

**Project Name:Crime Track-Online Crime Reporting System**

**Team ID : LTVIP2025TMID24711**

**Team Leader : G Narasimha**

**Team member : B Sandhya**

## **Crime Track - Online Crime Reporting System**

Crime Track - The Crime Reporting Platform aims to provide a comprehensive and userfriendly solution for individuals seeking to report criminal activities while also accessing vital information about various crimes. The platform is designed with an intuitive interface that allows users to quickly and securely report incidents in their area. Users can submit detailed reports, including the type of crime, location, time, and any relevant evidence or witnesses. This streamlined reporting process encourages community participation and helps law enforcement agencies receive timely information, ultimately contributing to enhanced public safety.

In addition to crime reporting, the platform serves as an educational resource, offering extensive information about common crimes, their impact on communities, and preventive measures individuals can take. Users can access articles, infographics, and resources related to different types of crimes, such as thefts, vandalism, assault, and cybercrime. This information empowers users to recognize potential threats and make informed decisions to protect themselves and their communities. The platform also features real-time updates on crime trends and statistics, allowing users to stay informed about safety concerns in their neighbourhood.

To further engage the community, the platform includes features such as discussion forums, where users can share experiences, safety tips, and advice. Users can also subscribe to alerts and notifications for specific crime categories or local areas of interest, ensuring they remain vigilant and aware. By fostering a sense of community involvement and collaboration, your crime reporting platform aims to create a safer environment where individuals feel empowered to take action against crime and contribute to the well-being of their neighbourhood.

### **Scenario Based Case Study:**

**Background :** Priya, a tech student ,went curious to know about the types of crimes and their preventions and what are the legal actions against the same.

**Problem:** Priya, a student while returning from her college, was witnessing that some unknown stranger was following her.

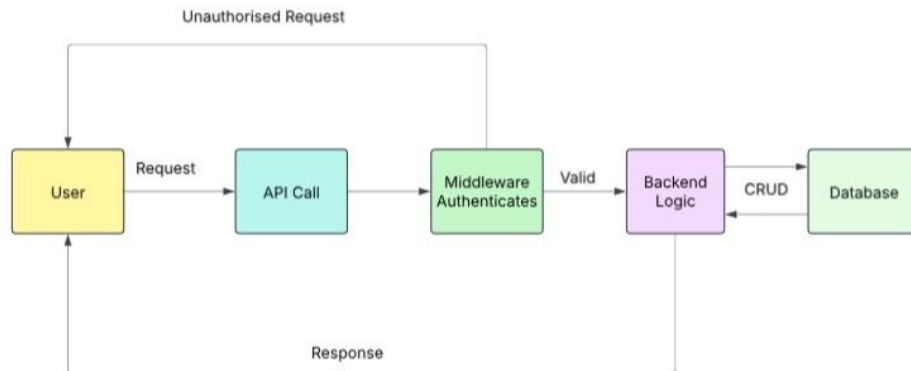
**Solution :** Priya got fed up because of the stranger so she visited Crime Track and read about the crime and reported it. The report has been taken into consideration as her problem has been solved by the police.

### **Usage:**

Customer Usage:

- o If an individual signs up ,They will see the information and a reporting form.
- o The user can access the information and get benefitted by it at the same time he/she can report the crime.

## Technical Architecture:



In this technical architecture , Crime Reporting System consist of several components:



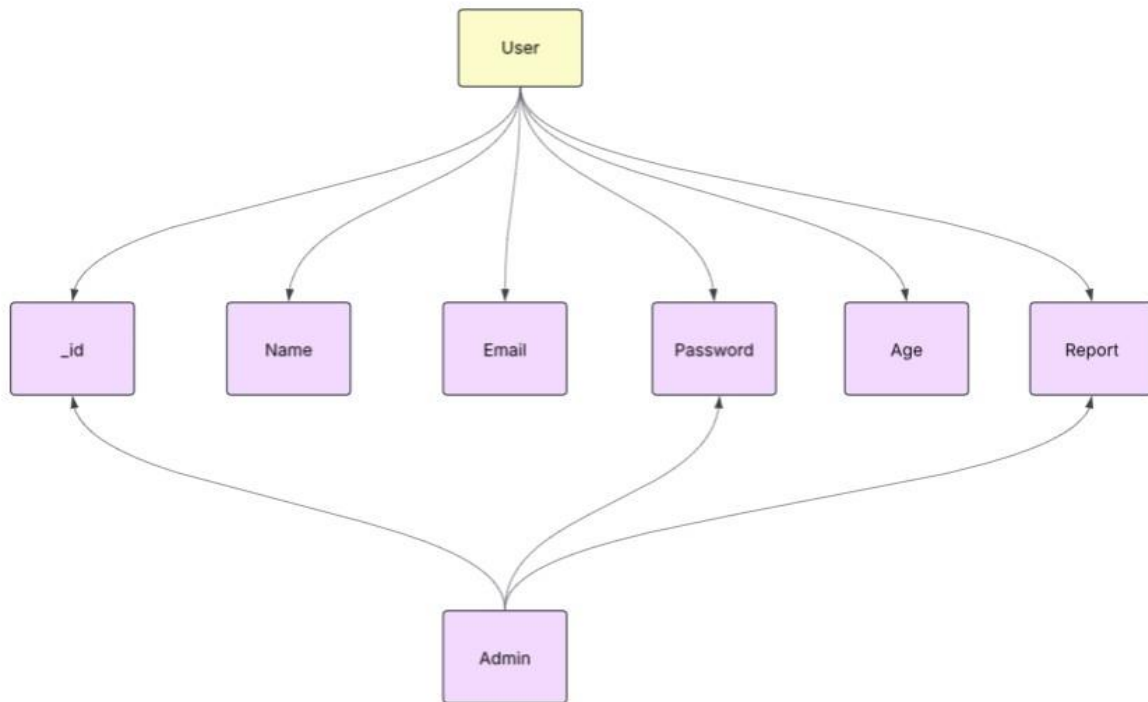
Authentication : The signup for the User and the user data will be stored in the respective collection.



