

CASE STUDY ON

CRIME ANALYSIS AND REPORTING SYSTEM

Team Members:

Pooja Shree M

Shakthi Priyanka S

Swathi Baskaran

Abstract

C.A.R.S. (Crime Analysis and Reporting System) is a Python-based digital platform designed to streamline the management, analysis, and reporting of criminal activities for law enforcement agencies. The system modernizes traditional, paper-based workflows by providing a centralized, secure, and scalable solution to handle incidents, victims, suspects, cases, and evidence. It enables real-time data access, role-based control, and activity tracking to ensure data integrity and operational accountability.

Built with a modular, layered architecture, C.A.R.S. supports role-specific functionalities for both officers and administrators, including incident registration, suspect/victim profiling, case creation, and evidence management. Advanced features like multi-factor authentication, GIS crime mapping, AI-driven analytics, and automated backups prepare the system for future-ready policing.

By enabling predictive policing and data-driven decision-making, C.A.R.S. serves as a transformative tool that strengthens public safety, improves inter-departmental collaboration, and enhances the overall efficiency of law enforcement operations.

Introduction

Crime analysis and reporting are crucial for maintaining public safety and ensuring efficient law enforcement operations. Traditional methods often rely heavily on manual paperwork, which is time-consuming and prone to errors. In an effort to modernize and digitize crime data management, we have developed C.A.R.S. (Crime Analysis and Reporting System), a robust Python-based application designed to help police officers and administrators record, manage, and analyze crime-related data in a secure and organized manner.

The C.A.R.S. system addresses common challenges faced by law enforcement agencies, such as delayed data access, inefficient report generation, and lack of centralized crime records. By shifting from paper-based documentation to a digital platform, the system enhances real-time information sharing, data consistency, and operational coordination across departments. This leads to quicker response times and more accurate investigations.

Incorporating modern technologies like multi-factor authentication, GIS mapping, AI-driven analytics, and automated backups, C.A.R.S. not only ensures data security and integrity but also enables predictive policing capabilities. It provides officers with powerful tools to visualize crime hotspots, detect patterns, and make informed decisions. Designed with scalability and user-friendliness in mind, the system is adaptable to the evolving needs of modern law enforcement, making it a vital asset in the digital transformation of public safety services.

Objective

The main goal of **C.A.R.S. (Crime Analysis and Reporting System)** is to build a centralized and intelligent system that streamlines the registration and tracking of criminal incidents. It facilitates comprehensive management of key components such as suspects, victims, evidence, and ongoing or closed cases. By organizing this information within a structured digital framework, C.A.R.S. ensures that law enforcement personnel can easily access accurate data for analysis and reporting. Additionally, it incorporates strict role-based access control to safeguard sensitive information, ensuring that only authorized users can view or modify specific records, thereby maintaining both security and accountability.

C.A.R.S. is designed to improve operational efficiency by reducing manual workloads and minimizing errors commonly associated with paper-based systems. The intuitive interface allows users to quickly input, retrieve, and update information, which not only saves time but also promotes better coordination across departments. Whether used by patrol officers, detectives, or administrative staff, the system fosters transparency and consistency throughout every stage of case management.

Moreover, the system's integrated analytical tools and visual dashboards empower agencies to identify crime trends and generate insightful reports for strategic planning. This enables law enforcement leaders to make informed decisions, allocate resources effectively, and anticipate potential threats. As an all-in-one platform tailored to the unique demands of modern policing, C.A.R.S. stands out as a transformative solution for strengthening public safety infrastructure.

Business Logic

The business logic of C.A.R.S. is encapsulated in a service-oriented structure, specifically in the CrimeAnalysisServiceImpl class, and covers the following core functionalities:

1. Role-Based Access Control

- Users register with a role-specific code and log in to access features tied to their role (admin or officer).
- Admins have full CRUD access; Officers have restricted access for reporting and viewing.

2. Incident Management

- createIncident(): Inserts a new incident with associated details into the database.
- viewIncidentsByDateRange(start_date, end_date): Fetches incidents within a time frame.
- searchIncidentsByType(type): Retrieves incidents by specific crime type.
- generateIncidentReport(incident_id): Gathers all linked records (victims, suspects, evidence) and formats them into a printable report.

3. Victim and Suspect Handling

- addSuspect(suspect_data): Admin-only feature to register suspects, with optional linkage to prior criminal records.
- viewVictimsByIncidentID(incident_id),
viewSuspectsByIncidentID(incident_id): List all victims or suspects connected to a given incident.

4. Case Management

- `createCase(case_description)`: Admin creates a case to group multiple incidents.
- `linkIncidentToCase(incident_id, case_id)`: Connects incidents with cases (many-to-many).
- `viewCaseDetails(case_id)`: Lists all incidents, victims, suspects, and evidence related to a case.

5. Evidence Management

- `viewEvidenceByIncidentID(incident_id)`: Returns a list of all evidence records for a given incident.

6. Criminal Record Handling

- `addCriminalRecord(suspect_id, crime_details, punishment_details)`: Links a suspect to a new criminal record entry.
- `viewSuspectCriminalHistory(suspect_id)`: Returns the history of crimes and punishments for a suspect.

7. Analytics & Reports (Admin Only)

- `getOfficerPerformanceMetrics()`: Evaluates officer activity based on cases handled, reports generated, etc.
- `generateCrimePatternAnalysis()`: Leverages available data to identify recurring trends.
- `finalizeReportsAndUpdateCase(case_id)`: Updates the status of cases and confirms report closure.

8. Activity Logging

- `logUserAction(user_id, action_description)`: Every major action is logged with a timestamp for audit trails.

9. Public Access (Optional Read-Only Menu)

- Allows civilians or general users to view limited incident data (e.g., by city or date range) without login.

System Architecture and Design

Technology Stack:

1. **Programming Language:** Python
2. **Database:** MySQL (for persistent data storage)
3. **Modules & Libraries:**
 4. **tabulate:** For displaying tabular data in a readable format.
 5. **dotenv:** For securely loading environment variables.
 6. **mysql.connector:** For database interactions.
 7. **Custom modules:** Entity classes (Incident, Case, Suspect), service layer, utility classes for DB connections and logging.

Modular Design:

The system follows a layered architecture to ensure scalability and maintainability:

1. **Presentation Layer:** Command-line interface for user interaction.
2. **Service Layer:** Business logic encapsulated within the CrimeServiceImpl.
3. **Data Access Layer:** Utility functions to handle database operations.
4. **Entity Layer:** Represents real-world objects like Incidents, Cases, Suspects, etc.

User Roles:

The application distinguishes between two primary roles:

1. Admin

1. Full control over incidents, suspects, and cases.
2. Can view and update all data, including activity logs.
3. Responsible for high-level system oversight.

2. Officer

1. Limited to reporting and viewing incidents and related data.
2. Cannot modify case details or manage users.

3.Authentication & Registration

1. Secure Registration: Users must enter a role-specific secret code (stored as environment variables) to register.
2. Login System: Validates user credentials against the database. Upon successful login, the system provides access based on the user's role.

Key Functionalities:

Incident Management

- Create Incident: Officers and admins can record new incidents, including type, date, area, city, description, status, and the officer in charge.
- View Incidents by Date Range: Retrieve incidents reported within a specified time frame, useful for trend analysis and case tracking.
- Search Incidents by Type: Allows targeted queries, such as "Robbery" or "Fraud", to analyze specific types of crime.
- Generate Incident Report: Produces a structured summary of an incident, including all crucial details.

Victim & Suspect Management

- View Victims by Incident ID: Displays a list of all victims associated with a specific incident.
- View Suspects by Incident ID: Shows suspects linked to a particular incident, along with their criminal IDs if available.
- Add Suspect (Admin only): Capture detailed suspect information such as personal details, address, contact info, and potential criminal records.

Evidence Management

- View Evidence by Incident ID: Displays collected evidence items related to a specific incident.

Case Management (Admin only)

- Create Case: Group multiple incidents under one case, providing a broader investigative perspective.

- View Case Details: Retrieve detailed descriptions of a case and its linked incidents.
- Update Case: Modify the case description to reflect new information.
- View All Cases: Comprehensive list of all cases for oversight and tracking.

Suspect-Criminal Record Analysis

- Retrieve combined suspect and criminal record data for deeper investigative insights (e.g., history of crimes, punishments).

Activity Logging (Admin only)

- Track user actions to ensure transparency and accountability. Every action (e.g., creating incidents, updating cases) is logged with a timestamp.

Database Schema Overview

Key tables include:

- Users: Stores credentials and role information.
- Incidents: Contains all details about criminal incidents.
- Victims: Holds victim-related data linked to incidents.
- Suspects: Stores suspect information, potentially linked to criminal records.
- Evidence: Documents all evidence items.
- Cases: Represents cases grouping multiple incidents.
- UserActivityLog: Logs all user activities for audit purposes.

Strengths of C.A.R.S.

- Role-based Access Control: Ensures that sensitive operations are restricted to authorized users only.
- Comprehensive Logging: Maintains a detailed activity log for traceability.
- Modular Codebase: Facilitates maintenance and future scalability.
- Detailed Data Handling: Supports deep analysis via suspect-criminal records and incident linking.

Areas for Improvement:

- Password Security: Current implementation stores passwords as plain text; implementing hashing (e.g., bcrypt) is recommended.

- Enhanced Error Handling: More robust checks (e.g., for invalid inputs, SQL injection) would improve stability.
- UI Enhancement: Migrating from CLI to a web-based interface (e.g., using Flask or Django) could improve usability.
- Reporting & Visualization: Integration of data visualization libraries (e.g., matplotlib, plotly) to generate charts and heatmaps.
- Notification System: Automatic email or SMS alerts for important updates.

Database Schema Design:

Tables

1 Users

- UserID (Primary Key): Unique identifier.
- Username: Officer/admin login name.
- Password: (currently plain text; should be hashed).
- Role: 'admin' or 'officer'.

2 Incidents

- IncidentID (Primary Key).
- IncidentType: Type of crime (e.g., robbery, assault).
- IncidentDate: Date it happened.
- Area: Area name.
- City: City where incident occurred.
- Description: Narrative details.
- Status: Open, closed, under investigation, etc.
- OfficerID (Foreign Key to Users.UserID): Who reported or is assigned.

3 Victims

- VictimID (Primary Key).
- FirstName, LastName.
- DOB: Date of birth.
- Gender.
- Address, ContactNumber.

- IncidentID (Foreign Key to Incidents.IncidentID).

4 Suspects

- SuspectID (Primary Key).
- FirstName, LastName.
- DOB, Gender.
- ResidentialAddress, ContactNumber, AadhaarNumber.
- IncidentID (Foreign Key to Incidents.IncidentID).
- CriminalID (Nullable): If they have a linked criminal record.

5 Evidence

- EvidenceID (Primary Key).
- EvidenceType: Type (e.g., fingerprint, CCTV footage).
- Description.
- IncidentID (Foreign Key to Incidents.IncidentID).

6 Criminals

- CriminalID (Primary Key).
- CrimeDetails.
- PunishmentDetails.

7 Cases

- CaseID (Primary Key).
- CaseDescription.

8 Incident_Case_Link

- CaseID (Foreign Key to Cases.CaseID).
- IncidentID (Foreign Key to Incidents.IncidentID).
- (Composite Primary Key on both columns).
- Purpose: To handle many-to-many between incidents and cases.

9 UserActivityLog

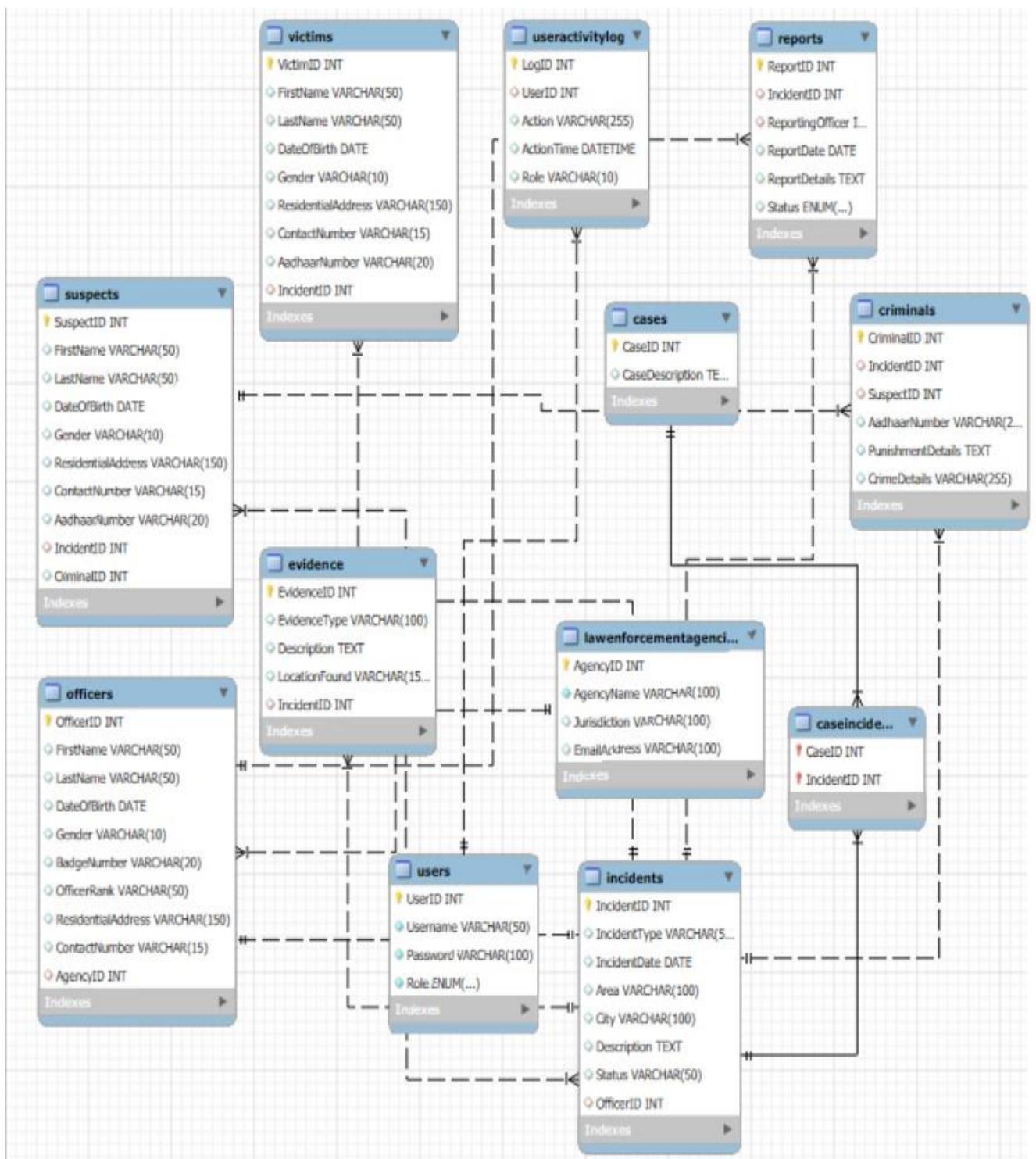
- LogID (Primary Key).
- UserID (Foreign Key to Users.UserID).

- Action: Description of activity.
- Action Time: Timestamp.

Relationships Summary

- Users → Incidents: One-to-many (an officer can handle multiple incidents).
- Incidents → Victims: One-to-many.
- Incidents → Suspects: One-to-many.
- Incidents → Evidence: One-to-many.
- Cases ↔ Incidents: Many-to-many through Incident_Case_Link.
- Suspects → Criminals: Optional one-to-one (if suspect has a criminal record).
- Users → UserActivityLog: One-to-many.

ER Diagram:



Console Output:

1. Public Menu: Viewing recent incidents

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Victus Laptop\Downloads\Hexaware\Python Training\Case Study\CARS> python main.py
== Crime Reporting System ==

== Public Menu ==
1. View recent incidents
2. View incidents by date range
3. View incidents by area/city
4. Login as officer/admin
0. Exit
Enter your choice: 1

RECENT INCIDENTS REPORT (LAST 30 DAYS)
=====
DATE | TYPE | AREA | CITY | DESCRIPTION
-----
2025-06-28 | Fraud | T Nagar, | Chennai | Fake lottery scam targeting seniors
2025-06-25 | Theft | Banjara Hills, | Hyderabad | Mobile phone theft in mall
2025-06-18 | Domestic Violen | Whitefield, | Bengaluru | Wife assaulted by husband
2025-06-12 | Cyber Crime | Gurgaon, | Delhi-NCR | Online job scam defrauding 50 people
2025-06-05 | Robbery | Marine Lines, | Mumbai | Chain snatching incident near station
```

2. Public Menu: View incidents by date range

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Enter your choice: 2
Start date (YYYY-MM-DD): 2023-01-31
End date (YYYY-MM-DD): 2023-03-15

Incidents that happened between '2023-01-31' and '2023-03-15':
-----
Incident Type: Burglary
Incident Date: 2023-02-03
Area: Saket
City: Delhi
Description: Residential burglary during daytime
-----
Incident Type: Cyber Crime
Incident Date: 2023-02-10
Area: Koramangala
City: Bengaluru
Description: Online financial fraud
-----
Incident Type: Assault
Incident Date: 2023-02-18
Area: Banjara Hills
City: Hyderabad
Description: Bar fight leading to serious injuries
-----
```

3. Public Menu: View incidents by area/city

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

3. View incidents by area/city
4. Login as officer/admin
0. Exit
Enter your choice: 3

Leave the field blank if you don't want that filter
Enter area:
Enter city: Mumbai
Incidents that happened in Mumbai:
-----
Incident Type: Robbery
Incident Date: 2025-06-05
City: Mumbai
Description: Chain snatching incident near station
-----
```

4. Public Menu: Login as officer

```
==== Public Menu ====
1. View recent incidents
2. View incidents by date range
3. View incidents by area/city
4. Login as officer/admin
0. Exit
Enter your choice: 4

==== Login ====
Username: rajesh.kumar
Password:
Welcome, rajesh.kumar!

==== Officer Menu ===

-- Incident Management --
1. View all incident details with assignments
2. View my assigned incidents
3. Filter incidents
```

5. Officer Menu: View all incident details with assignments

```
0. Create/Update Case Report
9. View case report
10. Logout
Enter choice: 1

==== All Incidents ===

Incident 1:
ID      : 1
Type    : Robbery
Date    : 2023-01-15
Area    : Andheri West
City    : Mumbai
Description : Armed robbery at jewelry store
Status   : Closed
Officer ID : 1
-----

Incident 2:
ID      : 2
Type    : Burglary
Date    : 2023-02-03
Area    : Saket
City    : Delhi
Description : Residential burglary during daytime
Status   : Under Investigation
Officer ID : 2
-----
```

6. Officer Menu: View my assigned incidents

```
-- Case Management --
7. View case details (victims/suspects/evidence)
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 2
===== My Assignments =====

Incident 1:
ID : 26
Type : Robbery
Date : 2025-06-05
Area : Marine Lines
City : Mumbai
Description : Chain snatching incident near station
Status : Under Investigation
Officer ID : 1

==== Officer Menu ====
```

7. Officer Menu: Filter Incidents

```
10. Logout
Enter choice: 3

===== Filtering the Incidents =====

Fill the parameters on whose basis you wish to filter and leave the rest blank

Date Filters:
Enter the start date (YYYY-MM-DD): 2023-02-28
Enter the end date (YYYY-MM-DD):
Enter the area:
Enter the city: Chennai
Enter the type of incident: Theft
Enter the Officer ID:
===== Filtered Results =====

Incident 1:
ID : 5
Type : Theft
Date : 2023-03-05
Area : T Nagar
City : Chennai
Description : Pickpocketing in crowded market
Status : Under Investigation
Officer ID : 5
```

8. Officer Menu: Access suspects database

```
-- Case Management --
7. View case details (victims/suspects/evidence)
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 4
===== Suspects Database =====
Suspect ID          : 1032
Name                : John Doe
Date of Birth       : 1990-01-01
Aadhaar Number     : 123456789012
Address             : Some Street
Contact Number      : 9876543210
Incident ID         : 1
Incident Type       : Robbery
Incident Date       : 2023-01-15
Description          : Armed robbery at jewelry store
Status              : Closed
Role in Incident    : Suspect in prior theft
Added By Officer   : 1
Previous Criminal Record : No

-----
Suspect ID          : 1031
Name                : Sunita Sharma
Date of Birth       : 1986-03-25
Aadhaar Number     : 565656565656
Address             : Flat 7, Kothrud, Pune
Contact Number      : 7210789017
Incident ID         : 27
Incident Type       : Cyber Crime
Incident Date       : 2025-06-12
Description          : Online job scam defrauding 50 people
Status              : Open
Role in Incident    : Handled financial transactions for scam
Added By Officer   : 2
Previous Criminal Record : No
```

9. Officer Menu: Create new suspect record

```
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 5
===== Adding a new Suspect =====
First Name: Krishna
Last Name: Kumar
Date of Birth (YYYY-MM-DD): 1990-05-31
Gender: Male
Residential Address: 89, Park Road, Mumbai
Contact Number: 9800345004
Aadhaar Number: 123456123456
Incident ID involved: 2
Your Officer ID: 1
Role in Incident: Was caught running with a bundle of cash in hand.
VALUES INSERTED: ('Krishna', 'Kumar', datetime.date(1990, 5, 31), 'Male', '89, Park Road, Mumbai', 9800345004, 123456123456)
Suspect added successfully with ID: 1033
```

10.Officer Menu: Access criminals database

```
-- Case Management --
7. View case details (victims/suspects/evidence)
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 6
===== Criminals Database =====
Criminal ID      : 1
Incident ID      : 1
Punishment Details : 5 years imprisonment under IPC 392
Aadhaar Number   : 123412341234
First Name       : Vijay
Last Name        : Malhotra
Date of Birth    : 1980-05-15
Address          : Room No. 12, Chawl No.5, Dharavi, Mumbai
Contact Number   : 9876123450
Gender           : Male
-----
Criminal ID      : 2
Incident ID      : 4
Punishment Details : 2 years imprisonment under IPC 326
Aadhaar Number   : 456745674567
First Name       : Ramesh
Last Name        : Patel
Date of Birth    : 1982-04-18
Address          : Flat 34, Old City, Hyderabad
Contact Number   : 6547456783
Gender           : Male
```

11.Officer Menu: View case details (Victims/Suspects/Evidence)

```
9. View case report
10. Logout
Enter choice: 7
View Case Related Resources:

Enter the Incident ID: 3
Victims related to Case 3:
Victim ID: 102
First Name: Rahul
Last Name: Iyer
Date Of Birth: 1990-11-08
Gender: Male
Residential Address: No. 45, 5th Cross, Koramangala, Bengaluru
Contact Number: 7654321098
Aadhaar Number: 345678901234
-----
Suspects related to Case 3:
Suspect ID: 1002
First Name: Suresh
Last Name: Kumar
Date Of Birth: 1978-11-30
Gender: Male
Residential Address: No. 56, Shivajinagar, Bengaluru
Contact Number: 765634563456
Aadhaar Number: 345634563456
Role in Incident: Created fake investment website
Added By Officer: 3
```

12.Officer Menu: Generate/Update case report

```
// View case details (victims/suspects/evidence)
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 8
Do you wish to generate an incident report or update an existing report(Generate/update): generate

==== Generate Case Report ===
Enter Incident ID: 31
Enter Report Details: A male individual suspected to have killed a 70 year old man at late night on the streets
Enter Report Status (Draft/Finalized): Draft
Report generated successfully! Report ID: 125

==== Officer Menu ===
```

```
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 8
Do you wish to generate an incident report or update an existing report(Generate/update): update
==== Update Case Report ====

Enter Incident ID: 31
Enter Report Details: A male individual named Mohammed Yadav with suspect ID 1034 has been identified as the criminal
Enter Report Status (Draft/Finalized): Finalized
Report updated successfully!

