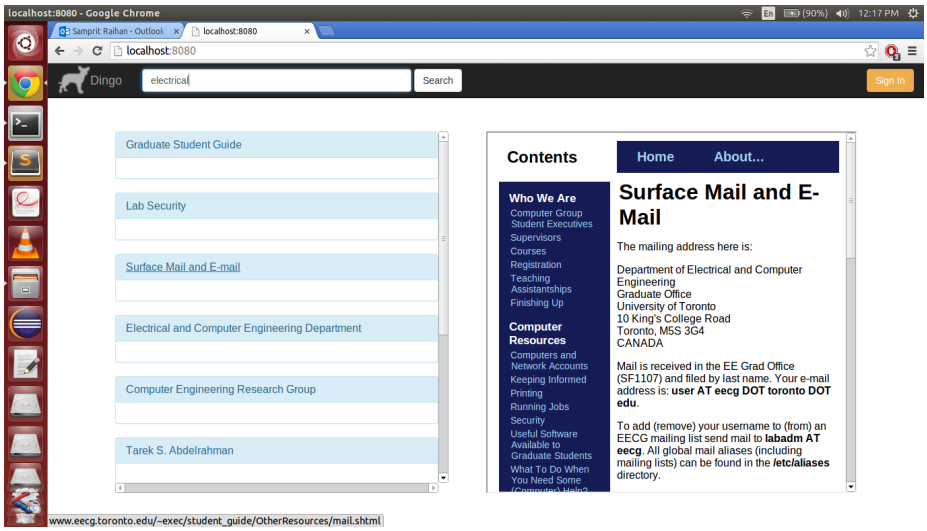
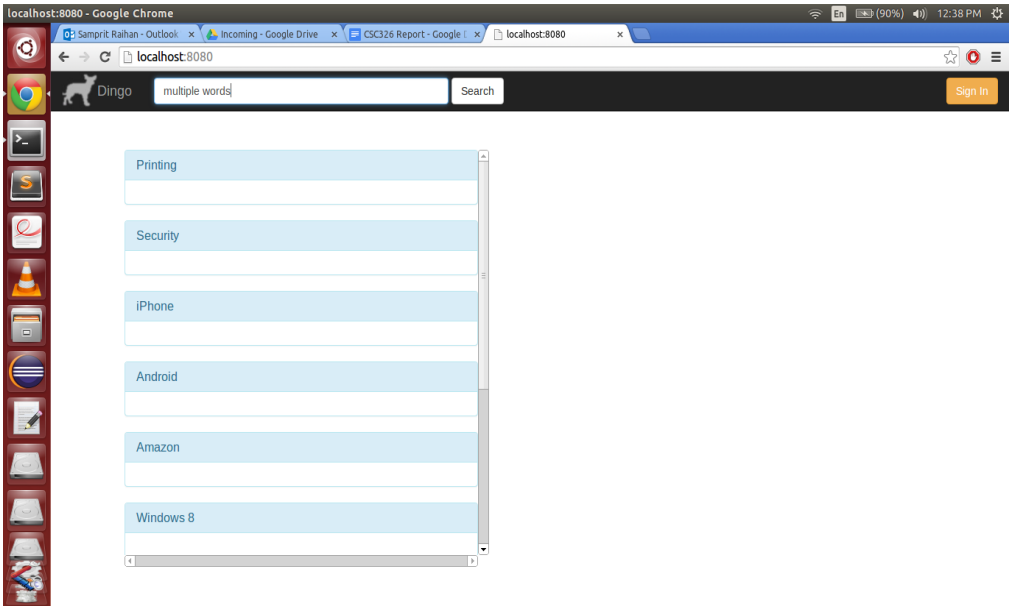
1.) Group:
    a.) Jon Erik Suero (suerojon) [998-171-954]
    b.) Samprit Raihan (raihansa) [998-138-830]

