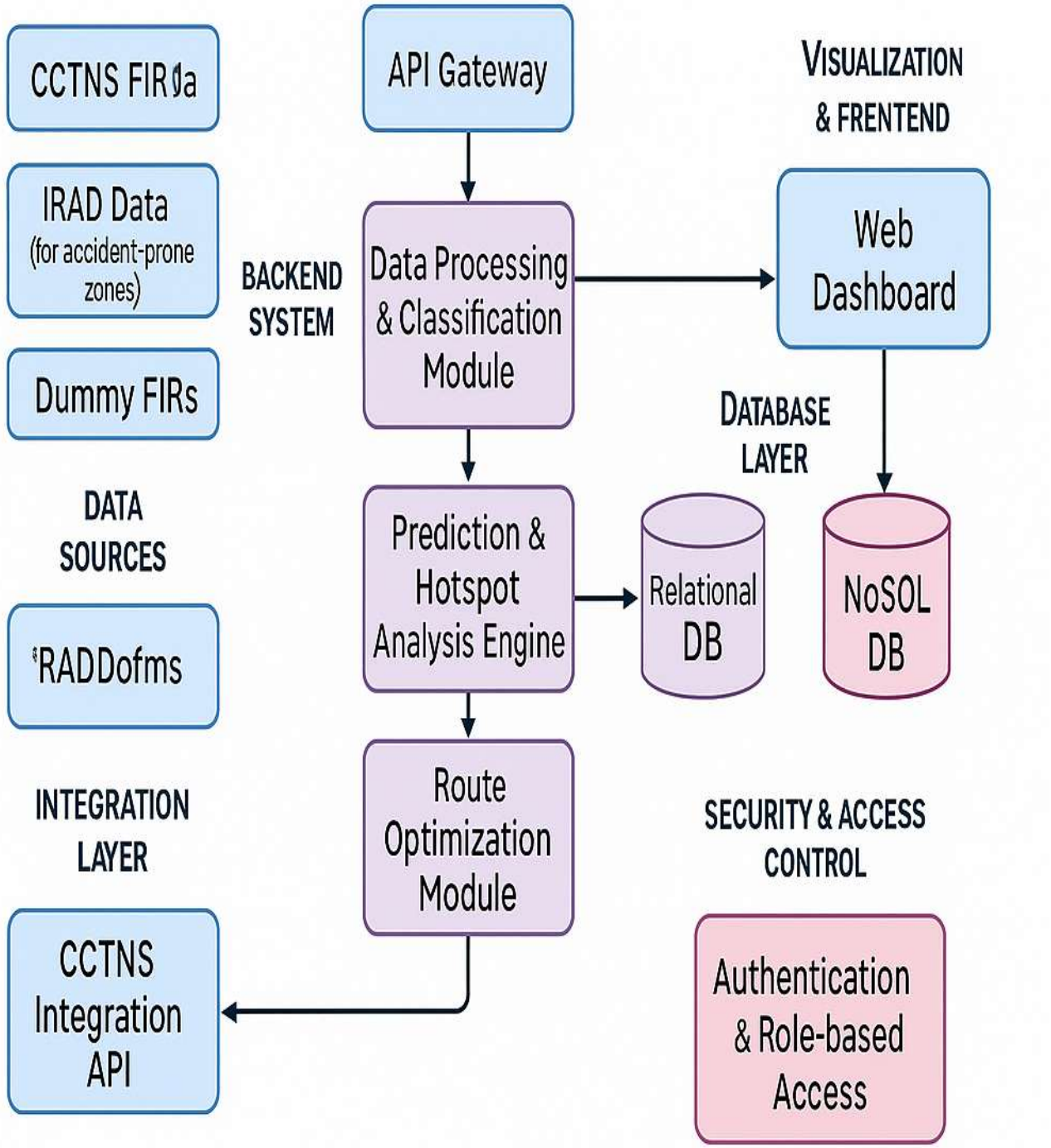
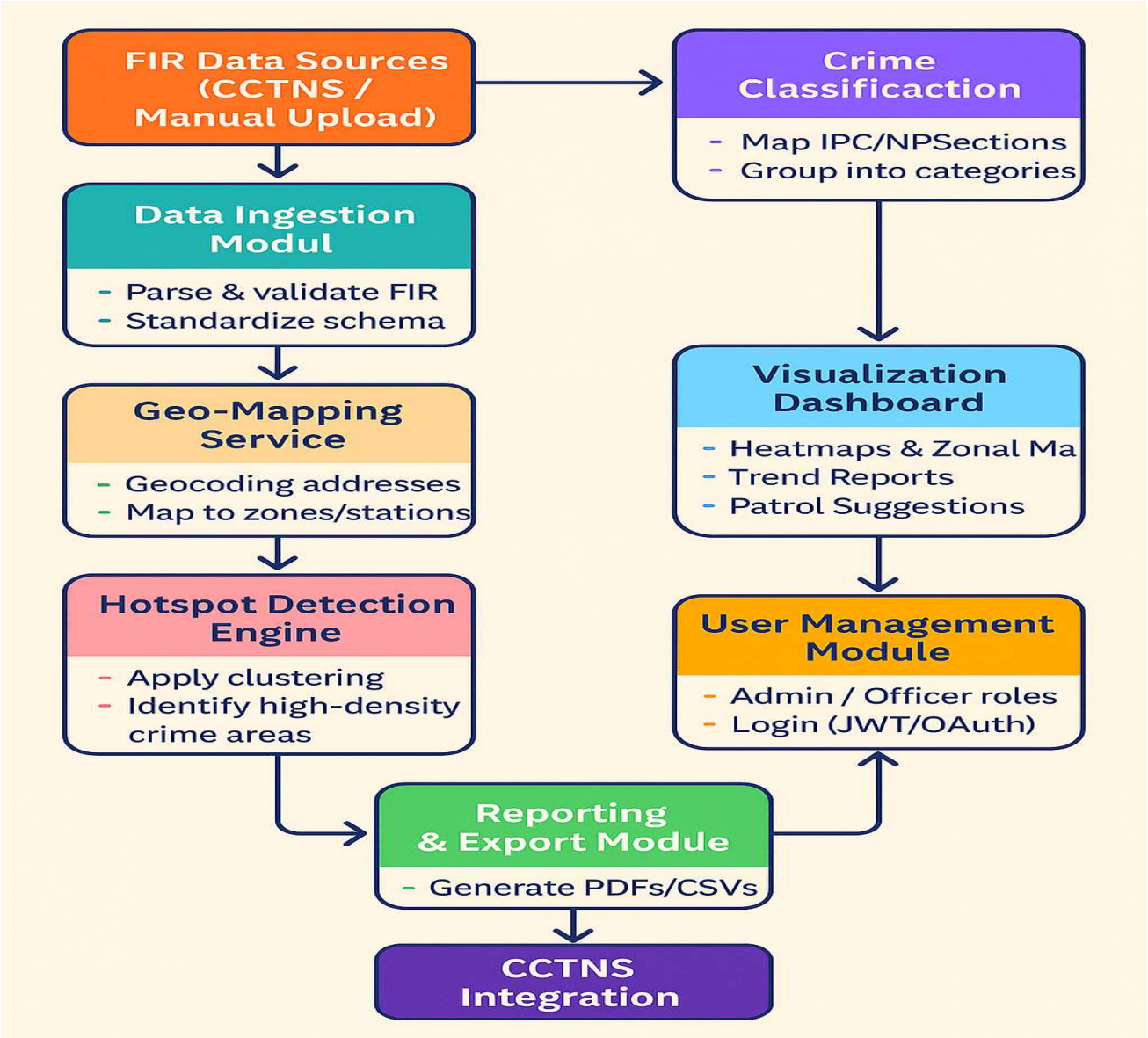
CRIME PREDICTIVE MODEL/TOOL FOR HOTSPOT MAPPING

High-Level System Architecture:



Step 1: Detailed Requirements Analysis:



1.1 Objectives

- Build a predictive model to identify crime-prone areas (hotspots) based on FIR data.
- Visualize crime data as heatmaps and zonal maps.
- Integrate with CCTNS for real-time data updates.
- Enable police officers and analysts to plan patrols and interventions.

