Take home assignment - System architect (UAV)

Problem Statement:

You are tasked with designing and implementing a prototype for a Ground Control Software (GCS) that can automate basic operations for a UAV imaging payload. The prototype will focus on creating a robust backend system that manages UAV commands, payload operations (like imaging), and basic analytics, while also considering the interaction with hardware components. The goal is to assess your ability to architect software solutions that interface with hardware, ensure scalability, and maintain code quality.

Functional Requirements:

Command and Control Interface:

- Implement a RESTful API that allows users to send basic commands to the UAV, such as:
  - `start_mission`: Initializes a mission and prepares the UAV.
  - `capture_image`: Triggers the UAV payload to capture an image.
  - `stop_mission`: Safely terminates the mission.
- The API should respond with appropriate status codes and messages.

Payload Operations:

- Simulate the payload operations (e.g., imaging) as background processes that are triggered by the above API using multiprocessing and/ or priority scheduling to achieve minimum latency possible.
- Implement basic error handling for situations where the payload operation might fail.

Data Storage and Retrieval:

- Design a simple database schema to store mission logs, captured images metadata, and analytics results.
- Provide an endpoint to retrieve this data, with filters such as mission ID and time range.

System Architecture:

- Design the system architecture using UML diagrams (HLD and LLD).
- The architecture should highlight interactions between the software components, hardware blocks, and APIs.

CI/CD Automation:

- Implement a basic CI/CD pipeline using GitHub Actions or a similar tool that:
  - Runs unit tests for the API and image processing functions.
  - Deploys the prototype on a local server.

Scalability Considerations:

- Document how you would scale the system to support multiple UAVs simultaneously.
- Discuss how you would ensure backward compatibility with future hardware upgrades.

Deployment Instructions:

- The prototype should be deployable on a local machine running Linux.
- Provide a `README` file with instructions on how to:
  - Set up the environment (including any dependencies).
  - Run the CI/CD pipeline.
  - Deploy the server and interact with the API.
  - Access the database and view logs/analytics.

Evaluation Criteria:

1. **Completeness (25%):**
    - Does the prototype meet all functional requirements?
    - Are all the required features implemented and working as expected?
2. **Code Quality (20%):**
    - Is the code well-structured, modular, and easy to understand?
    - Are best practices followed in terms of style, commenting, and error handling?
3. **System Design (20%):**
    - Is the system architecture well thought out and clearly documented?
    - Do the UML diagrams accurately reflect the system's components and interactions?
4. **Scalability and Future-Proofing (15%):**
    - Are scalability considerations well-documented?
    - Is there a clear plan for how to adapt the system for future needs?
5. **CI/CD Implementation (10%):**
    - Is the CI/CD pipeline correctly implemented?
    - Does it automate testing and deployment effectively?
6. **Deployment and Usability (10%):**
    - Is the system easy to set up and run on a local machine?
    - Are the deployment instructions clear and comprehensive?

Assume whatever deems fit for this use case. We expect you to set up everything on a local-host. You are free to take it up a notch ahead and deploy it live on Heroku, etc. Should you need any clarifications, please get back.

Please note that this assignment should be submitted in the form of a GitHub repository.

Brownie points will be awarded if you take care of application/API security, industry standards, design abstractions and optimizations if any ;-).