Final Project

Overview

In this project, students will work on designing and implementing a software application of their choice using Python. This project will allow students to apply the software engineering principles learned throughout the course, including version control, clean code practices, and thorough testing. Each student will have the freedom to select a project that aligns with their interests and **scientific** focus. Past research or anything submitted for a current or former class can be used. You can also use code that other people have written if you are improving it.

Objectives

By the end of this project, students will:

- Apply version control using Git and GitHub for collaborative development.
- Write clean, maintainable, and well-documented Python code.
- Implement unit tests and integration tests to ensure code quality.
- Present their project and demonstrate its functionality.

Project Guidelines

1. **Project Selection**

Each student must choose a software application that serves a scientific purpose or addresses a scientific problem. Examples include:

- Data analysis tools
- Simulation models
- Visualization applications
- Machine learning models
- Automation scripts for data collection or processing

The project should be feasible within the given timeline and should not exceed the complexity of a typical software engineering project.

2. **Version Control with Git and GitHub**

Each student will create a GitHub repository for their project. Students will practice branching, merging, and pull requests. Commit messages should be clear and descriptive, following best practices. Students will maintain a project board on GitHub to track tasks and progress.

3. **System Design**

Define the system architecture, focusing on a modular design. Create diagrams (class diagrams, sequence diagrams) to illustrate system components and interactions. Design a data model relevant to the chosen application, including classes and data structures.

4. **Implementation**

Develop the application using Python and ensure the code follows clean code principles, including proper naming conventions, modular design, code comments, etc.

5. **Testing**

Write unit tests for individual components using a testing framework (e.g., unittest or pytest).
Implement integration tests to verify that different parts of the system work together as expected.

6. **Documentation**

Create a README file that includes:

- Project title and description
- Installation instructions
- Usage examples
- Contribution guidelines

Maintain inline documentation throughout the codebase to explain functionality and logic.

Presentation

Each student will prepare a presentation (MAX 10 minutes) to showcase their project.
Presentations should cover:

- Overview of the application and its features
- Demonstration of the application
- Discussion of challenges faced and how they were overcome
- Reflection on the software engineering principles applied during the project

Deliverables

- ☐ A fully functional software application or analysis code implemented in Python, R or another language.
- ☐ A GitHub repository containing the complete codebase, documentation, and testing results.
- ☐ A presentation slide deck summarizing the project.

Evaluation Criteria

Functionality (40%): The application meets the specified requirements and functions as intended.
Code Quality (30%): Code is clean, well-organized, and follows best practices.

Testing (20%): Adequate unit and integration tests are provided.
Presentation (10%): Clarity, engagement, and thoroughness of the project presentation.