

SW Engineering

CSC 648 [06Z]

Fall 2023

Team 3

Louis Leblond Team Lead

louis.leblond@epitech.eu

Roy Meyer Database, Backend

Sungjun Yun Frontend

Milestone 4

11/07/2023

Version	Date
1.00	2023-11-08

1. Product summary

Emergency Track is a comprehensive crisis management solution meticulously engineered by SFSU software engineering students to deliver real-time, accurate information and foster a safer, more informed California.

Major Committed Functions:

- **User Authentication and Authorization:** Emergency Track will feature a secure sign-in process with encryption.
- **Customizable Alerts:** Users can tailor alert preferences, ensuring they receive vital information that is pertinent to their safety and well-being.
- **Advanced Search and Filters:** Efficient data retrieval is made possible through sophisticated search and filtering tools.
- **User-Friendly Registration and Account Management:** A straightforward registration process and easy account preference modifications enhance the user experience.
- **Intuitive Information Display:** Information is presented clearly, with visuals and customizable dashboards for different user types.
- **Comprehensive Admin Controls:** System administrators are equipped with extensive controls for monitoring and managing the platform.

Unique Selling Points:

- Emergency Track stands out with its user-centric design, allowing for real-time, location-based emergency alerts and information which are easy to navigate even for users with minimal technical experience.
- The platform is unique in its ability to serve both registered and public users, providing critical information with or without account creation.

URL to the Product:

<http://146.190.120.47:8080>

2. Usability test plan

Test Objectives:

The usability test aims to assess the search functionality and the associated display of departmental markers on Google Maps. The intent is to evaluate the ease of use, the intuitiveness of the search interface, the accuracy of the search results, and the clarity with which the markers are displayed on the map. This test focuses on user satisfaction and their interaction with the map feature to ensure a positive user experience.

Test Background and Setup:

System Setup: Testing will be conducted on a high-resolution monitor connected to a computer with sufficient processing power and RAM to ensure smooth operation. The latest version of Google Chrome will be installed.

Starting Point: The test will commence at the application's homepage, where the Google Map is embedded, and the search functionality is accessible.

Intended Users: The target users are individuals who rely on mapping services to locate various departmental incidents, such as traffic, weather, or security-related events.

URL: The test will be conducted on the development version of the application, accessible at a specified URL provided to testers.

Measurements: User satisfaction will be measured using a Likert scale questionnaire focusing on the user's ease of interaction and satisfaction with the search function and map markers.

Usability Task Description: Testers will be instructed to use the search functionality to locate different departmental incidents within a specified county.

1. They would input a county name into the search bar.
2. They would select a department from a dropdown menu.
3. They would initiate the search to display relevant markers on the map.

Measurement of Effectiveness:

Effectiveness will be measured by the ability of the users to complete their intended search and correctly identify the markers corresponding to different departments on the map without errors.

Measurement of Efficiency:

Efficiency will be gauged by the time it takes for users to complete a search and successfully identify the departmental markers on the map.

Likert Scale Questionnaire:

1. "The search function was easy to understand and use."
 - Strongly Disagree
 - Disagree
 - Neither Agree nor Disagree
 - Agree
 - Strongly Agree

2. "The markers on the map accurately represented the searched department."
 - Strongly Disagree
 - Disagree
 - Neither Agree nor Disagree
 - Agree
 - Strongly Agree

3. "Overall, I am satisfied with the search and map marker functionality of the application."
 - Strongly Disagree
 - Disagree
 - Neither Agree nor Disagree
 - Agree
 - Strongly Agree

3. QA test plan

Test Objectives:

The objective of the QA test plan is to ensure that the search and marker display functionality within the React&Django application operates correctly and as expected across Chrome browsers.

