

**Lahore Campus**

**OBJECT ORIENTED PROGRAMMING**

**Crime Reporting System Project**

**Submitted to:** Sir Shahid Bhatti

**Submitted by:** Laraib Qadeer, Namirah Manzoor

**Registration no.:** FA24-BCS-047, FA24-BCS-089.

**Section:** B

**Date:** 15/05/2025

**ABSTRACT**

The Crime Reporting System acts as an innovative platform that advances the practices of crime incident reporting and management along with resolution tasks. The application utilizes Java OOP methods to create a better public security framework which allows for effective crime report access by Public, PoliceOfficers and Admins. The system maintains secure controlled data exchange through its capacity for anonymous reporting and delivers immediate updates and supports multimedia content use.

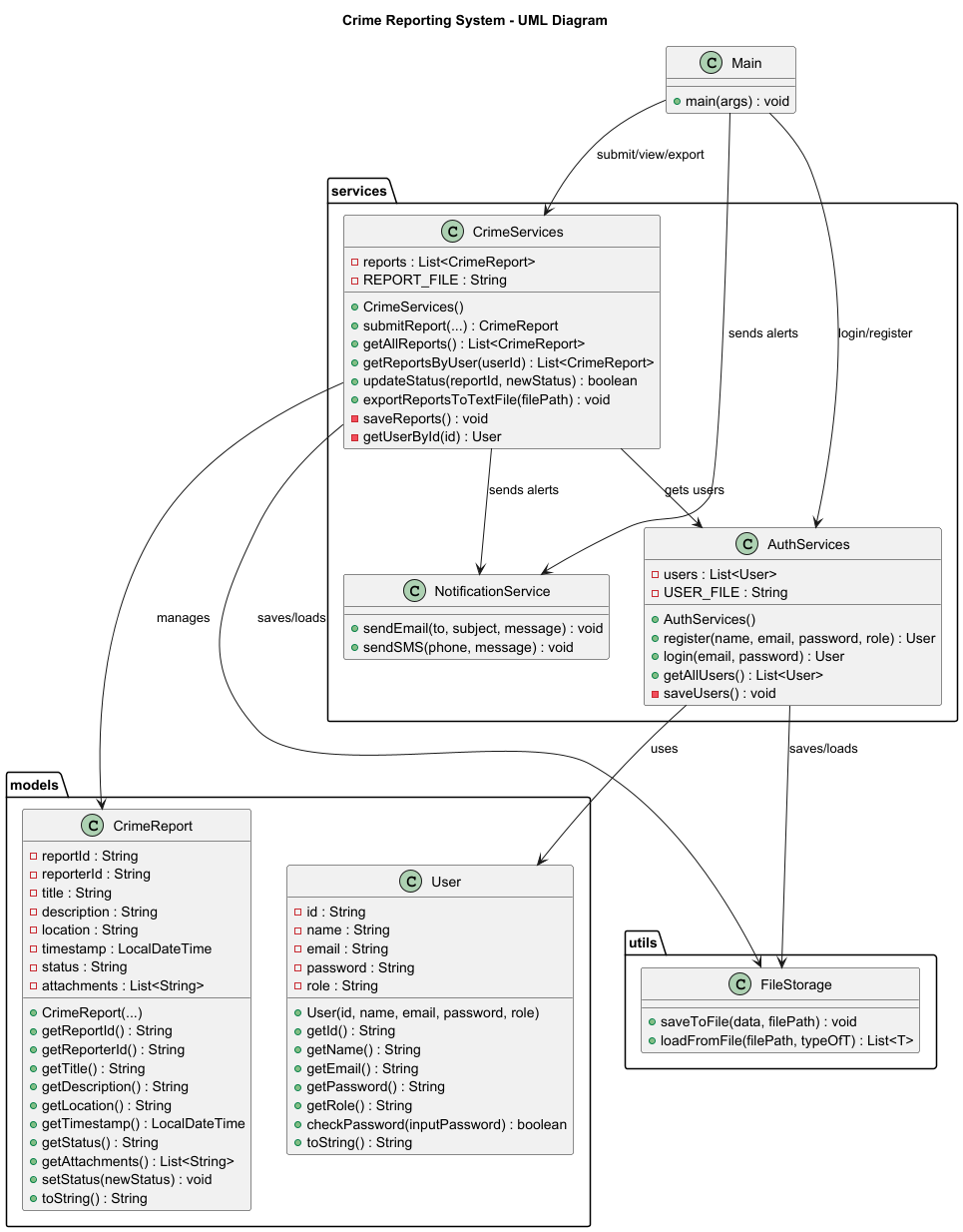
This system employs role-based access control enabled by Inheritance for different user types including Public and PoliceOfficer and Admin where each category possesses unique permissions. The technique of encapsulation safeguards important data including passwords and report information by ensuring complete privacy and safety measures. Through polymorphism the system enables users to access reports according to their assigned permissions which enables them to resolve reports when performing officer duties while managing user accounts as administrative personnel.