==== Officer Menu ===
```

13.Officer Menu: View case report

```
9. View case report
10. Logout
Enter choice: 9
Enter Incident ID: 27

===== Incident Report =====
Incident ID : 27
Incident Type : Cyber Crime
Area : Gurgaon
City : Delhi-NCR
Description : Online job scam defrauding 50 people
Incident Status : Open
Report Date : 2025-06-13
Report Details : Online job scam defrauding 50 people of approximately ₹25 lakhs collectively.
Report Status : Draft
```

14. Admin Menu: Login

```
4. Login as officer/admin
0. Exit
Enter your choice: 4

==== Login ====
Username: vikram.singh
Password:
Welcome, vikram.singh!

==== Admin Menu (vikram.singh) ===
```

15.Admin Menu: Create criminal record

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 1

==== Add Criminal Record ===
Enter Incident ID: 31
Enter Aadhaar Number of criminal: 678900678900
Enter Punishment Details: Sentenced to 7 years in prison
Criminal added successfully. Criminal ID: 23

==== Admin Menu (vikram.singh) ===
```

16.Admin Menu: Create incident record

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 2

==== Create New Incident ===
Incident Type: Murder
Incident Date (YYYY-MM-DD): 2025-06-30
Area: Porur
City: Chennai
Description: A late night street murder of an old man

Searching for officers in the area...

Available Officers:
Officer ID : 5
Name : Arun Iyer
Posting City : Chennai
Posting State : Tamil Nadu
Current Cases : 2
-----
```

```
Incident Date (YYYY-MM-DD): 2025-06-30
Area: Porur
City: Chennai
Description: A late night street murder of an old man

Searching for officers in the area...

Available Officers:
Officer ID : 5
Name : Arun Iyer
Posting City : Chennai
Posting State : Tamil Nadu
Current Cases : 2
-----
Officer ID : 20
Name : Anita Menon
Posting City : Chennai
Posting State : Tamil Nadu
Current Cases : 1
-----
Enter Officer ID to assign this case to: 20
Incident created with ID 31 and assigned to officer 20
```

17.Admin Menu: Add new officer

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 3

==== Create New Officer Account ====
First Name: Radhika
Last Name: Rakesh
Date of Birth (YYYY-MM-DD): 1994-03-25
Gender: Female
Badge Number: TN-9873
Ranking (e.g., Inspector, Sub-Inspector): Sub-Inspector
Posting City: Chennai
Posting State: Tamilnadu
Service Joining Date (YYYY-MM-DD): 2024-06-01
Residential Address: 67, Arunachalam Road, Koyambedu
Contact Number: 9002390432
Agency ID: 505

--- Login Credentials ---
Username: radhika.rakesh
Password: chennai@178
Officer account created successfully. ID: 36
```

18. Admin Menu: View case assignments

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 4

==== All Incidents ====

Incident 1:
ID : 1
Type : Robbery
Date : 2023-01-15
Area : Andheri West
City : Mumbai
Description : Armed robbery at jewelry store
Status : Closed
Officer ID : 1

-----
Incident 2:
ID : 2
Type : Burglary
Date : 2023-02-03
Area : Saket
City : Delhi
Description : Residential burglary during daytime
Status : Under Investigation
```

19. Admin Menu: View all law enforcement agencies

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 5

==== Law Enforcement Agencies ====
Agency ID: 500
Agency Name: Mumbai Police
Jurisdiction: Mumbai
Email Address: mumbaipolice@maha.gov.in

-----
Agency ID: 501
Agency Name: Delhi Police
Jurisdiction: Delhi
Email Address: delhipolice@delhi.gov.in
```

20.Admin Menu: Officer Performance Dashboard

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 6

===== Officer Dashboard =====

-----
Officer ID      : 1
Name            : Rajesh Kumar
Badge Number    : MH-4521
Ranking         : Inspector
Posting Location : Mumbai, Maharashtra
Service Joined On : 2005-07-10
Total Cases     : 2
Cases Closed    : 1

-----
Officer ID      : 2
Name            : Priya Sharma
Badge Number    : DL-7854
Ranking         : Sub-Inspector
Posting Location : Delhi, Delhi
Service Joined On : 2010-03-15
Total Cases     : 2
Cases Closed    : 0
```

21.Admin Menu: Criminal Analytics

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 7

===== Criminal Analytics Dashboard =====

Top 5 Most Frequent Criminals:
1. Vijay Malhotra - 1 cases
2. Raju Khan - 1 cases
3. Suresh Kumar - 1 cases
4. Ramesh Patel - 1 cases
5. Mohan Iyer - 1 cases

Crime Type Distribution:
Robbery : 2
Burglary : 1
Cyber Crime : 2
Assault : 1
Theft : 2
Homicide : 1
Vehicle Theft : 1
Fraud : 2
Kidnapping : 1
Domestic Violence : 2
Drug Offense : 1
Sexual Assault : 1
Arson : 1
Public Nuisance : 1
Hit and Run : 1
Vandalism : 1
Identity Theft : 1
Harassment : 1
Bribery : 1
Child Abuse : 1
Eve Teasing : 1
Illegal Gambling : 1
Forgery : 1
Extortion : 1
Trespassing : 1
```

22.Admin Menu: Crime Pattern analytics

```
9. Finalize reports & update case status
10. Logout
Enter choice: 8

===== Crime Analytics Dashboard =====

Top 5 Crime Hotspots (City & Area):
1. Delhi, Saket - 2 incidents
2. Hyderabad, Banjara Hills - 2 incidents
3. Chennai, T Nagar - 2 incidents
4. Mumbai, Andheri West - 1 incidents
5. Bengaluru, Koramangala - 1 incidents

Crime Count by Month:
January 2023 : 1
February 2023 : 3
March 2023 : 3
April 2023 : 3
May 2023 : 3
June 2023 : 3
July 2023 : 3
August 2023 : 3
September 2023 : 3
June 2025 : 5
```

23.Admin Menu: Finalize reports & update case status

```
-- Case Oversight --
9. Finalize reports & update case status
10. Logout
Enter choice: 9

==== Reports Pending Review ===

Incident ID      : 1
Type             : Robbery
Area/City        : Andheri West, Mumbai
Status           : Closed
Report Date      : 2023-01-16
Report Details   : Robbery at jewelry store with estimated loss of ₹25 lakhs. One suspect apprehended.

Incident ID      : 3
Type             : Cyber Crime
Area/City        : Koramangala, Bengaluru
Status           : Open
Report Date      : 2023-02-11
Report Details   : Online fraud case involving ₹12 lakhs transferred to fake accounts.
```

```
Incident ID      : 28
Type             : Domestic Violence
Area/City        : Whitefield, Bengaluru
Status           : Under Investigation
Report Date      : 2025-06-19
Report Details   : Domestic violence case with victim requiring hospitalization. Husband absconding.

Incident ID      : 31
Type             : Murder
Area/City        : Porur, Chennai
Status           : Open
Report Date      : 2025-06-30
Report Details   : A male individual named Mohammed Yadav with suspect ID 1034 has been identified as the criminal

Enter the Incident ID to close: 31
Case status of Incident: 31 updated to 'CLOSED' successfully.
```

24.Pytest Output

The screenshot shows a code editor interface with two tabs: 'main.py' and 'test_usertestcases.py'. The 'test_usertestcases.py' tab is active, displaying Python test code. The code includes several test functions: `test_add_duplicate_suspect`, `test_add_invalid_suspect_data`, and others. The test session starts with the platform information: 'platform win32 ... Python 3.13.1, pytest-8.4.1, pluggy-1.6.0 ... C:\Python313\python.exe'. It then lists the collected items: 'collected 10 items'. The test results are shown in a table with columns for the test name and status (PASSED). A progress bar at the bottom indicates the completion of the test session.

Test Name	Status
tests/test_usertestcases.py::test_public_access	PASSED
tests/test_usertestcases.py::test_login_invalid	PASSED
tests/test_usertestcases.py::test_get_officers_by_location	PASSED
tests/test_usertestcases.py::test_get_incidents_in_date_range_public_no_results	PASSED
tests/test_usertestcases.py::test_get_incidents_by_area_city_public_city_only	PASSED
tests/test_usertestcases.py::test_get_incidents_by_area_city_public_invalid	PASSED
tests/test_usertestcases.py::test_filtered_incidents_combination	PASSED
tests/test_usertestcases.py::test_view_my_incidents_invalid	PASSED
tests/test_usertestcases.py::test_add_duplicate_suspect	PASSED
tests/test_usertestcases.py::test_add_invalid_suspect_data	PASSED

===== 10 passed in 0.71s =====

PS D:\Victus Laptop\Downloads\Hexaware\Python Training\Case Study\CARS>

Implementation and Testing :

Dao Folder :

#CrimeAnalysisService.py

```
CrimeAnalysisService.py x
dao > CrimeAnalysisService.py > iCrimeAnalysisService > get_incidents_in_date_range_public
1  from abc import ABC, abstractmethod
2
3  class ICrimeAnalysisService(ABC):
4      @abstractmethod
5      def create_incident(self, incident):
6          pass
7
8      @abstractmethod
9      def update_incident_status(self, status, incidentID):
10         pass
11
12     @abstractmethod
13     def get_incidents_in_date_range_public(self, start_date, end_date):
14         pass
15
16     @abstractmethod
17     def get_incidents_by_area_city_public(self, area, city):
18         pass
19
20     @abstractmethod
21     def get_filtered_incidents(self, start_date, end_date, area, city, incident_type, officer_id):
22         pass
23
24     @abstractmethod
25     def create_officer(self, officer_data):
26         pass
27
28     @abstractmethod
29     def get_officers_by_location(self, city, area):
30         pass
31
32     @abstractmethod
33     def count_open_incidents_by_officer(self, officer_id):
34         pass
35
```

```
dao > 🐍 CrimeAnalysisService.py > 🏷 ICrimeAnalysisService > 📁 get_incidents_in_date_range_public
 3   class ICrimeAnalysisService(ABC):
 4     ...
 5     @abstractmethod
 6     def generate_incident_report(self, report):
 7       pass
 8
 9     @abstractmethod
10     def view_incident_report(self, incidentID):
11       pass
12
13     @abstractmethod
14     def get_victims_by_incident(self, incidentID):
15       pass
16
17     @abstractmethod
18     def get_suspects_by_incident(self, incidentID):
19       pass
20
21     @abstractmethod
22     def get_evidence_by_incident(self, incidentID):
23       pass
24
25     @abstractmethod
26     def update_incident_status(self, incidentID, updatedReport):
27       pass
28
29     @abstractmethod
30     def viewIncident(self, incidentID):
31       pass
32
33     @abstractmethod
34     def viewMyIncidents(self, officerID):
35       pass
36
37     @abstractmethod
38     def viewAllIncidents(self):
39       pass
```

```
3  class ICrimeAnalysisService(ABC):
4      pass
5
6      @abstractmethod
7      def suspectCriminalRelation(self, aadhaarID):
8          pass
9
10     @abstractmethod
11     def addSuspect(self, suspect):
12         pass
13
14     @abstractmethod
15     def addCriminal(self, criminal):
16         pass
17
18     @abstractmethod
19     def viewAllSuspects(self):
20         pass
21
22     @abstractmethod
23     def viewAllCriminals(self):
24         pass
25
26     @abstractmethod
27     def get_crime_analytics(self):
28         pass
29
30     @abstractmethod
31     def get_criminal_analytics(self):
32         pass
33
34     @abstractmethod
35     def get_all_officer_dashboard(self):
36         pass
37
38     @abstractmethod
39     def print_recent_incidents(self):
40         pass
41
42
43     @abstractmethod
44     def get_reports_by_status(self, status):
45         pass
```

#CrimeAnalysisServiceImpl.py

```
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl
1  from dao.CrimeAnalysisService import ICrimeAnalysisService
2  from util.DB_Connections import DBConnections
3  from entity.Incidents import Incidents
4  from entity.Victims import Victims
5  from entity.Suspects import Suspects
6  from entity.Criminals import Criminals
7  from entity.Evidence import Evidence
8  from entity.SuspectIncidents import SuspectIncidents
9  from entity.LawEnforcementAgencies import LawEnforcementAgencies
10 from exceptions.user_defined_exceptions import DatabaseError, IncidentNotFoundException, DuplicateEntryException, SuspectNotFoundException,
11
12
13 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
14     def __init__(self):
15         self.connection = DBConnections.get_connection('db.properties')
16
17     def create_incident(self, incident):
18         try:
19             cursor = self.connection.cursor()
20             query = """
21                 INSERT INTO Incidents (IncidentType, IncidentDate, Area, City,
22                                         Description, Status, OfficerID)
23                 VALUES (%s, %s, %s, %s, %s, %s, %s)
24             """
25             cursor.execute(query,
26                           incident.incidentType, incident.incidentDate, incident.area,
27                           incident.city, incident.description, incident.status, incident.officerID
28                         )
29             self.connection.commit()
30             return cursor.lastrowid
31
32         except Exception as e:
33             self.connection.rollback()
34             raise DatabaseError(f"Failed to create incident: {str(e)}")
35
36
```

```
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
14
15      def update_incident_status(self, status, incidentID):
16          try:
17              cursor = self.connection.cursor()
18              query = "UPDATE Incidents SET Status = %s WHERE IncidentID = %s"
19              cursor.execute(query, (status, incidentID))
20              self.connection.commit()
21              return cursor.rowcount > 0
22
23          except IncidentNotFoundException as e:
24              print("Incident not found.")
25          except Exception as e:
26              self.connection.rollback()
27              raise DatabaseError(f"Failed to update incident status: {str(e)}")
28
29      def get_incidents_in_date_range_public(self, start_date, end_date):
30          try:
31              incidents = []
32              cursor = self.connection.cursor(dictionary=True)
33              query = """
34                  SELECT IncidentType, IncidentDate, Area, City, Description FROM Incidents
35                  WHERE IncidentDate BETWEEN %s AND %s
36                  ORDER BY IncidentDate
37              """
38              cursor.execute(query, (start_date, end_date))
39
40              for row in cursor:
41                  incidents.append(Incidents(
42                      IncidentType = row['IncidentType'],
43                      IncidentDate = row['IncidentDate'],
44                      Area = row['Area'],
45                      City = row['City'],
46                      Description = row['Description']
47                    ))
48
49              return incidents
50          except IncidentNotFoundException as e:
51              print("Incident not found.")
52          except Exception as e:
53              raise DatabaseError(f"Failed to fetch incidents: {str(e)}")
54
```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_incidents_by_area_city_public
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
77
78      def get_incidents_by_area_city_public(self, area=None, city=None):
79          try:
80              cursor = self.connection.cursor(dictionary=True)
81
82              query = "SELECT IncidentType, IncidentDate, Area, City, Description FROM Incidents WHERE 1=1"
83              params = []
84
85              if area:
86                  query += " AND Area = %s"
87                  params.append(area)
88
89              if city:
90                  query += " AND City = %s"
91                  params.append(city)
92
93                  query += " ORDER BY IncidentDate DESC"
94
95              cursor.execute(query, params)
96              incidents = cursor.fetchall()
97
98              return -incidents
99
100
101         except IncidentNotFoundException as e:
102             print("Incident not found.")
103         except Exception as e:
104             raise DatabaseError(f"Error fetching incidents by area/city: {str(e)}")
105
106     def get_filtered_incidents(self, start_date=None, end_date=None, area=None, city=None, incident_type=None, officer_id=None):
107
108         incidents = []
109         try:
110             cursor = self.connection.cursor(dictionary=True)
111
112             query = "SELECT * FROM Incidents WHERE 1=1"
113             params = []
114
115             if start_date:
116                 query += " AND IncidentDate >= %s"
117                 params.append(start_date)
118
119             if end_date:
120                 query += " AND IncidentDate <= %s"
121                 params.append(end_date)
122
123             if area:
124                 query += " AND Area = %s"
125                 params.append(area)
126
127             if city:
128                 query += " AND City = %s"
129                 params.append(city)
130
131             if incident_type:
132                 query += " AND IncidentType = %s"
133                 params.append(incident_type)
134
135             if officer_id:
136                 query += " AND OfficerID = %s"
137                 params.append(officer_id)
138
139             query += " ORDER BY IncidentDate DESC"
140
141             cursor.execute(query, params)
142
143             for row in cursor:
144                 incidents.append({
145                     'IncidentID': row['IncidentID'],
146                     'IncidentType': row['IncidentType'],
147                     'IncidentDate': row['IncidentDate'].strftime('%Y-%m-%d') if row['IncidentDate'] else None,
148                     'Area': row['Area'],
149                     'City': row['City'],
150                     'Description': row['Description'],
151                     'Status': row['Status'],
152                     'OfficerID': row['OfficerID']
153                 })
154
155             return incidents
156
157
158         except IncidentNotFoundException as e:
159             print("Incident not found.")
160         except Exception as e:
161             raise DatabaseError(f"Filter error: {str(e)}")
```

```

1 CrimeAnalysisServiceImpl > CrimeAnalysisServiceImpl > get_incidents_by_area_city_public
2 class CrimeAnalysisServiceImpl implements ICrimeAnalysisService:
3
4     def create_officer(self, officer_data):
5         try:
6             cursor = self.connection.cursor()
7             cursor.execute("SELECT COUNT(*) FROM lawenforcementagencies WHERE AgencyID = %s",
8                           (officer_data['agencyID'],))
9             result = cursor.fetchone()
10            if result[0] == 0:
11                raise AgencyNotFoundException(f"AgencyID {officer_data['agencyID']} does not exist.")
12            officer_query = """
13                INSERT INTO Officers (FirstName, LastName, DateOfBirth, Gender, BadgeNumber, Ranking, PostingCity, PostingState, ServiceJoiningDate, ResidentialAddress, ContactNumber, AgencyID)
14                VALUES (%s, %s, %s)
15                """
16
17            cursor.execute(officer_query,
18                          officer_data['firstName'],
19                          officer_data['lastName'],
20                          officer_data['dateOfBirth'],
21                          officer_data['gender'],
22                          officer_data['badgeNumber'],
23                          officer_data['ranking'],
24                          officer_data['postingCity'],
25                          officer_data['postingState'],
26                          officer_data['serviceJoiningDate'],
27                          officer_data['residentialAddress'],
28                          officer_data['contactNumber'],
29                          officer_data['agencyID']
30                      )
31
32            officerID = cursor.lastrowid
33
34            user_query = """
35                INSERT INTO Users (Username, Password, Role, OfficerID)
36                VALUES (%s, %s, %s, %s)
37                """
38
39            cursor.execute(user_query,
40                          officer_data['username'],
41                          officer_data['password'],
42                          'Officer', officerID
43                      )
44
45            self.connection.commit()
46            return officerID
47
48        except AgencyNotFoundException as e:
49            print("Agency not found.")
50        except Exception as e:
51            self.connection.rollback()
52            raise DatabaseError(f"Failed to add officer: {str(e)}")
53
54    def get_officers_by_location(self, city, area):
55        try:
56            cursor = self.connection.cursor(dictionary=True)
57            query = """
58                SELECT * FROM Officers
59                WHERE PostingCity = %s OR PostingState = %s
60                """
61
62            cursor.execute(query, (city, area))
63            return cursor.fetchall()
64
65        except OfficerNotFoundException as e:
66            print("Officers not found. ")
67        except Exception as e:
68            raise DatabaseError(f"Failed to fetch officers: {str(e)}")
69
70    def count_open_incidents_by_officer(self, officer_id):
71        try:
72            cursor = self.connection.cursor()
73            query = """
74                SELECT COUNT(*) FROM Incidents
75                WHERE OfficerID = %s AND Status != 'Closed'
76                """
77
78            cursor.execute(query, (officer_id,))
79            count = cursor.fetchone()[0]
80            return count
81
82        except OfficerNotFoundException as e:
83            print("Officers not found. ")
84        except IncidentNotFoundException as e:
85            print("Incident not found. ")
86        except Exception as e:
87            raise DatabaseError(f"Failed to count open cases: {str(e)}")

```

```
CrimeAnalysisService.py CrimeAnalysisServiceImpl.py
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > generate_incident_report
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def generate_incident_report(self, report):
        try:
            cursor = self.connection.cursor()

            check_officer_query = """
                SELECT OfficerID FROM Incidents WHERE IncidentID = %s
            """
            cursor.execute(check_officer_query, (report.incidents.incidentID,))
            result = cursor.fetchone()
            if not result:
                raise IncidentNotFoundException(f"Incident ID {report.incidents.incidentID} not found.")

            officer_in_charge = result[0]
            if officer_in_charge != report.officers.officerID:
                raise PermissionError(f"You are not authorized to generate a report for Incident ID {report.incidents.incidentID}.")

