

Common platforms for ports, terminals, custodians

# **JAVA SWING BASED –COMMON PLATFORMS FOR PORTS, TERMINALS, CUSTODIANS–SQL CONNECTIVITY USING JDBC**

*A Report*

*Submitted in partial fulfillment of the Requirements  
for the COURSE*

## **DATABASE MANAGEMENT SYSTEMS By**

**M.SAI RISHIK 1602-21-737-302**

**Under the guidance of Ms B. Leelavathy**



**Department of Information Technology  
Vasavi College of Engineering (Autonomous)  
(Affiliated to Osmania University)  
Ibrahimbagh, Hyderabad-31  
2022-2023**

## **BONAFIDE CERTIFICATE**

This is to certify that this project report titled  
***‘COMMON PLATFORM FOR PORTS,TERMINALS,CUSTODIANS’***  
is a project work of **M.SAI RISHIK** bearing roll no. 1602-21-737-  
302 who carried out this project under my supervision in the IV  
semester for the academic year 2022- 2023

Signature  
External Examiner

Signature  
Internal Examiner

## **ABSTRACT**

Common platforms have emerged as a transformative trend in the logistics industry, enabling ports, terminals, and custodians to streamline operations, enhance collaboration, and improve efficiency. These centralized digital ecosystems leverage advanced technologies, facilitate real-time data sharing, automate processes, and promote transparency, resulting in optimized resource allocation, reduced turnaround times.

## **Requirement Analysis**

### **List of Tables:**

-Terminal

-Ports

-Custodians

-Shipment

### **List of Attributes with their Domain Types:**

#### **Table: Ports**

- port\_id: INT (Primary Key)
- port\_name: VARCHAR(100) NOT NULL
- country: VARCHAR(50) NOT NULL

### Table: Terminals

- terminal\_id: INT (Primary Key)
- terminal\_name: VARCHAR(100) NOT NULL
- port\_id: INT (Foreign Key referencing port\_id in the Ports table)

### Table: Custodians

- 
- custodian\_id: INT (Primary Key)
- custodian\_name: VARCHAR(100) NOT NULL
- terminal\_id: INT (Foreign Key referencing terminal\_id in the Terminals table)

### Table: Shipments

- 
- shipment\_id: INT (Primary Key)
- shipment\_name: VARCHAR2(100) NOT NULL
- terminal\_id: INT (Foreign Key referencing terminal\_id in the Terminals table)
- shipment\_status: VARCHAR2(20) NOT NULL (with a constraint to allow only 'In Progress' or 'Completed' values)
- NULL VARCHAR2(10)

## AIM AND PRIORITY OF THE PROJECT

To create a **Java GUI-based** desktop application of port management system. It takes values like Ports, terminals, custodians, shipments ..etc through forms which are then updated in the database using JDBC connectivity.

## ARCHITECTURE AND TECHNOLOGY

### **Software used:**

Java, Oracle 11g Database, Java SE version 14, Run SQL.

### **Java SWING:**

**Java SWING** is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

## **JAVA FX:**

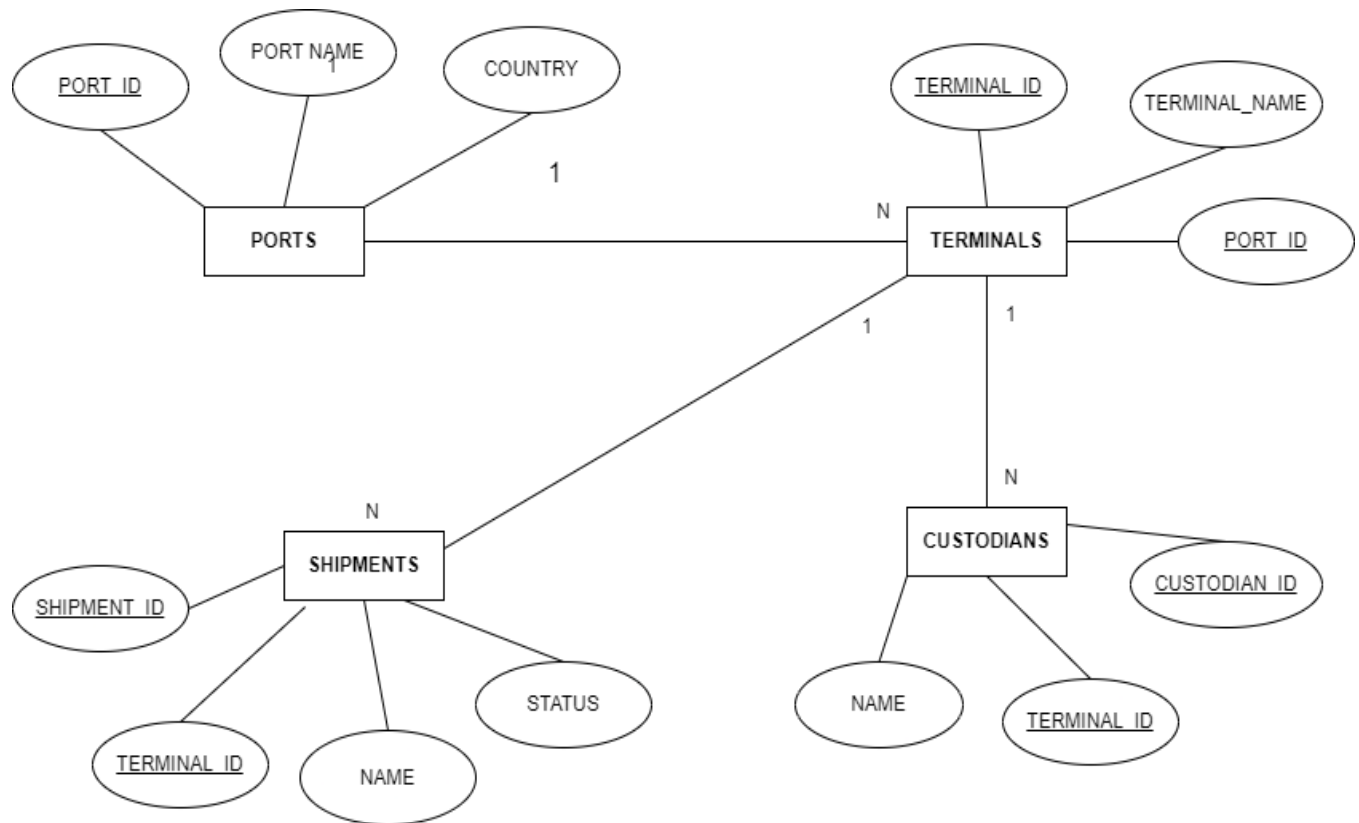
JavaFX is a versatile framework for creating visually appealing and interactive user interfaces in Java. With its extensive library of UI components, developers can design and customize buttons, labels, tables, and more. JavaFX's scene graph architecture facilitates efficient management of UI elements, event handling, and dynamic updates. Additionally, it supports advanced features such as animations and 3D graphics, making it suitable for developing desktop, mobile, and embedded applications with rich and engaging user experiences.

## **SQL:**

Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

# DESIGN

## Entity Relationship Diagram



## TABLE CREATED IN SQL:

TNAME	TABTYPE	CLUSTERID
-----	-----	-----
CUSTODIANS	TABLE	
PORTS	TABLE	
SHIPMENTS	TABLE	
TERMINALS	TABLE	

### 1.PORTS AND TERMINALS

```
SQL> CREATE TABLE Ports (  
  2   port_id INT PRIMARY KEY,  
  3   port_name VARCHAR(100) NOT NULL,  
  4   country VARCHAR(50) NOT NULL,  
  5   UNIQUE (port_name)  
  6 );
```

Table created.

```
SQL> CREATE TABLE Terminals (  
  2   terminal_id INT PRIMARY KEY,  
  3   terminal_name VARCHAR(100) NOT NULL,  
  4   port_id INT,  
  5   FOREIGN KEY (port_id) REFERENCES Ports(port_id)  
  6   ON DELETE CASCADE,  
  7   UNIQUE (terminal_name)  
  8 );
```

Table created.



```
SQL> DESC PORTS;
```

Name	Null?	Type
PORT_ID	NOT NULL	NUMBER(38)
PORT_NAME	NOT NULL	VARCHAR2(100)
COUNTRY	NOT NULL	VARCHAR2(50)

```
SQL> DESC TERMINALS;
```

Name	Null?	Type
TERMINAL_ID	NOT NULL	NUMBER(38)
TERMINAL_NAME	NOT NULL	VARCHAR2(100)
PORT_ID		NUMBER(38)

## 2.CUSTODIANS

```
SQL> CREATE TABLE Custodians (  
2     custodian_id INT PRIMARY KEY,  
3     custodian_name VARCHAR(100) NOT NULL,  
4     terminal_id INT,  
5     FOREIGN KEY (terminal_id) REFERENCES Terminals(terminal_id)  
6     ON DELETE CASCADE,  
7     UNIQUE (custodian_name)  
8 );
```

Table created.

```
SQL> DESC CUSTODIANS;
```

Name	Null?	Type
CUSTODIAN_ID	NOT NULL	NUMBER(38)
CUSTODIAN_NAME	NOT NULL	VARCHAR2(100)
TERMINAL_ID		NUMBER(38)

### 3.SHIPMENTS

```
SQL> CREATE TABLE Shipments (  
  2   shipment_id INT PRIMARY KEY,  
  3   shipment_name VARCHAR2(100) NOT NULL,  
  4   terminal_id INT,  
  5   FOREIGN KEY (terminal_id) REFERENCES Terminals(terminal_id)  
  6   ON DELETE CASCADE,  
  7   shipment_status VARCHAR2(20) NOT NULL,  
  8   CONSTRAINT shipment_status_check CHECK (shipment_status IN ('In Progr  
ess', 'Completed'))  
  9 );
```

Table created.

```
SQL> DESC SHIPMENTS;
```

Name	Null?	Type
SHIPMENT_ID	NOT NULL	NUMBER(38)
SHIPMENT_NAME	NOT NULL	VARCHAR2(100)
TERMINAL_ID		NUMBER(38)
SHIPMENT_STATUS	NOT NULL	VARCHAR2(20)

## DATABASE DESIGN:

TNAME	TABTYPE	CLUSTERID
CUSTODIANS	TABLE	
PORTS	TABLE	
SHIPMENTS	TABLE	
TERMINALS	TABLE	

SQL> DESC CUSTODIANS;

Name	Null?	Type
CUSTODIAN_ID	NOT NULL	NUMBER(38)
CUSTODIAN_NAME	NOT NULL	VARCHAR2(100)
TERMINAL_ID		NUMBER(38)

SQL> DESC PORTS;

Name	Null?	Type
PORT_ID	NOT NULL	NUMBER(38)
PORT_NAME	NOT NULL	VARCHAR2(100)
COUNTRY	NOT NULL	VARCHAR2(50)

SQL> DESC SHIPMENTS;

Name	Null?	Type
SHIPMENT_ID	NOT NULL	NUMBER(38)
SHIPMENT_NAME	NOT NULL	VARCHAR2(100)
TERMINAL_ID		NUMBER(38)
SHIPMENT_STATUS	NOT NULL	VARCHAR2(20)

SQL> DESC TERMINALS;

Name	Null?	Type
TERMINAL_ID	NOT NULL	NUMBER(38)
TERMINAL_NAME	NOT NULL	VARCHAR2(100)
PORT_ID		NUMBER(38)

## DML Operations on Ports

```
SQL> INSERT INTO Ports (port_id, port_name, country)
  2  VALUES (2, 'Port B', 'Country B');
```

1 row created.

```
SQL> UPDATE Ports
  2  SET country = 'New Country'
  3  WHERE port_id = 1;
```

1 row updated.

```
SQL> DELETE FROM Ports
  2  WHERE port_id = 2;
```

1 row deleted.