Database: it uses three different collections for users, and Reports.




User component : User component consists of the few components of the crime which contains all the types of crimes, the next section is report in which the user can report the crime. **ER-Diagram**




□  
User:


.  
UserID

.  
Name

.   
Email

.   
Phone

.   
Report

.   
Age

### Key Features:

1. **User Dashboard:** A dashboard where it displays the types of crimes and their preventions and a reporting form where the crime can be reported.
2. **Report:** A Reporting form to report the crime or and uneven activity.

### PRE REQUISITES

To develop a Backend Platform using Node.js, Express.js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download: <https://nodejs.org/en/download/>

• Installation : <https://nodejs.org/en/download/package-manager/>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service. • Download:

<https://www.mongodb.com/try/download/community> • Installation instructions : <https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development. • Installation: open your command prompt or terminal and run the following command: `npm install express` **Visual Studio Code**: Download and install [Visual Studio code](#)

### **Postman or ThunderClient:**

Download and install POSTMAN to check the output ○

Use this link to download [POSTMAN](#)

You can also use Thunder Client extension in VS code

### **Roles and Responsibility**

#### **□ User:**

? Profile : The user has to sign up to access the content.

? Crime Info: Different types of crimes are listed to create awareness regarding the crimes and the prevention tips.

? Reporting Form: The reporting form is available to report the crime and the uneven activity.

**User :**



1. Start: This is the entry point of the user flow.
2. User : The user should signup/login.
3. Dashboard: The dashboard contains a Crime info section which lists the various types of crimes and the prevention and legal actions.
4. Report: The form is to report the crime

### **Project Flow**

Let's start with the project development with the help of the given activities.

### **Project Setup and Configuration**

#### **1. Install required tools and software:**

? Node.js.

? MongoDB.

## 2. Create project folders and files:

? Server folders.

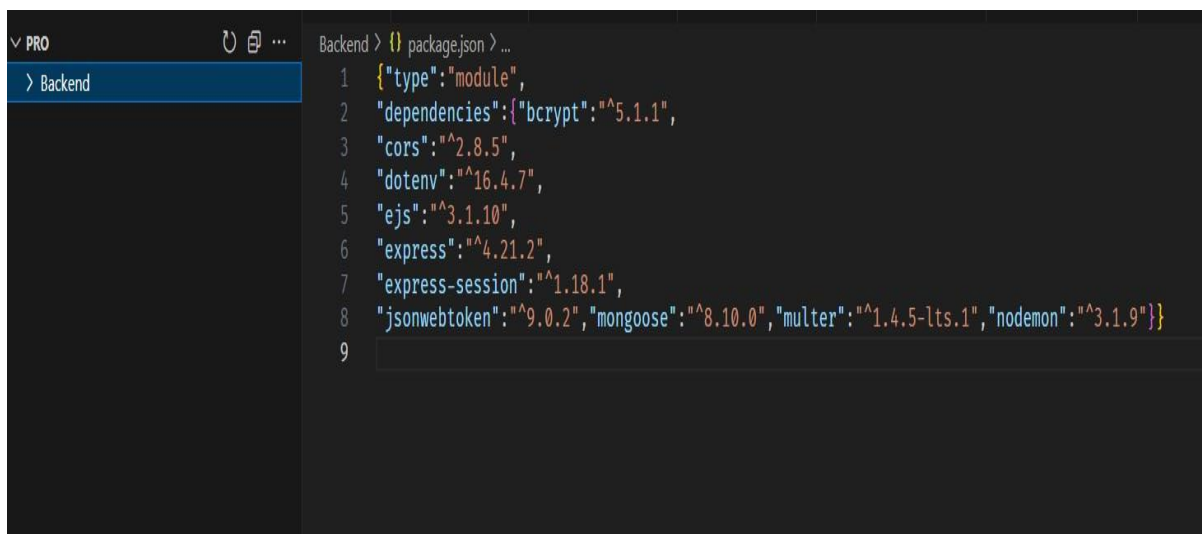
## 3. Install Packages: Backend npm Packages ? Express.

? dotenv.

? Nodemon.

? Mongoose

? JWT



```
Backend > {} package.json > ...
1  {"type": "module",
2  "dependencies": {"bcrypt": "^5.1.1",
3  "cors": "^2.8.5",
4  "dotenv": "^16.4.7",
5  "ejs": "^3.1.10",
6  "express": "^4.21.2",
7  "express-session": "^1.18.1",
8  "jsonwebtoken": "^9.0.2", "mongoose": "^8.10.0", "multer": "^1.4.5-lts.1", "nodemon": "^3.1.9"}}
9
```

## Backend Development

□

### Setup express server

- Create index.js file in the server (backend folder).
- Create a .env file and define port number to access it globally. ? Configure the server by adding cors, body-parser.