            cursor.execute("SELECT ReportID FROM Reports WHERE IncidentID = %s", (report.incidents.incidentID,))
            if cursor.fetchone():
                raise DuplicateEntryException(f"A report already exists for Incident ID {report.incidents.incidentID}.")

            insert_query = """
                INSERT INTO Reports (IncidentID, ReportingOfficerID, ReportDate, ReportDetails, Status)
                VALUES (%s, %s, %s, %s, %s)
            """
            cursor.execute(insert_query, (
                report.incidents.incidentID,
                report.officers.officerID,
                report.reportDate,
                report.reportDetails,
                report.status
            ))
            self.connection.commit()
            return cursor.lastrowid
        except (IncidentNotFoundException, PermissionError, DuplicateEntryException):
            raise
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to generate report: {str(e)}")

    def update_incident_report(self, report):
        try:
            cursor = self.connection.cursor()

            check_officer_query = """
                SELECT OfficerID FROM Incidents WHERE IncidentID = %s
            """
            cursor.execute(check_officer_query, (report.incidents.incidentID,))
            result = cursor.fetchone()
            if not result:
                raise IncidentNotFoundException(f"Incident ID {report.incidents.incidentID} not found.")

            officer_in_charge = result[0]
            if officer_in_charge != report.officers.officerID:
                raise PermissionError(f"You are not authorized to update the report for Incident ID {report.incidents.incidentID}.")

            update_query = """
                UPDATE Reports
                SET ReportDetails = %s, Status = %s, ReportDate = %s
                WHERE IncidentID = %s
            """
            cursor.execute(update_query, (
                report.reportDetails,
                report.status,
                report.reportDate,
                report.incidents.incidentID
            ))
            if cursor.rowcount == 0:
                raise ReportNotFoundException(f"No report found for Incident ID {report.incidents.incidentID}")

            self.connection.commit()
            return True

        except (IncidentNotFoundException, PermissionError, ReportNotFoundException):
            raise
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to update report: {str(e)}")
```

```
CrimeAnalysisService.py | CrimeAnalysisServiceImpl.py X
o > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > generate_incident_report
13 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
14
15
16
17
18     def view_incident_report(self, incidentID):
19         try:
20             cursor = self.connection.cursor(dictionary=True)
21             query = """
22                 SELECT i.IncidentID, i.IncidentType, i.Area, i.City, i.Description, i.Status,
23                 r.ReportDetails, r.ReportDate, r.Status AS ReportStatus
24                 FROM Incidents i
25                 JOIN Reports r
26                 ON i.IncidentID = r.IncidentID
27                 WHERE i.IncidentID = %s
28             """
29             cursor.execute(query, (incidentID,))
30             report = cursor.fetchone()
31             return report
32
33
34         except IncidentNotFoundException as e:
35             print("Incident not found.")
36         except Exception as e:
37             raise DatabaseError(f"Failed to display report: {str(e)}")
38
39     def get_victims_by_incident(self, incidentID):
40         try:
41             cursor = self.connection.cursor()
42             query = "SELECT * FROM Victims WHERE IncidentID = %s"
43             cursor.execute(query, (incidentID,))
44             results = cursor.fetchall()
45
46             victims = []
47             for result in results:
48                 victim = Victims(
49                     VictimID = result[0],
50                     FirstName = result[1],
51                     LastName = result[2],
52                     DateOfBirth = result[3],
53                     Gender = result[4],
54                     ResidentialAddress = result[5],
55                     ContactNumber = result[6],
56                     AadhaarNumber = result[7],)
57                 victims.append(victim)
58
59             return victims
60
61         except IncidentNotFoundException as e:
62             print("Incident not found.")
63         except VictimNotFoundException as e:
64             print("Victim not found.")
65         except Exception as e:
66             raise DatabaseError(f"Failed to fetch victims list: {str(e)}")
```

```

1 CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > generate_incident_report
2 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
3
4     def get_suspects_by_incident(self, incidentID):
5         try:
6             cursor = self.connection.cursor()
7             query = """
8                 SELECT S.*, SI.IncidentID, SI.AddedByOfficerID, SI.RoleDescription
9                 FROM Suspects S
10                JOIN SuspectIncidents SI ON S.AadhaarNumber = SI.AadhaarNumber
11               WHERE SI.IncidentID = %s
12             """
13
14             cursor.execute(query, (incidentID,))
15             results = cursor.fetchall()
16
17             suspect_incidents = []
18             for row in results:
19                 suspect = Suspects(
20                     SuspectID=row[0],
21                     FirstName=row[1],
22                     LastName=row[2],
23                     DateOfBirth=row[3],
24                     Gender=row[4],
25                     ResidentialAddress=row[5],
26                     ContactNumber=row[6],
27                     AadhaarNumber=row[7]
28                 )
29
30                 suspect_incident = SuspectIncidents(
31                     Suspects=suspect,
32                     Incidents=row[8],
33                     Officers=row[9],
34                     RoleDescription=row[10]
35                 )
36                 suspect_incidents.append(suspect_incident)
37
38             return suspect_incidents
39
40         except SuspectNotFoundException as e:
41             print("Suspect not found.")
42         except IncidentNotFoundException as e:
43             print("Incident not found.")
44         except Exception as e:
45             raise DatabaseError(f"Failed to fetch suspects list: {str(e)}")
46
47 CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > generate_incident_report
48 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
49
50     def get_evidence_by_incident(self, incidentID):
51         try:
52             cursor = self.connection.cursor()
53             query = "SELECT * FROM Evidence WHERE IncidentID = %s"
54             cursor.execute(query, (incidentID,))
55             results = cursor.fetchall()
56
57             evidences = []
58             for result in results:
59                 evidence = Evidence(
60                     EvidenceID = result[0],
61                     EvidenceName = result[1],
62                     Description = result[2],
63                     LocationFound = result[3]
64                 )
65                 evidences.append(evidence)
66
67             return evidences
68
69         except IncidentNotFoundException as e:
70             print("Incident not found.")
71         except EvidenceNotFoundException as e:
72             print("Victim not found.")
73         except Exception as e:
74             raise DatabaseError(f"Failed to fetch evidence list: {str(e)}")
75
76     def update_incident_status(self, incidentID, updatedReport):
77         try:
78             cursor = self.connection.cursor()
79             query = "UPDATE Incidents SET Status = %s, ReportDetails = %s, ReportDate = %s WHERE IncidentID = %s"
80             cursor.execute(query, (updatedReport.status, updatedReport.reportDetails, updatedReport.reportDate, incidentID))
81             self.connection.commit()
82             return cursor.rowcount
83
84         except IncidentNotFoundException as e:
85             print("Incident not found.")
86         except Exception as e:
87             self.connection.rollback()
88             raise DatabaseError(f"Failed to update the status of the incident: {str(e)}")
89
90     def viewIncident(self, incidentID):
91         try:
92             cursor = self.connection.cursor()
93             query = "SELECT * FROM Incidents WHERE IncidentID = %s"
94             cursor.execute(query, (incidentID,))
95             result = cursor.fetchone()
96             return result
97
98         except IncidentNotFoundException as e:
99             print("Incident not found.")
100        except Exception as e:
101            raise DatabaseError(f"Couldn't display the incident: {str(e)}")
102
```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllIncidents
3 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
2
4     def viewMyIncidents(self, officerID):
5         try:
6             cursor = self.connection.cursor(dictionary=True)
7             query = "SELECT * FROM Incidents WHERE OfficerID = %s AND STATUS != 'Closed'"
8             cursor.execute(query, (officerID,))
9             results = cursor.fetchall()
10
11            incidents = []
12            for row in results:
13                incident = Incidents(
14                    IncidentID=row['IncidentID'],
15                    IncidentType=row['IncidentType'],
16                    IncidentDate=row['IncidentDate'],
17                    Area=row['Area'],
18                    City=row['City'],
19                    Description=row['Description'],
20                    Status=row['Status'],
21                    OfficerID=row['OfficerID']
22                )
23                incidents.append(incident)
24            return results
25
26        except IncidentNotFoundException as e:
27            print("Incident not found.")
28        except Exception as e:
29            raise DatabaseError(f"Couldn't display the incidents: {str(e)}")
30
31    def viewAllIncidents(self):
32        try:
33            cursor = self.connection.cursor()
34            query = "SELECT * FROM Incidents"
35            cursor.execute(query)
36            results = cursor.fetchall()
37
38            incidents = []
39            for result in results:
40                incident = Incidents(
41                    IncidentID = result[0],
42                    IncidentType = result[1],
43                    IncidentDate = result[2],
44                    Area = result[3],
45                    City = result[4],
46                    Description = result[5],
47                    Status = result[6],
48                    OfficerID = result[7]
49                )
50                incidents.append(incident)
51
52            return
53        except IncidentNotFoundException as e:
54            print("Incident not found.")
55        except Exception as e:
56            raise DatabaseError(f"Couldn't display the incidents: {str(e)}")
```

```

> CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllIncidents
3 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
>
6     def suspectCriminalRelation(self, aadhaarNumber):
7         try:
8             cursor = self.connection.cursor()
9             query = """
0             SELECT C.CriminalID, C.IncidentID, C.PunishmentDetails
1             FROM Suspects S
2             JOIN Criminals C
3             ON S.AadhaarNumber = C.AadhaarNumber
4             WHERE S.AadhaarNumber = %s
5             """
6             cursor.execute(query, (aadhaarNumber,))
7             results = cursor.fetchall()
8
9             criminalRecords = []
10            for result in results:
11                criminalRecord = Criminals(
12                    CriminalID = result[0],
13                    IncidentID = result[1],
14                    PunishmentDetails = result[2]
15                )
16                criminalRecords.append(criminalRecord)
17
18            return criminalRecords
19
20        except SuspectNotFoundException as e:
21            print("Suspects not found")
22        except CriminalNotFoundException as e:
23            print("Criminal not found")
24        except Exception as e:
25            raise DatabaseError(f"Failed to show the suspect and criminal relations: {str(e)}")
26

```

```

> CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > addSuspect
3 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
4
5     def addSuspect(self, suspect, incidentID, officerID, roleDescription):
6         try:
7             cursor = self.connection.cursor()
8             check_query = """
9                 SELECT SuspectID FROM Suspects
10                WHERE AadhaarNumber = %s
11                LIMIT 1
12             """
13             cursor.execute(check_query, (suspect.aadhaarNumber,))
14             existing_suspect = cursor.fetchone()
15             if existing_suspect:
16                 suspectID = existing_suspect[0]
17                 check_incident_query = """
18                     SELECT * FROM SuspectIncidents
19                     WHERE AadhaarNumber = %s AND IncidentID = %s
20                 """
21                 cursor.execute(check_incident_query, (suspect.aadhaarNumber, incidentID))
22                 existing_link = cursor.fetchone()
23                 if existing_link:
24                     raise DuplicateEntryException(f"Duplicate Entry: Suspect already assigned to Incident {incidentID}.")
25                 else:
26                     insert_incident_query = """
27                         INSERT INTO SuspectIncidents (SuspectID, AadhaarNumber, IncidentID, AddedByOfficerID, RoleDescription)
28                         VALUES (%s, %s, %s, %s, %s)
29                     """
30                     cursor.execute(insert_incident_query, (suspectID, suspect.aadhaarNumber, incidentID, officerID, roleDescription))
31                     self.connection.commit()
32                     return suspectID
33
34             else:
35                 insert_query = """
36                     INSERT INTO Suspects (
37                         FirstName, LastName, DateOfBirth, Gender,
38                         ResidentialAddress, ContactNumber, AadhaarNumber
39                     ) VALUES (%s, %s, %s, %s, %s, %s, %s)
40                 """
41                 cursor.execute(insert_query, (
42                     suspect.firstName, suspect.lastName, suspect.dateOfBirth,
43                     suspect.gender, suspect.residentialAddress, suspect.contactNumber, suspect.aadhaarNumber
44                 ))
45                 suspectID = cursor.lastrowid
46                 insert_incident_query = """
47                     INSERT INTO SuspectIncidents (SuspectID, AadhaarNumber, IncidentID, AddedByOfficerID, RoleDescription)
48                     VALUES (%s, %s, %s, %s, %s)
49                 """
50                 cursor.execute(insert_incident_query, (suspectID, suspect.aadhaarNumber, incidentID, officerID, roleDescription))
51                 self.connection.commit()
52                 return suspectID
53
54         except IncidentNotFoundException as e:
55             print("Incident not found")
56         except OfficerNotFoundException as e:
57             print("Officer not found")
58         except Exception as e:
59             self.connection.rollback()
60             raise DatabaseError(f"Failed to add suspect: {str(e)}")
61

```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > addSuspect
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def addCriminal(self, criminal):
        try:
            cursor = self.connection.cursor()
            query_suspect = "SELECT SuspectID FROM Suspects WHERE AadhaarNumber = %s"
            cursor.execute(query_suspect, (criminal.aadhaarNumber,))
            result = cursor.fetchone()

            if not result:
                raise SuspectNotFoundException(f"No suspect found with Aadhaar: {criminal.aadhaarNumber}")

            suspect_id = result[0]

            duplicate_query = """
                SELECT CriminalID FROM Criminals
                WHERE IncidentID = %s AND AadhaarNumber = %s
            """
            cursor.execute(duplicate_query, (criminal.incidentID, criminal.aadhaarNumber))
            duplicate = cursor.fetchone()

            if duplicate:
                print(f"Duplicate entry: Criminal record already exists for Incident ID {criminal.incidentID}")
                return None

            insert_query = """
                INSERT INTO Criminals (IncidentID, SuspectID, AadhaarNumber, PunishmentDetails)
                VALUES (%s, %s, %s, %s)
            """
            cursor.execute(insert_query, (
                criminal.incidentID,
                suspect_id,
                criminal.aadhaarNumber,
                criminal.punishmentDetails
            ))

            self.connection.commit()
            return cursor.lastrowid

        except IncidentNotFoundException as e:
            print("Incidents not found.")
        except SuspectNotFoundException as e:
            print("Suspect not found.")
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to add criminal: {str(e)}")
```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > addVictim
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def addVictim(self, victim, incidentID, officerID, roleDescription):
        try:
            cursor = self.connection.cursor()
            cursor.execute("SELECT IncidentID FROM Incidents WHERE IncidentID = %s", (incidentID,))
            incident = cursor.fetchone()
            if not incident:
                raise IncidentNotFoundException(f"Incident ID {incidentID} not found.")

            cursor.execute("""
                SELECT VictimID FROM Victims
                WHERE AadhaarNumber = %s AND IncidentID = %s
            """, (victim.aadhaarNumber, incidentID))
            existing = cursor.fetchone()
            if existing:
                print("Victim already exists for this incident.")
                return existing[0]

            insert_query = """
                INSERT INTO Victims (
                    FirstName, LastName, DateOfBirth, Gender,
                    ResidentialAddress, ContactNumber, AadhaarNumber, IncidentID
                ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
            """
            cursor.execute(insert_query, (
                victim.firstName, victim.lastName, victim.dateOfBirth, victim.gender,
                victim.residentialAddress, victim.contactNumber, victim.aadhaarNumber, incidentID
            ))

            self.connection.commit()
            new_victim_id = cursor.lastrowid
            return new_victim_id

        except Exception as e:
            print(f"Database Error: {str(e)}")
            raise
```

```
> CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > addSuspect
13 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
14
15     def viewAllSuspects(self):
16         try:
17             cursor = self.connection.cursor()
18             query = """
19                 SELECT SI.SuspectID, SI.IncidentID, SI.RoleDescription, SI.AddedByOfficerID,
20                 FirstName, S.LastName, S.DateOfBirth, S.AadhaarNumber, S.ResidentialAddress, S.ContactNumber,
21                 I.IncidentDate, I.IncidentType, I.Description, I.Status
22                 FROM SuspectIncidents SI JOIN Suspects S ON SI.AadhaarNumber = S.AadhaarNumber
23                 JOIN Incidents I ON SI.IncidentID = I.IncidentID
24                 ORDER BY SI.SuspectID DESC
25             """
26
27             cursor.execute(query)
28             results = cursor.fetchall()
29
30             suspectsInfo = []
31             for result in results:
32                 suspect = Suspects(
33                     SuspectID = result[0],
34                     FirstName = result[4],
35                     LastName = result[5],
36                     DateOfBirth = result[6],
37                     AadhaarNumber = result[7],
38                     ResidentialAddress = result[8],
39                     ContactNumber = result[9]
40                 )
41
42                 incident = Incidents(
43                     IncidentID = result[1],
44                     IncidentDate = result[10],
45                     IncidentType = result[11],
46                     Description = result[12],
47                     Status = result[13]
48                 )
49
50                 suspectIncident = SuspectIncidents(
51                     Suspects = suspect,
52                     Incidents = incident,
53                     RoleDescription = result[2],
54                     Officers = result[3]
55                 )
56                 suspectsInfo.append(suspectIncident)
57
58             return suspectsInfo
59
60         except IncidentNotFoundException as e:
61             print("Incidents not found.")
62         except SuspectNotFoundException as e:
63             print("Suspect not found.")
64         except Exception as e:
65             raise DatabaseError(f"Couldn't display all suspects: {str(e)}")
```

```
5 > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllCriminals
13 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
14
15     def viewAllCriminals(self):
16         try:
17             cursor = self.connection.cursor()
18             query = """
19                 SELECT C.CriminalID, C.IncidentID, C.PunishmentDetails,
20                 S.FirstName, S.LastName, S.DateOfBirth, S.Gender, S.ResidentialAddress,
21                 S.ContactNumber, S.AadhaarNumber
22                 FROM Criminals C
23                 JOIN Suspects S
24                 ON S.AadhaarNumber = C.AadhaarNumber
25             """
26
27             cursor.execute(query)
28             results = cursor.fetchall()
29
30             criminals = []
31             for result in results:
32                 criminal_id      = result[0]
33                 incident_id     = result[1]
34                 aadhaar_number   = result[9]
35                 punishment      = result[2]
36                 first_name       = result[3]
37                 last_name        = result[4]
38                 dob              = result[5]
39                 gender           = result[6]
40                 address          = result[7]
41                 contact          = result[8]
42                 suspect = Suspects(
43                     FirstName = first_name,
44                     LastName = last_name,
45                     DateOfBirth = dob,
46                     Gender = gender,
47                     ResidentialAddress = address,
48                     ContactNumber = contact,
49                     AadhaarNumber = aadhaar_number
50                 )
51                 criminal = Criminals(
52                     CriminalID = criminal_id,
53                     IncidentID = incident_id,
54                     AadhaarNumber = aadhaar_number,
55                     PunishmentDetails = punishment
56                 )
57                 criminal.suspect = suspect
58                 criminals.append(criminal)
59
60             return criminals
61         except IncidentNotFoundException as e:
62             print("Incidents not found.")
63         except SuspectNotFoundException as e:
64             print("Suspect not found.")
65         except CriminalNotFoundException as e:
66             print("Criminal not found")
67         except Exception as e:
68             raise DatabaseError(f"Couldn't display all criminals: {str(e)}")
```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllCriminals
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def get_crime_analytics(self):
        try:
            cursor = self.connection.cursor(dictionary=True)
            hotspot_query = """
                SELECT City, Area, COUNT(*) as crime_count
                FROM Incidents
                GROUP BY City, Area
                ORDER BY crime_count DESC
                LIMIT 5
            """
            cursor.execute(hotspot_query)
            hotspots = cursor.fetchall()

            monthly_trends_query = """
                SELECT
                    DATE_FORMAT(IncidentDate, '%M %Y') as month,
                    COUNT(*) as count
                FROM Incidents
                GROUP BY month
                ORDER BY MIN(IncidentDate)
            """
            cursor.execute(monthly_trends_query)
            monthly_trends = {row['month']: row['count'] for row in cursor}

        return {
            'hotspots': hotspots,
            'monthly_trends': monthly_trends
        }

    except IncidentNotFoundException as e:
        print("Incident not found")
    except Exception as e:
        raise DatabaseError(f"Failed to generate the crime analytics: {str(e)}")
```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllCriminals
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def get_criminal_analytics(self):
        try:
            cursor = self.connection.cursor(dictionary=True)

            top_criminals_query = """
                SELECT s.FirstName, s.LastName, COUNT(*) as crime_count
                FROM Criminals c
                JOIN Suspects s ON c.SuspectID = s.SuspectID
                GROUP BY c.SuspectID
                ORDER BY crime_count DESC
                LIMIT 5
            """
            cursor.execute(top_criminals_query)
            top_criminals = cursor.fetchall()

            crime_type_distribution_query = """
                SELECT IncidentType, COUNT(*) as count
                FROM Incidents
                GROUP BY IncidentType
            """
            cursor.execute(crime_type_distribution_query)
            crime_types = {row['IncidentType']: row['count'] for row in cursor}

        return {
            'top_criminals': top_criminals,
            'crime_types': crime_types
        }

    except CriminalNotFoundException as e:
        print("Criminal not found")
    except IncidentNotFoundException as e:
        print("Incident not found")
    except Exception as e:
        raise DatabaseError(f"Failed to generate the criminal analytics: {str(e)}")
```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllCriminals
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def get_all_officer_dashboard(self):
        try:
            cursor = self.connection.cursor(dictionary=True)

            query = """
            SELECT
            o.OfficerID,
            o.FirstName,
            o.LastName,
            o.BadgeNumber,
            o.Ranking,
            o.PostingCity,
            o.PostingState,
            o.ServiceJoiningDate,
            COUNT(i.IncidentID) AS total_cases,
            SUM(CASE WHEN i.Status = 'Closed' THEN 1 ELSE 0 END) AS cases_closed
            FROM Officers o
            LEFT JOIN Incidents i ON o.OfficerID = i.OfficerID
            GROUP BY
            o.OfficerID, o.FirstName, o.LastName, o.BadgeNumber,
            o.Ranking, o.PostingCity, o.PostingState, o.ServiceJoiningDate
            ORDER BY total_cases DESC
            """

            cursor.execute(query)
            dashboard_data = cursor.fetchall()
            return dashboard_data

        except OfficerNotFoundException as e:
            print("Officers not found")
        except IncidentNotFoundException as e:
            print("Incidents not found")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch officer dashboard: {str(e)}")
```

