Milestone 4: Beta Launch and Final Project Reviews

Team #17/ Team 17

Project: CityInfo

CEN4010-001 Principles of Software Engineering Spring 2021

04/8/2021

[James Macdonald](https://canvas.fau.edu/groups/78058/users/148875): macdonaldj2016@fau.edu

Ethan Fisher: efisher2016@fau.edu

[LaDarrius Johnson](https://canvas.fau.edu/groups/78058/users/244622): Ladarriusjoh2019@fau.edu

[Nicholas Le](https://canvas.fau.edu/groups/78058/users/226120): nle2019@fau.edu

[Michael Romero](https://canvas.fau.edu/groups/78058/users/278315): michaelromer2020@fau.edu

[Dennis Sonjaco](https://canvas.fau.edu/groups/78058/users/188808): [dsonjaco2018@fau.edu](mailto:dsonjaco2018@fau.edu)

Product Summary:

Welcome to CityInfo. This is a single page website that would give the user a number of stats concerning any city within the US to start. For our initial MVP (Minimum Viable Product), we will have a page where the user can enter a city name and immediately after they submit their text entry, the system will display the current number of COVID-19 cases in that area in addition to the weather. For our coming revisions we will also include a great number of new stats such as: crime rates, map previews, geospatial analysis, entertainment, businesses, and even more! CityInfo takes a unique and innovative stance on the travel industry. Instead of simply being economically incentivized to advertise a destination, we discuss many of the aspects more concerned travelers would like to be informed about from a completely unbiased view. This lack of bias will come from the fact that we have no monetization mechanisms at the moment, we are simply taking information from the internet and giving it to the people in one central location free of tracking and cookies.

Usability Test Plan:

QA Test Plan:

Code Review:

Self check on best practices for security:

Self-check: Adherence to original Non-functional specs

Copy all original non-functional specs as in high level application document published at the very beginning of the class and then for each say DONE if it is done (which is expected and required); ON TRACK if it is in the process of being done and you are sure it will be completed on time; or ISSUE meaning you have some problems and

then explain it.

2.5 Code Review.

This is a part of the code for the Yelp API. This was written by [LaDarrius Johnson](https://canvas.fau.edu/groups/78058/users/244622) and reviewed by Dennis Sonjaco. Our coding style is java based in terms of the use of curley brace,indentation and naming of variables.

Comment:

Overall, the code is very well written. The naming of variables needs to be changed to java style just for the sake of consistency in the overall project. The comments are well placed in a way that it is very readable and easy to understand.

|  |  |
| --- | --- |
|  |  |
|  | var CITY = prompt("Please enter your name", "Harry Potter"); |
|  | var myurl = "http://52.91.156.204:8080/https://api.yelp.com/v3/businesses/search?location=" + CITY; |
|  | $.ajax({ |
|  | url: myurl, |
|  | headers: { |
|  | 'Authorization':'Bearer eTFFe41PY3fW6sZUph2s6jYZyYqA6Yi2\_5d88yYAUPSCts6Y8wEISL4FpaBjWU9ZALdvF53L07MKXEikvor2pzDtNbgwVl4MKVRQhDm5lj9tV2AD6p-mdvMH--SqX3Yx', |
|  | }, |
|  | method: 'GET', |
|  | dataType: 'json', // Lets change to TotalResults |
|  | success: function(data){ |
|  | // Grab the results from the API JSON return |
|  | var totalresults = data.total; |
|  | // If our results are greater than 0, continue |
|  | if (totalresults > 0){ |
|  | // Display a header on the page with the number of results |
|  | $('#results').append('<h5>We discovered ' + totalresults + ' results!</h5>'); |
|  | // Itirate through the JSON array of 'businesses' which was returned by the API |
|  | $.each(data.businesses, function(i, item) { |
|  | // Store each business's object in a variable |
|  | var id = item.id; //rename to JAVA style |
|  | var alias = item.alias; |
|  | var phone = item.display\_phone; |
|  | var image = item.image\_url; |
|  | var name = item.name; |
|  | var rating = item.rating; |
|  | var reviewcount = item.review\_count; |
|  | var address = item.location.address1; |
|  | var city = item.location.city; |
|  | var state = item.location.state; |
|  | var zipcode = item.location.zip\_code; |
|  | // Append our result into our page |
|  | $('#results').append('<div id="' + id + '" style="margin-top:50px;margin-bottom:50px;"><img src="' + image + '" style="width:200px;height:150px;"><br>We found <b>' + name + '</b> (' + alias + ')<br>Business ID: ' + id + '<br> Located at: ' + address + ' ' + city + ', ' + state + ' ' + zipcode + '<br>The phone number for this business is: ' + phone + '<br>This business has a rating of ' + rating + ' with ' + reviewcount + ' reviews.</div>'); |
|  | }); |
|  | } else { |
|  | // If our results are 0; no businesses were returned by the JSON therefor we display on the page no results were found |
|  | $('#results').append('<h5>We discovered no results!</h5>'); |
|  | } |
|  |
|  |
|  |
|  |
|  |
|  |

2.6

1)List major assets you are protecting

There is not really a list of assets we are protecting. Since our website is mainly for informational purpose we are only protecting a user’s profile with a password.

2) and 3)

To make our application work in the mold of SpringWeb and Spring Security, there is a configuration class, WebSecurity.java, that connects the UserDetailsService to Security, and enables the security settings for the various routes possible. The LoginController.java has all the endpoints for routing mapped to views which are Thyme Templates to allow for server side rendering of information and handling of login, registration, and logout commands. The MyUserDetailsService.java implements the UserDetailsServiceinterface which connects the required user database to the Spring Security components. The UserService gives access to the UserRepository and RoleRepository models stored in the database by making use of automatically generated JPA repositories.

At a lower level there are the models User and Role that are stored in the database and are made to match the User and Role objects needed for Spring Security. The attributes of User are id, name, and password which is encrypted. The role is just a lookup table for roles based on an id and name. These are also represented in the java program as Entities which is JPA annotation. Currently, all content is being served by the API’s we are calling so nothing is stored server side except user information. So there will be no content saved server side nor searching of the database outside of usernames which is handled by JPA

2.7

Self-check: Adherence to original Non-functional specs

Dependability Requirements: Done

Usability Requirements: Done

Performance Requirements: Done

Operational requirements: Done

Development Requirements : On Track (Code clean up and add more to the yelp api to accept Zip codes.)

Regulatory requirements: Done