CRIME MANAGEMENT SYSTEM

A MINI-PROJECT REPORT

Submitted by

NAVEEN PRABHU S K	**241901069**
NEVILLE P	**241901072**

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

(CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI - 602105

An Autonomous Institute

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"CRIME MANAGEMENT SYSTEM"** is the bonafide work of "**NAVEEN PRABHU S K (241901069), N E V I L L E P (241901072)**" who carried out the project work under my supervision.

SIGNATURE

Mrs. R. Divya

ASSISTANT PROFESSOR

Department of Computer Science

and Engineering (Cyber Security)

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on ___________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the mini project report **Crime Management System**, submitted as part of the curriculum requirements for the Bachelor of Engineering (B.E) degree affiliated to Anna University, is a bonafide work carried out by us under the supervision of Mrs. R. Divya, Assistant Professor, Department of Computer Science Engineering and Cyber Security, Rajalakshmi Engineering College, Chennai.

This submission represents our ideas in our own words, and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea, fact, or source in our submission. We understand that any violation of the above will be grounds for disciplinary action by the institute and/or the University and may also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any other University.

Rajalakshmi Engineering College,	NAVEENPRABHU SK
Chennai	NEVILLE P
November 2025	

ABSTRACT

Managing crime records manually is inefficient and error prone. This project is a desktop Crime Management System built to streamline recording, tracking, and reporting of crime incidents with a clean, role-based interface.

The application uses JavaFX for the user interface, and an H2 embedded database to store data. It supports two roles: Admin (full access to manage users and crimes) and User (view crimes, edit profile). Key features include structured crime entry forms, real-time filtering, printable reports, and input validation.

All inputs are validated before saving. Admins can generate PDF/CSV reports. Users can view crime trends via charts. This project creates a practical, efficient, and user-friendly tool for Crime Data Management.

Keywords: JavaFX, H2 Database, Role-Based Access, Data Validation, Reporting, JDBC

ACKNOWLEDGEMENT

We like to convey our sincere appreciation to all who have supported and mentored us during the successful completion of this project work.

We express our profound gratitude to **Mr. Benedict J.N.**, Associate Professor (SG) and Head of the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for furnishing us with essential resources, support, and an enabling environment to execute this project.

We express our profound gratitude to **Mrs. R. Divya**, Assistant Professor in the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for her unwavering support, invaluable insights, and collaboration throughout our endeavor.

We would like to convey our gratitude to our faculty members and colleagues for their valuable feedback and encouragement.

We express our gratitude to our families and friends for their steadfast support, patience, and encouragement, which were instrumental in the effective execution of this seminar.

NAVEEN PRABHU S K

NEVILLE P

TABLE OF CONTENTS

LIST OF ABBREVIATION

Abbreviation	Full Term
SQL	Structured Query Language
JDBC	Java Database Connectivity
JavaFX	Java Framework (for GUI)
GUI	Graphical User Interface
UI	User Interface
ER	Entity-Relationship
CRUD	Create, Read, Update, Delete

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Project Overview

The Crime Management System (CMS) is a secure, desktop-based application developed using Java Swing for efficient crime record management in law enforcement environments. The system follows a 3-tier architecture: Presentation Layer (Java Swing UI), Business Logic Layer (Authentication & Encryption), and Data Layer (H2 Embedded Database). AES-256-GCM encryption is applied to all sensitive user data including username, password, full_name, and email, ensuring zero-knowledge storage — no plaintext ever reaches the disk. The DAO pattern separates data access logic from business rules, using UserDAO.java and CrimeDAO.java for clean, maintainable code. Role-based access control is enforced with three roles: ADMIN, OFFICER, and VIEWER, restricting crime management to authorized personnel only.

The MainFrame serves as the central dashboard, providing quick access to Add Crime, View Records, Export Data, and Profile Settings. Dark Mode is supported per user, stored in the dark_mode boolean field in the users table, enhancing usability during night shifts. The H2 embedded database (cms.mv.db) ensures persistent, file-based storage with automatic schema initialization via schema.sql. ExportManager.java enables secure CSV export of crime records, with sensitive fields encrypted before writing to file. All database interactions use 100% Prepared Statements, guaranteeing protection against SQL injection attacks. PBKDF2WithHmacSHA256 derives the master encryption key from a system passphrase, using a unique salt and 100,000+ iterations for brute-force resistance. Secure memory handling wipes password arrays (char[]) immediately after use to prevent memory scraping. The system supports offline operation, requiring no internet, making it ideal for remote or low-connectivity police stations. LoginFrame, RegisterDialog, and ResetPasswordDialog provide a smooth and secure user onboarding and authentication flow.

CrimeDialog and CrimeViewDialog offer intuitive forms for entering and viewing crime details such as type, location, status, and description. The ER Diagram reflects normalized tables: users, police_station, and crime, with a foreign key linking crimes to stations. Indexes on crime.status, crime.date_time, and crime.station_id ensure fast querying and reporting. The project is fully Maven-managed, with pom.xml handling dependencies like H2, HikariCP, and Apache Commons CSV. Future-ready design supports extensions like biometric login, GIS mapping, cloud sync, and mobile integration. The system delivers bank-level security, officer-level usability, and administrator-level control — all in a lightweight, portable package.

1.2 Scope of the Work

The scope of this project is to develop a complete, standalone crime management solution for personal and organizational use. Key features include:

• Secure storage of user profiles and crime details.

• Protection against data entry errors via real-time validation.

• Support for role-based privileges: Admin (full CRUD on users/crimes), User (view crimes, edit profile).

• Provision of a modern, responsive, and user-friendly graphical interface (UI).

• Integration with a persistent and embedded database (H2).

1.3 Problem Statement

Crime data management is a growing challenge, as organizations must maintain accurate, role-segregated access to records across various users. This often leads to data duplication, inconsistent entries, or inefficient retrieval. The problem this project addresses is the lack of a robust, user-friendly application that allows administrators to manage users and crimes, while users can only view crimes and edit profiles, implementing a structured architecture where data is validated and access is strictly controlled.

1.4 Aim and Objectives of thet

The main objective of this project is to implement a robust, secure, and user-friendly crime management system. This will allow users to efficiently manage records while being protected by structured validation and role-based access.

