# ANUDIP FOUNDATION

## Project Title

## “CRIME RECORD MANAGEMENT SYSTEM”

**By**


| Name | Enrollment No |
|---|---|
| Abdul Karim Nadkar | AF0481837 |


**Under Guidance**

**Of**

**Rajshri Thete**

# ABSTRACT

The **Crime Record Management System (CRMS)** is a software application designed to streamline the management, storage, and retrieval of crime-related data within law enforcement agencies. This system replaces traditional manual processes with a secure and efficient digital platform, allowing authorized personnel to record, update, and track criminal cases, suspect profiles, FIRs (First Information Reports), chargesheets, and investigation status.

The CRMS provides role-based access to users such as police officers, administrators, and investigation units, ensuring that sensitive information is handled securely. It supports functionalities like data entry, search and filter options, real-time case tracking, report generation, and statistical crime analysis to assist in decision-making and resource allocation.

By integrating database management and security protocols, the system enhances operational efficiency, reduces paperwork, minimizes errors, and improves data accessibility. The implementation of such a system helps in maintaining transparency, accountability, and faster processing of crime records, ultimately contributing to better law enforcement and public safety.

# ACKNOWLEDGEMENT

The project “Speed Typing Tester” is the Project work carried out by

| Name | Enrollment No |
| --- | --- |
| **Abdul Karim Nadkar** | **AF0481837** |

Under the Guidance

We are thankful to my project guide for guiding me to complete the Project. His suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics.

We are also thankful to my family member and friends who were always there to provide support and moral boost up.

# INDEX

# INTRODUCTION

In the modern age of digital transformation, law enforcement agencies face mounting challenges in maintaining, retrieving, and analyzing vast amounts of crime-related data. Traditionally, crime records have been managed manually or through outdated software systems, leading to inefficiencies, security vulnerabilities, and delays in justice delivery. A Crime Record Management System (CRMS) aims to overcome these limitations by providing a centralized, secure, and efficient platform for managing criminal records and case information.

The Crime Record Management System is a comprehensive software solution designed to facilitate the electronic recording, tracking, and management of criminal cases and suspect profiles within law enforcement departments. It is structured to serve multiple stakeholders, including police officers, investigators, administrators, and government agencies. The system enables real-time access to information, improves case tracking efficiency, ensures data consistency across departments, and supports data-driven decision-making through integrated analytics.

This system also offers features like biometric data storage, photo identity linkage, real-time FIR (First Information Report) filing, case status monitoring, automated report generation, and advanced search capabilities. Furthermore, it ensures that sensitive data is protected through secure access controls, encryption, and audit trails.

By transitioning to a digital platform, CRMS enhances operational transparency, speeds up investigations, and improves public trust in law enforcement. It provides the foundation for inter-agency collaboration, supports national crime databases, and plays a critical role in modern policing and criminal justice.

# MOTIVATION

Law enforcement agencies across the world face an increasing burden in managing growing volumes of criminal data and case files. Traditional systems—often manual or based on outdated software—fail to meet the speed, accuracy, and efficiency required for modern crime investigation and prevention. This inefficiency directly impacts the ability of police departments to track offenders, link related cases, and respond swiftly to emerging threats. The motivation behind developing a Crime Record Management System (CRMS) stem from the pressing need to digitize, centralize, and secure crime-related information.

## Key Motivational Factors:

### ❖ Inefficiencies in Manual Record-Keeping: -

Manual data entry in registers or spreadsheets is time-consuming and prone to human error. Retrieving old case files or cross-referencing data becomes a slow process, delaying investigations and hampering justice. CRMS automates data entry and storage, significantly reducing the margin for error and improving response time.

### ❖ Increasing Crime Rates and Data Complexity: -

As the population grows and urbanizes, the number and complexity of crimes also increase. Law enforcement agencies must deal with thousands of FIRs, suspect profiles, forensic records, and evidence logs. CRMS provides tools for categorizing, searching, and analyzing this complex data efficiently.

### ❖ Lack of Centralized and Real-time Access: -

In many jurisdictions, records are stored at individual police stations with little to no connectivity to other stations. This makes it difficult to identify repeat offenders, track case progress across departments, or coordinate

investigations. CRMS offers centralized access to data from any authorized location, improving coordination and oversight.

❖ **Enhancing Public Safety and Trust: -**

Efficient record-keeping and faster investigation times contribute to improved public safety and build trust between the public and law enforcement. CRMS ensures transparency and accountability by keeping a log of all user activities and actions performed on records.

❖ **Security and Confidentiality: -**

Crime records contain highly sensitive information, including personal details of victims and suspects, confidential case notes, and forensic evidence. CRMS enables advanced security mechanisms like role-based access control, encrypted storage, and audit logs, ensuring that only authorized personnel can view or modify data.

# PROBLEM STATEMENT

Law enforcement agencies are tasked with the critical responsibility of ensuring public safety, maintaining order, and investigating crimes. One of the most fundamental requirements in this process is the efficient management of criminal records, case files, evidence data, and investigation progress. However, in many regions, these tasks are still performed using manual record-keeping systems or outdated software, resulting in numerous operational and administrative challenges.

❖ **Challenges in the Existing System: -**

1. Manual and Paper-Based Record Keeping: -
   A significant number of police stations still rely on handwritten records, registers, and files to store information about criminals, FIRs, cases, and evidence. These records are difficult to organize, vulnerable to damage or loss, and time-consuming to retrieve.

2. Lack of Centralized Data: -
   Records are typically stored locally within a particular police station, with little or no integration with other departments or districts. This makes it difficult to track the movement of suspects across regions, identify repeat offenders, or share information between units in real time.

3. Inefficient Search and Retrieval: -
   Retrieving case details or past criminal history often requires combing through large volumes of physical files or unstructured digital data, which consumes valuable time and resources—particularly during investigations where speed is critical.

4. Limited Data Security and Privacy: -
   Sensitive crime-related data stored in physical files or unsecured systems is vulnerable to unauthorized access, theft, tampering, or destruction. There is often no audit trail to monitor who accessed or modified records.

5. Lack of Analytical Capabilities: -
   Traditional systems do not support analytics, pattern detection, or crime trend visualization. This restricts law enforcement's ability to make informed decisions, allocate resources efficiently, or predict crime hotspots.

6. Difficulty in Report Generation and Monitoring: -
   Preparing crime statistics reports, departmental performance reviews, or administrative summaries is a manual process, often plagued by inaccuracies and delays. This affects transparency and slows down decision-making processes.

7. No Real-Time Updates: -
   In the absence of real-time data updates, information may be outdated or incomplete, leading to misinformed decisions and ineffective coordination between police personnel.

# PURPOSE, OBJECTIVE AND GOALS

The primary purpose of the Crime Record Management System (CRMS) is to provide a centralized, secure, and efficient digital platform for storing, managing, and retrieving crime-related information. It aims to eliminate the limitations of manual record-keeping and legacy systems by automating the process of managing FIRs, criminal profiles, and case details. This system ensures that only authorized administrative users can access and manage sensitive data, improving both the security and integrity of crime records.

By using modern web technologies like Django (Python framework), MySQL, HTML, CSS, and JavaScript, the system offers a robust, scalable, and user-friendly interface designed specifically for internal use by law enforcement administrators.

## ❖ OBJECTIVE: -

1. To Digitize Crime Records: -
   Convert manual and paper-based crime records into a digital format to improve accessibility, accuracy, and efficiency.

2. To Centralize Data Storage: -
   Create a centralized database using MySQL for storing all criminal records, FIRs, and case updates in a structured and organized way.

3. To Ensure Secure Admin Access: -
   Restrict access to authorized admin users only, with secure login credentials, session management, and role-based access control.

4. To Improve Data Retrieval Efficiency: -
   Enable fast and accurate search functionalities to retrieve case information, criminal profiles, or FIRs without delay.

5. To Facilitate Easy Record Updates: -
   Allow administrators to add, edit, or delete records easily with audit logs to track changes.

## ❖ GOALS: -

6. **Automation**: - Eliminate redundant manual processes in crime record handling and reporting.

7. **Security**: - Implement robust user authentication and secure data storage to prevent unauthorized access.

8. **Data Accuracy**: - Minimize errors in record-keeping through standardized digital entry forms and validations.

9. **Scalability**: - Design the system to handle a growing volume of data and users over time.

10. **User Experience**: - Provide a clean, responsive, and intuitive interface using HTML, CSS, and JavaScript.

11. **Maintainability**: - Ensure that the system is easy to maintain, debug, and upgrade by following Django's modular development practices.

# LITERATURE SURVEY

The **literature survey** provides an overview of existing research, systems, and technologies related to Crime Record Management. It helps identify the current state of crime data management, highlight gaps in existing systems, and establish the need for a more efficient and secure solution using modern web technologies such as Django, MySQL, HTML, CSS, and JavaScript.

- **Manual Crime Record Management: -**
  In many police departments, especially in developing regions, crime records are still maintained in paper-based registers or Excel sheets. This method is prone to:
- Human error
- Misplacement or damage of records
- Time-consuming retrieval processes
- Lack of real-time accessibility

- **Legacy Crime Management Systems: -**
  Some law enforcement agencies use standalone desktop applications for crime record keeping. These systems are often built with outdated technologies (e.g., VB.NET, MS Access) and lack scalability and centralized access.

- **E-Governance and Smart Policing Initiatives: -**
  Governments in countries like India have initiated projects like CCTNS (Crime and Criminal Tracking Network and Systems) under the National e-Governance Plan to create a centralized criminal database accessible across all police stations.

- Web-Based Solutions in Recent Research: -
  Several academic projects and research papers have explored web-based crime management systems using PHP, Java, or Python. These systems typically offer features like:

- Admin login and dashboard

- FIR entry and case tracking
- Suspect profile management
- Report generation

## ❖ Gap Identified: -

While several crime record systems exist, most are either too basic for real-world deployment or too complex for small/local use**.** There's a need for a lightweight, admin-only, secure, and easy-to-deploy web application that:

- Centralizes crime records
- Offers user-friendly interface and secure login
- Supports case management and search
- Is built using open-source technologies like Django and MySQL

# PROJECT SCOPE AND LIMITATIONS

❖ **Project Scope: -**

The Crime Record Management System (CRMS) aims to digitize and centralize the process of storing, managing, and retrieving crime-related information. Built using Django, MySQL, HTML, CSS, and JavaScript, this system is designed for administrative use only, ensuring secure and controlled access to sensitive data.

❖ **The scope of the project includes: -**

❖ **Admin Authentication System: -**

**A secure login module to allow only authorized admin users to access the system.**

❖ **Crime Record Entry and Management: -**

**Admins can add, update, and delete records of crimes, FIRs, suspects, victims, and case statuses.**

❖ **Centralized Database: -**

**MySQL is used to store all data in a structured, searchable format.**

❖ **Efficient Search and Filter Options: -**

**Admins can quickly retrieve information about cases, suspects, or specific crime types using filters or keyword searches.**

## ❖ Project Limitations: -

Despite its usefulness, the CRMS has certain limitations due to its intended scope and the technologies used:

❖ **No Multi-user Roles or Public Access: -**
The system supports only a single role: admin. There are no features for police officers, investigators, or the public to log in or view data.

❖ **No Biometric or Facial Recognition Integration: -**
The system does not support linking biometric data like fingerprints or facial recognition due to project scope and resource constraints.

❖ **Lack of Cloud Hosting or Mobile Access: -**
The system is web-based but not optimized for cloud deployment or mobile platforms (unless separately configured).

❖ **No Real-Time Crime Mapping or Predictive Analytics: -**
The system does not include advanced crime trend analytics, heat maps, or prediction models due to limited data and project duration.

❖ **Basic Security Features: -**
While Django provides some in-built security, the system lacks advanced features like two-factor authentication or intrusion detection.

❖ **Not Integrated with National or Local Police Databases: -**
This is a standalone system and does not connect with government databases such as CCTNS or NCIC.

# ❖Future Enhancements: -

- **Multi-User Role Management: -**

  - Police Officer
  - Investigating Officer
  - Station Head
  - Public/User (limited access)

- **Mobile Application Integration: -**

  - File reports remotely
  - Access case files in the field
  - Receive real-time alerts or updates

- **Integration with Government Databases: -**

  - CCTNS (Crime and Criminal Tracking Network and Systems)
  - Aadhaar or other identity databases for suspect verification
  - Enable data sharing across different police stations and jurisdictions

- **Advanced Crime Analytics and Dashboards: -**

  - Use data visualization tools (e.g., charts, maps) to:
  - Analyze crime trends by location, type, or time
  - Identify hotspots and recurring crime patterns
  - Integrate predictive analytics using machine learning for crime forecasting

- **Biometric and Facial Recognition: -**

  - Fingerprint storage and matching
  - Face recognition using CCTV footage or suspect images

# COMPARATIVE STUDY AND EXISTING SYSTEM

❖ **Manual Record-Keeping System (Paper-Based): -**

- **Records are maintained in physical files/registers.**
- **Advantages: -**
  - Simple and low-cost.
  - No technical setup required.
- **Disadvantages: -**
  - Easily damaged or lost.
  - Difficult to retrieve/search information.
  - No data backup.
  - Time-consuming and error-prone

❖ **Spreadsheet-Based Systems (e.g., MS Excel): -**

- **Data stored in digital spreadsheets on local computers.**
- **Advantages: -**
  - Better than paper for searching and sorting.
  - Easy to create without programming.
- **Disadvantages: -**
  - No data security or access control.
  - Risk of data corruption or accidental deletion.
  - Not suitable for multi-user environments.
  - No automation or reporting features.

