SW Engineering CSC648/848 Section 02 Spring 2018

Software Engineering Term Project:
Cleansweep Application

Team Number 11 – Local

Dylan Abrames, Team Lead: dabrames@mail.sfsu.edu
Brady Helkenn
Jia Jun Guan
Avi Mukherjee
Rodolfo Salgado
Frank Hood

Milestone 4
5-10-18

History Table	
Submitted for Review	5-17-18

Product Summary – Cleansweep Application

Functional Requirements

1. All users shall be able to search for Reports by zip code.
2. All users shall be able to search for Reports by Park name.
3. All users shall be able to view the status of submitted Reports.
4. All users shall be able to view the location of the park in submitted Reports through Google Maps.
5. Unregistered Users shall be allowed to fill out a Report, but must register to submit the Report.
6. Registered Users shall be able to post images when submitting a Report.
7. Registered Users shall be able to log in using a username and password.
8. City Users shall be able to view all Reports submitted by users.
9. City Users and Admins shall be able to view user information attached to submitted Reports.
10. City Users shall be able to adjust the Status of a submitted Report between open, in-progress, and complete.
11. Admins shall be able to delete submitted Reports, by using PGAdmin.
12. Admins shall not be able to edit submitted Reports.
13. Admins shall be able to delete user accounts, by using PGAdmin.

What is unique?

Any user can search and view details of reports on our website. Our competitors only offer the ability to report an issue - nothing else.

Final Deployed Product

CSC648/848 Team 11 Cleansweep Application, http://csc648-team11.herokuapp.com

Usability and Test Plan

Test Objectives

Verify the usability of the “Report Issue” function of the Cleansweep Application. Test validity of button placement per completion time and field placement by requesting a full report and noting any fields left empty.

Test Plan

System Setup – Windows 10 operating system, the Cleansweep Application open in the Windows Chrome web browser.

Starting Point – The Cleansweep Application home page, with a Registered User account logged in and a relevant and specific per test image on the computer desktop.

Task to Be Accomplished – Submit a full report based on the image provided.

Intended User – A person with no prior experience with the Cleansweep Application nor advanced computer skills; an average member of the public.

Completion Criteria – Report present on homepage in list of Recent Issues. Report Title, Park Name, Issue Type, Issue Description, and Image fields all filled.

Benchmark – Completion time: one minute.

URL of the System to Be Tested – http://csc648-team11.herokuapp.com

Questionnaire

Question 1

The Report Issue button was easily visible.

Strongly Disagree	Disagree	Neither Agree or Disagree	Agree	Strongly Agree

Question 2

The Report Issue button was intuitively placed.

Strongly Disagree	Disagree	Neither Agree or Disagree	Agree	Strongly Agree

Question 3

The Report Issue form fields were clearly labeled.

Strongly Disagree	Disagree	Neither Agree or Disagree	Agree	Strongly Agree

Question 4

The GUI (Graphic User Interface) was easy to use.

Strongly Disagree	Disagree	Neither Agree or Disagree	Agree	Strongly Agree

Comments:

QA Test Plan

Test Objective – Verify the function of all Submit a Report form fields. Verify Report data are properly stored and retrieved from the database. Search functionality is assumed to have already been tested and found functional and will be used to perform this test.

Hardware Setup – Current computer(s) with internet access, as needed to run the Google Chrome, Mozilla Firefox, and Apple Safari web browsers.

Software Setup – The latest two versions of the Google Chrome, Mozilla Firefox, and Apple Safari web browsers, each with all updates installed.

Test environment – Cleansweep Application with register user logged in.

Feature to be tested – Report Issue.

