SW Engineering CSC648/848 Section 2 Spring 2018

WWW Site for Reporting and Managing Environmental Issues "Park Report"

Team 13

Damico Shields (dshields@mail.sfsu.edu)

Leo Wang

Jaimes Subroto

Justice Chase

Kimyou By

Andrew Hutzel

Milestone 4

May 16, 2018

Version	Date	Description
1.0	5/17/2018	Initial draft

Product Summary

Park Report is a website for reporting and managing reports of environmental issues.

Functions:

1. All users of **Park Report** can easily look up their local parks by park name or postal code to find any reports of environmental issues submitted by their fellow citizens.
2. They can refine their search by limiting it to specific types of issues, such as oil spills, or medical waste.
3. Should they wish to read a report, they can click on it and learn when it was submitted, what type it is, what the details are, which park it is in, and they are provided with an image of the issue and an embedded Google Map centered on the park in question.
4. Should a user wish to post a report of their own, they will first be prompted to log in or register. Registration simply requires a username, a password and a working email.
5. Once logged in, a user can post a new report, choosing a park and an issue type from the lists, supplying some extra details about the issue, and uploading an image.
6. Registered city managers can look at a special list of all reports, sorting by park name, type, etc.
7. The city manager can also edit reports: they can correct inaccurate categorization, update the status, and delete reports with inappropriate content.
8. Website administrators will have access to a special administration page, where they can delete reports or add parks and issue types to the options available to the public.
9. These administrators will also be able to see which user submitted each report.
10. Finally, they can upgrade a user to be a city manager, giving them the privilege to access the aforementioned list of reports.

What makes **Park Report** stand out compared to existing services? Let's say you notice a pile of garbage dumped in the playground when you took your child to play. You could call the Parks and Recreation department, where you will be put on hold. Eventually, a receptionist will pick up, and after explaining yourself, you will be transferred to the proper office, where you might wait on hold again. After explaining yourself again, answering a list of questions, you will receive a report number. You are now in the dark: other than actually returning to the park to see if any cleanup has occurred, your only choice is to call the office again, and ask about the progress of your report number, suffering through the holds and questions again. Or, you could snap a picture with your phone, go home and upload it on our website, and check the progress from the comfort of your own computer.

Website URL: http://35.230.88.48/
Alternate link: https://csc648team13.com/

Usability Test Plan: Search

Test objectives: We wish to test the usability of our implemented search function. We want to make sure that it is intuitive enough that normal people can find reports in the parks they care about, as we feel this will be the primary use of the website.

Test plan:
Setup: The tester will be sat in front of a computer with Google Chrome set to the front page of the website.
Task: “Please find any reports at Heron’s Head Park, then find any reports of medical waste, then reports of medical waste at Heron’s Head Park (postal code 94124, in case they don’t know that).”
Intended user: A normal citizen familiar with the city and its parks. Should only have average computer knowledge for someone not in a technological field.
Completion criteria: They have found searched an empty search bar with the All option, they have searched an empty field with the medical waste option, and they have searched for Heron’s Head Park or its postal code with the medical waste option.
URL: https://csc648team13.com/

Questionnaire:

	1 Strongly Disagree	2 Disagree	3 Neutral	4 Agree	5 Strongly Agree
1. I quickly knew where to search for environmental issues.	○	○	○	○	○
2. I easily understood how to use the dropdown menu to refine my search.	○	○	○	○	○
3. I understood what the results I was shown meant.	○	○	○	○	○
4. There were clear instructions on how to fix any invalid searches.	○	○	○	○	○
5. Please leave any additional comments:					

QA Test Plan: Search Function

Test Objectives:
We wish to test the correctness of the implemented search function and its various error messages, to ensure that users are either correctly finding what they searched for or are informed of how their search should be improved.

