SW Engineering CSC 648/848
Section 01 Team 07
Eco Hazards
Milestone 4

Spring 2018

Sean Sutherland(ssutherl@mail.sfsu.edu), Lance Larsen, Corey Humeston,
Mark Soriano, Girish Tiwale, Ali Alavi, Amelie Cameron

Revision	Date
1.0	05/17/2018
1.1	

1. Product Summary

Environmental issues and hazard are impacting the world around us. Did you ever think you were able to make a difference, whether it be in your community or the environment? A group of students at San Francisco State University are trying to accomplish this task, to help bring people together to clean up the environment in the easiest/most efficient way possible. This design with change the way environmental issues are solved today. The use of mobile devices in this day and age will only help with the final product to change the way environmental issues are dealt with.

In this report we present Eco Hazards, an application created by a student start-up company. Eco Hazards has the potential to help clean up the environment in an easy and task-free way. Our application started as a way for people in the Bay Area to walk around in certain locations, parks, bridges, streets, etc.., and monitor the environmental hazards around them. When a hazard was discovered, the user would simply pull up the application, add their location, a quick comment, and even a picture of the hazard itself.

After the hazard is put into the system, a moderator will monitor the status of the hazard and change it from Resolved to Unresolved if the hazard was to be cleaned up. The application will work with Bay Area cleanup services and anyone who is willing to help the environment on their own time.

The product we offer in our application is quick, reliable, and easy to use. Our frontend team worked extremely hard on making the UI as user-friendly as possible with simple 'one-button' clicks to get the user where they want to go. The backend team even put their work in to make sure the best of algorithms were used for search functions, post functionality, and google map API's.

Eco Hazards provide the following features :

- Users shall be able to post information about environmental hazards in their area. .
- User shall be able to add all relevant information when uploading a report, including images, excluding status and who it is assigned to
- Authentication to post hazard reports shall be provided by either Login or by user providing identifying information
- Users shall be able to view posted (active) hazard reports and associated information.
- Users shall be able to search for hazard reports by zip code or location

Eco Hazards is now live and the link to the web application in the URL
http://csc648team07.herokuapp.com/

2. Usability Test Plan

Test Objectives:
The usability test plan focuses on testing features of our application for its ease of use. We selected our search function to analyze and report in detail from a testing standpoint.

Test Plan:
System Setup:
User will need access to the internet and a web browser. In the web browser running a recent version of either Chrome, or Firefox in which the user would need to type the URL provided which would navigate them to the homepage of the website.

User will need access to the internet and any updated web browser (Chrome, Firefox, Safari). With the chosen web browser up and running, the user will type the URL provided. The URL will navigate the user to our homepage website.

Starting Point:
Go to: http://csc648team07.herokuapp.com

Tasks:
1. From the homepage search bar, search for a zip code (94132).
2. From the homepage search bar, search for an oil spill.
3. From the homepage search bar, search for a fire.
4. From the homepage search bar, search for San Francisco.

Intended Users:
The intended users for Eco Hazards are the local residents who live in the Bay Area and want to help clean up the environment.

Completion Criteria:
User perform a search function on the homepage using a valid keyword. The valid keyword searches the database to find an item to report in each step.

3. Questionnaire

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

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Eco Hazards user interface was easy to understand	o	o	o	o	o

Additional Comments

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The search results of hazard reports were simple and easy to view	o	o	o	o	o

Additional Comments

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
It was easy to search for hazard reports	O	O	O	O	O

Additional Comments

3. QA Test Plan

Test Objectives:
Ensure the usability and functionality of searching for hazards, logging in and out, uploading a post, finding last report, and opening media page.

Hardware and Software Setup:
Application supports Chrome, FireFox, and Safari. Internet connection is required.

Hardware:
Processor: 2.9 Ghz Intel Core i5 RAM: 8GB
OS: macOS High Sierra

Testing:
The application testing targets the search functionality. Its features ensure the stability of a user friendly UI.

