");
        viewDashboardButton.setBounds(350, 370, 150, 40);
        viewDashboardButton.setFont(viewDashboardButton.getFont().deriveFont(14.0f));
        viewDashboardButton.addActionListener(e -> {
            updateViewDashboardPanel();
            cardLayout.show(mainPanel, VIEW_DASHBOARD);
        });
        panel.add(viewDashboardButton);

        JButton logoutButton = new JButton("Logout");
        logoutButton.setBounds(350, 420, 150, 40);
        logoutButton.setFont(logoutButton.getFont().deriveFont(14.0f));
        logoutButton.addActionListener(e -> {
            currentUser = null;
            cardLayout.show(mainPanel, MAIN_MENU);
        });
        panel.add(logoutButton);
        return panel;
    }

    private JPanel createAuthorityLoginPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Authority Login", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setBounds(50, 80, 150, 30);
        JTextField usernameField = new JTextField();
        usernameField.setBounds(200, 80, 600, 30);
        panel.add(usernameLabel);
        panel.add(usernameField);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(50, 120, 150, 30);
        JPasswordField passwordField = new JPasswordField();
        passwordField.setBounds(200, 120, 600, 30);
        panel.add(passwordLabel);
        panel.add(passwordField);

        JButton loginButton = new JButton("Login");
        loginButton.setBounds(300, 160, 150, 40);
        loginButton.setFont(loginButton.getFont().deriveFont(14.0f));
        loginButton.addActionListener(e -> {
            String username = usernameField.getText().trim();
            String password = new String(passwordField.getPassword()).trim();
            if (username.isEmpty() || password.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Username and Password are required.");
                return;
            }
            loginAuthority(username, password);
        });
        panel.add(loginButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 160, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, MAIN_MENU));
        panel.add(backButton);
        return panel;
    }

    private void loginAuthority(String username, String password) {
        String query = "SELECT * FROM authorities WHERE username = ? AND password = ?";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setString(1, username);
            stmt.setString(2, password);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    int id = rs.getInt("id");
                    String name = rs.getString("name");
                    currentUser = new Authority(id, name, Role.AUTHORITY, username, password);
                    JOptionPane.showMessageDialog(frame, "Welcome " + name);
                    cardLayout.show(mainPanel, AUTHORITY_DASHBOARD);
                } else {
                    JOptionPane.showMessageDialog(frame, "Authentication failed.");
                }
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error during login: " + e.getMessage());
        }
    }

    private JPanel createAuthorityDashboardPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Authority Dashboard", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel welcomeLabel = new JLabel("", SwingConstants.CENTER);
        welcomeLabel.setBounds(50, 80, 800, 30);
        if (currentUser != null) {
            welcomeLabel.setText("Welcome " + currentUser.getName());
        }
        panel.add(welcomeLabel);

        JButton assignVendorButton = new JButton("Assign Vendor");
        assignVendorButton.setBounds(350, 120, 150, 40);
        assignVendorButton.setFont(assignVendorButton.getFont().deriveFont(14.0f));
        assignVendorButton.addActionListener(e -> cardLayout.show(mainPanel,
ASSIGN_VENDOR));
```

```
    panel.add(assignVendorButton);

    JButton reviewIssuesButton = new JButton("Review Issues");
    reviewIssuesButton.setBounds(350, 170, 150, 40);
    reviewIssuesButton.setFont(reviewIssuesButton.getFont().deriveFont(14.0f));
    reviewIssuesButton.addActionListener(e -> {
        updateReviewIssuesPanel();
        cardLayout.show(mainPanel, REVIEW_ISSUES);
    });
    panel.add(reviewIssuesButton);

    JButton reviewProposalsButton = new JButton("Review Proposals");
    reviewProposalsButton.setBounds(350, 220, 150, 40);
    reviewProposalsButton.setFont(reviewProposalsButton.getFont().deriveFont(14.0f));
    reviewProposalsButton.addActionListener(e -> {
        updateReviewProposalsPanel();
        cardLayout.show(mainPanel, REVIEW_PROPOSALS);
    });
    panel.add(reviewProposalsButton);

    JButton viewCitizensButton = new JButton("View Citizens");
    viewCitizensButton.setBounds(350, 270, 150, 40);
    viewCitizensButton.setFont(viewCitizensButton.getFont().deriveFont(14.0f));
    viewCitizensButton.addActionListener(e -> {
        updateViewCitizensPanel();
        cardLayout.show(mainPanel, VIEW_CITIZENS);
    });
    panel.add(viewCitizensButton);

    JButton logoutButton = new JButton("Logout");
    logoutButton.setBounds(350, 320, 150, 40);
    logoutButton.setFont(logoutButton.getFont().deriveFont(14.0f));
    logoutButton.addActionListener(e -> {
        currentUser = null;
        cardLayout.show(mainPanel, MAIN_MENU);
    });
    panel.add(logoutButton);

    return panel;
}

private JPanel createVendorLoginPanel() {
    JPanel panel = new JPanel(null);
    JLabel titleLabel = new JLabel("Vendor Portal", SwingConstants.CENTER);
```

```
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel subtitleLabel = new JLabel("Please register or login to access the portal",
SwingConstants.CENTER);
        subtitleLabel.setBounds(50, 80, 800, 30);
        subtitleLabel.setFont(subtitleLabel.getFont().deriveFont(16.0f));
        panel.add(subtitleLabel);