## DML Operations on Terminals

```
SQL> INSERT INTO Terminals (terminal_id, terminal_name, port_id)
  2  VALUES (2, 'Terminal Y', 1);
```

1 row created.

```
SQL> UPDATE Terminals
  2  SET terminal_name = 'New Terminal Name'
  3  WHERE terminal_id = 1;
```

1 row updated.

```
SQL> DELETE FROM Terminals
  2  WHERE terminal_id = 2;
```

1 row deleted.

## DML Operations on Shipments

```
1 row created.

SQL> INSERT INTO Shipments (shipment_id, shipment_name, terminal_id, shipment_status)
  2 VALUES (2, 'Shipment B', 1, 'Completed');

1 row created.

SQL> UPDATE Shipments
  2 SET shipment_status = 'Completed'
  3 WHERE shipment_id = 1;

1 row updated.

SQL> DELETE FROM Shipments
  2 WHERE shipment_id = 2;

1 row deleted.
```

## DML Operations on Custodians

```
SQL> INSERT INTO Custodians (custodian_id, custodian_name, terminal_id)
  2 VALUES (2, 'Custodian B', 1);

1 row created.

SQL> UPDATE Custodians
  2 SET custodian_name = 'New Custodian Name'
  3 WHERE custodian_id = 1;

1 row updated.

SQL> DELETE FROM Custodians
  2 WHERE custodian_id = 2;

1 row deleted.
```

# IMPLEMENTATION

## JAVA-SQL Connectivity using JDBC:

**Java Database Connectivity (JDBC)** is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

### DatabaseManager.java

```
import java.sql.*;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class DatabaseManager {
    private Connection connection;
    private final String DB_URL = "jdbc:oracle:thin:@localhost:1521:xe";
    private final String DB_USER = "rishik";
    private final String DB_PASSWORD = "1819";

    public void connect() throws SQLException {
        connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
        System.out.println("Connected to the database.");
    }

    public void disconnect() throws SQLException {
        if (connection != null && !connection.isClosed()) {
            connection.close();
            System.out.println("Disconnected from the database.");
        }
    }

    public List<Port> getAllPorts() throws SQLException {
```

Common platforms for ports, terminals, custodians

```
List<Port> ports = new ArrayList<>();
String query = "SELECT * FROM Ports";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
while (resultSet.next()) {
    int portId = resultSet.getInt("port_id");
    String portName = resultSet.getString("port_name");
    String country = resultSet.getString("country");
    Port port = new Port(portId, portName, country);
    ports.add(port);
}
return ports;
}

public List<Terminal> getAllTerminals() throws SQLException {
    List<Terminal> terminals = new ArrayList<>();
    String query = "SELECT * FROM Terminals";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(query);
    while (resultSet.next()) {
        int terminalId = resultSet.getInt("terminal_id");
        String terminalName = resultSet.getString("terminal_name");
        int portId = resultSet.getInt("port_id");
        Terminal terminal = new Terminal(terminalId, terminalName, portId);
        terminals.add(terminal);
    }
    return terminals;
}

public List<Custodian> getAllCustodians() throws SQLException {
    List<Custodian> custodians = new ArrayList<>();
    String query = "SELECT * FROM Custodians";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(query);
    while (resultSet.next()) {
        int custodianId = resultSet.getInt("custodian_id");
        String custodianName = resultSet.getString("custodian_name");
        int terminalId = resultSet.getInt("terminal_id");
        Custodian custodian = new Custodian(custodianId, custodianName, terminalId);
        custodians.add(custodian);
    }
    return custodians;
}

public List<Shipment> getAllShipments() throws SQLException {
    List<Shipment> shipments = new ArrayList<>();
    String query = "SELECT * FROM Shipments";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(query);
```

Common platforms for ports, terminals, custodians

```
while (resultSet.next()) {
    int shipmentId = resultSet.getInt("shipment_id");
    String shipmentName = resultSet.getString("shipment_name");
    int terminalId = resultSet.getInt("terminal_id");
    String shipmentStatus = resultSet.getString("shipment_status");
    Shipment shipment = new Shipment(shipmentId, shipmentName, terminalId, shipmentStatus);
    shipments.add(shipment);
}
return shipments;
}
```

// Ports

```
public void insertPort(int portId, String portName, String country) throws SQLException {
    String query = "INSERT INTO Ports (port_id, port_name, country) VALUES (?, ?, ?)";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, portId);
    statement.setString(2, portName);
    statement.setString(3, country);
    statement.executeUpdate();
}
```

```
public void updatePort(int portId, String portName, String country) throws SQLException {
    String query = "UPDATE Ports SET port_name = ?, country = ? WHERE port_id = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, portName);
    statement.setString(2, country);
    statement.setInt(3, portId);
    statement.executeUpdate();
}
```

```
public void deletePort(int portId) throws SQLException {
    String query = "DELETE FROM Ports WHERE port_id = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, portId);
    statement.executeUpdate();
}
```

// Custodians

```
public void insertCustodian(int custodianId, String custodianName, int terminalId) throws SQLException
{
    // Create the SQL query with placeholders for the parameters
    String query = "INSERT INTO custodians (custodian_id, custodian_name, terminal_id) VALUES
    (?, ?, ?)";
}
```



Common platforms for ports, terminals, custodians

```
// Prepare the statement and set the values for the parameters
PreparedStatement statement = connection.prepareStatement(query);
statement.setInt(1, custodianId);
statement.setString(2, custodianName);
statement.setInt(3, terminalId);

// Execute the query
statement.executeUpdate();
}
```

```
public void updateCustodian(int custodianId, String custodianName) throws SQLException {
    String query = "UPDATE Custodians SET custodian_name = ? WHERE custodian_id = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, custodianName);
    statement.setInt(2, custodianId);
    statement.executeUpdate();
}
```

```
public void deleteCustodian(int custodianId) throws SQLException {
    String query = "DELETE FROM Custodians WHERE custodian_id = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, custodianId);
    statement.executeUpdate();
}
```

// Terminals

```
public void insertTerminal(int terminalId, String terminalName, int portId) throws SQLException {
    String query = "INSERT INTO Terminals (terminal_id, terminal_name, port_id) VALUES (?, ?, ?)";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, terminalId);
    statement.setString(2, terminalName);
    statement.setInt(3, portId);
    statement.executeUpdate();
}
```

```
public void updateTerminal(int terminalId, String terminalName, int portId) throws SQLException {
    String query = "UPDATE Terminals SET terminal_name = ?, port_id = ? WHERE terminal_id = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, terminalName);
    statement.setInt(2, portId);
    statement.setInt(3, terminalId);
    statement.executeUpdate();
}
```

Common platforms for ports, terminals, custodians

```
public void deleteTerminal(int terminalId) throws SQLException {  
    String query = "DELETE FROM Terminals WHERE terminal_id = ?";  
    PreparedStatement statement = connection.prepareStatement(query);  
    statement.setInt(1, terminalId);  
    statement.executeUpdate();  
}
```

// Shipments

```
public void insertShipment(int shipmentId, String shipmentName, int terminalId, String shipmentStatus)  
throws SQLException {  
    String query = "INSERT INTO Shipments (shipment_id, shipment_name, terminal_id, shipment_status)  
VALUES (?, ?, ?, ?)";  
    PreparedStatement statement = connection.prepareStatement(query);  
    statement.setInt(1, shipmentId);  
    statement.setString(2, shipmentName);  
    statement.setInt(3, terminalId);  
    statement.setString(4, shipmentStatus);  
    statement.executeUpdate();  
}
```

```
public void updateShipment(int shipmentId, String shipmentName, int terminalId, String shipmentStatus)  
throws SQLException {  
    String query = "UPDATE Shipments SET shipment_name = ?, terminal_id = ?, shipment_status = ?  
WHERE shipment_id = ?";  
    PreparedStatement statement = connection.prepareStatement(query);  
    statement.setString(1, shipmentName);  
    statement.setInt(2, terminalId);  
    statement.setString(3, shipmentStatus);  
    statement.setInt(4, shipmentId);  
    statement.executeUpdate();  
}
```

```
public void deleteShipment(int shipmentId) throws SQLException {  
    String query = "DELETE FROM Shipments WHERE shipment_id = ?";  
    PreparedStatement statement = connection.prepareStatement(query);  
    statement.setInt(1, shipmentId);  
    statement.executeUpdate();  
}  
}
```

## Front-end Programs (User Interfaces):

### Main ui:

```
//javac -cp ojdbc6.jar MainUI.java PortsForm.java CustodiansForm.java TerminalsForm.java
ShipmentsForm.java DatabaseManager.java
//java -cp .;ojdbc6.jar MainUI
```

```
import javafx.animation.FadeTransition;
import java.util.Optional;
import javafx.scene.control.Alert.AlertType;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.text.FontWeight;
import javafx.util.Callback;
import javafx.stage.Stage;
import javafx.scene.Scene;

import javafx.geometry.Insets;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import javafx.scene.Scene;
import java.sql.SQLException;
import javafx.scene.layout.HBox;
import java.util.List;
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.MenuItem;
import javafx.scene.layout.BorderPane;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.util.Duration;
```

```
public class MainUI extends Application {
    private Stage primaryStage;
    //private VBox container;
```

