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 I**ncident_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**

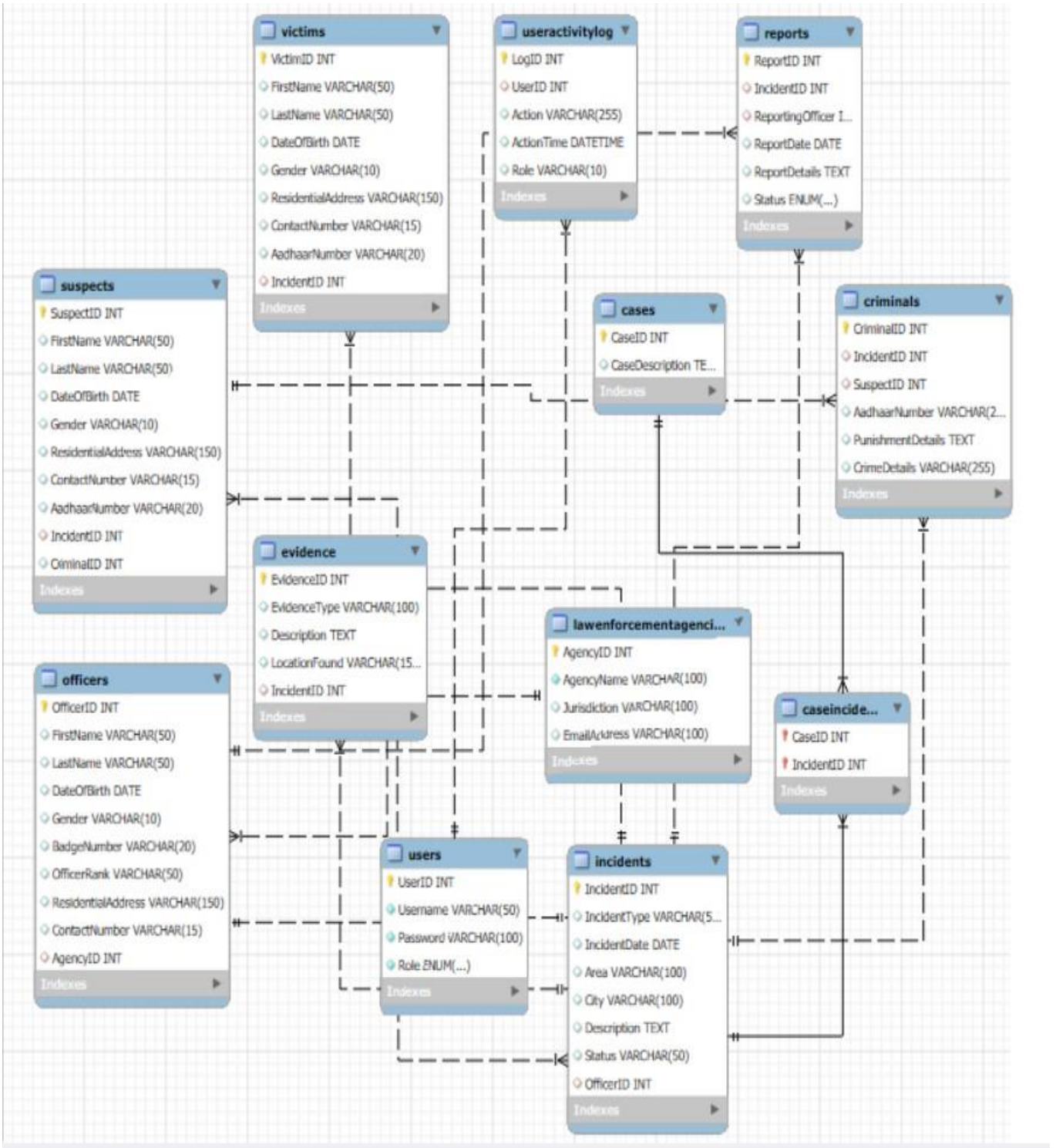
- LogID (Primary Key).
- UserID (Foreign Key to Users.UserID).

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

Relationships Summary

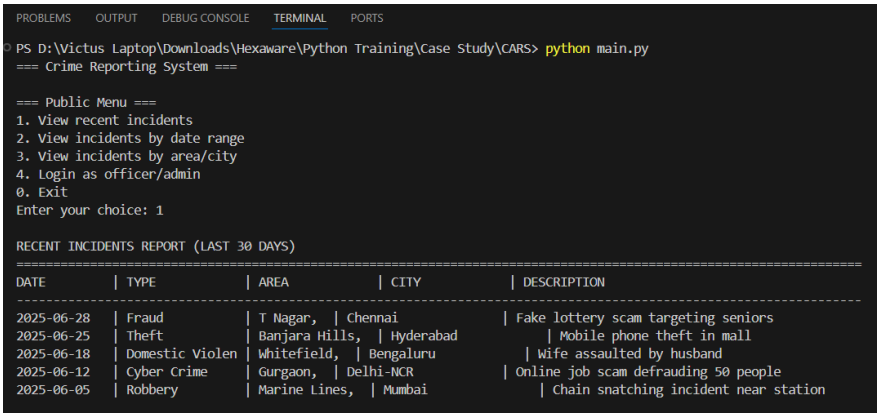
- Users → Incidents: One-to-many (an officer can handle multiple incidents).
- Incidents → Victims: One-to-many.
- Incidents → Suspects: One-to-many.
- Incidents → Evidence: One-to-many.
- Cases ↔ Incidents: Many-to-many through Incident_Case_Link.
- Suspects → Criminals: Optional one-to-one (if suspect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

ER Diagram:

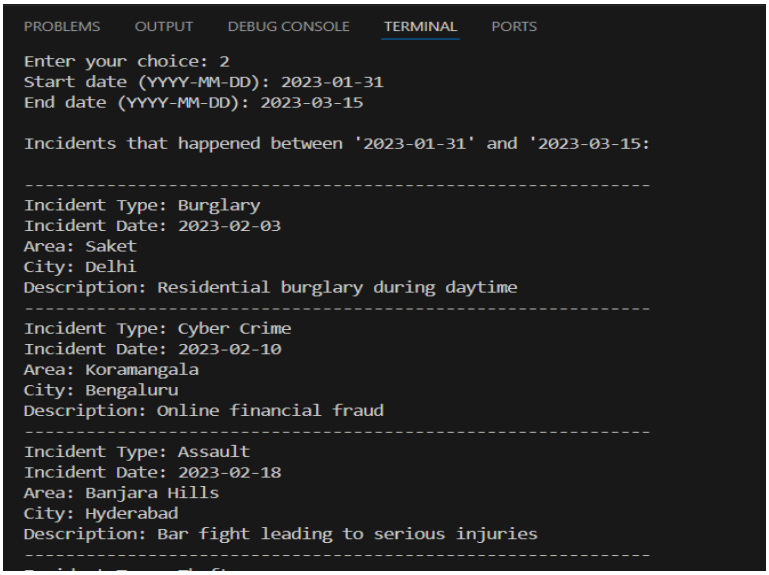

victims
VictimID INT
FirstName VARCHAR(50)
LastName VARCHAR(50)
DateOfBirth DATE
Gender VARCHAR(10)
ResidentialAddress VARCHAR(150)
ContactNumber VARCHAR(15)
AadhaarNumber VARCHAR(20)
IncidentID INT
Indexes
useractivitylog
LogID INT
UserID INT
Action VARCHAR(255)
ActionTime DATETIME
Role VARCHAR(10)
Indexes
reports
ReportID INT
IncidentID INT
ReportingOfficer I...
ReportDate DATE
ReportDetails TEXT
Status ENUM(...)
Indexes
suspects
SuspectID INT
FirstName VARCHAR(50)
LastName VARCHAR(50)
DateOfBirth DATE
Gender VARCHAR(10)
ResidentialAddress VARCHAR(150)
ContactNumber VARCHAR(15)
AadhaarNumber VARCHAR(20)
IncidentID INT
CriminalID INT
Indexes
cases
CaseID INT
CaseDescription TE...
Indexes
criminals
CriminalID INT
IncidentID INT
SuspectID INT
AadhaarNumber VARCHAR(2...
PunishmentDetails TEXT
CrimeDetails VARCHAR(255)
Indexes
evidence
EvidenceID INT
EvidenceType VARCHAR(100)
Description TEXT
LocationFound VARCHAR(15...
IncidentID INT
Indexes
lawenforcementagenci...
AgencyID INT
AgencyName VARCHAR(100)
Jurisdiction VARCHAR(100)
EmailAddress VARCHAR(100)
Indexes
officers
OfficerID INT
FirstName VARCHAR(50)
LastName VARCHAR(50)
DateOfBirth DATE
Gender VARCHAR(10)
BadgeNumber VARCHAR(20)
OfficerRank VARCHAR(50)
ResidentialAddress VARCHAR(150)
ContactNumber VARCHAR(15)
AgencyID INT
Indexes
caseincide...
CaseID INT
IncidentID INT
Indexes
users
UserID INT
Username VARCHAR(50)
Password VARCHAR(100)
Role ENUM(...)
Indexes
incidents
IncidentID INT
IncidentType VARCHAR(5...
IncidentDate DATE
Area VARCHAR(100)
City VARCHAR(100)
Description TEXT
Status VARCHAR(50)
OfficerID INT
Indexes

Console Output:

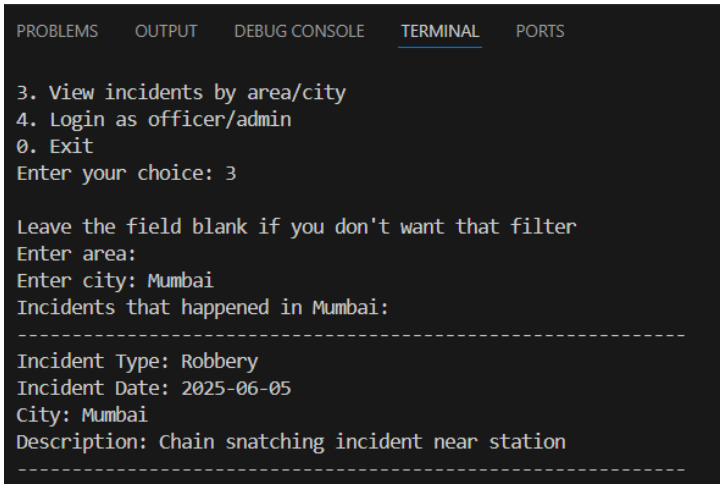
1. Public Menu: Viewing recent incidents



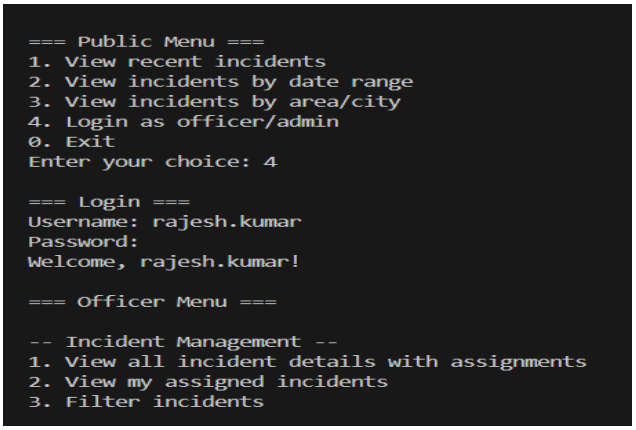
2. Public Menu: View incidents by date range



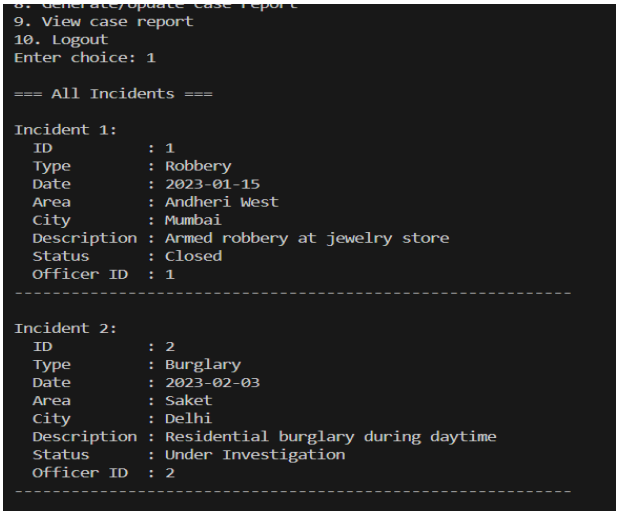
3. Public Menu: View incidents by area/city



4. Public Menu: Login as officer



5. Officer Menu: View all incident details with assignments



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
```

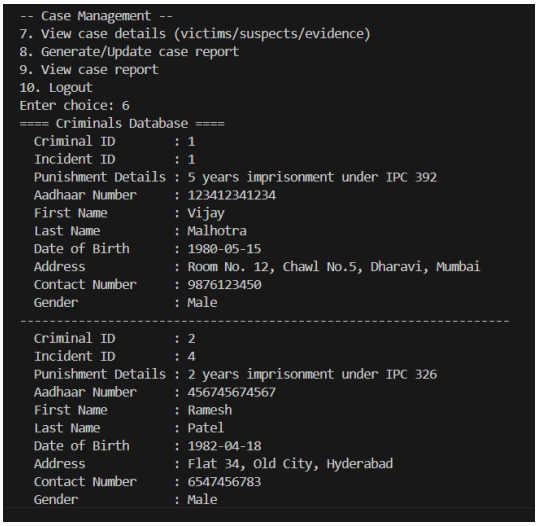
8. Officer Menu: Access suspects database

```
-- 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
------------------------------------------------------------
```

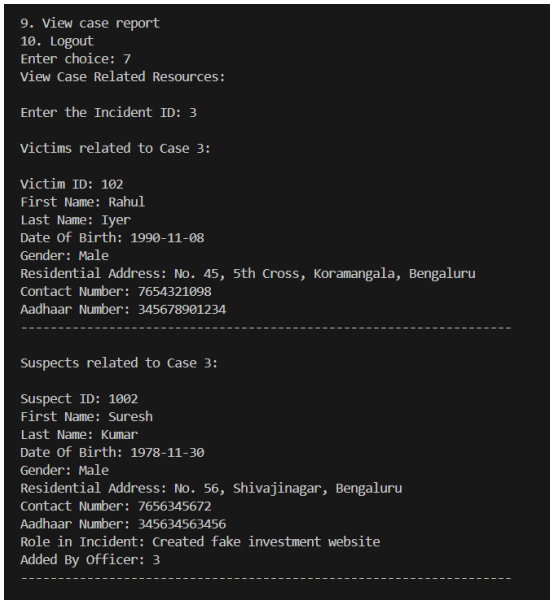
9. Officer Menu: Create new suspect record

```
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
```

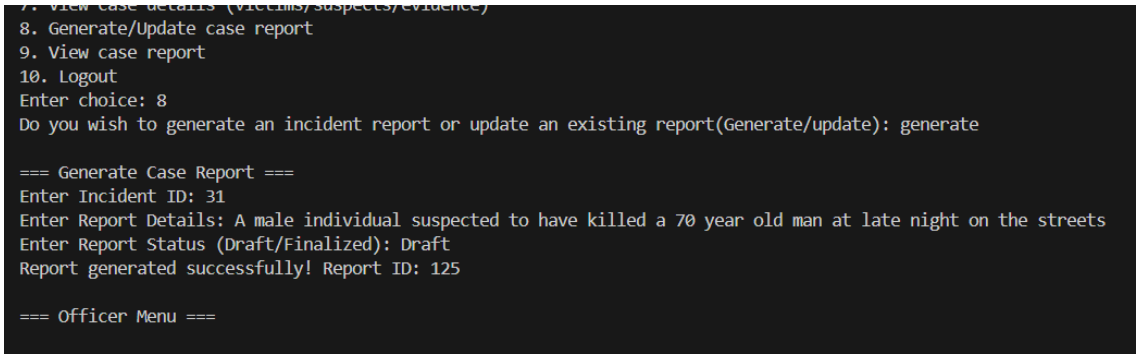
10.Officer Menu: Access criminals database



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



12.Officer Menu: Generate/Update case report



```
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
--------------------------------
```

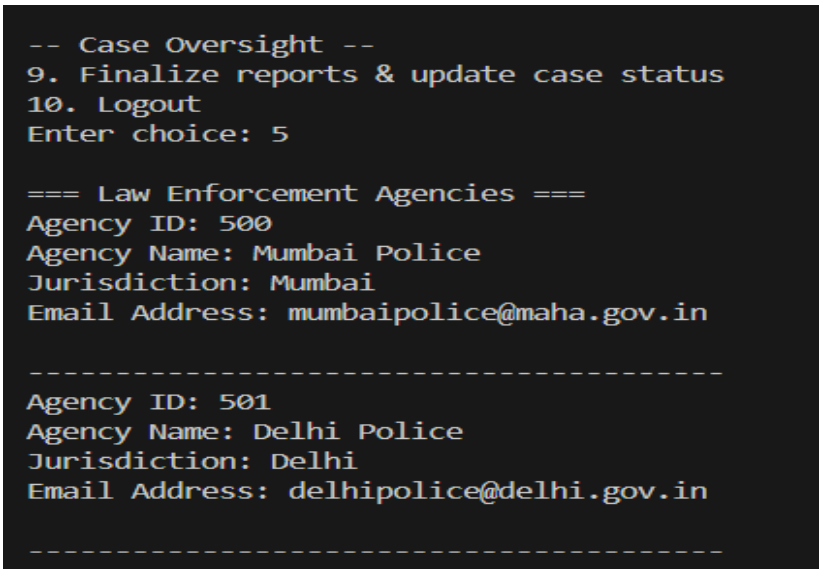
```
Incident Date (YYYY-MM-DD): 2025-06-30
Area: Porur
City: Chennai
Description: A late night street murder of an old man

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

19.Admin Menu: View all law enforcement agencies



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.
```

Implementation and Testing :

Dao Folder :

#CrimeAnalysisService.py

CrimeAnalysisService.py

dao > CrimeAnalysisService.py > ICrimeAnalysisService > get_incidents_in_date_range_public

```python
from abc import ABC, abstractmethod

class ICrimeAnalysisService(ABC):
    @abstractmethod
    def create_incident(self, incident):
        pass

    @abstractmethod
    def update_incident_status(self, status, incidentID):
        pass

    @abstractmethod
    def get_incidents_in_date_range_public(self, start_date, end_date):
        pass

    @abstractmethod
    def get_incidents_by_area_city_public(self, area, city):
        pass

    @abstractmethod
    def get_filtered_incidents(self, start_date, end_date, area, city, incident_type, officer_id):
        pass

