**CRIMINAL DETECTOR MANAGEMENT SYSTEM**

**A MINI PROJECT REPORT**

| **Submitted by** |  |  |
| --- | --- | --- |
| **SANDHYA SREE M** |  | **231801146** |
| **RITHIKA SMITHI S** |  | **231801138** |
| **RAGHUL S** |  | **231801131** |

In partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE OF DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM

CHENNAI-602105

2024 - 2025

**BONAFIDE CERTIFICATE**

Certified that this project report “**CRIME DETECTOR MANAGEMENT**” is the Bonafide work of **“SANDHYASREE M(231801146), RITHIKA SMITHI S (231801138), RAGHUL S(231801131)”**

who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

* The proposed system focuses on Crime Records Management for all police stations across the country. It aims to centralize Information Management in Crime for efficient sharing of critical information across all stations. The system will initially be implemented across cities and towns, but will eventually be interlinked to allow police detectives to access information across all records in the state
* The system will also generate information for proactive and preventive measures against crime. The project is designed with a distributed architecture, centralized database storage, and a user interface using SQL server constructs and DOT Net technologies. The application handles various modules and associated reports, adhering to administrative staff's strategies and standard

**TABLE OF CONTENTS**

1. **INTRODUCTION:**
   1. INTRODUCTION
   2. OBJECTIVES
   3. MODULES
2. **SURVEY OF TECHNOLOGIES:**
   1. SOFTWARE DESCRIPTION
   2. LANGUAGES:

2.2.1 SQL

* + 1. JAVA

1. **REQUIREMENTS AND ANALYSIS:**

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIRE

3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

3.5 NORMALIZATION

**4. PROGRAM CODE**

**5. RESULTS AND DISCUSSION**

**6. CONCLUSION**

**7. REFERENCES**

1. **INTRODUCTION:**

This project Online Crime Reporting System has been developed on C#, ASP and SQL Server. The main aim for this project is to provide all crime management solutions which are easily accessible to everyone. This system starts with the every people who want to log a complaint through the internet so this project is very useful for police department and social worker to find out the problem in the society without people are coming to the police station every time. The main purpose of this system is to manage criminal details in a centralized database and provide solution for public to give complaint through online and get online and get online service.

This project provides a lot of features to manage all the data in very well manner. The system has been developed to override the problems prevailing in the practicing manual system. This project is supported to eliminate and in some cases reduce the hardships

faced by this existing system. Moreover this system is designed for the particular need of the company to carry out operations in a smooth and effective manner. The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly.

Online crime reporting system as described above, can lead

to error free, secure, reliable and fast management system. It can

assist the user to concentrate on their other activities rather to concentrate on the record keeping. The purpose of this project is to automate the existing manual system by the help of computerized equipment’s and fullfledged computer system, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the process.

**Objectives of the Supermarket Management System**

**DATA ENTRY:**

The importance of user-friendly data entry interfaces, standardized input formats, validation rules, automated data collection methods, and access controls to ensure data integrity and security, while also reducing manual errors and ensuring only authorized personnel can access the system.

**DATA VISUALIZATION:**

The system includes dynamic dashboards for real-time crime trends, geospatial mapping for resource allocation, customizable reports, and various visualization techniques. It also includes user training to ensure accurate interpretation of data and efficient use of visualization tools.

**TESTING AND QUALITY ASSURANCE:**

The system's testing process includes unit testing, integration testing, performance testing, user acceptance testing, and security testing. Quality assurance is ensured through clear standards, regular audits, robust error handling mechanisms, and a culture of continuous improvement. Regular audits ensure compliance with established standards, while error handling mechanisms track and resolve issues. A comprehensive documentation of system processes, data standards, and testing protocols facilitates onboarding of new users.

**MODULES:**

**A crime detector management system(CDMS) requires a robust database to efficiently store and manage various types of data. Here are some essential modules that you can consider incorporating into your CDMS using a database management system:**

**Core Modules:**

* Searching a crime
* Reporting a crime

**Additional Modules:**

* Community Engagement
* Integrations

1. **SURVEY OF TECHNOLOGIES:**

2.1 Software Description :

This project utilizes a combination of software tools to create a comprehensive and efficient supermarket management system:

* **Database Management System (DBMS):** MySQL is chosen as the DBMS for its reliability, performance, and widespread use in various applications.