1.2 Key Functional Requirements

1. **Data Ingestion**
   - Receive FIR data (from CCTNS or manually uploaded).
   - FIR fields: case ID, crime type (IPC/NDPS), date, time, location, police station, description.
2. **Crime Classification**
   - Classify FIRs based on legal codes (e.g., IPC Sections, NDPS Act).
   - Group crimes into categories (e.g., violent, property, narcotics).

---

3. **Geo-mapping**
    - Convert textual addresses to geo-coordinates.
    - Overlay police station boundaries and jurisdiction zones.
4. **Hotspot Detection**
    - Apply clustering algorithms to identify high-density crime zones.
    - Visualize intensity with colors (heatmaps).
5. **Visualization Dashboard**
    - Interactive map with filters (crime type, date range, zone).
    - Patrol route recommendations.
    - Crime trend reports (daily, weekly, monthly).
6. **CCTNS Integration**
    - Real-time or scheduled data pull from CCTNS.
    - Ensure latest FIRs are processed automatically.
7. **User Management & Security**
    - Roles: Admin, Officer, Analyst.
    - Secure login (JWT, OAuth).
    - Access control based on role.
8. **Reporting & Export**
    - Generate reports for high-crime areas.
    - Export heatmaps, zone reports as PDF/CSV.

1.3 Non-Functional Requirements

- **Performance**: Real-time updates of maps as new FIRs arrive.
- **Scalability**: Handle large datasets (thousands of FIRs).
- **Security**: Data encryption, role-based access, audit logs.
- **User Experience**: Simple UI for officers with filters & map interactions.

1.4 Stakeholders

- **Police Department**: End-users (officers, analysts).
- **IT Team**: Developers & system administrators.
- **Legal Department**: For FIR data compliance.

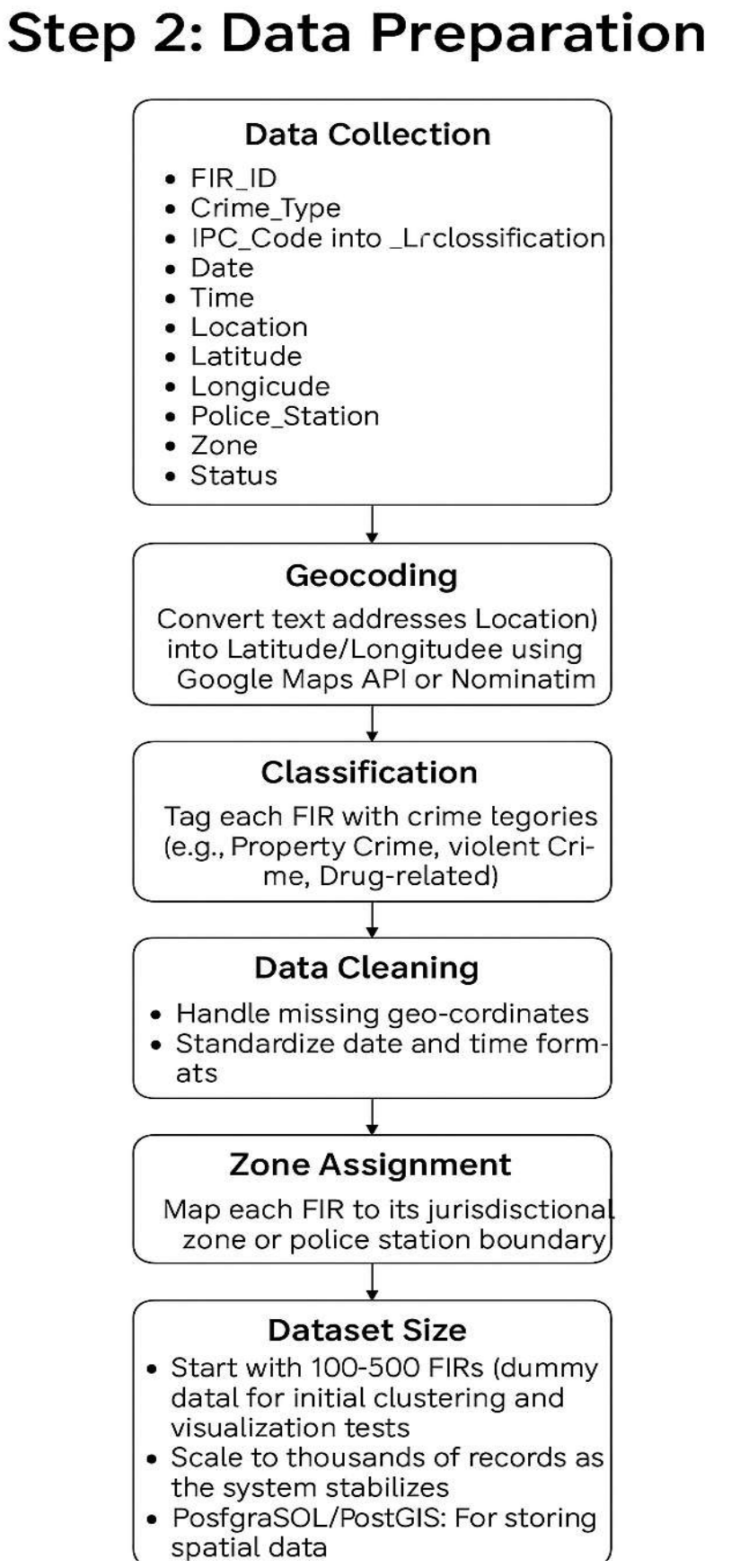
1.5 Constraints

- Data privacy laws (e.g., ensuring FIR data is securely stored).
- Limited access to live CCTNS for development (may use mock data initially).

1.6 Deliverables

- Functional web tool with heatmaps and zone analysis.
- APIs for CCTNS integration.
- User roles and access controls.
- Optional mobile app (future phase).

Step 2: <u>Data Preparation</u>:



2.1 Data Collection

For development and testing, we'll prepare **sample FIR datasets** with fields reflecting the structure of real CCTNS FIRs.

Sample FIR Data Fields

Field	Description
FIR_ID	Unique identifier
Crime_Type	IPC/NDPS/Arms Act classification
IPC_Code	Legal code of the offense
Date	Date of incident
Time	Time of incident
Location	Text address
Latitude	Latitude (geo-coordinate)
Longitude	Longitude (geo-coordinate)
Police_Station	Name of the reporting station
Description	Short description of the incident
Zone	Administrative area (optional)
Status	Open/Closed FIR

2.2 Sample FIR Data Example

FIR_ID	Crime_Type	IPC_Code	Date	Time	Location	Latitude	Longitude	Police_Station	Description	Zone	Status
FIR 001	Theft	379	2025-05-01	14:30	123 Market St, City Center	28.6139	77.2090	Connaught Place PS	Mobile phone snatched	Central	Open
FIR 002	Assault	323	2025-05-02	22:10	45 Block A, Green Park	28.5562	77.2005	Green Park PS	Street fight between two individuals	South	Open
FIR 003	NDPS	21	2025-05-03	18:45	Near Metro Station, Rohini	28.7359	77.1320	Rohini Sector 18 PS	Suspected drug dealing	North-West	Closed
FIR 004	Burglary	457	2025-05-04	03:20	8, Industrial Area	28.6200	77.3000	Okhla PS	Warehouse break-in reported	South-East	Open

2.3 Data Processing

1. **Geocoding**: Convert text addresses (Location) into Latitude/Longitude using Google Maps API or Nominatim.
2. **Classification**: Tag each FIR with crime categories (e.g., Property Crime, Violent Crime, Drug-related).
3. **Data Cleaning**:
    - Handle missing geo-coordinates.
    - Standardize date and time formats.
4. **Zone Assignment**: Map each FIR to its jurisdictional zone or police station boundary.