    @abstractmethod
    def create_officer(self, officer_data):
        pass

    @abstractmethod
    def get_officers_by_location(self, city, area):
        pass

    @abstractmethod
    def count_open_incidents_by_officer(self, officer_id):
        pass
```

dao > CrimeAnalysisService.py > ICrimeAnalysisService > get_incidents_in_date_range_public

```python
class ICrimeAnalysisService(ABC):
    @abstractmethod
    def generate_incident_report(self, report):
        pass

    @abstractmethod
    def view_incident_report(self, incidentID):
        pass

    @abstractmethod
    def get_victims_by_incident(self, incidentID):
        pass

    @abstractmethod
    def get_suspects_by_incident(self, incidentID):
        pass

    @abstractmethod
    def get_evidence_by_incident(self, incidentID):
        pass

    @abstractmethod
    def update_incident_status(self, incidentID, updatedReport):
        pass

    @abstractmethod
    def viewIncident(self, incidentID):
        pass

    @abstractmethod
    def viewMyIncidents(self,officerID):
        pass

    @abstractmethod
    def viewAllIncidents(self):
        pass
```

```python
class ICrimeAnalysisService(ABC):
    @abstractmethod
    def suspectCriminalRelation(self, aadhaarID):
        pass

    @abstractmethod
    def addSuspect(self, suspect):
        pass

    @abstractmethod
    def addCriminal(self, criminal):
        pass
    @abstractmethod
    def viewAllSuspects(self):
        pass
    @abstractmethod
    def viewAllCriminals(self):
        pass

    @abstractmethod
    def get_crime_analytics(self):
        pass

    @abstractmethod
    def get_criminal_analytics(self):
        pass
    @abstractmethod
    def get_all_officer_dashboard(self):
        pass
    @abstractmethod
    def print_recent_incidents(self):
        pass

    @abstractmethod
    def get_reports_by_status(self, status):
        pass
```

#CrimeAnalysisServiceImpl.py

dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl

```python
from dao.CrimeAnalysisService import ICrimeAnalysisService
from util.DB_Connections import DBConnections
from entity.Incidents import Incidents
from entity.Victims import Victims
from entity.Suspects import Suspects
from entity.Criminals import Criminals
from entity.Evidence import Evidence
from entity.SuspectIncidents import SuspectIncidents
from entity.LawEnforcementAgencies import LawEnforcementAgencies
from exceptions.user_defined_exceptions import DatabaseError, IncidentNotFoundException, DuplicateEntryException, SuspectNotFoundException,


class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def __init__(self):
        self.connection = DBConnections.get_connection('db.properties')

    def create_incident(self, incident):
        try:
            cursor = self.connection.cursor()
            query = """
            INSERT INTO Incidents (IncidentType, IncidentDate, Area, City,
                                   Description, Status, OfficerID)
            VALUES (%s, %s, %s, %s, %s, %s, %s)
            """
            cursor.execute(query, (
                incident.incidentType, incident.incidentDate, incident.area,
                incident.city, incident.description, incident.status, incident.officerID
            ))
            self.connection.commit()
            return cursor.lastrowid

        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to create incident: {str(e)}")
```

---

dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl

```python
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def update_incident_status(self, status, incidentID):
        try:
            cursor = self.connection.cursor()
            query = "UPDATE Incidents SET Status = %s WHERE IncidentID = %s"
            cursor.execute(query, (status, incidentID))
            self.connection.commit()
            return cursor.rowcount > 0

        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to update incident status: {str(e)}")

    def get_incidents_in_date_range_public(self, start_date, end_date):
        try:
            incidents = []
            cursor = self.connection.cursor(dictionary=True)
            query = """
            SELECT IncidentType, IncidentDate, Area, City, Description FROM Incidents
            WHERE IncidentDate BETWEEN %s AND %s
            ORDER BY IncidentDate
            """
            cursor.execute(query, (start_date, end_date))

            for row in cursor:
                incidents.append(Incidents(
                    IncidentType = row['IncidentType'],
                    IncidentDate = row['IncidentDate'],
                    Area = row['Area'],
                    City = row['City'],
                    Description = row['Description']
                ))
            return incidents
        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch incidents: {str(e)}")
```

```
o > 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         except IncidentNotFoundException as e:
101             print("Incident not found.")
102         except Exception as e:
103             raise DatabaseError(f"Error fetching incidents by area/city: {str(e)}")
104
105     def get_filtered_incidents(self, start_date=None, end_date=None, area=None, city=None, incident_type=None, officer_id=None):
106
107         incidents = []
108         try:
109             cursor = self.connection.cursor(dictionary=True)
110
111             query = "SELECT * FROM Incidents WHERE 1=1"
112             params = []
113
114             if start_date:
115                 query += " AND IncidentDate >= %s"
116                 params.append(start_date)
117             if end_date:
118                 query += " AND IncidentDate <= %s"
119                 params.append(end_date)
120             if area:
121                 query += " AND Area = %s"
122                 params.append(area)
123             if city:
124                 query += " AND City = %s"
125                 params.append(city)
126             if incident_type:
127                 query += " AND IncidentType = %s"
128                 params.append(incident_type)
129             if officer_id:
130                 query += " AND OfficerID = %s"
131                 params.append(officer_id)
132
133             query += " ORDER BY IncidentDate DESC"
134
135             cursor.execute(query, params)
136
137             for row in cursor:
138                 incidents.append({
139                     'IncidentID': row['IncidentID'],
140                     'IncidentType': row['IncidentType'],
141                     'IncidentDate': row['IncidentDate'].strftime('%Y-%m-%d') if row['IncidentDate'] else None,
142                     'Area': row['Area'],
143                     'City': row['City'],
144                     'Description': row['Description'],
145                     'Status': row['Status'],
146                     'OfficerID': row['OfficerID']
147                 })
148             return incidents
149
150         except IncidentNotFoundException as e:
151             print("Incident not found.")
152         except Exception as e:
153             raise DatabaseError(f"Filter error: {str(e)}")
154
```

```python
CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_incidents_by_area_city_public
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def create_officer(self, officer_data):
        try:
            cursor = self.connection.cursor()
            cursor.execute("SELECT COUNT(*) FROM lawenforcementagencies WHERE AgencyID = %s",
                           (officer_data['agencyID'],))
            result = cursor.fetchone()
            if result[0] == 0:
                raise AgencyNotFoundException(f"AgencyID {officer_data['agencyID']} does not exist.")
            officer_query = """
            INSERT INTO Officers (FirstName, LastName, DateOfBirth, Gender, BadgeNumber, Ranking, PostingCity, PostingState, ServiceJoiningDate, ResidentialAddress, ContactNumber, AgencyID)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
            """
            cursor.execute(officer_query, (
                officer_data['firstName'],
                officer_data['lastName'],
                officer_data['dateOfBirth'],
                officer_data['gender'],
                officer_data['badgeNumber'],
                officer_data['ranking'],
                officer_data['postingCity'],
                officer_data['postingState'],
                officer_data['serviceJoiningDate'],
                officer_data['residentialAddress'],
                officer_data['contactNumber'],
                officer_data['agencyID']
            ))
            officerID = cursor.lastrowid

            user_query = """
            INSERT INTO Users (Username, Password, Role, OfficerID)
            VALUES (%s, %s, %s, %s)
            """
            cursor.execute(user_query, (
                officer_data['username'],
                officer_data['password'],
                'Officer', officerID
            ))

            self.connection.commit()
            return officerID

        except AgencyNotFoundException as e:
            print("Agency not found.")
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to add officer: {str(e)}")

    def get_officers_by_location(self, city, area):
        try:
            cursor = self.connection.cursor(dictionary=True)
            query = """
            SELECT * FROM Officers
            WHERE PostingCity = %s OR PostingState = %s
            """
            cursor.execute(query, (city, area))
            return cursor.fetchall()

        except OfficerNotFoundException as e:
            print("Officers not found. ")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch officers: {str(e)}")

    def count_open_incidents_by_officer(self, officer_id):
        try:
            cursor = self.connection.cursor()
            query = """
            SELECT COUNT(*) FROM Incidents
            WHERE OfficerID = %s AND Status != 'Closed'
            """
            cursor.execute(query, (officer_id,))
            count = cursor.fetchone()[0]
            return count

        except OfficerNotFoundException as e:
            print("Officers not found. ")
        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            raise DatabaseError(f"Failed to count open cases: {str(e)}")
```

```
meAnalysisService.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

CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > generate_incident_report

```
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
317
318     def view_incident_report(self, incidentID):
319         try:
320             cursor = self.connection.cursor(dictionary=True)
321             query = """
322                 SELECT i.IncidentID, i.IncidentType, i.Area, i.City, i.Description, i.Status,
323                 r.ReportDetails, r.ReportDate, r.Status AS ReportStatus
324                 FROM Incidents i
325                 JOIN Reports r
326                 ON i.IncidentID = r.IncidentID
327                 WHERE i.IncidentID = %s
328             """
329             cursor.execute(query, (incidentID,))
330             report = cursor.fetchone()
331             return report
332
333         except IncidentNotFoundException as e:
334             print("Incident not found.")
335         except Exception as e:
336             raise DatabaseError(f"Failed to display report: {str(e)}")
337
338     def get_victims_by_incident(self, incidentID):
339         try:
340             cursor = self.connection.cursor()
341             query = "SELECT * FROM Victims WHERE IncidentID = %s"
342             cursor.execute(query, (incidentID,))
343             results = cursor.fetchall()
344
345             victims = []
346             for result in results:
347                 victim = Victims(
348                     VictimID = result[0],
349                     FirstName = result[1],
350                     LastName = result[2],
351                     DateOfBirth = result[3],
352                     Gender = result[4],
353                     ResidentialAddress = result[5],
354                     ContactNumber = result[6],
355                     AadhaarNumber = result[7],)
356                 victims.append(victim)
357
358             return victims
359
360         except IncidentNotFoundException as e:
361             print("Incident not found.")
362         except VictimNotFoundException as e:
363             print("Victim not found.")
364         except Exception as e:
365             raise DatabaseError(f"Failed to fetch victims list: {str(e)}")
```

CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > generate_incident_report

```python
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def get_suspects_by_incident(self, incidentID):
        try:
            cursor = self.connection.cursor()
            query = """
                SELECT S.*, SI.IncidentID, SI.AddedByOfficerID, SI.RoleDescription
                FROM Suspects S
                JOIN SuspectIncidents SI ON S.AadhaarNumber = SI.AadhaarNumber
                WHERE SI.IncidentID = %s
            """
            cursor.execute(query, (incidentID,))
            results = cursor.fetchall()

            suspect_incidents = []
            for row in results:
                suspect = Suspects(
                    SuspectID=row[0],
                    FirstName=row[1],
                    LastName=row[2],
                    DateOfBirth=row[3],
                    Gender=row[4],
                    ResidentialAddress=row[5],
                    ContactNumber=row[6],
                    AadhaarNumber=row[7]
                )

                suspect_incident = SuspectIncidents(
                    Suspects=suspect,
                    Incidents=row[8],
                    Officers=row[9],
                    RoleDescription=row[10]
                )
                suspect_incidents.append(suspect_incident)

            return suspect_incidents

        except SuspectNotFoundException as e:
            print("Suspect not found.")
        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch suspects list: {str(e)}")
```

CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > generate_incident_report

```python
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def get_evidence_by_incident(self, incidentID):
        try:
            cursor = self.connection.cursor()
            query = "SELECT * FROM Evidence WHERE IncidentID = %s"
            cursor.execute(query, (incidentID,))
            results = cursor.fetchall()

            evidences = []
            for result in results:
                evidence = Evidence(
                    EvidenceID = result[0],
                    EvidenceName = result[1],
                    Description = result[2],
                    LocationFound = result[3])
                evidences.append(evidence)

            return evidences

        except IncidentNotFoundException as e:
            print("Incident not found.")
        except EvidenceNotFoundException as e:
            print("Victim not found.")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch evidence list: {str(e)}")

    def update_incident_status(self, incidentID, updatedReport):
        try:
            cursor = self.connection.cursor()
            query = "UPDATE Incidents SET Status = %s, ReportDetails = %s, ReportDate = %s WHERE IncidentID = %s"
            cursor.execute(query, (updatedReport.status, updatedReport.reportDetails, updatedReport.reportDate, incidentID))
            self.connection.commit()
            return cursor.rowcount

        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to update the status of the incident: {str(e)}")

    def viewIncident(self, incidentID):
        try:
            cursor = self.connection.cursor()
            query = "SELECT * FROM Incidents WHERE IncidentID = %s"
            cursor.execute(query,(incidentID))
            result = cursor.fetchone()
            return result

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

CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllIncidents

```python
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def viewMyIncidents(self,officerID):
        try:
            cursor = self.connection.cursor(dictionary=True)
            query = "SELECT * FROM Incidents WHERE OfficerID = %s AND STATUS != 'Closed'"
            cursor.execute(query,(officerID,))
            results = cursor.fetchall()

            incidents = []
            for row in results:
                incident = Incidents(
                    IncidentID=row['IncidentID'],
                    IncidentType=row['IncidentType'],
                    IncidentDate=row['IncidentDate'],
                    Area=row['Area'],
                    City=row['City'],
                    Description=row['Description'],
                    Status=row['Status'],
                    OfficerID=row['OfficerID']
                )
                incidents.append(incident)
            return results

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

    def viewAllIncidents(self):
        try:
            cursor = self.connection.cursor()
            query = "SELECT * FROM Incidents"
            cursor.execute(query)
            results = cursor.fetchall()

            incidents = []
            for result in results:
                incident = Incidents(
                    IncidentID = result[0],
                    IncidentType = result[1],
                    IncidentDate = result[2],
                    Area = result[3],
                    City = result[4],
                    Description = result[5],
                    Status = result[6],
                    OfficerID = result[7]
                )
                incidents.append(incident)

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

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

    def suspectCriminalRelation(self, aadhaarNumber):
        try:
            cursor = self.connection.cursor()
            query = """
            SELECT C.CriminalID, C.IncidentID, C.PunishmentDetails
            FROM Suspects S
            JOIN Criminals C
            ON S.AadhaarNumber = C.AadhaarNumber
            WHERE S.AadhaarNumber = %s
            """
            cursor.execute(query,(aadhaarNumber,))
            results = cursor.fetchall()

            criminalRecords = []
            for result in results:
                criminalRecord = Criminals(
                    CriminalID = result[0],
                    IncidentID = result[1],
                    PunishmentDetails = result[2]
                )
                criminalRecords.append(criminalRecord)

            return criminalRecords

        except SuspectNotFoundException as e:
            print("Suspects not found")
        except CriminalNotFoundException as e:
            print("Criminal not found")
        except Exception as e:
            raise DatabaseError(f"Failed to show the suspect and criminal relations: {str(e)}")
```

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

    def addSuspect(self, suspect, incidentID, officerID, roleDescription):
        try:
            cursor = self.connection.cursor()
            check_query = """
                SELECT SuspectID FROM Suspects
                WHERE AadhaarNumber = %s
                LIMIT 1
            """
            cursor.execute(check_query, (suspect.aadhaarNumber,))
            existing_suspect = cursor.fetchone()
            if existing_suspect:
                suspectID = existing_suspect[0]
                check_incident_query = """
                    SELECT * FROM SuspectIncidents
                    WHERE AadhaarNumber = %s AND IncidentID = %s
                """
                cursor.execute(check_incident_query, (suspect.aadhaarNumber, incidentID))
                existing_link = cursor.fetchone()
                if existing_link:
                    raise DuplicateEntryException(f"Duplicate Entry: Suspect already assigned to Incident {incidentID}.")
                else:
                    insert_incident_query = """
                        INSERT INTO SuspectIncidents (SuspectID, AadhaarNumber, IncidentID, AddedByOfficerID, RoleDescription)
                        VALUES (%s, %s, %s, %s, %s)
                    """
                    cursor.execute(insert_incident_query, (suspectID, suspect.aadhaarNumber, incidentID, officerID, roleDescription))
                    self.connection.commit()
                    return suspectID
            else:
                insert_query = """
                    INSERT INTO Suspects (
                        FirstName, LastName, DateOfBirth, Gender,
                        ResidentialAddress, ContactNumber, AadhaarNumber
                    ) VALUES (%s, %s, %s, %s, %s, %s, %s)
                """
                cursor.execute(insert_query, (
                    suspect.firstName, suspect.lastName, suspect.dateOfBirth,
                    suspect.gender, suspect.residentialAddress, suspect.contactNumber, suspect.aadhaarNumber
                ))
                suspectID = cursor.lastrowid
                insert_incident_query = """
                    INSERT INTO SuspectIncidents (SuspectID, AadhaarNumber, IncidentID, AddedByOfficerID, RoleDescription)
                    VALUES (%s, %s, %s, %s, %s)
                """
                cursor.execute(insert_incident_query, (suspectID, suspect.aadhaarNumber, incidentID, officerID, roleDescription))
                self.connection.commit()
                return suspectID
        except IncidentNotFoundException as e:
            print("Incident not found")
        except OfficerNotFoundException as e:
            print("Officer not found")
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to add suspect: {str(e)}")
```

CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > addSuspect

```python
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 > addSuspect

```python
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

```python
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def viewAllSuspects(self):
        try:
            cursor = self.connection.cursor()
            query = """
            SELECT SI.SuspectID, SI.IncidentID, SI.RoleDescription, SI.AddedByOfficerID,
            FirstName, S.LastName, S.DateOfBirth, S.AadhaarNumber, S.ResidentialAddress, S.ContactNumber,
            I.IncidentDate, I.IncidentType, I.Description, I.Status
            FROM SuspectIncidents SI JOIN Suspects S ON SI.AadhaarNumber = S.AadhaarNumber
            JOIN Incidents I ON SI.IncidentID = I.IncidentID
            ORDER BY SI.SuspectID DESC
            """
            cursor.execute(query)
            results = cursor.fetchall()

            suspectsInfo = []
            for result in results:
                suspect = Suspects(
                    SuspectID = result[0],
                    FirstName = result[4],
                    LastName = result[5],
                    DateOfBirth = result[6],
                    AadhaarNumber = result[7],
                    ResidentialAddress = result[8],
                    ContactNumber = result[9]
                )

                incident = Incidents(
                    IncidentID = result[1],
                    IncidentDate = result[10],
                    IncidentType = result[11],
                    Description = result[12],
                    Status = result[13]
                )

                suspectIncident = SuspectIncidents(
                    Suspects = suspect,
                    Incidents = incident,
                    RoleDescription = result[2],
                    Officers = result[3]
                )
                suspectsInfo.append(suspectIncident)

            return suspectsInfo

        except IncidentNotFoundException as e:
            print("Incidents not found.")
        except SuspectNotFoundException as e:
            print("Suspect not found.")
        except Exception as e:
            raise DatabaseError(f"Couldn't display all suspects: {str(e)}")
```

