Automated Anomaly Detection System

Objective

Evaluate the candidate's ability to design and implement a modular, ML-integrated anomaly detection system using a Node.js backend, with emphasis on core business logic, decoupling, and modern AI-assisted development (**vibe coding**).

1. Requirements

1.1. Frontend Framework

- Use **Angular** or **React** to implement the GUI.
- If React is chosen, **TypeScript** (.ts / .tsx) must be used.

1.2. Frontend GUI

- Implement:
  - A **search page** with search criteria filters and results table.
  - A **modal dialog** for displaying detailed anomaly alerts (see GUI.png for reference).
- Use **Material-UI (MUI)**.
- Provided UI HTML files may be used as a base.
- UI sophistication is not required — focus on core logic.

1.3. Backend Framework

- Use **Node.js** for backend services.
- Deploy backend on **AWS EC2**.

1.4. Backend Storage

- Use **AWS RDS** to store anomaly alert data.
- Implement **decoupled CRUD operations**.
- Provide a reasonable solution for storing **frame data**.

1.5. Machine Learning Integration

- Integrate pretrained **YOLO** (object detection) and optionally **LSTM** (behavior analysis).
- Use samples from the **Stanford Drone Dataset**.
- Deploy models in-browser using **WebAssembly (WASM)** or **WebGPU**.
- Focus is not on model accuracy — use off-the-shelf pretrained models.

1.6. Anomaly Detection System

- Users must be able to upload video clips via the web app.
- Implement anomaly detection with custom rules.
- Upon detection, create an alert with:
  - `timestamp`, `alert_type`, `message`, `frame`, `details`

2. Non-Functional Requirements

2.1. Unit Testing

- Use **Jest**.
- Target **80%+ code coverage**.

2.2. Code Quality

- Pass all **ESLint** checks.

2.3. AWS Implementation

- Use a **free AWS trial account**.
- Provide access credentials to reviewers for verification.

3. Vibe Coding and AI-Assisted Development

3.1. Coding with LLMs

- We require candidates to go with a **vibe coding** workflow using AI tools instead of writing all code manually.
- Suggested tools:
  - Roo-code
  - Any LLM-powered IDE (e.g., VS Code with Copilot-like extensions)
- You may use public/free LLM APIs like:
  - NVIDIA Build: LLaMA 3.3 Nemotron Super 49B

3.2. Evaluation Focus

- Effective interaction with LLMs to generate and refine working code
- Logical prompt engineering and understanding model capabilities
- Adherence to design patterns and architectural decoupling
- Diagnostic and debugging skills with AI-generated code

4. Submission

- Provide a link to your **GitHub repository** with:
  - All source code, including prompts used to generate the code via LLMs.
- Share your **AWS trial account credentials** for implementation review.
- **If you are unable to complete any part of the assignment, do not fabricate or guess the answer. Instead, include a brief explanation of why it could not be completed and describe your intended approach or solution strategy.**