SW Engineering CSC648/848 Spring 2023
SafetyHub

Section: 02
Team: 03

Diego FLORES - Team Lead - Database - dfloresflores@mail.sfsu.edu
Carmelo DE GUZMÁN - Front End Lead - cdeguzman@mail.sfsu.edu
Coline SEGURET - Github Master - cseguret@sfsu.edu
Sumith SHRESTHA - Front End - Sshrestha12@sfsu.edu
Adrian VAZQUEZ - Back End Lead - Database - avazquez10@mail.sfsu.edu
Isabel FALCON - Back End - ifalcon@mail.sfsu.edu

Milestone 2
03/14/2023

Revision History Table

Description	**Date Submitted**	**Date Revised**
Initial Document Outline	03/12/23	
Functional Requirements (Prioritized)	03/13/23	
UI Mockups and Storyboards	03/12/23	
High Level Architecture, Database Organization	03/13/23	
High Level UML Diagrams	03/13/23	
Identify Key Risks	03/12/23	
Project Management	03/12/23	
Document Submission	03/14/23	

1. Functional Requirements (Prioritized):

Priority 1 (Must Have)

1. Users shall be able to receive notification alerts
    - 1.1 Current and past alerts will be in a chatbox that the user can scroll through
9. Users shall have login authentication.
    - 9.1 To be done via sessions so the user does not have to log in whenever they access a new page.
5. Users shall be able to search through categories or by text
    - 5.1 The search will be “Fuzzy” using %Like
3. Users shall be able to view entities on the map
    - 3.1 These entities will consist of covid, weather, etc.
8. System shall have a registration form where users can register
    - 8.1 Shall contain fields such as name, email, etc. that will be stored in the database
10. System shall have authorization levels (user, admin)
    - 10.1 There will be two authorization levels: user and admin. The admin can post content (input data/metrics) whereas the user cannot.
6. Users shall be able to filter by type of emergency
    - 6.1 Examples of such emergencies include covid, weather, etc. to avoid cluttering the map
15. Admin shall be able to input metrics
    - 15.1 As an input in the admin management dashboard

Priority 2 (Desired)

4. Users shall be able to zoom in/out of the county map
14. System shall allow users to reset/change their password
    - 14.1 In the case that the user knows their password but wishes to change it
11. System shall contain historical data
    - 11.1 Historical data will consist of past emergencies posted on the site.
    - 11.2 Historical data will be stored in the database.
18. System shall have Social Media Bot integration
    - 18.1 Whenever an admin posts data on the website the bot will post the same data on the social media platform
21. System shall be able to display the user’s recent search history
    - 21.1 Displays the last 3-5 search queries

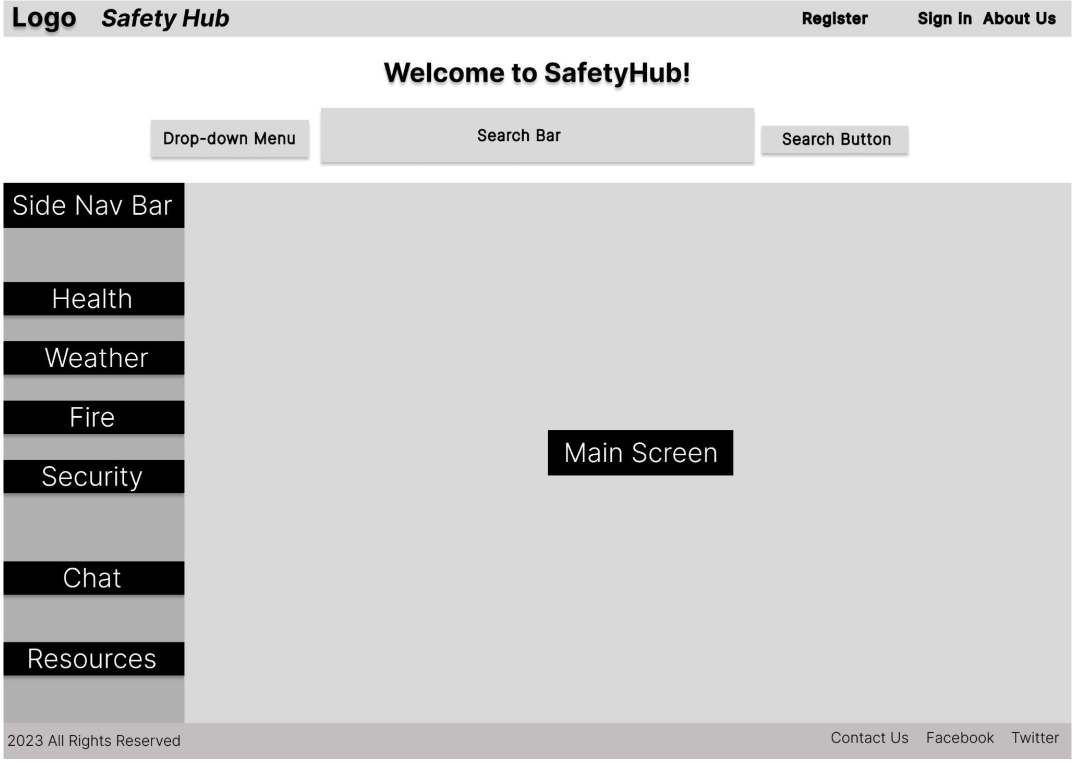
22. System shall provide users the opportunity to log in if they forgot their password
16. Admin shall be able to review/approve alerts before going live on the website
    16.1 Located in the admin management dashboard
17. Admin shall be able to remove registered users if not in compliance with TOS
    17.1 Accomplished by searching for the user id in the admin management dashboard

Priority 3 (Opportunistic)