**JSP :**

**JavaServer Pages** (**JSP**) is a Java technology that allows

software developers to dynamically generate HTML, XML or other

types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

* **Visual studio :** HTML,CSS ,JavaScript.

2.2 Languages :

* **SQL:** Structured Query Language is used to interact with the MySQL database, defining data structures, performing queries, and managing data integrity.
* **JAVA:**

The Java programming language is a high-level language that can

be characterized by all of the following buzzwords:

 Simple

 Architecture neutral

 Object oriented

 Portable

 Distributed

 High performance

**2.2.1 SQL :**

SQL plays a crucial role in the supermarket management system by:

* Creating database tables: Defining the structure of tables to store product information, customer details, sales transactions, employee data, and more.
* Performing queries: Retrieving, updating, inserting, and deleting data from the database tables based on specific criteria.
* Managing data integrity: Ensuring data consistency and accuracy through constraints like primary keys, foreign keys, and data types.
* Generating reports: Creating customized reports based on SQL queries to analyze sales trends, inventory levels, customer behavior, and other key metrics.

**2.2.2**·**JAVA :**

**Backend Development:**

Java handles **backend** logic for managing crime data, including CRUD operations for incidents, suspects, and investigations.Spring Boot is used to build secure, scalable web applications for CMS.

**Database Integration:**

Java supports seamless integration with databases (e.g., MySQL, PostgreSQL) via JDBC and JPA.

Stores and retrieves crime-related data, such as reports, evidence, and criminal profiles.

**Security:**

Java provides robust encryption and authentication mechanisms to protect sensitive crime data.

Spring Security enables role-based access control for authorized personnel.

**Data Analytics & Reporting:**

Java supports data analysis tools like Weka for identifying crime patterns and trends.

JasperReports is used to generate detailed reports, such as crime summaries and investigation statuses.

**Mobile Application Development:**

Java can be used to develop Android applications for on-the-go access to the CMS for law enforcement officers.

**3**.**REQUIREMENTS AND ANALYSIS :**

**3.1 Requirement Specification**

**When developing a Crime Detection Management System, it's essential to conduct a thorough requirements analysis to ensure the system meets the needs of law enforcement agencies, stakeholders, and the community. This section outlines the key requirements, categorized into functional, non-functional, and technical requirements.**

**Functional Requirements :**

#### Incident Management:

* + - Incident Reporting
    - **Incident Status Tracking**

#### Suspect and Witness Management:

* **Profile Management**
* **Search and Filter**
* **Evidence Management**
* **Evidence Entry**
* **Chain of Custody**

#### User Management

* **Role-Based Access Control**
* **User Profiles**

**Non-Functional Requirements:**

* User-Friendly Interface
* Training and Documentation

**Performance**

* Response Time
* Scalability

**Security**

* Data Protection
* Access Control

**Reliability**

* System Availability
* Data Backup and Recovery

**3. Technical Requirements**

**Database Management System**

* RDBMS Selection

**B. Development Environment**

* **Programming Languages**: Using appropriate programming languages for front-end (JavaScript, HTML/CSS) and back-end ( Java).

**Hosting and Infrastructure**

* **Cloud Services**: Evaluate cloud platforms (e.g., AWS, Azure) for hosting the application and database.
* **Network Infrastructure**: Ensure adequate network bandwidth and security measures for data transmission.

**3.2 Hardware and Software Requirements :**

**Hardware Requirements**

**Server Specifications:**

* **Processor:** Quad-core (e.g., Intel Xeon or AMD Ryzen) minimum; multi-core recommended.
* **RAM:** 16 GB minimum; 32 GB recommended for larger datasets.
* **Storage:** 500 GB SSD minimum; 1 TB SSD recommended with additional HDD for backups.
* **Network Interface:** Gigabit Ethernet.

**Client Workstations:**

* **Processor:** Dual-core minimum; quad-core recommended.
* **RAM:** 8 GB minimum; 16 GB recommended.
* **Storage:** 256 GB SSD minimum; 512 GB SSD recommended.
* **Display:** 15-inch monitor (1080p minimum); dual monitors recommended.