        JButton registerButton = new JButton("Register");
        registerButton.setBounds(350, 120, 150, 40);
        registerButton.setFont(registerButton.getFont().deriveFont(14.0f));
        registerButton.addActionListener(e -> cardLayout.show(mainPanel,
VENDOR_REGISTER));
        panel.add(registerButton);

        JButton loginButton = new JButton("Login");
        loginButton.setBounds(350, 170, 150, 40);
        loginButton.setFont(loginButton.getFont().deriveFont(14.0f));
        loginButton.addActionListener(e -> cardLayout.show(mainPanel,
VENDOR_LOGIN_FORM));
        panel.add(loginButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 220, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, MAIN_MENU));
        panel.add(backButton);

        return panel;
    }

    private JPanel createVendorRegisterPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Vendor Registration", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel nameLabel = new JLabel("Name:");
        nameLabel.setBounds(50, 80, 150, 30);
        JTextField nameField = new JTextField();
        nameField.setBounds(200, 80, 600, 30);
```

```
panel.add(nameLabel);
panel.add(nameField);

JLabel emailLabel = new JLabel("Email:");
emailLabel.setBounds(50, 120, 150, 30);
JTextField emailField = new JTextField();
emailField.setBounds(200, 120, 600, 30);
panel.add(emailLabel);
panel.add(emailField);

JLabel mobileLabel = new JLabel("Mobile No.:");
mobileLabel.setBounds(50, 160, 150, 30);
JTextField mobileField = new JTextField();
mobileField.setBounds(200, 160, 600, 30);
panel.add(mobileLabel);
panel.add(mobileField);

JLabel expertiseLabel = new JLabel("Expertise:");
expertiseLabel.setBounds(50, 200, 150, 30);
JTextField expertiseField = new JTextField();
expertiseField.setBounds(200, 200, 600, 30);
panel.add(expertiseLabel);
panel.add(expertiseField);

JLabel usernameLabel = new JLabel("Username:");
usernameLabel.setBounds(50, 240, 150, 30);
JTextField usernameField = new JTextField();
usernameField.setBounds(200, 240, 600, 30);
panel.add(usernameLabel);
panel.add(usernameField);

JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setBounds(50, 280, 150, 30);
JPasswordField passwordField = new JPasswordField();
passwordField.setBounds(200, 280, 600, 30);
panel.add(passwordLabel);
panel.add(passwordField);

JButton submitButton = new JButton("Submit");
submitButton.setBounds(300, 320, 150, 40);
submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
submitButton.addActionListener(e -> {
    String name = nameField.getText().trim();
    String email = emailField.getText().trim();
```

```
            String mobileNo = mobileField.getText().trim();
            String expertise = expertiseField.getText().trim();
            String username = usernameField.getText().trim();
            String password = new String(passwordField.getPassword()).trim();

            if (name.isEmpty() || username.isEmpty() || password.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Name, Username, and Password are
required.");
                return;
            }

            try {
                Category.valueOf(expertise.toUpperCase());
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(frame, "Expertise must be a valid category (e.g.,
WATER, ROADS).");
                return;
            }

            registerVendor(name, email, mobileNo, expertise, username, password);
            cardLayout.show(mainPanel, VENDOR_LOGIN);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 320, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, VENDOR_LOGIN));
        panel.add(backButton);

        return panel;
    }

    private void registerVendor(String name, String email, String mobileNo, String expertise,
String username, String password) {
        String query = "INSERT INTO vendors (name, email, mobile_no, expertise, username,
password) VALUES (?, ?, ?, ?, ?, ?)";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query,
PreparedStatement.RETURN_GENERATED_KEYS)) {
            stmt.setString(1, name);
            stmt.setString(2, email);
            stmt.setString(3, mobileNo);
            stmt.setString(4, expertise);
            stmt.setString(5, username);
            stmt.setString(6, password);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into vendors: " + rowsAffected);

            ResultSet rs = stmt.getGeneratedKeys();
            int id = -1;
            if (rs.next()) {
                id = rs.getInt(1);
            }
            Vendor vendor = new Vendor(String.valueOf(id), name,
Category.valueOf(expertise.toUpperCase()), mobileNo, 0);
            currentUser = new VendorUser(id, name, Role.VENDOR, username, password,
mobileNo, vendor);
            JOptionPane.showMessageDialog(frame, "Registration Successful for " + name);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error registering vendor: " + e.getMessage());
        }
    }

    private JPanel createVendorLoginFormPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Vendor Login", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setBounds(50, 80, 150, 30);
        JTextField usernameField = new JTextField();
        usernameField.setBounds(200, 80, 600, 30);
        panel.add(usernameLabel);
        panel.add(usernameField);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(50, 120, 150, 30);
        JPasswordField passwordField = new JPasswordField();
        passwordField.setBounds(200, 120, 600, 30);
        panel.add(passwordLabel);
        panel.add(passwordField);

