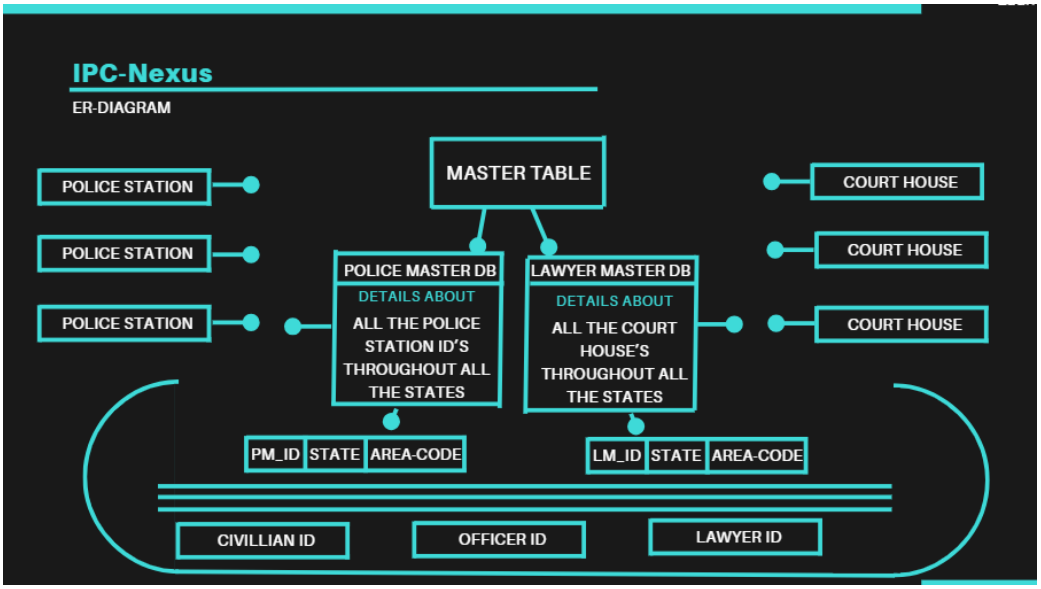
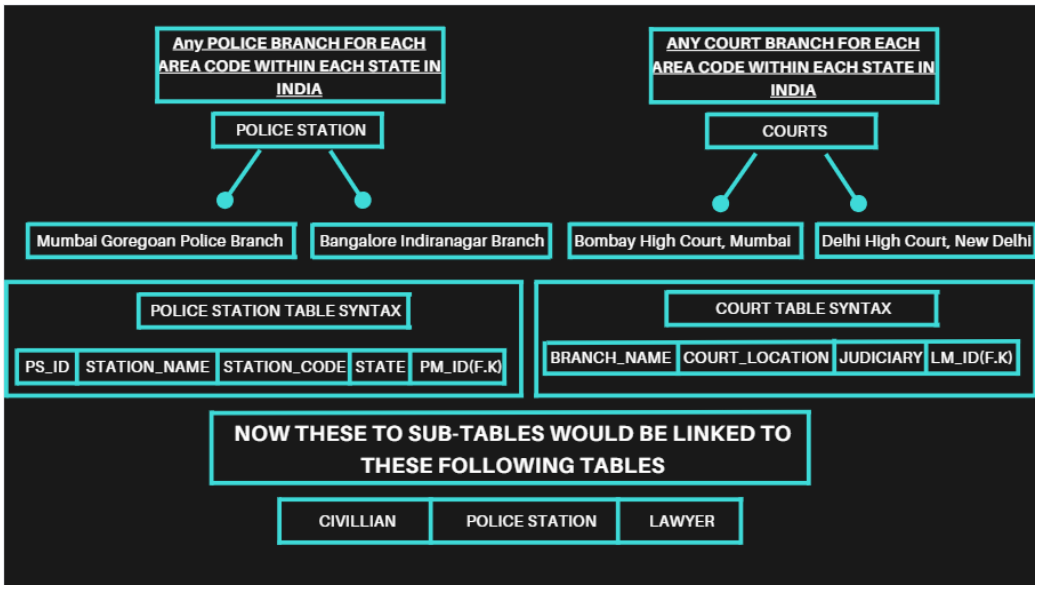
