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)

Modules & Libraries:

1. **tabulate**: For displaying tabular data in a readable format.
2. **dotenv**: For securely loading environment variables.
3. **mysql. connector**: For database interactions.
4. **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 three primary roles:

1. Admin

The Admin has full control over the system, including incidents, suspects, and case management. They can view, update, and oversee all data, including access to activity logs. Admins are responsible for the overall supervision and maintenance of the system.

2. Officer

Officers can report and view incidents along with related data. However, they do not have permissions to modify case details or manage user accounts.

3. Public

The Public role does not require authentication. Users can access limited incident data based on area, time period, or recent events, but cannot view internal or sensitive case details.

Authentication & Registration

Secure Registration: Users must enter a role-specific secret code (stored as environment variables) to register.

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 for security.
- Role: Role of the user (‘admin’ or ‘officer’).
- OfficerID (Foreign Key): Links the user to the corresponding officer profile.

2. Incidents

- IncidentID (Primary Key): Unique identifier.
- IncidentType: Type of crime (e.g., robbery, assault).
- IncidentDate: Date it occurred.
- Area: Area name.
- City: City of occurrence.
- Description: Narrative of the incident.
- Status: Open, Closed, Under Investigation.
- OfficerID (Foreign Key to Officers.OfficerID): Officer who reported or is assigned to the case.

3. Victims

- VictimID (Primary Key): Unique identifier.
- FirstName, LastName: Name of the victim.
- DateOfBirth: Date of birth.
- Gender
- ResidentialAddress, ContactNumber
- AadhaarNumber
- IncidentID (Foreign Key to Incidents.IncidentID): Incident linked to this victim.

4. Suspects

- SuspectID (Primary Key): Unique identifier.
- FirstName, LastName
- DateOfBirth, Gender
- ResidentialAddress, ContactNumber, AadhaarNumber (Unique)

5. Evidence

- EvidenceID (Primary Key): Unique identifier.
- EvidenceName: Name or type (e.g., fingerprint, CCTV).
- Description: Description of the evidence.
- LocationFound: Where it was located.
- IncidentID (Foreign Key): Linked incident.

6. Criminals

- CriminalID (Primary Key): Unique identifier.
- IncidentID (Foreign Key): Incident linked to this criminal.
- SuspectID (Foreign Key): Suspect linked to this criminal record.
- AadhaarNumber: Aadhaar (Unique).
- PunishmentDetails: Punishments or court outcomes.

7. Cases

- CaseID (Primary Key): Unique identifier for each case.
- CaseDescription: Summary or high-level view of the case.

8. Reports

- ReportID (Primary Key): Unique identifier.
- IncidentID (Foreign Key): Linked incident.
- ReportingOfficerID (Foreign Key): Officer who wrote the report.
- ReportDate: Date of reporting.
- ReportDetails: Full report.

- Status: Draft, Finalized, or Closed.

9. SuspectIncidents

- SuspectID, IncidentID (Composite Primary Key).
- AadhaarNumber
- RoleDescription: Role played in the incident.
- AddedByOfficerID (Foreign Key): Officer who added this info.

10. LawEnforcementAgencies

- AgencyID (Primary Key): Unique identifier.
- AgencyName
- Jurisdiction
- EmailAddress

11. Officers

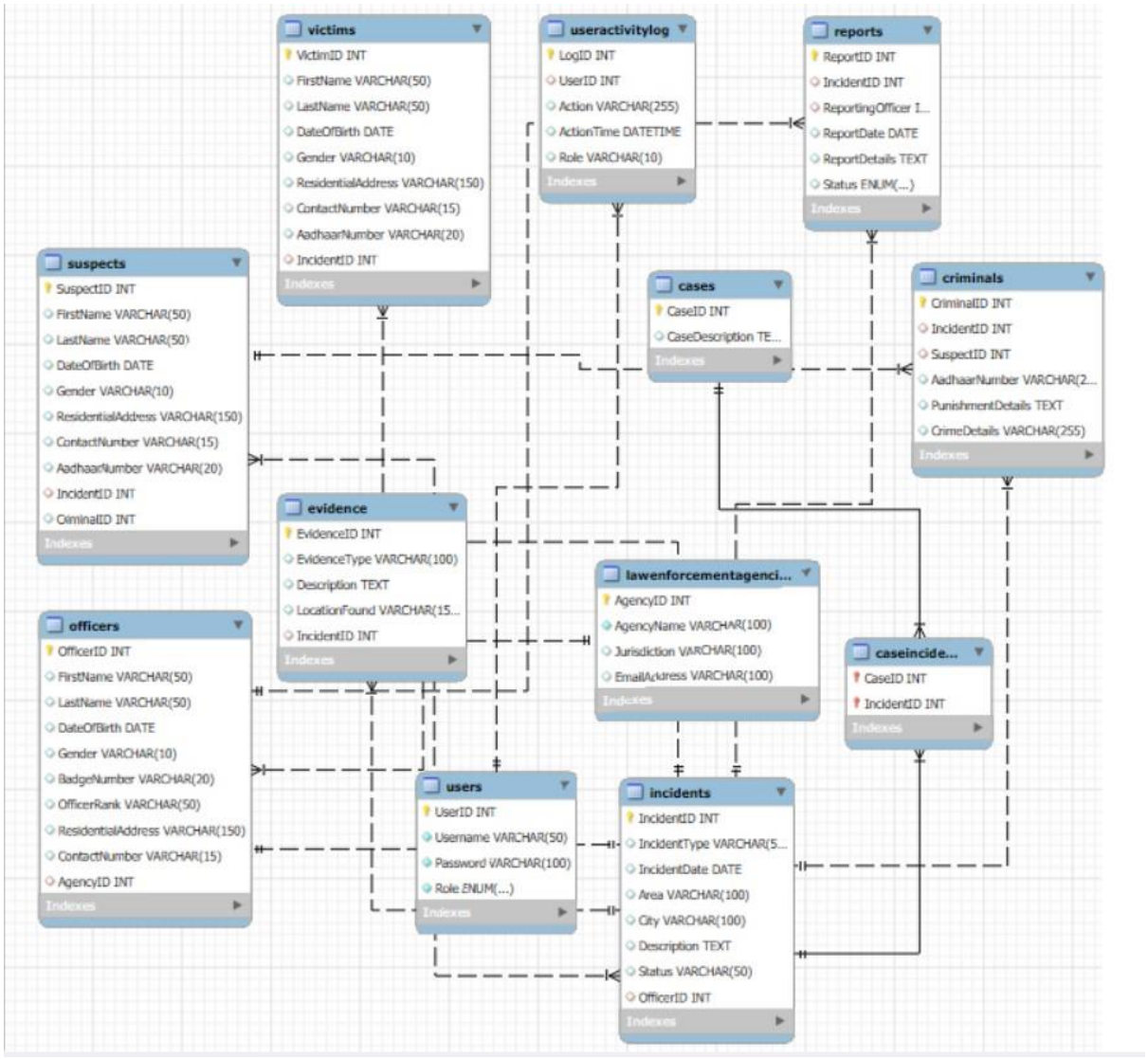
- OfficerID (Primary Key): Unique identifier.
- Personal Details: Name, DOB, Gender, BadgeNumber, Ranking.
- Posting Info: City, State, ServiceJoiningDate.
- Contact Info: Address, ContactNumber.
- AgencyID (Foreign Key): Associated law enforcement agency.

Relationships Summary

- Users ↔ Officers: Each user may optionally map to one officer.
- Officers ↔ Agencies: Each officer belongs to one agency; an agency has many officers.
- Incidents ↔ Officers: Each incident is reported by one officer; an officer can report many incidents.
- Incidents ↔ Agencies: Each incident is linked to one agency via the reporting officer.

- Incidents ↔ Victims: One incident can have many victims; each victim belongs to one incident.
- Incidents ↔ Suspects: Many-to-many via SuspectIncidents.
- Incidents ↔ Evidence: One incident can have many evidence entries.
- Incidents ↔ Reports: One incident can have many reports by different officers.
- Suspects ↔ Criminals: A suspect may become a criminal; each criminal links to one suspect.

ER Diagram:



Implementation and Testing

Dao:

CrimeAnalysisService.py:

main.py ● CrimeAnalysisService.py ×

dao > CrimeAnalysisService.py > ...

```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

    @abstractmethod
    def generate_incident_report(self, report):
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

main.py ● CrimeAnalysisService.py ×

dao > CrimeAnalysisService.py > ...

```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
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

```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
```

```python
    @abstractmethod
    def print_recent_incidents(self):
        pass

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

CrimeAnalysisServiceImpl.py:

```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, SuspectNotFoundE


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)}")


    def update_incident_status(self, status, incidentID):
```

main.py ● CrimeAnalysisServiceImpl.py ×

dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl

```
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.")
```

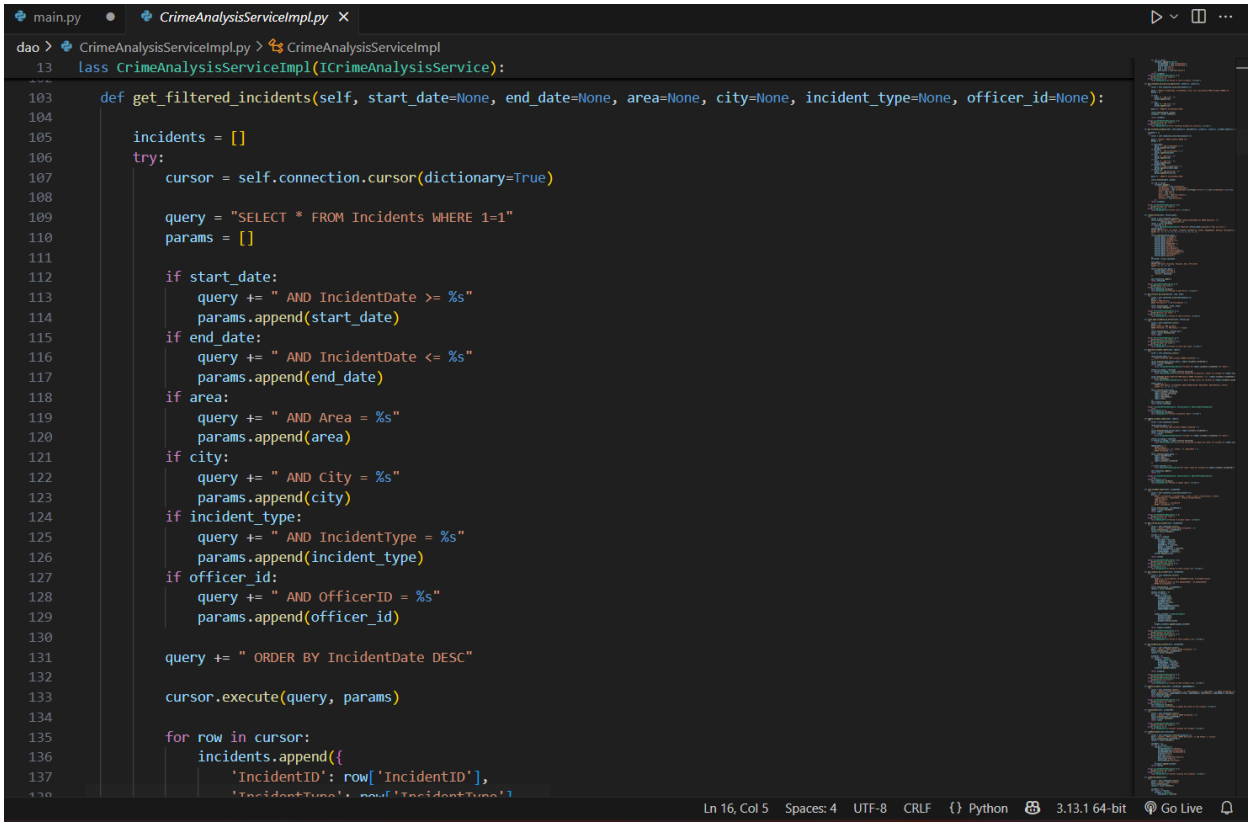
Ln 16, Col 5 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

main.py ● CrimeAnalysisServiceImpl.py ×

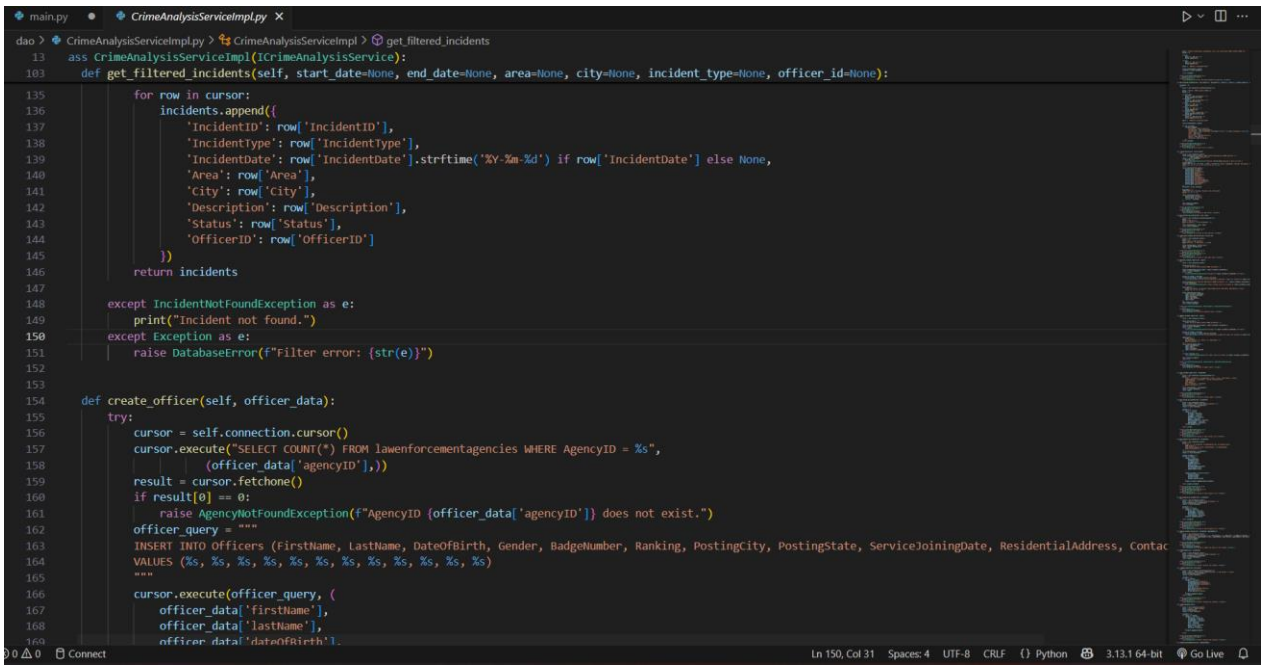
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl

```
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def get_incidents_in_date_range_public(self, start_date, end_date):
            return incidents
        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch incidents: {str(e)}")

    def get_incidents_by_area_city_public(self, area=None, city=None):
        try:
            cursor = self.connection.cursor(dictionary=True)

            query = "SELECT IncidentType, IncidentDate, Area, City, Description FROM Incidents WHERE 1=1"
            params = []

            if area:
                query += " AND Area = %s"
                params.append(area)

            if city:
                query += " AND City = %s"
                params.append(city)

            query += " ORDER BY IncidentDate DESC"

            cursor.execute(query, params)
            incidents = cursor.fetchall()

            return incidents

        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            raise DatabaseError(f"Error fetching incidents by area/city: {str(e)}")

    def get_filtered_incidents(self, start_date=None, end_date=None, area=None, city=None, incident_type=None, officer_id=None):

```

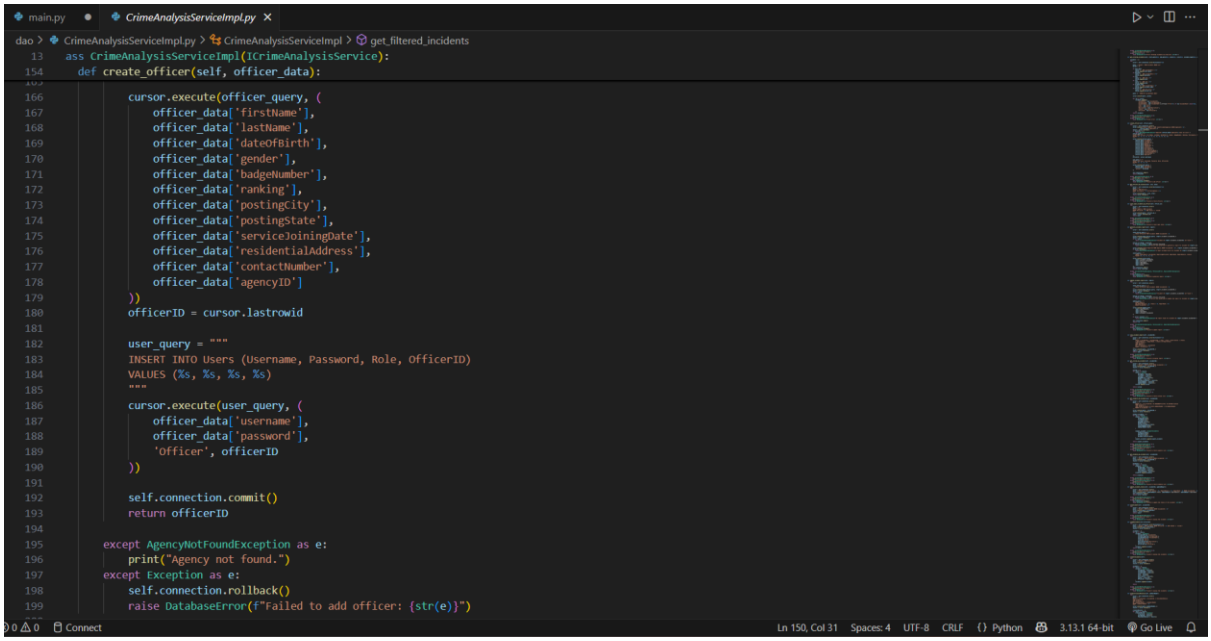
Ln 16, Col 5 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live



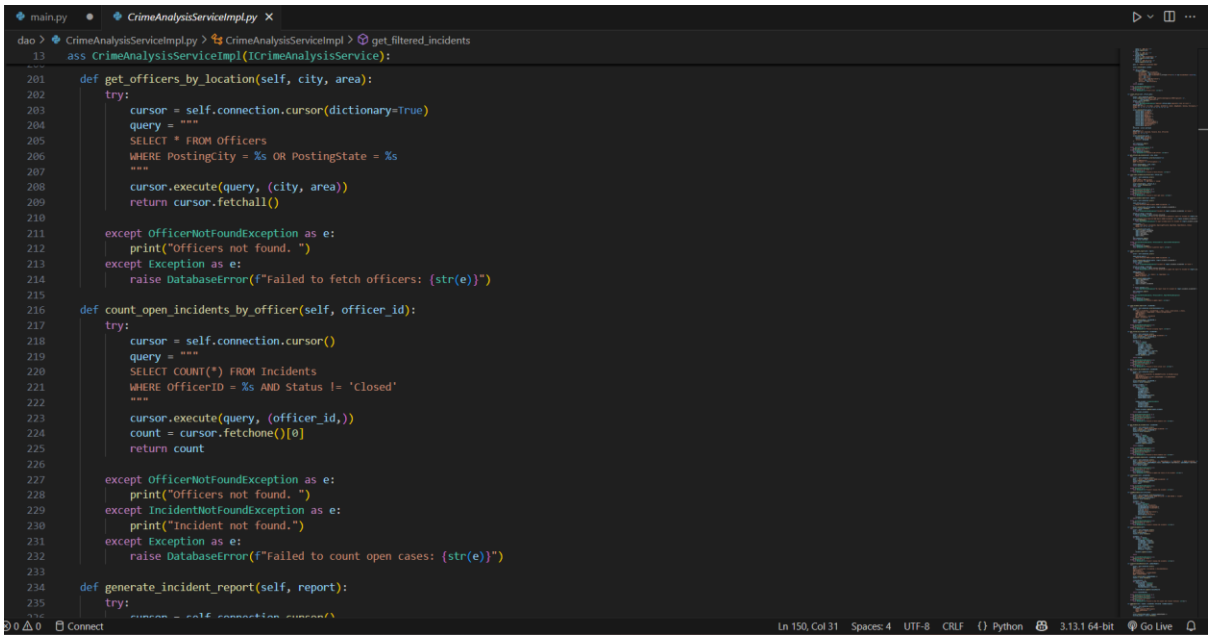
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl
13  lass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
103     def get_filtered_incidents(self, start_date=None, end_date=None, area=None, city=None, incident_type=None, officer_id=None):
104
105         incidents = []
106         try:
107             cursor = self.connection.cursor(dictionary=True)
108
109             query = "SELECT * FROM Incidents WHERE 1=1"
110             params = []
111
112             if start_date:
113                 query += " AND IncidentDate >= %s"
114                 params.append(start_date)
115             if end_date:
116                 query += " AND IncidentDate <= %s"
117                 params.append(end_date)
118             if area:
119                 query += " AND Area = %s"
120                 params.append(area)
121             if city:
122                 query += " AND City = %s"
123                 params.append(city)
124             if incident_type:
125                 query += " AND IncidentType = %s"
126                 params.append(incident_type)
127             if officer_id:
128                 query += " AND OfficerID = %s"
129                 params.append(officer_id)
130
131             query += " ORDER BY IncidentDate DESC"
132
133             cursor.execute(query, params)
134
135             for row in cursor:
136                 incidents.append({
137                     'IncidentID': row['IncidentID'],
                    'IncidentType': row['IncidentType']
Ln 16, Col 5  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



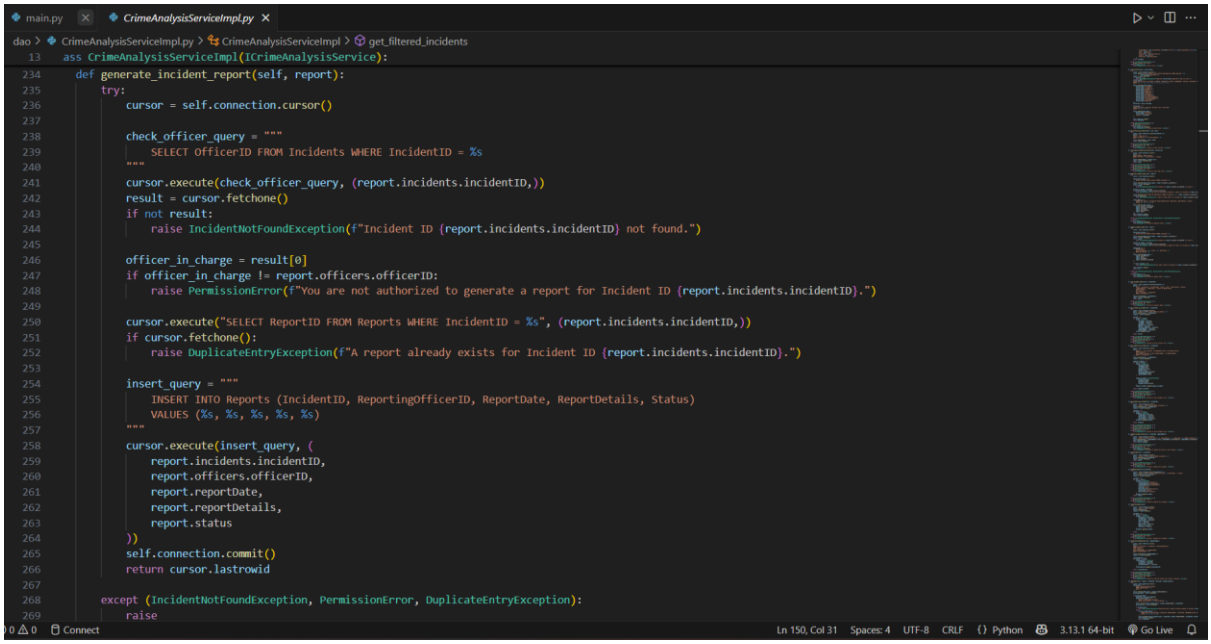
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
103     def get_filtered_incidents(self, start_date=None, end_date=None, area=None, city=None, incident_type=None, officer_id=None):
135             for row in cursor:
136                 incidents.append({
137                     'IncidentID': row['IncidentID'],
138                     'IncidentType': row['IncidentType'],
139                     'IncidentDate': row['IncidentDate'].strftime('%Y-%m-%d') if row['IncidentDate'] else None,
140                     'Area': row['Area'],
141                     'City': row['City'],
142                     'Description': row['Description'],
143                     'Status': row['Status'],
144                     'OfficerID': row['OfficerID']
145                 })
146             return incidents
147
148         except IncidentNotFoundException as e:
149             print("Incident not found.")
150         except Exception as e:
151             raise DatabaseError(f"Filter error: {str(e)}")
152
153
154     def create_officer(self, officer_data):
155         try:
156             cursor = self.connection.cursor()
157             cursor.execute("SELECT COUNT(*) FROM lawenforcementagencies WHERE AgencyID = %s",
158                            (officer_data['agencyID'],))
159             result = cursor.fetchone()
160             if result[0] == 0:
161                 raise AgencyNotFoundException(f"AgencyID {officer_data['agencyID']} does not exist.")
162             officer_query = """
163             INSERT INTO Officers (FirstName, LastName, DateOfBirth, Gender, BadgeNumber, Ranking, PostingCity, PostingState, ServiceJoiningDate, ResidentialAddress, Contac
164             VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
165             """
166             cursor.execute(officer_query, (
167                 officer_data['firstName'],
168                 officer_data['lastName'],
169                 officer_data['dateOfBirth'],
0 ⚠ 0  Connect  Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



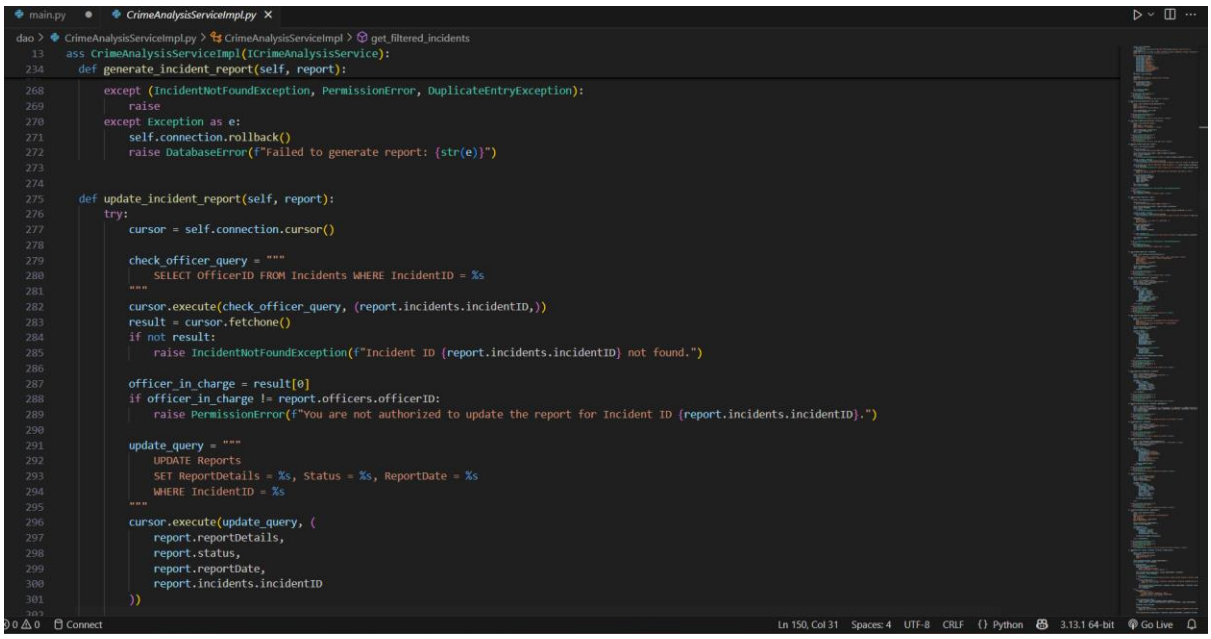
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
154   def create_officer(self, officer_data):
166           cursor.execute(officer_query, (
167               officer_data['firstName'],
168               officer_data['lastName'],
169               officer_data['dateOfBirth'],
170               officer_data['gender'],
171               officer_data['badgeNumber'],
172               officer_data['ranking'],
173               officer_data['postingCity'],
174               officer_data['postingState'],
175               officer_data['serviceJoiningDate'],
176               officer_data['residentialAddress'],
177               officer_data['contactNumber'],
178               officer_data['agencyID']
179           ))
180           officerID = cursor.lastrowid
181
182           user_query = """
183           INSERT INTO Users (Username, Password, Role, OfficerID)
184           VALUES (%s, %s, %s, %s)
185           """
186           cursor.execute(user_query, (
187               officer_data['username'],
188               officer_data['password'],
189               'Officer', officerID
190           ))
191
192           self.connection.commit()
193           return officerID
194
195       except AgencyNotFoundException as e:
196           print("Agency not found.")
197       except Exception as e:
198           self.connection.rollback()
199           raise DatabaseError(f"Failed to add officer: {str(e)}")
```