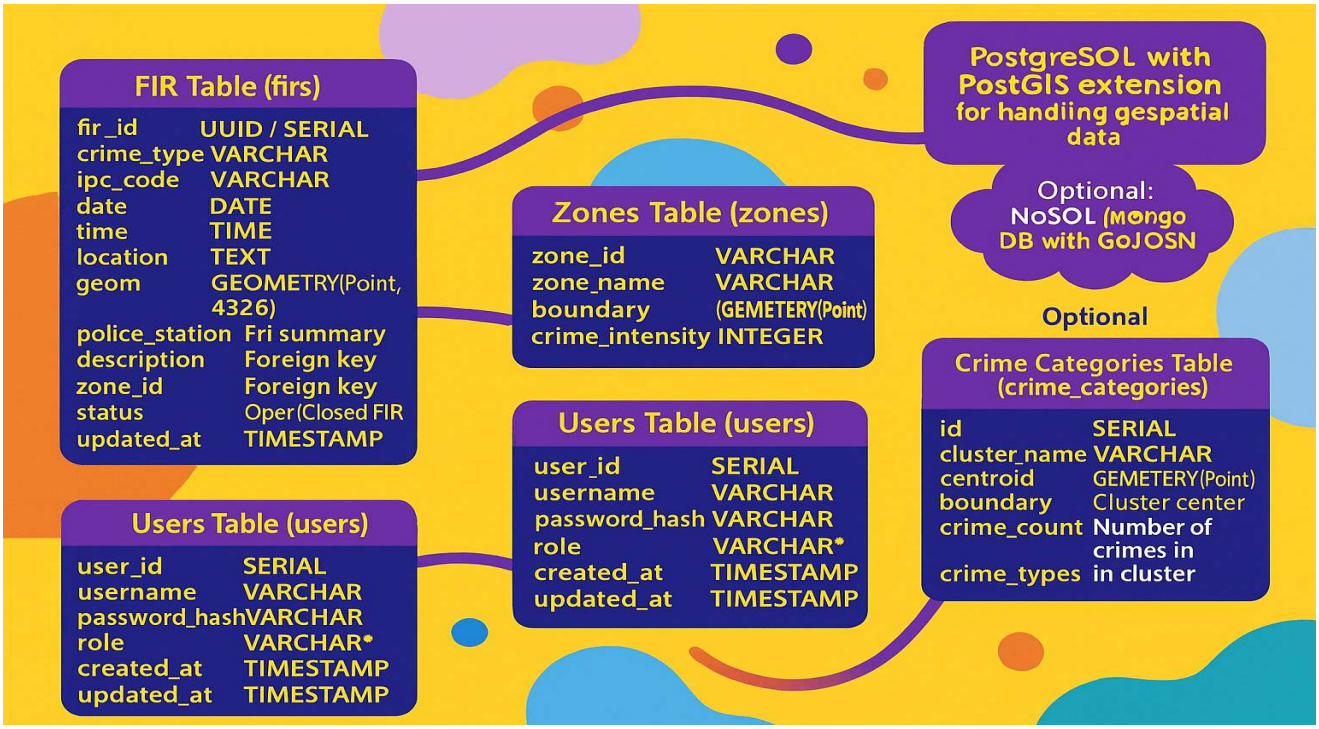
2.4 Dataset Size

- Start with **100–500 FIRs** (dummy data) for initial clustering and visualization tests.
- Scale to **thousands of records** as the system stabilizes.

2.5 Tools for Data Preparation

- **Excel/CSV**: For initial sample dataset.
- **Python/Pandas**: For cleaning, transforming, and geocoding addresses.
- **PostgreSQL/PostGIS**: For storing spatial data.

Step 3: Database Schema Design:



3.1 Database Type

- **Relational Database**: PostgreSQL with PostGIS extension for handling geospatial data.
- Optional: NoSQL (MongoDB with GeoJSON) for scalability.

3.2 Key Tables

3.2.1 FIR Table (firs)

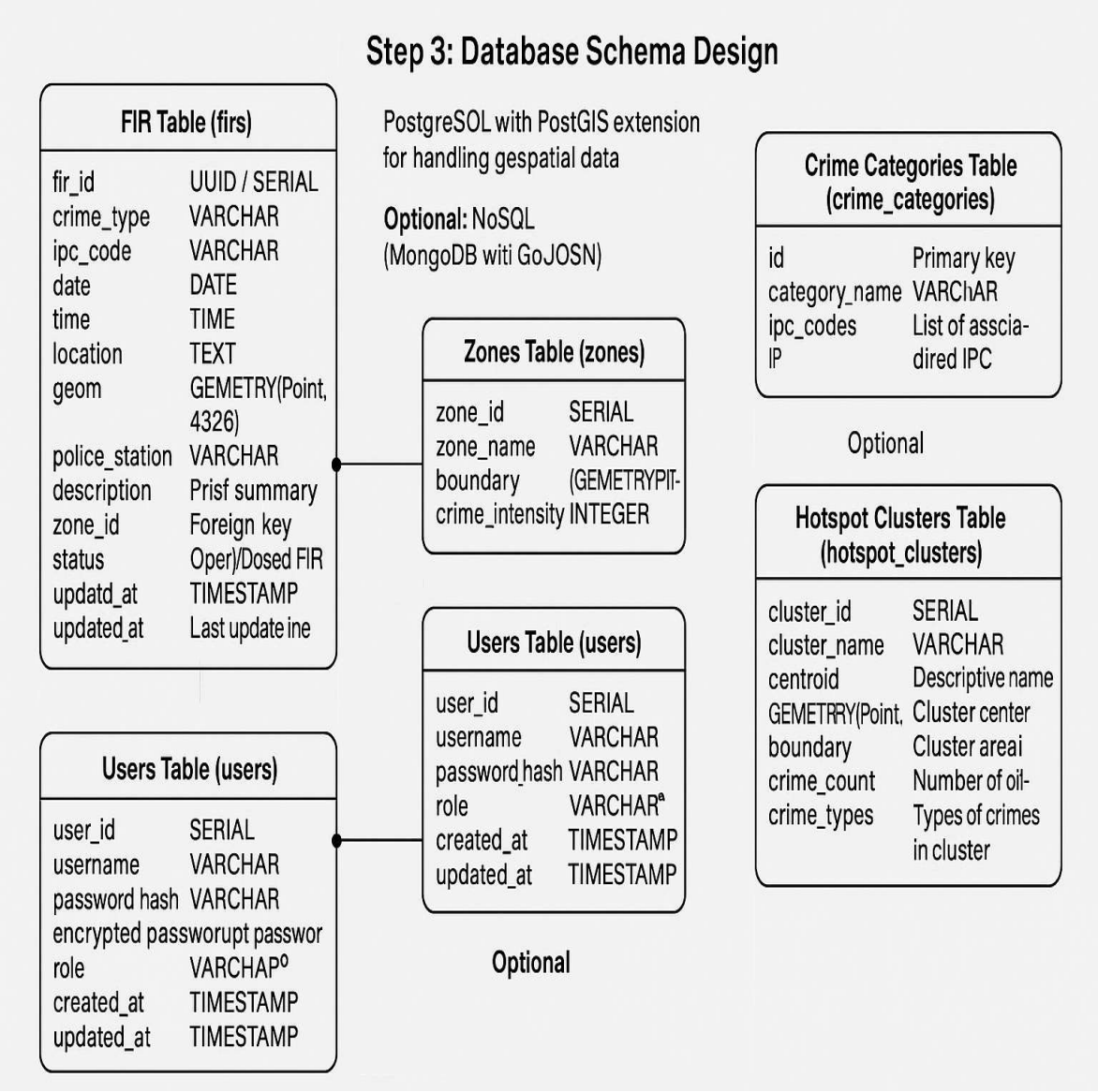
Field	Type	Description
fir_id	UUID / SERIAL	Unique FIR identifier
crime_type	VARCHAR	Crime category (e.g., Theft, Assault)
ipc_code	VARCHAR	Legal code (e.g., 379)
date	DATE	Date of crime
time	TIME	Time of crime
location	TEXT	Address
geom	GEOMETRY(Point, 4326)	Geospatial location (longitude/latitude)
police_station	VARCHAR	Reporting police station
description	TEXT	Brief summary
zone_id	INTEGER	Foreign key (link to zones)
status	VARCHAR	Open/Closed FIR
created_at	TIMESTAMP	Record creation time
updated_at	TIMESTAMP	Last update time