The report submission process adopts composition through the CrimeReport class that accepts Location and Media objects permitting users to create complete reporting forms with their positioning information and multimedia content.

The DatabaseManager class applies Encapsulation as its main design principle for data persistence duties involving file storage or simple database storage systems. The system uses individual FollowUp objects linked to crime reports to manage their follow-up process.

The Crime Reporting System implements Java’s OOP principles to create an effective platform which combines scalability with security and efficient data management for crime management.

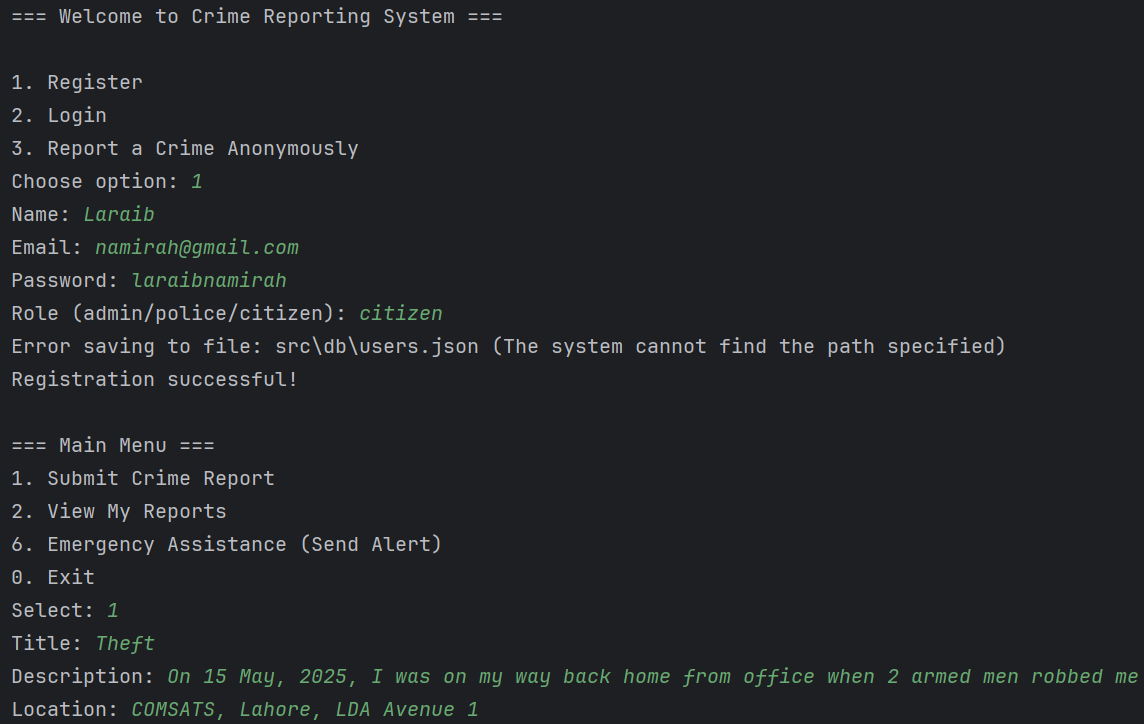
**UML**



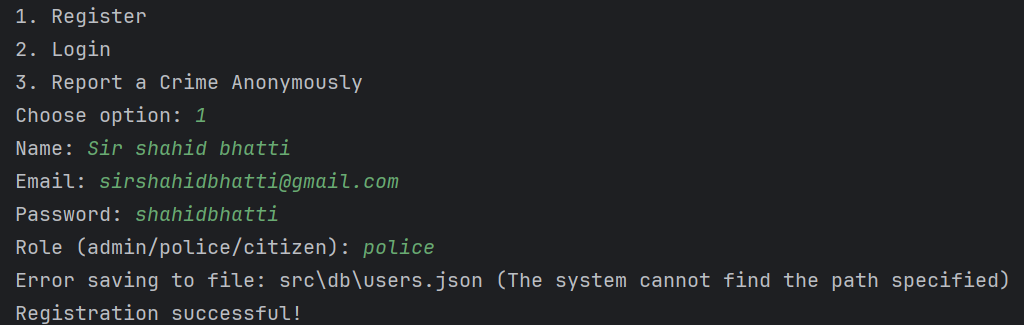
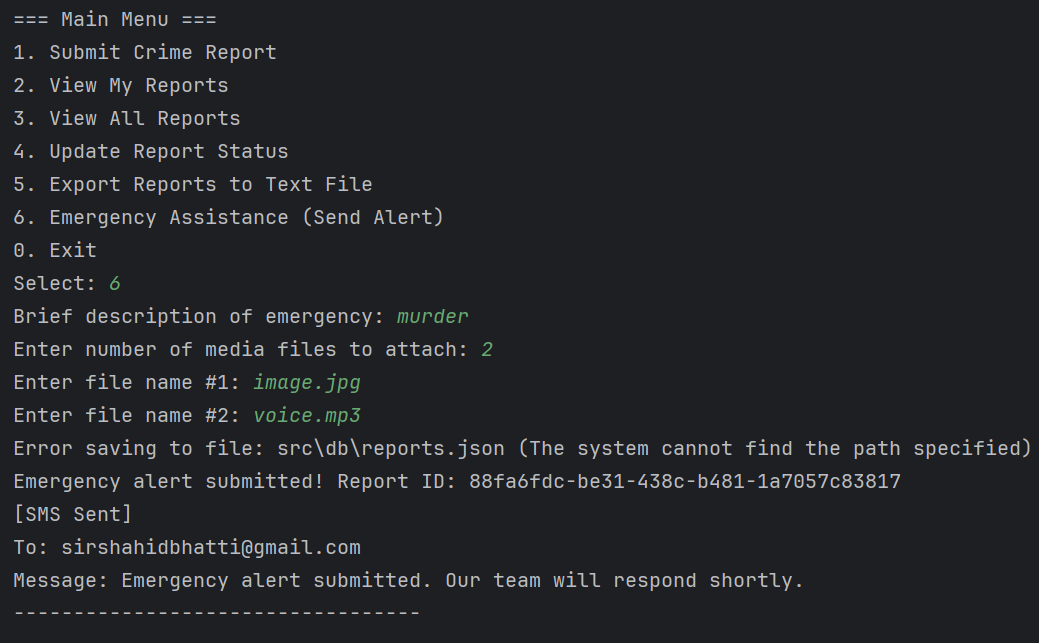
**OUTPUT**

**CASE 1(for Citizen)**

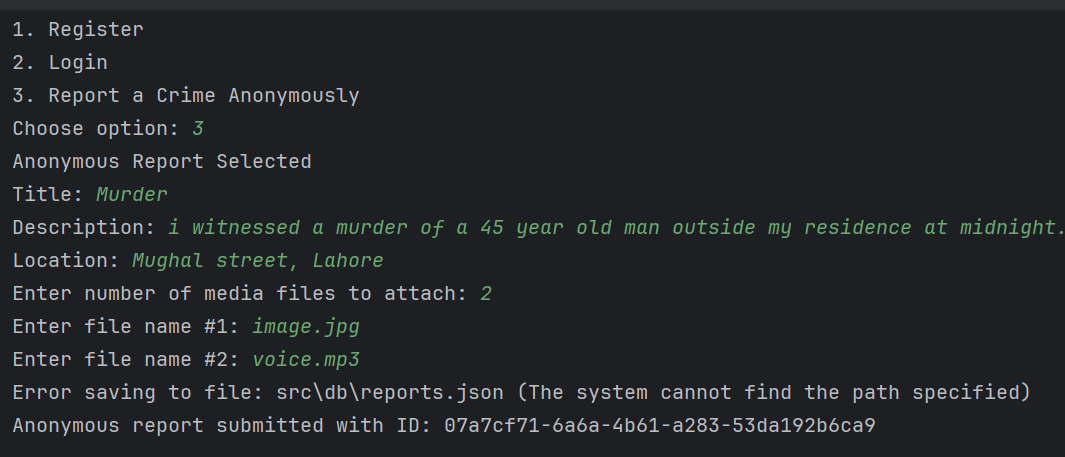
A screenshot of a computer

AI-generated content may be incorrect.