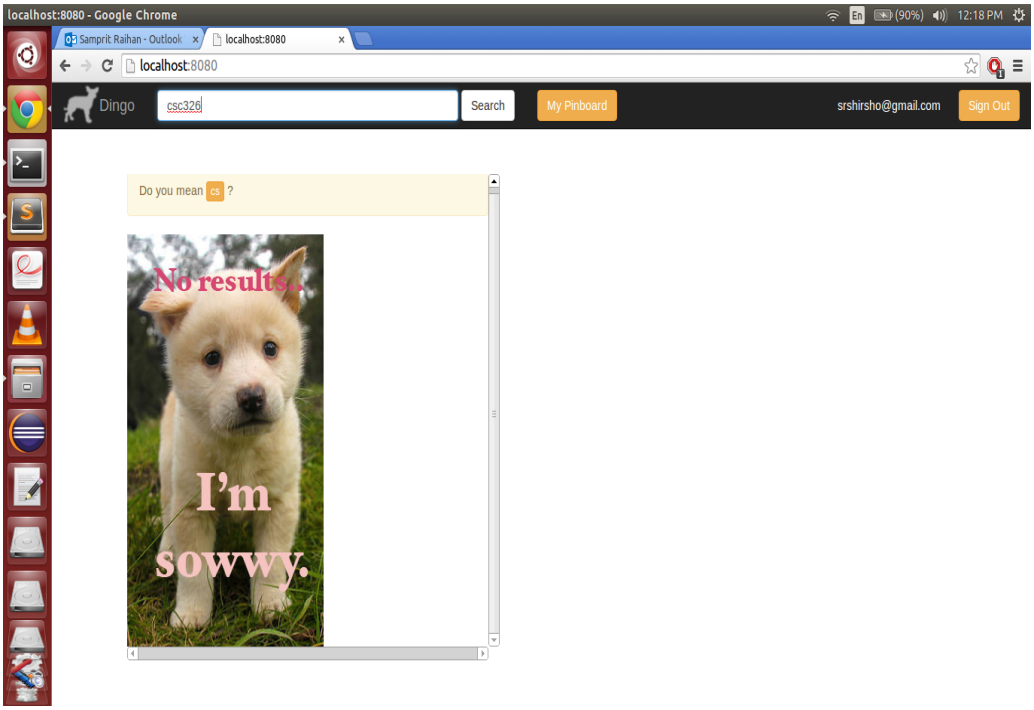
2) Design:
    a) For deployment:
        i) cd csc326/
        ii) python pre_deployer.rb
    b) For termination:
        i) cd csc326/
        ii) python terminator.rb
    c) Scroll to see 10 more URLs (using AJAX)