## Common platforms for ports, terminals, custodians

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

@Override
public void start(Stage primaryStage) {
    this.primaryStage = primaryStage;

    primaryStage.setTitle("Port Management System");

    BorderPane root = new BorderPane();

    // Create and configure the animated text
    Text animatedText = new Text("Welcome to Port Management System");
    animatedText.setFont(Font.font("Arial", FontWeight.BOLD, 40));
    animatedText.setFill(Color.BLACK);

    // Create fade transition for text
    FadeTransition fadeTransition = new FadeTransition(Duration.seconds(3.5), animatedText);
    fadeTransition.setFromValue(0);
    fadeTransition.setToValue(1);
    fadeTransition.setCycleCount(1);

    fadeTransition.play();

    root.setCenter(animatedText);

    MenuBar menuBar = createMenuBar();
    root.setTop(menuBar);

    Scene scene = new Scene(root, 800, 600);
    scene.setFill(Color.DARKBLUE);
    primaryStage.setScene(scene);
    primaryStage.show();
}

private MenuBar createMenuBar() {
    MenuBar menuBar = new MenuBar();
    Menu dataMenu = new Menu("View Data");

    MenuItem displayMenuItem = new MenuItem("Display");
    displayMenuItem.setOnAction(e -> displayData());
    dataMenu.getItems().add(displayMenuItem);

    // Add other menu items for ports, terminals, custodians, shipments
```

## Common platforms for ports, terminals, custodians

```
        menuBar.getMenus().add(dataMenu);

Menu portMenu = new Menu("Ports");
MenuItem portMenuItem = new MenuItem("Add Port");
portMenuItem.setOnAction(e -> showPortsForm());
portMenu.getItems().add(portMenuItem);

Menu shipmentMenu = new Menu("Shipments");
MenuItem shipmentMenuItem = new MenuItem("Add Shipments");
shipmentMenuItem.setOnAction(e -> showShipmentsForm());
shipmentMenu.getItems().add(shipmentMenuItem);

Menu custodianMenu = new Menu("Custodians");
MenuItem custodianMenuItem = new MenuItem("Add Custodians");
custodianMenuItem.setOnAction(e -> showCustodiansForm());
custodianMenu.getItems().add(custodianMenuItem);

Menu terminalMenu = new Menu("Terminals");
MenuItem terminalMenuItem = new MenuItem("Add Terminals");
terminalMenuItem.setOnAction(e -> showTerminalsForm());
terminalMenu.getItems().add(terminalMenuItem);

menuBar.getMenus().addAll(portMenu, shipmentMenu, custodianMenu, terminalMenu);

return menuBar;
}

/*private void showPortsForm() {
    PortsForm portsForm = new PortsForm();
    Scene scene = new Scene(portsForm, 600, 400);
    Stage stage = new Stage();
    stage.setScene(scene);
    stage.setTitle("Manage Ports");
    stage.show();
}*/

private void showShipmentsForm() {
    ShipmentsForm shipmentsForm = new ShipmentsForm();
    Scene scene = new Scene(shipmentsForm, 600, 400);
    Stage stage = new Stage();
    stage.setScene(scene);
    stage.setTitle("Manage Shipments");

    stage.setOnHidden(e -> {
        primaryStage.show();
    });
}
```

## Common platforms for ports, terminals, custodians

```
        stage.close(); // Close the shipments form
    });

    stage.showAndWait(); // Show the shipments form and wait for it to close
}
private VBox container; // Declare container as a class member

private void displayData() {
    try {
        // Fetch the data from the database using the DatabaseManager class
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        List<Port> ports = databaseManager.getAllPorts();
        List<Terminal> terminals = databaseManager.getAllTerminals();
        List<Custodian> custodians = databaseManager.getAllCustodians();
        List<Shipment> shipments = databaseManager.getAllShipments();
        databaseManager.disconnect();

        // Create UI components to display the data
        TableView<Port> portTable = createPortTable(ports);
        TableView<Terminal> terminalTable = createTerminalTable(terminals);
        TableView<Custodian> custodianTable = createCustodianTable(custodians);
        TableView<Shipment> shipmentTable = createShipmentTable(shipments);

        // Create a VBox to hold the UI components
        VBox container = new VBox();
        container.setSpacing(10);
        container.setPadding(new Insets(10));

        // Create the back button
        Button backButton = new Button("Back");
        backButton.setOnAction(e -> showHome());
        container.getChildren().add(backButton);

        // Add the UI components to the container pane
        container.getChildren().addAll(portTable, terminalTable, custodianTable,
shipmentTable);

        // Create a scene and set it in the primaryStage
        Scene scene = new Scene(container, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.show();

    } catch (SQLException ex) {
        showErrorMessage("Error fetching data: " + ex.getMessage());
    }
}
```

Common platforms for ports, terminals, custodians

}

```
private TableView<Port> createPortTable(List<Port> ports) {
    TableView<Port> table = new TableView<>();

    TableColumn<Port, Integer> portIdColumn = new TableColumn<>("Port ID");
    portIdColumn.setCellValueFactory(new PropertyValueFactory<>("portId"));

    TableColumn<Port, String> portNameColumn = new TableColumn<>("Port Name");
    portNameColumn.setCellValueFactory(new PropertyValueFactory<>("portName"));

    TableColumn<Port, String> countryColumn = new TableColumn<>("Country");
    countryColumn.setCellValueFactory(new PropertyValueFactory<>("country"));

    TableColumn<Port, Void> actionsColumn = new TableColumn<>("Actions");
    actionsColumn.setCellFactory(param -> {
        return new TableCell<Port, Void>() {
            private final Button updateButton = new Button("Update");
            private final Button deleteButton = new Button("Delete");

            {
                updateButton.setOnAction(event -> {
                    Port port = (Port) getTableRow().getItem();
                    openPortForm(port);
                });

                deleteButton.setOnAction(event -> {
                    Port port = (Port) getTableRow().getItem();
                    deletePort(port);
                });
            }

            @Override
            protected void updateItem(Void item, boolean empty) {
                super.updateItem(item, empty);
                if (empty) {
                    setGraphic(null);
                } else {
                    setGraphic(new HBox(updateButton, deleteButton));
                }
            }
        };
    });
};
```

## Common platforms for ports, terminals, custodians

```
        table.getColumns().addAll(portIdColumn, portNameColumn, countryColumn,
actionsColumn);
        table.getItems().addAll(ports);

        return table;
    }
    // Inside the MainUI class

    private void openPortForm(Port port) {
        PortsForm portsForm = new PortsForm(port);
        portsForm.setOwnerStage(primaryStage); // Set the owner stage
        portsForm.setTitle("Update Port");
        portsForm.show();
    }

    private void deletePort(Port port) {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Confirmation");
        alert.setHeaderText("Delete Port");
        alert.setContentText("Are you sure you want to delete the port?");

        Optional<ButtonType> result = alert.showAndWait();
        if (result.isPresent() && result.get() == ButtonType.OK) {
            try {
                DatabaseManager databaseManager = new DatabaseManager();
                databaseManager.connect();
                databaseManager.deletePort(port.getPortId());
                databaseManager.disconnect();
                showInformationMessage("Port deleted successfully.");
                displayData();
            } catch (SQLException ex) {
                showErrorMessage("Error deleting port: " + ex.getMessage());
            }
        }
    }

    private void showInformationMessage(String message) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information");
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```



## Common platforms for ports, terminals, custodians

```
private void showErrorMessage(String message) {  
    Alert alert = new Alert(AlertType.ERROR);  
    alert.setTitle("Error");  
    alert.setHeaderText(null);  
    alert.setContentText(message);  
    alert.showAndWait();  
}
```

```
private void showCustodiansForm() {  
    CustodiansForm custodiansForm = new CustodiansForm();  
    Scene scene = new Scene(custodiansForm, 600, 400);  
    Stage stage = new Stage();  
    stage.setScene(scene);  
    stage.setTitle("Manage Custodians");  
    stage.show();  
}
```

```
private void showTerminalsForm() {  
    TerminalsForm terminalsForm = new TerminalsForm();  
    Scene scene = new Scene(terminalsForm, 600, 400);  
    Stage stage = new Stage();  
    stage.setScene(scene);  
    stage.setTitle("Manage Terminals");  
    stage.show();  
}
```

// Inside the MainUI class

```
private void openTerminalForm(Terminal terminal) {  
    // Create a new instance of the TerminalsForm  
    TerminalsForm terminalsForm = new TerminalsForm(terminal);  
  
    // Set the title for the form  
    terminalsForm.setTitle("Update Terminal");  
  
    // Show the form  
    terminalsForm.show();  
  
    // Refresh the terminal table after the form is closed  
    if (terminalsForm.isFormSubmitted()) {  
        displayData();  
    }  
}
```

## Common platforms for ports, terminals, custodians

```
}

private void deleteTerminal(Terminal terminal) {
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle("Confirmation");
    alert.setHeaderText("Delete Terminal");
    alert.setContentText("Are you sure you want to delete the terminal?");

    Optional<ButtonType> result = alert.showAndWait();
    if (result.isPresent() && result.get() == ButtonType.OK) {
        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.deleteTerminal(terminal.getTerminalId());
            databaseManager.disconnect();
            showInformationMessage("Terminal deleted successfully.");
            displayData();
        } catch (SQLException ex) {
            showErrorMessage("Error deleting terminal: " + ex.getMessage());
        }
    }
}

private void openCustodianForm(Custodian custodian) {
    // Create a new instance of the CustodiansForm
    CustodiansForm custodiansForm = new CustodiansForm(custodian);

    // Set the title for the form
    custodiansForm.setTitle("Update Custodian");

    // Show the form
    custodiansForm.show();

    // Refresh the custodian table after the form is closed
    if (custodiansForm.isFormSubmitted()) {
        displayData();
    }
}

private void deleteCustodian(Custodian custodian) {
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle("Confirmation");
    alert.setHeaderText("Delete Custodian");
    alert.setContentText("Are you sure you want to delete the custodian?");
}
```