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

    def viewAllCriminals(self):
        try:
            cursor = self.connection.cursor()
            query = """
            SELECT C.CriminalID, C.IncidentID, C.PunishmentDetails,
            S.FirstName, S.LastName, S.DateOfBirth, S.Gender,S.ResidentialAddress,
            S.ContactNumber, S.AadhaarNumber
            FROM Criminals C
            JOIN Suspects S
            ON S.AadhaarNumber = C.AadhaarNumber
            """
            cursor.execute(query)
            results = cursor.fetchall()

            criminals = []
            for result in results:
                criminal_id      = result[0]
                incident_id      = result[1]
                aadhaar_number   = result[9]
                punishment       = result[2]
                first_name       = result[3]
                last_name        = result[4]
                dob              = result[5]
                gender           = result[6]
                address          = result[7]
                contact          = result[8]
                suspect = Suspects(
                    FirstName = first_name,
                    LastName = last_name,
                    DateOfBirth = dob,
                    Gender = gender,
                    ResidentialAddress = address,
                    ContactNumber = contact,
                    AadhaarNumber = aadhaar_number
                )
                criminal = Criminals(
                    CriminalID = criminal_id,
                    IncidentID = incident_id,
                    AadhaarNumber = aadhaar_number,
                    PunishmentDetails = punishment
                )
                criminal.suspect = suspect
                criminals.append(criminal)
            return criminals
        except IncidentNotFoundException as e:
            print("Incidents not found.")
        except SuspectNotFoundException as e:
            print("Suspect not found.")
        except CriminalNotFoundException as e:
            print("Criminal not found")
        except Exception as e:
            raise DatabaseError(f"Couldn't display all criminals: {str(e)}")
```

CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > viewAllCriminals

```python
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

```python
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

```python
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

```python
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'] + ', '} | "
                    f"{incident['City']:<20} | "
                    f"{incident['Description']}"
                )

        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

```python
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

```python
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):

    def close_case_by_admin(self, incident_id):
        try:
            cursor = self.connection.cursor()
            update_query = """
                UPDATE Reports
                SET Status = 'Closed'
                WHERE IncidentID = %s AND Status = 'Finalized'
            """
            cursor.execute(update_query, (incident_id,))
            self.connection.commit()
            return cursor.rowcount

        except ReportNotFoundException as e:
            print("Report not found")
        except IncidentNotFoundException as e:
            print("Incident not found")
        except Exception as e:
            self.connection.rollback()
            raise DatabaseError(f"Failed to close case: {str(e)}")
```

Database Folder

#CARS_CreatingTables.sql

database > CARS_CreatingTables.sql > ...

```sql
CREATE DATABASE CrimeReportingSystem;
USE CrimeReportingSystem;

-- Creating LawEnforcementAgencies table
CREATE TABLE LawEnforcementAgencies
(
    AgencyID INT AUTO_INCREMENT PRIMARY KEY,
    AgencyName VARCHAR(150),
    Jurisdiction VARCHAR(100),
    EmailAddress VARCHAR(250)
)AUTO_INCREMENT = 500;

-- Creating Officers Table
CREATE TABLE Officers (
    OfficerID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    Gender VARCHAR(20),
    BadgeNumber VARCHAR(20),
    Ranking VARCHAR(30),
    PostingCity VARCHAR(30),
    PostingState VARCHAR(30),
    ServiceJoiningDate DATE,
    ResidentialAddress TEXT,
    ContactNumber BIGINT,
    AgencyID INT,
    CONSTRAINT fk_Officers_LawEnforcementAgencies
    FOREIGN KEY(AgencyID)
    REFERENCES LawEnforcementAgencies(AgencyID)
)AUTO_INCREMENT = 1;

```

```sql
-- Creating Incidents Table
CREATE TABLE Incidents (
    IncidentID INT AUTO_INCREMENT PRIMARY KEY,
    IncidentType VARCHAR(100),
    IncidentDate DATE,
    Area VARCHAR(100),
    City VARCHAR(100),
    Description TEXT,
    Status ENUM ('Open', 'Under Investigation', 'Closed') NOT NULL DEFAULT 'Open',
    OfficerID INT,
    CONSTRAINT fk_Incidents_Officers
    FOREIGN KEY(OfficerID)
    REFERENCES Officers(OfficerID)
)AUTO_INCREMENT = 1;

-- Creating Victims table
CREATE TABLE Victims (
    VictimID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    Gender VARCHAR(50),
    ResidentialAddress TEXT,
    ContactNumber BIGINT,
    AadhaarNumber BIGINT,
    IncidentID INT,
    CONSTRAINT fk_Victims_Incidents
    FOREIGN KEY(IncidentID)
    REFERENCES Incidents(IncidentID)
)AUTO_INCREMENT = 100;
```

database > CARS_CreatingTables.sql > ...

```sql
-- Creating Suspects table
CREATE TABLE Suspects
(
    SuspectID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    DateOfBirth DATE,
    Gender VARCHAR(50),
    ResidentialAddress TEXT,
    ContactNumber BIGINT,
    AadhaarNumber BIGINT UNIQUE
)AUTO_INCREMENT = 1000;

-- Creating Evidence Table
CREATE TABLE Evidence (
    EvidenceID INT AUTO_INCREMENT PRIMARY KEY,
    EvidenceName VARCHAR(20),
    Description TEXT,
    LocationFound TEXT,
    IncidentID INT,
    CONSTRAINT fk_Evidence_Incidents
    FOREIGN KEY(IncidentID)
    REFERENCES Incidents(IncidentID)
)AUTO_INCREMENT = 1;

-- Creating Reports Table
CREATE TABLE Reports (
    ReportID INT AUTO_INCREMENT PRIMARY KEY,
    IncidentID INT,
    ReportingOfficerID INT,
    ReportDate DATE,
    ReportDetails TEXT,
    STATUS ENUM('Draft', 'Finalized', 'Closed') NOT NULL DEFAULT 'Draft',
    CONSTRAINT fk_Reports_Incidents FOREIGN KEY(IncidentID) REFERENCES Incidents(IncidentID),
    CONSTRAINT fk_Reports_Officers FOREIGN KEY(ReportingOfficerID) REFERENCES Officers(OfficerID)
)AUTO_INCREMENT = 100;
```

database > CARS_CreatingTables.sql > ...

```sql

CREATE TABLE Criminals (
    CriminalID INT AUTO_INCREMENT PRIMARY KEY,
    IncidentID INT,
    SuspectID INT,
    AadhaarNumber BIGINT UNIQUE,
    PunishmentDetails TEXT,
    CONSTRAINT fk_Criminals_Incidents FOREIGN KEY(IncidentID) REFERENCES Incidents(IncidentID),
    CONSTRAINT fk_Criminals_Suspects FOREIGN KEY(SuspectID) REFERENCES Suspects(SuspectID)
)AUTO_INCREMENT = 1;

CREATE TABLE Users (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(50) UNIQUE NOT NULL,
    Password VARCHAR(100) NOT NULL,
    Role ENUM('Officer', 'Admin') NOT NULL,
    OfficerID INT NULL,
    FOREIGN KEY (OfficerID) REFERENCES Officers(OfficerID)
);

CREATE TABLE SuspectIncidents (
    SuspectID INT,
    IncidentID INT,
    AadhaarNumber BIGINT,
    RoleDescription TEXT,
    AddedByOfficerID INT,
    PRIMARY KEY(SuspectID, IncidentID),
    CONSTRAINT fk_SuspectIncidents_Suspects
    FOREIGN KEY(SuspectID)
    REFERENCES Suspects(SuspectID),
    CONSTRAINT fk_SuspectIncidents_Incidents
    FOREIGN KEY(IncidentID)
    REFERENCES Incidents(IncidentID),
    CONSTRAINT fk_SuspectIncidents_Officers
    FOREIGN KEY(AddedByOfficerID)
    REFERENCES Officers(OfficerID)
);
```