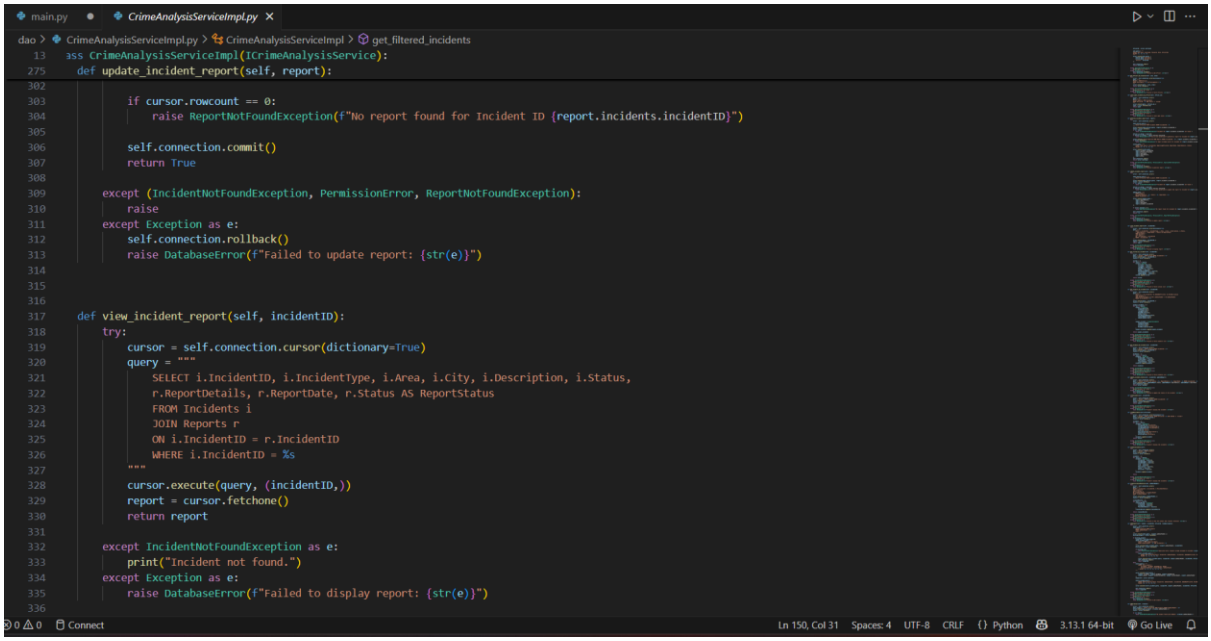
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
201   def get_officers_by_location(self, city, area):
202       try:
203           cursor = self.connection.cursor(dictionary=True)
204           query = """
205           SELECT * FROM Officers
206           WHERE PostingCity = %s OR PostingState = %s
207           """
208           cursor.execute(query, (city, area))
209           return cursor.fetchall()
210
211       except OfficerNotFoundException as e:
212           print("Officers not found. ")
213       except Exception as e:
214           raise DatabaseError(f"Failed to fetch officers: {str(e)}")
215
216   def count_open_incidents_by_officer(self, officer_id):
217       try:
218           cursor = self.connection.cursor()
219           query = """
220           SELECT COUNT(*) FROM Incidents
221           WHERE OfficerID = %s AND Status != 'Closed'
222           """
223           cursor.execute(query, (officer_id,))
224           count = cursor.fetchone()[0]
225           return count
226
227       except OfficerNotFoundException as e:
228           print("Officers not found. ")
229       except IncidentNotFoundException as e:
230           print("Incident not found.")
231       except Exception as e:
232           raise DatabaseError(f"Failed to count open cases: {str(e)}")
233
234   def generate_incident_report(self, report):
235       try:
```



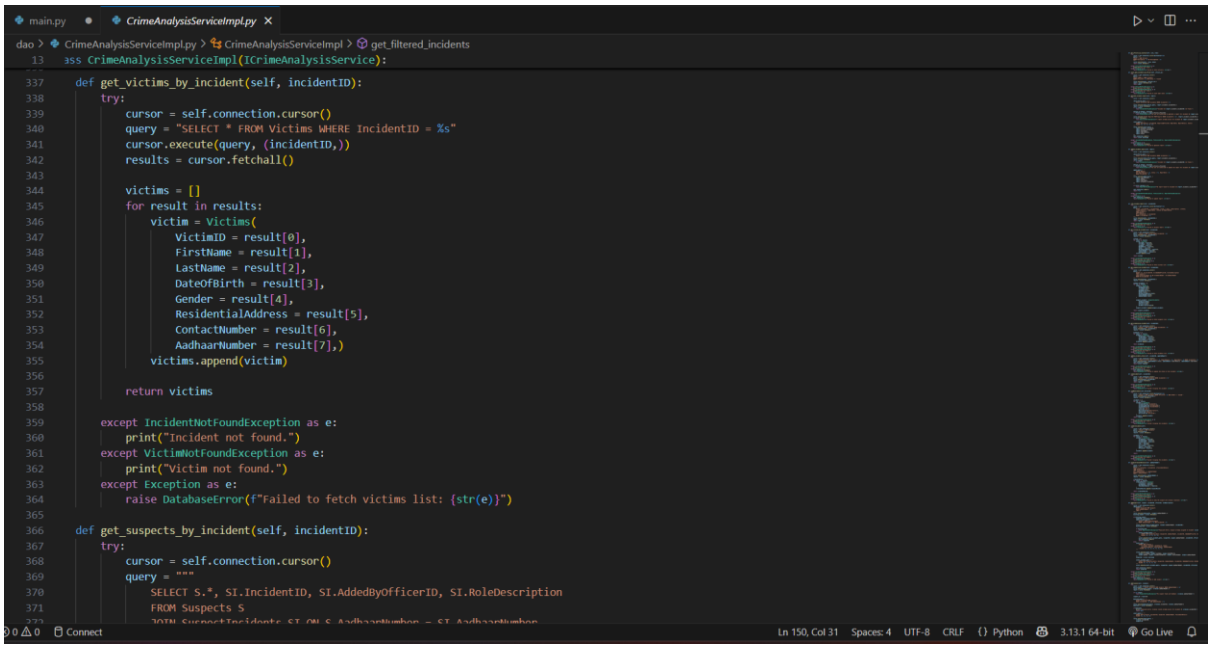
```
main.py  CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def generate_incident_report(self, report):
        try:
            cursor = self.connection.cursor()

            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
```



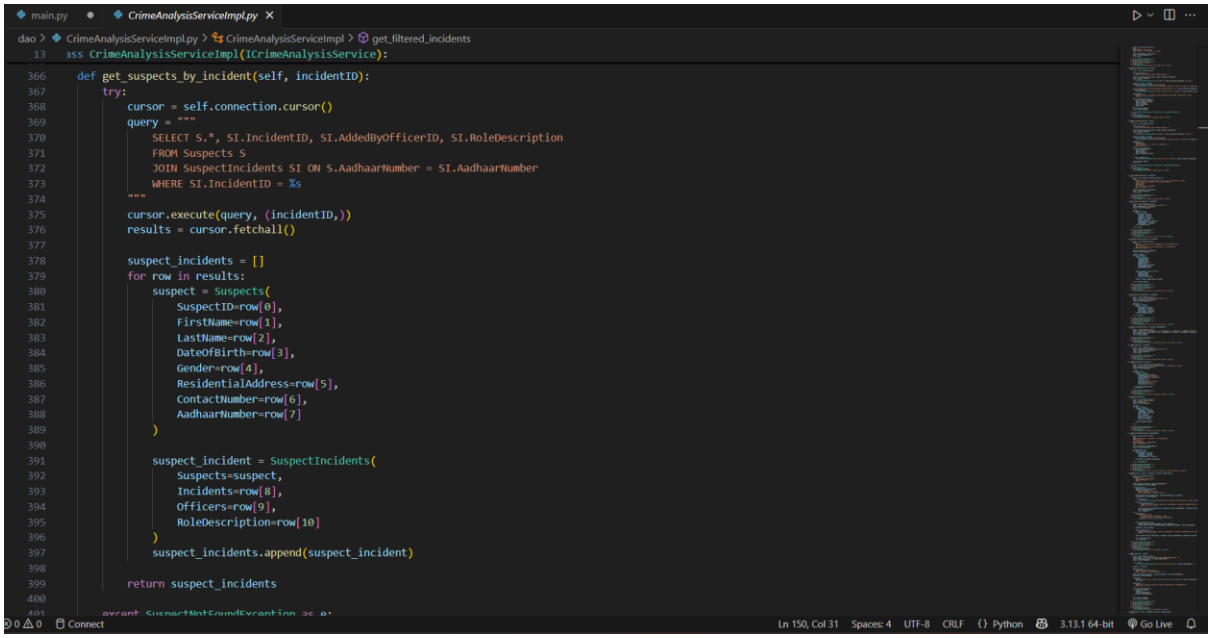
```
main.py  CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def generate_incident_report(self, report):
        except (IncidentNotFoundException, PermissionError, DuplicateEntryException):
            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
            ))
```



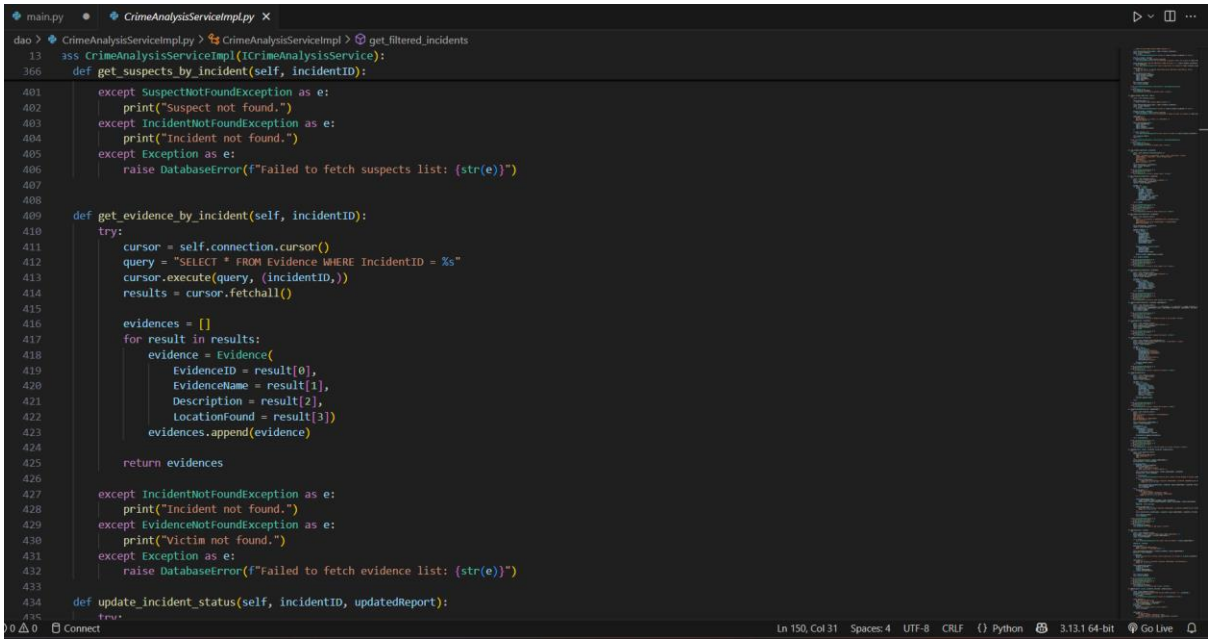
```
ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def update_incident_report(self, report):

            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)}")



    def view_incident_report(self, incidentID):
        try:
            cursor = self.connection.cursor(dictionary=True)
            query = """
                SELECT i.IncidentID, i.IncidentType, i.Area, i.City, i.Description, i.Status,
                r.ReportDetails, r.ReportDate, r.Status AS ReportStatus
                FROM Incidents i
                JOIN Reports r
                ON i.IncidentID = r.IncidentID
                WHERE i.IncidentID = %s
            """
            cursor.execute(query, (incidentID,))
            report = cursor.fetchone()
            return report

        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            raise DatabaseError(f"Failed to display report: {str(e)}")
```



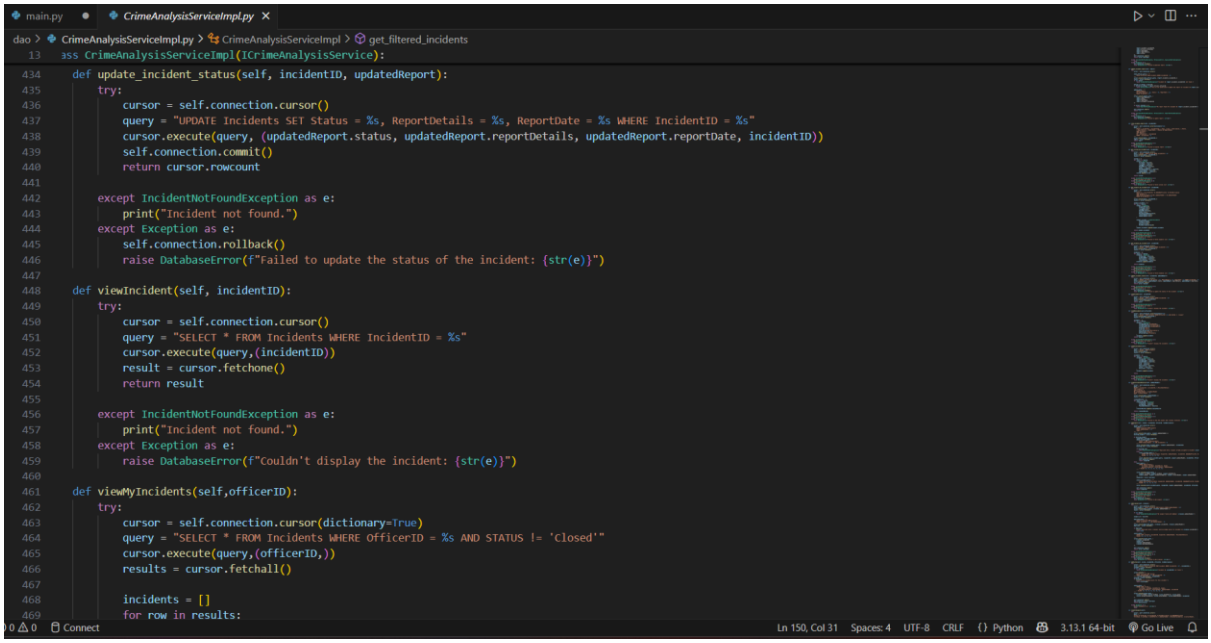
```
ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def get_victims_by_incident(self, incidentID):
        try:
            cursor = self.connection.cursor()
            query = "SELECT * FROM Victims WHERE IncidentID = %s"
            cursor.execute(query, (incidentID,))
            results = cursor.fetchall()

            victims = []
            for result in results:
                victim = Victims(
                    VictimID = result[0],
                    FirstName = result[1],
                    LastName = result[2],
                    DateOfBirth = result[3],
                    Gender = result[4],
                    ResidentialAddress = result[5],
                    ContactNumber = result[6],
                    AadhaarNumber = result[7],)
                victims.append(victim)

            return victims

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

    def get_suspects_by_incident(self, incidentID):
        try:
            cursor = self.connection.cursor()
            query = """
                SELECT S.*, SI.IncidentID, SI.AddedByOfficerID, SI.RoleDescription
                FROM Suspects S
```



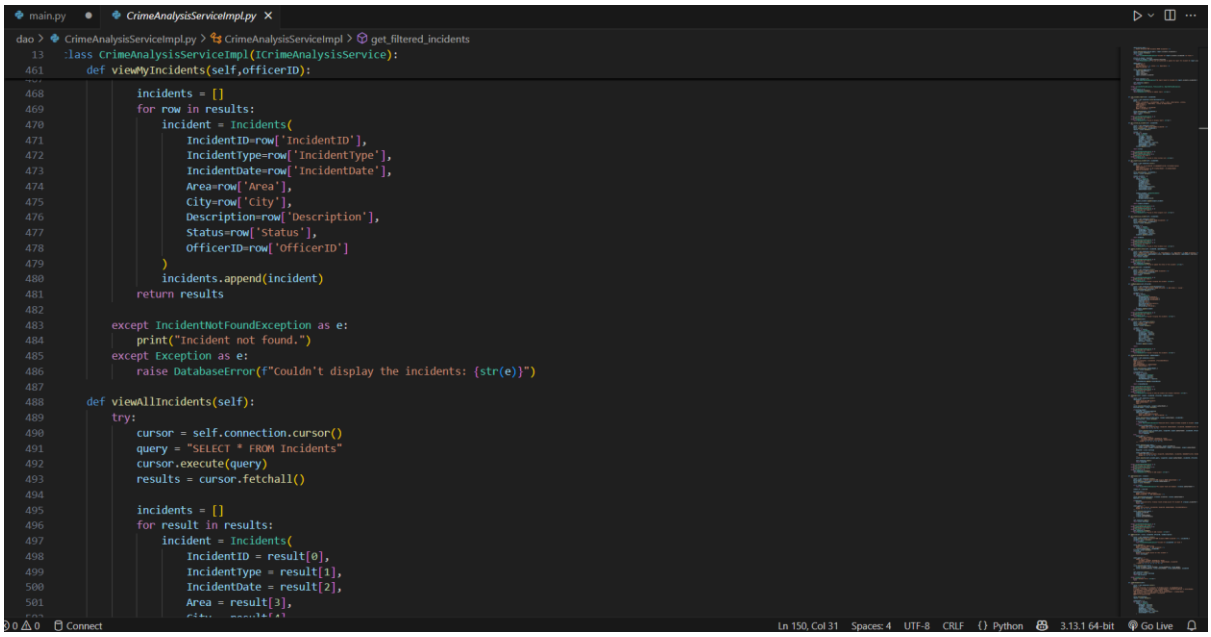
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def get_suspects_by_incident(self, incidentID):
        try:
            cursor = self.connection.cursor()
            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:
0 ⚠ 0  Connect                Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



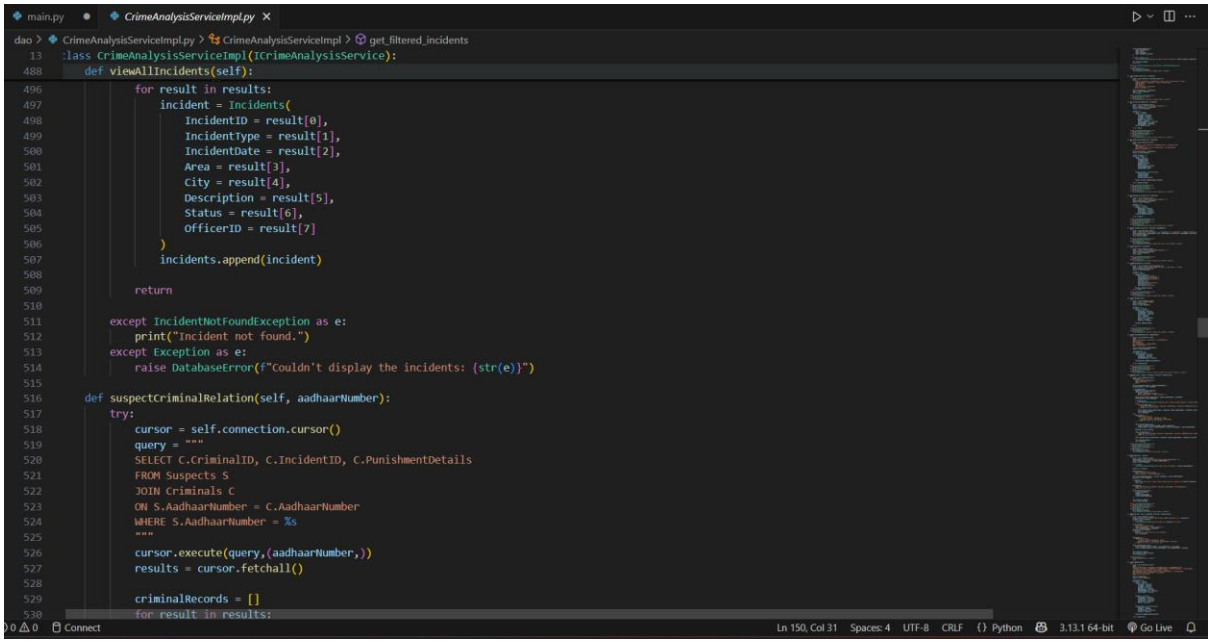
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
    def get_suspects_by_incident(self, incidentID):
        except SuspectNotFoundException as e:
            print("Suspect not found.")
        except IncidentNotFoundException as e:
            print("Incident not found.")
        except Exception as e:
            raise DatabaseError(f"Failed to fetch suspects list: {str(e)}")


    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:
0 ⚠ 0  Connect                Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



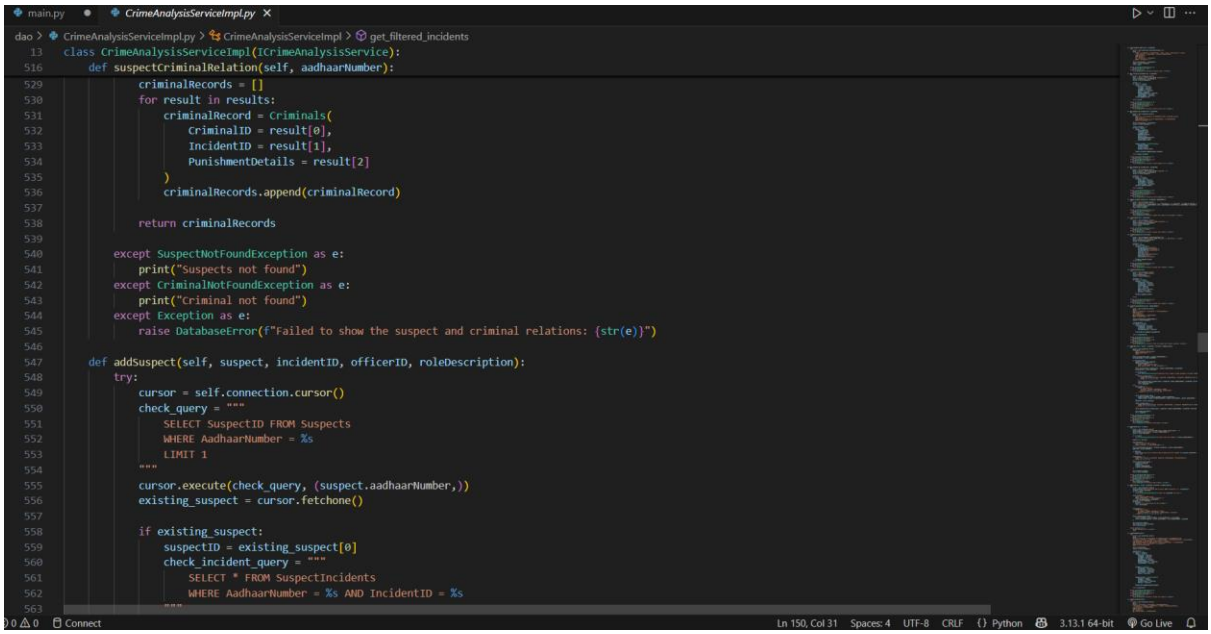
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  ass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
434     def update_incident_status(self, incidentID, updatedReport):
435         try:
436             cursor = self.connection.cursor()
437             query = "UPDATE Incidents SET Status = %s, ReportDetails = %s, ReportDate = %s WHERE IncidentID = %s"
438             cursor.execute(query, (updatedReport.status, updatedReport.reportDetails, updatedReport.reportDate, incidentID))
439             self.connection.commit()
440             return cursor.rowcount
441
442         except IncidentNotFoundException as e:
443             print("Incident not found.")
444         except Exception as e:
445             self.connection.rollback()
446             raise DatabaseError(f"Failed to update the status of the incident: {str(e)}")
447
448     def viewIncident(self, incidentID):
449         try:
450             cursor = self.connection.cursor()
451             query = "SELECT * FROM Incidents WHERE IncidentID = %s"
452             cursor.execute(query,(incidentID))
453             result = cursor.fetchone()
454             return result
455
456         except IncidentNotFoundException as e:
457             print("Incident not found.")
458         except Exception as e:
459             raise DatabaseError(f"Couldn't display the incident: {str(e)}")
460
461     def viewMyIncidents(self,officerID):
462         try:
463             cursor = self.connection.cursor(dictionary=True)
464             query = "SELECT * FROM Incidents WHERE OfficerID = %s AND STATUS != 'Closed'"
465             cursor.execute(query,(officerID,))
466             results = cursor.fetchall()
467
468             incidents = []
469             for row in results:
0 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



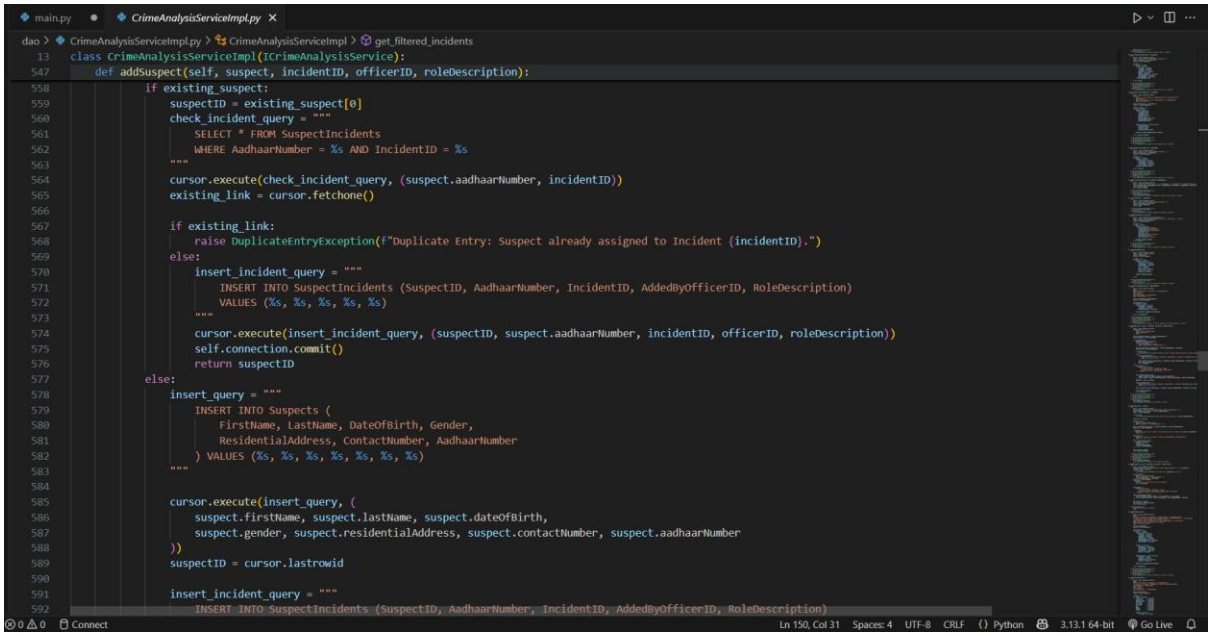
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  :lass CrimeAnalysisServiceImpl(ICrimeAnalysisService):
461     def viewMyIncidents(self,officerID):
468             incidents = []
469             for row in results:
470                 incident = Incidents(
471                     IncidentID=row['IncidentID'],
472                     IncidentType=row['IncidentType'],
473                     IncidentDate=row['IncidentDate'],
474                     Area=row['Area'],
475                     City=row['City'],
476                     Description=row['Description'],
477                     Status=row['Status'],
478                     OfficerID=row['OfficerID']
479                 )
480                 incidents.append(incident)
481             return results
482
483         except IncidentNotFoundException as e:
484             print("Incident not found.")
485         except Exception as e:
486             raise DatabaseError(f"Couldn't display the incidents: {str(e)}")
487
488     def viewAllIncidents(self):
489         try:
490             cursor = self.connection.cursor()
491             query = "SELECT * FROM Incidents"
492             cursor.execute(query)
493             results = cursor.fetchall()
494
495             incidents = []
496             for result in results:
497                 incident = Incidents(
498                     IncidentID = result[0],
499                     IncidentType = result[1],
500                     IncidentDate = result[2],
501                     Area = result[3],
0 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```

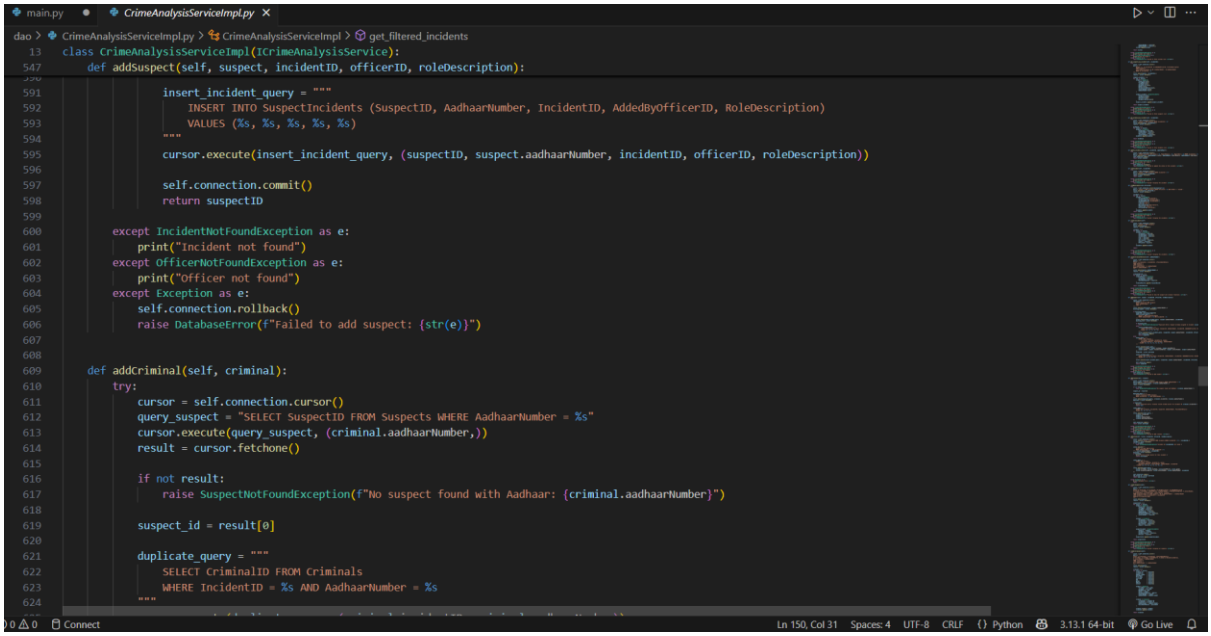


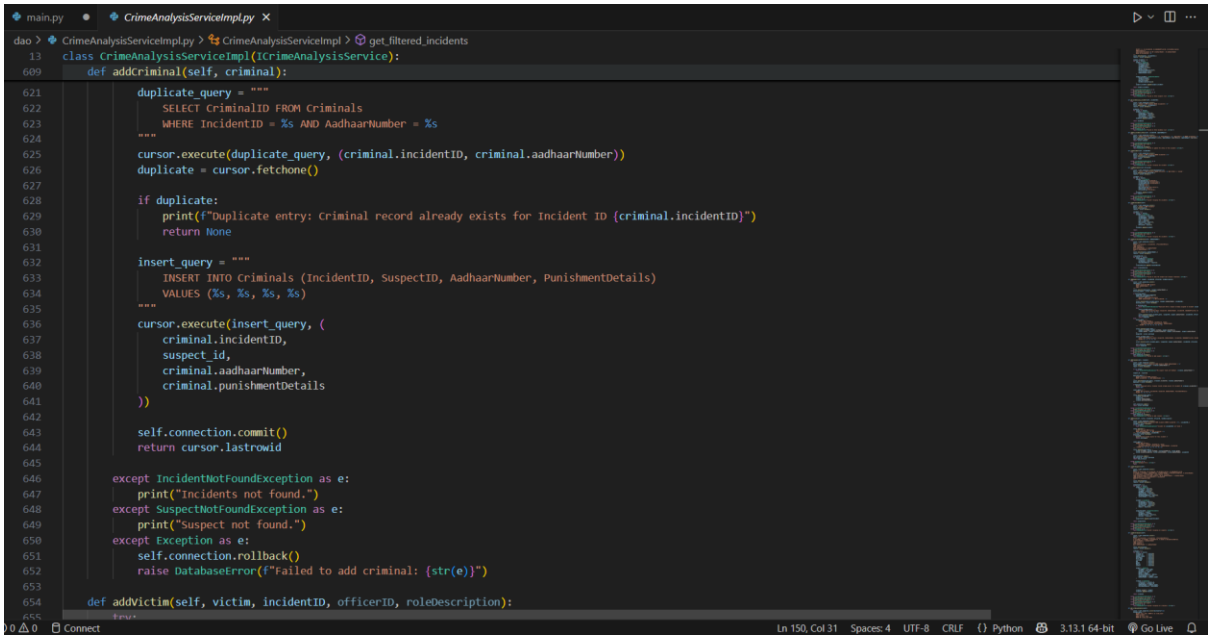
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
488     def viewAllIncidents(self):
496             for result in results:
497                 incident = Incidents(
498                     IncidentID = result[0],
499                     IncidentType = result[1],
500                     IncidentDate = result[2],
501                     Area = result[3],
502                     City = result[4],
503                     Description = result[5],
504                     Status = result[6],
505                     OfficerID = result[7]
506                 )
507                 incidents.append(incident)
508
509             return
510
511         except IncidentNotFoundException as e:
512             print("Incident not found.")
513         except Exception as e:
514             raise DatabaseError(f"Couldn't display the incidents: {str(e)}")
515
516     def suspectCriminalRelation(self, aadhaarNumber):
517         try:
518             cursor = self.connection.cursor()
519             query = """
520             SELECT C.CriminalID, C.IncidentID, C.PunishmentDetails
521             FROM Suspects S
522             JOIN Criminals C
523             ON S.AadhaarNumber = C.AadhaarNumber
524             WHERE S.AadhaarNumber = %s
525             """
526             cursor.execute(query,(aadhaarNumber,))
527             results = cursor.fetchall()
528
529             criminalRecords = []
530             for result in results:
0 0 Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



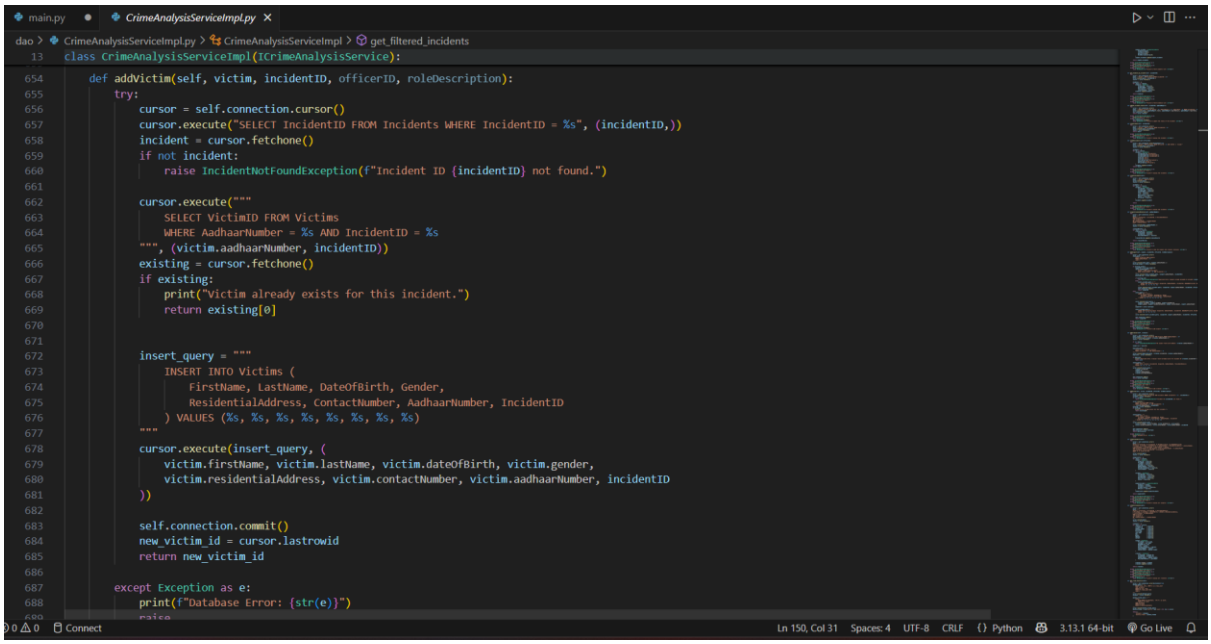
```
main.py ● CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
516     def suspectCriminalRelation(self, aadhaarNumber):
529             criminalRecords = []
530             for result in results:
531                 criminalRecord = Criminals(
532                     CriminalID = result[0],
533                     IncidentID = result[1],
534                     PunishmentDetails = result[2]
535                 )
536                 criminalRecords.append(criminalRecord)
537
538             return criminalRecords
539
540         except SuspectNotFoundException as e:
541             print("Suspects not found")
542         except CriminalNotFoundException as e:
543             print("Criminal not found")
544         except Exception as e:
545             raise DatabaseError(f"Failed to show the suspect and criminal relations: {str(e)}")
546
547     def addSuspect(self, suspect, incidentID, officerID, roleDescription):
548         try:
549             cursor = self.connection.cursor()
550             check_query = """
551                 SELECT SuspectID FROM Suspects
552                 WHERE AadhaarNumber = %s
553                 LIMIT 1
554             """
555             cursor.execute(check_query, (suspect.aadhaarNumber,))
556             existing_suspect = cursor.fetchone()
557
558             if existing_suspect:
559                 suspectID = existing_suspect[0]
560                 check_incident_query = """
561                     SELECT * FROM SuspectIncidents
562                     WHERE AadhaarNumber = %s AND IncidentID = %s
563                 """
0 0 Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```


