**Software Testing Q&A: Assignment 3**

<https://github.com/TanHess/Software-QA-Assignment-3>

Name: Tanner Hess | NetID: tmh648 | GitHub: TanHess  
Bagley College of Engineering, Mississippi State University   
Professor Tanmay Bhowmik   
April 7, 2022

**Deployment pipeline:**

For my deployment pipeline, I used CircleCI. CircleCI is a continuous integration and continuous deployment tool that allows implementation of test automation throughout a project. In my implementation, I created a docker image within the CircleCI, downloaded all the requirements for my project, and ran my tests (Unit tests and html route tests). The use of this pipeline ensured that each time I committed to GitHub my new code did not break my old code (regression testing).

The greatest challenges to setting up this deployment pipeline were simply learning the syntax of the .yml file. Learning how to setup the environment and run the tests was the toughest portion of setting up CircleCI. This also means that after learning CircleCI it would be much easier to use (as with anything).

Circle CI is a useful tool as it makes regression testing very easy and the ability to collaborate through GitHub or other similar platforms even more effective. Using Docker through CircleCI makes setting up a testing environment and testing your app in a controlled way incredibly simple. While other CI/CD platforms exist, in my experience with CircleCI, it is made incredibly easy. Below is a screenshot of my CircleCI running on my project:

Graphical user interface, text, application, email

Description automatically generated

**Manual Testing:**

Test 1) Navigate to the website URL. When running on local server, this will be <http://127.0.0.1:5000/>. After going to the URL, ensure the webpage exists and has 3 text fields asking for “feet”, “inches”, and “pounds”. Next, enter the following numbers for the parameters: feet: 5, inches: 5, pounds: 123. Click the “Calculate BMI” button and ensure that the page displays: “Your BMI: 21; You are Normal for your size”.

Test 2) Navigate to the website URL. When running on local server, this will be <http://127.0.0.1:5000/>. After going to the URL, ensure the webpage exists and has 3 text fields asking for “feet”, “inches”, and “pounds”. Next, enter the following numbers for the parameters: feet: 5, inches: 0, pounds: 140. Click the “Calculate BMI” button and ensure that the page displays: “Your BMI: 28; You are Overweight for your size”.

**Unit Testing:**

For unit testing, Pytest (python testing framework) was used. The unit tests for this program were focused on the BMI\_calc function. In all, there were 18 unit tests. The Nx1 boundary testing method was used to determine the unit test cases. These tests were also run within the CircleCI pipeline to implement regression testing. In all, there ae 5 tests for underweight, 5 tests for normal weight, 5 tests for overweight, and 3 tests for Obese.

**Tool Description:**

Assignment-3, 100 Points   
   
   
Objective   
Develop testing and deployment plans that enable continuous deployment of an existing   
software system that is extended for web access. Create automated acceptance tests (end-to-end   
testing) and integration (regression) tests.   
Scenario   
You have been asked to create a web interface for the application you created for Assignment-  
2. The VP of engineering at your firm also wants to ensure that you can continuously deploy   
new features, bug fixes, and changes to the application. He also wants to ensure the quality of   
the overall system by adopting a quality assurance and test plan.   
You will augment the existing application and make it accessible via a web interface and build   
& document a deployment pipeline for the system using various tools and cloud infrastructure.   
Requirements   
Web Interface - Add a web interface to your the app you created for Assignment-2 and make   
it accessible via the Google Cloud Platform (e.g., container engine [vm], Google App Engine,   
etc.).   
Deployment Pipeline - Setup a deployment pipeline using continuous integration and delivery   
tools (can make your GitHub project public).   
Steps for pushing to production environment:   
1. Source control   
2. Continuous Integration   
3. Static analysis (e.g., code / style linter, static bug checker [e.g., SpotBugs]).   
4. Automated unit tests   
5. Automated end-to-end tests (at least one per functionality)   
6. Automated deploy to staging environment   
7. Manual push to production   
8. Connect 3rd Party Code Coverage tool —> https://coveralls.io/, others for coverage reporting

**Assignment**   
Project Report (submitted to Canvas as a \*PDF\* File): Write a report summarizing your   
efforts. Your report should consist of the following content:   
• (10 pts) Report should be Professionally Organized / Presented (content should not   
solely be a list of bullets that delimit the required content - write in prose / paragraph   
form). NO ORDER SPECIFIED – Include Name, NetID, and Github username, and   
link to your Github repository on first page of report [4 pts for including appropriate   
items on the first page, 6 pts for organization].   
   
• (20 pts) Discuss your Deployment Pipeline. Explain each step, tools used, and   
benefits provided [15 pts]. What challenges did you encounter (e.g., setup,   
implementation) [5 pts]?   
   
• (10 pts) Detailed setup and execution instructions (put as a separate SECTION in the   
report).   
   
• (10 pts) Test Cases for Manual Testing: Use Test Case Specification to specify 2 tests   
that should be performed during the manual testing phase [5 pts for each test].   
   
• (10 pts) Discuss automated unit testing (using unit testing tool) you performed.   
   
• (20 pts) Tool Description - briefly describe tools used in the process (purpose,   
functionality provided, ease of use, setup, and do you recommend others to use it?) [15   
pts, 3 pts each]. How does the tool integrate with the rest of the pipeline [5 pts]?   
   
• (10 pts) Google Cloud Platform Usage: Brief discussion of challenges [5 pts], and   
benefits [5 pts] using Google Cloud Platform.   
   
• (10 pts) Code Coverage Report: Include output from code coverage tool. Turn off any   
ignored lines or disabled functions from coverage analysis (i.e., coverage should   
indicate actual lines covered by tests and not ignored)   
   
   
   
Notes & Resources   
• Tools you may consider   
o Continuous Integration   
♣ CircleCI, TravisCI, Bamboo\*, Jenkins\*, Team City, others   
o End-to-End Testing / Acceptance Testing / Web Interface Testing   
♣ [PhantomJS | Nightwatch] —> Selenium, Fitnesse, others

o Backends/Hosting   
♣ Google App Engine, Google Kubernetes Engine, Google Compute   
Engine   
o Code coverage reporting   
♣ https://coveralls.io/, or others for coverage reporting   
• May use GitHub flow or similar to manage contributions (Main pristine branch with   
feature branches for each function). Can use fork and pull request flow.   
• Recommended to use GitHub project boards for task / issue tra

a