JAVA SWINGS BASED- CRIME DETECTION

- SQL CONNECTIVITY USING JDBC

A

Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree

of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

B Shiva Shankar<1602-21-737-050 > Under

the guidance of Ms. B. Leelavathy

Department of Information Technology

Vasavi College of Engineering

(Autonomous) (Affiliated to Osmania

University) Ibrahimbagh, Hyderabad-31

BONAFIDE CERTIFICATE

This is to certify that this project report titled **‘CRIME DETECTION POLICIES’** is a project work of B Shiva Shankar bearing roll no. 1602-21-737-050 who carried out the project under my supervision in the IV semester for the academic year 2022- 2023.

Signature	Signature
Internal Examiner	External Examiner

ABSTRACT

In this project we are going to create the tables of admin, post_case, department, victim by using these tables The project Crime Detection Policies management system used to save the data of the crime spot , to save the information of the department who solved the case, the information about victim , and the admin who dealing the case.

REQUIREMENT ANALYSIS:

Tables Needed to Implement

1. Table Name: `victim`
- Columns:
- `ID`: Number (3), primary key
- `Name`: Varchar2 (2), not null
- `PhoneNo`: Number (10), not null
- `Address`: Varchar2 (30)

2. Table Name: `policy_`
- Columns:
- `policyId`: Number (3), primary key
- `PolicyName`: Varchar2 (20), not null
- `Description`: Varchar2 (50)

3. Table Name: `dept`
- Columns:
- `deptid`: Number (3), primary key
- `deptname`: Varchar2 (20)
- `HelpLine`: Number (10)
- `location`: Varchar2 (20)

4. Table Name: `deptOfficer`
- Columns:
- `OfficerId`: Number (3), primary key
- `Name`: Varchar2 (20)
- `deptid`: Number (3)
- `role`: Varchar2 (20)
- Foreign Key: `deptid` references `dept(deptid)`

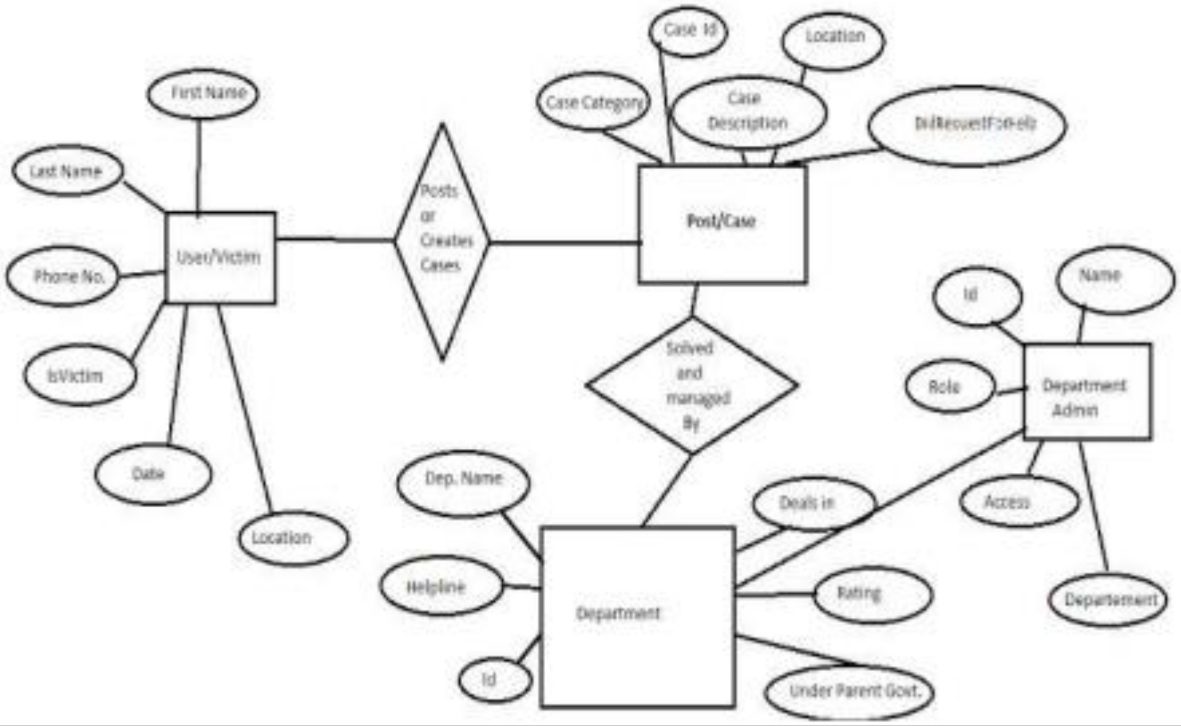
5. Table Name: `crime`
- Columns:
- `crimeId`: Number (3), primary key
- `crimeName`: Varchar2 (20), unique

6. Table Name: `crimeScene`
- Columns:
- `victimId`: Number (3), primary key
- `crimeName`: Varchar2 (20)
- `PolicyName`: Varchar2 (20)
- `OfficerId`: Number (3)
- `location`: Varchar2 (20)
- `crimedate`: Date, not null
- Foreign Keys:
- `victimId` references `victim(ID)`
- `crimeName` references `crime(crimeName)`
- `PolicyName` references `policy_(PolicyName)`
- `OfficerId` references `deptOfficer(OfficerId)`

These tables define the structure and relationships between various entities in the database, such as victims, policies, departments, officers, crimes, and crime scenes.

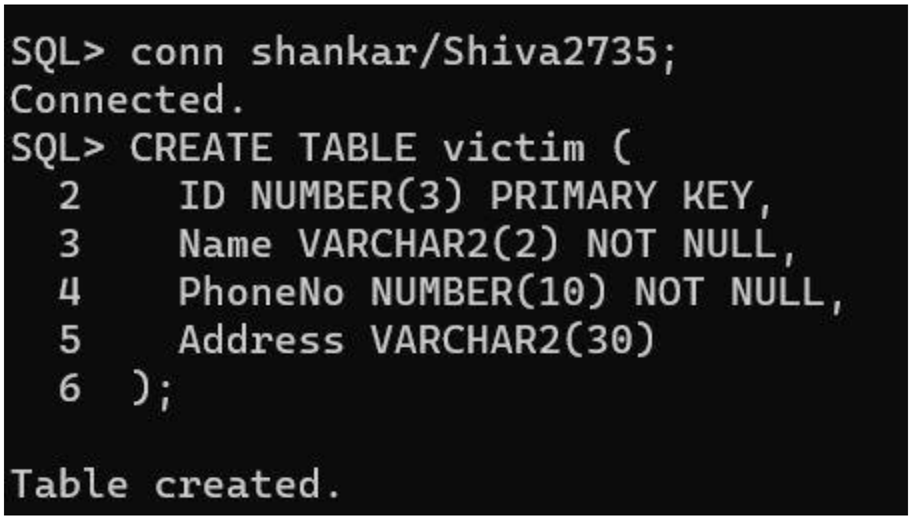
DESIGN:

ER DIAGRAM:



DDL COMMANDS:

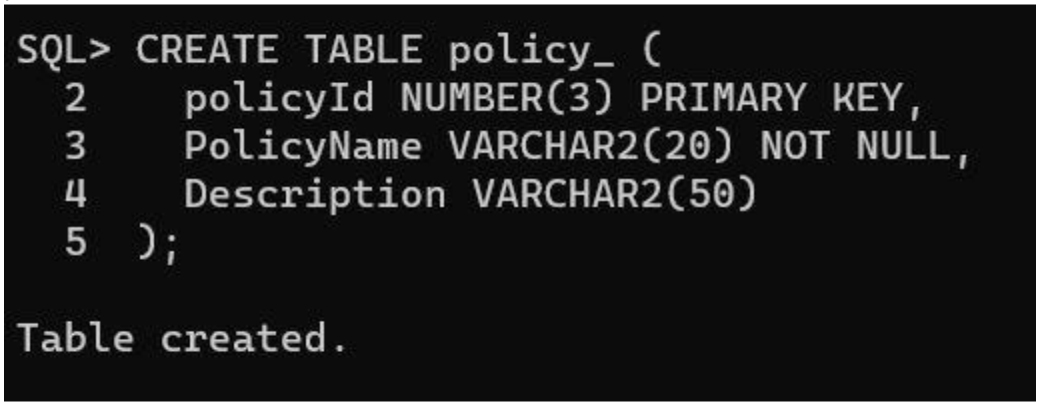
```
CREATE TABLE victim (
  ID NUMBER(3) PRIMARY KEY,
  Name VARCHAR2(2) NOT NULL,
  PhoneNo NUMBER(10) NOT NULL,
  Address VARCHAR2(30)
);
```



```
SQL> conn shankar/Shiva2735;
Connected.
SQL> CREATE TABLE victim (
  2    ID NUMBER(3) PRIMARY KEY,
  3    Name VARCHAR2(2) NOT NULL,
  4    PhoneNo NUMBER(10) NOT NULL,
  5    Address VARCHAR2(30)
  6  );

Table created.
```

2>Policy Table:

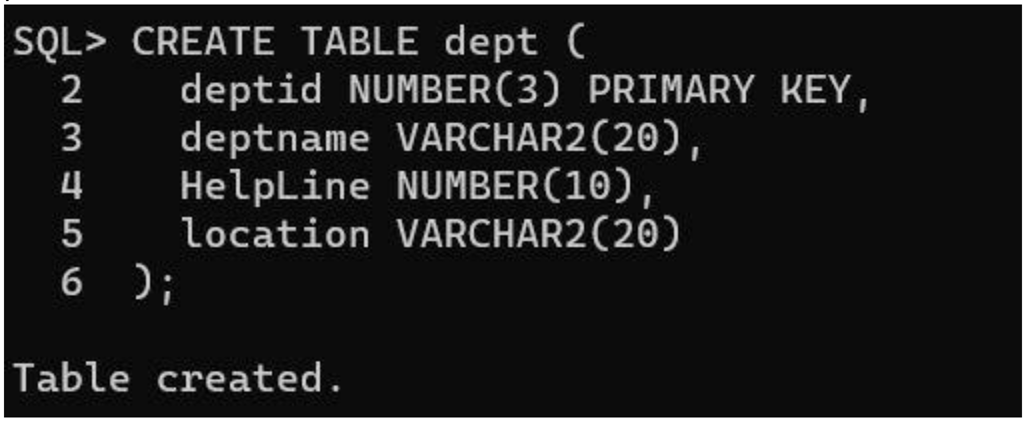
```
CREATE TABLE policy_ (
  policyId NUMBER(3) PRIMARY KEY,
  PolicyName VARCHAR2(20) NOT NULL,
  Description VARCHAR2(50)
);
```



```
SQL> CREATE TABLE policy_ (
  2    policyId NUMBER(3) PRIMARY KEY,
  3    PolicyName VARCHAR2(20) NOT NULL,
  4    Description VARCHAR2(50)
  5  );

Table created.
```

3>Department Table:

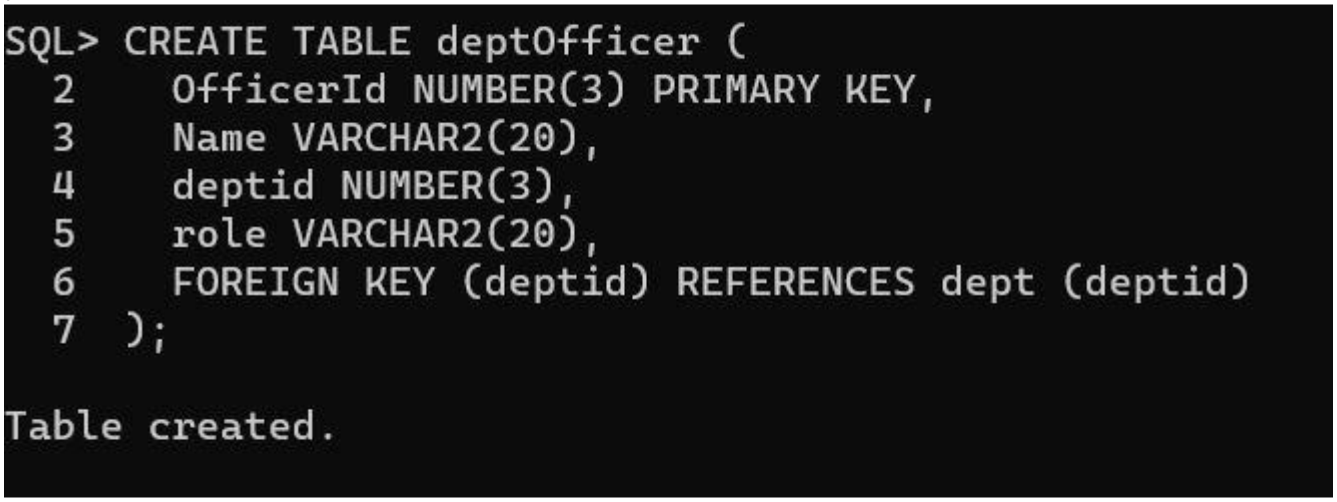
```
CREATE TABLE dept (
  deptid NUMBER(3) PRIMARY KEY,
  deptname VARCHAR2(20),
  HelpLine NUMBER(10),
  location VARCHAR2(20)
);
```



```
SQL> CREATE TABLE dept (
  2    deptid NUMBER(3) PRIMARY KEY,
  3    deptname VARCHAR2(20),
  4    HelpLine NUMBER(10),
  5    location VARCHAR2(20)
  6  );

Table created.
```

4>Department officer Table:

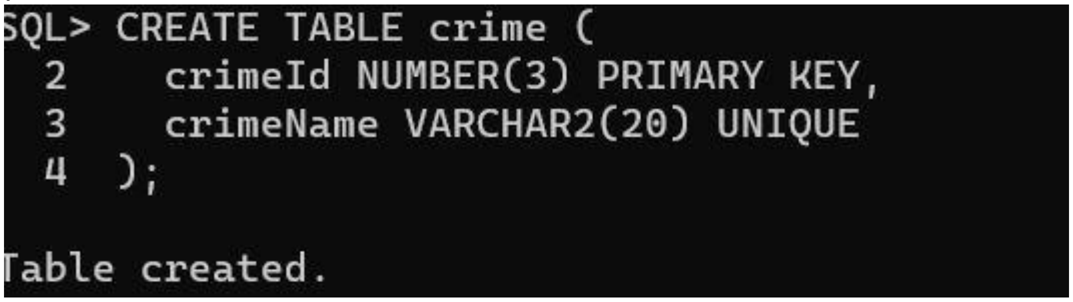
```
CREATE TABLE deptOfficer (
  OfficerId NUMBER(3) PRIMARY KEY,
  Name VARCHAR2(20),
  deptid NUMBER(3),
  role VARCHAR2(20),
  FOREIGN KEY (deptid) REFERENCES dept (deptid)
);
```



```
SQL> CREATE TABLE deptOfficer (
  2    OfficerId NUMBER(3) PRIMARY KEY,
  3    Name VARCHAR2(20),
  4    deptid NUMBER(3),
  5    role VARCHAR2(20),
  6    FOREIGN KEY (deptid) REFERENCES dept (deptid)
  7  );

Table created.
```

5>Crime Table:

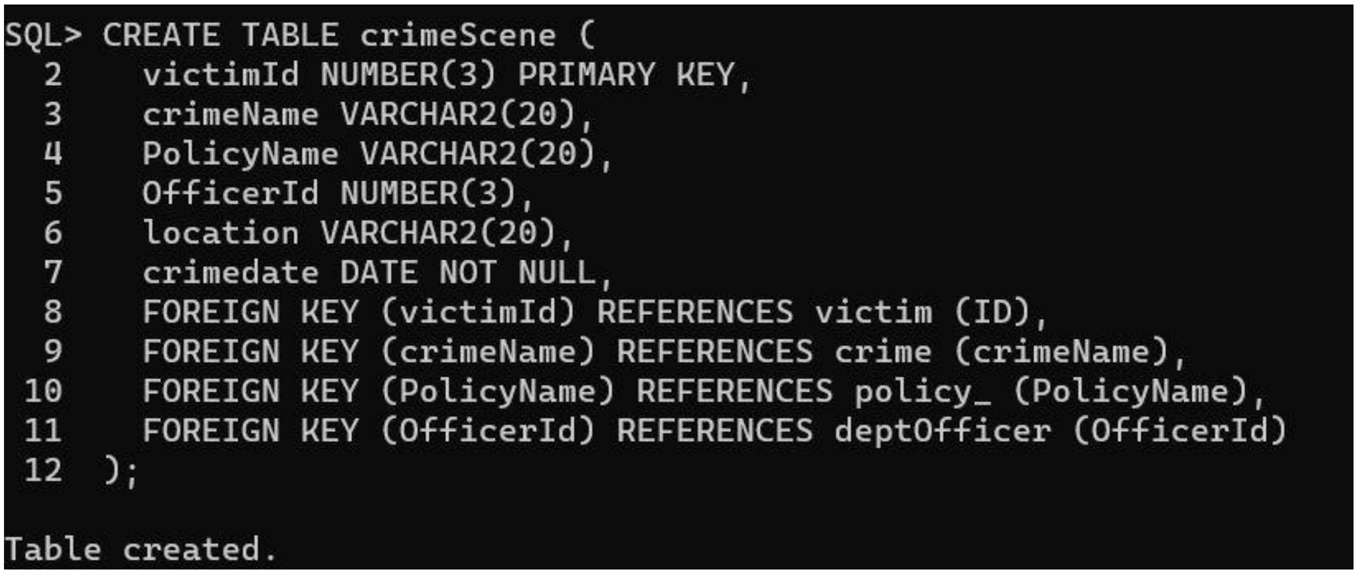
```
CREATE TABLE crime (
  crimeId NUMBER(3) PRIMARY KEY,
  crimeName VARCHAR2(20) UNIQUE
);
```



```
SQL> CREATE TABLE crime (
  2    crimeId NUMBER(3) PRIMARY KEY,
  3    crimeName VARCHAR2(20) UNIQUE
  4  );

Table created.
```

6>CrimeScene Table:

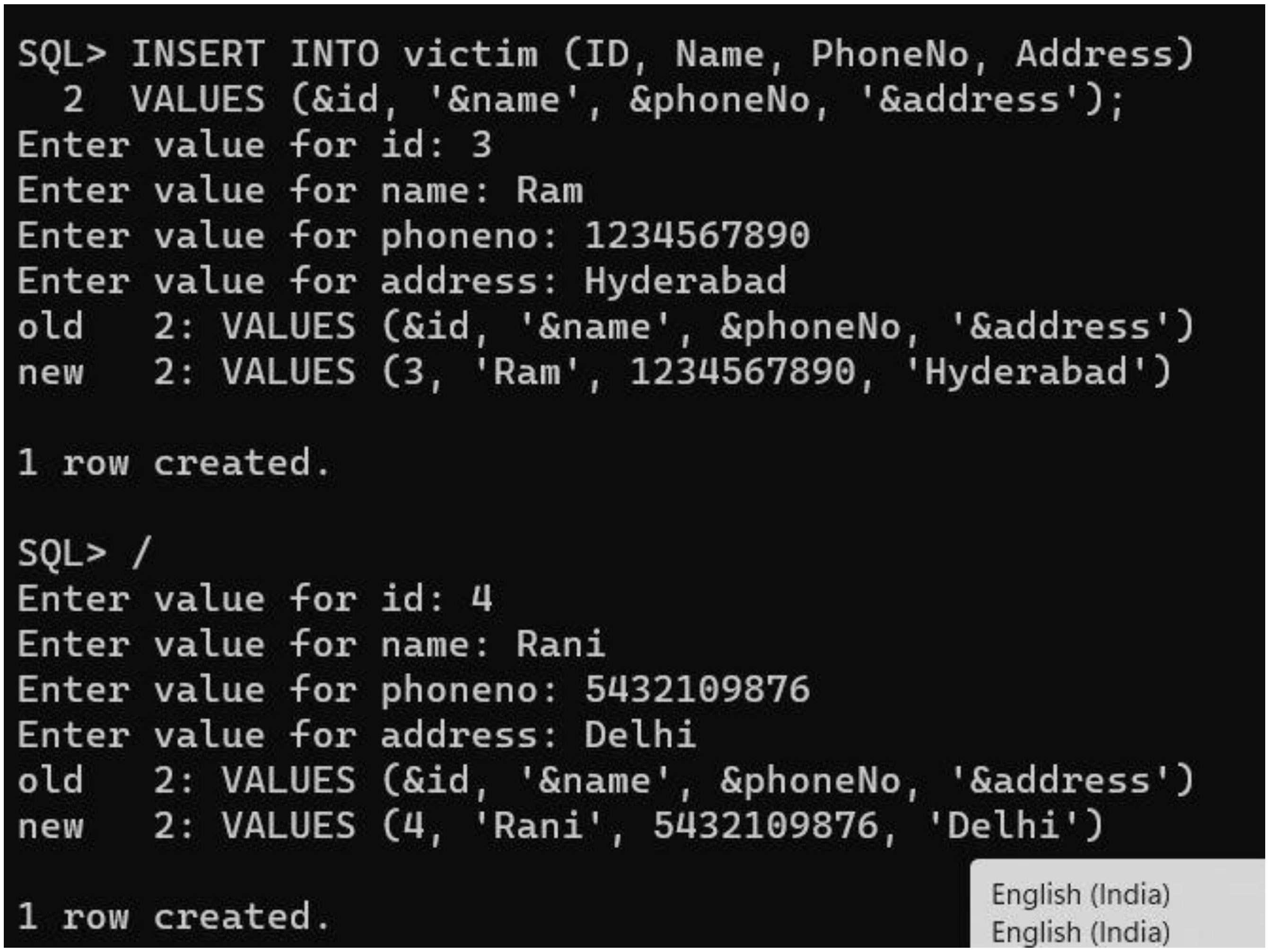
```
CREATE TABLE crimeScene (
  victimId NUMBER(3) PRIMARY KEY,
  crimeName VARCHAR2(20),
  PolicyName VARCHAR2(20),
  OfficerId NUMBER(3),
  location VARCHAR2(20),
  crimedate DATE NOT NULL,
  FOREIGN KEY (victimId) REFERENCES victim (ID),
  FOREIGN KEY (crimeName) REFERENCES crime (crimeName),
  FOREIGN KEY (PolicyName) REFERENCES policy_ (PolicyName),
  FOREIGN KEY (OfficerId) REFERENCES deptOfficer (OfficerId)
);
```



```
SQL> CREATE TABLE crimeScene (
  2    victimId NUMBER(3) PRIMARY KEY,
  3    crimeName VARCHAR2(20),
  4    PolicyName VARCHAR2(20),
  5    OfficerId NUMBER(3),
  6    location VARCHAR2(20),
  7    crimedate DATE NOT NULL,
  8    FOREIGN KEY (victimId) REFERENCES victim (ID),
  9    FOREIGN KEY (crimeName) REFERENCES crime (crimeName),
 10    FOREIGN KEY (PolicyName) REFERENCES policy_ (PolicyName),
 11    FOREIGN KEY (OfficerId) REFERENCES deptOfficer (OfficerId)
 12  );

Table created.
```

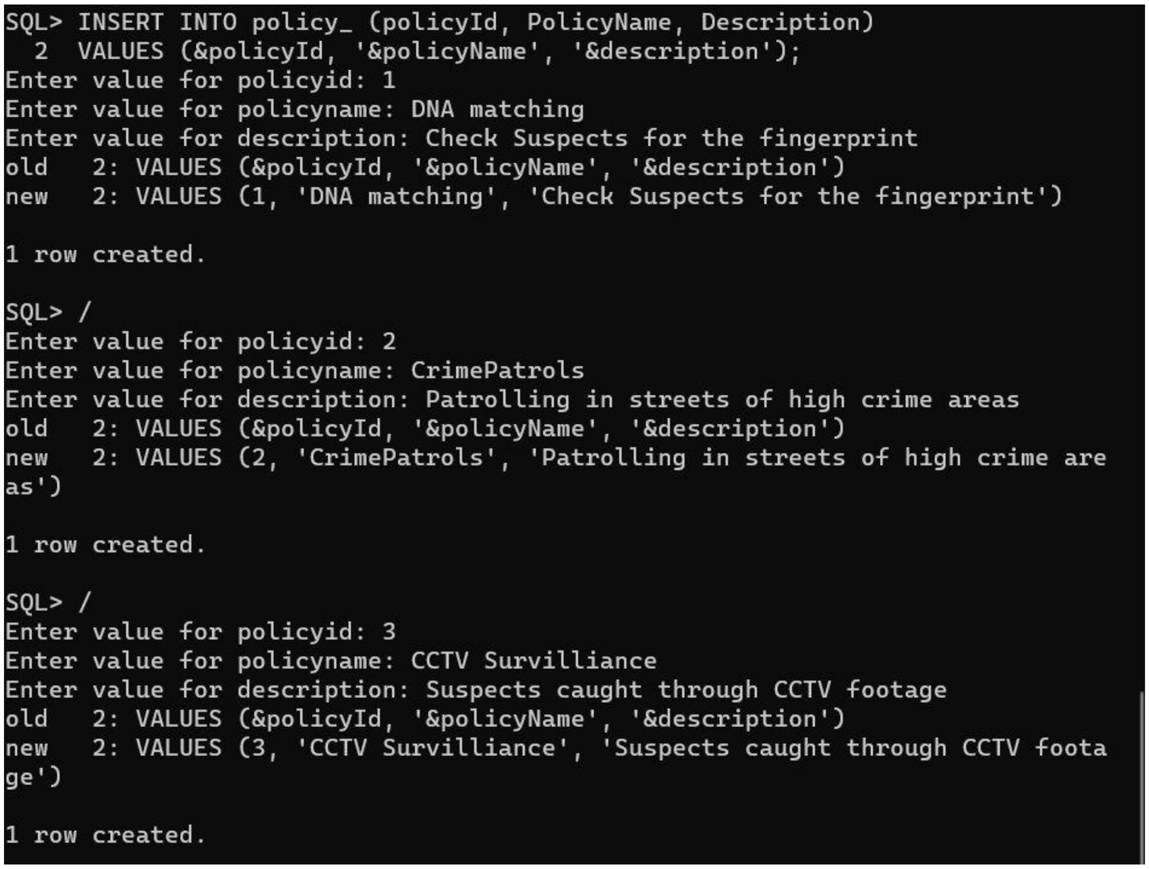
Table Insertions:

1>Victim Table:



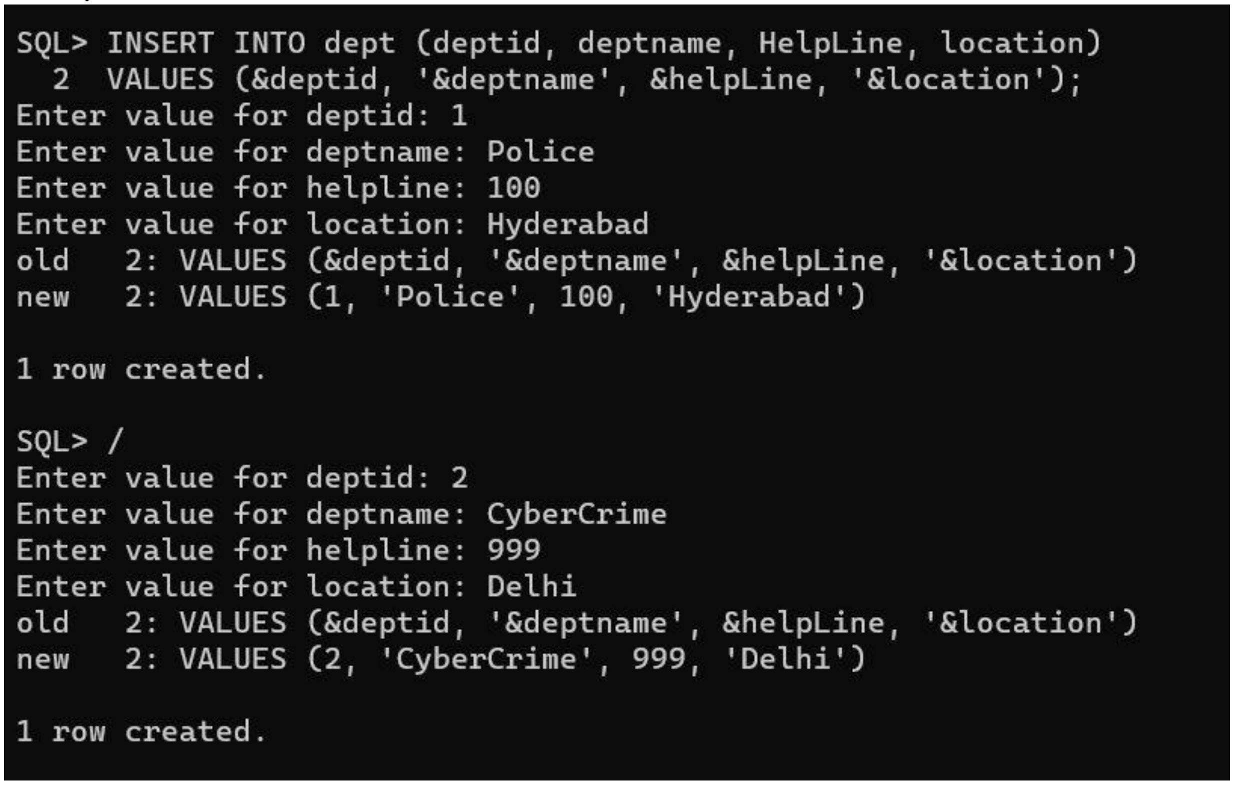
```
SQL> INSERT INTO victim (ID, Name, PhoneNo, Address)
  2  VALUES (&id, '&name', &phoneNo, '&address');
Enter value for id: 3
Enter value for name: Ram
Enter value for phoneno: 1234567890
Enter value for address: Hyderabad
old   2: VALUES (&id, '&name', &phoneNo, '&address')
new   2: VALUES (3, 'Ram', 1234567890, 'Hyderabad')

1 row created.

SQL> /
Enter value for id: 4
Enter value for name: Rani
Enter value for phoneno: 5432109876
Enter value for address: Delhi
old   2: VALUES (&id, '&name', &phoneNo, '&address')
new   2: VALUES (4, 'Rani', 5432109876, 'Delhi')

1 row created.
```

2>Policy Table:



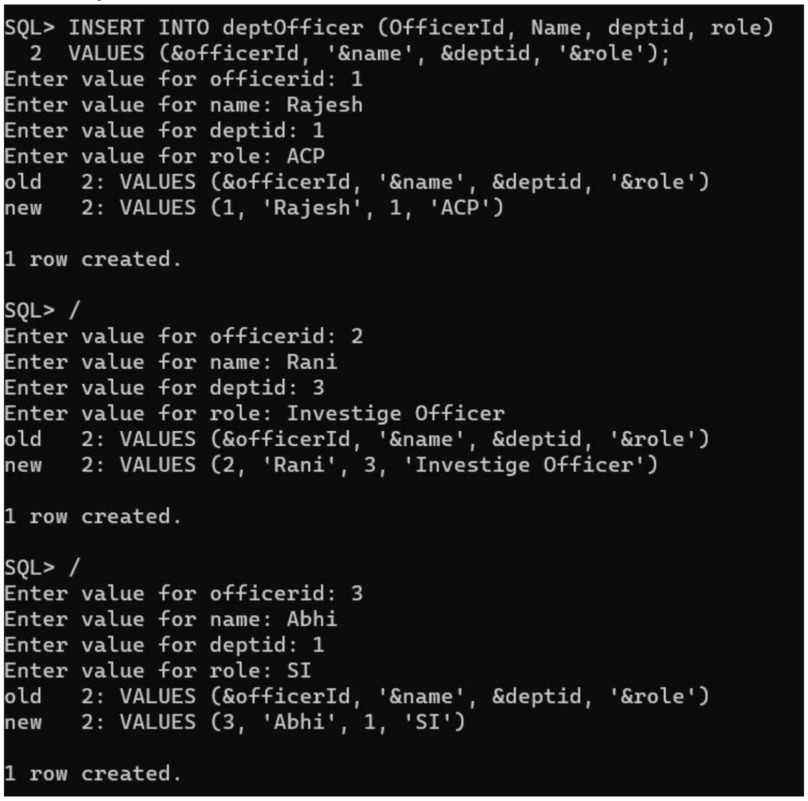
```
SQL> INSERT INTO policy_ (policyId, PolicyName, Description)
  2  VALUES (&policyId, '&policyName', '&description');
Enter value for policyid: 1
Enter value for policyname: DNA matching
Enter value for description: Check Suspects for the fingerprint
old   2: VALUES (&policyId, '&policyName', '&description')
new   2: VALUES (1, 'DNA matching', 'Check Suspects for the fingerprint')

1 row created.

SQL> /
Enter value for policyid: 2
Enter value for policyname: CrimePatrols
Enter value for description: Patrolling in streets of high crime areas
old   2: VALUES (&policyId, '&policyName', '&description')
new   2: VALUES (2, 'CrimePatrols', 'Patrolling in streets of high crime are
as')

1 row created.

SQL> /
Enter value for policyid: 3
Enter value for policyname: CCTV Survilliance
Enter value for description: Suspects caught through CCTV footage
old   2: VALUES (&policyId, '&policyName', '&description')
new   2: VALUES (3, 'CCTV Survilliance', 'Suspects caught through CCTV foota
ge')

1 row created.
```

3>Department Table:



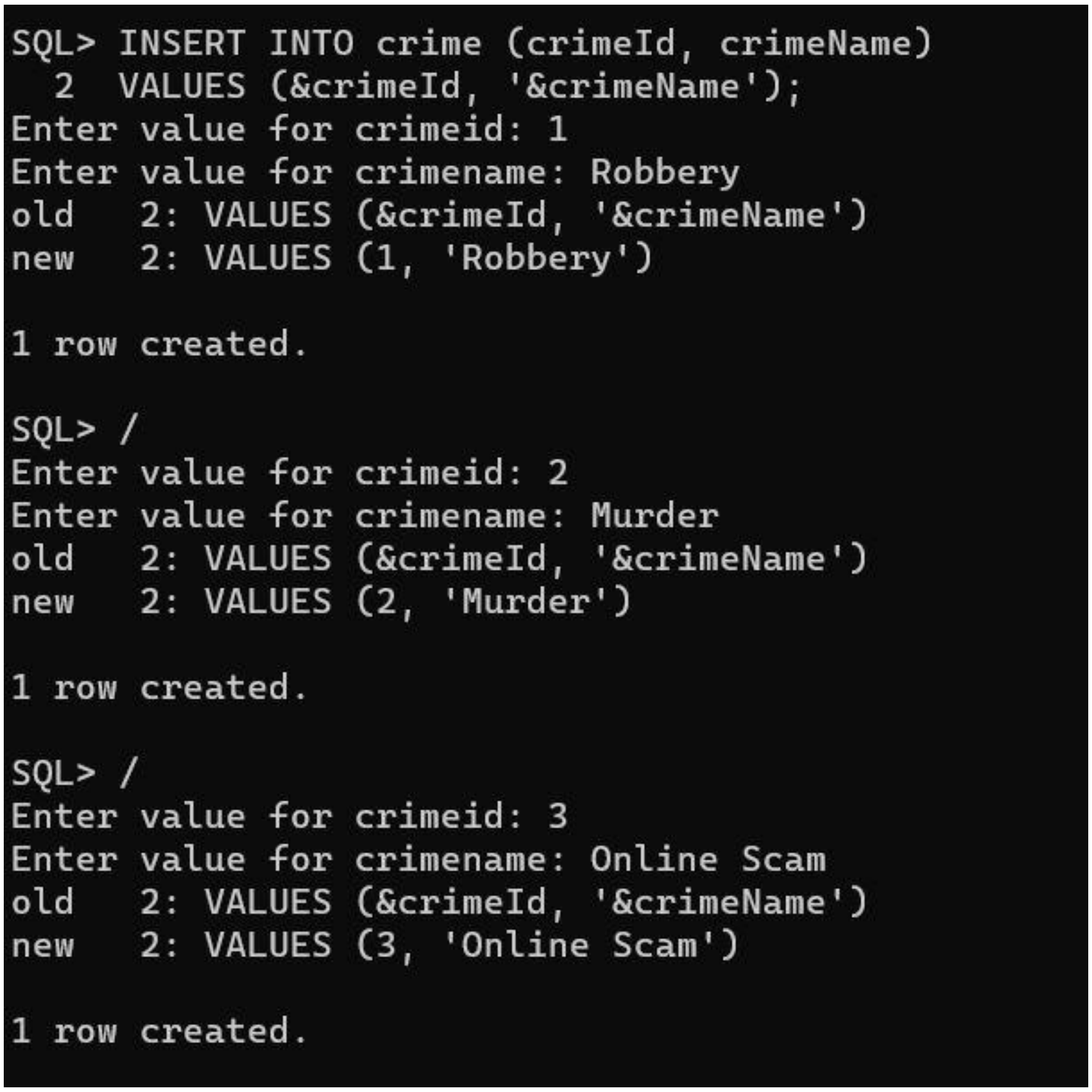
```
SQL> INSERT INTO dept (deptid, deptname, HelpLine, location)
  2  VALUES (&deptid, '&deptname', &helpLine, '&location');
Enter value for deptid: 1
Enter value for deptname: Police
Enter value for helpline: 100
Enter value for location: Hyderabad
old   2: VALUES (&deptid, '&deptname', &helpLine, '&location')
new   2: VALUES (1, 'Police', 100, 'Hyderabad')

1 row created.

SQL> /
Enter value for deptid: 2
Enter value for deptname: CyberCrime
Enter value for helpline: 999
Enter value for location: Delhi
old   2: VALUES (&deptid, '&deptname', &helpLine, '&location')
new   2: VALUES (2, 'CyberCrime', 999, 'Delhi')

1 row created.
```

4>DeptOfficer Table:



```
SQL> INSERT INTO deptOfficer (OfficerId, Name, deptid, role)
  2  VALUES (&officerId, '&name', &deptid, '&role');
Enter value for officerid: 1
Enter value for name: Rajesh
Enter value for deptid: 1
Enter value for role: ACP
old   2: VALUES (&officerId, '&name', &deptid, '&role')
new   2: VALUES (1, 'Rajesh', 1, 'ACP')

1 row created.

SQL> /
Enter value for officerid: 2
Enter value for name: Rani
Enter value for deptid: 3
Enter value for role: Investige Officer
old   2: VALUES (&officerId, '&name', &deptid, '&role')
new   2: VALUES (2, 'Rani', 3, 'Investige Officer')

1 row created.

SQL> /
Enter value for officerid: 3
Enter value for name: Abhi
Enter value for deptid: 1
Enter value for role: SI
old   2: VALUES (&officerId, '&name', &deptid, '&role')
new   2: VALUES (3, 'Abhi', 1, 'SI')

1 row created.
```

5>CrimeTable:



```
SQL> INSERT INTO crime (crimeId, crimeName)
  2  VALUES (&crimeId, '&crimeName');
Enter value for crimeid: 1
Enter value for crimename: Robbery
old   2: VALUES (&crimeId, '&crimeName')
new   2: VALUES (1, 'Robbery')

1 row created.

SQL> /
Enter value for crimeid: 2
Enter value for crimename: Murder
old   2: VALUES (&crimeId, '&crimeName')
new   2: VALUES (2, 'Murder')

1 row created.

SQL> /
Enter value for crimeid: 3
Enter value for crimename: Online Scam
old   2: VALUES (&crimeId, '&crimeName')
new   2: VALUES (3, 'Online Scam')

1 row created.
```

6>CrimeScene:

INSERT INTO crimeScene (victimId, crimeName, PolicyName, OfficerId, location, crimedate)
VALUES (1, 'Robbery', 'CCTV Survilliance', 3, 'Hyderabad', TO_DATE('2023-05-10', 'YYYY-MM-DD'));

IMPLEMENTATION:

Front end programs and its connectivity

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 aJava-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.