main.py
CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
def addSuspect(self, suspect, incidentID, officerID, roleDescription):
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)
Connect
Ln 150, Col 31
Spaces: 4
UTF-8
CRLF
Python
3.13.1 64-bit
Go Live

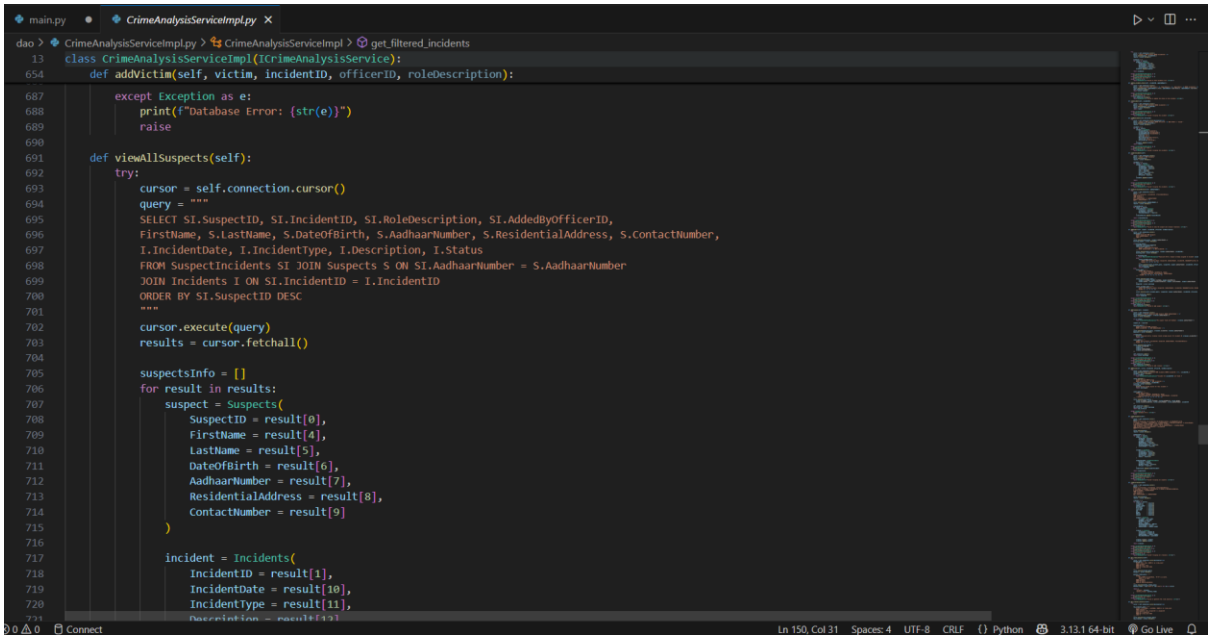

main.py
CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
def addSuspect(self, suspect, incidentID, officerID, roleDescription):
insert_incident_query = """
INSERT INTO SuspectIncidents (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)}")
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
"""
Connect
Ln 150, Col 31
Spaces: 4
UTF-8
CRLF
Python
3.13.1 64-bit
Go Live



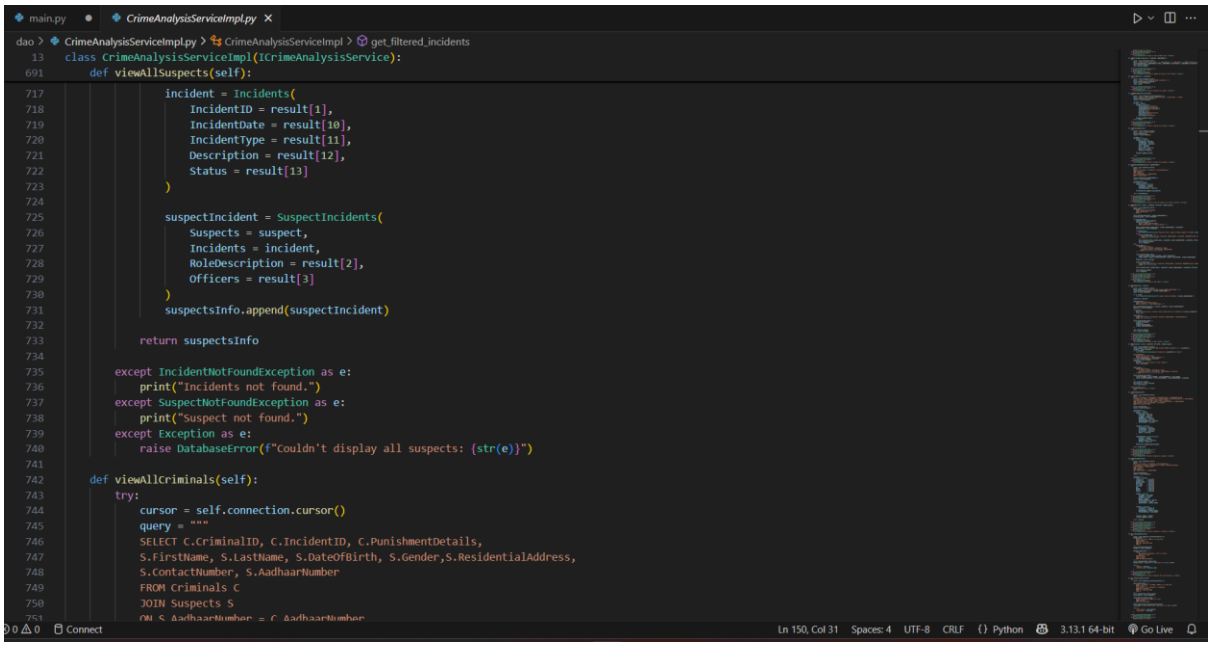
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
609     def addCriminal(self, criminal):
621             duplicate_query = """
622                 SELECT CriminalID FROM Criminals
623                 WHERE IncidentID = %s AND AadhaarNumber = %s
624             """
625             cursor.execute(duplicate_query, (criminal.incidentID, criminal.aadhaarNumber))
626             duplicate = cursor.fetchone()
627
628             if duplicate:
629                 print(f"Duplicate entry: Criminal record already exists for Incident ID {criminal.incidentID}")
630                 return None
631
632             insert_query = """
633                 INSERT INTO Criminals (IncidentID, SuspectID, AadhaarNumber, PunishmentDetails)
634                 VALUES (%s, %s, %s, %s)
635             """
636             cursor.execute(insert_query, (
637                 criminal.incidentID,
638                 suspect_id,
639                 criminal.aadhaarNumber,
640                 criminal.punishmentDetails
641             ))
642
643             self.connection.commit()
644             return cursor.lastrowid
645
646         except IncidentNotFoundException as e:
647             print("Incidents not found.")
648         except SuspectNotFoundException as e:
649             print("Suspect not found.")
650         except Exception as e:
651             self.connection.rollback()
652             raise DatabaseError(f"Failed to add criminal: {str(e)}")
653
654     def addVictim(self, victim, incidentID, officerID, roleDescription):
655         try:
0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



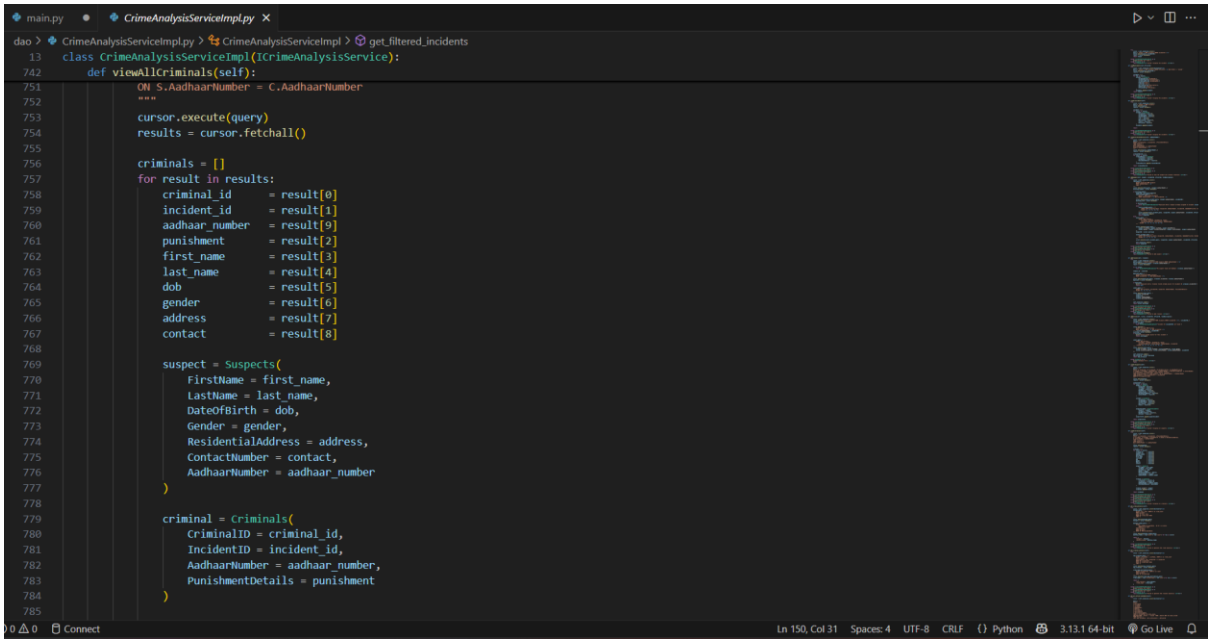
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
654     def addVictim(self, victim, incidentID, officerID, roleDescription):
655         try:
656             cursor = self.connection.cursor()
657             cursor.execute("SELECT IncidentID FROM Incidents WHERE IncidentID = %s", (incidentID,))
658             incident = cursor.fetchone()
659             if not incident:
660                 raise IncidentNotFoundException(f"Incident ID {incidentID} not found.")
661
662             cursor.execute("""
663                 SELECT VictimID FROM Victims
664                 WHERE AadhaarNumber = %s AND IncidentID = %s
665             """, (victim.aadhaarNumber, incidentID))
666             existing = cursor.fetchone()
667             if existing:
668                 print("Victim already exists for this incident.")
669                 return existing[0]
670
671
672             insert_query = """
673                 INSERT INTO Victims (
674                     FirstName, LastName, DateOfBirth, Gender,
675                     ResidentialAddress, ContactNumber, AadhaarNumber, IncidentID
676                 ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
677             """
678             cursor.execute(insert_query, (
679                 victim.firstName, victim.lastName, victim.dateOfBirth, victim.gender,
680                 victim.residentialAddress, victim.contactNumber, victim.aadhaarNumber, incidentID
681             ))
682
683             self.connection.commit()
684             new_victim_id = cursor.lastrowid
685             return new_victim_id
686
687         except Exception as e:
688             print(f"Database Error: {str(e)}")
689             raise
0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```

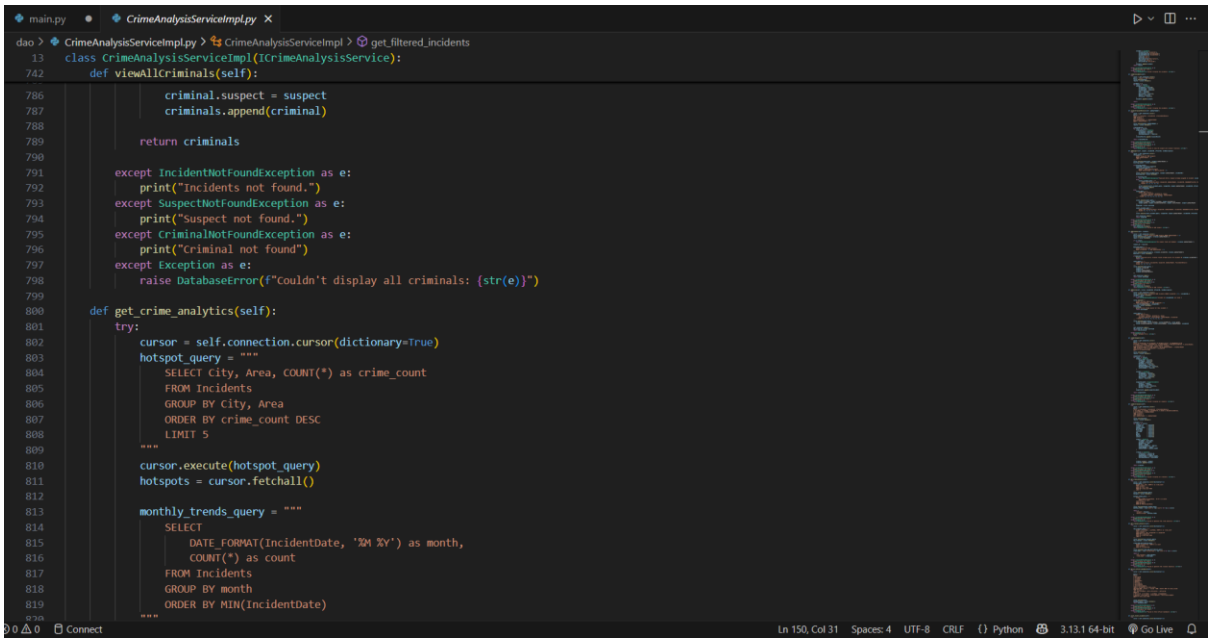


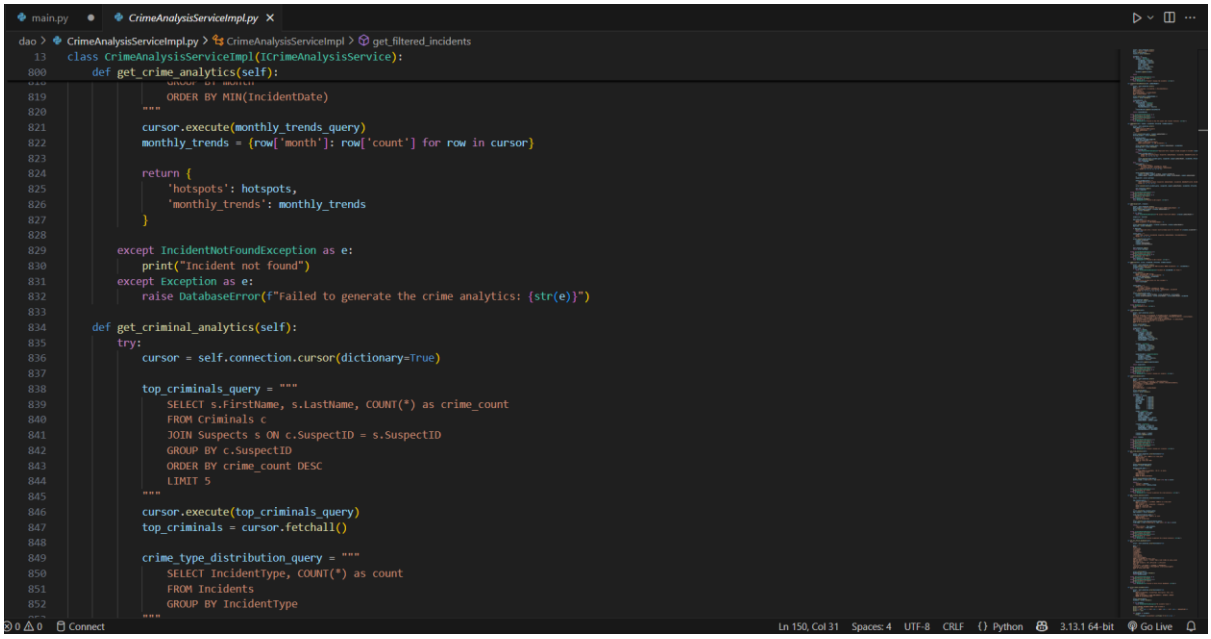
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
654     def addVictim(self, victim, incidentID, officerID, roleDescription):

687         except Exception as e:
688             print(f"Database Error: {str(e)}")
689             raise
690
691     def viewAllSuspects(self):
692         try:
693             cursor = self.connection.cursor()
694             query = """
695             SELECT SI.SuspectID, SI.IncidentID, SI.RoleDescription, SI.AddedByOfficerID,
696             FirstName, S.LastName, S.DateOfBirth, S.AadhaarNumber, S.ResidentialAddress, S.ContactNumber,
697             I.IncidentDate, I.IncidentType, I.Description, I.Status
698             FROM SuspectIncidents SI JOIN Suspects S ON SI.AadhaarNumber = S.AadhaarNumber
699             JOIN Incidents I ON SI.IncidentID = I.IncidentID
700             ORDER BY SI.SuspectID DESC
701             """
702             cursor.execute(query)
703             results = cursor.fetchall()
704
705             suspectsInfo = []
706             for result in results:
707                 suspect = Suspects(
708                     SuspectID = result[0],
709                     FirstName = result[4],
710                     LastName = result[5],
711                     DateOfBirth = result[6],
712                     AadhaarNumber = result[7],
713                     ResidentialAddress = result[8],
714                     ContactNumber = result[9]
715                 )
716
717                 incident = Incidents(
718                     IncidentID = result[1],
719                     IncidentDate = result[10],
720                     IncidentType = result[11],
721                     Description = result[12]

0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



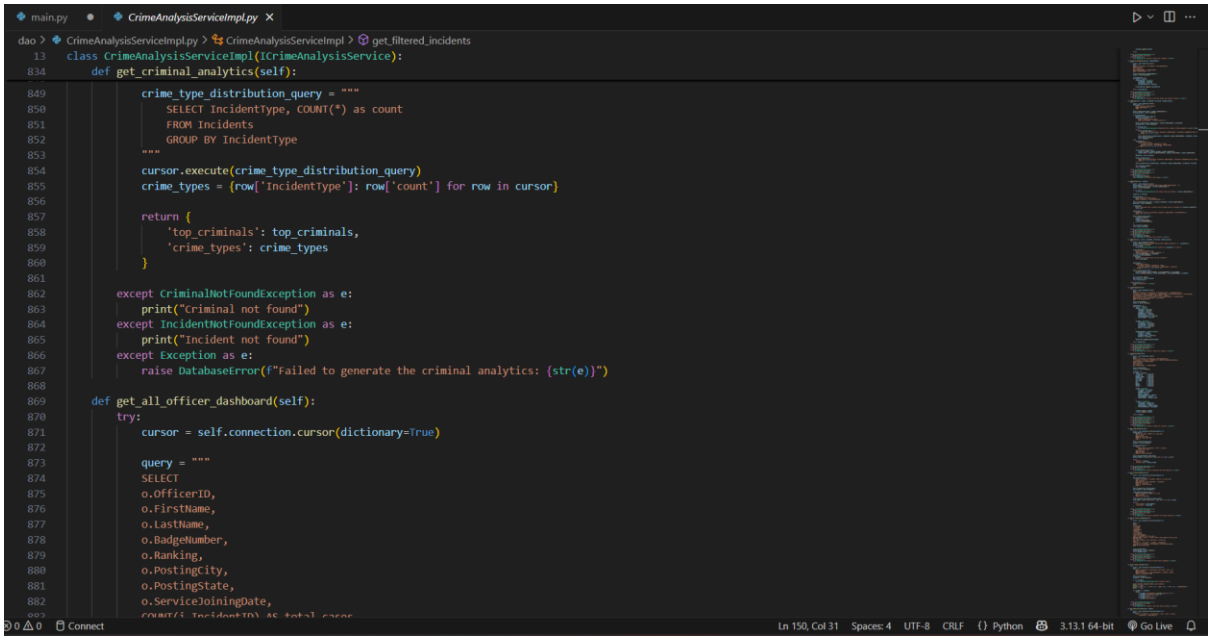
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
691     def viewAllSuspects(self):