7. Users shall be able to find information about resources
    7.1 This will be in the form of a help page that the user can access on the website. This page will contain links and things such as helpline numbers if the user requires psychological aid.
19. System shall have multilingual support
20. System shall be able to export data in CSV or PDF format

2. <u>UI Mockups and Storyboards (High Level):</u>

2.0 - ***Home Page***



- This is the UI for our landing page that shall serve as the main home page for our website SafetyHub. As for our UI's functionalities, we shall categorize it into two groups, *static* and *dynamic*, where static areas shall remain relatively fixed onto the website layout while dynamic areas shall change depending on the content currently being viewed by the user
  - Both the *header* and the *footer* sections of the UI shall remain static whenever the user scrolls up or down the webpage
    - However, clickable buttons/links such as the register button, about us link, contact us link, etc. shall always be made accessible to the user
  - The *side navigation bar* shall also remain static throughout the website, although, we plan on having the option to auto-hide this navigation bar through a clickable button, if the user desires it
    - In the navigation bar, there are clickable tabs that shall redirect the user to a different page such as the weather or health section of the site
  - As for the *main screen* portion of our website UI, this area shall remain dynamic, where it shall change depending on the content being viewed

        - We shall plan on having it be integrated with our Map API to show the “mini map” of a randomly-selected county, where it shall display current weather and COVID-related metrics
    - In terms of the *search bar*, it shall also serve as a dynamic function, where the user shall interact with it depending on their intended action
        - The user shall be able to click on the *drop-down menu bar* to search by categories while also being able to click on the *search button*
        - We shall also plan on adding in a feature where, if the user clicks on the search bar, a drop-down menu shall display their recent search history

2.1 - ***Chat***

Figure 2.1.1

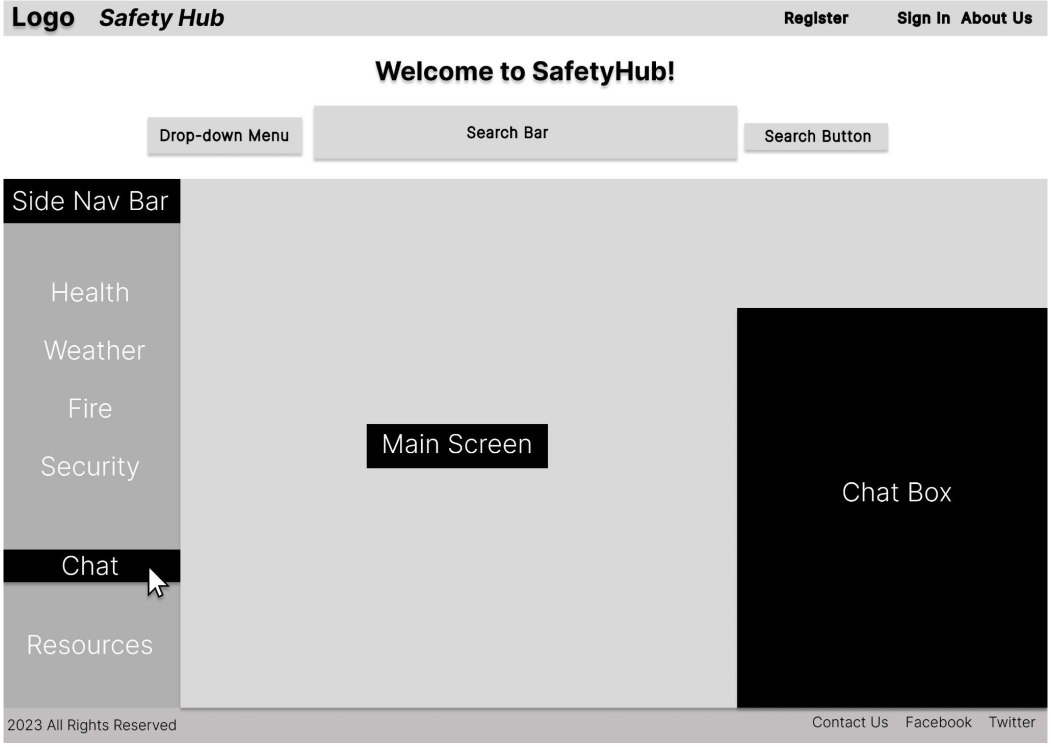
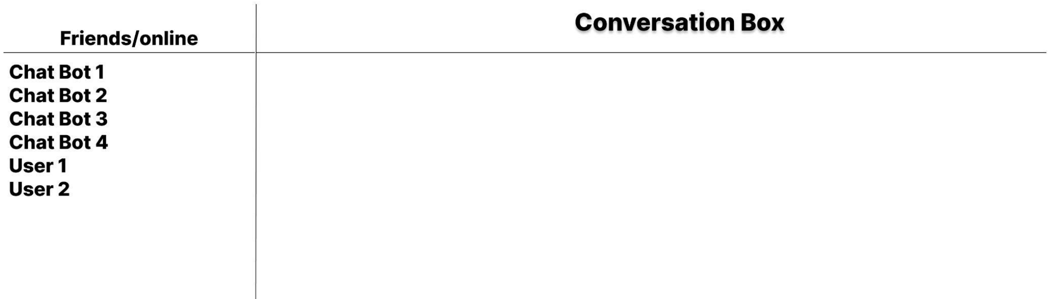


Figure 2.1.2



- In regards to the website UI for our chat use case, Figure 2.1.1, we shall plan on having the user click on the *chat* button on the side navigation bar, where a mini-chat box shall appear on the screen
    - As shown in Figure 2.1.2, if the user clicks on the mini-chat box to expand it, it shall redirect them to a page that displays the full window UI of the chat box