❖ **Legacy Desktop Applications (e.g., VB.NET + MS Access): -**

- **Standalone software installed on individual computers.**
- **Advantages: -**
  - Offers some level of automation.
  - Useful for small-scale setups.

- **Disadvantages: -**
    - No web-based access.
    - Poor scalability and maintainability.

❖ **CCTNS (Crime and Criminal Tracking Network and Systems): -**

- **A national-level digital policing initiative by the Indian government.**
- **Advantages: -**
    - Centralized database accessible nationwide.
    - Integration with biometric and identification systems.
    - Real-time case tracking and secure login.

- **Disadvantages: -**
    - Requires high infrastructure and training.
    - Not suitable for small/local deployments.
    - High development and maintenance cost.

❖ **Proposed Crime Record Management System (Django + MySQL + HTML/CSS/JS): -**

- **A web-based, admin-only system built using modern open-source technologies.**
- **Advantages: -**
    - Centralized, secure database (MySQL).
    - Admin authentication and access control.
    - User-friendly interface with search and reporting.
    - Scalable and modular – ready for future upgrades**.**

- **Disadvantages: -**
    - Only admin has access in current version.
    - No public reporting or multi-role support yet.
    - Limited analytics (planned for future versions).

# PROJECT PERSPECTIVE, FEATURES

The Crime Record Management System (CRMS) is designed to provide a centralized, secure, and efficient platform for storing and managing crime-related data within a police department or administrative unit. It replaces traditional paper-based or unstructured systems and offers a web-based solution that allows an authorized admin to access, manage, and monitor crime records.

This system is ideal for local law enforcement units, small stations, or training purposes where simplicity, accuracy, and control are more important than complex integrations or multi-user public access.

❖ **FEATURES: -**

- **Admin Login and Authentication: -**
  - Secure login system for administrators.
  - Prevents unauthorized access using Django's built-in user authentication.
  - Admin can view, add, update, or delete crime data**.**

- **Crime Record Management: -**
- Add new crime reports with details like:
  - Case number
  - Crime type
  - Location
  - Date and time
  - Description
- Edit or delete existing records as needed**.**

- **Suspect and Victim Management: -**
    - Maintain profiles of suspects and victims associated with each case.
    - Store details such as name, age, gender, address, and status (arrested, released, etc.).

- **Search and Filter Functionality: -**
- Search records by: -
    - Case number
    - Crime type
    - Date range
    - Suspect or victim name
- Filter data to quickly find relevant information.

- **Database Integration (MySQL): -**
    - All data is stored in a structured, relational MySQL database.
    - Ensures data consistency, integrity, and easy access.

- **Modular and Scalable Design: -**
- The system is designed to be easily extended in the future with:
    - New user roles (officer, public)
    - Analytics dashboards
    - Mobile compatibility
    - Biometric integration

# STAKEHOLDERS

❖ **System Administrator (Admin): -**

- **Role:** Primary user with full control over the system.
- **Responsibilities: -**
  - Manage user access (if multi-role system is introduced later).
  - Add, update, and delete crime records.
  - Oversee system security and data accuracy**.**
- **Importance: -** Ensures smooth functioning and data integrity of the system

❖ **Developers / IT Team: -**

- **Role: -** Technical stakeholders responsible for development, deployment, and maintenance of the system.
- **Responsibilities: -**
  - Implement new features and fix bugs.
  - Ensure system performance, backups, and security updates.
- **Importance: -** Provide technical backbone for the system's success.

❖ **Senior Police Officials / Supervisors: -**

- **Role: -** Use system reports for crime analysis and operational decisions.
- **Responsibilities: -**
  - **Monitor performance of officers.**
  - **Use statistics for planning and resource allocation.**
- **Importance: -** Use system data for high-level decision-making.

❖ **Police Officers / Investigating Officers: -**

- **Role: -** End users who may eventually be granted access to input and retrieve case-related information.
- **Responsibilities: -**
  - Record FIRs, suspects, evidence.

    - Track investigation progress.
    - Close or update case status.
- **Importance: -** Directly responsible for data entry and operational use in crime investigations.

❖ **Government and Regulatory Bodies: -**

- **Role: -** Oversight and policy-setting stakeholders.
- **Responsibilities: -**
    - Define data protection policies.
    - Enforce standard operating procedures.
- **Importance: -** Ensure compliance with legal frameworks and data protection norms.

# REQUIREMENT ANALYSIS

## ❖ Functional Requirements: -

- **Admin Authentication: -**
    - The system must provide a secure login for the admin.
    - Only authorized users should access the dashboard.

- **Crime Record Management: -**
- Admin must be able to **create, read, update, and delete** (CRUD) crime records.
- Each crime entry includes: -
    - Crime type
    - Date and time
    - Location
    - Description
    - Suspect/victim info
    - Case status

- **Search and Filter Crime Records: -**
- Admin can search cases by: -
    - Case ID
    - Suspect name
    - Crime type
    - Date range, etc.

❖ Case Status Update: -

❖ Admin should be able to update the progress/status of each case (e.g., "Open", "Under Investigation", "Closed").

## ❖ Performance Requirements: -

### ❖ Fast Data Retrieval: -

- The system must retrieve crime records in **less than 2 seconds** for search queries involving filters like crime type, date, or location.

### ❖ Concurrent Users: -

- The system should support **at least 5** simultaneous admin **sessions** without performance degradation (expandable for future multi-user roles).

### ❖ Scalability: -

- Should handle a minimum of **10,000+ crime entries** in the database without lag.
- System design must support **future upgrades** to handle larger datasets and concurrent access.

### ❖ Efficient CRUD Operations: -

- All Create, Read, Update, Delete operations on crime records must complete in under 3 seconds under normal load.

## ❖ Security Requirements: -

### ❖ Authentication & Authorization: -

- Only authorized admins can access the system via a secure login.
- Admin sessions must expire after a period of inactivity.

### ❖ Data Protection: -

- Passwords should be stored using hashing algorithms (Django uses PBKDF2 by default).

- Sensitive fields (e.g., suspect details) should not be visible without login.

❖ **Protection Against Common Attacks: -**

- SQL Injection (handled by Django ORM)
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF) (handled via Django's CSRF tokens)
- Brute force login attempts

❖ **Data Backup & Recovery: -**

- The database should support regular backups (manual or automated).
- Admins must be able to recover data in case of unexpected failure**.**

❖ **HTTPS Support (Recommended for Deployment): -**

- When deployed online, the system should use **HTTPS encryption** to ensure secure data transmission.

❖ **Audit Trails (Future Enhancement): -**

- A system log should be implemented in future versions to track admin activities (e.g., who edited/deleted records and when).

# DESIGN CONSTRAIN

❖ Technology Stack Constraint: -

❖ The system must be developed using:
- Django (Python framework) for backend
- MySQL for database management
- HTML, CSS, JavaScript for frontend design

❖ **Single User Role (Admin Only): -**

❖ The system supports only admin-level access.
❖ No support for multiple roles like police officers, investigators, or public users.
❖ All actions must be controlled through a single admin dashboard.

❖ **Web-Based Interface Only: -**

- The system is designed for desktop/laptop access through web browsers.
- No mobile application or device-optimized version is included in the current phase.

❖ **Local Deployment: -**

- The system is designed to run on **local servers** (e.g., WAMP, XAMPP, or Django local server).
- It is not optimized for **cloud deployment** or distributed access over multiple branches.

❖ **Limited Security Implementation**: -

- Relies on Django's default authentication system.
- No two-factor authentication (2FA), OTP verification, or biometric login is included.

### ❖ Static Data Entry: -

- All crime data must be manually entered by the admin.
- No integration with external systems (e.g., police databases, national crime portals, or sensors).

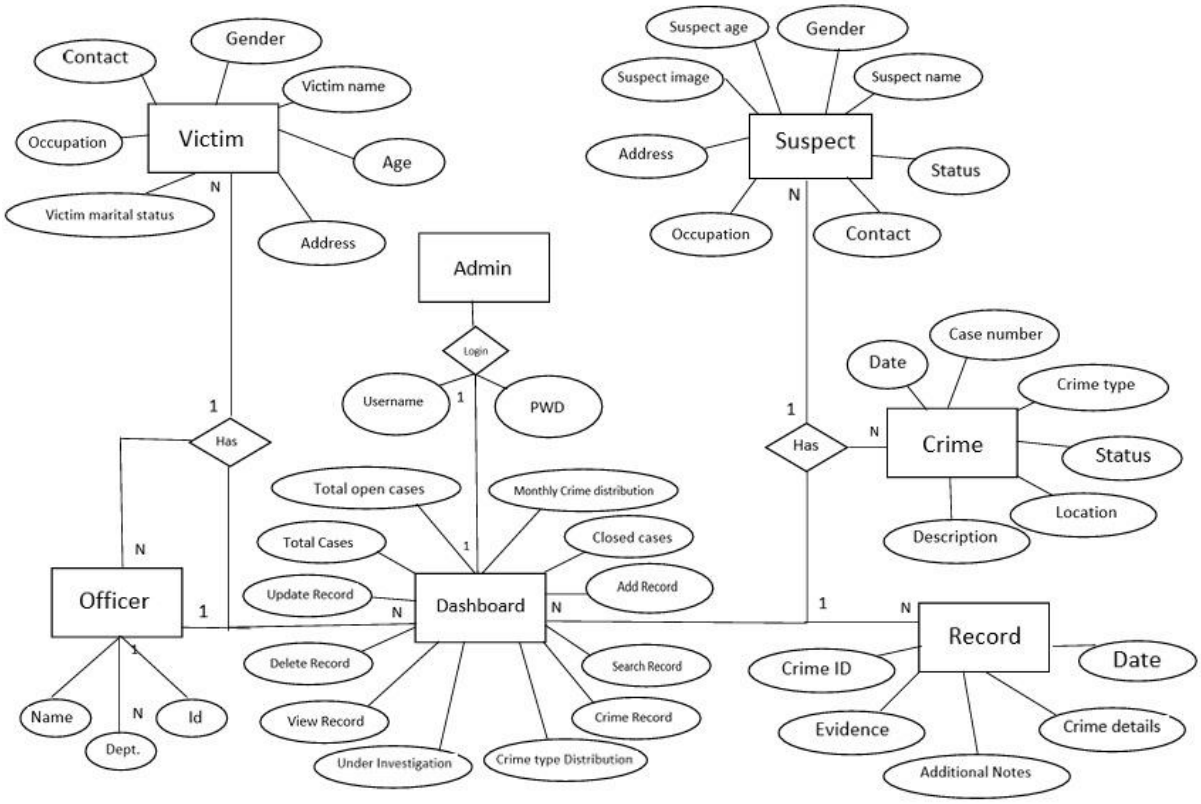
### ❖ Basic User Interface: -

- The UI is designed using basic HTML/CSS/JS, without frameworks like Bootstrap or Tailwind CSS.
- Design may not be fully responsive or compatible with all device screen sizes.