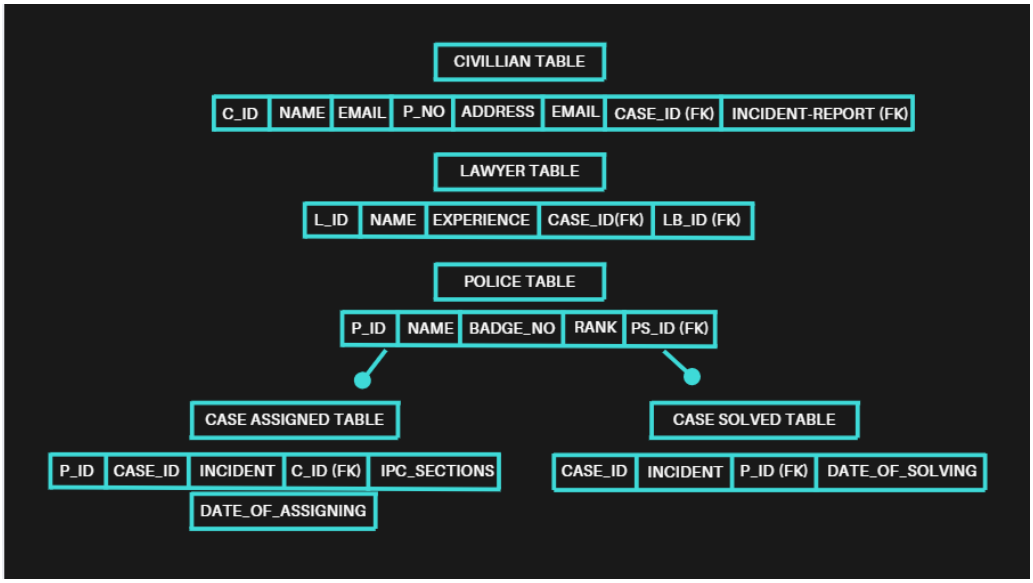
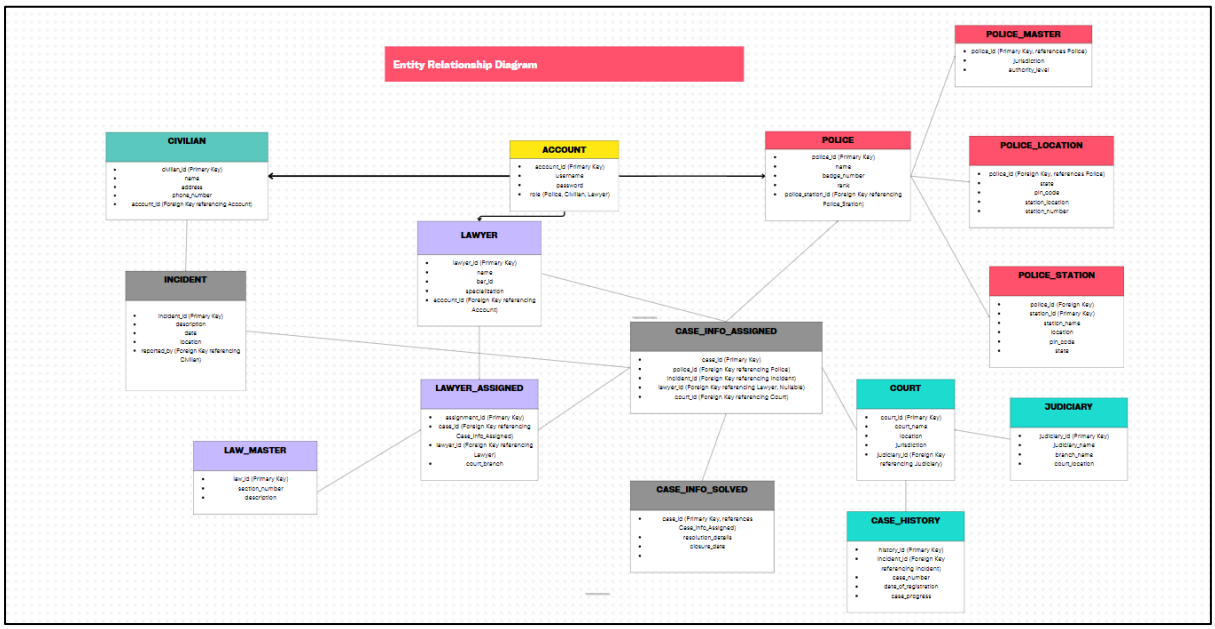
DATABASE FLOW:







ER-DIAGRAM:



DATABASE CODE STRUCTURE:

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_cors import CORS
from werkzeug.security import generate_password_hash, check_password_hash
import os

app = Flask(_name_)
CORS(app, resources={r"/api/": {"origins": ""}}, supports_credentials=True)

# Database config
DB_PATH = r"D:\MTHREE\UPDATED_PROJECT\Backend\Database\law_enforcement.db"
os.makedirs(os.path.dirname(DB_PATH), exist_ok=True)

app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{DB_PATH}'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# -------------------- MODELS --------------------

class Account(db.Model):
    _tablename_ = 'accounts'
    account_id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    email = db.Column(db.String(100), unique=True, nullable=True)
    phoneno = db.Column(db.String(15), nullable=False)
```

```
    password_hash = db.Column(db.String(128), nullable=False)

    civilian = db.relationship('Civilian', backref='account', uselist=False)
    lawyer = db.relationship('Lawyer', backref='account', uselist=False)
    police = db.relationship('Police', backref='account', uselist=False)


class Civilian(db.Model):
    _tablename_ = 'civilians'
    civilian_id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'),
nullable=False)

    cases = db.relationship('Case', backref='civilian')
    incidents = db.relationship('Incident', backref='civilian')


class Lawyer(db.Model):
    _tablename_ = 'lawyers'
    lawyer_id = db.Column(db.String(20), primary_key=True)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'),
nullable=False)

    case = db.relationship('Case', backref='lawyer', uselist=False)
    court_location_id = db.Column(db.Integer,
db.ForeignKey('court_locations.court_id'))


class Police(db.Model):
    _tablename_ = 'police'
    badge_id = db.Column(db.String(20), primary_key=True)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'),
nullable=False)
    police_station_id = db.Column(db.Integer,
db.ForeignKey('police_stations.ps_id'))

    case_assignments = db.relationship('CaseAssign', backref='police')
    case_solved = db.relationship('CaseSolved', backref='police')


class Case(db.Model):
    _tablename_ = 'cases'
    case_id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=False)
    civilian_id = db.Column(db.Integer,
db.ForeignKey('civilians.civilian_id'))
    lawyer_id = db.Column(db.String(20), db.ForeignKey('lawyers.lawyer_id'))
```

```
    case_assign = db.relationship('CaseAssign', backref='case', uselist=False)
    case_solved = db.relationship('CaseSolved', backref='case', uselist=False)


class CaseAssign(db.Model):
    _tablename_ = 'case_assign'
    assign_id = db.Column(db.Integer, primary_key=True)
    case_id = db.Column(db.Integer, db.ForeignKey('cases.case_id'))
    police_id = db.Column(db.String(20), db.ForeignKey('police.badge_id'))


class CaseSolved(db.Model):
    _tablename_ = 'case_solved'
    solved_id = db.Column(db.Integer, primary_key=True)
    case_id = db.Column(db.Integer, db.ForeignKey('cases.case_id'))
    police_id = db.Column(db.String(20), db.ForeignKey('police.badge_id'))


class PoliceStation(db.Model):
    _tablename_ = 'police_stations'
    ps_id = db.Column(db.Integer, primary_key=True)
    station_name = db.Column(db.String(100), nullable=False)
    police = db.relationship('Police', backref='station', uselist=False)

    master_id = db.Column(db.Integer, db.ForeignKey('police_master.pm_id'))


class PoliceMaster(db.Model):
    _tablename_ = 'police_master'
    pm_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    stations = db.relationship('PoliceStation', backref='master')


class CourtLocation(db.Model):
    _tablename_ = 'court_locations'
    court_id = db.Column(db.Integer, primary_key=True)
    location = db.Column(db.String(100), nullable=False)

    master_id = db.Column(db.Integer, db.ForeignKey('lawyer_master.lm_id'))
    lawyers = db.relationship('Lawyer', backref='court_location')


class LawyerMaster(db.Model):
    _tablename_ = 'lawyer_master'
    lm_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    courts = db.relationship('CourtLocation', backref='master')
```