        JButton loginButton = new JButton("Login");
        loginButton.setBounds(300, 160, 150, 40);
        loginButton.setFont(loginButton.getFont().deriveFont(14.0f));
        loginButton.addActionListener(e -> {
            String username = usernameField.getText().trim();
            String password = new String(passwordField.getPassword()).trim();
            if (username.isEmpty() || password.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Username and Password are required.");
                return;
            }
            loginVendor(username, password);
        });
        panel.add(loginButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 160, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, VENDOR_LOGIN));
        panel.add(backButton);

        return panel;
    }

    private void loginVendor(String username, String password) {
        String query = "SELECT * FROM vendors WHERE username = ? AND password = ?";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setString(1, username);
            stmt.setString(2, password);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    int id = rs.getInt("id");
                    String name = rs.getString("name");
                    String email = rs.getString("email");
                    String mobileNo = rs.getString("mobile_no");
                    String expertise = rs.getString("expertise");
                    try {
                        Category.valueOf(expertise.toUpperCase());
                    } catch (IllegalArgumentException ex) {
                        JOptionPane.showMessageDialog(frame, "Invalid expertise category in
database.");
                        return;
                    }
                    Vendor vendor = new Vendor(String.valueOf(id), name,
Category.valueOf(expertise.toUpperCase()), mobileNo, 0);
                    currentUser = new VendorUser(id, name, Role.VENDOR, username, password,
mobileNo, vendor);
                    JOptionPane.showMessageDialog(frame, "Welcome " + name);
                    cardLayout.show(mainPanel, VENDOR_DASHBOARD);
                } else {
                    JOptionPane.showMessageDialog(frame, "Authentication failed.");
                }
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error during login: " + e.getMessage());
        }
    }

    private JPanel createVendorDashboardPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Vendor Dashboard", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel welcomeLabel = new JLabel("", SwingConstants.CENTER);
        welcomeLabel.setBounds(50, 80, 800, 30);
        if (currentUser != null) {
            welcomeLabel.setText("Welcome " + currentUser.getName());
        }
        panel.add(welcomeLabel);

        JButton submitQuotationButton = new JButton("Submit Quotation");
        submitQuotationButton.setBounds(350, 120, 150, 40);
        submitQuotationButton.setFont(submitQuotationButton.getFont().deriveFont(14.0f));
        submitQuotationButton.addActionListener(e -> cardLayout.show(mainPanel,
SUBMIT_QUOTATION));
        panel.add(submitQuotationButton);

        JButton viewQuotationsButton = new JButton("View Quotations");
        viewQuotationsButton.setBounds(350, 170, 150, 40);
        viewQuotationsButton.setFont(viewQuotationsButton.getFont().deriveFont(14.0f));
        viewQuotationsButton.addActionListener(e -> {
            updateViewQuotationsPanel();
            cardLayout.show(mainPanel, VIEW_QUOTATIONS);
        });
        panel.add(viewQuotationsButton);

        JButton viewAssignmentsButton = new JButton("View Assignments");
        viewAssignmentsButton.setBounds(350, 220, 150, 40);
        viewAssignmentsButton.setFont(viewAssignmentsButton.getFont().deriveFont(14.0f));
        viewAssignmentsButton.addActionListener(e -> {
            updateViewAssignmentsPanel();
```

```
            cardLayout.show(mainPanel, VIEW_ASSIGNMENTS);
        });
        panel.add(viewAssignmentsButton);

        JButton logoutButton = new JButton("Logout");
        logoutButton.setBounds(350, 270, 150, 40);
        logoutButton.setFont(logoutButton.getFont().deriveFont(14.0f));
        logoutButton.addActionListener(e -> {
            currentUser = null;
            cardLayout.show(mainPanel, MAIN_MENU);
        });
        panel.add(logoutButton);

        return panel;
    }

    private JPanel createReportIssuePanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Report Issue", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel descriptionLabel = new JLabel("Description:");
        descriptionLabel.setBounds(50, 80, 150, 30);
        JTextArea descriptionArea = new JTextArea();
        descriptionArea.setBounds(200, 80, 600, 60);
        descriptionArea.setLineWrap(true);
        panel.add(descriptionLabel);
        panel.add(descriptionArea);

        JLabel stateLabel = new JLabel("State:");
        stateLabel.setBounds(50, 150, 150, 30);
        JTextField stateField = new JTextField();
        stateField.setBounds(200, 150, 600, 30);
        panel.add(stateLabel);
        panel.add(stateField);

        JLabel districtLabel = new JLabel("District:");
        districtLabel.setBounds(50, 190, 150, 30);
        JTextField districtField = new JTextField();
        districtField.setBounds(200, 190, 600, 30);
        panel.add(districtLabel);
        panel.add(districtField);
        JLabel localityLabel = new JLabel("Locality:");
        localityLabel.setBounds(50, 230, 150, 30);
        JTextField localityField = new JTextField();
        localityField.setBounds(200, 230, 600, 30);
        panel.add(localityLabel);
        panel.add(localityField);

        JLabel categoryLabel = new JLabel("Category:");
        categoryLabel.setBounds(50, 270, 150, 30);
        JComboBox<Category> categoryCombo = new JComboBox<>(Category.values());
        categoryCombo.setBounds(200, 270, 600, 30);
        panel.add(categoryLabel);
        panel.add(categoryCombo);