### User Authentication

- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

### Define API Routes



- Create separate route files for different API functionalities such as users orders, and authentication.
- Define the necessary routes for listing products, handling user registration and login, managing orders, etc.
- Implement route handlers using Express.js to handle requests and interact with the database.

### **User Authentication:**

- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

### **Error Handling:**

- Implement error handling middleware to catch and handle any errors that occur during the API requests.
- Return appropriate error responses with relevant error messages and HTTP status codes.

### **Reference Video :**

<https://drive.google.com/file/d/1O84dsRjcvYzVbFF0zIXquBojTEi2HMB/view?usp=sharing>

## **Database**

### **1. Configure MongoDB:**

- ? Install Mongoose.
- ? Create database connections.
- ? Create Schemas & Models.

### **2. Connect database to backend :**

Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.

```
const PORT = 5000;

mongoose.connect(process.env.MONGO_URI)
  .then(() => console.log('MongoDB connected successfully'))
  .catch(err => console.error('MongoDB connection error:', err));
```

### **3. Configure Schema:**

Firstly, configure the Schemas for MongoDB database, to store the data in such a pattern. Use the data from the ER diagrams to create the schemas. The schemas are looks like for the Application.

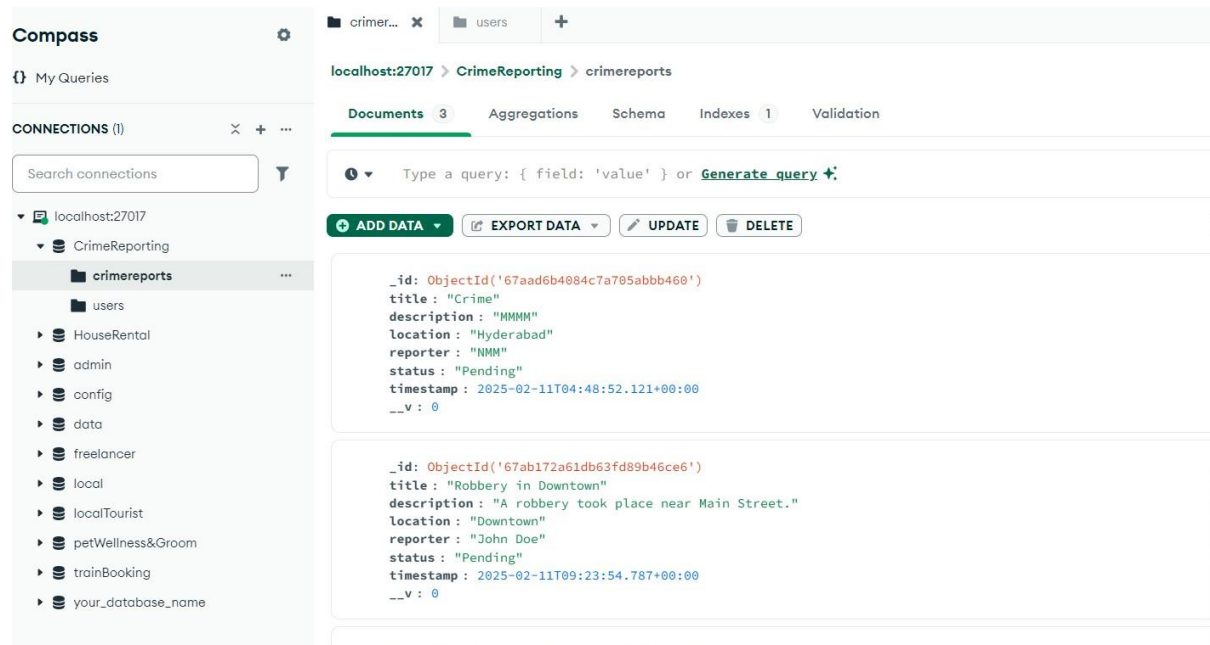
```
Backend > models > JS userModel.js > ...
1  import mongoose from 'mongoose';
2
3  const userSchema = mongoose.Schema({
4    username: { type: String, required: true, unique: true },
5    password: { type: String, required: true },
6    email:{ type: String, required: true},
7    age:{ type: String, required: true }
8  });
9
10 const User = mongoose.model('User', userSchema);
11 export default User;
12 |
```

**userModel.js** ? Stores user details

```
Backend > models > JS reportModel.js > ...
1  import mongoose from 'mongoose';
2
3  const reportSchema = mongoose.Schema({
4    title: String,
5    description: String,
6    location: String,
7    reporter: String,
8    assignedOfficer: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
9    status: { type: String, default: 'Pending' },
10    timestamp: { type: Date, default: Date.now }
11  });
12
13 const CrimeReport = mongoose.model('CrimeReport', reportSchema);
14 export default CrimeReport;
15 |
```

