Session 4 - Graded TP

Deadline : 07/12.2023 - 23h59.

In this nearly 5-hour work project, you will develop a web application comprising a backend, frontend, and database integration. The primary focus is on the backend, while the frontend serves as an exposition. Key requirements include the use of Docker for deployment, comprehensive testing (unit, integration, end-to-end), and automation using tools like GitHub Actions.

Functional Requirements:

- **Backend Development (Python):**
  - Create a backend for the application using the Python programming language and a web framework of their choice (e.g., Flask, Django).
- **Frontend Development (Exposition):**
  - While the frontend is not the primary focus, you should create a basic frontend interface that showcases the functionality of the backend.
- **Database Integration:**
  - Integrate a database system (e.g., SQLite, PostgreSQL) with the Python backend to store and retrieve data.
- **Docker Containers:**
  - Use Docker containers whenever necessary to ensure portability and ease of deployment.
- **Testing:**
  - Fully test all parts of their Python code, including unit tests for individual components, integration tests for interactions between components, and end-to-end tests for the complete system.
- **Automation Pipeline:**
  - Automate the development process using a pipeline orchestration tool such as GitHub Actions or a similar CI/CD system.
- **Deployment: (BONUS)**
  - Deploy a version of your Docker images on a hub of your choice
    - - Docker Hub
    - - Github

Non-Functional Requirements:

- **Project Idea (Personal Interest):**
  - You can/should choose a project idea based on their personal interests, hobbies, or passions. The project should be simple but meaningful to you. You can use it to have a better chance in your next internships and jobs
- **Frontend Design**:
  - While the frontend is not the main focus, it should have a basic yet presentable user interface that reflects the chosen project idea.
- **Intuitiveness**:
  - The application should remain intuitive and easy to work with, ensuring that the chosen project idea is effectively communicated to users and developers.
- **Performance**:
  - The application should perform efficiently, with acceptable response times for typical operations.
- **Documentation**:
  - Provide clear and comprehensive documentation that explains how to set up and run the application, including instructions for developers and users.
- **Code Quality:**

    - Maintain clean and well-structured Python code, following best practices for code readability and maintainability. (PEP8). This part should be integrated to your CI pipeline
- **Deployment:**
    - Ensure the application can be easily deployed to a production environment, following best practices for deployment and environment configuration.
- **Version Control:**
    - Use version control (e.g., Git) to track changes and collaborate on the project, with clear commit messages and branching strategies.