Herb Controller: Version Control and Testing Standards

Vinicius Petrelli Cicerone, Dylan Lafleur, Paxton Plum

13 Dec 2019

Version Control Standards

For the Herb Controller version control the group decided to use git, more specifically the open-source framework Github. Our repository can be found at the following URL: https://github.com/DonPetrelone/HerbControl.git

- **Cloning the Repository**

  In order to clone the repository into your machine you need to have collaborator privileges and also have access to the login credentials. Once you are at this point run the following command in your shell:

  git clone https://github.com/DonPetrelone/HerbControl.git

  Afterwards you will be prompted to type the credentials.

- **Merging Branch**

  If any changes were made in the current branch, it is necessary to commit them before merging. Please view the section on commiting code.

  In order to merge first switch to the branch you want to merge, merge and then you can delete the previous branch. Example shell commands for merging into the master branch:

  git checkout master

  git merge branchname

  git branch -d branchname

- **Create a New Branch**

  There are two approaches to creating a new branch. You can create the branch and switch to it with two seperate shell commands, or combine both into one command. The first approach is as follows:

  git branch branchname - where branch name is the name of your branch

  git checkout branchname

  The second approach is as follows:

  git checkout -b branchname

- **Pulling a Branch.**

  In order to incorporate changes to your current branch type the following command in your shell:

  git pull

  You may be asked to type in your account credentials. After this you should have all the most recent changes.

- **Committing a Branch**

  Before committing it is necessary to pass all unit tests for that piece of code. Please view testing standards and how to run unit tests. It is also important to not commit directly to the master branch before code review, so please commit to your side branch and then create a pull request. Make sure all commit messages follow the standards. From your shell use the following commands:

  git add filename - filename is the name of the file you want to add

  git commit -m "message"

- **Creating a Pull Request for Code Review**

In order for the group to review code before committing to the master branch, we will use Github's code review tool. In order to submit code for review follow the steps:

- Make sure you have committed the code to a side branch and not the master branch.
- Make sure you have tested before commiting.
- In your browser go to the repository where you branch is and create a new pull request.
- Follow the prompted steps after that.

- **Other Important Standards**
  - Do not force push.
  - Commit messages should include a title and description like the example below:

    Fixed bugs

    The bug in the addition function was fixed by performing...
  - Make each commit a logical unit, meaning that each commit should have a piece of code that implements a single purpose. This will make it easier for debugging and testing.
  - Avoid indiscriminate commits. This means that you should not use "-a" when committing, instead you should specify the files you have made changes to. This avoids committing of content that was not meant to be committed.
  - Incorporate changes frequently. In order to have the most recent code versions locally, run git pull every so often.

Testing Standards

- **Unit Testing**

  The group will use the Python Unit Testing framework PyTest for this portion of testing. This framework will be using both for testing the code that will control the hardware and also for testing the code of the web page, which will be done using Flask a Python web development framework.

  In order to automatically run all unit tests, move into the directory where the tests are located using your shell terminal. Afterwards use the following command:

  pytest -v

  In order to run a specific unit test only first make sure the test method is located in its own file the run the following shell command:

  pytest "filename" -v

- **Integration Testing**

  After unit testing has been completed, each component in each story can start to be integrated. As each of these combinations is made the group will perform integration testing by treating each module as a unit and will build tests using the same criteria as was used for unit tests. The commands to run them using Pytest will be the same as the used to unit test.

  In the case that a certain functionality is ready and it needs to be integrated and tested with another functionality that is not yet implemented, Pytest provides a mock object plugin called pytest-mock. The group will use this when performing tests with units that need to communicate with our database, for an example.

- **System Testing**

For this phase of testing the group has decided to perform blackbox testing.

In order to perform this type of testing, first the group will determine some test inputs. Both valid and invalid are desired because the system has to be capable of dealing with both. The group will also determine what is the expected output from the given input. Afterwards, the test cases will be constructed and executed. Lastly results will be evaluated and compared to the expected ones. With this the group can make decisions if the system is ready or not for the next phase of testing.

- **User Acceptance Testing**

  We plan to run acceptance tests with our client following the conclusion of all other tests and the unofficial release of version 1.0. This will involve reviewing the final stories document corresponding to version 1.0, introducing all features to the client, and allowing him/her to test the application and provide feedback.

- **Accessibility Testing**

  The Accessibility tests will be run on the Web Accessibility Evaluation Tool (WAVE). The goal of the group is to make the developed interface compliant with the Web Content Accessibility Guidelines 2.1 (WCAG).

  WAVE provides an automated system to run tests that evaluate for WCAG compatibility. To run the tests simply access their webpage at "https://wave.webaim.org/" and paste the URL of the page to be tested. Afterwards, WAVE will display the detailed results.

- **Usability Testing**

  Test users will be recruited who do not have a technical background. They will not be instructed on how to use the system. User will be encouraged to express their perception of usability without our input on the matter. Sessions will be filmed. The target number of users is 5 per segment to be tested. Use of the webpage to control any hardware that

can be controlled from it will be considered a testing segment. Control of any hardware that must be controlled manually in a given release will also be considered a testing segment. Metrics will be established in order to make the results of usability testing less subjective. Some examples of these metrics are time on task, task success, and expectation matching (Foggia, 2018).

- **Security Testing**

  Vulnerability scanning can be performed automatically with Owasp. This can be complemented by a line by line review of the code looking for vulnerabilities.

- **Hardware Testing**

  In order to measure temperature of certain hardware components to check for overheating, the group will use a digital laser thermometer. A voltmeter will be used to measure the voltage across and the current through certain hardware component and make sure they comply with the standards. The group will also use light meters that are able to detect what light wavelength is being emitted and what is the light intensity in order to conduct the required light tests.

Programming Standards

The group will follow the Python Enhancement Proposal (PEP) 8 style guidelines for Python code. Please refer to their documentation for specific information.

References

Chacon, S., & Straub, B. (2014). Git Documentation. Retrieved December 11, 2019, from https://git-scm.com/doc.

Ernst, M. (2018, March 3). Version control concepts and best practices. Retrieved December 10, 2019, from https://homes.cs.washington.edu/~mernst/advice/version-control.html.

Foggia, L. (2018, November 19). Usability testing: what is it and how to do it? Retrieved December 11, 2019, from https://uxdesign.cc/

PEP 8 -- Style Guide for Python Code. (n.d.). Retrieved December 11, 2019, from https://www.python.org/dev/peps/pep-0008/.

What is Security Testing? Types with Example. (n.d.). Retrieved December 11, 2019, from https://www.guru99.com/what-is-security-testing.html.