3.2.2 Crime Categories Table (crime_categories)

Field	Type	Description
id	SERIAL	Primary key
category_name	VARCHAR	E.g., Violent Crime, Property Crime
ipc_codes	TEXT	List of associated IPC codes

3.2.3 Zones Table (zones)

Field	Type	Description
zone_id	SERIAL	Primary key
zone_name	VARCHAR	Name of the zone
boundary	GEOMETRY(Polygon, 4326)	Polygon representing the zone
crime_intensity	INTEGER	Optional: crime count or intensity metric



3.2.4 Users Table (users)

Field	Type	Description
user_id	SERIAL	Primary key
username	VARCHAR	User login
password_hash	VARCHAR	Encrypted password
role	VARCHAR	Admin, Officer, Analyst
created_at	TIMESTAMP	Creation time
updated_at	TIMESTAMP	Last update time

3.2.5 Audit Logs Table (audit_logs)

Field	Type	Description
log_id	SERIAL	Primary key
user_id	INTEGER	Who performed the action
action	TEXT	Description of action
timestamp	TIMESTAMP	When it occurred

3.3 Relationships

- firs.zone_id → zones.zone_id: FIRs are linked to a zone.
- firs.crime_type → crime_categories.category_name (optional): Can standardize classification.
- audit_logs.user_id → users.user_id: Tracks user actions.

3.4 Special Considerations

- **Geospatial Indexing**: Create GIST index on firs.geom and zones.boundary for fast location queries.
- **Time-Series Indexing**: Index firs.date and firs.time for filtering.
- **Security**: Store passwords hashed (bcrypt/argon2).

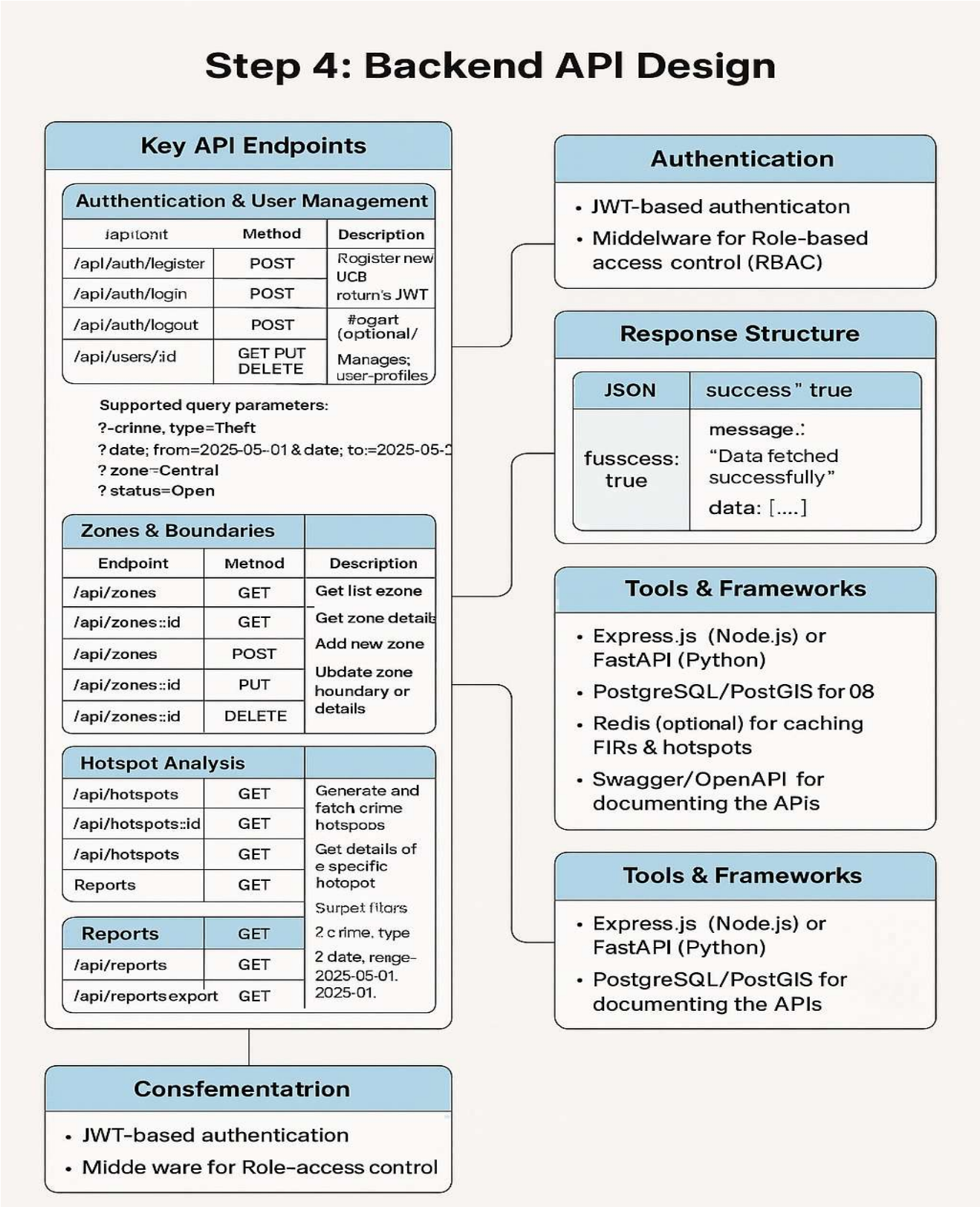
3.5 Optional: Clustered Tables for Hotspot Analysis

You can add a table hotspot_clusters:

Field	Type	Description
cluster_id	SERIAL	Cluster identifier
cluster_name	VARCHAR	Descriptive name
centroid	GEOMETRY(Point, 4326)	Cluster center
boundary	GEOMETRY(Polygon, 4326)	Cluster area
crime_count	INTEGER	Number of crimes in cluster
crime_types	TEXT[]	Types of crimes in cluster

Step 4: Backend API Design:

Our backend will use a framework like **Node.js (Express)** or **Python (FastAPI/Django)**, with PostgreSQL/PostGIS as the database.