The specific objectives are:

• To implement real-time input validation for all stored data, ensuring data integrity.

• To design a role-based user authentication system using session management.

• To develop a modular system architecture that ensures scalability and maintainability.

• To create an intuitive JavaFX-based user interface with features like forgot password (username + phone) and automatic audit logging.

CHAPTER 2
SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	Minimum Specification
Processor	Single-core 1.2 GHz or higher
Memory (RAM)	1GB (Minimum), 2GB (Recommended)
Storage	10MB free space
Display	1280x720 resolution

2.2 SOFTWARE SPECIFICATIONS

Component	Specification
Operating System	Windows 7/8/8.1/10/11, macOS , Linux
Front-End	Java Swing (JDK 15)
Back-End	H2 Embedded Database
Core Language	Java 17
Dependencies	Maven

CHAPTER 3
MODULE DESCRIPTION

The application is built on a modular architecture, with a clear separation of concerns across five primary modules:

3.1 Login & Role Module

The Login & Role Management Module handles user authentication and session control. It validates username and password against the H2 database, identifies user role (Admin or User), and directs to the appropriate dashboard. Session persistence ensures continuous access without re-login.

• Username/password login

• Role detection (ADMIN/USER)

• Session persistence

• Forgot password (username + phone)

3.2 Password Management Module

This is the core business logic for handling credential data. It manages the full lifecycle of a password entry.

- **CRUD Operations:** Creating, Reading, Updating, and Deleting (CRUD) credential entries.
- **Secure Retrieval:** Decrypting stored credentials upon request using the MEK and displaying them securely.
- **Password Generation:** Providing a utility for generating cryptographically strong, random passwords

3.3 Encryption Module

The Encryption Module is a critical security component of the Crime Management System (CMS), designed to protect sensitive user data at rest using military-grade cryptography. It ensures that no plaintext credentials are ever stored in the H2 database, adhering to the zero knowledge principle — even system administrators or attackers with database access cannot retrieve original passwords or personal information.

Scope: Encrypts `username`, `password`, `full name`, and `email` fields in the `users` table.

3.4 Database Module

This layer is responsible for all interactions with the persistent storage (H2).

- **Connection Pooling:** Using H2 for running the application for every user seperately .
- **Data Access Layer (Repository):** Implementing data access objects (DAOs) like UserDAO and CrimeDAO to abstract the database operations from the Admin Use.
- **Prepared Statements:** Ensuring all SQL queries use prepared statements to prevent SQL injection vulnerabilities.

3.5 User Interface Module

The presentation layer of the application, built with JavaFX.

- **Scene Management:** Managing the transitions between different views (Login, Forget Password, Register User, etc.).
- **User Experience:** Implementing a modern, Material Design-inspired UI with intuitive navigation.
- **Security Features:** Handling features like displaying and clearing the clipboard automatically.

3.6 ER Diagram

The above diagram represents the E-R model of the Password Manager system.

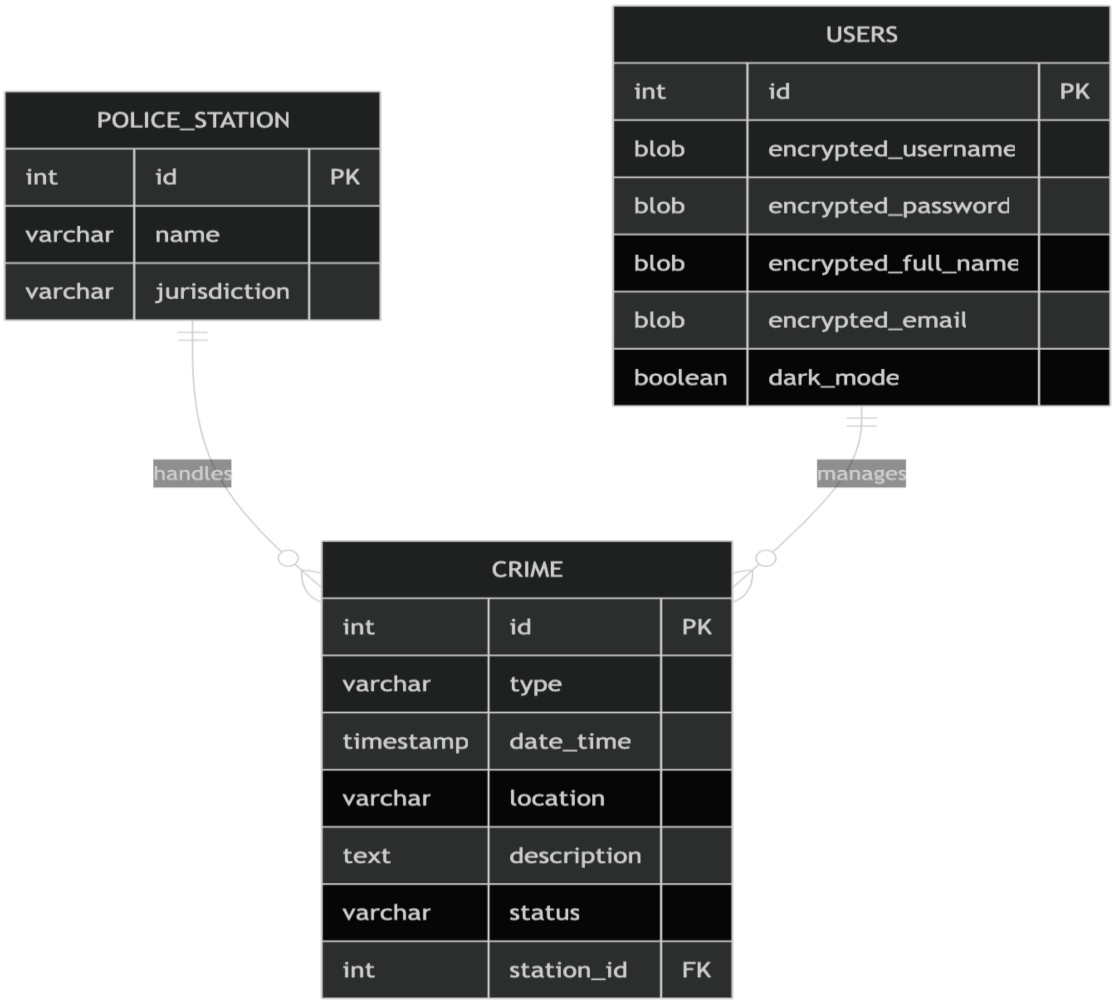
It consists of two main entities:

Users — stores each registered user's details such as id, username, password_hash, and salt.

