

KULLIYAH OF INFORMATION & COMMUNICATION TECHNOLOGY

BICS 1304 OBJECT-ORIENTED PROGRAMMING **SEMESTER 2, 2024/2025** **SECTION 10**

GROUP D

ASSIGNMENT 2 **IIUM ACCIDENT DETECTION SYSTEM**

PREPARED BY:

NAME	MATRIC NO.	ROLE	PARTICIPATION
SARAH YASMIN BINTI RODZMAN	2413034	PROJECT MANAGER	100%
NUR BALQIS BINTI MOHD KAMARULZAMAN	2410006	FILE HANDLER	100%
PUTRI AIMI BATRISYIA BINTI MUHAMMAD YUSRI	2320206	BACKEND DEVELOPER	100%
NURIN SOFINA BINTI YUSDI	2221372	GUI DESIGNER	100%

LECTURER

DR. DINI OKTARINA DWI HANDAYANI

DUE

30 MAY 2025

Table of Content

Table of Content.....	3
1.0 JavaFX Implementation.....	4
1.1 User Dashboard.....	4
1.2 Admin Dashboard.....	10
1.3 Responder Dashboard.....	16
2.0 JavaFx Coding.....	28
2.1 Login Window.....	31
2.2 User Dashboard.....	35
2.3 Admin Dashboard.....	40
2.4 Responder Dashboard.....	43
2.5 Accident Report.....	56
2.6 Main app.....	61

1.0 JavaFX Implementation

1.1 User Dashboard

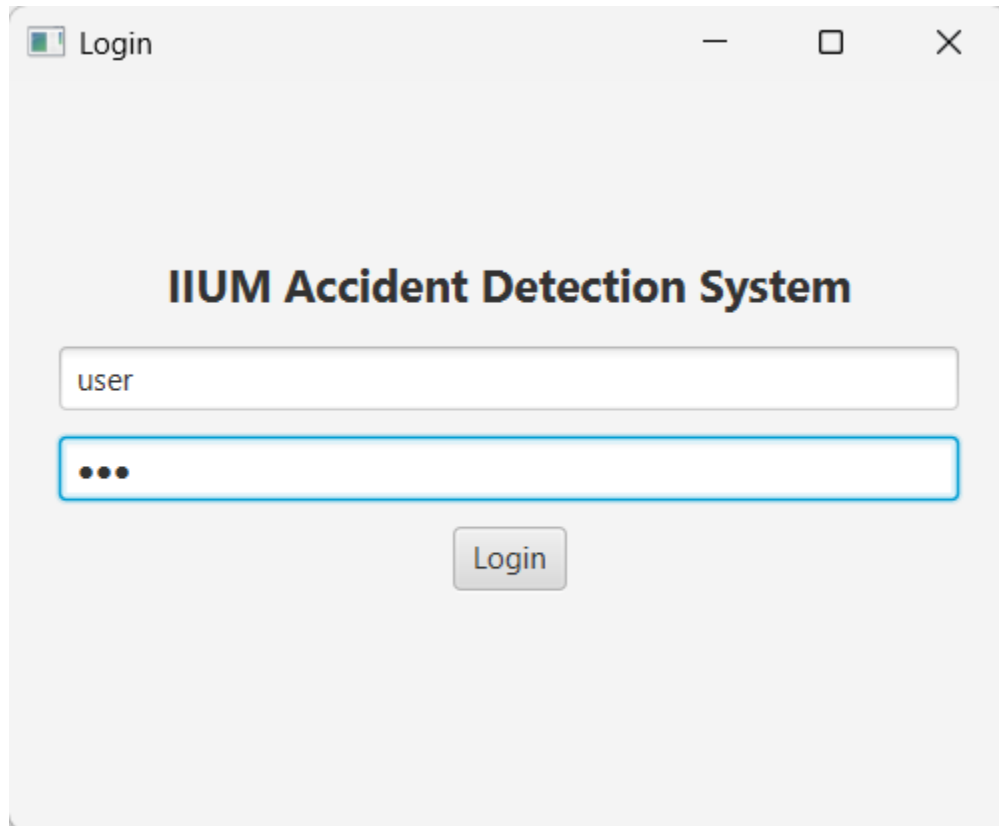


Figure 1: User Dashboard Interface

Users enter their username as “user” and password “123” to login into the user dashboard interface.

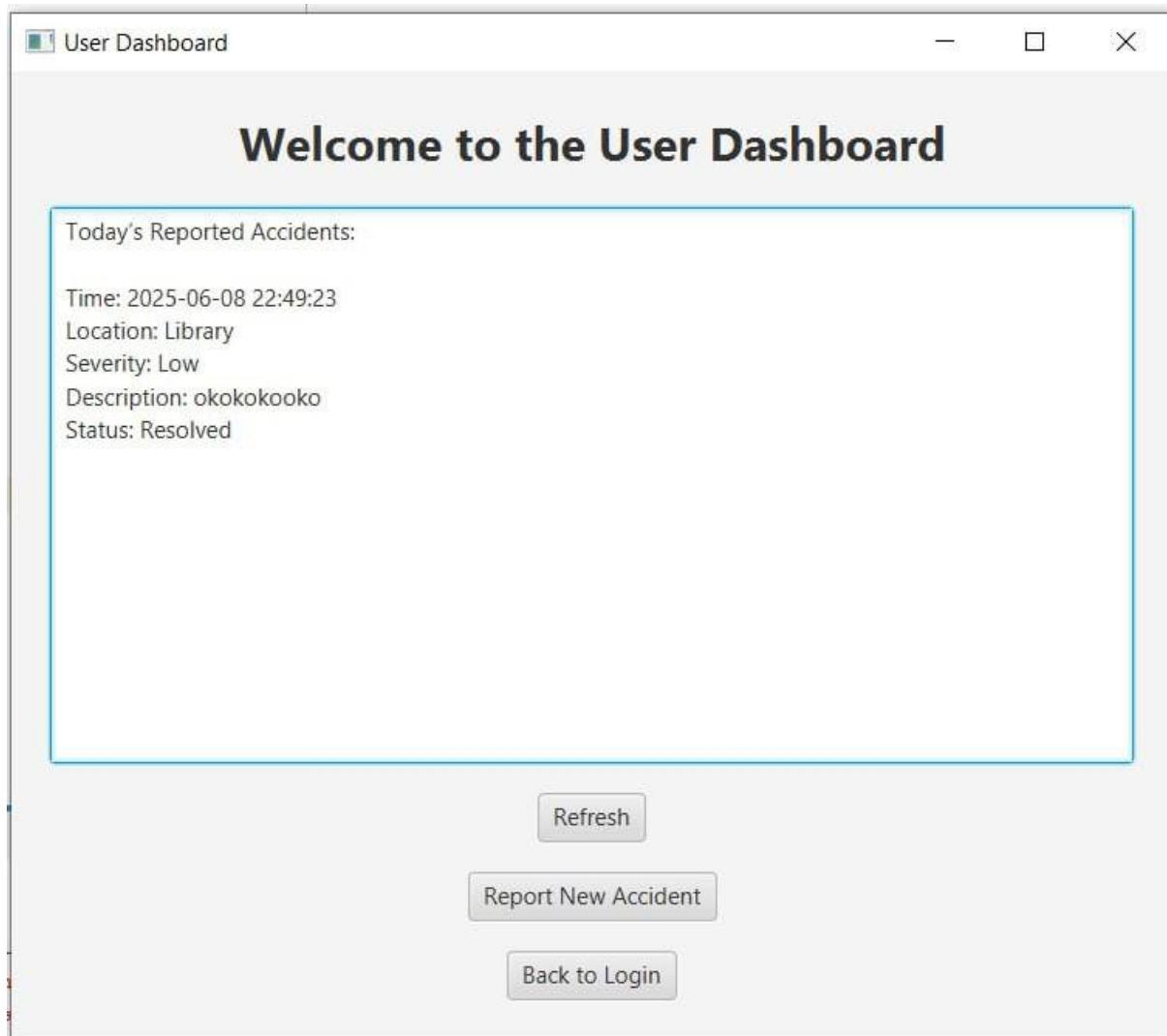


Figure 2: Display the report of accident

The users can see accidents reported on the current day. The button of "Report New Accident" will takes users to the accident reporting form, the button of refresh will update the current accident and the button of "Back to Login" can allows users to return to the login screen

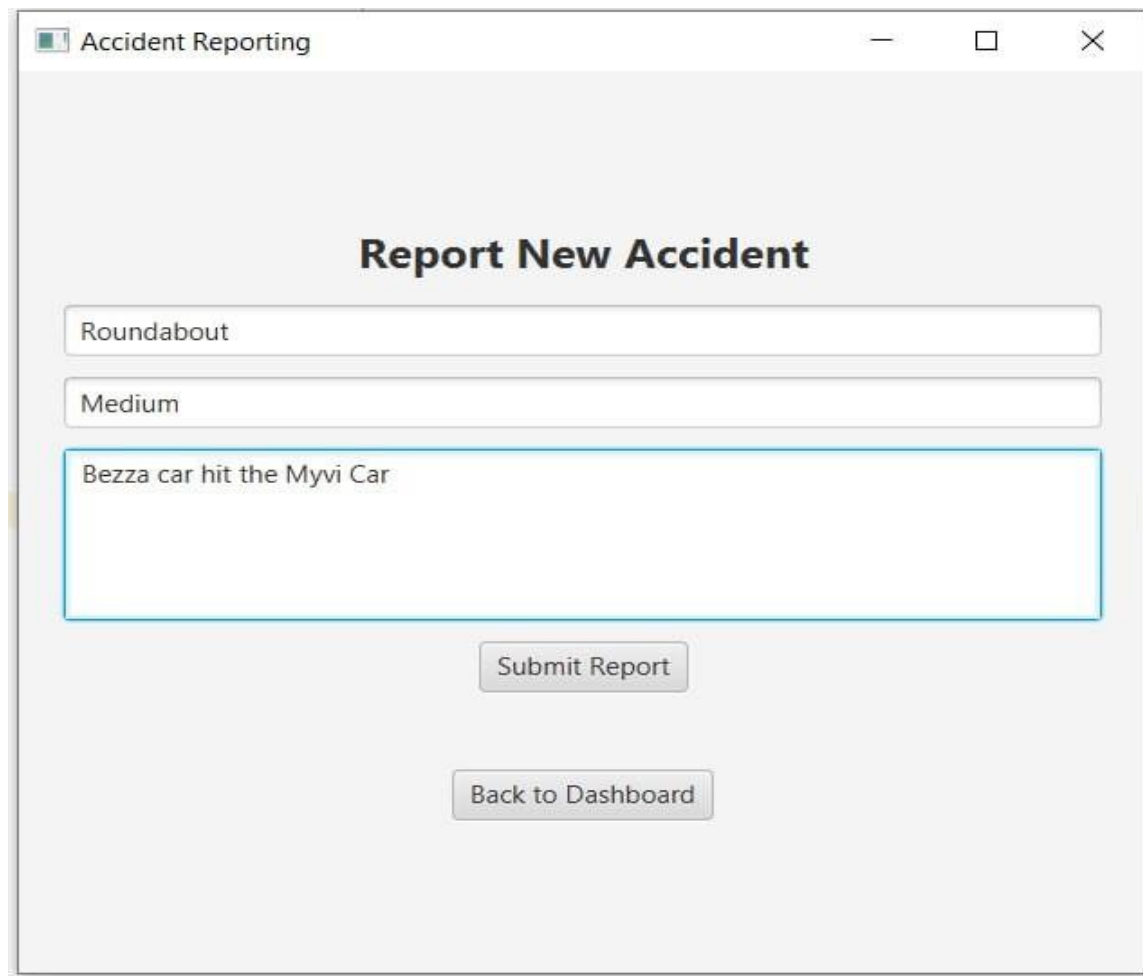
The screenshot shows a window titled "Accident Reporting" with standard window controls (minimize, maximize, close). The main heading is "Report New Accident". Below the heading are three input fields: a text box for the accident location (highlighted with a blue border), a dropdown menu for "Enter severity level (Low/Medium/High)", and a larger text area for "Describe the accident details". At the bottom of the form are two buttons: "Submit Report" and "Back to Dashboard".

