TrashPosters

SW Engineering CSC648/848 Section 01 Spring 2018

Team 05 – Super Heroes In Training

Team is local

James Quintero - jamesaquint@gmail.com

Stanley Liu

Danielle Nunez

Tumar Temirova

Alex Hernandez

Jianhao Zhong

Milestone 4

16 May 2018

Revision Number	Name	Date
1.0	Initial Beta Launch	5/16/18

1. Product Summary

TrashPosters, where you can view and report environmental issues. http://trashposters.com

Priority 1 functions

Unregistered users:

- 1.01 Users shall be able to register
- 1.08 Users shall be able to browse posts
- 1.10 Users shall be able to filter search by issue type - tag
- 1.13 Users shall be able to filter search by keyword in post - limited to area/timeframe

Registered users:

- Registered users shall have functionality as unregistered users
- 2.02 Users shall be able to post about environmental issues
- 2.03 Users shall be able to include images and location in posts
- 2.07 Users shall have a profile page

City Officials:

- City Officials shall have functionality as registered users
- 3.05 Users shall be able to post official statements that stand out from regular posts
- 3.06 Users shall be given a city official account by a site admin

In this day and age, environmental issues should be prioritized as we progress as a society. Our web application offers a solution to getting regular citizens and elected officials involved in working together to troubleshoot the environment in a familiar way. Our platform, TrashPosters, allows people to communicate issues in a way people are already familiar with about environment hazards with each other and also to get in contact with city officials on ways to resolve complaints that would otherwise pile high on an unpaid intern's inbox in city hall. Our product allows city officials to have their own unique accounts in order to post official statements about environmental issues in their jurisdiction. Citizens will be informed and kept in the loop with what's going on in their area.

City Officials being on the site will attract users through advertising done by the City Officials. The City Officials could advertise the site as a place to receive up-to-date information on the current environmental status of the city. This will keep unregistered and registered users in the loop with their city. Users will also be attracted to our platform because the city officials will legitimize the service. TrashPosters won't just be a site where people complain, but a site where citizens and the city work together to clean up and help the environment.

We aim to build a visually appealing web application that will incentivize its users to visit on a daily basis by gamifying being active in the community by mimicking a social network scheme, something most internet users today are already familiar with. Instead of giving the users a chore with the feel of doing government paperwork, users shall be able to post quickly and easily to provide the government information about the environment and get in contact with city officials. Furthermore, users shall be able to search for results relevant to their communities and shall be given an opportunity to comment on local issues or subscribe to environmental news local to their area. Lastly, users shall be given an opportunity to personalize their own profile and get to know other community members and leaders in their neighborhood that they would otherwise never interact with.

2. Usability Test Plan

Search will be tested for usability. The goals of usability testing is to determine whether the user can use the search function without issue, and to determine user satisfaction in using the search function.

- **Problem statement**: Specific questions you want resolved
    - Want to see that user can successfully and easily filter results
    - Possible errors in completing tasks:
    - Navigation issues: where the users can't locate proper filters or results.
    - Excessive keystrokes: where the users have to click on too many buttons or dropdowns in order to complete a task
    - Improper values: where the users input improper values for search
- **Test plan and objectives**: tasks the user will do
    - Task 1:
        - Task: View the most recent posts.
        - Machine state: Home page.
        - Success: Most recent posts are displayed in list format.
        - Benchmark: Completed in 10 seconds.
    - Task 2:
        - Task: Search for posts containing "jellyfish" in the post title.
        - Machine state: Home page.
        - Success: posts in list format with every title containing "jellyfish".
        - Benchmark: Completed in 15 seconds.
    - Task 3:
        - Task: Search for posts containing "trash" in the post description.
        - Machine state: Home page.
        - Success: posts in list format with every description containing "trash".
        - Benchmark: Completed in 15 seconds.
    - Task 4:
        - Task: Search for posts from "San Francisco".
        - Machine state: Home page.
        - Success: posts in list format with every location from "San Francisco".
        - Benchmark: Completed in 15 seconds.
    - Task 5:
        - Task: Search for posts from user "daniellenunez"
        - Machine state: Home page
        - Success: Posts in list format with all from user "daniellenunez"
- **User Profile**: Who will be the users
    - Students at SFSU, 18-25 year old regular internet user
    - 35-45 year old worker, semi-regular internet user
    - 65+ year old retiree