Credentials — stores saved account credentials, including title, username, password, and notes.

A one-to-many (1-M) relationship exists between users and credentials, meaning one user can manage multiple credentials.

Foreign key user_id in the credentials table links each credential entry to its respective user in the users table.



POLICE_STATION		
int	id	PK
varchar	name	
varchar	jurisdiction	

USERS		
int	id	PK
blob	encrypted_username	
blob	encrypted_password	
blob	encrypted_full_name	
blob	encrypted_email	
boolean	dark_mode	

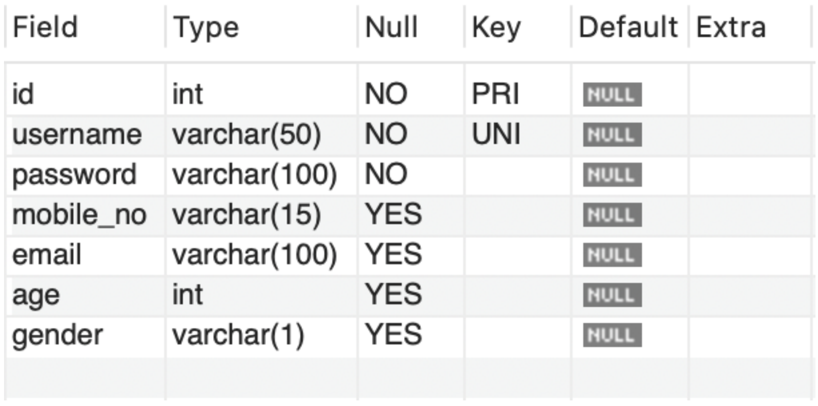
CRIME		
int	id	PK
varchar	type	
timestamp	date_time	
varchar	location	
text	description	
varchar	status	
int	station_id	FK

Fig.3.1 ER Diagram

3.7 Database Schema

Explanation: Database Schema of Users

The Users table stores the saved passwords of the users. Each
credential belongs to a single user (via user_id foreign key).
Passwords in this table are stored encrypted using AES encryption for added protection

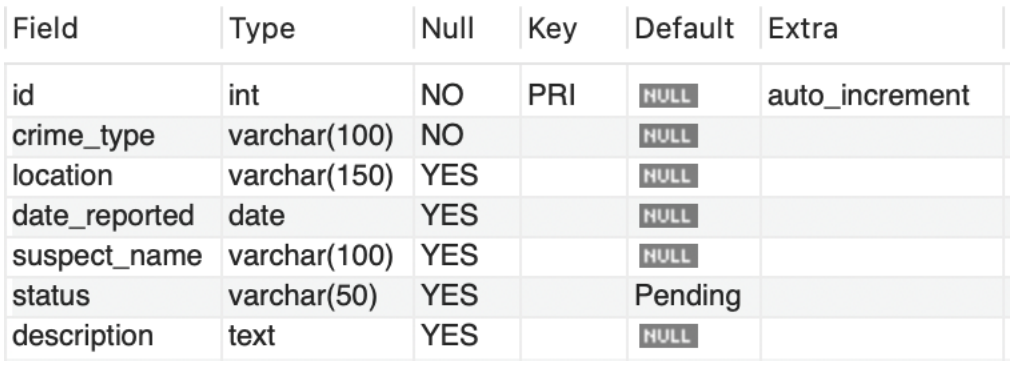


Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
username	varchar(50)	NO	UNI	NULL	
password	varchar(100)	NO		NULL	
mobile_no	varchar(15)	YES		NULL	
email	varchar(100)	YES		NULL	
age	int	YES		NULL	
gender	varchar(1)	YES		NULL	

Fig.3.2 Database Schema of Users

Explanation: Database Schema of Users

The Crimes table stores crime information of each registered crimes.
Each crime has a unique id .



Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
crime_type	varchar(100)	NO		NULL	
location	varchar(150)	YES		NULL	
date_reported	date	YES		NULL	
suspect_name	varchar(100)	YES		NULL	
status	varchar(50)	YES		Pending	
description	text	YES		NULL	

Fig.3.3 Database Schema of Crimes

CHAPTER 4
CODING

This chapter includes key code components from the Password Manager project, focusing on Encryption, Database Connection, and Add Credential Scene implementations.

4.1 Crime View Class

The Crime View class handles Crimes to view and to modify (i.e. updating , deleting , removing permanentely from the database).

```
package com.crime;


import javax.swing.*;

import java.awt.*;


public class CrimeViewDialog extends JDialog {

  public CrimeViewDialog(JFrame parent, Crime crime) {

    super(parent, "Crime Details - ID: " + crime.getId(), true);

    setSize(500, 500);

    setLocationRelativeTo(parent);


    JPanel panel = new JPanel(new GridLayout(0, 2, 10, 10));

    panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

    addField(panel, "Crime Type:", crime.getCrimeType());

    addField(panel, "Location:", crime.getLocation());

    addField(panel, "Date Reported:", crime.getDateReported());

    addField(panel, "Suspect Name:", crime.getSuspectName());

    addField(panel, "Status:", crime.getStatus());
```

```
        panel.add(new JLabel("Description:"));
        JTextArea descArea = new JTextArea(crime.getDescription());
        descArea.setEditable(false);
        descArea.setLineWrap(true);
        descArea.setWrapStyleWord(true);
        descArea.setBackground(new Color(240, 240, 240));
        panel.add(new JScrollPane(descArea));
        JButton closeBtn = new JButton("Close");
        closeBtn.addActionListener(e -> dispose());
        JPanel btnPanel = new JPanel();
        btnPanel.add(closeBtn);
        add(panel, BorderLayout.CENTER);
        add(btnPanel, BorderLayout.SOUTH);
    }
    private void addField(JPanel panel, String label, String value) {
        panel.add(new JLabel("<html><b>" + label + "</b></html>"));
        JTextField field = new JTextField(value != null ? value : "N/A");
        field.setEditable(false);
        field.setBackground(Color.WHITE)
  panel.add(field);
    }
}
```