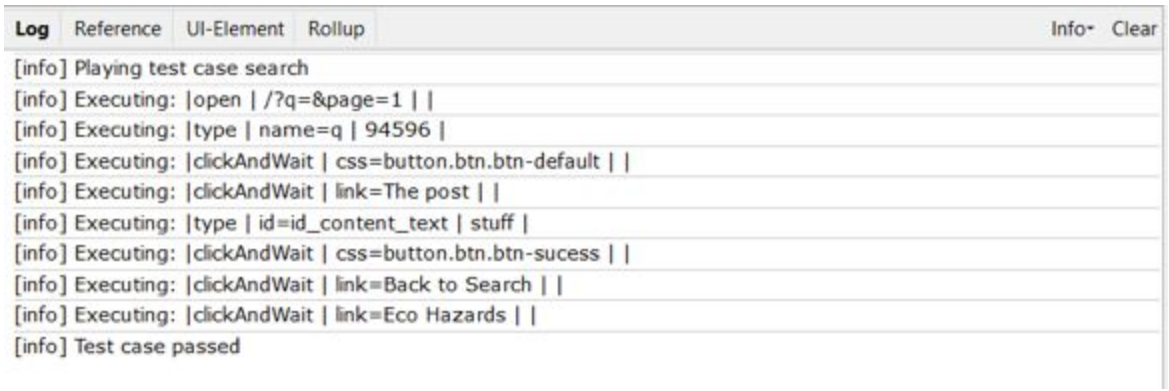
Test Number	Description	Test Input	Expected Output	PASS/FAIL Google Chrome	PASS/FAIL Firefox
1.	Open the webpage: http://csc648team07.herokuapp.com Search for zip code: 94132	**Category:** Zipcode **Search:** "94132 "	Hazards that fall in the "zipcode" category: **8 Results**	Pass	Pass
2.	Search for an oil spill.	**Category:** "Oil spill" **Search:** "oil spill "	Hazards that fall in the "oil spill" category: **5 Results**	Pass	Pass

3.	From the homepage search bar, search for a fire.	**Category:** Fire **Search:**"fire"	Hazards that fall in the "fire" category: **10 Results**	Pass	Pass

4.	From the homepage search bar, search for San Francisco.	**Category:** location **Search:** "San Francisco "	Hazards that fall in the "San Francisco" category: **48 Results**	Pass	Pass

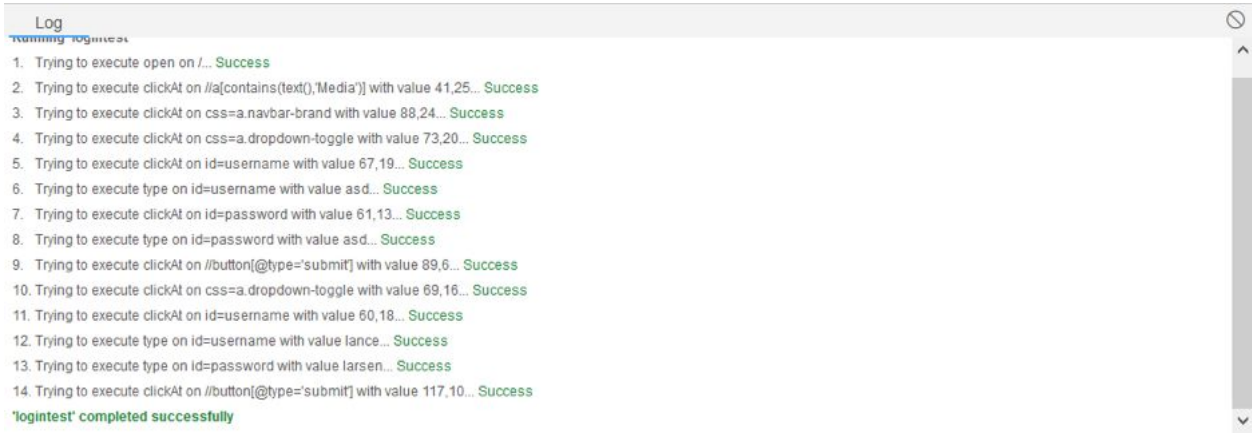
1) Upload a Post and return to Homepage.

Search for Zip “94596”. Select first result then leave a comment. Return to homepage.



Find your last report under My Reports. Return to homepage.

Open media then return to homepage. Fail to login, then log in.