## Common platforms for ports, terminals, custodians

```
Optional<ButtonType> result = alert.showAndWait();
if (result.isPresent() && result.get() == ButtonType.OK) {
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.deleteCustodian(custodian.getCustodianId());
        databaseManager.disconnect();
        showInformationMessage("Custodian deleted successfully.");
        displayData();
    } catch (SQLException ex) {
        showErrorMessage("Error deleting custodian: " + ex.getMessage());
    }
}

private void openShipmentForm(Shipment shipment) {
    // Create a new instance of the ShipmentsForm
    ShipmentsForm shipmentsForm = new ShipmentsForm(shipment);

    // Set the title for the form
    shipmentsForm.setTitle("Update Shipment");

    // Show the form
    shipmentsForm.show();

    // Refresh the shipment table after the form is closed
    if (shipmentsForm.isFormSubmitted()) {
        displayData();
    }
}

private void deleteShipment(Shipment shipment) {
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle("Confirmation");
    alert.setHeaderText("Delete Shipment");
    alert.setContentText("Are you sure you want to delete the shipment?");

    Optional<ButtonType> result = alert.showAndWait();
    if (result.isPresent() && result.get() == ButtonType.OK) {
        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.deleteShipment(shipment.getShipmentId());
            databaseManager.disconnect();
            showInformationMessage("Shipment deleted successfully.");
            displayData();
        }
    }
}
```

Common platforms for ports, terminals, custodians

```
    } catch (SQLException ex) {
        showErrorMessag("Error deleting shipment: " + ex.getMessage());
    }
}

private TableView<Terminal> createTerminalTable(List<Terminal> terminals) {
    TableView<Terminal> table = new TableView<>();

    TableColumn<Terminal, Integer> terminalIdColumn = new TableColumn<>("Terminal
ID");
    terminalIdColumn.setCellValueFactory(new PropertyValueFactory<>("terminalId"));

    TableColumn<Terminal, String> terminalNameColumn = new
TableColumn<>("Terminal Name");
    terminalNameColumn.setCellValueFactory(new
PropertyValueFactory<>("terminalName"));

    TableColumn<Terminal, Void> actionsColumn = new TableColumn<>("Actions");
    actionsColumn.setCellFactory(new Callback<TableColumn<Terminal, Void>,
TableCell<Terminal, Void>>() {
        @Override
        public TableCell<Terminal, Void> call(TableColumn<Terminal, Void> param) {
            return new TableCell<Terminal, Void>() {
                private final Button updateButton = new Button("Update");
                private final Button deleteButton = new Button("Delete");

                {
                    updateButton.setOnAction(event -> {
                        Terminal terminal = (Terminal)
getTableRow().getItem();

                        openTerminalForm(terminal);
                    });

                    deleteButton.setOnAction(event -> {
                        Terminal terminal = (Terminal)
getTableRow().getItem();

                        deleteTerminal(terminal);
                    });
                }

                @Override
                protected void updateItem(Void item, boolean empty) {
                    super.updateItem(item, empty);
                    if (empty) {
                        setGraphic(null);
                    }
                }
            };
        }
    });
}
```

Common platforms for ports, terminals, custodians

```

        } else {
            setGraphic(new HBox(updateButton,
deleteButton));
        }
    };
}

});

table.getColumnModel().addAll(terminalIdColumn, terminalNameColumn, actionsColumn);
table.getItems().addAll(terminals);

return table;
}

private TableView<Custodian> createCustodianTable(List<Custodian> custodians) {
    TableView<Custodian> table = new TableView<>();

    TableColumn<Custodian, Integer> custodianIdColumn = new
TableColumn<>("Custodian ID");
    custodianIdColumn.setCellValueFactory(new PropertyValueFactory<>("custodianId"));

    TableColumn<Custodian, String> custodianNameColumn = new
TableColumn<>("Custodian Name");
    custodianNameColumn.setCellValueFactory(new
PropertyValueFactory<>("custodianName"));

    TableColumn<Custodian, Integer> terminalIdColumn = new TableColumn<>("Terminal
ID");
    terminalIdColumn.setCellValueFactory(new PropertyValueFactory<>("terminalId"));

    TableColumn<Custodian, Void> actionsColumn = new TableColumn<>("Actions");
    actionsColumn.setCellFactory(param -> {
        return new TableCell<Custodian, Void>() {
            private final Button updateButton = new Button("Update");
            private final Button deleteButton = new Button("Delete");

            {
                updateButton.setOnAction(event -> {
                    Custodian custodian = (Custodian)
getTableRow().getItem();

                    openCustodianForm(custodian);
                });

                deleteButton.setOnAction(event -> {

```

Common platforms for ports, terminals, custodians

```
getTableRow().getItem();

Custodian custodian = (Custodian)
deleteCustodian(custodian);
});
}

@Override
protected void updateItem(Void item, boolean empty) {
    super.updateItem(item, empty);
    if (empty) {
        setGraphic(null);
    } else {
        setGraphic(new HBox(updateButton, deleteButton));
    }
}

});

table.getColumns().addAll(custodianIdColumn, custodianNameColumn,
terminalIdColumn, actionsColumn);
table.getItems().addAll(custodians);

return table;
}

private TableView<Shipment> createShipmentTable(List<Shipment> shipments) {
    TableView<Shipment> table = new TableView<>();

    TableColumn<Shipment, Integer> shipmentIdColumn = new TableColumn<>("Shipment
ID");
    shipmentIdColumn.setCellValueFactory(new PropertyValueFactory<>("shipmentId"));

    TableColumn<Shipment, String> shipmentNameColumn = new
TableColumn<>("Shipment Name");
    shipmentNameColumn.setCellValueFactory(new
PropertyValueFactory<>("shipmentName"));

    TableColumn<Shipment, Integer> terminalIdColumn = new TableColumn<>("Terminal
ID");
    terminalIdColumn.setCellValueFactory(new PropertyValueFactory<>("terminalId"));

    TableColumn<Shipment, String> shipmentStatusColumn = new
TableColumn<>("Shipment Status");
    shipmentStatusColumn.setCellValueFactory(new
PropertyValueFactory<>("shipmentStatus"));
```

## Common platforms for ports, terminals, custodians

```
TableColumn<Shipment, Void> actionsColumn = new TableColumn<>("Actions");
actionsColumn.setCellFactory(param -> {
    return new TableCell<Shipment, Void>() {
        private final Button updateButton = new Button("Update");
        private final Button deleteButton = new Button("Delete");

        {
            updateButton.setOnAction(event -> {
                Shipment shipment = (Shipment)
getTableRow().getItem();

                openShipmentForm(shipment);
            });

            deleteButton.setOnAction(event -> {
                Shipment shipment = (Shipment)
getTableRow().getItem();

                deleteShipment(shipment);
            });
        }

        @Override
        protected void updateItem(Void item, boolean empty) {
            super.updateItem(item, empty);
            if (empty) {
                setGraphic(null);
            } else {
                setGraphic(new HBox(updateButton,
deleteButton));
            }
        }
    };
});

table.getColumns().addAll(shipmentIdColumn, shipmentNameColumn,
terminalIdColumn, shipmentStatusColumn, actionsColumn);
table.getItems().addAll(shipments);

return table;
}
private void showPortsForm() {
    PortsForm portsForm = new PortsForm();
    Scene scene = new Scene(portsForm, 600, 400);
    Stage stage = new Stage();
    stage.setScene(scene);
```

Common platforms for ports, terminals, custodians

```
        stage.setTitle("Manage Ports");

        // Set the owner stage for portsForm
        portsForm.setOwnerStage(primaryStage);

        stage.show();
    }
    private void showHome() {
        // Recreate the initial scene and set it in the primaryStage
        primaryStage.setScene(createHomePage());
    }

    private Scene createHomePage() {
        BorderPane root = new BorderPane();

        // Create and configure the animated text
        Text animatedText = new Text("Welcome to Port Management System");
        animatedText.setFont(Font.font("Arial", FontWeight.BOLD, 40));
        animatedText.setFill(Color.BLACK);

        // Create fade transition for text
        FadeTransition fadeTransition = new FadeTransition(Duration.seconds(3.5),
animatedText);
        fadeTransition.setFromValue(0);
        fadeTransition.setToValue(1);
        fadeTransition.setCycleCount(1);

        fadeTransition.play();

        root.setCenter(animatedText);

        // Create the back button
        Button backButton = new Button("Back");
        backButton.setOnAction(e -> showHome());
        root.setBottom(backButton);

        MenuBar menuBar = createMenuBar();
        root.setTop(menuBar);

        Scene scene = new Scene(root, 800, 600);
        scene.setFill(Color.DARKBLUE);

        return scene;
    }
}
```



## PortsForm.java

```
import javafx.geometry.Insets;
import javafx.scene.control.*;
import javafx.stage.Stage;
import javafx.scene.Scene;
import java.sql.SQLException;
import javafx.scene.layout.GridPane;
import javafx.scene.control.Alert.AlertType;

public class PortsForm extends GridPane {
    private TextField portIdField;
    private TextField portNameField;
    private TextField countryField;
    private Button addButton;

    private Port portToUpdate; // Variable to hold the port being updated
    private boolean formSubmitted = false;

    private Stage ownerStage;

    public PortsForm(Port port) {
        this(); // Call the default constructor to initialize the form

        // Set the port to update and fill the fields with its data
        portToUpdate = port;
        fillFieldsWithData(portToUpdate);

        // Change the button text to "Update" and update the event handler
        addButton.setText("Update");
        addButton.setOnAction(e -> updatePort());
    }

    public PortsForm() {
        setPadding(new Insets(10));
        setHgap(10);
        setVgap(10);

        Label portIdLabel = new Label("Port ID:");
        Label portNameLabel = new Label("Port Name:");
        Label countryLabel = new Label("Country:");
```

## Common platforms for ports, terminals, custodians

```
portIdField = new TextField();
portNameField = new TextField();
countryField = new TextField();
addButton = new Button("Add");

add(portIdLabel, 0, 0);
add(portIdField, 1, 0);
add(portNameLabel, 0, 1);
add(portNameField, 1, 1);
add(countryLabel, 0, 2);
add(countryField, 1, 2);
add(addButton, 0, 3);

addButton.setOnAction(e -> addPort());
}

private void addPort() {
    int portId = Integer.parseInt(portIdField.getText());
    String portName = portNameField.getText();
    String country = countryField.getText();

    // Perform the add operation
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.insertPort(portId, portName, country);
        databaseManager.disconnect();
        clearFields();
        showSuccessMessage("Port added successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error adding port: " + ex.getMessage());
    }
}

public void setTitle(String title) {
    // Set the title of the form
    if (ownerStage != null) {
        ownerStage.setTitle(title);
    }
}

public void setOwnerStage(Stage ownerStage) {
    this.ownerStage = ownerStage;
}

public void show() {
```