**CASE 2(for police)**



**CASE 3(For Anonymous)**



**CASE 4 (login)**

A screenshot of a computer

AI-generated content may be incorrect.

As the email does not match to the already registered email.

**CODE**

User.java

package models;  
  
public class User {  
 private String id;  
 private String name;  
 private String email;  
 private String password;  
 private String role; // "admin", "police", "citizen"  
  
 public User(String id, String name, String email, String password, String role) {  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 this.password = password;  
 this.role = role;  
 }  
  
 public String getId() {  
 return id;  
 }  
 public String getName() {  
 return name;  
 }  
 public String getEmail() {  
 return email;  
 }  
 public String getPassword() {  
 return password;  
 }  
 public String getRole() {  
 return role;  
 }  
  
 public boolean checkPassword(String inputPassword) {  
 return this.password.equals(inputPassword);  
 }  
  
 @Override  
 public String toString() {  
 return "[" + role.toUpperCase() + "] " + name + " <" + email + ">";  
 }  
}

AuthServices.java

package services;  
  
import com.google.gson.reflect.TypeToken;  
import models.User;  
import utils.FileStorage;  
  
import java.lang.reflect.Type;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.UUID;  
  
public class AuthServices {  
 private List<User> users;  
 private final String USER\_FILE = "src/db/users.json";  
  
 public AuthServices() {  
 // Load users from file when AuthService is created  
 Type userListType = new TypeToken<List<User>>() {}.getType();  
 users = FileStorage.*loadFromFile*(USER\_FILE, userListType);  
 if (users == null) {  
 users = new ArrayList<>();  
 }  
 }  
  
 // Register a new user  
 public User register(String name, String email, String password, String role) {  
 String id = UUID.*randomUUID*().toString();  
 User newUser = new User(id, name, email, password, role);  
 users.add(newUser);  
 saveUsers(); // Save to file  
 return newUser;  
 }  
  
 // Login  
 public User login(String email, String password) {  
 for (User user : users) {  
 if (user.getEmail().equalsIgnoreCase(email) && user.checkPassword(password)) {  
 return user;  
 }  
 }  
 return null;  
 }  
  
 // Save user list to file  
 private void saveUsers() {  
 FileStorage.*saveToFile*(users, USER\_FILE);  
 }  
  
 public List<User> getAllUsers() { //bcz users are private  
 return users;  
 }  
}

CrimeReport.java

package models;  
  
import java.time.LocalDateTime;  
import java.util.List;  
  
public class CrimeReport {  
 private String reportId;  
 private String reporterId;  
 private String title;  
 private String description;  
 private String location;  
 private LocalDateTime timestamp;  
 private String status; // "Pending", "Verified", "Rejected"  
 private List<String> attachments; // photo/video/audio file names  
  
 public CrimeReport(String reportId, String reporterId, String title, String description, String location, List<String> attachments) {  
 this.reportId = reportId;  
 this.reporterId = reporterId;  
 this.title = title;  
 this.description = description;  
 this.location = location;  
 this.attachments = attachments;  
 this.timestamp = LocalDateTime.*now*();// automatically captures current date and time everytime new report is submitted  
 this.status = "Pending";  
 }  
  
 public String getReportId() {  
 return reportId;  
 }  
 public String getReporterId() {  
 return reporterId;  
 }  
 public String getTitle() {  
 return title;  
 }  
 public String getDescription() {  
 return description;  
 }  
 public String getLocation() {  
 return location;  
 }  
 public LocalDateTime getTimestamp() {  
 return timestamp;  
 }  
 public String getStatus() {  
 return status;  
 }  
 public List<String> getAttachments() {  
 return attachments;  
 }  
  
  
 public void setStatus(String status) {  
 this.status = status;  
 }  
  
 @Override  
 public String toString() {  
 String attachCount = (attachments != null) ? attachments.size() + " file(s)" : "No files";  
 return "[Report: " + title + " | Status: " + status + " | Media: " + attachCount + "]";  
 }  
}