Figure 3: Display the form accident reporting

When the user clicks the button of "Report New Accident" at user dashboard, the system will ask user to fill with three input fields:

1. Enter the accident location
2. Enter the severity level (with options like Low/Medium/High)
3. Describing the accident details

At the bottom, there's a "Submit Report" button, along with the button of "Back to Dashboard" to go back to the previous screen



The screenshot shows a web application window titled "Accident Reporting". Inside the window, the heading "Report New Accident" is centered. Below the heading, there are three input fields: a text field containing "Roundabout", a text field containing "Medium", and a larger text area containing "Bezza car hit the Myvi Car". The text area is highlighted with a blue border. Below the input fields, there are two buttons: "Submit Report" and "Back to Dashboard".

Accident Reporting

Report New Accident

Roundabout

Medium

Bezza car hit the Myvi Car

Submit Report

Back to Dashboard

Figure 4: Example if user fill the text field.

This is an example if user fill the information.

The image shows a web application window titled "Accident Reporting". Inside the window, the heading "Report New Accident" is centered. Below the heading are three input fields: "Enter accident location", "Enter severity level (Low/Medium/High)", and "Describe the accident details". A "Submit Report" button is positioned below the input fields. Below the button, the text "Accident reported successfully." is displayed. At the bottom, there is a "Back to Dashboard" button.

Figure 5: View when the user submit report

When the user clicks the button "Submit Report" the text will appear button that said that "Accident reported successfully".

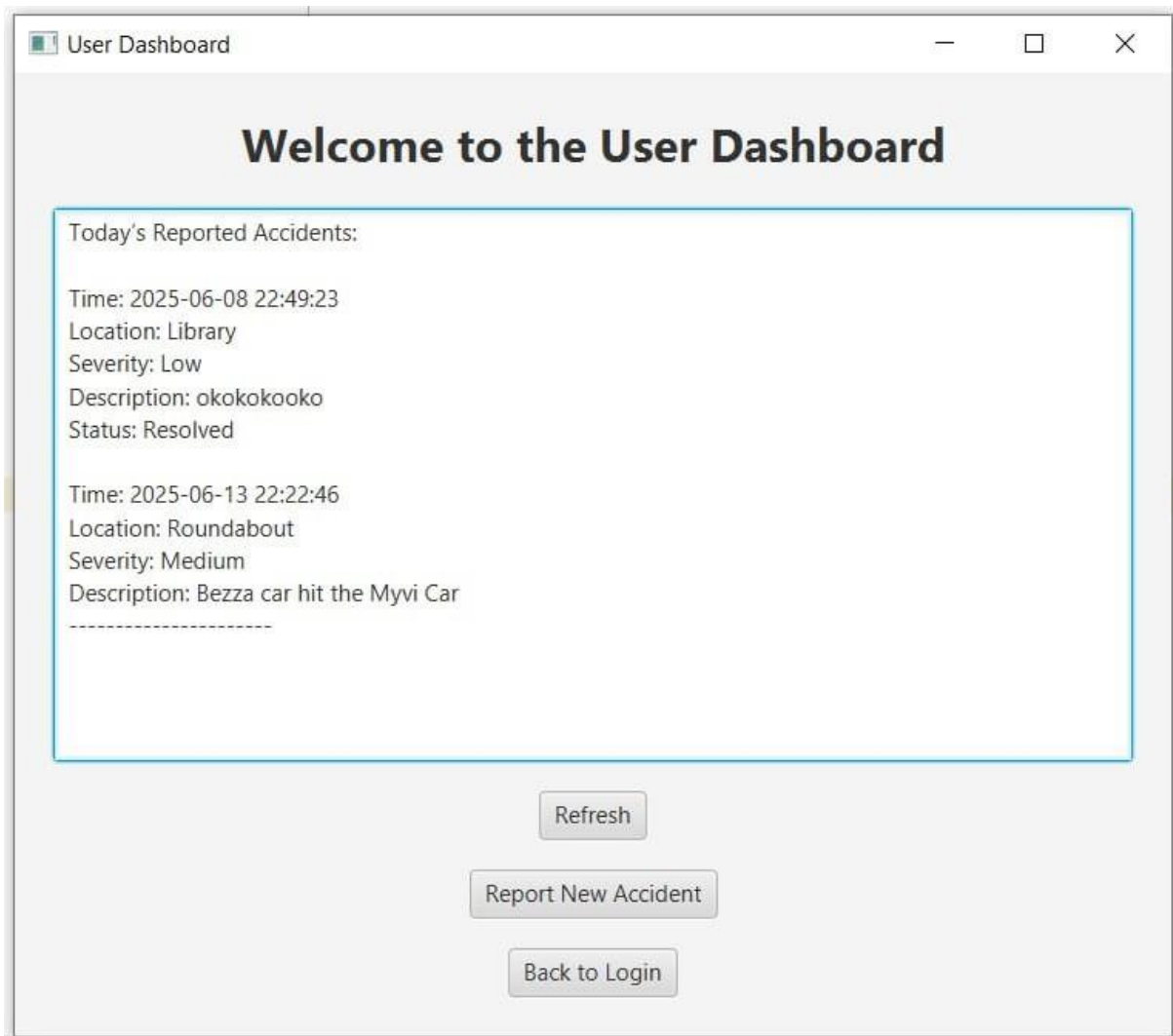
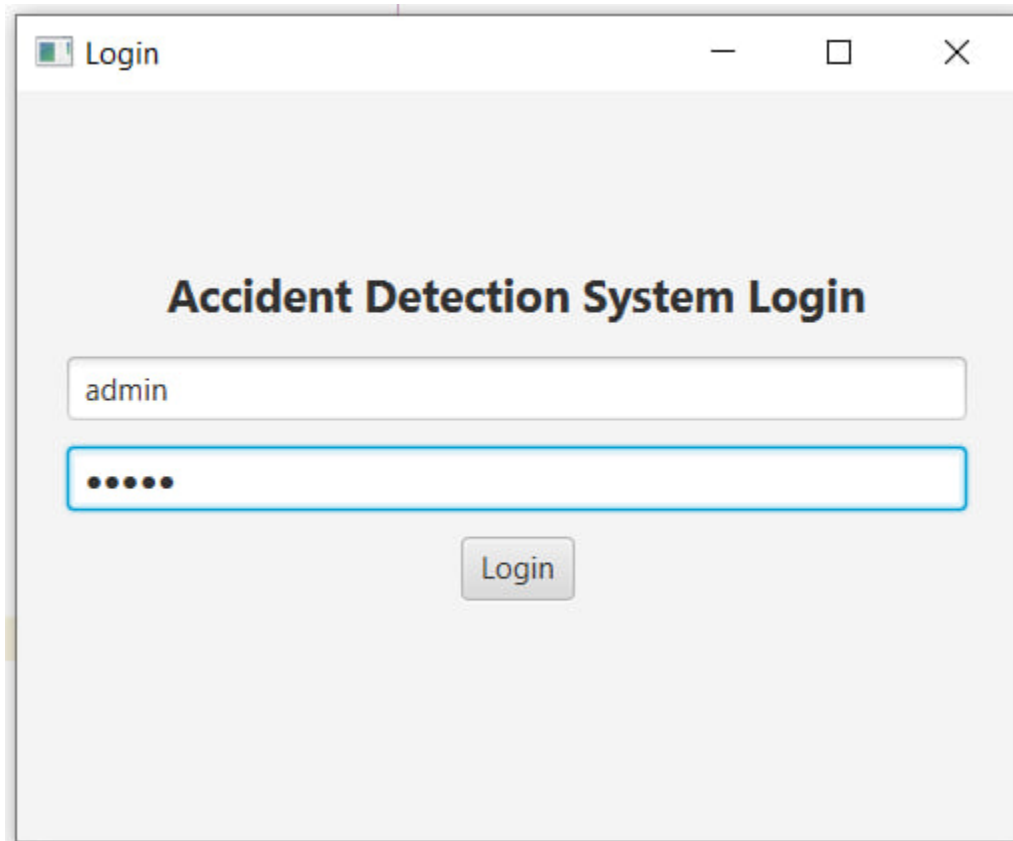


Figure 6: Update the report of accident

After the user submit the report and go back to the dashboard, the information that user just enter will appear in the screen

1.2 Admin Dashboard



The image shows a web browser window titled "Login". The main heading is "Accident Detection System Login". Below the heading, there is a username input field containing the text "admin". Below the username field is a password input field with five dots representing the password. A "Login" button is positioned below the password field.

Figure 7: Admin Login Interface

Admins enter their username "admin" and password "admin" to login into the admin dashboard interface.

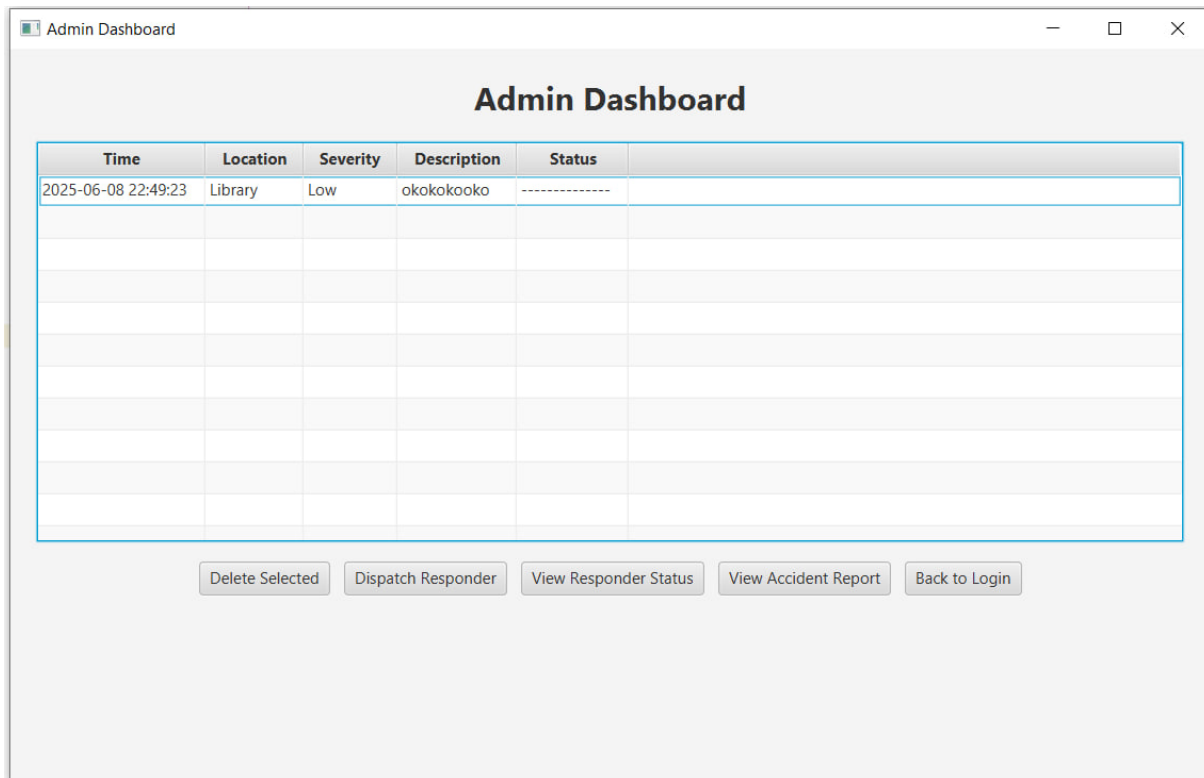


Figure 8: Admin Dashboard Interface

The admin can view the accident reported by the user. Then, they need to click a desired row below the "Status" column to update the accident's status.