Security:

Report Title field

- Only alphanumeric characters
- Maximum of 40 characters

Park Name field

- Only alphanumeric characters
- Maximum of 40 characters

Issue Description field

- Maximum of 500 characters

Image Upload

- Only JPG or BMG format

Validity:

Park Name

- Map shows park entered

Issue Type

- Dropdown works as expected

System:

Lazy Log In

- Window opens and pauses submitting Report.
- Does not interfere with Report being added to the database

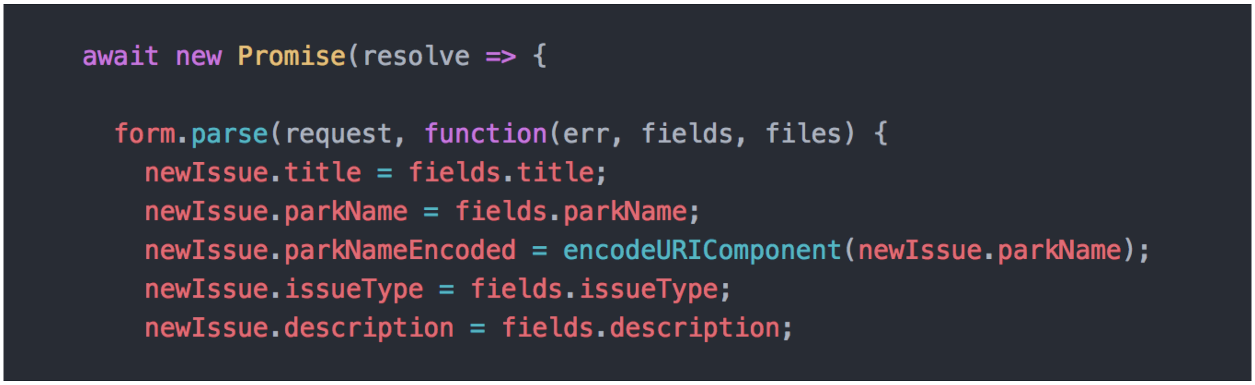
Test Cases

Test #	**Title**	**Description**	**Input**	**Expected Correct Output**	**Results**
1	Alphanumeric and reasonable length Report Title	Test input validation, Report Title	Enter non- alphanumeric characters into Report Title field. Click Submit. Clear and enter 41 alphanumeric characters. Click Submit. Clear and enter 40 alphanumeric characters. Click Submit.	Form will not submit on first or second attempt. Form submits on third attempt	PASS - Firefox
2	Alphanumeric and reasonable length Park Name	Test input validation, Park Name	Enter non- alphanumeric characters into Park Name field. Click Submit. Clear and enter 41 alphanumeric characters. Click Submit. Clear and enter 40 alphanumeric characters. Click Submit.	Form will not submit on first or second attempt. Form submits on third attempt	PASS - Firefox
3	Reasonable length Description	Test input length, Report Description	Enter 501 characters into Report Description field, then repeat with 500.	Form will not submit on first attempt. Form will submit on second attempt.	PASS – Chrome
4	Lazy Log In occurrence and function	Test Lazy Log in window interrupts submit process, and completes on log in.	Log out, click “Report Issue” and then “Report Issue” leaving the auto fill text, do not log in. Repeat and do log in.	Window opens, and only one Report is added to database.	PASS - Firefox PASS - Chrome

Code Review

Coding Style:

- Camel Case for identifier names, both variables and functions.
- Spaces around operators and after commas.
- 2 spaces for indentation of code blocks.
- Global variables written in all uppercase letters.



```
await new Promise(resolve => {

  form.parse(request, function(err, fields, files) {
    newIssue.title = fields.title;
    newIssue.parkName = fields.parkName;
    newIssue.parkNameEncoded = encodeURIComponent(newIssue.parkName);
    newIssue.issueType = fields.issueType;
    newIssue.description = fields.description;
```

Complex and Compound Statements:

- The opening bracket is at the end of the first line.
- There is one space before the opening bracket.
- The closing bracket is on a new line and without leading spaces.
- Such statements are not ended with a semicolon.