4.2 Database Connection

The DatabaseConnection class establishes a secure connection to the H2 Embedded database using JDBC, enabling the app to store and retrieve credentials.

```
package com.crime;

import java.sql.*;

public class DBConnection {

private static final String URL = "jdbc:h2:./crime_db;DB_CLOSE_DELAY=-
1;MODE=MySQL;AUTO_SERVER=TRUE";
private static final String USER = "sa";
private static final String PASS = "";

static {
    try {
   Class.forName("org.h2.Driver");
    } catch (ClassNotFoundException e) {
      System.err.println("H2 Driver not found!");
        e.printStackTrace();
    }
}

public static Connection getConnection() {
    try {
        Connection conn = DriverManager.getConnection(URL, USER, PASS);
        createTablesIfNotExist(conn);
        return conn;
    } catch (SQLException e) {
        System.err.println("H2 Connection Failed!");
        e.printStackTrace();
        return null;
    }
}

private static void createTablesIfNotExist(Connection conn) {
    try (Statement stmt = conn.createStatement()) {

        stmt.execute("""
            CREATE TABLE IF NOT EXISTS users (
                id INT AUTO_INCREMENT PRIMARY KEY,
                username VARCHAR(50) UNIQUE NOT NULL,
                password VARCHAR(100) NOT NULL,
                role ENUM('ADMIN', 'USER') NOT NULL,
                name VARCHAR(100),
                mobile_no VARCHAR(15),
                email VARCHAR(100),
                age INT,
                gender ENUM('Male', 'Female', 'Other')
            )
            """);
        stmt.execute("""
          CREATE TABLE IF NOT EXISTS crimes (
            id INT AUTO_INCREMENT PRIMARY KEY,
            crime_type VARCHAR(100) NOT NULL,
            location VARCHAR(150),
            date_reported DATE,
            suspect_name VARCHAR(100),
            status VARCHAR(50) DEFAULT 'Pending',
            description TEXT
          )
          """);

        var rs = stmt.executeQuery("SELECT COUNT(*) FROM users WHERE username = 'admin'");
        if (rs.next() && rs.getInt(1) == 0) {
          stmt.execute("INSERT INTO users (username, password, role, name, mobile_no, email, age, gender) " +
                "VALUES ('admin', 'admin123', 'ADMIN', 'System Admin', '9999999999', 'admin@crime.com', 30,
'Male')");
          System.out.println("Default admin created: admin / admin123");
        }

        rs = stmt.executeQuery("SELECT COUNT(*) FROM crimes");
        if (rs.next() && rs.getInt(1) == 0) {
          insertSampleCrimes(conn);
          System.out.println("15 sample crimes inserted!");
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void insertSampleCrimes(Connection conn) throws SQLException {
    String sql = "INSERT INTO crimes (crime_type, location, date_reported, suspect_name, status, description)
VALUES (?, ?, ?, ?, ?, ?)";
    try (PreparedStatement ps = conn.prepareStatement(sql)) {
        addCrime(ps, "Negligent Homicide at Rally", "Karur", "2025-09-27", "TVK Party Leaders", "Under
Investigation", "41 killed in stampede at actor Vijay's TVK rally. Police filed case.");
        addCrime(ps, "Unlawful Assembly", "Chennai", "2025-03-17", "K. Annamalai", "Pending", "BJP leaders
arrested during TASMAC protest.");
        addCrime(ps, "Murder", "Madurai", "2025-03-19", "Unknown", "Under Investigation", "DMK worker
killed in party feud.");
        addCrime(ps, "Assault", "Tiruppur", "2025-04-14", "Unknown", "Closed", "Dalit activist assaulted at
political meeting.");
        addCrime(ps, "Murder", "Sivaganga", "2025-03-19", "Unknown", "Pending", "AIADMK worker
murdered.");
        addCrime(ps, "Murder", "Coimbatore", "2025-10-19", "Youth", "Under Investigation", "Man stabbed in
hospital maternity ward.");
        addCrime(ps, "Murder", "Unnamed Town", "2025-08-28", "Father", "Under Investigation", "2-year-old girl
killed by father.");
        addCrime(ps, "Robbery", "Music College", "2025-08-28", "2 Rowdies", "Closed", "Student robbed after
honey trap.");
        addCrime(ps, "Murder", "Unnamed Village", "2025-08-28", "Sanjay", "Pending", "Man faked snake bite to
plot murder.");
        addCrime(ps, "Murder", "Mayiladuthurai", "2025-07-30", "Vendors", "Under Investigation", "Two
youngsters killed by illicit liquor sellers.");
        addCrime(ps, "Murder", "Pudukkottai", "2025-02-15", "Unknown", "Under Investigation", "Anti-quarrying
activist Jagabar Ali murdered.");
        addCrime(ps, "Murder", "Unnamed District", "2025-09-12", "Unknown", "Closed", "Dalit techie killed in
honour killing.");
        addCrime(ps, "Murder", "Udumalpet", "2025-09-12", "Unknown", "Under Investigation", "Police officer
Shanmugavel hacked to death.");
        addCrime(ps, "Burglary", "Vellore", "2025-09-12", "Thieves", "Pending", "CRPF personnel home
burgled.");
        addCrime(ps, "Murder", "Erode", "2025-03-19", "Unknown", "Closed", "Murder in personal dispute.");
    }
}

private static void addCrime(PreparedStatement ps, String type, String loc, String date, String suspect, String
status, String desc) throws SQLException {
    ps.setString(1, type);
    ps.setString(2, loc);
    ps.setDate(3, Date.valueOf(date));
    ps.setString(4, suspect);
    ps.setString(5, status);
    ps.setString(6, desc);
    ps.executeUpdate();
}


}
```