Code:

Victim:

```
public class Victim {
    private int ID;
    private String name;
    private long phoneNo;
    private String address;

    public Victim(int ID, String name, long phoneNo, String address) {
        this.ID = ID;
        this.name = name;
        this.phoneNo = phoneNo;
        this.address = address;
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public long getPhoneNo() {
        return phoneNo;
    }

    public void setPhoneNo(long phoneNo) {
        this.phoneNo = phoneNo;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}
```

Policy.java:

```
public class Policy {
    private int policyId;
    private String policyName;
    private String description;

    public Policy(int policyId, String policyName, String description) {
        this.policyId = policyId;
        this.policyName = policyName;
        this.description = description;
    }

    public int getPolicyId() {
        return policyId;
    }

    public void setPolicyId(int policyId) {
        this.policyId = policyId;
    }

    public String getPolicyName() {
        return policyName;
    }

    public void setPolicyName(String policyName) {
        this.policyName = policyName;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

3.Department.java:

```
public class Department {
```

```
private int deptId;
private String deptName;
private long helpLine;
private String location;

public Department(int deptId, String deptName, long helpLine, String location) {
    this.deptId = deptId;
    this.deptName = deptName;
    this.helpLine = helpLine;
    this.location = location;
}

public int getDeptId() {
    return deptId;
}

public void setDeptId(int deptId) {
    this.deptId = deptId;
}

public String getDeptName() {
    return deptName;
}

public void setDeptName(String deptName) {
    this.deptName = deptName;
}

public long getHelpLine() {
    return helpLine;
}
```

```
    public void setHelpLine(long helpLine) {
        this.helpLine = helpLine;
    }

    public String getLocation() {
        return location;
    }

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

4.DeptOfficer.java:

```
public class DeptOfficer {
    private int officerId;
    private String name;
    private int deptId;
    private String role;

    public DeptOfficer(int officerId, String name, int deptId, String role) {
        this.officerId = officerId;
        this.name = name;
        this.deptId = deptId;
        this.role = role;
    }

    public int getOfficerId() {
        return officerId;
    }
    public void setOfficerId(int officerId) {
this.officerId = officerId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getDeptId() {
        return deptId;
    }

    public void setDeptId(int deptId) {
        this.deptId = deptId;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
}
```

5.Crime.java:

```
public class Crime {
    private int crimeId;
    private String crimeName;

    public Crime(int crimeId, String crimeName) {
        this.crimeId = crimeId;
        this.crimeName = crimeName;
    }

    public int getCrimeId() {
        return crimeId;
    }

    public void setCrimeId(int crimeId) {
        this.crimeId = crimeId;
    }

    public String getCrimeName() {
        return crimeName;
    }

    public void setCrimeName(String crimeName) {
        this.crimeName = crimeName;
    }
}
```

6.CrimeScene.java:

```
import java.util.Date;

public class CrimeScene {
    private int victimId;
    private String crimeName;
    private String policyName;
    private int officerId;
    private String location;
    private Date crimeDate;

    public CrimeScene(int victimId, String crimeName, String policyName, int officerId, String location, Date crimeDate) {
        this.victimId = victimId;
        this.crimeName = crimeName;
        this.policyName = policyName;
        this.officerId = officerId;
        this.location = location;
        this.crimeDate = crimeDate;
    }

    public int getVictimId() {
        return victimId;
    }

    public void setVictimId(int victimId) {
        this.victimId = victimId;
    }

    public String getCrimeName() {
        return crimeName;
    }

    public void setCrimeName(String crimeName) {
        this.crimeName = crimeName;
    }

    public String getPolicyName() {
        return policyName;
    }

    public void setPolicyName(String policyName) {
        this.policyName = policyName;
    }

    public int getOfficerId() {
        return officerId;
    }

    public void setOfficerId(int officerId) {
        this.officerId = officerId;
    }

    public String getLocation() {
        return location;
    }

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

    public Date getCrimeDate() {
        return crimeDate;
    }

    public void setCrimeDate(Date crimeDate) {
        this.crimeDate = crimeDate;
    }
}
```

DatabaseManager.java:

```
import java.sql.*;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.sql.Date;
import java.time.LocalDate;

public class DatabaseManager {
    private Connection connection;
    private final String DB_URL = "jdbc:oracle:thin:@localhost:1521:xe";
    private final String DB_USER = "shiva050";
    private final String DB_PASSWORD = "Shiva2735";

    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<Victim> getAllVictims() throws SQLException {
      List<Victim> victims = new ArrayList<>();
      String query = "SELECT * FROM Victim";
      Statement statement = connection.createStatement();
      ResultSet resultSet = statement.executeQuery(query);
      while (resultSet.next()) {
                  int id = resultSet.getInt("ID");
            String name = resultSet.getString("Name");
            long phoneNo = resultSet.getLong("PhoneNo");
            String address = resultSet.getString("Address");
            Victim victim = new Victim(id, name, phoneNo, address);
            victims.add(victim);
    }
    return victims;
}
public List<Policy> getAllPolicies() throws SQLException {
    List<Policy> policies = new ArrayList<>();
    String query = "SELECT * FROM Policy_";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(query);
    while (resultSet.next()) {
            int policyId = resultSet.getInt("policyId");
            String policyName = resultSet.getString("PolicyName");
            String description = resultSet.getString("Description");
            Policy policy = new Policy(policyId, policyName, description);
            policies.add(policy);
    }
    return policies;
}
public List<Department> getAllDepartments() throws SQLException {
    List<Department> departments = new ArrayList<>();
    String query = "SELECT * FROM Dept";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(query);
    while (resultSet.next()) {
            int deptId = resultSet.getInt("deptid");
            String deptName = resultSet.getString("deptname");
            long helpLine = resultSet.getLong("HelpLine");
            String location = resultSet.getString("location");
            Department department = new Department(deptId, deptName, helpLine, location);
            departments.add(department);
    }
    return departments;
}
public List<DeptOfficer> getAllDeptOfficers() throws SQLException {
    List<DeptOfficer> deptOfficers = new ArrayList<>();
    String query = "SELECT * FROM DeptOfficer";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(query);
    while (resultSet.next()) {
            int officerId = resultSet.getInt("OfficerId");
            String name = resultSet.getString("Name");
            int deptId = resultSet.getInt("deptid");
            String role = resultSet.getString("role");
            DeptOfficer deptOfficer = new DeptOfficer(officerId, name, deptId, role);
            deptOfficers.add(deptOfficer);
    }
    return deptOfficers;
}
public List<Crime> getAllCrimes() throws SQLException {
    List<Crime> crimes = new ArrayList<>();
    String query = "SELECT * FROM Crime";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(query);
    while (resultSet.next()) {
            int crimeId = resultSet.getInt("crimeId");
            String crimeName = resultSet.getString("crimeName");
            Crime crime = new Crime(crimeId, crimeName);
            crimes.add(crime);
```

```
        }
        return crimes;
    }
    public List<CrimeScene> getAllCrimeScenes() throws SQLException {
        List<CrimeScene> crimeScenes = new ArrayList<>();
        String query = "SELECT * FROM CrimeScene";
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query);
        while (resultSet.next()) {
                int victimId = resultSet.getInt("victimId");
                String crimeName = resultSet.getString("crimeName");
                String policyName = resultSet.getString("PolicyName");
                int officerId = resultSet.getInt("OfficerId");
                String location = resultSet.getString("location");
                Date crimeDate = resultSet.getDate("crimedate");
                LocalDate localDate = crimeDate.toLocalDate();
                CrimeScene crimeScene = new CrimeScene(victimId, crimeName, policyName, officerId, location, crimeDate);
                crimeScenes.add(crimeScene);
        }
        return crimeScenes;
    }
    // Victim

      public void insertVictim(int id, String name, Long phoneNo, String address) throws SQLException {
        String query = "INSERT INTO Victim (ID, Name, PhoneNo, Address) VALUES (?, ?, ?, ?)";
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setInt(1, id);
        statement.setString(2, name);
        statement.setLong(3, phoneNo);
        statement.setString(4, address);
        statement.executeUpdate();
```

```
}

public void updateVictim(int id, String name, Long phoneNo, String address) throws SQLException {
    String query = "UPDATE Victim SET Name = ?, PhoneNo = ?, Address = ? WHERE ID = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, name);
    statement.setLong(2, phoneNo);
    statement.setString(3, address);
    statement.setInt(4, id);
    statement.executeUpdate();
}

public void deleteVictim(int id) throws SQLException {
    String query = "DELETE FROM Victim WHERE ID = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, id);
    statement.executeUpdate();
}

// Policy_

public void insertPolicy(int policyId, String policyName, String description) throws SQLException {
    String query = "INSERT INTO Policy_ (policyId, PolicyName, Description) VALUES (?, ?, ?)";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, policyId);
    statement.setString(2, policyName);
    statement.setString(3, description);
    statement.executeUpdate();
}

public void updatePolicy(int policyId, String policyName, String description) throws SQLException {
    String query = "UPDATE Policy_ SET PolicyName = ?, Description = ? WHERE policyId = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, policyName);
    statement.setString(2, description);
    statement.setInt(3, policyId);
    statement.executeUpdate();
}

public void deletePolicy(int policyId) throws SQLException {
    String query = "DELETE FROM Policy_ WHERE policyId = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, policyId);
    statement.executeUpdate();
}

// Dept

public void insertDept(int deptId, String deptName, long helpLine, String location) throws SQLException {
    String query = "INSERT INTO Dept (deptid, deptname, HelpLine, location) VALUES (?, ?, ?, ?)";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, deptId);
    statement.setString(2, deptName);
    statement.setLong(3, helpLine);
    statement.setString(4, location);
    statement.executeUpdate();
}

public void updateDept(int deptId, String deptName, long helpLine, String location) throws SQLException {
    String query = "UPDATE Dept SET deptname = ?, HelpLine = ?, location = ? WHERE deptid = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, deptName);
    statement.setLong(2, helpLine);
    statement.setString(3, location);
    statement.setInt(4, deptId);
    statement.executeUpdate();
}

public void deleteDept(int deptId) throws SQLException {
    String query = "DELETE FROM Dept WHERE deptid = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, deptId);
    statement.executeUpdate();
}

// DeptOfficer

public void insertDeptOfficer(int officerId, String name, int deptId, String role) throws SQLException {
    String query = "INSERT INTO DeptOfficer (OfficerId, Name, deptid, role) VALUES (?, ?, ?, ?)";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, officerId);
    statement.setString(2, name);
    statement.setInt(3, deptId);
    statement.setString(4, role);
    statement.executeUpdate();
}

public void updateDeptOfficer(int officerId, String name, int deptId, String role) throws SQLException {
    String query = "UPDATE DeptOfficer SET Name = ?, deptid = ?, role = ? WHERE OfficerId = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, name);
    statement.setInt(2, deptId);
    statement.setString(3, role);
    statement.setInt(4, officerId);
    statement.executeUpdate();
}

public void deleteDeptOfficer(int officerId) throws SQLException {
    String query = "DELETE FROM DeptOfficer WHERE OfficerId = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, officerId);
    statement.executeUpdate();
}

// Crime

public void insertCrime(int crimeId, String crimeName) throws SQLException {
    String query = "INSERT INTO Crime (crimeId, crimeName) VALUES (?, ?)";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setInt(1, crimeId);
    statement.setString(2, crimeName);
    statement.executeUpdate();
}

public void updateCrime(int crimeId, String crimeName) throws SQLException {
    String query = "UPDATE Crime SET crimeName = ? WHERE crimeId = ?";
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, crimeName);
    statement.setInt(2, crimeId);
    statement.executeUpdate();
}

public void deleteCrime(int crimeId) throws SQLException {
    String query = "DELETE FROM Crime WHERE crimeId = ?";
```

```
        PreparedStatement statement = connection.prepareStatement(query);

        statement.setInt(1, crimeId);

        statement.executeUpdate();

    }



    // CrimeScene



    public void insertCrimeScene(int victimId, String crimeName, String policyName, int officerId, String location, String crimeDate)
throws SQLException {

        String query = "INSERT INTO CrimeScene (victimId, crimeName, PolicyName, OfficerId, location, crimedate) VALUES (?, ?,
?, ?, ?, ?)";

        PreparedStatement statement = connection.prepareStatement(query);

        statement.setInt(1, victimId);

        statement.setString(2, crimeName);

        statement.setString(3, policyName);

        statement.setInt(4, officerId);

        statement.setString(5, location);

        statement.setString(6, crimeDate);

        statement.executeUpdate();

    }



    public void updateCrimeScene(int victimId, int newVictimId, String crimeName, String policyName, int officerId, String location,
String crimeDate) throws SQLException {

        String query = "UPDATE CrimeScene SET victimId = ?, crimeName = ?, PolicyName = ?, OfficerId = ?, location = ?,
crimedate = ? WHERE victimId = ?";

        PreparedStatement statement = connection.prepareStatement(query);

        statement.setInt(1, newVictimId);

        statement.setString(2, crimeName);

        statement.setString(3, policyName);

        statement.setInt(4, officerId);

        statement.setString(5, location);

        statement.setString(6, crimeDate);

        statement.setInt(7, victimId);
        statement.executeUpdate();
    }


    public void deleteCrimeScene(int victimId) throws SQLException {
        String query = "DELETE FROM CrimeScene WHERE victimId = ?";
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setInt(1, victimId);
        statement.executeUpdate();
    }
public List<LocationCrimeCount> getCrimeCountsByLocation() throws SQLException {
        String query = "SELECT location, COUNT(*) AS crimeCount FROM crimeScene GROUP BY location";
        List<LocationCrimeCount> crimeCounts = new ArrayList<>();


        try (PreparedStatement statement = connection.prepareStatement(query);
             ResultSet resultSet = statement.executeQuery()) {
            while (resultSet.next()) {
                String location = resultSet.getString("location");
                int crimeCount = resultSet.getInt("crimeCount");
                LocationCrimeCount crimeCountObj = new LocationCrimeCount(location, crimeCount);
                crimeCounts.add(crimeCountObj);
            }
        }


        return crimeCounts;
    }
public List<CrimeTypeCount> getCrimeCountsByType() throws SQLException {
      String query = "SELECT crimeName, COUNT(*) AS crimeCount FROM crimeScene GROUP BY crimeName";
      List<CrimeTypeCount> crimeCounts = new ArrayList<>();


      try (PreparedStatement statement = connection.prepareStatement(query);
                   ResultSet resultSet = statement.executeQuery()) {
```

```
            while (resultSet.next()) {
                String crimeName = resultSet.getString("crimeName");
                int crimeCount = resultSet.getInt("crimeCount");
                CrimeTypeCount crimeCountObj = new CrimeTypeCount(crimeName, crimeCount);
                crimeCounts.add(crimeCountObj);
            }
        }

        return crimeCounts;
    }



}
```

VictimForm.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 VictimForm extends GridPane {
    private TextField idField;
    private TextField nameField;
    private TextField phoneNoField;
    private TextField addressField;
    private Button addButton;

    private Victim victimToUpdate;
    private boolean formSubmitted = false;

    private Stage ownerStage;

    public VictimForm(Victim victim) {
        this();
        victimToUpdate = victim;
        fillFieldsWithData(victimToUpdate);

        addButton.setText("Update");
        addButton.setOnAction(e -> updateVictim());
    }

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

        Label idLabel = new Label("ID:");
        Label nameLabel = new Label("Name:");
        Label phoneNoLabel = new Label("Phone No:");
        Label addressLabel = new Label("Address:");
        idField = new TextField();
        nameField = new TextField();
        phoneNoField = new TextField();
        addressField = new TextField();
        addButton = new Button("Add");

        add(idLabel, 0, 0);
        add(idField, 1, 0);
        add(nameLabel, 0, 1);
```

```
        add(nameField, 1, 1);
        add(phoneNoLabel, 0, 2);
        add(phoneNoField, 1, 2);
        add(addressLabel, 0, 3);
        add(addressField, 1, 3);
        add(addButton, 0, 4);

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

    private void addVictim() {
        int id = Integer.parseInt(idField.getText());
        String name = nameField.getText();
        long phoneNo = Long.parseLong(phoneNoField.getText());
        String address = addressField.getText();

        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.insertVictim(id, name, phoneNo, address);
            databaseManager.disconnect();
            clearFields();
            showSuccessMessage("Victim added successfully.");
        } catch (SQLException ex) {
            showErrorMessage("Error adding victim: " + ex.getMessage());
        }
    }

    public void setTitle(String title) {
        if (ownerStage != null) {
            ownerStage.setTitle(title);
        }
    }

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

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

    public boolean isFormSubmitted() {
        return formSubmitted;
    }

    private void updateVictim() {
        if (victimToUpdate == null) {
            return;
        }

        int id = Integer.parseInt(idField.getText());
        String name = nameField.getText();
        long phoneNo = Long.parseLong(phoneNoField.getText());
        String address = addressField.getText();

        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
```

```
            databaseManager.updateVictim(victimToUpdate.getID(), name, phoneNo, address);

            databaseManager.disconnect();

            showSuccessMessage("Victim updated successfully.");

        } catch (SQLException ex) {

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

        }


        Stage stage = (Stage) getScene().getWindow();

        stage.close();

    }


    private void fillFieldsWithData(Victim victim) {

        idField.setText(String.valueOf(victim.getID()));

        nameField.setText(victim.getName());

        phoneNoField.setText(String.valueOf(victim.getPhoneNo()));

        addressField.setText(victim.getAddress());

    }


    private void clearFields() {

        idField.clear();

        nameField.clear();

        phoneNoField.clear();

        addressField.clear();