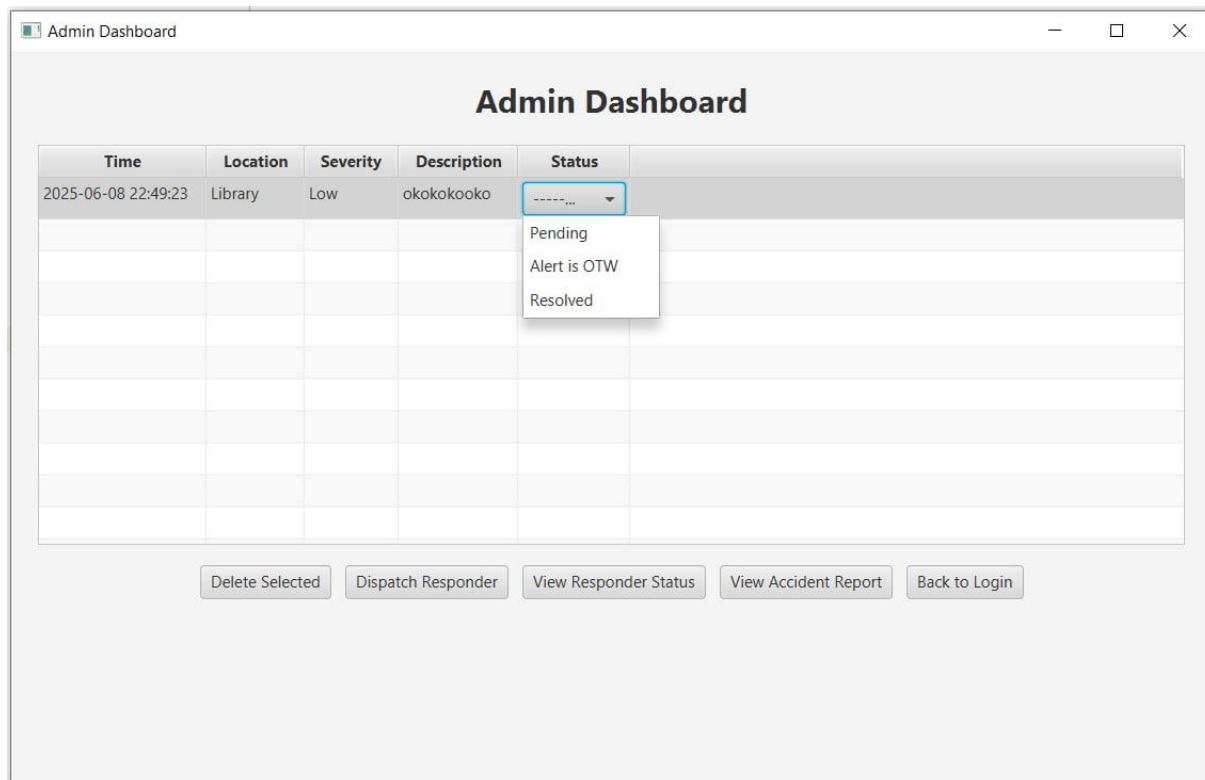
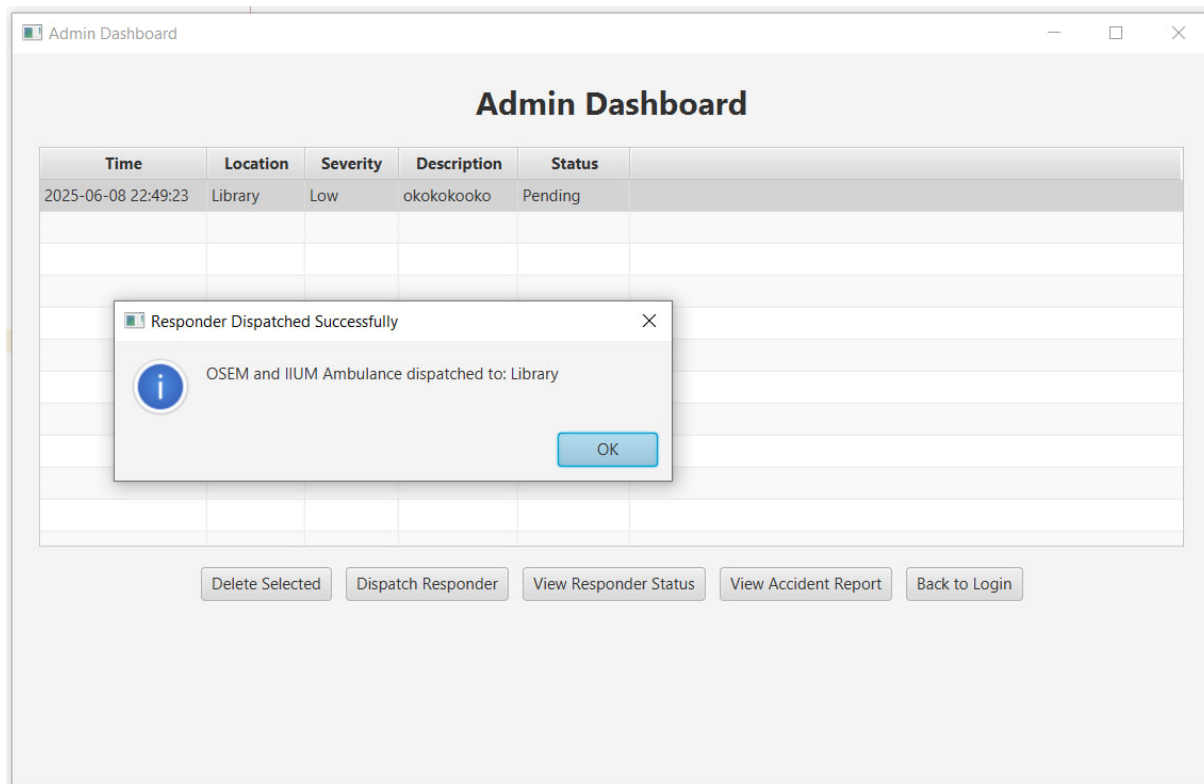


Figure 9: Options for Status in Admin Dashboard

Admins can select “Pending”, “Alert is OTW”, or “Resolved” to update the accident status.



After clicking the “Dispatch Responder” button, the admin can see the message displayed. It shows the respondent in charge (OSEM and IIUM Ambulance) as well as the accident location that reported by the user earlier. For example, OSEM and IIUM Ambulance dispatched to: Library.

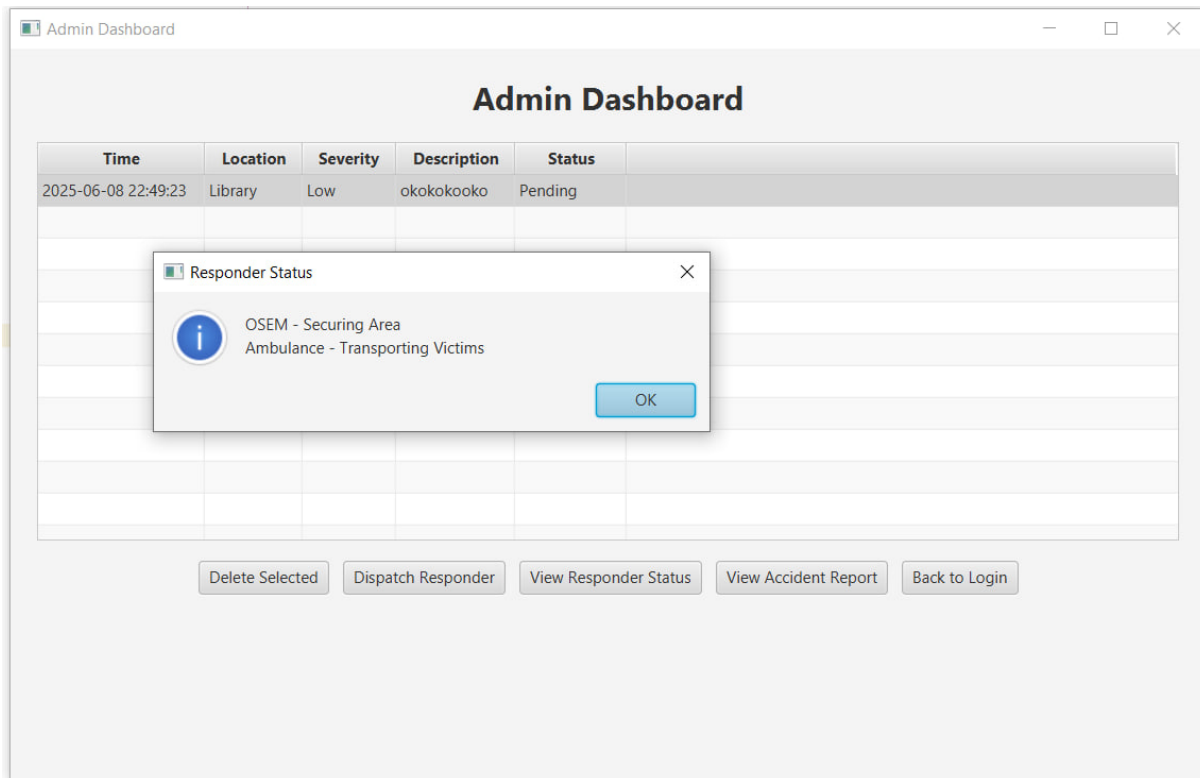


Figure 11: View Responder Status display

After clicking the “View Responder Status” button, the admin can see the message displayed. It is basically updating and showing each role of these two respondents.

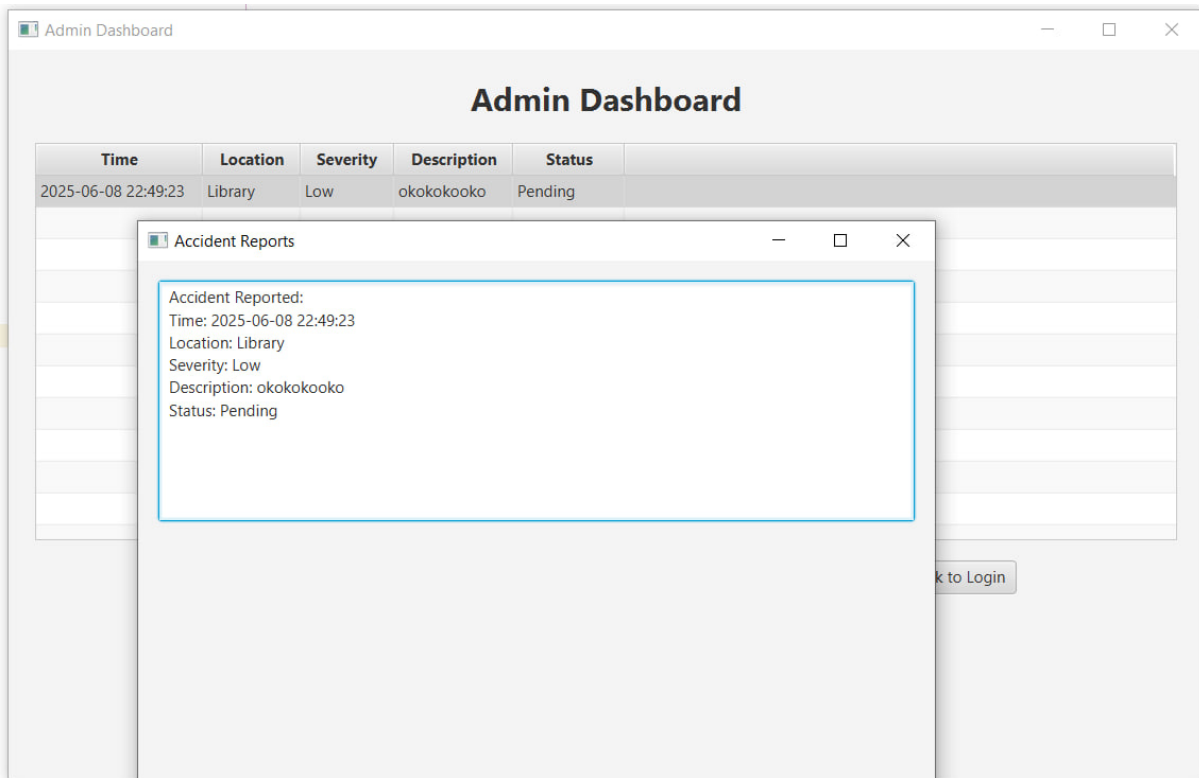


Figure 12: View Accident Report display

After clicking the “View Accident Report” button, the admin can see the message displayed. It will display all the accident reported, concisely a summary.