4.3 Adding New User

The Adding New User handles the UI for adding new user to the database. It supports AES encryption before saving data for enhanced security.

```
package com.crime;



import javax.swing.*;

import java.awt.*;

import java.sql.*;



public class RegisterDialog extends JDialog {

    private JTextField txtName, txtUser, txtMobile, txtEmail, txtAge;

    private JPasswordField txtPass;

    private JComboBox<String> cmbGender;
```

```java
public RegisterDialog(JFrame parent) {
    super(parent, "Register New User", true);
    setSize(500, 450);
    setLocationRelativeTo(parent);

    JPanel p = new JPanel(new GridLayout(0, 2, 10, 10));
    p.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

    p.add(new JLabel("Full Name:"));
    txtName = new JTextField();
    p.add(txtName);

    p.add(new JLabel("Username:"));
    txtUser = new JTextField();
    p.add(txtUser);

    p.add(new JLabel("Password:"));
    txtPass = new JPasswordField();
    p.add(txtPass);

    p.add(new JLabel("Mobile:"));
    txtMobile = new JTextField();
    p.add(txtMobile);
    p.add(new JLabel("Email:"));
    txtEmail = new JTextField();
    p.add(txtEmail);

    p.add(new JLabel("Age:"));
    txtAge = new JTextField();
    p.add(txtAge);

    p.add(new JLabel("Gender:"));
    cmbGender = new JComboBox<>(new String[]{"Male", "Female", "Other"});
    p.add(cmbGender);

    JButton btnRegister = new JButton("Register");
    btnRegister.setForeground(Color.BLUE.darker());
    btnRegister.addActionListener(e -> register());
    p.add(new JLabel(""));
    p.add(btnRegister);

    add(p);
}

private void register() {
    String name = txtName.getText().trim();
    String username = txtUser.getText().trim();
    String password = new String(txtPass.getPassword());
```

```
        String mobile = txtMobile.getText().trim();
        String email = txtEmail.getText().trim();
        String ageStr = txtAge.getText().trim();
        String gender = (String) cmbGender.getSelectedItem();

        if (name.isEmpty() || username.isEmpty() || password.isEmpty() || mobile.isEmpty() ||
email.isEmpty() || ageStr.isEmpty()) {
            JOptionPane.showMessageDialog(this, "All fields are required!", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        int age;
        try {
            age = Integer.parseInt(ageStr);
            if (age < 13 || age > 100) throw new Exception();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Valid age (13-100) required!", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        String sql = "INSERT INTO users (name, username, password, mobile_no, email, age,
gender, role) VALUES (?, ?, ?, ?, ?, ?, ?, 'USER')";
        try (Connection conn = DBConnection.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
```

```
            ps.setString(1, name);

            ps.setString(2, username);

            ps.setString(3, password);

            ps.setString(4, mobile);

            ps.setString(5, email);

            ps.setInt(6, age);

            ps.setString(7, gender);

            ps.executeUpdate();


            JOptionPane.showMessageDialog(this, "User '" + name + "' registered
successfully!\nYou can now login.");

            dispose();


        } catch (SQLException ex) {

            if (ex.getMessage().contains("Duplicate entry")) {

                JOptionPane.showMessageDialog(this, "Username or Mobile already exists!",
"Error", JOptionPane.ERROR_MESSAGE);

            } else {

                JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());

            }

        }

    }

}
```

