4 Testing

Testing is an extremely important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopting test strategy and instruments. In this overarching introduction, given an overview of the testing strategy. Emphasize any unique challenges to testing for your system/design.

4.1 Unit Testing

What units are being tested? How? Tools?

The units that will be tested are most of our function calls that have a critical render that will change the UI in react. We will use what is supposed to do and also have a few edge cases for most of the test cases. Example units would be our login functionality.

Each member in the group will create their own unit test cases for the functions that they create so the next step in the testing process will go smoothly.

The tools that will be used are react testing libraries Jest & Cypress.

4.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

The interface testing that will be done is for our calls to the web server that we are using. For the project it is AWS and we will use multiple different services such as, Coginto, Amplify, Dynamo DB, DataStore, DataSync, and CloudWatch.

We will validate any edge cases so these services do not crash and it will be scalable with our web application.

4.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

Integration testing will be important for inserting everyone's work into each other and validating that nothing from someone else's part will break another. The critical integration path would be from moving from page to page in the design from any part of the project. Also, validating that when we are working on a part of the website that requires a submission it will be able to store the item as a draft.

4.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

Inventory:

- Add/Edit Inventory items
- Auto increment inventory amount by shipment
- Filter and search inventory list

Shipment

- Add/Remove/Edit shipments in list
- Filter and search shipment list

4.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure they do not break? Is it driven by requirements? Tools?

Any pushes that affect old functionality will run through our CI/CD pipeline inside of gitlab. The CI/CD will have to be updated with the new additions.

4.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

We would involve our client in the acceptance testing by allowing them to use our project and design. Since the clients are testing it, this ensures that the software meets and performs according to the client's expectations.

4.7 Security Testing (if applicable)

N/A (handled by Amazon)

4.8 Results

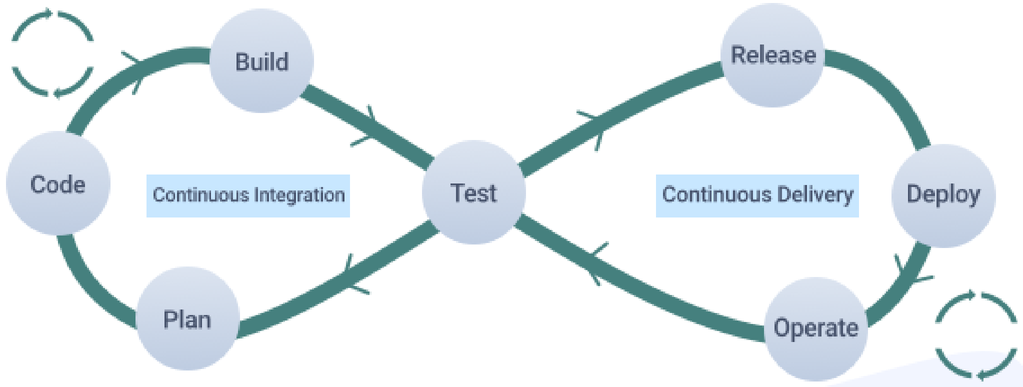
What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

Our test-implementation phase concludes on the 9th, so we haven't completed our testing requirements yet, but once we are done, we will ensure that we have tests that meet our requirements for handling shipping and inventory, namely incrementing and querying inventory counts and adding and searching shipment records. We will expect all tests to pass, and will modify our code to conform to specs if tests are failing.

XENONSTACK

CI/CD

(Continuous Integration/Continuous Delivery)


Build
Release
Code
Continuous Integration
Test
Continuous Delivery
Deploy
Plan
Operate