- **Method and test design**: how will you observe it, how will you collect the data
  - Observe by looking over shoulder and video/audio recording
  - Collect data through video/audio recording, and handwritten notes.
- **Test environment and equipment:**
  - Environment will be a classroom at a standard table
  - Equipment will be a provided laptop running Windows with latest version of Google Chrome.
- **Test monitor role**:
  - To guide the users through the tasks, logging when they start each task, any comments they have for each task, and when they complete each task.
- **Evaluation measures and data to be collected**: how will you collect the feedback and how will you evaluate it:
  - Test monitors will log user comments, and a review of the video/audio will provide feedback on the user's mouse movements for each task. The mouse movements and clicks will tell us where the users expect features to be when they're not.
- **Legal issues:**
  - Each test user will be a volunteer, and will not have to sign an NDA.
- **Report:** what will final report contain
  - Final report will contain ISO/IEC 9126-4 usability metrics. Effectiveness, Efficiency, and Satisfaction for the search function.
- **URL to be tested:**
  - http://trashposters.com

Questionnaire: 3 Lickert scale questions, in a form easy to be used by reviewer (check class slides) – 3/4 page

- The search feature was easy to use
  - Strongly agree
  - Agree
  - Neutral
  - Disagree
  - Strongly Disagree
  - Comments on ease of use?
- The search GUI was pleasing to look at
  - Strongly agree
  - Agree
  - Neutral
  - Disagree
  - Strongly Disagree
  - Comments on look of search GUI?
- The search results were informative
  - Strongly agree
  - Agree

- Neutral
- Disagree
- Strongly Disagree
- Comments on list of posts?

3. QA Test Plan

Test Objectives:
Our objective is to remove all bugs from the search function of Trashposters. This will require we test multiple types of input: alphabetical, numerical, alphanumerical, special character, no input, and apostrophe. We will compare the number of posts returned for each test input, and compare it to what should be returned.

Hardware & Software setup:

Asus Laptop running Windows with Chrome or Firefox web browser.
Go to TrashPosters homepage.
Enter desired inputs into the search bar at the top of the page.

Feature to be Tested: Search function

Test #	Test title	Description	Browser	Input	Actual output	Correct output	Test results
1	alphabetical title search	Test % LIKE in search for title field	Chrome	jellyfish	4 results, all have "jellyfish" in post title	4 results, all have "jellyfish" in post title	PASS
2	Empty post title search	Test % LIKE in search for title field	Chrome		Crash	24 results, all posts	FAIL
3	Special character post title search	Test % LIKE in search for title field	Chrome	[^$&*()@{}|	Crash	No posts returned, so here are most recent posts	FAIL
4	Integer post title search	Test % LIKE in search for title field	Chrome	123456	No posts returned, so here are most recent posts	No posts returned, so here are most recent posts	PASS
5	Apostrophe post title search	Test % LIKE in search for title	Chrome	it's	Crash	2 results, all have "it's" in	FAIL

		field				post title	
7	alphabetical description search	Test % LIKE in search for description field	Chrome	gross	3 results, all have "gross" in post description	3 results, all have "gross" in post description	PASS
8	Empty post description search	Test % LIKE in search for description field	Chrome		Crash	24 results, all posts	FAIL
9	Special character post description search	Test % LIKE in search for description field	Chrome	[^$&*()@{}|	Crash	No posts returned, so here are most recent posts	FAIL
10	Integer post description search	Test % LIKE in search for description field	Chrome	123456	No posts returned, so here are most recent posts	No posts returned, so here are most recent posts	PASS
11	Apostrophe post description search	Test % LIKE in search for description field	Chrome	it's	Crash	2 results, all have "it's" in post title	FAIL
13	alphabetical title search	Test % LIKE in search for title field	Firefox	jellyfish	4 results, all have "jellyfish" in post title	4 results, all have "jellyfish" in post title	PASS
14	Empty post title search	Test % LIKE in search for title field	Firefox		Crash	24 results, all posts	FAIL
15	Special character post title search	Test % LIKE in search for title field	Firefox	[^$&*()@{}|	Crash	No posts returned, so here are most recent	FAIL

