

# 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
  - Edit 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 (or upload) 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

---

 **Important: You must practice the entire demonstration process ahead of time and come fully prepared.**

 You are allowed to use supporting materials (laptop, notebook, printed materials, etc.) during the demo.

During the final demonstration, you will be required to perform the following steps live:

1. **Share your GitHub repository** with the instructor prior to the demonstration.
2. **Clone your GitHub repository** to the machine that will be used for the demo.
3. **Present and explain your Python code:**
  - Open your Python code files (main file and functions file).
  - Explain the structure of your code and how the logic of your application is implemented.
  - Highlight key functions, data structures, and error handling techniques.
4. **Run your Python application locally:**
  - Run the Python application from your local machine.
  - Demonstrate its functionality (adding, listing, editing, deleting, searching contacts).
  - Enter *invalid and incorrect data* during the demo to show how your application handles errors gracefully.
5. **Build and run your application inside a local Docker container:**
  - Use your `Dockerfile` to build a **local image** and a **local container**.
  - Run the container on your local machine.
  - Demonstrate the full functionality of the **web interface** running inside the container.
6. **Deploy to AWS:**
  - Open a new AWS sandbox environment.
  - Upload your Docker image to AWS ECR.
  - Use your pre-written CloudFormation template and/or Beanstalk configuration to deploy the application:
    - Create a High Availability environment based on EC2 instances across **two different Availability Zones**.
  - Explain each step of your deployment process.

7. **Verify and demonstrate your web application on AWS:**

- Access the public URL (via Load Balancer).
- Show that your web application is fully functional on AWS.
- Demonstrate that the application continues to work when one EC2 instance is stopped.
- Stop the other instance and verify again.

8. **Summary and Q&A:**

- Be prepared to explain your architectural choices, deployment process, and implementation decisions.

---

**Good luck! 😊**