**rReportModel.js**:Stores crime reports & officer assignments

## MongoDB Compass:



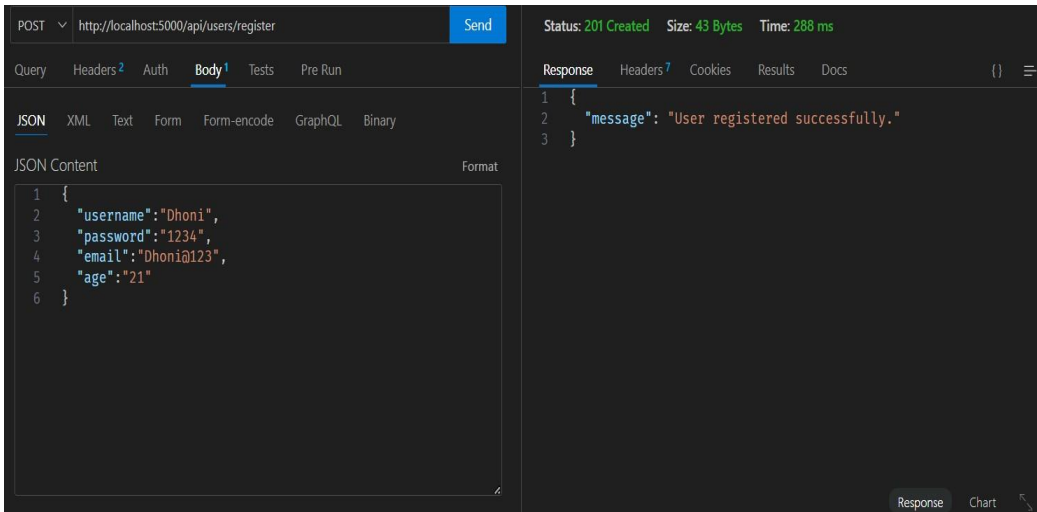
## Reference Video:

<https://drive.google.com/file/d/1Nk0lhCBfujb2Z3Mqwr7zcAuG9o3bQ4N/view?usp=sharing>

## API Testing

Finally, after finishing coding the projects we run the whole project to test it's working process and look for bugs. Now, let's have a final look at the work.