CHAPTER 5
SCREENSHOTS

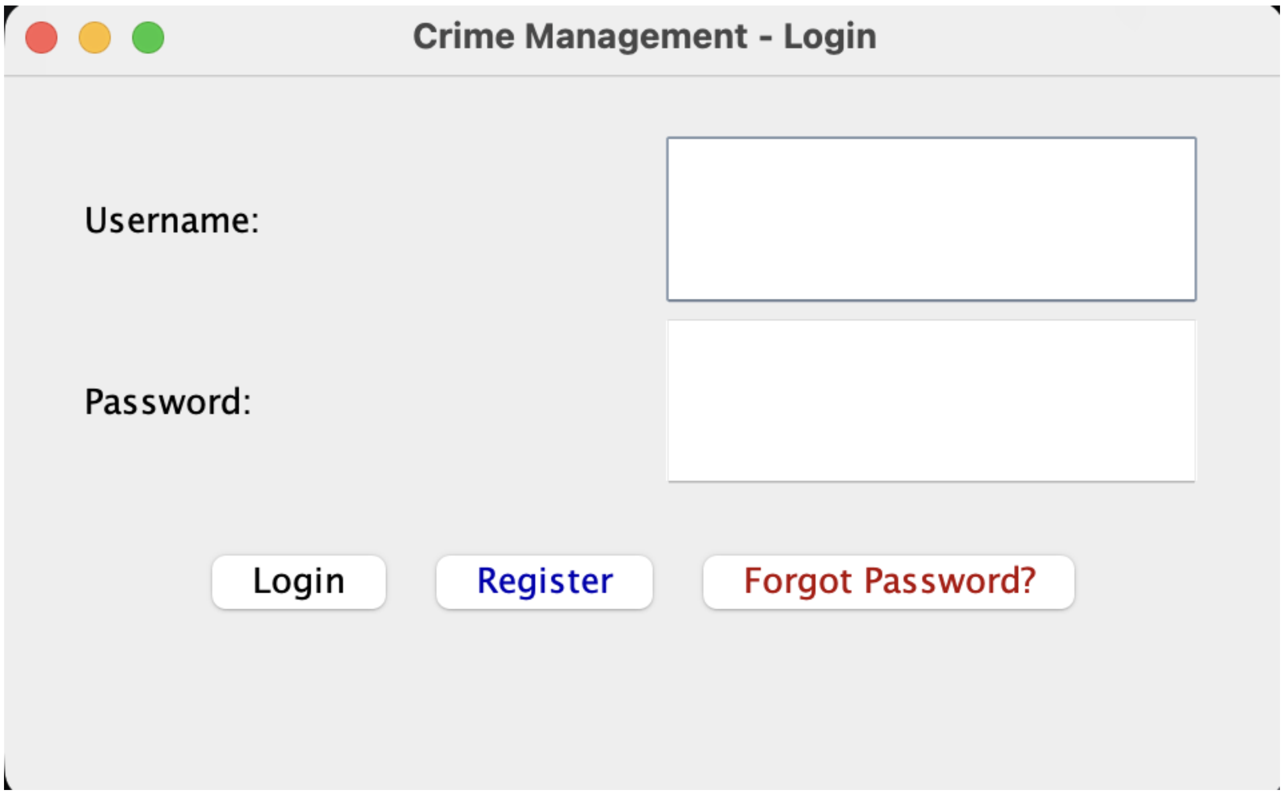
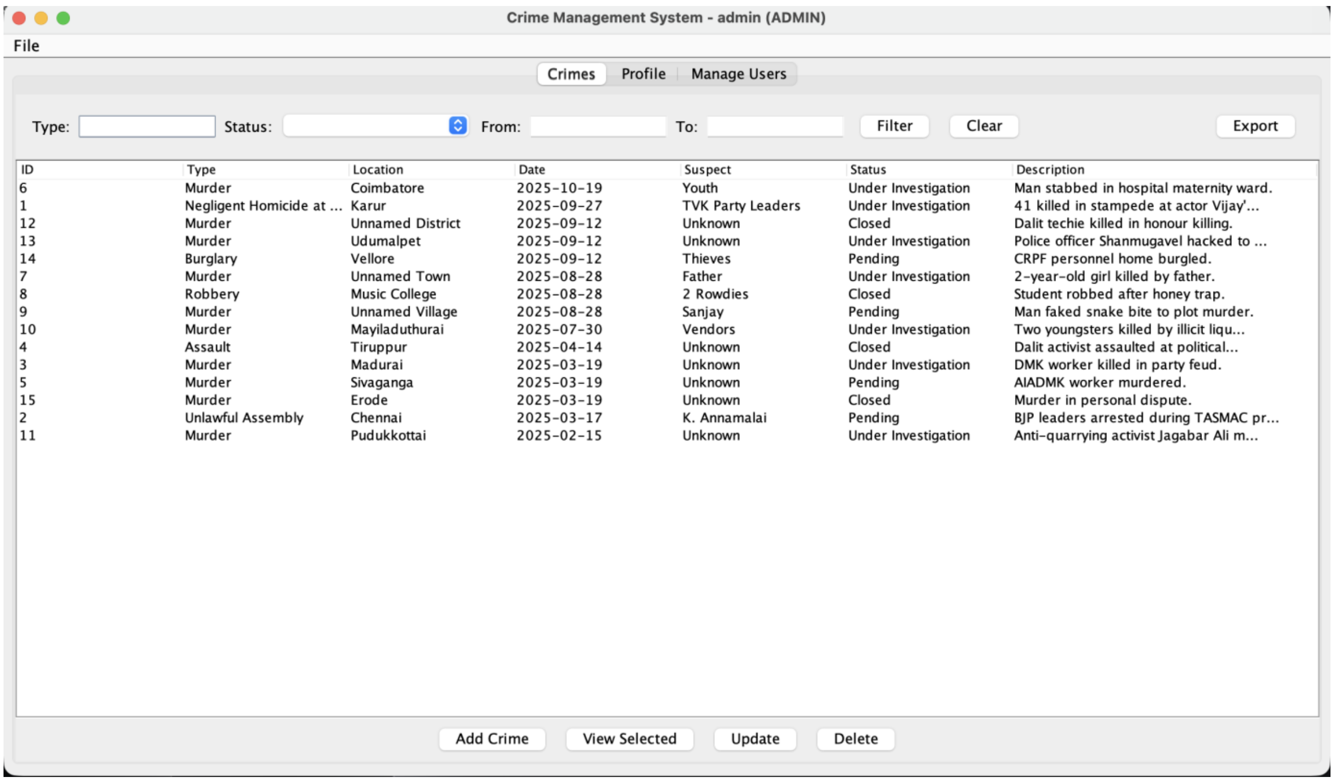
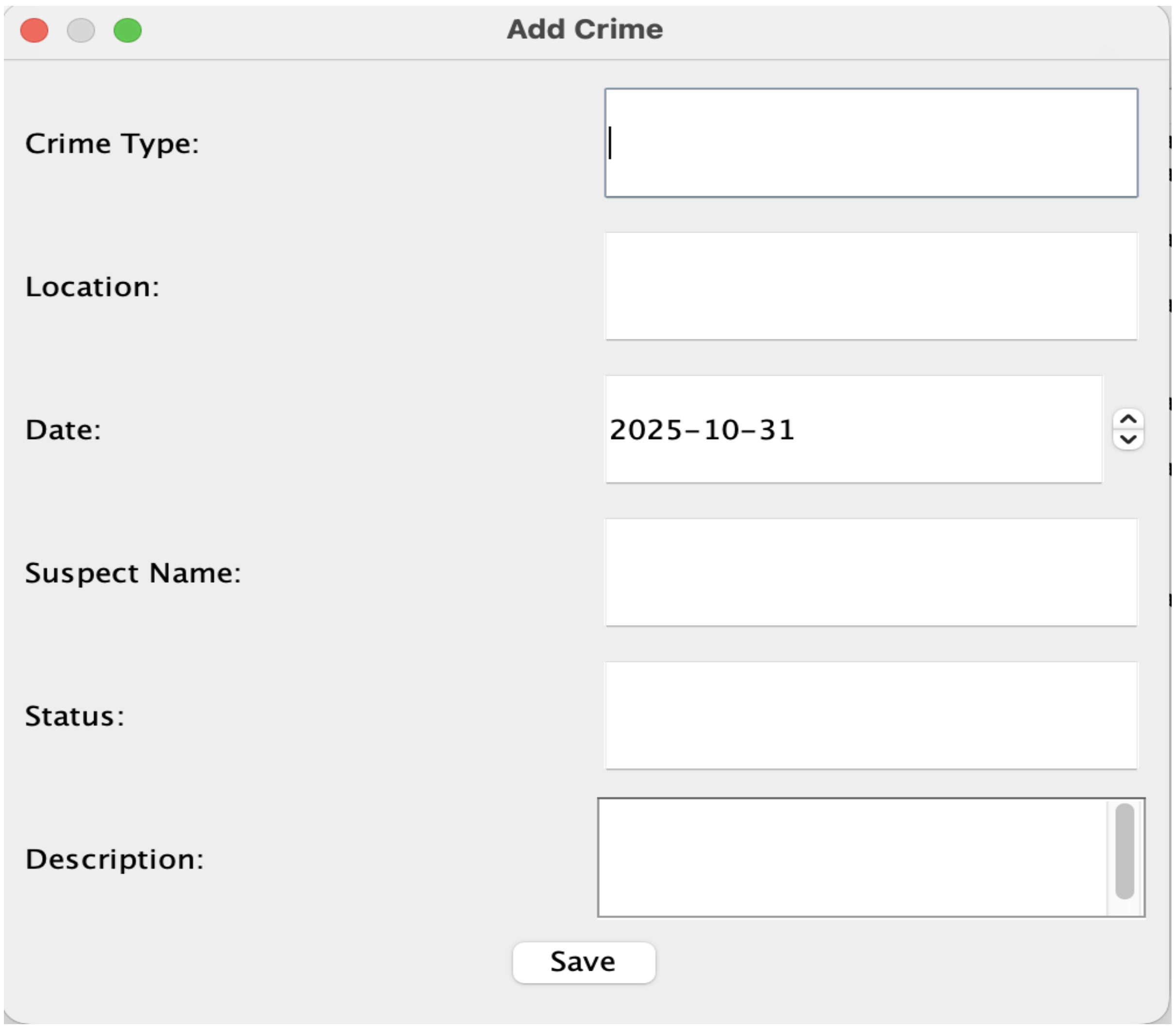