2.2 ***Emergency Push Notifications***

Figure 2.2.1

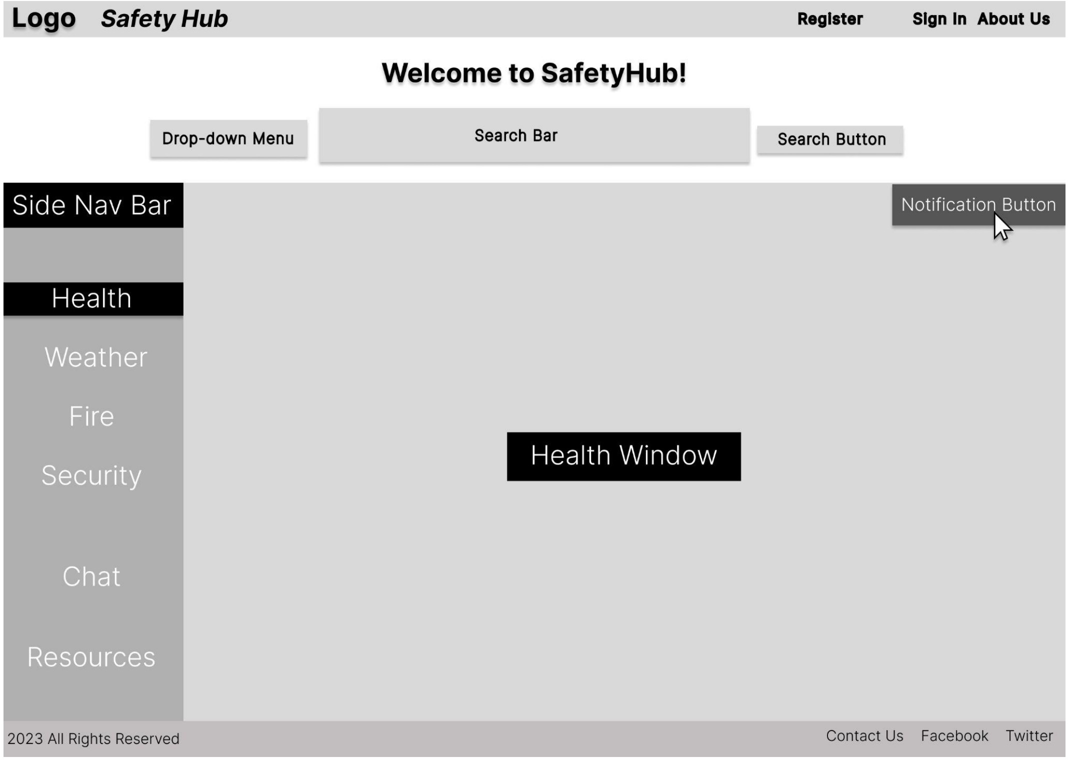
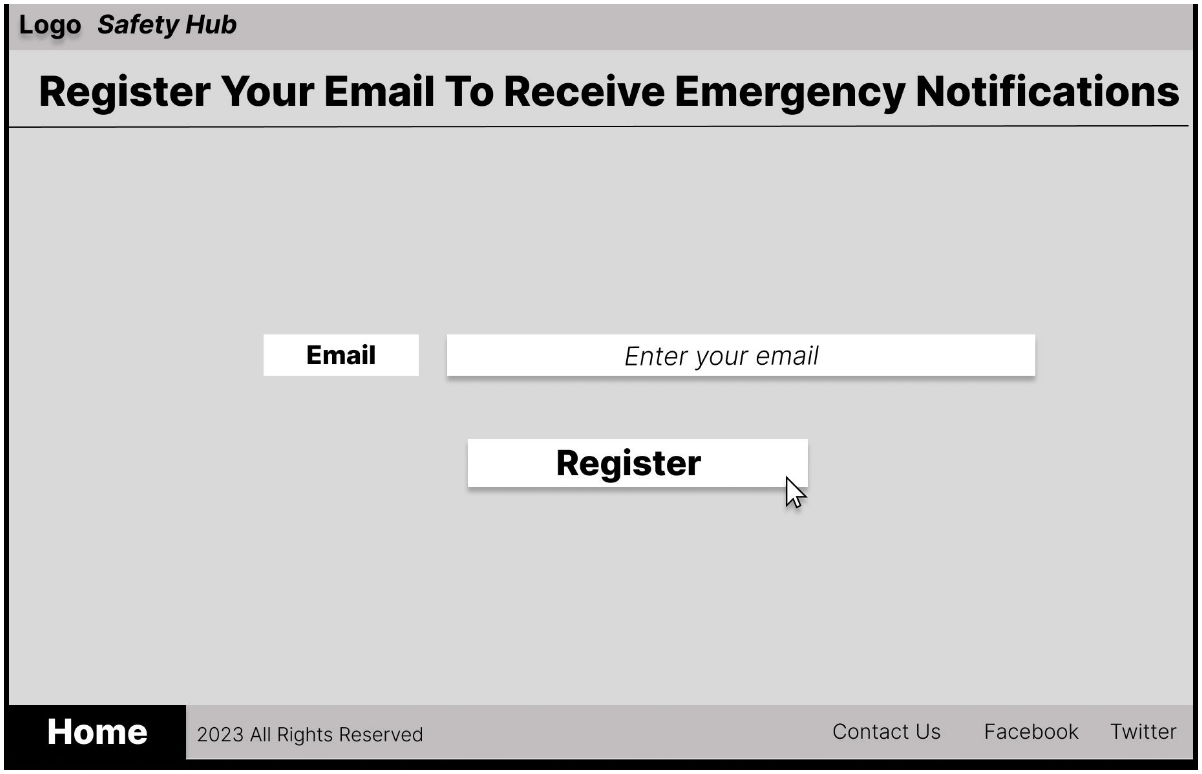