4) Code Review:

As responsible developers, we adhered to the coding style of the set framework Django, it follows **MVC** pattern very closely but it uses slightly different terminology. **Django is** essentially an MTV (Model-Template-View) framework. **Django** uses the term Templates for Views and Views for Controller. In other words, in **Django** views **are** called templates and controllers **are** called views.

Model / Database Code

```
# Create your models here.
# These are items for our database; Hazard Reports


class Category(models.Model):
    id = models.AutoField(primary_key=True)
    description = models.TextField()
    title = models.CharField(max_length=45)

# hazard post Content and text lengths limit
class HazardReport(models.Model):
    pub_date = models.DateTimeField(auto_now_add = True)
    title_text = models.CharField(max_length=26)
    content_text = models.TextField(max_length=240)
    zipcode = models.CharField(max_length=5)
    location = models.CharField(max_length=50)
    image = models.FileField(null=True, blank=True)
    user = models.ForeignKey(User, related_name="hazardposts",
                             on_delete=models.CASCADE)
    category = models.ForeignKey(Category,
                             on_delete=models.CASCADE)

    def __str__(self):
        return self.title_text

# Date Field
# Hazard post comment Content and text lengths
class HazardReportComment(models.Model):
    pub_date = models.DateField(auto_now_add = True)
    content_text = models.TextField()
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    hazardreport = models.ForeignKey(HazardReport, related_name='comments',
                                 on_delete=models.CASCADE)
```

View / Template Code

```
{% if hazardreport.image %}
    <img src="{{ hazardreport.image.url }}" class="img-responsive" />
    {% endif %}
  <h2>{{ hazardreport.title_text }}</h2>
  <h3>{{ hazardreport.zipcode }}</h3>
  <h3>Hazard Category: {{ hazardreport.category.title }}</h3>
  <p>Location: <span class='location'>{{ hazardreport.location }}</span></p>
  <p>{{ hazardreport.pub_date }}</p>
  <div class='map' id="map-canvas" style="width: 320px; height: 320px"></div>
</div>
  <p>Author: <a href="#">{{ hazardreport.user.username }}</a></p>
  <span>{{ hazardreport.content_text }}</span>
  <br /><br /><br />
  <b>Comments:</b>
  {% if comments %}
    {% for item in comments %}
      <p>{{ item.pub_date }}<a href="{% url 'ecohazards:userposts' item.user.username %}"> {{ item.user.username }} </a>
      <p> {{ item.content_text }} </p>
      <br />
    {% endfor %}
  {% else %}
    <p>There are no comments yet</p>
  {% endif %}
  <!-- Show view for authenthicated users -->
  {% if request.user.is_authenticated %}
    <form class="form-horizontal" action="" method="POST" encrypt="multipart/from-data">
        {% csrf_token %}
        {% include "hazard/form-template.html" %}
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <button type="submit" class="btn btn-sucess" />Add comment</button>
            </div>
        </div>
    </form>
  {% endif %}

{% endblock content %}
```

Controller / Functions Code

```
def search_process(request):
    search = request.GET['q']
    search_list = HazardReport.objects. \
        filter(Q(title_text__icontains=search)
        | Q(content_text__icontains=search)
        | Q(pub_date__contains=search)
        | Q(zipcode__contains=search))
    # number of search results 6
    current_post_list, num_pages = get_paginator(request, search_list, 6)
    context = {
            'list': current_post_list,
            'num_pages': num_pages,
            'search': search
    }
    return render(request, 'hazard/search_results.html', context)
```