**Software Requirements**

**Operating System:**

* **Server:** Windows Server.
* **Client:** Windows 10/11

**Database Management System (DBMS):**

* **Options:** MySQL

**Development Tools:**

* **Languages:** Backend (Java). Frontend (JavaScript with frameworks like React or Angular).
* **IDE:** Visual Studio Code.

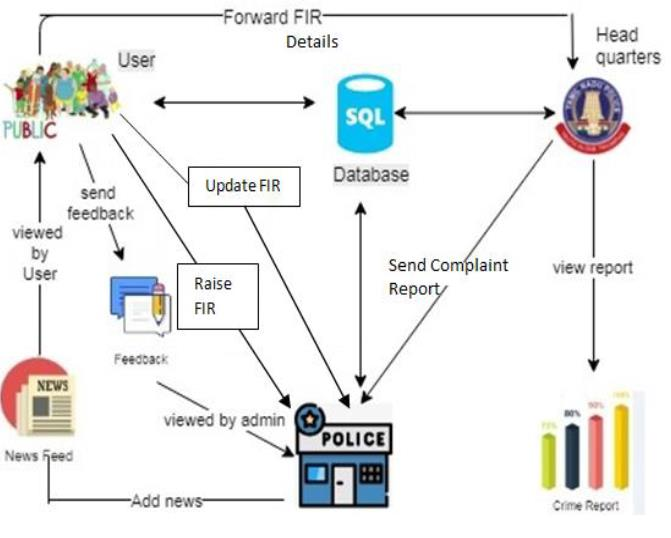
**Web Server:**

* **Options:** Apache HTTP Server or Nginx.

**Security Software:**

* **Firewall:** Hardware/software firewall.
* **Antivirus:** Reliable endpoint protection.
* **Encryption Tools:** SSL/TLS for data security.

**3.3 ARCHITECTURE DIAGRAM :**



**3.4 ER DIAGRAM** :

A diagram of a company

Description automatically generated

#### 1. ****Application Class (Main class)****

package com.crime.management;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CrimeManagementApplication {

public static void main(String[] args) {

SpringApplication.run(CrimeManagementApplication.class, args);

}

}

#### ****Model Classes****

Here, we’ll create basic models for **User** and **CrimeReport** to map to the database tables.

java

Copy code

package com.crime.management.model;

import javax.persistence.\*;

@Entitypublic class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long userId;

private String username;

private String password;

// Getters and Setters

}

@Entitypublic class CrimeReport {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long crimeId;

private String crimeType;

private String description;

private String dateOfCrime;

private String location;

// Getters and Setters

}

#### 3. ****Repository Layer****

Create repositories for interacting with the database:

java

Copy code

package com.crime.management.repository;

import com.crime.management.model.User;import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

User findByUsername(String username);

}

package com.crime.management.repository;

import com.crime.management.model.CrimeReport;import org.springframework.data.jpa.repository.JpaRepository;

public interface CrimeReportRepository extends JpaRepository<CrimeReport, Long> {

}

#### ****Service Layer****

The service layer contains business logic such as user validation and crime report handling.

java

Copy code

package com.crime.management.service;

import com.crime.management.model.User;import com.crime.management.repository.UserRepository;import org.springframework.beans.factory.annotation.Autowired;import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;import org.springframework.stereotype.Service;

@Servicepublic class UserService {

@Autowired

private UserRepository userRepository;

private BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();

public User validateUser(String username, String password) {

User user = userRepository.findByUsername(username);

if (user != null && passwordEncoder.matches(password, user.getPassword())) {

return user;

}

return null;

}

}

#### 5. ****Controller Layer****

This is where we handle the routing, user sessions, and HTML templates.

java

Copy code

package com.crime.management.controller;

import com.crime.management.model.CrimeReport;import com.crime.management.model.User;import com.crime.management.service.UserService;import org.springframework.beans.factory.annotation.Autowired;import org.springframework.beans.factory.annotation.Value;import org.springframework.context.annotation.ComponentScan;import org.springframework.security.core.context.SecurityContextHolder;import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;import org.springframework.ui.Model;import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpSession;

@Controllerpublic class CrimeController {

@Autowired

private UserService userService;

