**Testing and GitHub setup guide group MD 13**

Testing:

I propose we use 5 different types of testing for our mini project. These include unit testing, integration testing, system testing, acceptance testing, and performance testing. The rest of this section of the document will outline each of the diffrent types of testing and ways I suggest we impliment them.

Unit Testing:

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually scrutinized for proper operation. This would be something like an individual function like this:

A screen shot of a computer

Description automatically generated

Where no other functions are called. And if another function is called, we can call a dummy result instead of the function that will always return the correct value.

So instead of this:

A screenshot of a computer program

Description automatically generated

We do this:

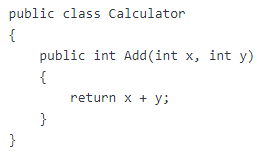
A screenshot of a computer program

Description automatically generated

In terms of testing incCounter.

How to write Unit tests?

We can both hard code the unit tests, something like this:



A screenshot of a computer code

Description automatically generated

Or we can use GitHub actions to write unit test. I personally am not sure how to do this. If no one else knows either. I will be happy to try and figure it out if we want or must go that direction.

Integration Testing:

To my understanding integration testing is a type of software testing in which various modules in an application are tested as a combined unit. Meaning we are taking different parts of our software that interact with each other and testing them as one module.

How to write integration test:

The following website gives a fantastic step by step guide to integration testing: <https://blog.hubspot.com/marketing/integration-testing>

Feel free to read the whole article but I will give a summery of each step below.

1. Define the scope of integration testing:

Find all components and modules that are going to be evaluated.

1. Identify test scenarios:

We need to lay out all of the scenarios that these components and modules will be used for and write tests for all of them.

1. Prepare test data:

Generate the appropriate data that will be employed to run these tests. This test data should include all potential input conditions and must contain both valid and invalid information.

1. Create test cases (the actual tests):

Create test cases for each scenario. Create clear inputs, expected outcomes and pass/fail criteria.

1. Set up a test environment.
2. Execute the tests.
3. Examine the results and act as needed.

I don’t know if this type of testing can be done using GitHub actions. But I think it will be a lot easier to just create a standalone testing environment in normal code.

System Testing:

Once integration testing is done System Testing can begin. Now that we have tested all the different modules and components are working. We need to test the entire system as a whole and how all modules and components interact with each other. This is quite like integration testing just on a larger scale.

How Do You Write System Test Cases?

<https://testsigma.com/guides/system-testing/>

Here are sample steps to show you how to write a system test case:

1. Identify the system requirements: The first step is identifying the system requirements. This includes the functional requirements, non-functional requirements, and any other relevant requirements.
2. Create test cases: Once the system requirements are identified, you can create test cases. A test case should describe a single test scenario. It should include the following information:

* Test case ID: A unique identifier for the test case.
* Test case description: A brief description of the test scenario.
* Preconditions: The conditions must be met before the test case can be executed.
* Steps: The steps that need to be taken to execute the test case.
* Expected results: The results that are expected from the test case.
* Actual results: The actual results that were obtained from the test case.
* Pass/fail criteria: The criteria that determine whether the test case passed or failed.

1. Review the test cases: Once they have been created, they should be reviewed by a team of testers. This will help ensure the test cases are complete, accurate, and feasible.
2. Execute the test cases: Once reviewed, they can be executed. The results of the test cases should be recorded and analysed.
3. Report the test results: The results of the test cases should be reported to the development team. This will help the development team to identify and fix any defects in the system.

Performance Testing:

I personally do not see the need for this type of testing on a mini-project scale. I am also not sure how we would implement it. But of course, let me know how you guys feel about it.

Acceptance Testing:

<https://www.geeksforgeeks.org/acceptance-testing-software-testing/>

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not.

From what I understand this is done by the end user. So, we as developers “use” the product in its final state as if we were end users to make sure it satisfies all system and business requirements.

We can document this, but no actual code will have to be written for this type of testing.

GitHub:

Branches:

From my experience I think 2 branches would work best and prevent any type of lost code. We would have 1 main branch and 1 dev branch. Initially the main branch and dev branch are exact copies of each other. But all of us can only create branches from the dev branch and merge to the dev branch. We can then at intervals of let’s say every 2 days merge from the dev branch to the main branch. This though must be done by the project manager and the people who worked on that specific feature together. This makes sure that multiple people are checking the code that is being merged to our main product. This is only a suggestion though. Please let me know your thoughts.

Branch Naming:

I think the best way would be to create a branch in the following format:

namesOfPeopleWorkingOnFeature-Feature

E.g.

AlexPretorius-EmojiReactions

That concludes this document. Please let me know your guy’s thoughts and suggestions to any changes.