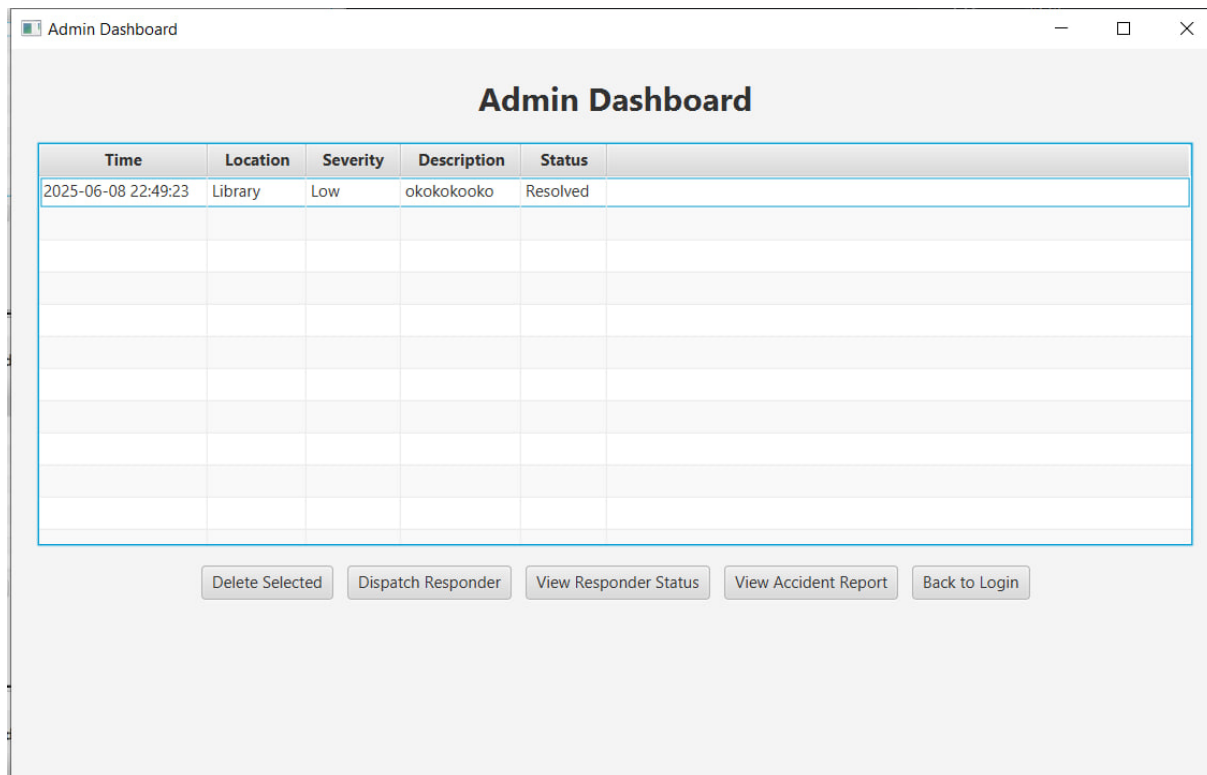


Figure 13: Back to Login

Lastly, the admin can click “Back to Login” button to go to the login page again.

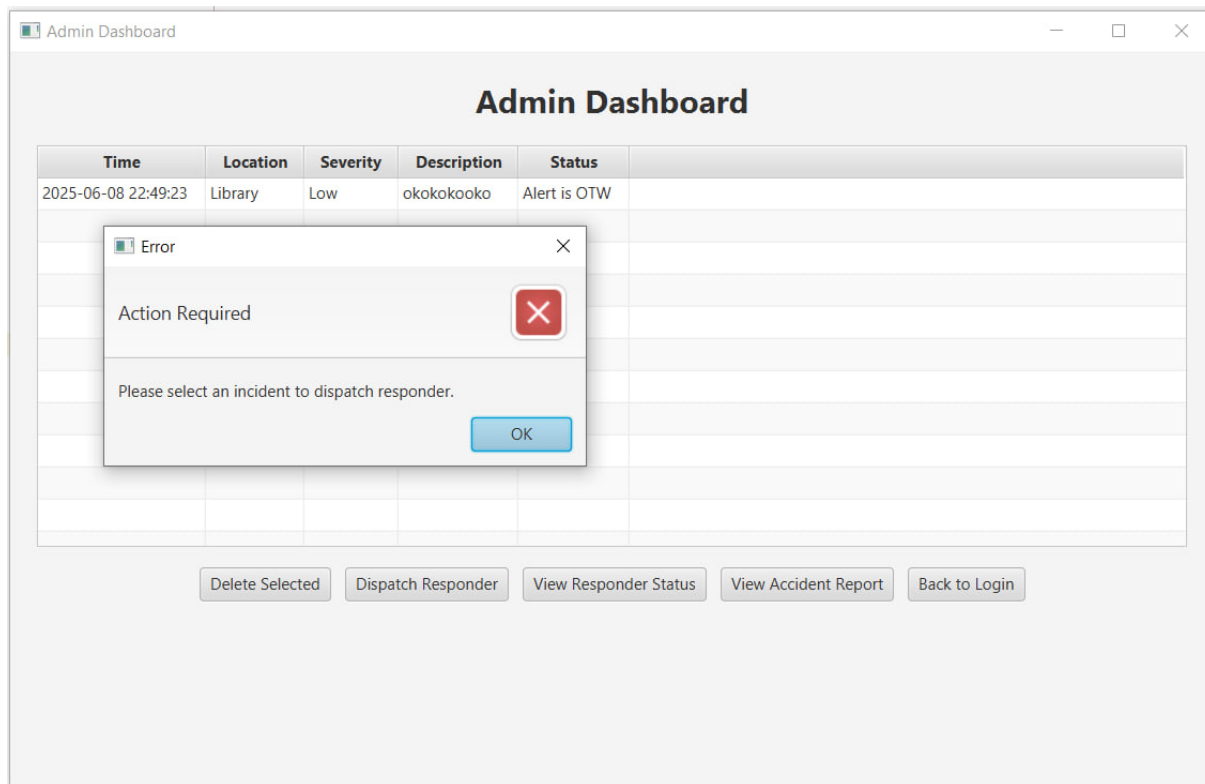
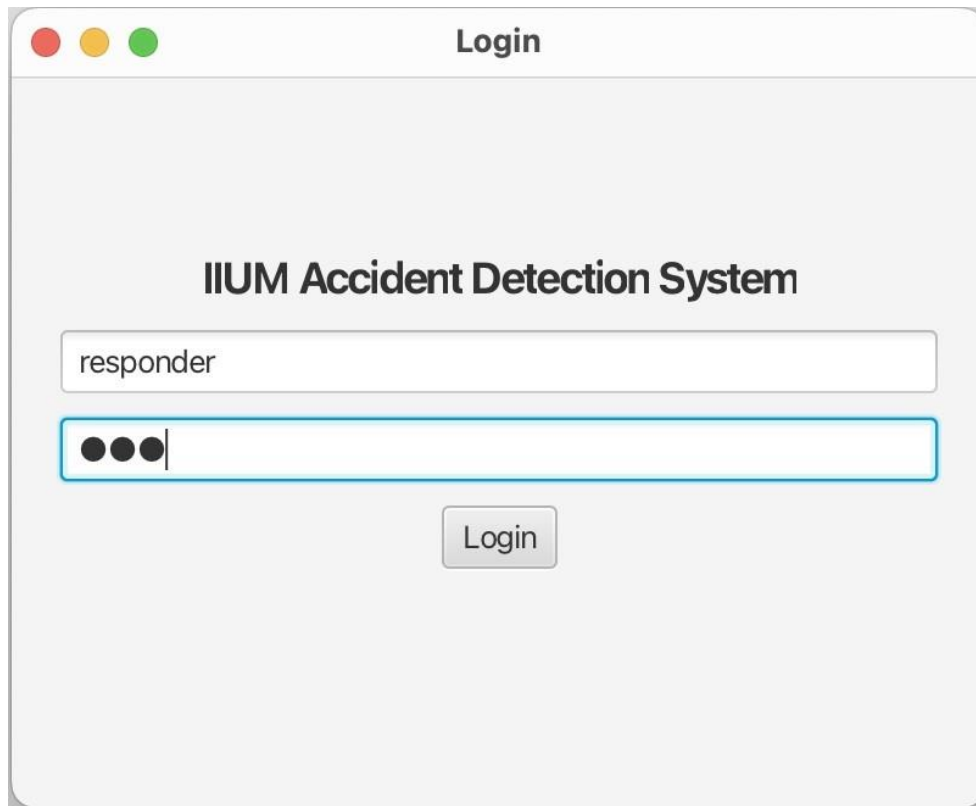


Figure 14: Error Feature in Admin Dashboard

An error will occur if the admin does not select which report they want to update. The message will be displayed as above.

1.3 Responder Dashboard



The image shows a login window titled "Login" with a standard macOS-style title bar (red, yellow, green buttons). The main content area has a light gray background. At the top center, the text "IIUM Accident Detection System" is displayed in a bold, black font. Below this, there are two input fields. The first field contains the text "responder". The second field contains three black dots, indicating a password field. Below the input fields is a button labeled "Login".

Figure 15: Admin Login Interface

To enter the Responder dashboard, the user must enter the username as 'responder' and the password as '111'. After that, the button 'Login' must be clicked to gain access to the Responder dashboard.

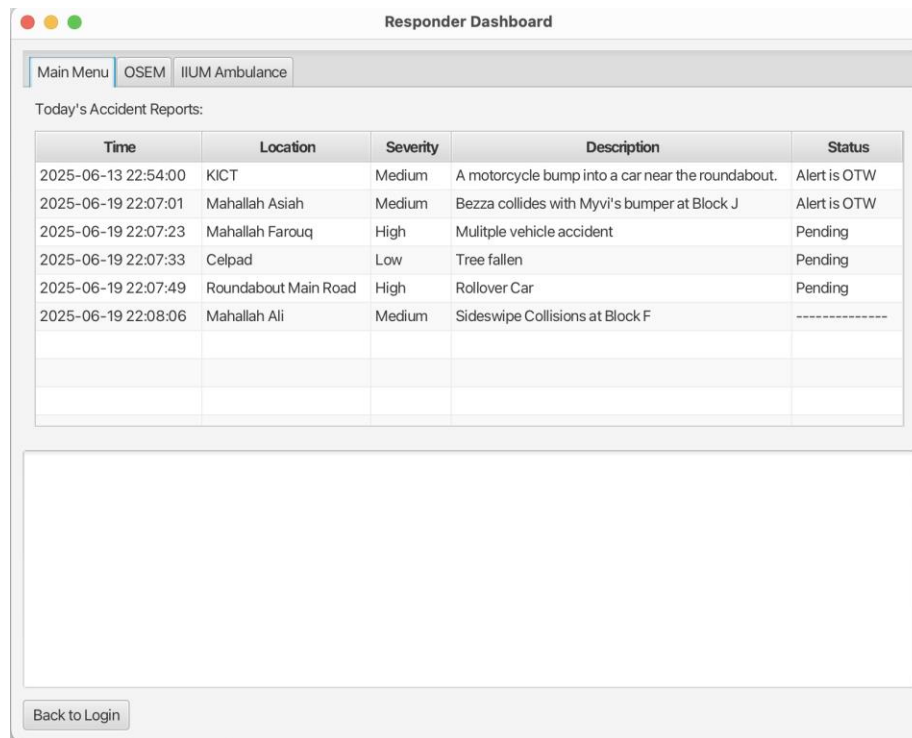


Figure 16: Display Responder Dashboard

This will be the interface of the Responder dashboard that the user will see once they have successfully entered. Daily accident reports will be visible for them to view with their specific time, location, severity, description and status. The responder is responsible for updating the status of the accident. There are several buttons for users to interact with in the first page of Responder dashboard such as the me "OSEM", "IIUMAmbulance".

