https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

*Online Store*

Documentation

Internet Programming

Albert Hambardzumyan

Fall 2015

https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

**Table of Contents**

|  |  |
| --- | --- |
| [Introduction](#h.m6u8vhq41itr) | 3 |
| [System Architecture](#h.cg24mu4ei374) | 4 |
| [RESTful API](#h.6tj7o6uzhwkk) | 4 |
| [Sentiment Analysis](#h.ob0b85pnpqes) | 5 |
| [Collaborative Filtering](#h.ayqtdk4hqmgf) | 6 |
| [Discussion of Potential Issues](#h.uetrkwb430bd) | 6 |
| [Data Source](#h.7sh7tfhsqwp2) | 7 |
| [How It Works](#h.aum4vfmzla8e) | 8 |
| [Home Page Overview](#h.3m87rosg3zyb) | 8 |
| [Item Page Overview](#h.46nwivzsk4k) | 8 |
| [How to Configure and Run](#h.3b3j157fjp4r) | 10 |
| [Challenges](#h.58fj2h9so7pg) | 13 |
| [Conclusion and Remarks](#h.x0fvsieanmw7) | 14 |
| [References](#h.t75xxc8htq04) | 15 |

# https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

# Introduction

TBA

# https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

# System Architecture

TBA

https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

# How It Works

## Home Page Overview

As the application loads it requests for a user id to be inputted. Note that this can be skipped, but in that case there will not be any collaborative filtering recommendations.Next, a search box will appear and allows searching for types of items. The items will be loaded from the database after the desired items is specified. Note, within this project scope we are dealing with only one type of the item, which is “Hotel”. This is to limit the scope of this project, and because we found a good source of data for hotels in New York. This is described in the “Data Source” section. For future reference, the system is able to handle other types of items, e.g. tourist attractions, cities in general etc.  
  
Clicking on one of the loaded items redirects to another page where the detailed information of that item is shown.

## Item Page Overview

As the page loads, it takes the ‘hotel id’ and ‘user id’ stored in the local storage by previous page. Based on this data, four independent requests goes to the server side to resolve the data related to this item and user.  
  
The ‘Rank’ value indicates how popular the selected item is, and the range of its value is between 1 and 5. The calculation is done in offline stage using the feedback data. The sentiment analysis module returns the number of positive, negative and neutral values. Based on the sentiment analyses, a simple arithmetic is done to calculate the ‘Rank’ value. The formula is: positive / total \* 5. More details description of the Sentiment Analysis module can be found in the ‘Sentiment Analysis’ section above. The sentiment analysis part can be tested directly using the admin panel (see the ‘How to configure and run’ section).  
  
The stars in the ‘Give a rate’ component indicates the user’s previous rating for this item. If the item has not been rated by the user, no star is filled and the user can rate the item. As a result of rating a hotel, a new call will be made to the server and the rating score will be stored in the database, which will be taken into account the next time collaborative filtering is done.  
  
The map section takes the coordinates of the item, and shows the item on the map. It sounds simple, however our original data source lacked the coordinates for the hotels, and we had to manually go through and add these. It was necessary not only for the map section, but also for computing recommendations for closest hotels with highest rank.  
  
The ‘Nearby ones with highest rank’ works in the following way: a message is passed to the server; the server then takes all the hotels and computes the distance between the item and other ones. All items within the range go to next stage of filtering. Allowed distance number is specified in a config file, and can be easily changed. Currently it is set to 1 km. In the next stage, hotels are filtering by their rank. First, we take all of the hotels having rank between 4 and 5. If the number of such hotels does not satisfy the minimum allowed number, then we consider ones with rank 3 and 4 too. This continues until we get the specified amount of recommendations, which is 7. This number of recommendations is specified in a config file and can be changed easily as well.  
  
The collaborative filtering section shows recommendations based on the user and item data. The collaborative filtering module returns items that the particular user can be interested in. More details description of the Collaborative Filtering module you can find in the ‘Collaborative Filtering’ section. Like with the sentiment analysis, the admin panel can be used to test it from the client side.  
  
Clicking on any of the recommended items will redirect to the page of that item. All the undergoing processes are available for inspection in the console both in the client and server sides.  
  