#CARS_InsertionOfData.sql

```sql
-- Inserting data in Law Enforcement Agencies table
INSERT INTO LawEnforcementAgencies (AgencyName, Jurisdiction, EmailAddress) VALUES
('Mumbai Police', 'Mumbai', 'mumbaipolice@maha.gov.in'),
('Delhi Police', 'Delhi', 'delhipolice@delhi.gov.in'),
('Bengaluru City Police', 'Bengaluru', 'bcp@kar.gov.in'),
('Hyderabad Police', 'Hyderabad', 'hydpolice@telangana.gov.in'),
('Chennai City Police', 'Chennai', 'chennaipolice@tn.gov.in'),
('Kolkata Police', 'Kolkata', 'kolkatapolice@wb.gov.in'),
('Pune City Police', 'Pune', 'punepolice@maha.gov.in'),
('Ahmedabad Police', 'Ahmedabad', 'ahdpolice@gujarat.gov.in'),
('Jaipur Police', 'Jaipur', 'jaipurpolice@rajasthan.gov.in'),
('Lucknow Police', 'Lucknow', 'lucknowpolice@up.gov.in'),
('Chandigarh Police', 'Chandigarh', 'chdpolice@chd.gov.in'),
('Kochi City Police', 'Kochi', 'kochipolice@kerala.gov.in'),
('Bhopal Police', 'Bhopal', 'bhopalpolice@mp.gov.in'),
('Patna Police', 'Patna', 'patnapolice@bihar.gov.in'),
('Surat City Police', 'Surat', 'suratpolice@gujarat.gov.in'),
('Nagpur Police', 'Nagpur', 'nagpurpolice@maha.gov.in'),
('Indore Police', 'Indore', 'indorepolice@mp.gov.in'),
('Thane Police', 'Thane', 'thanepolice@maha.gov.in'),
('Ghaziabad Police', 'Ghaziabad', 'ghaziabadpolice@up.gov.in'),
('Coimbatore Police', 'Coimbatore', 'coimbatorepolice@tn.gov.in');

-- Inserting data in Officers table
INSERT INTO Officers (FirstName, LastName, DateOfBirth, Gender, BadgeNumber, Ranking, PostingCity, PostingState, ServiceJoiningDate, ResidentialAddress, ContactNumber, AgencyID) VALUES
('Rajesh', 'Kumar', '1988-05-15', 'Male', 'MH-4521', 'Inspector', 'Mumbai', 'Maharashtra', '2005-07-10', 'Flat 12, Shivaji Nagar, Mumbai', 9876543210, 500),
('Priya', 'Sharma', '1985-08-22', 'Female', 'DL-7854', 'Sub-Inspector', 'Delhi', 'Delhi', '2010-03-15', 'H.No. 45, RK Puram, Delhi', 8765432109, 501),
('Vikram', 'Singh', '1978-11-30', 'Male', 'KA-3214', 'Deputy Commissioner', 'Bengaluru', 'Karnataka', '2000-12-05', 'No. 56, Koramangala, Bengaluru', 7654321098, 502),
('Anjali', 'Patel', '1982-04-18', 'Female', 'TS-6547', 'Inspector', 'Hyderabad', 'Telangana', '2006-09-20', 'Flat 34, Banjara Hills, Hyderabad', 6543210987, 503),
('Arun', 'Iyer', '1987-07-25', 'Male', 'TN-9874', 'Sub-Inspector', 'Chennai', 'Tamil Nadu', '2012-05-12', 'No. 78, T Nagar, Chennai', 9432109876, 504),
('Meera', 'Banerjee', '1975-02-14', 'Female', 'WB-3698', 'Inspector', 'Kolkata', 'West Bengal', '1998-11-08', '12B, Park Street, Kolkata', 8321098765, 505),
('Sanjay', 'Deshmukh', '1983-09-09', 'Male', 'MH-8521', 'Sub-Inspector', 'Pune', 'Maharashtra', '2008-06-30', 'Flat 5, Kothrud, Pune', 7210987654, 506),
('Neha', 'Gupta', '1988-12-03', 'Female', 'GJ-7412', 'Constable', 'Ahmedabad', 'Gujarat', '2015-04-22', 'H.No. 32, Navrangpura, Ahmedabad', 6109876543, 507),
('Rahul', 'Sharma', '1979-06-28', 'Male', 'RJ-9632', 'Inspector', 'Jaipur', 'Rajasthan', '2004-08-17', 'No. 15, Malviya Nagar, Jaipur', 5098765432, 508),
('Sunita', 'Yadav', '1984-03-19', 'Female', 'UP-2587', 'Sub-Inspector', 'Lucknow', 'Uttar Pradesh', '2009-10-05', 'H.No. 7, Gomti Nagar, Lucknow', 4087654321, 509),
('Amit', 'Khanna', '1981-07-22', 'Male', 'CH-1478', 'Inspector', 'Chandigarh', 'Chandigarh', '2007-01-14', 'Sector 22B, Chandigarh', 3876543210, 510),
('Deepa', 'Nair', '1986-10-11', 'Female', 'KL-3698', 'Sub-Inspector', 'Kochi', 'Kerala', '2013-09-28', 'Flat 8, Marine Drive, Kochi', 2765432109, 511),
('Vijay', 'Malhotra', '1977-04-05', 'Male', 'MP-6541', 'Deputy Commissioner', 'Bhopal', 'Madhya Pradesh', '1999-12-15', 'No. 42, Arera Colony, Bhopal', 1654321098, 512),
('Pooja', 'Verma', '1989-01-30', 'Female', 'BR-9873', 'Constable', 'Patna', 'Bihar', '2016-07-08', 'H.No. 23, Patliputra Colony, Patna', 9543210987, 513),
('Ramesh', 'Joshi', '1980-08-17', 'Male', 'GJ-3214', 'Inspector', 'Surat', 'Gujarat', '2005-11-25', 'Flat 16, Athwa Lines, Surat', 8432109876, 514),
('Smita', 'Reddy', '1983-05-14', 'Female', 'MH-7852', 'Sub-Inspector', 'Mumbai', 'Maharashtra', '2008-02-18', 'Flat 9, Andheri East, Mumbai', 7321098765, 500),
('Alok', 'Mishra', '1976-12-08', 'Male', 'DL-4569', 'Inspector', 'Delhi', 'Delhi', '2001-04-30', 'H.No. 34, Saket, Delhi', 6210987654, 501),
('Kavita', 'Singh', '1987-09-21', 'Female', 'KA-1478', 'Constable', 'Bengaluru', 'Karnataka', '2014-08-12', 'No. 51, Indiranagar, Bengaluru', 5109876543, 502),
('Prakash', 'Rao', '1982-02-25', 'Male', 'TS-9632', 'Sub-Inspector', 'Hyderabad', 'Telangana', '2007-06-19', 'Flat 22, Jubilee Hills, Hyderabad', 4098765432, 503),
('Anita', 'Menon', '1985-11-13', 'Female', 'TN-2587', 'Inspector', 'Chennai', 'Tamil Nadu', '2010-03-07', 'No. 67, Adyar, Chennai', 3987654321, 504),
('Rajiv', 'Bose', '1978-07-04', 'Male', 'WB-7412', 'Deputy Commissioner', 'Kolkata', 'West Bengal', '2003-10-22', '15C, Salt Lake, Kolkata', 2876543210, 505),
('Shweta', 'Pawar', '1989-04-16', 'Female', 'MH-3698', 'Constable', 'Pune', 'Maharashtra', '2017-01-09', 'Flat 3, Wakad, Pune', 1765432109, 506),
('Manoj', 'Patel', '1981-10-29', 'Male', 'GJ-6541', 'Sub-Inspector', 'Ahmedabad', 'Gujarat', '2006-05-14', 'H.No. 18, Satellite, Ahmedabad', 8654321098, 507),
('Divya', 'Sharma', '1984-03-07', 'Female', 'RJ-9873', 'Inspector', 'Jaipur', 'Rajasthan', '2009-09-26', 'No. 24, Vaishali Nagar, Jaipur', 9543210987, 508),
('Vivek', 'Trivedi', '1979-08-12', 'Male', 'UP-3214', 'Deputy Commissioner', 'Lucknow', 'Uttar Pradesh', '2004-12-03', 'H.No. 9, Alambagh, Lucknow', 8432109876, 509),
('Anjali', 'Mehta', '1986-01-24', 'Female', 'CH-7852', 'Sub-Inspector', 'Chandigarh', 'Chandigarh', '2011-07-17', 'Sector 35C, Chandigarh', 7321098765, 510),
('Sunil', 'Nair', '1980-06-18', 'Male', 'KL-4569', 'Inspector', 'Kochi', 'Kerala', '2005-11-08', 'Flat 11, Panampilly Nagar, Kochi', 6210987654, 511),
('Preeti', 'Verma', '1983-09-05', 'Female', 'MP-1478', 'Sub-Inspector', 'Bhopal', 'Madhya Pradesh', '2008-04-21', 'No. 33, Shahpura, Bhopal', 5109876543, 512),
('Rakesh', 'Yadav', '1977-02-28', 'Male', 'BR-9632', 'Inspector', 'Patna', 'Bihar', '2002-08-15', 'H.No. 5, Danapur, Patna', 4098765432, 513),
('Sonia', 'Kapoor', '1988-07-19', 'Female', 'GJ-2587', 'Constable', 'Surat', 'Gujarat', '2015-02-11', 'Flat 7, Vesu, Surat', 3087654321, 514),
('Nitin', 'Shah', '1982-09-28', 'Male', 'MH-8523', 'Inspector', 'Nagpur', 'Maharashtra', '2007-08-14', 'Flat 5, Sitabuldi, Nagpur', 9876543201, 515),
('Ananya', 'Desai', '1987-04-15', 'Female', 'MP-7531', 'Sub-Inspector', 'Indore', 'Madhya Pradesh', '2012-11-22', 'No. 12, Vijay Nagar, Indore', 8765432108, 516),
('Vikas', 'Malhotra', '1980-12-03', 'Male', 'MH-9635', 'Inspector', 'Thane', 'Maharashtra', '2006-05-19', 'Flat 8, Ghodbunder Road, Thane', 7654321097, 517),
('Rashmi', 'Saxena', '1985-07-21', 'Female', 'UP-8524', 'Sub-Inspector', 'Ghaziabad', 'Uttar Pradesh', '2010-09-30', 'H.No. 34, Raj Nagar, Ghaziabad', 6543210986, 518),
('Sanjeev', 'Iyer', '1979-02-14', 'Male', 'TN-7418', 'Inspector', 'Coimbatore', 'Tamil Nadu', '2004-10-25', 'No. 56, RS Puram, Coimbatore', 5432109875, 519);

-- Inserting data in Incidents table
INSERT INTO Incidents (IncidentType, IncidentDate, Area, City, Description, Status, OfficerID) VALUES
('Robbery', '2023-01-15', 'Andheri West', 'Mumbai', 'Armed robbery at jewelry store', 'Closed', 1),
('Burglary', '2023-02-03', 'Saket', 'Delhi', 'Residential burglary during daytime', 'Under Investigation', 2),
('Cyber Crime', '2023-02-10', 'Koramangala', 'Bengaluru', 'Online financial fraud', 'Open', 3),
('Assault', '2023-02-18', 'Banjara Hills', 'Hyderabad', 'Bar fight leading to serious injuries', 'Closed', 4),
('Theft', '2023-03-05', 'T Nagar', 'Chennai', 'Pickpocketing in crowded market', 'Under Investigation', 5),
('Homicide', '2023-03-12', 'Park Street', 'Kolkata', 'Domestic dispute turned violent', 'Under Investigation', 6),
('Vehicle Theft', '2023-03-20', 'Kothrud', 'Pune', 'Motorcycle stolen from parking lot', 'Open', 7),
('Fraud', '2023-04-02', 'Navrangpura', 'Ahmedabad', 'Real estate scam', 'Closed', 8),
('Kidnapping', '2023-04-15', 'Malviya Nagar', 'Jaipur', 'Child abduction case', 'Under Investigation', 9),
('Domestic Violence', '2023-04-22', 'Gomti Nagar', 'Lucknow', 'Spousal abuse complaint', 'Closed', 10),
('Drug Offense', '2023-05-05', 'Sector 22', 'Chandigarh', 'Drug peddling bust', 'Closed', 11),
('Sexual Assault', '2023-05-12', 'Marine Drive', 'Kochi', 'Molestation case', 'Under Investigation', 12),
('Arson', '2023-05-20', 'Arera Colony', 'Bhopal', 'Deliberate fire at commercial property', 'Open', 13),
('Public Nuisance', '2023-06-03', 'Patliputra Colony', 'Patna', 'Loud party disturbance', 'Closed', 14),
('Hit and Run', '2023-06-10', 'Athwa Lines', 'Surat', 'Pedestrian hit by speeding vehicle', 'Under Investigation', 15),
('Vandalism', '2023-06-18', 'Andheri East', 'Mumbai', 'Graffiti on public property', 'Open', 16),
('Identity Theft', '2023-07-02', 'Saket', 'Delhi', 'Credit card fraud', 'Under Investigation', 17),
('Harassment', '2023-07-15', 'Indiranagar', 'Bengaluru', 'Workplace harassment complaint', 'Closed', 18),
('Bribery', '2023-07-22', 'Jubilee Hills', 'Hyderabad', 'Government official caught taking bribe', 'Under Investigation', 19),
('Child Abuse', '2023-08-05', 'Adyar', 'Chennai', 'School teacher accused of abuse', 'Open', 20),
('Eve Teasing', '2023-08-12', 'Salt Lake', 'Kolkata', 'Street harassment complaint', 'Closed', 21),
('Illegal Gambling', '2023-08-28', 'Wakad', 'Pune', 'Underground gambling den raid', 'Under Investigation', 22),
('Forgery', '2023-09-03', 'Satellite', 'Ahmedabad', 'Fake document racket', 'Open', 23),
('Extortion', '2023-09-10', 'Vaishali Nagar', 'Jaipur', 'Shopkeepers threatened for money', 'Under Investigation', 24),
('Trespassing', '2023-09-18', 'Alambagh', 'Lucknow', 'Unauthorized entry into private property', 'Closed', 25),
('Robbery', '2025-06-05', 'Marine Lines', 'Mumbai', 'Chain snatching incident near station', 'Under Investigation', 1),
('Cyber Crime', '2025-06-12', 'Gurgaon', 'Delhi-NCR', 'Online job scam defrauding 50 people', 'Open', 2),
('Domestic Violence', '2025-06-18', 'Whitefield', 'Bengaluru', 'Wife assaulted by husband', 'Under Investigation', 3),
('Theft', '2025-06-25', 'Banjara Hills', 'Hyderabad', 'Mobile phone theft in mall', 'Open', 4),
('Fraud', '2025-06-28', 'T Nagar', 'Chennai', 'Fake lottery scam targeting seniors', 'Under Investigation', 5);
```

```sql

-- Inserting data in Victims table
INSERT INTO Victims (FirstName, LastName, DateOfBirth, Gender, ResidentialAddress, ContactNumber, AadhaarNumber, IncidentID) VALUES
('Aarav', 'Shah', '1975-08-12', 'Male', 'Shop No.5, Juhu Road, Andheri West, Mumbai', 9876543210, 123456789012, 1),
('Neha', 'Kapoor', '1982-03-25', 'Female', 'Flat 302, Saket Apartments, Delhi', 8765432100, 234567890123, 2),
('Rahul', 'Iyer', '1990-11-08', 'Male', 'No. 45, 5th Cross, Koramangala, Bengaluru', 7654321098, 345678901234, 3),
('Priya', 'Reddy', '1988-07-14', 'Female', 'Flat 8, Banjara Residency, Hyderabad', 6543210987, 456789012345, 4),
('Vikram', 'Menon', '1979-04-30', 'Male', 'No. 12, T Nagar Main Road, Chennai', 9432109876, 567890123456, 5),
('Ananya', 'Banerjee', '1995-01-18', 'Female', '12B, Park Street, Kolkata', 8321098765, 678901234567, 6),
('Rohan', 'Deshpande', '1987-09-22', 'Male', 'Flat 3, Kothrud Society, Pune', 7210987654, 789012345678, 7),
('Meera', 'Patel', '1973-12-05', 'Female', 'H.No. 32, Navrangpura, Ahmedabad', 6109876543, 890123456789, 8),
('Arjun', 'Sharma', '2005-06-15', 'Male', 'No. 15, Malviya Nagar, Jaipur', 5098765432, 901234567890, 9),
('Sunita', 'Yadav', '1988-02-28', 'Female', 'H.No. 7, Gomti Nagar, Lucknow', 4987654321, 112233445566, 10),
('Amit', 'Khanna', '1992-10-11', 'Male', 'Sector 22B, Chandigarh', 3876543210, 223344556677, 11),
('Deepika', 'Nair', '1984-05-19', 'Female', 'Flat 8, Marine Drive, Kochi', 2765432109, 334455667788, 12),
('Vijay', 'Malhotra', '1968-08-24', 'Male', 'No. 42, Arera Colony, Bhopal', 1654321098, 445566778899, 13),
('Pooja', 'Verma', '1998-03-07', 'Female', 'H.No. 23, Patliputra Colony, Patna', 9543210987, 556677889900, 14),
('Ramesh', 'Joshi', '1977-11-15', 'Male', 'Flat 16, Athwa Lines, Surat', 8432109876, 667788990011, 15),
('Smita', 'Reddy', '1989-07-30', 'Female', 'Flat 9, Andheri East, Mumbai', 7321098765, 778899001122, 16),
('Alok', 'Mishra', '1991-04-02', 'Male', 'H.No. 34, Saket, Delhi', 6210987654, 889900112233, 17),
('Kavita', 'Singh', '1983-12-18', 'Female', 'No. 51, Indiranagar, Bengaluru', 5109876543, 990011223344, 18),
('Prakash', 'Rao', '1976-09-21', 'Male', 'Flat 22, Jubilee Hills, Hyderabad', 4098765432, 112233445500, 19),
('Anita', 'Menon', '1993-02-14', 'Female', 'No. 67, Adyar, Chennai', 3087654321, 223344556611, 20),
('Rajiv', 'Mehta', '1978-05-17', 'Male', '15C, Salt Lake, Kolkata', 2876543210, 334455667722, 21),
('Shweta', 'Pawar', '1985-09-23', 'Female', 'Flat 3, Wakad, Pune', 1765432109, 445566778833, 22),
('Manoj', 'Patel', '1988-12-07', 'Male', 'H.No. 18, Satellite, Ahmedabad', 0654321098, 556677889944, 23),
('Divya', 'Sharma', '1987-04-11', 'Female', 'No. 24, Vaishali Nagar, Jaipur', 9543210988, 667788990055, 24),
('Vivek', 'Trivedi', '1990-08-29', 'Male', 'H.No. 9, Alambagh, Lucknow', 8432109877, 778899001166, 25),
('Anjali', 'Mehta', '1982-01-14', 'Female', 'Sector 35C, Chandigarh', 7321098766, 889900112277, 26),
('Sunil', 'Nair', '1979-07-03', 'Male', 'Flat 11, Panampilly Nagar, Kochi', 6210987655, 990011223388, 27),
('Preeti', 'Verma', '1986-10-19', 'Female', 'No. 33, Shahpura, Bhopal', 5109876544, 101010101010, 28),
('Rakesh', 'Yadav', '1975-03-26', 'Male', 'H.No. 5, Danapur, Patna', 4098765433, 112211221122, 29),
('Sonia', 'Kapoor', '1988-12-09', 'Female', 'Flat 7, Vesu, Surat', 3087654322, 223322332233, 30);

-- Inserting data in Suspects table
INSERT INTO Suspects (FirstName, LastName, DateOfBirth, Gender, ResidentialAddress, ContactNumber, AadhaarNumber) VALUES
('Vijay', 'Malhotra', '1988-05-15', 'Male', 'Room No. 12, Chawl No.5, Dharavi, Mumbai', 9876123450, 123412341234),
('Raju', 'Khan', '1985-08-22', 'Male', 'H.No. 45, Seelampur, Delhi', 8765234561, 234523452345),
('Suresh', 'Kumar', '1978-11-30', 'Male', 'No. 56, Shivajinagar, Bengaluru', 7656345672, 345634563456),
('Ramesh', 'Patel', '1982-04-18', 'Male', 'Flat 34, Old City, Hyderabad', 6547456783, 456745674567),
('Mohan', 'Iyer', '1987-07-25', 'Male', 'No. 78, Washermanpet, Chennai', 9438567894, 567856785678),
('Babu', 'Banerjee', '1975-02-14', 'Male', '12B, Howrah, Kolkata', 8320678905, 678967806780),
('Sanju', 'Deshmukh', '1983-09-09', 'Male', 'Flat 5, Pimpri, Pune', 7218789016, 789078907890),
('Ravi', 'Gupta', '1988-12-03', 'Male', 'H.No. 32, Naroda, Ahmedabad', 6181890127, 890189018901),
('Mahesh', 'Sharma', '1979-06-28', 'Male', 'No. 15, Jhotwara, Jaipur', 5092901238, 901290129012),
('Sunil', 'Yadav', '1984-03-10', 'Male', 'H.No. 7, Chinhat, Lucknow', 4983012349, 112211221122),
('Aakash', 'Khanna', '1981-07-22', 'Male', 'Sector 22D, Chandigarh', 3874123450, 223322332233),
('Deepak', 'Nair', '1986-10-11', 'Male', 'Flat 8, Fort Kochi, Kochi', 2765234561, 334433443344),
('Vinod', 'Malhotra', '1977-04-05', 'Male', 'No. 42, Karond, Bhopal', 1656345672, 445544554455),
('Pappu', 'Verma', '1989-01-30', 'Male', 'H.No. 23, Phulwari Sharif, Patna', 9547456783, 556655665566),
('Rajesh', 'Joshi', '1988-08-17', 'Male', 'Flat 16, Varachha, Surat', 8438567894, 667766776677),
('Suman', 'Reddy', '1983-05-14', 'Female', 'Flat 9, Govandi, Mumbai', 7329678905, 778877887788),
('Alok', 'Mishra', '1976-12-08', 'Male', 'H.No. 34, Narela, Delhi', 6218789016, 889988998899),
('Kiran', 'Singh', '1987-09-21', 'Female', 'No. 51, Yeshwantpur, Bengaluru', 5101890127, 990090000900),
('Prakash', 'Rao', '1982-02-25', 'Male', 'Flat 22, Secunderabad, Hyderabad', 4092901238, 101010101010),
('Anil', 'Menon', '1985-11-13', 'Male', 'No. 67, Tambaram, Chennai', 3083012349, 202020202020),
('Rahul', 'Bose', '1979-08-14', 'Male', '15D, Salt Lake, Kolkata', 2874123450, 303030303030),
('Shiva', 'Pawar', '1984-06-27', 'Male', 'Flat 4, Wakad, Pune', 1765234561, 404040404040),
('Mohan', 'Patel', '1981-11-05', 'Male', 'H.No. 19, Satellite, Ahmedabad', 0656345672, 505050505050),
('Dinesh', 'Sharma', '1986-03-18', 'Male', 'No. 25, Vaishali Nagar, Jaipur', 9547456784, 606060606060),
('Vikas', 'Trivedi', '1991-09-22', 'Male', 'H.No. 10, Alambagh, Lucknow', 8438567895, 707070707070),
('Ramesh', 'Pillai', '1982-07-18', 'Male', 'Room No. 8, Chawl No.3, Dharavi, Mumbai', 9876123451, 808080808080),
('Sanjay', 'Verma', '1987-04-22', 'Male', 'H.No. 12, Laxmi Nagar, Delhi', 8765234562, 909090909090),
('Anita', 'Desai', '1984-11-15', 'Female', 'No. 34, MG Road, Bengaluru', 7656345673, 121212121212),
('Vishal', 'Joshi', '1990-02-28', 'Male', 'Flat 12, Begumpet, Hyderabad', 6547456784, 232323232323),
('Pooja', 'Reddy', '1988-09-05', 'Female', 'No. 45, Anna Nagar, Chennai', 9438567895, 343434343434),
('Rakesh', 'Malhotra', '1983-06-12', 'Male', '22B, Park Street, Kolkata', 8320678906, 454545454545),
('Sunita', 'Sharma', '1986-03-25', 'Female', 'Flat 7, Kothrud, Pune', 7218789017, 565656565656);

-- Inserting data in Evidence table
INSERT INTO Evidence (EvidenceName, Description, LocationFound, IncidentID) VALUES
('Knife', '6-inch blade with wooden handle', 'Near jewelry store counter', 1),
('CCTV Footage', 'Digital recording of burglary', 'Building security room', 2),
('Laptop', 'Used for online fraud', 'Suspect''s residence', 3),
('Broken Bottle', 'Used in assault with blood traces', 'Bar floor', 4),
('Wallet', 'Victim''s stolen wallet found discarded', 'Near market dustbin', 5),
('Bloody Knife', 'Murder weapon with fingerprints', 'Kitchen sink', 6),
('Helmet', 'Left at vehicle theft scene', 'Parking lot corner', 7),
('Documents', 'Fake property papers', 'Suspect''s office', 8),
('Child''s Toy', 'Found near abduction site', 'Park bench', 9),
('Medical Report', 'Victim injury documentation', 'Hospital records', 10),
('Drug Packets', 'Containing narcotics', 'Suspect''s car trunk', 11),
('Clothing', 'Torn dress of victim', 'Alleyway', 12),
('Gas Can', 'Used to start fire', 'Behind burned building', 13),
('Audio Recording', 'Noise complaint evidence', 'Neighbor''s phone', 14),
('Paint Scraps', 'From hit-and-run vehicle', 'Victim''s clothing', 15),
('Spray Cans', 'Used for graffiti', 'Abandoned near scene', 16),
('Credit Card', 'Cloned card found', 'ATM vestibule', 17),
('Emails', 'Harassment evidence', 'Victim''s work computer', 18),
('Cash', 'Bribe money marked', 'Official''s drawer', 19),
('Text Messages', 'Threatening messages', 'Victim''s phone', 20),
('Gold Chain', 'Snatched from victim', 'Pawn shop recovery', 26),
('Laptop', 'Used for scam operations', 'Cyber cafe raid', 27),
('Belt', 'Used in domestic assault', 'Victim''s residence', 28),
('Mobile Phone', 'Stolen device with fingerprints', 'Flea market recovery', 29),
('Lottery Tickets', 'Fake winning tickets', 'Suspect''s residence', 30);

-- Inserting data in Reports table
```

```sql

-- Inserting data in Reports table
INSERT INTO Reports (IncidentID, ReportingOfficerID, ReportDate, ReportDetails, STATUS) VALUES
(1, 1, '2023-01-16', 'Robbery at jewelry store with estimated loss of ₹25 lakhs. One suspect apprehended.', 'Finalized'),
(2, 2, '2023-02-04', 'Daytime burglary with stolen valuables worth ₹5 lakhs. Investigation ongoing.', 'Draft'),
(3, 3, '2023-02-11', 'Online fraud case involving ₹12 lakhs transferred to fake accounts.', 'Finalized'),
(4, 4, '2023-02-19', 'Bar fight resulting in serious injuries to two patrons. Case closed after settlement.', 'Finalized'),
(5, 5, '2023-03-06', 'Pickpocketing incident in crowded market. No suspects identified yet.', 'Draft'),
(6, 6, '2023-03-13', 'Domestic dispute turned fatal. Husband in custody.', 'Finalized'),
(7, 7, '2023-03-21', 'Motorcycle theft from residential parking. CCTV footage being analyzed.', 'Draft'),
(8, 8, '2023-04-03', 'Real estate scam involving fake property documents. Three suspects arrested.', 'Finalized'),
(9, 9, '2023-04-16', 'Child abduction case. AMBER alert issued. Child recovered safely.', 'Finalized'),
(10, 10, '2023-04-23', 'Domestic violence complaint. Husband charged under IPC 498A.', 'Finalized'),
(11, 11, '2023-05-06', 'Drug peddling bust with seizure of 2kg narcotics. Two arrested.', 'Finalized'),
(12, 12, '2023-05-13', 'Molestation case reported by college student. Suspect identified.', 'Draft'),
(13, 13, '2023-05-21', 'Deliberate fire at commercial property causing ₹50 lakhs damage.', 'Draft'),
(14, 14, '2023-06-04', 'Noise complaint against late-night party. Warning issued.', 'Finalized'),
(15, 15, '2023-06-11', 'Hit-and-run case with pedestrian critically injured. Vehicle identified.', 'Draft'),
(16, 16, '2023-06-19', 'Vandalism of public property with graffiti. Local gang suspected.', 'Draft'),
(17, 17, '2023-07-03', 'Credit card fraud involving ₹3.2 lakhs unauthorized transactions.', 'Draft'),
(18, 18, '2023-07-16', 'Workplace harassment complaint by female employee. HR involved.', 'Finalized'),
(19, 19, '2023-07-23', 'Bribery case against municipal officer caught red-handed.', 'Finalized'),
(20, 20, '2023-08-06', 'Child abuse complaint against school teacher. Investigation ongoing.', 'Draft'),
(26, 1, '2025-06-06', 'Chain snatching incident near Marine Lines station. Victim sustained minor injuries.', 'Finalized'),
(27, 2, '2025-06-13', 'Online job scam defrauding 50 people of approximately ₹25 lakhs collectively.', 'Draft'),
(28, 3, '2025-06-19', 'Domestic violence case with victim requiring hospitalization. Husband absconding.', 'Finalized'),
(29, 4, '2025-06-26', 'Mobile phone theft reported in Banjara Hills mall. CCTV footage available.', 'Draft'),
(30, 5, '2025-06-29', 'Fake lottery scam targeting senior citizens. Three suspects identified.', 'Draft');

-- Inserting data in Criminals table
INSERT INTO Criminals (IncidentID, SuspectID, AadhaarNumber, PunishmentDetails) VALUES
(1, 1000, 123412341234, '5 years imprisonment under IPC 392'),
(4, 1003, 456745674567, '2 years imprisonment under IPC 326'),
(6, 1005, 678967896789, 'Life imprisonment under IPC 302'),
(8, 1007, 890189018901, '7 years imprisonment under IPC 420'),
(9, 1008, 901290129012, '10 years imprisonment under IPC 363'),
(10, 1009, 112211221122, '3 years imprisonment under IPC 498A'),
(11, 1010, 223322332233, '10 years rigorous imprisonment under NDPS Act'),
(14, 1013, 556655665566, 'Fine of ₹10,000 under Noise Pollution Rules'),
(18, 1017, 889988998890, '2 years imprisonment under IPC 354'),
(19, 1018, 990090089900, '5 years imprisonment under Prevention of Corruption Act'),
(2, 1001, 234523452345, '3 years imprisonment under IPC 454'),
(3, 1002, 345634563456, '4 years imprisonment under IT Act 2008'),
(5, 1004, 567856785678, '1 year imprisonment under IPC 379'),
(12, 1011, 334433443344, '7 years imprisonment under IPC 354'),
(15, 1014, 667766776677, '5 years imprisonment under IPC 279/338'),
(16, 1015, 778877887788, '6 months imprisonment under IPC 427'),
(17, 1016, 899988998899, '3 years imprisonment under IT Act 66C'),
(20, 1019, 990090089900, '7 years imprisonment under POCSO Act'),
(21, 1020, 101010101010, '1 year imprisonment under IPC 294'),
(22, 1021, 202020202020, '2 years imprisonment under Public Gambling Act'),
(26, 1022, 303030303030, '3 years imprisonment under IPC 392 (pending appeal)'),
(27, 1023, 404040404040, '5 years imprisonment under IT Act 66D (under trial)');

-- Inserting data in Users table
INSERT INTO Users (Username, Password, Role, OfficerID) VALUES
('rajesh.kumar', 'mumbai@123', 'Officer', 1),
('priya.sharma', 'delhi@456', 'Officer', 2),
('vikram.singh', 'bengaluru@789', 'Admin', 3),
('anjali.patel', 'hyderabad@321', 'Officer', 4),
('arun.iyer', 'chennai@654', 'Officer', 5),
('meera.banerjee', 'kolkata@987', 'Officer', 6),
('sanjay.deshmukh', 'pune@123', 'Officer', 7),
('neha.gupta', 'ahmedabad@456', 'Officer', 8),
('rahul.sharma', 'jaipur@789', 'Officer', 9),
('sunita.yadav', 'lucknow@321', 'Officer', 10),
('amit.khanna', 'chandigarh@654', 'Officer', 11),
('deepa.nair', 'kochi@987', 'Officer', 12),
('vijay.malhotra', 'bhopal@123', 'Admin', 13),
('pooja.verma', 'patna@456', 'Officer', 14),
('ramesh.joshi', 'surat@789', 'Officer', 15),
('smita.reddy', 'mumbai2@321', 'Officer', 16),
('alok.mishra', 'delhi2@654', 'Officer', 17),
('kavita.singh', 'bengaluru2@987', 'Officer', 18),
('prakash.rao', 'hyderabad2@123', 'Officer', 19),
('anita.menon', 'chennai2@456', 'Officer', 20),
('rajiv.bose', 'kolkata2@789', 'Admin', 21),
('shweta.pawar', 'pune2@321', 'Officer', 22),
('manoj.patel', 'ahmedabad2@654', 'Officer', 23),
('divya.sharma', 'jaipur2@987', 'Officer', 24),
('vivek.trivedi', 'lucknow2@123', 'Officer', 25),
('anjali.mehta', 'chandigarh2@456', 'Officer', 26),
('sunil.nair', 'kochi2@789', 'Officer', 27),
('preeti.verma', 'bhopal2@321', 'Officer', 28),
('rakesh.yadav', 'patna2@654', 'Officer', 29),
('sonia.kapoor', 'surat2@987', 'Officer', 30),
('nitin.shah', 'nagpur@123', 'Admin', 31),
('ananya.desai', 'indore@456', 'Officer', 32),
('vikas.malhotra', 'thane@789', 'Officer', 33),
('rashmi.saxena', 'ghaziabad@321', 'Officer', 34),
('sanjeev.iyer', 'coimbatore@654', 'Officer', 35);

