SW Engineering CSC 648/848
Section 01, Team 01
Spring 2018

R-Earth

Umang Mathur
(umathur@mail.sfsu.edu)
Ryan Liszewski
Rosalba Rodriguez
Chloe Zirbel
Lorenzo Moises
Taylor Marquez
Oleksandr Nibyt

Milestone 4

Document Version	Notes	Submission Date
Version 1.0	First Draft	05/17/2018
Version 2.0	Fix QA issues	5/20/2018

1. Product Summary

R-Earth is a web application that allows citizens of San Francisco to post their concerns about local environmental issues that need resolution. Environmental agents and officials can in turn access these listings in order to direct their day-to-day work.

List of features made available by R-Earth:

- Browse all posted listings, sorted by most recent date
- Search all listings by zip code with filtering options for category and status
- View the full details of an individual listing
- Google maps integration to allow users to easily visualize location and proximity of environmental issues
- Create and login to an account with R-Earth
- Submit a listing that describes a current local environmental issue
- Environmental agents are given an additional Dashboard tool to help them quickly review and update listings based on their work

R-Earth is unique in its approach to making data easily accessible and understandable to the common user. This solution focuses on providing an improved, intuitive interface for reporting an environmental issue. Citizens and environmental agencies can access crowdsourced data from throughout San Francisco in a rich format that heavily relies on location data and images. This facilitates a better understanding of the issue at hand and provides greater context to the environmental agencies who are responsible for resolving the problems reported by citizens.

R-Earth can be found at: https://csc648team01.herokuapp.com/

2. Usability Test Plan

Test Objectives:

The purpose of this test is to determine the usability of the application with regards to the submission of an environmental listing. That is, if an unregistered user visits the website and wishes to submit a report about an environmental issue, will they find the application easy to use. This particular scenario requires the user to visit a number of screens within the application. We hope to elicit feedback on the ease of use of all involved screens and thus identify any major issues with overall usability of the site, as well as with the submission page specifically. We intend to this usability test to act as an evaluation of the three major usability metrics: effectiveness, efficiency, and satisfaction.

Test Plan:

- System setup: using Windows 10 and Google Chrome browser
- Starting point: home page of R-Earth (found at the link below)
- Task to be accomplished: create and submit a listing about a local environmental issue
- Intended user: unregistered, first-time user (novice)
- Completion criteria: user sees their listing posted to the homepage
- URL: https://csc648team01.herokuapp.com/

Questionnaire:

	Strongly Agree	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
The homepage was easy to navigate	1	2	3	4	5
The submission page was easy to understand	1	2	3	4	5
The website is pleasant to look at	1	2	3	4	5

Additional Comments:	

3. QA Test Plan

Test Objectives:

The objective of this test is to evaluate the ability of the application to perform to specifications without any bugs with regards to the submission of an environmental listing. This test will be conducted in two iterations, the first on a laptop and the second on a mobile device to ensure uniform behavior across device type. A different browser will be used based on device type as well. This test seeks to expose any functionality issues at any point within the system, whether backend or frontend.

Hardware and Software Setup:

The web application is deployed on Heroku, with browsers on the laptop and the mobile phone acting as clients. The laptop is a Dell running Windows 10. The mobile device is an iPhone.

Postgresql was used as the database management system. Node.js was used to implement the backend. The front end was developed using Bootstrap and EJS.

Feature to be Tested:

This test addresses the submission feature of R-Earth. The submission pages displays a number of different input fields. In order to create a submission, all fields should be filled with appropriate values.

Actual Test Cases (Chrome):

Test #	Title	Description	Input	Expected Correct Output	Result
1	Valid Submission	User enters correct information for each field when creating a new submission	Title: "Air Pollution", Address & Zip code: press Auto Detect button, Category: "Air", Description: "There is air pollution in my neighborhood", Photo: jpg file (220 KB)	Application redirects to homepage where user can see their listing displayed with all the same information as entered on the submission page. Search by zip code shows the newly posted listing.	PASS

