**Crime Analysis Platform**

**A MINI-PROJECT BY:**

**Laksheta SV 230701161**

**Miruthula KG 230701183**

***in partial fulfillment of the award of the degree***

***OF***

## *BACHELOR OF ENGINEERING*

**IN**

## COMPUTER SCIENCE AND ENGINEERING



**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**

**An Autonomous Institute**

**CHENNAI**

**NOVEMBER 2024**

# BONAFIDE CERTIFICATE

Certified that this project **“Crime analysis platform”** is the bonafide work of **“LAKSHETA SV(230701161) ,MIRUTHULA KG(230701183)”** who carried out the project work under my supervision.

Submitted for the practical examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

I would like to extend my sincere gratitude to everyone who has contributed to the successful completion of this mini-project.

First and foremost, I am deeply thankful to my professor, **Mrs. K. Maheshmeena**, my project advisor, for their invaluable guidance, insightful feedback, and continuous support throughout the duration of this mini-project. Their expertise and encouragement have been instrumental in shaping my research and bringing this mini-project to completion.

I would also like to express my appreciation to the faculty and staff of the **Computer Science and Engineering Department** at **Rajalakshmi Engineering College** for providing the necessary resources and a conductive learning environment. We express our sincere thanks to **Dr. P. Kumar, M.E., Ph.D.,** Professor, and Head of the Department, Computer Science and Engineering, for his guidance and encouragement throughout the project work.

My heartfelt thanks go to my peers and friends for their collaboration, constructive criticism, and moral support.

Thank you all for your contributions, both direct and indirect, to the success of this project.

# ABSTRACT

The Crime Data Analysis System is a database-driven application designed to assist law enforcement agencies, policymakers, and researchers in managing crime-related data. This system serves as a comprehensive platform for recording, analyzing, and retrieving information about crimes, offenders, incident reports, and officer activities.

The project tests the ability to design and implement a scalable and relational database system, incorporating real-time data analysis and visualization to support crime prevention and resolution. While there is guidance in the development phase, the project showcases independent problem-solving and technical skills.

Our system aims to streamline crime data management and provide actionable insights. Crimes are recorded with details such as type, location, and date. Offenders' profiles are maintained, including risk levels and prior offenses. Reports on incidents and associated officer activities are tracked, ensuring accountability and transparency.

The system enables law enforcement agencies to access detailed crime data, analyze patterns, and make informed decisions. By facilitating seamless data storage and retrieval, the Crime Data Analysis System enhances the efficiency of crime tracking and resolution while supporting advanced analytical capabilities like identifying crime hotspots and offender patterns.

**TABLE OF CONTENTS**

## 1. INTRODUCTION

**1.1 INTRODUCTION**

**1.2 IMPLEMENTATION**

**1.3 SCOPE OF THE PROJECT**

**1.4 WEBSITE FEATURES**

## 2. SYSTEM SPECIFICATION

**2.1 HARDWARE SPECIFICATION**

**2.2 SOFTWARE SPECIFICATION**

## 3. SAMPLE CODE :

**3.1 Dasboard design**

**3.2 Main page design**

**3.3 Crime records page design**

**3.4 Incident records page design**

**3.5 Offender profile page design**

**3.6 Main controller**

**3.7 Dashboard controller**

**3.8 Crime records page controller**

**3.9 Incident reports controller**

**3.10 Offender profile controller**

**3.11 Crime model**

**3.12 Crime data model**

**3.13 Incident reports model**

**3.14 Offender model**

**3.15 Database jdbc connection**

**3.16 Database Mysql files**

**3.17 Main.java**

## 4. SNAPSHOTS :

**4.1 Dashboard page**

**4.2 Crime records page**

**4.3 Incident report page**

**4.4 Inserted new record**

**4.5 Deleted crime record**

1. **CONCLUSION**

1. **REFERENCES**

# INTRODUCTION

**1.1 Introduction**

The **Crime Data Analysis System** is designed to assist law enforcement agencies, policymakers, and researchers in managing and analyzing crime-related data. This platform offers a structured approach to store and retrieve crime information, offender profiles, incident reports, and related data. The system aims to enhance the efficiency of crime management, enable advanced analysis, and support decision-making processes to improve public safety.

**1.2 Implementation**

The Crime Data Analysis System is implemented using:

- **JavaFX:** For building the graphical user interface (GUI), providing an intuitive and interactive experience for end-users.

**- MySQL**: For managing relational data, including crimes, offenders, and reports, ensuring data integrity and scalability.

- **JDBC**: To connect the JavaFX frontend with the MySQL database, facilitating efficient data retrieval and manipulation.

**1.3 Scope of the Project**

The system provides a comprehensive solution for crime data management by integrating various functionalities, including data storage, querying, and visualization. It is designed to assist law enforcement agencies in tracking crimes, analyzing patterns, and allocating resources effectively. The platform's scalable structure allows for future enhancements, such as integrating predictive analytics or geospatial analysis.

**1.4 Features**

- **Crime Records Management**: Ability to store, retrieve, and update details of reported crimes.

**- Offender Profiling**: Detailed profiles of offenders, including risk assessment and history of offenses.

- **Incident Reporting**: Record and track the status of crime investigations.

- **Location Tracking**: Geographical representation of crime hotspots.

- **Analytics Dashboard**: Displays trends and summaries of crime data.

# 

# SYSTEM SPECIFICATIONS

**2.1 HARDWARE SPECIFICATIONS:**

PROCESSOR : Intel i5

MEMORY SIZE : 4GB(Minimum)