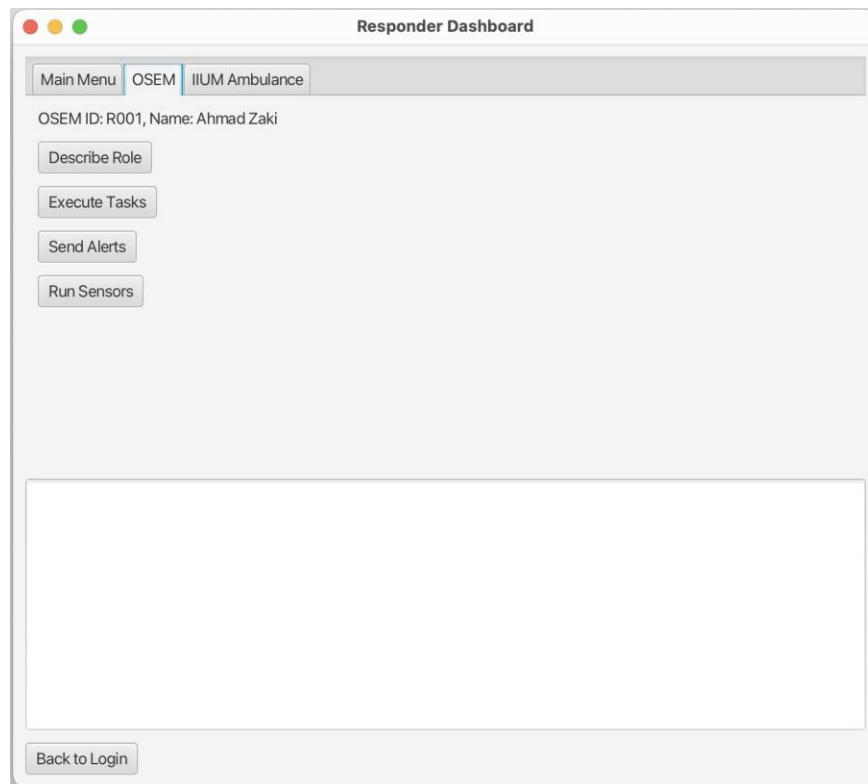


Figure 17: View of OSEM

If the user clicks on the “OSEM” menu, the officer on shift information will reveal: “OSEM ID: R001, Name: Ahmad Zaki”. There are also several buttons for user to choose from:

- Describe Roles - to show duties and responsibilities of responders.
- Execute Task - allow responders to initiate response procedures.
- Send Alerts - trigger notifications and warnings to related responders.
- Run Sensors - to activate the accident detection sensors.

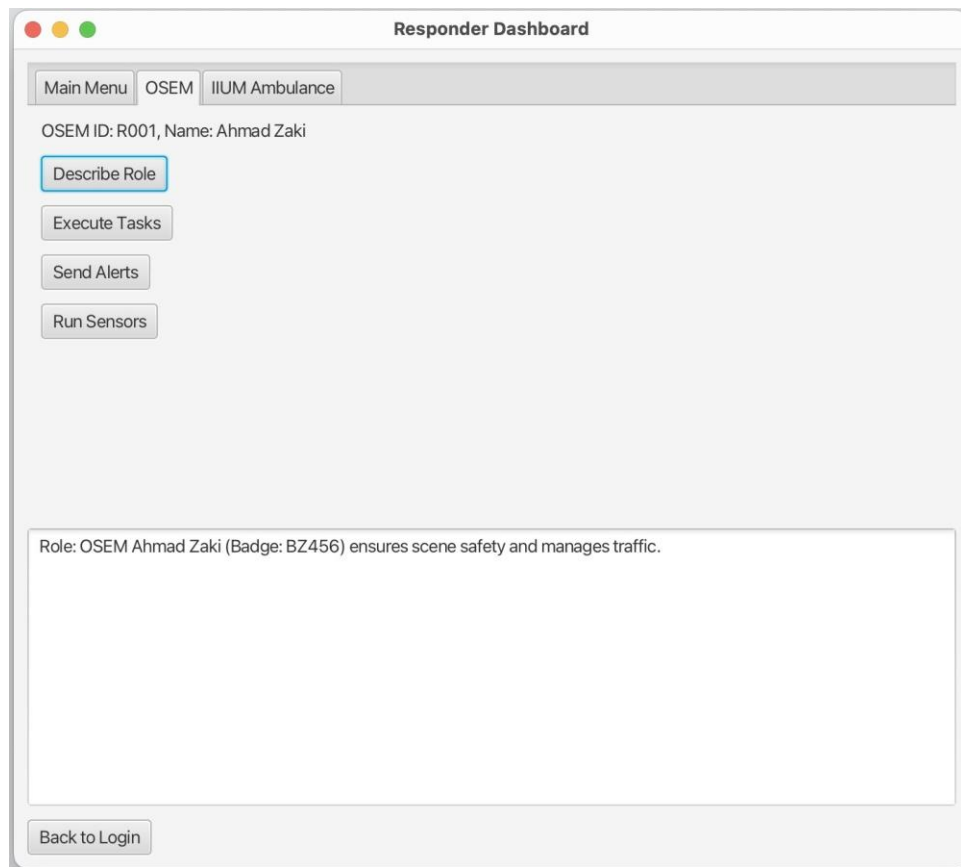


Figure 18: Describe Role button clicked in OSEM interface

If the user clicks on the “Describe Role” button, then the description of the officers on shift will be displayed as shown.

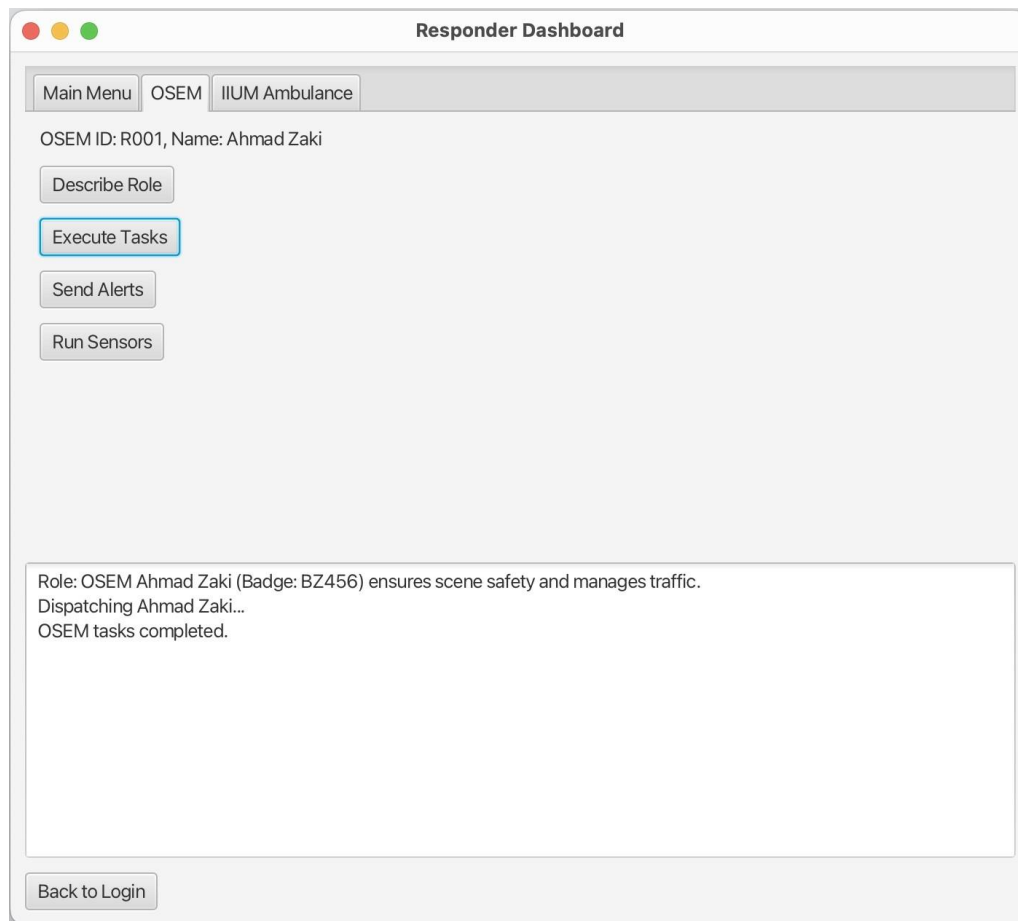


Figure 19: Execute Tasks button clicked in OSEM interface

If the user clicks on the “Execute Tasks” button, the output will show the message that the system is dispatching the officer in charge and will inform the user once the tasks have been completed.

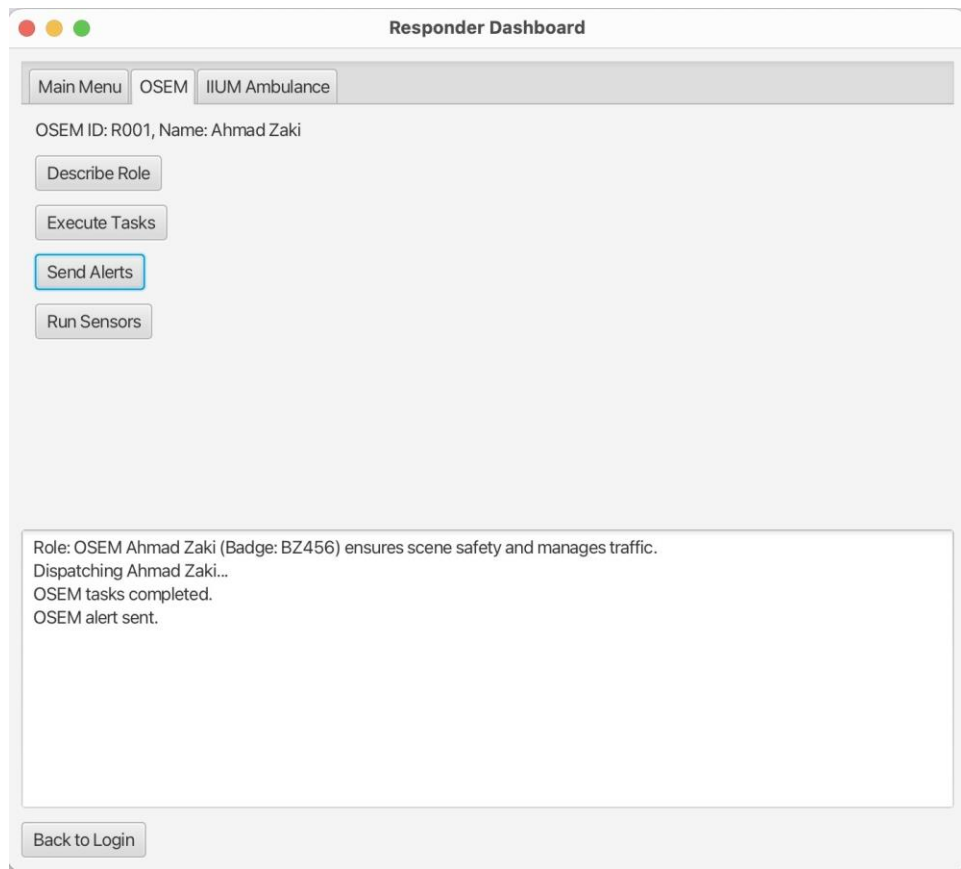


Figure 20: Send Alerts button clicked in OSEM interface

If the user clicks on the “Send Alerts” button, it will notify the OSEM officer in charge and will display the message “OSEM alert sent”.