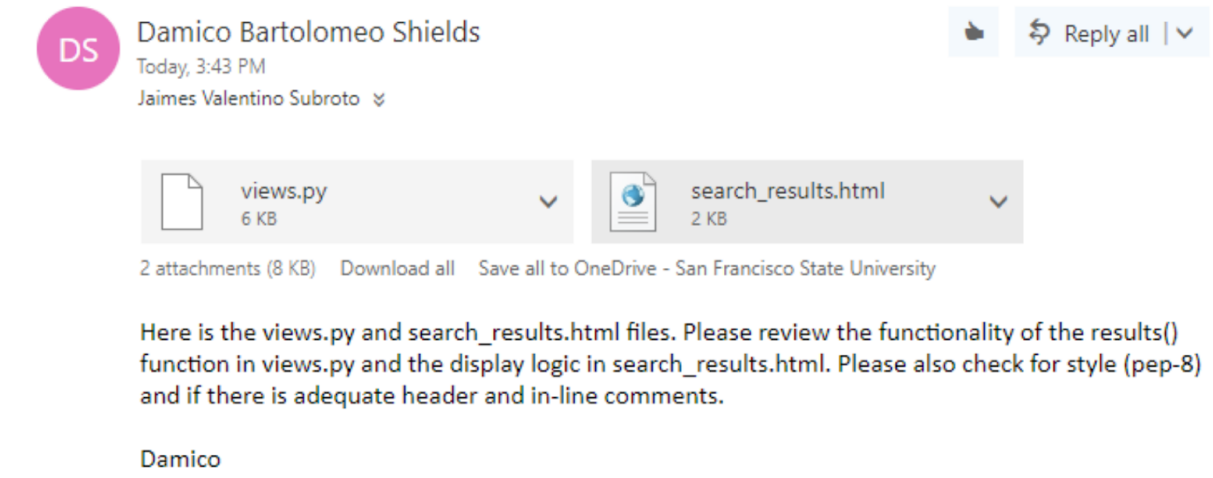
Hardware: One laptop computer, of average specs.
Software: Two browsers open to the website: Google Chrome, Mozilla Firefox.
Feature to be tested: The search bar and the results or errors that come from it.

Test cases:

Number	Title	Description	Input	Expected Output	Results
1	Chrome Park	Test % like for name	“All” “head”	Get 2 reports with Heron’s Head Park	Pass
2	Chrome Invalid	Catch invalid input	“All” “-1234”	**“**Please enter alphanumeric characters only. Search either by zip code, park name or city name.**”**	Pass
3	Chrome Similar	Test if there are zero results.	“Bathroom” “94110”	Get 4 reports with type bathroom. Message: “No park matches your search criteria, here are reports that share a category.”	Fail, no message
4	Firefox Park	Test % like for name	“All” “head”	Get 2 reports with Heron’s Head Park	Pass
5	Firefox Invalid	Catch invalid input	“All” “-1234”	**“**Please enter alphanumeric characters only. Search either by zip code, park name or city name.**”**	Pass
6	Firefox Similar	Test if there are zero results.	“Bathroom” “94110”	Get 4 reports with type bathroom. Message: “No park matches your search criteria, here are reports that share a category.”	Fail, no message

Code Review

We have chosen to follow the PEP 8 style.