HARD DISK : 500 GB of free space

**2.2 SOFTWARE SPECIFICATIONS:**

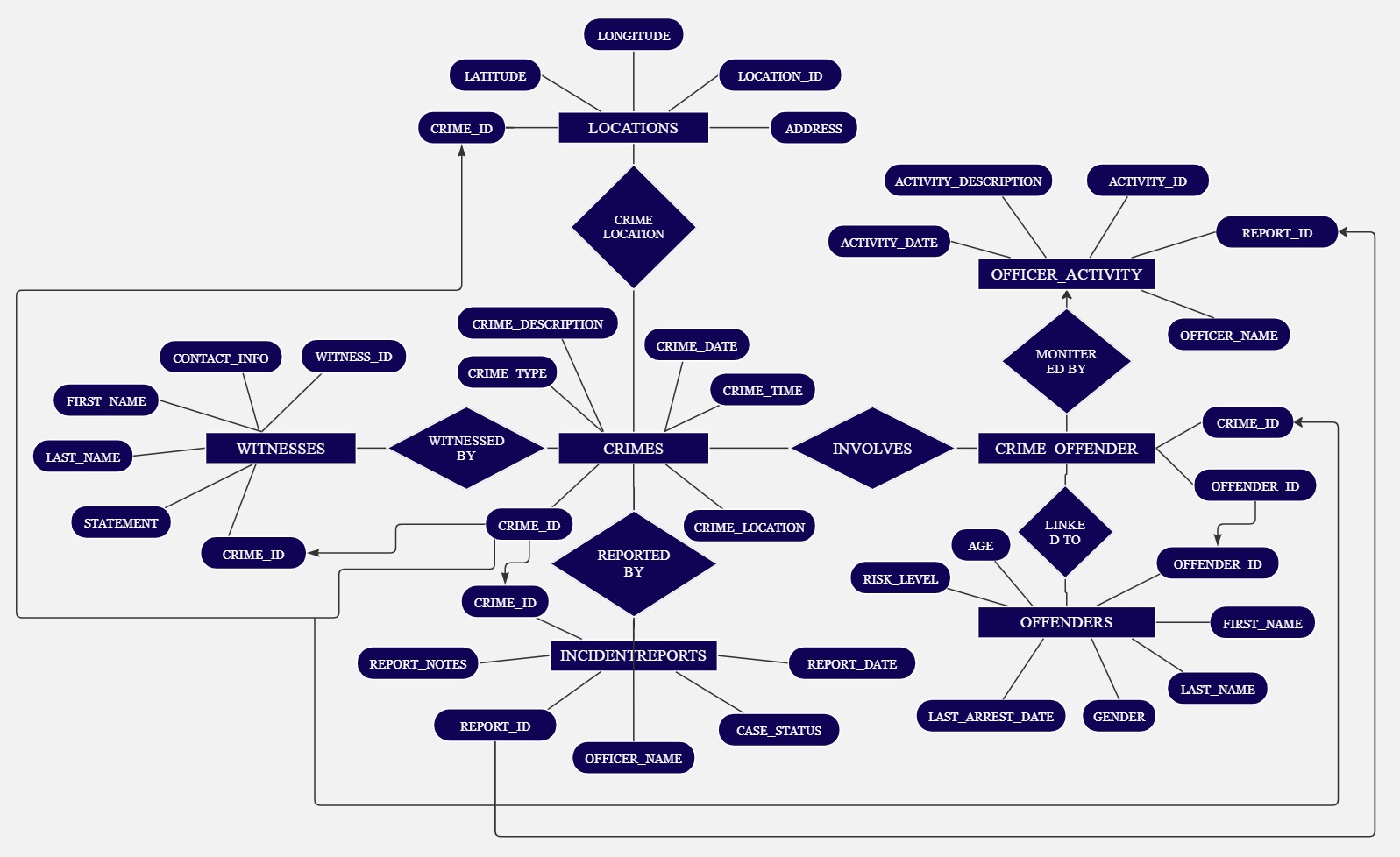
PROGRAMMING LANGUAGE : Java, MySQL

FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 10

**2.3 ER Diagram :**

****

**2.4 NORMALIZATION:**

**Normalization** in a database ensures the data is organized efficiently by reducing redundancy and dependency. It involves structuring a database into tables and establishing relationships based on rules known as normal forms (NF). For Crime Data Analysis System, normalization ensures that the data is logically arranged, maintains integrity, and avoids duplication.

### ****1. First Normal Form (1NF): Eliminate Repeating Groups****

* Ensure each table has a unique identifier (primary key).
* Each column contains atomic values (single, indivisible values).
* Each record (row) contains unique data.

**Example: Issues in Raw Data**

| **Crime\_ID** | **Crime\_Type** | **Crime\_Location** | **Witness1\_Name** | **Witness2\_Name** |
| --- | --- | --- | --- | --- |
| 1 | Burglary | 123 Main St. | John Doe | Jane Doe |

**Normalization into 1NF** Break down repeating groups into separate rows or tables:

* Create a Witnesses table to separate witness information.

**Normalized Tables**:

1. **Crimes Table**

| **Crime\_ID** | **Crime\_Type** | **Crime\_Location** |
| --- | --- | --- |
| 1 | Burglary | 123 Main St. |

1. **Witnesses Table**

| **Witness\_ID** | **Crime\_ID** | **Witness\_Name** |
| --- | --- | --- |
| 1 | 1 | John Doe |
| 2 | 1 | Jane Doe |

### ****2. Second Normal Form (2NF): Remove Partial Dependencies****

* Ensure all non-key attributes are fully dependent on the primary key.
* Eliminate attributes that depend only on part of a composite primary key.

**Example: Issues in 1NF Data**

| **Crime\_ID** | **Offender\_ID** | **Offender\_Name** | **Crime\_Type** |
| --- | --- | --- | --- |
| 1 | 101 | John Smith | Burglary |

Here, Offender\_Name depends only on Offender\_ID and not the composite key (Crime\_ID, Offender\_ID).

**Normalization into 2NF**

* Separate offender details into their own table.

**Normalized Tables**:

1. **Crimes Table**

| **Crime\_ID** | **Crime\_Type** |
| --- | --- |
| 1 | Burglary |

1. **Offenders Table**

| **Offender\_ID** | **Offender\_Name** |
| --- | --- |
| 101 | John Smith |

1. **Crime\_Offender Table** (Associative Table)

| **Crime\_ID** | **Offender\_ID** |
| --- | --- |
| 1 | 101 |

### ****3. Third Normal Form (3NF): Remove Transitive Dependencies****

* Ensure all non-key attributes are directly dependent on the primary key, not other non-key attributes.

**Example: Issues in 2NF Data**

| **Crime\_ID** | **Location\_ID** | **Address** | **City** |
| --- | --- | --- | --- |
| 1 | 201 | 123 Main St. | Gotham |

Here, City is dependent on Address, which is not the primary key.

**Normalization into 3NF**

* Separate location details into a new table.

**Normalized Tables**:

1. **Crimes Table**

| **Crime\_ID** | **Location\_ID** |
| --- | --- |
| 1 | 201 |

1. **Locations Table**

| **Location\_ID** | **Address** | **City** |
| --- | --- | --- |
| 201 | 123 Main St. | Gotham |

### ****4. Higher Normal Forms (Optional)****

If further optimization is needed, consider:

* **Boyce-Codd Normal Form (BCNF)**: Resolves any remaining anomalies not addressed in 3NF.
* **Fourth Normal Form (4NF)**: Addresses multi-valued dependencies (rare in most applications).

