Grumpy Old Men

Jason Konikow	jk4057
Christopher Thomas	cpt2132
Billy Armfield	wsa2113
Luigi Pastore Pica	lap2204

COMS 4156
Instructor: Dr. Gail Kaiser
IA: Sara Samuel

First Iteration Demo Report

The demo took place from 2:00 pm to 2:20 pm, on Thursday, November 08.

Challenges During Demo:

- Testing is not as cohesive or as integrated as possible (explained later in document).

User Stories Presented:

- Destination History:
  - The destination history was presented in an early stage. The condition of satisfaction was met for showing 10 previous searches.
  - For the first iteration there is no concept of a user so the search history is populated based on a master list of searches.
  - Iteration 2 will implement user classes and segregate search history into a “by user” basis.
- Map Overlay:
  - Map Overlay was demonstrated in its entirety and successfully displays columbia's campus.
  - The second iteration will improve upon the map overlay by introducing constraints on zoom and pan so that a student can not zoom out from campus view and can not pan beyond the campus limits.

Additional Changes Since Project Proposal:

Persistent Storage: Our team has elected to use Google's Firebase for backend storage as opposed to the proposed Mongodb database. The reasoning for this was that firebase has much cleaner integration in android studio than Mongodb.

Continuous Integration:

For CI, the team made the decision to utilize Travis CI. Testing has proven to be the area where the most learning has been accrued for us through mistakes. I will explain how our CI currently functions, and then i will explain how it will function moving forward.

Currently:
As of now Travis is configured to run a shell script that will run our unit tests through the gradle wrapper executable and output the entire log of the Travis run process to a log directory in our repository. In addition, gradle generates html reports of the results of the unit tests to its test directory. Unit tests themselves are utilized using the Junit framework with mocking performed by Mockito.

The issue:
While our CI infrastructure is in place, we ran into a lot of trouble with the unit tests themselves. Android is a notoriously difficult platform to integrate unit tests for we discovered this the hard way. The difficulty lies in the fact that android apps (ours included) have a considerable number of dependencies, including even the android framework itself. To access the methods to be tested in the first place we would need to instantiate the android activity that contains them, which simply is not allowed. This doesn't even begin to get into the problems with testing a method that both has java logic and android logic that rely on things such as screen touches and UI interaction that cannot easily be simulated from a command line.

The Solution:
We plan on refactoring our code to utilize Model View Presenter architecture which was created out of the need for making unit testing more feasible in android applications. This architecture creates a separation of UI based code, and behavioral code in such a manner that pure unit tests can be run on the behavioral logic with minimal mocking required, and no dependencies causing conflicts.

Github Repository:

https://github.com/jkon1513/ASE2018.git