DS

Damico Bartolomeo Shields

Today, 3:43 PM

Jaimes Valentino Subroto

Reply all

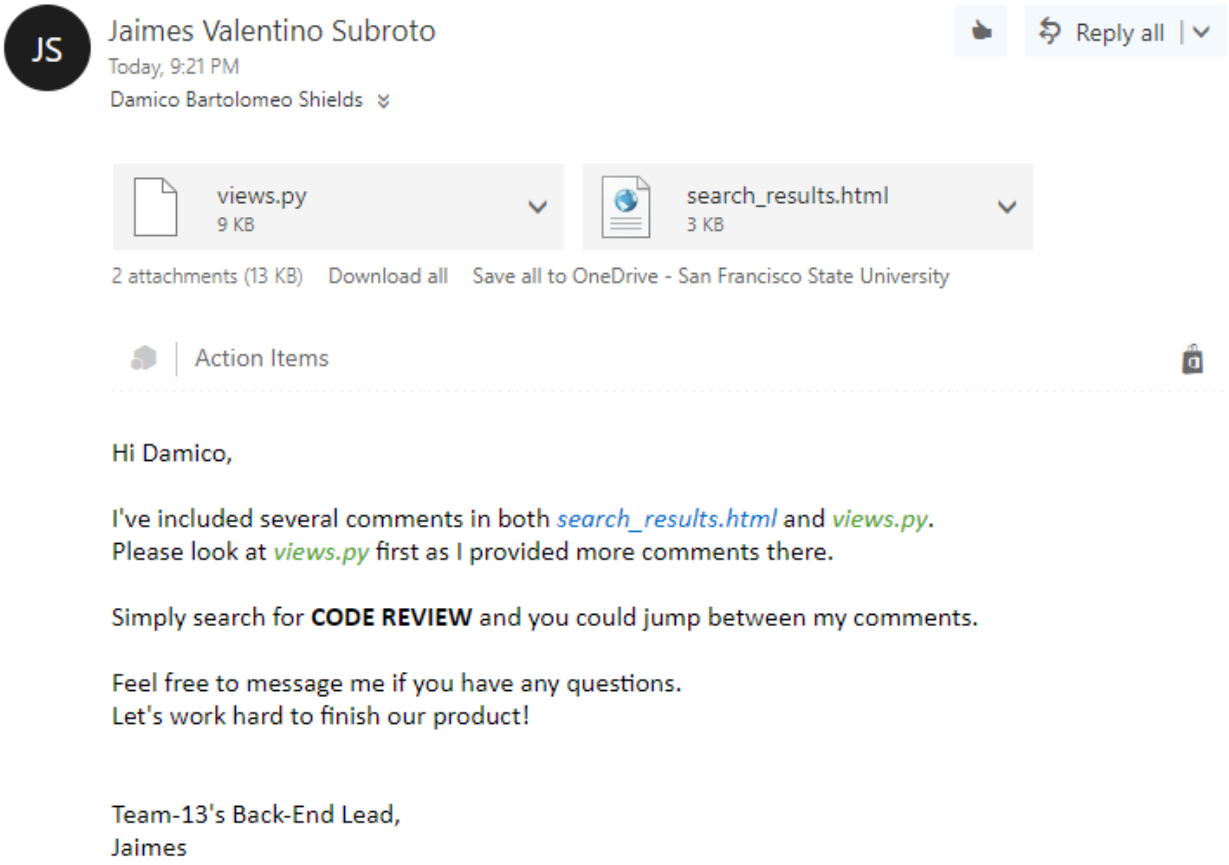
views.py
6 KB

search_results.html
2 KB

2 attachments (8 KB) Download all Save all to OneDrive - San Francisco State University

Here is the views.py and search_results.html files. Please review the functionality of the results() function in views.py and the display logic in search_results.html. Please also check for style (pep-8) and if there is adequate header and in-line comments.

Damico



JS

Jaimes Valentino Subroto

Today, 9:21 PM

Damico Bartolomeo Shields

Reply all

views.py
9 KB

search_results.html
3 KB

2 attachments (13 KB) Download all Save all to OneDrive - San Francisco State University

Action Items

Hi Damico,

I've included several comments in both *search_results.html* and *views.py*.
Please look at *views.py* first as I provided more comments there.

Simply search for **CODE REVIEW** and you could jump between my comments.

Feel free to message me if you have any questions.
Let's work hard to finish our product!

Team-13's Back-End Lead,
Jaimes

Code Review Examples

Not complete code, only sections with comments included here:

Views.py

```
##################( CODE REVIEW )#####################
## Reviewed by Jaimes Subroto (Team 13: Back-End Lead)
#####################################################
# The main search functionality works fine. However, I have added suggestions below to
# improve code readability as well as to fix some bugs.
# Find 'CODE REVIEW' to jump through my comments.
#####################################################

        ##################( CODE REVIEW )#####################
        # Change 'All' to 'All Categories'
        #####################################################
        # Category set to All
        if category == 'All':
            ##################( CODE REVIEW )#####################
            # Use '\' before the OR operator and move the
            # the | Report.obj... to the next line
            #####################################################
            reports = Report.objects.filter(park__name__icontains=query) |
Report.objects.filter(
                park__zip_code__iexact=query)
                ##################( CODE REVIEW )#####################
                # Should be: reports = Report.objects.all()
                # since the category is set to 'All Categories'
                #####################################################
                reports = Report.objects.filter(type__type__iexact=category)
            return render(request, 'search/search_results.html', {
                'reports': reports, 'query': query, 'categories': categories, 'cat':
category, 'is_reports': True})
        ##################( CODE REVIEW )#####################
        # Change comment to Search/Query specified
        #####################################################
        # Category specified
        else:
            ##################( CODE REVIEW )#####################
            # Same thing here, use '\' before the OR operator
            # and move the | Report.obj... to the next line
            #####################################################
            reports = Report.objects.filter(park__name__icontains=query,
                                            type__type__iexact=category) |
Report.objects.filter(
                park__zip_code__iexact=query, type__type__iexact=category)
'error': error})

    ##################( CODE REVIEW )#####################
    # Append to the comment '(when search is empty)'
    #####################################################
    # Category search
    else:
        ##################( CODE REVIEW )#####################
        # Change 'All' to 'All Categories'
        #####################################################
        if category == 'All':
            reports = Report.objects.all()
```

```
            ##################( CODE REVIEW )#####################
            # Set a msg variable here to send to the template
            # giving the error message that the query was
            # empty, and that all reports were returned.
            #
            # Also, move categories (line 133) -> line 134
            # for better code readability.
            #
            # See search_results.html line 40
            ########################################################
            return render(request, 'search/search_results.html',
                          {'reports': reports, 'query': False, 'categories':
categories.exclude(type__iexact=category),
                           'cat': category})
        else:
            ##################( CODE REVIEW )#####################
            # Same thing as my last suggestion.
            ########################################################
            reports = Report.objects.filter(type__type__iexact=category)
            return render(request, 'search/search_results.html',
                          {'reports': reports, 'query': False, 'categories':
categories.exclude(type__iexact=category),
                           'cat': category})
```

