**IoT Theft Detection Web Dashboard – FastAPI Implementation Plan**

**1️⃣ Objective**

**To provide a real-time, user-friendly web dashboard that:**

* **Displays logs of known and unknown faces**
* **Shows intruder images captured by the IoT system**
* **Provides analytics on detections**
* **Works seamlessly with Raspberry Pi**

**2️⃣ Technical Stack**

| **Layer** | **Technology / Tool** | **Purpose** |
| --- | --- | --- |
| **Backend** | **Python + FastAPI** | **Serve API endpoints & dashboard pages** |
| **Frontend** | **HTML + TailwindCSS / Bootstrap** | **Responsive UI design** |
| **JS Libraries** | **Chart.js or Plotly** | **Interactive charts and analytics** |
| **Data Storage** | **JSON (known.json / unknown.json) + captures/ images** | **Persistent logs and snapshots** |
| **Real-Time** | **AJAX polling or WebSockets (FastAPI WebSocket)** | **Instant updates on detections** |
| **Deployment** | **Raspberry Pi OS** | **Local network access; optional remote access** |

**3️⃣ Folder Structure**

**iot\_dashboard/**

**│**

**├─ app.py # FastAPI backend**

**├─ templates/**

**│ └─ index.html # Main dashboard HTML**

**├─ static/**

**│ ├─ css/**

**│ ├─ js/**

**│ └─ captures/ # Intruder images**

**├─ logs/**

**│ ├─ known.json**

**│ └─ unknown.json**

**├─ utils/**

**│ └─ logger.py # Logging utilities (from IoT system)**

**└─ requirements.txt # Dependencies**

**4️⃣ Backend Design (FastAPI)**

**API Endpoints**

1. **GET /api/known\_logs → Returns all known face logs**
2. **GET /api/unknown\_logs → Returns all unknown face logs**
3. **GET /api/latest\_alerts → Optional: return last N intrusions**
4. **GET /captures/{image\_name} → Serves captured images**

**Backend Features**

* **Read JSON files safely (utils/logger.py)**
* **Serve data as JSON for frontend**
* **Serve static files (images, CSS, JS)**
* **Optional: implement WebSocket for real-time updates**

**5️⃣ Frontend Design**

**Dashboard Sections**

1. **Event Tables**
   * **Known Faces Table**
   * **Unknown Faces Table (intruders)**
   * **Columns: Timestamp, Face Name, Objects Detected, Alert, Capture Image**
2. **Image Viewer**
   * **Display captured images as thumbnails**
   * **Click to enlarge**
3. **Analytics & Charts**
   * **Pie chart: known vs unknown faces**
   * **Bar chart: most detected objects**
   * **Timeline chart: detections over time**
4. **Filters & Sorting**
   * **Filter by face type, timestamp, or object**
   * **Sort by alert status or time**
5. **Real-Time Updates**
   * **Refresh tables/charts automatically using AJAX or WebSockets**

**6️⃣ Implementation Steps**

**Step 1 – Environment Setup**

* **Install FastAPI: pip install fastapi uvicorn**
* **Install dependencies: pip install jinja2 python-multipart pydantic (optional: Tailwind / Chart.js CDN)**

**Step 2 – Backend API**

* **Implement endpoints to read JSON logs and serve data**
* **Serve captured images via /static/captures**

**Step 3 – Frontend UI**

* **Create index.html with Tailwind or Bootstrap**
* **Tables for logs, image previews, chart placeholders**

**Step 4 – Data Visualization**

* **Add Chart.js scripts to visualize analytics**
* **Fetch JSON via AJAX /api/known\_logs and /api/unknown\_logs**

**Step 5 – Real-Time Updates**

* **Implement AJAX polling every 5–10 seconds OR WebSocket for instant updates**

**Step 6 – Testing**

* **Run backend on Raspberry Pi: uvicorn app:app --host 0.0.0.0 --port 8000**
* **Test API endpoints in browser or Postman**
* **Check logs, images, charts update correctly**

**Step 7 – Optional Enhancements**

* **User authentication (Flask-login or FastAPI OAuth2)**
* **Export logs (CSV / JSON)**
* **Remote secure access (HTTPS + Nginx reverse proxy)**

**7️⃣ Workflow Overview**

1. **IoT System detects faces → logs events in JSON + saves image**
2. **FastAPI backend reads JSON logs and serves API endpoints**
3. **Frontend dashboard fetches logs, displays tables and charts**
4. **Real-time updates notify users immediately about intruders**