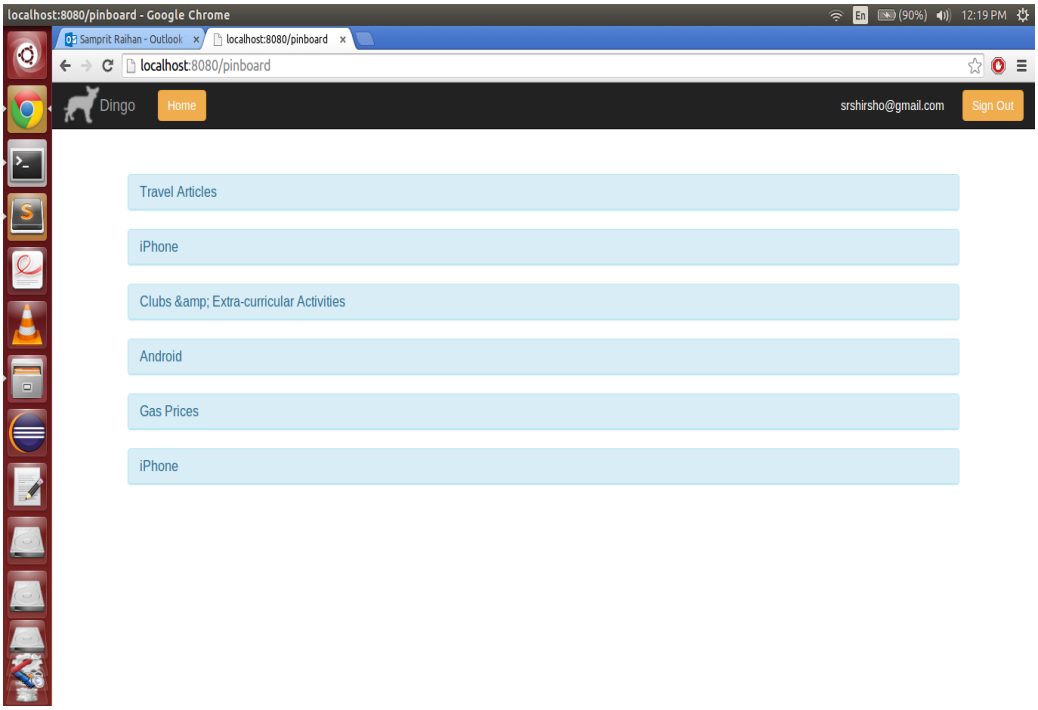
    d) Search for more than one word



e) Recommendation System: (Click the Recommended Search Query)



f) Pinboard: Bookmarking/pinning URLs from the search results



3) We used manual QA methods to test our application. We listed corner cases such as empty strings to make sure that the application will not break. (Explain your testing strategy during the development. Describe how you identify the corner cases.)

4) The lessons that we learned from this project:
    a) Decorators such as “@route” can be used to pair routes with functions.
    b) Easy but time-consuming tasks such as web designing could be better accomplished by using existing web frameworks instead of manual designing

5) Our answers:
    a) Things we would have done differently if we had to do it again.
        i) Easy but time-consuming tasks such as web designing should have been done on earlier labs.
        ii) Instead of iframes for previewing urls, we could have taken screenshots of top websites using our crawler. Loading one image/screenshot is much faster than loading a website especially if it has multiple large images.
    b) Things we would do if we had more time.
        i) Use a frontend framework such as Ember.js and Angular.js
        ii) Multi-threaded crawler (using MapReduce).
    c) Tasks took longer that we thought
        i) Writing Deployment and Termination Script: AWS takes a while to create/terminate instances. Therefore, during testing the scripts, we have to wait. It also took us a while to figure out how to run terminal commands in the server using another computer. Moreover, since “screen -dm python server.py” does not print error messages, it took us a while to realize that the server.py was not running at all due to errors.
        ii) Properly designing a web page with CSS took a while also because designing is an opinion-based task.

6) The material from the course helped us understand some key features of Bottle.py. For example, “@route” are decorators that maps the function to the specified route. The course also taught us how to manipulate files through python (for example, reading json files for Google Authentication).

7) It took us at least 5 days to create this Lab 4.

8) The most important parts of the project are the following:
    a) The web crawler: Search is a fundamental problem on the web today, and it was useful to learn the inner workings of crawling the web and mining for information.

b) Search quality: Besides crawling the web for merely gathering data, it is also important that we sanitize the data received and cater it in accordance to user preferences.
c) Integration with external platforms (Google login): This is important because the social graph on the web is expanding, and it is critical for web applications to leverage integration with more popular platforms to enhance its own growth.

9) The useless parts of the project are the following:
    a) Benchmarking should have been optional. Since in web development, productivity is much more important than speed/optimization. Web apps tend to morph frequently (i.e. Features are continually added, modified or deprecated.) Because of these frequent changes, the benchmarking will have to re-done multiple times.

10) Other feedback or recommendations for the course:
    a) Bottle framework is a too simple a framework and is not supported by a large community. Other more popular Python frameworks such as Flask or Django are better alternatives. They are simpler to use, and supported by larger of communities of developers.

11) Contributions of each member for Lab 4:
    a) Jon Erik Suero
        i) AWS with Deployment and Termination scripts
        ii) Redesigning using Bootstrap
        iii) AJAX for loading 10 new urls
        iv) Preview of URLs (iframe)
        v) Search for Multiple Words
        vi) Recommendation System for Typos
        vii) Refactoring/Code Segregation to Modules (i.e. Split code to different files. For example: “csc326/bottle-0.12.7/lib/”)
    b) Samprit Raihan
        i) Extracting page title from URLs and displaying them on the results page instead of plain URL
        ii) Implementing a Pinboard for signed in users who can bookmark any links from the search results that they find interesting. These are displayed under the ‘My Pinboard’ page