```python
class Incident(db.Model):
    _tablename_ = 'incidents'
    incident_id = db.Column(db.Integer, primary_key=True)
    description = db.Column(db.Text, nullable=False)
    location = db.Column(db.String(100), nullable=False)
    incident_date = db.Column(db.String(20), nullable=False)
    civilian_id = db.Column(db.Integer,
db.ForeignKey('civilians.civilian_id'))


class Support(db.Model):
    _tablename_ = 'support'
    support_id = db.Column(db.Integer, primary_key=True)
    message = db.Column(db.Text, nullable=False)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'))


# -------------------- ROUTES --------------------

@app.route("/api/civilian/signup", methods=["POST"])
def signup():
    data = request.json
    username = data.get('username')
    phoneno = data.get('phoneno')
    password = data.get('password')

    if not username or not password:
        return jsonify({'message': 'Username and Password required'}), 400

    existing_user = Account.query.filter_by(username=username).first()
    if existing_user:
        return jsonify({'message': 'User with same username exists!'}), 409

    hashed_password = generate_password_hash(password)
    account = Account(username=username, phoneno=phoneno,
password_hash=hashed_password)
    db.session.add(account)
    db.session.commit()

    civilian = Civilian(username=username, account_id=account.account_id)
    db.session.add(civilian)
    db.session.commit()

    return jsonify({"message": "Signup successful"}), 201


@app.route("/api/civilian/login", methods=["POST"])
```

```
def login():
    data = request.json
    username = data.get('idOrUsername')
    password = data.get('password')

    if not username or not password:
        return jsonify({"message": "Username and Password required"}), 400

    account = Account.query.filter_by(username=username).first()
    if not account or not check_password_hash(account.password_hash,
password):
        return jsonify({"message": "Invalid credentials"}), 401

    civilian = Civilian.query.filter_by(account_id=account.account_id).first()
    if not civilian:
        return jsonify({"message": "No civilian profile found"}), 404

    return jsonify({
        "message": "Login successful",
        "account_id": account.account_id,
        "civilian_id": civilian.civilian_id
    }), 200


@app.route("/api/lawyer/signup", methods=["POST"])
def lawyer_signup():
    data = request.json
    lawyer_id = data.get('id')
    phoneno = data.get('phoneno')
    password = data.get('password')
    email = data.get('email')

    if not lawyer_id or not password:
        return jsonify({'message': 'Lawyer ID and Password required'}), 400

    existing_user = Lawyer.query.filter_by(lawyer_id=lawyer_id).first()
    if existing_user:
        return jsonify({'message': 'Lawyer with same ID exists!'}), 409

    hashed_password = generate_password_hash(password)
    account = Account(email=email, phoneno=phoneno,
password_hash=hashed_password, username=lawyer_id)
    db.session.add(account)
    db.session.commit()

    lawyer = Lawyer(lawyer_id=lawyer_id, account_id=account.account_id)
    db.session.add(lawyer)
    db.session.commit()
```

```
    return jsonify({"message": "Lawyer signup successful"}), 201


@app.route("/api/lawyer/login", methods=["POST"])
def lawyer_login():
    data = request.json
    lawyer_id = data.get('idOrUsername')
    password = data.get('password')

    if not lawyer_id or not password:
        return jsonify({"message": "Lawyer ID and Password required"}), 400

    lawyer = Lawyer.query.filter_by(lawyer_id=lawyer_id).first()
    if not lawyer:
        return jsonify({"message": "No lawyer profile found"}), 404

    account = Account.query.filter_by(account_id=lawyer.account_id).first()
    if not account or not check_password_hash(account.password_hash,
password):
        return jsonify({"message": "Invalid credentials"}), 401