CrimeServices.java

package services;  
  
import com.google.gson.reflect.TypeToken;  
import models.CrimeReport;  
import models.User;  
import utils.FileStorage;  
  
import java.lang.reflect.Type;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.UUID;  
  
public class CrimeServices {  
 private List<CrimeReport> reports;  
 private final String REPORT\_FILE = "src/db/reports.json";  
  
 public CrimeServices() {  
 Type reportListType = new TypeToken<List<CrimeReport>>() {}.getType();  
 reports = FileStorage.*loadFromFile*(REPORT\_FILE, reportListType);  
 if (reports == null) {  
 reports = new ArrayList<>();  
 }  
 }  
 // submit report with media  
 public CrimeReport submitReport(String reporterId, String title, String description, String location, List<String> attachments) {  
 String id = UUID.*randomUUID*().toString();  
 CrimeReport report = new CrimeReport(id, reporterId, title, description, location, attachments);  
 reports.add(report);  
 saveReports();  
 return report;  
 }  
  
 public CrimeReport submitReport(String reporterId, String title, String description, String location) {  
 return submitReport(reporterId, title, description, location, new ArrayList<>());//overloaded method  
 }  
 //Used by admin or police to view everything.  
 public List<CrimeReport> getAllReports() {  
 return reports;  
 }  
 // view reprt by any specific user  
 public List<CrimeReport> getReportsByUser(String userId) {  
 List<CrimeReport> userReports = new ArrayList<>();  
 for (CrimeReport r : reports) {  
 if (r.getReporterId().equals(userId)) {  
 userReports.add(r);// Filters the full report list to return only those filed by a specific user  
 }  
 }  
 return userReports;  
 }  
  
 public boolean updateStatus(String reportId, String newStatus) {  
 for (CrimeReport report : reports) {  
 if (report.getReportId().equals(reportId)) {  
 report.setStatus(newStatus);  
 saveReports();  
  
 // Notify user  
 User user = getUserById(report.getReporterId());  
 // prevents from sending email to anonymous reporter  
 if (user != null && !user.getId().equals("anonymous")) {  
 NotificationService.*sendEmail*(  
 user.getEmail(),  
 "Report Status Updated",  
 "Your report titled '" + report.getTitle() + "' is now marked as: " + newStatus  
 );  
 }  
  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public void exportReportsToTextFile(String filePath) {  
 try {  
 List<CrimeReport> allReports = getAllReports();  
 StringBuilder builder = new StringBuilder();  
 //Then it loops over each report and appends its details to a long string:  
 for (CrimeReport r : allReports) {  
 builder.append("Report ID: ").append(r.getReportId()).append("\n");  
 builder.append("Title: ").append(r.getTitle()).append("\n");  
 builder.append("Description: ").append(r.getDescription()).append("\n");  
 builder.append("Location: ").append(r.getLocation()).append("\n");  
 builder.append("Status: ").append(r.getStatus()).append("\n");  
 builder.append("Reporter ID: ").append(r.getReporterId()).append("\n");  
 builder.append("Attachments: ").append(r.getAttachments()).append("\n");  
 builder.append("Timestamp: ").append(r.getTimestamp()).append("\n");  
 builder.append("--------------------------------------------------\n");  
 }  
 //writing to a file:  
 Files.*write*(Paths.*get*(filePath), builder.toString().getBytes());  
 System.*out*.println("Reports exported to: " + filePath);  
 } catch (Exception e) {  
 System.*out*.println("Failed to export reports: " + e.getMessage());  
 }  
 }  
 //save to json file  
 private void saveReports() {  
 FileStorage.*saveToFile*(reports, REPORT\_FILE);  
 }  
 // find user who reported and u can send him notification:  
 private User getUserById(String id) {  
 AuthServices authService = new AuthServices();  
 for (User u : authService.getAllUsers()) {  
 if (u.getId().equals(id)) return u;  
 }  
 return null;  
 }  
}

NotificationService.java

package services;  
  
public class NotificationService {  
  
 public static void sendEmail(String to, String subject, String message) {  
 System.*out*.println("[Email Sent]");  
 System.*out*.println("To: " + to);  
 System.*out*.println("Subject: " + subject);  
 System.*out*.println("Message: " + message);  
 System.*out*.println("----------------------------------");  
 }  
  
 public static void sendSMS(String phone, String message) {  
 System.*out*.println("[SMS Sent]");  
 System.*out*.println("To: " + phone);  
 System.*out*.println("Message: " + message);  
 System.*out*.println("----------------------------------");  
 }  
}