## Common platforms for ports, terminals, custodians

```
// Show the form
Stage stage = new Stage();
Scene scene = new Scene(this);
stage.setScene(scene);
stage.show();
}

public boolean isFormSubmitted() {
    // Return whether the form was submitted or not
    return formSubmitted;
}

private void updatePort() {
    // Check if the port to update is set
    if (portToUpdate == null) {
        return;
    }

    // Get the updated values from the form fields
    int portId = Integer.parseInt(portIdField.getText());
    String portName = portNameField.getText();
    String country = countryField.getText();

    // Update the port in the database
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.updatePort(portToUpdate.getPortId(), portName, country);
        databaseManager.disconnect();
        showSuccessMessage("Port updated successfully.");
    } catch (SQLException ex) {

        showErrorMessage("Error updating port: " + ex.getMessage());
    }

    // Close the form's window
    Stage stage = (Stage) getScene().getWindow();
    stage.close();
}

private void fillFieldsWithData(Port port) {
    portIdField.setText(String.valueOf(port.getPortId()));
    portNameField.setText(port.getPortName());
    countryField.setText(port.getCountry());
}
```

## Common platforms for ports, terminals, custodians

```
private void clearFields() {
    portIdField.clear();
    portNameField.clear();
    countryField.clear();
}

private void showSuccessMessage(String message) {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Success");
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();

    // Close the form's window
    Stage stage = (Stage) getScene().getWindow();
    stage.close();
}

private void showErrorMessage(String message) {
    Alert alert = new Alert(AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();
}
}
```

## CustoianForm.java

```
import javafx.geometry.Insets;
import javafx.scene.control.*;
import javafx.stage.Stage;
import javafx.scene.Scene;
import java.sql.SQLException;
import javafx.scene.layout.GridPane;
import javafx.scene.control.Alert.AlertType;

public class CustodiansForm extends GridPane {
    private TextField custodianIdField;
    private TextField custodianNameField;
    private Button addButton;
    private TextField terminalIdField;

    private Custodian custodianToUpdate; // Variable to hold the custodian being updated
    private boolean formSubmitted = false;

    private Stage ownerStage;

    public CustodiansForm(Custodian custodian) {
        this(); // Call the default constructor to initialize the form

        // Set the custodian to update and fill the fields with its data
        custodianToUpdate = custodian;
        fillFieldsWithData(custodianToUpdate);

        // Change the button text to "Update" and update the event handler
        addButton.setText("Update");
        addButton.setOnAction(e -> updateCustodian());
    }

    public CustodiansForm() {
        setPadding(new Insets(10));
        setHgap(10);
        setVgap(10);

        Label custodianIdLabel = new Label("Custodian ID:");
        Label custodianNameLabel = new Label("Custodian Name:");
        custodianIdField = new TextField();
        custodianNameField = new TextField();
        terminalIdField = new TextField();
        add(new Label("Terminal ID:"), 0, 2);
    }
}
```

## Common platforms for ports, terminals, custodians

```
        add(terminalIdField, 1, 2);
        addButton = new Button("Add");

        add(custodianIdLabel, 0, 0);
        add(custodianIdField, 1, 0);
        add(custodianNameLabel, 0, 1);
        add(custodianNameField, 1, 1);
        add(addButton, 0, 3);

        addButton.setOnAction(e -> addCustodian());
    }

    private void addCustodian() {
        int custodianId = Integer.parseInt(custodianIdField.getText());
        String custodianName = custodianNameField.getText();
        int terminalId = Integer.parseInt(terminalIdField.getText()); // Retrieve the terminal ID

        // Perform the add operation
        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.insertCustodian(custodianId, custodianName, terminalId); // Pass the
terminal ID to the insertCustodian() method
            databaseManager.disconnect();
            clearFields();
            showSuccessMessage("Custodian added successfully.");
        } catch (SQLException ex) {
            showErrorMessage("Error adding custodian: " + ex.getMessage());
        }
    }

    public void setTitle(String title) {
        // Set the title of the form
        if (ownerStage != null) {
            ownerStage.setTitle(title);
        }
    }

    public void setOwnerStage(Stage ownerStage) {
        this.ownerStage = ownerStage;
    }

    public void show() {
        // Show the form
        Stage stage = new Stage();
```

## Common platforms for ports, terminals, custodians

```
Scene scene = new Scene(this);
stage.setScene(scene);
stage.show();
}

public boolean isFormSubmitted() {
    // Return whether the form was submitted or not
    return formSubmitted;
}

private void updateCustodian() {
    // Check if the custodian to update is set
    if (custodianToUpdate == null) {
        return;
    }

    // Get the updated values from the form fields
    int custodianId = Integer.parseInt(custodianIdField.getText());
    String custodianName = custodianNameField.getText();

    // Update the custodian in the database
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.updateCustodian(custodianToUpdate.getCustodianId(),
custodianName);
        databaseManager.disconnect();
        showMessage("Custodian updated successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error updating custodian: " + ex.getMessage());
    }

    // Close the form's window
    Stage stage = (Stage) getScene().getWindow();
    stage.close();
}

private void fillFieldsWithData(Custodian custodian) {
    custodianIdField.setText(String.valueOf(custodian.getCustodianId()));
    custodianNameField.setText(custodian.getCustodianName());
}

private void clearFields() {
    custodianIdField.clear();
    custodianNameField.clear();
}
```

## Common platforms for ports, terminals, custodians

```
private void showSuccessMessage(String message) {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Success");
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();

    // Close the form's window
    Stage stage = (Stage) getScene().getWindow();
    stage.close();
}

private void showErrorMessage(String message) {
    Alert alert = new Alert(AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();
}
}
```



## ShipmentForm.java

```
public class ShipmentsForm extends GridPane {
    private TextField shipmentIdField;
    private TextField shipmentNameField;
    private TextField terminalIdField;
    private TextField shipmentStatusField;
    private Button addButton;
        private Stage ownerStage;
        private boolean formSubmitted = false;

    private Shipment shipmentToUpdate; // Variable to hold the shipment being updated

    public ShipmentsForm(Shipment shipment) {
        this(); // Call the default constructor to initialize the form

        // Set the shipment to update and fill the fields with its data
        shipmentToUpdate = shipment;
        fillFieldsWithData(shipmentToUpdate);

        // Change the button text to "Update" and update the event handler
        addButton.setText("Update");
        addButton.setOnAction(e -> updateShipment());
    }

    public ShipmentsForm() {
        setPadding(new Insets(10));
        setHgap(10);
        setVgap(10);

        Label shipmentIdLabel = new Label("Shipment ID:");
        Label shipmentNameLabel = new Label("Shipment Name:");
        Label terminalIdLabel = new Label("Terminal ID:");
        Label shipmentStatusLabel = new Label("Shipment Status:");
        shipmentIdField = new TextField();
        shipmentNameField = new TextField();
        terminalIdField = new TextField();
        shipmentStatusField = new TextField();
        addButton = new Button("Add");

        add(shipmentIdLabel, 0, 0);
        add(shipmentIdField, 1, 0);
        add(shipmentNameLabel, 0, 1);
```

Common platforms for ports, terminals, custodians

```
add(shipmentNameField, 1, 1);
add(terminalIdLabel, 0, 2);
add(terminalIdField, 1, 2);
add(shipmentStatusLabel, 0, 3);
add(shipmentStatusField, 1, 3);
add(addButton, 0, 4);

addButton.setOnAction(e -> addShipment());
}

private void addShipment() {
    int shipmentId = Integer.parseInt(shipmentIdField.getText());
    String shipmentName = shipmentNameField.getText();
    int terminalId = Integer.parseInt(terminalIdField.getText());
    String shipmentStatus = shipmentStatusField.getText();

    // Perform the add operation
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.insertShipment(shipmentId, shipmentName, terminalId, shipmentStatus);
        databaseManager.disconnect();
        clearFields();
        showSuccessMessage("Shipment added successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error adding shipment: " + ex.getMessage());
    }
}

private void updateShipment() {
    // Check if the shipment to update is set
    if (shipmentToUpdate == null) {
        return;
    }

    // Get the updated values from the form fields
    int shipmentId = Integer.parseInt(shipmentIdField.getText());
    String shipmentName = shipmentNameField.getText();
    int terminalId = Integer.parseInt(terminalIdField.getText());
    String shipmentStatus = shipmentStatusField.getText();

    // Update the shipment in the database
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
```

Common platforms for ports, terminals, custodians

```
databaseManager.updateShipment(shipmentToUpdate.getShipmentId(), shipmentName,  
terminalId, shipmentStatus);
```

```
        databaseManager.disconnect();  
        showMessage("Shipment updated successfully.");  
    } catch (SQLException ex) {  
        showError("Error updating shipment: " + ex.getMessage());  
    }  
  
    getScene().getWindow().hide(); // Close the form's window  
}
```

```
private void fillFieldsWithData(Shipment shipment) {  
    shipmentIdField.setText(String.valueOf(shipment.getShipmentId()));  
    shipmentNameField.setText(shipment.getShipmentName());  
    terminalIdField.setText(String.valueOf(shipment.getTerminalId()));  
    shipmentStatusField.setText(shipment.getShipmentStatus());  
}
```

```
private void clearFields() {  
    shipmentIdField.clear();  
    shipmentNameField.clear();  
    terminalIdField.clear();  
    shipmentStatusField.clear();  
}
```

```
private void showMessage(String message) {  
    Alert alert = new Alert(Alert.AlertType.INFORMATION);  
    alert.setTitle("Success");  
    alert.setHeaderText(null);  
    alert.setContentText(message);  
    alert.showAndWait();  
    getScene().getWindow().hide(); // Close the form's window  
}
```

```
private void showError(String message) {  
    Alert alert = new Alert(Alert.AlertType.ERROR);  
    alert.setTitle("Error");  
    alert.setHeaderText(null);  
    alert.setContentText(message);  
    alert.showAndWait();  
}
```

```
    public void setTitle(String title) {  
        // Set the title of the form  
        if (ownerStage != null) {
```

Common platforms for ports, terminals, custodians

```
        ownerStage.setTitle(title);
    }
}

public void show() {
    Stage stage = new Stage();
    Scene scene = new Scene(this);
    stage.setScene(scene);
    stage.showAndWait();
}

public boolean isFormSubmitted() {
    // Return whether the form was submitted or not
    return formSubmitted;
}

    public void setOwnerStage(Stage ownerStage) {
        this.ownerStage = ownerStage;
    }

}
```