    return jsonify({"message": "Lawyer login successful"}), 200


@app.route("/api/police/signup", methods=["POST"])
def police_signup():
    data = request.json
    badge_id = data.get('id')
    phoneno = data.get('phoneno')
    password = data.get('password')
    email = data.get('email')

    if not badge_id or not password:
        return jsonify({'message': 'Badge ID and Password required'}), 400

    existing_user = Police.query.filter_by(badge_id=badge_id).first()
    if existing_user:
        return jsonify({'message': 'Police with same Badge ID exists!'}), 409

    hashed_password = generate_password_hash(password)
    account = Account(email=email, phoneno=phoneno,
password_hash=hashed_password, username=badge_id)
    db.session.add(account)
    db.session.commit()

    police = Police(badge_id=badge_id, account_id=account.account_id)
    db.session.add(police)
    db.session.commit()
```

```
    return jsonify({"message": "Police signup successful"}), 201


@app.route("/api/police/login", methods=["POST"])
def police_login():
    data = request.json
    badge_id = data.get('idOrUsername')
    password = data.get('password')

    if not badge_id or not password:
        return jsonify({"message": "Badge ID and Password required"}), 400

    police = Police.query.filter_by(badge_id=badge_id).first()
    if not police:
        return jsonify({"message": "No police profile found"}), 404

    account = Account.query.filter_by(account_id=police.account_id).first()
    if not account or not check_password_hash(account.password_hash,
password):
        return jsonify({"message": "Invalid credentials"}), 401

    return jsonify({"message": "Police login successful"}), 200


@app.route("/api/civilian/complaint", methods=["POST"])
def register_complaint():
    data = request.json
    description = data.get('description')
    location = data.get('location')
    incident_date = data.get('incident_date')
    civilian_id = data.get('civilian_id')

    if not description or not location or not incident_date or not
civilian_id:
        return jsonify({'message': 'All fields are required'}), 400

    civilian = Civilian.query.get(civilian_id)
    if not civilian:
        return jsonify({'message': 'Civilian not found'}), 404

    incident = Incident(description=description, location=location,
incident_date=incident_date, civilian_id=civilian_id)
    db.session.add(incident)
    db.session.commit()

    return jsonify({'message': 'Complaint registered successfully'}), 201


if _name_ == "_main_":
```

```
    with app.app_context():
        db.create_all()
    app.run(debug=True)
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_cors import CORS
from werkzeug.security import generate_password_hash, check_password_hash
import os

app = Flask(_name_)
CORS(app, resources={r"/api/": {"origins": ""}}, supports_credentials=True)

# Database config
DB_PATH = r"D:\MTHREE\UPDATED_PROJECT\Backend\Database\law_enforcement.db"
os.makedirs(os.path.dirname(DB_PATH), exist_ok=True)

app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{DB_PATH}'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# -------------------- MODELS --------------------

class Account(db.Model):
    _tablename_ = 'accounts'
    account_id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    email = db.Column(db.String(100), unique=True, nullable=True)
    phoneno = db.Column(db.String(15), nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)

    civilian = db.relationship('Civilian', backref='account', uselist=False)
    lawyer = db.relationship('Lawyer', backref='account', uselist=False)
    police = db.relationship('Police', backref='account', uselist=False)


class Civilian(db.Model):
    _tablename_ = 'civilians'
    civilian_id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'),
nullable=False)

    cases = db.relationship('Case', backref='civilian')
    incidents = db.relationship('Incident', backref='civilian')


class Lawyer(db.Model):
    _tablename_ = 'lawyers'
```

```
    lawyer_id = db.Column(db.String(20), primary_key=True)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'),
nullable=False)

    case = db.relationship('Case', backref='lawyer', uselist=False)
    court_location_id = db.Column(db.Integer,
db.ForeignKey('court_locations.court_id'))


class Police(db.Model):
    _tablename_ = 'police'
    badge_id = db.Column(db.String(20), primary_key=True)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'),
nullable=False)
    police_station_id = db.Column(db.Integer,
db.ForeignKey('police_stations.ps_id'))