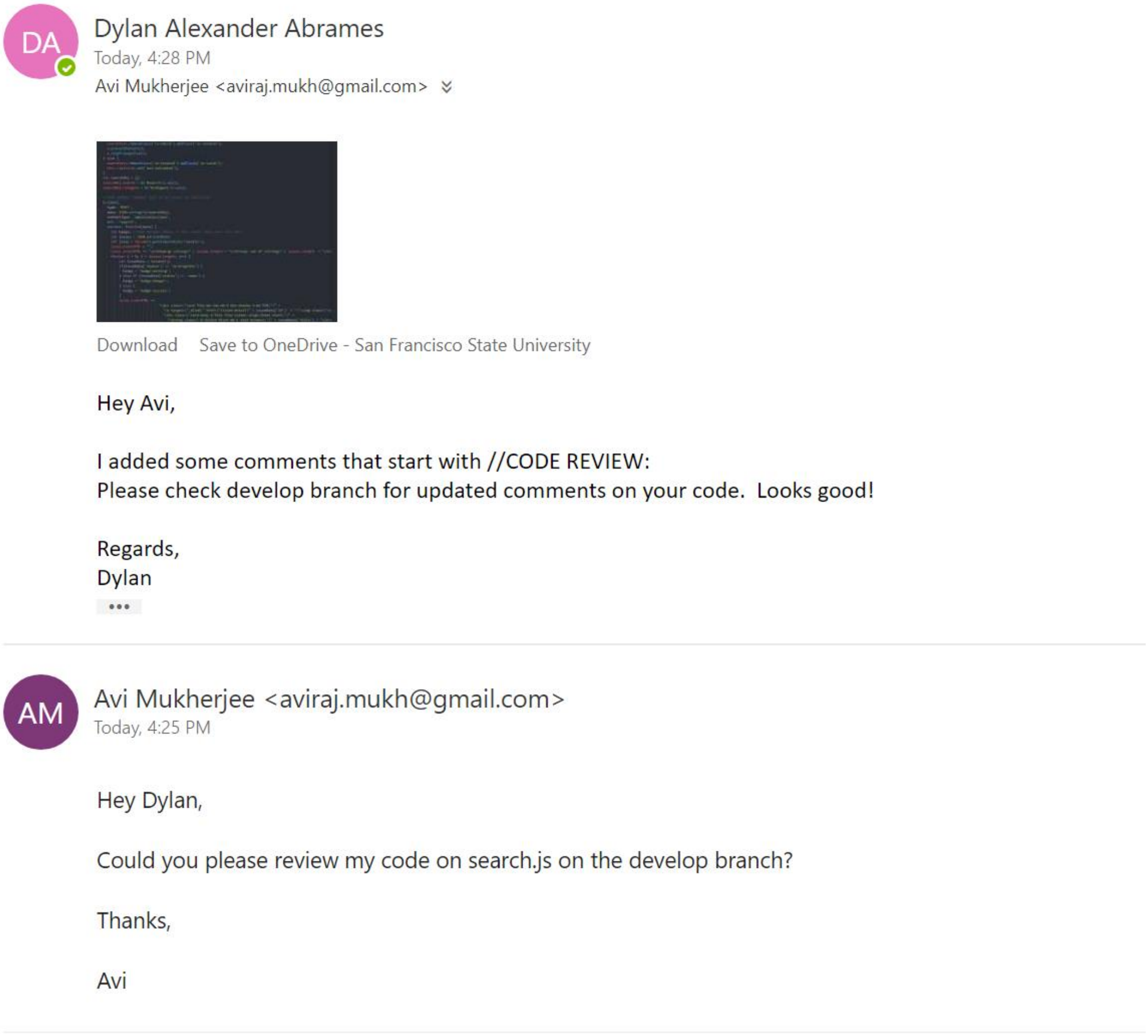
Object Definitions:

- The opening bracket is on the same line as the object name.
- There is a colon and a space between each property and its value.
- There are quotes around string values and not around numeric values.
- There is not a comma after the last property-value pair.
- The closing bracket is on a new line without leading spaces.
- Object definitions always end with a semicolon.

General:

- No lines are longer than 80 characters.

Complete Review



DA Dylan Alexander Abrames

Today, 4:28 PM

Avi Mukherjee <aviraj.mukh@gmail.com>

Hey Avi,

I added some comments that start with //CODE REVIEW:
Please check develop branch for updated comments on your code. Looks good!

