* **Name:** Viki Kumar
* **Course:** B.Tech in Computer Science Engineering (CSE)
* **University:** Lovely Professional University
* **Email:** Vikikumar0505@gmail.com
* **LinkedIn:** www.linkedin.com/in/viki-kumar
* **GitHub:** https://github.com/vikikumar05
* **Date of Submission:** 4th of May

**Project Title**

**HumanChain AI Safety Incident Log API**

**Introduction**

This project is a **RESTful API** designed to log and manage hypothetical AI safety incidents. It is built using **Node.js** and **Express**, with **MongoDB** as the database. The API allows users to log incidents, retrieve them, and delete them via HTTP requests.

**Objective**

The goal is to assess fundamental backend development skills, including API design, request handling, and data persistence in a real-world context related to AI safety.

**Project Overview**

The **AI Safety Incident Log API** enables the creation, retrieval, and deletion of AI safety incidents stored in a database. This project is built using **Node.js**, **Express**, and **MongoDB**.

Key Features:

* **Log Incidents**: Ability to log AI safety incidents with a title, description, and severity.
* **Retrieve All Incidents**: Fetch all incidents stored in the database.
* **Retrieve Specific Incident**: Retrieve an incident using its unique ID.
* **Delete Incident**: Remove an incident from the database.

**Technologies Used**

* **Node.js** - Server-side JavaScript runtime
* **Express** - Web framework for Node.js
* **MongoDB** - NoSQL database
* **Mongoose** - MongoDB ODM for data modeling
* **dotenv** - Loads environment variables from .env files

**Features**

* **GET /incidents**: Retrieve all incidents.
* **POST /incidents**: Log a new incident to the database.
* **GET /incidents/{id}**: Retrieve a specific incident by ID.
* **DELETE /incidents/{id}**: Delete a specific incident by ID.

**Why This Project?**

This project was chosen because it aligns with the growing focus on AI safety and the need to log and manage incidents for better transparency and accountability in AI systems. The simple design of the API allows it to be easily integrated into various platforms and expanded with additional features as needed.

**API Endpoints**

**1.**GET /incidents

* **Description**: Retrieves all AI safety incidents stored in the database.
* **Response**: 200 OK with a JSON array of incident objects.
* **Example**:

[

{

"id": 1,

"title": "Incorrect Translation",

"description": "AI translation led to diplomatic misunderstanding.",

"severity": "Low",

"reported\_at": "2025-05-04T05:09:12.168Z"

}

]

**2.**POST /incidents

* **Description**: Log a new incident.
* **Request Body**:

{

"title": "Misleading Chatbot Output",

"description": "Chatbot gave misleading medical advice.",

"severity": "High"

}

* **Response**: 201 Created with the newly created incident, including id and reported\_at.
* **Error Response**: If required fields are missing, returns 400 Bad Request.

**3.**GET /incidents/{id}

* **Description**: Retrieve a specific incident by its ID.
* **Response**: 200 OK with the incident object if found, 404 Not Found if not found.
* **Example**:

{

"id": 1,

"title": "Incorrect Translation",

"description": "AI translation led to diplomatic misunderstanding.",

"severity": "Low",

"reported\_at": "2025-05-04T05:09:12.168Z"

}

**4.**DELETE /incidents/{id}

* **Description**: Delete the specified incident.
* **Response**: 204 No Content if successful, 404 Not Found if incident does not exist.

**Setup Instructions**

**Prerequisites:**

* **Node.js** and **npm** installed on your machine.
* **MongoDB** installed locally or use **MongoDB Atlas**.

**Steps to Run the Project:**

1. **Clone the Repository**:
2. **Install Dependencies**:

npm install

1. **Set up Environment Variables**:

PORT=5000

MONGO\_URI=mongodb://127.0.0.1:27017/incidentlog

If you're using **MongoDB Atlas**, replace the MONGO\_URI with your Atlas connection string.

1. **Run MongoDB**:  
   If using local MongoDB, run the following in a separate terminal:

mongod

1. **Start the Application**:

npm run dev

The API will be available at [http://localhost:5000](http://localhost:5000/).

**Testing the API**

You can use **Postman** or **curl** to test the API.

* **GET All Incidents**:

curl http://localhost:5000/incidents

* **POST New Incident**:

http://localhost:5000/incidents -H "Content-Type: application/json" -d '{"title": "New Incident", "description": "This is a new incident.", "severity": "Medium"}'

* **GET Incident by ID:**

http://localhost:5000/incidents/6816f678994dcc6e5667cc42

* **DELETE Incident by ID**:

http://localhost:5000/incidents/6816f678994dcc6e5667cc42

**Challenges & Design Decisions**

* **Handling MongoDB Object IDs**: Since MongoDB generates unique \_id for each document, I used mongoose.Types.ObjectId.isValid(id) to validate the format of the provided ID.
* **Error Handling**: Implemented validation for request body in POST and GET requests, ensuring valid data is submitted.
* **Database Persistence**: Used MongoDB and Mongoose for handling the incidents, ensuring data is stored in a flexible, scalable format.

**Conclusion**

This project demonstrates my backend skills with **Node.js**, **Express**, and **MongoDB**, and it provides a functional API to manage AI safety incidents. The project adheres to REST principles and includes basic error handling and data validation.