## TerminalForm.java

```
public class TerminalsForm extends GridPane {
    private TextField terminalIdField;
    private TextField terminalNameField;
    private TextField portIdField;
        private Stage ownerStage;

    private Button addButton;
        private boolean formSubmitted = false;

    private Terminal terminalToUpdate; // Variable to hold the terminal being updated

    public TerminalsForm(Terminal terminal) {
        this(); // Call the default constructor to initialize the form

        // Set the terminal to update and fill the fields with its data
        terminalToUpdate = terminal;
        fillFieldsWithData(terminalToUpdate);

        // Change the button text to "Update" and update the event handler
        addButton.setText("Update");
        addButton.setOnAction(e -> updateTerminal());
    }

    public TerminalsForm() {
        setPadding(new Insets(10));
        setHgap(10);
        setVgap(10);

        Label terminalIdLabel = new Label("Terminal ID:");
        Label terminalNameLabel = new Label("Terminal Name:");
        Label portIdLabel = new Label("Port ID:");
        terminalIdField = new TextField();
        terminalNameField = new TextField();
        portIdField = new TextField();
        addButton = new Button("Add");

        add(terminalIdLabel, 0, 0);
        add(terminalIdField, 1, 0);
        add(terminalNameLabel, 0, 1);
        add(terminalNameField, 1, 1);
    }
}
```

## Common platforms for ports, terminals, custodians

```
add(portIdLabel, 0, 2);
add(portIdField, 1, 2);
add(addButton, 0, 3);

addButton.setOnAction(e -> addTerminal());
}

private void addTerminal() {
    int terminalId = Integer.parseInt(terminalIdField.getText());
    String terminalName = terminalNameField.getText();
    int portId = Integer.parseInt(portIdField.getText());

    // Perform the add operation
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.insertTerminal(terminalId, terminalName, portId);

        databaseManager.disconnect();
        clearFields();
        showSuccessMessage("Terminal added successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error adding terminal: " + ex.getMessage());
    }
}

private void updateTerminal() {
    // Check if the terminal to update is set
    if (terminalToUpdate == null) {
        return;
    }

    // Get the updated values from the form fields
    int terminalId = Integer.parseInt(terminalIdField.getText());
    String terminalName = terminalNameField.getText();
    int portId = Integer.parseInt(portIdField.getText());

    // Update the terminal in the database
    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.updateTerminal(terminalToUpdate.getTerminalId(), terminalName, portId);

        databaseManager.disconnect();
    }
}
```

## Common platforms for ports, terminals, custodians

```
        showSuccessMessage("Terminal updated successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error updating terminal: " + ex.getMessage());
    }

    getScene().getWindow().hide(); // Close the form's window
}

private void fillFieldsWithData(Terminal terminal) {
    terminalIdField.setText(String.valueOf(terminal.getTerminalId()));
    terminalNameField.setText(terminal.getTerminalName());
    portIdField.setText(String.valueOf(terminal.getPortId()));
}

private void clearFields() {
    terminalIdField.clear();
terminalNameField.clear();
    portIdField.clear();
}

private void showSuccessMessage(String message) {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Success");
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();
    getScene().getWindow().hide(); // Close the form's window
}

private void showErrorMessage(String message) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();
}

    public void setOwnerStage(Stage ownerStage) {
        this.ownerStage = ownerStage;
    }

    public void setTitle(String title) {
        // Set the title of the form
        if (ownerStage != null) {
            ownerStage.setTitle(title);
        }
    }
```

Common platforms for ports, terminals, custodians

```
}
```

```
public void show() {  
    if (ownerStage != null) {  
        Scene scene = new Scene(this);  
        ownerStage.setScene(scene);  
        ownerStage.show();  
    }  
}
```

```
public boolean isFormSubmitted() {  
    // Return whether the form was submitted or not  
    return formSubmitted;  
}  
}
```



## Port.java

```
public class Port {
    private int portId;
    private String portName;
    private String country;

    public Port(int portId, String portName, String country) {
        this.portId = portId;
        this.portName = portName;
        this.country = country;
    }

    public int getPortId() {
        return portId;
    }

    public void setPortId(int portId) {
        this.portId = portId;
    }

    public String getPortName() {
        return portName;
    }

    public void setPortName(String portName) {
        this.portName = portName;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }
}
```

## Custodian.java

```
public class Custodian {
    private int custodianId;
    private String custodianName;
    private int terminalId;

    public Custodian(int custodianId, String custodianName, int terminalId) {
        this.custodianId = custodianId;
        this.custodianName = custodianName;
        this.terminalId = terminalId;
    }

    public int getCustodianId() {
        return custodianId;
    }

    public void setCustodianId(int custodianId) {
        this.custodianId = custodianId;
    }

    public String getCustodianName() {
        return custodianName;
    }

    public void setCustodianName(String custodianName) {
        this.custodianName = custodianName;
    }

    public int getTerminalId() {
        return terminalId;
    }

    public void setTerminalId(int terminalId) {
        this.terminalId = terminalId;
    }
}
```

## Terminal.java

```
public class Terminal {
    private int terminalId;
    private String terminalName;
    private int portId;
    private String location;
    public Terminal(int terminalId, String terminalName, int portId) {
        this.terminalId = terminalId;
        this.terminalName = terminalName;
        this.portId = portId;
    }

    public int getTerminalId() {
        return terminalId;
    }
    public void setTerminalId(int terminalId) {
        this.terminalId = terminalId;
    }

    public String getTerminalName() {
        return terminalName;
    }

    public void setTerminalName(String terminalName) {
        this.terminalName = terminalName;
    }

    public int getPortId() {
        return portId;
    }

    public void setPortId(int portId) {
        this.portId = portId;
    }
    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }
}
```

## Shipment.java

```
public class Shipment {
    private int shipmentId;
    private String shipmentName;
    private int terminalId;
    private String shipmentStatus;

    public Shipment(int shipmentId, String shipmentName, int terminalId, String shipmentStatus) {
        this.shipmentId = shipmentId;
        this.shipmentName = shipmentName;
        this.terminalId = terminalId;
        this.shipmentStatus = shipmentStatus;
    }

    public int getShipmentId() {
        return shipmentId;
    }

    public void setShipmentId(int shipmentId) {
        this.shipmentId = shipmentId;
    }

    public String getShipmentName() {
        return shipmentName;
    }

    public void setShipmentName(String shipmentName) {
        this.shipmentName = shipmentName;
    }

    public int getTerminalId() {
        return terminalId;
    }

    public void setTerminalId(int terminalId) {
        this.terminalId = terminalId;
    }

    public String getShipmentStatus() {
        return shipmentStatus;
    }
}
```

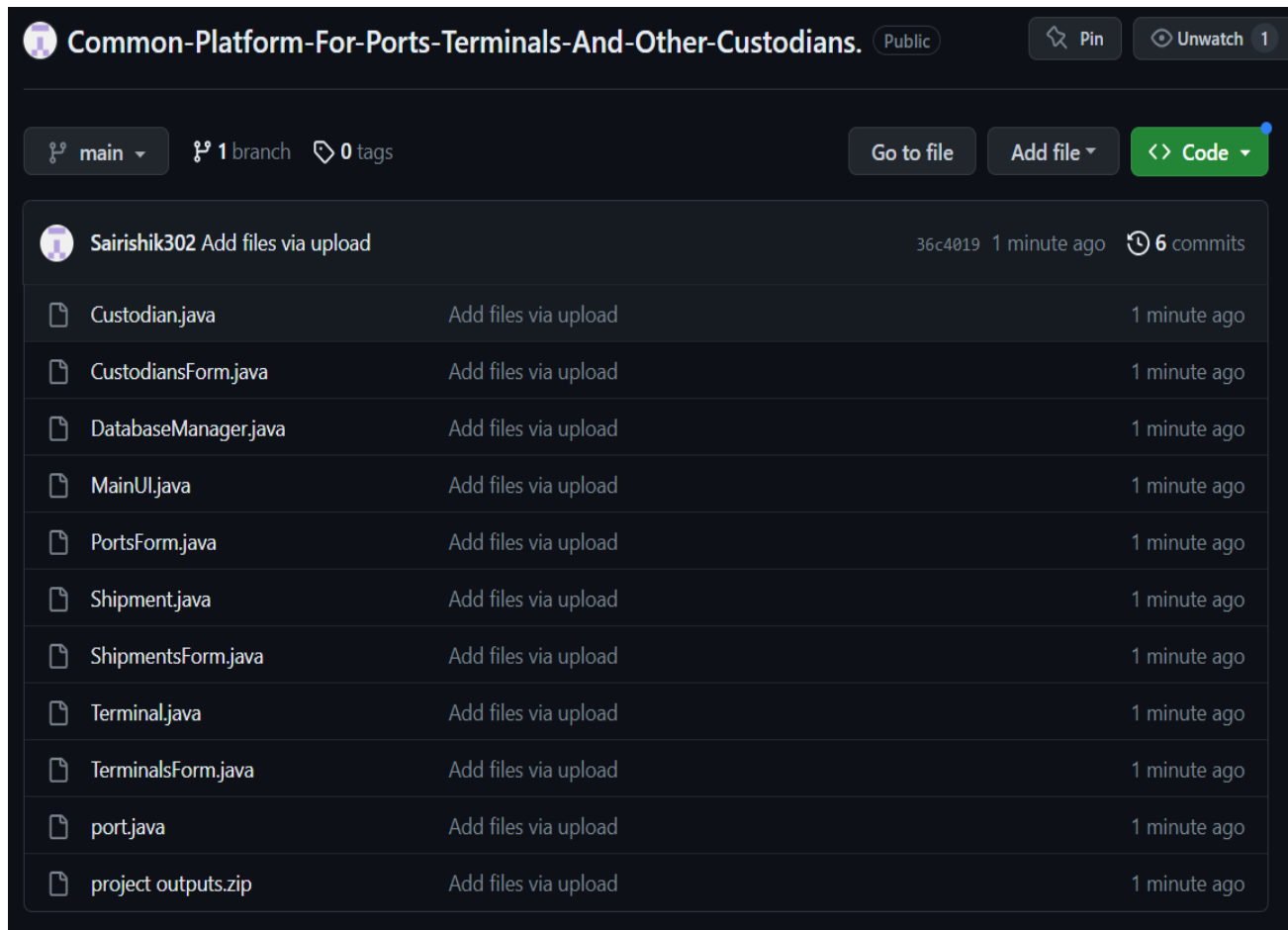
## Common platforms for ports, terminals, custodians

```
public void setShipmentStatus(String shipmentStatus) {  
    this.shipmentStatus = shipmentStatus;  
}  
}
```