						posts	
16	Integer post title search	Test % LIKE in search for title field	Firefox	123456	No posts returned, so here are most recent posts	No posts returned, so here are most recent posts	PASS
17	Apostrophe post title search	Test % LIKE in search for title field	Firefox	we're	Crash	2 results, all have "it's" in post title	FAIL
19	alphabetical title search	Test % LIKE in search for description field	Firefox	jellyfish	3 results, all have "gross" in post description	3 results, all have "gross" in post description	PASS
20	Empty post title search	Test % LIKE in search for description field	Firefox		Crash	24 results, all posts	FAIL
21	Special character post title search	Test % LIKE in search for description field	Firefox	[^$&*()@{}|	Crash	No posts returned, so here are most recent posts	FAIL
22	Integer post title search	Test % LIKE in search for description field	Firefox	123456	No posts returned, so here are most recent posts	No posts returned, so here are most recent posts	PASS
23	Apostrophe post title search	Test % LIKE in search for description field	Firefox	we're	Crash	2 results, all have "it's" in post title	FAIL

4. Code Review

PEP 8 guidlines:

- Use 4 spaces per indentation level
- Limit lines to 79 characters
- Consistent line breaks for long if statements
- Comments should be complete sentences, first word capitlized, etc.
- Write docstring for all public modules, functions, classes, and methods.
- …

Rest can be found at: https://www.python.org/dev/peps/pep-0008/

Search function code:

```
def search_empty(request):
    """
    Handles an empty search bar.
    @:param     An http request.
    @:return    Renders a page with all posts listed.
    """
    all_posts = Posts.objects.all()
    context = {'posts': all_posts,
               'extra_posts': all_posts,
               'select': "",
               'keyword': ""}
    return render(request, 'new_regular/search.html', context)

def search_by(request, select, query):
    """
    Searches difference aspects of a post depending on what the user selects
    @:param     An http request with a title keyword.
    @:return    Renders a page with all matching posts listed.
    """

    context = {'posts': None,
               'extra_posts': None,
               'select': select,
               'keyword': query}

    if(select=="title"):
        context['posts'] =  Posts.objects.filter(title__icontains=query)
    elif (select=="description"):
        context['posts'] = Posts.objects.filter(description__icontains=query)
    elif(select=="user"):
        context['posts'] = Posts.objects.filter(user_id__username__exact=query)
    elif(select=="hazard_type"):
        context['posts'] = Posts.objects.filter(hazard_type__hazard_name__exact=query)
    elif (select=="location"):
        context['posts'] = Posts.objects.filter(location__icontains=query)

    #gets all posts if no results
    if not context['posts']:
        context['extra_posts'] = Posts.objects.all()

    return render(request, 'new_regular/search.html', context)
```

Emails:

---

jamesaquint@gmail.com:
Hi Danielle, can you please review the code for the search function? Methods search_empty and search_by.

