System and Unit Test Report
Smart-Irrigation Backend
Team: The Wubadubdubs
Nov 28th 2015

Sprint 1

A.) User Story 2 from Sprint 1 : As a developer, I want the data from the raspberry pi to be stored on a MySql database so I can easily query the data.

Scenario :

1.) Start SqlPullTest.pl with python SqlPull.py in same diectory.
    a.) Enter in a range of readings to request.
2.) User should see range of values reported from MySQL table.

Sprint 2

A.) As a developer, I want to be able to generate dummy data for testing so that I can continue development regardless of the PI sensor's state.

Scenario :

1.) Run RandCSV.py
    ...

B.) As a frontend developer, I want to be able to access data through an API so that I can provide functionality to the user.

Scenario :

1.) Enter the /smart-irrigation/Backend/QA directory.
2.) Open 'testlist' and ensure the tests on the API you'd like to run are present.
    a.) Tests can be viewed in the /Tests subdirectory of QA.
3.) Execute TestHarness.py in the QA directory.
4.) terminal should report the queries performed through the API and their successes and failures.

Sprint 3

A.) As an administrator, I want to implement security into our API so that only the frontend has access to the information stored in the database.

Scenario :

1.) Enter the /smart-irrigation/Backend/QA directory

2.) Open the ‘testlist’ file.

a.) Ensure the testlist contains the tests httpauthentication_bad and httpauthentication_good.

3.) Execute TestHarness.py in the QA directory.

a.) Observe the terminal output for httpauthentication tests. good will use a correct username and password. bad will not.

b.) good should result in observed output “All Hail Bob!”

c.) bad should result a failure to login.

4.) Examine TestingLog_<yourdate>_at_<yourtime> and ensure all tests completed successfully.

B.) As an administrator, I want to implement security into our API so that only the frontend has access to the information on the database.

Scenario :

1.) Enter the /smart-irrigation/Backend/QA directory

2.) Open the ‘testlist’ file.

a.) Ensure the testlist contains the tests httpauthentication_bad and httpauthentication_good.

3.) Execute TestHarness.py in the QA directory.

b.) Observe the terminal output for httpauthentication tests. good will use a correct username and password. bad will not.

c.) good should result in observed output “All Hail Bob!”

d.) bad should result a failure to login.

4.) Examine TestingLog_<yourdate>_at_<yourtime> and ensure all tests completed successfully.