    }


    private void showSuccessMessage(String message) {

        Alert alert = new Alert(AlertType.INFORMATION);

        alert.setTitle("Success");

        alert.setHeaderText(null);

        alert.setContentText(message);

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

PolicyForm.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 PolicyForm extends GridPane {
    private TextField policyIdField;
    private TextField policyNameField;
    private TextField descriptionField;
    private Button addButton;

    private Policy policyToUpdate;
    private boolean formSubmitted = false;
private Stage ownerStage;


public PolicyForm(Policy policy) {
    this();
    policyToUpdate = policy;
    fillFieldsWithData(policyToUpdate);


    addButton.setText("Update");
    addButton.setOnAction(e -> updatePolicy());
}


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


    Label policyIdLabel = new Label("Policy ID:");
    Label policyNameLabel = new Label("Policy Name:");
    Label descriptionLabel = new Label("Description:");
    policyIdField = new TextField();
    policyNameField = new TextField();
    descriptionField = new TextField();
    addButton = new Button("Add");


    add(policyIdLabel, 0, 0);
    add(policyIdField, 1, 0);
    add(policyNameLabel, 0, 1);
    add(policyNameField, 1, 1);
    add(descriptionLabel, 0, 2);
    add(descriptionField, 1, 2);
    add(addButton, 0, 3);
```

```
        addButton.setOnAction(e -> addPolicy());
    }

    private void addPolicy() {
        int policyId = Integer.parseInt(policyIdField.getText());
        String policyName = policyNameField.getText();
        String description = descriptionField.getText();

        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.insertPolicy(policyId, policyName, description);
            databaseManager.disconnect();
            clearFields();
            showSuccessMessage("Policy added successfully.");
        } catch (SQLException ex) {
            showErrorMessage("Error adding policy: " + ex.getMessage());
        }
    }

    public void setTitle(String title) {
        if (ownerStage != null) {
            ownerStage.setTitle(title);
        }
    }

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

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

public boolean isFormSubmitted() {
    return formSubmitted;
}

private void updatePolicy() {
    if (policyToUpdate == null) {
        return;
    }

    int policyId = Integer.parseInt(policyIdField.getText());
    String policyName = policyNameField.getText();
    String description = descriptionField.getText();

    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.updatePolicy(policyToUpdate.getPolicyId(), policyName, description);
        databaseManager.disconnect();
        showSuccessMessage("Policy updated successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error updating policy: " + ex.getMessage());
    }

    Stage stage = (Stage) getScene().getWindow();
```

```
        stage.close();
    }

    private void fillFieldsWithData(Policy policy) {
        policyIdField.setText(String.valueOf(policy.getPolicyId()));
        policyNameField.setText(policy.getPolicyName());
        descriptionField.setText(policy.getDescription());
    }

    private void clearFields() {
        policyIdField.clear();
        policyNameField.clear();
        descriptionField.clear();
    }

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

        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();
    }
}
```

DepartmentForm.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 DepartmentForm extends GridPane {
    private TextField deptIdField;
    private TextField deptNameField;
    private TextField helpLineField;
    private TextField locationField;
    private Button addButton;

    private Department departmentToUpdate;
    private boolean formSubmitted = false;

    private Stage ownerStage;

    public DepartmentForm(Department department) {
        this();
        departmentToUpdate = department;
        fillFieldsWithData(departmentToUpdate);

        addButton.setText("Update");
        addButton.setOnAction(e -> updateDepartment());
```

```
}

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

    Label deptIdLabel = new Label("Department ID:");
    Label deptNameLabel = new Label("Department Name:");
    Label helpLineLabel = new Label("Help Line:");
    Label locationLabel = new Label("Location:");
    deptIdField = new TextField();
    deptNameField = new TextField();
    helpLineField = new TextField();
    locationField = new TextField();
    addButton = new Button("Add");

    add(deptIdLabel, 0, 0);
    add(deptIdField, 1, 0);
    add(deptNameLabel, 0, 1);
    add(deptNameField, 1, 1);
    add(helpLineLabel, 0, 2);
    add(helpLineField, 1, 2);
    add(locationLabel, 0, 3);
    add(locationField, 1, 3);
    add(addButton, 0, 4);

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

private void addDepartment() {
        int deptId = Integer.parseInt(deptIdField.getText());
        String deptName = deptNameField.getText();
        long helpLine = Long.parseLong(helpLineField.getText());
        String location = locationField.getText();

        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.insertDept(deptId, deptName,(long) helpLine, location);
            databaseManager.disconnect();
            clearFields();
            showSuccessMessage("Department added successfully.");
        } catch (SQLException ex) {
            showErrorMessage("Error adding department: " + ex.getMessage());
        }
    }

    public void setTitle(String title) {
        if (ownerStage != null) {
            ownerStage.setTitle(title);
        }
    }

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

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

    public boolean isFormSubmitted() {
        return formSubmitted;
    }

    private void updateDepartment() {
        if (departmentToUpdate == null) {
            return;
        }

        int deptId = Integer.parseInt(deptIdField.getText());
        String deptName = deptNameField.getText();
        long helpLine =Long.parseLong(helpLineField.getText());
        String location = locationField.getText();

        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.updateDept(departmentToUpdate.getDeptId(), deptName, helpLine, location);
            databaseManager.disconnect();
            showSuccessMessage("Department updated successfully.");
        } catch (SQLException ex) {
            showErrorMessage("Error updating department: " + ex.getMessage());
        }

        Stage stage = (Stage) getScene().getWindow();
        stage.close();
    }
```

```
private void fillFieldsWithData(Department department) {
    deptIdField.setText(String.valueOf(department.getDeptId()));
    deptNameField.setText(department.getDeptName());
    helpLineField.setText(String.valueOf(department.getHelpLine()));
    locationField.setText(department.getLocation());
}

private void clearFields() {
    deptIdField.clear();
    deptNameField.clear();
    helpLineField.clear();
    locationField.clear();
}

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

    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();
    }
}
```

DeptOfficer.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 DeptOfficerForm extends GridPane {
    private TextField officerIdField;
    private TextField nameField;
    private TextField deptIdField;
    private TextField roleField;
    private Button addButton;

    private DeptOfficer officerToUpdate;
    private boolean formSubmitted = false;

    private Stage ownerStage;

    public DeptOfficerForm(DeptOfficer officer) {
        this();
        officerToUpdate = officer;
        fillFieldsWithData(officerToUpdate);

        addButton.setText("Update");
        addButton.setOnAction(e -> updateOfficer());
    }
```

```
public DeptOfficerForm() {
    setPadding(new Insets(10));
    setHgap(10);
    setVgap(10);

    Label officerIdLabel = new Label("Officer ID:");
    Label nameLabel = new Label("Name:");
    Label deptIdLabel = new Label("Department ID:");
    Label roleLabel = new Label("Role:");
    officerIdField = new TextField();
    nameField = new TextField();
    deptIdField = new TextField();
    roleField = new TextField();
    addButton = new Button("Add");

    add(officerIdLabel, 0, 0);
    add(officerIdField, 1, 0);
    add(nameLabel, 0, 1);
    add(nameField, 1, 1);
    add(deptIdLabel, 0, 2);
    add(deptIdField, 1, 2);
    add(roleLabel, 0, 3);
    add(roleField, 1, 3);
    add(addButton, 0, 4);

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

private void addOfficer() {
    int officerId = Integer.parseInt(officerIdField.getText());
```

```
        String name = nameField.getText();

        int deptId = Integer.parseInt(deptIdField.getText());

        String role = roleField.getText();


        try {

            DatabaseManager databaseManager = new DatabaseManager();

                        databaseManager.connect();

            databaseManager.insertDeptOfficer(officerId, name, deptId, role);

            databaseManager.disconnect();

            clearFields();

            showSuccessMessage("Department Officer added successfully.");

        } catch (SQLException ex) {

            showErrorMessage("Error adding Department Officer: " + ex.getMessage());

        }

    }


    public void setTitle(String title) {

        if (ownerStage != null) {

            ownerStage.setTitle(title);

        }

    }


    public void setOwnerStage(Stage ownerStage) {

        this.ownerStage = ownerStage;

    }


    public void show() {

        Stage stage = new Stage();

        Scene scene = new Scene(this);

        stage.setScene(scene);

        stage.show();
```

```
}

public boolean isFormSubmitted() {
    return formSubmitted;
}

private void updateOfficer() {
    if (officerToUpdate == null) {
        return;
    }

    int officerId = Integer.parseInt(officerIdField.getText());
    String name = nameField.getText();
    int deptId = Integer.parseInt(deptIdField.getText());
    String role = roleField.getText();

    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.updateDeptOfficer(officerToUpdate.getOfficerId(), name, deptId, role);
        databaseManager.disconnect();
        showSuccessMessage("Department Officer updated successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error updating Department Officer: " + ex.getMessage());
    }

    Stage stage = (Stage) getScene().getWindow();
    stage.close();
}

private void fillFieldsWithData(DeptOfficer officer) {
        officerIdField.setText(String.valueOf(officer.getOfficerId()));
        nameField.setText(officer.getName());
        deptIdField.setText(String.valueOf(officer.getDeptId()));
        roleField.setText(officer.getRole());
    }

    private void clearFields() {
        officerIdField.clear();
        nameField.clear();
        deptIdField.clear();
        roleField.clear();
    }

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

        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();
    }
```

```
}

CrimeForm.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 DeptOfficerForm extends GridPane {

    private TextField officerIdField;

    private TextField nameField;

    private TextField deptIdField;

    private TextField roleField;

    private Button addButton;


    private DeptOfficer officerToUpdate;

    private boolean formSubmitted = false;


    private Stage ownerStage;


    public DeptOfficerForm(DeptOfficer officer) {

        this();

        officerToUpdate = officer;

        fillFieldsWithData(officerToUpdate);


        addButton.setText("Update");

        addButton.setOnAction(e -> updateOfficer());

    }
```

```
public DeptOfficerForm() {
    setPadding(new Insets(10));
    setHgap(10);
    setVgap(10);

    Label officerIdLabel = new Label("Officer ID:");
    Label nameLabel = new Label("Name:");
    Label deptIdLabel = new Label("Department ID:");
    Label roleLabel = new Label("Role:");
    officerIdField = new TextField();
    nameField = new TextField();
    deptIdField = new TextField();
    roleField = new TextField();
    addButton = new Button("Add");

    add(officerIdLabel, 0, 0);
    add(officerIdField, 1, 0);
    add(nameLabel, 0, 1);
    add(nameField, 1, 1);
    add(deptIdLabel, 0, 2);
    add(deptIdField, 1, 2);
    add(roleLabel, 0, 3);
    add(roleField, 1, 3);
    add(addButton, 0, 4);

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

private void addOfficer() {
    int officerId = Integer.parseInt(officerIdField.getText());
    String name = nameField.getText();
```

```
        int deptId = Integer.parseInt(deptIdField.getText());
        String role = roleField.getText();

        try {
            DatabaseManager databaseManager = new DatabaseManager();
                        databaseManager.connect();
            databaseManager.insertDeptOfficer(officerId, name, deptId, role);
            databaseManager.disconnect();
            clearFields();
            showSuccessMessage("Department Officer added successfully.");
        } catch (SQLException ex) {
            showErrorMessage("Error adding Department Officer: " + ex.getMessage());
        }
    }

    public void setTitle(String title) {
        if (ownerStage != null) {
            ownerStage.setTitle(title);
        }
    }

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

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

```
public boolean isFormSubmitted() {
    return formSubmitted;
}

private void updateOfficer() {
    if (officerToUpdate == null) {
        return;
    }

    int officerId = Integer.parseInt(officerIdField.getText());
    String name = nameField.getText();
    int deptId = Integer.parseInt(deptIdField.getText());
    String role = roleField.getText();

    try {
        DatabaseManager databaseManager = new DatabaseManager();
        databaseManager.connect();
        databaseManager.updateDeptOfficer(officerToUpdate.getOfficerId(), name, deptId, role);
        databaseManager.disconnect();
        showSuccessMessage("Department Officer updated successfully.");
    } catch (SQLException ex) {
        showErrorMessage("Error updating Department Officer: " + ex.getMessage());
    }

    Stage stage = (Stage) getScene().getWindow();
    stage.close();
}

private void fillFieldsWithData(DeptOfficer officer) {
    officerIdField.setText(String.valueOf(officer.getOfficerId()));
```

```
        nameField.setText(officer.getName());

        deptIdField.setText(String.valueOf(officer.getDeptId()));

        roleField.setText(officer.getRole());

    }


    private void clearFields() {

        officerIdField.clear();

        nameField.clear();

        deptIdField.clear();

        roleField.clear();

    }


    private void showSuccessMessage(String message) {

        Alert alert = new Alert(AlertType.INFORMATION);

        alert.setTitle("Success");

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();


        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();

    }

}
```

CrimeSceneForm.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;

import java.text.SimpleDateFormat;


public class CrimeSceneForm extends GridPane {

    private TextField victimIdField;

    private TextField crimeNameField;

    private TextField policyNameField;

    private TextField officerIdField;

    private TextField locationField;

    private TextField crimeDateField;

    private Button addButton;


    private CrimeScene crimeSceneToUpdate;

    private boolean formSubmitted = false;


    private Stage ownerStage;


    public CrimeSceneForm(CrimeScene crimeScene) {

        this();

        crimeSceneToUpdate = crimeScene;

        fillFieldsWithData(crimeSceneToUpdate);


        addButton.setText("Update");

        addButton.setOnAction(e -> updateCrimeScene());
```

```
}

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

    Label victimIdLabel = new Label("Victim ID:");
    Label crimeNameLabel = new Label("Crime Name:");
    Label policyNameLabel = new Label("Policy Name:");
    Label officerIdLabel = new Label("Officer ID:");
    Label locationLabel = new Label("Location:");
    Label crimeDateLabel = new Label("Crime Date:");
    victimIdField = new TextField();
    crimeNameField = new TextField();
    policyNameField = new TextField();
    officerIdField = new TextField();
    locationField = new TextField();
    crimeDateField = new TextField();
    addButton = new Button("Add");

    add(victimIdLabel, 0, 0);
    add(victimIdField, 1, 0);
    add(crimeNameLabel, 0, 1);
    add(crimeNameField, 1, 1);
    add(policyNameLabel, 0, 2);
    add(policyNameField, 1, 2);
    add(officerIdLabel, 0, 3);
    add(officerIdField, 1, 3);
    add(locationLabel, 0, 4);
    add(locationField, 1, 4);
```

```
        add(crimeDateLabel, 0, 5);
        add(crimeDateField, 1, 5);
        add(addButton, 0, 6);

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

    private void addCrimeScene() {
        int victimId = Integer.parseInt(victimIdField.getText());
        String crimeName = crimeNameField.getText();
        String policyName = policyNameField.getText();
        int officerId = Integer.parseInt(officerIdField.getText());
        String location = locationField.getText();
        String crimeDate = crimeDateField.getText();

        try {
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
            databaseManager.insertCrimeScene(victimId, crimeName, policyName, officerId, location, crimeDate);
            databaseManager.disconnect();
            clearFields();
            showSuccessMessage("Crime Scene added successfully.");
        } catch (SQLException ex) {
            showErrorMessage("Error adding Crime Scene: " + ex.getMessage());
        }
    }

    public void setTitle(String title) {
        if (ownerStage != null) {
            ownerStage.setTitle(title);
        }
```

```
}

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

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

public boolean isFormSubmitted() {
    return formSubmitted;
}

private void updateCrimeScene() {
    if (crimeSceneToUpdate == null) {
        return;
    }

    int victimId = Integer.parseInt(victimIdField.getText());
    String crimeName = crimeNameField.getText();
    String policyName = policyNameField.getText();
    int officerId = Integer.parseInt(officerIdField.getText());
    String location = locationField.getText();
    String crimeDate = crimeDateField.getText();

    try {
        DatabaseManager databaseManager = new DatabaseManager();
```

```
            databaseManager.connect();

            databaseManager.updateCrimeScene(crimeSceneToUpdate.getVictimId(), victimId, crimeName, policyName, officerId, location, crimeDate);

            databaseManager.disconnect();

            showSuccessMessage("Crime Scene updated successfully.");

        } catch (SQLException ex) {

            showErrorMessage("Error updating Crime Scene: " + ex.getMessage());

        }



        Stage stage = (Stage) getScene().getWindow();

        stage.close();

    }



    private void fillFieldsWithData(CrimeScene crimeScene) {

        victimIdField.setText(String.valueOf(crimeScene.getVictimId()));

        crimeNameField.setText(crimeScene.getCrimeName());

        policyNameField.setText(crimeScene.getPolicyName());

        officerIdField.setText(String.valueOf(crimeScene.getOfficerId()));

        locationField.setText(crimeScene.getLocation());

      SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

        String formattedDate = dateFormat.format(crimeScene.getCrimeDate());

        crimeDateField.setText(formattedDate);

    }



    private void clearFields() {

        victimIdField.clear();

        crimeNameField.clear();

        policyNameField.clear();

        officerIdField.clear();

        locationField.clear();

        crimeDateField.clear();
    }


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


        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();
    }
}
```

LocationCrimeCount.java:

```
public class LocationCrimeCount {
    private String location;
    private int crimeCount;


    public LocationCrimeCount(String location, int crimeCount) {
        this.location = location;
        this.crimeCount = crimeCount;
    }
    public String getLocation() {
        return location;
    }

    public int getCrimeCount() {
        return crimeCount;
    }
}
```

CrimeTypeCount.java:

```
public class CrimeTypeCount {
    private String crimeName;
    private int crimeCount;

    public CrimeTypeCount(String crimeName, int crimeCount) {
        this.crimeName = crimeName;
        this.crimeCount = crimeCount;
    }

    public String getCrimeName() {
        return crimeName;
    }

    public int getCrimeCount() {
        return crimeCount;
    }
}
```

CountData.java:

```
public class CountData {
    private String data1;
    private int data2;
    public CountData(String data1, int data2) {
        this.data1 = data1;
        this.data2 = data2;
    }

    public String getData1() {
        return data1;
    }

    public int getData2() {
        return data2;
    }
}
```

MainUI.java:

```
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 java.time.LocalDate;
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;

    public static void main(String[] args) {
        launch(args);
    }
@Override
    public void start(Stage primaryStage) {
        this.primaryStage = primaryStage;

        primaryStage.setTitle("Crime Detection Policies");

        BorderPane root = new BorderPane();

        // Create and configure the animated text
        Text animatedText = new Text("Welcome to Crime Detection Policies");
        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, 900, 900);
        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);



        Menu victimMenu = new Menu("Victims");
        MenuItem victimMenuItem = new MenuItem("Add Victim");
```

```
 victimMenuItem.setOnAction(e -> showVictimForm());
 victimMenu.getItems().add(victimMenuItem);

 Menu crimeMenu = new Menu("Crimes");
 MenuItem crimeMenuItem = new MenuItem("Add Crime");
 crimeMenuItem.setOnAction(e -> showCrimeForm());
 crimeMenu.getItems().add(crimeMenuItem);

 Menu departmentMenu = new Menu("Departments");
 MenuItem departmentMenuItem = new MenuItem("Add Department");
 departmentMenuItem.setOnAction(e -> showDepartmentForm());
 departmentMenu.getItems().add(departmentMenuItem);

