Task: Develop a Simple Cyber Asset Management API

Objective

The goal of this project is to create a RESTful API that allows users to manage a database of cyber assets, such as networking equipment, servers, personal computers, etc. You will use any programming language of your choice and utilize GIT for version control. Please host your code in a public or private Git repository (like GitHub, GitLab, etc.), making sure to commit often to demonstrate your version control skills. The code must be able to compile/run, and instructions must be provided. This assessment shouldn't take more than an hour to complete, you may limit yourself to one hour and submit what you have completed.

Part 1: Basic API Setup and Initial Endpoints

Requirements:

- Initialize a new git repository for your project.
- Create a basic API that connects to a SQLite database.
- Implement the following endpoints:
    - GET /assets: Returns a list of all cyber assets in the database.
    - POST /assets: Adds a new cyber asset to the database. This endpoint should accept cyber asset data in JSON format (e.g., device name, type, serial number, operating system, etc).
    - GET /assets/{id}: Returns details of a specific cyber asset by ID.
    - DELETE /assets/{id}: Deletes a specific cyber asset by ID.

Deliverables

- Source code checked into GIT, with clear, atomic commit messages.
- Documentation on how to set up and run the API.

Part 2: Enhancing Functionality and Adding Error Handling

Requirements

- Update the API to handle errors gracefully, returning appropriate HTTP status codes.
- Add functionality to update existing cyber assets:
    - PUT /assets/{id}: Updates a specific cyber asset's information by ID.
- Include filtering for the GET /assets endpoint to retrieve specific cyber asset types (networking equipment, servers, etc)
- Include pagination for the GET /assets endpoint to handle large volumes of data efficiently.

Deliverables

- Updated source code in GIT with new endpoints.
- Updated documentation including new endpoints and any additional setup required.

Part 3: Reflections

Requirements

- Create a TODO file in your repository reflecting on how you would continue to enhance the functionality of this API
    - What would you change given extra time?
    - What would you add to the API to make it more secure?
    - Would you continue using the same database?
    - Could the project be more portable?

Deliverables

- Final source code in GIT
- A brief reflection on the project, challenges faced, and key learnings.

Submission Instructions

- Email the link to the GIT repository to hr@corkinc.com
- Ensure your GIT history reflects the progression through these parts.
- Provide a README file with detailed instructions on setting up and testing the API, including any necessary environment setup.
- OPTIONAL: Include any Postman collections or scripts that can be used to demonstrate the API's functionality.