## GitHub Links and Folder Structure

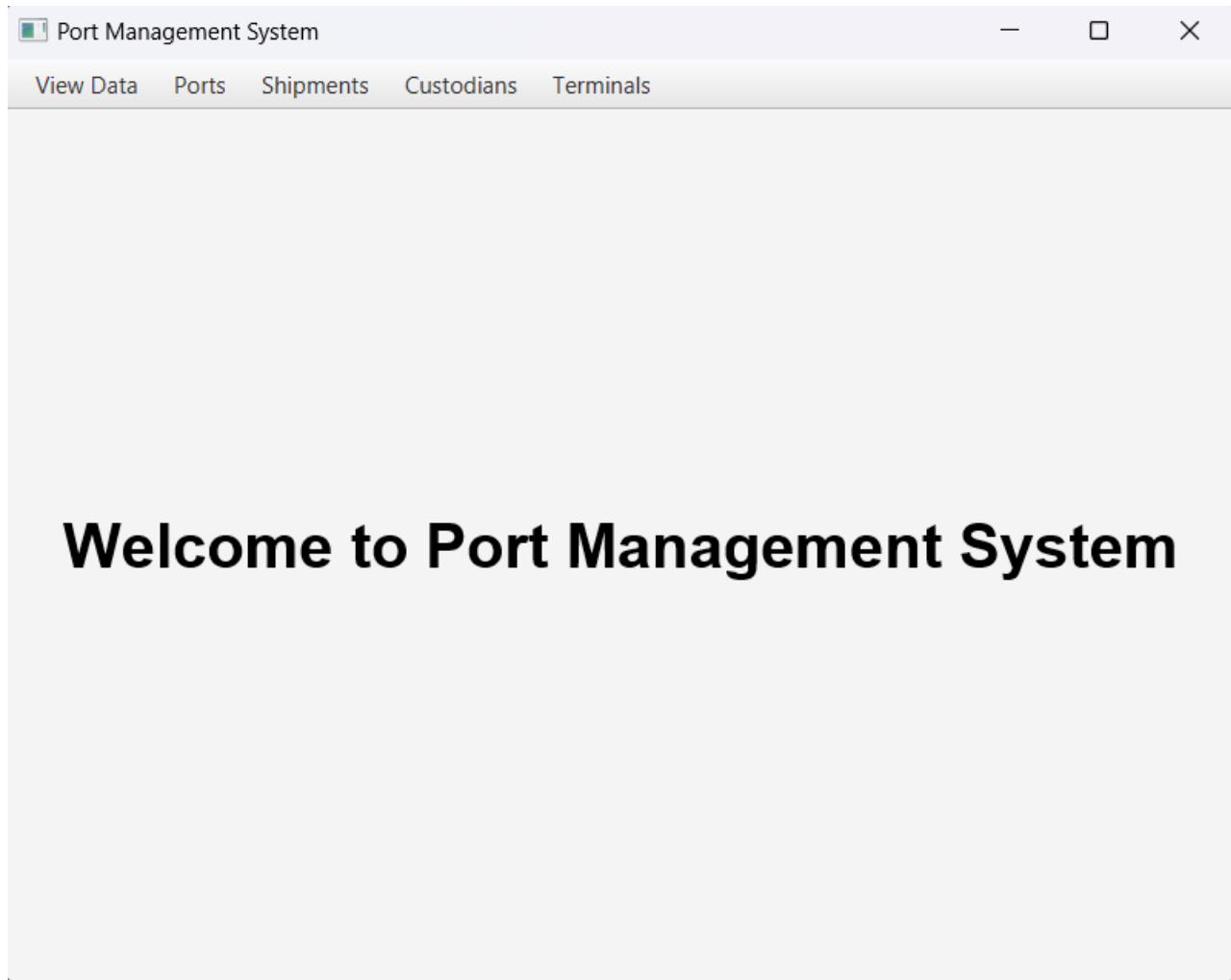
**GitHub link:** <https://github.com/Sairishik302/Common-Platform-For-Ports-Terminals-And-Other-Custodians./tree/main>

### Folder Structure:



## TESTING

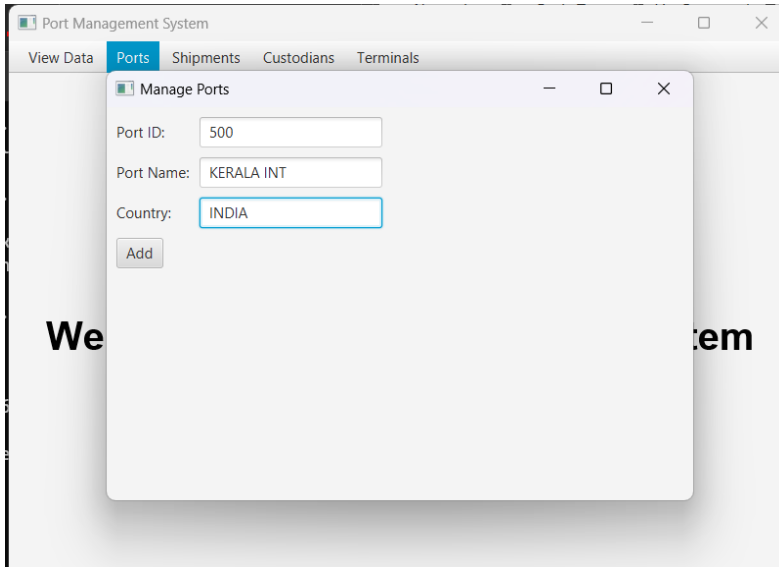
Main Interface:



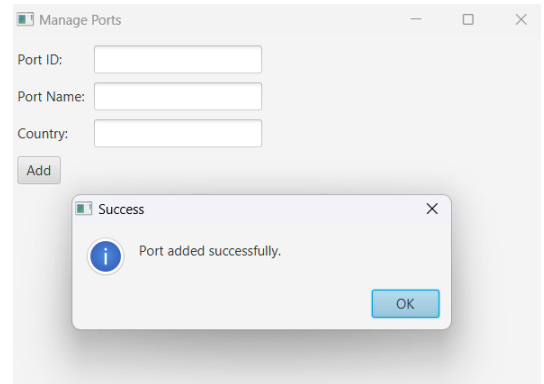
## Common platforms for ports, terminals, custodians

Insert operations : (Port,Terminal,Custodian,Shipment)

Port:

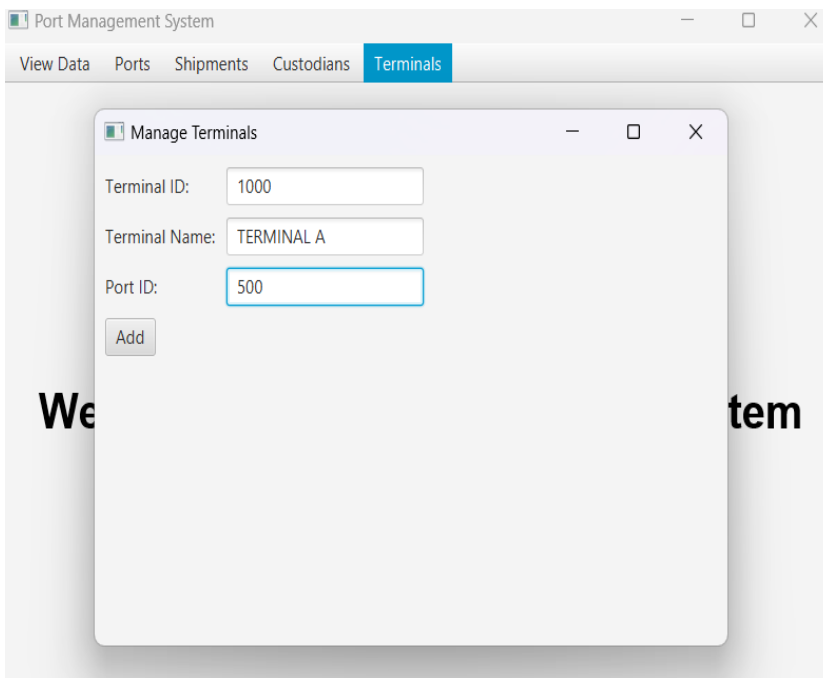


The screenshot shows the 'Port Management System' window with the 'Ports' tab selected. A 'Manage Ports' dialog box is open, containing the following fields: 'Port ID' with the value '500', 'Port Name' with the value 'KERALA INT', and 'Country' with the value 'INDIA'. There is an 'Add' button at the bottom left of the dialog box.

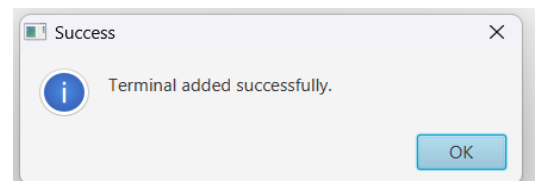


The screenshot shows the 'Manage Ports' dialog box with empty fields for 'Port ID', 'Port Name', and 'Country'. An 'Add' button is at the bottom left. A 'Success' dialog box is overlaid on top, displaying a blue information icon and the text 'Port added successfully.' with an 'OK' button at the bottom right.

Terminals:



The screenshot shows the 'Port Management System' window with the 'Terminals' tab selected. A 'Manage Terminals' dialog box is open, containing the following fields: 'Terminal ID' with the value '1000', 'Terminal Name' with the value 'TERMINAL A', and 'Port ID' with the value '500'. There is an 'Add' button at the bottom left of the dialog box.

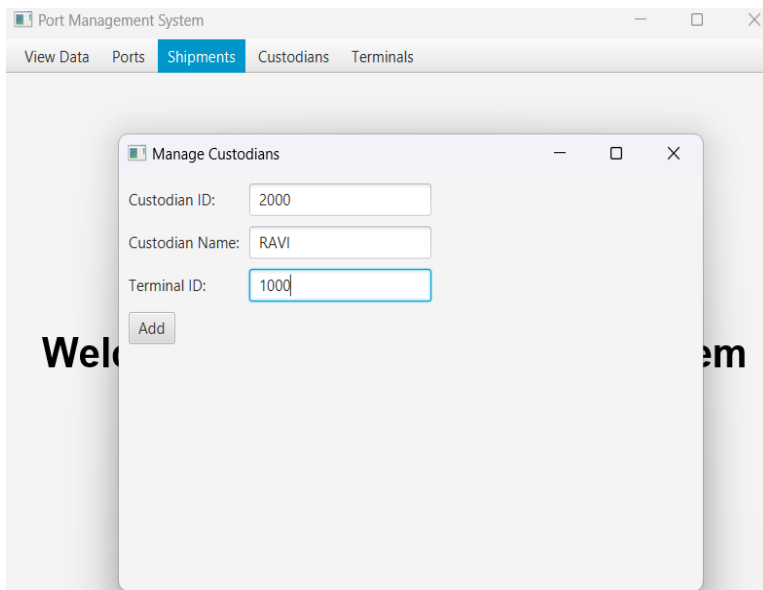


The screenshot shows the 'Manage Terminals' dialog box with empty fields for 'Terminal ID', 'Terminal Name', and 'Port ID'. An 'Add' button is at the bottom left. A 'Success' dialog box is overlaid on top, displaying a blue information icon and the text 'Terminal added successfully.' with an 'OK' button at the bottom right.