@Autowired

private CrimeReportRepository crimeReportRepository;

@GetMapping("/")

public String loginPage() {

return "login";

}

@PostMapping("/login")

public String loginUser(@RequestParam String username, @RequestParam String password, HttpSession session, Model model) {

User user = userService.validateUser(username, password);

if (user != null) {

session.setAttribute("userId", user.getUserId());

return "redirect:/dashboard";

}

model.addAttribute("error", "Invalid username or password.");

return "login";

}

@GetMapping("/dashboard")

public String dashboard(HttpSession session, Model model) {

if (session.getAttribute("userId") == null) {

return "redirect:/";

}

return "dashboard";

}

@GetMapping("/report\_crime")

public String reportCrimePage(HttpSession session) {

if (session.getAttribute("userId") == null) {

return "redirect:/";

}

return "report\_crime";

}

@PostMapping("/report\_crime")

public String reportCrime(@RequestParam String crimeType, @RequestParam String description,

@RequestParam String dateOfCrime, @RequestParam String location, HttpSession session) {

if (session.getAttribute("userId") == null) {

return "redirect:/";

}

CrimeReport crimeReport = new CrimeReport();

crimeReport.setCrimeType(crimeType);

crimeReport.setDescription(description);

crimeReport.setDateOfCrime(dateOfCrime);

crimeReport.setLocation(location);

crimeReportRepository.save(crimeReport);

return "redirect:/dashboard";

}

@GetMapping("/search\_crime")

public String searchCrimePage(HttpSession session) {

if (session.getAttribute("userId") == null) {

return "redirect:/";

}

return "search\_crime";

}

@PostMapping("/search\_crime")

public String searchCrime(@RequestParam String location, HttpSession session, Model model) {

if (session.getAttribute("userId") == null) {

return "redirect:/";

}

model.addAttribute("crimes", crimeReportRepository.findAll());

return "crime\_results";

}

@GetMapping("/logout")

public String logout(HttpSession session) {

session.invalidate();

return "redirect:/";

}

}

### Database Configuration

In your application.properties or application.yml, configure your MySQL connection:

spring.datasource.url=jdbc:mysql://localhost:3306/CrimeManagementSystem

spring.datasource.username=root

spring.datasource.password=Sanjay@2005

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

Front end

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login - Crime Management System</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f7f6;

color: #333;

margin: 0;

padding: 0;

}

.container {

max-width: 400px;

margin: 100px auto;

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

input[type="text"], input[type="password"] {

width: 100%;

padding: 10px;

margin: 10px 0;

border-radius: 5px;

border: 1px solid #ccc;

}

button {

background-color: #5cb85c;

border: none;

padding: 10px 20px;

color: white;

border-radius: 5px;

cursor: pointer;

width: 100%;

}

button:hover {

background-color: #4cae4c;

}

.alert-danger {

background-color: #f8d7da;

color: #721c24;

padding: 10px;

border-radius: 5px;

}

</style>

<script>

function validateLoginForm() {

var username = document.getElementById("username").value;

var password = document.getElementById("password").value;

if (username === "" || password === "") {

alert("Both username and password are required.");

return false;

}

return true;

}

</script>

</head>

<body>

<div class="container">

<h2>Login</h2>

<!-- Flash Message for Errors -->

<!-- You can uncomment the below section for Flask flash message -->

<!--

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<div class="alert-danger">

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

</div>

{% endif %}

{% endwith %}

-->

<form method="POST" action="/login" onsubmit="return validateLoginForm()">

<input type="text" id="username" name="username" placeholder="Username" required>

<input type="password" id="password" name="password" placeholder="Password" required>