        JButton submitButton = new JButton("Submit");
        submitButton.setBounds(300, 310, 150, 40);
        submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
        submitButton.addActionListener(e -> {
            String description = descriptionArea.getText().trim();
            String state = stateField.getText().trim();
            String district = districtField.getText().trim();
            String locality = localityField.getText().trim();
            Category category = (Category) categoryCombo.getSelectedItem();
            if (description.isEmpty() || state.isEmpty() || district.isEmpty() || locality.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "All fields are required.");
                return;
            }
            reportIssue(description, state, district, locality, category);
            cardLayout.show(mainPanel, CITIZEN_DASHBOARD);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 310, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
CITIZEN_DASHBOARD));
        panel.add(backButton);

        return panel;
    }
    private void reportIssue(String description, String state, String district, String locality, Category
category) {
        String query = "INSERT INTO issues (description, state, district, locality, citizen_id,
category, status, votes, reported_date) VALUES (?, ?, ?, ?, ?, ?, ?, ?, CURDATE())";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setString(1, description);
            stmt.setString(2, state);
            stmt.setString(3, district);
            stmt.setString(4, locality);
            stmt.setInt(5, currentUser.getId());
            stmt.setString(6, category.toString());
            stmt.setString(7, "REPORTED");
            stmt.setInt(8, 0);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into issues: " + rowsAffected);
            JOptionPane.showMessageDialog(frame, "Issue reported successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error reporting issue: " + e.getMessage());
        }
    }

    private JPanel createSubmitProposalPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Submit Proposal", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel descriptionLabel = new JLabel("Description:");
        descriptionLabel.setBounds(50, 80, 150, 30);
        JTextArea descriptionArea = new JTextArea();
        descriptionArea.setBounds(200, 80, 600, 60);
        descriptionArea.setLineWrap(true);
        panel.add(descriptionLabel);
        panel.add(descriptionArea);

        JLabel stateLabel = new JLabel("State:");
        stateLabel.setBounds(50, 150, 150, 30);
        JTextField stateField = new JTextField();
        stateField.setBounds(200, 150, 600, 30);
        panel.add(stateLabel);
        panel.add(stateField);

        JLabel districtLabel = new JLabel("District:");
        districtLabel.setBounds(50, 190, 150, 30);
        JTextField districtField = new JTextField();
        districtField.setBounds(200, 190, 600, 30);
        panel.add(districtLabel);
        panel.add(districtField);

        JLabel localityLabel = new JLabel("Locality:");
        localityLabel.setBounds(50, 230, 150, 30);
        JTextField localityField = new JTextField();
        localityField.setBounds(200, 230, 600, 30);
        panel.add(localityLabel);
        panel.add(localityField);

        JButton submitButton = new JButton("Submit");
        submitButton.setBounds(300, 270, 150, 40);
        submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
        submitButton.addActionListener(e -> {
            String description = descriptionArea.getText().trim();
            String state = stateField.getText().trim();
            String district = districtField.getText().trim();
            String locality = localityField.getText().trim();
            if (description.isEmpty() || state.isEmpty() || district.isEmpty() || locality.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "All fields are required.");
                return;
            }
            submitProposal(description, state, district, locality);
            cardLayout.show(mainPanel, CITIZEN_DASHBOARD);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 270, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
CITIZEN_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void submitProposal(String description, String state, String district, String locality) {
        String query = "INSERT INTO proposals (description, state, district, locality, citizen_id,
status, votes, submitted_date) VALUES (?, ?, ?, ?, ?, ?, ?, CURDATE())";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setString(1, description);
            stmt.setString(2, state);
            stmt.setString(3, district);
            stmt.setString(4, locality);
            stmt.setInt(5, currentUser.getId());
            stmt.setString(6, "SUBMITTED");
            stmt.setInt(7, 0);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into proposals: " + rowsAffected);
            JOptionPane.showMessageDialog(frame, "Proposal submitted successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error submitting proposal: " +
e.getMessage());
        }
    }

    private JPanel createFileGrievancePanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("File Grievance", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel descriptionLabel = new JLabel("Description:");
        descriptionLabel.setBounds(50, 80, 150, 30);
        JTextArea descriptionArea = new JTextArea();
        descriptionArea.setBounds(200, 80, 600, 60);
        descriptionArea.setLineWrap(true);
        panel.add(descriptionLabel);
        panel.add(descriptionArea);

        JLabel issueIdLabel = new JLabel("Issue ID:");
        issueIdLabel.setBounds(50, 150, 150, 30);
        JTextField issueIdField = new JTextField();
        issueIdField.setBounds(200, 150, 600, 30);
        panel.add(issueIdLabel);
        panel.add(issueIdField);

        JButton submitButton = new JButton("Submit");
        submitButton.setBounds(300, 190, 150, 40);
        submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
        submitButton.addActionListener(e -> {
            String description = descriptionArea.getText().trim();
            String issueIdText = issueIdField.getText().trim();
            if (description.isEmpty() || issueIdText.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "All fields are required.");
                return;
            }
            int issueId;
            try {
                issueId = Integer.parseInt(issueIdText);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Issue ID must be a valid number.");
                return;
            }
            fileGrievance(description, issueId);
            cardLayout.show(mainPanel, CITIZEN_DASHBOARD);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 190, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
CITIZEN_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void fileGrievance(String description, int issueId) {
        String query = "INSERT INTO grievances (description, issue_id, citizen_id, filed_date,
status) VALUES (?, ?, ?, CURDATE(), ?)";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setString(1, description);
            stmt.setInt(2, issueId);
            stmt.setInt(3, currentUser.getId());
            stmt.setString(4, "FILED");
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into grievances: " + rowsAffected);
            JOptionPane.showMessageDialog(frame, "Grievance filed successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error filing grievance: " + e.getMessage());
        }
    }

    private JPanel createAssignVendorPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Assign Vendor", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel assignmentIdLabel = new JLabel("Assignment ID:");
        assignmentIdLabel.setBounds(50, 80, 150, 30);
        JTextField assignmentIdField = new JTextField();
        assignmentIdField.setBounds(200, 80, 600, 30);
        panel.add(assignmentIdLabel);
        panel.add(assignmentIdField);