Figure 2.2.2



- As for the emergency push notifications UI use case, the user shall be able to click on a button on the top-right corner of the *main screen*, Figure 2.2.1 and Figure 2.2.2, to be redirect to the registration page for push notifications
  - As seen in Figure 2.2.2, the user shall be able to register their email address to receive emergency push notifications through their email provider

2.3 ***Maps***

Figure 2.3.1

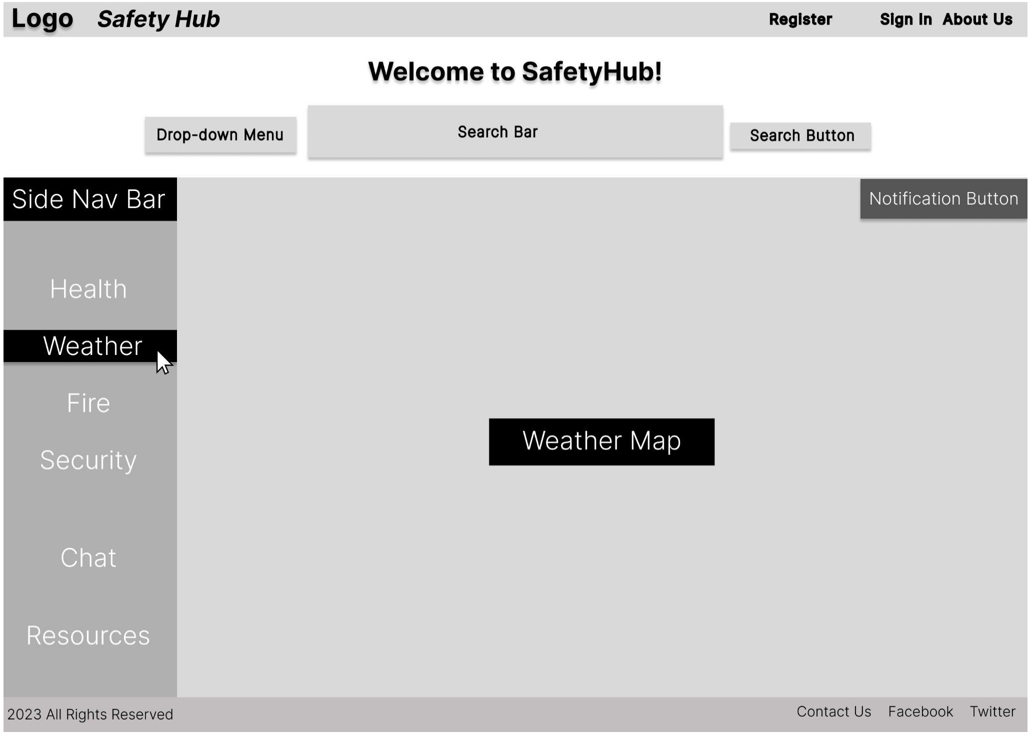


Figure 2.3.2