<button type="submit">Login</button>

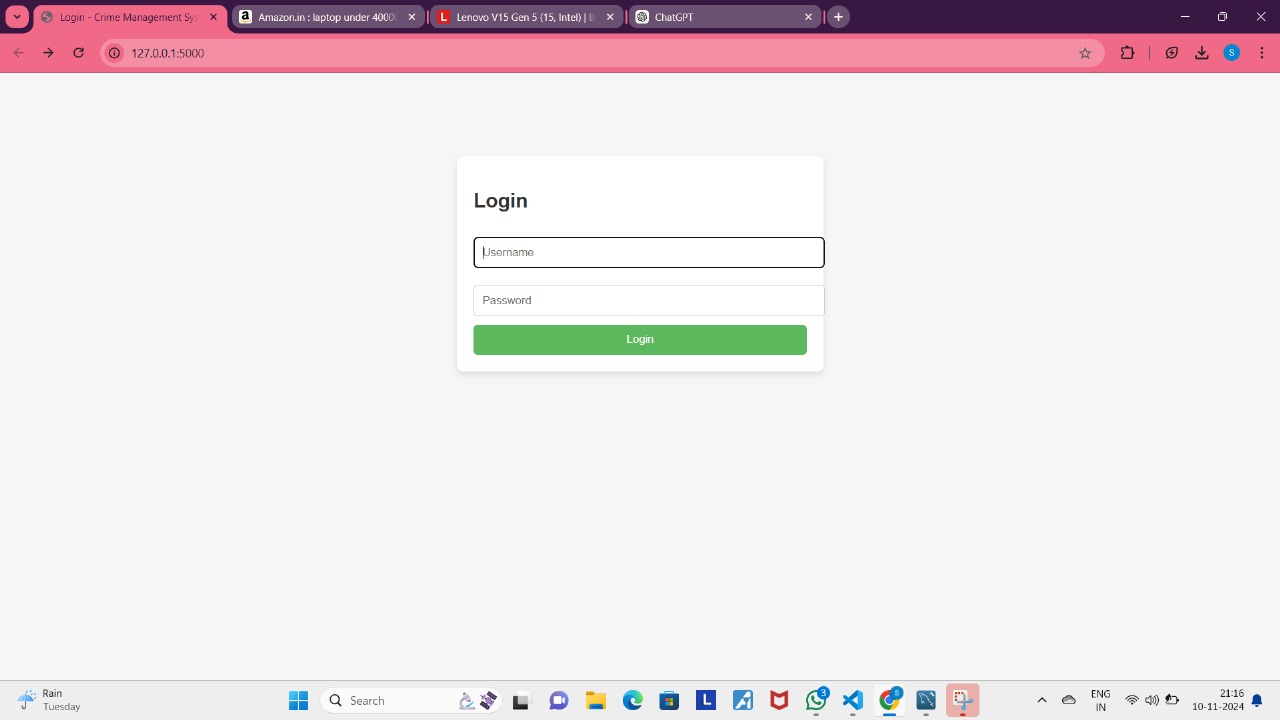
</form>

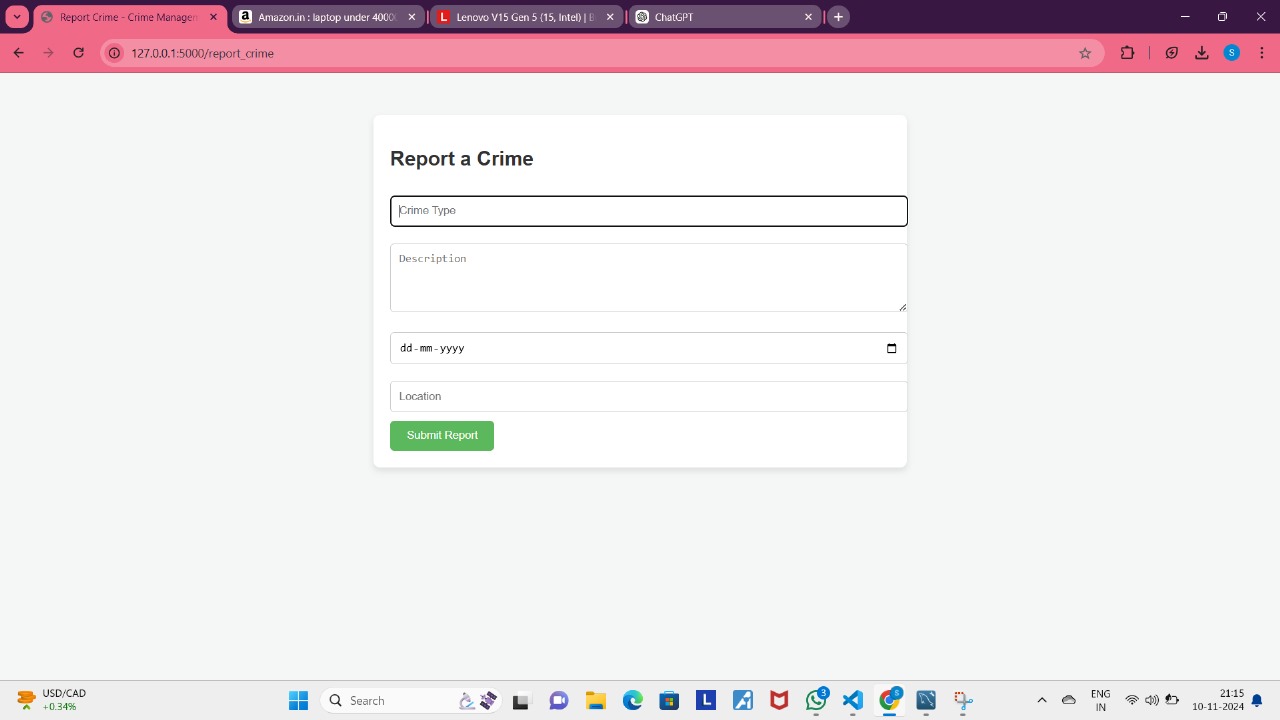
</div>

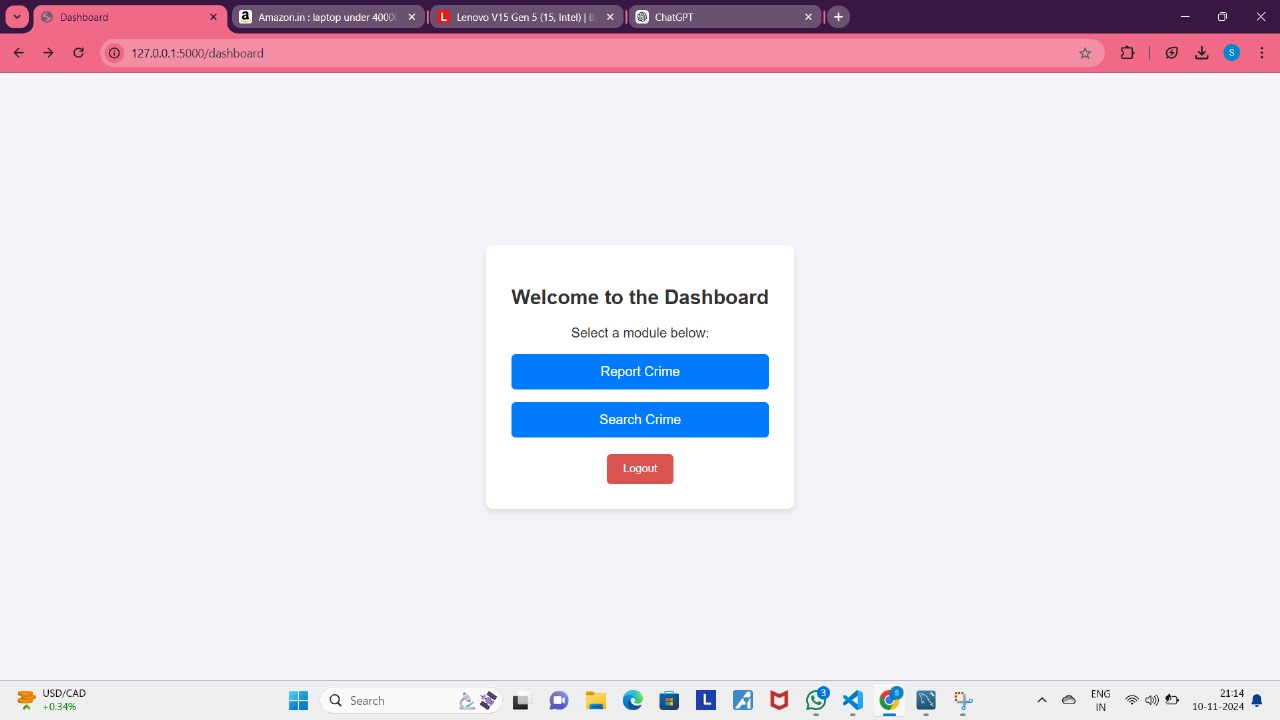
</body>

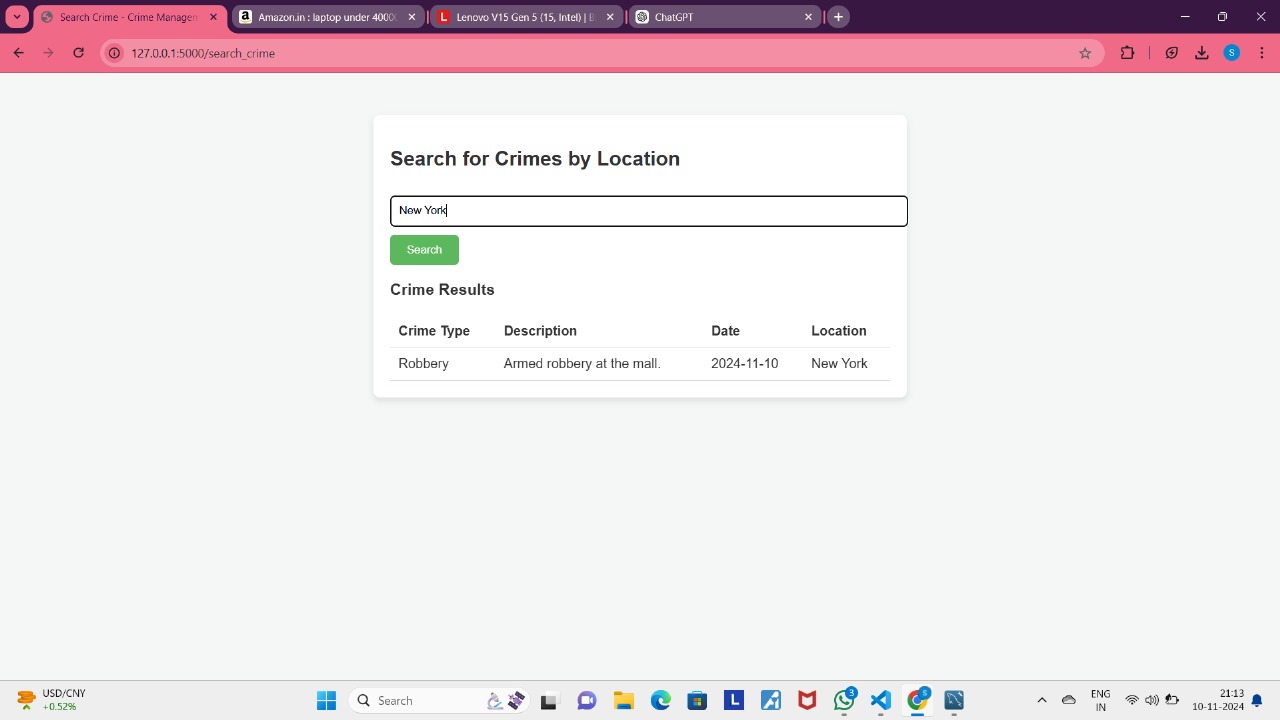
</html>

OUTPUT:









**CONCLUSION:**

### Conclusion for Crime Detector Management System in DBMS

The ‘Crime Detector System’ enhances law enforcement capabilities by leveraging data-driven insights to detect, analyze, and predict criminal activities. By collecting and organizing crime data (e.g., crime types, locations, and suspects) into a structured database, the system helps identify patterns, optimize resource allocation, and improve response times. Through advanced SQL queries and analytics, the system provides actionable insights for crime prevention and decision-making.

The ‘Crime Detector System’ improves public safety by enabling proactive measures, increasing transparency, and fostering collaboration between law enforcement and the community. While challenges like data privacy and accuracy exist, the system's future potential, especially with AI integration and real-time data, promises to further enhance crime detection and prevention efforts, making communities safer and more resilient.

**7.REFERENCES :**

* **From google,chatgpt**
* **Github**
* **Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.**
* **McConnell, S. (2004). *Code Complete* (2nd ed.). Microsoft Press**
* **For web development : Youtube**