### ****Normalized Schema for Crime Analysis Project****

After applying normalization:

1. **Crimes Table**

| **Crime\_ID** | **Crime\_Type** | **Crime\_Date** | **Crime\_Time** | **Crime\_Description** | **Location\_ID** |
| --- | --- | --- | --- | --- | --- |

1. **Offenders Table**

| **Offender\_ID** | **First\_Name** | **Last\_Name** | **Age** | **Gender** | **Prior\_Offenses** | **Risk\_Level** | **Last\_Arrest\_Date** |
| --- | --- | --- | --- | --- | --- | --- | --- |

1. **Crime\_Offender Table**

| **Crime\_ID** | **Offender\_ID** |
| --- | --- |

1. **Locations Table**

| **Location\_ID** | **Address** | **Latitude** | **Longitude** |
| --- | --- | --- | --- |

1. **IncidentReports Table**

| **Report\_ID** | **Crime\_ID** | **Officer\_Name** | **Case\_Status** | **Report\_Notes** | **Report\_Date** |
| --- | --- | --- | --- | --- | --- |

1. **Witnesses Table**

| **Witness\_ID** | **Crime\_ID** | **First\_Name** | **Last\_Name** | **Statement** | **Contact\_Info** |
| --- | --- | --- | --- | --- | --- |

1. **OfficerActivity Table**

| **Activity\_ID** | **Officer\_Name** | **Report\_ID** | **Activity\_Date** | **Activity\_Description** |
| --- | --- | --- | --- | --- |

### ****Benefits of Normalization in Crime Data Analysis****

1. **Reduced Data Redundancy**: Data is stored once and referenced, avoiding duplicates.
2. **Improved Data Integrity**: Ensures consistency and accuracy by maintaining clear relationships.
3. **Easier Data Maintenance**: Changes are easier to implement without cascading errors.
4. **Enhanced Query Performance**: Well-structured tables speed up data retrieval.
5. **Scalability**: The system can handle additional data and complexity without significant rework.

This structured approach will help your Crime Data Analysis System maintain a clean, efficient, and robust database. Let me know if you'd like further assistance!

**3.PROGRAM CODE**

## 3.1 Main page design :

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.VBox?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.layout.HBox?>

<?import javafx.geometry.Insets?>

<VBox xmlns:fx="http://javafx.com/fxml" fx:controller="com.crimedata.controllers.MainController" spacing="10" style="-fx-background-color: #e0e0e0;">

    <!-- Header Section -->

    <HBox alignment="CENTER" spacing="20" style="-fx-background-color: #A7C7E7; -fx-padding: 15;">

        <Button text="Dashboard" onAction="#handleDashboard" style="-fx-font-size: 14px; -fx-background-color: #ffffff;" />

        <Button text="Crime Records" onAction="#handleCrimeRecords" style="-fx-font-size: 14px; -fx-background-color: #ffffff;" />

        <Button text="Incident Reports" onAction="#handleIncidentReports" style="-fx-font-size: 14px; -fx-background-color: #ffffff;" />

    </HBox>

    <!-- Content Pane where pages will load -->

    <Pane fx:id="contentPane" style="-fx-background-color: #f0f0f0; -fx-padding: 20;" VBox.vgrow="ALWAYS">

        <!-- This pane will be filled with dynamic content based on navigation -->

    </Pane>

</VBox>

**3.2 Dashboard page design:**

<?xml version="1.0" encoding="UTF-8"?>

## <?import javafx.scene.layout.VBox?>

## <?import javafx.scene.control.Label?>

## <?import javafx.scene.text.Font?>

## <?import javafx.scene.layout.HBox?>

## <?import javafx.scene.layout.BorderPane?>

## 

## <BorderPane xmlns:fx="http://javafx.com/fxml" fx:controller="com.crimedata.controllers.DashboardController"  style="-fx-background-color: #f7f7f7;">

## <top>

## <HBox alignment="CENTER" style="-fx-background-color: #3f51b5; -fx-padding: 20;">

## <Label text="Crime Data Analysis System" style="-fx-font-size: 24px; -fx-text-fill: white; -fx-font-weight: bold;"/>

## </HBox>

## </top>

## <!-- Total Offenders Card -->

## <VBox alignment="CENTER" style="-fx-background-color: #fce4ec; -fx-padding: 20; -fx-border-radius: 10; -fx-background-radius: 10; -fx-min-width: 150;">

## <Label text="Total Offenders" style="-fx-font-size: 18px; -fx-font-weight: bold;"/>

## <Label fx:id="totalOffenders" text="0" style="-fx-font-size: 30px; -fx-font-weight: bold; -fx-text-fill: #d81b60;"/>

## </VBox>

## 

## 

## 

## <!-- Open Cases Card -->

## <VBox alignment="CENTER" style="-fx-background-color: #e8f5e9; -fx-padding: 20; -fx-border-radius: 10; -fx-background-radius: 10; -fx-min-width: 150;">

## <Label text="Open Cases" style="-fx-font-size: 18px; -fx-font-weight: bold;"/>

## <Label fx:id="openCases" text="0" style="-fx-font-size: 30px; -fx-font-weight: bold; -fx-text-fill: #388e3c;"/>

## </VBox>

## </HBox>

## 

## 

## <HBox alignment="CENTER" spacing="20" style="-fx-padding: 20;">

## <Label text="Explore more crime statistics" style="-fx-font-size: 16px;"/>

## </HBox>

## </VBox>

## </center>

## 

## 

## <bottom>

## <HBox alignment="CENTER" style="-fx-background-color: #3f51b5; -fx-padding: 10;">

## <Label text="© 2024 Crime Data Analysis System" style="-fx-font-size: 12px; -fx-text-fill: white;"/>

## </HBox>

## </bottom>

## </BorderPane>

## 3.3 Crime Records Page design:

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?

<AnchorPane xmlns:fx="http://javafx.com/fxml" fx:controller="com.crimedata.controllers.CrimeRecordsController">

    <children>

        <!-- Crime Type Field -->

        <Label text="Crime Type:" layoutX="20" layoutY="20"/>

        <TextField fx:id="typeField" layoutX="100" layoutY="20" promptText="Enter crime type"/>

        <!-- Crime Location Field -->

        <Label text="Crime Location:" layoutX="20" layoutY="60"/>

        <TextField fx:id="locationField" layoutX="100" layoutY="60" promptText="Enter crime location"/>

        <!-- Crime Date Field -->

        <Label text="Crime Date:" layoutX="20" layoutY="100"/>

        <TextField fx:id="dateField" layoutX="100" layoutY="100" promptText="Enter crime date"/>

        <!-- Add Crime Button -->

        <Button text="Add Crime" layoutX="20" layoutY="140" onAction="#addCrime"/>

        <!-- Delete Crime Button -->

        <Button text="Delete Crime" layoutX="120" layoutY="140" onAction="#deleteCrime"/>

        <!-- Crime Table -->

        <TableView fx:id="crimeTable" layoutX="20" layoutY="180" prefWidth="460" prefHeight="200">

            <columns>

                <TableColumn fx:id="crimeIdColumn" text="ID" prefWidth="50"/>

                <TableColumn fx:id="crimeTypeColumn" text="Type" prefWidth="130"/>

                <TableColumn fx:id="crimeLocationColumn" text="Location" prefWidth="130"/>

                <TableColumn fx:id="crimeDateColumn" text="Date" prefWidth="130"/>

            </columns>

        </TableView>

    </children>

</AnchorPane>

**3.4 Incident Reports page design :**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Alert?>

<?import javafx.scene.control.Alert.AlertType?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.HBox?>

