EnviRepair

Milestone 4

5/17/2018

Gary Straub gstraub@sfsu.edu
Gayoung Kim
Gerren Penaloza
Justin Abarquez
Lily Linh Lan
Uzair Inamdar

Submission	5/17/18
Revised and Frozen	

Contents

1) <u>Product Summary</u>

1) Users shall be able to browse enviromental issues upon arriving to the website.
2) Users shall be able to search enviromental issues by category if they choose.
3) Users shall eb able to view an enviromental issues location on a map in the Detailed view.
4) Unregistered Users shall be able to register and become registered users.
5) Registered users shall be able to post enviromental issues
6) Registered users shall be able to view their posting history on their profile page.
7) Registered users shall be able to comment on issues that have been posted
8) City manager shall be able to change the status of current issue based on internal workflow.

Our product stands out from other products on the market in its open browsability. There are very few if any products that allow its users to brows current, past, and in-progress enviromental issues. EnviRepair

2) Usability Test Plan (Search)

Purpose: Users being able to search the website for posted issues, though a small part of the site, it’s a key part in the user expericence. Through these tests we want to find out how users feel about searching for issues as well as how they feel about the search results being displayed.
Problem Statement: We want to know how users feel about the placement, and the number of options given to refine searches. In addition to the presentation of the results presented.
Test Plan & Objectives: Users will search for enviromental issues, after browsing will answer a questionaire.
User Profile: In this test we hope to have users of all skill levels, though users with less technical experience will be preferred.
Method and Test Design: The best way to preform this test is going to be through focus groups.
Test Environment and Equipment: In order to emulate the most likely scenarios tests should be conducted with a set of laptops in conference room. This should give an average environment to the testing users, where most will be at their house, or office navigating the website planning for future outings or having arrived home to search for something they saw.
Test Monitor: The test monitor will be collecting user feedback (notes on overall topic of conversation in the focus groups), as well as playing the role of moderator in discussions.
Evaluation Measures and data to be collected: Most of the information shall be collected by the test monitor during the focus group. However, users will be given a survey before leaving to gauge how they felt on a numerical scale.
Legal Issues: There are no foreseeable legal issues, though user privacy is always a primary concern at EnviRepair.
Report: The report should contain at the very least a general sense of how users want the information displayed to them. For example: if 70% of users requested that the search not display items that did not pertain to their search but were displayed anyway to fill the void. In addition to a report with stats on how users felt based on the surveys given to them.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The search returned information based on your search	0	1	2	3	4
Searchbar categories are helpfull	0	1	2	3	4
Search items returrned are of an appropriate size	0	1	2	3	4

3) QA Test Plan

Test Obective: Here we intend on intending the functionality of our search algorithms in an effort to make sure data is being display appropriatly. This shall be done by creating a database with preset values where the search queries can be predicted accuratly and the results can be used to guage the effectiveness of the search algorithm.

HW and SW Set Up: This test can be done either on a local machine or on the live server. It is imprtant that the testing enviroment does not have any related merge conflicts.

Feature to Be tested: Search, search by catogory.

Number	Description	Test Input	Expected Output	Pass/Fail
1	Test % Search for the main search field.	San	8 Search results should be returned	Pass
2	Test category search	pollution	3 results should be returned	Pass
3	Test full search	Blank	Should return all issues in the database available to users (14 returned)	Pass

