Midterm Project Requirements – DevOps Course

Scenario

Welcome aboard! You've just joined **DeployNova**, a fast-growing global SaaS company specializing in cloud-native DevOps solutions.
Your first mission as a DevOps engineer is to initiate and develop a new flagship application that will showcase the company's innovation, scalability, and technical excellence.

You are free to choose the topic of your application, as long as it meets the technical and functional requirements outlined below.

Product Functionality

1. Project Topic

Each student must independently choose a topic for their project.

The project will be based on a Python-based application.

The chosen application must include at least the following features:

- A central theme or use case (chosen by the student)
- A welcome screen
- A user menu
- A data store based on Python data structures (e.g., lists, dictionaries)
- Menu options for:
  - Adding entries
  - Removing entries
  - Displaying entries
  - Sorting entries
  - Performing calculations based on the stored data

2. Python Application

The application will be structured professionally, with:

- Clear documentation (comments, docstrings, meaningful variable names)
- At least two separate files:
  - A `main` file that runs the application
  - A `functions` file that contains reusable logic

3. Version Control

- The project will be stored in a GitHub repository.
- The repository will include **subfolders** for each project part:
  - Python code
  - Website
  - Docker file
  - AWS Beanstalk
  - AWS CloudFormation
- Your `README.md` file should clearly describe:
  - The overall folder structure of the project
  - The purpose and role of each file included
  - The folder structure for the Python app
  - Documentation of code and functions

4. Website

Your application will include a web-based interface that presents the functionality of your Python project.

You are free to convert your Python application into a web-based interface using any method or framework you prefer.

We recommend that you investigate and choose one of the following free tools:

- Streamlit
- Gradio
- Flask
- NiceGUI

5. Containerization

- You will write a `Dockerfile` to create an Ubuntu-based container that runs the web application.
- The `Dockerfile` will:
  - Include all required installations
  - Pull necessary files from GitHub
  - Run the full web application inside the container

AWS Deployment

You will deploy your application on AWS using the following services:

- **ECR**: Store your Docker image in AWS ECR.
- **Elastic Beanstalk / CloudFormation**:
  - The deployment will be fully automated using one of these technologies.
  - The deployment process will create a highly available (HA) environment running the web application on at least two EC2 instances across two different Availability Zones (AZs).

Demonstration

During the final demonstration, you will be required to:

1. **Clone your GitHub repository** to the AWS environment or your local machine.
2. **Explain your GitHub structure** and the organization of your project files and folders.
3. **Review your code**:
   - Show your Python application structure (main file and functions file).
   - Show your website code and explain how each menu item corresponds to a function.
   - Show and explain your Dockerfile and how it installs dependencies and pulls required files.
4. **Show your AWS deployment process step by step**:
   - Demonstrate the full deployment process live using the AWS sandbox provided.
   - Show your ECR repository and the Docker image you uploaded.
   - Deploy the application to run on an AWS container (EC2 and/or Elastic Beanstalk).
   - Show your CloudFormation template(s) and explain how the infrastructure is created.
   - Show your Beanstalk configuration files and explain any custom settings.
5. **Demonstrate the application running live** on AWS:
   - Access the web application through the public endpoint.
   - Demonstrate adding, listing, editing, deleting, and searching entries in the web interface.
6. **Summary and Q&A**:
   - Be prepared to answer questions about your architecture, design choices, and implementation.
   - The AWS environment must be a High Availability (HA) setup created using CloudFormation and/or Beanstalk.