        JLabel issueIdLabel = new JLabel("Issue ID:");
        issueIdLabel.setBounds(50, 120, 150, 30);
        JTextField issueIdField = new JTextField();
        issueIdField.setBounds(200, 120, 600, 30);
        panel.add(issueIdLabel);
        panel.add(issueIdField);

        JLabel vendorIdLabel = new JLabel("Vendor ID:");
        vendorIdLabel.setBounds(50, 160, 150, 30);
        JTextField vendorIdField = new JTextField();
        vendorIdField.setBounds(200, 160, 600, 30);
        panel.add(vendorIdLabel);
        panel.add(vendorIdField);

        JButton submitButton = new JButton("Assign");
        submitButton.setBounds(300, 200, 150, 40);
        submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
        submitButton.addActionListener(e -> {
            String assignmentId = assignmentIdField.getText().trim();
            String issueIdText = issueIdField.getText().trim();
            String vendorIdText = vendorIdField.getText().trim();
            if (assignmentId.isEmpty() || issueIdText.isEmpty() || vendorIdText.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "All fields are required.");
                return;
            }
            int issueId, vendorId;
            try {
                issueId = Integer.parseInt(issueIdText);
                vendorId = Integer.parseInt(vendorIdText);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Issue ID and Vendor ID must be valid
numbers.");
```

```
                return;
            }
            assignVendor(assignmentId, issueId, vendorId);
            cardLayout.show(mainPanel, AUTHORITY_DASHBOARD);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 200, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
AUTHORITY_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void assignVendor(String assignmentId, int issueId, int vendorId) {
        String query = "INSERT INTO assignments (assignment_id, issue_id, vendor_id,
assigned_date, status) VALUES (?, ?, ?, CURDATE(), ?)";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setString(1, assignmentId);
            stmt.setInt(2, issueId);
            stmt.setInt(3, vendorId);
            stmt.setString(4, "ASSIGNED");
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into assignments: " + rowsAffected);

            String updateQuery = "UPDATE issues SET status = 'ASSIGNED' WHERE issue_id =
?";
            try (PreparedStatement updateStmt =
app.getConnection().prepareStatement(updateQuery)) {
                updateStmt.setInt(1, issueId);
                updateStmt.executeUpdate();
            }
            JOptionPane.showMessageDialog(frame, "Vendor assigned successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error assigning vendor: " + e.getMessage());
        }
    }

    private JPanel createSubmitQuotationPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Submit Quotation", SwingConstants.CENTER);
```

```
titleLabel.setBounds(50, 20, 800, 50);
titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
panel.add(titleLabel);

JLabel issueIdLabel = new JLabel("Issue ID:");
issueIdLabel.setBounds(50, 80, 150, 30);
JTextField issueIdField = new JTextField();
issueIdField.setBounds(200, 80, 600, 30);
panel.add(issueIdLabel);
panel.add(issueIdField);

JLabel descriptionLabel = new JLabel("Description:");
descriptionLabel.setBounds(50, 120, 150, 30);
JTextArea descriptionArea = new JTextArea();
descriptionArea.setBounds(200, 120, 600, 60);
descriptionArea.setLineWrap(true);
panel.add(descriptionLabel);
panel.add(descriptionArea);

JLabel priceLabel = new JLabel("Price:");
priceLabel.setBounds(50, 190, 150, 30);
JTextField priceField = new JTextField();
priceField.setBounds(200, 190, 600, 30);
panel.add(priceLabel);
panel.add(priceField);

JLabel estimatedDaysLabel = new JLabel("Estimated Days:");
estimatedDaysLabel.setBounds(50, 230, 150, 30);
JTextField estimatedDaysField = new JTextField();
estimatedDaysField.setBounds(200, 230, 600, 30);
panel.add(estimatedDaysLabel);
panel.add(estimatedDaysField);

JLabel detailsLabel = new JLabel("Details:");
detailsLabel.setBounds(50, 270, 150, 30);
JTextArea detailsArea = new JTextArea();
detailsArea.setBounds(200, 270, 600, 60);
detailsArea.setLineWrap(true);
panel.add(detailsLabel);
panel.add(detailsArea);

JButton submitButton = new JButton("Submit");
submitButton.setBounds(300, 340, 150, 40);
submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
```