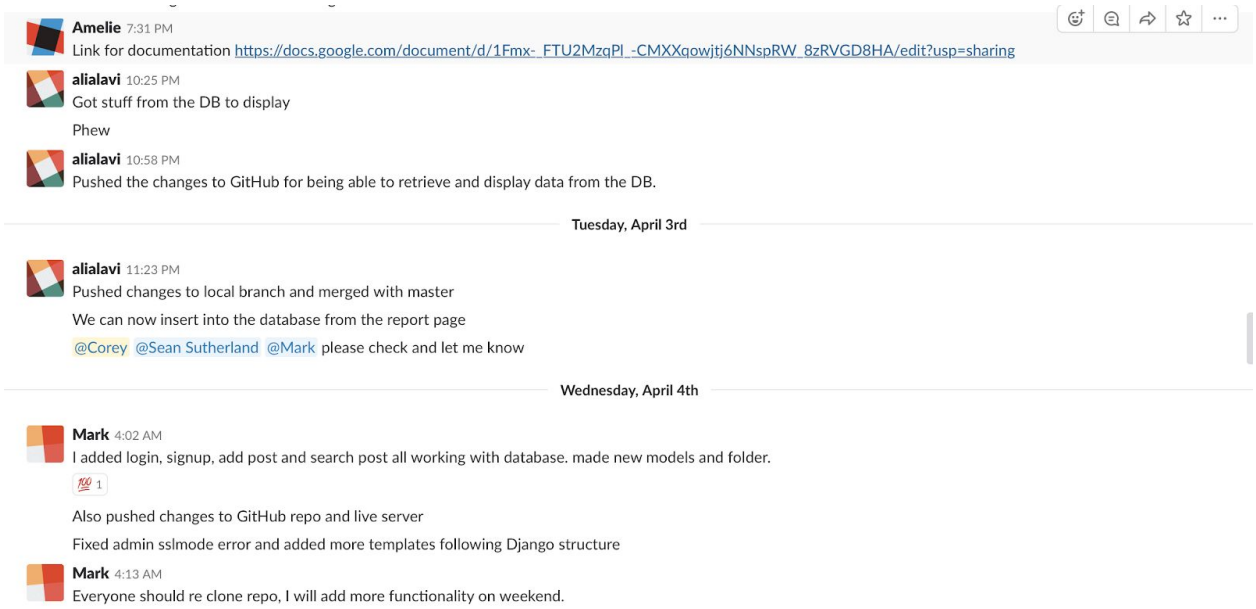
```
# USER HAS LOGIN WITH USER AUTHENTHICATION REDIRECT TO INDEX

def login_process(request):
    if request.method == 'POST':
        form = AuthenticationForm(request.POST)
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)

        if user is not None:
            if user.is_active:
                login(request, user)
                return redirect('ecohazards:index')
        else:
            messages.error(request,'Login Failed : User name or password incorrect')
            return redirect('ecohazards:index')
    else:
        form = AuthenticationForm()
    return render(request, 'ecohazards:index', {'form': form})
```

```python
# ---- CREATE NEW HAZARD REPORT VIEW ----
class HazardReportCreate(CreateView):
    """Creates new post """
    form_class = HazardReportForm
    template_name = "hazard/hazardreport_form.html"
    categories = Category.objects.all()

    print(categories)

    def get(self, request):
        form = self.form_class(None)
        return render(request, self.template_name, {'form': form, 'categories': self.categories})

    def post(self, request):
        form = self.form_class(request.POST or None, request.FILES or None)

        if form.is_valid():
            new_post = form.save(commit=False)
            new_post.user_id = request.user.id
            new_post.save()
        return redirect('ecohazards:post', hazardreport_id=new_post.id)
```

Peer Review

We used git, github, asana, and slack as a platform for peer review. We also resolved our conflicts with building the application, merging, and Milestones through face-to-face contact and slack communication. Github was used to help us track issues with our code (bugs). When a persistent bug crept along, github and asana were extremely helpful in assigning people to figure it out. Milestones were all collaborated on together with google docs, where everyone had a certain space to fill out.

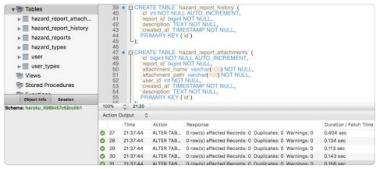
Screenshots:



alialavi 3:46 PM
Got it

alialavi 9:45 PM
The schema is there on the live db

alialavi 9:45 PM
uploaded this image: live schema.png



Corey 10:54 PM
Great job

Thursday, March 15th

Corey 10:46 AM
Hey, @Sean Sutherland. My .idea folder and other random stuff are pushing to Github.
I thought we adde .gitignore files? Not too sure why it's happening.