4.1 Key API Endpoints

4.1.1 Authentication & User Management

Endpoint	Method	Description
/api/auth/register	POST	Register new user
/api/auth/login	POST	User login, returns JWT
/api/auth/logout	POST	Logout (optional)
/api/users	GET	Get list of users (Admin only)
/api/users/:id	GET/PUT/DELETE	Manage user profiles

4.1.2 FIR Management

Endpoint	Method	Description
/api/firs	GET	Fetch list of FIRs (supports filtering)
/api/firs/:id	GET	Get FIR details
/api/firs	POST	Create new FIR (Admin/Officer)
/api/firs/:id	PUT	Update FIR
/api/firs/:id	DELETE	Delete FIR (Admin)

Query parameters for /api/firs:

- ?crime_type=Theft
- ?date_from=2025-05-01&date_to=2025-05-31
- ?zone=Central
- ?status=Open

4.1.3 Zones & Boundaries

Endpoint	Method	Description
/api/zones	GET	Get list of zones
/api/zones/:id	GET	Get zone details
/api/zones	POST	Add new zone
/api/zones/:id	PUT	Update zone boundary or details
/api/zones/:id	DELETE	Delete zone

4.1.4 Hotspot Analysis

Endpoint	Method	Description
/api/hotspots	GET	Generate and fetch crime hotspots
/api/hotspots/:id	GET	Get details of a specific hotspot
/api/hotspots/export	GET	Export hotspot data as PDF/CSV

Supports filters:

- ?crime_type=Violent
- ?date_range=2025-05-01,2025-05-31

4.1.5 CCTNS Data Integration

Endpoint	Method	Description
/api/cctns/import	POST	Import FIRs from CCTNS (batch)
/api/cctns/sync	POST	Trigger real-time sync with CCTNS

4.1.6 Reports

Endpoint	Method	Description
/api/reports	GET	Generate summary reports
/api/reports/export	GET	Export reports as PDF/Excel

4.2 Authentication

- JWT-based authentication.
- Middleware for **Role-based access control (RBAC)**.

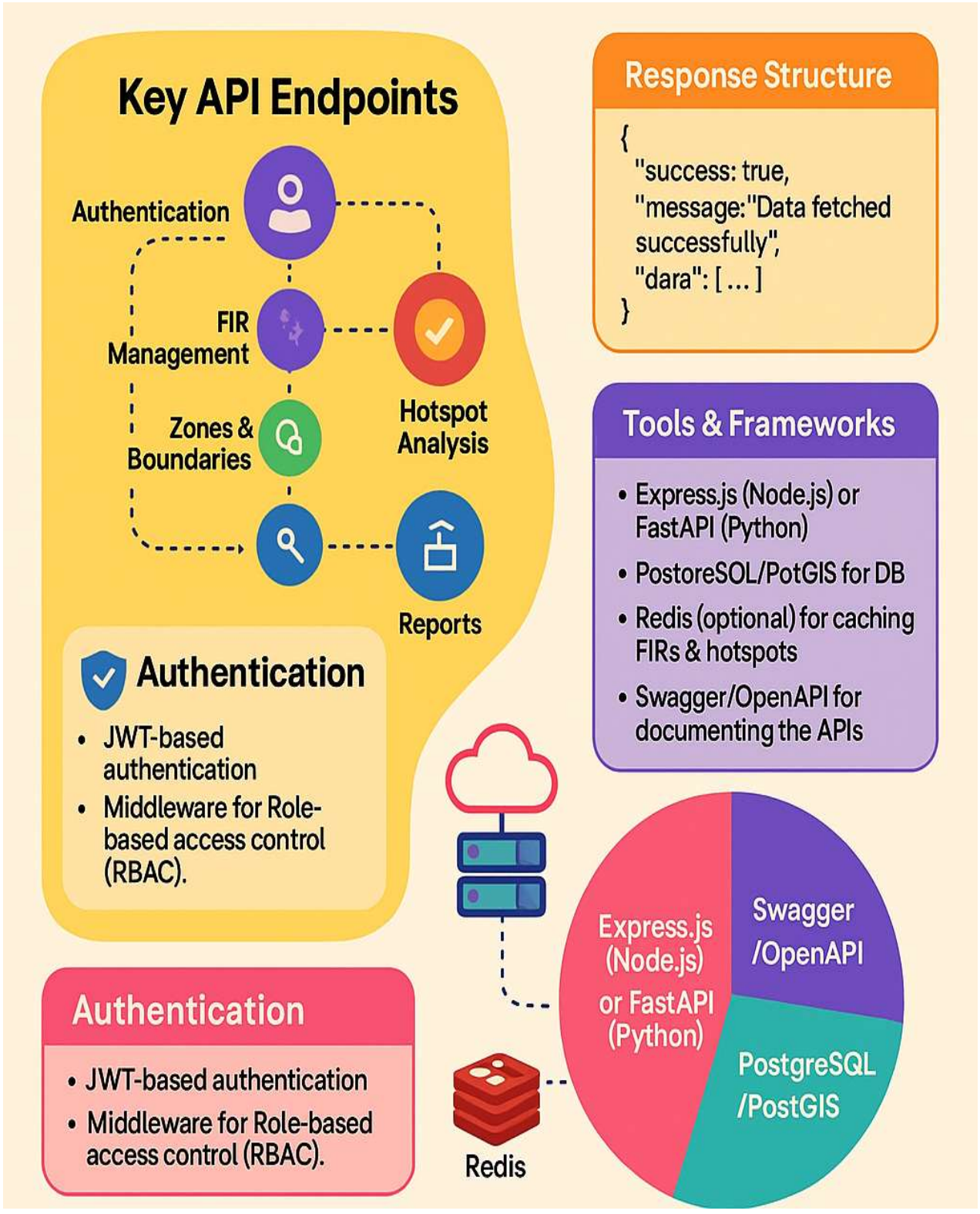
4.3 Response Structure

[Json]