```
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllCriminals
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def print_recent_incidents(self):
        try:
            cursor = self.connection.cursor(dictionary=True)
            query = """
            SELECT IncidentDate, IncidentType, Description, Area, City
            FROM Incidents
            WHERE IncidentDate >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
            ORDER BY IncidentDate DESC
            """
            cursor.execute(query)
            incidents = cursor.fetchall()

            if not incidents:
                raise IncidentNotFoundException("No incidents found.")

            print("\nRECENT INCIDENTS REPORT (LAST 30 DAYS)")
            print("-" * 115)
            print(f"{'DATE':<12} | {'TYPE':<15} | {'AREA':<15} | {'CITY':<15} | {'DESCRIPTION'}")
            print("-" * 115)

            for incident in incidents:
                print(
                    f"{incident['IncidentDate'].strftime('%Y-%m-%d')}<12} | "
                    f"{incident['IncidentType'][:15]}<15} | "
                    f"{incident['Area']} + ', '}<15} | "
                    f"{incident['City'][:20]}<20} | "
                    f"{incident['Description']}<15}"
                )

        except IncidentNotFoundException as e:
            print("Incident not found")
        except Exception as e:
            raise DatabaseError(f"Couldn't print the recent incidents: {str(e)}")
```

```
> 🐍 CrimeAnalysisServiceImpl.py > 🐍 CrimeAnalysisServiceImpl > ⚒ view_all_agencies
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def get_reports_by_status(self, status):
        try:
            cursor = self.connection.cursor(dictionary=True)
            query = """
                SELECT i.IncidentID, i.IncidentType, i.Area, i.City, i.Status,
                       r.ReportDate, r.ReportDetails
                FROM Reports r
                JOIN Incidents i ON r.IncidentID = i.IncidentID
                WHERE r.Status = %s
            """
            cursor.execute(query, (status,))
            return cursor.fetchall()

        except IncidentNotFoundException as e:
            print("Incident not found")
        except ReportNotFoundException as e:
            print("Report not found")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch reports by status: {str(e)}")

    def view_all_agencies(self):
        try:
            cursor = self.connection.cursor(dictionary=True)
            query = "SELECT * FROM LawEnforcementAgencies"
            cursor.execute(query)
            results = cursor.fetchall()

            agencies = []
            for row in results:
                agency = LawEnforcementAgencies(
                    AgencyID = row['AgencyID'],
                    AgencyName = row['AgencyName'],
                    Jurisdiction = row['Jurisdiction'],
                    EmailAddress = row['EmailAddress']
                )
                agencies.append(agency)
            return agencies

        except AgencyNotFoundException as e:
            print("Agency not found")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch law enforcement agencies: {str(e)}")
```

```
dao > 🐍 CrimeAnalysisServiceImpl.py > 🐍 CrimeAnalysisServiceImpl > ⚒ view_all_agencies
13 class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
971
972     def close_case_by_admin(self, incident_id):
973         try:
974             cursor = self.connection.cursor()
975             update_query = """
976                 UPDATE Reports
977                 SET Status = 'Closed'
978                 WHERE IncidentID = %s AND Status = 'Finalized'
979             """
980             cursor.execute(update_query, (incident_id,))
981             self.connection.commit()
982             return cursor.rowcount
983
984         except ReportNotFoundException as e:
985             print("Report not found")
986         except IncidentNotFoundException as e:
987             print("Incident not found")
988         except Exception as e:
989             self.connection.rollback()
990             raise DatabaseError(f"Failed to close case: {str(e)}")
991
```

Database Folder

#CARS_CreatingTables.sql

```
database > CARS_CreatingTables.sql > ...
1  CREATE DATABASE CrimeReportingSystem;
2  USE CrimeReportingSystem;
3
4  -- Creating LawEnforcementAgencies table
5  CREATE TABLE LawEnforcementAgencies
6  (
7      AgencyID INT AUTO_INCREMENT PRIMARY KEY,
8      AgencyName VARCHAR(150),
9      Jurisdiction VARCHAR(100),
10     EmailAddress VARCHAR(250)
11 )AUTO_INCREMENT = 500;
12
13 -- Creating Officers Table
14 CREATE TABLE Officers (
15     OfficerID INT AUTO_INCREMENT PRIMARY KEY,
16     FirstName VARCHAR(50),
17     LastName VARCHAR(50),
18     DateOfBirth DATE,
19     Gender VARCHAR(20),
20     BadgeNumber VARCHAR(20),
21     Ranking VARCHAR(30),
22     PostingCity VARCHAR(30),
23     PostingState VARCHAR(30),
24     ServiceJoiningDate DATE,
25     ResidentialAddress TEXT,
26     ContactNumber BIGINT,
27     AgencyID INT,
28     CONSTRAINT fkOfficers_LawEnforcementAgencies
29     FOREIGN KEY(AgencyID)
30     REFERENCES LawEnforcementAgencies(AgencyID)
31 )AUTO_INCREMENT = 1;
32
```

```

52
53 -- Creating Incidents Table
54 CREATE TABLE Incidents (
55     IncidentID INT AUTO_INCREMENT PRIMARY KEY,
56     IncidentType VARCHAR(100),
57     IncidentDate DATE,
58     Area VARCHAR(100),
59     City VARCHAR(100),
60     Description TEXT,
61     Status ENUM ('Open', 'Under Investigation', 'Closed') NOT NULL DEFAULT 'Open',
62     OfficerID INT,
63     CONSTRAINT fk_IncidentsOfficers
64     FOREIGN KEY(OfficerID)
65     REFERENCES Officers(OfficerID)
66 )AUTO_INCREMENT = 1;
67
68 -- Creating Victims table
69 CREATE TABLE Victims (
70     VictimID INT AUTO_INCREMENT PRIMARY KEY,
71     FirstName VARCHAR(50),
72     LastName VARCHAR(50),
73     DateOfBirth DATE,
74     Gender VARCHAR(50),
75     ResidentialAddress TEXT,
76     ContactNumber BIGINT,
77     AadhaarNumber BIGINT,
78     IncidentID INT,
79     CONSTRAINT fk_VictimsIncidents
80     FOREIGN KEY(IncidentID)
81     REFERENCES Incidents(IncidentID)
82 )AUTO_INCREMENT = 100;
83
84

```

```

database > CARS_CreatingTables.sql > ...
64 -- Creating Suspects table
65 CREATE TABLE Suspects (
66 (
67     SuspectID INT AUTO_INCREMENT PRIMARY KEY,
68     FirstName VARCHAR(100),
69     LastName VARCHAR(100),
70     DateOfBirth DATE,
71     Gender VARCHAR(50),
72     ResidentialAddress TEXT,
73     ContactNumber BIGINT,
74     AadhaarNumber BIGINT UNIQUE
75 )AUTO_INCREMENT = 1000;
76
77 -- Creating Evidence Table
78 CREATE TABLE Evidence (
79     EvidenceID INT AUTO_INCREMENT PRIMARY KEY,
80     EvidenceName VARCHAR(20),
81     Description TEXT,
82     LocationFound TEXT,
83     IncidentID INT,
84     CONSTRAINT fk_EvidenceIncidents
85     FOREIGN KEY(IncidentID)
86     REFERENCES Incidents(IncidentID)
87 )AUTO_INCREMENT = 1;
88
89 -- Creating Reports Table
90 CREATE TABLE Reports (
91     ReportID INT AUTO_INCREMENT PRIMARY KEY,
92     IncidentID INT,
93     ReportingOfficerID INT,
94     ReportDate DATE,
95     ReportDetails TEXT,
96     STATUS ENUM('Draft', 'Finalized', 'Closed') NOT NULL DEFAULT 'Draft',
97     CONSTRAINT fk_ReportsIncidents FOREIGN KEY(IncidentID) REFERENCES Incidents(IncidentID),
98     CONSTRAINT fk_ReportsOfficers FOREIGN KEY(ReportingOfficerID) REFERENCES Officers(OfficerID)
99 )AUTO_INCREMENT = 100;
100

```

```
database > CARS_CreatingTables.sql > ...
100
101 CREATE TABLE Criminals (
102     CriminalID INT AUTO_INCREMENT PRIMARY KEY,
103     IncidentID INT,
104     SuspectID INT,
105     AadhaarNumber BIGINT UNIQUE,
106     PunishmentDetails TEXT,
107     CONSTRAINT fk_Criminals_Incidents FOREIGN KEY(IncidentID) REFERENCES Incidents(IncidentID),
108     CONSTRAINT fk_Criminals_Suspects FOREIGN KEY(SuspectID) REFERENCES Suspects(SuspectID)
109 )AUTO_INCREMENT = 1;
110
111 CREATE TABLE Users (
112     UserID INT AUTO_INCREMENT PRIMARY KEY,
113     Username VARCHAR(50) UNIQUE NOT NULL,
114     Password VARCHAR(100) NOT NULL,
115     Role ENUM('Officer', 'Admin') NOT NULL,
116     OfficerID INT NULL,
117     FOREIGN KEY (OfficerID) REFERENCES Officers(OfficerID)
118 );
119
120 CREATE TABLE SuspectIncidents (
121     SuspectID INT,
122     IncidentID INT,
123     AadhaarNumber BIGINT,
124     RoleDescription TEXT,
125     AddedByOfficerID INT,
126     PRIMARY KEY(SuspectID, IncidentID),
127     CONSTRAINT fk_SuspectIncidents_Suspects
128     FOREIGN KEY(SuspectID)
129     REFERENCES Suspects(SuspectID),
130     CONSTRAINT fk_SuspectIncidents_Incidents
131     FOREIGN KEY(IncidentID)
132     REFERENCES Incidents(IncidentID),
133     CONSTRAINT fk_SuspectIncidentsOfficers
134     FOREIGN KEY(AddedByOfficerID)
135     REFERENCES Officers(OfficerID)
136 );
```

#CARS_InsertionOfData.sql

```

database > # CARS_InsertionOfData.sql

-- Inserting data in Law Enforcement Agencies table
1  INSERT INTO LawEnforcementAgencies (AgencyName, Jurisdiction, EmailAddress) VALUES
2  ('Mumbai Police', 'Mumbai', 'mumbai@police.maha.gov.in'),
3  ('Delhi Police', 'Delhi', 'delhipolice@delhi.gov.in'),
4  ('Bengaluru City Police', 'Bengaluru', 'bcp@kar.gov.in'),
5  ('Hyderabad Police', 'Hyderabad', 'hydpolice@telangana.gov.in'),
6  ('Chennai City Police', 'Chennai', 'chennai@police@tn.gov.in'),
7  ('Kolkata Police', 'Kolkata', 'kolkatapolice@wb.gov.in'),
8  ('Pune City Police', 'Pune', 'pune@police.maha.gov.in'),
9  ('Ahmedabad Police', 'Ahmedabad', 'ahdpolice@gujarat.gov.in'),
10 ('Jaipur Police', 'Jaipur', 'jaipur@police@rajasthan.gov.in'),
11 ('Lucknow Police', 'Lucknow', 'lucknow@police@up.gov.in'),
12 ('Chandigarh Police', 'Chandigarh', 'chdpolice@chd.gov.in'),
13 ('Kochi City Police', 'Kochi', 'kochipolice@kerala.gov.in'),
14 ('Bhopal Police', 'Bhopal', 'bhopalpolice@mp.gov.in'),
15 ('Patna Police', 'Patna', 'patnapolice@bihar.gov.in'),
16 ('Surat City Police', 'Surat', 'surat@police@gujarat.gov.in'),
17 ('Nagpur Police', 'Nagpur', 'nagpur@police@maha.gov.in'),
18 ('Indore Police', 'Indore', 'indore@police@mp.gov.in'),
19 ('Thane Police', 'Thane', 'thane@police@maha.gov.in'),
20 ('Ghaziabad Police', 'Ghaziabad', 'ghaziabad@police@mp.gov.in'),
21 ('Coimbatore Police', 'Coimbatore', 'coimbatore@police@tn.gov.in'),
22

-- Inserting data in Officers table
23 INSERT INTO Officers (FirstName, LastName, DateOfBirth, Gender, BadgeNumber, Ranking, PostingCity, PostingState, ServiceJoiningDate, ResidentialAddress, ContactNumber, AgencyID) VALUES
24 ('Rajesh', 'Kumar', '1988-05-15', 'Male', 'MH-4521', 'Inspector', 'Mumbai', 'Maharashtra', '2005-07-10', 'Flat 12, Shivaji Nagar, Mumbai', '0876543210, 580'),
25 ('Priya', 'Sharma', '1985-08-22', 'Female', 'TN-7854', 'Sub-Inspector', 'Delhi', 'Delhi', '2010-05-15', 'H.No. 45, RK Puram, Delhi', '8765432100, 581'),
26 ('Vikram', 'Singh', '1978-11-30', 'Male', 'KA-3214', 'Deputy Commissioner', 'Bengaluru', 'Karnataka', '2000-12-05', 'No. 56, Koramangala, Bengaluru', '7654321008, 582'),
27 ('Anjali', 'Patel', '1982-04-18', 'Female', 'TS-6547', 'Inspector', 'Hyderabad', 'Telangana', '2006-09-28', 'Flat 34, Banjara Hills, Hyderabad', '6543210087, 583'),
28 ('Arun', 'Iyer', '1987-07-25', 'Male', 'TN-0874', 'Sub-Inspector', 'Chennai', 'Tamil Nadu', '2012-05-12', 'No. 78, T Nagar, Chennai', '0432100876, 584'),
29 ('Meera', 'Banerjee', '1975-02-14', 'Female', 'WB-3608', 'Inspector', 'Kolkata', 'West Bengal', '1998-11-08', '128, Park Street, Kolkata', '8521008765, 585'),
30 ('Sanjay', 'Deshmukh', '1983-09-09', 'Male', 'MH-8521', 'Sub-Inspector', 'Pune', 'Maharashtra', '2008-06-30', 'Flat 5, Kothrud, Pune', '7210087654, 586'),
31 ('Wohu', 'Gupta', '1988-12-03', 'Female', 'GJ-7412', 'Constable', 'Ahmedabad', 'Gujarat', '2014-04-22', 'H.No. 32, Navrangpura, Ahmedabad', '6100876543, 587'),
32 ('Rahul', 'Sharma', '1979-06-28', 'Male', 'RJ-0632', 'Inspector', 'Jaipur', 'Rajasthan', '2004-08-17', 'No. 15, Malviya Nagar, Jaipur', '5008765432, 588'),
33 ('Sunita', 'Yadav', '1984-03-19', 'Female', 'UP-2587', 'Sub-Inspector', 'Lucknow', 'Uttar Pradesh', '2009-10-05', 'H.No. 7, Gomti Nagar, Lucknow', '4087654321, 589'),
34 ('Amit', 'Khanna', '1981-07-22', 'Male', 'CH-1478', 'Inspector', 'Chandigarh', 'Chandigarh', '2011-01-14', 'Sector 28B, Chandigarh', '3876543210, 581'),
35 ('Deepa', 'Nair', '1986-10-11', 'Female', 'KL-3608', 'Sub-Inspector', 'Kochi', 'Kerala', '2013-09-28', 'Flat 8, Marine Drive, Kochi', '2765432100, 511'),
36 ('Vijay', 'Malhotra', '1977-04-05', 'Male', 'MP-6541', 'Deputy Commissioner', 'Bhopal', 'Madhya Pradesh', '1999-12-15', 'No. 42, Azera Colony, Bhopal', '1654321008, 512'),
37 ('Pooja', 'Verma', '1989-01-30', 'Female', 'BR-0875', 'Constable', 'Patna', 'Bihar', '2016-07-08', 'No. 25, Patliputra Colony, Patna', '0543210087, 513'),
38 ('Ramesh', 'Joshi', '1988-08-17', 'Male', 'GJ-3214', 'Inspector', 'Surat', 'Gujarat', '2005-11-25', 'Flat 16, Athwa Lines, Surat', '8432100876, 514'),
39 ('Smita', 'Roddy', '1983-05-14', 'Female', 'MH-7852', 'Sub-Inspector', 'Mumbai', 'Maharashtra', '2008-02-18', 'Flat 9, Andheri East, Mumbai', '7521008765, 580'),
40 ('Alok', 'Misra', '1976-12-08', 'Male', 'DL-4560', 'Inspector', 'Delhi', 'Delhi', '2001-04-30', 'H.No. 34, Saket, Delhi', '6210087654, 581'),
41 ('Kavita', 'Singh', '1987-09-21', 'Female', 'KA-1478', 'Constable', 'Bengaluru', 'Karnataka', '2014-08-12', 'No. 51, Indiranagar, Bengaluru', '5100876543, 582'),
42 ('Prakash', 'Rao', '1982-02-25', 'Male', 'TS-0632', 'Sub-Inspector', 'Hyderabad', 'Telangana', '2007-06-19', 'Flat 22, Jubilee Hills, Hyderabad', '4008765432, 583),
43 ('Anita', 'Manon', '1985-11-13', 'Female', 'TN-2587', 'Inspector', 'Chennai', 'Tamil Nadu', '2010-05-07', 'No. 67, Adyar, Chennai', '5087654321, 584),
44 ('Rajiv', 'Bose', '1978-07-04', 'Male', 'WB-7412', 'Deputy Commissioner', 'Kolkata', 'West Bengal', '2003-10-22', '15C, Salt Lake, Kolkata', '2876543210, 585),
45 ('Shweta', 'Patwari', '1989-04-16', 'Female', 'MH-3608', 'Constable', 'Pune', 'Maharashtra', '2017-01-09', 'Flat 3, Wakad, Pune', '1765432100, 586'),
46 ('Manoj', 'Patel', '1981-05-20', 'Male', 'GJ-6540', 'Sub-Inspector', 'Ahmedabad', 'Gujarat', '2008-05-14', 'H.No. 18, Satellite, Ahmedabad', '8654321008, 587'),
47 ('Divya', 'Sharma', '1984-03-07', 'Female', 'RJ-0873', 'Inspector', 'Jaipur', 'Rajasthan', '2009-09-26', 'No. 24, Vaishali Nagar, Jaipur', '0543210087, 588'),
48 ('Vivek', 'Tripathi', '1979-08-12', 'Male', 'UP-3214', 'Deputy Commissioner', 'Lucknow', 'Uttar Pradesh', '2004-12-03', 'H.No. 9, Alambagh, Lucknow', '8432100876, 589'),
49 ('Anjali', 'Mehta', '1986-01-24', 'Female', 'CH-7852', 'Sub-Inspector', 'Chandigarh', 'Chandigarh', '2011-07-17', 'Sector 35C, Chandigarh', '7321008765, 516),
50 ('Sunil', 'Nair', '1988-06-18', 'Male', 'KL-4569', 'Inspector', 'Kochi', 'Kerala', '2005-11-08', 'Flat 11, Panampilly Nagar, Kochi', '6210087654, 511),
51 ('Preeti', 'Verma', '1983-09-05', 'Female', 'MP-1478', 'Sub-Inspector', 'Bhopal', 'Madhya Pradesh', '2008-04-21', 'No. 33, Shahpura, Bhopal', '5100876543, 512),
52 ('Rakesh', 'Yadav', '1977-02-28', 'Male', 'BR-0632', 'Inspector', 'Patna', 'Bihar', '2002-08-15', 'H.No. 5, Danapur, Patna', '4098765432, 513),
53 ('Sonia', ' Kapoor', '1988-07-10', 'Female', 'GJ-2587', 'Constable', 'Surat', 'Gujarat', '2015-02-11', 'Flat 7, Vosu, Surat', '5087654321, 514),
54 ('Nitin', 'Shah', '1982-09-28', 'Male', 'MH-8523', 'Inspector', 'Nagpur', 'Maharashtra', '2007-08-14', 'Flat 5, Sicihduli, Nagpur', '9876543281, 515),
55 ('Ananya', 'Desai', '1987-04-15', 'Female', 'MP-7531', 'Sub-Inspector', 'Indore', 'Madhya Pradesh', '2012-11-22', 'No. 12, Vijay Nagar, Indore', '8765432100, 516),
56 ('Vikas', 'Malhotra', '1988-12-03', 'Male', 'MH-0635', 'Inspector', 'Thane', 'Maharashtra', '2006-05-10', 'Flat 8, Ghodbhunder Road, Thane', '7654321007, 517),
57 ('Rashmi', 'Saxena', '1985-07-21', 'Female', 'UP-8524', 'Sub-Inspector', 'Ghaziabad', 'Uttar Pradesh', '2010-09-30', 'H.No. 34, Raj Nagar, Ghaziabad', '6543210086, 518),
58 ('Sanjeev', 'Iyer', '1979-02-14', 'Male', 'TN-7418', 'Inspector', 'Coimbatore', 'Tamil Nadu', '2004-10-25', 'No. 56, RS Puram, Coimbatore', '5432100875, 519);