717                 incident = Incidents(
718                     IncidentID = result[1],
719                     IncidentDate = result[10],
720                     IncidentType = result[11],
721                     Description = result[12],
722                     Status = result[13]
723                 )
724
725                 suspectIncident = SuspectIncidents(
726                     Suspects = suspect,
727                     Incidents = incident,
728                     RoleDescription = result[2],
729                     Officers = result[3]
730                 )
731                 suspectsInfo.append(suspectIncident)
732
733             return suspectsInfo
734
735         except IncidentNotFoundException as e:
736             print("Incidents not found.")
737         except SuspectNotFoundException as e:
738             print("Suspect not found.")
739         except Exception as e:
740             raise DatabaseError(f"Couldn't display all suspects: {str(e)}")
741
742     def viewAllCriminals(self):
743         try:
744             cursor = self.connection.cursor()
745             query = """
746             SELECT C.CriminalID, C.IncidentID, C.PunishmentDetails,
747             S.FirstName, S.LastName, S.DateOfBirth, S.Gender,S.ResidentialAddress,
748             S.ContactNumber, S.AadhaarNumber
749             FROM Criminals C
750             JOIN Suspects S
751             ON S.AadhaarNumber = C.AadhaarNumber

0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```


main.py
CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
def viewAllCriminals(self):
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
)
Connect
Ln 150, Col 31
Spaces: 4
UTF-8
CRLF
Python
3.13.1 64-bit
Go Live

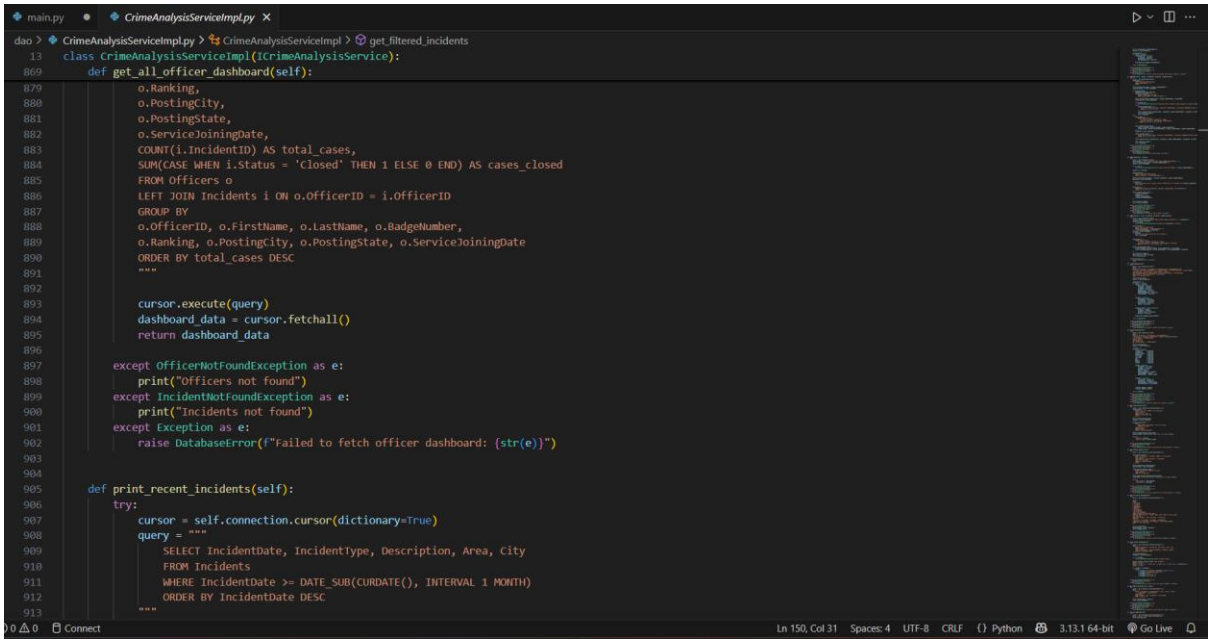

main.py
CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
def viewAllCriminals(self):
criminal.suspect = suspect
criminals.append(criminal)
return criminals
except IncidentNotFoundException 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)}")
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)
"""
Connect
Ln 150, Col 31
Spaces: 4
UTF-8
CRLF
Python
3.13.1 64-bit
Go Live



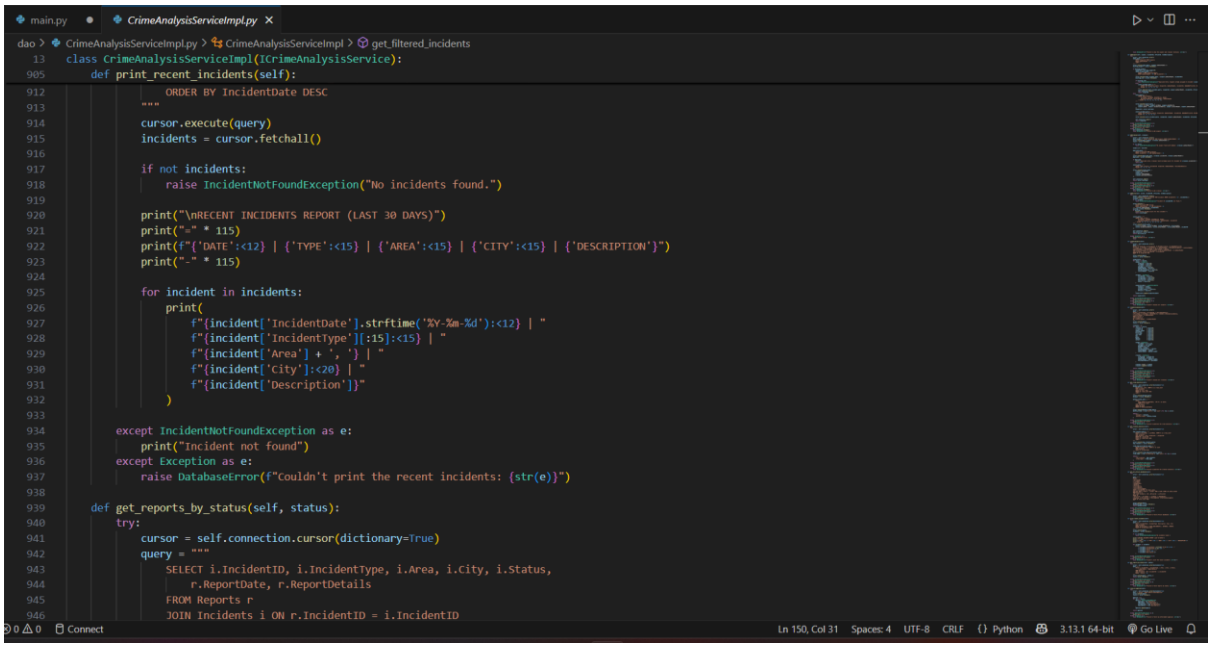
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
800     def get_crime_analytics(self):
819                 ORDER BY MIN(IncidentDate)
820             """
821             cursor.execute(monthly_trends_query)
822             monthly_trends = {row['month']: row['count'] for row in cursor}
823
824             return {
825                 'hotspots': hotspots,
826                 'monthly_trends': monthly_trends
827             }
828
829         except IncidentNotFoundException as e:
830             print("Incident not found")
831         except Exception as e:
832             raise DatabaseError(f"Failed to generate the crime analytics: {str(e)}")
833
834     def get_criminal_analytics(self):
835         try:
836             cursor = self.connection.cursor(dictionary=True)
837
838             top_criminals_query = """
839                 SELECT s.FirstName, s.LastName, COUNT(*) as crime_count
840                 FROM Criminals c
841                 JOIN Suspects s ON c.SuspectID = s.SuspectID
842                 GROUP BY c.SuspectID
843                 ORDER BY crime_count DESC
844                 LIMIT 5
845             """
846             cursor.execute(top_criminals_query)
847             top_criminals = cursor.fetchall()
848
849             crime_type_distribution_query = """
850                 SELECT IncidentType, COUNT(*) as count
851                 FROM Incidents
852                 GROUP BY IncidentType
            """
0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



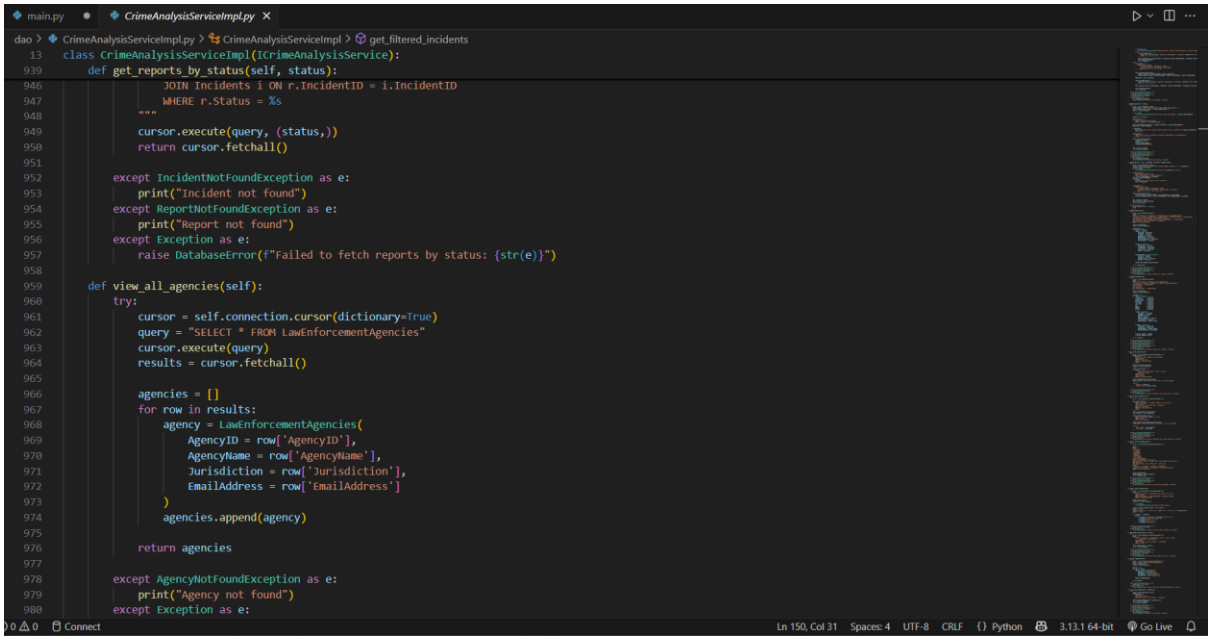
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
834     def get_criminal_analytics(self):
849             crime_type_distribution_query = """
850                 SELECT IncidentType, COUNT(*) as count
851                 FROM Incidents
852                 GROUP BY IncidentType
853             """
854             cursor.execute(crime_type_distribution_query)
855             crime_types = {row['IncidentType']: row['count'] for row in cursor}
856
857             return {
858                 'top_criminals': top_criminals,
859                 'crime_types': crime_types
860             }
861
862         except CriminalNotFoundException as e:
863             print("Criminal not found")
864         except IncidentNotFoundException as e:
865             print("Incident not found")
866         except Exception as e:
867             raise DatabaseError(f"Failed to generate the criminal analytics: {str(e)}")
868
869     def get_all_officer_dashboard(self):
870         try:
871             cursor = self.connection.cursor(dictionary=True)
872
873             query = """
874             SELECT
875             o.OfficerID,
876             o.FirstName,
877             o.LastName,
878             o.BadgeNumber,
879             o.Ranking,
880             o.PostingCity,
881             o.PostingState,
882             o.ServiceJoiningDate,
            COUNT(i.IncidentID) AS total_cases
0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



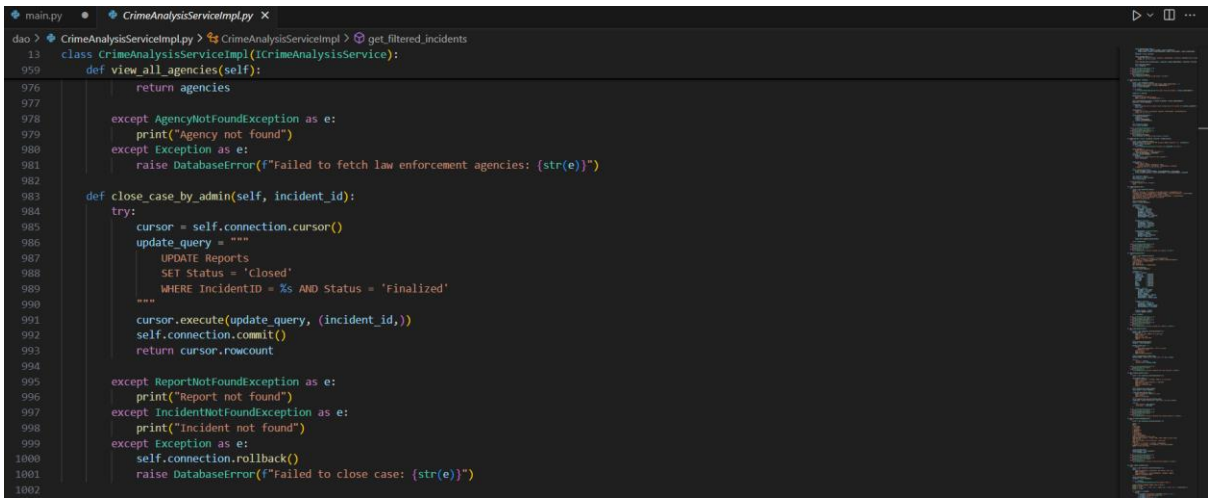
```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
 13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
869      def get_all_officer_dashboard(self):
879              o.Ranking,
880              o.PostingCity,
881              o.PostingState,
882              o.ServiceJoiningDate,
883              COUNT(i.IncidentID) AS total_cases,
884              SUM(CASE WHEN i.Status = 'Closed' THEN 1 ELSE 0 END) AS cases_closed
885              FROM Officers o
886              LEFT JOIN Incidents i ON o.OfficerID = i.OfficerID
887              GROUP BY
888              o.OfficerID, o.FirstName, o.LastName, o.BadgeNumber,
889              o.Ranking, o.PostingCity, o.PostingState, o.ServiceJoiningDate
890              ORDER BY total_cases DESC
891              """
892
893              cursor.execute(query)
894              dashboard_data = cursor.fetchall()
895              return dashboard_data
896
897          except OfficerNotFoundException as e:
898              print("Officers not found")
899          except IncidentNotFoundException as e:
900              print("Incidents not found")
901          except Exception as e:
902              raise DatabaseError(f"Failed to fetch officer dashboard: {str(e)}")
903
904
905      def print_recent_incidents(self):
906          try:
907              cursor = self.connection.cursor(dictionary=True)
908              query = """
909                  SELECT IncidentDate, IncidentType, Description, Area, City
910                  FROM Incidents
911                  WHERE IncidentDate >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
912                  ORDER BY IncidentDate DESC
913              """
0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



```
main.py ●  CrimeAnalysisServiceImpl.py ×
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
 13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
905      def print_recent_incidents(self):
912                  ORDER BY IncidentDate DESC
913              """
914              cursor.execute(query)
915              incidents = cursor.fetchall()
916
917              if not incidents:
918                  raise IncidentNotFoundException("No incidents found.")
919
920              print("\nRECENT INCIDENTS REPORT (LAST 30 DAYS)")
921              print("-" * 115)
922              print(f"{'DATE':<12} | {'TYPE':<15} | {'AREA':<15} | {'CITY':<15} | {'DESCRIPTION'}")
923              print("-" * 115)
924
925              for incident in incidents:
926                  print(
927                      f"{incident['IncidentDate'].strftime('%Y-%m-%d'):<12} | "
928                      f"{incident['IncidentType'][:15]:<15} | "
929                      f"{incident['Area'] + ', ' } | "
930                      f"{incident['City']:<20} | "
931                      f"{incident['Description']}"
932                  )
933
934          except IncidentNotFoundException as e:
935              print("Incident not found")
936          except Exception as e:
937              raise DatabaseError(f"Couldn't print the recent incidents: {str(e)}")
938
939      def get_reports_by_status(self, status):
940          try:
941              cursor = self.connection.cursor(dictionary=True)
942              query = """
943                  SELECT i.IncidentID, i.IncidentType, i.Area, i.City, i.Status,
944                      r.ReportDate, r.ReportDetails
945                  FROM Reports r
946                  JOIN Incidents i ON r.IncidentID = i.IncidentID
0 ⚠ 0  Connect   Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



```
main.py    CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
939     def get_reports_by_status(self, status):
946                 JOIN Incidents i ON r.IncidentID = i.IncidentID
947                 WHERE r.Status = %s
948             """
949             cursor.execute(query, (status,))
950             return cursor.fetchall()
951
952         except IncidentNotFoundException as e:
953             print("Incident not found")
954         except ReportNotFoundException as e:
955             print("Report not found")
956         except Exception as e:
957             raise DatabaseError(f"Failed to fetch reports by status: {str(e)}")
958
959     def view_all_agencies(self):
960         try:
961             cursor = self.connection.cursor(dictionary=True)
962             query = "SELECT * FROM LawEnforcementAgencies"
963             cursor.execute(query)
964             results = cursor.fetchall()
965
966             agencies = []
967             for row in results:
968                 agency = LawEnforcementAgencies(
969                     AgencyID = row['AgencyID'],
970                     AgencyName = row['AgencyName'],
971                     Jurisdiction = row['Jurisdiction'],
972                     EmailAddress = row['EmailAddress']
973                 )
974                 agencies.append(agency)
975
976             return agencies
977
978         except AgencyNotFoundException as e:
979             print("Agency not found")
980         except Exception as e:
0 0  Connect    Ln 150, Col 31  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



```
main.py    CrimeAnalysisServiceImpl.py
dao > CrimeAnalysisServiceImpl.py > CrimeAnalysisServiceImpl > get_filtered_incidents
13  class CrimeAnalysisServiceImpl(ICrimeAnalysisService):
959     def view_all_agencies(self):
976             return agencies
977
978         except AgencyNotFoundException as e:
979             print("Agency not found")
980         except Exception as e:
981             raise DatabaseError(f"Failed to fetch law enforcement agencies: {str(e)}")
982
983     def close_case_by_admin(self, incident_id):
984         try:
985             cursor = self.connection.cursor()
986             update_query = """
987                 UPDATE Reports
988                 SET Status = 'Closed'
989                 WHERE IncidentID = %s AND Status = 'Finalized'
990             """
991             cursor.execute(update_query, (incident_id,))
992             self.connection.commit()
993             return cursor.rowcount
994
995         except ReportNotFoundException as e:
996             print("Report not found")
997         except IncidentNotFoundException as e:
998             print("Incident not found")
999         except Exception as e:
1000            self.connection.rollback()
1001            raise DatabaseError(f"Failed to close case: {str(e)}")
1002
```

Database:

CARS_CreatingTables.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;

-- 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;

-- 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;
```

-- Creating Criminals Table

```
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;
```

-- Creating Users Table

```
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)
);
```

-- Creating SuspectIncidents Table

```
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)
);
```

Database

CARS_InsertionOfData.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', '1980-05-15', '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', 4987654321, 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', 0654321098, 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', 3987654321, 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', '2023-08-20', '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);

-- 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', 8765432109, 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', '1980-02-28', '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', 3987654321, 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', '1980-12-07', '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', 3987654322, 223322332233, 30);

-- Inserting data in Suspects table

INSERT INTO Suspects (FirstName, LastName, DateOfBirth, Gender, ResidentialAddress, ContactNumber, AadhaarNumber) VALUES

('Vijay', 'Malhotra', '1980-05-15', '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', 8329678905, 678967896789),

('Sanju', 'Deshmukh', '1983-09-09', 'Male', 'Flat 5, Pimpri, Pune', 7210789016, 789078907890),

('Ravi', 'Gupta', '1988-12-03', 'Male', 'H.No. 32, Naroda, Ahmedabad', 6101890127, 890189018901),

('Mahesh', 'Sharma', '1979-06-28', 'Male', 'No. 15, Jhotwara, Jaipur', 5092901238, 901290129012),

('Sunil', 'Yadav', '1984-03-19', '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', '1980-08-17', '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', 6210789016, 889988998899),

('Kiran', 'Singh', '1987-09-21', 'Female', 'No. 51, Yeshwantpur, Bengaluru', 5101890127, 990099009900),

('Prakash', 'Rao', '1982-02-25', 'Male', 'Flat 22, Secunderabad, Hyderabad', 4092901238, 101010101010),

('Anil', 'Menon', '1985-11-13', 'Male', 'No. 67, Tambaram, Chennai', 3983012349, 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', 8329678906, 454545454545),

('Sunita', 'Sharma', '1986-03-25', 'Female', 'Flat 7, Kothrud, Pune', 7210789017, 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

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, 889988998899, '2 years imprisonment under IPC 354'),

(19, 1018, 999099009900, '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 2000'),

(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, 990099009900, '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);

-- 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),

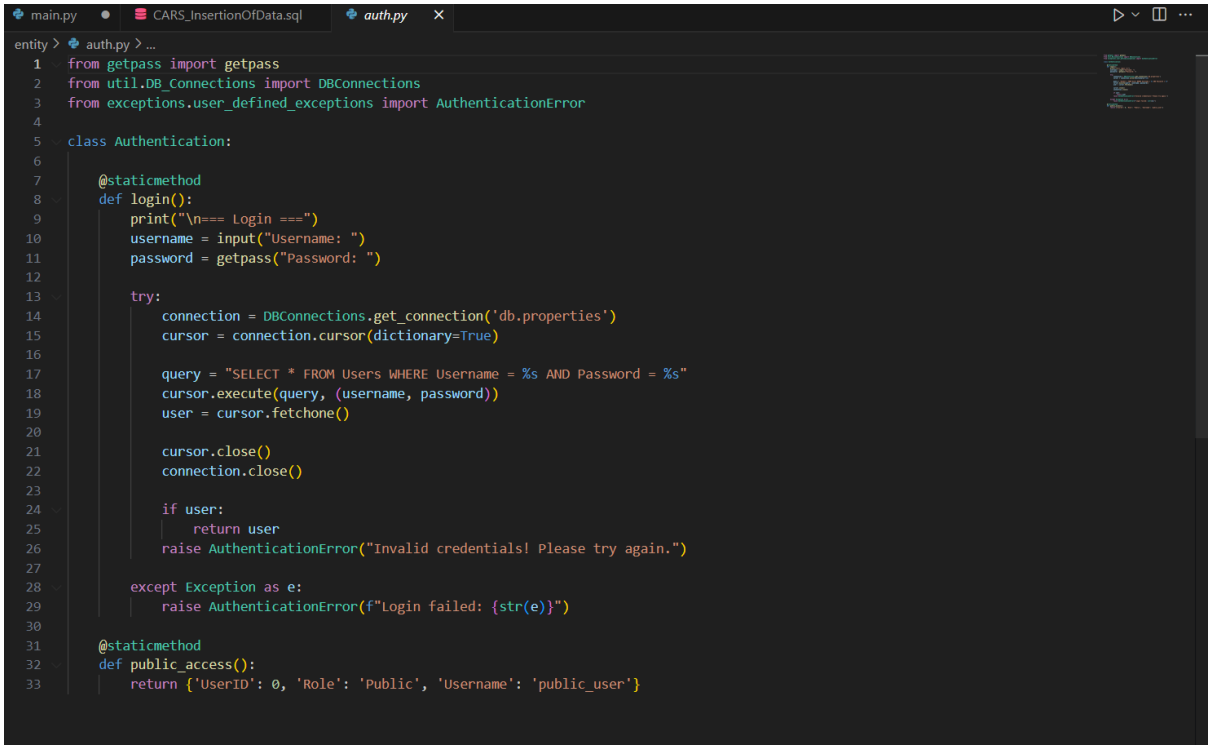
(1029, 30, 343434343434, 'Scammed seniors with fake lottery', 5),

(1030, 26, 454545454545, 'Accomplice in chain snatching', 1),

(1031, 27, 565656565656, 'Handled financial transactions for scam', 2);

Entity:

auth.py:



Criminals.py:

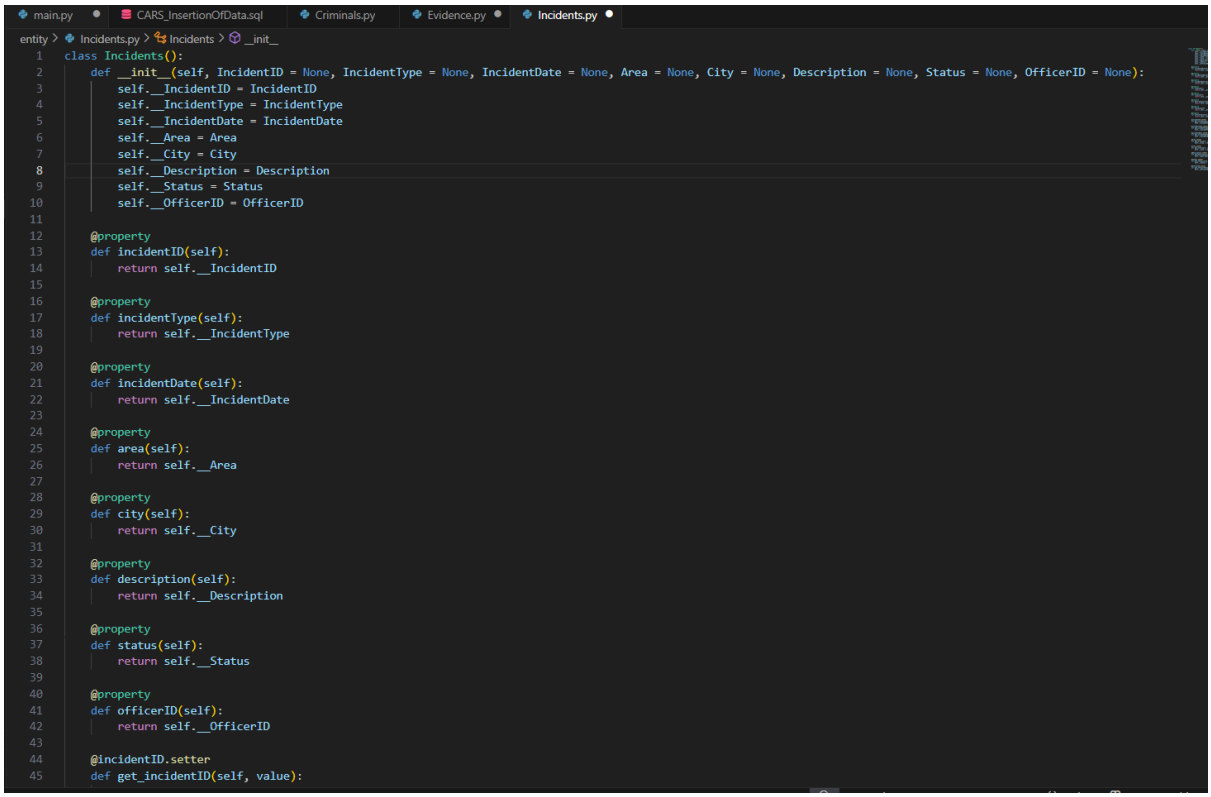
```python
class Criminals():
    def __init__(self, CriminalID = None, IncidentID = None, SuspectID = None, AadhaarNumber = None, PunishmentDetails = None):
        self.__CriminalID = CriminalID
        self.__IncidentID = IncidentID
        self.__SuspectID = SuspectID
        self.__AadhaarNumber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

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

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

Incidents.py:



```
main.py | CARS_InsertionOfData.sql | Criminals.py | Evidence.py | Incidents.py
entity > Incidents.py > Incidents > __init__
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):
```

```
    @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:

main.py ● | CARS_InsertionOfData.sql | Criminals.py | Evidence.py ● | Incidents.py ● | *LawEnforcementAgencies.py* ×

entity > LawEnforcementAgencies.py > LawEnforcementAgencies

```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):
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

```python

    @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}"


```

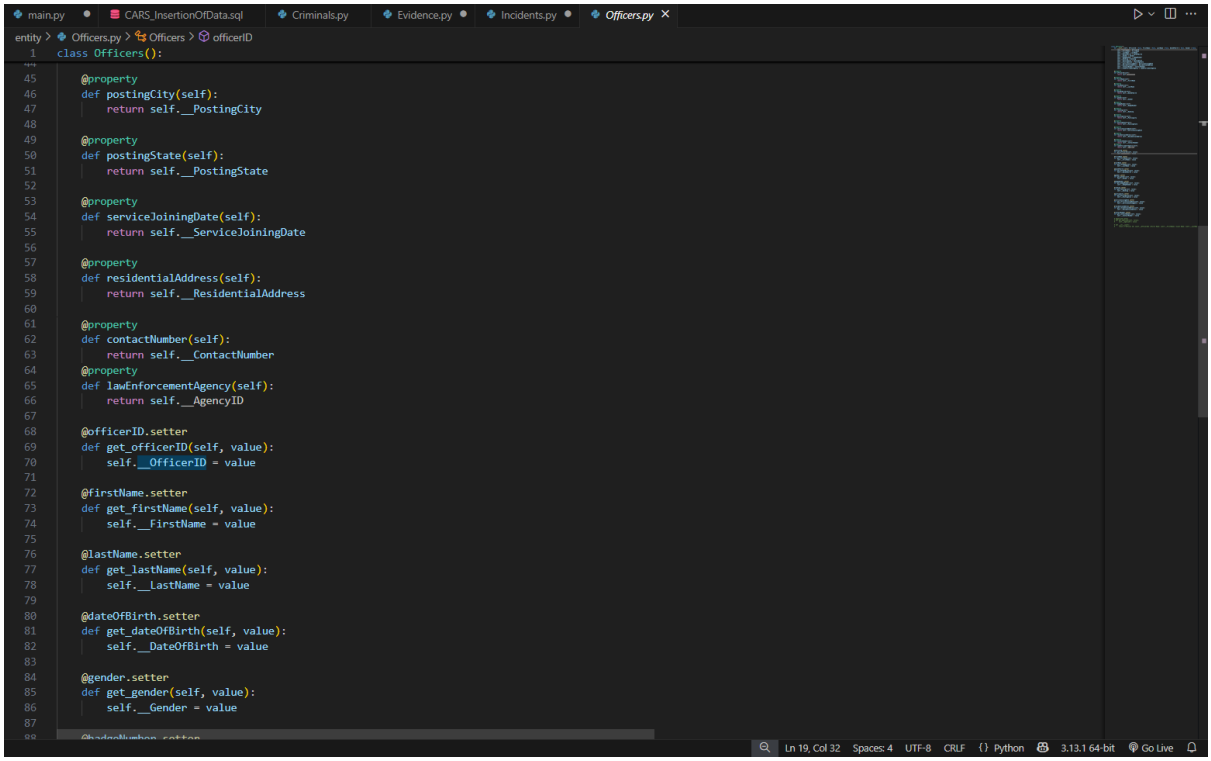
Officers.py:

main.py | CARS_InsertionOfData.sql | Criminals.py | Evidence.py ● | Incidents.py ● | *Officers.py* ×

entity > Officers.py > Officers > officerID

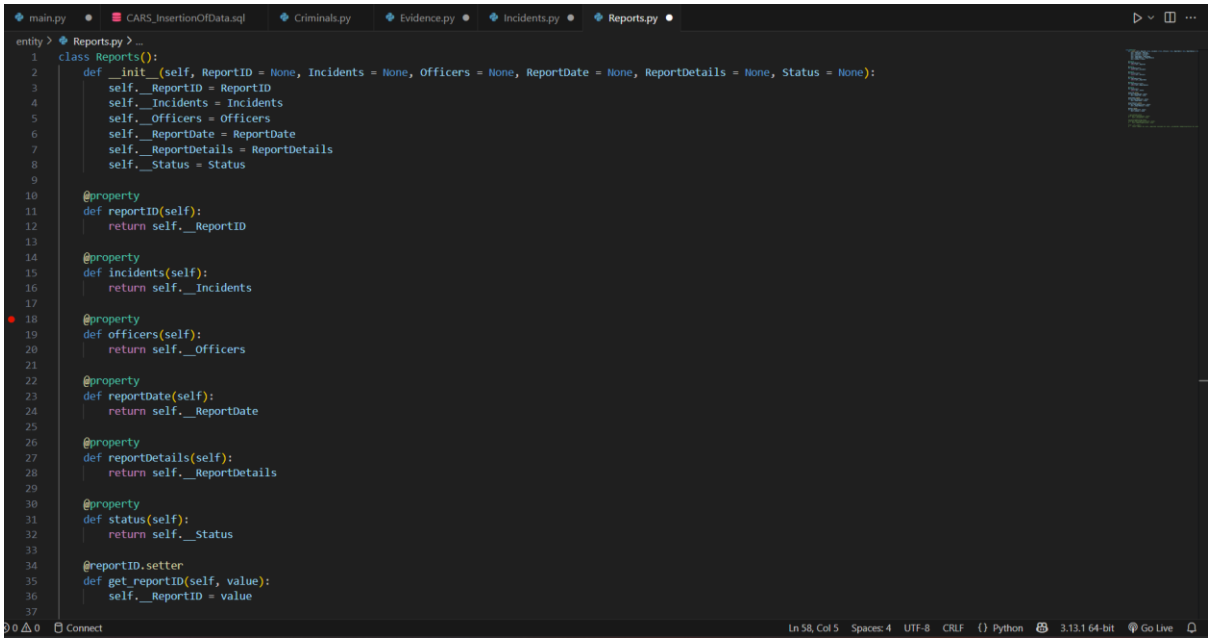
```python
class Officers():
    def __init__(self, OfficerID = None, FirstName = None, LastName = None, DateOfBirth = None, Gender = None, BadgeNumber = None, Ranking = None, PostingCity = None, Po
        self.__OfficerID = OfficerID
        self.__FirstName = FirstName
        self.__LastName = LastName
        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

    @property
```

Ln 19, Col 32 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

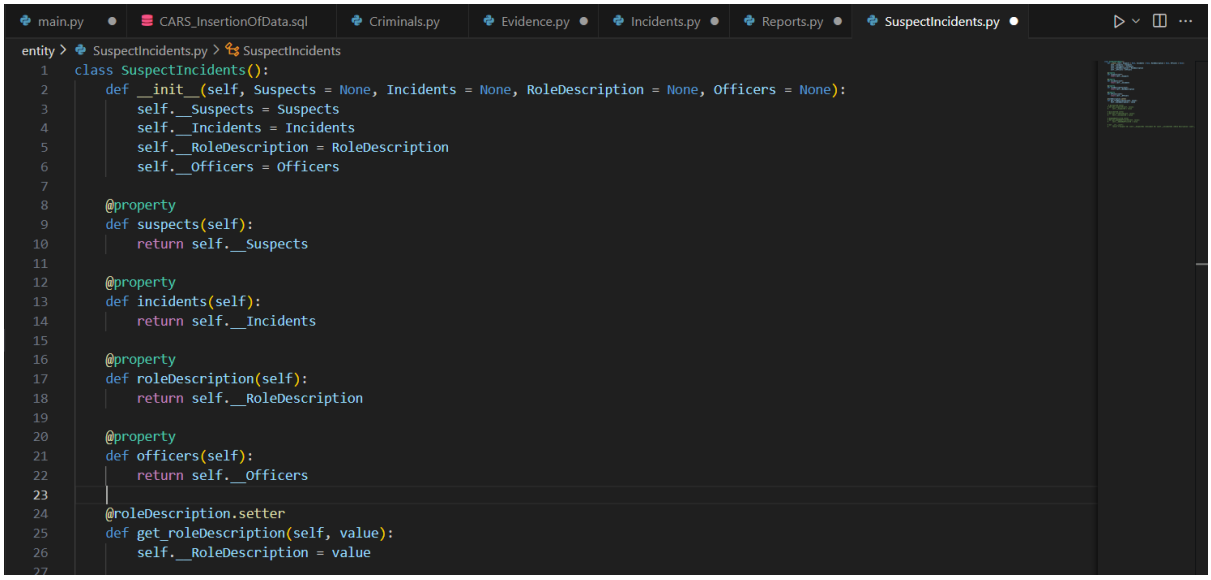

main.py
CARS_InsertionOfData.sql
Criminals.py
Evidence.py
Incidents.py
Officers.py
entity > Officers.py > Officers > officerID
class Officers():
@property
def postingCity(self):
return self.__PostingCity
@property
def postingState(self):
return self.__PostingState
@property
def serviceJoiningDate(self):
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
Ln 19, Col 32
Spaces: 4
UTF-8
CRLF
Python
3.13.1 64-bit
Go Live


@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

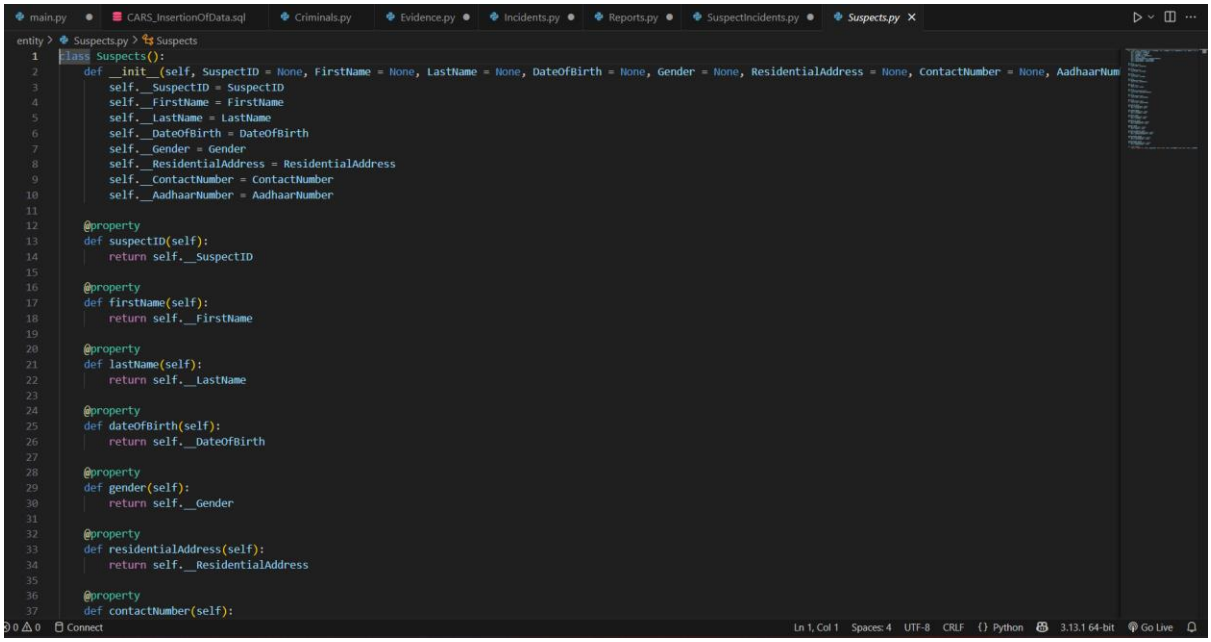
Reports.py:


main.py
CARS_InsertionOfData.sql
Criminals.py
Evidence.py
Incidents.py
Reports.py
entity > Reports.py > ...
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
Connect
Ln 58, Col 5
Spaces: 4
UTF-8
CRLF
Python
3.13.1 64-bit
Go Live

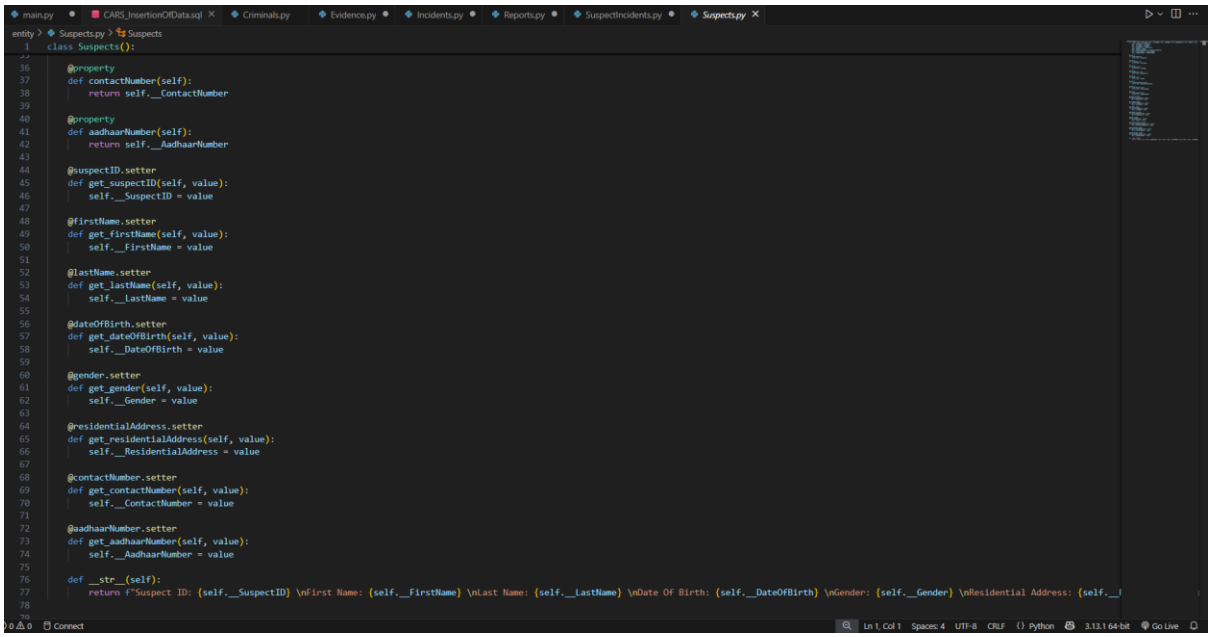

@reportDate.setter
def get_reportDate(self, value):
self.__ReportDate = value
@reportDetails.setter
def get_reportDetails(self, value):
self.__ReportDetails = value
@status.setter
def get_status(self, value):
self.__Status = value


main.py
CARS_InsertionOfData.sql
Criminals.py
Evidence.py
Incidents.py
Reports.py
SuspectIncidents.py
entity > SuspectIncidents.py > SuspectIncidents
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
@roleDescription.setter
def get_roleDescription(self, value):
self.__RoleDescription = value

Suspects.py:



```python
class Suspects():
    def __init__(self, SuspectID = None, FirstName = None, LastName = None, DateOfBirth = None, Gender = None, ResidentialAddress = None, ContactNumber = None, AadhaarNum
        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):
```



```python
class Suspects():

    @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.__FirstName} \nLast Name: {self.__LastName} \nDate Of Birth: {self.__DateOfBirth} \nGender: {self.__Gender} \nResidential Address: {self.__
```

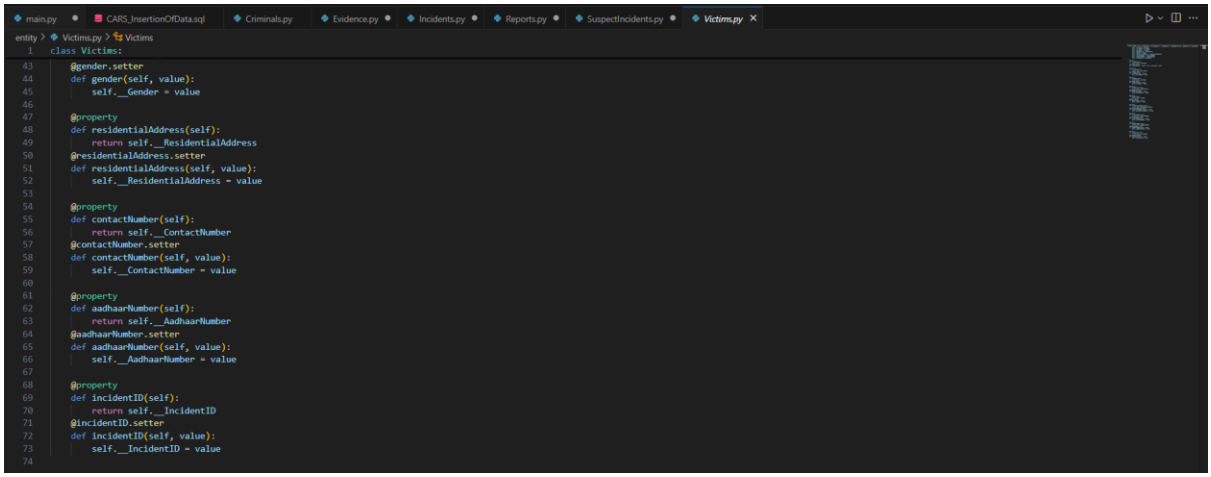
Users.py:

```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
```

```python
    @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:

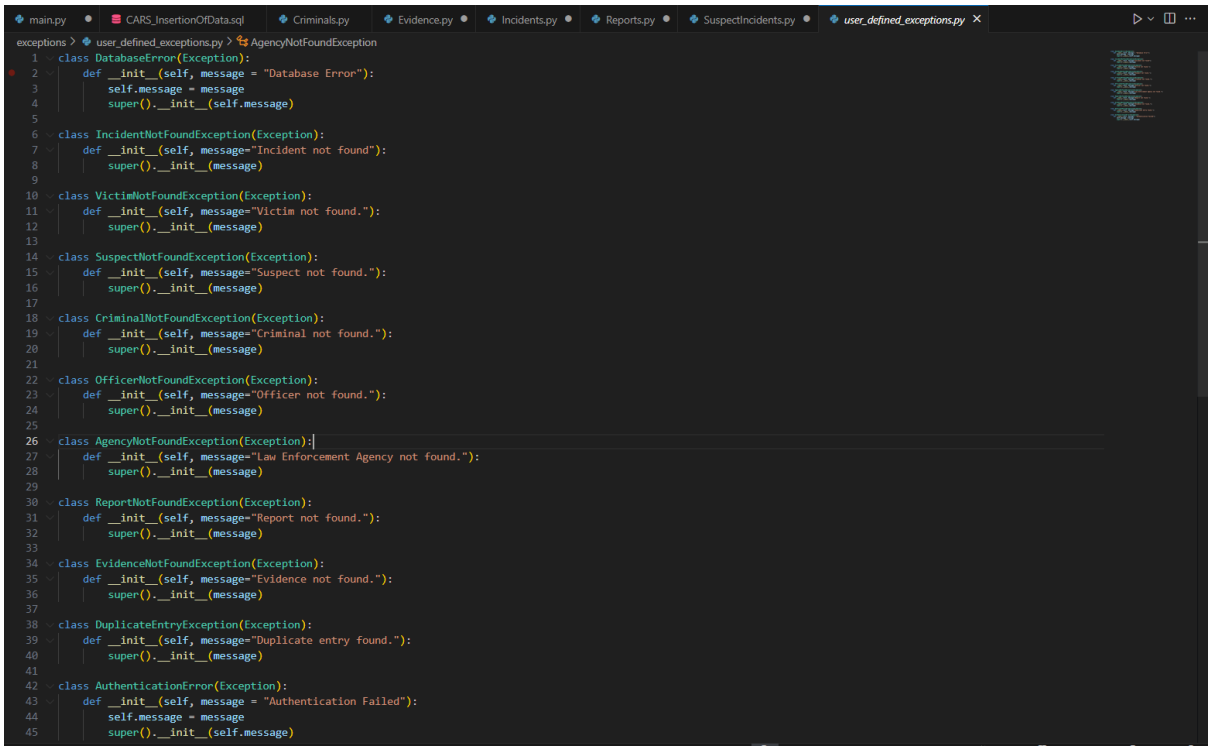
```python
class Victims:
    def __init__(self, VictimID=None, FirstName=None, LastName=None, DateOfBirth=None, Gender=None, ResidentialAddress=None, ContactNumber=None, AadhaarNumber=None, IncidentID=None):
        self.__VictimID = VictimID
        self.__FirstName = FirstName
        self.__LastName = 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
```



```
class Victims:
    @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

user_defined_exceptions.py



```
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

test_usertestcases.py

tests > test_usertestcases.py > ...

```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)
```

Ln 114, Col 33 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

tests > test_usertestcases.py > ...

```python
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)

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"
```

Ln 114, Col 33 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

```
    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")
```

Util

DB_Connections.py:

```
util > DB_Connections.py > ...
from util.DB_Properties_Util import DBProperties
from exceptions.user_defined_exceptions import DatabaseError
import mysql.connector

class DBConnections():
    @staticmethod
    def get_connection(property_file_name):
        try:
            conn_params = DBProperties.get_connection_string('db.properties')

            if not conn_params:
                raise DatabaseError("Database Error!")

            connection = mysql.connector.connect(
                host = conn_params['host'],
                database = conn_params['database'],
                user = conn_params['user'],
                password = conn_params['password'],
                port = int(conn_params.get('port', 3306)),
            )

            return connection

        except mysql.connector.Error as e:
            raise DatabaseError (f"Database interaction failed: {e.msg}") from e


```

DB_Properties_Util.py:

util > DB_Properties_Util.py > ...

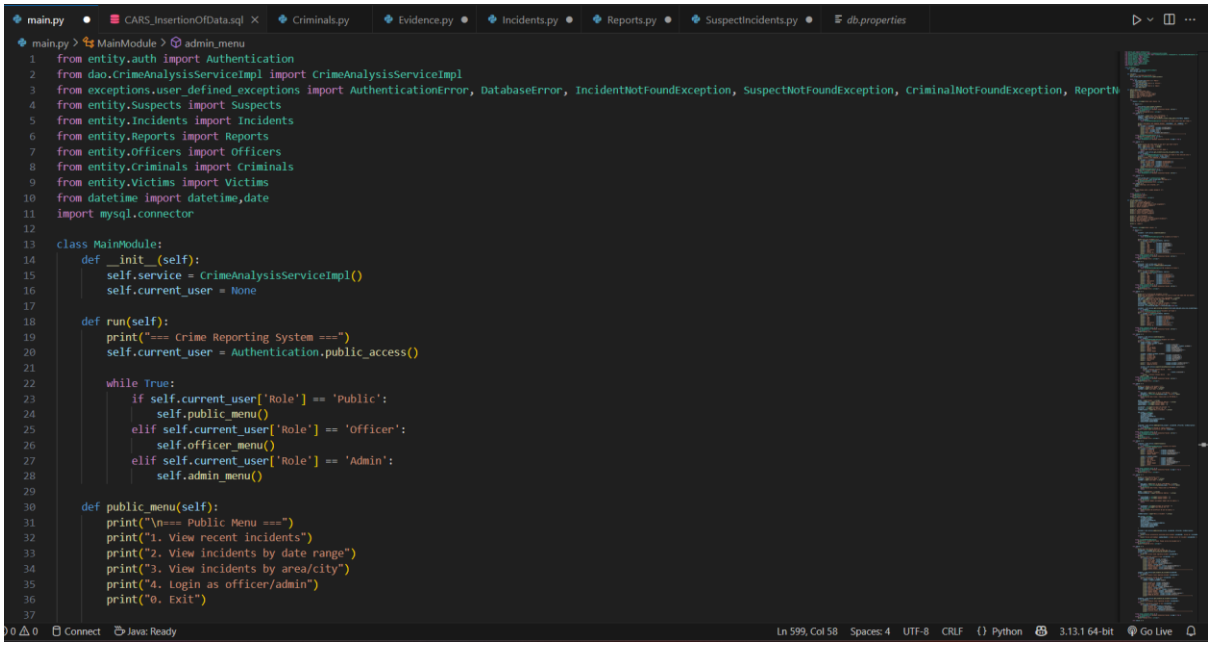
```python
import configparser
import os
from exceptions.user_defined_exceptions import DatabaseError
import mysql.connector

class DBProperties():
    @staticmethod
    def get_connection_string(property_file_name):
        try:
            if not os.path.exists(property_file_name):
                raise FileNotFoundError(f"'{property_file_name}' file not found")

            config = configparser.ConfigParser()
            config.read(property_file_name)

            if 'database' not in config:
                raise DatabaseError ("Database not found")

            return{
                'host' : config.get('database', 'host'),
                'database' : config.get('database', 'database'),
                'user' : config.get('database', 'user'),
                'password' : config.get('database', 'password'),
                'port' : config.get('database', 'port')
            }

        except mysql.connector.Error as e:
            raise DatabaseError(f"Database interaction failed: {e.msg}") from e
```

db.properties:

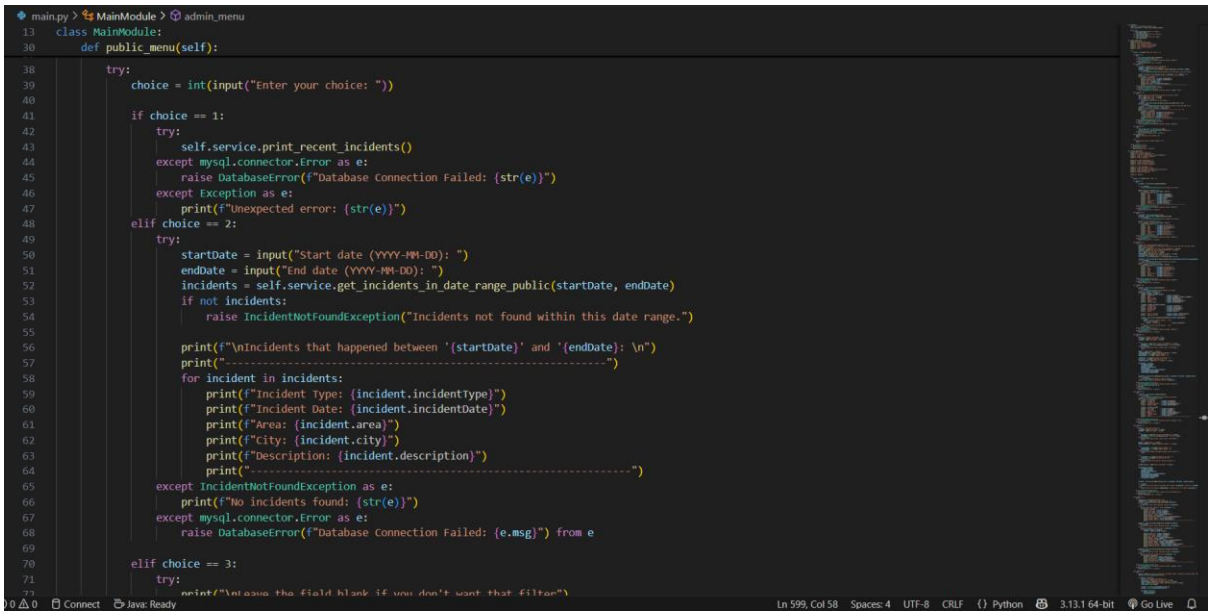
db.properties

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

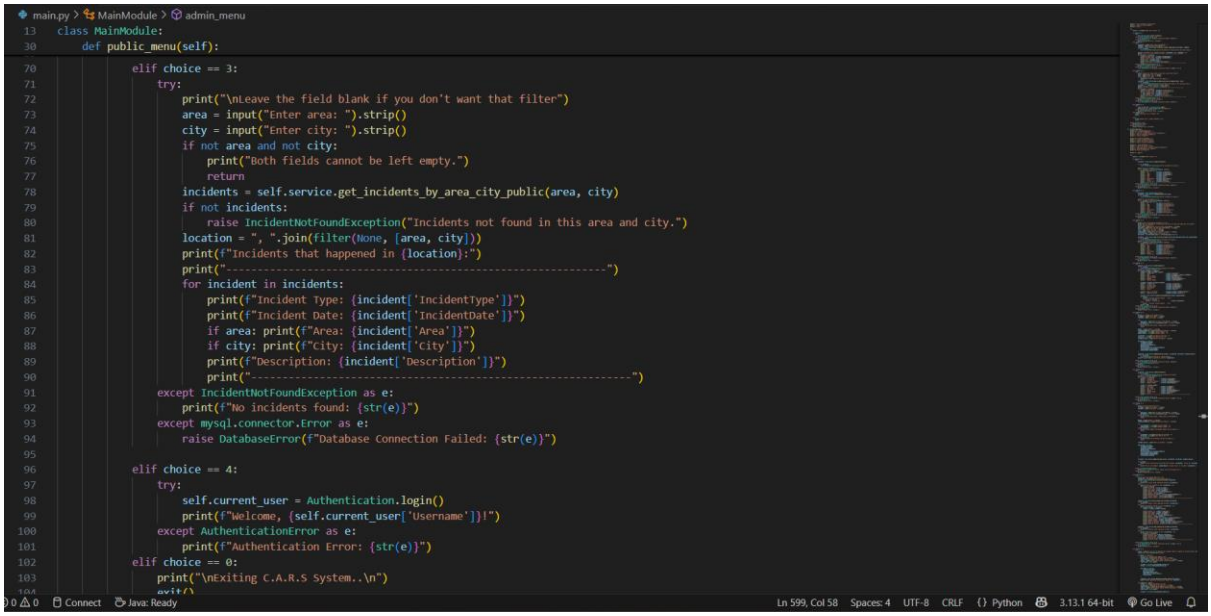
main.py:



```python
from entity.auth import Authentication
from dao.CrimeAnalysisServiceImpl import CrimeAnalysisServiceImpl
from exceptions.user_defined_exceptions import AuthenticationError, DatabaseError, IncidentNotFoundException, SuspectNotFoundException, CriminalNotFoundException, ReportN
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")
```



```python
class MainModule:
    def public_menu(self):
        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)}")
            elif choice == 2:
                try:
                    startDate = input("Start date (YYYY-MM-DD): ")
                    endDate = input("End date (YYYY-MM-DD): ")
                    incidents = self.service.get_incidents_in_date_range_public(startDate, endDate)
                    if not incidents:
                        raise IncidentNotFoundException("Incidents not found within this date range.")

                    print(f"\nIncidents that happened between '{startDate}' and '{endDate}: \n")
                    print("------------------------------------------------------------")
                    for incident in incidents:
                        print(f"Incident Type: {incident.incidentType}")
                        print(f"Incident Date: {incident.incidentDate}")
                        print(f"Area: {incident.area}")
                        print(f"City: {incident.city}")
                        print(f"Description: {incident.description}")
                        print("------------------------------------------------------------")
                except IncidentNotFoundException as e:
                    print(f"No incidents found: {str(e)}")
                except mysql.connector.Error as e:
                    raise DatabaseError(f"Database Connection Failed: {e.msg}") from e

            elif choice == 3:
                try:
                    print("\nplease the field blank if you don't want that filter")
```



