System and Unit Test Report
Smart Irrigation-Front-end
November 29, 2015

System Tests:
Sprint 1:

1. As a developer, I want a stable webserver to host dynamic content on. □□

   Scenario :
   - Install server software such as CentOS
   - Install Latest versions of Java + Groovy
   - Install Latest version of Grails
   - use the command “grails run-app” inside the server folder that contains the grails-app sub-folder
   - Testing for connectivity is required

2. As a developer I want remote access to the server for developing in a real server environment.

   Scenario :
   - Remote server access can be accomplished by port-forwarding the port 8080 to the outside using NAT.
   - This port should be mapped to 80 from the outside world because of the server being a web-server.
   - SSH can also be setup by forwarding port 22, allowing for the connection of a SSH client for development on the server.

Sprint 2:

1. As a developer, I want a finalized layout of the website so that data pushed from the back end can be displayed in intuitive ways.

   Scenario :
   - Log onto the website using credentials provided to you.
   - Navigate to the home page to see content.

2. As a developer, I want to be able to use the API created from the backend team to pull data from database and use represent the data with a google chart.

   Scenario :
   - Use a database client that connects with CURL or HTTP to their server at smart-irrigation.elasticbeanstalk.com
   - Accept and octal-stream of characters from the website.
   - Fit the data received into the agreed JSON format.

Sprint 3:

1. As a developer, I want to be able to create a secure website where the user can access their garden.

    Scenario :

    - Secure account creation can be shown by logging into two separate accounts to show the different content in each.
    - Log into user: ian password: password
    - Logout of ian, login to user: admin password: admin and see the unique secure data.

2. As a developer, I want to be able to add a simple animation that will help in the design of the website. Also, I want to be able to style the website using Cascading Style Sheets (CSS).

    Scenario :

    - Start Processing (For instructions on installation follow the System Requirements document)
    - With Processing, open the file called “flowers.pde” inside the /Frontend/Testing/FlowerAnimationTest/Flowers folder
    - Run the animation,
        - If code is successful, then FlowerAnimation will be displayed on the monitor
        - If code fails, then FlowerAnimation will display errors and code will have to be modified.

3. As an administrator, I want to be able to be able to have access to the secure website and make necessary changes to modify the data represented on the website.

    Scenario :

    - Use grails server running on server box.
    - Change the data by changing the HTML and GSP documents in real time.
    - See the changes in real time.

4. As a user, I want to be able to see a graphic description that contains information about the temperature and moisture sensor readings. As a user, I also want to see a summary generated by the grails-app that should tell me when to water my garden again.

    Scenario :

    1. start by going to http://smart-irrigation.no-ip.info/
        - username: ian
        - password: password

2. After log-in, user should be on the home window
    a. user should be able to see the weather information
    b. user should be able to see temperature graphs displayed with Google Charts
    c. user should be able to see moisture graphs displayed with Google Charts

3. After user is done monitoring their garden information
    a. user should be able to log out
    b. user should have an update for next watering cycle

Unit tests:

First and foremost, the testing apparatus does not work outside the folder "\Frontend\SmartIrrigate\grails-app\controllers\example" since the module containing the tests must be run as a Grails controller, making the test have to be run on the server while its running. Therefore, for brevity, the testing controller will be provided in the /Testing folder but will not work.

Unit tests are as follows:

<u>Test Login Controller:</u>

This unit test runs the module for the login controller directly to get the current user, and check valid login information. It is compared with the already known logged in user to check for success.

<u>Test Database Client:</u>

This unit test tests the URL connection to the elastic-beanstalk database server. The two functions that are checked are get with range and the regular get. Both are tested against static input to verify success.

<u>Test Index Controller:</u>

This unit test tests the GetUserData function, and calls upon the functions in database client to get the data. The index controller has just this function and is tested against the data gathered from the raw database client with fixed parameters.

<u>Test MyUserDetails:</u>

This unit test is run with the test login controller when it tests the functions of MyUserDetails controller. My user details hold the account information associated with a given login authority. This information is passed to the testing platform to determine the success of the currently logged in user.