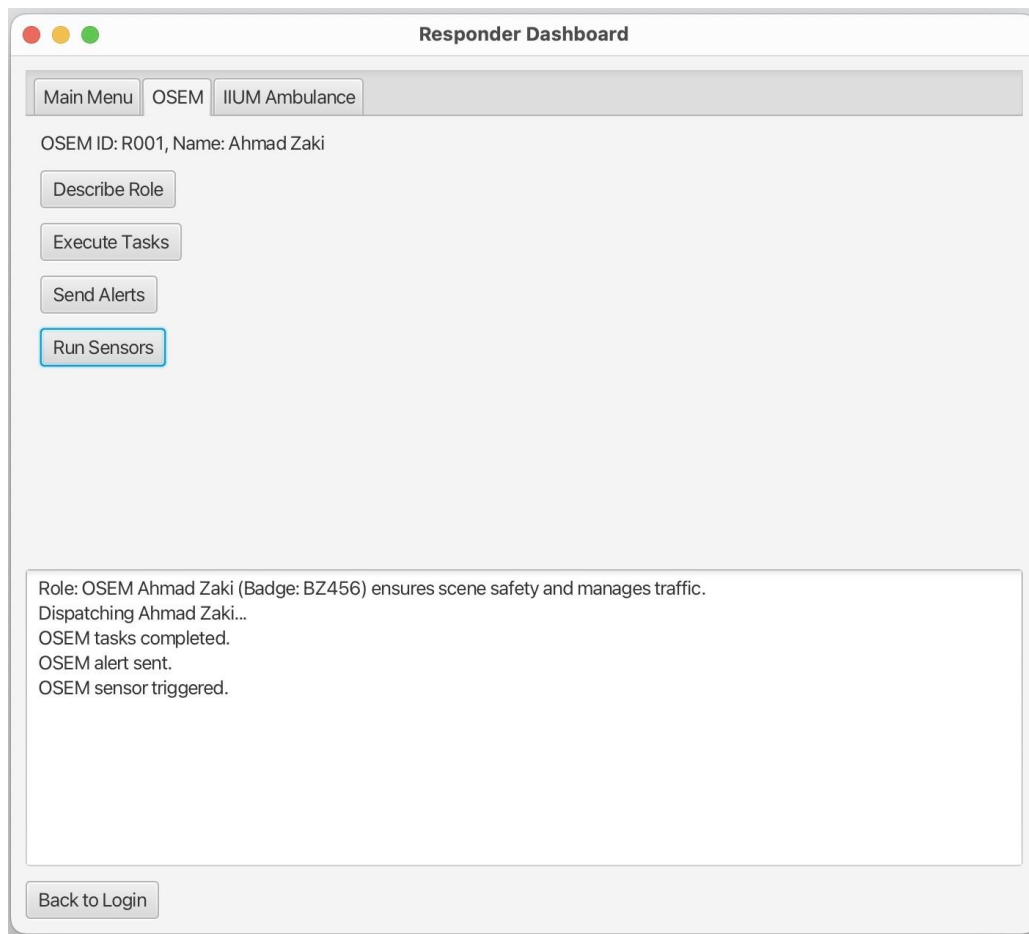


Figure 21: Run Sensors button clicked in OSEM interface

If the user clicks on the “Run Sensors” button, the program will activate the monitoring and detection system and will display the message “OSEM sensor triggered”.

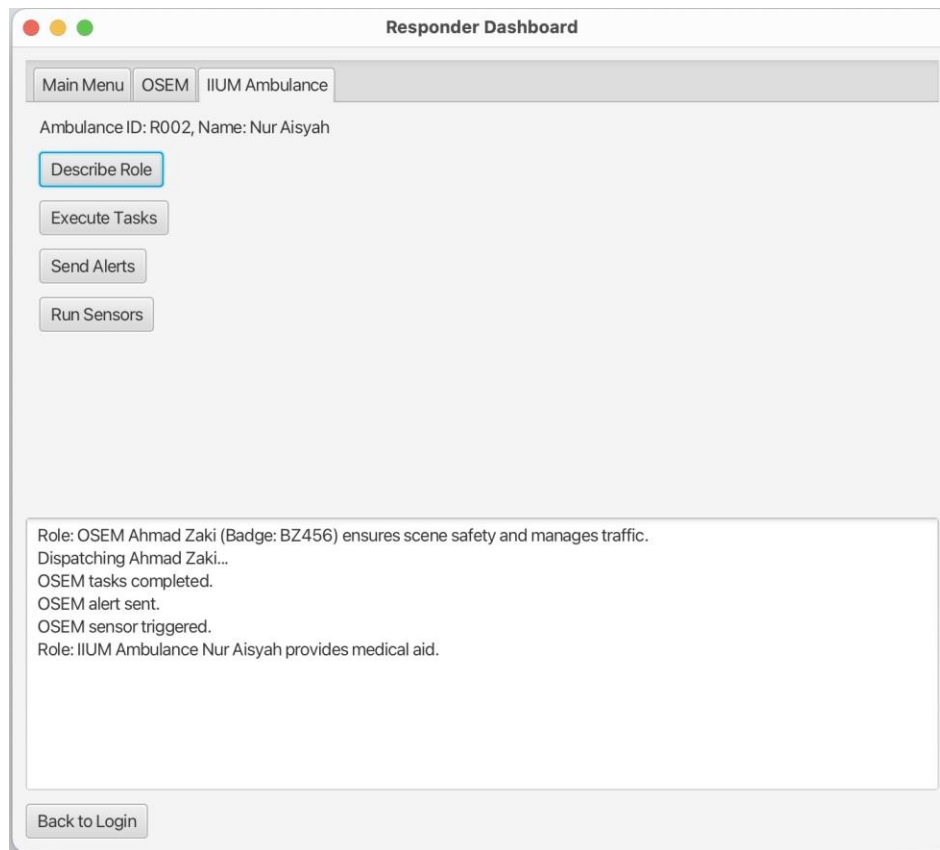


Figure 22: Describe Role button clicked in IIUMAmbulance interface

If the user chooses the menu “IIUMAmbulance”, the interface will display the ambulance ID and the medical person in charge for the day.

If the user clicks on the button “Describe Role”, then the user will be able to see the description of the medical officer appointed.

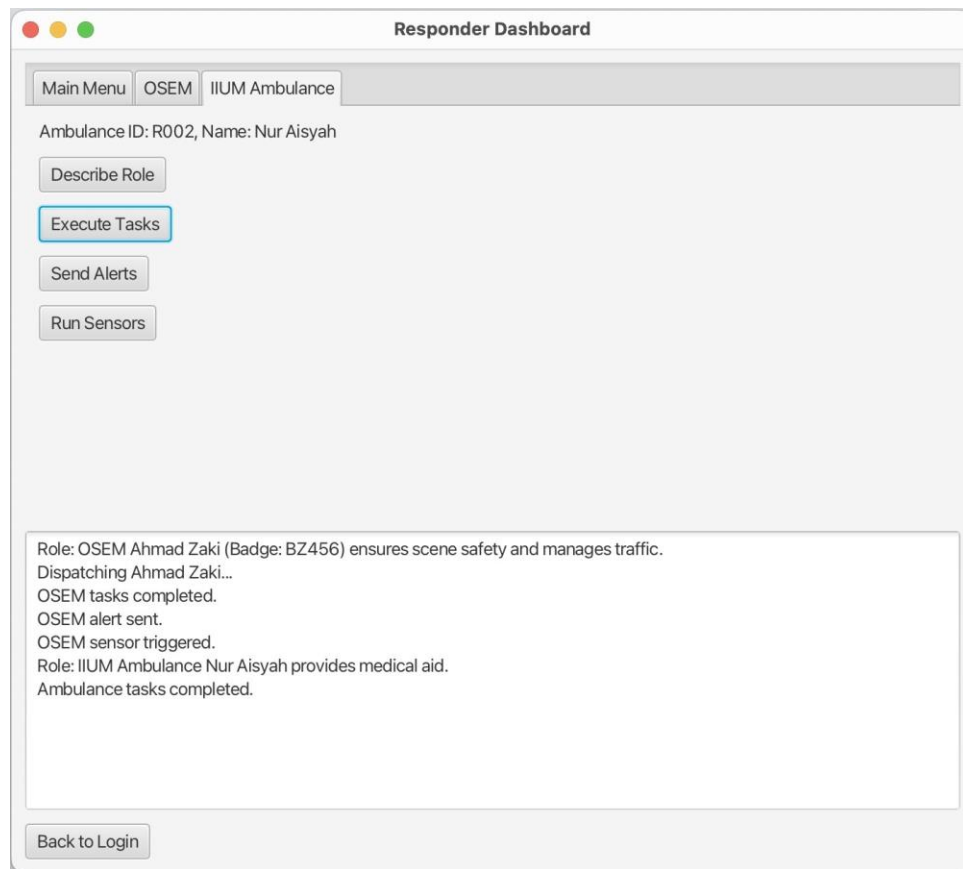


Figure 23: Execute Tasks button clicked in IIUMAmbulance interface

If the user clicks on the “Execute Tasks” button, the program will show their operational duties and display a message to inform the user when the tasks have been completed.

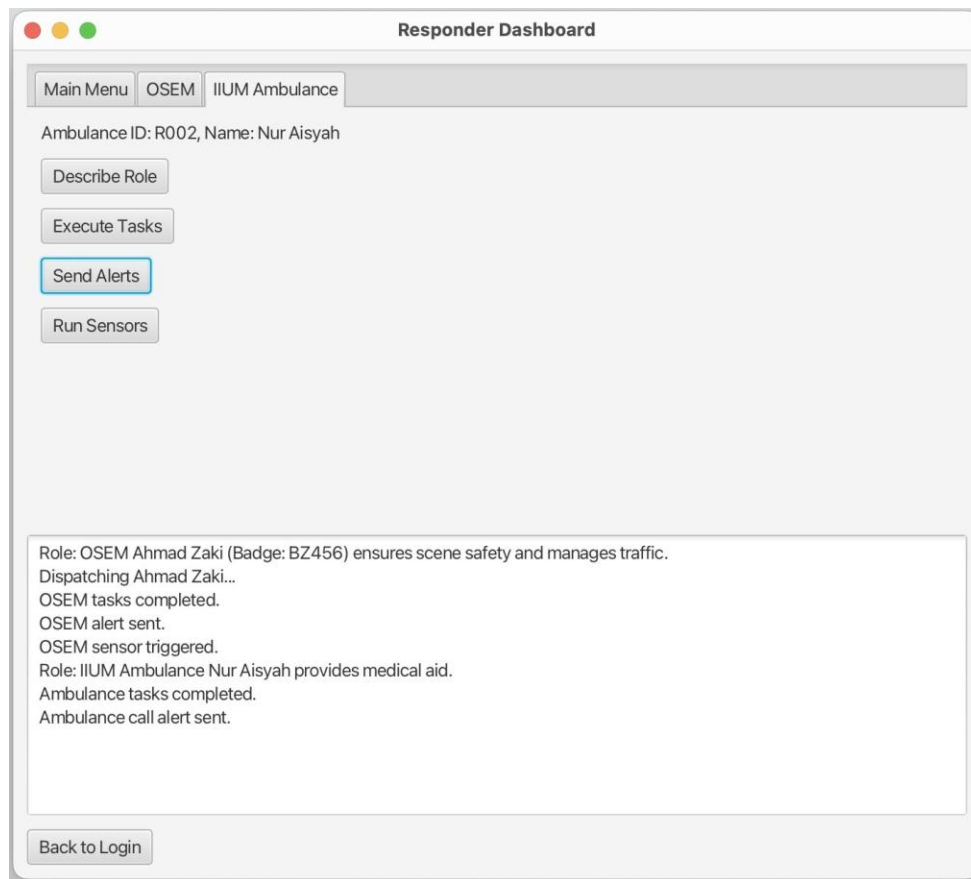


Figure 24: Send Alerts button clicked in IIUMAmbulance interface

If the user clicks on the “Send Alerts” button, the system will notify the relevant authorities, such as medical officers to make an action. The message of alert sent will be displayed to user.

