Psychic Waffle
Homework 2

Andrew Grant
amg2215@columbia.edu

Jake Weissman
jdw2159@columbia.edu

October 2, 2016

1 GitHub URL

https://github.com/jdw2159/furry-octo-fiesta

2 System Description

For this assignment we developed simple web app that keeps track of users who visit the website and enter their name. One page asks the user what his or her name is and upon submitting it, that information is stored in a database. The other page prints a list of all the people who have visited our site.

3 Technologies Used / Lessons Learned

3.1 Flask Framework

We created a very simple application using the Python Flask framework. Flask takes care of a lot of the low level networking and web server details; we wrote some python code for the business logic, and html code for what was to be displayed in the browser. It was rather simple to follow an online guide to set up a very basic flask application. We also both had experience working with Python virtual environments so that part was not especially difficult; nonetheless we were able to successfully create a Python virtual environment on both of our machines which enabled us to sandbox our Python app.

Flask has front end components that made it easy enough to render a couple of templates and display information by iterating through a list. Creating a display for users to input their name was fairly simple and reading that input was also rather straightforward; for this we created some rather simple html files.

3.2 Back End Database

Our next objective was to set up a persistent store for the user input. Both of us have experience with PostgreSQL so we were able to spin up a database on our laptop and add a table that has one text column to store a name. There was enough documentation online to figure out how to connect the app to the database, but installing all of the necessary packages was tricky and took some time.

3.3 Unit Testing

Once all of that was in place, we wrote a few lines of code to read and write to the database and the app was up and running! Once all of that was in place, we added some unit tests which was very straightforward. We took advantage of the unittest library for this. This library makes it super simple to add unit tests to separate files, which can subsequently be invoked from the command line. The suite instantly reported that all of our, albeit simple, tests passed. When it comes to continuous integration, having the ability to so easily run unit tests will be invaluable.

3.4 Static Analysis

Finally, we ran our code through pylint, a python static analysis tool. There were a few syntax errors but given the simplicity of the app there wasn't much interesting. Nonetheless, we learned about pylint's capabilities.