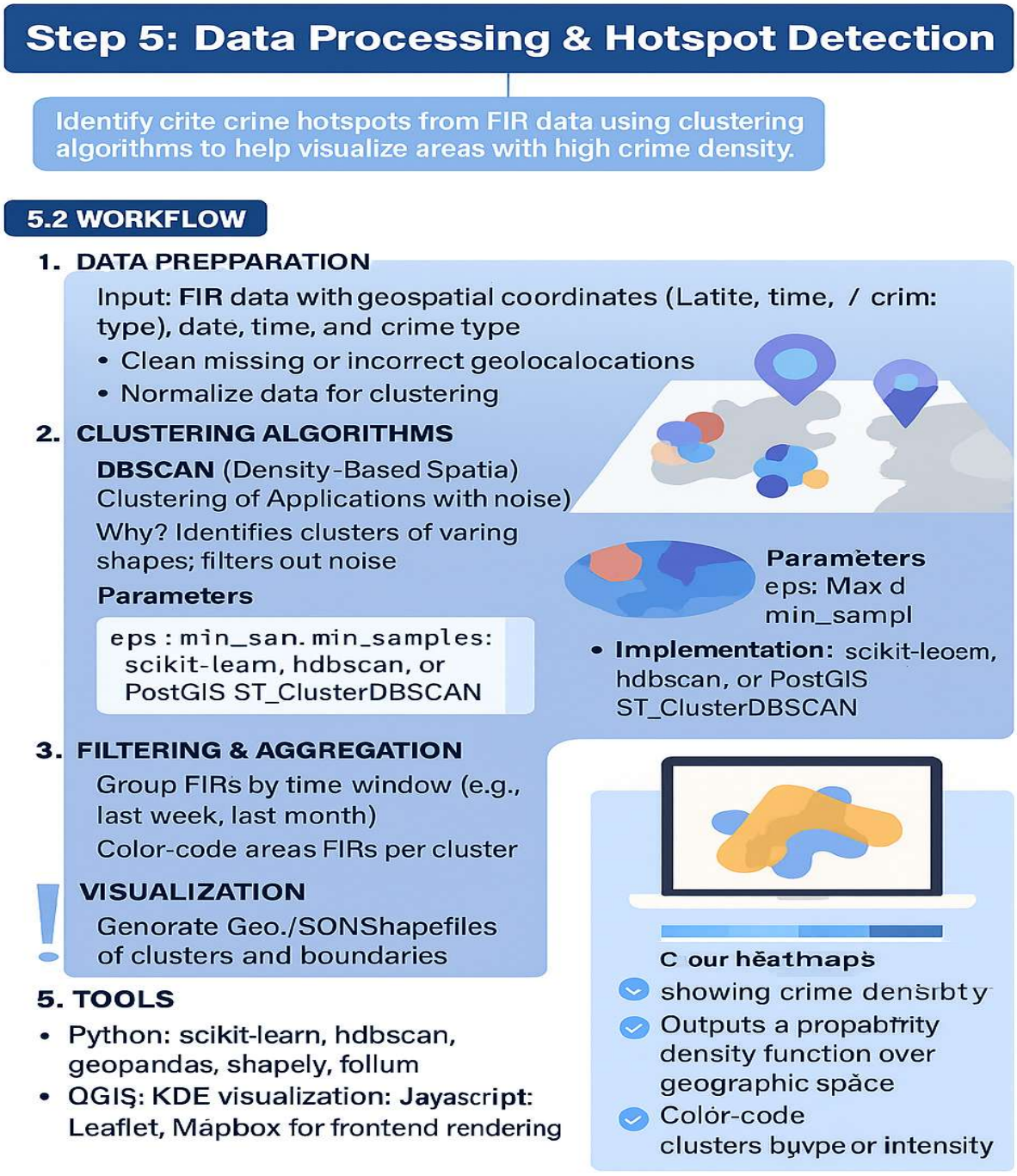
```
{
  "success": true,
  "message": "Data fetched successfully",
  "data": [ ... ]
}
```

4.4 Tools & Frameworks

- **Express.js (Node.js)** or **FastAPI (Python)**.
- **PostgreSQL/PostGIS** for DB.
- **Redis** (optional) for caching FIRs & hotspots.
- **Swagger/OpenAPI** for documenting the APIs.


Key API Endpoints
Authentication
FIR Management
Zones & Boundaries
Hotspot Analysis
Reports
Response Structure
{
"success: true,
"message:"Data fetched successfully",
"dara": [...]
}
Tools & Frameworks
• Express.js (Node.js) or FastAPI (Python)
• PostoreSOL/PotGIS for DB
• Redis (optional) for caching FIRs & hotspots
• Swagger/OpenAPI for documenting the APIs
Authentication
• JWT-based authentication
• Middleware for Role-based access control (RBAC).
Authentication
• JWT-based authentication
• Middleware for Role-based access control (RBAC).
Redis
Express.js (Node.js) or FastAPI (Python)
Swagger /OpenAPI
PostgreSQL /PostGIS

Step 5: Data Processing & Hotspot Detection:



5.1 Objective

Identify **crime hotspots** from FIR data using clustering algorithms to help visualize areas with high crime density.

5.2 Workflow

5.2.1 Data Preparation

- Input: FIR data with geospatial coordinates (Latitude, Longitude), date, time, and crime type.
- Clean missing or incorrect geolocations.
- Normalize data for clustering (e.g., convert coordinates to a projection system for distance).

5.2.2 Clustering Algorithms

A. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- **Why?** Identifies clusters of varying shapes, filters out noise (outlier FIRs).
- **Parameters**:
  - eps: Max distance between two samples for them to be considered in the same cluster.
  - min_samples: Minimum number of FIRs to form a cluster.
- **Implementation**: scikit-learn, hdbscan, or PostGIS ST_ClusterDBSCAN.

B. K-Means Clustering (Optional)

- Groups FIRs into **K fixed clusters** (less flexible for hotspot mapping).
- Requires specifying K (number of clusters).

C. Kernel Density Estimation (KDE)

- Creates **heatmaps** showing crime density.
- Outputs a **probability density function** over geographic space.
- Tools: QGIS, Python with seaborn, scipy, or PostGIS.

5.2.3 PostGIS-Based Clustering (Optional)

If you're using **PostgreSQL/PostGIS**, you can run clustering SQL:

[Sql]

```
SELECT ST_ClusterDBSCAN(geom, eps := 0.001, minpoints := 3) OVER () AS cluster_id, *
FROM firs;
```

5.2.4 Time-Based Hotspots (Optional)

- Partition FIRs by **time window** (e.g., last week, last month).
- Compare clusters over time to detect **emerging hotspots**.

5.2.5 Filtering & Aggregation

- Group FIRs by **crime type**, **zone**, or **date range**.
- Count number of FIRs per cluster.
- Calculate **cluster centroids** and **boundaries (convex hulls)**.

5.2.6 Visualization

- Generate **GeoJSON/Shapefiles** of clusters and boundaries.
- Render on **Leaflet.js**, **Mapbox**, or **Google Maps**.
- Color-code clusters by **crime type** or **intensity**.