The Search functions above have been tested in edge and Chrome

4) Code Review

Coding style: Java/Javascript camelCase variable names

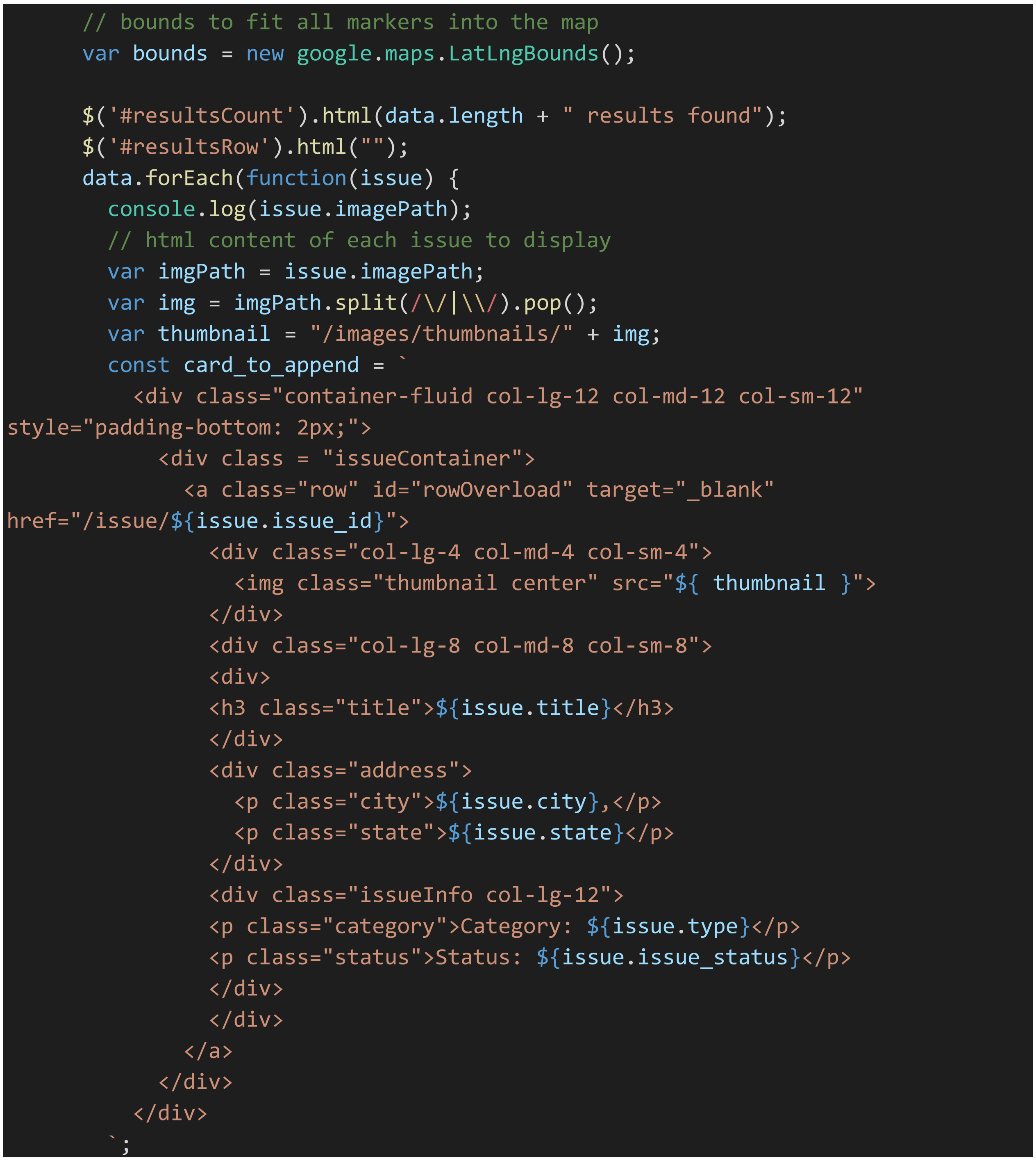
```
let filter = "all";

// Funciton to update the filter name on the button
const setFilter = function(element) {
  filter = $(element).text().trim();
  if(filter != 'All' ) {
    $('#filterMenu').text(filter);
    $('#searchBar').attr("placeholder", `Search By ${filter}...`);
  } else {
    $('#filterMenu').text("Search All");
    $('#searchBar').attr("placeholder", 'Search by Title, Description, City, or
Zipcode...');
  }
};


// Function to load and put search results on the page
const loadData = function(e){
  $('#results-count').html('');
  console.log(e.which);
  if(e.which == 13 || e.type == 'click') {
  const searchValue = document.getElementById("searchBar").value;
  console.log( searchValue );

  $.get('/search?keyword=' + searchValue +"&filter=" + filter.toLowerCase(),
function(data) {
      // console.log(JSON.stringify(data);

      const myCenter = new google.maps.LatLng(37.720460, -122.478124);
      const mapProp= {
        center:myCenter,
        zoom:14,
        scrollwheel: false,
      };
      const map=new
google.maps.Map(document.getElementById("googleMap"),mapProp);
      // geocoder will convert address into LongLat location on the map
      const geocoder = new google.maps.Geocoder();
      // label to display each marker on the map
      const labels = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
      let index = 0;
      // dict to map each marker location to a list of issues on that location
      let locations = [];
```



```
      // bounds to fit all markers into the map
      var bounds = new google.maps.LatLngBounds();

      $('#resultsCount').html(data.length + " results found");
      $('#resultsRow').html("");
      data.forEach(function(issue) {
        console.log(issue.imagePath);
        // html content of each issue to display
        var imgPath = issue.imagePath;
        var img = imgPath.split(/\/|\\/).pop();
        var thumbnail = "/images/thumbnails/" + img;
        const card_to_append = `
          <div class="container-fluid col-lg-12 col-md-12 col-sm-12"
style="padding-bottom: 2px;">
            <div class = "issueContainer">
              <a class="row" id="rowOverload" target="_blank"
href="/issue/${issue.issue_id}">
                <div class="col-lg-4 col-md-4 col-sm-4">
                  <img class="thumbnail center" src="${ thumbnail }">
                </div>
                <div class="col-lg-8 col-md-8 col-sm-8">
                <div>
                <h3 class="title">${issue.title}</h3>
                </div>
                <div class="address">
                  <p class="city">${issue.city},</p>
                  <p class="state">${issue.state}</p>
                </div>
                <div class="issueInfo col-lg-12">
                <p class="category">Category: ${issue.type}</p>
                <p class="status">Status: ${issue.issue_status}</p>
                </div>
                </div>
              </a>
            </div>
          </div>
        `;
```