```

database > CARS_InsertionOfData.sql > ...

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

SELECT * FROM Victims;
DELETE FROM Victims WHERE VictimID = 31;
```

Entity Folder

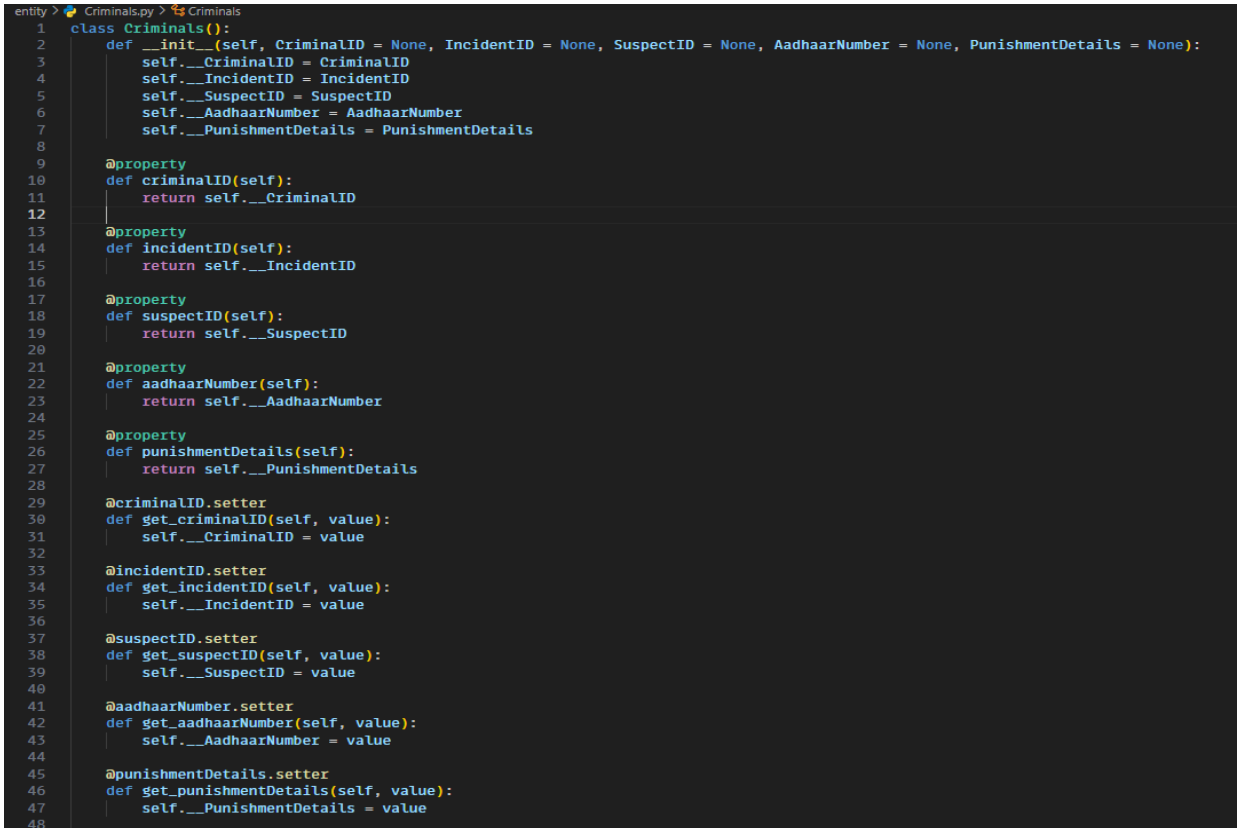
auth.py

CrimeAnalysisService.py | CrimeAnalysisServiceImpl.py | CARS_CreatingTables.sql | CARS_InsertionOfData.sql | auth.py

entity > auth.py > Authentication > login

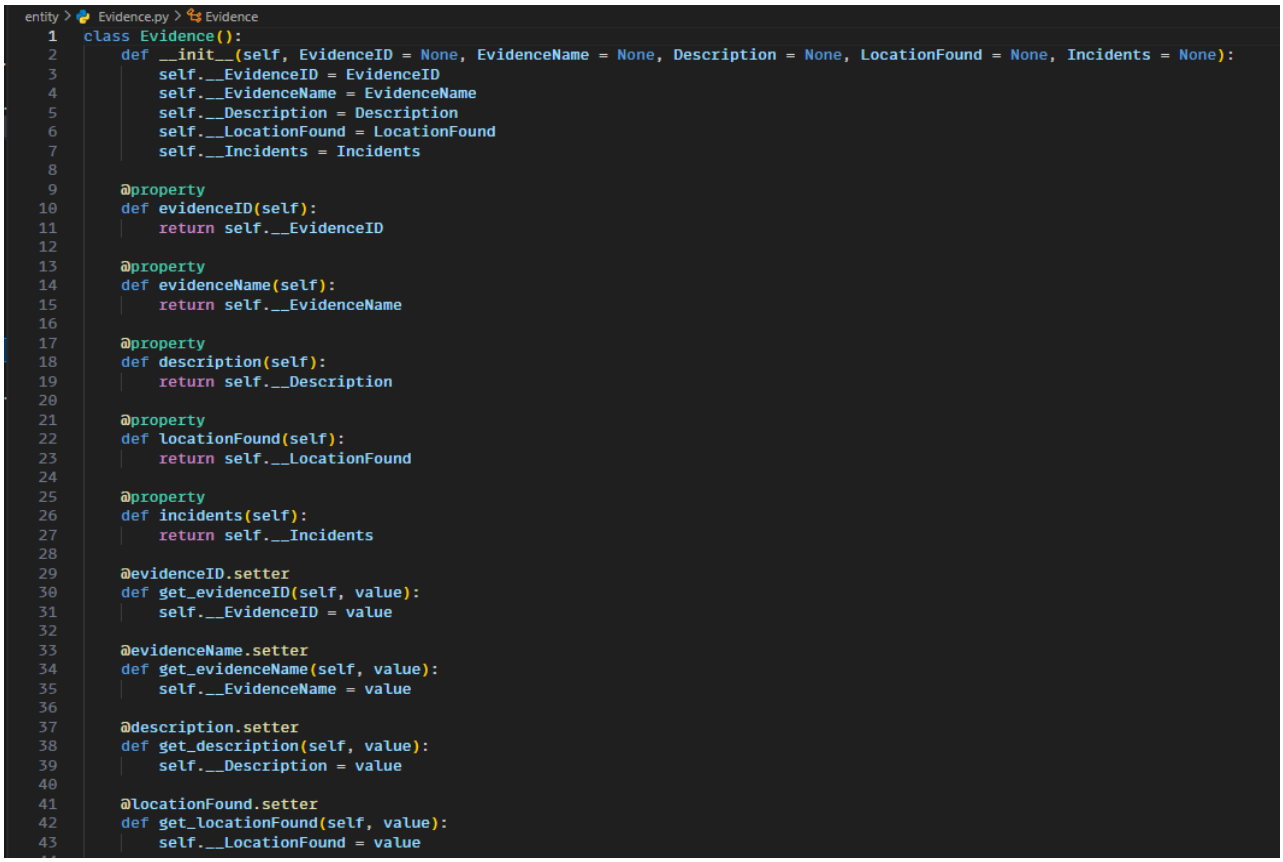
```python
from getpass import getpass
from util.DB_Connections import DBConnections
from exceptions.user_defined_exceptions import AuthenticationError

class Authentication:

    @staticmethod
    def login():
        print("\n=== Login ===")
        username = input("Username: ")
        password = getpass("Password: ")

        try:
            connection = DBConnections.get_connection('db.properties')
            cursor = connection.cursor(dictionary=True)

            query = "SELECT * FROM Users WHERE Username = %s AND Password = %s"
            cursor.execute(query, (username, password))
            user = cursor.fetchone()

            cursor.close()
            connection.close()

            if user:
                return user
            raise AuthenticationError("Invalid credentials! Please try again.")

        except Exception as e:
            raise AuthenticationError(f"Login failed: {str(e)}")

    @staticmethod
    def public_access():
        return {'UserID': 0, 'Role': 'Public', 'Username': 'public_user'}
```

Criminals.py



```python
class Criminals():
    def __init__(self, CriminalID = None, IncidentID = None, SuspectID = None, AadhaarNumber = None, PunishmentDetails = None):
        self.__CriminalID = CriminalID
        self.__IncidentID = IncidentID
        self.__SuspectID = SuspectID
        self.__AadhaarNumber = AadhaarNumber
        self.__PunishmentDetails = PunishmentDetails

    @property
    def criminalID(self):
        return self.__CriminalID

    @property
    def incidentID(self):
        return self.__IncidentID

    @property
    def suspectID(self):
        return self.__SuspectID

    @property
    def aadhaarNumber(self):
        return self.__AadhaarNumber

    @property
    def punishmentDetails(self):
        return self.__PunishmentDetails

    @criminalID.setter
    def get_criminalID(self, value):
        self.__CriminalID = value

    @incidentID.setter
    def get_incidentID(self, value):
        self.__IncidentID = value

    @suspectID.setter
    def get_suspectID(self, value):
        self.__SuspectID = value

    @aadhaarNumber.setter
    def get_aadhaarNumber(self, value):
        self.__AadhaarNumber = value

    @punishmentDetails.setter
    def get_punishmentDetails(self, value):
        self.__PunishmentDetails = value
```

Evidence.py



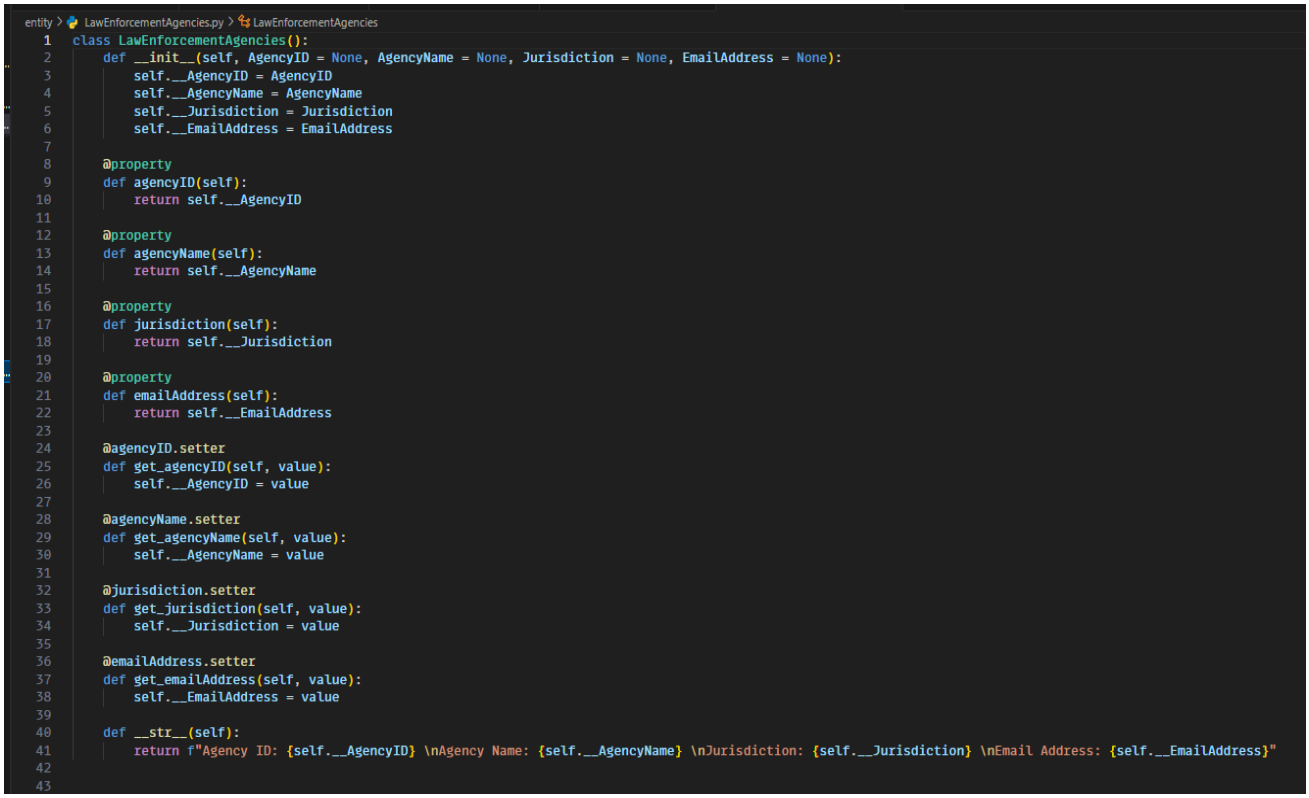
```python
class Evidence():
    def __init__(self, EvidenceID = None, EvidenceName = None, Description = None, LocationFound = None, Incidents = None):
        self.__EvidenceID = EvidenceID
        self.__EvidenceName = EvidenceName
        self.__Description = Description
        self.__LocationFound = LocationFound
        self.__Incidents = Incidents

    @property
    def evidenceID(self):
        return self.__EvidenceID

    @property
    def evidenceName(self):
        return self.__EvidenceName

    @property
    def description(self):
        return self.__Description

    @property
    def locationFound(self):
        return self.__LocationFound

    @property
    def incidents(self):
        return self.__Incidents

    @evidenceID.setter
    def get_evidenceID(self, value):
        self.__EvidenceID = value

    @evidenceName.setter
    def get_evidenceName(self, value):
        self.__EvidenceName = value

    @description.setter
    def get_description(self, value):
        self.__Description = value