Thank you

Attached file: Views.py

---

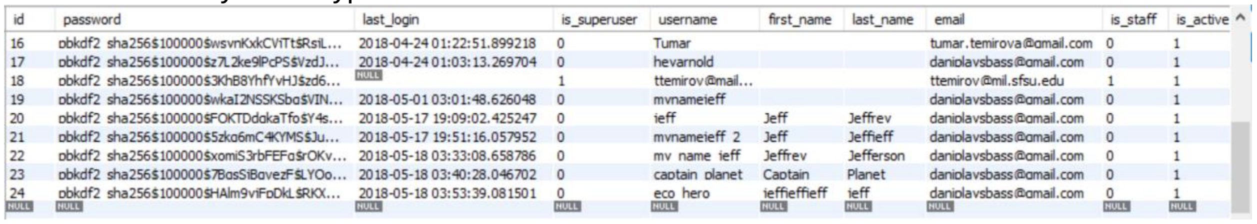
dnunez@mail.sfsu.edu
Hi James,

The first three quarters of the code you sent are all follow PEP8 style documentation. That's great, I'm glad our team could agree to a code style and stick together. However, the last few functions starting from line 303 are missing comments. It's really similar to javadoc style commenting where you have param, return, etc. underneath the function definition. We should review it as a team next time.
We could probably clean up the imports at the top of the file later one once we move our code towards the final milestone. I noticed that there are a ton of unused imports. However, they're all on top of the file which is up to PEP8's standards.
Another note, I'm glad to see that the code's white space consists of entirely spaces rather than a mixture of tabs and spaces. It's good to see that the team has agreed on the preferred method of spaces.

---

5. Self-check on best practices for security

- Major assets we are protecting:
    - Passwords
    - Emails
    - First and Last name
    - Confirm that you encrypt PW in the DB



id	password	last_login	is_superuser	username	first_name	last_name	email	is_staff	is_active
16	pbkdf2_sha256$100000$wsvnKxkCViTt$RsiL...	2018-04-24 01:22:51.899218	0	Tumar			tumar.temirova@gmail.com	0	1
17	pbkdf2_sha256$100000$z7L2ke9lPcPS$VzdJ...	2018-04-24 01:03:13.269704	0	hevarnold			daniplaysbass@gmail.com	0	1
18	pbkdf2_sha256$100000$3KhB8YhfYvHJ$zd6...	NULL	1	ttemirov@mail...			ttemirov@mil.sfsu.edu	1	1
19	pbkdf2_sha256$100000$wkaI2NSSKSbq$VIN...	2018-05-01 03:01:48.626048	0	mynamejeff			daniplaysbass@gmail.com	0	1
20	pbkdf2_sha256$100000$FOKTDdqkaTfo$Y4s...	2018-05-17 19:09:02.425247	0	jeff	Jeff	Jeffrey	daniplaysbass@gmail.com	0	1
21	pbkdf2_sha256$100000$5zkq6mC4KYMS$Ju...	2018-05-17 19:51:16.057952	0	mynamejeff_2	Jeff	Jeffjeff	daniplaysbass@gmail.com	0	1
22	pbkdf2_sha256$100000$xomiS3rbFEFq$rOKv...	2018-05-18 03:33:08.658786	0	my_name_jeff	Jeffrey	Jefferson	daniplaysbass@gmail.com	0	1
23	pbkdf2_sha256$100000$7BqsSjBgvezF$LYOo...	2018-05-18 03:40:28.046702	0	captain_planet	Captain	Planet	daniplaysbass@gmail.com	0	1
24	pbkdf2_sha256$100000$HAlm9viFpDkL$RKX...	2018-05-18 03:53:39.081501	0	eco_hero	jeffjeffjeff	jeff	daniplaysbass@gmail.com	0	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Confirm Input data validation (list what is being validated and what code you used) – we request you validate search bar input;
    - Currently validated forms:
    - Registration
    - Login
    - Post
    - Comment
- Future validated forms:
    - Search

Example Registration form validation:

```
def clean(self):
    """
    Extends the clean() function so that the form throws ValidationErrors
    if the passwords and emails don't match.
    :return: cleaned_data if everything looks good.
    :raisesL Validation errors if the fields don't match.
    """
    email = self.cleaned_data.get('email', None)
    re_email = self.cleaned_data.get('re_email', None)
    password = self.cleaned_data.get('password', None)
    re_password = self.cleaned_data.get('re_password', None)
    if email and re_email and (email == re_email):
        if password and re_password and (password == re_password):
            return self.cleaned_data
        raise forms.ValidationError("Your passwords don't match")
    else:
        raise forms.ValidationError("Your emails don't match")
```

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. DONE
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. DONE
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. DONE
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). **ON TRACK**
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**