## User Register :

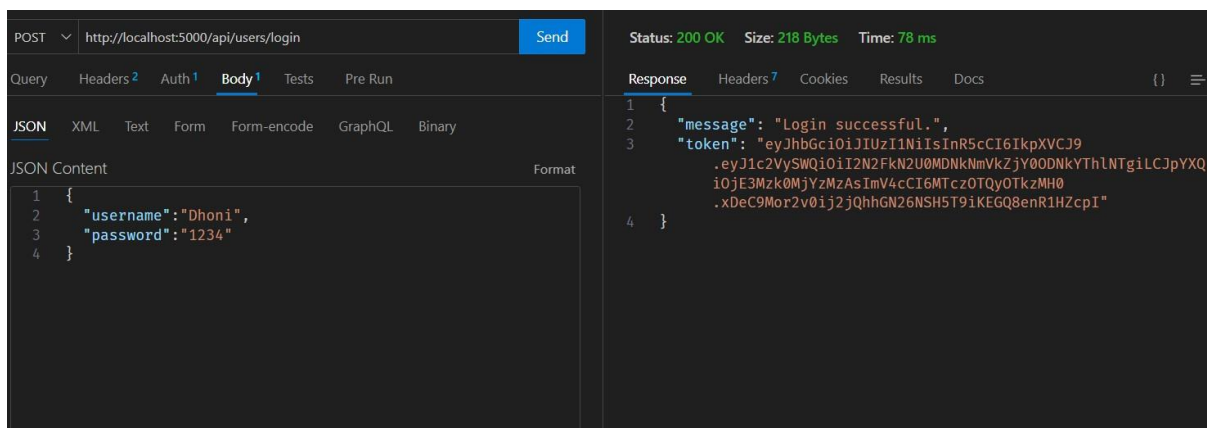


**Method:** POST

**URL:** /api/users/register

**Description:** Registers a new user.

## User Login:

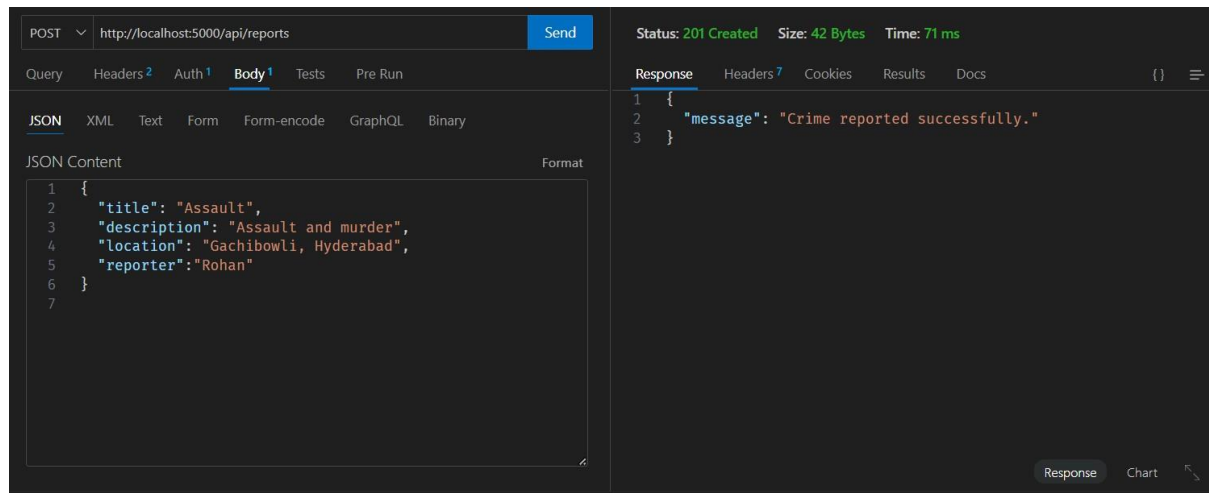


**Method:** POST

**URL:** /api/users/login

**Description:** Authenticates a user and returns a JWT token.

### Create a New Crime Report :

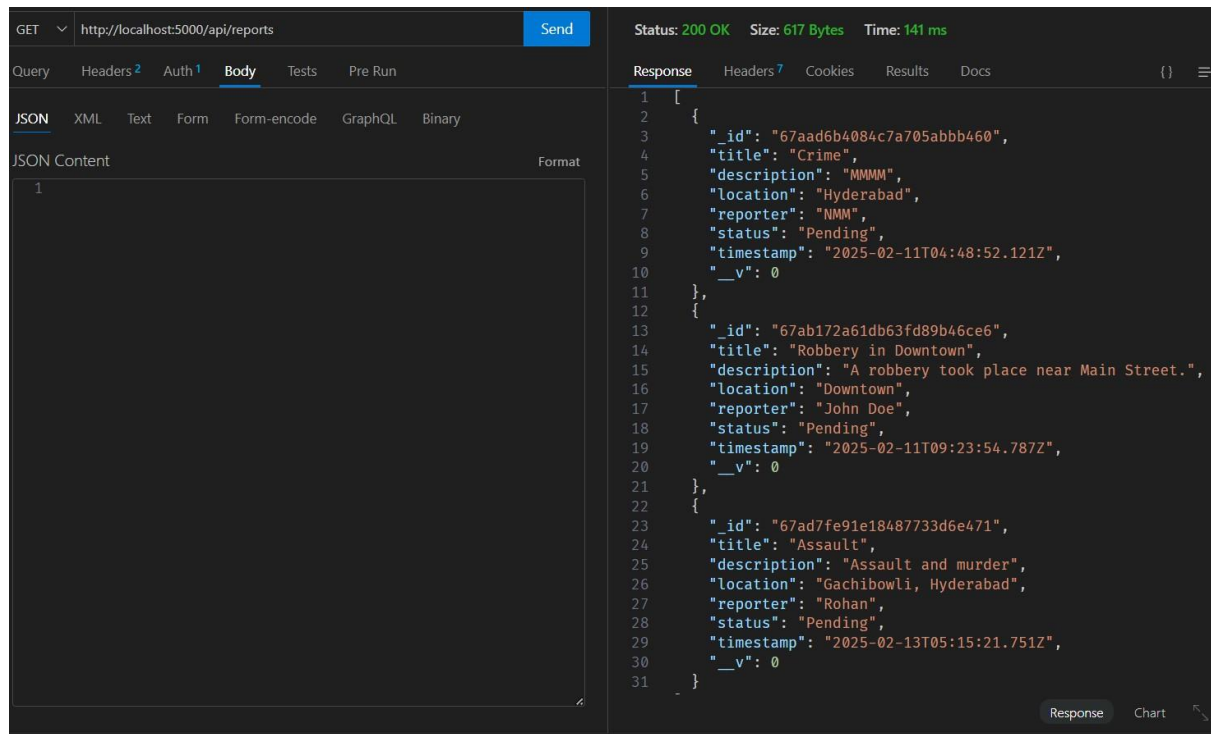


**Method:** POST

**URL:** /api/reports

**Description:** Allows users to report a new crime.

### Get All Crime Reports :

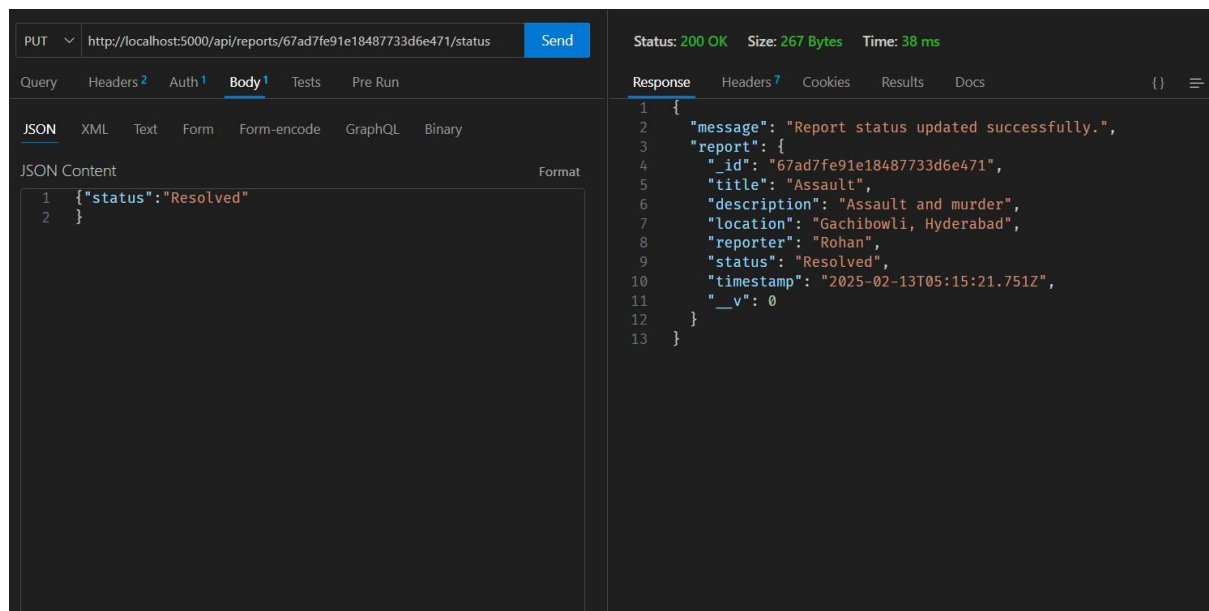