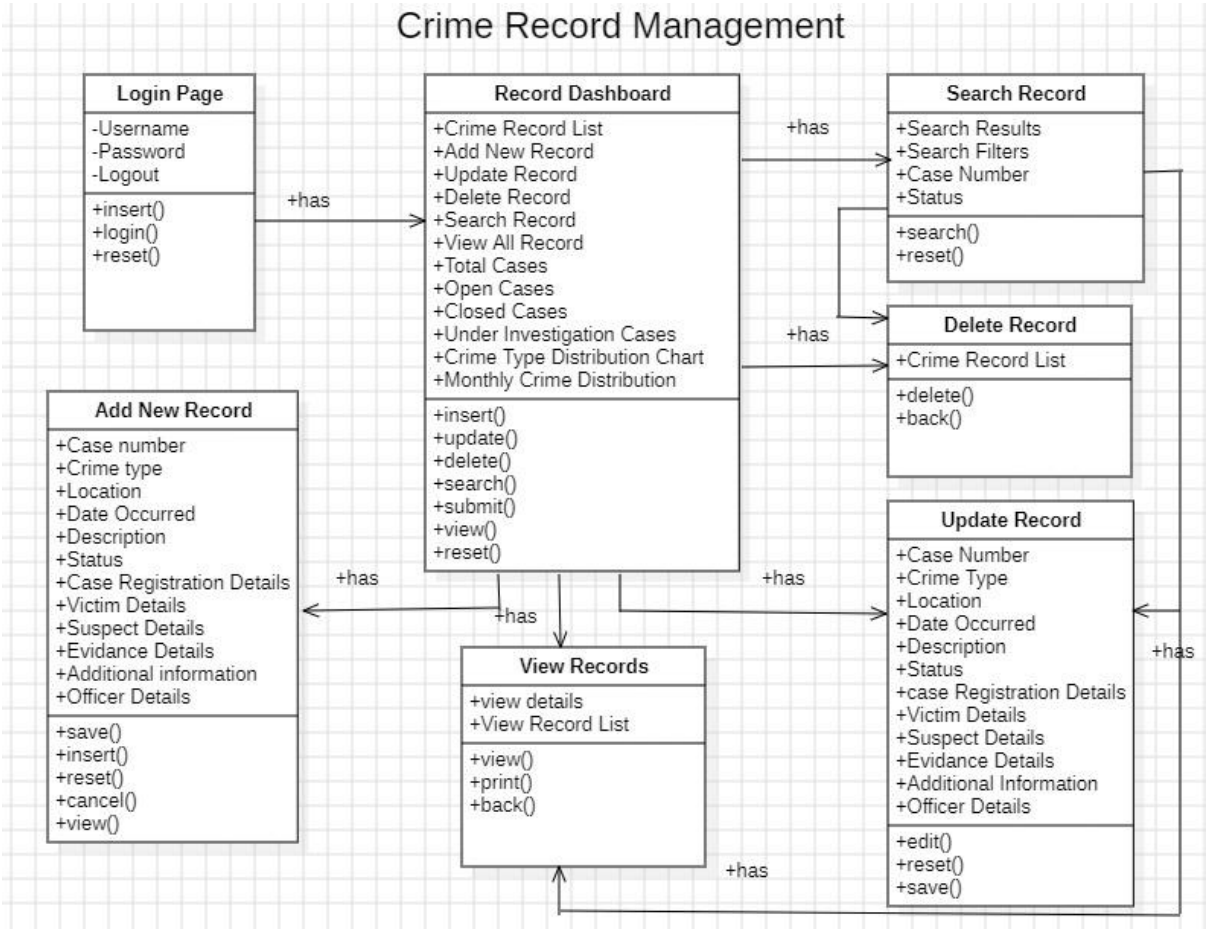
### ❖ No Real-Time Updates: -

- The system does not support real-time notifications, live tracking, or push updates.
- All updates are reflected only after manual data refresh or submission.