FileStorage,java

package utils;  
  
import com.google.gson.Gson;  
import com.google.gson.reflect.TypeToken;  
  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.lang.reflect.Type;  
import java.util.List;  
  
public class FileStorage {  
 private static final Gson *gson* = new Gson();  
  
 public static <T> void saveToFile(List<T> data, String filePath) {//Saves a list of Java objects to a file in JSON format  
 try (FileWriter writer = new FileWriter(filePath)) {  
 *gson*.toJson(data, writer);  
 } catch (Exception e) {  
 System.*out*.println("Error saving to file: " + e.getMessage());  
 }  
 }  
  
 public static <T> List<T> loadFromFile(String filePath, Type typeOfT) {  
 try (FileReader reader = new FileReader(filePath)) {  
 return *gson*.fromJson(reader, typeOfT);  
 } catch (Exception e) {  
 System.*out*.println("Error loading from file: " + e.getMessage());  
 return null;  
 }  
 }  
}

Main.java

import models.User;  
import models.CrimeReport;  
import services.AuthServices;  
import services.CrimeServices;  
import services.NotificationService;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 AuthServices authService = new AuthServices();  
 CrimeServices crimeService = new CrimeServices();  
  
 System.*out*.println("=== Welcome to Crime Reporting System ===");  
  
 User currentUser = null;  
  
 while (currentUser == null) {  
 System.*out*.println("\n1. Register");  
 System.*out*.println("2. Login");  
 System.*out*.println("3. Report a Crime Anonymously");  
 System.*out*.print("Choose option: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine();  
  
 if (choice == 1) {  
 System.*out*.print("Name: ");  
 String name = scanner.nextLine();  
 System.*out*.print("Email: ");  
 String email = scanner.nextLine();  
 System.*out*.print("Password: ");  
 String password = scanner.nextLine();  
 System.*out*.print("Role (admin/police/citizen): ");  
 String role = scanner.nextLine();  
  
 currentUser = authService.register(name, email, password, role);  
 System.*out*.println("Registration successful!");  
  
 } else if (choice == 2) {  
 System.*out*.print("Email: ");  
 String email = scanner.nextLine();  
 System.*out*.print("Password: ");  
 String password = scanner.nextLine();  
  
 currentUser = authService.login(email, password);  
 if (currentUser != null) {  
 System.*out*.println("Login successful! Welcome " + currentUser.getName());  
 } else {  
 System.*out*.println("Invalid email or password.");  
 }  
  
 } else if (choice == 3) {  
 System.*out*.println("Anonymous Report Selected");  
 System.*out*.print("Title: ");  
 String title = scanner.nextLine();  
 System.*out*.print("Description: ");  
 String description = scanner.nextLine();  
 System.*out*.print("Location: ");  
 String location = scanner.nextLine();  
  
 List<String> attachments = new ArrayList<>();  
 System.*out*.print("Enter number of media files to attach: ");  
 int fileCount = scanner.nextInt();  
 scanner.nextLine();  
 for (int i = 1; i <= fileCount; i++) {  
 System.*out*.print("Enter file name #" + i + ": ");  
 attachments.add(scanner.nextLine());  
 }  
  
 CrimeReport report = crimeService.submitReport(  
 "anonymous", title, description, location, attachments  
 );  
 System.*out*.println("Anonymous report submitted with ID: " + report.getReportId());  
  
 } else {  
 System.*out*.println("Invalid choice. Try again.");  
 }  
 }  
  
 boolean exit = false;  
 while (!exit) {  
 System.*out*.println("\n=== Main Menu ===");  
 System.*out*.println("1. Submit Crime Report");  
 System.*out*.println("2. View My Reports");  
 if (currentUser.getRole().equalsIgnoreCase("admin") || currentUser.getRole().equalsIgnoreCase("police")) {  
 System.*out*.println("3. View All Reports");  
 System.*out*.println("4. Update Report Status");  
 System.*out*.println("5. Export Reports to Text File");  
 }  
 System.*out*.println("6. Emergency Assistance (Send Alert)");  
 System.*out*.println("0. Exit");  
 System.*out*.print("Select: ");  
 int option = scanner.nextInt();  
 scanner.nextLine();  
  
 switch (option) {  
 case 1:  
 System.*out*.print("Title: ");  
 String title = scanner.nextLine();  
 System.*out*.print("Description: ");  
 String description = scanner.nextLine();  
 System.*out*.print("Location: ");  
 String location = scanner.nextLine();  
  
