Software Testing Plan

April 3, 2018

Sponsor:
Dr. Benjamin L. Ruddell

Team Mentor:
Dr. Eck Doerry

Team Members:
Luis Arroyo
Logan Brewer
Ryan Ladwig
Kelli Ruddy

Table of Contents

Introduction

This document describes the software testing plan for the Citizens Science Mobile App for Hydrology Reporting project. This mobile application will allow users to collect data very quickly and efficiently from already set up gauges and submit this data which will federate to the national HydroServer. The primary goal of this application is to increase the knowledge about how water is moving in small and ephemeral waterways where this data currently isn't being collected.

First, this document will describe the unit testing that will take place on the system. This will be done primarily through testing limits on submitted data for the different gauge locations as well as logging in validation and making sure users can view data submitted by themselves in comparison to others. Next we will describe the integration testing we will be doing to make sure the core components of our application are working together correctly. Finally we will describe our usability testing to ensure that this application will work as our client desires for people who use our application.

Unit Testing

This section describes the various unit tests that will be used to check the accuracy of our application. We will be testing this by submitting different values and seeing how the system reacts to these different value breakpoints.

The following is our unit testing plan of the different modules of our system.

- Unit Test Plan
    - Test Items
        - User login form
        - User registration form
        - User data submission form
        - HydroServer CSV file
        - Accuracy of Image Processing
        - Accuracy of user adjustments
    - Features to be tested
        - We will be testing these by checking for validation on logging in and registering. For the data submission we will be checking that the data is submitted in a valid location and be submitting valid data for the water height.
    - Test deliverables
        - Unit testing report
    - Testing tasks
        - Before testing, the task and the values used will be documented for re-testing.
        - During testing, we will record how each module reacts to the values used.

        ■ After testing, we will create the unit testing report that will show all of the above information compiled into one document.

    ○ Environmental needs
        ■ All the tests will be run on our Meteor application through a localhost server. We will also be using MongoDB to check submitted values.
    ○ Grading Criteria
        ■ A test will pass if the modules work as they should.
        ■ A test will fail if there is an error or the modules do not work as they should.
    ○ Resolution
        ■ If a test fails, we will fix the code until the test passes as we expect it to work.

Integration Testing

This section covers the integration testing of the system. Since our framework involves the web environment, all components must conform to HTTP and JavaScript format standards.

We broke up the integration testing of our system into key points to address the core components and figure out whether or not they are working or not.

Integration points:

Database Functionality: We will be using MongoDB in order to store the immediately collected values as well as user accounts. This is important because it will allow us to validate users and give users ID's for them to view data submitted by them in comparison to other collected data. We will also be using this to create our CSV file that will submit this data to the national HydroServer.

Test harness used: Checking that the MongoDB submitted values are accurate by going through and looking at each submission as well as checking that the generated CSV file has the information necessary for submission to the HydroServer.

Module Interaction:

Data Visualization: We will be using the submitted data to graph and displaying the data that is correct for the given user and the given gauge. This is important because it will allow us to give the user feedback on what they are submitting and see it in comparison to other data.

Test harness used: We will be checking that these values are correct by looking at the database and seeing the values that are submitted as well as the gauge they were submitted at.

Offline Caching: We will be submitting data from the Mongo database to the HydroServer while the user is online and offline. This is important because some of these water gauges are out in the wildlife and the user does not receive signal. So when the user submits the data while offline, it will obtain the water height, description of how it is outside, the date they submitted data, and their geocoordinates.

Test harness used:
We will be checking if the data is being submitted successfully while the user is offline by looking at the if the database and as well as grabbing their data when they are

Performance Testing:

We will run a performance test that will test how long the process of submitting and viewing data takes. If the speed to perform these two use cases is too long for our client than we will work on streamlining the submission process as well as the process of viewing submitted data.

Usability Testing

Usability testing is focused on the interactions between the software system and the end user. This testing is vitally important for our application because without an easy to use interface we will lose our user interest and will not be able to collect the amount of data points necessary to improve the knowledge we have of hydrological data. Our end user could be those who are not particularly patient or technically savvy. With usability testing, we will be able to improve our user interface based on participant feedback and continue testing until we feel as though users are easily utilizing all aspects of our application.

In order to conduct usability testing, we will using two groups of users who will be able to give detailed feedback on our user interface design choices. We will be creating a Google Form questionnaire that participants will be using once they have completed usage of our application.

The participants we will be utilizing in our usability testing can be outlined below:

1. Have 4- 6 users in no CS classes and no prior experience in computer science test our application and answer questions on a questionnaire. These participants will be used because they model our end user who may not be very technically inclined but must still be able to use our application with ease.

2. Have 4-6 users in at least 200+ level CS courses test our application and answer questions on a questionnaire. These students will be used in a test because they may be able to give us high level feedback regarding system architecture.

Each set of participants will have the same process which looks like:

    a. Users will walk around the engineering quad where there will be 2-3 gauges.
    b. The user will receive notifications letting them know that they are near a gauge.
    c. The user will take a photo and submit this photo.
    d. The user will be able to see their submission on a chart for that specific gauge.
    e. The users will give feedback on the questionnaire about their experience using the application.

The main difference between the two groups is that we will only run one test with CS students whereas we plan on testing with users with no CS background multiple times until we feel we have a easy to use interface. Tests will be run multiple times in order to ensure optimal usability and update our interface of our application based off of feedback from participants.

The questionnaire given via a Google Form will have questions which will provide the team with feedback that will help refine our application.

Some questions may include:

1. How would you rate your overall experience(1-5). Please justify your rating.
2. Were any aspects of the application interface unclear?(i.e. Could you not understand what was being asked of the application or how to interact with any aspect of the application)
3. What recommendations do you have to improve this application?
4. What did you find most difficult when using this application?