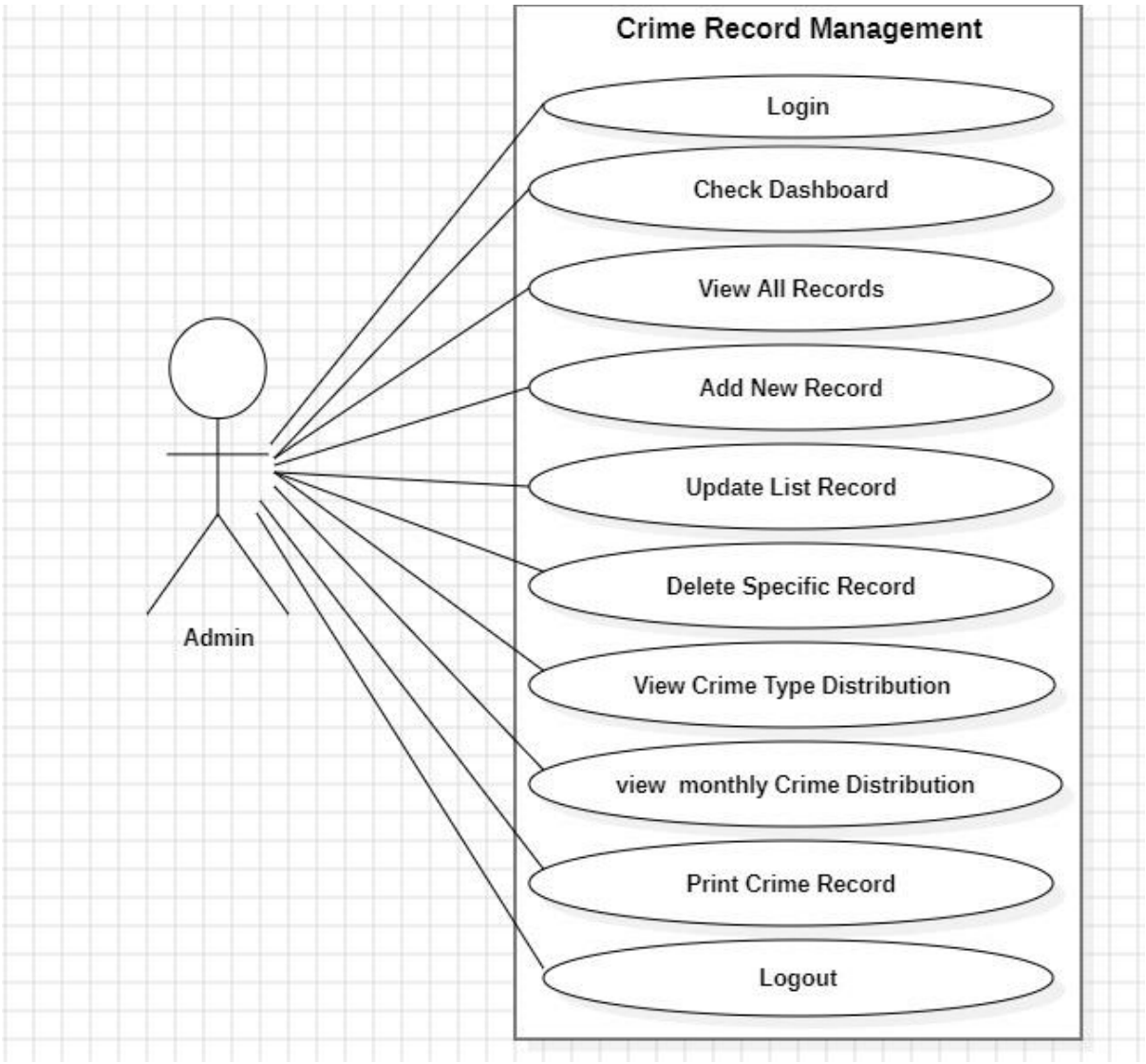
# ENTITY RELATIONSHIP DIAGRAM



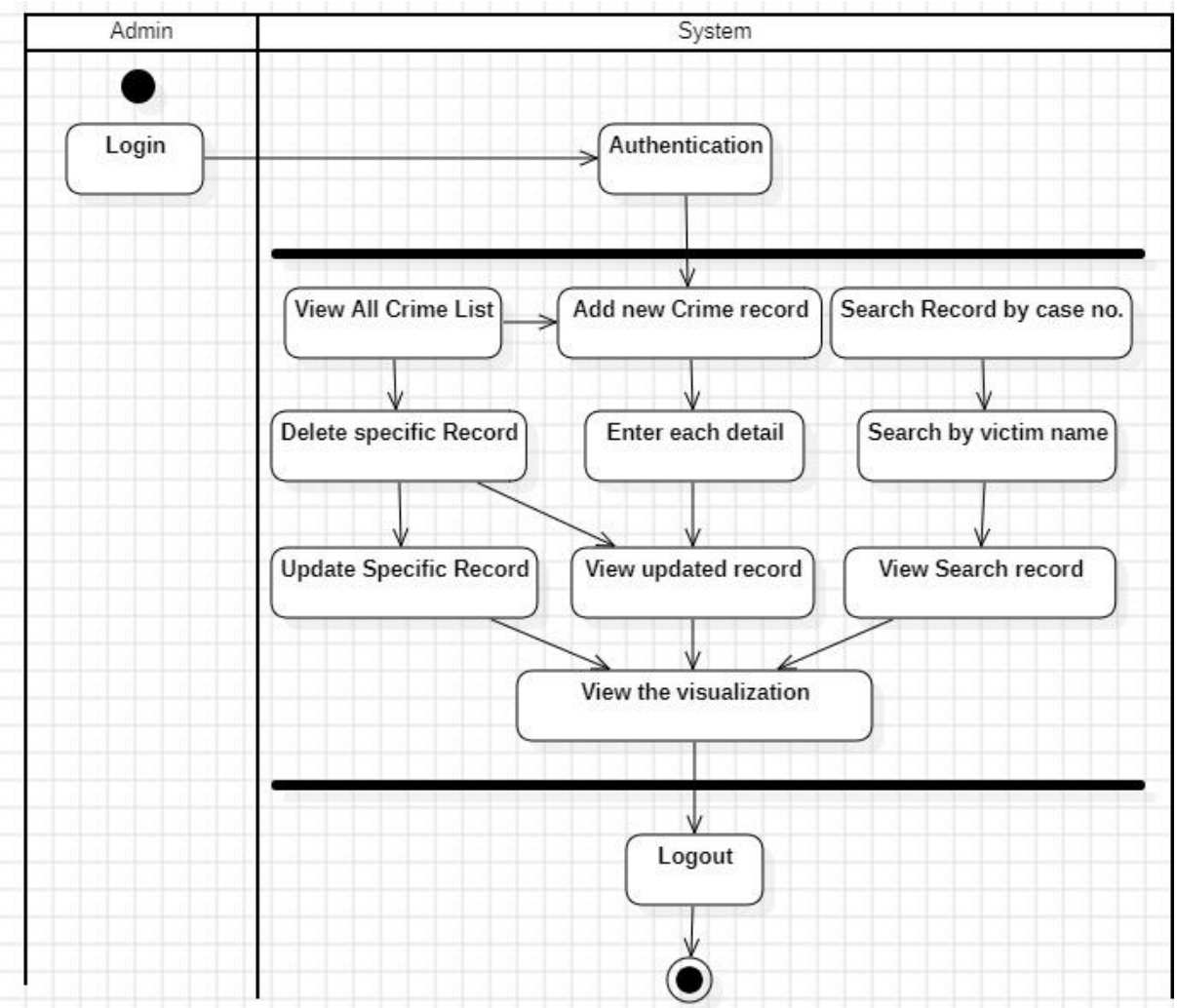
# CLASS DIAGRAM



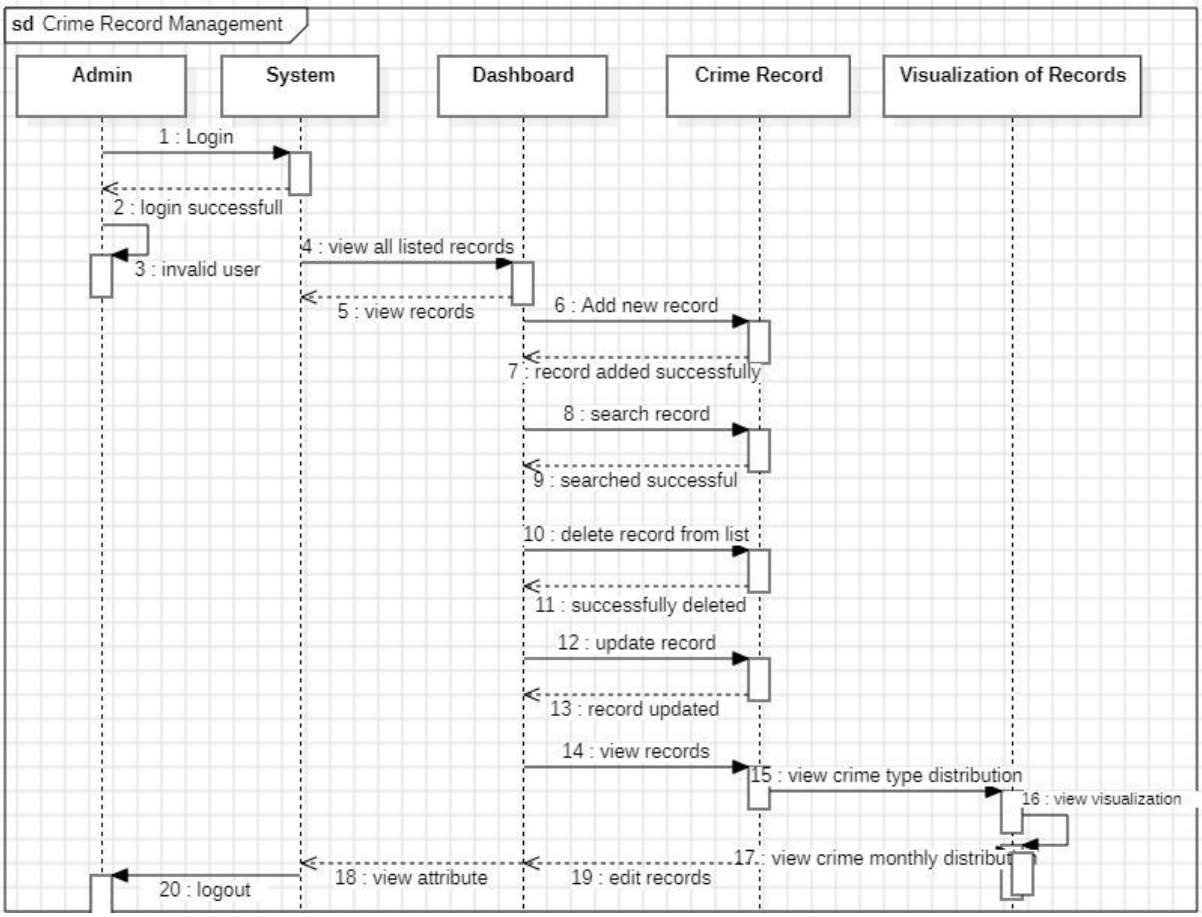
# USE CASE DIAGRAM



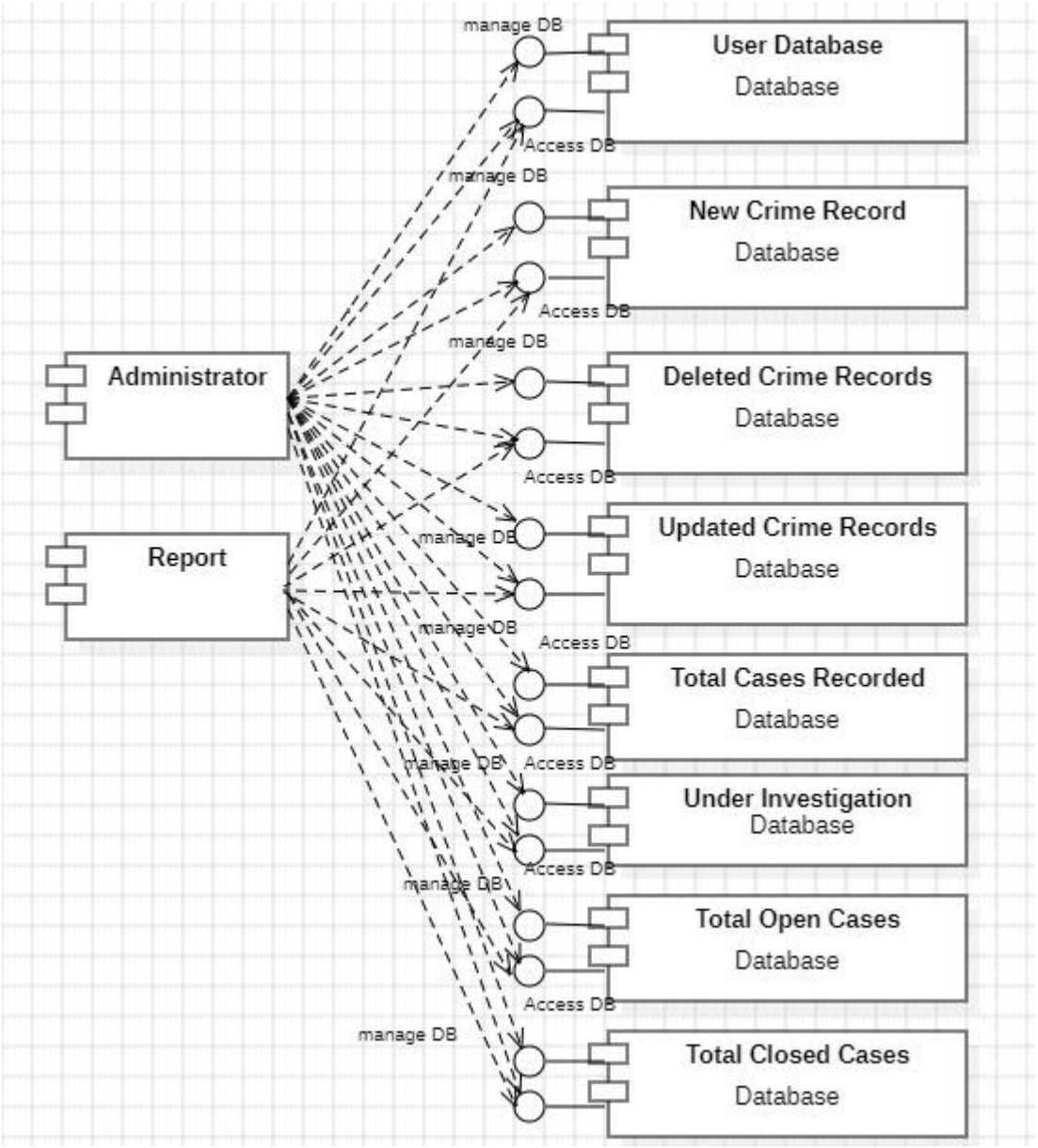
# ACTIVITY DIAGRAM



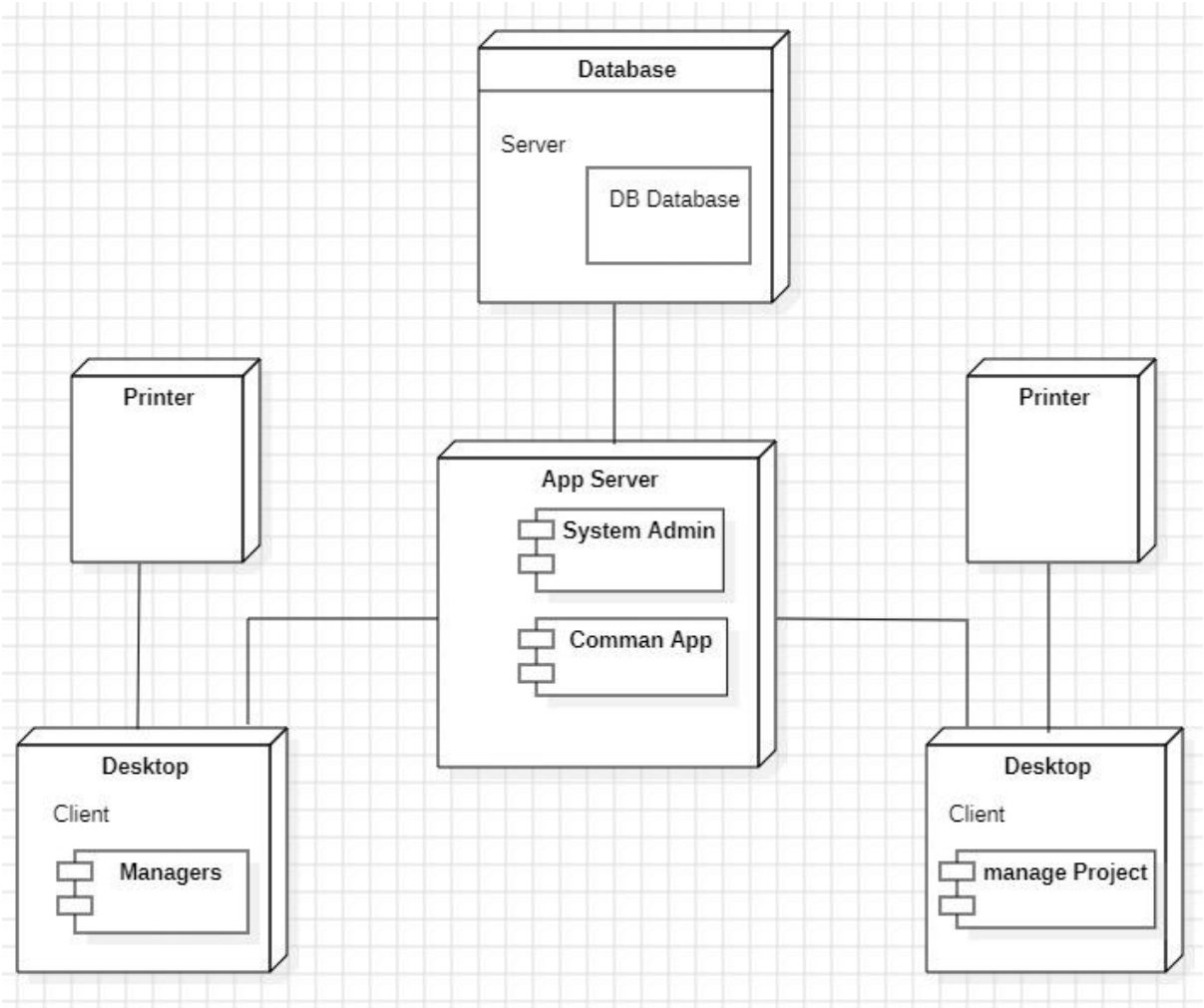
# SEQUENCE DIAGRAM



# COMPONENT DIAGRAM



# DEPLOYEMENT DIAGRAM



# DATA DICTIONARY

- **Users: -**
  - Stores login details of police/admin users.
  - Fields: user id, username, password, role, created at.
- **Criminals**: -
  - Stores details of criminals.
  - Fields: criminal id, name, gender, dob, address, photo.
- **Crimes**
  - Records crime information.
  - Fields: crime id, title, type, description, location, status.
- **Case Registration Detail: -**
  - Stores official First Information Reports.
  - Fields: FIR Number, FIR Date, Court Case Number.
- **Suspects: -**
  - Contains data on suspects related to crimes.
  - Fields: suspect name, Suspect Age, Supect_Gender, Suspect Contact, Suspect Occupation, Suspect Address
- Evidence: -
  - Keeps digital or physical evidence records.
  - Fields: - Evidence Type, Evidence Collection Date, Evidence Description, Evidence Location, Evidence Image.

# TEST PROCEDURE & IMPLEMENTATION

- Test Procedure: -
  Testing ensures the Crime Record Management System functions correctly and securely. The following procedures were used:

- **Unit Testing: -**
- Each module (e.g., login, FIR entry, crime record update) was tested separately.
- Focused on input validation, correct outputs, and error handling.

- **Integration Testing: -**
  - Verified that connected modules (e.g., FIR linked with crimes and users) worked together smoothly.
  - Ensured data flowed correctly between tables and screens.

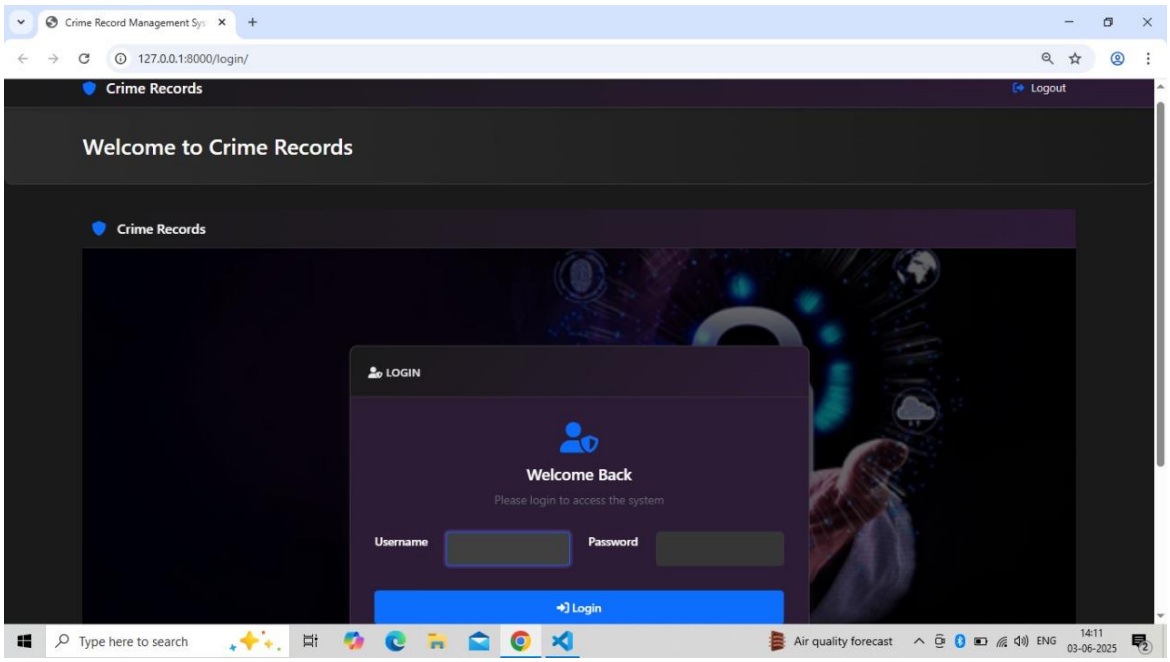
- **System Testing: -**
  - End-to-end testing of the entire application.
  - Tested with sample data to simulate real use cases.