    case_assignments = db.relationship('CaseAssign', backref='police')
    case_solved = db.relationship('CaseSolved', backref='police')


class Case(db.Model):
    _tablename_ = 'cases'
    case_id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=False)
    civilian_id = db.Column(db.Integer,
db.ForeignKey('civilians.civilian_id'))
    lawyer_id = db.Column(db.String(20), db.ForeignKey('lawyers.lawyer_id'))

    case_assign = db.relationship('CaseAssign', backref='case', uselist=False)
    case_solved = db.relationship('CaseSolved', backref='case', uselist=False)


class CaseAssign(db.Model):
    _tablename_ = 'case_assign'
    assign_id = db.Column(db.Integer, primary_key=True)
    case_id = db.Column(db.Integer, db.ForeignKey('cases.case_id'))
    police_id = db.Column(db.String(20), db.ForeignKey('police.badge_id'))


class CaseSolved(db.Model):
    _tablename_ = 'case_solved'
    solved_id = db.Column(db.Integer, primary_key=True)
    case_id = db.Column(db.Integer, db.ForeignKey('cases.case_id'))
    police_id = db.Column(db.String(20), db.ForeignKey('police.badge_id'))


class PoliceStation(db.Model):
```

```
    _tablename_ = 'police_stations'
    ps_id = db.Column(db.Integer, primary_key=True)
    station_name = db.Column(db.String(100), nullable=False)
    police = db.relationship('Police', backref='station', uselist=False)

    master_id = db.Column(db.Integer, db.ForeignKey('police_master.pm_id'))


class PoliceMaster(db.Model):
    _tablename_ = 'police_master'
    pm_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    stations = db.relationship('PoliceStation', backref='master')


class CourtLocation(db.Model):
    _tablename_ = 'court_locations'
    court_id = db.Column(db.Integer, primary_key=True)
    location = db.Column(db.String(100), nullable=False)

    master_id = db.Column(db.Integer, db.ForeignKey('lawyer_master.lm_id'))
    lawyers = db.relationship('Lawyer', backref='court_location')


class LawyerMaster(db.Model):
    _tablename_ = 'lawyer_master'
    lm_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    courts = db.relationship('CourtLocation', backref='master')


class Incident(db.Model):
    _tablename_ = 'incidents'
    incident_id = db.Column(db.Integer, primary_key=True)
    description = db.Column(db.Text, nullable=False)
    location = db.Column(db.String(100), nullable=False)
    incident_date = db.Column(db.String(20), nullable=False)
    civilian_id = db.Column(db.Integer,
db.ForeignKey('civilians.civilian_id'))


class Support(db.Model):
    _tablename_ = 'support'
    support_id = db.Column(db.Integer, primary_key=True)
    message = db.Column(db.Text, nullable=False)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.account_id'))


# -------------------- ROUTES --------------------
```

```python
@app.route("/api/civilian/signup", methods=["POST"])
def signup():
    data = request.json
    username = data.get('username')
    phoneno = data.get('phoneno')
    password = data.get('password')

    if not username or not password:
        return jsonify({'message': 'Username and Password required'}), 400

    existing_user = Account.query.filter_by(username=username).first()
    if existing_user:
        return jsonify({'message': 'User with same username exists!'}), 409

    hashed_password = generate_password_hash(password)
    account = Account(username=username, phoneno=phoneno,
password_hash=hashed_password)
    db.session.add(account)
    db.session.commit()

    civilian = Civilian(username=username, account_id=account.account_id)
    db.session.add(civilian)
    db.session.commit()

    return jsonify({"message": "Signup successful"}), 201


@app.route("/api/civilian/login", methods=["POST"])
def login():
    data = request.json
    username = data.get('idOrUsername')
    password = data.get('password')

    if not username or not password:
        return jsonify({"message": "Username and Password required"}), 400

    account = Account.query.filter_by(username=username).first()
    if not account or not check_password_hash(account.password_hash,
password):
        return jsonify({"message": "Invalid credentials"}), 401

    civilian = Civilian.query.filter_by(account_id=account.account_id).first()
    if not civilian:
        return jsonify({"message": "No civilian profile found"}), 404

    return jsonify({
        "message": "Login successful",
        "account_id": account.account_id,
```