```
        submitButton.addActionListener(e -> {
            String issueIdText = issueIdField.getText().trim();
            String description = descriptionArea.getText().trim();
            String priceText = priceField.getText().trim();
            String estimatedDaysText = estimatedDaysField.getText().trim();
            String details = detailsArea.getText().trim();
            if (issueIdText.isEmpty() || description.isEmpty() || priceText.isEmpty() ||
estimatedDaysText.isEmpty() || details.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "All fields are required.");
                return;
            }
            int issueId, estimatedDays;
            double price;
            try {
                issueId = Integer.parseInt(issueIdText);
                price = Double.parseDouble(priceText);
                estimatedDays = Integer.parseInt(estimatedDaysText);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Issue ID, Price, and Estimated Days must
be valid numbers.");
                return;
            }
            submitQuotation(issueId, description, price, estimatedDays, details);
            cardLayout.show(mainPanel, VENDOR_DASHBOARD);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 340, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
VENDOR_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void submitQuotation(int issueId, String description, double price, int estimatedDays,
String details) {
        String query = "INSERT INTO quotations (issue_id, vendor_id, description, price,
estimated_days, details, submission_date, status) VALUES (?, ?, ?, ?, ?, ?, CURDATE(), ?)";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setInt(1, issueId);
            stmt.setInt(2, currentUser.getId());
            stmt.setString(3, description);
            stmt.setDouble(4, price);
            stmt.setInt(5, estimatedDays);
            stmt.setString(6, details);
            stmt.setString(7, "SUBMITTED");
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into quotations: " + rowsAffected);
            JOptionPane.showMessageDialog(frame, "Quotation submitted successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error submitting quotation: " +
e.getMessage());
        }
    }

    private JPanel createVotePanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Vote on Issue/Proposal", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel typeLabel = new JLabel("Vote On:");
        typeLabel.setBounds(50, 80, 150, 30);
        String[] types = {"Issue", "Proposal"};
        JComboBox<String> typeCombo = new JComboBox<>(types);
        typeCombo.setBounds(200, 80, 600, 30);
        panel.add(typeLabel);
        panel.add(typeCombo);

        JLabel idLabel = new JLabel("ID:");
        idLabel.setBounds(50, 120, 150, 30);
        JTextField idField = new JTextField();
        idField.setBounds(200, 120, 600, 30);
        panel.add(idLabel);
        panel.add(idField);

        JButton voteButton = new JButton("Vote");
        voteButton.setBounds(300, 160, 150, 40);
        voteButton.setFont(voteButton.getFont().deriveFont(14.0f));
        voteButton.addActionListener(e -> {
            String type = (String) typeCombo.getSelectedItem();
            String idText = idField.getText().trim();
            if (idText.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "ID is required.");
                return;
            }
            int id;
            try {
                id = Integer.parseInt(idText);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "ID must be a valid number.");
                return;
            }
            if (type.equals("Issue")) {
                voteOnIssue(id);
            } else {
                voteOnProposal(id);
            }
            cardLayout.show(mainPanel, CITIZEN_DASHBOARD);
        });
        panel.add(voteButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 160, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
    CITIZEN_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void voteOnIssue(int issueId) {
        String query = "UPDATE issues SET votes = votes + 1 WHERE issue_id = ?";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setInt(1, issueId);
            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(frame, "Voted on issue successfully.");
            } else {
                JOptionPane.showMessageDialog(frame, "Issue ID not found.");
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error voting on issue: " + e.getMessage());
        }
    }

    private void voteOnProposal(int proposalId) {
        String query = "UPDATE proposals SET votes = votes + 1 WHERE proposal_id = ?";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setInt(1, proposalId);
            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(frame, "Voted on proposal successfully.");
            } else {
                JOptionPane.showMessageDialog(frame, "Proposal ID not found.");
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error voting on proposal: " +
e.getMessage());
        }
    }

    private JPanel createCommentPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Comment on Issue/Proposal", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JLabel typeLabel = new JLabel("Comment On:");
        typeLabel.setBounds(50, 80, 150, 30);
        String[] types = {"Issue", "Proposal"};
        JComboBox<String> typeCombo = new JComboBox<>(types);
        typeCombo.setBounds(200, 80, 600, 30);
        panel.add(typeLabel);
        panel.add(typeCombo);

        JLabel idLabel = new JLabel("ID:");
        idLabel.setBounds(50, 120, 150, 30);
        JTextField idField = new JTextField();
        idField.setBounds(200, 120, 600, 30);
        panel.add(idLabel);
        panel.add(idField);

        JLabel commentLabel = new JLabel("Comment:");
        commentLabel.setBounds(50, 160, 150, 30);
        JTextArea commentArea = new JTextArea();
        commentArea.setBounds(200, 160, 600, 60);
        commentArea.setLineWrap(true);
        panel.add(commentLabel);
        panel.add(commentArea);
        JButton submitButton = new JButton("Submit");
        submitButton.setBounds(300, 230, 150, 40);
        submitButton.setFont(submitButton.getFont().deriveFont(14.0f));
        submitButton.addActionListener(e -> {
            String type = (String) typeCombo.getSelectedItem();
            String idText = idField.getText().trim();
            String comment = commentArea.getText().trim();
            if (idText.isEmpty() || comment.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "ID and Comment are required.");
                return;
            }
            int id;
            try {
                id = Integer.parseInt(idText);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "ID must be a valid number.");
                return;
            }
            if (type.equals("Issue")) {
                commentOnIssue(id, comment);
            } else {
                commentOnProposal(id, comment);
            }
            cardLayout.show(mainPanel, CITIZEN_DASHBOARD);
        });
        panel.add(submitButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(500, 230, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
CITIZEN_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void commentOnIssue(int issueId, String comment) {
        String query = "INSERT INTO comments (citizen_id, issue_id, proposal_id, comment_text,
comment_date) VALUES (?, ?, NULL, ?, CURDATE())";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setInt(1, currentUser.getId());
            stmt.setInt(2, issueId);
            stmt.setString(3, comment);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into comments: " + rowsAffected);
            JOptionPane.showMessageDialog(frame, "Comment added to issue successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error commenting on issue: " +
e.getMessage());
        }
    }