-- Inserting data in Incidents table
59 INSERT INTO Incidents (IncidentType, IncidentDate, Area, City, Description, Status, OfficerID) VALUES
60 ('Robbery', '2023-01-15', 'Andheri West', 'Mumbai', 'Armed robbery at jewelry store', 'Closed', 1),
61 ('Burglary', '2023-02-03', 'Saket', 'Delhi', 'Residential burglary during daytime', 'Under Investigation', 2),
62 ('Cyber Crime', '2023-02-10', 'Koramangala', 'Bengaluru', 'Online financial fraud', 'Open', 3),
63 ('Assault', '2023-02-18', 'Banjara Hills', 'Hyderabad', 'Bar fight leading to serious injuries', 'Closed', 4),
64 ('Theft', '2023-03-05', 'T Nagar', 'Chennai', 'Pickpocketing in crowded market', 'Under Investigation', 5),
65 ('Homicide', '2023-03-12', 'Park Street', 'Kolkata', 'Domestic dispute turned violent', 'Under Investigation', 6),
66 ('Vehicle Theft', '2023-03-28', 'Kothrud', 'Pune', 'Motorcycle stolen from parking lot', 'Open', 7),
67 ('Fraud', '2023-04-02', 'Navrangpura', 'Ahmedabad', 'Real estate scam', 'Closed', 8),
68 ('Kidnapping', '2023-04-15', 'Malviya Nagar', 'Jaipur', 'Child abduction case', 'Under Investigation', 9),
69 ('Domestic Violence', '2023-04-22', 'Gomti Nagar', 'Lucknow', 'Spousal abuse complaint', 'Closed', 10),
70 ('Drug Offense', '2023-05-05', 'Sector 22', 'Chandigarh', 'Drug peddling bust', 'Closed', 11),
71 ('Sexual Assault', '2023-05-12', 'Marine Drive', 'Kochi', 'Molestation case', 'Under Investigation', 12),
72 ('Arson', '2023-05-28', 'Azaara Colony', 'Bhopal', 'Deliberate fire at commercial property', 'Open', 13),
73 ('Public Nuisance', '2023-06-03', 'Patliputra Colony', 'Patna', 'Loud party disturbance', 'Closed', 14),
74 ('Hit and Run', '2023-06-10', 'Athwa Lines', 'Surat', 'Pedestrian hit by speeding vehicle', 'Under Investigation', 15),
75 ('Vandalism', '2023-06-18', 'Andheri East', 'Mumbai', 'Graffiti on public property', 'Open', 16),
76 ('Identity Theft', '2023-07-02', 'Saket', 'Delhi', 'Credit card fraud', 'Under Investigation', 17),
77 ('Harassment', '2023-07-15', 'Indiranagar', 'Bengaluru', 'Workplace harassment complaint', 'Closed', 18),
78 ('Bribery', '2023-07-22', 'Jubilee Hills', 'Hyderabad', 'Government official caught taking bribe', 'Under Investigation', 19),
79 ('Child Abuse', '2023-08-03', 'Alambagh', 'Lucknow', 'School teacher accused of abuse', 'Open', 20),
80 ('Eve Teasing', '2023-08-12', 'Salt Lake', 'Kolkata', 'Street harassment complaint', 'Closed', 21),
81 ('Illegal Gambling', '2023-08-28', 'Wakad', 'Pune', 'Underground gambling den raid', 'Under Investigation', 22),
82 ('Forgery', '2023-09-03', 'Satellite', 'Ahmedabad', 'Fake document racket', 'Open', 23),
83 ('Extortion', '2023-09-10', 'Vaishali Nagar', 'Jaipur', 'Shopkeepers threatened for money', 'Under Investigation', 24),
84 ('Trespassing', '2023-09-18', 'Alambagh', 'Lucknow', 'Unauthorized entry into private property', 'Closed', 25),
85 ('Robbery', '2023-06-05', 'Marine Lines', 'Mumbai', 'Chain snatching incident near station', 'Under Investigation', 1),
86 ('Cyber Crime', '2023-06-12', 'Gurgaon', 'Delhi-NCR', 'Online job scam defrauding 50 people', 'Open', 2),
87 ('Domestic Violence', '2023-06-18', 'Whitefield', 'Bengaluru', 'Wife assaulted by husband', 'Under Investigation', 3),
88 ('Theft', '2023-06-25', 'Banjara Hills', 'Hyderabad', 'Mobile phone theft in mall', 'Open', 4),
89 ('Fraud', '2023-06-28', 'T Nagar', 'Chennai', 'Fake lottery scam targeting seniors', 'Under Investigation', 5);

```

```

94    -- Inserting data in Victims table
95    INSERT INTO Victims (FirstName, LastName, DateOfBirth, Gender, ResidentialAddress, ContactNumber, AadhaarNumber, IncidentID) VALUES
96    ('Aarav', 'Shah', '1975-08-12', 'Male', 'Shop No.5, Juhu Road, Andheri West, Mumbai', 0876543210, 123456789012, 1),
97    ('Neha', 'Kapoor', '1982-03-25', 'Female', 'Flat 302, Saket Apartments, Delhi', 8765432100, 234567890123, 2),
98    ('Rahul', 'Iyer', '1990-11-08', 'Male', 'No. 45, 5th Cross, Koramangala, Bengaluru', 7654321008, 345678901234, 3),
99    ('Priya', 'Reddy', '1988-07-14', 'Female', 'Flat 8, Banjara Residency, Hyderabad', 6543210087, 456789012345, 4),
100   ('Vikram', 'Menon', '1970-04-30', 'Male', 'No. 12, T Nagar Main Road, Chennai', 8432100876, 567890123456, 5),
101   ('Ananya', 'Banerjee', '1995-01-18', 'Female', '12B, Park Street, Kolkata', 8321008765, 678901234567, 6),
102   ('Rohan', 'Deshpande', '1987-09-22', 'Male', 'Flat 3, Kothrud Society, Pune', 7210087654, 780812345678, 7),
103   ('Moza', 'Patel', '1973-12-05', 'Female', 'H.No. 52, Navrangpura, Ahmedabad', 6100876543, 808123456780, 8),
104   ('Arjun', 'Sharma', '2005-06-15', 'Male', 'No. 15, Malviya Nagar, Jaipur', 5008765432, 981234567890, 9),
105   ('Sunita', 'Yadav', '1988-02-28', 'Female', 'H.No. 7, Gomti Nagar, Lucknow', 4087654321, 112233445566, 10),
106   ('Amit', 'Khanna', '1992-10-11', 'Male', 'Sector 22B, Chandigarh', 3876543210, 223344556677, 11),
107   ('Deopika', 'Naik', '1984-05-19', 'Female', 'Flat 8, Maxine Drive, Kochi', 2765432109, 334455667788, 12),
108   ('Vijay', 'Malhotra', '1968-08-24', 'Male', 'No. 42, Aroha Colony, Bhopal', 1654321008, 445566778809, 13),
109   ('Pooja', 'Verma', '1998-03-07', 'Female', 'H.No. 23, Patliputra Colony, Patna', 0543210087, 556677880908, 14),
110   ('Ramesh', 'Joshi', '1977-11-15', 'Male', 'Flat 16, Athwa Lines, Surat', 8432100876, 667788000811, 15),
111   ('Smita', 'Reddy', '1989-07-30', 'Female', 'Flat 9, Andheri East, Mumbai', 7321008765, 778800081122, 16),
112   ('Alok', 'Mishra', '1991-04-02', 'Male', 'H.No. 34, Saket, Delhi', 6210087654, 889000112233, 17),
113   ('Kavita', 'Singh', '1985-12-18', 'Female', 'No. 51, Indiranagar, Bengaluru', 5100876543, 900011223344, 18),
114   ('Prakash', 'Rao', '1976-09-21', 'Male', 'Flat 22, Jubilee Hills, Hyderabad', 4808765432, 112233445500, 19),
115   ('Anita', 'Menon', '1993-02-14', 'Female', 'No. 67, Adyar, Chennai', 3087654321, 223344556611, 20),
116   ('Rajiv', 'Mehta', '1985-05-17', 'Male', '15C, Salt Lake, Kolkata', 2876543210, 334455667722, 21),
117   ('Shweta', 'Pawar', '1985-00-23', 'Female', 'Flat 3, Wakad, Pune', 1765432109, 445566778833, 22),
118   ('Manoj', 'Patel', '1988-12-07', 'Male', 'H.No. 18, Satellite, Ahmedabad', 0654321008, 556677880944, 23),
119   ('Divya', 'Sharma', '1987-04-11', 'Female', 'No. 24, Vaishali Nagar, Jaipur', 0543210088, 667788000855, 24),
120   ('Vivek', 'Trivedi', '1990-09-20', 'Male', 'H.No. 9, Alambagh, Lucknow', 8432100877, 778800081166, 25),
121   ('Anjali', 'Mehta', '1982-01-14', 'Female', 'Sector 35C, Chandigarh', 7321008766, 880000112277, 26),
122   ('Sunil', 'Nair', '1970-07-03', 'Male', 'Flat 11, Panampilly Nagar, Kochi', 6210087655, 900011223388, 27),
123   ('Preeti', 'Verma', '1986-10-19', 'Female', 'No. 33, Shahpura, Bhopal', 5100876544, 161810181018, 28),
124   ('Rakesh', 'Yadav', '1975-03-26', 'Male', 'H.No. 5, Danapur, Patna', 4087654333, 112211221122, 29),
125   ('Sonia', 'Kapoor', '1988-12-09', 'Female', 'Flat 7, Vesu, Surat', 3987654322, 223322332233, 30),
126
127  -- Inserting data in Suspects table
128  INSERT INTO Suspects (FirstName, LastName, DateOfBirth, Gender, ResidentialAddress, ContactNumber, AadhaarNumber) VALUES
129  ('Vijay', 'Malhotra', '1988-05-15', 'Male', 'Room No. 12, Chawl No.5, Dhaxavi, Mumbai', 0876123458, 123412341234),
130  ('Raju', 'Khan', '1985-08-22', 'Male', 'H.No. 45, Seelampur, Delhi', 8765234561, 234523452345),
131  ('Suzesh', 'Kumar', '1978-11-30', 'Male', 'No. 56, Shivajinagar, Bengaluru', 7656345672, 345634563456),
132  ('Ramesh', 'Patel', '1982-04-18', 'Male', 'Flat 34, Old City, Hyderabad', 6547456783, 456745674567),
133  ('Mohan', 'Iyer', '1987-07-25', 'Male', 'No. 78, Wasermanpet, Chennai', 9438567894, 567856785678),
134  ('Babu', 'Banerjee', '1975-02-14', 'Male', '12B, Howrah, Kolkata', 8329678085, 678067806780),
135  ('Sanju', 'Deshmukh', '1983-00-09', 'Male', 'Flat 5, Pimpri, Pune', 7218780016, 780878087809),
136  ('Ravi', 'Gupta', '1988-12-03', 'Male', 'H.No. 32, Naroda, Ahmedabad', 6181008127, 808100810001),
137  ('Mahesh', 'Sharma', '1970-06-28', 'Male', 'No. 15, Jhotwara, Jaipur', 5002081238, 081208120012),
138  ('Sunil', 'Yadav', '1984-03-19', 'Male', 'H.No. 7, Chinhat, Lucknow', 4983812349, 112211221122),
139  ('Aakash', 'Khanna', '1981-07-22', 'Male', 'Sector 22D, Chandigarh', 3874123458, 223322332233),
140  ('Deepak', 'Naik', '1986-10-11', 'Male', 'Flat 8, Fort Kochi, Kochi', 2765234561, 334433443344),
141  ('Vinod', 'Malhotra', '1977-04-05', 'Male', 'No. 42, Karond, Bhopal', 1656345672, 445544554455),
142  ('Pappu', 'Verma', '1980-01-30', 'Male', 'H.No. 23, Phulwari Sharif, Patna', 0547456783, 556655665556),
143  ('Rajesh', 'Joshi', '1988-08-17', 'Male', 'Flat 16, Varachha, Surat', 8438567804, 667766776677),
144  ('Suman', 'Reddy', '1983-05-14', 'Female', 'Flat 9, Govandi, Mumbai', 7329678085, 778877887788),
145  ('Alok', 'Mishra', '1976-12-08', 'Male', 'H.No. 34, Narela, Delhi', 6210087656, 880088000800),
146  ('Kiran', 'Singh', '1987-09-21', 'Female', 'No. 51, Yeshwantpur, Bengaluru', 5181008127, 008000000000),
147  ('Prakash', 'Rao', '1982-02-25', 'Male', 'Flat 22, Secunderabad, Hyderabad', 4802008128, 181010181018),
148  ('Anil', 'Menon', '1985-11-13', 'Male', 'No. 67, Tambaram, Chennai', 3083812340, 282828282828),
149  ('Rahul', 'Bose', '1979-08-14', 'Male', '15D, Salt Lake, Kolkata', 2874123458, 383803030308),
150  ('Shiva', 'Pawar', '1984-06-27', 'Male', 'Flat 4, Wakad, Pune', 1765234561, 484848484848),
151  ('Mohan', 'Patel', '1981-11-05', 'Male', 'H.No. 19, Satellite, Ahmedabad', 0656345672, 585958505050),
152  ('Dinesh', 'Sharma', '1986-03-18', 'Male', 'No. 25, Vaishali Nagar, Jaipur', 0547456784, 668666866666),
153  ('Vikas', 'Trivedi', '1991-00-22', 'Male', 'H.No. 18, Alambagh, Lucknow', 8438567805, 787787787878),
154  ('Ramesh', 'Pillai', '1982-07-18', 'Male', 'Room No. 8, Chawl No.3, Dhaxavi, Mumbai', 0876123451, 888888888888),
155  ('Sanjay', 'Verma', '1987-04-22', 'Male', 'H.No. 12, Laxmi Nagar, Delhi', 8765234562, 080800000800),
156  ('Anita', 'Desai', '1990-11-15', 'Female', 'No. 34, MG Road, Bengaluru', 7656345673, 121212121212),
157  ('Vishal', 'Joshi', '1990-02-28', 'Male', 'Flat 12, Begumpet, Hyderabad', 6547456784, 232323232323),
158  ('Pooja', 'Reddy', '1988-00-05', 'Female', 'No. 45, Anna Nagar, Chennai', 9438567805, 343434343434),
159  ('Rakesh', 'Malhotra', '1983-06-12', 'Male', '22B, Park Street, Kolkata', 8329678086, 454545454545),
160  ('Sunita', 'Sharma', '1986-03-25', 'Female', 'Flat 7, Kothrud, Pune', 7218780017, 565656565656),
161
162  -- Inserting data in Evidence table
163  INSERT INTO Evidence (EvidenceName, Description, LocationFound, IncidentID) VALUES
164  ('Knife', '6-inch blade with wooden handle', 'Near jewelry store counter', 1),
165  ('CCTV Footage', 'Digital recording of burglary', 'Building security room', 2),
166  ('Laptop', 'Used for online fraud', 'Suspect's residence', 3),
167  ('Broken Bottle', 'Used in assault with blood traces', 'Bar floor', 4),
168  ('Wallet', 'Victim's stolen wallet found discarded', 'Near market dustbin', 5),
169  ('Bloody Knife', 'Murder weapon with fingerprints', 'Kitchen sink', 6),
170  ('Helmet', 'Left at vehicle theft scene', 'Parking lot corner', 7),
171  ('Documents', 'Fake property papers', 'Suspect's office', 8),
172  ('Child's Toy', 'Found near abduction site', 'Park bench', 9),
173  ('Medical Report', 'Victim injury documentation', 'Hospital records', 10),
174  ('Drug Packets', 'Containing narcotics', 'Suspect's car trunk', 11),
175  ('Clothing', 'Torn dress of victim', 'Alleyway', 12),
176  ('Gas Can', 'Used to start fire', 'Behind burned building', 13),
177  ('Audio Recording', 'Noise complaint evidence', 'Neighbour's phone', 14),
178  ('Paint Scraps', 'From hit-and-run vehicle', 'Victim's clothing', 15),
179  ('Spray Cans', 'Used for graffiti', 'Abandoned near scene', 16),
180  ('Credit Card', 'Cloned card found', 'ATM vestibule', 17),
181  ('Emails', 'Harassment evidence', 'Victim's work computer', 18),
182  ('Cash', 'Bribe money marked', 'Official's drawer', 19),
183  ('Text Messages', 'Threatening messages', 'Victim's phone', 20),
184  ('Gold Chain', 'Snatched from victim', 'Pawn shop recovery', 26),
185  ('Laptop', 'Used for scam operations', 'Cyber cafe raid', 27),
186  ('Belt', 'Used in domestic assault', 'Victim's residence', 28),
187  ('Mobile Phone', 'Stolen device with fingerprints', 'Flea market recovery', 29),
188  ('Lottery Tickets', 'Fake winning tickets', 'Suspect's residence', 30),
189
190  -- Inserting data in Reports table

```

```

108
109 -- Inserting data in Reports table
110 INSERT INTO Reports (IncidentID, ReportingOfficerID, ReportDate, ReportDetails, STATUS) VALUES
111 (1, 1, '2023-01-16', 'Robbery at jewelry store with estimated loss of ₹25 lakhs. One suspect apprehended.', 'Finalized'),
112 (2, 2, '2023-02-04', 'Daytime burglary with stolen valuables worth ₹5 lakhs. Investigation ongoing.', 'Draft'),
113 (3, 3, '2023-02-11', 'Online fraud case involving ₹12 lakhs transferred to fake accounts.', 'Finalized'),
114 (4, 4, '2023-02-19', 'Bar fight resulting in serious injuries to two patrons. Case closed after settlement.', 'Finalized'),
115 (5, 5, '2023-03-06', 'Pickpocketing incident in crowded market. No suspects identified yet.', 'Draft'),
116 (6, 6, '2023-03-15', 'Domestic dispute turned fatal. Husband in custody.', 'Finalized'),
117 (7, 7, '2023-03-21', 'Motorcycle theft from residential parking. CCTV footage being analyzed.', 'Draft'),
118 (8, 8, '2023-04-05', 'Real estate scam involving fake property documents. Three suspects arrested.', 'Finalized'),
119 (9, 9, '2023-04-16', 'Child abduction case. AMBER alert issued. Child recovered safely.', 'Finalized'),
120 (10, 10, '2023-04-23', 'Domestic violence complaint. Husband charged under IPC 408A.', 'Finalized'),
121 (11, 11, '2023-05-06', 'Drug peddling bust with seizure of 2kg narcotics. Two arrested.', 'Finalized'),
122 (12, 12, '2023-05-15', 'Molestation case reported by college student. Suspect identified.', 'Draft'),
123 (13, 13, '2023-05-21', 'Deliberate fire at commercial property causing ₹50 lakhs damage.', 'Draft'),
124 (14, 14, '2023-06-04', 'Noise complaint against late-night party. Warning issued.', 'Finalized'),
125 (15, 15, '2023-06-11', 'Hit-and-run case with pedestrian critically injured. Vehicle identified.', 'Draft'),
126 (16, 16, '2023-06-19', 'Vandalism of public property with graffiti. Local gang suspected.', 'Draft'),
127 (17, 17, '2023-07-03', 'Credit card fraud involving ₹3.2 lakhs unauthorized transactions.', 'Draft'),
128 (18, 18, '2023-07-16', 'Workplace harassment complaint by female employee. HR involved.', 'Finalized'),
129 (19, 19, '2023-07-23', 'Bribery case against municipal officer caught red-handed.', 'Finalized'),
130 (20, 20, '2023-08-06', 'Child abuse complaint against school teacher. Investigation ongoing.', 'Draft'),
131 (21, 21, '2023-08-06', 'Chain snatching incident near Marine Lines station. Victim sustained minor injuries.', 'Finalized'),
132 (22, 22, '2023-08-15', 'Online job scam defrauding 50 people of approximately ₹25 lakhs collectively.', 'Draft'),
133 (23, 23, '2023-08-19', 'Domestic violence case with victim requiring hospitalization. Husband absconding.', 'Finalized'),
134 (24, 24, '2023-08-26', 'Mobile phone theft reported in Banjara Hills mall. CCTV footage available.', 'Draft'),
135 (25, 25, '2023-08-29', 'Fake lottery scam targeting senior citizens. Three suspects identified.', 'Draft');

136
137 -- Inserting data in Criminals table
138 INSERT INTO Criminals (IncidentID, SuspectID, AadhaarNumber, PunishmentDetails) VALUES
139 (1, 1000, 123412341234, '5 years imprisonment under IPC 392'),
140 (2, 1001, 456745674567, '2 years imprisonment under IPC 326'),
141 (3, 1005, 678906789067890, 'Life imprisonment under IPC 382'),
142 (4, 1007, 890180818081, '7 years imprisonment under IPC 428'),
143 (5, 1008, 981208120812, '10 years imprisonment under IPC 363'),
144 (6, 1009, 112211221122, '3 years imprisonment under IPC 408A'),
145 (7, 1010, 223322332233, '10 years rigorous imprisonment under NDPS Act'),
146 (8, 1011, 556655665566, 'Fine of ₹10,000 under Noise Pollution Rules'),
147 (9, 1012, 880988098809, '2 years imprisonment under IPC 354'),
148 (10, 1013, 999999999999, '5 years imprisonment under Prevention of Corruption Act'),
149 (11, 1014, 234523452345, '3 years imprisonment under IPC 454'),
150 (12, 1015, 345634563456, '4 years imprisonment under IT Act 2000'),
151 (13, 1016, 567856785678, '1 year imprisonment under IPC 370'),
152 (14, 1017, 667766776677, '7 years imprisonment under IPC 354'),
153 (15, 1018, 778877887788, '5 years imprisonment under IPC 270/338'),
154 (16, 1019, 778877887788, '6 months imprisonment under IPC 427'),
155 (17, 1020, 809988998899, '3 years imprisonment under IT Act 66C'),
156 (18, 1021, 908090809080, '7 years imprisonment under POCSO Act'),
157 (19, 1022, 181818181818, '1 year imprisonment under IPC 294'),
158 (20, 1023, 282828282828, '2 years imprisonment under Public Gambling Act'),
159 (21, 1024, 383838383838, '3 years imprisonment under IPC 392 (pending appeal)'),
160 (22, 1025, 484848484848, '5 years imprisonment under IT Act 66D (under trial)');
161
162
163 -- Inserting data in Users table
164 INSERT INTO Users (Username, Password, Role, OfficerID) VALUES
165 ('rajesh.kumar', 'mumbai@123', 'Officer', 1),
166 ('priya.sharma', 'delhi@456', 'Officer', 2),
167 ('vikram.singh', 'bengaluru@789', 'Admin', 3),
168 ('anjali.patel', 'hyderabad@321', 'Officer', 4),
169 ('arun.iyer', 'chennai@654', 'Officer', 5),
170 ('moora.banerjee', 'kolkata@087', 'Officer', 6),
171 ('sanjay.deshmukh', 'pune@123', 'Officer', 7),
172 ('neha.gupta', 'ahmedabad@456', 'Officer', 8),
173 ('rahul.sharma', 'jaipur@789', 'Officer', 9),
174 ('sunita.yadav', 'lucknow@321', 'Officer', 10),
175 ('amit.khanna', 'chandigarh@654', 'Officer', 11),
176 ('deepa.nair', 'kochia@987', 'Officer', 12),
177 ('vijay.malhotra', 'bhopal@123', 'Admin', 13),
178 ('pooja.verma', 'patna@456', 'Officer', 14),
179 ('ramesh.joshi', 'surat@789', 'Officer', 15),
180 ('smita.xddy', 'mumbai@321', 'Officer', 16),
181 ('alok.mishra', 'delhi@654', 'Officer', 17),
182 ('kavita.singh', 'bengaluru@087', 'Officer', 18),
183 ('prakash.rao', 'hyderabad@123', 'Officer', 19),
184 ('anita.monon', 'chennai@456', 'Officer', 20),
185 ('rajiv.bose', 'kolkata@789', 'Admin', 21),
186 ('shweta.pawar', 'pune@321', 'Officer', 22),
187 ('manoj.patel', 'ahmedabad@654', 'Officer', 23),
188 ('divya.sharma', 'jaipur@087', 'Officer', 24),
189 ('vivek.trivedi', 'lucknow@123', 'Officer', 25),
190 ('anjali.mohta', 'chandigarh@456', 'Officer', 26),
191 ('sunil.nair', 'kochi@789', 'Officer', 27),
192 ('preeti.verma', 'bhopal@321', 'Officer', 28),
193 ('rakesh.yadav', 'patna@654', 'Officer', 29),
194 ('sonia.kapoor', 'surat@087', 'Officer', 30),
195 ('nitin.shah', 'nagpur@123', 'Admin', 31),
196 ('ananya.desai', 'indore@456', 'Officer', 32),
197 ('vikas.malhotra', 'chennai@789', 'Officer', 33),
198 ('rashmi.saxena', 'ghaziabad@321', 'Officer', 34),
199 ('sanjeev.iyer', 'coimbatore@654', 'Officer', 35);
200
201

```