Sunday, March 18th

Corey 11:00 PM
@everyone Documentation for Milestone 2 is complete and sent in an email. Check our prototype online! Have a good break y'all.
😍 1 💯 1

Friday, March 23rd

alialavi 1:19 PM
Hey @Mark and @Corey please review my pull request for the autocomplete address functionality. I was able to test it by making a new post and it worked fine

Corey 1:48 PM
How would we incorporate it?
Ohhhh
I see.

alialavi 1:50 PM
It's done, we can merge it and test it on live

Amelie 8:12 PM
I made two columns for the post list, I'll make more example posts and print screenshots tomorrow for our review. Does he want a second search bar on the results page too?

Monday, April 23rd

Amelie 4:26 PM
So far I made screenshots of the home page, search results, map view, signup/login, about, media, and account posts. Are there any other ones missing?

Mark 4:27 PM
looks god
good*

Sean Sutherland 4:27 PM
Details page for an individual post, maybe, otherwise that covers everything, thank you

Amelie 4:28 PM
Thanks forgot that one

Tuesday, April 24th

5) Self-check on best practices for security

Major assets we aim to protect include:

User's confidential information such as his/her email address and password.

We protect against invalid hazard reports being posted by requiring the user to register and login to post a new hazard report

Confirm that you encrypt PW in the DB

Passwords are encrypted before being stored in the database. It uses django encrypting algorithm to securely store the passwords safely when it send data of registered users to the database.

id	password	last_login	is_superuser	username
1	pbkdf2_sha256$100000$a5Zbavzv7iIe$lO/nMYi...	2018-05-10 19:58:41.333122	1	hCorbear
11	pbkdf2_sha256$100000$9baS4iXXwxcZ$66bdo...	NULL	0	untletowev
21	pbkdf2_sha256$100000$RlOBwMeBwLOR$PMn...	2018-04-03 06:01:58.234235	1	mark
31	pbkdf2_sha256$100000$pO2dzOipwd77$twlbX...	2018-04-04 09:15:02.395132	0	User
41	pbkdf2_sha256$100000$P8CvBB0LPDeh$U/K1...	2018-04-04 10:19:38.774590	0	onetrwo
51	pbkdf2_sha256$100000$VA8NBGnhOkbU$65Xa...	2018-05-16 00:11:40.931697	1	mrk
61	pbkdf2_sha256$100000$VmU8dmM6xWHh$0N...	2018-04-05 05:46:27.392630	0	srsutherland
71	pbkdf2_sha256$100000$LTeIM7F85aAu$zaLKn...	2018-04-05 18:25:53.056574	0	Amelie
81	pbkdf2_sha256$100000$AutRAvk64m3r$O7U4...	2018-04-07 08:24:55.724730	0	user4
91	pbkdf2_sha256$100000$kI47mcTvnN8H$ABha...	2018-05-16 00:54:24.438141	0	girishtiwale
101	pbkdf2_sha256$100000$8NGMD6KOCZHt$ehE...	2018-05-11 00:02:17.734735	0	Amelie C
111	pbkdf2_sha256$100000$8f70UfD063dG$YVlaN...	2018-05-07 23:45:03.838205	0	srsutherland2
112	pbkdf2_sha256$100000$AuDBaZ4vSVbT$if+kn...	2018-04-16 00:12:04.364996	0	a
121	pbkdf2_sha256$100000$dbi8aRHaMKZl$Sn/Jo8...	2018-05-07 23:21:00.637426	0	test
131	pbkdf2_sha256$100000$ooIwT3za5ekn$1WKO...	2018-05-14 06:49:36.229102	0	lancevdoa
141	pbkdf2_sha256$100000$VCTfzw9UIXiz$apwOV...	2018-05-14 06:54:57.912919	0	lance
NULL	NULL	NULL	NULL	NULL

Form input validation

1. Registration form is validated by using {% csrf_token %} from django framework and JQuery Validation from bootstrap easy-to-use protection against Cross Site Request Forgeries and it will check if the user filled in the required fields: Email Address, User Name, Password, and Confirm Password, Captcha.

2. Creating a posting is validated using bootstrap validation and {% csrf_token %} and it will check if the user filled in the required fields: hazard report title, hazard report description, category, address and zip code.

6. 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. **(ON TRACK)** 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 e-mail 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).