<GridPane fx:controller="com.crimedata.controllers.IncidentReportsController"

          xmlns:fx="http://javafx.com/fxml/1" alignment="CENTER" hgap="10" vgap="10">

    <!-- Table for displaying incident reports -->

    <TableView fx:id="incidentReportsTable" GridPane.columnSpan="2" GridPane.rowIndex="0">

        <columns>

            <TableColumn fx:id="reportIdColumn" text="Report ID" />

            <TableColumn fx:id="crimeIdColumn" text="Crime ID" />

            <TableColumn fx:id="reportDescriptionColumn" text="Description" />

            <TableColumn fx:id="reportDateColumn" text="Report Date" />

        </columns>

    </TableView>

    <!-- Form fields for adding new reports -->

    <TextField fx:id="reportIdField" promptText="Report ID" GridPane.rowIndex="1" GridPane.columnIndex="0" />

    <TextField fx:id="crimeIdField" promptText="Crime ID" GridPane.rowIndex="1" GridPane.columnIndex="1" />

    <TextField fx:id="descriptionField" promptText="Description" GridPane.rowIndex="2" GridPane.columnIndex="0" GridPane.columnSpan="2" />

    <TextField fx:id="dateField" promptText="Date" GridPane.rowIndex="3" GridPane.columnIndex="0" GridPane.columnSpan="2" />

    <!-- Submit Button -->

    <HBox spacing="10" GridPane.rowIndex="4" GridPane.columnSpan="2" alignment="CENTER">

        <Button text="Submit Report" onAction="#handleSubmitReport"/>

    </HBox>

</GridPane>

3.5 **Offender profiles page design :**

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Alert?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.control.Label?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.input.KeyCode?>

<AnchorPane xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.crimedata.controllers.OffenderProfilesController">

    <!-- Main Layout -->

    <GridPane layoutX="14.0" layoutY="14.0" vgap="15" hgap="15" padding="20">

        <!-- Table for Offender List -->

        <TableView fx:id="offenderTable" GridPane.columnSpan="2" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #f9f9f9;">

            <columns>

                <TableColumn fx:id="offenderIdColumn" text="ID" prefWidth="100" cellValueFactory="new PropertyValueFactory<Offender, Integer>('offenderId')" />

                <TableColumn fx:id="firstNameColumn" text="First Name" prefWidth="150" cellValueFactory="new PropertyValueFactory<Offender, String>('firstName')" />

                <TableColumn fx:id="lastNameColumn" text="Last Name" prefWidth="150" cellValueFactory="new PropertyValueFactory<Offender, String>('lastName')" />

                <TableColumn fx:id="ageColumn" text="Age" prefWidth="100" cellValueFactory="new PropertyValueFactory<Offender, Integer>('age')" />

                <TableColumn fx:id="genderColumn" text="Gender" prefWidth="100" cellValueFactory="new PropertyValueFactory<Offender, String>('gender')" />

                <TableColumn fx:id="priorOffensesCountColumn" text="Prior Offenses Count" prefWidth="150" cellValueFactory="new PropertyValueFactory<Offender, Integer>('priorOffensesCount')" />

                <TableColumn fx:id="riskLevelColumn" text="Risk Level" prefWidth="150" cellValueFactory="new PropertyValueFactory<Offender, String>('riskLevel')" />

                <TableColumn fx:id="lastArrestDateColumn" text="Last Arrest Date" prefWidth="150" cellValueFactory="new PropertyValueFactory<Offender, String>('lastArrestDate')" />

            </columns>

        </TableView>

        <!-- Add Offender Form -->

        <Label text="Add Offender Profile" GridPane.rowIndex="1" style="-fx-font-size: 18px; -fx-font-weight: bold;" />

        <!-- Input Fields -->

        <GridPane GridPane.rowIndex="2" GridPane.columnSpan="2" vgap="10" hgap="10">

            <Label text="First Name" />

            <TextField fx:id="nameField" promptText="Enter First Name" GridPane.columnIndex="1" prefWidth="200.0"/>

            <Label text="Risk Level" GridPane.rowIndex="1" />

            <TextField fx:id="riskLevelField" promptText="Enter Risk Level" GridPane.columnIndex="1" GridPane.rowIndex="1" prefWidth="200.0" />

        </GridPane>

        <!-- Add Offender Button -->

        <HBox GridPane.rowIndex="3" GridPane.columnSpan="2" spacing="15" alignment="center">

            <Button layoutX="300.0" layoutY="450.0" text="Add Offender" onAction="#addOffender" style="-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-weight: bold;" tooltip="Click to Add Offender" />

        </HBox>

    </GridPane>

    <!-- Styling for TableView Rows (Alternating Colors) -->

    <style>

        .table-row-cell:nth-child(odd) {

            -fx-background-color: #f1f1f1;

        }

        .table-row-cell:nth-child(even) {

            -fx-background-color: #ffffff;

        }

        .table-header {

            -fx-background-color: #2196F3;

            -fx-text-fill: white;

        }

    </style>

</AnchorPane>

**Controller pages for javafx**:

**3.6 Main page Controller:**

package com.crimedata.controllers;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.layout.Pane;

import java.io.IOException;

public class MainController {

    @FXML

    private Pane contentPane; // This is the container where pages will load dynamically

    // Method to load a page into the content pane

    private void loadPage(String fxmlFile) {

        try {

            // Load the specified FXML file

            FXMLLoader loader = new FXMLLoader(getClass().getResource("/com/crimedata/views/" + fxmlFile));

            Parent root = loader.load();

            // Clear existing content and set the new content

            contentPane.getChildren().clear();

            contentPane.getChildren().add(root);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    // Event handler for loading the Dashboard page

    @FXML

    private void handleDashboard() {

        loadPage("Dashboard.fxml");

    }

    // Event handler for loading the Crime Records page

    @FXML

    private void handleCrimeRecords() {

        loadPage("CrimeRecords.fxml");

    }

    // Event handler for loading the Offender Profiles page

    @FXML

    private void handleOffenderProfiles() {

        loadPage("OffenderProfiles.fxml");

    }

    // Event handler for loading the Incident Reports page

    @FXML

    private void handleIncidentReports() {

        loadPage("IncidentReports.fxml");

    }

    // Event handler for loading the Analytics page

    @FXML

    private void handleAnalytics() {

        loadPage("Analytics.fxml");

    }

}

**3.7 Dashboard controller page :**

package com.crimedata.controllers;

import com.crimedata.utils.DatabaseUtil;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class DashboardController {

    @FXML

    private Label totalCrimes;

    @FXML

    private Label totalOffenders;

    @FXML

    private Label openCases;

    public void initialize() {

        Connection conn = DatabaseUtil.getConnection();

        if (conn != null) {

            System.out.println("Connection successful!");

        } else {

            System.out.println("Connection failed!");

        }

        loadStatistics();