    @locationFound.setter
    def get_locationFound(self, value):
        self.__LocationFound = value
```

Incidents.py

entity > Incidents.py > Incidents

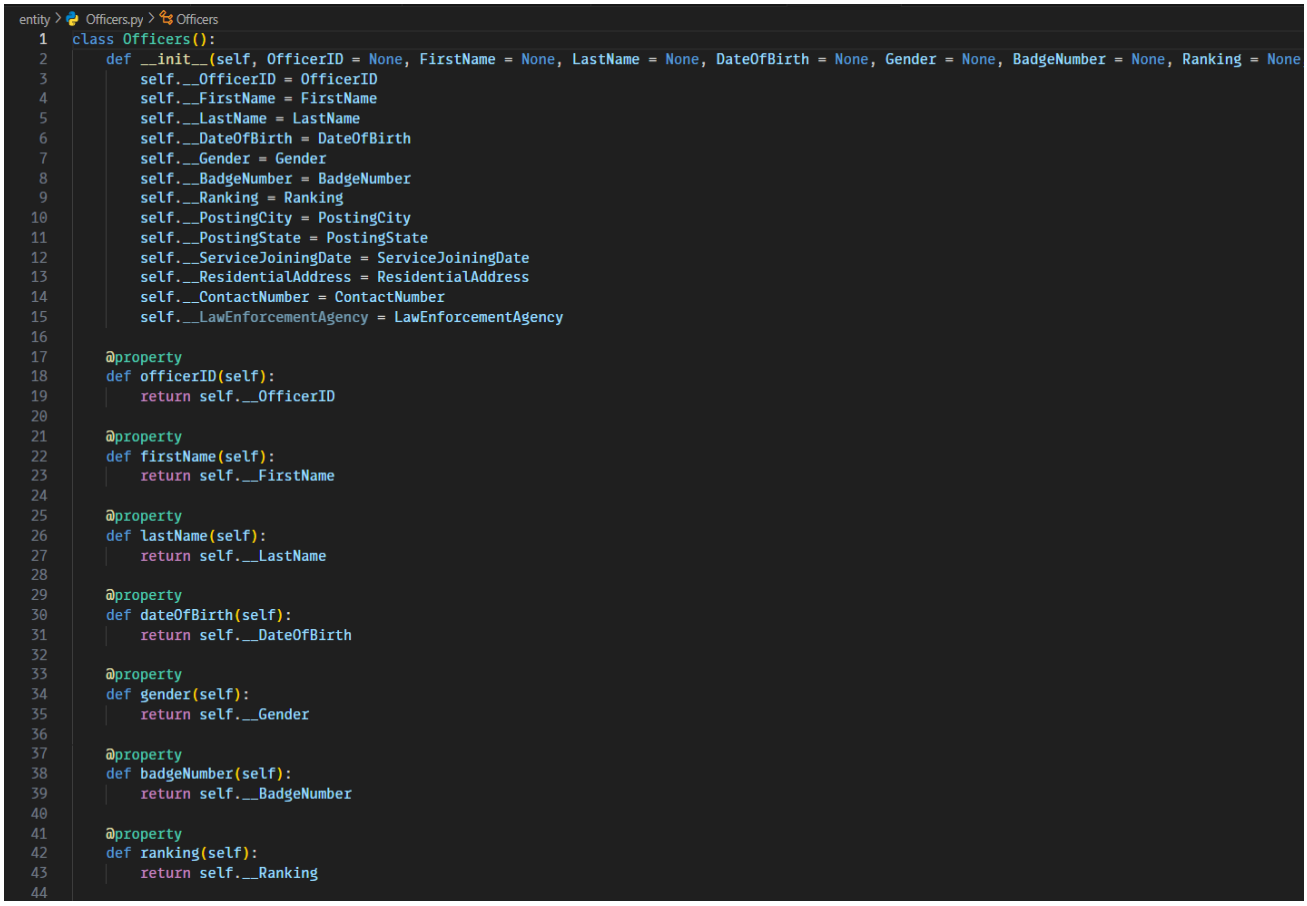
```python
class Incidents():
    def __init__(self, IncidentID = None, IncidentType = None, IncidentDate = None, Area = None, City = None, Description = None, Status = None, OfficerID = None):
        self.__IncidentID = IncidentID
        self.__IncidentType = IncidentType
        self.__IncidentDate = IncidentDate
        self.__Area = Area
        self.__City = City
        self.__Description = Description
        self.__Status = Status
        self.__OfficerID = OfficerID

    @property
    def incidentID(self):
        return self.__IncidentID

    @property
    def incidentType(self):
        return self.__IncidentType

    @property
    def incidentDate(self):
        return self.__IncidentDate

    @property
    def area(self):
        return self.__Area

    @property
    def city(self):
        return self.__City

    @property
    def description(self):
        return self.__Description

    @property
    def status(self):
        return self.__Status

    @property
    def officerID(self):
        return self.__OfficerID

    @incidentID.setter
    def get_incidentID(self, value):
        self.__IncidentID = value

    @incidentType.setter
    def get_incidentType(self, value):
        self.__IncidentType = value

    @incidentDate.setter
    def get_incidentDate(self, value):
        self.__IncidentDate = value

    @area.setter
    def get_area(self, value):
        self.__Area = value

    @city.setter
    def get_city(self, value):
        self.__City = value

    @description.setter
    def get_description(self, value):
        self.__Description = value

    @status.setter
    def get_status(self, value):
        self.__Status = value

    @officerID.setter
    def get_officerID(self, value):
        self.__OfficerID = value

```

LawEnforcementAgencies.py



```python
class LawEnforcementAgencies():
    def __init__(self, AgencyID = None, AgencyName = None, Jurisdiction = None, EmailAddress = None):
        self.__AgencyID = AgencyID
        self.__AgencyName = AgencyName
        self.__Jurisdiction = Jurisdiction
        self.__EmailAddress = EmailAddress

    @property
    def agencyID(self):
        return self.__AgencyID

    @property
    def agencyName(self):
        return self.__AgencyName

    @property
    def jurisdiction(self):
        return self.__Jurisdiction

    @property
    def emailAddress(self):
        return self.__EmailAddress

    @agencyID.setter
    def get_agencyID(self, value):
        self.__AgencyID = value

    @agencyName.setter
    def get_agencyName(self, value):
        self.__AgencyName = value

    @jurisdiction.setter
    def get_jurisdiction(self, value):
        self.__Jurisdiction = value

    @emailAddress.setter
    def get_emailAddress(self, value):
        self.__EmailAddress = value

    def __str__(self):
        return f"Agency ID: {self.__AgencyID} \nAgency Name: {self.__AgencyName} \nJurisdiction: {self.__Jurisdiction} \nEmail Address: {self.__EmailAddress}"
```

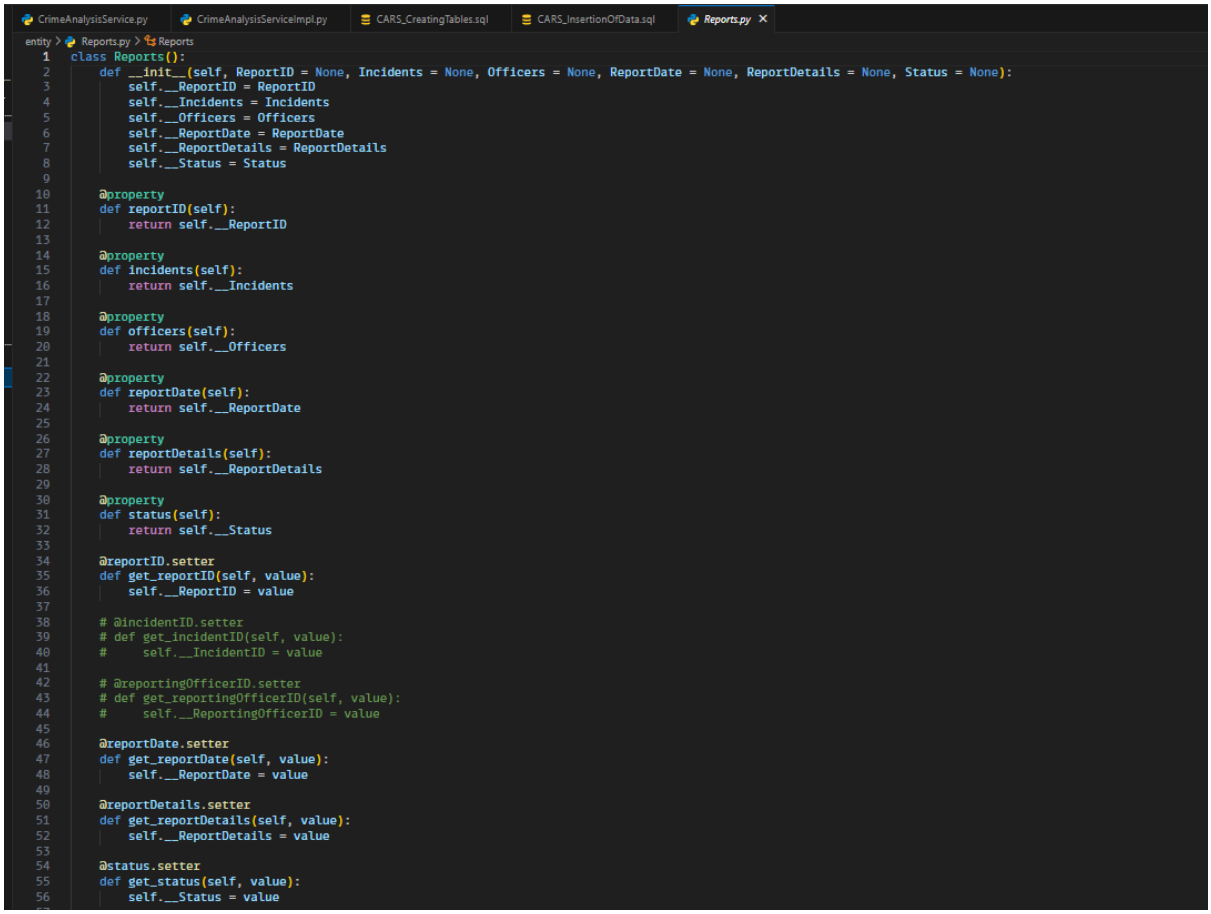
Officers.py



```python
class Officers():
    def __init__(self, OfficerID = None, FirstName = None, LastName = None, DateOfBirth = None, Gender = None, BadgeNumber = None, Ranking = None
        self.__OfficerID = OfficerID
        self.__FirstName = FirstName
        self.__LastName = LastName
        self.__DateOfBirth = DateOfBirth
        self.__Gender = Gender
        self.__BadgeNumber = BadgeNumber
        self.__Ranking = Ranking
        self.__PostingCity = PostingCity
        self.__PostingState = PostingState
        self.__ServiceJoiningDate = ServiceJoiningDate
        self.__ResidentialAddress = ResidentialAddress
        self.__ContactNumber = ContactNumber
        self.__LawEnforcementAgency = LawEnforcementAgency

    @property
    def officerID(self):
        return self.__OfficerID

    @property
    def firstName(self):
        return self.__FirstName

    @property
    def lastName(self):
        return self.__LastName

    @property
    def dateOfBirth(self):
        return self.__DateOfBirth

    @property
    def gender(self):
        return self.__Gender

    @property
    def badgeNumber(self):
        return self.__BadgeNumber

    @property
    def ranking(self):
        return self.__Ranking
```

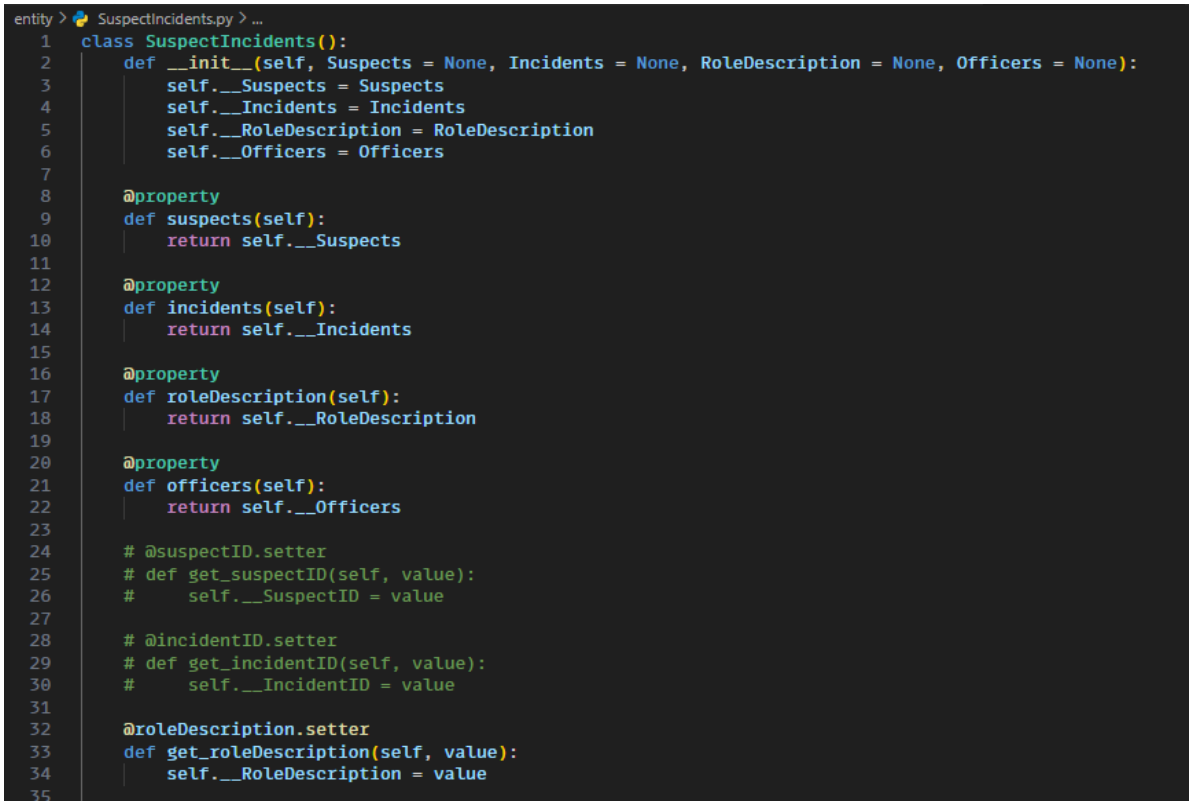

entity > Officers.py > Officers > get_officerID
class Officers():
@property
def ranking(self):
return self.__Ranking
@property
def postingCity(self):
return self.__PostingCity
@property
def postingState(self):
return self.__PostingState
@property
def serviceJoiningDate(self):
return self.__ServiceJoiningDate
@property
def residentialAddress(self):
return self.__ResidentialAddress
@property
def contactNumber(self):
return self.__ContactNumber
@property
def lawEnforcementAgency(self):
return self.__AgencyID
@officerID.setter
def get_officerID(self, value):
self.__OfficerID = value
@firstName.setter
def get_firstName(self, value):
self.__FirstName = value
@lastName.setter
def get_lastName(self, value):
self.__LastName = value
@dateOfBirth.setter
def get_dateOfBirth(self, value):
self.__DateOfBirth = value
@gender.setter
def get_gender(self, value):
self.__Gender = value
@badgeNumber.setter
def get_badgeNumber(self, value):
self.__BadgeNumber = value
@ranking.setter
def get_ranking(self, value):
self.__Ranking = value
@postingCity.setter
def get_postingCity(self, value):
self.__PostingCity = value
@serviceJoiningDate.setter
def get_serviceJoiningDate(self, value):
self.__ServiceJoiningDate = value
@residentialAddress.setter
def get_residentialAddress(self, value):
self.__ResidentialAddress = value
@contactNumber.setter
def get_contactNumber(self, value):
self.__ContactNumber = value

Reports.py



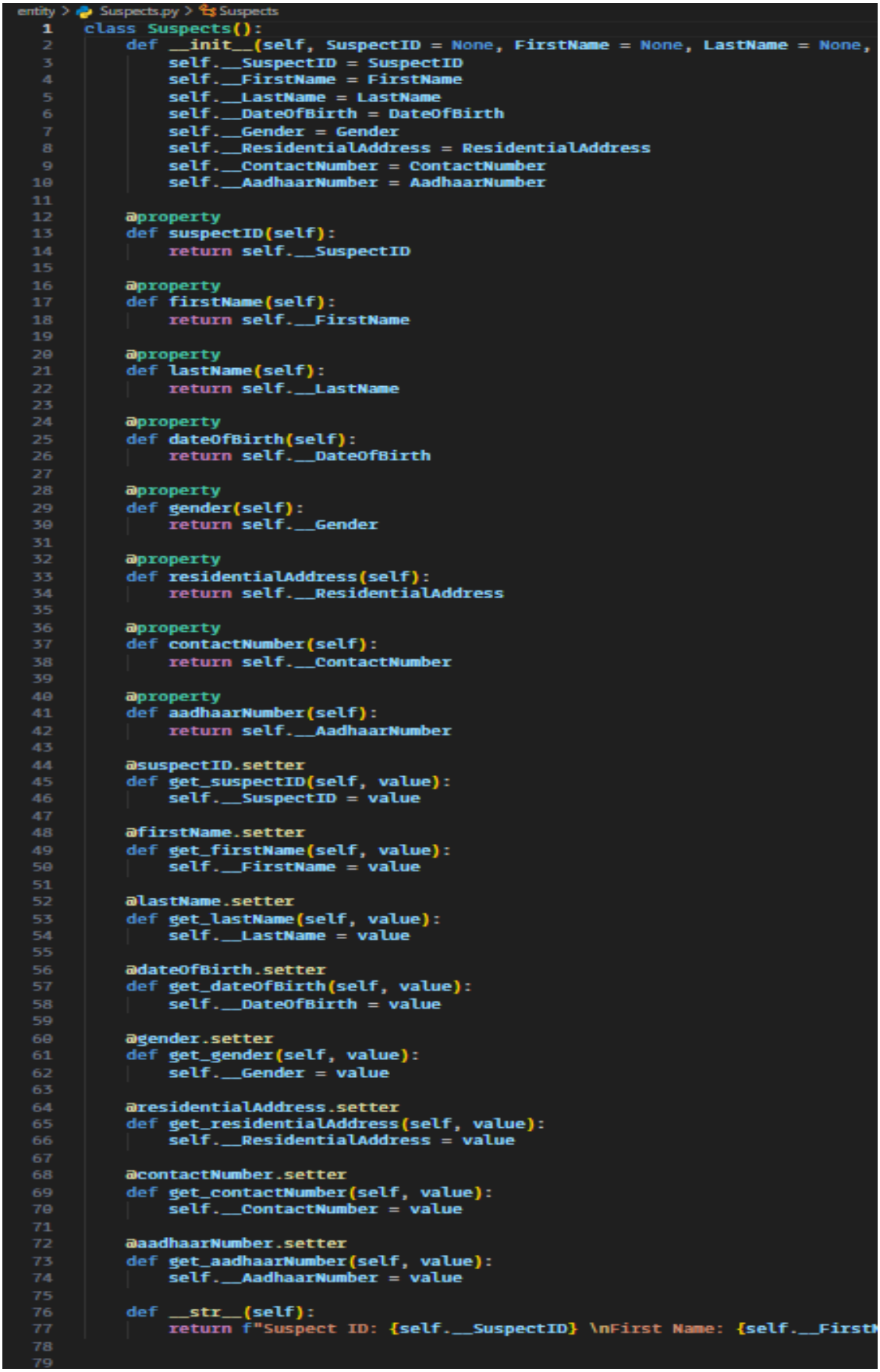
```python
class Reports():
    def __init__(self, ReportID = None, Incidents = None, Officers = None, ReportDate = None, ReportDetails = None, Status = None):
        self.__ReportID = ReportID
        self.__Incidents = Incidents
        self.__Officers = Officers
        self.__ReportDate = ReportDate
        self.__ReportDetails = ReportDetails
        self.__Status = Status

