General information

In this first part of the assignment your goal is to design a test plan for an E-commerce store (a web application). In the second part the application is to be tested by following the test plan.

The application – an E-commerce store

The application is an E-commerce store selling food products from various small producers. Users can search products by category, price, product contents, and producer. Products can be added to a shopping cart. Shopping cart automatically updates and shows the total price. Checkout and payment process is handled with a third-party solution. The food producers can add their products via a previously created portal, or by using a front-end application. The producers can leave some fields blank if they do not want to specify some attributes like category or contents.

The application consists of a REST back-end, which provides the interfaces for executing the core functionality of the E-commerce store (searching, buying, adding and removing products). The back-end is also responsible for accessing third party services.

Additionally, the application includes a front-end built with React. The React application uses an utility library, which can be found in https://github.com/otula/COMP.SE.200-2023-2024-1. Unit tests for the applicable parts of the utility library must be included in your test plan. The React application is used for the store functionality and can also be used to access the portal mentioned above. The React application performs validation for user inputs (e.g. check that product descriptions look similar, first word of a sentence start with an upper-case letter, prices are added with two decimal accuracy, user does not input malformed data when ordering products, etc.).

The goal of the first part of the assignment

In the first part of the assignment you should consider the wider aspects of testing the application: what components should be tested, and how? Is integration or usability testing required, or perhaps performance testing needs to be done? Create end-to-end scenarios for the common use cases of the E-commerce store (based on the application and portal description above), and describe what kind of tests are needed to validate the scenarios, and what should be kept in mind, when designing and implementing the tests. How will the test results be documented? If needed, provide a template for your issue and bug reports.

Check out the provided utility library (https://github.com/otula/COMP.SE.200-2023-2024-1). Based on your end-to-end scenarios, decide what components (in this case, functions/source files) of the library would be required in the implementation of the E-commerce store. Because of (imaginary) time constraints, you can only test 10 source files. Thus, after studying the library (and your end-to-end scenarios), use a prioritization method to select up to 10 files/functions for which you will implement unit tests in the part 2 of the assignment.

When writing your test report (in part 2 of the assignment) you should reflect your findings with the described scenarios – well-designed scenarios will also help you with part 2.

And finally, consider which tools could be used to implement the required tests. Take a look at the tools in the list below, and try them out to see what they are used for. You can also experiment with other testing tools that are not included in the list. How you test the tools is up to you (e.g. pick a file from the provided library, https://github.com/otula/COMP.SE.200-2023-2024-1, and write a simple unit test, or simply study the documentation provided for the selected tools). Include in your plan a *brief explanation* of the tools you tested, the tools' features and what experiments you did – use screenshots or example scripts if needed. The purpose of this process is to give you an overview of the available tools to help you implement and execute the actual tests in part 2 of the assignment.

Tools

- https://www.npmjs.com/package/c8 (Test coverage)
- https://www.npmjs.com/package/chai (Assertion library)
- https://www.npmjs.com/package/coveralls (Test coverage service integration)
- https://github.com (Source code repository)
- https://github.com/features/actions (GitHub Actions)
- https://www.npmjs.com/package/jest (Test framework)
- https://www.npmjs.com/package/jest-chain (Utility library to make Jest tests chainable)
- https://www.npmjs.com/package/jest-extended (Extension for Jest assertions)
- https://www.npmjs.com/package/mocha (Test framework)
- https://www.npmjs.com/package/mocha-lcov-reporter (Test reporter)
- https://www.npmjs.com/package/mochawesome (Test reporter)
- https://www.websequencediagrams.com/ (Tool for creating simple UML sequence diagrams)

Implementing the tests (in part 2)

In part 1, you are **not required (or recommended) to implement any tests**. In part 2, you will be required to implement *unit tests only for the applicable components of the provided library* (https://github.com/otula/COMP.SE.200-2023-2024-1). *Applicable*, in this case, means the files you select for testing in this plan.

Note that the library (https://github.com/otula/COMP.SE.200-2023-2024-1) includes a hidden .internal directory, which is to be excluded from any plans, tests, test reports and coverage reports. No additional material will be provided in the part 2 of the assignment. The back-end REST interface, detailed documentation of the application, front-end source codes or example data will *not* be provided.

The Test Plan

Your ultimate goal is to design the test plan. The plan is the only document you will be returning in the first part of the assignment. The overall style, format and structure for the test plan can be freely chosen, but certain crucial parts should be included in the document:

Cover page

- Name of the document,
- course ID and course name,
- the names of students as well as the student numbers.

Definitions, acronyms and abbrevations

- List of any definitions, acronyms and abbrevations used in the document.

Introduction

- Short introduction to the contents of the document.
- What is in the document?

- What is the purpose of the document?