 Menu officerMenu = new Menu("Department Officers");
 MenuItem officerMenuItem = new MenuItem("Add Department Officer");
 officerMenuItem.setOnAction(e -> showDeptOfficerForm());
 officerMenu.getItems().add(officerMenuItem);

 Menu policyMenu = new Menu("Policies");
 MenuItem policyMenuItem = new MenuItem("Add Policy");
 policyMenuItem.setOnAction(e -> showPolicyForm());
 policyMenu.getItems().add(policyMenuItem);

 Menu crimeSceneMenu = new Menu("Crime Scenes");
 MenuItem crimeSceneMenuItem = new MenuItem("Add Crime Scene");
 crimeSceneMenuItem.setOnAction(e -> showCrimeSceneForm());
 crimeSceneMenu.getItems().add(crimeSceneMenuItem);


Menu crimeInfoMenu = new Menu("Crime Info");
 MenuItem locationWiseItem = new MenuItem("CrimeInfo");
```

```
        locationWiseItem.setOnAction(event -> CrimeInfo());



        crimeInfoMenu.getItems().addAll(locationWiseItem);


        menuBar.getMenus().addAll(victimMenu,        crimeMenu,        departmentMenu,        officerMenu,        policyMenu,
crimeSceneMenu,dataMenu,crimeInfoMenu);


        return menuBar;

    }


private VBox container;

private TableView<Victim> createVictimTable(List<Victim> victims) {

       TableView<Victim> table = new TableView<>();


       TableColumn<Victim, Integer> idColumn = new TableColumn<>("ID");

       idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));


       TableColumn<Victim, String> nameColumn = new TableColumn<>("Name");

       nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));


       TableColumn<Victim, Integer> phoneColumn = new TableColumn<>("PhoneNo");

       phoneColumn.setCellValueFactory(new PropertyValueFactory<>("phoneNo"));


       TableColumn<Victim, String> addressColumn = new TableColumn<>("Address");

       addressColumn.setCellValueFactory(new PropertyValueFactory<>("address"));


       TableColumn<Victim, Void> actionsColumn = new TableColumn<>("Actions");

       actionsColumn.setCellFactory(param -> {

                return new TableCell<Victim, Void>() {

                        private final Button updateButton = new Button("Update");
```

```
                    private final Button deleteButton = new Button("Delete");


                    {
                              updateButton.setOnAction(event -> {
                                        Victim victim = (Victim) getTableRow().getItem();
                                        openVictimForm(victim);
                              });


                              deleteButton.setOnAction(event -> {
                                        Victim victim = (Victim) getTableRow().getItem();
                                        deleteVictim(victim);
                              });
                    }


                    @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(idColumn, nameColumn, phoneColumn, addressColumn, actionsColumn);
table.getItems().addAll(victims);


return table;
    }
```

```
private TableView<Policy> createPolicyTable(List<Policy> policies) {
    TableView<Policy> table = new TableView<>();

    TableColumn<Policy, Integer> policyIdColumn = new TableColumn<>("Policy ID");
    policyIdColumn.setCellValueFactory(new PropertyValueFactory<>("policyId"));

    TableColumn<Policy, String> policyNameColumn = new TableColumn<>("Policy Name");
    policyNameColumn.setCellValueFactory(new PropertyValueFactory<>("policyName"));

    TableColumn<Policy, String> descriptionColumn = new TableColumn<>("Description");
    descriptionColumn.setCellValueFactory(new PropertyValueFactory<>("description"));

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

                    {
                            updateButton.setOnAction(event -> {
                                    Policy policy = (Policy) getTableRow().getItem();
                                    openPolicyForm(policy);
                            });

                            deleteButton.setOnAction(event -> {
                                    Policy policy = (Policy) getTableRow().getItem();
                                    deletePolicy(policy);
                            });
                    }

                    @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(policyIdColumn, policyNameColumn, descriptionColumn, actionsColumn);
        table.getItems().addAll(policies);

        return table;
    }
    private TableView<Department> createDepartmentTable(List<Department> departments) {
        TableView<Department> table = new TableView<>();

        TableColumn<Department, Integer> deptIdColumn = new TableColumn<>("Department ID");
        deptIdColumn.setCellValueFactory(new PropertyValueFactory<>("deptId"));

        TableColumn<Department, String> deptNameColumn = new TableColumn<>("Department Name");
        deptNameColumn.setCellValueFactory(new PropertyValueFactory<>("deptName"));

        TableColumn<Department, Integer> helpLineColumn = new TableColumn<>("HelpLine");
        helpLineColumn.setCellValueFactory(new PropertyValueFactory<>("helpLine"));

        TableColumn<Department, String> locationColumn = new TableColumn<>("Location");
        locationColumn.setCellValueFactory(new PropertyValueFactory<>("location"));
```

```
});

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

                {
                        updateButton.setOnAction(event -> {
                                Department department = (Department) getTableRow().getItem();
                                openDepartmentForm(department);
                        });

                        deleteButton.setOnAction(event -> {
                                Department department = (Department) getTableRow().getItem();
                                deleteDepartment(department);
                        });
                }

                @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(deptIdColumn, deptNameColumn, helpLineColumn, locationColumn, actionsColumn);
```

```
        table.getItems().addAll(departments);

        return table;
    }
    private TableView<DeptOfficer> createDeptOfficerTable(List<DeptOfficer> deptOfficers) {
        TableView<DeptOfficer> table = new TableView<>();

        TableColumn<DeptOfficer, Integer> officerIdColumn = new TableColumn<>("Officer ID");
        officerIdColumn.setCellValueFactory(new PropertyValueFactory<>("officerId"));

        TableColumn<DeptOfficer, String> nameColumn = new TableColumn<>("Name");
        nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

        TableColumn<DeptOfficer, Integer> deptIdColumn = new TableColumn<>("Department ID");
        deptIdColumn.setCellValueFactory(new PropertyValueFactory<>("deptId"));

        TableColumn<DeptOfficer, String> roleColumn = new TableColumn<>("Role");
        roleColumn.setCellValueFactory(new PropertyValueFactory<>("role"));

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

                        {
                                updateButton.setOnAction(event -> {
                                        DeptOfficer deptOfficer = (DeptOfficer) getTableRow().getItem();
                                        openDeptOfficerForm(deptOfficer);
                                });
                deleteButton.setOnAction(event -> {
                    DeptOfficer deptOfficer = (DeptOfficer) getTableRow().getItem();
                    deleteDeptOfficer(deptOfficer);
                });
            }

            @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(officerIdColumn, nameColumn, deptIdColumn, roleColumn, actionsColumn);
    table.getItems().addAll(deptOfficers);

    return table;
}
private TableView<Crime> createCrimeTable(List<Crime> crimes) {
    TableView<Crime> table = new TableView<>();

    TableColumn<Crime, Integer> crimeIdColumn = new TableColumn<>("Crime ID");
    crimeIdColumn.setCellValueFactory(new PropertyValueFactory<>("crimeId"));

    TableColumn<Crime, String> crimeNameColumn = new TableColumn<>("Crime Name");
    crimeNameColumn.setCellValueFactory(new PropertyValueFactory<>("crimeName"));
```

```
        };

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

                {
                        updateButton.setOnAction(event -> {
                                Crime crime = (Crime) getTableRow().getItem();
                                openCrimeForm(crime);
                        });

                        deleteButton.setOnAction(event -> {
                                Crime crime = (Crime) getTableRow().getItem();
                                deleteCrime(crime);
                        });
                }

                @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(crimeIdColumn, crimeNameColumn, actionsColumn);
        table.getItems().addAll(crimes);

        return table;
    }
    private TableView<CrimeScene> createCrimeSceneTable(List<CrimeScene> crimeScenes) {
        TableView<CrimeScene> table = new TableView<>();

        TableColumn<CrimeScene, Integer> victimIdColumn = new TableColumn<>("Victim ID");
        victimIdColumn.setCellValueFactory(new PropertyValueFactory<>("victimId"));

        TableColumn<CrimeScene, String> crimeNameColumn = new TableColumn<>("Crime Name");
        crimeNameColumn.setCellValueFactory(new PropertyValueFactory<>("crimeName"));

        TableColumn<CrimeScene, String> policyNameColumn = new TableColumn<>("Policy Name");
        policyNameColumn.setCellValueFactory(new PropertyValueFactory<>("policyName"));

        TableColumn<CrimeScene, Integer> officerIdColumn = new TableColumn<>("Officer ID");
        officerIdColumn.setCellValueFactory(new PropertyValueFactory<>("officerId"));

        TableColumn<CrimeScene, String> locationColumn = new TableColumn<>("Location");
        locationColumn.setCellValueFactory(new PropertyValueFactory<>("location"));

        TableColumn<CrimeScene, LocalDate> crimeDateColumn = new TableColumn<>("Crime Date");
        crimeDateColumn.setCellValueFactory(new PropertyValueFactory<>("crimeDate"));

        TableColumn<CrimeScene, Void> actionsColumn = new TableColumn<>("Actions");
        actionsColumn.setCellFactory(param -> {
                return new TableCell<CrimeScene, Void>() {
                        private final Button updateButton = new Button("Update");
                        private final Button deleteButton = new Button("Delete");
                        {
                        updateButton.setOnAction(event -> {
                                        CrimeScene crimeScene = (CrimeScene) getTableRow().getItem();
                                        openCrimeSceneForm(crimeScene);
                                });

                                deleteButton.setOnAction(event -> {
                                        CrimeScene crimeScene = (CrimeScene) getTableRow().getItem();
                                        deleteCrimeScene(crimeScene);
                                });
                        }

                        @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(victimIdColumn, crimeNameColumn, policyNameColumn, officerIdColumn, locationColumn, 
crimeDateColumn, actionsColumn);
        table.getItems().addAll(crimeScenes);


        return table;
    private void displayData() {
        try {
```

```
// Fetch the data from the database using the DatabaseManager class
DatabaseManager databaseManager = new DatabaseManager();
databaseManager.connect();
List<Victim> victims = databaseManager.getAllVictims();
List<Crime> crimes = databaseManager.getAllCrimes();
List<Department> departments = databaseManager.getAllDepartments();
List<DeptOfficer> departmentOfficers = databaseManager.getAllDeptOfficers();
List<Policy> policies = databaseManager.getAllPolicies();
List<CrimeScene> crimeScenes = databaseManager.getAllCrimeScenes();
databaseManager.disconnect();

// Create UI components to display the data
TableView<Victim> victimTable = createVictimTable(victims);
TableView<Crime> crimeTable = createCrimeTable(crimes);
TableView<Department> departmentTable = createDepartmentTable(departments);
TableView<DeptOfficer> officerTable = createDeptOfficerTable(departmentOfficers);
TableView<Policy> policyTable = createPolicyTable(policies);
TableView<CrimeScene> crimeSceneTable = createCrimeSceneTable(crimeScenes);

// 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(victimTable, crimeTable, departmentTable, officerTable, policyTable,
}                 crimeSceneTable);


            // Create a scene and set it in the primaryStage
            Scene scene = new Scene(container, 1000, 1000);
            primaryStage.setScene(scene);
            primaryStage.show();

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




private void openCrimeSceneForm(CrimeScene crimeScene) {
      // Create a new instance of the CrimeSceneForm
      CrimeSceneForm crimeSceneForm = new CrimeSceneForm(crimeScene);

      // Set the title for the form
      crimeSceneForm.setTitle("Update Crime Scene");

      // Show the form
      crimeSceneForm.show();

      // Refresh the crime scene table after the form is closed
      if (crimeSceneForm.isFormSubmitted()) {
                    displayData();
      }
}
```

```
private void deleteCrimeScene(CrimeScene crimeScene) {
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle("Confirmation");
    alert.setHeaderText("Delete Crime Scene");
    alert.setContentText("Are you sure you want to delete the crime scene?");

    Optional<ButtonType> result = alert.showAndWait();
    if (result.isPresent() && result.get() == ButtonType.OK) {
            try {
                    DatabaseManager databaseManager = new DatabaseManager();
                    databaseManager.connect();
                    databaseManager.deleteCrimeScene(crimeScene.getVictimId());
                    databaseManager.disconnect();
                    showInformationMessage("Crime Scene deleted successfully.");
                    displayData();
            } catch (SQLException ex) {
                    showErrorMessage("Error deleting crime scene: " + ex.getMessage());
            }
    }
}

private void showCrimeSceneForm() {
    CrimeSceneForm crimeSceneForm = new CrimeSceneForm();
    Scene scene = new Scene(crimeSceneForm, 600, 400);
    Stage stage = new Stage();
    stage.setScene(scene);
    stage.setTitle("Manage Crime Scenes");
    stage.show();
}
private void openPolicyForm(Policy policy) {
    // Create a new instance of the PolicyForm
    PolicyForm policyForm = new PolicyForm(policy);

    // Set the title for the form
    policyForm.setTitle("Update Policy");

    // Show the form
    policyForm.show();

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

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

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

```
                }
        }
}

private void showPolicyForm() {
        PolicyForm policyForm = new PolicyForm();
        Scene scene = new Scene(policyForm, 600, 400);
        Stage stage = new Stage();
        stage.setScene(scene);
        stage.setTitle("Manage Policies");
        stage.show();
}

// Victim

    private void openVictimForm(Victim victim) {
        // Create a new instance of the VictimForm
        VictimForm victimForm = new VictimForm(victim);

        // Set the title for the form
        victimForm.setTitle("Update Victim");

        // Show the form
        victimForm.show();

        // Refresh the victim table after the form is closed
        if (victimForm.isFormSubmitted()) {
                displayData();
        }
}
```

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

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

private void showVictimForm() {
    VictimForm victimForm = new VictimForm();
    Scene scene = new Scene(victimForm, 600, 400);
    Stage stage = new Stage();
    stage.setScene(scene);
    stage.setTitle("Manage Victims");
    stage.show();
}

// Department
```

```
private void openDepartmentForm(Department department) {
    // Create a new instance of the DepartmentForm
    DepartmentForm departmentForm = new DepartmentForm(department);

    // Set the title for the form
    departmentForm.setTitle("Update Department");

    // Show the form
    departmentForm.show();

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

private void deleteDepartment(Department department) {
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle("Confirmation");
    alert.setHeaderText("Delete Department");
     alert.setContentText("Are you sure you want to delete the department?");
    Optional<ButtonType> result = alert.showAndWait();
    if (result.isPresent() && result.get() == ButtonType.OK) {
            try {
                    DatabaseManager databaseManager = new DatabaseManager();
                    databaseManager.connect();
                    databaseManager.deleteDept(department.getDeptId());
                    databaseManager.disconnect();
                    showInformationMessage("Department deleted successfully.");
                    displayData();
```

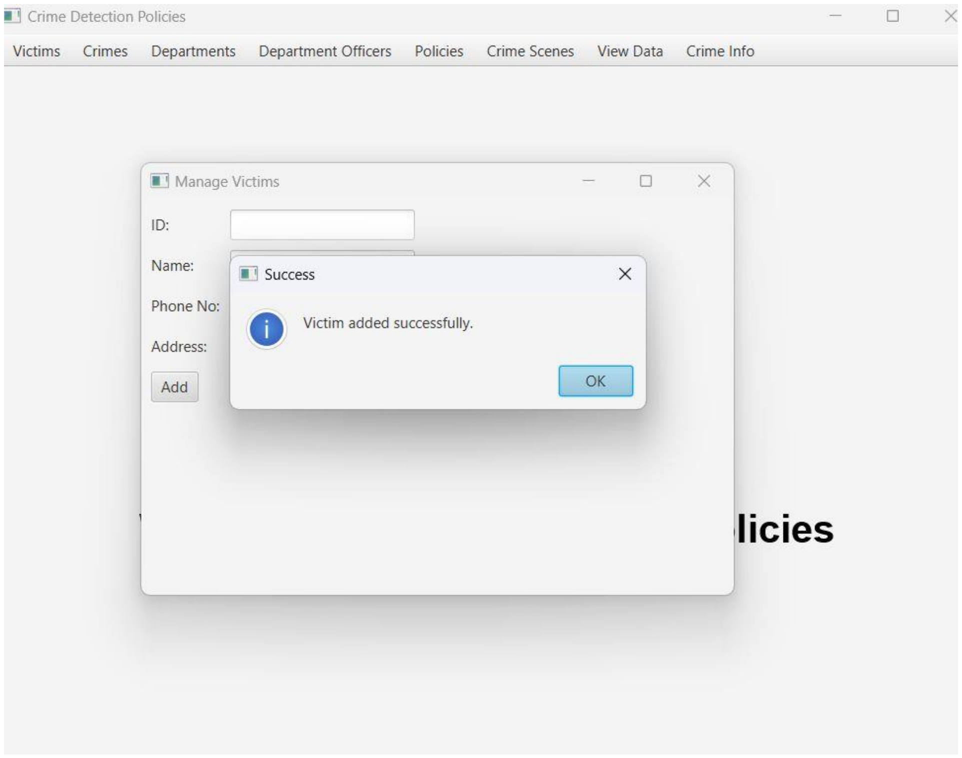
```
                } catch (SQLException ex) {
                        showErrorMessage("Error deleting department: " + ex.getMessage());
                }
        }
}
private void showDepartmentForm() {
        DepartmentForm departmentForm = new DepartmentForm();
        Scene scene = new Scene(departmentForm, 600, 400);
        Stage stage = new Stage();
        stage.setScene(scene);
        stage.setTitle("Manage Departments");
        stage.show();
}
private void openDeptOfficerForm(DeptOfficer deptOfficer) {
        // Create a new instance of the DeptOfficerForm
        DeptOfficerForm deptOfficerForm = new DeptOfficerForm(deptOfficer);
        // Set the title for the form
        deptOfficerForm.setTitle("Update Department Officer");

        // Show the form
        deptOfficerForm.show();

        // Refresh the deptOfficer table after the form is closed
        if (deptOfficerForm.isFormSubmitted()) {
                displayData();
        }
}
private void deleteDeptOfficer(DeptOfficer deptOfficer) {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Confirmation");
        alert.setHeaderText("Delete Department Officer");
```

```
    alert.setContentText("Are you sure you want to delete the department officer?");
    Optional<ButtonType> result = alert.showAndWait();
    if (result.isPresent() && result.get() == ButtonType.OK) {
            try {
                    DatabaseManager databaseManager = new DatabaseManager();
                    databaseManager.connect();
                    databaseManager.deleteDeptOfficer(deptOfficer.getOfficerId());
                    databaseManager.disconnect();
                    showInformationMessage("Department Officer deleted successfully.");
                    displayData();
            } catch (SQLException ex) {
                    showErrorMessage("Error deleting department officer: " + ex.getMessage());
            }
}
}
private void showDeptOfficerForm() {
    DeptOfficerForm deptOfficerForm = new DeptOfficerForm();
    Scene scene = new Scene(deptOfficerForm, 600, 400);
    Stage stage = new Stage();
    stage.setScene(scene);
    stage.setTitle("Manage Department Officers");
    stage.show();
}

    // Crime

private void openCrimeForm(Crime crime) {
    // Create a new instance of the CrimeForm
    CrimeForm crimeForm = new CrimeForm(crime);
    // Set the title for the form
    crimeForm.setTitle("Update Crime");
    // Show the form
    crimeForm.show();