2	Missing Photo	User attempts to create a new submission without a photo	Same as above, but without the photo	Submission page remains open with all entered information. User is given a hint/warning that they must upload a photo with their submission.	PASS
3	Incomplete Text Field	User attempts to create a new submission with an incomplete address	Same as above, but Address & Zip code are both: 94618	Submission page remains open with all entered information. User is given a hint/warning that they must enter a valid address.	FAIL - page remains open, but no hint is given

Actual Test Cases (Safari):

Test #	Title	Description	Input	Expected Correct Output	Result
1	Valid Submission	User enters correct information for each field when creating a new submission	Title: "Air Pollution", Address & Zip code: press Auto Detect button, Category: "Air", Description: "There is air pollution in my neighborhood", Photo: jpg file (220 KB)	Application redirects to homepage where user can see their listing displayed with all the same information as entered on the submission page. Search by zip code shows the newly posted listing.	PASS
2	Missing Photo	User attempts to create a new submission without a photo	Same as above, but without the photo	Submission page remains open with all entered information. User is given a hint/warning that they must upload a photo with their submission.	PASS
3	Incomplete Text Field	User attempts to create a new submission with an incomplete address	Same as above, but Address & Zip code are both: 94618	Submission page remains open with all entered information. User is given a hint/warning that they must enter a valid address.	FAIL - page remains open, but no hint is given

4. Code Review

- **Coding Style**

    - Naming Conventions:
        - Camel Case for identifier names (variables and functions)
        - All names start with a letter
        - Global variables and constants in UPPERCASE

    - General rules for writing functions:
        - Put the opening bracket at the end of the first line
        - Use one space before the opening bracket
        - Put the closing bracket on a new line, without leading spaces

    - General rules for object definitions:
        - Place the opening bracket on the same line as the object name
        - Use colon plus one space between each property and its value
        - Place the closing bracket on a new line, without leading spaces.
        - Always end an object definition with a semicolon.

    - File Naming Conventions:
        - Use lower case file names only
        - Abbreviations should not be used in file names to avoid confusion

    - Other miscellaneous conventions to be followed:
        - Always put spaces around operators (= + - * /), and after commas
        - Always use 1 ‘tab’ for indentation of code blocks
        - Always end a simple statement with a semicolon

    - Documentation and comments:
        - Headers (of classes, modules, functions) – should contain comments which indicate basic information like purpose/function, methods, attributes
        - In-code comments for complex methods which contain code that may not be easy to understand for a new developer.

- **Sample Code Review**

Email Communication:

from: czirbel@mail.sfsu.edu to: umathur@mail.sfsu.edu subject: Code review request for submit report functionality Hi Umang I've recently completed the submit a report functionality for R-Earth. Can you please review the code at this link: https://pastebin.com/C55rzCVX Best, Chloe Zirbel Backend Lead, R-Earth
from: umathur@mail.sfsu.edu to: czirbel@mail.sfsu.edu subject: Re: [Code review request for submit report functionality] Hi Chloe I've reviewed the code submitted at the link provided. Overall, the functionality looks good and the code seems logically structured. However, in some places, the code needs to be refactored(I have commented at those places within the code). Once those fixes are made, I believe that this code can be pushed to production. Thanks for the great job done ! Regards, Umang Mathur Team Lead, R-Earth