Scenarios

- The descriptions of your end-to-end scenarios.
- What components of the E-commerce store will be tested, how and why?
- What tools will be used for the testing (if you can think of any based on the provided documentation)?
- A (UML) diagram of your scenario. The diagram can be, for example, an activity or sequence diagram, whatever feels most natural. The diagram can be high-level (abstract). If you know which tools are should to be used, describe how they are used in relation to your diagram.
- List the source code files you selected for your unit tests from the provided utility library (https://github.com/otula/COMP.SE.200-2023-2024-1). Use a prioritization method to validate your file selections. Remember your end-to-end scenarios!

Tools

- Description of the chosen tools, packages, libraries, etc. including possible links to web sources (tutorials, homepages, etc). Why did you select these tools and how did you test them? **This includes any AI tools you used.**
- How would the tools be used to test your end-to-end scenarios?

Tests

- What kind of tests are to be performed (unit, integration, …), and what is their importance for the end-to-end scenarios you described?
- What components of the E-commerce store will be tested and what will *not* be tested, and why?
- How will the test results be documented?
- If bugs or issues are found, how are they classified or categorized? Add a template of you bug/issue report if needed.
- When the tests are considered "passed"?
- How to guarantee adequate code coverage?

References

List of any and all references used in the document. Add URL links in references (as opposed to inline links within the text).

BONUS POINT (in part 2): use an AI

Using an AI is optional, but learning to use one, can save your time both in general, as well as in doing this assignment. How you use an AI, is up to you. It can be used, for example, for designing your tests or implementing them – just remember to document any tools you used as instructed in the task assignment.

If you want the bonus point, a more in-depth look on the use of AI is required, and you should **add a seperate section for this in your test report (part 2 document).** You should consider the advantages and disadvantages of using AI tools **in the scope of the testing task described in the assignment**. Especially:

- Did you, in your opinion, find the tools useful? Was something lacking, could the tools somehow be better?
- Did the AI produce something you did not expect? If you used it for design, did it, for example, design tests you would not have considered yourself, or if you used it for implementation, did it produce (test) code you might not have tought yourself to include?
- Did the AI clearly not consider something it should have? Was something missing that you had to add yourself?
- Would you, in your opinion, consider the AI designed or generated tests reliable? Why? Why not?

Only generating text for the plan or the report (documents) will not award the bonus point. The usage must be in either deisgn or implementation.

Tools like Copilot (https://docs.github.com/en/copilot) and ChatGPT (https://openai.com/blog/chatgpt) are good, simple tools to begin, if you have not used AI before. ChatGPT has a free version that uses the older GPT-3.5, and Copilot has both a free trial and a free education license (https://techcommunity.microsoft.com/t5/educator-developer-blog/step-by-step-setting-up-github-student-and-github-copilot-as-an/ba-p/3736279 or https://education.github.com/discount_requests/application). A photo of a student card or screenshot of Sisu showing your enrollment to Tampere University should be enough proof to get your education license.

Tips for making the test plan

- There is no need to create excessive amount of end-to-end scenarios (that would be challenging with the provided documentation). A minimum of three (3) *well-described* scenarios will be enough.
- You can use your imagination to fill in the gaps, if you have difficulties in creating reasonable end-to-end scenarios with the provided documentation. There is no single right approach in designing your plan, and later, in implementing your tests, but remember to be consistent throughout your documentation.
- Remember prioritizing - you do not need to test everything, but make sure to describe why something will not be tested. Furthermore, ”lack of documentation” is a valid reason for *not* testing something, but do not overuse this particular excuse! You should consider the application testing as a whole despite the brief documentation.
- Check the source code for the provided library, it contains some basic examples of the functions. These might help you to understand the functions and their purpose, whether the source codes file(s) should be tested, and how they could be tested.
- **In the part two of the assignment your minimum testing environment must consist of GitHub Actions (CI pipeline) and Coveralls (test coverage reporting). Your project will be hosted in GitHub.** This is something you should take into account when designing your test plan.

- Use of a virtual machine is strongly recommended, especially when trying out various libraries and development tools (so that you do not break your running everyday configuration). Oracle's VirtualBox is a popular free choice.
- Use of AI (tools) is accepted both in design and implementation, but the use must be documented in the design plan. Producing documentation is accepted, but a word of warning: **ChatGPT hallucinations or excessive use of AI generated text in the design plan and/or in the test report will lead to point reductions**! The goal of the assignment is that you will also learn testing, even if AI already knows it.
- And finally, **remember that the documents are not made for you, they are made for others**. The reader of the document (e.g. a hypothetical new employee) should be able to understand your testing approach, prioritization, as well as the reasons for your tool and test choises, by simply reading your test plan (*without* knowning the details of this course assignment).

Returning the part 1 of the assignment

Save the test plan in PDF format with name test_plan.pdf. Zip the test plan and possible attachments. The zip file is to be returned to Moodle and must be named studentnumber1_studentnumber2_2023_part1.zip where student numbers are the students numbers of the group members.

Pay close attention that you include all the necessary files in the zip file and that you follow the naming conventions. Unclear / late returns / failure to follow the naming convention results in a failed assignment.