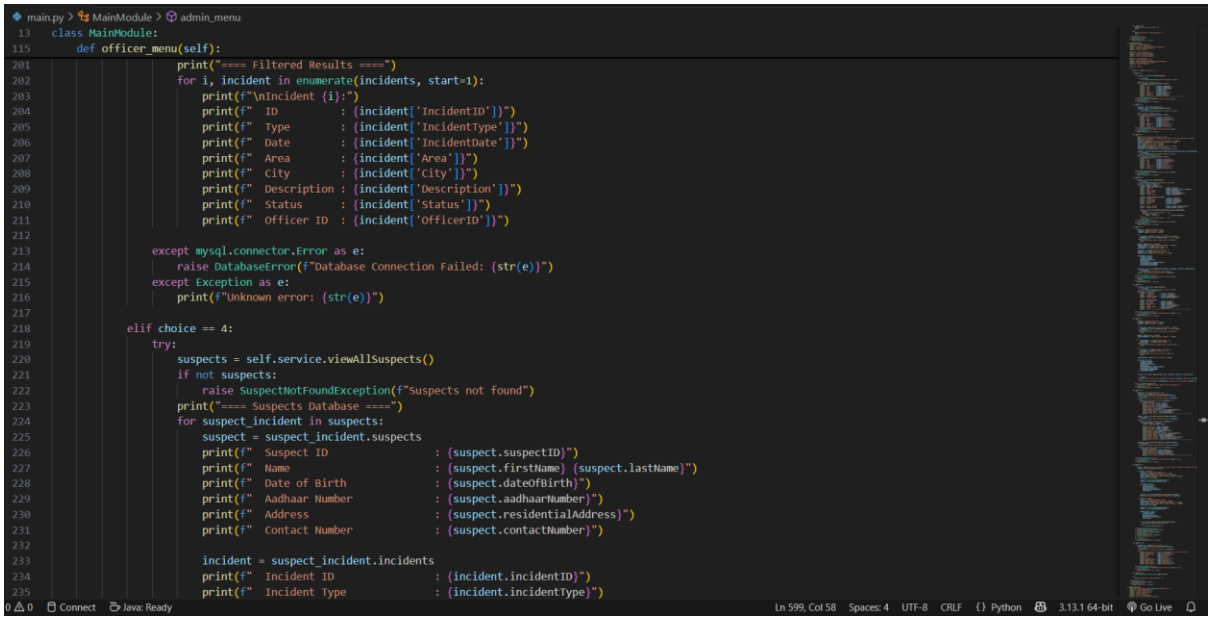
```
main.py > MainModule > admin_menu
13  class MainModule:
30      def public_menu(self):
70              elif choice == 3:
71                  try:
72                      print("\nLeave the field blank if you don't want that filter")
73                      area = input("Enter area: ").strip()
74                      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
96              elif choice == 4:
97                  try:
98                      self.current_user = Authentication.login()
99                      print(f"Welcome, {self.current_user['Username']}!")
100                 except AuthenticationError as e:
101                     print(f"Authentication Error: {str(e)}")
102             elif choice == 0:
103                 print("\nExiting C.A.R.S System..\n")
104                 exit()
```


```
main.py > MainModule > admin_menu
13  class MainModule:
30      def public_menu(self):
102             elif choice == 0:
103                 print("\nExiting C.A.R.S System..\n")
104                 exit()
105
106             else:
107                 print("Please enter a number between 0 -3")
108                 return
109
110         except ValueError as e:
111             print("Invalid Input")
112         except Exception as e:
113             print(f"Unknown Error: {str(e)}")
114
115     def officer_menu(self):
116         print(f"\n=== Officer Menu ===")
117         print("\n-- Incident Management --")
118         print("1. View all incident details with assignments")
119         print("2. View my assigned incidents")
120         print("3. Filter incidents")
121
122         print("\n-- Suspect Management --")
123         print("4. Access suspects database")
124         print("5. Create new suspect record")
125         print("6. Access criminals database")
126
127         print("\n-- Case Management --")
128         print("7. Create new victim record")
129         print("8. View case details (victims/suspects/evidence)")
130         print("9. Generate/Update case report")
131         print("10. View case report")
132
133         print("11. Logout")
134
135         try:
136             choice = int(input("Enter choice: "))
```

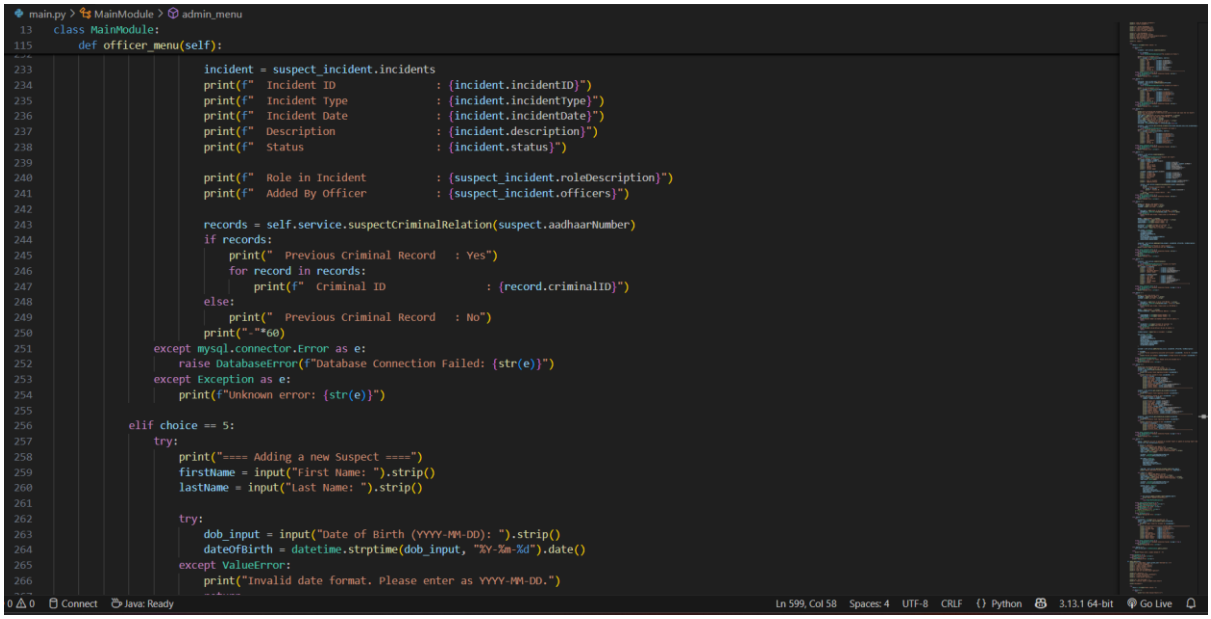
```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
135         try:
136             choice = int(input("Enter choice: "))
137
138             if choice == 1:
139                 try:
140                     incidents = self.service.viewAllIncidents()
141
142                     if not incidents:
143                         raise IncidentNotFoundException(f"No incidents are found.")
144
145                     print("\n=== All Incidents ===")
146                     for i, incident in enumerate(incidents, start=1):
147                         print(f"\nIncident {i}:")
148                         print(f"  ID          : {incident.incidentID}")
149                         print(f"  Type        : {incident.incidentType}")
150                         print(f"  Date        : {incident.incidentDate}")
151                         print(f"  Area        : {incident.area}")
152                         print(f"  City        : {incident.city}")
153                         print(f"  Description : {incident.description}")
154                         print(f"  Status      : {incident.status}")
155                         print(f"  Officer ID  : {incident.officerID}")
156                         print("-------------------------------------------------------------")
157                 except mysql.connector.Error as e:
158                     raise DatabaseError(f"Database Connection Failed: {str(e)}")
159                 except Exception as e:
160                     print(f"Unknown error: {str(e)}")
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 ====")
```



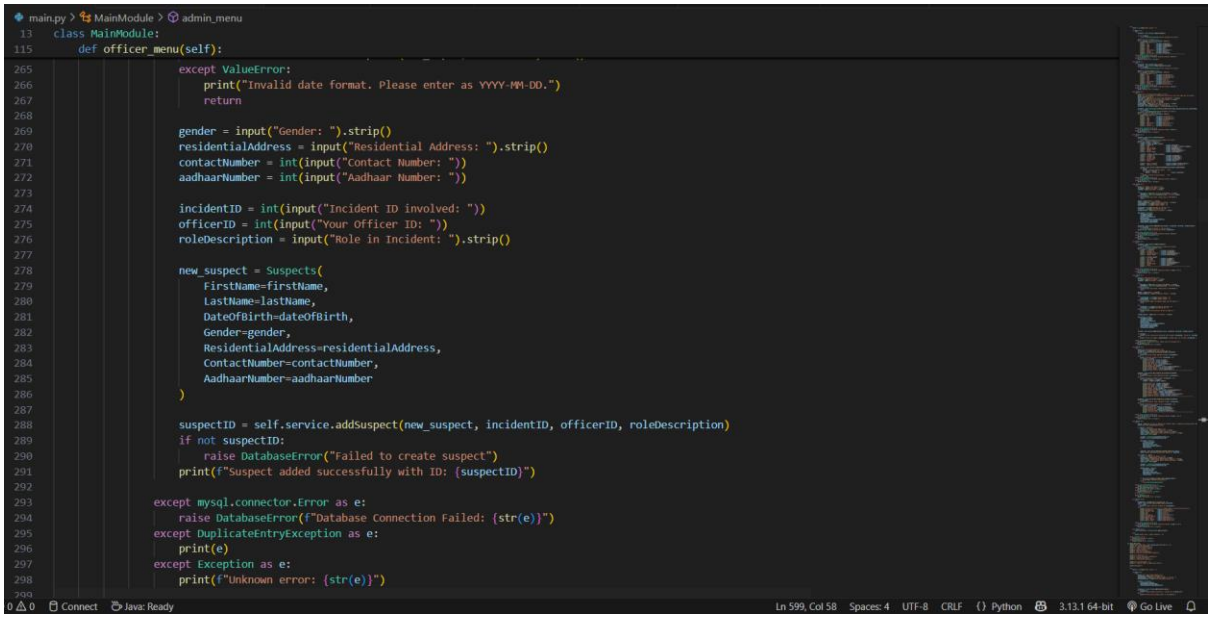
```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
169                     print("==== My Assignments ====")
170                     for i, incident in enumerate(incidents, start=1):
171                         print(f"\nIncident {i}:")
172                         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):
```



```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
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
218         elif choice == 4:
219             try:
220                 suspects = self.service.viewAllSuspects()
221                 if not suspects:
222                     raise SuspectNotFoundException(f"Suspects not found")
223                 print("==== Suspects Database ====")
224                 for suspect_incident in suspects:
225                     suspect = suspect_incident.suspects
226                     print(f"  Suspect ID                : {suspect.suspectID}")
227                     print(f"  Name                      : {suspect.firstName} {suspect.lastName}")
228                     print(f"  Date of Birth             : {suspect.dateOfBirth}")
229                     print(f"  Aadhaar Number            : {suspect.aadhaarNumber}")
230                     print(f"  Address                   : {suspect.residentialAddress}")
231                     print(f"  Contact Number            : {suspect.contactNumber}")
232
233                     incident = suspect_incident.incidents
234                     print(f"  Incident ID               : {incident.incidentID}")
235                     print(f"  Incident Type             : {incident.incidentType}")
```

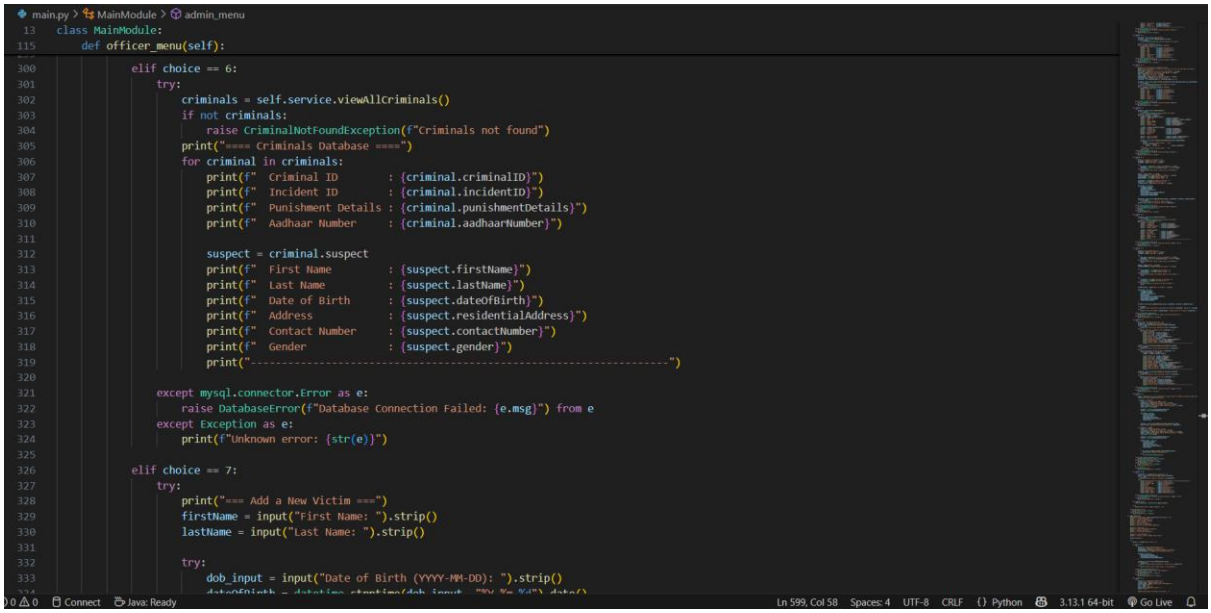



```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
233                     incident = suspect_incident.incidents
234                     print(f"  Incident ID               : {incident.incidentID}")
235                     print(f"  Incident Type             : {incident.incidentType}")
236                     print(f"  Incident Date             : {incident.incidentDate}")
237                     print(f"  Description               : {incident.description}")
238                     print(f"  Status                    : {incident.status}")
239
240                     print(f"  Role in Incident          : {suspect_incident.roleDescription}")
241                     print(f"  Added By Officer          : {suspect_incident.officers}")
242
243                     records = self.service.suspectCriminalRelation(suspect.aadhaarNumber)
244                     if records:
245                         print("  Previous Criminal Record   : Yes")
246                         for record in records:
247                             print(f"  Criminal ID               : {record.criminalID}")
248                     else:
249                         print("  Previous Criminal Record   : No")
250                     print("-"*60)
251             except mysql.connector.Error as e:
252                 raise DatabaseError(f"Database Connection Failed: {str(e)}")
253             except Exception as e:
254                 print(f"Unknown error: {str(e)}")
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.")
```



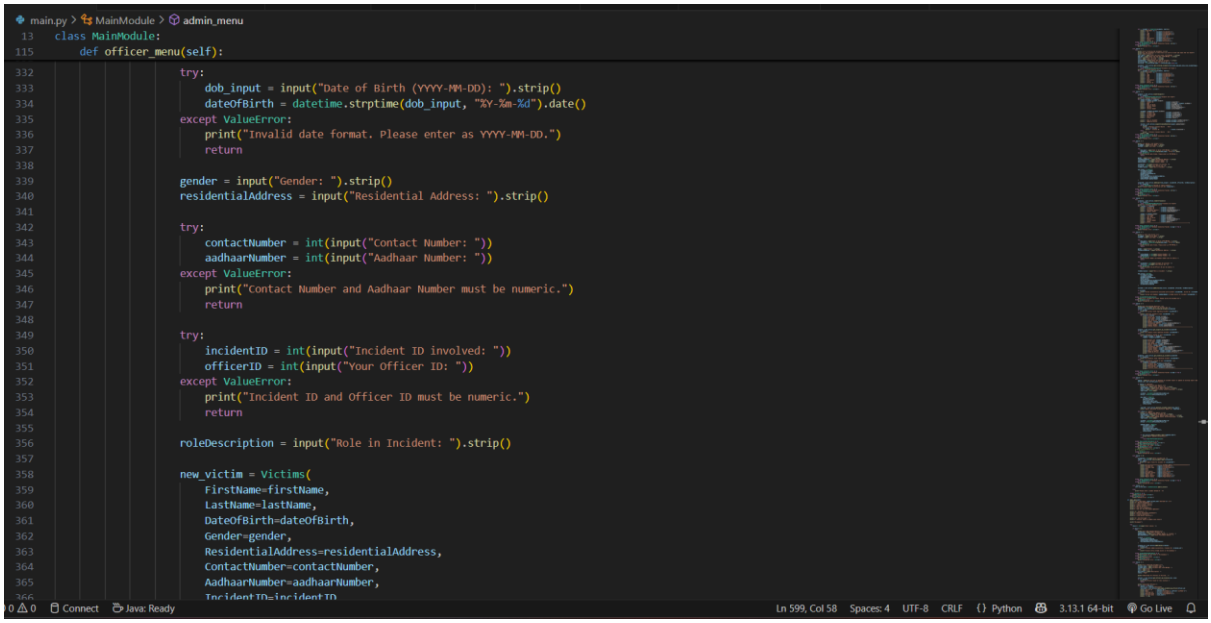
```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
265                 except ValueError:
266                     print("Invalid date format. Please enter as YYYY-MM-DD.")
267                     return
268
269                 gender = input("Gender: ").strip()
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)}")
```

0 △ 0 Connect Java: Ready Ln 599, Col 58 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

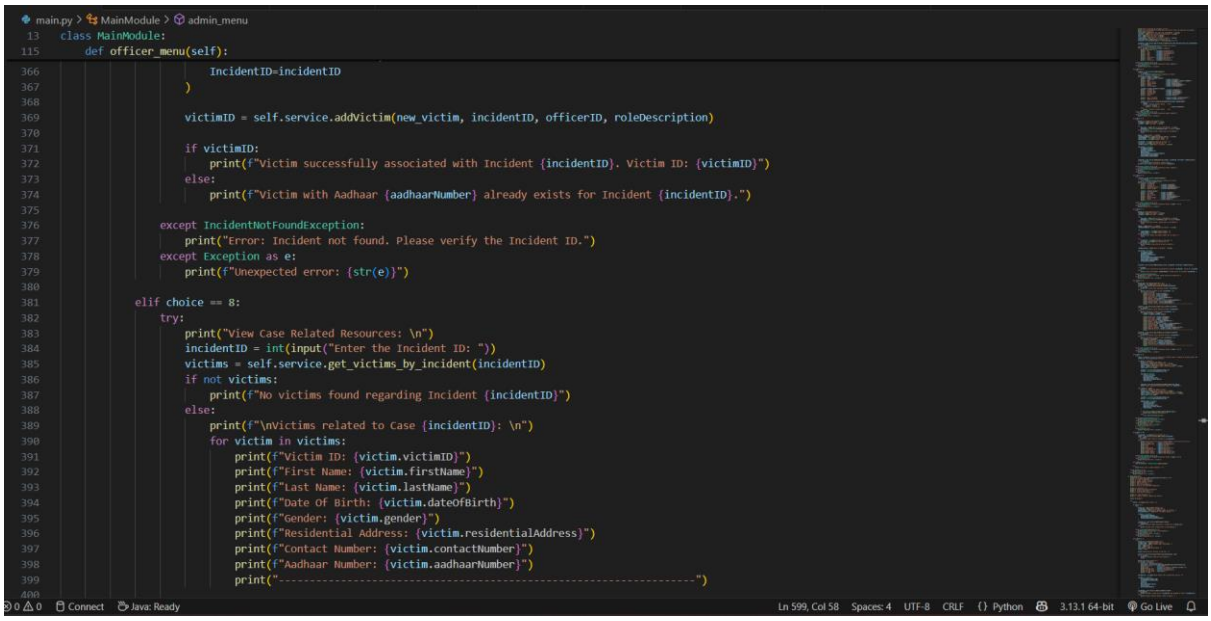


```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
300             elif choice == 6:
301                 try:
302                     criminals = self.service.viewAllCriminals()
303                     if not criminals:
304                         raise CriminalNotFoundException(f"Criminals not found")
305                     print("==== Criminals Database ====")
306                     for criminal in criminals:
307                         print(f"  Criminal ID        : {criminal.criminalID}")
308                         print(f"  Incident ID        : {criminal.incidentID}")
309                         print(f"  Punishment Details : {criminal.punishmentDetails}")
310                         print(f"  Aadhaar Number     : {criminal.aadhaarNumber}")
311
312                         suspect = criminal.suspect
313                         print(f"  First Name         : {suspect.firstName}")
314                         print(f"  Last Name          : {suspect.lastName}")
315                         print(f"  Date of Birth      : {suspect.dateOfBirth}")
316                         print(f"  Address            : {suspect.residentialAddress}")
317                         print(f"  Contact Number     : {suspect.contactNumber}")
318                         print(f"  Gender             : {suspect.gender}")
319                         print("-----------------------------------------------------------------")
320
321                 except mysql.connector.Error as e:
322                     raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
323                 except Exception as e:
324                     print(f"Unknown error: {str(e)}")
325
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()
```

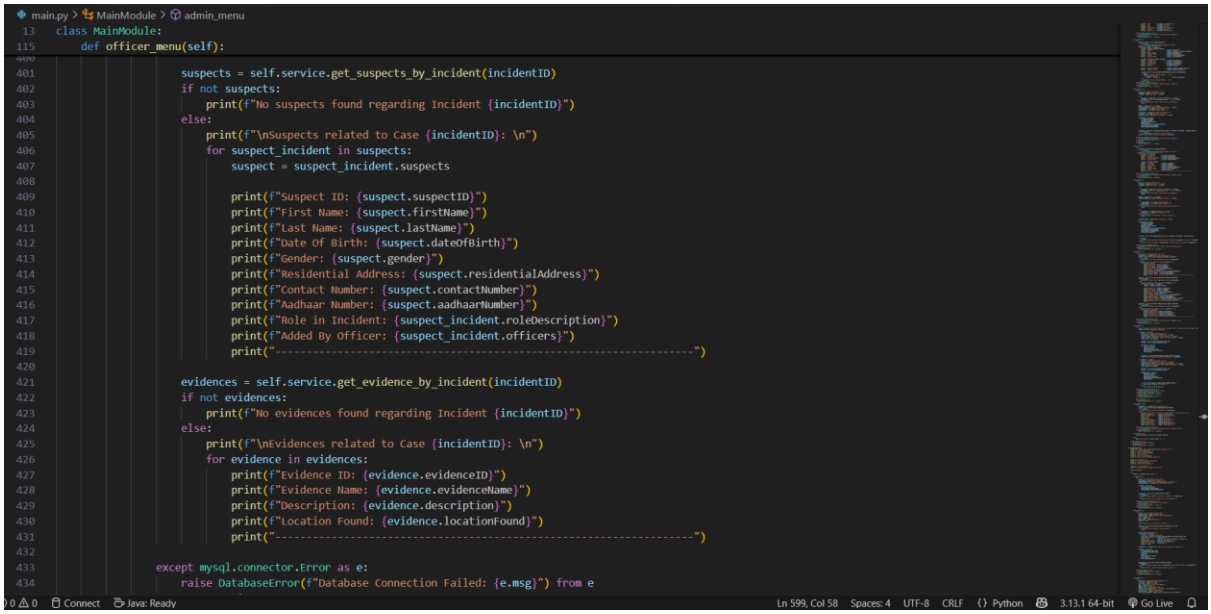
0 △ 0 Connect Java: Ready Ln 599, Col 58 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live



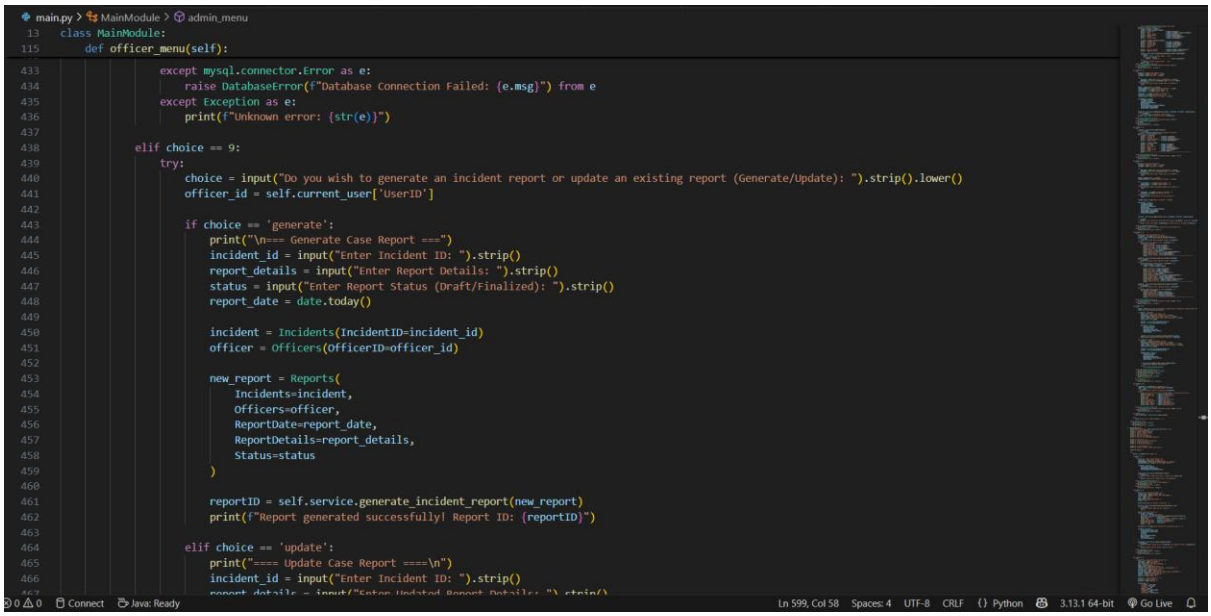
```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
332                 try:
333                     dob_input = input("Date of Birth (YYYY-MM-DD): ").strip()
334                     dateOfBirth = datetime.strptime(dob_input, "%Y-%m-%d").date()
335                 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
356                 roleDescription = input("Role in Incident: ").strip()
357
358                 new_victim = Victims(
359                     FirstName=firstName,
360                     LastName=lastName,
361                     DateOfBirth=dateOfBirth,
362                     Gender=gender,
363                     ResidentialAddress=residentialAddress,
364                     ContactNumber=contactNumber,
365                     AadhaarNumber=aadhaarNumber,
366                     IncidentID=incidentID
```

```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
366                     IncidentID=incidentID
367                 )
368
369                 victimID = self.service.addVictim(new_victim, incidentID, officerID, roleDescription)
370
371                 if victimID:
372                     print(f"Victim successfully associated with Incident {incidentID}. Victim ID: {victimID}")
373                 else:
374                     print(f"Victim with Aadhaar {aadhaarNumber} already exists for Incident {incidentID}.")
375
376             except IncidentNotFoundException:
377                 print("Error: Incident not found. Please verify the Incident ID.")
378             except Exception as e:
379                 print(f"Unexpected error: {str(e)}")
380
381         elif choice == 8:
382             try:
383                 print("View Case Related Resources: \n")
384                 incidentID = int(input("Enter the Incident ID: "))
385                 victims = self.service.get_victims_by_incident(incidentID)
386                 if not victims:
387                     print(f"No victims found regarding Incident {incidentID}")
388                 else:
389                     print(f"\nVictims related to Case {incidentID}: \n")
390                     for victim in victims:
391                         print(f"Victim ID: {victim.victimID}")
392                         print(f"First Name: {victim.firstName}")
393                         print(f"Last Name: {victim.lastName}")
394                         print(f"Date Of Birth: {victim.dateOfBirth}")
395                         print(f"Gender: {victim.gender}")
396                         print(f"Residential Address: {victim.residentialAddress}")
397                         print(f"Contact Number: {victim.contactNumber}")
398                         print(f"Aadhaar Number: {victim.aadhaarNumber}")
399                         print("-----------------------------------------------------------------")
400
```

