BMS COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

LAB CYCLE – AY2024-2025 [ODD]

Assignment Assessment -1

24th Oct 2024, 9 AM – 10.30 AM

COURSE: DevOps **SECTION: A, B, C & D**

COURSE CODE: 21IS7PEDVR **TOTAL CREDITS: 3 Marks**

Choose any ONE of the following projects and demonstrate a working solution on your laptop or on a lab system

Free to use any internet resources

Assignment 1: Dockerized Multi-Container Web Application with Networking and Container Management

OR

Assignment 2: Secured Multi-Container Python Application with AppArmor and Automated Health Monitoring

---

Assignment 1: Dockerized Multi-Container Web Application with Networking and Container Management

Objective: Build a Python-based web application with a database backend, containerize both services and manage the containers and networks programmatically using Python scripts.

Project Requirements:

1. **Web Application:**
    - Develop a Python Flask web application that includes basic CRUD (Create, Read, Update, Delete) functionality. The data should be stored in an SQLite database or MongoDB.

- Create a **Dockerfile** for the Flask app and containerize it.

2. **Database Container:**
   - Containerize an SQLite or MongoDB database using a separate Docker container. The database should persist data across container restarts using Docker volumes.
3. **Networking:**
   - Use the Docker SDK for Python to create a custom bridge network and connect the Flask application container with the SQLite or MongoDB database container.
   - Write a Python script to manage and inspect the Docker network, ensuring proper container communication.
4. **Container Management:**
   - Write a Python script that lists all active containers, checks the health of the Flask application container, and automatically restarts it if it's unhealthy.
5. **Demonstration:**
   - Present the solution and demonstrate network connectivity and container management.

Assignment 2: Secured Multi-Container Python Application with AppArmor and Automated Health Monitoring

Objective: Create a secure, containerized Python application with a database backend using Docker. Implement AppArmor security profiles and automate container health monitoring with Python scripts.

Project Requirements:

1. **Web Application:**
   - Develop a Python Flask application that performs basic user authentication (login and registration).
   - Containerize the Flask application using a Dockerfile.
2. **Database Backend:**
   - Set up a separate SQLite database or MongoDB container to store user data. Ensure that the database container persists data using Docker volumes.
   - Use Docker networking to allow the Flask application container to communicate with the SQLite or MongoDB container.
3. **Security with AppArmor:**

- Write an AppArmor security profile to restrict the Flask application container from accessing certain system resources (e.g., file access restrictions).
- Use the Docker SDK for Python to apply the AppArmor profile and verify its enforcement on the Flask container.

4. **Health Monitoring Automation:**
    - Create a Python script that periodically checks the health of the Flask container.
    - If the container is unhealthy (e.g., due to a failed login service), the script should automatically restart the container and log the issue.
    - Set up a notification system that sends an alert (e.g., email or console log) when the container has been restarted due to a health check failure.
5. **Demonstration:**
    - Present the project with a live demo showcasing container security, networking, and automated health monitoring.