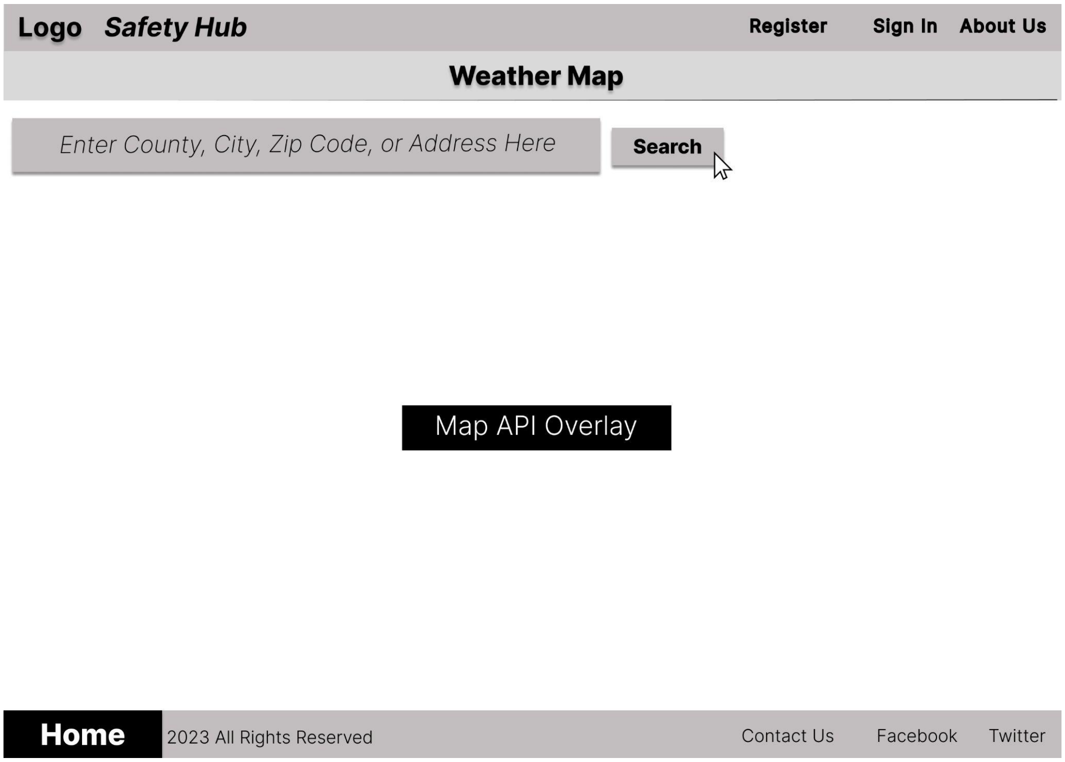


Figure 2.2.3

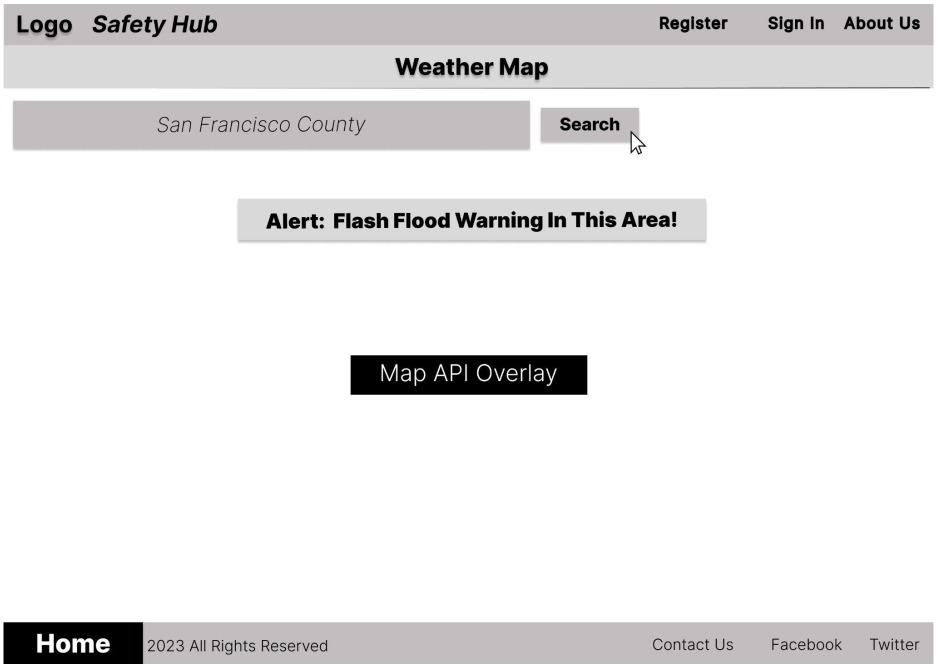


Figure 2.2.4

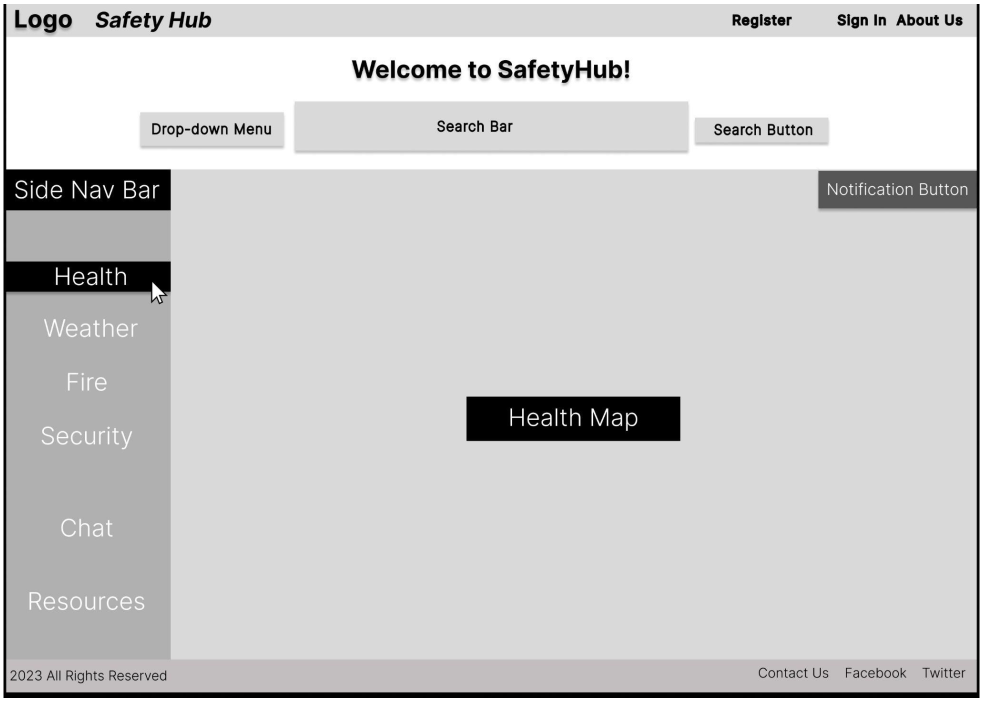


Figure 2.3.5

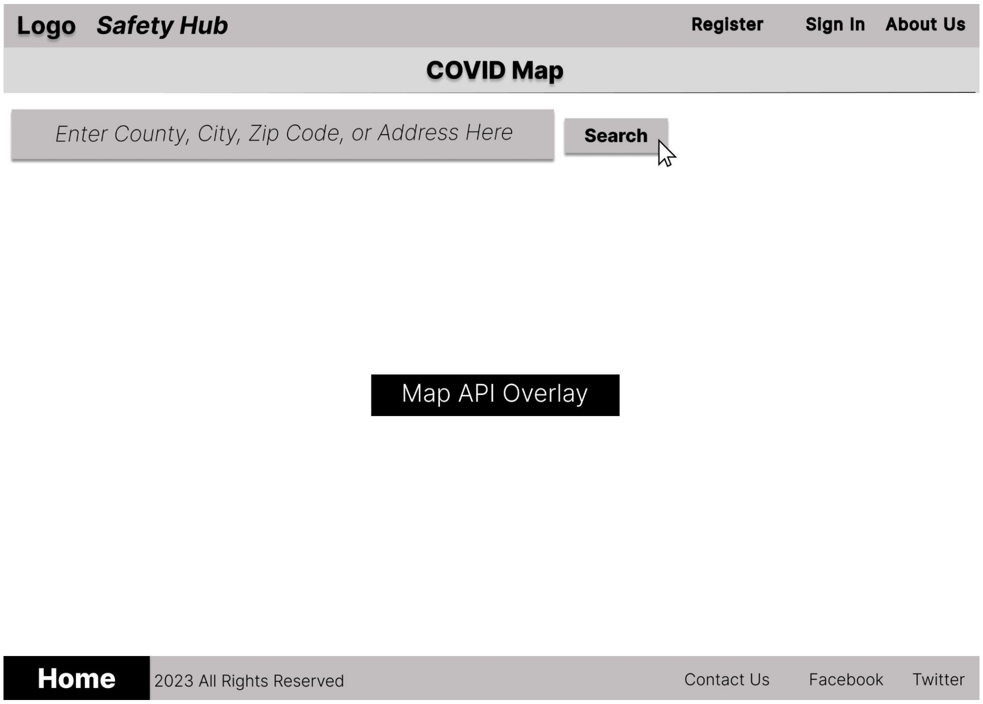
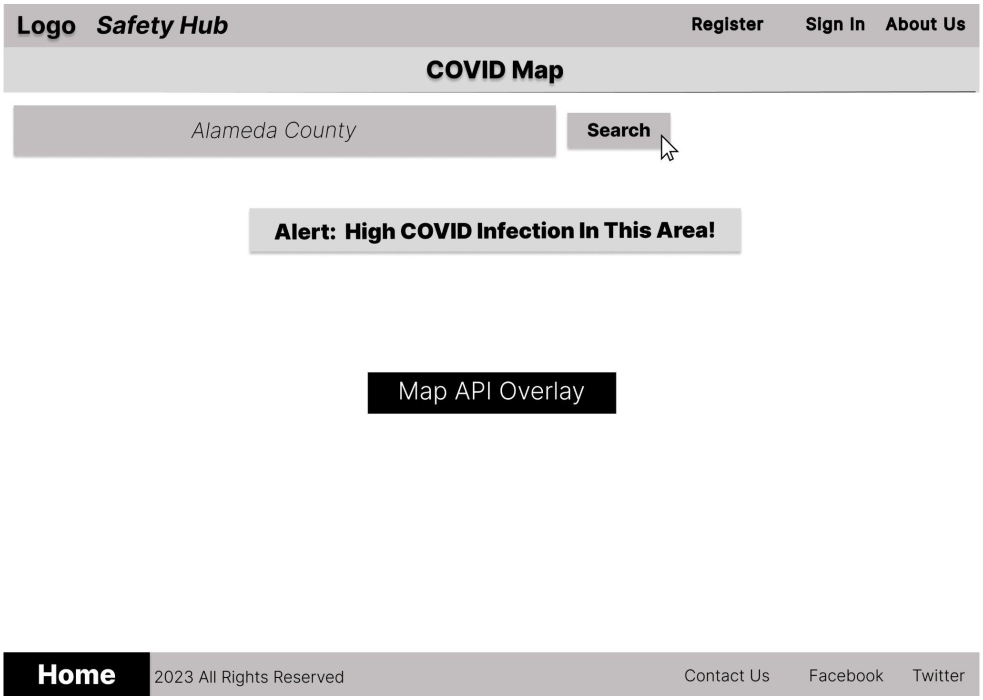


Figure 2.3.6



Alameda County

Alert: High COVID Infection In This Area!

- As for the website UI for the Maps use case, users shall be able to click on any of the tab links in the side navigation bar to display a map of a randomly selected county (Figures 2.3.1, 2.3.4), where it shall contain current weather and COVID-related metrics
    - When the user wants to check on a specific area for weather information or COVID-metrics, they shall be able to type in a specific county, city, zip code, or address in the map search bar (Figure 2.3.2, 2.3.5)
    - Depending on the search area specified, the user shall receive a pop-up alert, informing them that severe weather or high COVID infection rate was reported in that area (Figure 2.3.3, 2.3.6)