```

database > CARS_InsertionOfData.sql > ...
-- Inserting data in SuspectIncidents table
282 INSERT INTO SuspectIncidents (SuspectID, IncidentID, AadhaarNumber, RoleDescription, AddedByOfficerID) VALUES
284 (1000, 1, 123412341234, 'Primary robber with weapon', 1),
285 (1001, 2, 234523452345, 'Burglar caught on CCTV', 2),
286 (1002, 3, 345634563456, 'Created fake investment website', 3),
287 (1003, 4, 456745674567, 'Initiated bar fight with bottle', 4),
288 (1004, 5, 567856785678, 'Pickpocket working in team', 5),
289 (1005, 6, 678967896789, 'Husband who committed murder', 6),
290 (1006, 7, 789078907890, 'Stolen motorcycle found in possession', 7),
291 (1007, 8, 890189018901, 'Mastermind of real estate scam', 8),
292 (1008, 9, 901290129012, 'Abducted child from park', 9),
293 (1009, 10, 112211221122, 'Husband accused of domestic violence', 10),
294 (1010, 11, 223322332233, 'Drug supplier caught with narcotics', 11),
295 (1011, 12, 334433443344, 'Molested college student', 12),
296 (1012, 13, 445544554455, 'Set fire to business rival's shop', 13),
297 (1013, 14, 556655665566, 'Organized loud party violating norms', 14),
298 (1014, 15, 667766776677, 'Hit pedestrian while drunk driving', 15),
299 (1015, 16, 778877887788, 'Spray painted public walls', 16),
300 (1016, 17, 889988998899, 'Created cloned credit cards', 17),
301 (1017, 18, 990099009900, 'Sent inappropriate emails to colleague', 18),
302 (1018, 19, 101010101010, 'Accepted bribe for contract approval', 19),
303 (1019, 20, 202020202020, 'Physically abused students', 20),
304 (1020, 21, 303030303030, 'Eve teased college girls', 21),
305 (1021, 22, 404040404040, 'Ran illegal gambling operation', 22),
306 (1022, 23, 505050505050, 'Prepared fake documents', 23),
307 (1023, 24, 606060606060, 'Extorted money from shopkeepers', 24),
308 (1024, 25, 707070707070, 'Trespassed into private property', 25),
309 (1025, 26, 808080808080, 'Chain snatcher on motorcycle', 1),
310 (1026, 27, 909090909090, 'Operated fake job portal', 2),
311 (1027, 28, 121212121212, 'Assaulted wife with belt', 3),
312 (1028, 29, 232323232323, 'Stole mobile phones in mall', 4),
313 (1029, 30, 343434343434, 'Scammed seniors with fake lottery', 5),
314 (1030, 26, 454545454545, 'Accomplice in chain snatching', 1),
315 (1031, 27, 565656565656, 'Handled financial transactions for scam', 2);
316
317 SELECT * FROM Victims;
318 DELETE FROM Victims WHERE VictimID = 31;
319

```

Entity Folder

auth.py

```

CrimeAnalysisService.py CrimeAnalysisServiceImpl.py CARS_CreatingTables.sql CARS_InsertionOfData.sql auth.py
entity > auth.py > Authentication > login
  1  from getpass import getpass
  2  from util.DB_Connections import DBConnections
  3  from exceptions.user_defined_exceptions import AuthenticationError
  4
  5  class Authentication:
  6
  7      @staticmethod
  8      def login():
  9          print("\n==== Login ====")
 10          username = input("Username: ")
 11          password = getpass("Password: ")
 12
 13          try:
 14              connection = DBConnections.get_connection('db.properties')
 15              cursor = connection.cursor(dictionary=True)
 16
 17              query = "SELECT * FROM Users WHERE Username = %s AND Password = %s"
 18              cursor.execute(query, (username, password))
 19              user = cursor.fetchone()
 20
 21              cursor.close()
 22              connection.close()
 23
 24              if user:
 25                  return user
 26                  raise AuthenticationError("Invalid credentials! Please try again.")
 27
 28          except Exception as e:
 29              raise AuthenticationError(f"Login failed: {str(e)}")
 30
 31      @staticmethod
 32      def public_access():
 33          return {'UserID': 0, 'Role': 'Public', 'Username': 'public_user'}

```

Criminals.py

```
entity > 🏷 Criminals.py > 📄 Criminals
 1  class Criminals():
 2      def __init__(self, CriminalID = None, IncidentID = None, SuspectID = None, AadhaarNumber = None, PunishmentDetails = None):
 3          self.__CriminalID = CriminalID
 4          self.__IncidentID = IncidentID
 5          self.__SuspectID = SuspectID
 6          self.__AadhaarNumber = AadhaarNumber
 7          self.__PunishmentDetails = PunishmentDetails
 8
 9      @property
10      def criminalID(self):
11          return self.__CriminalID
12
13      @property
14      def incidentID(self):
15          return self.__IncidentID
16
17      @property
18      def suspectID(self):
19          return self.__SuspectID
20
21      @property
22      def aadhaarNumber(self):
23          return self.__AadhaarNumber
24
25      @property
26      def punishmentDetails(self):
27          return self.__PunishmentDetails
28
29      @CriminalID.setter
30      def get_criminalID(self, value):
31          self.__CriminalID = value
32
33      @incidentID.setter
34      def get_incidentID(self, value):
35          self.__IncidentID = value
36
37      @suspectID.setter
38      def get_suspectID(self, value):
39          self.__SuspectID = value
40
41      @aadhaarNumber.setter
42      def get_aadhaarNumber(self, value):
43          self.__AadhaarNumber = value
44
45      @punishmentDetails.setter
46      def get_punishmentDetails(self, value):
47          self.__PunishmentDetails = value
48
```

Evidence.py

```
entity > 🏷 Evidence.py > 📄 Evidence
 1  class Evidence():
 2      def __init__(self, EvidenceID = None, EvidenceName = None, Description = None, LocationFound = None, Incidents = None):
 3          self.__EvidenceID = EvidenceID
 4          self.__EvidenceName = EvidenceName
 5          self.__Description = Description
 6          self.__LocationFound = LocationFound
 7          self.__Incidents = Incidents
 8
 9      @property
10      def evidenceID(self):
11          return self.__EvidenceID
12
13      @property
14      def evidenceName(self):
15          return self.__EvidenceName
16
17      @property
18      def description(self):
19          return self.__Description
20
21      @property
22      def locationFound(self):
23          return self.__LocationFound
24
25      @property
26      def incidents(self):
27          return self.__Incidents
28
29      @evidenceID.setter
30      def get_evidenceID(self, value):
31          self.__EvidenceID = value
32
33      @evidenceName.setter
34      def get_evidenceName(self, value):
35          self.__EvidenceName = value
36
37      @description.setter
38      def get_description(self, value):
39          self.__Description = value
40
41      @locationFound.setter
42      def get_locationFound(self, value):
43          self.__LocationFound = value
```

Incidents.py

```
entity > Incidents.py > Incidents
1  class Incidents():
2      def __init__(self, IncidentID = None, IncidentType = None, IncidentDate = None, Area = None, City = None, Description = None, Status = None, OfficerID = None):
3          self.__IncidentID = IncidentID
4          self.__IncidentType = IncidentType
5          self.__IncidentDate = IncidentDate
6          self.__Area = Area
7          self.__City = City
8          self.__Description = Description
9          self.__Status = Status
10         self.__OfficerID = OfficerID
11
12     @property
13     def incidentID(self):
14         return self.__IncidentID
15
16     @property
17     def incidentType(self):
18         return self.__IncidentType
19
20     @property
21     def incidentDate(self):
22         return self.__IncidentDate
23
24     @property
25     def area(self):
26         return self.__Area
27
28     @property
29     def city(self):
30         return self.__City
31
32     @property
33     def description(self):
34         return self.__Description
35
36     @property
37     def status(self):
38         return self.__Status
39
40     @property
41     def officerID(self):
42         return self.__OfficerID
43
44     @incidentID.setter
45     def get_incidentID(self, value):
46         self.__IncidentID = value
47
48     @incidentType.setter
49     def get_incidentType(self, value):
50         self.__IncidentType = value
51
52     @incidentDate.setter
53     def get_incidentDate(self, value):
54         self.__IncidentDate = value
55
56     @area.setter
57     def get_area(self, value):
58         self.__Area = value
59
60     @city.setter
61     def get_city(self, value):
62         self.__City = value
63
64     @description.setter
65     def get_description(self, value):
66         self.__Description = value
67
68     @status.setter
69     def get_status(self, value):
70         self.__Status = value
71
72     @officerID.setter
73     def get_officerID(self, value):
74         self.__OfficerID = value
75
```

LawEnforcementAgencies.py

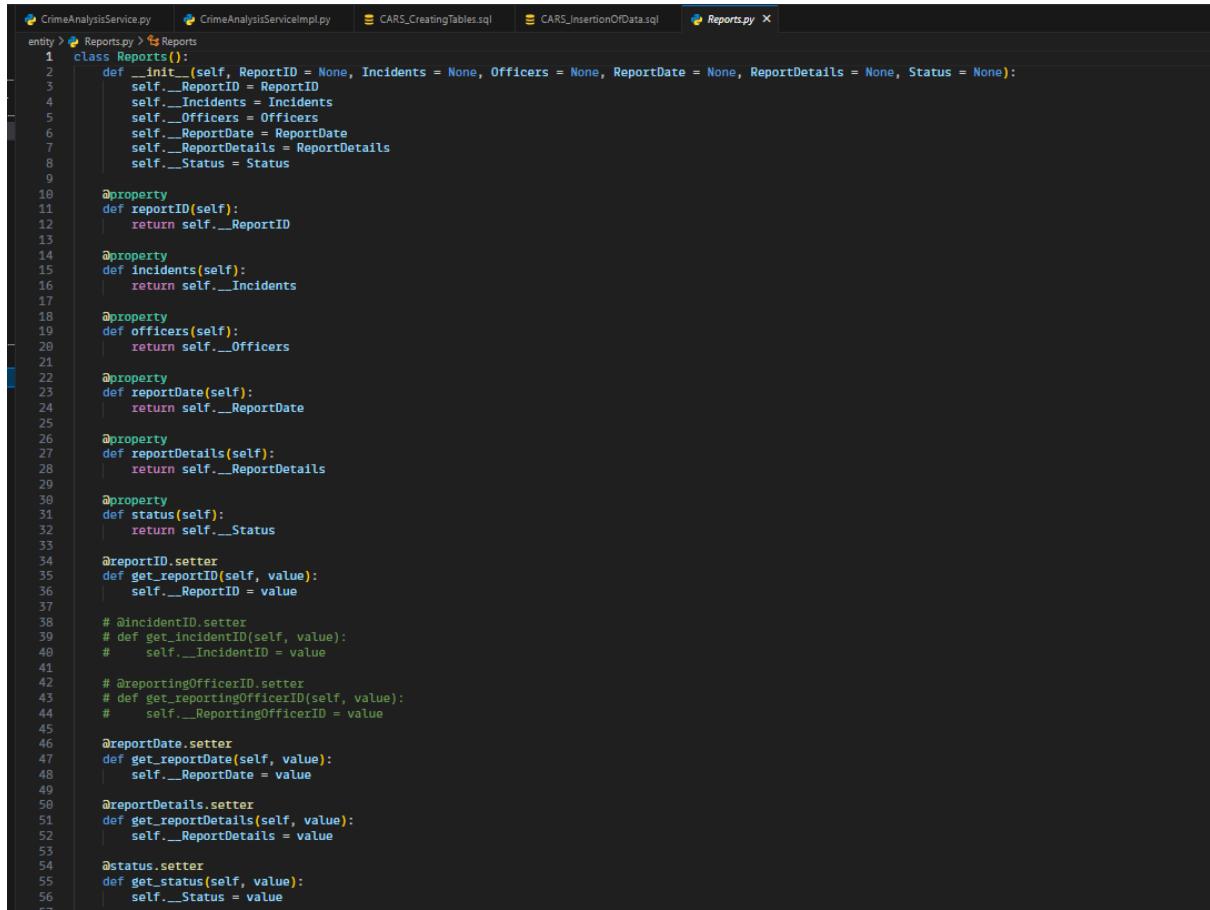
```
entity > 🐍 LawEnforcementAgencies.py > 📄 LawEnforcementAgencies
 1  class LawEnforcementAgencies():
 2      def __init__(self, AgencyID = None, AgencyName = None, Jurisdiction = None, EmailAddress = None):
 3          self.__AgencyID = AgencyID
 4          self.__AgencyName = AgencyName
 5          self.__Jurisdiction = Jurisdiction
 6          self.__EmailAddress = EmailAddress
 7
 8      @property
 9      def agencyID(self):
10          return self.__AgencyID
11
12      @property
13      def agencyName(self):
14          return self.__AgencyName
15
16      @property
17      def jurisdiction(self):
18          return self.__Jurisdiction
19
20      @property
21      def emailAddress(self):
22          return self.__EmailAddress
23
24      @agencyID.setter
25      def get_agencyID(self, value):
26          self.__AgencyID = value
27
28      @agencyName.setter
29      def get_agencyName(self, value):
30          self.__AgencyName = value
31
32      @jurisdiction.setter
33      def get_jurisdiction(self, value):
34          self.__Jurisdiction = value
35
36      @emailAddress.setter
37      def get_emailAddress(self, value):
38          self.__EmailAddress = value
39
40      def __str__(self):
41          return f"Agency ID: {self.__AgencyID} \nAgency Name: {self.__AgencyName} \nJurisdiction: {self.__Jurisdiction} \nEmail Address: {self.__EmailAddress}"
42
43
```

Officers.py

```
entity > 🐍 Officers.py > 📄 Officers
 1  class Officers():
 2      def __init__(self, OfficerID = None, FirstName = None, LastName = None, DateOfBirth = None, Gender = None, BadgeNumber = None, Ranking = None):
 3          self.__OfficerID = OfficerID
 4          self.__FirstName = FirstName
 5          self.__LastName = LastName
 6          self.__DateOfBirth = DateOfBirth
 7          self.__Gender = Gender
 8          self.__BadgeNumber = BadgeNumber
 9          self.__Ranking = Ranking
10          self.__PostingCity = PostingCity
11          self.__PostingState = PostingState
12          self.__ServiceJoiningDate = ServiceJoiningDate
13          self.__ResidentialAddress = ResidentialAddress
14          self.__ContactNumber = ContactNumber
15          self.__LawEnforcementAgency = LawEnforcementAgency
16
17      @property
18      def officerID(self):
19          return self.__OfficerID
20
21      @property
22      def firstName(self):
23          return self.__FirstName
24
25      @property
26      def lastName(self):
27          return self.__LastName
28
29      @property
30      def dateOfBirth(self):
31          return self.__DateOfBirth
32
33      @property
34      def gender(self):
35          return self.__Gender
36
37      @property
38      def badgeNumber(self):
39          return self.__BadgeNumber
40
41      @property
42      def ranking(self):
43          return self.__Ranking
44
```

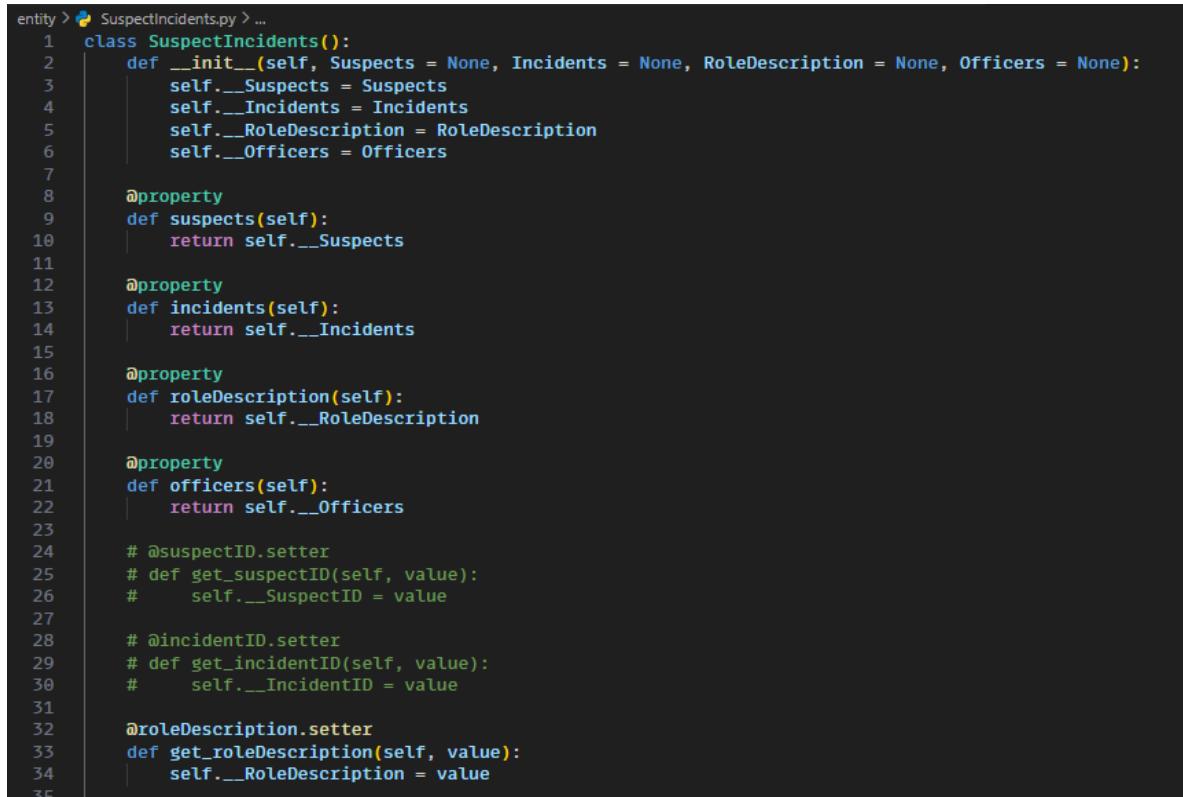
```
entity > Officers.py > Officers > get_officerID
 1  class Officers():
 2
 3      @property
 4      def ranking(self):
 5          return self.__Ranking
 6
 7      @property
 8      def postingCity(self):
 9          return self.__PostingCity
10
11      @property
12      def postingState(self):
13          return self.__PostingState
14
15      @property
16      def serviceJoiningDate(self):
17          return self.__ServiceJoiningDate
18
19      @property
20      def residentialAddress(self):
21          return self.__ResidentialAddress
22
23      @property
24      def contactNumber(self):
25          return self.__ContactNumber
26
27      @property
28      def lawEnforcementAgency(self):
29          return self.__AgencyID
30
31      @officerID.setter
32      def get_officerID(self, value):
33          self.__OfficerID = value
34
35      @firstName.setter
36      def get(firstName, self, value):
37          self.__FirstName = value
38
39      @lastName.setter
40      def get(lastName, self, value):
41          self.__LastName = value
42
43      @dateOfBirth.setter
44      def get_dateOfBirth(self, value):
45          self.__DateOfBirth = value
46
47      @gender.setter
48      def get_gender(self, value):
49          self.__Gender = value
50
51      @badgeNumber.setter
52      def get_badgeNumber(self, value):
53          self.__BadgeNumber = value
54
55      @ranking.setter
56      def get_ranking(self, value):
57          self.__Ranking = value
58
59      @postingCity.setter
60      def get_postingCity(self, value):
61          self.__PostingCity = value
62
63      @serviceJoiningDate.setter
64      def get_serviceJoiningDate(self, value):
65          self.__ServiceJoiningDate = value
66
67      @residentialAddress.setter
68      def get_residentialAddress(self, value):
69          self.__ResidentialAddress = value
70
71      @contactNumber.setter
72      def get_contactNumber(self, value):
73          self.__ContactNumber = value
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
```

Reports.py



```
entity > Reports.py > CrimeAnalysisService.py CrimeAnalysisServiceImpl.py CARS_CreatingTables.sql CARS_InsertionOfData.sql Reports.py x
1 class Reports():
2     def __init__(self, ReportID = None, Incidents = None, Officers = None, ReportDate = None, ReportDetails = None, Status = None):
3         self.__ReportID = ReportID
4         self.__Incidents = Incidents
5         self.__Officers = Officers
6         self.__ReportDate = ReportDate
7         self.__ReportDetails = ReportDetails
8         self.__Status = Status
9
10    @property
11    def reportID(self):
12        return self.__ReportID
13
14    @property
15    def incidents(self):
16        return self.__Incidents
17
18    @property
19    def officers(self):
20        return self.__Officers
21
22    @property
23    def reportDate(self):
24        return self.__ReportDate
25
26    @property
27    def reportDetails(self):
28        return self.__ReportDetails
29
30    @property
31    def status(self):
32        return self.__Status
33
34    @reportID.setter
35    def get_reportID(self, value):
36        self.__ReportID = value
37
38    # @incidentID.setter
39    # def get_incidentID(self, value):
40    #     self.__IncidentID = value
41
42    # @reportingOfficerID.setter
43    # def get_reportingOfficerID(self, value):
44    #     self.__ReportingOfficerID = value
45
46    @reportDate.setter
47    def get_reportDate(self, value):
48        self.__ReportDate = value
49
50    @reportDetails.setter
51    def get_reportDetails(self, value):
52        self.__ReportDetails = value
53
54    @status.setter
55    def get_status(self, value):
56        self.__Status = value
```

SuspectIncidents.py