**Method:** GET

**URL:** /api/reports

**Description:** Fetches a list of all reported crimes.

**Update Crime Report Status:**

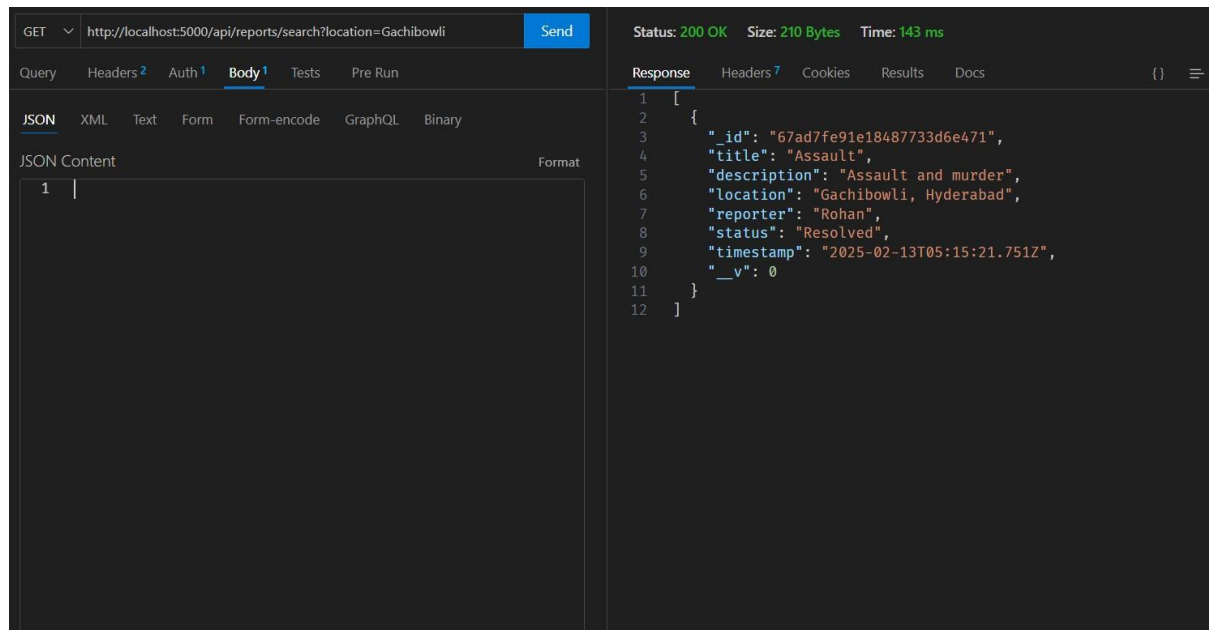


**Method:** PUT

**URL:** /api/reports/:id/status

**Description:** Updates the status of a specific crime report.

**Search Reports by Location:**

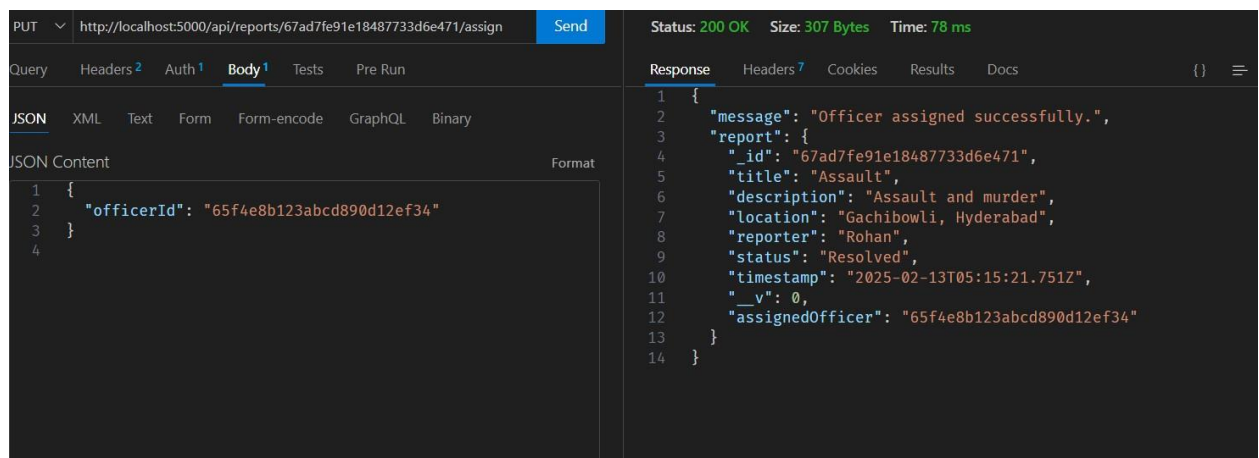


**Method:** GET

**URL:** /api/reports/search?location=Downtown

**Description:** Filters crime reports based on location.

### Assign a Crime Report to an Officer:



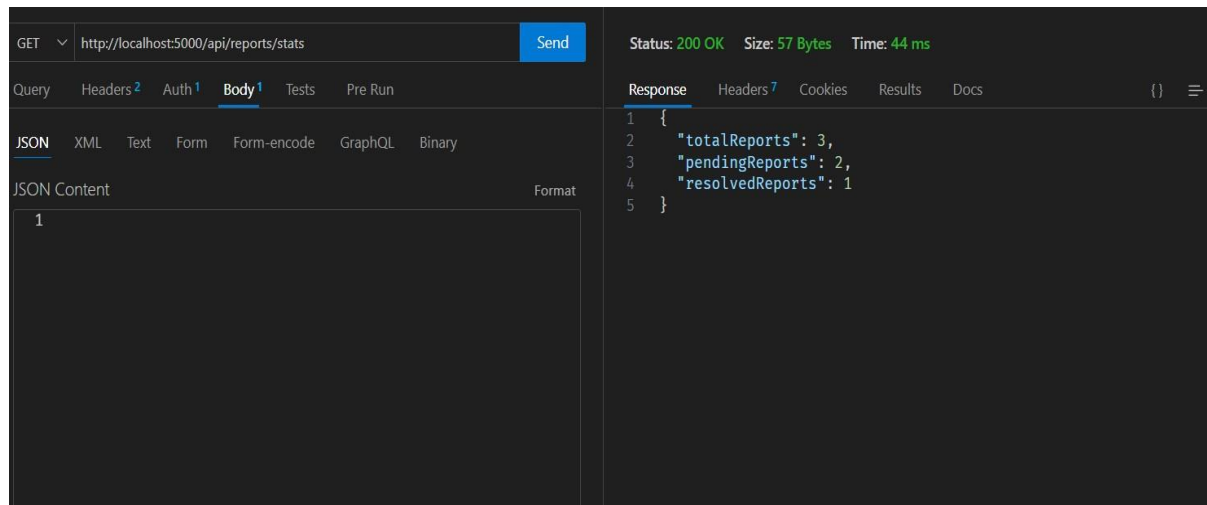
**Method:** PUT

**URL:** /api/reports/:id/assign

**Description:** Assigns a report to a police officer.

**Get Crime Report Statistics :**



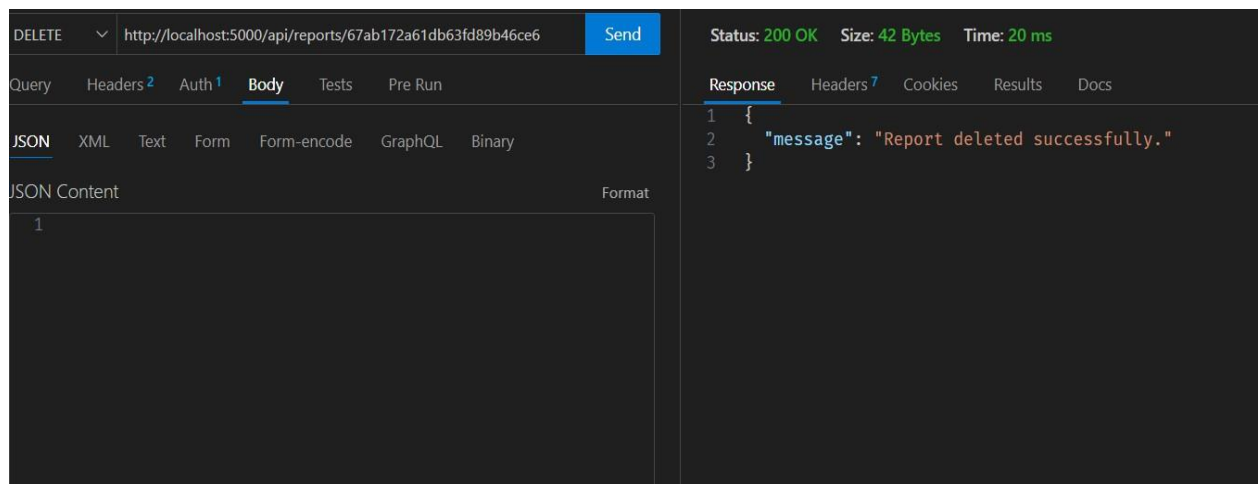