Search_results.html

```
<!--###################( CODE REVIEW )###################
## Reviewed by Jaimes Subroto (Team 13: Back-End Lead)
#########################################################
# Search results looks good to me. Just add the msg from views.py when someone does an
# "empty" search query. Also, there's a minor bug where the reports are smaller in the
# search results than in the homepage. Simply change the bootstrap classes.
# Find 'CODE REVIEW' to jump through my comments.
#########################################################-->
###( CODE REVIEW )#####################
        # Add the msg variable from views.py here
        ########################################################-->
        <p>No park matches your search criteria, here are reports that share a
category.</p><hr>
    {% endif %}
    <hr>
    <!-- display Park info if search are found -->
    {% for issue in reports %}
        <div class="row">
            <!--###################( CODE REVIEW )#####################
            # Change 'col-sm-3' to 'col-3' so that the sizing of the report will be
            # consistent with our homepage.
            ########################################################-->
            <div class="col-sm-3">

            <!--###################( CODE REVIEW )#####################
            # 'col-sm-4' to 'col-4'
            ########################################################-->
            <div class="col-sm-4">

            <!--###################( CODE REVIEW )#####################
            # 'col-sm-4' to 'col-4'
            ########################################################-->
            <div class="col-sm-4">
                <iframe width="310" height="220" frameborder="0" style="border:0"
                        src="{{ issue.park.maps_string }}" allowfullscreen>
```

Security

Assets we are protecting:

1. User emails.
2. User passwords.

Passwords are encrypted in the database by the Django Authentication API.

For post report, login and registration forms, input is validated using Django's form validation API, namely the is_valid() method.

The search results use Django querysets, which are similarly protected against SQL attacks. Our custom search validation, namely of negative numbers, numbers that are too long, strings that are too long and strings containing non-alphanumeric characters is implemented in the results() function of the views.py file.

Adherence to Original Non-Functional Specs

1. Park Report shall be developed, tested and deployed using tools and servers approved by Class CTO. **DONE**
2. Park Report shall be optimized for standard desktop/laptop browser environments, e.g. shall render correctly on the two latest version of Chrome, Mozilla Firefox, Safari and Internet Explorer. **DONE**
3. Park Report shall have responsive UI code so that it can be rendered on mobile device browsers. **ON TRACK**
4. Data shall be stored in a MySQL database running on the Google Cloud Compute Engine. **DONE**
5. Application shall be media rich (at minimum contain images and Google Maps integrated) **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 shall be appropriately communicated to users. **ON TRACK**
8. The language on the website shall be English. **DONE**
9. Google analytics shall be used to track website traffic. **DONE**
10. No email clients shall be allowed. **DONE**
11. Pay functionality shall not be implemented nor simulated. **DONE**
12. Best practices for site security shall be applied. **DONE**
13. Modern software engineering process and practices shall be applied as specified in the class, including collaborative and continuous software development. **DONE**
14. The website shall display the following text on all pages “SFSU Software Engineering Project, Spring 2018. For Demonstration Only.” at the top of the WWW page. **DONE**
15. Application server shall be maintained during the semester. **DONE**
16. Application server costs on the Google Server shall not incur charges. **DONE**
17. The website shall be curated every day for any malicious content that may have been uploaded by users. **DONE**