```
entity > SuspectIncidents.py > ...
1 class SuspectIncidents():
2     def __init__(self, Suspects = None, Incidents = None, RoleDescription = None, Officers = None):
3         self.__Suspects = Suspects
4         self.__Incidents = Incidents
5         self.__RoleDescription = RoleDescription
6         self.__Officers = Officers
7
8     @property
9     def suspects(self):
10        return self.__Suspects
11
12     @property
13     def incidents(self):
14        return self.__Incidents
15
16     @property
17     def roleDescription(self):
18        return self.__RoleDescription
19
20     @property
21     def officers(self):
22        return self.__Officers
23
24     # @suspectID.setter
25     # def get_suspectID(self, value):
26     #     self.__SuspectID = value
27
28     # @incidentID.setter
29     # def get_incidentID(self, value):
30     #     self.__IncidentID = value
31
32     @roleDescription.setter
33     def get_roleDescription(self, value):
34        self.__RoleDescription = value
```

Suspects.py

```
entity > Suspects.py > Suspects
  1  class Suspects():
  2      def __init__(self, SuspectID = None, FirstName = None, LastName = None,
  3          self.__SuspectID = SuspectID
  4          self.__FirstName = FirstName
  5          self.__LastName = LastName
  6          self.__DateOfBirth = DateofBirth
  7          self.__Gender = Gender
  8          self.__ResidentialAddress = ResidentialAddress
  9          self.__ContactNumber = ContactNumber
 10         self.__AadhaarNumber = AadhaarNumber
 11
 12     @property
 13     def suspectID(self):
 14         return self.__SuspectID
 15
 16     @property
 17     def firstName(self):
 18         return self.__FirstName
 19
 20     @property
 21     def lastName(self):
 22         return self.__LastName
 23
 24     @property
 25     def dateOfBirth(self):
 26         return self.__DateOfBirth
 27
 28     @property
 29     def gender(self):
 30         return self.__Gender
 31
 32     @property
 33     def residentialAddress(self):
 34         return self.__ResidentialAddress
 35
 36     @property
 37     def contactNumber(self):
 38         return self.__ContactNumber
 39
 40     @property
 41     def aadhaarNumber(self):
 42         return self.__AadhaarNumber
 43
 44     @suspectID.setter
 45     def get_suspectID(self, value):
 46         self.__SuspectID = value
 47
 48     @firstName.setter
 49     def get(firstName, value):
 50         self.__FirstName = value
 51
 52     @lastName.setter
 53     def get(lastName, value):
 54         self.__LastName = value
 55
 56     @dateOfBirth.setter
 57     def get_dateOfBirth(self, value):
 58         self.__DateOfBirth = value
 59
 60     @gender.setter
 61     def get_gender(self, value):
 62         self.__Gender = value
 63
 64     @residentialAddress.setter
 65     def get_residentialAddress(self, value):
 66         self.__ResidentialAddress = value
 67
 68     @contactNumber.setter
 69     def get_contactNumber(self, value):
 70         self.__ContactNumber = value
 71
 72     @aadhaarNumber.setter
 73     def get_aadhaarNumber(self, value):
 74         self.__AadhaarNumber = value
 75
 76     def __str__(self):
 77         return f"Suspect ID: {self.__SuspectID} \nFirst Name: {self.__FirstN
```

Users.py

```
entity > 📁 Users.py > 🏷 User
1  class User:
2      def __init__(self, user_id=None, username=None, password=None, role='Officer', Officers=None):
3          self.__user_id = user_id
4          self.__username = username
5          self.__password = password
6          self.__role = role
7          self.__Officers = Officers
8
9      @property
10     def user_id(self):
11         return self.__user_id
12
13     @property
14     def username(self):
15         return self.__username
16
17     @property
18     def password(self):
19         return self.__password
20
21     @property
22     def role(self):
23         return self.__role
24
25     @property
26     def officers(self):
27         return self.__Officers
28
29     @user_id.setter
30     def user_id(self, value):
31         self.__user_id = value
32
33     @username.setter
34     def username(self, value):
35         self.__username = value
36
37     @password.setter
38     def password(self, value):
39         self.__password = value
40
41     @role.setter
42     def role(self, value):
43         if value not in ('Officer', 'Admin'):
44             raise ValueError("Role must be 'Officer' or 'Admin'")
45         self.__role = value
```

Victims.py

```
entity > 📁 Victims.py > 🏷 Victims
1  class Victims:
2      def __init__(self, VictimID=None, FirstName=None, LastName=None, DateOfBirth=None, Gender=None, ResidentialAddress=None, ContactNumber=None, AadhaarNumber=None, IncidentID=None):
3          self.__VictimID = VictimID
4          self.__FirstName = FirstName
5          self.__LastName = LastName
6          self.__DateOfBirth = DateOfBirth
7          self.__Gender = Gender
8          self.__ResidentialAddress = ResidentialAddress
9          self.__ContactNumber = ContactNumber
10         self.__AadhaarNumber = AadhaarNumber
11         self.__IncidentID = IncidentID
12
13     @property
14     def victimID(self):
15         return self.__VictimID
16     @victimID.setter
17     def victimID(self, value): self.__VictimID = value
18
19     @property
20     def firstName(self):
21         return self.__FirstName
22     @firstName.setter
23     def firstName(self, value):
24         self.__FirstName = value
25
26     @property
27     def lastName(self):
28         return self.__LastName
29     @lastName.setter
30     def lastName(self, value):
31         self.__LastName = value
32
33     @property
34     def dateOfBirth(self):
35         return self.__DateOfBirth
36     @dateOfBirth.setter
37     def dateOfBirth(self, value):
38         self.__DateOfBirth = value
39
40     @property
41     def gender(self):
42         return self.__Gender
43     @gender.setter
44     def gender(self, value):
45         self.__Gender = value
46
47     @property
48     def residentialAddress(self):
49         return self.__ResidentialAddress
50     @residentialAddress.setter
51     def residentialAddress(self, value):
52         self.__ResidentialAddress = value
53
54     @property
55     def contactNumber(self):
56         return self.__ContactNumber
57     @contactNumber.setter
58     def contactNumber(self, value):
59         self.__ContactNumber = value
60
61     @property
62     def aadhaarNumber(self):
63         return self.__AadhaarNumber
64     @aadhaarNumber.setter
65     def aadhaarNumber(self, value):
66         self.__AadhaarNumber = value
67
68     @property
69     def incidentID(self):
70         return self.__IncidentID
71     @incidentID.setter
72     def incidentID(self, value):
73         self.__IncidentID = value
74
```

Exceptions Folder

#user_defined_exceptions.py

```
exceptions > user_defined_exceptions.py > DatabaseError
1  class DatabaseError(Exception):
2      def __init__(self, message = "Database Error"):
3          self.message = message
4          super().__init__(self.message)
5
6  class IncidentNotFoundException(Exception):
7      def __init__(self, message="Incident not found"):
8          super().__init__(message)
9
10 class VictimNotFoundException(Exception):
11     def __init__(self, message="Victim not found."):
12         super().__init__(message)
13
14 class SuspectNotFoundException(Exception):
15     def __init__(self, message="Suspect not found."):
16         super().__init__(message)
17
18 class CriminalNotFoundException(Exception):
19     def __init__(self, message="Criminal not found."):
20         super().__init__(message)
21
22 class OfficerNotFoundException(Exception):
23     def __init__(self, message="Officer not found."):
24         super().__init__(message)
25
26 class AgencyNotFoundException(Exception):
27     def __init__(self, message="Law Enforcement Agency not found."):
28         super().__init__(message)
29
30 class ReportNotFoundException(Exception):
31     def __init__(self, message="Report not found."):
32         super().__init__(message)
33
34 class EvidenceNotFoundException(Exception):
35     def __init__(self, message="Evidence not found."):
36         super().__init__(message)
37
38 class DuplicateEntryException(Exception):
39     def __init__(self, message="Duplicate entry found."):
40         super().__init__(message)
41
42 class AuthenticationError(Exception):
43     def __init__(self, message = "Authentication Failed"):
44         self.message = message
45         super().__init__(self.message)
46
```

Tests Folder

#test_usertestcases.py

```
tests > 🐍 test_usertestcases.py > ⚙️ test_get_officers_by_location
1  import sys
2  import os
3  import pytest
4  from entity.Suspects import Suspects
5  from entity.auth import Authentication
6  from exceptions.user_defined_exceptions import AuthenticationError
7  from dao.CrimeAnalysisServiceImpl import CrimeAnalysisServiceImpl
8  from exceptions.user_defined_exceptions import DatabaseError
9
10 sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
11 service = CrimeAnalysisServiceImpl()
12 def test_public_access():
13     user = Authentication.public_access()
14     assert user['Role'] == 'Public'
15     assert user['Username'] == 'public_user'
16
17 def test_login_invalid(monkeypatch):
18     monkeypatch.setattr('builtins.input', lambda _: 'wrong_user')
19     monkeypatch.setattr('entity.auth.getpass', lambda _: 'wrong_pass')
20
21     with pytest.raises(AuthenticationError):
22         Authentication.login()
23
24
25 def test_get_officers_by_location():
26     officers = service.get_officers_by_location("coimbatore", "Coimbatore")
27     assert isinstance(officers, list)
28
29
30 def test_get_incidents_in_date_range_public_no_results():
31     results = service.get_incidents_in_date_range_public("1900-01-01", "1900-01-02")
32     assert isinstance(results, list)
33     assert len(results) == 0
34
35 def test_get_incidents_by_area_city_public_city_only():
36     results = service.get_incidents_by_area_city_public(area=None, city="coimbatore")
37     assert isinstance(results, list)
38
39 def test_get_incidents_by_area_city_public_invalid():
40     results = service.get_incidents_by_area_city_public(area="fakearea", city="fakecity")
41     assert isinstance(results, list)
42     assert len(results) == 0
43
44 def test_filtered_incidents_combination():
45     results = service.get_filtered_incidents(
46         start_date="2020-01-01",
47         end_date="2020-12-31",
48         area="Coimbatore",
49         city="coimbatore",
50         incident_type="theft",
51         officer_id=None
52     )
53     assert isinstance(results, list)
54
```

```
tests > ⚡ test_usertestcases.py > ⚡ test_view_all_suspects_returns_list
54
55     def test_view_my_incidents_invalid():
56         invalid_officer_id = -9999
57         result = service.viewMyIncidents(invalid_officer_id)
58
59         assert isinstance(result, list)
60         assert len(result) == 0
61
62     def test_add_duplicate_suspect():
63         suspect = Suspects(
64             FirstName="John",
65             LastName="Doe",
66             DateOfBirth="1990-01-01",
67             Gender="M",
68             ResidentialAddress="Some Street",
69             ContactNumber=9876543210,
70             AadhaarNumber=123456689112
71         )
72         incidentID = 1
73         officerID = 1
74         roleDescription = "Suspect in prior theft"
75
76         try:
77             service.addSuspect(suspect, incidentID, officerID, roleDescription)
78         except Exception:
79             pass
80
81         with pytest.raises(DatabaseError):
82             service.addSuspect(suspect, incidentID, officerID, roleDescription)
83
84     def test_add_invalid_suspect_data():
85         suspect = Suspects(
86             FirstName=None,
87             LastName="",
88             DateOfBirth="not-a-date",
89             Gender="",
90             ResidentialAddress=None,
91             ContactNumber="abc123",
92             AadhaarNumber=None
93         )
94         with pytest.raises(DatabaseError):
95             service.addSuspect(suspect, 1, 1, "Invalid data")
96
97     def test_view_all_suspects_returns_list():
98         suspects = service.viewAllSuspects()
99         assert isinstance(suspects, list)
100        if suspects:
101            item = suspects[0]
102            assert hasattr(item, "suspects")
103            assert hasattr(item, "incidents")
104            assert hasattr(item, "roleDescription")
105            assert hasattr(item, "officers")
106
107    def test_view_all_suspects_structure_should_fail():
108        suspects = service.viewAllSuspects()
109        assert isinstance(suspects, list)
110        if suspects:
111            item = suspects[0]
112            assert hasattr(item, "fakeField")
113    def test_view_my_incidents_invalid_should_fail():
114        invalid_officer_id = -9999
115        result = service.viewMyIncidents(invalid_officer_id)
116
117        assert isinstance(result, dict)
118        assert len(result) > 0
119    def test_add_duplicate_suspect():
120        with pytest.raises(DatabaseError):
121            service.addSuspect(suspect, incidentID, officerID, roleDescription)
```

Util Folder

#db.properties.py

```
⚙ db.properties
1 [database]
2 host = localhost
3 database = CrimeReportingSystem
4 user = root
5 password = mysql
6 port = 3306
```

Main Module

#main.py

```

main.py > MainModule > run
13  class MainModule:
14      def public_menu(self):
15          print("1. Get Incidents in Date Range")
16          print("2. Get Incidents by Area/City")
17          print("3. Get Incidents by Specific Fields")
18          print("4. Log In")
19          print("0. Exit")
20
21          choice = int(input("Enter choice: "))
22
23          if choice == 0:
24              print("\nExiting C.A.R.S System.. \n")
25              exit()
26
27          elif choice == 1:
28              try:
29                  startDate = input("Start date (YYYY-MM-DD): ")
30                  endDate = input("End date (YYYY-MM-DD): ")
31                  incidents = self.service.get_incidents_in_date_range_public(startDate, endDate)
32                  if not incidents:
33                      raise IncidentNotFoundException("Incidents not found within this date range.")
34
35                  print(f"\nIncidents that happened between '{startDate}' and '{endDate}': \n")
36                  print("-----")
37                  for incident in incidents:
38                      print(f"  Incident Type: {incident.incidentType}")
39                      print(f"  Incident Date: {incident.incidentDate}")
40                      print(f"  Area: {incident.area}")
41                      print(f"  City: {incident.city}")
42                      print(f"  Description: {incident.description}")
43                      print("-----")
44
45              except IncidentNotFoundException as e:
46                  print(f"No incidents found: {str(e)}")
47              except mysql.connector.Error as e:
48                  raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
49
50
51          elif choice == 2:
52              try:
53                  print("\nEnter the field blank if you don't want that filter")
54                  area = input("Enter area: ").strip()
55                  city = input("Enter city: ").strip()
56                  if not area and not city:
57                      print("Both fields cannot be left empty.")
58                      return
59
60                  incidents = self.service.get_incidents_by_area_city_public(area, city)
61                  if not incidents:
62                      raise IncidentNotFoundException("Incidents not found in this area and city.")
63                  location = ", ".join(filter(None, [area, city]))
64                  print(f"Incidents that happened in {location}: \n")
65                  print("-----")
66                  for incident in incidents:
67                      print(f"  Incident Type: {incident['IncidentType']}")
68                      print(f"  Incident Date: {incident['IncidentDate']}")
69                      if area: print(f"  Area: {incident['Area']}")
70                      if city: print(f"  City: {incident['City']}")
71                      print(f"  Description: {incident['Description']}")
72                      print("-----")
73
74              except IncidentNotFoundException as e:
75                  print(f"No incidents found: {str(e)}")
76              except mysql.connector.Error as e:
77                  raise DatabaseError(f"Database Connection Failed: {str(e)}")
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95

```

```

class MainModule:
    def public_menu(self):
        elif choice == 4:
            try:
                self.current_user = Authentication.login()
                print(f"Welcome, {self.current_user['Username']}!")
            except AuthenticationError as e:
                print(f"Authentication Error: {str(e)}")
        elif choice == 0:
            print("\nExiting C.A.R.S System.. \n")
            exit()

        else:
            print("Please enter a number between 0 - 3")
            return

    except ValueError as e:
        print("Invalid Input")
    except Exception as e:
        print(f"Unknown Error: {str(e)}")

```

```
main.py > MainModule > run
3  class MainModule:
4
5      def officer_menu(self):
6          print(f"\n--- Officer Menu ---")
7          print("\n-- Incident Management --")
8          print("1. View all incident details with assignments")
9          print("2. View my assigned incidents")
10         print("3. Filter incidents")
11
12        print("\n-- Suspect Management --")
13        print("4. Access suspects database")
14        print("5. Create new suspect record")
15        print("6. Access criminals database")
16
17        print("\n-- Case Management --")
18        print("7. Create new victim record")
19        print("8. View case details (victims/suspects/evidence)")
20        print("9. Generate/Update case report")
21        print("10. View case report")
22
23        print("11. Logout")
24
25    try:
26        choice = int(input("Enter choice: "))
27
28        if choice == 1:
29            try:
30                incidents = self.service.viewAllIncidents()
31
32                if not incidents:
33                    raise IncidentNotFoundException(f"No incidents are found.")
34
35                print("\n--- All Incidents ---")
36                for i, incident in enumerate(incidents, start=1):
37                    print(f"\nIncident {i}:")
38                    print(f" ID : {incident.incidentID}")
39                    print(f" Type : {incident.incidentType}")
40                    print(f" Date : {incident.incidentDate}")
41                    print(f" Area : {incident.area}")
42                    print(f" City : {incident.city}")
43                    print(f" Description : {incident.description}")
44                    print(f" Status : {incident.status}")
45                    print(f" Officer ID : {incident.officerID}")
46                    print("-----")
47
48            except mysql.connector.Error as e:
49                raise DatabaseError(f"Database Connection Failed: {str(e)}")
50            except Exception as e:
51                print(f"Unknown error: {str(e)}")
```

```
▶ main.py > ⌂ MainModule > ⌂ run
13  class MainModule:
115     def officer_menu(self):
116
117         elif choice == 2:
118             try:
119                 officerID = self.current_user['UserID']
120                 incidents = self.service.viewMyIncidents(officerID)
121                 if not incidents:
122                     raise IncidentNotFoundException(f"No incidents are found.")
123
124                 print("==== My Assignments ====")
125                 for i, incident in enumerate(incidents, start=1):
126                     print(f"\nIncident {i}:")
127                     print(f" ID : {incident['IncidentID']} ")
128                     print(f" Type : {incident['IncidentType']} ")
129                     print(f" Date : {incident['IncidentDate']} ")
130                     print(f" Area : {incident['Area']} ")
131                     print(f" City : {incident['City']} ")
132                     print(f" Description : {incident['Description']} ")
133                     print(f" Status : {incident['Status']} ")
134                     print(f" Officer ID : {incident['OfficerID']} ")
135             except mysql.connector.Error as e:
136                 raise DatabaseError(f"Database Connection Failed: {str(e)}")
137             except Exception as e:
138                 print(f"Unknown error: {str(e)}")
139
140
141         elif choice == 3:
142             try:
143                 print("\n==== Filtering the Incidents ====\n")
144                 print("Fill the parameters on whose basis you wish to filter and leave the rest blank")
145                 print("\nDate Filters:")
146                 start_date = input("Enter the start date (YYYY-MM-DD): ").strip()
147                 end_date = input("Enter the end date (YYYY-MM-DD): ").strip()
148                 area = input("Enter the area: ").strip()
149                 city = input("Enter the city: ").strip()
150                 incidentType = input("Enter the type of incident: ").strip()
151                 officerID_input = input("Enter the Officer ID: ")
152                 officerID = int(officerID_input) if officerID_input else None
153
154
155                 incidents = self.service.get_filtered_incidents(start_date, end_date, area, city, incidentType, officerID)
156                 if not incidents:
157                     raise IncidentNotFoundException("Incidents not found.")
158                 print("==== Filtered Results ====")
159                 for i, incident in enumerate(incidents, start=1):
160                     print(f"\nIncident {i}:")
161                     print(f" ID : {incident['IncidentID']} ")
162                     print(f" Type : {incident['IncidentType']} ")
163                     print(f" Date : {incident['IncidentDate']} ")
164                     print(f" Area : {incident['Area']} ")
165                     print(f" City : {incident['City']} ")
166                     print(f" Description : {incident['Description']} ")
167                     print(f" Status : {incident['Status']} ")
168                     print(f" Officer ID : {incident['OfficerID']} ")
169
170             except mysql.connector.Error as e:
171                 raise DatabaseError(f"Database Connection Failed: {str(e)}")
172             except Exception as e:
173                 print(f"Unknown error: {str(e)}")
```

```

class MainModule:
    def officer_menu(self):
        elif choice == 4:
            try:
                suspects = self.service.viewAllSuspects()
                if not suspects:
                    raise SuspectNotFoundException(f"Suspects not found")
                print("==== Suspects Database ====")
                for suspect_incident in suspects:
                    suspect = suspect_incident.suspects
                    print(f" Suspect ID : {suspect.suspectID}")
                    print(f" Name : {suspect.firstName} {suspect.lastName}")
                    print(f" Date of Birth : {suspect.dateOfBirth}")
                    print(f" Aadhaar Number : {suspect.aadhaarNumber}")
                    print(f" Address : {suspect.residentialAddress}")
                    print(f" Contact Number : {suspect.contactNumber}")

                    incident = suspect_incident.incidents
                    print(f" Incident ID : {incident.incidentID}")
                    print(f" Incident Type : {incident.incidentType}")
                    print(f" Incident Date : {incident.incidentDate}")
                    print(f" Description : {incident.description}")
                    print(f" Status : {incident.status}")

                    print(f" Role in Incident : {suspect_incident.roleDescription}")
                    print(f" Added By Officer : {suspect_incident.officers}")

                    records = self.service.suspectCriminalRelation(suspect.aadhaarNumber)
                    if records:
                        print(" Previous Criminal Record : Yes")
                        for record in records:
                            print(f" Criminal ID : {record.criminalID}")
                    else:
                        print(" Previous Criminal Record : No")
            except mysql.connector.Error as e:
                raise DatabaseError(f"Database Connection Failed: {str(e)}")
            except Exception as e:
                print(f"Unknown error: {str(e)}")

```