- **User Acceptance Testing (UAT): -**
  - Tested by intended users (police/admin staff) to check ease of use and interface design.
  - Feedback used to refine features and fix issues.

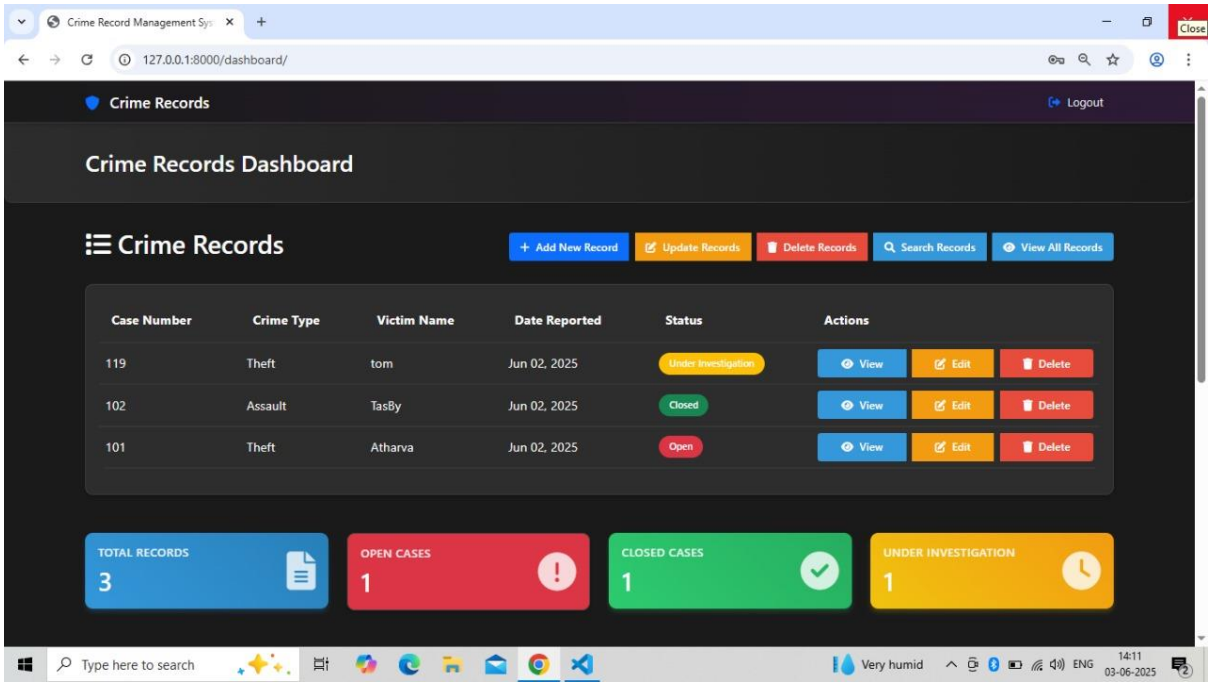
- **Security Testing: -**
  - Ensured secure login and role-based access control.
  - Tested for SQL injection and unauthorized data access.