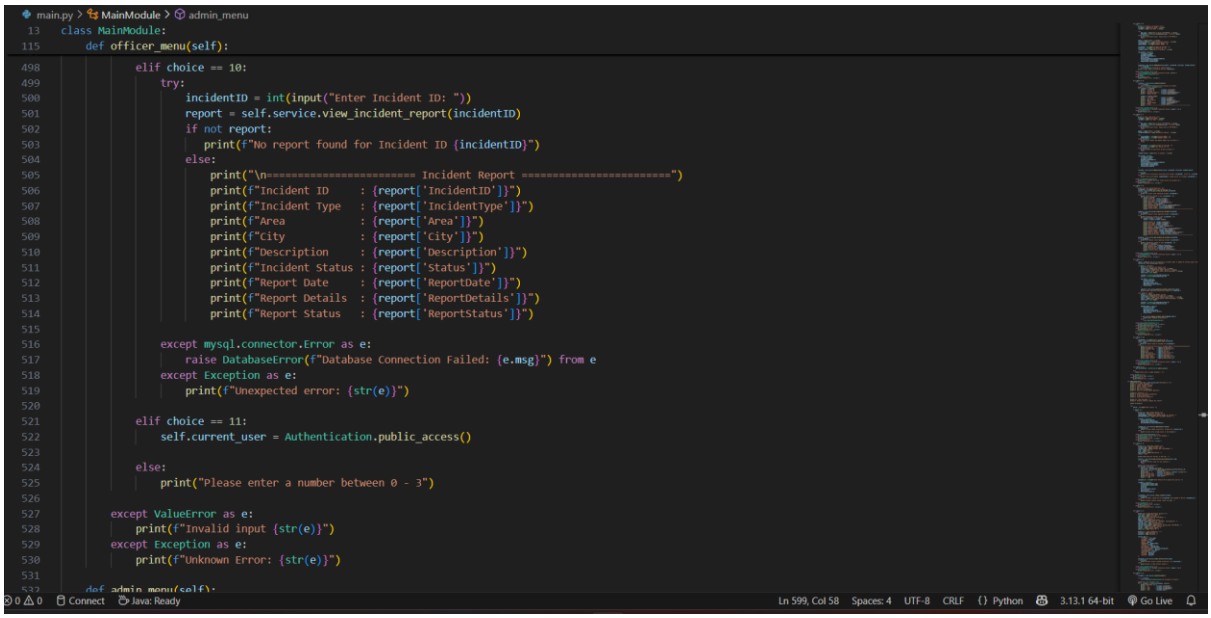


```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
401                     suspects = self.service.get_suspects_by_incident(incidentID)
402                     if not suspects:
403                         print(f"No suspects found regarding Incident {incidentID}")
404                     else:
405                         print(f"\nSuspects related to Case {incidentID}: \n")
406                         for suspect_incident in suspects:
407                             suspect = suspect_incident.suspects
408
409                             print(f"Suspect ID: {suspect.suspectID}")
410                             print(f"First Name: {suspect.firstName}")
411                             print(f"Last Name: {suspect.lastName}")
412                             print(f"Date Of Birth: {suspect.dateOfBirth}")
413                             print(f"Gender: {suspect.gender}")
414                             print(f"Residential Address: {suspect.residentialAddress}")
415                             print(f"Contact Number: {suspect.contactNumber}")
416                             print(f"Aadhaar Number: {suspect.aadhaarNumber}")
417                             print(f"Role in Incident: {suspect_incident.roleDescription}")
418                             print(f"Added By Officer: {suspect_incident.officers}")
419                             print("-----------------------------------------------------------------")
420
421                     evidences = self.service.get_evidence_by_incident(incidentID)
422                     if not evidences:
423                         print(f"No evidences found regarding Incident {incidentID}")
424                     else:
425                         print(f"\nEvidences related to Case {incidentID}: \n")
426                         for evidence in evidences:
427                             print(f"Evidence ID: {evidence.evidenceID}")
428                             print(f"Evidence Name: {evidence.evidenceName}")
429                             print(f"Description: {evidence.description}")
430                             print(f"Location Found: {evidence.locationFound}")
431                             print("-----------------------------------------------------------------")
432
433                 except mysql.connector.Error as e:
434                     raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
```

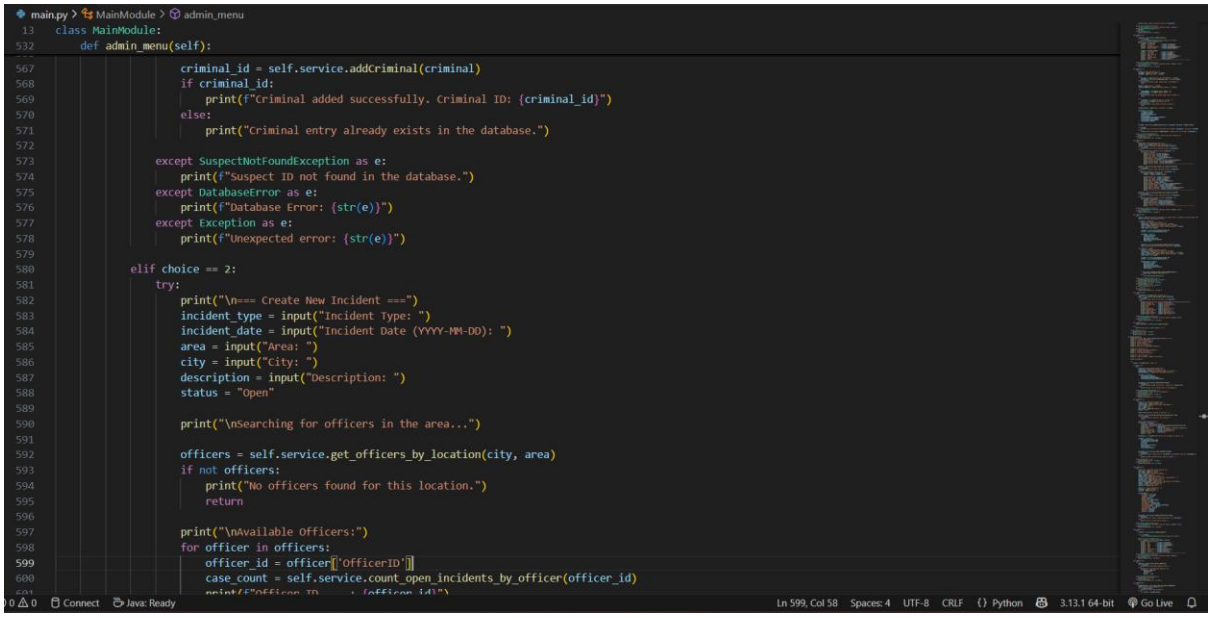
```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
433                 except mysql.connector.Error as e:
434                     raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
435                 except Exception as e:
436                     print(f"Unknown error: {str(e)}")
437
438             elif choice == 9:
439                 try:
440                     choice = input("Do you wish to generate an incident report or update an existing report (Generate/Update): ").strip().lower()
441                     officer_id = self.current_user['UserID']
442
443                     if choice == 'generate':
444                         print("\n=== Generate Case Report ===")
445                         incident_id = input("Enter Incident ID: ").strip()
446                         report_details = input("Enter Report Details: ").strip()
447                         status = input("Enter Report Status (Draft/Finalized): ").strip()
448                         report_date = date.today()
449
450                         incident = Incidents(IncidentID=incident_id)
451                         officer = Officers(OfficerID=officer_id)
452
453                         new_report = Reports(
454                             Incidents=incident,
455                             Officers=officer,
456                             ReportDate=report_date,
457                             ReportDetails=report_details,
458                             Status=status
459                         )
460
461                         reportID = self.service.generate_incident_report(new_report)
462                         print(f"Report generated successfully! Report ID: {reportID}")
463
464                     elif choice == 'update':
465                         print("==== Update Case Report ====\n")
466                         incident_id = input("Enter Incident ID: ").strip()
467                         report_details = input("Enter Updated Report Details: ").strip()
```

```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
464                 elif choice == 'update':
465                     print("==== Update Case Report ====\n")
466                     incident_id = input("Enter Incident ID: ").strip()
467                     report_details = input("Enter Updated Report Details: ").strip()
468                     status = input("Enter Updated Status (Draft/Finalized): ").strip()
469                     report_date = date.today()
470
471                     incident = Incidents(IncidentID=incident_id)
472                     officer = Officers(OfficerID=officer_id)
473
474                     updated_report = Reports(
475                         Incidents=incident,
476                         Officers=officer,
477                         ReportDate=report_date,
478                         ReportDetails=report_details,
479                         Status=status
480                     )
481
482                     if self.service.update_incident_report(updated_report):
483                         print("Report updated successfully!")
484                     else:
485                         raise ReportNotFoundException()
486
487                 except DuplicateEntryException as e:
488                     print(f"Duplicate Entry found. {str(e)}")
489                 except ReportNotFoundException as e:
490                     print(f"Report not found. {str(e)}")
491                 except DatabaseError as e:
492                     print(f"Database Error: {str(e)}")
493                 # except PermissionError as e:
494                 #     print("")
495                 except Exception as e:
496                     print(f"Unexpected Error: {str(e)}")
497
498             elif choice == 10:
```

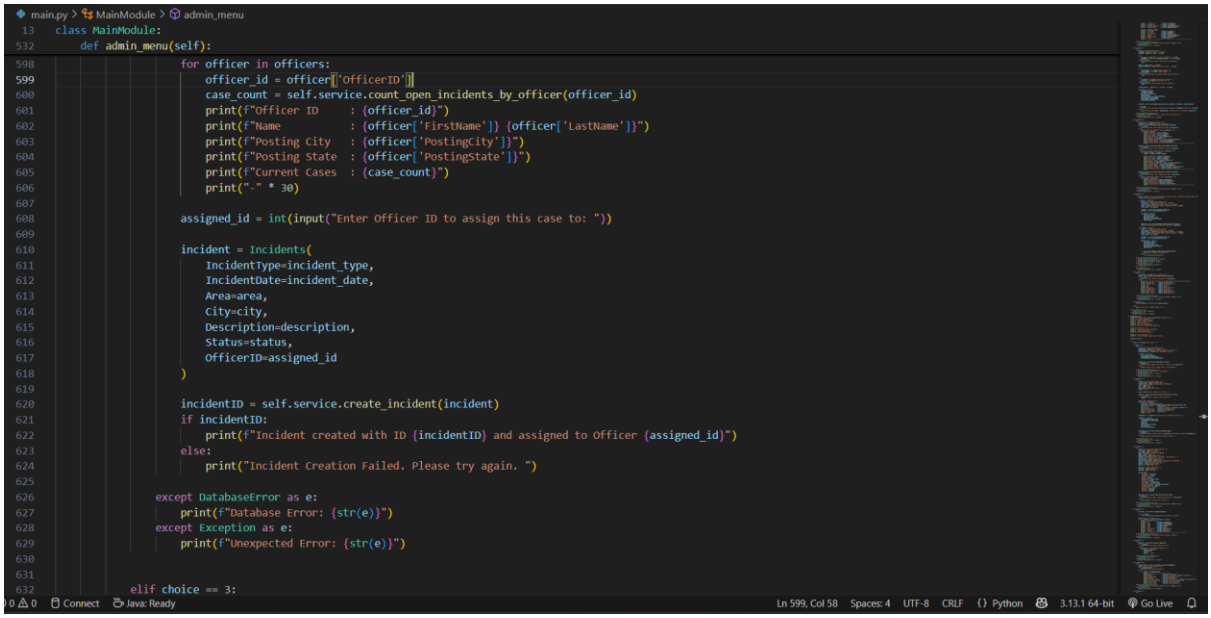


```
main.py > MainModule > admin_menu
13  class MainModule:
115     def officer_menu(self):
498             elif choice == 10:
499                 try:
500                     incidentID = int(input("Enter Incident ID: "))
501                     report = self.service.view_incident_report(incidentID)
502                     if not report:
503                         print(f"No report found for Incident ID {incidentID}")
504                     else:
505                         print("\n====================== Incident Report ======================")
506                         print(f"Incident ID     : {report['IncidentID']}")
507                         print(f"Incident Type   : {report['IncidentType']}")
508                         print(f"Area            : {report['Area']}")
509                         print(f"City            : {report['City']}")
510                         print(f"Description     : {report['Description']}")
511                         print(f"Incident Status : {report['Status']}")
512                         print(f"Report Date     : {report['ReportDate']}")
513                         print(f"Report Details  : {report['ReportDetails']}")
514                         print(f"Report Status   : {report['ReportStatus']}")
515
516                 except mysql.connector.Error as e:
517                     raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
518                 except Exception as e:
519                     print(f"Unexpected error: {str(e)}")
520
521             elif choice == 11:
522                 self.current_user = Authentication.public_access()
523
524             else:
525                 print("Please enter a number between 0 - 3")
526
527         except ValueError as e:
528             print(f"Invalid input {str(e)}")
529         except Exception as e:
530             print(f"Unknown Error: {str(e)}")
531
532     def admin_menu(self):
```

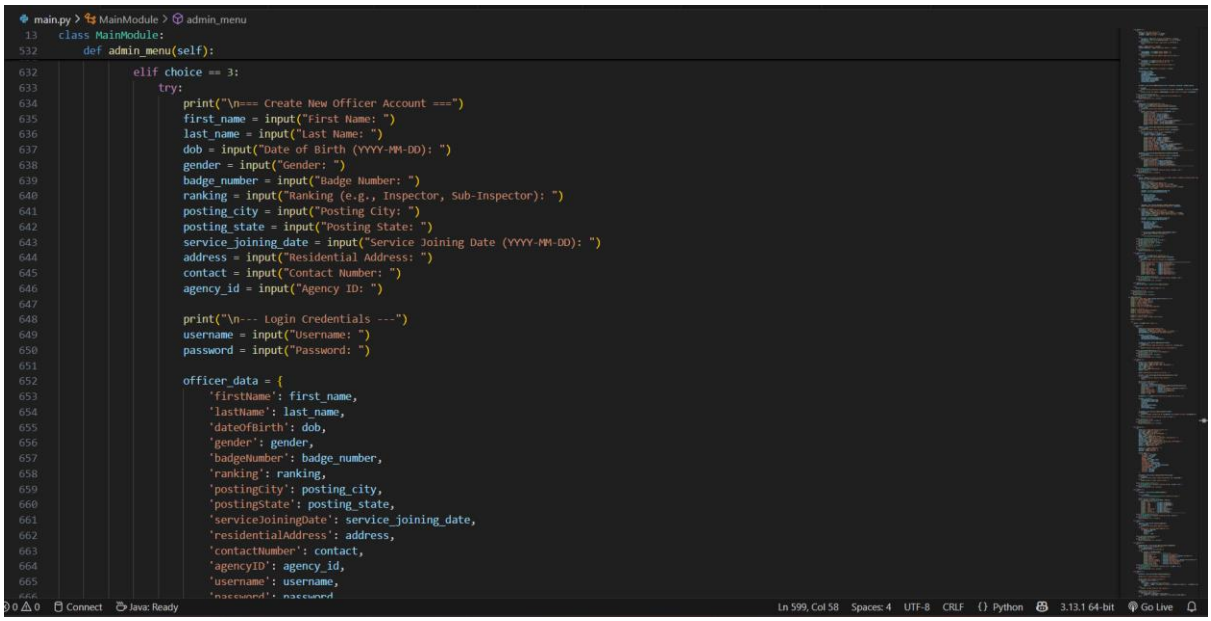
```
main.py > MainModule > admin_menu
13  class MainModule:
532     def admin_menu(self):
533         print(f"\n=== Admin Menu ({self.current_user['Username']}) ===")
534         print("\n-- Records Management --")
535         print("1. Create criminal record")
536         print("2. Create incident record")
537         print("3. Add new officer")
538         print("4. View case assignments")
539         print("5. View all law enforcement agencies")
540
541         print("\n-- Analytics --")
542         print("6. Officer performance dashboard")
543         print("7. Criminal analytics")
544         print("8. Crime pattern analytics")
545
546         print("\n-- Case Oversight --")
547         print("9. Finalize reports & update case status")
548
549         print("10. Logout")
550
551         try:
552             choice = int(input("Enter choice: "))
553
554             if choice == 1:
555                 try:
556                     print("\n=== Add Criminal Record ===")
557                     incidentID = input("Enter Incident ID: ")
558                     aadhaarNumber = input("Enter Aadhaar Number of criminal: ")
559                     punishmentDetails = input("Enter Punishment Details: ")
560
561                     criminal = Criminals(
562                         IncidentID=incidentID,
563                         AadhaarNumber=aadhaarNumber,
564                         PunishmentDetails=punishmentDetails
565                     )
566
567                     criminal_id = self.service.addCriminal(criminal)
```

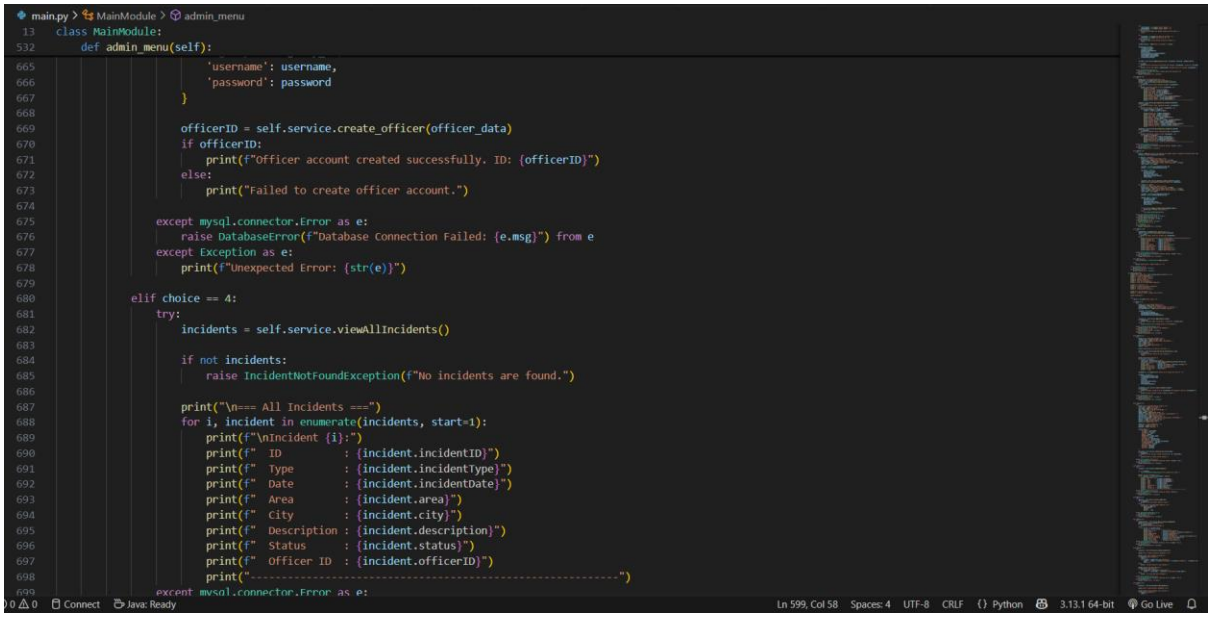
```
main.py > MainModule > admin_menu
13  class MainModule:
532     def admin_menu(self):
567                     criminal_id = self.service.addCriminal(criminal)
568                     if criminal_id:
569                         print(f"Criminal added successfully. Criminal ID: {criminal_id}")
570                     else:
571                         print("Criminal entry already exists in the database.")
572
573                 except SuspectNotFoundException as e:
574                     print(f"Suspect ID not found in the database.")
575                 except DatabaseError as e:
576                     print(f"Database Error: {str(e)}")
577                 except Exception as e:
578                     print(f"Unexpected error: {str(e)}")
579
580             elif choice == 2:
581                 try:
582                     print("\n=== Create New Incident ===")
583                     incident_type = input("Incident Type: ")
584                     incident_date = input("Incident Date (YYYY-MM-DD): ")
585                     area = input("Area: ")
586                     city = input("City: ")
587                     description = input("Description: ")
588                     status = "Open"
589
590                     print("\nSearching for officers in the area...")
591
592                     officers = self.service.get_officers_by_location(city, area)
593                     if not officers:
594                         print("No officers found for this location.")
595                         return
596
597                     print("\nAvailable Officers:")
598                     for officer in officers:
599                         officer_id = officer['OfficerID']
600                         case_count = self.service.count_open_incidents_by_officer(officer_id)
```



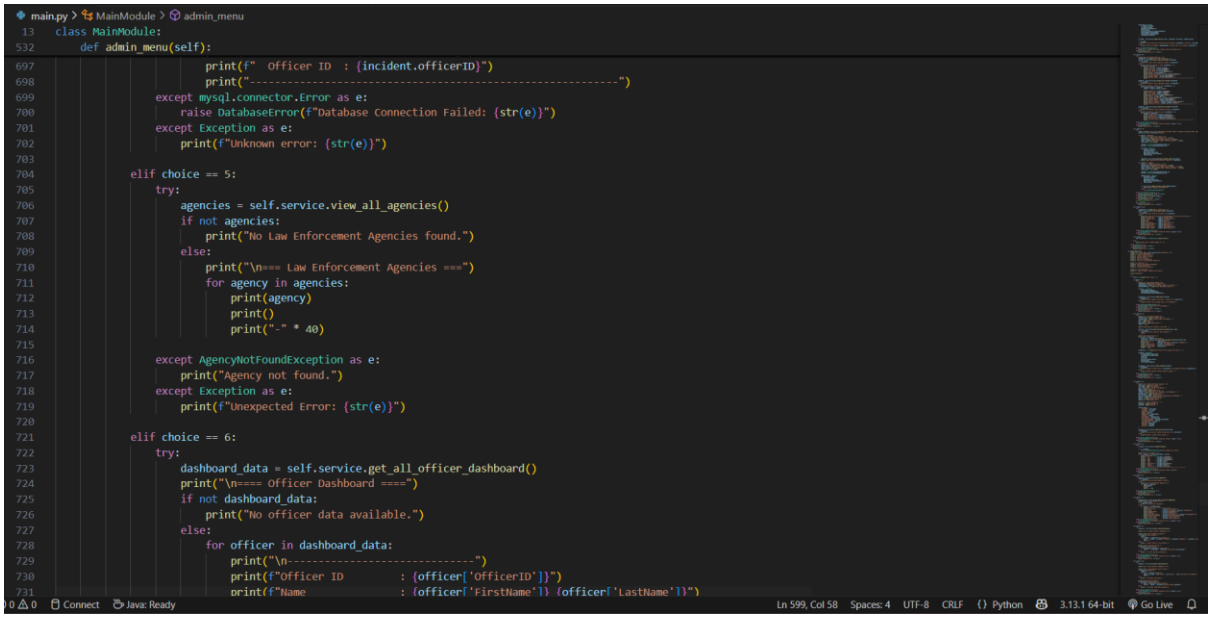
```
main.py > MainModule > admin_menu
 13  class MainModule:
532      def admin_menu(self):
598                  for officer in officers:
599                      officer_id = officer['OfficerID']
600                      case_count = self.service.count_open_incidents_by_officer(officer_id)
601                      print(f"Officer ID     : {officer_id}")
602                      print(f"Name           : {officer['FirstName']} {officer['LastName']}")
603                      print(f"Posting City   : {officer['PostingCity']}")
604                      print(f"Posting State  : {officer['PostingState']}")
605                      print(f"Current Cases  : {case_count}")
606                      print("-" * 30)
607
608                  assigned_id = int(input("Enter Officer ID to assign this case to: "))
609
610                  incident = Incidents(
611                      IncidentType=incident_type,
612                      IncidentDate=incident_date,
613                      Area=area,
614                      City=city,
615                      Description=description,
616                      Status=status,
617                      OfficerID=assigned_id
618                  )
619
620                  incidentID = self.service.create_incident(incident)
621                  if incidentID:
622                      print(f"Incident created with ID {incidentID} and assigned to Officer {assigned_id}")
623                  else:
624                      print("Incident Creation Failed. Please try again. ")
625
626              except DatabaseError as e:
627                  print(f"Database Error: {str(e)}")
628              except Exception as e:
629                  print(f"Unexpected Error: {str(e)}")
630
631
632          elif choice == 3:
```

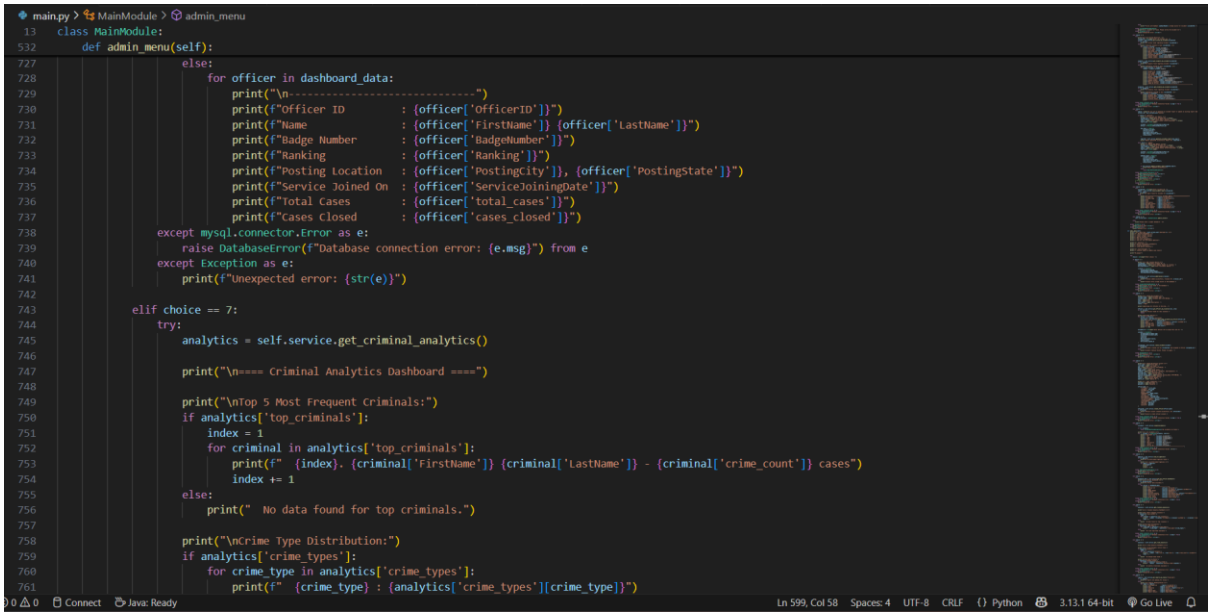
```
main.py > MainModule > admin_menu
 13  class MainModule:
532      def admin_menu(self):
632          elif choice == 3:
633              try:
634                  print("\n=== Create New Officer Account ===")
635                  first_name = input("First Name: ")
636                  last_name = input("Last Name: ")
637                  dob = input("Date of Birth (YYYY-MM-DD): ")
638                  gender = input("Gender: ")
639                  badge_number = input("Badge Number: ")
640                  ranking = input("Ranking (e.g., Inspector, Sub-Inspector): ")
641                  posting_city = input("Posting City: ")
642                  posting_state = input("Posting State: ")
643                  service_joining_date = input("Service Joining Date (YYYY-MM-DD): ")
644                  address = input("Residential Address: ")
645                  contact = input("Contact Number: ")
646                  agency_id = input("Agency ID: ")
647
648                  print("\n--- Login Credentials ---")
649                  username = input("Username: ")
650                  password = input("Password: ")
651
652                  officer_data = {
653                      'firstName': first_name,
654                      'lastName': last_name,
655                      'dateOfBirth': dob,
656                      'gender': gender,
657                      'badgeNumber': badge_number,
658                      'ranking': ranking,
659                      'postingCity': posting_city,
660                      'postingState': posting_state,
661                      'serviceJoiningDate': service_joining_date,
662                      'residentialAddress': address,
663                      'contactNumber': contact,
664                      'agencyID': agency_id,
665                      'username': username,
666                      'password': password
```