```

main.py > ⌂ MainModule > ⌂ run
13   class MainModule:
115     def officer_menu(self):
255
256         elif choice == 5:
257             try:
258                 print("==== Adding a new Suspect ====")
259                 firstName = input("First Name: ").strip()
260                 lastName = input("Last Name: ").strip()
261
262             try:
263                 dob_input = input("Date of Birth (YYYY-MM-DD): ").strip()
264                 dateOfBirth = datetime.strptime(dob_input, "%Y-%m-%d").date()
265             except ValueError:
266                 print("Invalid date format. Please enter as YYYY-MM-DD.")
267                 return
268
269             gender = input("Gender: ").strip()
270             residentialAddress = input("Residential Address: ").strip()
271             contactNumber = int(input("Contact Number: "))
272             aadhaarNumber = int(input("Aadhaar Number: "))
273
274             incidentID = int(input("Incident ID involved: "))
275             officerID = int(input("Your Officer ID: "))
276             roleDescription = input("Role in Incident: ").strip()
277
278             new_suspect = Suspects(
279                 FirstName=firstName,
280                 LastName=lastName,
281                 DateOfBirth=dateOfBirth,
282                 Gender=gender,
283                 ResidentialAddress=residentialAddress,
284                 ContactNumber=contactNumber,
285                 AadhaarNumber=aadhaarNumber
286             )
287
288             suspectID = self.service.addSuspect(new_suspect, incidentID, officerID, roleDescription)
289             if not suspectID:
290                 raise DatabaseError("Failed to create suspect")
291             print(f"Suspect added successfully with ID: {suspectID}")
292
293             except mysql.connector.Error as e:
294                 raise DatabaseError(f"Database Connection Failed: {str(e)}")
295             except DuplicateEntryException as e:
296                 print(e)
297             except Exception as e:
298                 print(f"Unknown error: {str(e)}")
299

```

```
ain.py > MainModule > run
class MainModule:
    def officer_menu(self):
        elif choice == 6:
            try:
                criminals = self.service.viewAllCriminals()
                if not criminals:
                    raise CriminalNotFoundException(f"Criminals not found")
                print("==== Criminals Database ====")
                for criminal in criminals:
                    print(f"  Criminal ID      : {criminal.criminalID}")
                    print(f"  Incident ID     : {criminal.incidentID}")
                    print(f"  Punishment Details : {criminal.punishmentDetails}")
                    print(f"  Aadhaar Number   : {criminal.aadhaarNumber}")

                    suspect = criminal.suspect
                    print(f"  First Name       : {suspect.firstName}")
                    print(f"  Last Name        : {suspect.lastName}")
                    print(f"  Date of Birth    : {suspect.dateOfBirth}")
                    print(f"  Address          : {suspect.residentialAddress}")
                    print(f"  Contact Number   : {suspect.contactNumber}")
                    print(f"  Gender           : {suspect.gender}")
                    print("-----")

            except mysql.connector.Error as e:
                raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
            except Exception as e:
                print(f"Unknown error: {str(e)}")
```

```
main.py > MainModule > officer_menu
13     class MainModule:
14         def officer_menu(self):
15             ...
16             elif choice == 7:
17                 try:
18                     print("== Add a New Victim ==")
19                     firstName = input("First Name: ").strip()
20                     lastName = input("Last Name: ").strip()
21
22                     try:
23                         dob_input = input("Date of Birth (YYYY-MM-DD): ").strip()
24                         dateOfBirth = datetime.strptime(dob_input, "%Y-%m-%d").date()
25                     except ValueError:
26                         print("Invalid date format. Please enter as YYYY-MM-DD.")
27                         return
28
29                     gender = input("Gender: ").strip()
30                     residentialAddress = input("Residential Address: ").strip()
31
32                     try:
33                         contactNumber = int(input("Contact Number: "))
34                         aadhaarNumber = int(input("Aadhaar Number: "))
35                     except ValueError:
36                         print("Contact Number and Aadhaar Number must be numeric.")
37                         return
38
39                     try:
40                         incidentID = int(input("Incident ID involved: "))
41                         officerID = int(input("Your Officer ID: "))
42                     except ValueError:
43                         print("Incident ID and Officer ID must be numeric.")
44                         return
45
46                     roleDescription = input("Role in Incident: ").strip()
47
48                     new_victim = Victims(
49                         FirstName=firstName,
50                         LastName=lastName,
51                         DateOfBirth=dateOfBirth,
52                         Gender=gender,
53                         ResidentialAddress=residentialAddress,
54                         ContactNumber=contactNumber,
55                         AadhaarNumber=aadhaarNumber,
56                         IncidentID=incidentID
57                     )
58                     victimID = self.service.addVictim(new_victim, incidentID, officerID, roleDescription)
59
60                     if victimID:
61                         print(f"Victim successfully associated with Incident {incidentID}. Victim ID: {victimID}")
62                     else:
63                         print(f"Victim with Aadhaar {aadhaarNumber} already exists for Incident {incidentID}.")
64
65                 except IncidentNotFoundException:
66                     print("Error: Incident not found. Please verify the Incident ID.")
67                 except Exception as e:
68                     print(f"Unexpected error: {str(e)}")
69
70             ...
71
72             print("Main menu")
73             choice = int(input("Enter choice (1-7): "))
74
75         ...
76
77     ...
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2290
2291
2292
2293
2294
2295
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2390
2391
2392
2393
2394
2395
2395
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
```

```
main.py > MainModule > officer_menu
13  class MainModule:
115     def officer_menu(self):
379         elif choice == 8:
380             try:
381                 print("View Case Related Resources: \n")
382                 incidentID = int(input("Enter the Incident ID: "))
383                 victims = self.service.get_victims_by_incident(incidentID)
384                 if not victims:
385                     print(f"No victims found regarding Incident {incidentID}")
386                 else:
387                     print(f"\nVictims related to Case {incidentID}: \n")
388                     for victim in victims:
389                         print(f"Victim ID: {victim.victimID}")
390                         print(f"First Name: {victim.firstName}")
391                         print(f"Last Name: {victim.lastName}")
392                         print(f"Date Of Birth: {victim.dateOfBirth}")
393                         print(f"Gender: {victim.gender}")
394                         print(f"Residential Address: {victim.residentialAddress}")
395                         print(f"Contact Number: {victim.contactNumber}")
396                         print(f"Aadhaar Number: {victim.aadhaarNumber}")
397                         print("-----")
398                     suspects = self.service.get_suspects_by_incident(incidentID)
399                     if not suspects:
400                         print(f"No suspects found regarding Incident {incidentID}")
401                     else:
402                         print(f"\nSuspects related to Case {incidentID}: \n")
403                         for suspect_incident in suspects:
404                             suspect = suspect_incident.suspects
405                             print(f"Suspect ID: {suspect.suspectID}")
406                             print(f"First Name: {suspect.firstName}")
407                             print(f"Last Name: {suspect.lastName}")
408                             print(f"Date Of Birth: {suspect.dateOfBirth}")
409                             print(f"Gender: {suspect.gender}")
410                             print(f"Residential Address: {suspect.residentialAddress}")
411                             print(f"Contact Number: {suspect.contactNumber}")
412                             print(f"Aadhaar Number: {suspect.aadhaarNumber}")
413                             print(f"Role in Incident: {suspect_incident.roleDescription}")
414                             print(f"Added By Officer: {suspect_incident.officers}")
415                             print("-----")
416                     evidences = self.service.get_evidence_by_incident(incidentID)
417                     if not evidences:
418                         print(f"No evidences found regarding Incident {incidentID}")
419                     else:
420                         print(f"\nEvidences related to Case {incidentID}: \n")
421                         for evidence in evidences:
422                             print(f"Evidence ID: {evidence.evidenceID}")
423                             print(f"Evidence Name: {evidence.evidenceName}")
424                             print(f"Description: {evidence.description}")
425                             print(f"Location Found: {evidence.locationFound}")
426                             print("-----")
427             except mysql.connector.Error as e:
428                 raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
429             except Exception as e:
430                 print(f"Unknown error: {str(e)}")
```



```

main.py > MainModule > officer_menu
13   class MainModule:
14     def officer_menu(self):
15       if choice == 10:
16         try:
17           incidentID = int(input("Enter Incident ID: "))
18           report = self.service.view_incident_report(incidentID)
19           if not report:
20             print(f"No report found for Incident ID {incidentID}")
21           else:
22             print("\n===== Incident Report =====")
23             print(f"Incident ID      : {report['IncidentID']}")
24             print(f"Incident Type    : {report['IncidentType']}")
25             print(f"Area            : {report['Area']}")
26             print(f"City             : {report['City']}")
27             print(f"Description     : {report['Description']}")
28             print(f"Incident Status : {report['Status']}")
29             print(f"Report Date     : {report['ReportDate']}")
30             print(f"Report Details  : {report['ReportDetails']}")
31             print(f"Report Status   : {report['ReportStatus']}")

32         except mysql.connector.Error as e:
33             raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
34         except Exception as e:
35             print(f"Unexpected error: {str(e)}")

36       elif choice == 11:
37         self.current_user = Authentication.public_access()
38
39       else:
40         print("Please enter a number between 0 - 3")

41     except ValueError as e:
42       print(f"Invalid input {str(e)}")
43     except Exception as e:
44       print(f"Unknown Error: {str(e)}")

```

```

main.py > MainModule > officer_menu
13   class MainModule:
14     def admin_menu(self):
15       print(f"\n--- Admin Menu ({self.current_user['Username']}) ---")
16       print("\n-- Records Management --")
17       print("1. Create criminal record")
18       print("2. Create incident record")
19       print("3. Add new officer")
20       print("4. View case assignments")
21       print("5. View all law enforcement agencies")

22       print("\n-- Analytics --")
23       print("6. Officer performance dashboard")
24       print("7. Criminal analytics")
25       print("8. Crime pattern analytics")

26       print("\n-- Case Oversight --")
27       print("9. Finalize reports & update case status")

28       print("10. Logout")

29     try:
30       choice = int(input("Enter choice: "))

31       if choice == 1:
32         try:
33           print("\n--- Add Criminal Record ---")
34           incidentID = input("Enter Incident ID: ")
35           aadhaarNumber = input("Enter Aadhaar Number of criminal: ")
36           punishmentDetails = input("Enter Punishment Details: ")

37           criminal = Criminals(
38             IncidentID=incidentID,
39             AadhaarNumber=aadhaarNumber,
40             PunishmentDetails=punishmentDetails
41           )

42           criminal_id = self.service.addCriminal(criminal)
43           if criminal_id:
44             print(f"Criminal added successfully. Criminal ID: {criminal_id}")
45           else:
46             print("Criminal entry already exists in the database.")

47         except SuspectNotFoundException as e:
48             print(f"Suspect ID not found in the database.")
49         except DatabaseError as e:
50             print(f"Database Error: {str(e)}")
51         except Exception as e:
52             print(f"Unexpected error: {str(e)}")

```

```
main.py > MainModule > officer_menu
13 class MainModule:
518     def admin_menu(self):
505
566         elif choice == 2:
567             try:
568                 print("\n==== Create New Incident ===")
569                 incident_type = input("Incident Type: ")
570                 incident_date = input("Incident Date (YYYY-MM-DD): ")
571                 area = input("Area: ")
572                 city = input("City: ")
573                 description = input("Description: ")
574                 status = "Open"
575
576                 print("\nSearching for officers in the area...")
577
578                 officers = self.service.getOfficersByLocation(city, area)
579                 if not officers:
580                     print("No officers found for this location.")
581                     return
582
583                 print("\nAvailable Officers:")
584                 for officer in officers:
585                     officer_id = officer['OfficerID']
586                     case_count = self.service.countOpenIncidentsByOfficer(officer_id)
587                     print(f"Officer ID : {officer_id}")
588                     print(f"Name : {officer['FirstName']} {officer['LastName']}")
589                     print(f"Posting City : {officer['PostingCity']}")
590                     print(f"Posting State : {officer['PostingState']}")
591                     print(f"Current Cases : {case_count}")
592                     print("-" * 30)
593
594                 assigned_id = int(input("Enter Officer ID to assign this case to: "))
595
596                 incident = Incidents(
597                     IncidentType=incident_type,
598                     IncidentDate=incident_date,
599                     Area=area,
600                     City=city,
601                     Description=description,
602                     Status=status,
603                     OfficerID=assigned_id
604                 )
605
606                 incidentID = self.service.createIncident(incident)
607                 if incidentID:
608                     print(f"Incident created with ID {incidentID} and assigned to Officer {assigned_id}")
609                 else:
610                     print("Incident Creation Failed. Please try again. ")
611
612             except DatabaseError as e:
613                 print(f"Database Error: {str(e)}")
614             except Exception as e:
615                 print(f"Unexpected Error: {str(e)}")
```

```
main.py > MainModule > officer_menu
13 class MainModule:
518     def admin_menu(self):
617
618         elif choice == 3:
619             try:
620                 print("\n==== Create New Officer Account ===")
621                 first_name = input("First Name: ")
622                 last_name = input("Last Name: ")
623                 dob = input("Date of Birth (YYYY-MM-DD): ")
624                 gender = input("Gender: ")
625                 badge_number = input("Badge Number: ")
626                 ranking = input("Ranking (e.g., Inspector, Sub-Inspector): ")
627                 posting_city = input("Posting City: ")
628                 posting_state = input("Posting State: ")
629                 service_joining_date = input("Service Joining Date (YYYY-MM-DD): ")
630                 address = input("Residential Address: ")
631                 contact = input("Contact Number: ")
632                 agency_id = input("Agency ID: ")
633
634                 print("\n--- Login Credentials ---")
635                 username = input("Username: ")
636                 password = input("Password: ")
637
638                 officer_data = {
639                     'firstName': first_name,
640                     'lastName': last_name,
641                     'dateOfBirth': dob,
642                     'gender': gender,
643                     'badgeNumber': badge_number,
644                     'ranking': ranking,
645                     'postingCity': posting_city,
646                     'postingState': posting_state,
647                     'serviceJoiningDate': service_joining_date,
648                     'residentialAddress': address,
649                     'contactNumber': contact,
650                     'agencyID': agency_id,
651                     'username': username,
652                     'password': password
653                 }
654
655                 officerID = self.service.create_officer(officer_data)
656                 if officerID:
657                     print(f"Officer account created successfully. ID: {officerID}")
658                 else:
659                     print("Failed to create officer account.")
660
661             except mysql.connector.Error as e:
662                 raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
663             except Exception as e:
664                 print(f"Unexpected Error: {str(e)}")
```

```

main.py > MainModule > officer_menu
13  class MainModule:
518      def admin_menu(self):
...
666          elif choice == 4:
667              try:
668                  incidents = self.service.viewAllIncidents()
669
670                  if not incidents:
671                      raise IncidentNotFoundException(f"No incidents are found.")
672
673                  print("\n==== All Incidents ===")
674                  for i, incident in enumerate(incidents, start=1):
675                      print(f"\nIncident {i}:")
676                      print(f" ID : {incident.incidentID}")
677                      print(f" Type : {incident.incidentType}")
678                      print(f" Date : {incident.incidentDate}")
679                      print(f" Area : {incident.area}")
680                      print(f" City : {incident.city}")
681                      print(f" Description : {incident.description}")
682                      print(f" Status : {incident.status}")
683                      print(f" Officer ID : {incident.officerID}")
684                      print("-" * 40)
685
686              except mysql.connector.Error as e:
687                  raise DatabaseError(f"Database Connection Failed: {str(e)}")
688              except Exception as e:
689                  print(f"Unknown error: {str(e)}")
690
691          elif choice == 5:
692              try:
693                  agencies = self.service.view_all_agencies()
694                  if not agencies:
695                      print("No Law Enforcement Agencies found.")
696                  else:
697                      print("\n==== Law Enforcement Agencies ===")
698                      for agency in agencies:
699                          print(agency)
700                          print()
701                          print("-" * 40)
702
703              except AgencyNotFoundException as e:
704                  print("Agency not found.")
705              except Exception as e:
706                  print(f"Unexpected Error: {str(e)}")

```

```

main.py > MainModule > officer_menu
13  class MainModule:
518      def admin_menu(self):
...
707          elif choice == 6:
708              try:
709                  dashboard_data = self.service.get_all_officer_dashboard()
710                  print("\n==== Officer Dashboard ===")
711                  if not dashboard_data:
712                      print("No officer data available.")
713                  else:
714                      for officer in dashboard_data:
715                          print("-" * 40)
716                          print(f"Officer ID : {officer['OfficerID']}")
717                          print(f"Name : {officer['FirstName']} {officer['LastName']}")
718                          print(f"Badge Number : {officer['BadgeNumber']}")
719                          print(f"Ranking : {officer['Ranking']}")
720                          print(f"Posting Location : {officer['PostingCity']}, {officer['PostingState']}")
721                          print(f"Service Joined On : {officer['ServiceJoiningDate']}")
722                          print(f"Total Cases : {officer['total_cases']}")
723                          print(f"Cases Closed : {officer['cases_closed']}")
724
725              except mysql.connector.Error as e:
726                  raise DatabaseError(f"Database connection error: {e.msg}") from e
727              except Exception as e:
728                  print(f"Unexpected error: {str(e)}")
729
730          elif choice == 7:
731              try:
732                  analytics = self.service.get_criminal_analytics()
733
734                  print("\n==== Criminal Analytics Dashboard ===")
735
736                  print("\nTop 5 Most Frequent Criminals:")
737                  if analytics['top_criminals']:
738                      index = 1
739                      for criminal in analytics['top_criminals']:
740                          print(f" {index}. {criminal['FirstName']} {criminal['LastName']} - {criminal['crime_count']} cases")
741                          index += 1
742
743                  else:
744                      print(" No data found for top criminals.")
745
746                  print("\nCrime Type Distribution:")
747                  if analytics['crime_types']:
748                      for crime_type in analytics['crime_types']:
749                          print(f" {crime_type} : {analytics['crime_types'][crime_type]}")
750
751              except mysql.connector.Error as e:
752                  raise DatabaseError(f"Database connection error: {e.msg}") from e
753              except Exception as e:
754                  print(f"Unknown error: {str(e)}")

```

```

main.py > MainModule > officer_menu
13   class MainModule:
518     def admin_menu(self):
122
756     elif choice == 8:
757       try:
758         analytics = self.service.get_crime_analytics()
759
760         print("\n==== Crime Analytics Dashboard ====")
761
762         print("\nTop 5 Crime Hotspots (City & Area):")
763         if analytics['hotspots']:
764           index = 1
765           for row in analytics['hotspots']:
766             print(f" {index}. {row['City']}, {row['Area']} - {row['crime_count']} incidents")
767             index += 1
768         else:
769           print(" No hotspot data found.")
770
771         print("\nCrime Count by Month:")
772         if analytics['monthly_trends']:
773           for month in analytics['monthly_trends']:
774             print(f" {month} : {analytics['monthly_trends'][month]}")
775         else:
776           print(" No monthly trend data available.")
777
778       except mysql.connector.Error as e:
779         raise DatabaseError(f"Database connection error: {e.msg}") from e
780       except Exception as e:
781         print(f"Unexpected error: {str(e)}")
782
783     elif choice == 9:
784       try:
785         reports = self.service.get_reports_by_status("Finalized")
786         if not reports:
787           print("No reports are pending for review.")
788         else:
789           print("\n==== Reports Pending Review ====")
790           for report in reports:
791             print(f"\nIncident ID      : {report['IncidentID']}")
792             print(f"Type            : {report['IncidentType']}")
793             print(f"Area/City       : {report['Area']}, {report['City']}")
794             print(f"Status          : {report['Status']}")
795             print(f"Report Date     : {report['ReportDate']}")
796             print(f"Report Details  : {report['ReportDetails']}")
797
798           incidentID = int(input("\nEnter the Incident ID to close: "))
799
800           rows_updated = self.service.close_case_by_admin(incidentID)
801           if rows_updated:
802             print(f"Case status of Incident: {incidentID} updated to 'CLOSED' successfully.")
803           else:
804             print("Failed to update case status.")
805       except Exception as e:
806         print(f"Unexpected Error: {str(e)}")
807
808
809
810
811
812
813
814
815
816
817
818

```

```

main.py > MainModule > officer_menu
13   class MainModule:
518     def admin_menu(self):
807
808       elif choice == 10:
809         self.current_user = Authentication.public_access()
810
811       except ValueError as e:
812         print("Please enter a number between 0 - 3")
813       except Exception as e:
814         print(f"Unexpected Error: {str(e)}")
815
816     if __name__ == "__main__":
817       app = MainModule()
818       app.run()

```

Conclusion:

C.A.R.S. successfully addresses the need for a centralized crime reporting and analysis tool. By digitizing incident and case management, it not only improves data accuracy and accessibility but also empowers law enforcement agencies to make data-driven decisions. The system lays a strong foundation for further development into a more advanced, fully-fledged crime analytics platform.

C.A.R.S. also enhances collaboration between agencies by centralizing data and streamlining communication workflows. Officers in the field, analysts in the office, and administrators can all interact within a unified digital ecosystem that supports better decision-making.

Moreover, the modular architecture of C.A.R.S. allows for easy upgrades and integration with emerging technologies, ensuring long-term sustainability and adaptability to evolving law enforcement needs.

This project bridges the gap between traditional paper-based workflows and modern digital systems in law enforcement. With its clear structure, thoughtful design, and extensibility, C.A.R.S. exemplifies how technology can transform critical public safety processes. By integrating advanced features such as AI-driven analytics, GIS visualization, and automated backup systems, C.A.R.S. not only improves operational efficiency but also enhances security, transparency, and responsiveness in law enforcement. As a comprehensive and scalable solution, C.A.R.S. sets a new standard for digital policing in the 21st century.

Future Enhancements:

- **Multi-factor authentication for higher security**

C.A.R.S. employs multi-factor authentication (MFA) to ensure that only authorized personnel gain access to sensitive law enforcement data. This added layer of security greatly reduces the risk of breaches and unauthorized access, thus protecting both operational integrity and public trust.

- **Integration with national criminal databases**

By seamlessly connecting with national criminal databases, the system enables officers to retrieve and cross-reference data in real time. This significantly accelerates the investigative process and improves the accuracy of crime tracking and suspect identification.

- **Automated data backups and recovery**

To safeguard critical records, C.A.R.S. includes automated backup and recovery systems. These ensure that valuable information is never lost, even in the event of a system failure or cyberattack, promoting operational continuity and resilience.

- **GIS (Geographic Information System) integration to visualize crime hotspots on maps**

The system integrates GIS technology to allow real-time mapping of crime patterns. Law enforcement agencies can visualize high-crime areas, track incident trends, and allocate resources more effectively to areas that need attention most.

- **AI-based pattern detection for predictive policing**

With AI-driven analytics, C.A.R.S. can detect crime patterns and generate predictive insights. This proactive approach helps law enforcement agencies anticipate criminal activity and respond more efficiently, potentially preventing crimes before they occur.