2.4 ***Resource Page***

Figure 2.4.1

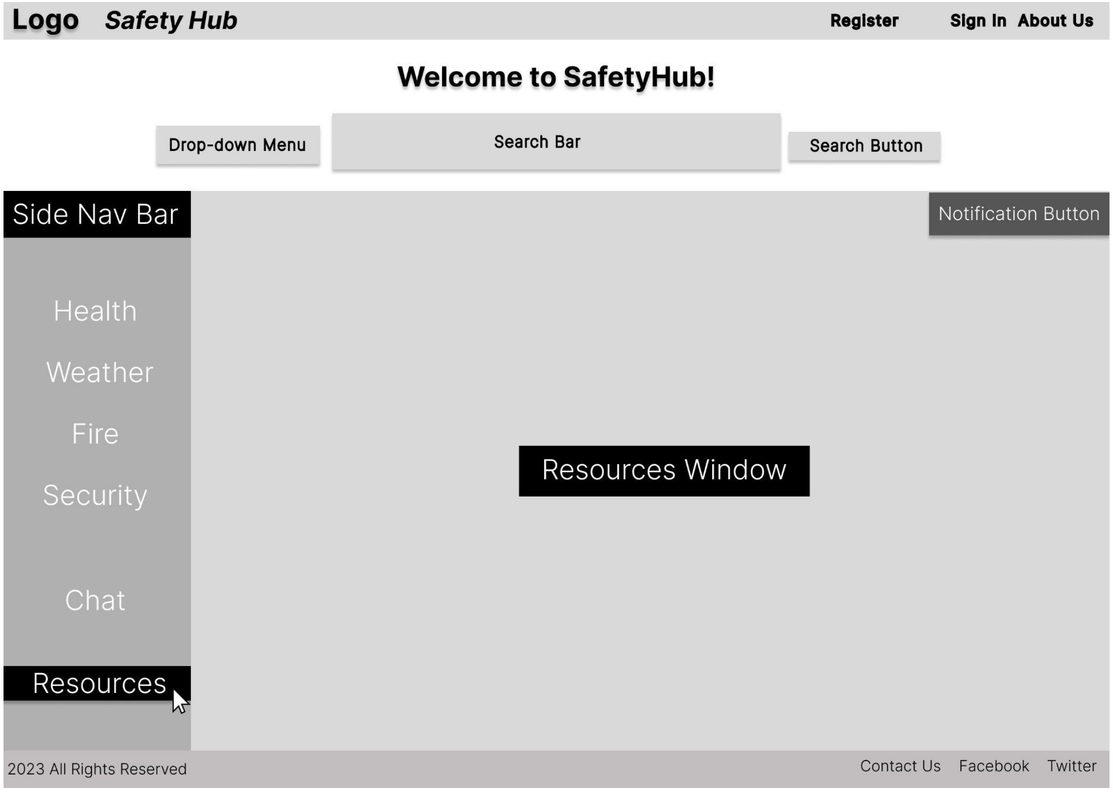
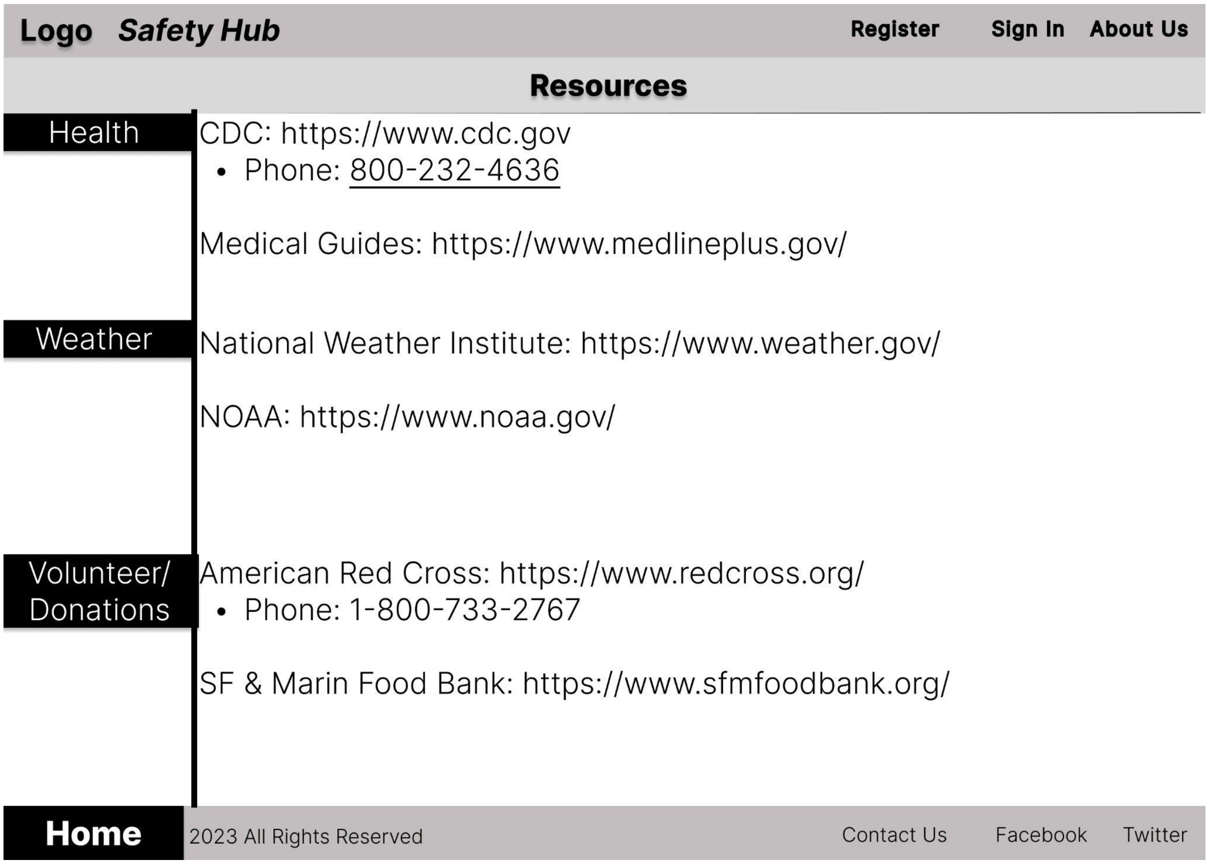


Figure 2.4.2



Resources

CDC: https://www.cdc.gov

- Phone: 800-232-4636

Medical Guides: https://www.medlineplus.gov/

National Weather Institute: https://www.weather.gov/

NOAA: https://www.noaa.gov/

American Red Cross: https://www.redcross.org/

- Phone: 1-800-733-2767

SF & Marin Food Bank: https://www.sfmfoodbank.org/

2023 All Rights Reserved

- In terms of the website UI for the *Resources* use case, users shall be able to click on the Resources tab to be redirected to the resource page of the our site (Figure 2.4.1)
  - In this page, users shall be presented with a list of contact and website resources for those who may need additional information such as from the CDC, the National Weather Institute, the American Red Cross, etc. (Figure 2.4.2)

3. High Level Architecture, Database Organization:

DB Organization:

Table counties:

Column name	Data type	Key	Description
id	uuid	PK	Unique identifier for the county
name	varchar		Name of the county
latitude	decimal(9,6)		The latitude of the county
longitude	decimal(9,6)		The longitude of the county

Table user:

Column name	Data type	Key	Description
id	uuid	PK	Unique identifier for the user
email	varchar		Email address of the user
password	varchar		Password for the user's account
county_id	uuid	FK	ID of the county the user belongs to
covid_alerts	boolean		Flag for whether the user has COVID-19 alerts enabled
fire_alerts	boolean		Flag for whether the user has fire alerts enabled
security_alerts	boolean		Flag for whether

			the user has security alerts enabled
user_type	varchar		Type of the user (Admin or User)