    // Refresh the crime table after the form is closed
    if (crimeForm.isFormSubmitted()) {
            displayData();
    }
}
private void deleteCrime(Crime crime) {
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle("Confirmation");
    alert.setHeaderText("Delete Crime");
    alert.setContentText("Are you sure you want to delete the crime?");
    Optional<ButtonType> result = alert.showAndWait();
    if (result.isPresent() && result.get() == ButtonType.OK) {
            try {
                    DatabaseManager databaseManager = new DatabaseManager();
                    databaseManager.connect();
                    databaseManager.deleteCrime(crime.getCrimeId());
                    databaseManager.disconnect();
                    showInformationMessage("Crime deleted successfully.");
                    displayData();
            } catch (SQLException ex){
                    showErrorMessage("Error deleting crime: " + ex.getMessage());
                    }
    }
}

private void showCrimeForm() {
    CrimeForm crimeForm = new CrimeForm();
```

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

        Stage stage = new Stage();

        stage.setScene(scene);

        stage.setTitle("Manage Crimes");

        stage.show();

}

private void showHome() {

         primaryStage.setScene(createHomePage());

     }

private Scene createHomePage() {

        BorderPane root = new BorderPane();


        // Create and configure the animated text

        Text animatedText = new Text("Welcome to Crime Detection Policies");

        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, 1200, 1200);
        scene.setFill(Color.DARKBLUE);

        return scene;
    }
    private TableView<LocationCrimeCount> createLocationWiseTable(List<LocationCrimeCount> locationCounts) {
        TableView<LocationCrimeCount> table = new TableView<>();

        TableColumn<LocationCrimeCount, String> locationColumn = new TableColumn<>("Location");
        locationColumn.setCellValueFactory(new PropertyValueFactory<>("location"));

        TableColumn<LocationCrimeCount, Integer> countColumn = new TableColumn<>("Crime Count");
        countColumn.setCellValueFactory(new PropertyValueFactory<>("crimeCount"));

        table.getColumns().addAll(locationColumn, countColumn);
        table.getItems().addAll(locationCounts);

        return table;
    }
    private TableView<CrimeTypeCount> createCrimeWiseTable(List<CrimeTypeCount> crimeCounts) {
        TableView<CrimeTypeCount> table = new TableView<>();

        TableColumn<CrimeTypeCount, String> crimeColumn = new TableColumn<>("Crime Name");
        crimeColumn.setCellValueFactory(new PropertyValueFactory<>("crimeName"));

        TableColumn<CrimeTypeCount, Integer> countColumn = new TableColumn<>("Crime Count");
        countColumn.setCellValueFactory(new PropertyValueFactory<>("crimeCount"));
        table.getColumns().addAll(crimeColumn, countColumn);
        table.getItems().addAll(crimeCounts);

        return table;
}
public void CrimeInfo(){
        try {
            // Fetch the data from the database using the DatabaseManager class
            DatabaseManager databaseManager = new DatabaseManager();
            databaseManager.connect();
                    List<LocationCrimeCount> locationCounts = databaseManager.getCrimeCountsByLocation();
                   List<CrimeTypeCount> crimeCounts = databaseManager.getCrimeCountsByType();
                   databaseManager.disconnect();
                   TableView<LocationCrimeCount> locationTable=createLocationWiseTable(locationCounts);
                   TableView<CrimeTypeCount> crimeTable=createCrimeWiseTable(crimeCounts);
                    VBox container = new VBox();
                   container.setSpacing(20);
                   container.setPadding(new Insets(20));
                   Button backButton = new Button("Back");
                   backButton.setOnAction(e -> showHome());
                   container.getChildren().add(backButton);
                   container.getChildren().addAll(locationTable,crimeTable);