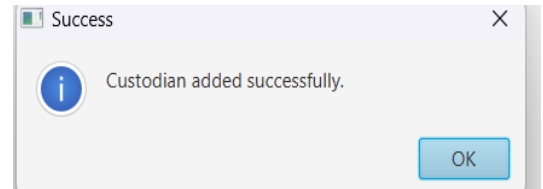


## Common platforms for ports, terminals, custodians

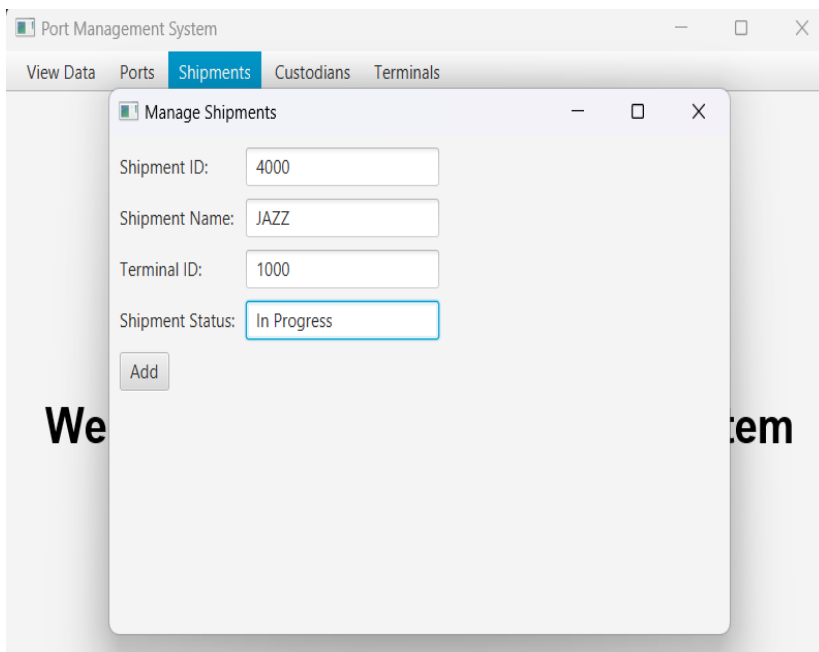
### Custodian:



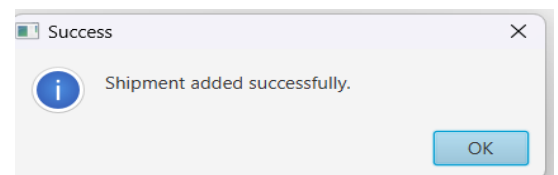
The screenshot shows the 'Port Management System' window with tabs for 'View Data', 'Ports', 'Shipments', 'Custodians', and 'Terminals'. The 'Shipments' tab is selected. A 'Manage Custodians' dialog box is open, featuring input fields for 'Custodian ID' (2000), 'Custodian Name' (RAVI), and 'Terminal ID' (1000). An 'Add' button is located at the bottom left of the dialog.



### Shipment:

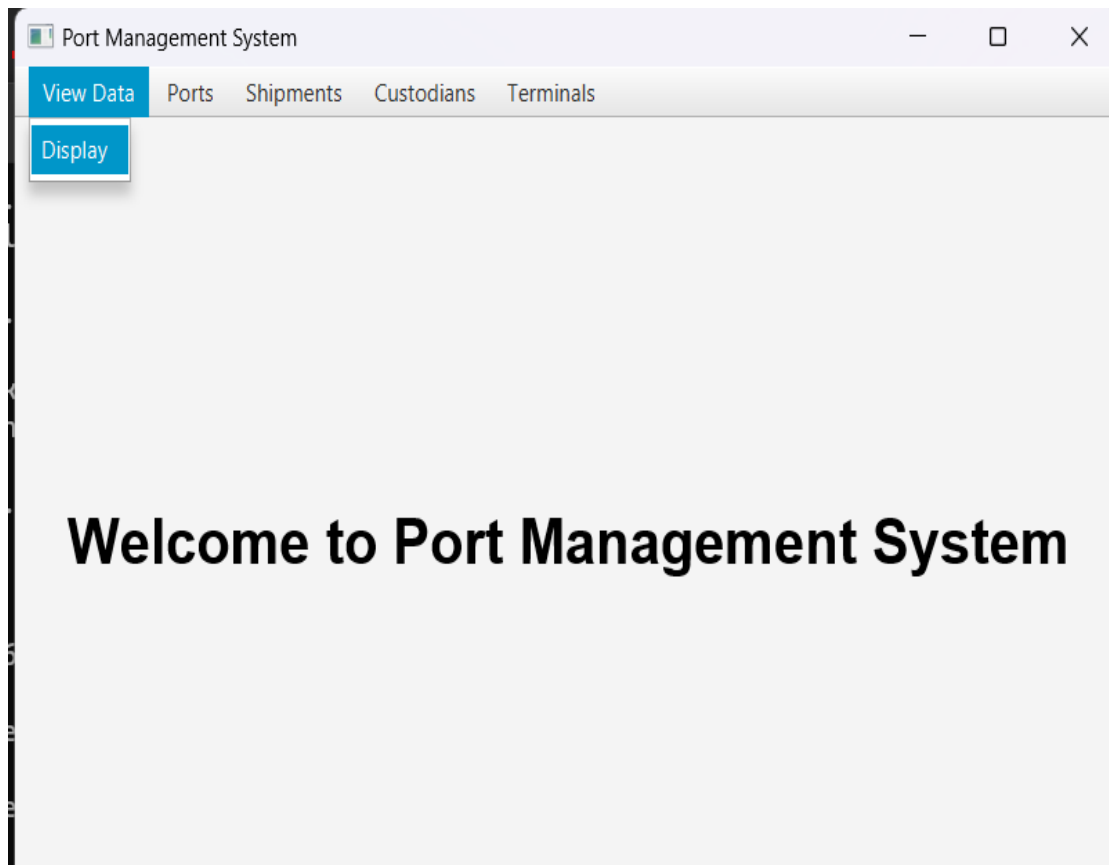


The screenshot shows the 'Port Management System' window with the 'Shipments' tab selected. A 'Manage Shipments' dialog box is open, featuring input fields for 'Shipment ID' (4000), 'Shipment Name' (JAZZ), 'Terminal ID' (1000), and 'Shipment Status' (In Progress). An 'Add' button is at the bottom left.



Common platforms for ports, terminals, custodians

Display data:



# Common platforms for ports, terminals, custodians

## TOTAL ENTRIES:

Port Management System

Back

Port ID	Port Name	Country	Actions	
302	orisa	india	<div>Update</div>	<div>Delete</div>
500	KERALA INT	INDIA	<div>Update</div>	<div>Delete</div>
22	bihar	india	<div>Update</div>	<div>Delete</div>

Terminal ID	Terminal Name	Actions	
1000	TERMINAL A	<div>Update</div>	<div>Delete</div>

Custodian ID	Custodian Name	Terminal ID	Actions	
2000	RAVI	1000	<div>Update</div>	<div>Delete</div>

Shipment ID	Shipment Name	Terminal ID	Shipment Status	Actions	
4000	JAZZ	1000	In Progress	<div>Update</div>	<div>Delete</div>

## Common platforms for ports, terminals, custodians

Update :

Port ID	Port Name	Country	Actions	
302	orrisa	india	Update	Delete
500	KERALA INT	INDIA	Update	Delete
22	bihar	india	Update	Delete

Port ID:

302

Port Name:

chandhigar

Country:

india

Update

Delete:

Update Port

Back

Port ID	Port Name	Country	Actions	
302	chandhigar	india	Update	Delete
500	KERALA INT	INDIA	Update	Delete
22	bihar	india	Update	Delete

Terminal ID	Terminal Name	Actions	
1000	TERMINAL A	Update	Delete

Custodian ID	Custodian Name	Terminal ID	Actions	
2000	RAVI	1000	Update	Delete

Shipment ID	Shipment Name	Terminal ID	Shipment Status	Actions	
4000	JAZZ	1000	In Progress	Update	Delete

Confirmation

Delete Port

Are you sure you want to delete the port?

OKCancel

FINAL TABLE ENTRY AFTER  
DELETION,UPDATION,INSERTION

Update Port

Back

Port ID	Port Name	Country	Actions	
302	chandhigar	india	<div>Update</div>	<div>Delete</div>
22	bihar	india	<div>Update</div>	<div>Delete</div>

Termina...	Termina...	Actions	
No content in table			

Custodi...	Custodi...	Termina...	Actions	
No content in table				

Shipme...	Shipme...	Termina...	Shipme...	Actions	
No content in table					

## RESULTS

I have successfully completed the mini-project “***COMMON PLATFORM FOR PORTS, TERMINALS, CUSTODIANS***”.

## DISCUSSION AND FUTURE WORK

Creating a common platform for ports, terminals, and custodians presents several advantages. Firstly, it streamlines operations and improves efficiency by integrating the management of these entities. Port authorities can easily oversee the activities and resources across different terminals, while custodians can efficiently coordinate with various ports and terminals. This integration promotes seamless communication and collaboration, enhancing overall productivity and customer service.

To further enhance the common platform, future work could focus on the development of advanced features and functionalities. This may include implementing real-time data synchronization between ports, terminals, and custodians, allowing for accurate tracking of shipments and resources. Integration with other systems, such as logistics or inventory management, could provide comprehensive insights and optimize the overall supply chain. Additionally, incorporating data analytics and predictive modeling can support proactive decision-making and performance optimization.

## REFERENCES

- <https://docs.oracle.com/javase/7/docs/api/>
- <https://www.javatpoint.com/java-swing>
- <https://stackoverflow.com/>