    private void commentOnProposal(int proposalId, String comment) {
        String query = "INSERT INTO comments (citizen_id, issue_id, proposal_id, comment_text,
comment_date) VALUES (?, NULL, ?, ?, CURDATE())";
        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setInt(1, currentUser.getId());
            stmt.setInt(2, proposalId);
            stmt.setString(3, comment);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Rows inserted into comments: " + rowsAffected);
            JOptionPane.showMessageDialog(frame, "Comment added to proposal successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error commenting on proposal: " +
e.getMessage());
        }
    }

    private JPanel createViewDashboardPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("View Dashboard", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JTextArea dashboardArea = new JTextArea();
        dashboardArea.setBounds(50, 80, 800, 400);
        dashboardArea.setEditable(false);
        panel.add(dashboardArea);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 500, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
CITIZEN_DASHBOARD));
        panel.add(backButton);
        return panel;
    }

    private void updateViewDashboardPanel() {
        JPanel panel = (JPanel) mainPanel.getComponent(getPanelIndex(VIEW_DASHBOARD));
        JTextArea dashboardArea = (JTextArea) panel.getComponent(1);
        StringBuilder dashboardData = new StringBuilder();
        String issueQuery = "SELECT * FROM issues WHERE citizen_id = ?";
        String proposalQuery = "SELECT * FROM proposals WHERE citizen_id = ?";
        String grievanceQuery = "SELECT * FROM grievances WHERE citizen_id = ?";

        try (PreparedStatement issueStmt = app.getConnection().prepareStatement(issueQuery)) {
            issueStmt.setInt(1, currentUser.getId());
            try (ResultSet rs = issueStmt.executeQuery()) {
                dashboardData.append("Issues Reported:\n");
                while (rs.next()) {
                    dashboardData.append("ID: ").append(rs.getInt("issue_id"))
                            .append(", Description: ").append(rs.getString("description"))
                            .append(", Status: ").append(rs.getString("status"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            dashboardData.append("Error fetching issues: ").append(e.getMessage()).append("\n");
        }

        try (PreparedStatement proposalStmt =
app.getConnection().prepareStatement(proposalQuery)) {
            proposalStmt.setInt(1, currentUser.getId());
            try (ResultSet rs = proposalStmt.executeQuery()) {
                dashboardData.append("\nProposals Submitted:\n");
                while (rs.next()) {
                    dashboardData.append("ID: ").append(rs.getInt("proposal_id"))
                            .append(", Description: ").append(rs.getString("description"))
                            .append(", Status: ").append(rs.getString("status"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            dashboardData.append("Error fetching proposals:
").append(e.getMessage()).append("\n");
        }
        try (PreparedStatement grievanceStmt =
app.getConnection().prepareStatement(grievanceQuery)) {
            grievanceStmt.setInt(1, currentUser.getId());
            try (ResultSet rs = grievanceStmt.executeQuery()) {
                dashboardData.append("\nGrievances Filed:\n");
                while (rs.next()) {
                    dashboardData.append("ID: ").append(rs.getInt("grievance_id"))
                            .append(", Description: ").append(rs.getString("description"))
                            .append(", Status: ").append(rs.getString("status"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            dashboardData.append("Error fetching grievances:
").append(e.getMessage()).append("\n");
        }

        dashboardArea.setText(dashboardData.toString());
    }

    private JPanel createReviewIssuesPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Review Issues", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JTextArea issuesArea = new JTextArea();
        issuesArea.setBounds(50, 80, 800, 400);
        issuesArea.setEditable(false);
        panel.add(issuesArea);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 500, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
AUTHORITY_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void updateReviewIssuesPanel() {
        JPanel panel = (JPanel) mainPanel.getComponent(getPanelIndex(REVIEW_ISSUES));
        JTextArea issuesArea = (JTextArea) panel.getComponent(1);
        StringBuilder issuesData = new StringBuilder();
        String query = "SELECT * FROM issues";

        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    issuesData.append("ID: ").append(rs.getInt("issue_id"))
                            .append(", Description: ").append(rs.getString("description"))
                            .append(", Category: ").append(rs.getString("category"))
                            .append(", Status: ").append(rs.getString("status"))
                            .append(", Votes: ").append(rs.getInt("votes"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            issuesData.append("Error fetching issues: ").append(e.getMessage());
        }

        issuesArea.setText(issuesData.toString());
    }

    private JPanel createReviewProposalsPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("Review Proposals", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JTextArea proposalsArea = new JTextArea();
        proposalsArea.setBounds(50, 80, 800, 400);
        proposalsArea.setEditable(false);
        panel.add(proposalsArea);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 500, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
AUTHORITY_DASHBOARD));
        panel.add(backButton);

        return panel;
    }
    private void updateReviewProposalsPanel() {
        JPanel panel = (JPanel)
mainPanel.getComponent(getPanelIndex(REVIEW_PROPOSALS));
        JTextArea proposalsArea = (JTextArea) panel.getComponent(1);
        StringBuilder proposalsData = new StringBuilder();
        String query = "SELECT * FROM proposals";