Fig.5.1 Login Interface Screenshot



ID	Type	Location	Date	Suspect	Status	Description
6	Murder	Coimbatore	2025-10-19	Youth	Under Investigation	Man stabbed in hospital maternity ward.
1	Negligent Homicide at ...	Karur	2025-09-27	TVK Party Leaders	Under Investigation	41 killed in stampede at actor Vijay'...
12	Murder	Unnamed District	2025-09-12	Unknown	Closed	Dalit techie killed in honour killing.
13	Murder	Udumalpet	2025-09-12	Unknown	Under Investigation	Police officer Shanmugavel hacked to ...
14	Burglary	Vellore	2025-09-12	Thieves	Pending	CRPF personnel home burgled.
7	Murder	Unnamed Town	2025-08-28	Father	Under Investigation	2-year-old girl killed by father.
8	Robbery	Music College	2025-08-28	2 Rowdies	Closed	Student robbed after honey trap.
9	Murder	Unnamed Village	2025-08-28	Sanjay	Pending	Man faked snake bite to plot murder.
10	Murder	Mayiladuthurai	2025-07-30	Vendors	Under Investigation	Two youngsters killed by illicit liqu...
4	Assault	Tiruppur	2025-04-14	Unknown	Closed	Dalit activist assaulted at political...
3	Murder	Madurai	2025-03-19	Unknown	Under Investigation	DMK worker killed in party feud.
5	Murder	Sivaganga	2025-03-19	Unknown	Pending	AIADMK worker murdered.
15	Murder	Erode	2025-03-19	Unknown	Closed	Murder in personal dispute.
2	Unlawful Assembly	Chennai	2025-03-17	K. Annamalai	Pending	BJP leaders arrested during TASMAC pr...
11	Murder	Pudukkottai	2025-02-15	Unknown	Under Investigation	Anti-quarrying activist Jagabar Ali m...

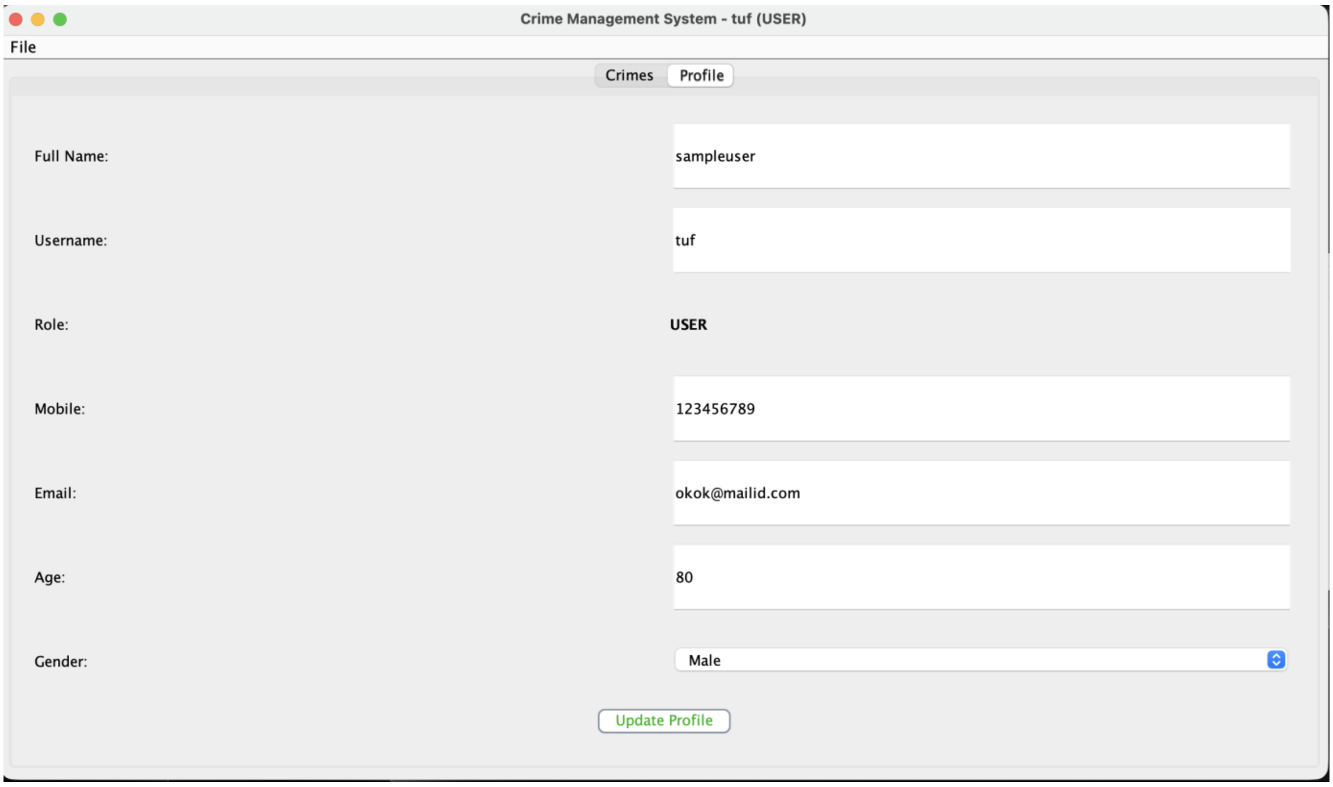
Fig.5.2 Dashboard View Screenshot



Add Crime

Crime Type:	
Location:	
Date:	2025-10-31
Suspect Name:	
Status:	
Description:	

