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.

- The unique challenge we face as a team is coming up with the best method for testing how well our GIS map satisfies the functional requirements and nonfunctional requirements that our client has given us. For example, testing how interactive our map is and whether it satisfies the client's expectations requires qualitative assessments of the features available with our application.

4.1 Unit Testing

What units are being tested? How? Tools?

- Our QGIS testing will be focused on getting individual features to work. We want each feature to be a map layer, so each individual feature would be a unit. For example, watersheds and grids on the map, precipitation and weather data overlaid, etc...
- Testing database api calls. We want to put our data in a database and make sure we can access it through the frontend web app and in QGIS. We would have to test that we can call individual pieces of data, as well as test that other data isn't corrupted because of that.
- Our web application will have unit testing through the frontend. Making sure all UI features work as intended with test values.
- We would use IDE testing tools with each of these. Our tech stack includes JavaScript, HTML/CSS on the frontend, Python with QGIS, and PHP in the backend. Since most of our unit tests would be run through the frontend, we would use VS Code or whatever IDE we pick to run unit tests using their specified frameworks/libraries. We would also use Postman to test on a mock server and test backend.

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?

One interface we will have to use is a maps interface. We have defined what our map interface will need to be, and we should now have ways we can test it. Testing the interface will probably consist more of testing communication between components. Depending on which map implementation we decide upon, we might end up having specific test cases be different:

- Verifying that functions within the interface are being called correctly (mocking)
- Checking that data retrieved is being processed efficiently (responsiveness of application)
- Verifying integrity of data actually being displayed (interface is actually doing something with data)
- Testing that the map view can be effectively navigated (seeing that our maps framework is reliable)

Another interface we will have to implement and test is how our data will be obtained. We will need a solid model for storing and obtaining data when requested. Testing our implementation of the data storage interface will primarily consist of:

- Verifying that data is being obtained and is present in a database (manually)
- Having a well defined set of APIs (test endpoints thoroughly)
- Testing actual communication between frontend and backend (verify data received from frontend API calls)
- Verify that backend controllers and services are communicating with database (mocking)

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?

- Our two critical integration paths for our design is integrating QGIS into a web application for the front end and integrating the database into a usable format for the back end. The first one is critical because the client has a requirement for the data to be in a visual format that is visible in a web application. The criticality for the second path comes from that if we can't get the right data from the database the map will no longer have correct or accurate data, which fails one of our requirements. We will test the QGIS by comparing what we know should be on what location and seeing whether the data is correct. For testing the database integration, we will make sure that all the data is parsed in correctly and no decrease in accuracy occurs. Most likely, other tools will not be used.

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?

- Our testing strategy for the entire system will be to go through the functional and non-functional requirements and check to see how well they are satisfied. We would be testing how the multiple integrations work in conjunction. We would use a unit/interface test that coordinates with a specific UI feature. Since our UI comprises an integrated maps component into our frontend, it would also be testing that integration.

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?

We will have to use separate unit tests for each task, continuous integration will be involved to make sure they do not break the old functionality, and keeping track of the old and new additions is

necessary. We need to ensure implemented critical features for both frontend and backend will not break. It will be driven by project functional, nonfunctional requirements and testing tools.

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?

- Given many of our functional requirements are UI based, it will be very difficult to have automated acceptance testing. Our plan is to instead manually evaluate the criteria with our client.

4.7 Security Testing (if applicable)

- Security testing may involve scanning our web server internally for any XSS or SQL injection that may take place. Also need to ensure that the integrity of our client's data is kept. The data points for precipitation, land cover, watershed boundaries, etc. are okay to be accessed, but any of the machine learning algorithms that generate the data need to be kept confidential.

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.

- One result of our testing is they allow us to make changes while ensuring previous functionality does not break. Another result is to help us to establish clear expectations from the start and use testing as a verification.

- This image shows the various levels of abstraction within our testing plan. It goes from unit testing, which is very selective individual component level testing, up to acceptance testing, which tests high level functional requirements.