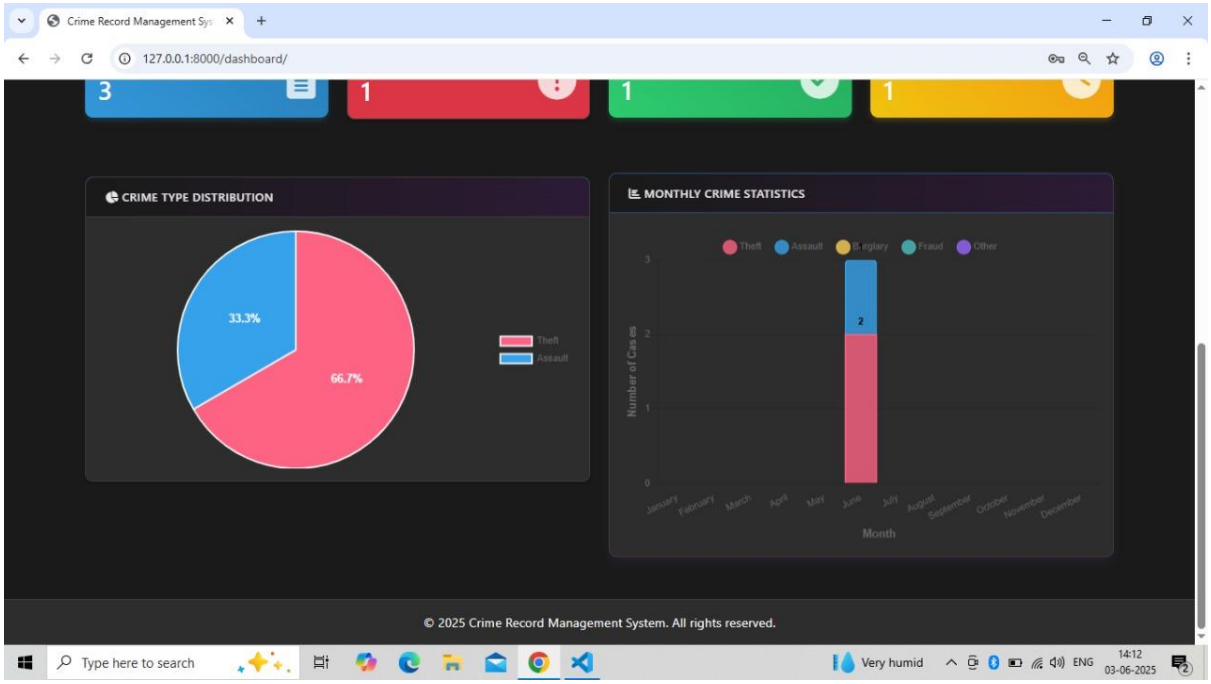
# SCREENSHORTS

- **Login Page: -**

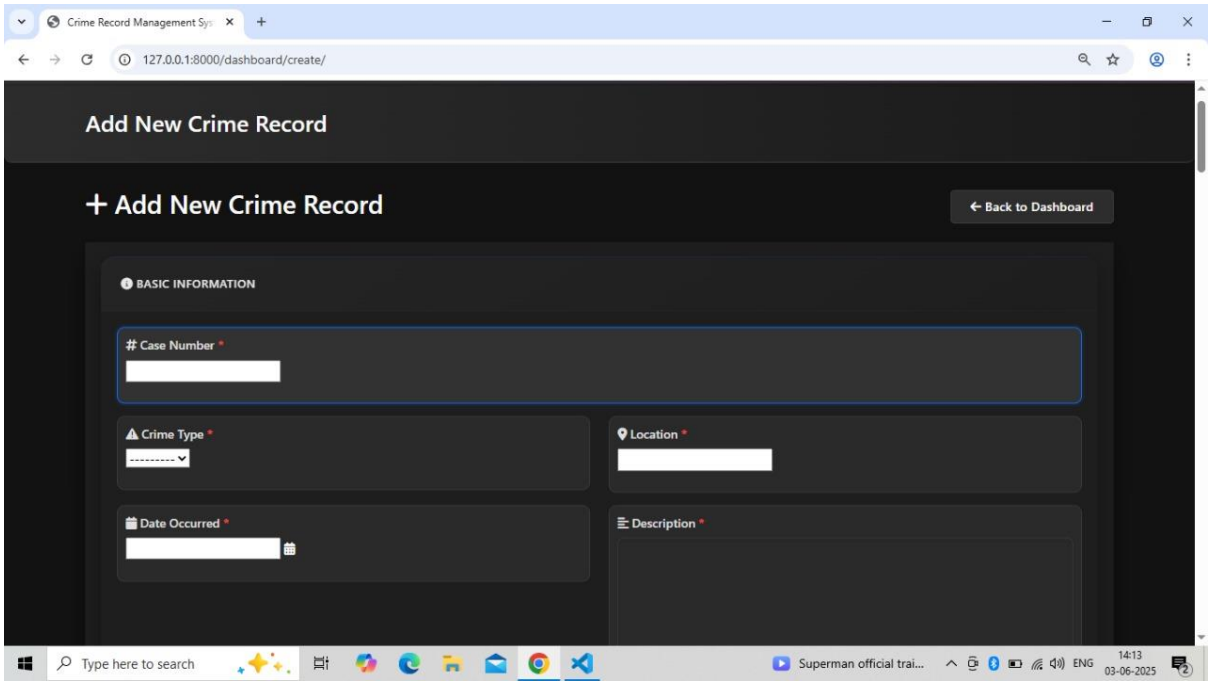


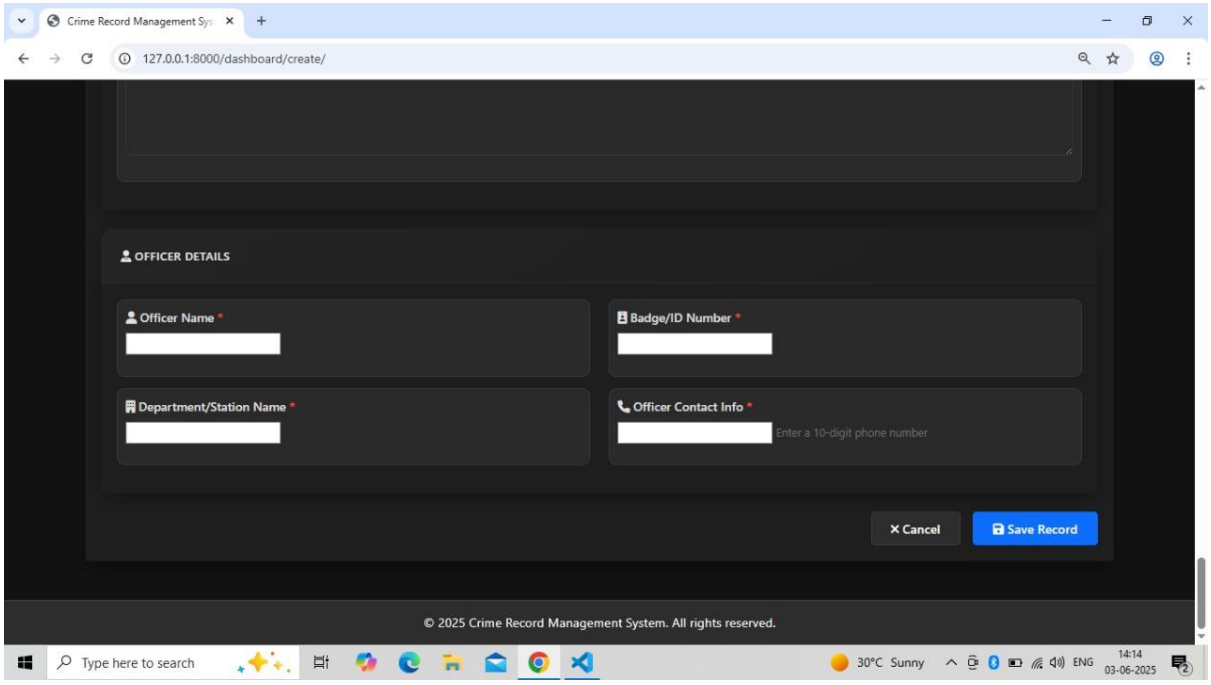
- **DashBoard Page: -**



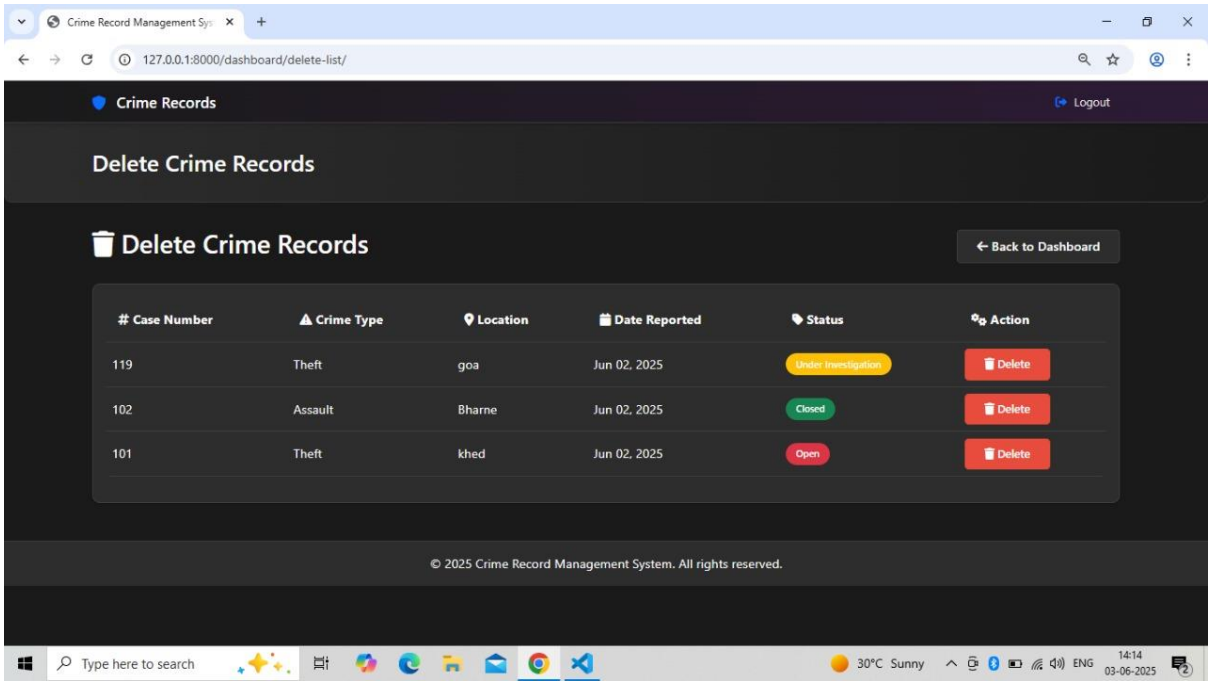



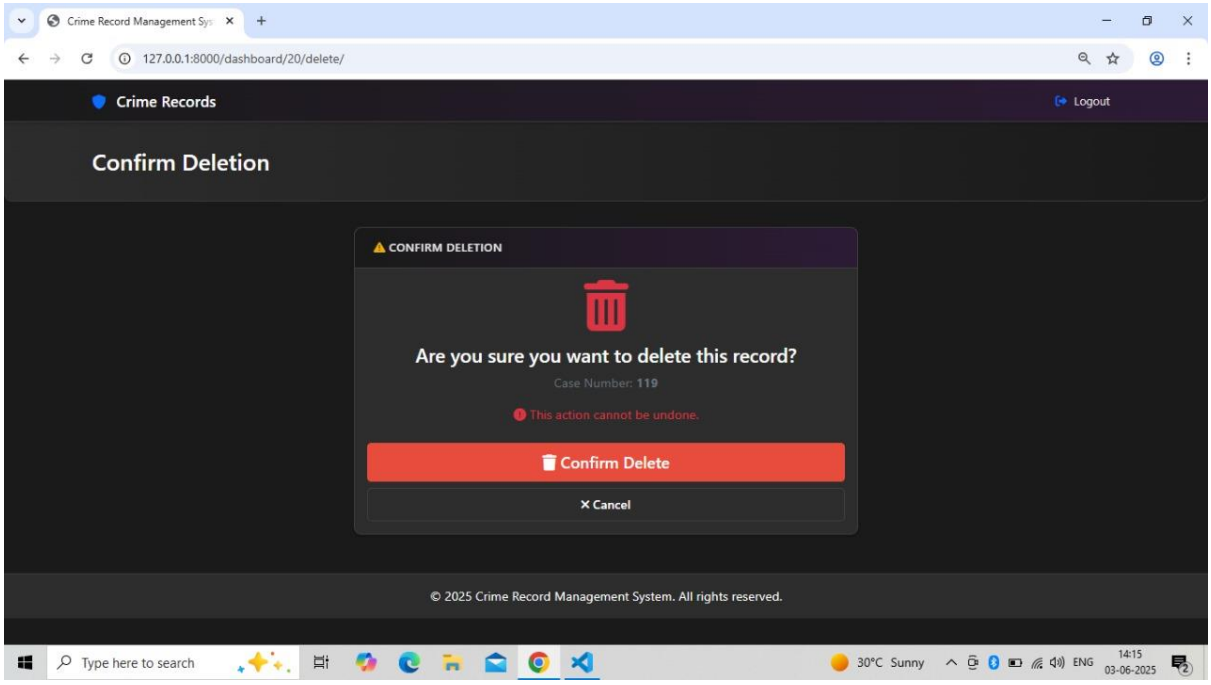
- **Add Crime : -**


Crime Record Management Sys
127.0.0.1:8000/dashboard/create/
Add New Crime Record
+ Add New Crime Record
← Back to Dashboard
BASIC INFORMATION
# Case Number *
Crime Type *
---------
Location *
Date Occurred *
Description *
Type here to search
Superman official trai...
ENG
14:13
03-06-2025

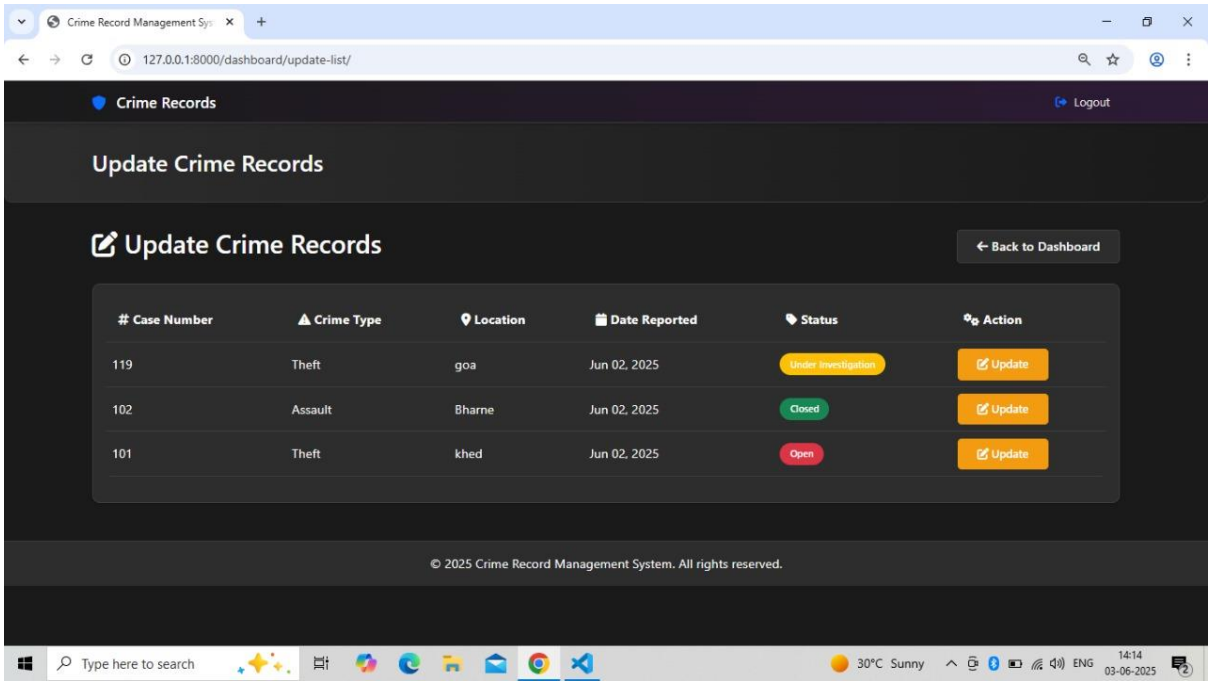

Crime Record Management Sys
127.0.0.1:8000/dashboard/create/
OFFICER DETAILS
Officer Name *
Badge/ID Number *
Department/Station Name *
Officer Contact Info *
Enter a 10-digit phone number
× Cancel
Save Record
© 2025 Crime Record Management System. All rights reserved.
Type here to search
30°C Sunny
ENG
14:14
03-06-2025