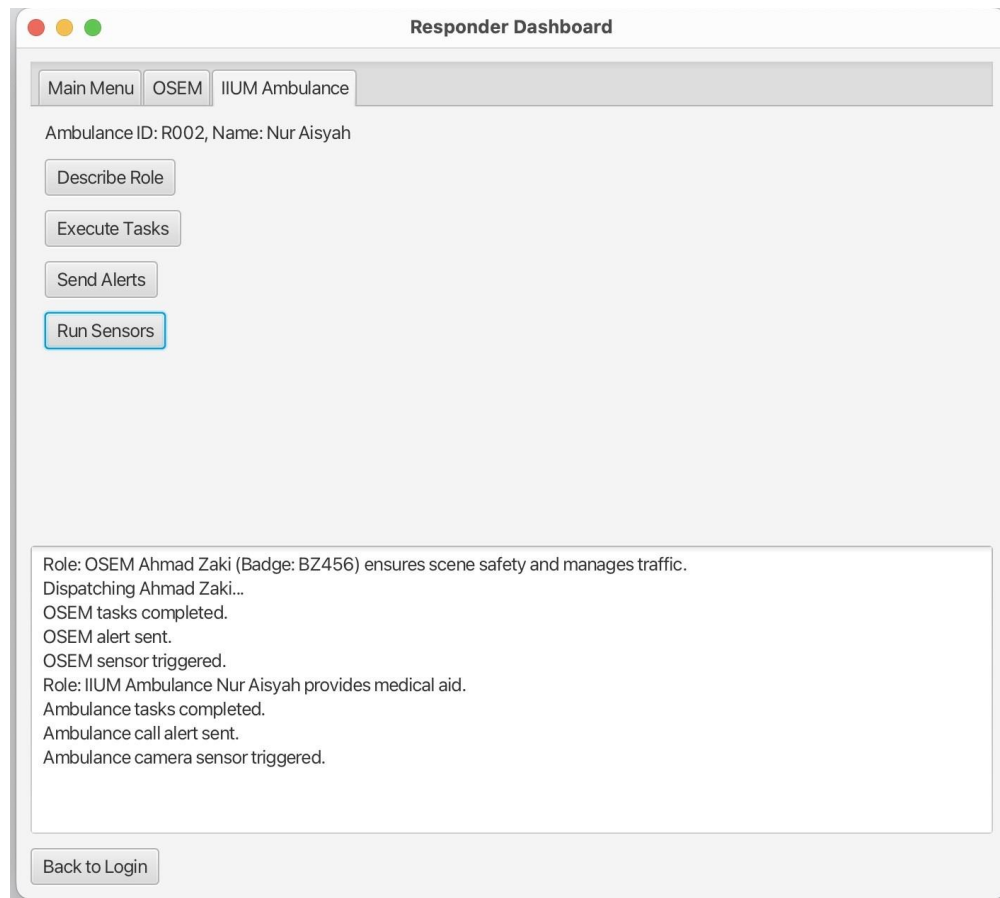


Figure 25: Run Sensors button clicked in IIUMAmbulance interface

If the user clicks on the “Run Sensors” button, it will activate the ambulance camera sensor for monitoring and the system will display a message to inform the user of its action.

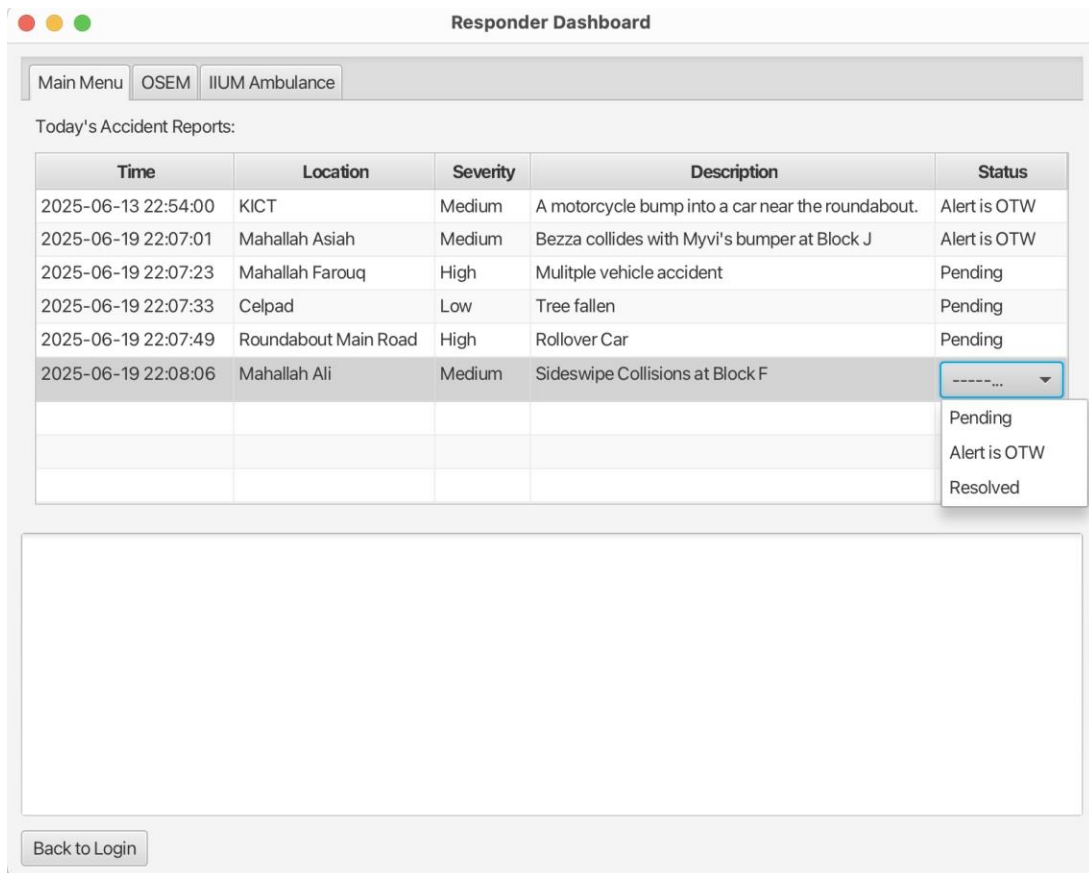


Figure 26: Main menu with new accident reports updated

Once an accident has been reported, it will appear at the Responder dashboard to give alert to the responder responsible to handle the case. Responder can also update the status of the accident for the user to keep track of with the dropdown button options: "Pending", "Alert is OTW", and "Resolved".

2.0 JavaFx Coding

2.1 Login Window

```
package ui;

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;

public class LoginWindow {

    public static void display(Stage stage) {

        Label titleLabel = new Label("Accident Detection System Login");
        titleLabel.setStyle("-fx-font-size: 18px; -fx-font-weight: bold;");

        TextField usernameField = new TextField();
        usernameField.setPromptText("Username");

        PasswordField passwordField = new PasswordField();
        passwordField.setPromptText("Password");

        Button loginButton = new Button("Login");
```

```
Label messageLabel = new Label();
```

```
loginButton.setOnAction(e -> {
```

```
    String username = usernameField.getText().trim().toLowerCase();
```

```
    String password = passwordField.getText().trim();
```

```
    switch (username) {
```

```
        case "user":
```

```
            if (password.equals("123")) {
```

```
                UserDashboardUI.display(stage);
```

```
            } else {
```

```
                messageLabel.setText("Invalid password for user.");
```

```
            }
```

```
            break;
```

```
        case "admin":
```

```
            if (password.equals("admin")) {
```

```
                AdminDashboardUI.display(stage);
```

```
            } else {
```

```
                messageLabel.setText("Invalid password for admin.");
```

```
            }
```

```
            break;
```

```
        case "responder":
```

```

        if (password.equals("111")) {
            ResponderDashboardUI.display(stage);
        } else {
            messageLabel.setText("Invalid password for responder.");
        }
        break;
    default:
        messageLabel.setText("Invalid username or password.");
    }
});

VBox layout = new VBox(10);
layout.setAlignment(Pos.CENTER);
layout.setPadding(new Insets(20));

layout.getChildren().addAll(titleLabel, usernameField, passwordField, loginButton,
messageLabel);

Scene scene = new Scene(layout, 400, 300);
stage.setScene(scene);
stage.setTitle("Login");
stage.show();
}
}

```

2.2 User Dashboard

```
package ui;

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

public class UserDashboardUI {

    private static final String ACCIDENT_FILE = "accidents.txt";

    public static void display(Stage stage) {

        stage.setTitle("User Dashboard");

        Label titleLabel = new Label("Welcome to the User Dashboard");
```



```
titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");
```

```
TextArea accidentArea = new TextArea();
```

```
accidentArea.setEditable(false);
```

```
accidentArea.setWrapText(true);
```

```
accidentArea.setPrefHeight(300);
```

```
accidentArea.setText("Loading today's accidents...");
```

```
Button btnRefresh = new Button("Refresh");
```

```
btnRefresh.setOnAction(e -> loadAccidents(accidentArea));
```

```
Button btnReportAccident = new Button("Report New Accident");
```

```
btnReportAccident.setOnAction(e -> AccidentReportUI.display(stage));
```

```
Button btnBack = new Button("Back to Login");
```

```
btnBack.setOnAction(e -> LoginWindow.display(stage));
```

```
VBox layout = new VBox(15);
```

```
layout.setAlignment(Pos.CENTER);
```

```
layout.setPadding(new Insets(20));
```

```
layout.getChildren().addAll(  
    titleLabel,
```

```

        accidentArea,

        btnRefresh,

        btnReportAccident,

        btnBack

    );

    loadAccidents(accidentArea);

    Scene scene = new Scene(layout, 600, 500);
    stage.setScene(scene);
    stage.show();
}

private static void loadAccidents(TextArea area) {
    area.clear();

    try {
        List<String> lines = Files.readAllLines(Paths.get(ACCIDENT_FILE));
        if (lines.isEmpty()) {
            area.setText("No accidents reported today.");
            return;
        }
    }
}

```

```

StringBuilder sb = new StringBuilder();

sb.append("Today's Reported Accidents:\n\n");

for (int i = 0; i < lines.size(); i++) {
    if (lines.get(i).startsWith("Time: ")) {
        sb.append(lines.get(i)).append("\n"); // Time
        sb.append(lines.get(++i)).append("\n"); // Location
        sb.append(lines.get(++i)).append("\n"); // Severity
        sb.append(lines.get(++i)).append("\n"); // Description
        sb.append(lines.get(++i)).append("\n"); // Status
        sb.append("\n");
    }
}

area.setText(sb.toString());
} catch (IOException e) {
    area.setText("Failed to load accident data.");
}
}

```

2.3 Admin Dashboard

```
package ui;

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;

public class AdminDashboardUI {

    public static void display(Stage stage) {

        stage.setTitle("Admin Dashboard");

        Label titleLabel = new Label("Admin Dashboard");
        titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

        Button btnMonitorIncidents = new Button("Monitor Incidents");
        Button btnDispatchResponders = new Button("Dispatch Responders");
        Button btnViewResponderStatus = new Button("View Responder Status");
        Button btnBack = new Button("Back to Login");
```

```
btnMonitorIncidents.setOnAction(e -> {  
    Alert alert = new Alert(Alert.AlertType.INFORMATION);  
    alert.setTitle("Incidents Overview");  
    alert.setHeaderText("Real-Time Incident Monitoring");  
    alert.setContentText("- Accident at Gate 1\n- Severity: Critical\n- Status: Awaiting  
Response");  
    alert.showAndWait();  
});
```

```
btnDispatchResponders.setOnAction(e -> {  
    Alert alert = new Alert(Alert.AlertType.INFORMATION);  
    alert.setTitle("Dispatch Responders");  
    alert.setHeaderText("Responder Dispatched Successfully");  
    alert.setContentText("OSEM and IIUM Ambulance have been dispatched to the  
scene.");  
    alert.showAndWait();  
});
```

```
btnViewResponderStatus.setOnAction(e -> {  
    Alert alert = new Alert(Alert.AlertType.INFORMATION);  
    alert.setTitle("Responder Status");  
    alert.setHeaderText("Current Responder Activities");
```

```
        alert.setContentText("OSEM - Securing Area\nAmbulance - Transporting  
Victims");
```

```
        alert.showAndWait();
```

```
    });
```

```
    btnBack.setOnAction(e -> LoginWindow.display(stage));
```

```
    VBox layout = new VBox(15);
```

```
    layout.setAlignment(Pos.CENTER);
```

```
    layout.setPadding(new Insets(20));
```

```
    layout.getChildren().addAll(titleLabel, btnMonitorIncidents, btnDispatchResponders,  
    btnViewResponderStatus, btnBack);
```

```
    Scene scene = new Scene(layout, 600, 400);
```

```
    stage.setScene(scene);
```

```
    stage.show();
```

```
    }
```

```
}
```