    @property
    def reportID(self):
        return self.__ReportID

    @property
    def incidents(self):
        return self.__Incidents

    @property
    def officers(self):
        return self.__Officers

    @property
    def reportDate(self):
        return self.__ReportDate

    @property
    def reportDetails(self):
        return self.__ReportDetails

    @property
    def status(self):
        return self.__Status

    @reportID.setter
    def get_reportID(self, value):
        self.__ReportID = value

    # @incidentID.setter
    # def get_incidentID(self, value):
    #     self.__IncidentID = value

    # @reportingOfficerID.setter
    # def get_reportingOfficerID(self, value):
    #     self.__ReportingOfficerID = value

    @reportDate.setter
    def get_reportDate(self, value):
        self.__ReportDate = value

    @reportDetails.setter
    def get_reportDetails(self, value):
        self.__ReportDetails = value

    @status.setter
    def get_status(self, value):
        self.__Status = value
```

SuspectIncidents.py



```python
class SuspectIncidents():
    def __init__(self, Suspects = None, Incidents = None, RoleDescription = None, Officers = None):
        self.__Suspects = Suspects
        self.__Incidents = Incidents
        self.__RoleDescription = RoleDescription
        self.__Officers = Officers

    @property
    def suspects(self):
        return self.__Suspects

    @property
    def incidents(self):
        return self.__Incidents

    @property
    def roleDescription(self):
        return self.__RoleDescription

    @property
    def officers(self):
        return self.__Officers

    # @suspectID.setter
    # def get_suspectID(self, value):
    #     self.__SuspectID = value

    # @incidentID.setter
    # def get_incidentID(self, value):
    #     self.__IncidentID = value

    @roleDescription.setter
    def get_roleDescription(self, value):
        self.__RoleDescription = value
```

Suspects.py

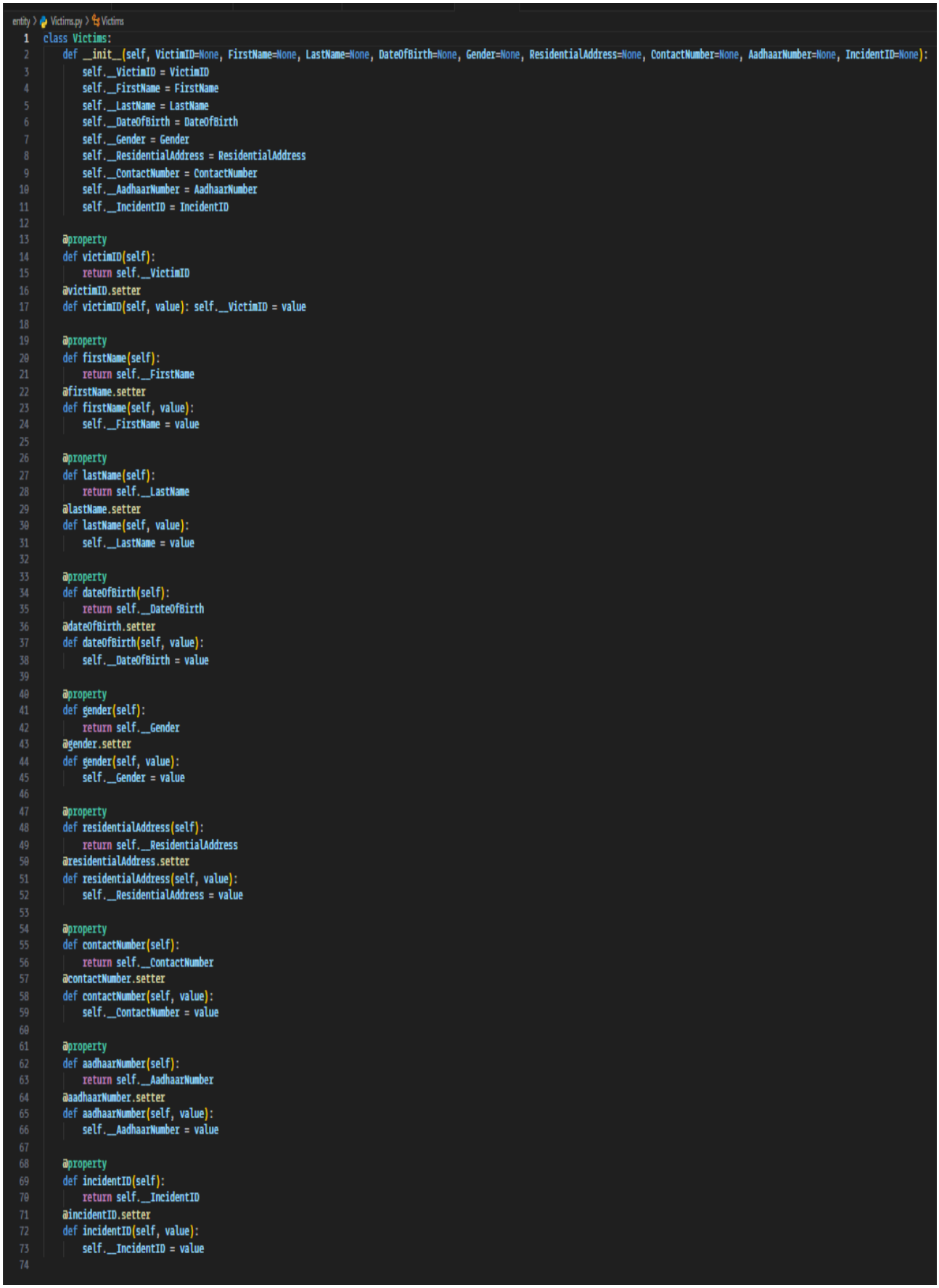

entity > Suspects.py > Suspects
class Suspects():
def __init__(self, SuspectID = None, FirstName = None, LastName = None,
self.__SuspectID = SuspectID
self.__FirstName = FirstName
self.__LastName = LastName
self.__DateOfBirth = DateOfBirth
self.__Gender = Gender
self.__ResidentialAddress = ResidentialAddress
self.__ContactNumber = ContactNumber
self.__AadhaarNumber = AadhaarNumber
@property
def suspectID(self):
return self.__SuspectID
@property
def firstName(self):
return self.__FirstName
@property
def lastName(self):
return self.__LastName
@property
def dateOfBirth(self):
return self.__DateOfBirth
@property
def gender(self):
return self.__Gender
@property
def residentialAddress(self):
return self.__ResidentialAddress
@property
def contactNumber(self):
return self.__ContactNumber
@property
def aadhaarNumber(self):
return self.__AadhaarNumber
@suspectID.setter
def get_suspectID(self, value):
self.__SuspectID = value
@firstName.setter
def get_firstName(self, value):
self.__FirstName = value
@lastName.setter
def get_lastName(self, value):
self.__LastName = value
@dateOfBirth.setter
def get_dateOfBirth(self, value):
self.__DateOfBirth = value
@gender.setter
def get_gender(self, value):
self.__Gender = value
@residentialAddress.setter
def get_residentialAddress(self, value):
self.__ResidentialAddress = value
@contactNumber.setter
def get_contactNumber(self, value):
self.__ContactNumber = value
@aadhaarNumber.setter
def get_aadhaarNumber(self, value):
self.__AadhaarNumber = value
def __str__(self):
return f"Suspect ID: {self.__SuspectID} \nFirst Name: {self.__FirstN

Users.py

entity > Users.py > User

```python
class User:
    def __init__(self, user_id=None, username=None, password=None, role='Officer', Officers=None):
        self.__user_id = user_id
        self.__username = username
        self.__password = password
        self.__role = role
        self.__Officers = Officers

    @property
    def user_id(self):
        return self.__user_id

    @property
    def username(self):
        return self.__username

    @property
    def password(self):
        return self.__password

    @property
    def role(self):
        return self.__role

    @property
    def officers(self):
        return self.__Officers

    @user_id.setter
    def user_id(self, value):
        self.__user_id = value

    @username.setter
    def username(self, value):
        self.__username = value

    @password.setter
    def password(self, value):
        self.__password = value

    @role.setter
    def role(self, value):
        if value not in ('Officer', 'Admin'):
            raise ValueError("Role must be 'Officer' or 'Admin'")
        self.__role = value
```

Victims.py



```
entity > Victims.py > Victims
class Victims:
    def __init__(self, VictimID=None, FirstName=None, LastName=None, DateOfBirth=None, Gender=None, ResidentialAddress=None, ContactNumber=None, AadhaarNumber=None, IncidentID=None):
        self.__VictimID = VictimID
        self.__FirstName = FirstName
        self.__LastName = LastName
        self.__DateOfBirth = DateOfBirth
        self.__Gender = Gender
        self.__ResidentialAddress = ResidentialAddress
        self.__ContactNumber = ContactNumber
        self.__AadhaarNumber = AadhaarNumber
        self.__IncidentID = IncidentID

    @property
    def victimID(self):
        return self.__VictimID
    @victimID.setter
    def victimID(self, value): self.__VictimID = value

    @property
    def firstName(self):
        return self.__FirstName
    @firstName.setter
    def firstName(self, value):
        self.__FirstName = value

    @property
    def lastName(self):
        return self.__LastName
    @lastName.setter
    def lastName(self, value):
        self.__LastName = value

    @property
    def dateOfBirth(self):
        return self.__DateOfBirth
    @dateOfBirth.setter
    def dateOfBirth(self, value):
        self.__DateOfBirth = value

    @property
    def gender(self):
        return self.__Gender
    @gender.setter
    def gender(self, value):
        self.__Gender = value

    @property
    def residentialAddress(self):
        return self.__ResidentialAddress
    @residentialAddress.setter
    def residentialAddress(self, value):
        self.__ResidentialAddress = value

    @property
    def contactNumber(self):
        return self.__ContactNumber
    @contactNumber.setter
    def contactNumber(self, value):
        self.__ContactNumber = value

    @property
    def aadhaarNumber(self):
        return self.__AadhaarNumber
    @aadhaarNumber.setter
    def aadhaarNumber(self, value):
        self.__AadhaarNumber = value

    @property
    def incidentID(self):
        return self.__IncidentID
    @incidentID.setter
    def incidentID(self, value):
        self.__IncidentID = value
```

Exceptions Folder

#user_defined_exceptions.py

exceptions > user_defined_exceptions.py > DatabaseError

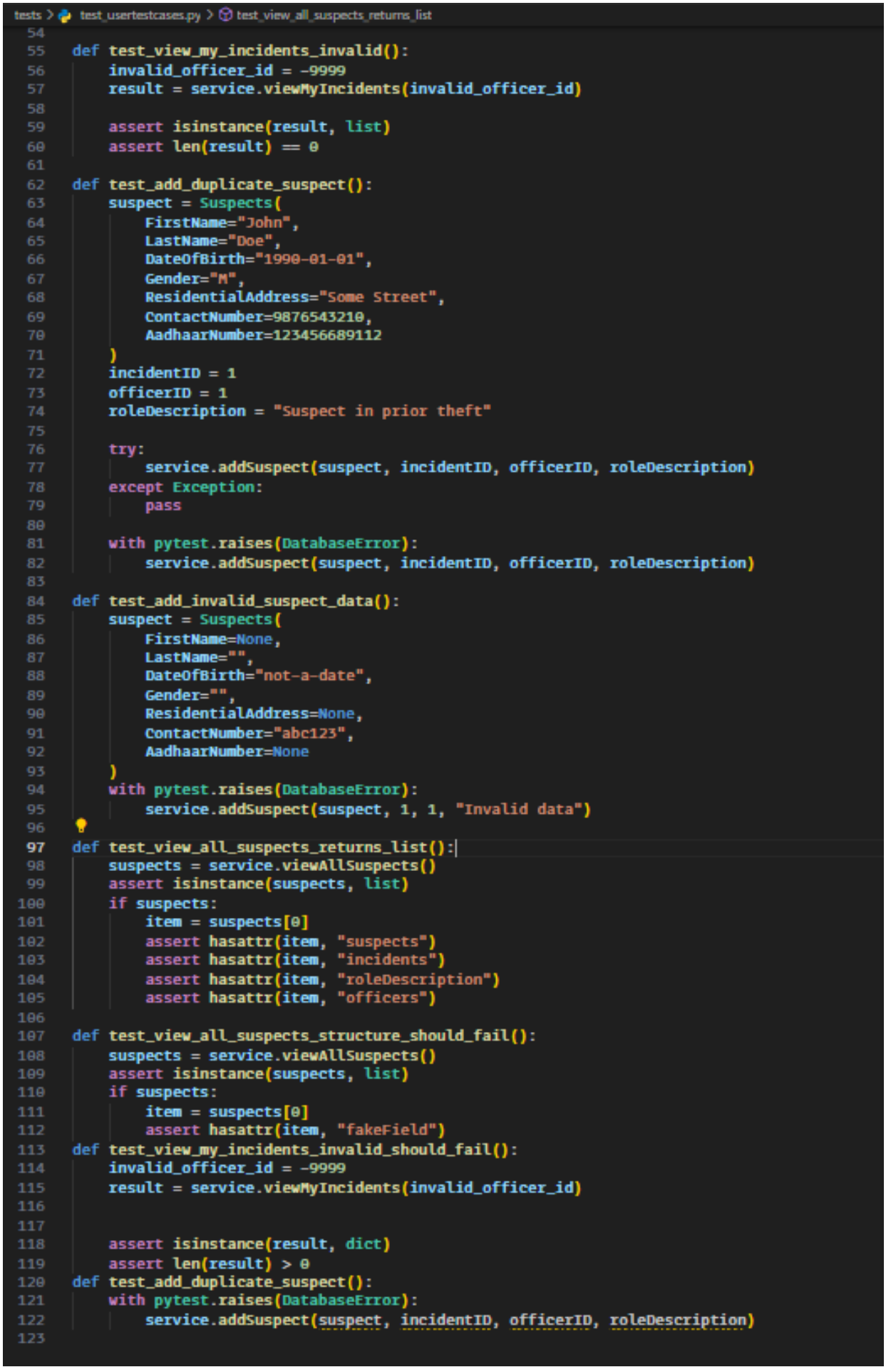
```python
class DatabaseError(Exception):
    def __init__(self, message = "Database Error"):
        self.message = message
        super().__init__(self.message)

class IncidentNotFoundException(Exception):
    def __init__(self, message="Incident not found"):
        super().__init__(message)

class VictimNotFoundException(Exception):
    def __init__(self, message="Victim not found."):
        super().__init__(message)

class SuspectNotFoundException(Exception):
    def __init__(self, message="Suspect not found."):
        super().__init__(message)

class CriminalNotFoundException(Exception):
    def __init__(self, message="Criminal not found."):
        super().__init__(message)

class OfficerNotFoundException(Exception):
    def __init__(self, message="Officer not found."):
        super().__init__(message)

class AgencyNotFoundException(Exception):
    def __init__(self, message="Law Enforcement Agency not found."):
        super().__init__(message)

class ReportNotFoundException(Exception):
    def __init__(self, message="Report not found."):
        super().__init__(message)

class EvidenceNotFoundException(Exception):
    def __init__(self, message="Evidence not found."):
        super().__init__(message)

class DuplicateEntryException(Exception):
    def __init__(self, message="Duplicate entry found."):
        super().__init__(message)

class AuthenticationError(Exception):
    def __init__(self, message = "Authentication Failed"):
        self.message = message
        super().__init__(self.message)

```

Tests Folder

#test_usertestcases.py

tests > test_usertestcases.py > test_get_officers_by_location

```python
import sys
import os
import pytest
from entity.Suspects import Suspects
from entity.auth import Authentication
from exceptions.user_defined_exceptions import AuthenticationError
from dao.CrimeAnalysisServiceImpl import CrimeAnalysisServiceImpl
from exceptions.user_defined_exceptions import DatabaseError

sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
service = CrimeAnalysisServiceImpl()
def test_public_access():
    user = Authentication.public_access()
    assert user['Role'] == 'Public'
    assert user['Username'] == 'public_user'

def test_login_invalid(monkeypatch):
    monkeypatch.setattr('builtins.input', lambda _: 'wrong_user')
    monkeypatch.setattr('entity.auth.getpass', lambda _: 'wrong_pass')

    with pytest.raises(AuthenticationError):
        Authentication.login()


def test_get_officers_by_location():
    officers = service.get_officers_by_location("coimbatore", "Coimbatore")
    assert isinstance(officers, list)


def test_get_incidents_in_date_range_public_no_results():
    results = service.get_incidents_in_date_range_public("1900-01-01", "1900-01-02")
    assert isinstance(results, list)
    assert len(results) == 0

def test_get_incidents_by_area_city_public_city_only():
    results = service.get_incidents_by_area_city_public(area=None, city="coimbatore")
    assert isinstance(results, list)

def test_get_incidents_by_area_city_public_invalid():
    results = service.get_incidents_by_area_city_public(area="fakearea", city="fakecity")
    assert isinstance(results, list)
    assert len(results) == 0

def test_filtered_incidents_combination():
    results = service.get_filtered_incidents(
        start_date="2020-01-01",
        end_date="2020-12-31",
        area="Coimbatore",
        city="coimbatore",
        incident_type="theft",
        officer_id=None
    )
    assert isinstance(results, list)

```


tests > test_usertestcases.py > test_view_all_suspects_returns_list
def test_view_my_incidents_invalid():
invalid_officer_id = -9999
result = service.viewMyIncidents(invalid_officer_id)
assert isinstance(result, list)
assert len(result) == 0
def test_add_duplicate_suspect():
suspect = Suspects(
FirstName="John",
LastName="Doe",
DateOfBirth="1990-01-01",
Gender="M",
ResidentialAddress="Some Street",
ContactNumber=9876543210,
AadhaarNumber=123456689112
)
incidentID = 1
officerID = 1
roleDescription = "Suspect in prior theft"
try:
service.addSuspect(suspect, incidentID, officerID, roleDescription)
except Exception:
pass
with pytest.raises(DatabaseError):
service.addSuspect(suspect, incidentID, officerID, roleDescription)
def test_add_invalid_suspect_data():
suspect = Suspects(
FirstName=None,
LastName="",
DateOfBirth="not-a-date",
Gender="",
ResidentialAddress=None,
ContactNumber="abc123",
AadhaarNumber=None
)
with pytest.raises(DatabaseError):
service.addSuspect(suspect, 1, 1, "Invalid data")
def test_view_all_suspects_returns_list():
suspects = service.viewAllSuspects()
assert isinstance(suspects, list)
if suspects:
item = suspects[0]
assert hasattr(item, "suspects")
assert hasattr(item, "incidents")
assert hasattr(item, "roleDescription")
assert hasattr(item, "officers")
def test_view_all_suspects_structure_should_fail():
suspects = service.viewAllSuspects()
assert isinstance(suspects, list)
if suspects:
item = suspects[0]
assert hasattr(item, "fakeField")
def test_view_my_incidents_invalid_should_fail():
invalid_officer_id = -9999
result = service.viewMyIncidents(invalid_officer_id)
assert isinstance(result, dict)
assert len(result) > 0
def test_add_duplicate_suspect():
with pytest.raises(DatabaseError):
service.addSuspect(suspect, incidentID, officerID, roleDescription)

Util Folder

#db.properties.py