- **Delete Crime Record: -**

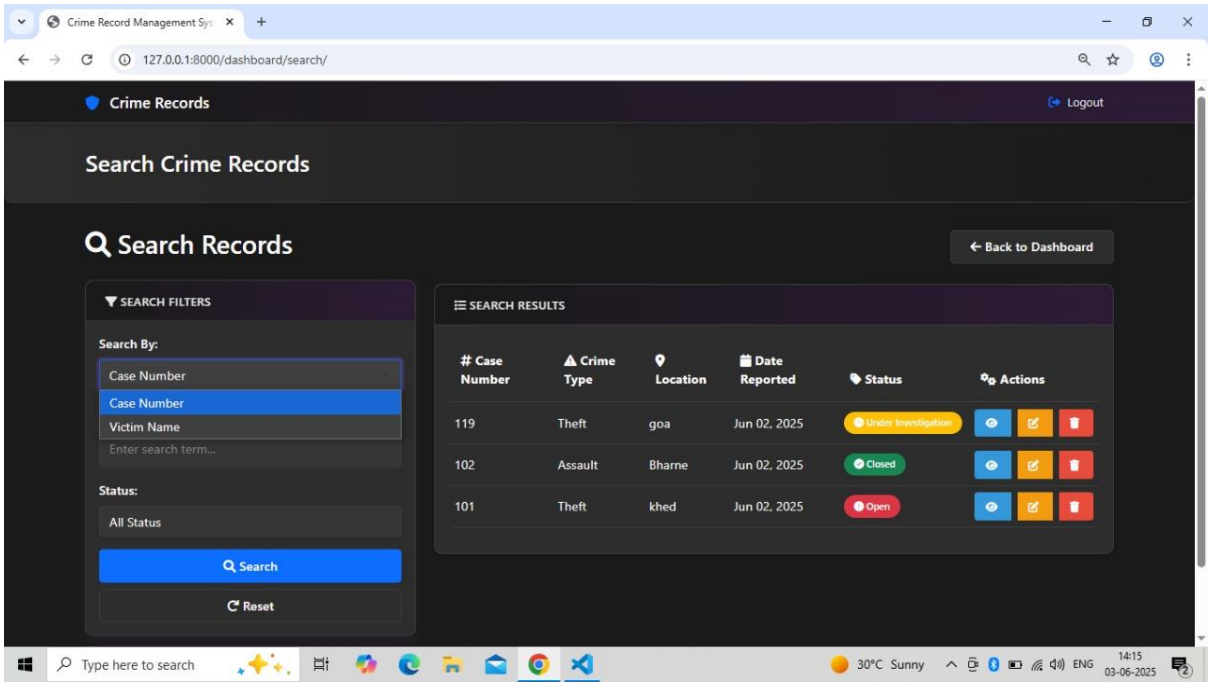


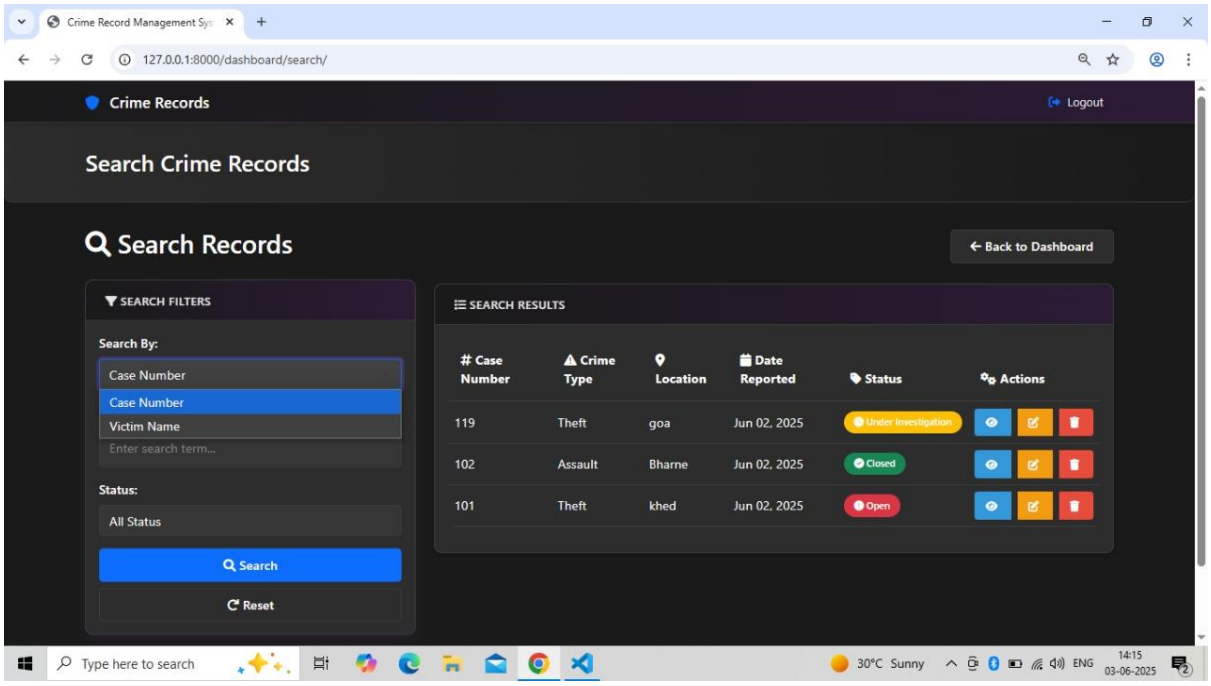
- **Update Crime Record: -**

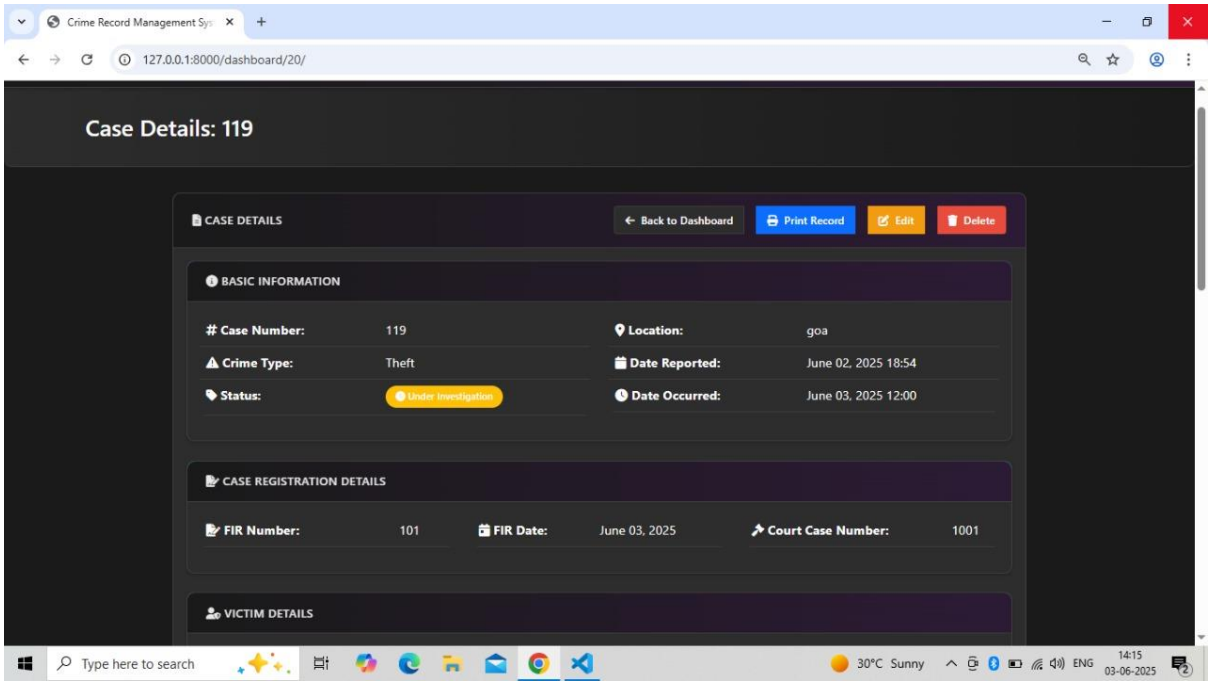



# Search Crime Record: -



# View Crime Record: -



# CODE

- **Model.py: -**

```
from django.db import models
from django.utils import timezone
from django.core.exceptions import ValidationError
from django.core.validators import RegexValidator


class CrimeRecord(models.Model):
    """
    This model represents a crime record in the system.
    It stores all information related to a crime case including victim details,
    suspect information, evidence, and investigating officer details.
    """

    # Predefined choices for crime types to maintain consistency
    CRIME_TYPES = [
        ('THEFT', 'Theft'),
        ('ASSAULT', 'Assault'),
        ('BURGLARY', 'Burglary'),
        ('FRAUD', 'Fraud'),
        ('OTHER', 'Other'),
    ]

    # Possible statuses for a crime case
    STATUS_CHOICES = [
```

```
        ('OPEN', 'Open'),                  # Case is newly filed
        ('UNDER_INVESTIGATION', 'Under Investigation'),  # Case is being investigated
        ('CLOSED', 'Closed'),              # Case has been resolved
    ]

    # Phone number validation - ensures exactly 10 digits
    phone_validator = RegexValidator(
        regex=r'^\d{10}$',
        message='Phone number must be exactly 10 digits.'
    )

    # ===== Basic Information Section =====
    # Unique identifier for the case
    case_number = models.CharField(max_length=50, unique=True, help_text="Unique case number for the crime record")
    # Type of crime committed
    crime_type = models.CharField(max_length=20, choices=CRIME_TYPES)
    # Detailed description of the crime
    description = models.TextField()
    # Where the crime occurred
    location = models.CharField(max_length=200)
    # When the crime was reported to authorities
    date_reported = models.DateTimeField(default=timezone.now)
    # When the crime actually occurred
    date_occurred = models.DateTimeField()
```

```
    # Current status of the case
    status = models.CharField(max_length=20,
choices=STATUS_CHOICES, default='OPEN')


    # ===== Case Registration Details =====
    # First Information Report number (FIR)
    fir_number = models.CharField(max_length=50, unique=True,
null=True, blank=True)
    # Date when FIR was filed
    fir_date = models.DateField(null=True, blank=True)
    # Court case number if case goes to trial
    court_case_number = models.CharField(max_length=50,
unique=True, null=True, blank=True)


    # ===== Victim Details Section =====
    # Personal information about the victim
    victim_name = models.CharField(max_length=100)
    victim_age = models.PositiveIntegerField(null=False, blank=False)
    victim_gender = models.CharField(max_length=10, choices=[
        ('MALE', 'Male'),
        ('FEMALE', 'Female'),
        ('OTHER', 'Other'),
    ], null=False, blank=False)
    # Contact information with 10-digit validation
    victim_contact = models.CharField(max_length=15, null=False,
blank=False, validators=[phone_validator])
    victim_address = models.TextField(null=False, blank=False)
```

```
    # Optional victim information
    victim_occupation = models.CharField(max_length=100,
blank=True)
    victim_marital_status = models.CharField(max_length=20,
choices=[
        ('SINGLE', 'Single'),
        ('MARRIED', 'Married'),
        ('DIVORCED', 'Divorced'),
        ('WIDOWED', 'Widowed'),
        ('OTHER', 'Other'),
    ], blank=True)

    # ===== Suspect Details Section =====
    # Required suspect information
    suspect_name = models.CharField(max_length=100, null=False,
blank=False)
    # Optional suspect information
    suspect_age = models.PositiveIntegerField(null=True, blank=True)
    suspect_gender = models.CharField(max_length=10, choices=[
        ('MALE', 'Male'),
        ('FEMALE', 'Female'),
        ('OTHER', 'Other'),
    ], blank=True)
    suspect_contact = models.CharField(max_length=15, blank=True)
    suspect_address = models.TextField(blank=True)
    suspect_occupation = models.CharField(max_length=100,
blank=True)
```

```
    # Current status of the suspect
    suspect_status = models.CharField(max_length=20, choices=[
        ('ARRESTED', 'Arrested'),
        ('AT_LARGE', 'At Large'),
        ('KNOWN', 'Known'),
        ('UNKNOWN', 'Unknown'),
    ], null=False, blank=False)
    # Photo of the suspect if available
    suspect_image = models.ImageField(upload_to='suspect_images/',
null=True, blank=True)


    # ===== Evidence Details Section =====
    # Type of evidence collected
    evidence_type = models.CharField(max_length=50, choices=[
        ('WEAPON', 'Weapon'),
        ('DOCUMENTS', 'Documents'),
        ('CCTV', 'CCTV Footage'),
        ('DNA', 'DNA Evidence'),
        ('FINGERPRINTS', 'Fingerprints'),
        ('PHYSICAL', 'Physical Evidence'),
        ('DIGITAL', 'Digital Evidence'),
        ('BIOLOGICAL', 'Biological Evidence'),
        ('TRACE', 'Trace Evidence'),
        ('OTHER', 'Other'),
    ], null=False, blank=False)
    # When the evidence was collected
```

```
    evidence_collection_date = models.DateField(null=True,
blank=True)
    # Detailed description of the evidence
    evidence_description = models.TextField(null=True, blank=True)
    # Where the evidence is stored
    evidence_location = models.CharField(max_length=200, null=False,
blank=False)
    # Chain of custody documentation
    evidence_chain_of_custody = models.TextField(blank=True)
    # Photo of the evidence if available
    evidence_image =
models.ImageField(upload_to='evidence_images/', null=True,
blank=True)

    # ===== Additional Information Section =====
    # General evidence notes
    evidence = models.TextField(blank=True)
    # Any additional notes about the case
    notes = models.TextField(blank=True)
    # Timestamps for record keeping
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    # ===== Officer Details Section =====
    # Information about the investigating officer
    officer_name = models.CharField(max_length=100, null=False,
blank=False)
```

```
    officer_badge = models.CharField(max_length=50, null=False, blank=False)
    officer_department = models.CharField(max_length=100, null=False, blank=False)
    # Contact information with 10-digit validation
    officer_contact = models.CharField(max_length=20, null=False, blank=False, validators=[phone_validator])


    def __str__(self):
        """
        Returns a string representation of the crime record.
        Format: "Case Number - Crime Type"
        """
        return f"{self.case_number} - {self.crime_type}"


    class Meta:
        """
        Meta class for CrimeRecord model.
        Orders records by date reported in descending order (newest first).
        """
        ordering = ['-date_reported']


    def clean(self):
        """
        Custom validation method to ensure case number uniqueness.
        Raises ValidationError if case number already exists.
```

```
        """
        super().clean()
        if
CrimeRecord.objects.filter(case_number=self.case_number).exclude(p
k=self.pk).exists():
            raise ValidationError({'case_number': 'This case number is
already in use.'})
```

- **View.py: -**

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView,
CreateView, UpdateView, DeleteView
from django.urls import reverse_lazy
from .models import CrimeRecord
from django.contrib import messages
from django.utils import timezone
from django.db.models import Count
from django.db.models.functions import TruncMonth
import json
from datetime import datetime, timedelta

# Create your views here.

class CrimeRecordListView(LoginRequiredMixin, ListView):
    model = CrimeRecord
    template_name = 'crime_records/crime_list.html'
    context_object_name = 'crime_records'
    ordering = ['-created_at']  # Order by most recent first
    paginate_by = 3  # Show only 3 records

    def get_queryset(self):
        queryset = super().get_queryset()
        search_query = self.request.GET.get('search_query')
        search_type = self.request.GET.get('search_type')

        if search_query and search_type:
            if search_type == 'case_number':
                queryset =
queryset.filter(case_number__icontains=search_query)
            elif search_type == 'victim_name':
                queryset =
queryset.filter(victim_name__icontains=search_query)
```

```
        # Always return the 3 most recent records
        return queryset[:3]

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        # Basic statistics
        context['total_records'] = CrimeRecord.objects.count()
        context['closed_cases'] =
CrimeRecord.objects.filter(status='CLOSED').count()
        context['under_investigation'] =
CrimeRecord.objects.filter(status='UNDER_INVESTIGATION').c
ount()
        context['open_cases'] =
CrimeRecord.objects.filter(status='OPEN').count()

        # Crime type distribution for pie chart
        crime_types =
CrimeRecord.objects.values('crime_type').annotate(count=Count('i
d'))
        context['crime_type_data'] = json.dumps([{
            'type':
dict(CrimeRecord.CRIME_TYPES)[item['crime_type']],
            'count': item['count']
        } for item in crime_types])

        # Monthly trend data for bar chart
        six_months_ago = timezone.now() - timedelta(days=180)
        monthly_trend = CrimeRecord.objects.filter(
            date_reported__gte=six_months_ago
        ).annotate(
            month=TruncMonth('date_reported')
        ).values('month').annotate(
            count=Count('id')
        ).order_by('month')
```

```
        # Get detailed records for each month
        monthly_records = {}
        for record in
CrimeRecord.objects.filter(date_reported__gte=six_months_ago):
            month_key = record.date_reported.strftime('%Y-%m')
            if month_key not in monthly_records:
                monthly_records[month_key] = []
            monthly_records[month_key].append({
                'crime_type':
dict(CrimeRecord.CRIME_TYPES)[record.crime_type],
                'location': record.location,
                'date_reported': record.date_reported.strftime('%B %d,
%Y'),
                'case_number': record.case_number,
                'status':
dict(CrimeRecord.STATUS_CHOICES)[record.status]
            })

        context['monthly_trend_data'] = json.dumps([{
            'month': item['month'].month,  # Just the month number
            'count': item['count'],
            'records': monthly_records.get(item['month'].strftime('%Y-
%m'), [])
        } for item in monthly_trend])

        # Location data for heat map
        locations =
CrimeRecord.objects.values('location').annotate(count=Count('id')
)
        context['location_data'] = json.dumps([{
            'location': item['location'],
            'count': item['count']
        } for item in locations])

        return context

class CrimeRecordDetailView(LoginRequiredMixin, DetailView):
```

```
    model = CrimeRecord
    template_name = 'crime_records/crime_detail.html'
    context_object_name = 'crime'

class CrimeRecordCreateView(LoginRequiredMixin, CreateView):
    model = CrimeRecord
    template_name = 'crime_records/crime_form.html'
    fields = [
        # Basic Information
        'case_number', 'crime_type', 'description', 'location',
        'date_occurred', 'status',