HW and SW Setup:

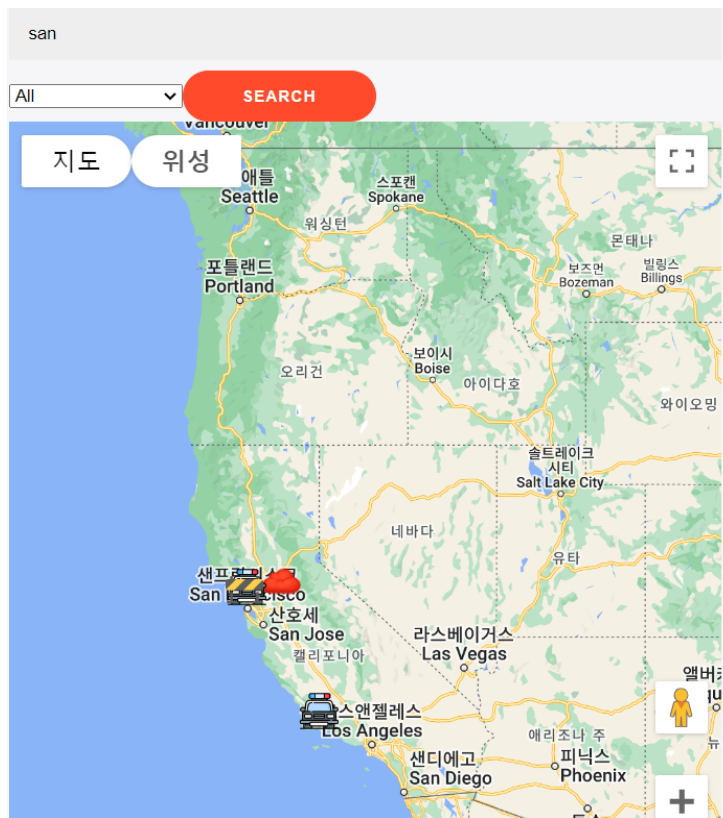
Test environment will include a variety of devices with different hardware specifications to simulate a range of user conditions. Software setup includes the latest versions of Google Chrome. The application will be hosted on a local development server with the URL provided during the testing session. Feature to be Tested: The search functionality and the subsequent display of departmental markers on Google Maps within the application.

QA Test Plan:

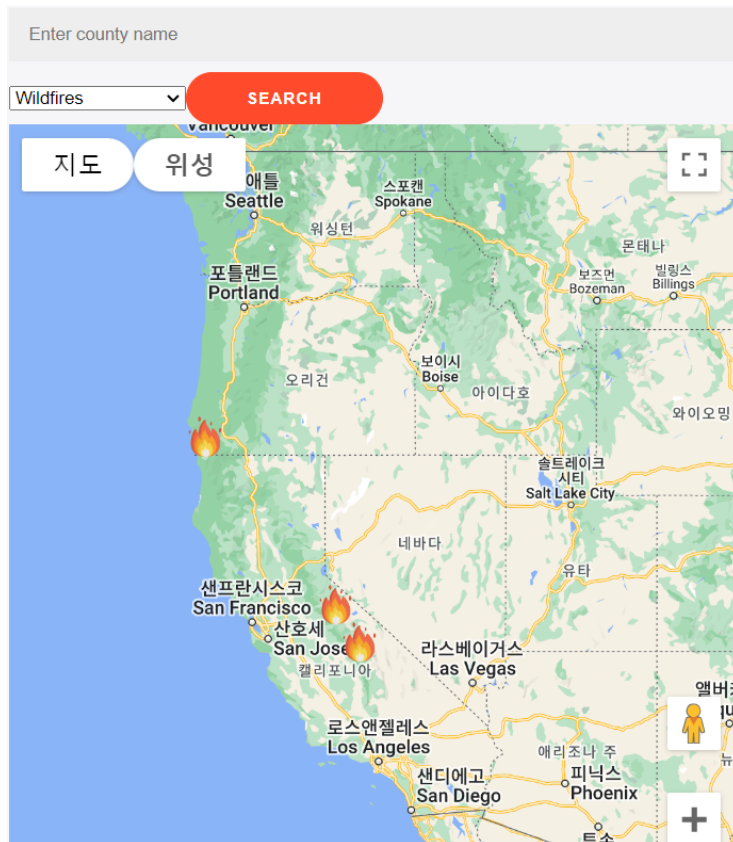
Develop a test case table that outlines each test scenario, including the expected results and the actual outcomes for each browser tested.