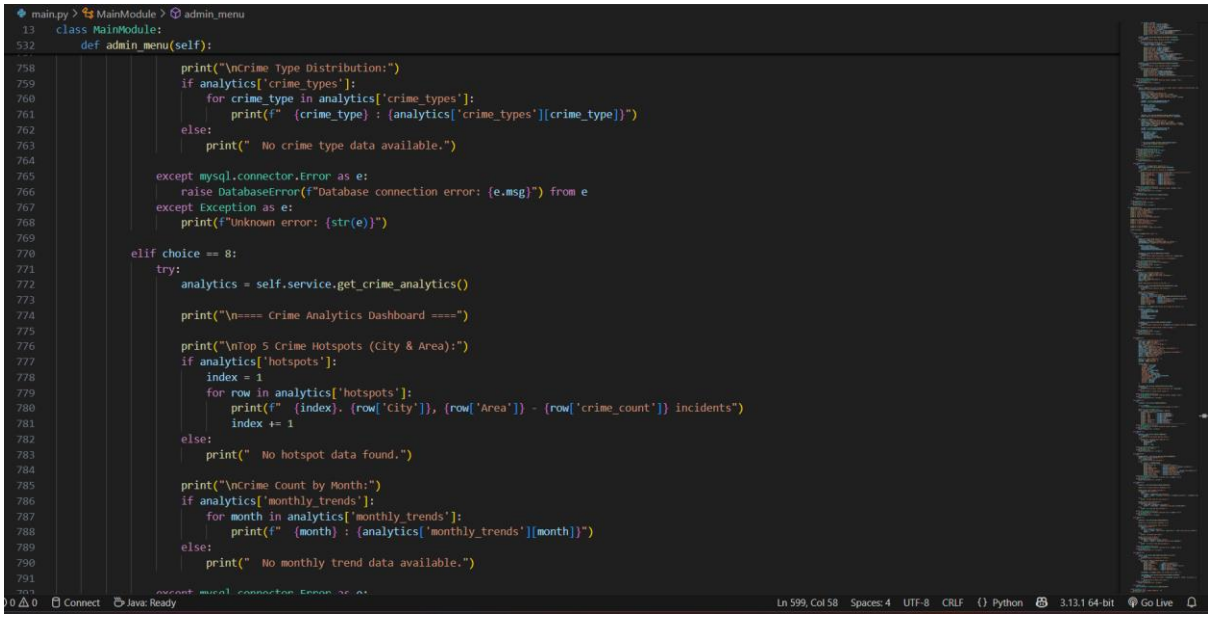
```
main.py > MainModule > admin_menu
13  class MainModule:
532     def admin_menu(self):
665                     'username': username,
666                     'password': password
667                 }
668
669                 officerID = self.service.create_officer(officer_data)
670                 if officerID:
671                     print(f"Officer account created successfully. ID: {officerID}")
672                 else:
673                     print("Failed to create officer account.")
674
675             except mysql.connector.Error as e:
676                 raise DatabaseError(f"Database Connection Failed: {e.msg}") from e
677             except Exception as e:
678                 print(f"Unexpected Error: {str(e)}")
679
680         elif choice == 4:
681             try:
682                 incidents = self.service.viewAllIncidents()
683
684                 if not incidents:
685                     raise IncidentNotFoundException(f"No incidents are found.")
686
687                 print("\n=== All Incidents ===")
688                 for i, incident in enumerate(incidents, start=1):
689                     print(f"\nIncident {i}:")
690                     print(f"  ID          : {incident.incidentID}")
691                     print(f"  Type        : {incident.incidentType}")
692                     print(f"  Date        : {incident.incidentDate}")
693                     print(f"  Area        : {incident.area}")
694                     print(f"  City        : {incident.city}")
695                     print(f"  Description : {incident.description}")
696                     print(f"  Status      : {incident.status}")
697                     print(f"  Officer ID  : {incident.officerID}")
698                     print("-----------------------------------------------------------")
699             except mysql.connector.Error as e:
0 0  Connect  Java: Ready   Ln 599, Col 58  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



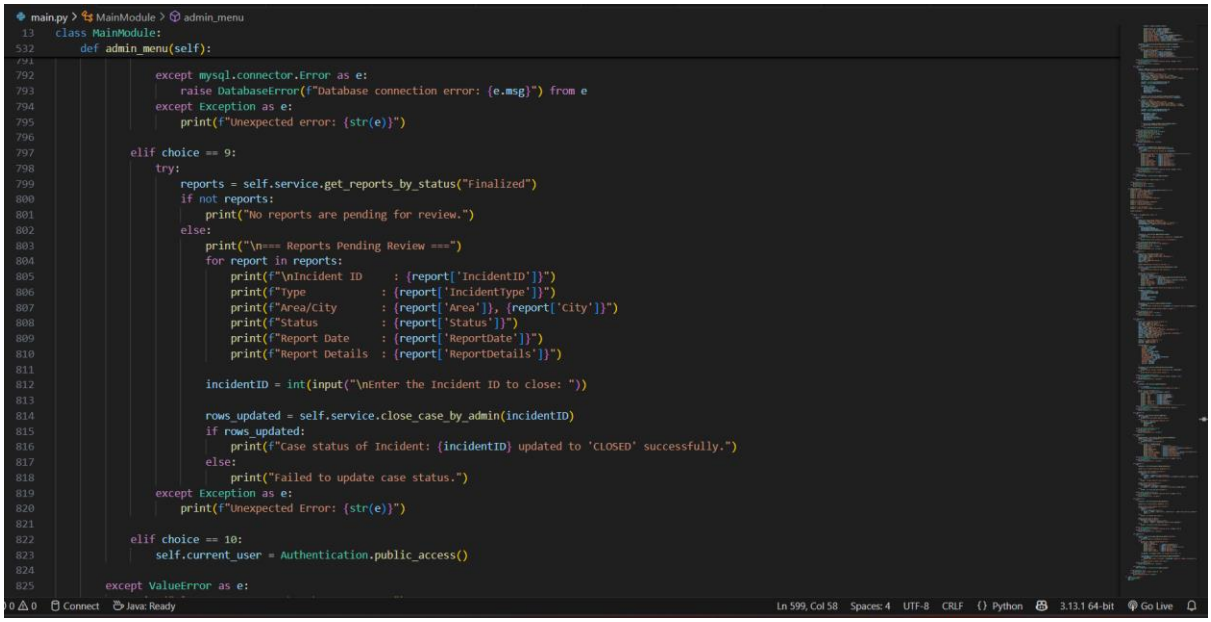
```
main.py > MainModule > admin_menu
13  class MainModule:
532     def admin_menu(self):
697                     print(f"  Officer ID  : {incident.officerID}")
698                     print("-----------------------------------------------------------")
699             except mysql.connector.Error as e:
700                 raise DatabaseError(f"Database Connection Failed: {str(e)}")
701             except Exception as e:
702                 print(f"Unknown error: {str(e)}")
703
704         elif choice == 5:
705             try:
706                 agencies = self.service.view_all_agencies()
707                 if not agencies:
708                     print("No Law Enforcement Agencies found.")
709                 else:
710                     print("\n=== Law Enforcement Agencies ===")
711                     for agency in agencies:
712                         print(agency)
713                         print()
714                         print("-" * 40)
715
716             except AgencyNotFoundException as e:
717                 print("Agency not found.")
718             except Exception as e:
719                 print(f"Unexpected Error: {str(e)}")
720
721         elif choice == 6:
722             try:
723                 dashboard_data = self.service.get_all_officer_dashboard()
724                 print("\n=== Officer Dashboard ====")
725                 if not dashboard_data:
726                     print("No officer data available.")
727                 else:
728                     for officer in dashboard_data:
729                         print("\n-------------------------------")
730                         print(f"Officer ID         : {officer['OfficerID']}")
731                         print(f"Name               : {officer['FirstName']} {officer['LastName']}")
0 0  Connect  Java: Ready   Ln 599, Col 58  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```



```
main.py > MainModule > admin_menu
 13   class MainModule:
532       def admin_menu(self):
727                   else:
728                       for officer in dashboard_data:
729                           print("\n------------------------------")
730                           print(f"Officer ID         : {officer['OfficerID']}")
731                           print(f"Name               : {officer['FirstName']} {officer['LastName']}")
732                           print(f"Badge Number       : {officer['BadgeNumber']}")
733                           print(f"Ranking            : {officer['Ranking']}")
734                           print(f"Posting Location   : {officer['PostingCity']}, {officer['PostingState']}")
735                           print(f"Service Joined On  : {officer['ServiceJoiningDate']}")
736                           print(f"Total Cases        : {officer['total_cases']}")
737                           print(f"Cases Closed       : {officer['cases_closed']}")
738               except mysql.connector.Error as e:
739                   raise DatabaseError(f"Database connection error: {e.msg}") from e
740               except Exception as e:
741                   print(f"Unexpected error: {str(e)}")
742
743           elif choice == 7:
744               try:
745                   analytics = self.service.get_criminal_analytics()
746
747                   print("\n==== Criminal Analytics Dashboard ====")
748
749                   print("\nTop 5 Most Frequent Criminals:")
750                   if analytics['top_criminals']:
751                       index = 1
752                       for criminal in analytics['top_criminals']:
753                           print(f"  {index}. {criminal['FirstName']} {criminal['LastName']} - {criminal['crime_count']} cases")
754                           index += 1
755                   else:
756                       print("  No data found for top criminals.")
757
758                   print("\nCrime Type Distribution:")
759                   if analytics['crime_types']:
760                       for crime_type in analytics['crime_types']:
761                           print(f"  {crime_type} : {analytics['crime_types'][crime_type]}")
```

```
main.py > MainModule > admin_menu
 13   class MainModule:
532       def admin_menu(self):
758                   print("\nCrime Type Distribution:")
759                   if analytics['crime_types']:
760                       for crime_type in analytics['crime_types']:
761                           print(f"  {crime_type} : {analytics['crime_types'][crime_type]}")
762                   else:
763                       print("  No crime type data available.")
764
765               except mysql.connector.Error as e:
766                   raise DatabaseError(f"Database connection error: {e.msg}") from e
767               except Exception as e:
768                   print(f"Unknown error: {str(e)}")
769
770           elif choice == 8:
771               try:
772                   analytics = self.service.get_crime_analytics()
773
774                   print("\n==== Crime Analytics Dashboard ====")
775
776                   print("\nTop 5 Crime Hotspots (City & Area):")
777                   if analytics['hotspots']:
778                       index = 1
779                       for row in analytics['hotspots']:
780                           print(f"  {index}. {row['City']}, {row['Area']} - {row['crime_count']} incidents")
781                           index += 1
782                   else:
783                       print("  No hotspot data found.")
784
785                   print("\nCrime Count by Month:")
786                   if analytics['monthly_trends']:
787                       for month in analytics['monthly_trends']:
788                           print(f"  {month} : {analytics['monthly_trends'][month]}")
789                   else:
790                       print("  No monthly trend data available.")
791
```

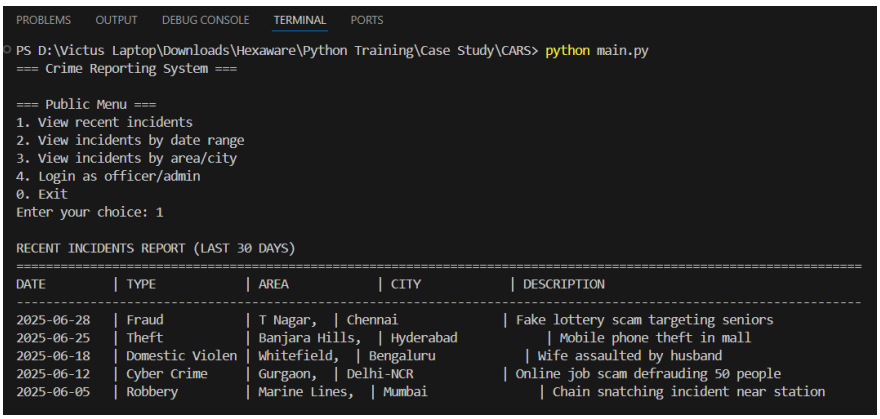


```
main.py > MainModule > admin_menu
13  class MainModule:
532     def admin_menu(self):
792                 except mysql.connector.Error as e:
793                     raise DatabaseError(f"Database connection error: {e.msg}") from e
794                 except Exception as e:
795                     print(f"Unexpected error: {str(e)}")
796
797             elif choice == 9:
798                 try:
799                     reports = self.service.get_reports_by_status("Finalized")
800                     if not reports:
801                         print("No reports are pending for review.")
802                     else:
803                         print("\n=== Reports Pending Review ===")
804                         for report in reports:
805                             print(f"\nIncident ID     : {report['IncidentID']}")
806                             print(f"Type            : {report['IncidentType']}")
807                             print(f"Area/city       : {report['Area']}, {report['city']}")
808                             print(f"Status          : {report['Status']}")
809                             print(f"Report Date     : {report['ReportDate']}")
810                             print(f"Report Details  : {report['ReportDetails']}")
811
812                         incidentID = int(input("\nEnter the Incident ID to close: "))
813
814                         rows_updated = self.service.close_case_by_admin(incidentID)
815                         if rows_updated:
816                             print(f"Case status of Incident: {incidentID} updated to 'CLOSED' successfully.")
817                         else:
818                             print("Failed to update case status.")
819                 except Exception as e:
820                     print(f"Unexpected Error: {str(e)}")
821
822             elif choice == 10:
823                 self.current_user = Authentication.public_access()
824
825         except ValueError as e:
0 0  Connect  Java: Ready   Ln 599, Col 58  Spaces: 4  UTF-8  CRLF  {} Python  3.13.1 64-bit  Go Live
```

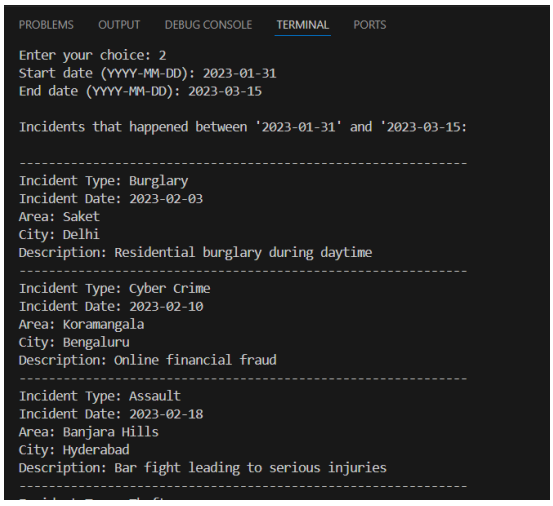
```
824
825         except ValueError as e:
826             print("Please enter a number between 0 - 3")
827         except Exception as e:
828             print(f"Unexpected Error: {str(e)}")
829
830 if __name__ == "__main__":
831     app = MainModule()
832     app.run()
```

Console Output

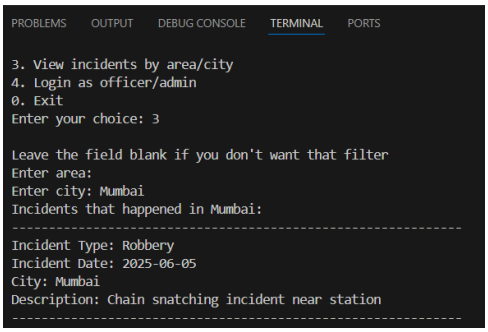
1. Public Menu: Viewing recent incidents



2. Public Menu: View incidents by date range



3. Public Menu: View incidents by area/city



4. Public Menu: Login as officer

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

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

=== Officer Menu ===

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

5. Officer Menu: View all incident details with assignments

```
9. View case report
10. Logout
Enter choice: 1

=== All Incidents ===

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

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

6. Officer Menu: View my assigned incidents

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

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

=== Officer Menu ===
```

7. Officer Menu: Filter Incidents

```
10. Logout
Enter choice: 3

==== Filtering the Incidents ====

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

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

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

8. Officer Menu: Access suspects database

```
-- Case Management --
7. View case details (victims/suspects/evidence)
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 4
==== Suspects Database ====
  Suspect ID               : 1032
  Name                     : John Doe
  Date of Birth            : 1990-01-01
  Aadhaar Number           : 123456789012
  Address                  : Some Street
  Contact Number           : 9876543210
  Incident ID              : 1
  Incident Type            : Robbery
  Incident Date            : 2023-01-15
  Description              : Armed robbery at jewelry store
  Status                   : Closed
  Role in Incident         : Suspect in prior theft
  Added By Officer         : 1
  Previous Criminal Record : No
------------------------------------------------------------
  Suspect ID               : 1031
  Name                     : Sunita Sharma
  Date of Birth            : 1986-03-25
  Aadhaar Number           : 565656565656
  Address                  : Flat 7, Kothrud, Pune
  Contact Number           : 7210789017
  Incident ID              : 27
  Incident Type            : Cyber Crime
  Incident Date            : 2025-06-12
  Description              : Online job scam defrauding 50 people
  Status                   : Open
  Role in Incident         : Handled financial transactions for scam
  Added By Officer         : 2
  Previous Criminal Record : No
------------------------------------------------------------
```

9. Officer Menu: Create new suspect record

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

10. Officer Menu: Access criminals database

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

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

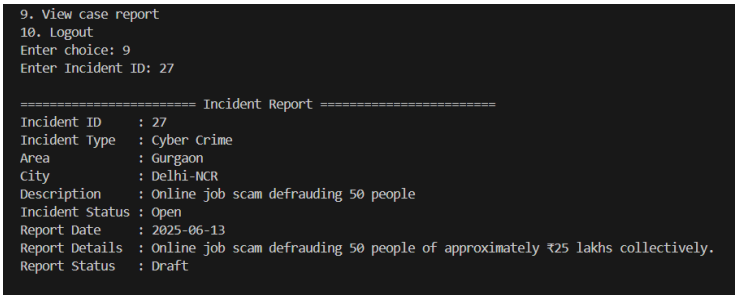
```
9. View case report
10. Logout
Enter choice: 7
View Case Related Resources:

Enter the Incident ID: 3

Victims related to Case 3:

Victim ID: 102
First Name: Rahul
Last Name: Iyer
Date Of Birth: 1990-11-08
Gender: Male
Residential Address: No. 45, 5th Cross, Koramangala, Bengaluru
Contact Number: 7654321098
Aadhaar Number: 345678901234
------------------------------------------------------------------

Suspects related to Case 3:

Suspect ID: 1002
First Name: Suresh
Last Name: Kumar
Date Of Birth: 1978-11-30
Gender: Male
Residential Address: No. 56, Shivajinagar, Bengaluru
Contact Number: 7656345672
Aadhaar Number: 345634563456
Role in Incident: Created fake investment website
Added By Officer: 3
------------------------------------------------------------------
```

12.Officer Menu: Generate/Update case report

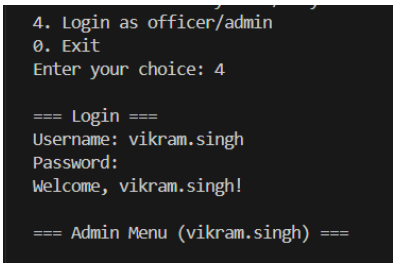
```
7. View case details (victims/suspects/evidence)
8. Generate/Update case report
9. View case report
10. Logout
Enter choice: 8
Do you wish to generate an incident report or update an existing report(Generate/update): generate

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

=== Officer Menu ===
```

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

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

=== Officer Menu ===
```

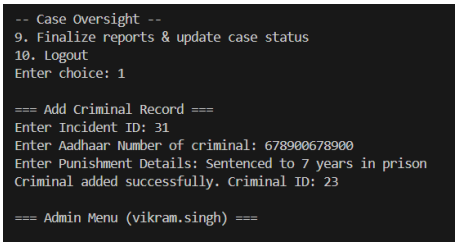
13.Officer Menu: View case report



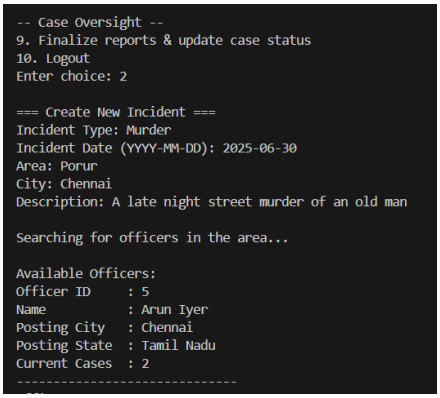
14. Admin Menu: Login



15.Admin Menu: Create criminal record



16.Admin Menu: Create incident record



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

Searching for officers in the area...

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

17.Admin Menu: Add new officer

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

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

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

18. Admin Menu: View case assignments

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

=== All Incidents ===

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

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

19.Admin Menu: View all law enforcement agencies

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

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

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

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

20.Admin Menu: Officer Performance Dashboard

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

==== Officer Dashboard ====

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

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

21.Admin Menu: Criminal Analytics

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

==== Criminal Analytics Dashboard ====

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

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

22.Admin Menu: Crime Pattern analytics

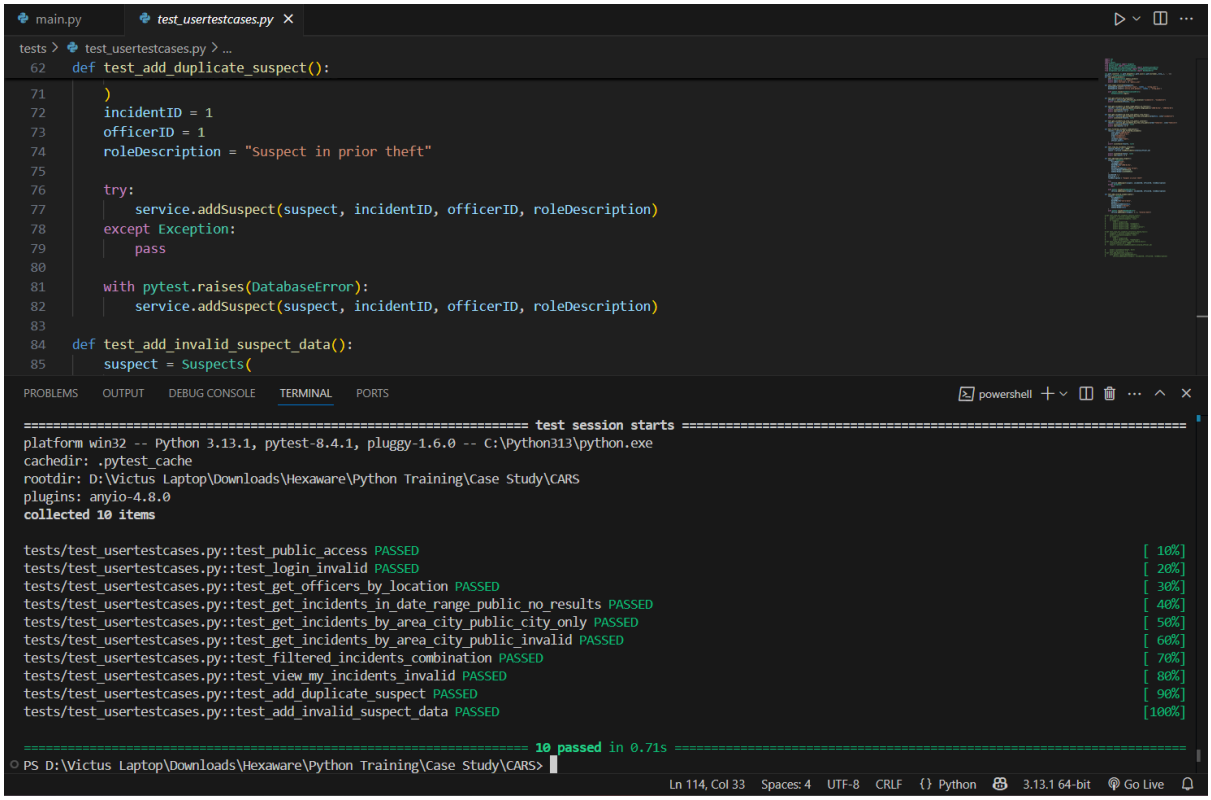
```
9. Finalize reports & update case status
10. Logout
Enter choice: 8

==== Crime Analytics Dashboard ====

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

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

23.Admin Menu: Finalize reports & update case status

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

=== Reports Pending Review ===

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

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

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

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

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

24.Testing


main.py
test_usertestcases.py
tests > test_usertestcases.py > ...
62 def test_add_duplicate_suspect():
71)
72 incidentID = 1
73 officerID = 1
74 roleDescription = "Suspect in prior theft"
75
76 try:
77 service.addSuspect(suspect, incidentID, officerID, roleDescription)
78 except Exception:
79 pass
80
81 with pytest.raises(DatabaseError):
82 service.addSuspect(suspect, incidentID, officerID, roleDescription)
83
84 def test_add_invalid_suspect_data():
85 suspect = Suspects(
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
test session starts
platform win32 -- Python 3.13.1, pytest-8.4.1, pluggy-1.6.0 -- C:\Python313\python.exe
cachedir: .pytest_cache
rootdir: D:\Victus Laptop\Downloads\Hexaware\Python Training\Case Study\CARS
plugins: anyio-4.8.0
collected 10 items
tests/test_usertestcases.py::test_public_access PASSED [10%]
tests/test_usertestcases.py::test_login_invalid PASSED [20%]
tests/test_usertestcases.py::test_get_officers_by_location PASSED [30%]
tests/test_usertestcases.py::test_get_incidents_in_date_range_public_no_results PASSED [40%]
tests/test_usertestcases.py::test_get_incidents_by_area_city_public_city_only PASSED [50%]
tests/test_usertestcases.py::test_get_incidents_by_area_city_public_invalid PASSED [60%]
tests/test_usertestcases.py::test_filtered_incidents_combination PASSED [70%]
tests/test_usertestcases.py::test_view_my_incidents_invalid PASSED [80%]
tests/test_usertestcases.py::test_add_duplicate_suspect PASSED [90%]
tests/test_usertestcases.py::test_add_invalid_suspect_data PASSED [100%]
10 passed in 0.71s
PS D:\Victus Laptop\Downloads\Hexaware\Python Training\Case Study\CARS>
Ln 114, Col 33 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit Go Live

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 – C.A.R.S.

C.A.R.S. employs multi-factor authentication (MFA) to ensure that only authorized personnel access sensitive data, significantly reducing the risk of breaches and enhancing public trust. By integrating with national criminal databases, the system allows officers to retrieve and cross-reference information in real time, speeding up investigations and improving accuracy in suspect identification.

Automated data backup and recovery mechanisms safeguard critical information, ensuring that records are not lost even in the event of system failures or cyberattacks. GIS integration enables real-time visualization of crime hotspots on interactive maps, helping law enforcement agencies identify patterns, allocate resources effectively, and respond to high-risk areas promptly.

AI-based pattern detection analyzes historical and real-time data to generate predictive insights, allowing proactive measures in law enforcement and potentially preventing crimes before they occur.