                   Scene scene = new Scene(container, 1000, 1000);
                   primaryStage.setScene(scene);
                   primaryStage.show();
        }catch (SQLException ex) {
                   showErrorMessage("Error fetching data: " + ex.getMessage());
        }
}
    private void showInformationMessage(String message) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Information");
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }

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

Output Screen Shots:

Victim:



Crime Detection Policies

Victims Crimes Departments Department Officers Policies Crime Scenes View Data Crime Info

Manage Victims

ID:

Name:

Phone No:

Address:

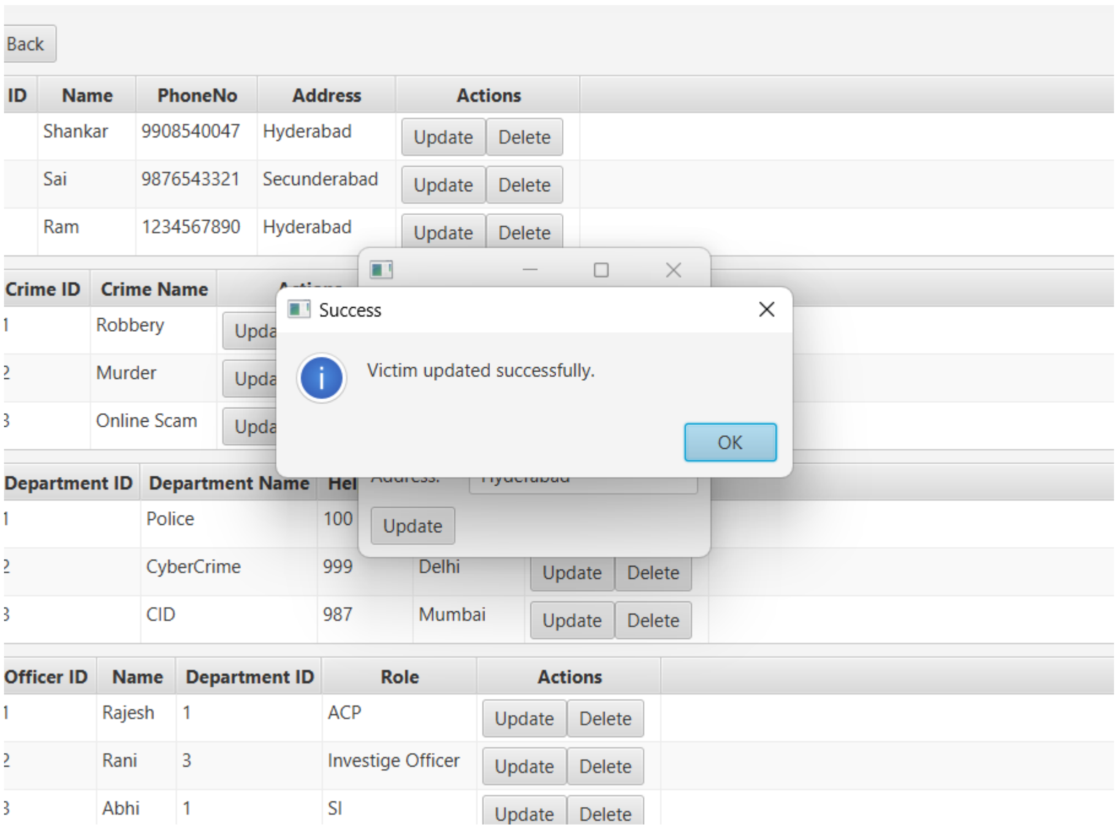
Add

Success

Victim added successfully.

OK

licies



Back

ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Department ID	Department Name	Hel		
1	Police	100		
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

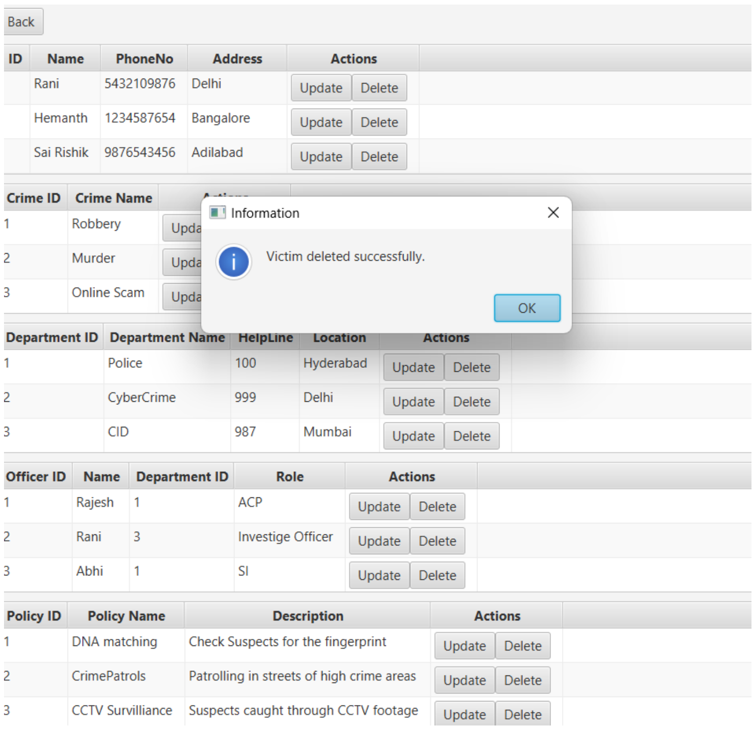
Success

Victim updated successfully.

OK

Address: Hyderabad

Update



Back

ID	Name	PhoneNo	Address	Actions
	Rani	5432109876	Delhi	Update Delete
	Hemanth	1234587654	Bangalore	Update Delete
	Sai Rishik	9876543456	Adilabad	Update Delete

Crime ID	Crime Name	
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Information

Victim deleted successfully.

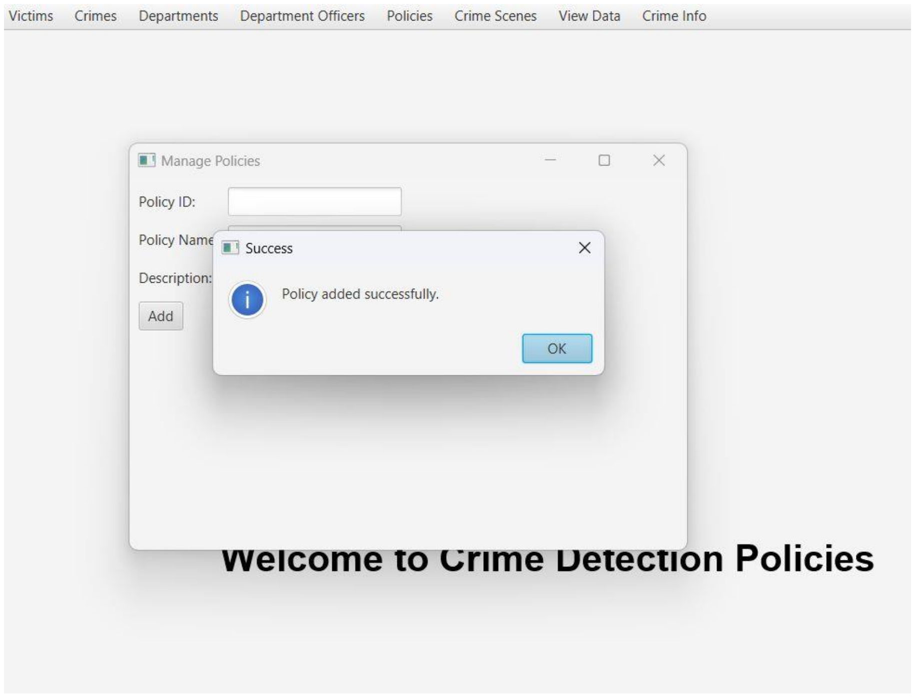
OK

Department ID	Department Name	HelpLine	Location	Actions
1	Police	100	Hyderabad	Update Delete
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Policy ID	Policy Name	Description	Actions
1	DNA matching	Check Suspects for the fingerprint	Update Delete
2	CrimePatrols	Patrolling in streets of high crime areas	Update Delete
3	CCTV Survilliance	Suspects caught through CCTV footage	Update Delete

Policy:



Victims Crimes Departments Department Officers Policies Crime Scenes View Data Crime Info

Manage Policies

Policy ID:

Policy Name

Description:

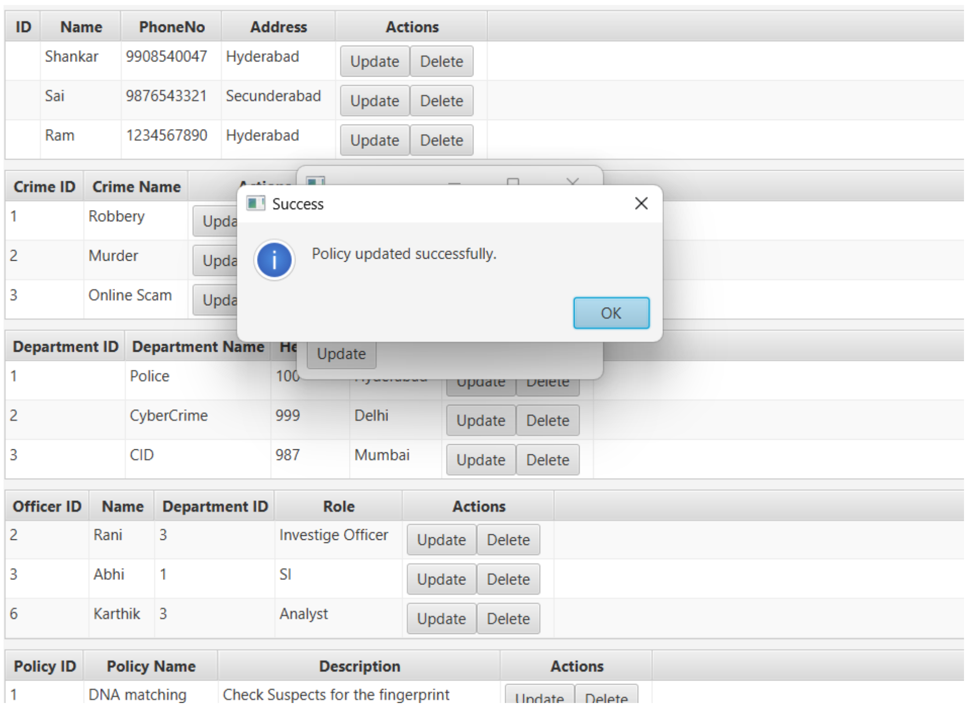
Add

Success

Policy added successfully.

OK

Welcome to Crime Detection Policies



ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Success

Policy updated successfully.

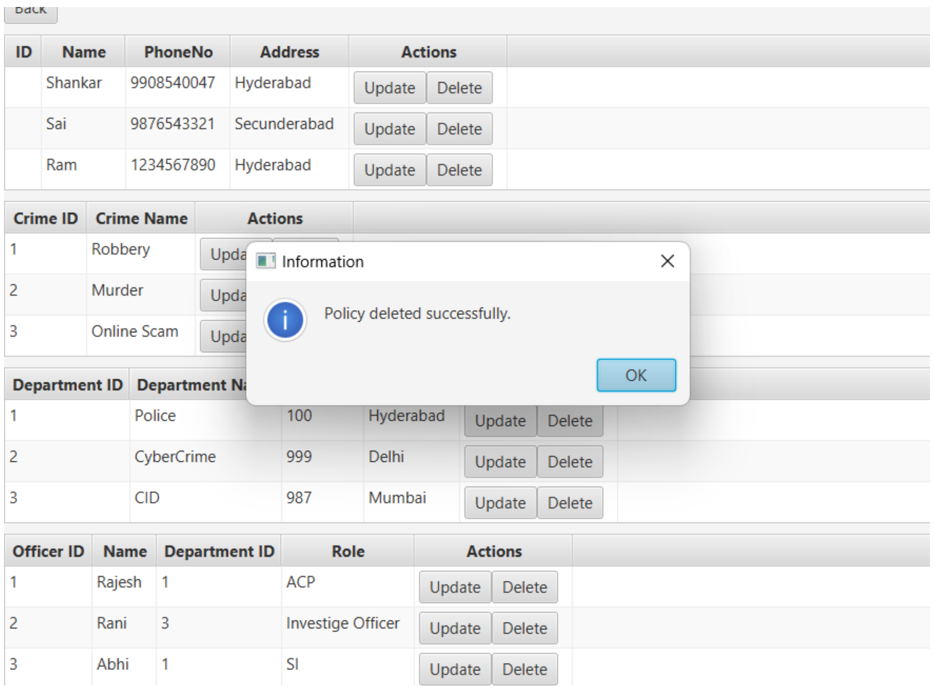
OK

Update

Department ID	Department Name	He		
1	Police	100		Update Delete
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

Officer ID	Name	Department ID	Role	Actions
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete
6	Karthik	3	Analyst	Update Delete

Policy ID	Policy Name	Description	Actions
1	DNA matching	Check Suspects for the fingerprint	Update Delete



ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Information

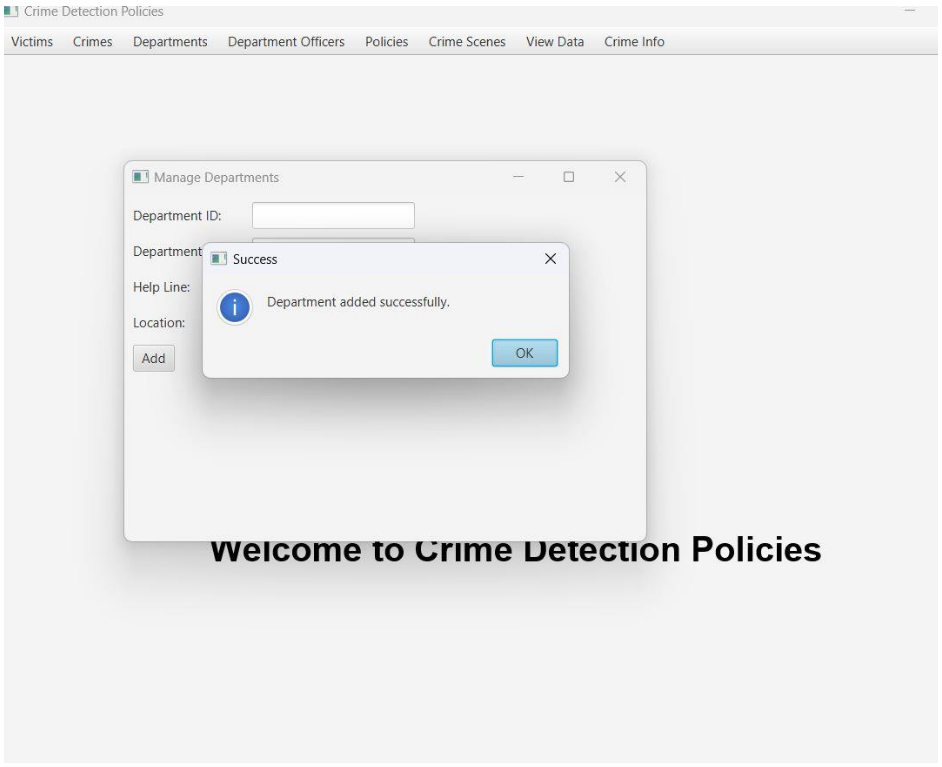
Policy deleted successfully.

OK

Department ID	Department Na			
1	Police	100	Hyderabad	Update Delete
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

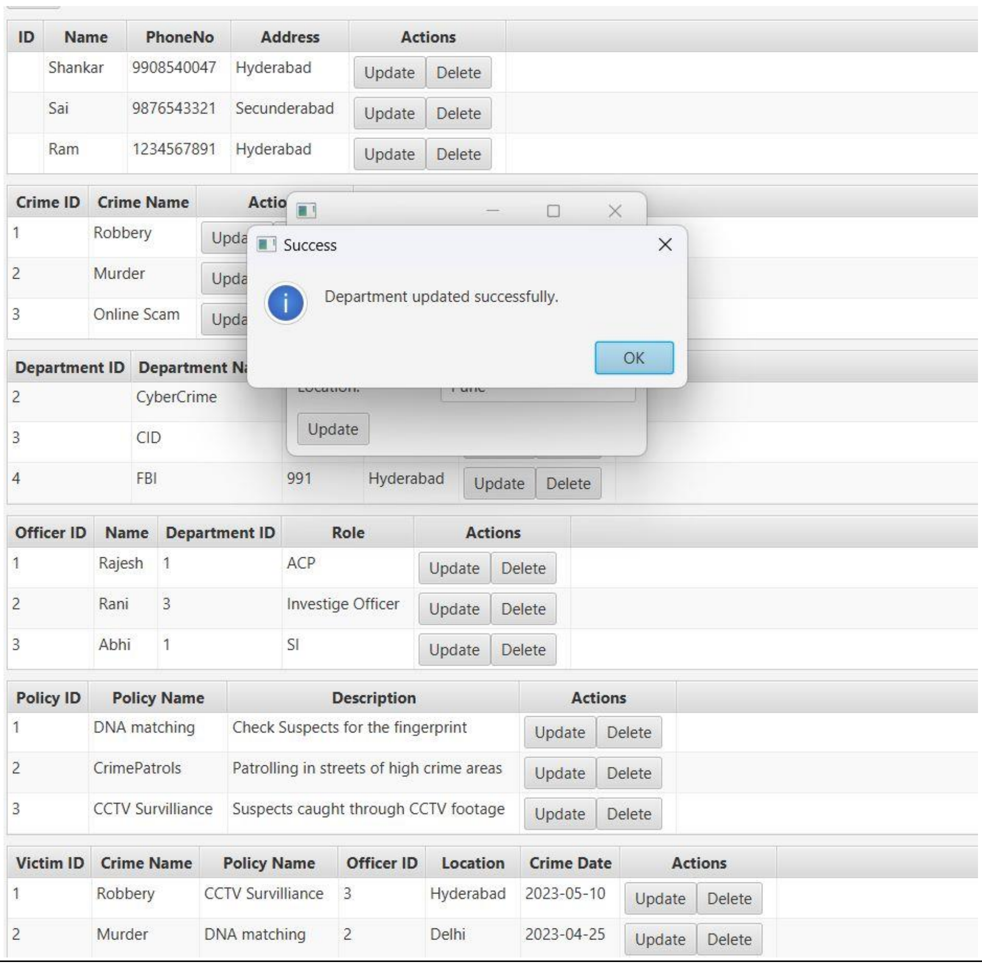
Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Department:



Department added successfully.

Welcome to Crime Detection Policies



ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567891	Hyderabad	Update Delete

Crime ID	Crime Name	Actio
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

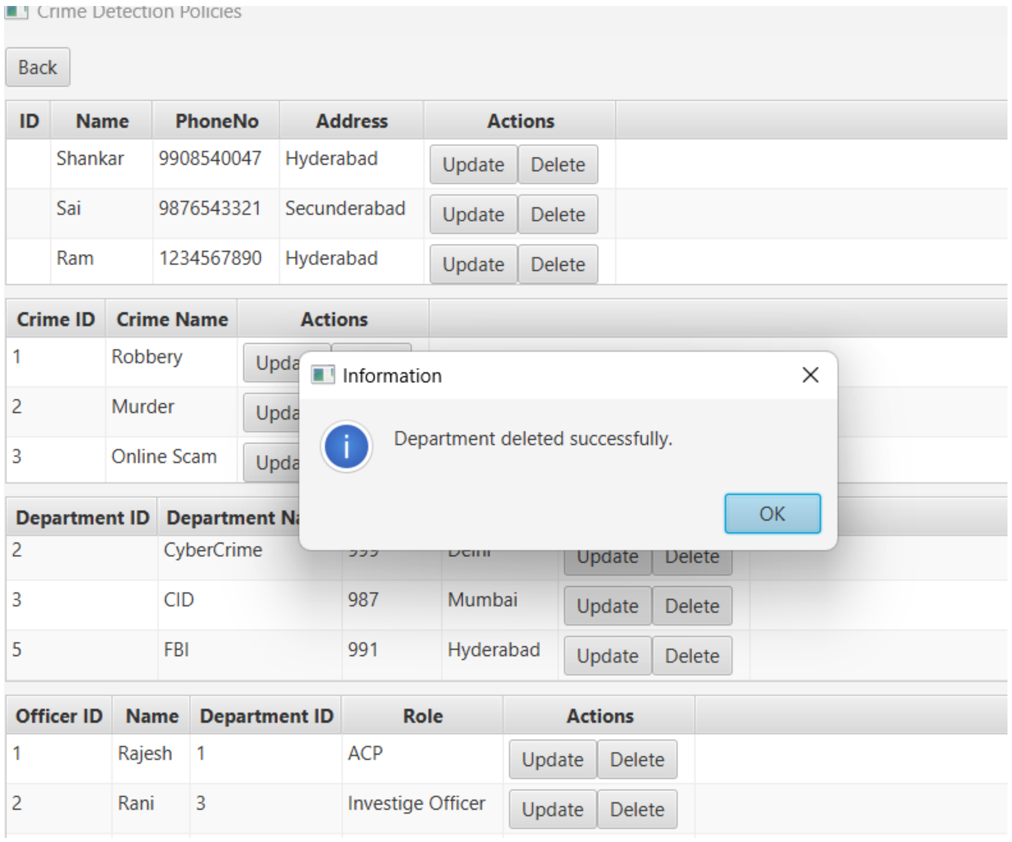
Department updated successfully.

Department ID	Department N			
2	CyberCrime			
3	CID			
4	FBI	991	Hyderabad	Update Delete

Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Policy ID	Policy Name	Description	Actions
1	DNA matching	Check Suspects for the fingerprint	Update Delete
2	CrimePatrols	Patrolling in streets of high crime areas	Update Delete
3	CCTV Survilliance	Suspects caught through CCTV footage	Update Delete

Victim ID	Crime Name	Policy Name	Officer ID	Location	Crime Date	Actions
1	Robbery	CCTV Survilliance	3	Hyderabad	2023-05-10	Update Delete
2	Murder	DNA matching	2	Delhi	2023-04-25	Update Delete



Crime Detection Policies

Back

ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Department ID	Department Na			
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete
5	FBI	991	Hyderabad	Update Delete

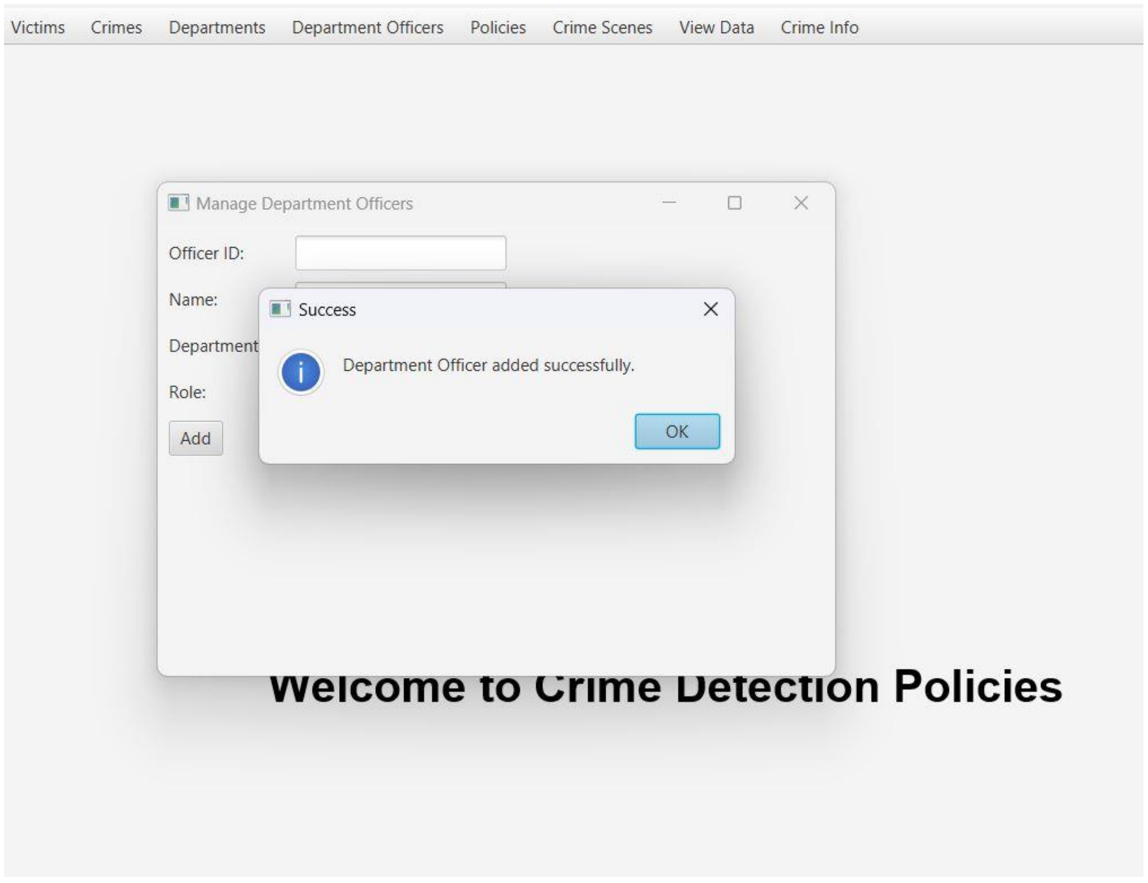