    }

    private void loadStatistics() {

        try (Connection conn = DatabaseUtil.getConnection()) {

            // Define the queries

            String crimeQuery = "SELECT COUNT(\*) FROM crimes";

            String offenderQuery = "SELECT COUNT(\*) FROM offenders";

            String openCasesQuery = "SELECT COUNT(\*) FROM incidentreports WHERE case\_status = 'Open'";

            // Execute the queries using PreparedStatements

            try (PreparedStatement crimeStmt = conn.prepareStatement(crimeQuery);

                 PreparedStatement offenderStmt = conn.prepareStatement(offenderQuery);

                 PreparedStatement openCasesStmt = conn.prepareStatement(openCasesQuery)) {

                // Get results for crimes count

                ResultSet crimeResult = crimeStmt.executeQuery();

                if (crimeResult.next()) {

                    totalCrimes.setText(String.valueOf(crimeResult.getInt(1)));

                }

                // Get results for offenders count

                ResultSet offenderResult = offenderStmt.executeQuery();

                if (offenderResult.next()) {

                    totalOffenders.setText(String.valueOf(offenderResult.getInt(1)));

                }

                // Get results for open cases count

                ResultSet openCasesResult = openCasesStmt.executeQuery();

                if (openCasesResult.next()) {

                    openCases.setText(String.valueOf(openCasesResult.getInt(1)));

                }

            }

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (Exception e) {

            e.printStackTrace();  // Catch other exceptions as well

        }

    }

}

**3.8 Crime Records controller page:**

package com.crimedata.controllers;

import com.crimedata.models.Crime;

import com.crimedata.utils.DatabaseUtil;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.ButtonType;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.TextField;

import javafx.scene.control.cell.PropertyValueFactory;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class CrimeRecordsController {

    @FXML

    private TableView<Crime> crimeTable;

    @FXML

    private TableColumn<Crime, Integer> crimeIdColumn;

    @FXML

    private TableColumn<Crime, String> crimeTypeColumn;

    @FXML

    private TableColumn<Crime, String> crimeLocationColumn;

    @FXML

    private TableColumn<Crime, String> crimeDateColumn;

    @FXML

    private TextField typeField, locationField, dateField;

    public void initialize() {

        // Setting up columns to bind to Crime properties

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

        crimeTypeColumn.setCellValueFactory(new PropertyValueFactory<>("type"));

        crimeLocationColumn.setCellValueFactory(new PropertyValueFactory<>("location"));

        crimeDateColumn.setCellValueFactory(new PropertyValueFactory<>("date"));

        // Load initial crime data

        loadCrimes();

    }

    private void loadCrimes() {

        crimeTable.getItems().clear();  // Clear existing items

        try (Connection conn = DatabaseUtil.getConnection()) {

            String query = "SELECT \* FROM crimes";

            PreparedStatement stmt = conn.prepareStatement(query);

            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {

                Crime crime = new Crime(rs.getInt("crime\_id"), rs.getString("crime\_type"), rs.getString("crime\_location"), rs.getString("crime\_date"));

                System.out.println("Adding crime: " + crime.getType());  // Debug print

                crimeTable.getItems().add(crime);

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    @FXML

    private void addCrime() {

        try (Connection conn = DatabaseUtil.getConnection()) {

            String query = "INSERT INTO crimes (crime\_type, crime\_location, crime\_date) VALUES (?, ?, ?)";

            PreparedStatement stmt = conn.prepareStatement(query);

            stmt.setString(1, typeField.getText());

            stmt.setString(2, locationField.getText());

            stmt.setString(3, dateField.getText());

            stmt.executeUpdate();

            // Refresh the list after adding a crime

            loadCrimes();

            // Show success alert

            Alert alert = new Alert(AlertType.INFORMATION);

            alert.setTitle("Success");

            alert.setHeaderText("Crime Record Added");

            alert.setContentText("The crime record has been successfully added.");

            alert.showAndWait();

        } catch (Exception e) {

            e.printStackTrace();

            // Show error alert if something goes wrong

            Alert alert = new Alert(AlertType.ERROR);

            alert.setTitle("Error");

            alert.setHeaderText("Failed to Add Crime");

            alert.setContentText("An error occurred while adding the crime record. Please try again.");

            alert.showAndWait();

        }

    }

    @FXML

    private void deleteCrime() {

        Crime selectedCrime = crimeTable.getSelectionModel().getSelectedItem();

        if (selectedCrime != null) {

            // Show confirmation alert

            Alert confirmationAlert = new Alert(Alert.AlertType.CONFIRMATION);

            confirmationAlert.setTitle("Confirmation");

            confirmationAlert.setHeaderText("Delete Crime Record");

            confirmationAlert.setContentText("Are you sure you want to delete the selected crime record?");

            // Wait for user response

            if (confirmationAlert.showAndWait().get() == ButtonType.OK) {

                try (Connection conn = DatabaseUtil.getConnection()) {

                    // Use the correct column name in the DELETE query

                    String query = "DELETE FROM crimes WHERE crime\_id = ?";

                    PreparedStatement stmt = conn.prepareStatement(query);

                    stmt.setInt(1, selectedCrime.getId());  // Ensure you are passing the correct 'crime\_id'

                    int rowsAffected = stmt.executeUpdate();

                    // If rows are affected, proceed with the deletion

                    if (rowsAffected > 0) {

                        crimeTable.getItems().remove(selectedCrime);

                        // Show success alert

                        Alert successAlert = new Alert(AlertType.INFORMATION);

                        successAlert.setTitle("Success");

                        successAlert.setHeaderText("Crime Record Deleted");

                        successAlert.setContentText("The crime record has been successfully deleted.");

                        successAlert.showAndWait();

                    } else {

                        // If no rows are affected, something went wrong

                        Alert errorAlert = new Alert(AlertType.ERROR);

                        errorAlert.setTitle("Error");

                        errorAlert.setHeaderText("Failed to Delete Crime");

                        errorAlert.setContentText("No crime record was deleted. Please try again.");

                        errorAlert.showAndWait();

                    }

                } catch (Exception e) {

                    e.printStackTrace();

                    // Show error alert if something goes wrong

                    Alert errorAlert = new Alert(AlertType.ERROR);

                    errorAlert.setTitle("Error");

                    errorAlert.setHeaderText("Failed to Delete Crime");

                    errorAlert.setContentText("An error occurred while deleting the crime record. Please try again.");

                    errorAlert.showAndWait();

                }

            }

        } else {

            // Show alert if no item is selected

            Alert noSelectionAlert = new Alert(AlertType.WARNING);

            noSelectionAlert.setTitle("No Selection");

            noSelectionAlert.setHeaderText("No Crime Record Selected");

            noSelectionAlert.setContentText("Please select a crime record to delete.");

            noSelectionAlert.showAndWait();