**Method:** GET

**URL:** /api/reports/stats

**Description:** Retrieves statistics about crime reports.

### Delete Report :

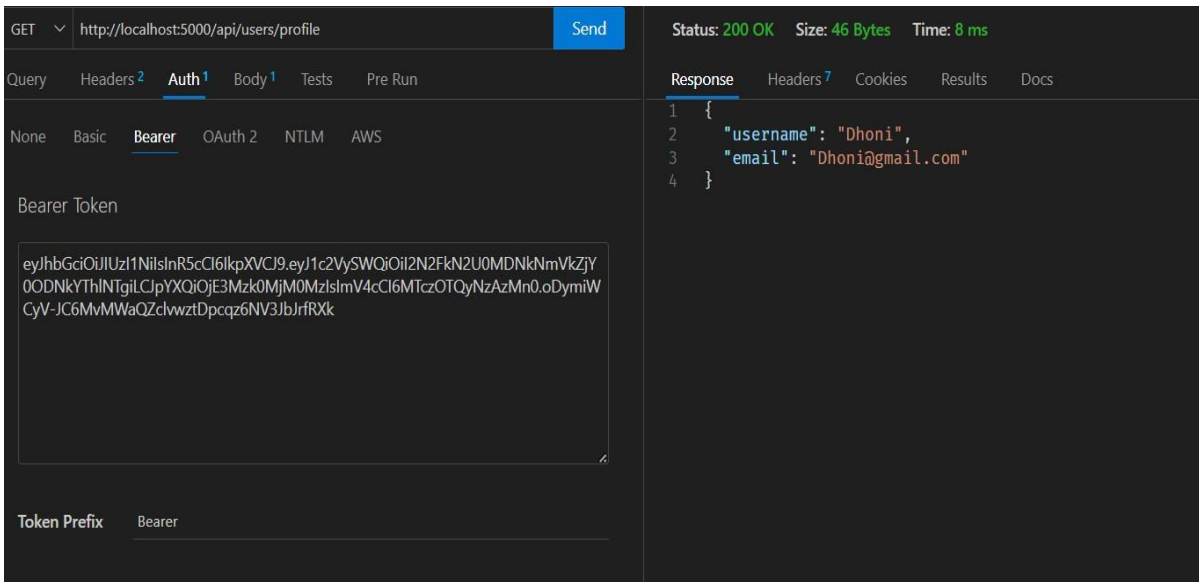


**Method:** DELETE

**URL:** /api/reports/:id

**Description:** Deletes a specific crime report

### User Profile :

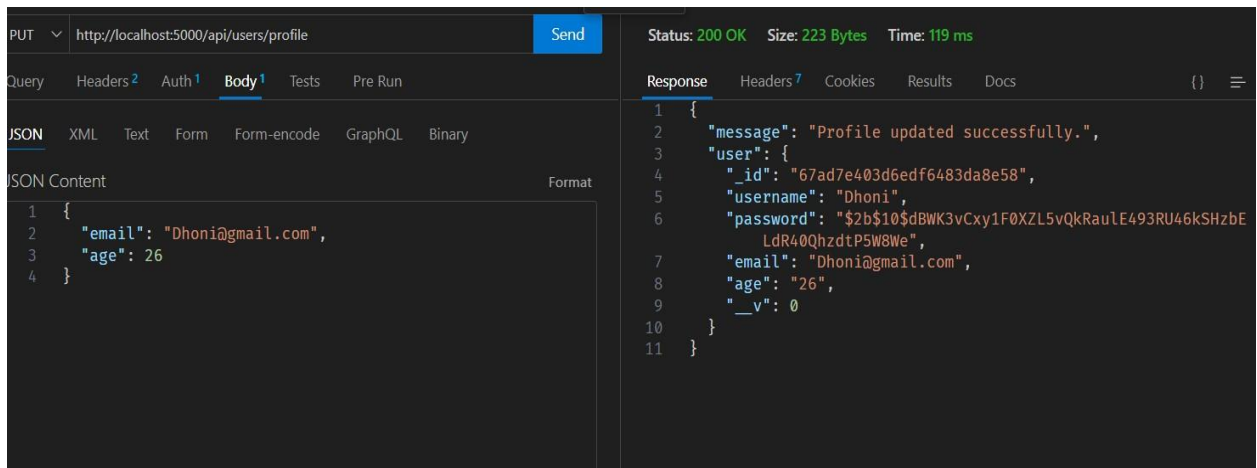


**Method:** GET

**URL:** /api/users/profile

**Description:** Fetches the logged-in user's profile.

## Update Profile



**Method:** PUT

**URL:** /api/users/profile

**Description:** Updates the logged-in user's profile.

### User Logout :

The screenshot displays a REST client interface with the following details:

- Method:** POST (dropdown menu)
- URL:** http://localhost:5000/api/users/logout
- Auth:** Bearer (selected)
- Bearer Token:** eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQjOiI2N2FkN2U0MDNkNmVlZjY0ODNkYThtNTgILCJpYXQjOiE3Mzk0MjM0MzIsImV4cCI6MTczOTQyNzAzMn0.oDymiW CyV-JC6MvMWaQZclvwztDpcqz6NV3JbJrFRXk
- Token Prefix:** Bearer
- Status:** 200 OK
- Size:** 32 Bytes
- Time:** 5 ms
- Response:**

```
1 {
2   "message": "Logout successful."
3 }
```

**Method:** POST

**URL:** /api/users/login

**Description:** Authenticates a user and returns a JWT token.