Regards,
Dylan

AM Avi Mukherjee <aviraj.mukh@gmail.com>

Today, 4:25 PM

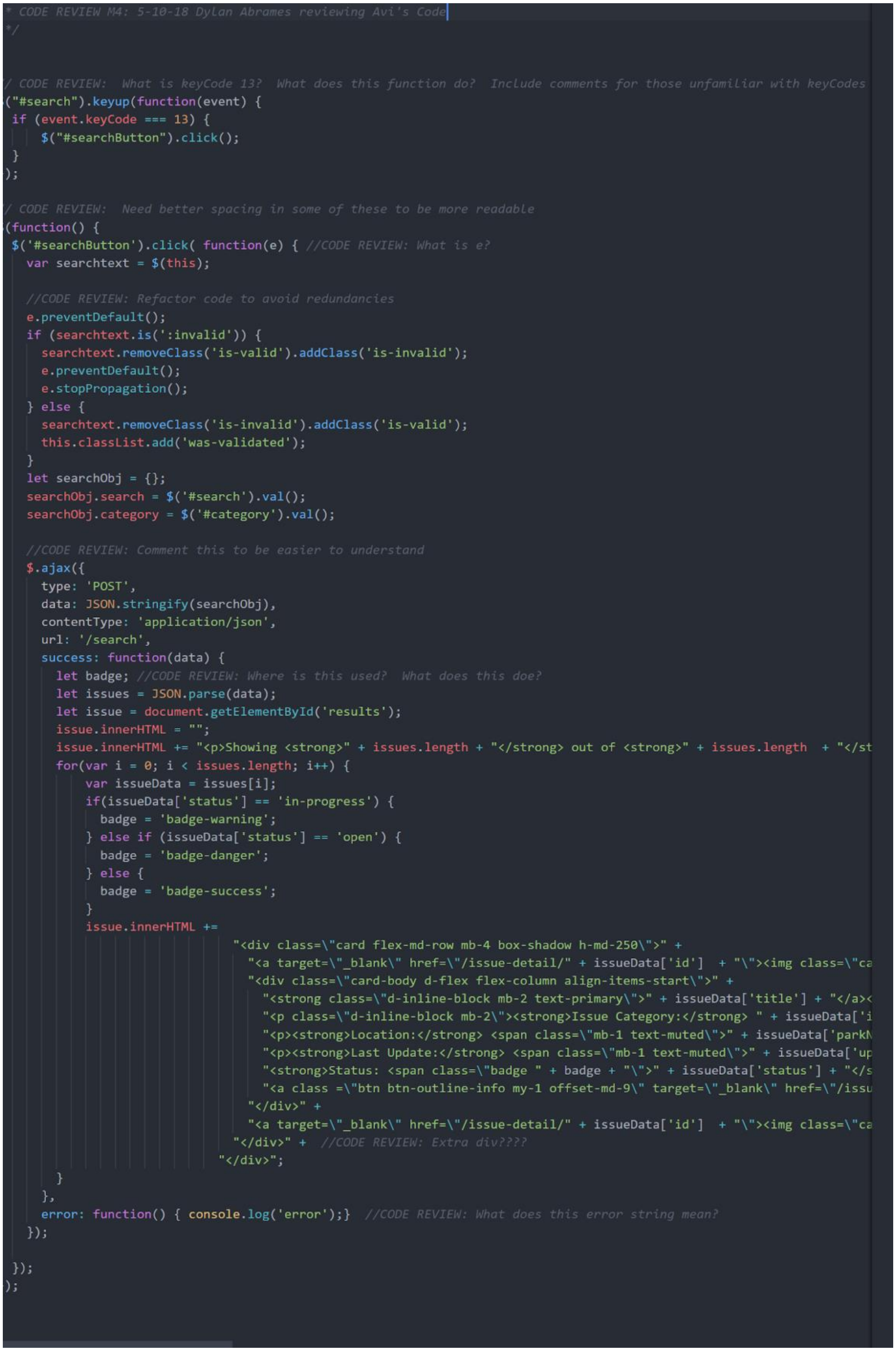
Hey Dylan,

Could you please review my code on search.js on the develop branch?

