# AI Threat Intelligence Platform - Project Documentation

*Document Version: 1.0 Date: June 02, 2025*

## 1. Introduction

This document provides comprehensive documentation for the AI Threat Intelligence Platform project. The platform is designed to collect, analyze, and display threat data from various sources, enabling security teams to proactively manage and mitigate potential cyber threats.

## 2. Project Overview

The AI Threat Intelligence Platform comprises a backend developed with FastAPI and a frontend built using Angular. The backend manages threat data ingestion, validation, and storage in an in-memory database, while the frontend delivers an intuitive user interface for interacting with threat intelligence data.

## 3. System Architecture

Backend (FastAPI): Manages API endpoints for threat analysis, log retrieval, and health status checks.

Frontend (Angular): Offers a web-based dashboard for visualizing and managing threat logs.

In-Memory Database: Serves as a temporary storage solution for threat logs during runtime.

CORS Middleware: Facilitates cross-origin resource sharing between the frontend and backend.

## 4. Directory Structure

Final\_AI\_Threat\_Intelligence\_Platform/  
 backend/  
 main.py  
 requirements.txt  
 frontend/  
 angular.json  
 package.json  
 tailwind.config.js  
 tsconfig.json  
 src/  
 index.html  
 main.ts  
 app/  
 app.component.ts  
 app.module.ts  
 environments/

## 5. Technology Stack

|  |  |
| --- | --- |
| **Category** | **Technologies** |
| **Backend** | Python 3.9+, FastAPI, Uvicorn, Pydantic |
| **Frontend** | Angular 15+, TypeScript, Tailwind CSS |
| **Other Tools** | Git & GitHub, Node.js & NPM, python-docx |

## 6. Backend Details

### 6.1. main.py Overview

The main.py file defines the FastAPI application with the following components:

ThreatData Model: A Pydantic model enforcing data validation for threat entries (e.g., source IP, destination IP, threat type, severity, timestamp).

CORS Middleware: Configured to allow all origins, methods, and headers.

In-Memory Database (threat\_log\_db): A list storing threat logs during runtime.

Utility Functions:

- ip\_in\_blacklist(ip: str) -> bool: Verifies if an IP is blacklisted.

### 6.2. API Endpoints

### 6.2.1. POST /analyze-threat/

Description: Accepts a JSON payload matching the ThreatData model, validates it, applies blacklist checks and severity thresholds, and stores valid threats.

Request Body (Example):

{  
 "source\_ip": "192.168.1.15",  
 "destination\_ip": "10.0.0.20",  
 "threat\_type": "DDoS",  
 "severity": 8,  
 "timestamp": "2025-05-30T14:30:00Z"  
}

Responses:

200 OK: Success message with threat data.

400 Bad Request: Validation errors (e.g., invalid IP, future timestamp).

403 Forbidden: Blacklisted source IP.

### 6.2.2. GET /threats/

Description: Returns all stored threat logs.

Response (Example):

[  
 {  
 "source\_ip": "192.168.1.15",  
 "destination\_ip": "10.0.0.20",  
 "threat\_type": "DDoS",  
 "severity": 8,  
 "timestamp": "2025-05-30T14:30:00Z"  
 }  
]

### 6.2.3. GET /threats/{source\_ip}

Description: Retrieves threat logs for a specified source IP.

Path Parameter: source\_ip (string, IP address format)

Responses:

200 OK: List of matching threat logs.

400 Bad Request: Invalid IP format.

404 Not Found: No threats found for the source IP.

### 6.2.4. GET /health

Description: Confirms the application is running.

Response:

{  
 "status": "running"  
}

## 7. Frontend Details

### 7.1. Overview

The frontend, built with Angular, provides a responsive UI for managing threat intelligence data. Key files include:

angular.json: Angular CLI configuration.

package.json: Frontend dependencies and scripts.

src/index.html: Main HTML file.

src/main.ts: Bootstraps the Angular application.

src/app/app.module.ts: Root module.

src/app/app.component.ts: Root component.

Tailwind CSS: Configured via tailwind.config.js and styles.css for styling.

## 8. Installation and Setup

### 8.1. Prerequisites

Python 3.9 or higher

Node.js v14.x or higher

Git

### 8.2. Backend Setup

Navigate to the backend directory:

cd Final\_AI\_Threat\_Intelligence\_Platform/backend

(Optional) Create and activate a virtual environment:

python -m venv venv  
source venv/bin/activate # Linux/Mac  
venv\Scripts\activate # Windows

Install dependencies:

pip install -r requirements.txt

### 8.3. Frontend Setup

Navigate to the frontend directory:

cd Final\_AI\_Threat\_Intelligence\_Platform/frontend

Install dependencies:

npm install

## 9. Running the Application

### 9.1. Backend

cd Final\_AI\_Threat\_Intelligence\_Platform/backend

uvicorn main:app --reload --host 0.0.0.0 --port 8000

### 9.2. Frontend

cd Final\_AI\_Threat\_Intelligence\_Platform/frontend

ng serve --open

## 10. Future Enhancements

Implement persistent storage (e.g., PostgreSQL, MongoDB).

Add data visualization dashboards (charts, graphs) to the frontend.

Introduce user authentication and role-based access control.

Integrate real-time threat feeds from external APIs.

Deploy using Docker and Kubernetes for scalability.

## 11. License

This project is licensed under the MIT License. See the LICENSE file for details.