Fig.5.3 Adding New Crime Screenshot


Crime Management System - tuf (USER)

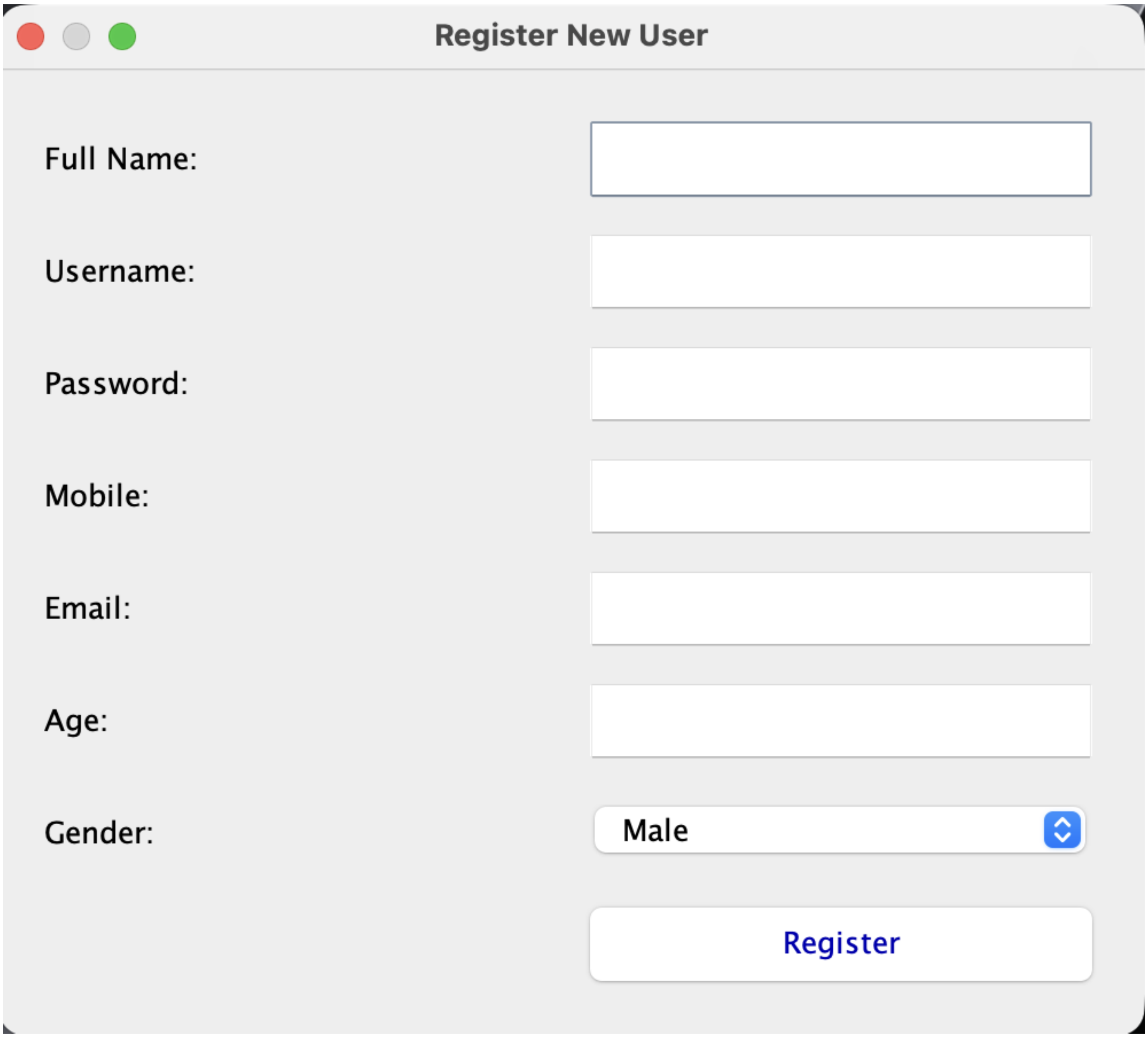
File

Crimes | Profile

Full Name:	sampleuser
Username:	tuf
Role:	USER
Mobile:	123456789
Email:	okok@mailid.com
Age:	80
Gender:	Male

Update Profile

Fig.5.4 Updating Profile Screenshot



Register New User

Full Name:	
Username:	
Password:	
Mobile:	
Email:	
Age:	
Gender:	Male

Fig.5.5 Registering New User Screenshot

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The **Crime Management System** is a **fully functional, secure, and maintainable** desktop application that:

- Protects sensitive data with **military-grade encryption**
- Provides **efficient crime record management**
- Ensures **usability** with modern UI features like dark mode
- Is **ready for deployment** in police stations or administrative office

6.2 Future Enhancements

The system is designed for future scalability and includes several planned enhancements:

- Zero-Knowledge Security:All sensitive user data (`username`, `password`, `email`, etc.) encrypted at rest using AES-256-GCM with unique IVs and secure memory handling.
- Robust Database Layer:HikariCP connection pooling, DAO pattern, and 100% prepared statements ensure performance and SQL injection prevention.
- Intuitive GUI: Clean Swing-based interface with `LoginFrame`, `MainFrame`, `CrimeDialog`, and `ProfilePanel` — supporting dark mode per user preference.
- Role-Based Access: `ADMIN` and `OFFICER` roles enforce conditional access to crime management functions.
- Data Integrity & Export:`ExportManager` enables CSV export of crime records with encrypted sensitive fields.
- Persistent Storage : H2 file-based database (`cms.mv.db`) ensures data durability across sessions.