**Exercise 4: Solution Reflection**

Last Updated 9.4.2018

Name: Akaninyene Eyoh, Daniel Yan, Ulysses Yu

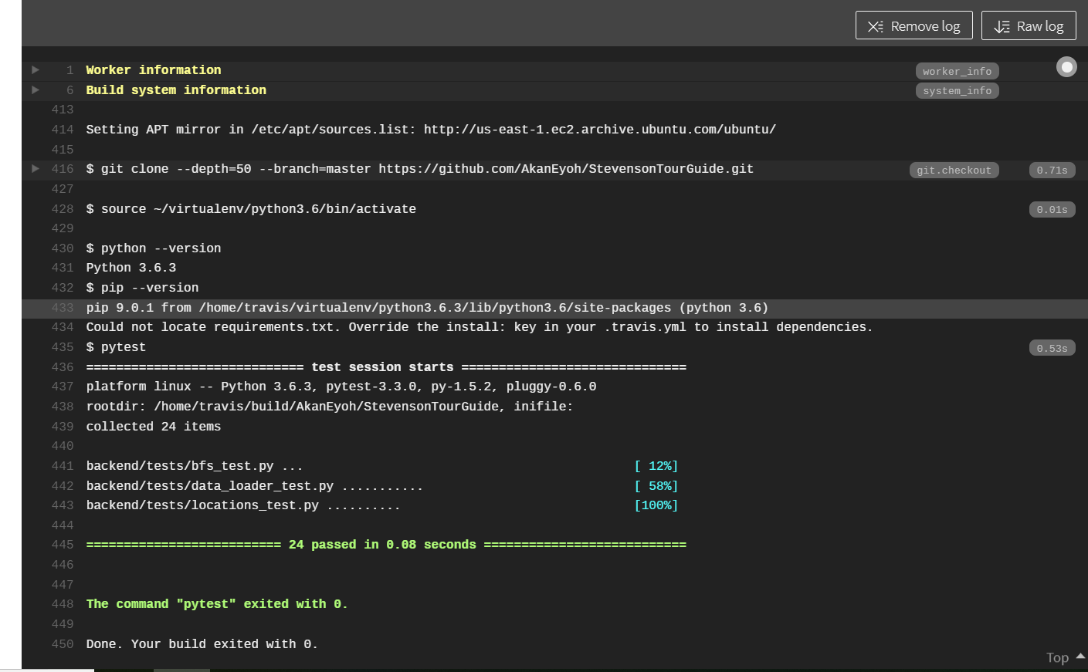
Date: 12-1-2018

Course: CS 278

**In the box below, show your prototype, either with a photograph or a sketch. Then, include a full, written description of your prototype.**



This prototype was our initial implementation of the backend that used breadth first search for the backend, which was at the first code review with Dr. White. At that point, we had only been working on the backend, so we had the files for the location objects (hallways, staircases, elevators), the data loader that contained the structure for the Stevenson Math Department (the project used the Stevenson Math department as a test case for Stevenson building), and the breadth first search algorithm, as well as tests for those files.

**In the box below, show your test of your prototype, either with photographs or sketches of your testing. Then, include a full, written description of your testing.**

The testing for our initial prototype consisted of unit tests for breadth first search, the data loader file containing the structure of the Stevenson math department, and the file for the locations objects. Unit tests were written with the pytest library and used Travis CI for continuous testing each time a commit was made.

**In the box below, describe in words or pictures your final solution.**

The final solution consisted of a backend that used Dijkstra’s shortest path to obtain the shortest path between two rooms in Stevenson, a Django server that would interface between the frontend and backend, and a Java Android app for the frontend.

The backend contains a map that translates room numbers to hallways, and a map structure that maps all the shortest paths, since the structure of Stevenson does not change. The backend would look up the shortest path and return it as a list of location objects.

The Django server by sends the room numbers from the frontend to the backend and sends the route from the backend to the frontend.

The frontend allows the user to enter in a start and end room number and makes a request to the Django server for the path, and prints out the path between the two rooms.

**Finally, tell us about at least one change you made to your prototype and why you made that change.**

At Dr. White’s suggestion after our first code review, we changed from using breadth first search to Dijkstra’s shortest path on the backend because the structure of Stevenson does not change. Thus, we would just have to run Dijkstra’s shortest path once to find all the shortest path and do a quick lookup to find the shortest path between two rooms, instead of search for a path each time.