        # Case Registration Details
        'fir_number', 'fir_date', 'court_case_number',

        # Victim Details
        'victim_name', 'victim_age', 'victim_gender', 'victim_contact',
        'victim_address', 'victim_occupation', 'victim_marital_status',

        # Suspect Details
        'suspect_name', 'suspect_age', 'suspect_gender',
'suspect_contact',
        'suspect_address', 'suspect_occupation', 'suspect_status',
'suspect_image',

        # Evidence Details
        'evidence_type', 'evidence_collection_date',
'evidence_description',
        'evidence_location', 'evidence_chain_of_custody',
'evidence_image',

        # Officer Details
        'officer_name', 'officer_badge', 'officer_department',
'officer_contact',

        # Additional Information
        'evidence', 'notes'
```

```
    ]
    success_url = reverse_lazy('crime-list')

    def form_valid(self, form):
        # Set the date_reported to current time
        form.instance.date_reported = timezone.now()
        messages.success(self.request, 'Crime record created
successfully!')
        return super().form_valid(form)

    def form_invalid(self, form):
        messages.error(self.request, 'Please correct the errors below.')
        return super().form_invalid(form)

class CrimeRecordUpdateView(LoginRequiredMixin,
UpdateView):
    model = CrimeRecord
    template_name = 'crime_records/crime_form.html'
    fields = [
        # Basic Information
        'case_number', 'crime_type', 'description', 'location',
        'date_occurred', 'status',

        # Case Registration Details
        'fir_number', 'fir_date', 'court_case_number',

        # Victim Details
        'victim_name', 'victim_age', 'victim_gender', 'victim_contact',
        'victim_address', 'victim_occupation', 'victim_marital_status',

        # Suspect Details
        'suspect_name', 'suspect_age', 'suspect_gender',
'suspect_contact',
        'suspect_address', 'suspect_occupation', 'suspect_status',
'suspect_image',

        # Evidence Details
```

```
        'evidence_type', 'evidence_collection_date',
'evidence_description',
        'evidence_location', 'evidence_chain_of_custody',
'evidence_image',

        # Officer Details
        'officer_name', 'officer_badge', 'officer_department',
'officer_contact',

        # Additional Information
        'evidence', 'notes'
    ]
    success_url = reverse_lazy('crime-list')

    def form_valid(self, form):
        messages.success(self.request, 'Crime record updated
successfully!')
        return super().form_valid(form)

class CrimeRecordDeleteView(LoginRequiredMixin, DeleteView):
    model = CrimeRecord
    template_name = 'crime_records/crime_confirm_delete.html'
    success_url = reverse_lazy('crime-list')

    def delete(self, request, *args, **kwargs):
        messages.success(request, 'Crime record deleted successfully!')
        return super().delete(request, *args, **kwargs)

class CrimeRecordUpdateListView(LoginRequiredMixin,
ListView):
    model = CrimeRecord
    template_name = 'crime_records/crime_update_list.html'
    context_object_name = 'crimes'
    paginate_by = 10

class CrimeRecordDeleteListView(LoginRequiredMixin, ListView):
    model = CrimeRecord
```

```
    template_name = 'crime_records/crime_delete_list.html'
    context_object_name = 'crimes'
    paginate_by = 10

class CrimeRecordViewAllListView(LoginRequiredMixin,
ListView):
    model = CrimeRecord
    template_name = 'crime_records/crime_view_all.html'
    context_object_name = 'crimes'
    paginate_by = 10

class CrimeRecordSearchView(LoginRequiredMixin, ListView):
    model = CrimeRecord
    template_name = 'crime_records/crime_search.html'
    context_object_name = 'crimes'
    paginate_by = 10

    def get_queryset(self):
        queryset = super().get_queryset()
        search_query = self.request.GET.get('search_query')
        search_type = self.request.GET.get('search_type')
        status = self.request.GET.get('status')

        if search_query and search_type:
            if search_type == 'case_number':
                queryset =
queryset.filter(case_number__icontains=search_query)
            elif search_type == 'victim_name':
                queryset =
queryset.filter(victim_name__icontains=search_query)

        if status:
            queryset = queryset.filter(status=status)

        return queryset
```

- **URls.py: -**

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.CrimeRecordListView.as_view(), name='crime-
list'),
    path('search/', views.CrimeRecordSearchView.as_view(),
name='crime-search'),
    path('create/', views.CrimeRecordCreateView.as_view(),
name='crime-create'),
    path('<int:pk>/', views.CrimeRecordDetailView.as_view(),
name='crime-detail'),
    path('<int:pk>/update/',
views.CrimeRecordUpdateView.as_view(), name='crime-update'),
    path('<int:pk>/delete/', views.CrimeRecordDeleteView.as_view(),
name='crime-delete'),
    path('update-list/',
views.CrimeRecordUpdateListView.as_view(), name='crime-
update-list'),
    path('delete-list/', views.CrimeRecordDeleteListView.as_view(),
name='crime-delete-list'),
    path('view-all/', views.CrimeRecordViewAllListView.as_view(),
name='crime-view-all'),
]
```

# REPORTS

In the **Crime Record Management System (CRMS)**, "reports" typically refer to the **automatically or manually generated summaries** and data outputs that help the admin analyze, present, or archive crime-related information.

### ❖ Crime Summary Report: -

- Lists total number of crimes recorded in the system
- Breakdown by category (e.g., theft, assault, cybercrime).
- Includes counts per city/area/station if applicable

### ❖ Crime by Type Report: -

- Filters all crimes by selected crime type (e.g., robbery).
- Shows date, location, suspect/victim details, status, etc.
- Useful for pattern analysis and resource allocation**.**

### ❖ Criminal Record Report: -

- Lists all crimes associated with a specific criminal.
- Includes personal data, arrest details, conviction status, etc.
- Helps in tracking repeat offenders.

### ❖ Area-Wise Crime Report: -

- Breakdown of crime occurrences by region or locality.
- Useful for identifying high-risk or crime-prone zones.

### ❖ Crime by Date Range Report: -

- Displays crimes committed between selected start and end dates.
- Useful for trend analysis and seasonal patterns.

### ❖ Officer-wise Report (if expanded to multi-role in future): -

- Number of cases handled by each officer/admin.
- Can show performance, resolution rate, etc.

# IMPLIMENTATION DETAILS

The Crime Record Management System is a web-based application built using Django as the backend framework, MySQL as the database, and HTML, CSS, and JavaScript for the frontend. It is designed for admin-only access, where an authorized user can log in, manage crime records, search, filter, and generate reports.

### ❖ Hardware Requirements: -

- Minimum: Dual-core processor, 4GB RAM, 250GB HDD.
- Standard USB keyboard and mouse
- Internet connection (Wi-Fi or Ethernet) for local or remote server access.
- Recommended: Intel i5/i7, 8GB RAM, 500GB SSD.

### ❖ Software Requirements: -

- Operating System: Windows 10, Windows 11
- Backend: Django framework (Python 3.9+)
- Database: MySQL 5.7 or higher.
- Frontend: HTML, CSS, JavaScript.
- Web Server: Django development server (during dev)
- Python Libraries: Django, MySQL Client, crispy-forms, etc.
- IDE/Text Editor: Visual Studio Code.

### ❖ Frontend Development: -

- Built with HTML and styled using CSS.
- JavaScript used for interactivity (e.g., form validation, filters).
- Admin-only user interface.
- Pages: Login, Dashboard, Add Crime, View Crimes, Search/Filter, Reports.

❖ Development Tools: -

- Visual Studio Code or PyCharm for coding.
- MySQL Workbench or phpMyAdmin for database management.
- Command line/Terminal for running Django commands.

❖ Browser Compatibility: -

- Fully tested on modern browsers including:
  - Google Chrome – Fully supported.
  - Mozilla Firefox – Fully supported.
  - Microsoft Edge – Fully supported.

# TESTING

❖ **Test Plan: -**

The testing plan ensures that all modules of the Crime Record Management System function as expected. It includes:

- Verifying admin login.
- Validating crime data input.
- Ensuring search, view, update, delete functionality works.
- Checking for UI consistency and security features.

❖ Black Box Testing (Data Validation Test Cases): -

| **Test Case ID** | Description | Input | Expected Output | Status |
|---|---|---|---|---|
| BB-01 | Login with valid credentials | Correct admin username & password | Admin dashboard loads successfully | Pass |
| BB-02 | Login with invalid credentials | Incorrect password | Error message: "Invalid credentials" | Pass |
| BB-03 | Add crime record with valid data | Complete crime form details | Crime record saved and confirmation shown | Pass |
| BB-04 | Search crime records by location | Missing required form fields | Validation error displayed for missing data | Pass |
| BB-05 | Delete crime record | Location = "New York" | Crime records filtered and displayed | Pass |
| BB-06 | Delete crime record | Select record and delete | Record removed from the database | Pass |

❖ Whit Box Testing **(**Functional Validation Test Cases): -

| **Test Case ID** | Description | Input | Expected Output | Status |
|---|---|---|---|---|
| WB-01 | Function: add_crime() | Valid crime data | New crime data inserted into database | Pass |
| WB-02 | Function: search_crime() | Search parameter (location) | Filtered list of crime records returned | Pass |
| WB-03 | Function: delete_crime() | Crime record ID | Crime record deleted from database | Pass |
| WB-04 | Function: update_crime() | Updated crime data | Crime record updated successfully | Pass |
| WB-05 | Input validation logic | Invalid/malicious inputs | Inputs rejected, safe data stored | Pass |

❖ **Test result Summary: -**

- Security Tests: Passed input validation and access control
- Total Test Cases: 11
- Passed: 11
- Failed: 0
- Functionality: All core functions (add, update, delete, search) working correctly
- User Interface: Forms and pages load without error
- Conclusion: System is stable, secure, and meets all functional requirements.

# FUTURE APPLICATION OF THE PROJECT

❖ **Integration with National Crime Databases: -**

- The system can be integrated with national and international crime databases (such as the NCRB or INTERPOL) to enable seamless data sharing, improve coordination between agencies, and track cross-border criminal activities.

❖ **Artificial Intelligence for Crime Prediction: -**

- Incorporating AI and machine learning can help analyze historical crime data to predict potential crime hotspots, assisting in proactive patrolling and preventive policing.

❖ **Biometric Integration: -**

- The system can include biometric features such as fingerprint, face recognition, or iris scanning to identify suspects and verify individuals during arrests or investigations.

❖ **Public Portal for FIR Filing and Updates: -**

- A user-friendly public interface could be introduced for citizens to file complaints, track FIR status, and communicate with police departments without visiting the station physically.

❖ **Cloud-Based Deployment: -**

- Moving the system to a secure cloud infrastructure can offer better scalability, availability, and disaster recovery capabilities.

❖ **Mobile Application for On-Field Officers: -**

- A mobile version of the CRMS can be developed to allow police officers to access or update case details in real-time while in the field, improving responsiveness and efficiency.

# BIBLIOGRAPHY


- **Django Documentation** – Official guide for Django web framework.
  https://docs.djangoproject.com
- **MySQL Developer Guide** – Reference for relational database integration and SQL commands.
  https://dev.mysql.com/doc/
- **Python Official Documentation** – Detailed explanation of Python libraries, syntax, and core modules.
  https://docs.python.org
- **W3Schools Django Tutorial** – Beginner-friendly resources on Django models, views, and templates.
  https://www.w3schools.com/django/
- **MDN Web Docs** – Front-end technologies reference (HTML, CSS, JavaScript).
  https://developer.mozilla.org/
- **GeeksforGeeks** – Tutorials and explanations for Django, Python, and database design.
  https://www.geeksforgeeks.org
- **Stack Overflow** – Community-driven solutions for debugging and code optimization.
  https://stackoverflow.com
- **Real Python** – Practical tutorials and advanced Python/Django articles.
  https://realpython.com

# CONCLUSION

The Crime Record Management System provides a centralized and efficient platform for managing crime-related data. Built using Django, MySQL, and web technologies, it enables administrators to securely add, update, delete, and search crime records.

While the current system focuses on admin-level functionality, it successfully addresses core requirements such as data organization, accessibility, and basic validation.

Though limited in scalability and multi-user support, the CRMS lays a strong foundation for digitizing and streamlining crime record management in small to medium law enforcement environments. Future improvements can expand its capabilities to support more roles, analytics, and security features.

The Crime Record Management System is a web-based solution developed using Django, MySQL, HTML, CSS, and JavaScript, with a focus on providing a structured and reliable way for administrators to manage crime data efficiently. The system is designed to store, retrieve, and manage crime records with ease, offering core functionalities such as adding new records, updating details, deleting old entries, and searching based on filters like location or crime type.