        }

    }

**}**

**3.9 Incident Reports controller :**

package com.crimedata.controllers;

import com.crimedata.models.IncidentReport;

import com.crimedata.utils.DatabaseUtil;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.TextField;

import javafx.scene.control.cell.PropertyValueFactory;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

public class IncidentReportsController {

    @FXML

    private TableView<IncidentReport> incidentReportsTable;

    @FXML

    private TableColumn<IncidentReport, Integer> reportIdColumn;

    @FXML

    private TableColumn<IncidentReport, Integer> crimeIdColumn;

    @FXML

    private TableColumn<IncidentReport, String> reportDescriptionColumn;

    @FXML

    private TableColumn<IncidentReport, String> reportDateColumn;

    @FXML

    private TextField reportIdField;

    @FXML

    private TextField crimeIdField;

    @FXML

    private TextField descriptionField;

    @FXML

    private TextField dateField;

    private List<IncidentReport> reportsList = new ArrayList<>();

    @FXML

    public void initialize() {

        // Set up TableView columns

        reportIdColumn.setCellValueFactory(new PropertyValueFactory<>("report\_Id"));

        crimeIdColumn.setCellValueFactory(new PropertyValueFactory<>("crime\_Id"));

        reportDescriptionColumn.setCellValueFactory(new PropertyValueFactory<>("report\_notes"));

        reportDateColumn.setCellValueFactory(new PropertyValueFactory<>("report\_date"));

        // Load incident reports from the database

        loadReports();

    }

    @FXML

    private void handleSubmitReport() {

        try {

            // Get input values

            int reportId = Integer.parseInt(reportIdField.getText());

            int crimeId = Integer.parseInt(crimeIdField.getText());

            String description = descriptionField.getText();

            String date = dateField.getText();

            // Insert the new report into the database

            addReportToDatabase(reportId, crimeId, description, date);

            // Clear the input fields after submission

            reportIdField.clear();

            crimeIdField.clear();

            descriptionField.clear();

            dateField.clear();

            // Reload reports from the database

            loadReports();

            // Optionally, show success message

            Alert alert = new Alert(AlertType.INFORMATION);

            alert.setTitle("Report Submitted");

            alert.setHeaderText("Incident Report Added");

            alert.setContentText("The incident report has been successfully added.");

            alert.showAndWait();

        } catch (NumberFormatException e) {

            // Handle invalid input (non-numeric for ID fields)

            Alert alert = new Alert(AlertType.ERROR);

            alert.setTitle("Invalid Input");

            alert.setHeaderText("Input Error");

            alert.setContentText("Please enter valid numbers for Report ID and Crime ID.");

            alert.showAndWait();

        }

    }

    private void addReportToDatabase(int reportId, int crimeId, String description, String date) {

        try (Connection conn = DatabaseUtil.getConnection()) {

            // SQL query to insert the report into the database

            String query = "INSERT INTO incidentreports (report\_id, crime\_id, report\_notes, report\_date) VALUES (?, ?, ?, ?)";

            PreparedStatement stmt = conn.prepareStatement(query);

            stmt.setInt(1, reportId);

            stmt.setInt(2, crimeId);

            stmt.setString(3, description);

            stmt.setString(4, date);

            stmt.executeUpdate();

        } catch (Exception e) {

            e.printStackTrace();

            Alert alert = new Alert(AlertType.ERROR);

            alert.setTitle("Database Error");

            alert.setHeaderText("Error Adding Report");

            alert.setContentText("There was an error adding the incident report to the database.");

            alert.showAndWait();

        }

    }

    private void loadReports() {

        reportsList.clear(); // Clear the list before adding updated data

        try (Connection conn = DatabaseUtil.getConnection()) {

            String query = "SELECT \* FROM incidentreports";

            PreparedStatement stmt = conn.prepareStatement(query);

            ResultSet rs = stmt.executeQuery();

            // Loop through the result set and populate the reports list

            while (rs.next()) {

                IncidentReport report = new IncidentReport(

                        rs.getInt("report\_id"),

                        rs.getInt("crime\_id"),

                        rs.getString("report\_notes"),

                        rs.getString("report\_date")

                );

                reportsList.add(report);

            }

            // Bind the list to the TableView

            incidentReportsTable.getItems().setAll(reportsList);

        } catch (Exception e) {

            e.printStackTrace();

            Alert alert = new Alert(AlertType.ERROR);

            alert.setTitle("Database Error");

            alert.setHeaderText("Error Loading Reports");

            alert.setContentText("There was an error loading the incident reports from the database.");

            alert.showAndWait();

        }

    }

}

**3.10 Offender profiles controller page:**

package com.crimedata.controllers;

import com.crimedata.models.Offender;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.TextField;

public class OffenderProfilesController {

    @FXML

    private TableView<Offender> offenderTable;

    @FXML

    private TableColumn<Offender, Integer> offenderIdColumn;

    @FXML

    private TableColumn<Offender, String> firstNameColumn;

    @FXML

    private TableColumn<Offender, String> lastNameColumn;

    @FXML

    private TableColumn<Offender, Integer> ageColumn;

    @FXML

    private TableColumn<Offender, String> genderColumn;

    @FXML

    private TableColumn<Offender, Integer> priorOffensesCountColumn;

    @FXML

    private TableColumn<Offender, String> riskLevelColumn;

    @FXML

    private TableColumn<Offender, String> lastArrestDateColumn;

    @FXML

    private TextField nameField;

    @FXML

    private TextField riskLevelField;

    private ObservableList<Offender> offenderList;

    // Initialize method to set up table columns and load data

    public void initialize() {

        offenderList = FXCollections.observableArrayList();

        offenderTable.setItems(offenderList);

        // Set up table columns

        offenderIdColumn.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getOffenderId()).asObject());

        firstNameColumn.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getFirstName()));

        lastNameColumn.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getLastName()));

        ageColumn.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getAge()).asObject());

        genderColumn.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getGender()));

        priorOffensesCountColumn.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getPriorOffensesCount()).asObject());

        riskLevelColumn.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getRiskLevel()));

        lastArrestDateColumn.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getLastArrestDate()));

    }

    // Action for the "Add Offender" button

    @FXML

    public void addOffender() {

        String firstName = nameField.getText();

        String riskLevel = riskLevelField.getText();

        if (firstName.isEmpty() || riskLevel.isEmpty()) {

            showErrorDialog("Form Error", "Please fill out all fields.");

            return;

        }

        // Sample data for the new offender

        int offenderId = offenderList.size() + 1; // Auto-increment offender ID

        String lastName = "Doe"; // Use default last name for simplicity

        int age = 30; // Placeholder age

        String gender = "M"; // Placeholder gender

        int priorOffensesCount = 0; // Placeholder for prior offenses

        String lastArrestDate = "N/A"; // Placeholder for last arrest date

        // Create a new offender object

        Offender newOffender = new Offender(offenderId, firstName, lastName, age, gender, priorOffensesCount, riskLevel, lastArrestDate);