The snippet of code shown here is actually an example of Code Style the team uses as refrence on their own pages. Though during our M3 review Dr. Petkovic mentioned the headers needed to be more detailed. In addition the HTML code shown towards the bottom is very confusing to read, it is very difficult to write HTML in a funtion like this so it is understandable when somoen hacks something together in this scenario (HTML within javascript). However the fact of the matter is that it would be best to maintain consistent indentation when and where needed.

Note: I have done my best to replicate the comments for this code. Many times when our team does code reeview we use slack. The system is great, though it does not keep a history for very long.

5) <u>Self Check on Best Practices for security</u>

The most important asset we are protecting in our database is a user’s password to access their account. We are using bcrypt to hash the user’s password once the password is submitted in the “register” form to create their account. Bcrypt has a compare function that we are using to check the hashed password against he plain-text password the registered user enters when they are attempting to log in. In our database, the password column in the user table is a hashed password stored as a string. On the form to submit a new issue, we are including checks to ensure the user can only submit an image if the extension to the file they choose is .jpg, .jpeg, .png, or .gif. On top of that, we also check the mimetype of the file to ensure that the minetype is png, jpeg, jpg, or gif. We use multer and regular expressions to ensure that these checks are done properly. An error is thrown onto the view that tells the user to only submit files of a specific type if the user tries to submit a file that is not an image. Multer also allows us to limit the size of the file a user tries to submit. For the search bar, we check the keyword the user enters against existing information in our database. The user can search by city, state, category, or status and we use SQL’s %LIKE functionality to best match their input to existing data and to return only valid search results. If the user enter invalid data, we output a message that tells the user their search keyword did not match any existing issues.

6) <u>Self Check: Adherence to original non functional specs</u>

a. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO). (**DONE**)
b. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. (**DONE**)
c. Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed (**In-Progress**)
d. Data shall be stored in the team's chosen database technology on the team's deployment server. (**DONE**)
e. Application shall be media rich (at minimum contain images and maps) (**DONE**)
f. No more than 50 concurrent users shall be accessing the application at any time (**DONE**)
g. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. (**DONE**)
h. The language used shall be English. (**DONE**)
i. Application shall be very easy to use and intuitive. (**In-Progress**)
j. Google analytics shall be added (**DONE**)
k. No e-mail clients shall be allowed (**DONE**)
l. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. (**DONE**)
m. Site security: basic best practices shall be applied (as covered in the class) (**In-Progress**)
n. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development (**In-Progress**)
o. The website shall <u>prominently</u> display the following <u>exact</u> text on all pages *"SFSU Software Engineering Project, Spring 2018. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). (**DONE**)