INFS2044 Assignment 2

Due date: Monday, 13 June 2022, 11:59 PM
Weighting: 40% of the course total marks.

Submission instructions are at the end of this document.

Instructions:

This assessment item must be conducted individually.

You will be creating an implementation design, implement it in python, and test the system. The assignment will be based on a case study and a partial design given to you. Refer to the document *Assignment 2 Case Study* on the course website for details about the case study.

Base your implementation on the code fragments given in *Assignment 2 Code for Students.zip* archive on the course website.

All submitted materials must be entirely your own work (except the source code given to you as part of the assignment specification).
Penalties apply for academic misconduct.

You will receive marks via learnonline within 3-4 weeks.

Answers will be scored based on correctness and completeness of your design and code, and the quality of the design and corresponding code will be assessed. Please ensure that your design and code are clean and self-documenting; code that works correctly but exhibits poor quality will receive low scores. Refer to the marking scheme provided in this document.

The word count for this assignment will not be verified. Please do not write excessively long answers (3k+ words). Excessively short answers, such as submitting only diagrams or code without reflection, are unlikely to satisfy the marking criteria.

It is your responsibility to use appropriate document management and source code control tools if required and to always maintain adequate backup. Hardware and software failure are not usually acceptable reasons for requesting extensions.

Start early and develop incrementally.

Assignment tasks

This assignment consists of four tasks. More detailed instructions about development approach and submission process can be found in the Submission Instructions at the end of this document.

Task 1: Class Implementation Design

Create a class-level design reflecting the to-be-created implementation of the system. Your design must comply with the given component decomposition and extend the given partial implementation that accompany the case study.

Use design principles, patterns, and principles of clean code to guide your design.

Document your implementation design as UML class diagrams. In addition, you may include interaction diagrams if you deem them helpful to illustrate aspects of your design. Interaction diagrams are however optional.

Discuss any major assumptions that you may have made and how they have influenced the design.

The answers to this task will form part of the report to be submitted on learnonline. See the Submission Instructions at the end of this document for details.

Task 2: Tests and Quality Assurance

You are expected to follow a test-driven development (TDD) approach when implementing the system. That is, *write the tests first*, *then write the source code* of the implementation to pass the tests.

Develop your tests and code base incrementally. That is, you will be interleaving the work for Task 2 and Task 3, and potentially extend and/or revise your implementation design created in Task 1 as you progress with these tasks.

Use *pytest* to create and run the tests.

Ensure that your tests don't modify the file system (that is, test should not create/delete files.)

Task 3: Source Code Implementation

Implement the design you have created in Task 1 so that the requirements of the case study are satisfied and all tests embodying the requirements run and pass.

At the end, you should present a complete implementation that

- matches the design documented in Task 1,
- completely implements the system specified in the case study,

- demonstrably satisfies the case study requirements as evidenced by test cases created during Task 2.

Use only standard packages that are available in a plain python 3.x distribution and those listed in the file requirements.txt in the provided code. No other third-party code is allowed unless approved by the Course Coordinator. Your code must be strictly cross-platform, that is, it must be pure python code.

The quality of your code will be assessed. While no coding standard is mandated for this assignment, it is expected that intention-revealing names are chosen in a consistent manner throughout your code.

Task 4: Reflection

Reflect on how you have conducted tasks 1-3, and how you have assured that the design and its implementation are correct and exhibit high quality.

Describe the design principles and patterns (if any) you have adopted in the design and implementation, as evidenced in the diagrams and source code.

Describe any major changes that you may have made during development and give reasons.

The answers to this task should be at most 1 page. Quality of discussion is preferred over quantity.

Advice

- Ensure that the design and its implementation match each other.
- Strive for clean, maintainable, self-documenting code.
- Focus your attention on the *quality* and *readability* of your code. Speed and memory efficiency are not primary concerns in this assignment.
- Use tests to evidence that your implementation works correctly.
- Version source code in github, bitbucket, or a similar system.

Marking Scheme

It is not sufficient to present code that works to pass this assignment. Your code must be designed and implemented according to the design and coding practices taught in this course. Code that merely works but does not reflect these principles and practices will be awarded a Fail grade.

Criteria	Marks (as a fraction of course total)
Task 1: Class-level Implementation Design • Implementation design covers all components in the decomposition • Class diagram(s) documenting the system implementation matches the actual implementation • Design satisfies design principles • Correct dependencies between components • Diagrams are syntactically correct • Class design is complete (includes classes, attributes, associations, public operations) • Design is implementable and will achieve use case narratives	10%
Task 2: Quality Assurance and Testing • Unit tests for all components defined • Each test has a single objective • Comprehensive test suite covering all normal situations, exception situations, and boundary situations • All tests run without errors	10%
Task 3: Source Code Implementation • Source code is complete to achieve the use cases • Source code matches the design created in Task 1 • Intention-revealing and consistent naming • Appropriate comments where comments are warranted • Uses only approved packages and libraries	15%
Task 4: Reflection • Discussion of changes made to the design as the implementation progressed (if any) • Discussion of the design principles and patterns that are evidenced in the design and corresponding code (if any) • Free of grammar/spelling mistakes • Diagrams easily legible when printed on A4 paper	5%
Total	**40%**

Submission Instructions

Submit two files:

1. a zip archive containing all source code and test code, and
2. a report in PDF format.

Source code

- Submit all source code (answers to tasks 2&3) as a ZIP archive to learnonline.
- The archive must contain separate source files for the system implementation and the unit tests. That is, tests should be in a separate file, not in the same file as the code implementing the system.
- The source code into modules, each in a separate source files.
- The system must be runnable by executing the "word_statistics_app.py" file.
- All tests must be runnable via pytest.
- The archive must contain all source code and test code to run and test the system. Please include only source code and documentation files; exclude generated files, IDE configuration files, etc.

Report

- The report shall contain the design documentation (UML diagrams), related discussion, and reflection as described in the Assignment tasks 1 & 4.
- Please use following overall structure:
    1. Cover page
        - Include your name, student identifier, and UniSA email address
    2. Class-Level Design
        - Present the UML diagram(s) and related discussion
        - State assumptions you have made and their impact on the design
    3. Quality Assurance
        - Briefly summarise your approach to quality assurance during creation of the design and source code.
        - Briefly describe your approach to test-driven development of the program
        - Do NOT include pytest and pylint reports in the submission.
    4. Reflection
        - Describe any major changes that you may have made during development (if any) and give reasons
        - Describe the design principles and patterns (if any) you have adopted in the design and implementation, as evidenced in the diagrams and source code.
- Submit your report to learnonline in PDF format.