        // Add the new offender to the list and refresh the table

        offenderList.add(newOffender);

        // Clear the input fields

        nameField.clear();

        riskLevelField.clear();

    }

    // Method to show error dialog

    private void showErrorDialog(String title, String message) {

        Alert alert = new Alert(AlertType.ERROR);

        alert.setTitle(title);

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();

    }

}

**Models :**

**3.11 Crime .java**

package com.crimedata.models;

import javafx.beans.property.IntegerProperty;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.StringProperty;

import javafx.beans.property.SimpleStringProperty;

public class Crime {

    private IntegerProperty id;

    private StringProperty type;

    private StringProperty location;

    private StringProperty date;

    public Crime(int id, String type, String location, String date) {

        this.id = new SimpleIntegerProperty(id);

        this.type = new SimpleStringProperty(type);

        this.location = new SimpleStringProperty(location);

        this.date = new SimpleStringProperty(date);

    }

    // Getters and Setters

    public int getId() {

        return id.get();

    }

    public void setId(int id) {

        this.id.set(id);

    }

    public String getType() {

        return type.get();

    }

    public void setType(String type) {

        this.type.set(type);

    }

    public String getLocation() {

        return location.get();

    }

    public void setLocation(String location) {

        this.location.set(location);

    }

    public String getDate() {

        return date.get();

    }

    public void setDate(String date) {

        this.date.set(date);

    }

    // Property methods for binding

    public IntegerProperty idProperty() {

        return id;

    }

    public StringProperty typeProperty() {

        return type;

    }

    public StringProperty locationProperty() {

        return location;

    }

    public StringProperty dateProperty() {

        return date;

    }

}

**3.12 Crime data.java:**

package com.crimedata.models;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.property.StringProperty;

public class CrimeData {

    private StringProperty crimeType;

    private StringProperty crimeLocation;

    private StringProperty crimeDate;

    private StringProperty offenderName;

    private StringProperty caseID;

    private StringProperty caseStatus;

    public CrimeData(String crimeType, String crimeLocation, String crimeDate,

                     String offenderName, String caseID, String caseStatus) {

        this.crimeType = new SimpleStringProperty(crimeType);

        this.crimeLocation = new SimpleStringProperty(crimeLocation);

        this.crimeDate = new SimpleStringProperty(crimeDate);

        this.offenderName = new SimpleStringProperty(offenderName);

        this.caseID = new SimpleStringProperty(caseID);

        this.caseStatus = new SimpleStringProperty(caseStatus);

    }

    public String getCrimeType() {

        return crimeType.get();

    }

    public String getCrimeLocation() {

        return crimeLocation.get();

    }

    public String getCrimeDate() {

        return crimeDate.get();

    }

    public String getOffenderName() {

        return offenderName.get();

    }

    public String getCaseID() {

        return caseID.get();

    }

    public String getCaseStatus() {

        return caseStatus.get();

    }

    // Getters for properties to bind with TableView

    public StringProperty crimeTypeProperty() {

        return crimeType;

    }

    public StringProperty crimeLocationProperty() {

        return crimeLocation;

    }

    public StringProperty crimeDateProperty() {

        return crimeDate;

    }

    public StringProperty offenderNameProperty() {

        return offenderName;

    }

    public StringProperty caseIDProperty() {

        return caseID;

    }

    public StringProperty caseStatusProperty() {

        return caseStatus;

    }

}

**3.13 Incidentreport .java :**

package com.crimedata.models;

public class IncidentReport {

    private int reportId;

    private int crimeId;

    private String description;

    private String date;

    // Constructor

    public IncidentReport(int reportId, int crimeId, String description, String date) {

        this.reportId = reportId;

        this.crimeId = crimeId;

        this.description = description;

        this.date = date;

    }

    // Getters and Setters

    public int getReportId() {

        return reportId;

    }

    public void setReportId(int reportId) {

        this.reportId = reportId;

    }

    public int getCrimeId() {

        return crimeId;

    }

    public void setCrimeId(int crimeId) {

        this.crimeId = crimeId;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getDate() {

        return date;

    }

    public void setDate(String date) {

        this.date = date;

    }

    // toString method (optional but helpful for debugging)

    @Override

    public String toString() {

        return "IncidentReport [reportId=" + reportId + ", crimeId=" + crimeId + ", description=" + description + ", date=" + date + "]";

    }

}

**3.14 Offender .java :**

package com.crimedata.models;

public class Offender {

    private int offenderId;

    private String firstName;

    private String lastName;

    private int age;

    private String gender;

    private int priorOffensesCount;

    private String riskLevel;

    private String lastArrestDate;

    // Constructors

    public Offender(int offenderId, String firstName, String lastName, int age, String gender, int priorOffensesCount, String riskLevel, String lastArrestDate) {

        this.offenderId = offenderId;

        this.firstName = firstName;

        this.lastName = lastName;

        this.age = age;

        this.gender = gender;

        this.priorOffensesCount = priorOffensesCount;

        this.riskLevel = riskLevel;

        this.lastArrestDate = lastArrestDate;

    }

    // Getters and Setters

    public int getOffenderId() {

        return offenderId;

    }

    public void setOffenderId(int offenderId) {

        this.offenderId = offenderId;

    }

    public String getFirstName() {

        return firstName;

    }

    public void setFirstName(String firstName) {

        this.firstName = firstName;

    }

    public String getLastName() {

        return lastName;

    }

    public void setLastName(String lastName) {

        this.lastName = lastName;

    }

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public String getGender() {

        return gender;

    }

    public void setGender(String gender) {

        this.gender = gender;

    }

    public int getPriorOffensesCount() {

        return priorOffensesCount;

    }

    public void setPriorOffensesCount(int priorOffensesCount) {

        this.priorOffensesCount = priorOffensesCount;

    }

    public String getRiskLevel() {

        return riskLevel;

    }

    public void setRiskLevel(String riskLevel) {

        this.riskLevel = riskLevel;

    }

    public String getLastArrestDate() {

        return lastArrestDate;

    }

    public void setLastArrestDate(String lastArrestDate) {

        this.lastArrestDate = lastArrestDate;

    }

}

**3.15 Database JDBC Connection:**

Databaseutil.java:

package com.crimedata.utils;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseUtil {

    private static final String DB\_URL = "jdbc:mysql://127.0.0.1:3306/prj";  // Database URL

    private static final String DB\_USER = "root";  // Database username

    private static final String DB\_PASSWORD = "Miruthula@24";  // Database password (change as needed)

    // Establish a database connection and return the connection object

    public static Connection getConnection() {

        try {

            // Load the MySQL JDBC driver

            Class.forName("com.mysql.cj.jdbc.Driver");

            // Attempt to connect to the database

            Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

            if (conn != null) {

                System.out.println("Database connected successfully!");

            } else {

                System.out.println("Failed to connect to the database.");

            }

            return conn;

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

            return null;