Information

Department deleted successfully.

OK

Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete

Departmentofficer:



Manage Department Officers

Officer ID:

Name:

Department

Role:

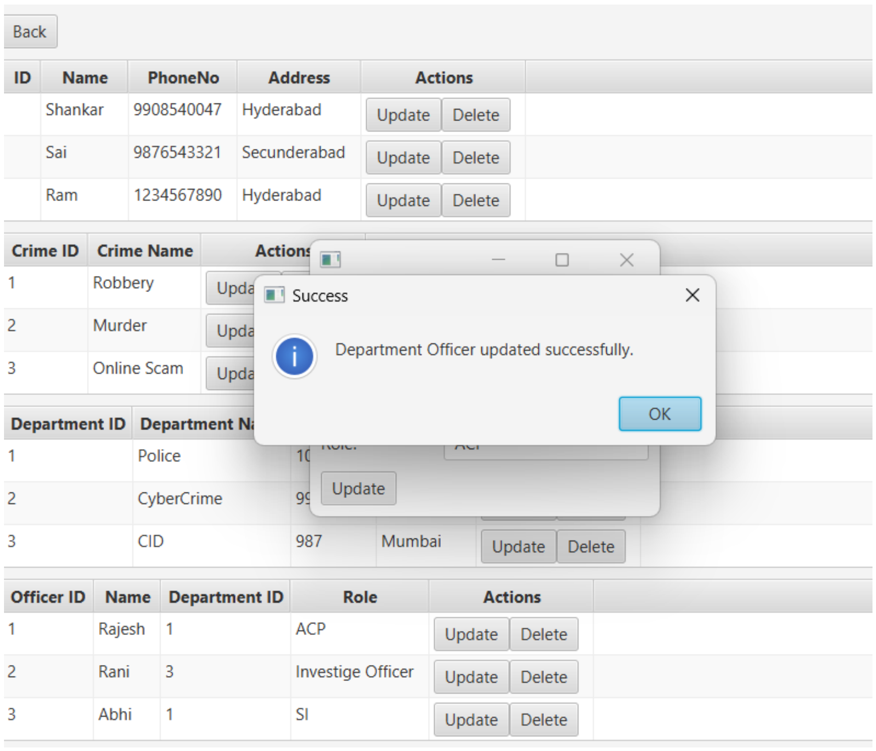
Add

Success

Department Officer added successfully.

OK

Welcome to Crime Detection Policies



Back

ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Department ID	Department N			
1	Police	10		
2	CyberCrime	99		
3	CID	987	Mumbai	Update Delete

Success

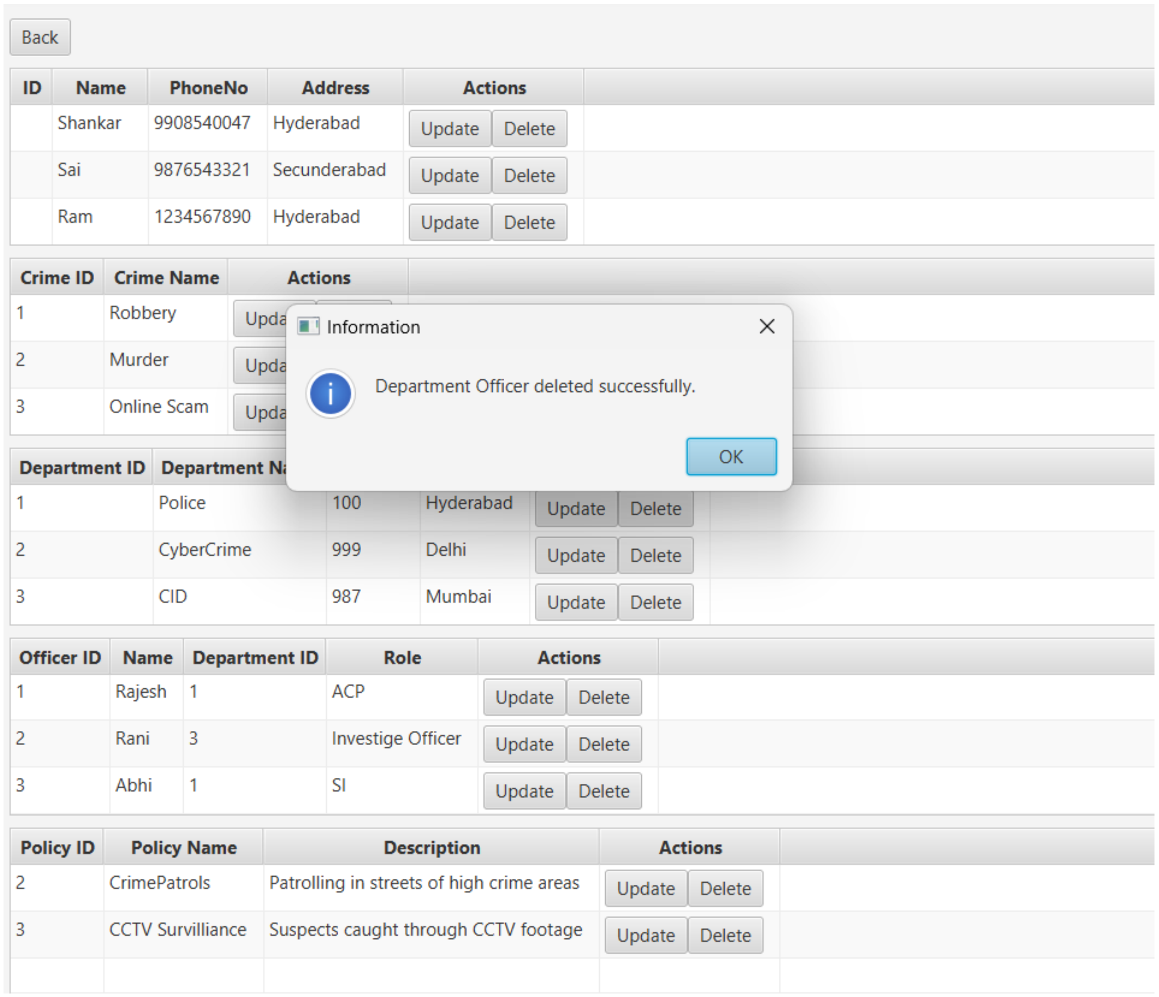
Department Officer updated successfully.

OK

Role: ACP

Update

Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete



Back

ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Information

Department Officer deleted successfully.

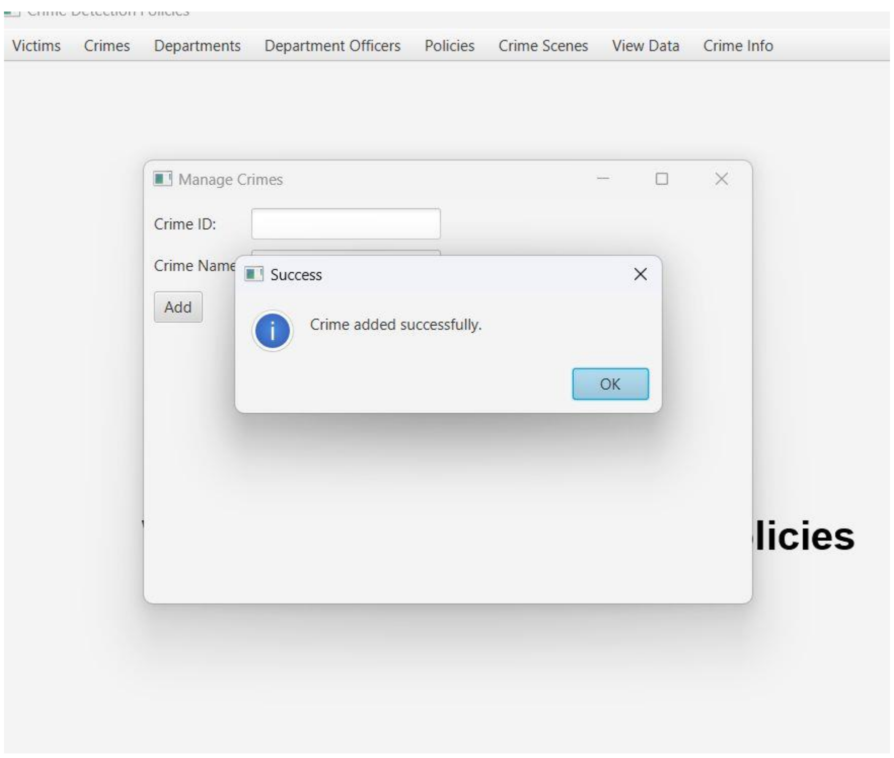
OK

Department ID	Department N			
1	Police	100	Hyderabad	Update Delete
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

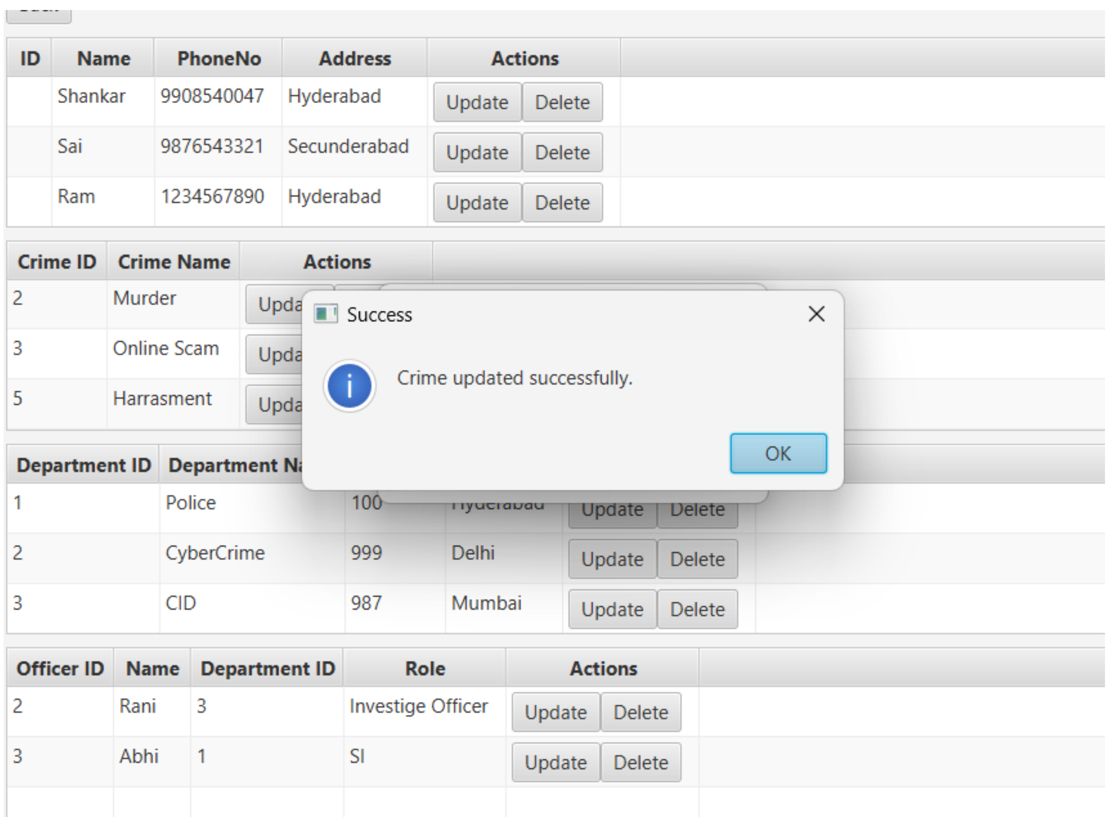
Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Policy ID	Policy Name	Description	Actions
2	CrimePatrols	Patrolling in streets of high crime areas	Update Delete
3	CCTV Survilliance	Suspects caught through CCTV footage	Update Delete

Crime:



Crime added successfully.



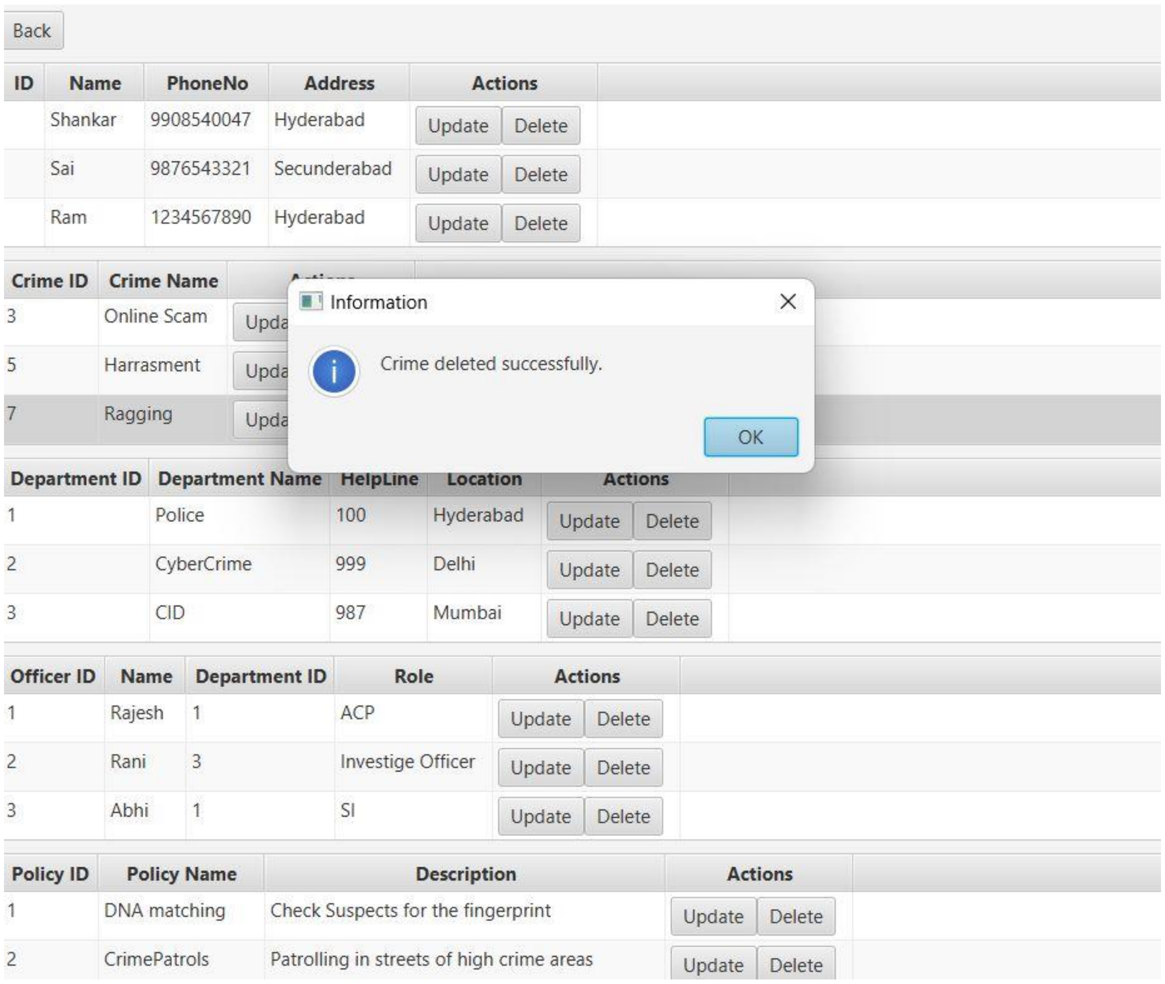
ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
2	Murder	Upda
3	Online Scam	Upda
5	Harrasment	Upda

Department ID	Department N			
1	Police	100		Update Delete
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

Officer ID	Name	Department ID	Role	Actions
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Crime updated successfully.



ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	
3	Online Scam	Upda
5	Harrasment	Upda
7	Ragging	Upda

Information

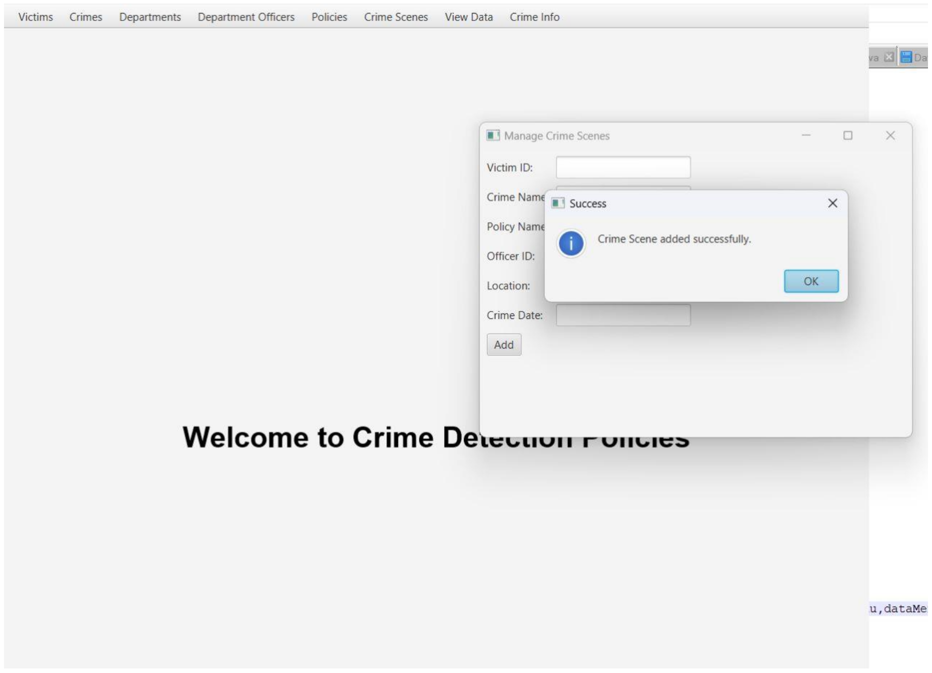
Crime deleted successfully.

Department ID	Department Name	HelpLine	Location	Actions
1	Police	100	Hyderabad	Update Delete
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Policy ID	Policy Name	Description	Actions
1	DNA matching	Check Suspects for the fingerprint	Update Delete
2	CrimePatrols	Patrolling in streets of high crime areas	Update Delete

CrimeScene:



Manage Crime Scenes

Victim ID:

Crime Name

Policy Name

Officer ID:

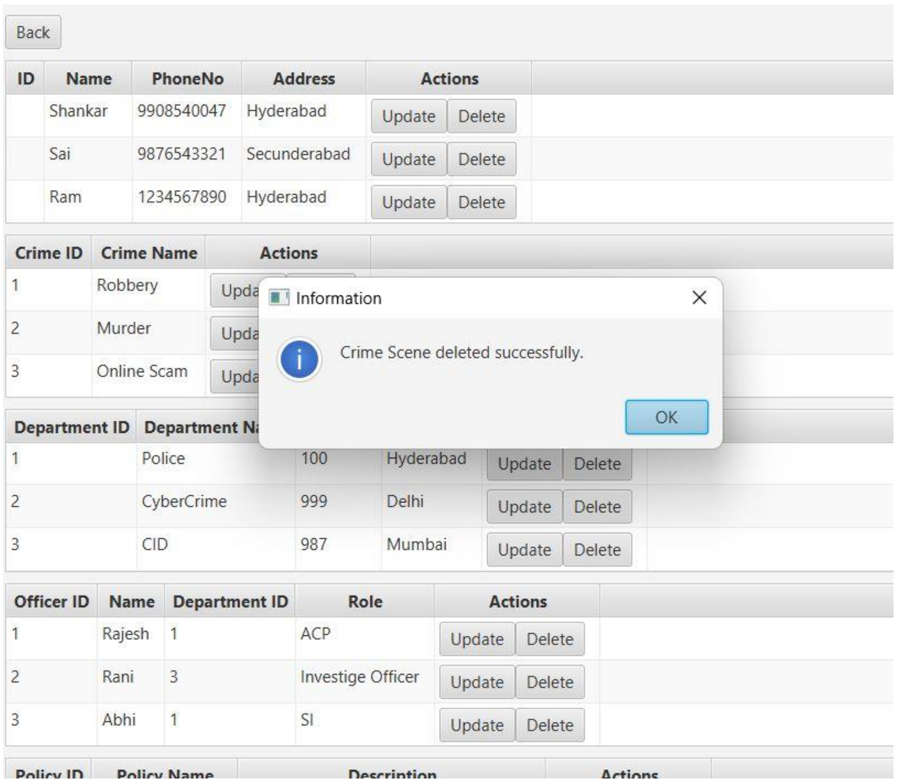
Location:

Crime Date:

Success

Crime Scene added successfully.

Welcome to Crime De



Back

ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
1	Robbery	Upda
2	Murder	Upda
3	Online Scam	Upda

Information

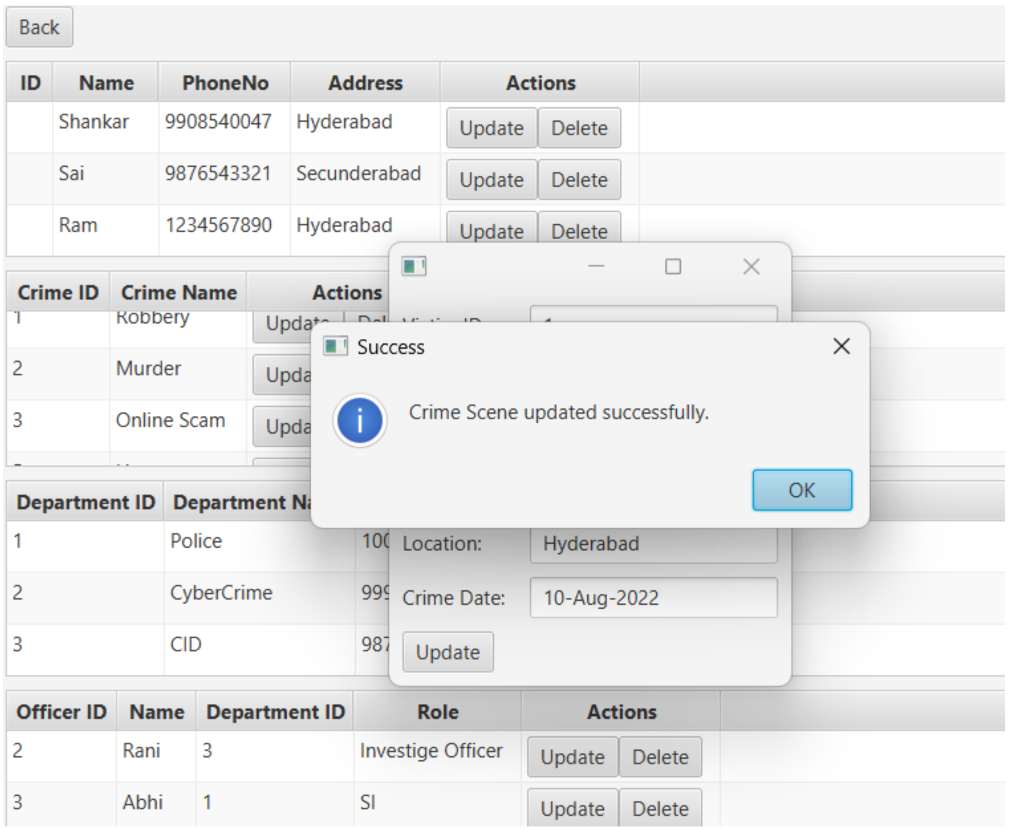
Crime Scene deleted successfully.

OK

Department ID	Department Na			
1	Police	100	Hyderabad	Update Delete
2	CyberCrime	999	Delhi	Update Delete
3	CID	987	Mumbai	Update Delete

Officer ID	Name	Department ID	Role	Actions
1	Rajesh	1	ACP	Update Delete
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Policy ID	Policy Name	Description	Actions



Back

ID	Name	PhoneNo	Address	Actions
	Shankar	9908540047	Hyderabad	Update Delete
	Sai	9876543321	Secunderabad	Update Delete
	Ram	1234567890	Hyderabad	Update Delete

Crime ID	Crime Name	Actions
1	Robbery	Update
2	Murder	Upda
3	Online Scam	Upda

Success

Crime Scene updated successfully.

OK

Location: Hyderabad

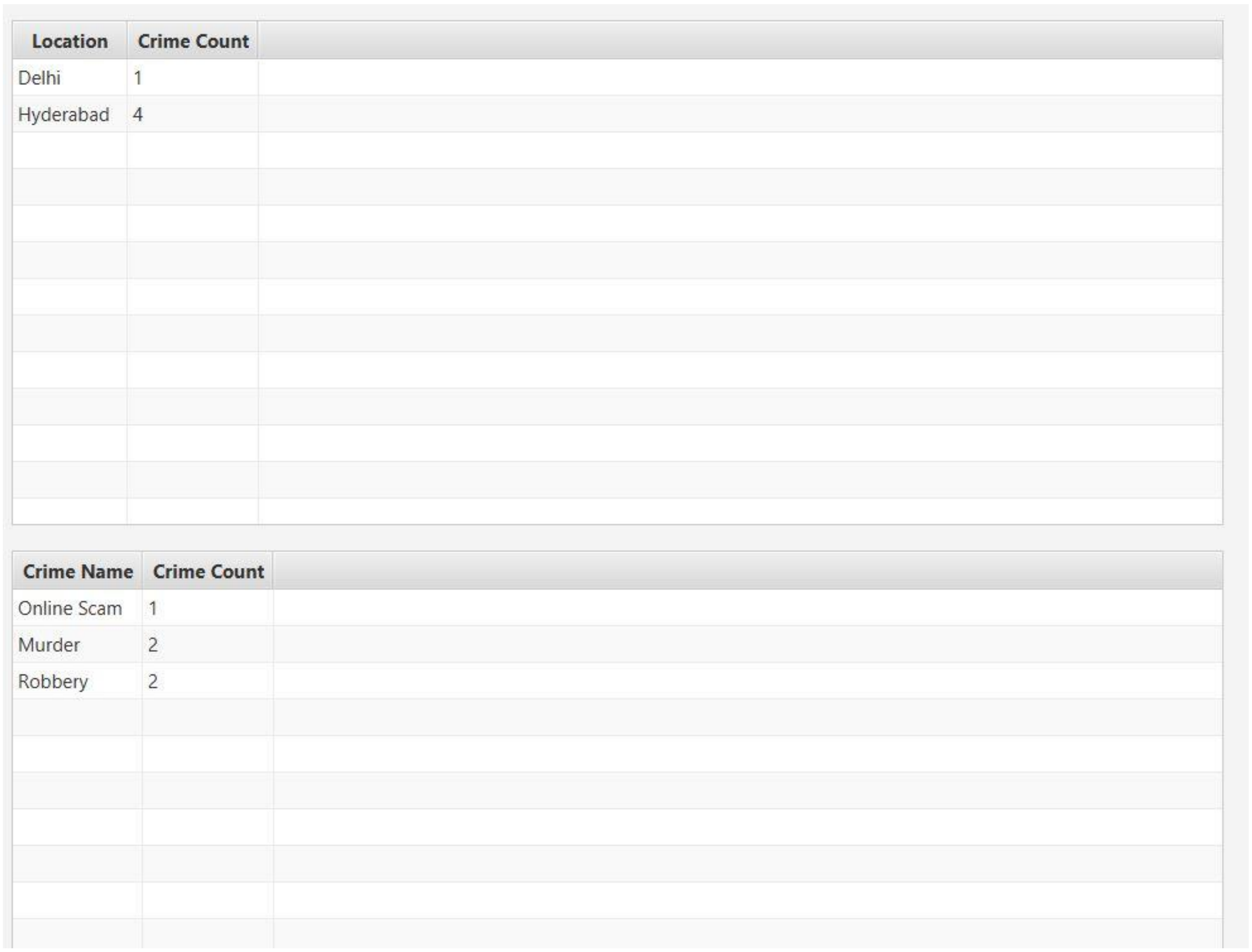
Crime Date: 10-Aug-2022

Update

Department ID	Department Na	
1	Police	10
2	CyberCrime	99
3	CID	98

Officer ID	Name	Department ID	Role	Actions
2	Rani	3	Investige Officer	Update Delete
3	Abhi	1	SI	Update Delete

Crime Count by location and crime type:’



Location	Crime Count
Delhi	1
Hyderabad	4

Crime Name	Crime Count
Online Scam	1
Murder	2
Robbery	2

Result:

I have successfully completed the mini-project “Crime detection policies”.

DISCUSSION AND FUTURE WORK

This project contains the min data required for any department to solve any case , in future Work we can also add about the total of the cases and enlist the various categoriesof the Case which have been divided based on the crime done .

REFERENCES:

1. Abraham Silberschatz, Henry F Korth, S. Sudarshan, Database System Concepts, 6thEdition, McGraw-Hill International Edition, 2010.