 List<String> attachments = new ArrayList<>();  
 System.*out*.print("Enter number of media files to attach: ");  
 int fileCount = scanner.nextInt();  
 scanner.nextLine();  
 for (int i = 1; i <= fileCount; i++) {  
 System.*out*.print("Enter file name #" + i + ": ");  
 attachments.add(scanner.nextLine());  
 }  
  
 CrimeReport report = crimeService.submitReport(  
 currentUser.getId(), title, description, location, attachments  
 );  
  
 System.*out*.println("Report submitted with ID: " + report.getReportId());  
  
 NotificationService.*sendEmail*(  
 currentUser.getEmail(),  
 "Report Submitted",  
 "Your report titled '" + title + "' has been successfully submitted."  
 );  
 break;  
  
 case 2:  
 List<CrimeReport> myReports = crimeService.getReportsByUser(currentUser.getId());  
 if (myReports.isEmpty()) {  
 System.*out*.println("No reports found.");  
 } else {  
 for (CrimeReport r : myReports) {  
 System.*out*.println(r);  
 }  
 }  
 break;  
  
 case 3:  
 if (currentUser.getRole().equalsIgnoreCase("admin") || currentUser.getRole().equalsIgnoreCase("police")) {  
 List<CrimeReport> allReports = crimeService.getAllReports();  
 for (CrimeReport r : allReports) {  
 System.*out*.println(r);  
 }  
 } else {  
 System.*out*.println("Unauthorized access.");  
 }  
 break;  
  
 case 4:  
 if (currentUser.getRole().equalsIgnoreCase("admin") || currentUser.getRole().equalsIgnoreCase("police")) {  
 System.*out*.print("Enter Report ID: ");  
 String reportId = scanner.nextLine();  
 System.*out*.print("New Status (Verified/Rejected): ");  
 String newStatus = scanner.nextLine();  
  
 boolean updated = crimeService.updateStatus(reportId, newStatus);  
 if (updated) {  
 System.*out*.println("Status updated.");  
 } else {  
 System.*out*.println("Report not found.");  
 }  
 } else {  
 System.*out*.println("Unauthorized access.");  
 }  
 break;  
  
 case 6:  
 System.*out*.print("Brief description of emergency: ");  
 String emergencyDesc = scanner.nextLine();  
 String emergencyLocation = "User's current location (not tracked in console version)";  
  
 List<String> emergencyFiles = new ArrayList<>();  
 System.*out*.print("Enter number of media files to attach: ");  
 int emergencyFileCount = scanner.nextInt();  
 scanner.nextLine();  
 for (int i = 1; i <= emergencyFileCount; i++) {  
 System.*out*.print("Enter file name #" + i + ": ");  
 emergencyFiles.add(scanner.nextLine());  
 }  
  
 CrimeReport emergencyReport = crimeService.submitReport(  
 currentUser.getId(), "🚨 Emergency Alert", emergencyDesc, emergencyLocation, emergencyFiles  
 );  
  
 System.*out*.println("Emergency alert submitted! Report ID: " + emergencyReport.getReportId());  
  
 NotificationService.*sendSMS*(  
 currentUser.getEmail(),  
 "Emergency alert submitted. Our team will respond shortly."  
 );  
 break;  
  
 case 5:  
 if (currentUser.getRole().equalsIgnoreCase("admin") || currentUser.getRole().equalsIgnoreCase("police")) {  
 System.*out*.print("Enter filename to export (e.g., reports.txt): ");  
 String filename = scanner.nextLine();  
 String exportPath = "db/" + filename;  
 crimeService.exportReportsToTextFile(exportPath);  
 } else {  
 System.*out*.println("Unauthorized access.");  
 }  
 break;  
  
 case 0:  
 exit = true;  
 System.*out*.println("Thank you for using the Crime Reporting System!");  
 break;  
  
 default:  
 System.*out*.println("Invalid option.");  
 }  
 }  
  
 scanner.close();  
 }  
}