        }

    }

    // Close the connection (if needed in controllers)

    public static void closeConnection(Connection connection) {

        if (connection != null) {

            try {

                connection.close();

                System.out.println("Connection closed.");

            } catch (SQLException e) {

                e.printStackTrace();

            }

        }

    }

}

**Database Files:**

**3.16 Create\_tables.sql:**

CREATE DATABASE IF NOT EXISTS CrimeData;

USE CrimeData;

CREATE TABLE IF NOT EXISTS Crimes (

    crime\_id INT AUTO\_INCREMENT PRIMARY KEY,

    crime\_type VARCHAR(255),

    crime\_date DATE,

    crime\_time TIME,

    crime\_location VARCHAR(255),

    crime\_description TEXT

);

CREATE TABLE IF NOT EXISTS Offenders (

    offender\_id INT AUTO\_INCREMENT PRIMARY KEY,

    first\_name VARCHAR(100),

    last\_name VARCHAR(100),

    age INT,

    gender VARCHAR(10),

    prior\_offenses\_count INT,

    risk\_level VARCHAR(50),

    last\_arrest\_date DATE

);

CREATE TABLE IF NOT EXISTS Locations (

    location\_id INT AUTO\_INCREMENT PRIMARY KEY,

    crime\_id INT,

    address VARCHAR(255),

    latitude DOUBLE,

    longitude DOUBLE,

    FOREIGN KEY (crime\_id) REFERENCES Crimes(crime\_id)

);

CREATE TABLE IF NOT EXISTS IncidentReports (

    report\_id INT AUTO\_INCREMENT PRIMARY KEY,

    crime\_id INT,

    officer\_name VARCHAR(255),

    case\_status ENUM('Open', 'Closed', 'Under Investigation'),

    report\_notes TEXT,

    report\_date DATE,

    FOREIGN KEY (crime\_id) REFERENCES Crimes(crime\_id)

);

CREATE TABLE IF NOT EXISTS Witnesses (

    witness\_id INT AUTO\_INCREMENT PRIMARY KEY,

    crime\_id INT,

    first\_name VARCHAR(100),

    last\_name VARCHAR(100),

    statement TEXT,

    contact\_info VARCHAR(255),

    FOREIGN KEY (crime\_id) REFERENCES Crimes(crime\_id)

);

CREATE TABLE IF NOT EXISTS Crime\_Offender (

    crime\_id INT,

    offender\_id INT,

    PRIMARY KEY (crime\_id, offender\_id),

    FOREIGN KEY (crime\_id) REFERENCES Crimes(crime\_id),

    FOREIGN KEY (offender\_id) REFERENCES Offenders(offender\_id)

);

CREATE TABLE IF NOT EXISTS OfficerActivity (

    activity\_id INT AUTO\_INCREMENT PRIMARY KEY,

    officer\_name VARCHAR(255),

    report\_id INT,

    activity\_date DATE,

    activity\_description TEXT,

    FOREIGN KEY (report\_id) REFERENCES IncidentReports(report\_id)

);

**3.17 Main.java :**

package com.crimedata;

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.fxml.FXMLLoader;

import javafx.stage.Stage;

public class Main extends Application {

    @Override

    public void start(Stage primaryStage) throws Exception {

        // Load the Dashboard FXML as the main entry page

        FXMLLoader loader = new FXMLLoader(getClass().getResource("/com/crimedata/views/Main.fxml"));

        Scene scene = new Scene(loader.load());

        primaryStage.setScene(scene);

        primaryStage.setTitle("Crime Data Analysis System");

        primaryStage.show();

    }

    public static void main(String[] args) {

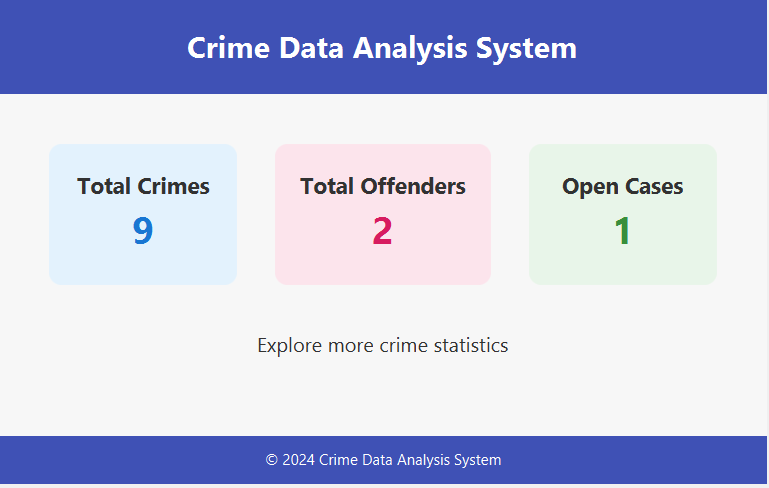
        Application.launch(args);

    }

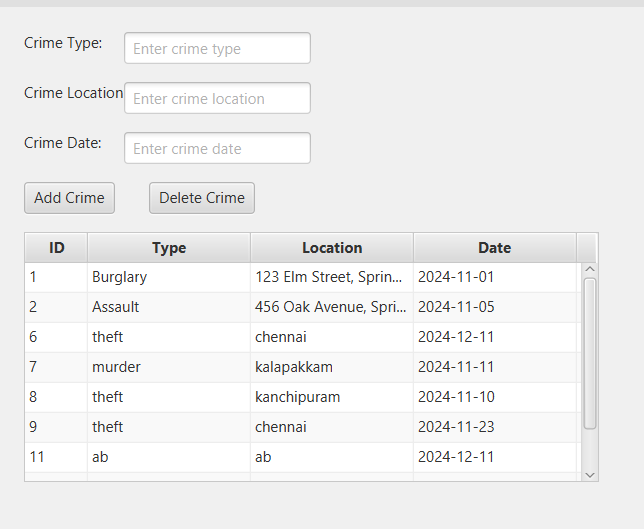
}

**SNAPSHOTS**

## 4.1 Dasboard:



## 4.2 Crime records page



## 4.3 Incident Reports Page



# CONCLUSION

With the help of our project, law enforcement agencies, policymakers, and researchers will be able to effortlessly manage and analyze crime-related data. The **Crime Data Analysis System** provides a centralized platform to record crimes, maintain offender profiles, track incident reports, and monitor officer activities. By organizing all critical data and making it easily accessible through an interactive dashboard, the system saves significant time and resources. It also enhances decision-making capabilities by providing advanced analytics, including crime trends, geographic hotspots, and offender patterns.

**REFERENCES**

1. [**https://www.javatpoint.com/java-tutorial**](https://www.javatpoint.com/java-tutorial)

1. <https://www.javatpoint.com/javafx-tutorial>

1. [SQL | Codecademy](https://www.codecademy.com/resources/docs/sql)
2. <https://www.w3schools.com/MySQL/default.asp>