Thanks,

Avi

Reviewed Code



```
* CODE REVIEW M4: 5-10-18 Dylan Abrames reviewing Avi's Code
*/


/ CODE REVIEW:  What is keyCode 13?  What does this function do?  Include comments for those unfamiliar with keyCodes
("#search").keyup(function(event) {
 if (event.keyCode === 13) {
     $("#searchButton").click();
 }
);

/ CODE REVIEW:  Need better spacing in some of these to be more readable
(function() {
 $('#searchButton').click( function(e) { //CODE REVIEW: What is e?
   var searchtext = $(this);

   //CODE REVIEW: Refactor code to avoid redundancies
   e.preventDefault();
   if (searchtext.is(':invalid')) {
     searchtext.removeClass('is-valid').addClass('is-invalid');
     e.preventDefault();
     e.stopPropagation();
   } else {
     searchtext.removeClass('is-invalid').addClass('is-valid');
     this.classList.add('was-validated');
   }
   let searchObj = {};
   searchObj.search = $('#search').val();
   searchObj.category = $('#category').val();

   //CODE REVIEW: Comment this to be easier to understand
   $.ajax({
     type: 'POST',
     data: JSON.stringify(searchObj),
     contentType: 'application/json',
     url: '/search',
     success: function(data) {
       let badge; //CODE REVIEW: Where is this used?  What does this doe?
       let issues = JSON.parse(data);
       let issue = document.getElementById('results');
       issue.innerHTML = "";
       issue.innerHTML += "<p>Showing <strong>" + issues.length + "</strong> out of <strong>" + issues.length  + "</st
       for(var i = 0; i < issues.length; i++) {
           var issueData = issues[i];
           if(issueData['status'] == 'in-progress') {
             badge = 'badge-warning';
           } else if (issueData['status'] == 'open') {
             badge = 'badge-danger';
           } else {
             badge = 'badge-success';
           }
           issue.innerHTML +=
                               "<div class=\"card flex-md-row mb-4 box-shadow h-md-250\">" +
                                 "<a target=\"_blank\" href=\"/issue-detail/" + issueData['id']  + "\"><img class=\"ca
                                 "<div class=\"card-body d-flex flex-column align-items-start\">" +
                                   "<strong class=\"d-inline-block mb-2 text-primary\">" + issueData['title'] + "</a><
                                   "<p class=\"d-inline-block mb-2\"><strong>Issue Category:</strong> " + issueData['i
                                   "<p><strong>Location:</strong> <span class=\"mb-1 text-muted\">" + issueData['parkN
                                   "<p><strong>Last Update:</strong> <span class=\"mb-1 text-muted\">" + issueData['up
                                   "<strong>Status: <span class=\"badge " + badge + "\">" + issueData['status'] + "</s
                                   "<a class =\"btn btn-outline-info my-1 offset-md-9\" target=\"_blank\" href=\"/issu
                                 "</div>" +
                                 "<a target=\"_blank\" href=\"/issue-detail/" + issueData['id']  + "\"><img class=\"ca
                               "</div>" +  //CODE REVIEW: Extra div????
                             "</div>";
       }
     },
     error: function() { console.log('error');}  //CODE REVIEW: What does this error string mean?
   });

 });
);
```

Self-Check on Best Practices for Security

Assets Protected:

- User passwords.
- Reports and accounts.
- System stability and availability.

Data Security:

- Passwords are encrypted using *bcrypt* and only the hashes are stored in the database.

Input Data Validation:

- Regex validation is used to screen inputs
  - Search bar:
    - Search bar only accepts 40 or less alphanumeric characters.
  - Registration Page:
    - Name field only accept letters.
    - Emails field only accept valid email address formats.
    - Registration fields are limited to 30 characters or less.
  - Report Issue Page:
    - Report Title only accepts alphanumeric characters.
    - Park Name only accepts alphanumeric characters.
    - Report Title only accepts up to 40 characters.
    - Park Name only accepts up to 40 characters.
    - Issue Description only accepts up to 500 characters
- Limited input format
  - Report Issue Page:
    - Upload Image only allows JPG and BMP formatted files.

Self-Check: Adherence to Original Non-Functional Specs

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