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Result
1	Basic County Search	Test basic search functionality with specific word and all department selection.	"San" "All"	Show all data of County which have word "san"	PASS
2	Basic Department Search	Test basic search functionality with all county and one specific department selection.	"%Nothing" "Wildfires"	Show all data of Wildfires!	PASS
3	Specific Department Search	Verify that selecting a specific department filters the markers correctly.	'Tulare County', 'wildfires'	Show Tulare County's Wildfires data	PASS
4	Basic Total Search	Check if interacting with a marker displays additional information.	"%Nothing" "All"	Show all Data	PASS

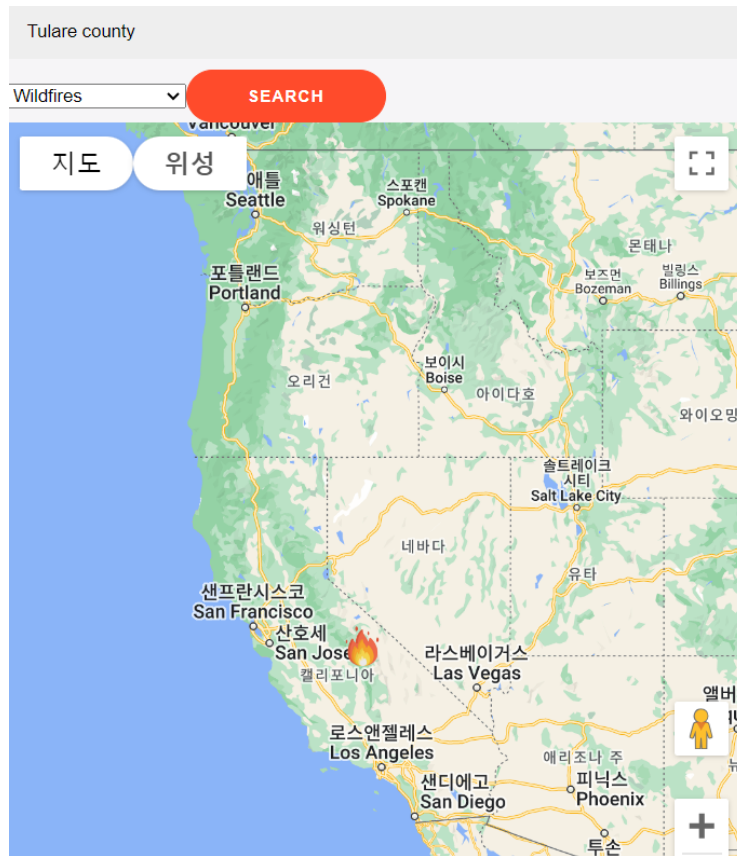
Test #1 Image



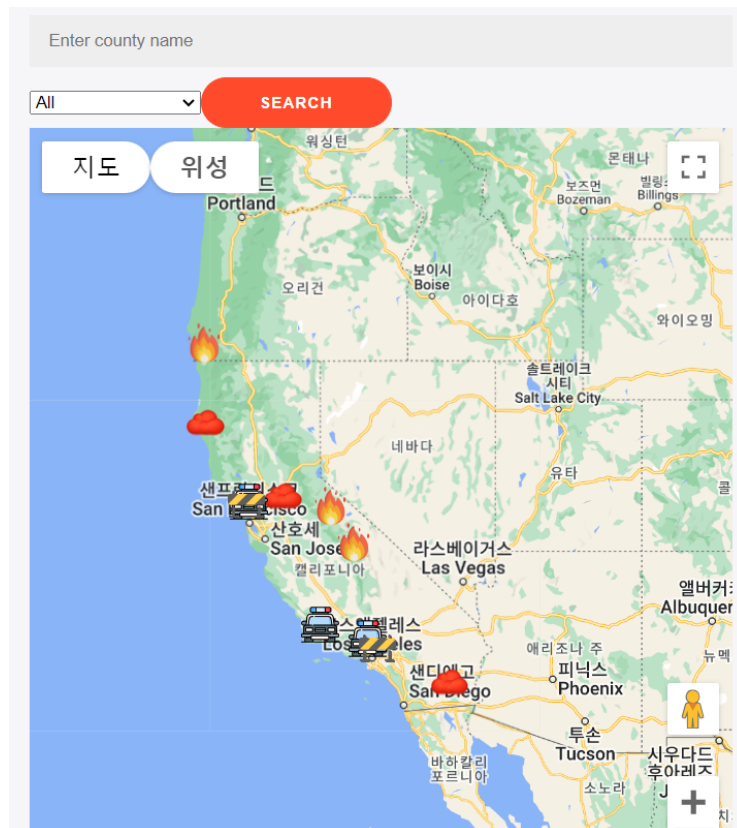
Test #2 image



Test #3 Image



Test #4 Image



4. Code Review

Selected Code for Review:

The review will focus on the React GoogleMap component that interfaces with the Google Maps API and handles the dynamic display of markers. Additionally, the Django view LocationsAPIView, which serves filtered location data to the frontend based on user search queries, is included in this review.

Coding Style Chosen:

Our development team adheres to the Airbnb React/JSX Style Guide for writing React components. This guide prioritizes readability and maintainability, advocating for practices such as using functional components, leveraging hooks for state and lifecycle management, destructuring props for clearer access, and favoring explicit returns in arrow functions. The guide also provides clear directives on structuring components and organizing files, which facilitates easier code navigation and consistency across our codebase.

Code Review Feedback:

Consolidation of API Calls:

The refactor to use a single API with query parameters instead of multiple endpoints is a modern approach that simplifies the frontend logic and is easier to maintain. Encoding Query Parameters: Properly encoding the query parameters is a best practice to avoid potential bugs related to improper URL formatting.

Custom Marker Icons:

The addition of custom icons for the markers greatly improves the user experience by providing visual cues that differentiate between departments.

Refactoring handle Search

Before Refactoring:

```
const departmentEndpoints = {
  BlockedRoad: "api/blocked_roads",
  Security: "api/security_concerns",
  Weather: "api/weather_events",
  Wildfire: "api/wildfires",
};

const endpoint =
  departmentEndpoints[selectedDept as keyof typeof departmentEndpoints];

// Ensure both county and department are selected
if (endpoint && selectedCounty) {
  clearMarkers(); // Clear existing markers before setting new ones
  try {
    const response = await axios.get(
      `http://localhost:8000/${endpoint}/${selectedCounty}`
    );
    const locations = response.data; // Response should be an array of location objects
    console.log(locations);

    // Create new markers and add to map
    const newMarkers = locations.map((location: Location) => {
      const lat = location.location_x_coordinate;
      const lng = location.location_y_coordinate;
      const marker = new google.maps.Marker({
        position: new google.maps.LatLng(lat, lng),
        map: googleMap,
        title: location.location_name,
      });
      console.log("Marker created:", marker); // Log the marker object
      return marker;
    });
```

- The code utilized a departmentEndpoints object to map departments to specific API endpoints.
- It constructed the endpoint URL by appending the selected department and county directly into the URL string.
- New marker creation did not specify an icon, implying default markers were used.

After Refactoring:

```
// Function to call the unified API and add markers to the map
const handleSearch = async () => {
  // Ensure both county and department are selected
  clearMarkers(); // Clear existing markers before setting new ones
  try {
    // Construct the API endpoint with query parameters
    const apiEndpoint = `http://localhost:8000/api/locations/?county=${encodeURIComponent(
      selectedCounty
    )}&department=${encodeURIComponent(selectedDept)}`;

    // Fetch data from the API
    const response = await axios.get(apiEndpoint);
    const locations = response.data; // Response should be an array of location objects
    console.log(locations);

    // Create new markers and add to map
    const newMarkers = locations.map((location: Location) => {
      const lat = Number(location.location_x_coordinate);
      const lng = Number(location.location_y_coordinate);
      // Define the icon with a scaled size
      const icon = {
        url: departmentMarkers[
          location.department as keyof typeof departmentMarkers
        ], // Get the icon based on the selected department
        scaledSize: new google.maps.Size(30, 30), // Set the size you want here
      };

      return new google.maps.Marker({
        position: { lat, lng },
        map: googleMap,
        title: selectedDept,
        icon: icon,
      });
    });
  }
};
```


- The code was simplified by consolidating multiple endpoints into a single locations API endpoint with query parameters for the county and department.
- The construction of the API endpoint URL now includes encodeURIComponent for the query parameters, ensuring special characters are properly encoded for the HTTP request.
- The new marker creation now includes an icon with a scaled size, providing a customized visual representation for different departments.