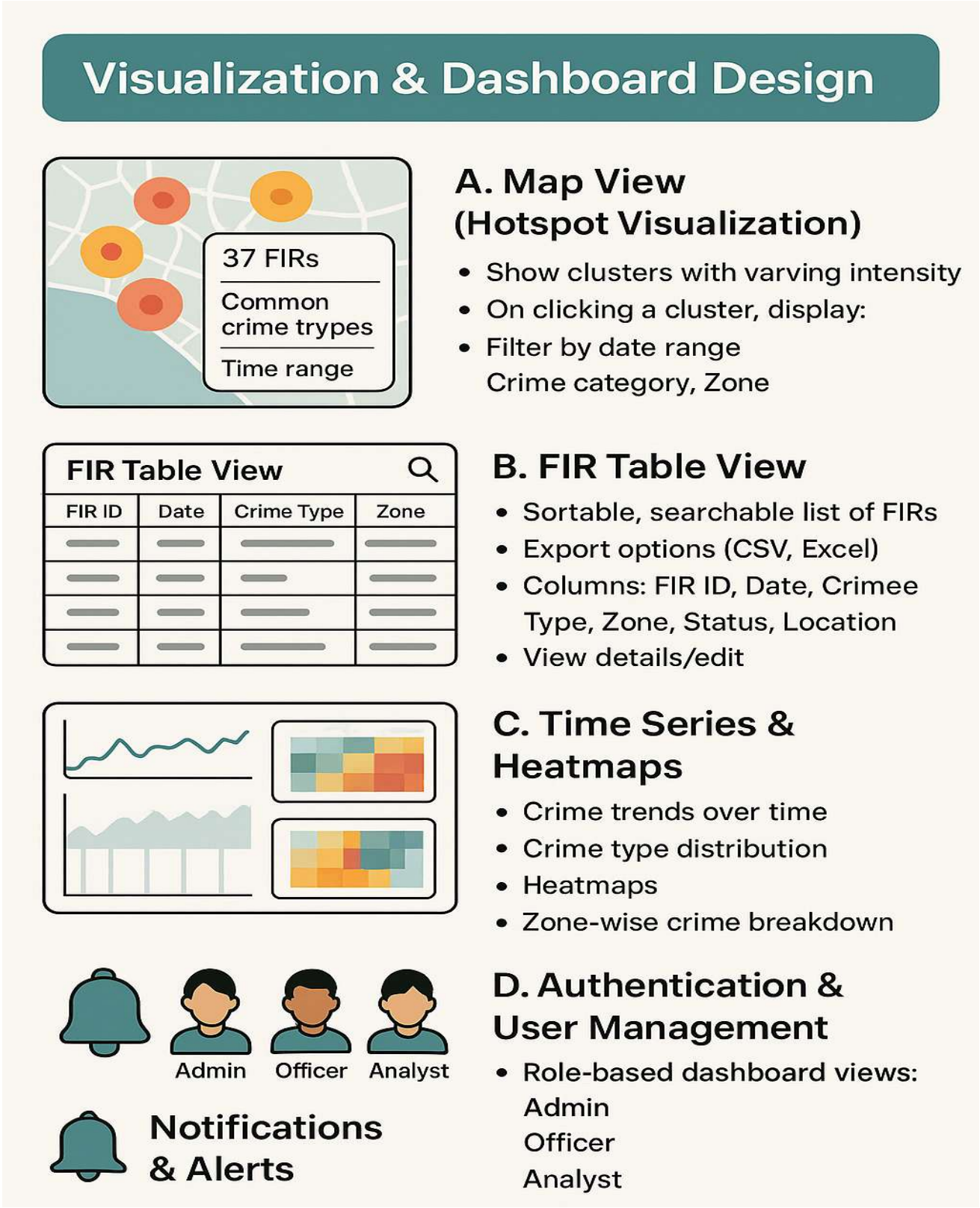
5.3 Tools

- **Python**: scikit-learn, hdbscan, geopandas, shapely, folium.
- **PostgreSQL/PostGIS**: Geospatial queries, clustering.
- **QGIS**: KDE visualization.
- **JavaScript**: Leaflet, Mapbox for frontend rendering.

Step 6: Visualization & Dashboard Design:

6.1 Objective

Create a **user-friendly dashboard** that visualizes crime data, FIRs, clusters (hotspots), and trends over time, enabling police officers, analysts, and administrators to make informed decisions.



6.2 Key Components of the Dashboard

A. Map View (Hotspot Visualization)

- **Technology**: Leaflet.js, Mapbox GL, Google Maps, or OpenLayers.
- **Features**:
    - Show **clusters** with varying intensity (color-coded).
    - On clicking a cluster, display:
        - Number of FIRs
        - Common crime types
        - Time range of reported crimes
    - Filter by:
        - Date range
        - Crime category (e.g., theft, assault)
        - Zone

B. FIR Table View

- **Technology**: DataTables.js, React Table, or equivalent.
- **Features**:
    - Sortable, searchable list of FIRs
    - Export options (CSV, Excel)
    - Columns: FIR ID, Date, Crime Type, Zone, Status, Location
    - View details/edit (role-based access)

C. Time Series & Heatmaps

- **Charts**: Line charts, bar charts, area charts (using Chart.js, Recharts, Highcharts).
- **Metrics**:
    - Crime trends over time (daily/weekly/monthly)
    - Crime type distribution (pie/donut)
    - Heatmaps for intensity over geography and time
    - Zone-wise crime breakdown

D. Authentication & User Management

- **Role-based dashboard views**:
    - Admin: Full control over users and settings.
    - Officer: Can manage FIRs.
    - Analyst: View analytics and trends.
- **Login/Logout** (JWT-based).

E. Notifications & Alerts

- Optional feature:
    - Push notifications for new FIRs in a hotspot zone.
    - Email alerts for emerging crime patterns.

6.3 High-Level Tech Stack

- **Frontend**: React.js, Tailwind CSS, shadcn/ui, Recharts, Leaflet/Mapbox.
- **Backend**: Node.js/FastAPI + PostgreSQL/PostGIS + Redis (optional).
- **API**: RESTful APIs with JWT-based authentication.
- **Deployment**: Docker, Nginx, AWS/GCP/Azure.

Step 7: Deployment Plan:

This step focuses on setting up the system to be **publicly accessible**, **secure**, and **scalable**, with options for **future expansion**.

DEPLOYMENT PLAN

Hosting Options

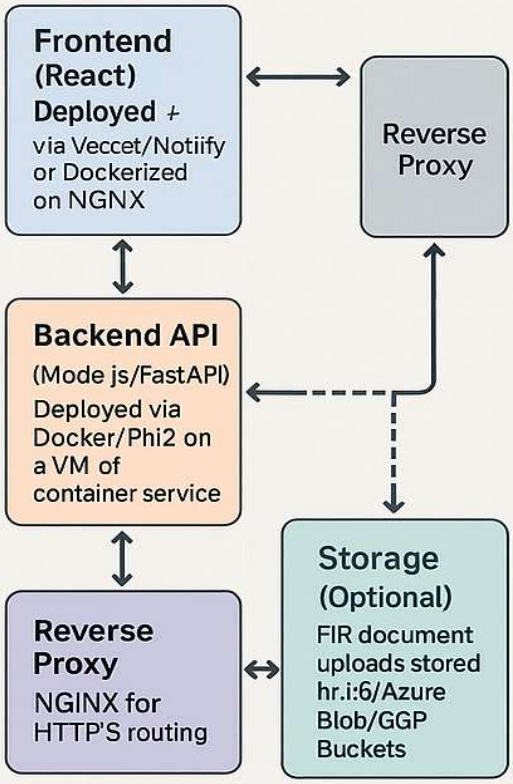
A Cloud Providers

- AWS: EC2/app serve)RDS (PostgreSQL/S3, S3 (filste storage, CloudFront (CDN)
- Azure: App-Services, Azure DB for PostgreSQL, Azure Maps
- GCP: Compute Engine Cloud SQL: Firebase Hosting

B Alternative (budget-friendly)

- Render, Railway, Fly io, DigitalOcean; Heroku

Deployment Architecture



Deployment Steps

1 · Environment Setup

- Provisionon cloud resources (VAI, DB, Storage)
- Install Docker, Node B/thon, NGINX

2 · CI/CD Pipeline

- GitHub → Actions, / GitLab CI / Render Auto Deploy
- Auto deploy on push to main brnch

3 Frontend Deployment

- Option 1: Vercel/Netlify (React frontend)
- Option 2: Serve with NGINX in Docker

4 · Backend Deployment

- Dockerize FastAPI/Node bockend
- Expose API on port 443 via NGINX reverse proxy

5 Database Setup

- Deploy PastAPI + enable PostGIS extension
- Apply schema using SQLAlcherny/migrations

6 Domain + HTTPS

- Buy a domain or use existting (ℓ.g.) crimewatch in)
- Use Let's Encrypt via Certbot for free SSL

Security Best Practices

- Use HTTPS everywhere
- JWT for API authentication
- Rate limiting & CORS setup
- PostgreSQL secured with user roles & IP-whitelisting
- Auto-backups for the database

7.1 Hosting Options

A. Cloud Providers

- **AWS**: EC2 (app server), RDS (PostgreSQL/PostGIS), S3 (file storage), CloudFront (CDN)
- **Azure**: App Services, Azure DB for PostgreSQL, Azure Maps
- **GCP**: Compute Engine, Cloud SQL, Firebase Hosting

B. Alternative (Budget-Friendly)

- **Render**, **Railway**, **Fly.io**, **DigitalOcean**, **Heroku**

7.2 Deployment Architecture

Components:

- **Frontend (React)** → Deployed via Vercel/Netlify or Dockerized on NGINX
- **Backend API (Node.js/FastAPI)** → Deployed via Docker/PM2 on a VM or container service
- **Database (PostgreSQL + PostGIS)** → Hosted on RDS/Azure DB/Cloud SQL or local server
- **Storage (Optional)** → FIR document uploads stored in S3/Azure Blob/GCP Buckets
- **Reverse Proxy** → NGINX for HTTPS & routing

7.3 Deployment Steps

Step 1: Environment Setup

- Provision cloud resources (VM, DB, Storage)
- Install Docker, Node.js/Python, NGINX

Step 2: CI/CD Pipeline

- GitHub → GitHub Actions / GitLab CI / Render Auto Deploy
- Auto deploy on push to main branch

Step 3: Frontend Deployment

- Option 1: Vercel/Netlify (React frontend)
- Option 2: Serve with NGINX in Docker

Step 4: Backend Deployment

- Dockerize FastAPI/Node backend
- Expose API on port 443 via NGINX reverse proxy

Step 5: Database Setup

- Deploy PostgreSQL + enable PostGIS extension
- Apply schema using SQLAlchemy/migrations

Step 6: Domain + HTTPS

- Buy domain or use existing (e.g., crimewatch.in)
- Use Let's Encrypt via Certbot for free SSL

Step 7: Monitoring & Logging

- Tools: Prometheus, Grafana, ELK Stack, Sentry
- Setup alerts for failures, slow queries, etc.