```
db.properties
[database]
host = localhost
database = CrimeReportingSystem
user = root
password = mysql
port = 3306
```

Main Module

#main.py

```python
main.py > MainModule > run
from entity.auth import Authentication
from dao.CrimeAnalysisServiceImpl import CrimeAnalysisServiceImpl
from exceptions.user_defined_exceptions import AuthenticationError, DatabaseError, IncidentNotFoundException,
from entity.Suspects import Suspects
from entity.Incidents import Incidents
from entity.Reports import Reports
from entity.Officers import Officers
from entity.Criminals import Criminals
from entity.Victims import Victims
from datetime import datetime,date
import mysql.connector

class MainModule:
    def __init__(self):
        self.service = CrimeAnalysisServiceImpl()
        self.current_user = None

    def run(self):
        print("=== Crime Reporting System ===")
        self.current_user = Authentication.public_access()

        while True:
            if self.current_user['Role'] == 'Public':
                self.public_menu()
            elif self.current_user['Role'] == 'Officer':
                self.officer_menu()
            elif self.current_user['Role'] == 'Admin':
                self.admin_menu()

    def public_menu(self):
        print("\n=== Public Menu ===")
        print("1. View recent incidents")
        print("2. View incidents by date range")
        print("3. View incidents by area/city")
        print("4. Login as officer/admin")
        print("0. Exit")

        try:
            choice = int(input("Enter your choice: "))

            if choice == 1:
                try:
                    self.service.print_recent_incidents()
                except mysql.connector.Error as e:
                    raise DatabaseError(f"Database Connection Failed: {str(e)}")
                except Exception as e:
                    print(f"Unexpected error: {str(e)}")
```

main.py > MainModule > run

```
13  class MainModule:
30      def public_menu(self):
48              elif choice == 2:
49                  try:
50                      startDate = input("Start date (YYYY-MM-DD): ")
51                      endDate = input("End date (YYYY-MM-DD): ")
52                      incidents = self.service.get_incidents_in_date_range_public(startDate, endDate)
53                      if not incidents:
54                          raise IncidentNotFoundException("Incidents not found within this date range.")
55
56                      print(f"\nIncidents that happened between '{startDate}' and '{endDate}: \n")
57                      print("------------------------------------------------------------")
58                      for incident in incidents:
59                          print(f"Incident Type: {incident.incidentType}")
60                          print(f"Incident Date: {incident.incidentDate}")
61                          print(f"Area: {incident.area}")
62                          print(f"City: {incident.city}")
63                          print(f"Description: {incident.description}")
64                          print("------------------------------------------------------------")
65                  except IncidentNotFoundException as e:
66                      print(f"No incidents found: {str(e)}")
67                  except mysql.connector.Error as e:
68                      raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
69
70              elif choice == 3:
71                  try:
72                      print("\nLeave the field blank if you don't want that filter")
73                      area = input("Enter area: ").strip()
74                      city = input("Enter city: ").strip()
75                      if not area and not city:
76                          print("Both fields cannot be left empty.")
77                          return
78                      incidents = self.service.get_incidents_by_area_city_public(area, city)
79                      if not incidents:
80                          raise IncidentNotFoundException("Incidents not found in this area and city.")
81                      location = ", ".join(filter(None, [area, city]))
82                      print(f"Incidents that happened in {location}:")
83                      print("------------------------------------------------------------")
84                      for incident in incidents:
85                          print(f"Incident Type: {incident['IncidentType']}")
86                          print(f"Incident Date: {incident['IncidentDate']}")
87                          if area: print(f"Area: {incident['Area']}")
88                          if city: print(f"City: {incident['City']}")
89                          print(f"Description: {incident['Description']}")
90                          print("------------------------------------------------------------")
91                  except IncidentNotFoundException as e:
92                      print(f"No incidents found: {str(e)}")
93                  except mysql.connector.Error as e:
94                      raise DatabaseError(f"Database Connection Failed: {str(e)}")
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)}")
```

```python
class MainModule:

    def officer_menu(self):
        print(f"\n=== Officer Menu ===")
        print("\n-- Incident Management --")
        print("1. View all incident details with assignments")
        print("2. View my assigned incidents")
        print("3. Filter incidents")

        print("\n-- Suspect Management --")
        print("4. Access suspects database")
        print("5. Create new suspect record")
        print("6. Access criminals database")

        print("\n-- Case Management --")
        print("7. Create new victim record")
        print("8. View case details (victims/suspects/evidence)")
        print("9. Generate/Update case report")
        print("10. View case report")

        print("11. Logout")

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

            if choice == 1:
                try:
                    incidents = self.service.viewAllIncidents()

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

                    print("\n=== All Incidents ===")
                    for i, incident in enumerate(incidents, start=1):
                        print(f"\nIncident {i}:")
                        print(f"  ID          : {incident.incidentID}")
                        print(f"  Type        : {incident.incidentType}")
                        print(f"  Date        : {incident.incidentDate}")
                        print(f"  Area        : {incident.area}")
                        print(f"  City        : {incident.city}")
                        print(f"  Description : {incident.description}")
                        print(f"  Status      : {incident.status}")
                        print(f"  Officer ID  : {incident.officerID}")
                        print("-----------------------------------------------------------")
                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):
161
162             elif choice == 2:
163                 try:
164                     officerID = self.current_user['UserID']
165                     incidents = self.service.viewMyIncidents(officerID)
166                     if not incidents:
167                         raise IncidentNotFoundException(f"No incidents are found.")
168
169                     print("==== My Assignments ====")
170                     for i, incident in enumerate(incidents, start=1):
171                         print(f"\nIncident {i}:")
172                         print(f"  ID          : {incident['IncidentID']}")
173                         print(f"  Type        : {incident['IncidentType']}")
174                         print(f"  Date        : {incident['IncidentDate']}")
175                         print(f"  Area        : {incident['Area']}")
176                         print(f"  City        : {incident['City']}")
177                         print(f"  Description : {incident['Description']}")
178                         print(f"  Status      : {incident['Status']}")
179                         print(f"  Officer ID  : {incident['OfficerID']}")
180                 except mysql.connector.Error as e:
181                     raise DatabaseError(f"Database Connection Failed: {str(e)}")
182                 except Exception as e:
183                     print(f"Unknown error: {str(e)}")
184
185             elif choice == 3:
186                 try:
187                     print("\n==== Filtering the Incidents ====\n")
188                     print("Fill the parameters on whose basis you wish to filter and leave the rest blank")
189                     print("\nDate Filters:")
190                     start_date = input("Enter the start date (YYYY-MM-DD): ").strip()
191                     end_date = input("Enter the end date (YYYY-MM-DD): ").strip()
192                     area = input("Enter the area: ").strip()
193                     city = input("Enter the city: ").strip()
194                     incidentType = input("Enter the type of incident: ").strip()
195                     officerID_input = input("Enter the Officer ID: ")
196                     officerID = int(officerID_input) if officerID_input else None
197
198                     incidents = self.service.get_filtered_incidents(start_date,end_date,area,city,incidentType,officerID)
199                     if not incidents:
200                         raise IncidentNotFoundException("Incidents not found.")
201                     print("==== Filtered Results ====")
202                     for i, incident in enumerate(incidents, start=1):
203                         print(f"\nIncident {i}:")
204                         print(f"  ID          : {incident['IncidentID']}")
205                         print(f"  Type        : {incident['IncidentType']}")
206                         print(f"  Date        : {incident['IncidentDate']}")
207                         print(f"  Area        : {incident['Area']}")
208                         print(f"  City        : {incident['City']}")
209                         print(f"  Description : {incident['Description']}")
210                         print(f"  Status      : {incident['Status']}")
211                         print(f"  Officer ID  : {incident['OfficerID']}")
212
213                 except mysql.connector.Error as e:
214                     raise DatabaseError(f"Database Connection Failed: {str(e)}")
215                 except Exception as e:
216                     print(f"Unknown error: {str(e)}")
217
```

```python
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")
                        print("-"*60)
                except mysql.connector.Error as e:
                    raise DatabaseError(f"Database Connection Failed: {str(e)}")
                except Exception as e:
                    print(f"Unknown error: {str(e)}")
```

```python
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:
115     def officer_menu(self):

326             elif choice == 7:
327                 try:
328                     print("=== Add a New Victim ===")
329                     firstName = input("First Name: ").strip()
330                     lastName = input("Last Name: ").strip()
331
332                     try:
333                         dob_input = input("Date of Birth (YYYY-MM-DD): ").strip()
334                         dateOfBirth = datetime.strptime(dob_input, "%Y-%m-%d").date()
335                     except ValueError:
336                         print("Invalid date format. Please enter as YYYY-MM-DD.")
337                         return
338
339                     gender = input("Gender: ").strip()
340                     residentialAddress = input("Residential Address: ").strip()
341
342                     try:
343                         contactNumber = int(input("Contact Number: "))
344                         aadhaarNumber = int(input("Aadhaar Number: "))
345                     except ValueError:
346                         print("Contact Number and Aadhaar Number must be numeric.")
347                         return
348
349                     try:
350                         incidentID = int(input("Incident ID involved: "))
351                         officerID = int(input("Your Officer ID: "))
352                     except ValueError:
353                         print("Incident ID and Officer ID must be numeric.")
354                         return
355                     roleDescription = input("Role in Incident: ").strip()
356
357                     new_victim = Victims(
358                         FirstName=firstName,
359                         LastName=lastName,
360                         DateOfBirth=dateOfBirth,
361                         Gender=gender,
362                         ResidentialAddress=residentialAddress,
363                         ContactNumber=contactNumber,
364                         AadhaarNumber=aadhaarNumber,
365                         IncidentID=incidentID
366                     )
367                     victimID = self.service.addVictim(new_victim, incidentID, officerID, roleDescription)
368
369                     if victimID:
370                         print(f"Victim successfully associated with Incident {incidentID}. Victim ID: {victimID}")
371                     else:
372                         print(f"Victim with Aadhaar {aadhaarNumber} already exists for Incident {incidentID}.")
373
374                 except IncidentNotFoundException:
375                     print("Error: Incident not found. Please verify the Incident ID.")
376                 except Exception as e:
377                     print(f"Unexpected error: {str(e)}")
```

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)}")
```

```python
class MainModule:
    def officer_menu(self):
            elif choice == 9:
                try:
                    choice = input("Do you wish to generate an incident report or update an existing report (Generate/Update): ").strip().lower()
                    officer_id = self.current_user['UserID']
                    if choice == 'generate':
                        print("\n=== Generate Case Report ===")
                        incident_id = input("Enter Incident ID: ").strip()
                        report_details = input("Enter Report Details: ").strip()
                        status = input("Enter Report Status (Draft/Finalized): ").strip()
                        report_date = date.today()

                        incident = Incidents(IncidentID=incident_id)
                        officer = Officers(OfficerID=officer_id)

                        new_report = Reports(
                            Incidents=incident,
                            Officers=officer,
                            ReportDate=report_date,
                            ReportDetails=report_details,
                            Status=status
                        )
                        reportID = self.service.generate_incident_report(new_report)
                        print(f"Report generated successfully! Report ID: {reportID}")
                    elif choice == 'update':
                        print("==== Update Case Report ====\n")
                        incident_id = input("Enter Incident ID: ").strip()
                        report_details = input("Enter Updated Report Details: ").strip()
                        status = input("Enter Updated Status (Draft/Finalized): ").strip()
                        report_date = date.today()

                        incident = Incidents(IncidentID=incident_id)
                        officer = Officers(OfficerID=officer_id)

                        updated_report = Reports(
                            Incidents=incident,
                            Officers=officer,
                            ReportDate=report_date,
                            ReportDetails=report_details,
                            Status=status
                        )
                        if self.service.update_incident_report(updated_report):
                            print("Report updated successfully!")
                        else:
                            raise ReportNotFoundException()
                except DuplicateEntryException as e:
                    print(f"Duplicate Entry found. {str(e)}")
                except ReportNotFoundException as e:
                    print(f"Report not found. {str(e)}")
                except DatabaseError as e:
                    print(f"Database Error: {str(e)}")
                except Exception as e:
                    print(f"Unexpected Error: {str(e)}")
```

```
main.py > MainModule > officer_menu
13  class MainModule:
15      def officer_menu(self):
84              elif choice == 10:
85                  try:
86                      incidentID = int(input("Enter Incident ID: "))
87                      report = self.service.view_incident_report(incidentID)
88                      if not report:
89                          print(f"No report found for Incident ID {incidentID}")
90                      else:
91                          print("\n======================== Incident Report ========================")
92                          print(f"Incident ID      : {report['IncidentID']}")
93                          print(f"Incident Type    : {report['IncidentType']}")
94                          print(f"Area             : {report['Area']}")
95                          print(f"City             : {report['City']}")
96                          print(f"Description      : {report['Description']}")
97                          print(f"Incident Status  : {report['Status']}")
98                          print(f"Report Date      : {report['ReportDate']}")
99                          print(f"Report Details   : {report['ReportDetails']}")
00                          print(f"Report Status    : {report['ReportStatus']}")
01
02                  except mysql.connector.Error as e:
03                      raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
04                  except Exception as e:
05                      print(f"Unexpected error: {str(e)}")
06
07              elif choice == 11:
08                  self.current_user = Authentication.public_access()
09
10              else:
11                  print("Please enter a number between 0 - 3")
12
13          except ValueError as e:
14              print(f"Invalid input {str(e)}")
15          except Exception as e:
16              print(f"Unknown Error: {str(e)}")
```

```
main.py > MainModule > officer_menu
13  class MainModule:
518     def admin_menu(self):
519         print(f"\n=== Admin Menu ({self.current_user['Username']}) ===")
520         print("\n-- Records Management --")
521         print("1. Create criminal record")
522         print("2. Create incident record")
523         print("3. Add new officer")
524         print("4. View case assignments")
525         print("5. View all law enforcement agencies")
526
527         print("\n-- Analytics --")
528         print("6. Officer performance dashboard")
529         print("7. Criminal analytics")
530         print("8. Crime pattern analytics")
531
532         print("\n-- Case Oversight --")
533         print("9. Finalize reports & update case status")
534
535         print("10. Logout")
536
537         try:
538             choice = int(input("Enter choice: "))
539
540             if choice == 1:
541                 try:
542                     print("\n=== Add Criminal Record ===")
543                     incidentID = input("Enter Incident ID: ")
544                     aadhaarNumber = input("Enter Aadhaar Number of criminal: ")
545                     punishmentDetails = input("Enter Punishment Details: ")
546
547                     criminal = Criminals(
548                         IncidentID=incidentID,
549                         AadhaarNumber=aadhaarNumber,
550                         PunishmentDetails=punishmentDetails
551                     )
552
553                     criminal_id = self.service.addCriminal(criminal)
554                     if criminal_id:
555                         print(f"Criminal added successfully. Criminal ID: {criminal_id}")
556                     else:
557                         print("Criminal entry already exists in the database.")
558
559                 except SuspectNotFoundException as e:
560                     print(f"Suspect ID not found in the database.")
561                 except DatabaseError as e:
562                     print(f"Database Error: {str(e)}")
563                 except Exception as e:
564                     print(f"Unexpected error: {str(e)}")
565
```

main.py > MainModule > officer_menu

```
 13  class MainModule:
518      def admin_menu(self):
565
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.get_officers_by_location(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.count_open_incidents_by_officer(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.create_incident(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)}")
616
```

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("-------------------------------------------------------------")
685                       except mysql.connector.Error as e:
686                           raise DatabaseError(f"Database Connection Failed: {str(e)}")
687                       except Exception as e:
688                           print(f"Unknown error: {str(e)}")
689
690                   elif choice == 5:
691                       try:
692                           agencies = self.service.view_all_agencies()
693                           if not agencies:
694                               print("No Law Enforcement Agencies found.")
695                           else:
696                               print("\n=== Law Enforcement Agencies ===")
697                               for agency in agencies:
698                                   print(agency)
699                                   print()
700                                   print("-" * 40)
701
702                       except AgencyNotFoundException as e:
703                           print("Agency not found.")
704                       except Exception as e:
705                           print(f"Unexpected Error: {str(e)}")
```

main.py > MainModule > officer_menu

```
 13   class MainModule:
518       def admin_menu(self):
706
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("\n-----------------------------")
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                   except mysql.connector.Error as e:
725                       raise DatabaseError(f"Database connection error: {e.msg}") from e
726                   except Exception as e:
727                       print(f"Unexpected error: {str(e)}")
728
729               elif choice == 7:
730                   try:
731                       analytics = self.service.get_criminal_analytics()
732
733                       print("\n==== Criminal Analytics Dashboard ====")
734
735                       print("\nTop 5 Most Frequent Criminals:")
736                       if analytics['top_criminals']:
737                           index = 1
738                           for criminal in analytics['top_criminals']:
739                               print(f"  {index}. {criminal['FirstName']} {criminal['LastName']} - {criminal['crime_count']} cases")
740                               index += 1
741                       else:
742                           print("  No data found for top criminals.")
743
744                       print("\nCrime Type Distribution:")
745                       if analytics['crime_types']:
746                           for crime_type in analytics['crime_types']:
747                               print(f"  {crime_type} : {analytics['crime_types'][crime_type]}")
748                       else:
749                           print("  No crime type data available.")
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):
```

```
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
```

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.