2.4 Responder Dashboard

```
import responder.IIUMAmbulance;

import responder.OSEM;

import responder.Responder;

import alert.CallAlert;

import alert.SMSAlert;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.List;

import sensor.CameraSensor;

import sensor.IoTSensor;

import sensor.Sensor;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.*;

import javafx.scene.control.cell.ComboBoxTableCell;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.*;

import javafx.stage.Stage;

public class ResponderDashboardUI {
```

```

private static final String ACCIDENT_FILE = "accidents.txt";

public static class AccidentRecord {

    private final String time;

    private final String location;

    private String severity;

    private final String description;

    private String status;

    public AccidentRecord(String time, String location, String severity, String description,
String status) {

        this.time = time;

        this.location = location;

        this.severity = severity;

        this.description = description;

        this.status = status;

    }

    public String getTime() {

        return time;

    }

    public String getLocation() {

        return location;

```



```

    }

    public String getSeverity() {

        return severity;

    }

    public String getDescription() {

        return description;

    }

    public String getStatus() {

        return status;

    }


    public void setSeverity(String severity) {

        this.severity = severity;

    }

    public void setStatus(String status) {

        this.status = status;

    }

}


public static void display(Stage stage) {

    stage.setTitle("Responder Dashboard");

```

```
TextArea statusArea = new TextArea();
```

```
statusArea.setEditable(false);
```

```
statusArea.setWrapText(true);
```

```
statusArea.setPrefHeight(250);
```

```
OSEM osem = new OSEM("R001", "Ahmad Zaki", "013-1234567", "BZ456");
```

```
IIUMAmbulance ambulance = new IIUMAmbulance("R002", "Nur Aisyah", "012-9876543", "VAN3021", "STF9981");
```

```
// Tab 1: Main Menu
```

```
TableView<AccidentRecord> table = new TableView<>();
```

```
table.setEditable(true);
```

```
table.setPrefHeight(300);
```

```
TableColumn<AccidentRecord, String> colTime = new TableColumn<>("Time");
```

```
colTime.setCellValueFactory(new PropertyValueFactory<>("time"));
```

```
TableColumn<AccidentRecord, String> colLocation = new  
TableColumn<>("Location");
```

```
colLocation.setCellValueFactory(new PropertyValueFactory<>("location"));
```

```
TableColumn<AccidentRecord, String> colSeverity = new  
TableColumn<>("Severity");
```

```

colSeverity.setCellValueFactory(new PropertyValueFactory<>("severity"));

    TableColumn<AccidentRecord, String> colDescription = new
TableColumn<>("Description");

    colDescription.setCellValueFactory(new PropertyValueFactory<>("description"));

    TableColumn<AccidentRecord, String> colStatus = new TableColumn<>("Status");
    colStatus.setCellValueFactory(new PropertyValueFactory<>("status"));
    colStatus.setCellFactory(column -> {
        ComboBoxTableCell<AccidentRecord, String> cell = new
ComboBoxTableCell<>("Pending", "Alert is OTW", "Resolved");

        cell.setTooltip(new Tooltip("Click to update status"));

        return cell;
    });
    colStatus.setOnEditCommit(event -> {
        AccidentRecord record = event.getRowValue();
        record.setStatus(event.getNewValue());
        updateStatusInFile(record);
    });

    table.getColumns().addAll(colTime, colLocation, colSeverity, colDescription,
colStatus);

    loadAccidents(table);

```

```

table.setRowFactory(tv -> {

    TableRow<AccidentRecord> row = new TableRow<>();

    row.setOnMouseClicked(event -> {

        if (!row.isEmpty() && event.getClickCount() == 1) {

            table.edit(row.getIndex(), colStatus);

        }

    });

    return row;

});

```

```

VBox mainLayout = new VBox(10);

mainLayout.setPadding(new Insets(10));

mainLayout.getChildren().addAll(new Label("Today's Accident Reports:"), table);

```

```

// Tab 2: OSEM

```

```

VBox osemLayout = new VBox(10);

osemLayout.setPadding(new Insets(10));

```

```

Label osemInfo = new Label("OSEM ID: " + osem.responderID + ", Name: " +
osem.responderName);

```

```

Button describeOsem = new Button("Describe Role");

```

```

describeOsem.setOnAction(e -> {

    statusArea.appendText("Role: OSEM " + osem.responderName + " (Badge:
BZ456) ensures scene safety and manages traffic.\n");

});

```

```

Button taskOsem = new Button("Execute Tasks");

taskOsem.setOnAction(e -> {

    statusArea.appendText("Dispatching " + osem.responderName + "... \n");

    osem.dispatchResponder();

    osem.updateStatus("On Site");

    osem.secureAccidentArea();

    osem.controlTraffic();

    statusArea.appendText("OSEM tasks completed.\n");

});

```

```

Button alertOsem = new Button("Send Alerts");

alertOsem.setOnAction(e -> {

    SMSAlert sms = new SMSAlert(101, "Dispatcher OSEM", "013-1111111",
"Accident at East Gate");

    sms.sendAlert();

    statusArea.appendText("OSEM alert sent.\n");

});

```

```

Button sensorOsem = new Button("Run Sensors");

sensorOsem.setOnAction(e -> {

    Sensor sensor = new IoTSensor(201, "Gate A", "Heavy Impact");

    sensor.detectImpact();

    sensor.sendImpactData();

    statusArea.appendText("OSEM sensor triggered.\n");

});

osemLayout.getChildren().addAll(osemInfo, describeOsem, taskOsem, alertOsem,
sensorOsem);

// Tab 3: IIUM Ambulance

VBox ambLayout = new VBox(10);

ambLayout.setPadding(new Insets(10));

Label ambInfo = new Label("Ambulance ID: " + ambulance.responderID + ", Name:
" + ambulance.responderName);

Button describeAmb = new Button("Describe Role");

describeAmb.setOnAction(e -> {

    statusArea.appendText("Role: IIUM Ambulance " + ambulance.responderName
+ " provides medical aid.\n");

});

```

```
Button taskAmb = new Button("Execute Tasks");

taskAmb.setOnAction(e -> {

    ambulance.dispatchResponder();

    ambulance.updateStatus("Providing Aid");

    ambulance.provideFirstAid();

    ambulance.transportVictims();

    statusArea.appendText("Ambulance tasks completed.\n");

});
```

```
Button alertAmb = new Button("Send Alerts");

alertAmb.setOnAction(e -> {

    CallAlert call = new CallAlert(102, "Dispatcher Ambulance", "999", 30);

    call.sendAlert();

    statusArea.appendText("Ambulance call alert sent.\n");

});
```

```
Button sensorAmb = new Button("Run Sensors");

sensorAmb.setOnAction(e -> {

    CameraSensor cam = new CameraSensor(202, "North Entrance", "Car
Movement", true);

    cam.detectFromCamera();

    cam.captureImage();

});
```

```
cam.sendDataToAIEngine();  
  
statusArea.appendText("Ambulance camera sensor triggered.\n");  
  
});
```

```
ambLayout.getChildren().addAll(ambInfo, describeAmb, taskAmb, alertAmb,  
sensorAmb);
```

```
TabPane tabPane = new TabPane();  
  
tabPane.setTabClosingPolicy(TabPane.TabClosingPolicy.UNAVAILABLE);
```

```
Tab tabMain = new Tab("Main Menu", mainLayout);  
  
Tab tabOsem = new Tab("OSEM", osemLayout);  
  
Tab tabAmbulance = new Tab("IIUM Ambulance", ambLayout);
```

```
tabPane.getTabs().addAll(tabMain, tabOsem, tabAmbulance);
```

```
Button btnBack = new Button("Back to Login");  
  
btnBack.setOnAction(e -> LoginWindow.display(stage));
```

```
VBox mainContainer = new VBox(10);  
  
mainContainer.setPadding(new Insets(10));  
  
mainContainer.getChildren().addAll(tabPane, statusArea, btnBack);
```



```

    Scene scene = new Scene(mainContainer, 700, 600);

    stage.setScene(scene);

    stage.show();
}

private static void updateStatusInFile(AccidentRecord record) {

    try {

        List<String> lines = Files.readAllLines(Paths.get(ACCIDENT_FILE));

        String identifier = "Time: " + record.getTime();

        for (int i = 0; i < lines.size(); i++) {

            if (lines.get(i).equals(identifier)) {

                lines.set(i + 4, "Status: " + record.getStatus());

                break;

            }

        }

        Files.write(Paths.get(ACCIDENT_FILE), lines);

    } catch (IOException e) {

        showError("Failed to update status.");

    }

}

private static void showError(String msg) {

    Alert alert = new Alert(Alert.AlertType.ERROR);

```

```

    alert.setTitle("Error");

    alert.setHeaderText("Action Required");

    alert.setContentText(msg);

    alert.showAndWait();
}

```

```

private static void loadAccidents(Table<AccidentRecord> table) {

    table.getItems().clear();

    try {

        List<String> lines = Files.readAllLines(Paths.get(ACCIDENT_FILE));

        for (int i = 0; i < lines.size(); i++) {

            if (lines.get(i).startsWith("Time: ")) {

                String time = lines.get(i).substring(6);

                String location = lines.get(++i).substring(10);

                String severity = lines.get(++i).substring(10);

                String description = lines.get(++i).substring(13);

                String status = lines.get(++i).substring(8);

                table.getItems().add(new AccidentRecord(time, location, severity,
description, status));

            }

        }

    } catch (IOException ignored) {

    }

}

```

}}

2.5 Accident Report

```
package ui;

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;

import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class AccidentReportUI {

    public static void display(Stage stage) {

        stage.setTitle("Accident Reporting");

        Label titleLabel = new Label("Report New Accident");
        titleLabel.setStyle("-fx-font-size: 20px; -fx-font-weight: bold;");
```

```
TextField tfLocation = new TextField();

tfLocation.setPromptText("Enter accident location");


TextField tfSeverity = new TextField();

tfSeverity.setPromptText("Enter severity level (Low/Medium/High)");


TextArea tfDescription = new TextArea();

tfDescription.setPromptText("Describe the accident details");

tfDescription.setWrapText(true);

tfDescription.setPrefRowCount(4);


Label messageLabel = new Label();


Button btnSubmit = new Button("Submit Report");

btnSubmit.setOnAction(e -> {

    String location = tfLocation.getText();

    String severity = tfSeverity.getText();

    String description = tfDescription.getText();


    if (location.isEmpty() || severity.isEmpty() || description.isEmpty()) {

        messageLabel.setText("All fields are required!");

        return;
    }
}
```

```
}
```

```
try (FileWriter writer = new FileWriter("accidents.txt", true)) {  
    String time =  
    LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));  
    writer.write("Accident Reported:\n");  
    writer.write("Time: " + time + "\n");  
    writer.write("Location: " + location + "\n");  
    writer.write("Severity: " + severity + "\n");  
    writer.write("Description: " + description + "\n");  
    writer.write("-----\n");  
    messageLabel.setText("Accident reported successfully.");  
  
    tfLocation.clear();  
    tfSeverity.clear();  
    tfDescription.clear();  
} catch (IOException ex) {  
    messageLabel.setText("Error writing to file.");  
}  
});
```

```
Button btnBack = new Button("Back to Dashboard");  
btnBack.setOnAction(e -> UserDashboardUI.display(stage));
```

```

VBox layout = new VBox(10);

layout.setPadding(new Insets(20));

layout.setAlignment(Pos.CENTER);

layout.getChildren().addAll(
    titleLabel, tfLocation, tfSeverity, tfDescription, btnSubmit, messageLabel,
    btnBack
);

Scene scene = new Scene(layout, 500, 450);

stage.setScene(scene);

stage.show();
}
}

```

2.6 Main app

```
package main;

import javafx.application.Application;
import javafx.stage.Stage;
import ui.LoginWindow;

public class MainApp extends Application {

    @Override

    public void start(Stage stage) {

        LoginWindow.display(stage);

    }

    public static void main(String[] args) {

        launch(args);

    }

}
```

GitHub Link: <https://github.com/Sarah-Yas/IIUM-Accident-Detection-System.git>