        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    proposalsData.append("ID: ").append(rs.getInt("proposal_id"))
                            .append(", Description: ").append(rs.getString("description"))
                            .append(", Status: ").append(rs.getString("status"))
                            .append(", Votes: ").append(rs.getInt("votes"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            proposalsData.append("Error fetching proposals: ").append(e.getMessage());
        }

        proposalsArea.setText(proposalsData.toString());
    }

    private JPanel createViewCitizensPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("View Citizens", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JTextArea citizensArea = new JTextArea();
        citizensArea.setBounds(50, 80, 800, 400);
        citizensArea.setEditable(false);
        panel.add(citizensArea);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 500, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
AUTHORITY_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void updateViewCitizensPanel() {
        JPanel panel = (JPanel) mainPanel.getComponent(getPanelIndex(VIEW_CITIZENS));
        JTextArea citizensArea = (JTextArea) panel.getComponent(1);
        StringBuilder citizensData = new StringBuilder();
        String query = "SELECT * FROM citizens";

        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    citizensData.append("ID: ").append(rs.getInt("id"))
                            .append(", Name: ").append(rs.getString("name"))
                            .append(", Email: ").append(rs.getString("email"))
                            .append(", Mobile: ").append(rs.getString("mobile_no"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            citizensData.append("Error fetching citizens: ").append(e.getMessage());
        }

        citizensArea.setText(citizensData.toString());
    }

    private JPanel createViewQuotationsPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("View Quotations", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JTextArea quotationsArea = new JTextArea();
        quotationsArea.setBounds(50, 80, 800, 400);
        quotationsArea.setEditable(false);
        panel.add(quotationsArea);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 500, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
VENDOR_DASHBOARD));
        panel.add(backButton);
        return panel;
    }

    private void updateViewQuotationsPanel() {
        JPanel panel = (JPanel) mainPanel.getComponent(getPanelIndex(VIEW_QUOTATIONS));
        JTextArea quotationsArea = (JTextArea) panel.getComponent(1);
        StringBuilder quotationsData = new StringBuilder();
        String query = "SELECT * FROM quotations WHERE vendor_id = ?";

        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setInt(1, currentUser.getId());
            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    quotationsData.append("ID: ").append(rs.getInt("quotation_id"))
                            .append(", Issue ID: ").append(rs.getInt("issue_id"))
                            .append(", Description: ").append(rs.getString("description"))
                            .append(", Price: ").append(rs.getDouble("price"))
                            .append(", Status: ").append(rs.getString("status"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            quotationsData.append("Error fetching quotations: ").append(e.getMessage());
        }

        quotationsArea.setText(quotationsData.toString());
    }

    private JPanel createViewAssignmentsPanel() {
        JPanel panel = new JPanel(null);
        JLabel titleLabel = new JLabel("View Assignments", SwingConstants.CENTER);
        titleLabel.setBounds(50, 20, 800, 50);
        titleLabel.setFont(titleLabel.getFont().deriveFont(24.0f));
        panel.add(titleLabel);

        JTextArea assignmentsArea = new JTextArea();
        assignmentsArea.setBounds(50, 80, 800, 400);
        assignmentsArea.setEditable(false);
        panel.add(assignmentsArea);

        JButton backButton = new JButton("Back");
        backButton.setBounds(350, 500, 150, 40);
        backButton.setFont(backButton.getFont().deriveFont(14.0f));
```

```
        backButton.addActionListener(e -> cardLayout.show(mainPanel,
VENDOR_DASHBOARD));
        panel.add(backButton);

        return panel;
    }

    private void updateViewAssignmentsPanel() {
        JPanel panel = (JPanel)
mainPanel.getComponent(getPanelIndex(VIEW_ASSIGNMENTS));
        JTextArea assignmentsArea = (JTextArea) panel.getComponent(1);
        StringBuilder assignmentsData = new StringBuilder();
        String query = "SELECT * FROM assignments WHERE vendor_id = ?";

        try (PreparedStatement stmt = app.getConnection().prepareStatement(query)) {
            stmt.setInt(1, currentUser.getId());
            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    assignmentsData.append("Assignment ID: ").append(rs.getString("assignment_id"))
                            .append(", Issue ID: ").append(rs.getInt("issue_id"))
                            .append(", Status: ").append(rs.getString("status"))
                            .append(", Assigned Date: ").append(rs.getDate("assigned_date"))
                            .append("\n");
                }
            }
        } catch (SQLException e) {
            assignmentsData.append("Error fetching assignments: ").append(e.getMessage());
        }

        assignmentsArea.setText(assignmentsData.toString());
    }

    private int getPanelIndex(String panelName) {
        for (int i = 0; i < mainPanel.getComponentCount(); i++) {
            if (mainPanel.getComponent(i).getName() != null &&
mainPanel.getComponent(i).getName().equals(panelName)) {
                return i;
            }
        }
        for (int i = 0; i < mainPanel.getComponentCount(); i++) {
            if (((JPanel) mainPanel.getComponent(i)).getLayout() instanceof CardLayout) {
                return i;
            }
        }
        return -1;
    }
}
```