7.4 Security Best Practices

- Use HTTPS everywhere
- JWT for API authentication
- Rate limiting & CORS setup
- PostgreSQL: Secure with user roles & IP whitelisting
- Auto-backups for the database

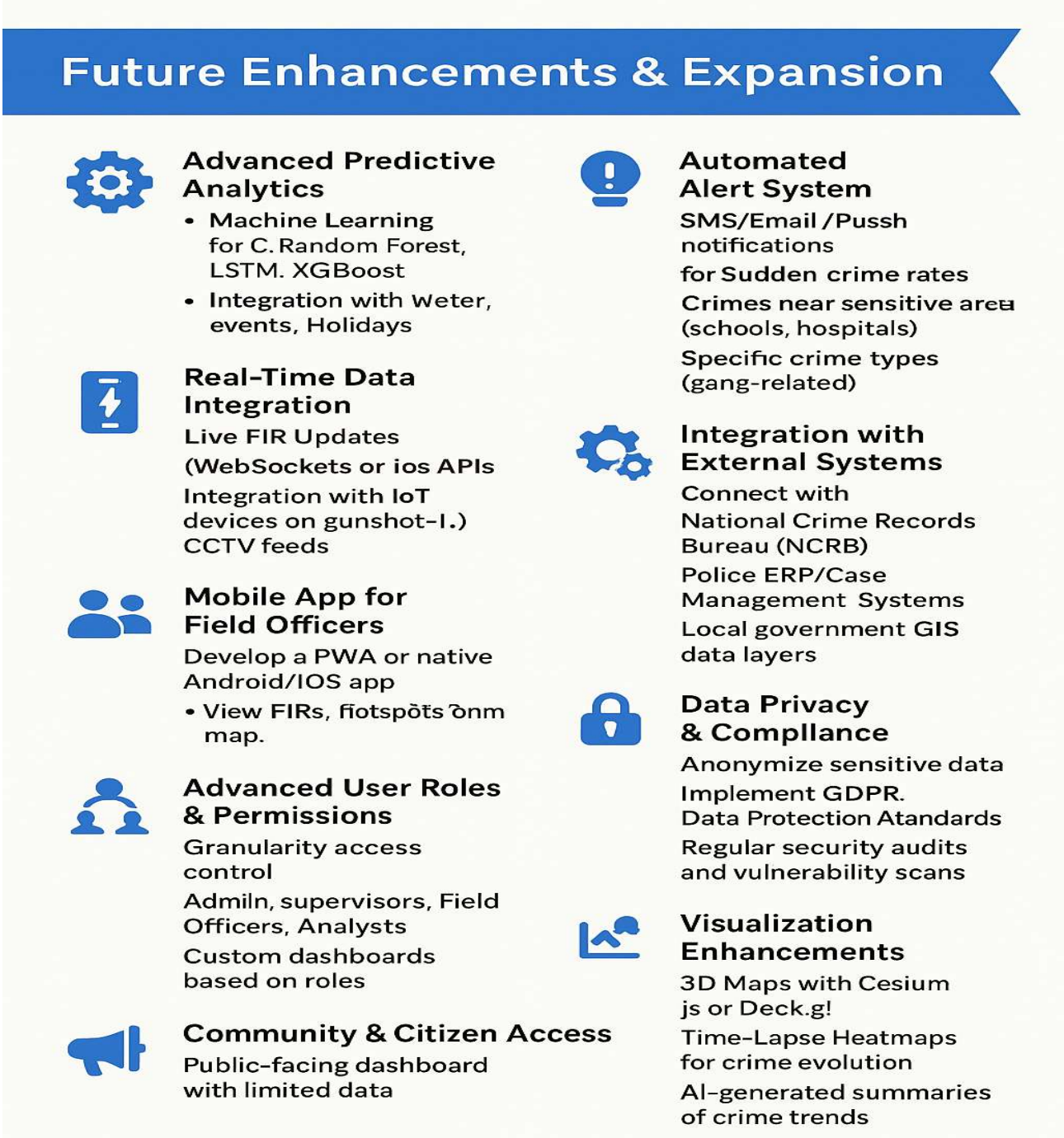
7.5 Scalability

- Use Load Balancer (Nginx or AWS ALB)
- Deploy with Docker Swarm/Kubernetes for scale
- Cache frequent queries (Redis)

7.6 Optional: Mobile Access

- Wrap the dashboard in a **PWA (Progressive Web App)** for mobile field access

Step 8: Future Enhancements & Expansion:



8.1 Advanced Predictive Analytics

- **Machine Learning for Crime Prediction**
  - Train models (e.g., Random Forest, LSTM, XGBoost) to predict:
    - Likelihood of crime in a location
    - Type of crime likely to occur
    - Temporal patterns (time of day/week)
- **Integration with Weather, Events, Holidays**
  - Correlate FIR data with weather conditions, events, and public holidays to enhance predictions.

8.2 Real-Time Data Integration

- **Live FIR Updates**
  - Stream new FIRs from police systems using WebSockets or APIs.
- **IoT & CCTV Feeds**
  - Integrate IoT devices (e.g., gunshot detectors) or CCTV systems to trigger real-time alerts.

8.3 Mobile App for Field Officers

- Develop a **PWA (Progressive Web App)** or **native Android/iOS app**:
  - View FIRs, hotspots on map.
  - Submit new FIRs directly from the field.
  - Receive notifications and alerts.

8.4 Automated Alert System

- SMS/Email/Push notifications for:
  - Sudden spikes in crime rates.
  - Crimes occurring near sensitive areas (schools, hospitals).
  - Specific crime types (e.g., gang-related).

8.5 Advanced User Roles & Permissions

- Add granular access control:
  - Admins, Supervisors, Field Officers, Analysts.
  - Custom dashboards based on roles.

8.6 Integration with External Systems

- Connect with:
  - National Crime Records Bureau (NCRB)
  - Police ERP/Case Management Systems
  - Local government GIS data layers.

8.7 Performance & Scalability

- Use **Kubernetes** for elastic scaling.
- **CDN** for faster frontend loading.
- **Database sharding & read replicas** for large-scale FIR data.

8.8 Data Privacy & Compliance

- Anonymize sensitive data.
- Implement **GDPR**, **Data Protection Act** standards.
- Regular security audits and vulnerability scans.

8.9 Visualization Enhancements

- **3D Maps** with Cesium.js or Deck.gl.
- **Time-Lapse Heatmaps** for crime evolution.
- **AI-generated summaries** of crime trends.

8.10 Community & Citizen Access (Optional)

- Public-facing dashboard with limited data.
- Citizens can report suspicious activity via app.