Screenshot of Peer review

Talking about Icons


 **Roy** 11/07/2023 2:22 PM
I should have studied for my midterm tomorrow but instead I took some time to make different icons for us.


I basically recolored these emojis as they are displayed by apple.


   





 **Sungjun** 11/07/2023 2:23 PM
Ok


 **Roy** 11/07/2023 2:23 PM
let me know if you have ideas for improvement



 **Sungjun** 11/07/2023 2:38 PM
I preferred my icons... except weather icon


 **Roy** 11/07/2023 2:53 PM
Okay. I like the Idea of having our color scheme inside the icons.
We can have Louis decide, wich ones we should use 😊

 **Louis** 11/07/2023 2:55 PM
I like both but prefer maybe sungjun ones
Except for the thunder with we can not see enough on the map


 **Roy** 11/07/2023 2:56 PM
okay 👍

 **Sungjun** 11/07/2023 3:01 PM
Roy can u send me red cloud image?


 **Roy** Click to see attachment 


 **Roy** 11/07/2023 3:02 PM
should be inside here

Combining API

 **Roy** 11/06/2023 3:35 PM
Okay the Api for the different events is also running.
Depending on the department/event you have to access a different url.

localhost:8000/api/blocked_roads/<str:name>
localhost:8000/api/security_concerns/<str:name>
localhost:8000/api/weather_events/<str:name>
localhost:8000/api/wildfires/<str:name>
if you request it without /<str:name> you get all the data (edited)

 **Sungjun** 11/06/2023 3:35 PM
name is for county?

 **Roy** 11/06/2023 3:35 PM
yes

 **Roy** 11/06/2023 7:09 PM
/api/locations/?county=[name]&department=[department] (edited)

 **Sungjun** 11/06/2023 7:46 PM
where are u guys

 **Roy** started a call that lasted 3 hours. 11/06/2023 7:46 PM

  **@Roy** /api/locations/?county=[name]&department=[department] (edited)

 **Sungjun** 11/06/2023 8:21 PM
So when u make this api just send the whole data

  **Roy** /api/locations/?county=[name]&department=[department] (edited)

 **Roy** 11/06/2023 9:00 PM
done

  **@Roy** /api/locations/?county=[name]&department=[department] (edited)

 **Sungjun** 11/06/2023 10:35 PM
this api

 **Roy** 11/06/2023 10:43 PM
all
wildfires
blocked_roads
security
weather

We usually call each other and feedback each others.

5. Self-check on best practices for security

Our Major asset is the user and of the citizens and especially of the officers which are stored in the database.

Our database access is password protected and we prevent SQL injection by using Django's Object Relational Mapping (ORM) layer whenever there is an SQL Request rather than direct SQL Queries. The ORM layer uses query parameterization to avoid sql injection and cross site scripting attacks.

Unfortunately our login and register isn't working at the moment, but we are planning to utilize Django's Password Hashing in order to encrypt the user passwords in the Database.

The input data validation is done by Django's ORM as well as its `is_valid()` method on the serializers. This method is always called, when a new dataset in the database is created or changed. The `is_valid()` method makes sure that the given user input fits the datatype of our database. The 40 character limit on each input is archived in react by including the `maxLength={40}` property on every `<input>` tag.

6. Self-check: Adherence to original Non-functional specs – performed by team leads

1. **ON TRACK:** 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 two major browsers.

3. **ON TRACK:** Selected application functions must render well on mobile devices (this is a plus).

4. **DONE:** Data shall be stored in the team's chosen database technology on the team's deployment server.

5. **ON TRACK:** Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users.

6. **DONE:** The language used shall be English.

7. **ON TRACK:** Application shall be very easy to use and intuitive.

8. **ON TRACK:** Google maps and analytics shall be added
9. **ON TRACK:** No email clients shall be allowed. You shall use webmail.
10. **DONE:** Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.
11. **ON TRACK:** Site security: basic best practices shall be applied (as covered in the class)
12. **ON TRACK:** Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
13. **DONE:** The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Fall 2023. For Demonstration Only" at the top of the WWW page. (Important so not to confuse this with a real application).