Due to our focus in finding data more appropriate for sentimental analysis and recommendations, our data does not include images for each item. Thus, the image section is static.

https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

# How to Configure and Run

|  |  |
| --- | --- |
| **Requirements** |  |
|  | Install python, make sure classpath is set, and that it runs $ pip install -U textblob $ python -m textblob.download\_corpora Install Java, make sure classpath is set, and that it runs Install NodeJS Install MySql |
| **Running** |  |
|  | Make configuration changes in api/app/config/config.js Important changes - path: the directory of the project, db configs Import the api/app/db/db\_dump.sql file to your database Download dependencies - npm install Run - env=dev npm start |

|  |  |
| --- | --- |
| **Testing** |  |
|  | Main route - <http://localhost:3000/api/v1/>  Python script test <http://localhost:3000/api/v1/python> Result - result :{"positive":0,"negative":0,"neutral":1}  Testing using admin panel Username: admin Password: admin  Sentiment Analysis Open the item page to check the rank value Before doing sentiment analysis, set the rank of the item to be zero:  *update hotels set rate=0 where id="usa\_new%20york%20city\_3\_west\_club";*   Check it by refreshing the item page. Do sentiment analysis. Refresh the item page. The item should have a rank.  Collaborative Filtering Open the item page to check that there are some recommendations for this user Before doing collaborative filtering, delete recommendations for that user:  *select \* from recommendations where userId=1;*  You see some recommendations. Now, delete them:  *delete from recommendations where userId=1;*  Check that there is no collaborative recommendations by refreshing the item page. Then, do collaborative filtering. Refresh the item page. The user should have recommendations based on collaborative filtering. |

Note, all the undergoing processes can be seen in the console, both on the client and server side.

# https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

# Challenges

One of the first challenges was the finding a decent source of data. There were difficulties in finding proper sources of data that were applicable for our system. The data we eventually found was adequate, though incomplete for our purposes, and we had to manually find and enter the coordinates for all of hotels. Additionally, there were a lot of unnecessary information and reviews unrelated to our hotels, and we had to spend a lot of time on sanitizing the data for use in our database.

Although the system consists of mainly two pages, it is quite complicated and a lot of data has to be transferred. For this reason, there was a need to create client-server communication. Both server and client side code is done in a way that it can be easily extended in the future. We spent a lot of time to set up appropriate structure on both client and server sides, and both of them are satisfying the standards of real product applications. Each piece of logic is separated in one module. Thus, the application consists of a number of services, controllers, templates, models, and configuration files.

The sentiment analysis for 259 hotels with 33861 feedbacks related to these hotels requires high computation cost. It was not possible to analyze all of them at once, due to machine CPU and RAM limitations. As a result, we had to do it for a maximum of 4-5 hotels at a time, which requires hard manual labor.

We ran into the same problem with collaborative filtering, and we had to do it manually for sets of 100 registered persons at a time, as our computer power is limited to do it for all the users at once.

We wanted the application to be interactive, and consequently it results in added complexity. On the other hand, we have ended up with a system that has many features, e.g. a rating feature, and an admin panel to demonstrate from the client side.

Another challenge was using the different technologies (Angular, Node, Python, and Java as mentioned earlier) and integrating them in such a way that they function as a coherent system.

Day by day, the code got longer and more complex, and in the end, we had thousands of lines of code. It was difficult to refactor and document it. However, we ended up with a pretty nice and scalable system. Also, we tried to make it as friendly as possible for demonstration purposes, providing logs of the each main actions.

# https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

# Conclusion and Remarks

In the end, our system does what we intended it to do. Therefore, we are happy with the result. While we took on a lot of work on the project, maybe too much, we learned a lot as well. Doing both sentiment analysis and two types of recommender systems gave us valuable practical experience and insight in two of main themes in this course.

For future work, the system should be tested more thorough. In particular, the collaborative filtering part needs testing with real users to be improved. Regarding the web page itself, more info on each hotel could be fetched and presented to give a more complete presentation. Such additional info should also improve the collaborative recommender, as the user easier could distinguish between different hotels and thus make more informed ratings.

https://lh5.googleusercontent.com/_dknrD8EsCZGVxicyxIz2TFLg32xTC55wU2SBEve2Nv8qgADwRdP0EA-dpliWX8oHzWoT9gxbufqz0APER4MiQXGvNkedMszyi0ppew2pr1rzGEhtiIevBXqFS6WZoXj3vo238IW

# References

[1] Hall, Marty, *Core Servlets and Java Server Pages,* Prentice Hall, Second Ed., Boston 2008.  
  
[2] Murach, Joel, *Murach’s Java Servlets and JSP*, Mike Murach, Second Edition, 2008