```
        "civilian_id": civilian.civilian_id
    }), 200


@app.route("/api/lawyer/signup", methods=["POST"])
def lawyer_signup():
    data = request.json
    lawyer_id = data.get('id')
    phoneno = data.get('phoneno')
    password = data.get('password')
    email = data.get('email')

    if not lawyer_id or not password:
        return jsonify({'message': 'Lawyer ID and Password required'}), 400

    existing_user = Lawyer.query.filter_by(lawyer_id=lawyer_id).first()
    if existing_user:
        return jsonify({'message': 'Lawyer with same ID exists!'}), 409

    hashed_password = generate_password_hash(password)
    account = Account(email=email, phoneno=phoneno,
password_hash=hashed_password, username=lawyer_id)
    db.session.add(account)
    db.session.commit()

    lawyer = Lawyer(lawyer_id=lawyer_id, account_id=account.account_id)
    db.session.add(lawyer)
    db.session.commit()

    return jsonify({"message": "Lawyer signup successful"}), 201


@app.route("/api/lawyer/login", methods=["POST"])
def lawyer_login():
    data = request.json
    lawyer_id = data.get('idOrUsername')
    password = data.get('password')

    if not lawyer_id or not password:
        return jsonify({"message": "Lawyer ID and Password required"}), 400

    lawyer = Lawyer.query.filter_by(lawyer_id=lawyer_id).first()
    if not lawyer:
        return jsonify({"message": "No lawyer profile found"}), 404

    account = Account.query.filter_by(account_id=lawyer.account_id).first()
    if not account or not check_password_hash(account.password_hash,
password):
        return jsonify({"message": "Invalid credentials"}), 401
```

```
    return jsonify({"message": "Lawyer login successful"}), 200


@app.route("/api/police/signup", methods=["POST"])
def police_signup():
    data = request.json
    badge_id = data.get('id')
    phoneno = data.get('phoneno')
    password = data.get('password')
    email = data.get('email')

    if not badge_id or not password:
        return jsonify({'message': 'Badge ID and Password required'}), 400

    existing_user = Police.query.filter_by(badge_id=badge_id).first()
    if existing_user:
        return jsonify({'message': 'Police with same Badge ID exists!'}), 409

    hashed_password = generate_password_hash(password)
    account = Account(email=email, phoneno=phoneno,
password_hash=hashed_password, username=badge_id)
    db.session.add(account)
    db.session.commit()

    police = Police(badge_id=badge_id, account_id=account.account_id)
    db.session.add(police)
    db.session.commit()

    return jsonify({"message": "Police signup successful"}), 201


@app.route("/api/police/login", methods=["POST"])
def police_login():
    data = request.json
    badge_id = data.get('idOrUsername')
    password = data.get('password')

    if not badge_id or not password:
        return jsonify({"message": "Badge ID and Password required"}), 400

    police = Police.query.filter_by(badge_id=badge_id).first()
    if not police:
        return jsonify({"message": "No police profile found"}), 404

    account = Account.query.filter_by(account_id=police.account_id).first()
    if not account or not check_password_hash(account.password_hash,
password):
        return jsonify({"message": "Invalid credentials"}), 401
```

```
    return jsonify({"message": "Police login successful"}), 200


@app.route("/api/civilian/complaint", methods=["POST"])
def register_complaint():
    data = request.json
    description = data.get('description')
    location = data.get('location')
    incident_date = data.get('incident_date')
    civilian_id = data.get('civilian_id')

    if not description or not location or not incident_date or not
civilian_id:
        return jsonify({'message': 'All fields are required'}), 400

    civilian = Civilian.query.get(civilian_id)
    if not civilian:
        return jsonify({'message': 'Civilian not found'}), 404

    incident = Incident(description=description, location=location,
incident_date=incident_date, civilian_id=civilian_id)
    db.session.add(incident)
    db.session.commit()

    return jsonify({'message': 'Complaint registered successfully'}), 201


if _name_ == "_main_":
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```