Code sample with comments by the reviewer(highlighted in red)

```
/*
* Insert detailed comment here indicating the purpose of this file/class
*/
var geocoder, latitude, longitude, address, zipcode, picture;
//Object initialization  of locationSpinner should be done inside the
$(document).ready() method.
var locationSpinner = document.getElementById('locationSpinner');
var geocoder, autocomplete;
var locationSpinner, sendImage;

$(document).ready(function () {//Function comment missing
    geocoder = new google.maps.Geocoder();
    autocomplete = new
google.maps.places.Autocomplete((document.getElementById('address')), {types:
['geocode']});
    autocomplete.addListener('place_changed', function() {
        onAddressSelectedFromDropdown();
```

```
    });
    document.getElementById('btnDetect').addEventListener("click", function() {
        getLocation();
    });

    //No newlines within functions. Newline space only between 2 functions
    locationSpinner = document.getElementById('locationSpinner');
    $("#form1").submit(function(){
        submitData();
        return false;
    })
});


/**Asynchronously fetches current location using HTML5's Geolocation API. If
successful, calls the 'reverseGeocodeLatLng' method.*/
function getLocation() {
    if (navigator.geolocation) {
        setVisibility(locationSpinner, true);
        navigator.geolocation.getCurrentPosition(function success(position) {
            var currentLatitude = position.coords.latitude;
            var currentLongitude = position.coords.longitude;
            reverseGeocodeLatLng(currentLatitude, currentLongitude);
        }, function failure(error) {
            setVisibility(locationSpinner, false);
            console.log(`Error. Code ${error.code}: ${error.message}`);
            //These 2 error message strings should be stored as constants and
not typed in directly between the code
            var errorMsgDenied = 'Failed to fetch location.';
            var genericErrorMsg = 'Unable to fetch your current location !!';
            window.alert((error.code==1) ? errorMsgDenied : genericErrorMsg);
        });
    } else {
        window.alert('Geolocation is not supported by this browser.');
    }
}
```

```
//Insert comment to indicate the utility of this method here.
function submitData() {
    var title = $('#title').val();
    var category = $('.dropdown-select').val();
    var address = $('#address').val();
    var zipcode = $('#zip').val();
    var description = $('#description').val();
    var picture = sendImage;
    if(title && category && address && zipcode && description && picture) {
        $.post('/submit', {
                title: title,
                category: category,
                address: address,
                zipcode: zipcode,
                description: description,
                longitude: longitude,
                latitude: latitude,
                picture:picture
            },
            function(response){
                console.log("function");
                if(response.status == "success"){
                    console.log("Success");
                    window.location.replace("/");
                } else
                    console.log("status: " + status);
            });
    }
}
```

5. Self-check: Best Practices for Security

- **Major assets being protected:**
  - **a. User information:**
    - i. The user enters their full name, username(alias) and password to register. These credentials need to be protected. The website itself exposes only the user's alias via its interface. The full name of the user is not disclosed to other people using website. The password is stored in an encrypted format.
    - ii. Cookies are used for session management. A user is automatically signed out after 24 hours. This reduces the probability of other people with access to the user's computer submitting malicious reports on his/her behalf.
    - iii. The website runs on 'https', thereby providing greater security to the data submitted by the user and is less prone to 'Man-in-the-middle' attacks.

  - b. **Environmental Hazards' data:**
    Each reported hazard's data is stored on a password protected relational database(PostgreSQL) on a secure server hosted on a reliable cloud service (Heroku).

- **Password Encryption:** (Status: Done)
  The 'bcrypt' password hashing function is used to encrypt user passwords before insertion into the database.

- **Input data validation:** (Status: Done)
  - Search bar input validation: Only numeric or empty, maximum of 5 characters (No minimum enforced to enable 'LIKE %' search)
  - Submit report validation:
    - i. Image upload: File formats restricted to .jpg and .png. Max upload size: 10mb.
    - ii. Zip code: Only numeric, exactly 5 characters
    - iii. Report title: alphanumeric input only, no special characters, max length of 50 characters, empty title not permitted.
    - iv. Address: Max input length of 80 characters, empty address not permitted.
    - v. Description: Max input length of 300 characters, empty not permitted.

Note: The JQuery javascript library was used for validation. Some sample code to validate the 'Title' field of the report submission form:

```
<input name="title" type="text" required="required"
class="validate[required,custom[onlyLetter],length[0,50]] feedback-input"
placeholder="Title*" id="title" required="required"/>
```

6. Self-check: Adherence to Original Non-Functional Specs

1. 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
2. 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. - ON TRACK
3. Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed - ON TRACK
4. Data shall be stored in the team's chosen database technology on the team's deployment server. - DONE
5. Application shall be media rich (at minimum contain images and maps) - DONE
6. No more than 50 concurrent users shall be accessing the application at any time - DONE
7. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. - ON TRACK
8. The language used shall be English. - DONE
9. Application shall be very easy to use and intuitive. - ON TRACK
10. Google analytics shall be added - ON TRACK
11. No e-mail clients shall be allowed - DONE
12. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. - DONE
13. Site security: basic best practices shall be applied (as covered in the class) - DONE
14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development - DONE
15. The website shall prominently display the following exact 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). - ON TRACK