Table covid_alerts:

Column name	Data type	Key	Description
id	uuid	PK	Unique identifier for the COVID-19 alert
county_id	uuid	FK	ID of the county the alert is for
message	varchar		Message for the COVID-19 alert
created_at	timestamp		Timestamp of when the alert was created

Table fire_alerts:

Column name	Data type	Key	Description
id	uuid	PK	Unique identifier for the fire alert
county_id	uuid	FK	ID of the county the alert is for
message	varchar		Message for the fire alert
created_at	timestamp		Timestamp of when the alert was created

Table security_alerts:

Column name	Data type	Key	Description
id	uuid	PK	Unique identifier for the security alert
county_id	uuid	FK	ID of the county the alert is for
message	varchar		Message for the security alert
created_at	timestamp		Timestamp of when the alert was created

Table posts:

Column name	Data type	Key	Description
id	uuid	PK	Unique identifier for the post
user_id	uuid	FK	ID of the user who created the post
title	varchar		Title of the post
content	text		Content of the post
created_at	timestamp		Timestamp of when the post was created
updated_at	timestamp		Timestamp of when the post was last updated

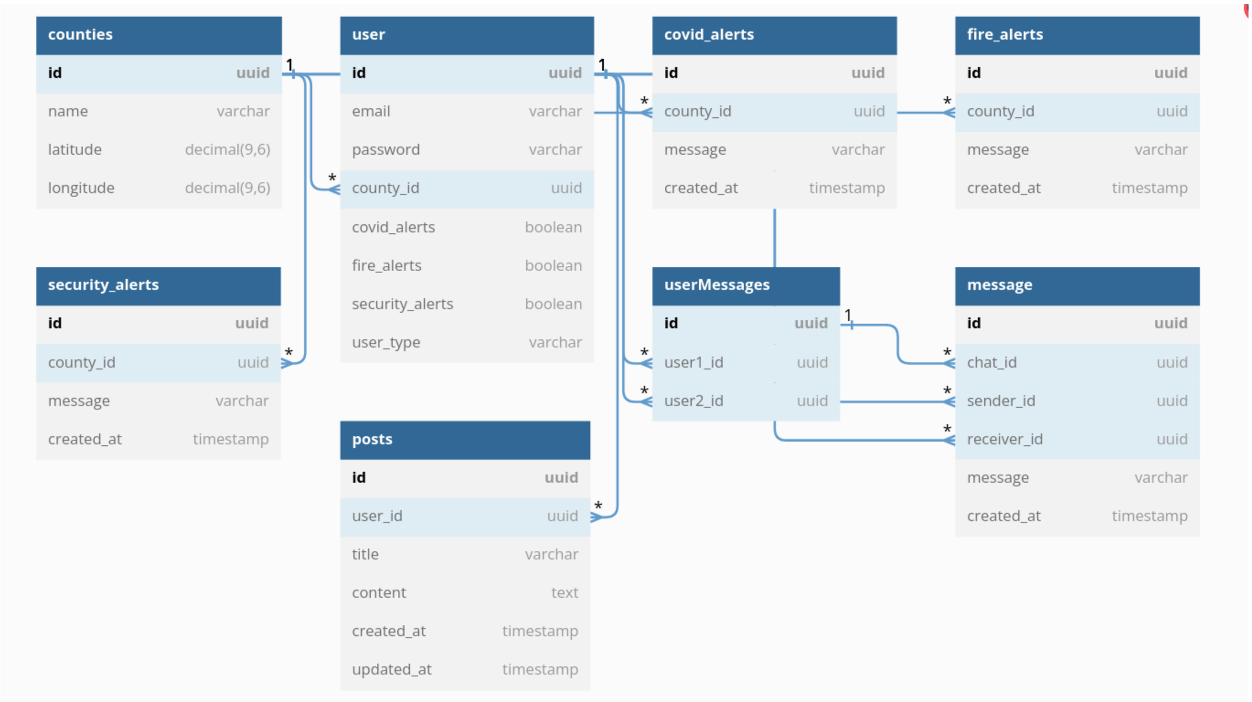
Table chat

Column Name	Data Type	Key	Description
id	uuid	PK	Unique identifier for each chat
user1_id	uuid	FK	Reference to the first user in the chat
user2_id	uuid	FK	Reference to the second user in the chat

Table messages:

Column Name	Data Type	Key	Description
id	uuid	PK	Unique identifier for each message
chat_id	uuid	FK	Reference to the chat where the message was sent
sender_id	uuid	FK	Reference to the user who sent the message
receiver_id	uuid	FK	Reference to the user who received the message
message	varchar		Text content of the message
created_at	timestamp		Date and time when the message was sent

You will find the link of the DBdiagrams for more interactivity



counties	
id	uuid
name	varchar
latitude	decimal(9,6)
longitude	decimal(9,6)

user	
id	uuid
email	varchar
password	varchar
county_id	uuid
covid_alerts	boolean
fire_alerts	boolean
security_alerts	boolean
user_type	varchar

covid_alerts	
id	uuid
county_id	uuid
message	varchar
created_at	timestamp

fire_alerts	
id	uuid
county_id	uuid
message	varchar
created_at	timestamp

security_alerts	
id	uuid
county_id	uuid
message	varchar
created_at	timestamp

userMessages	
id	uuid
user1_id	uuid
user2_id	uuid

message	
id	uuid
chat_id	uuid
sender_id	uuid
receiver_id	uuid
message	varchar
created_at	timestamp

posts	
id	uuid
user_id	uuid
title	varchar
content	text
created_at	timestamp
updated_at	timestamp

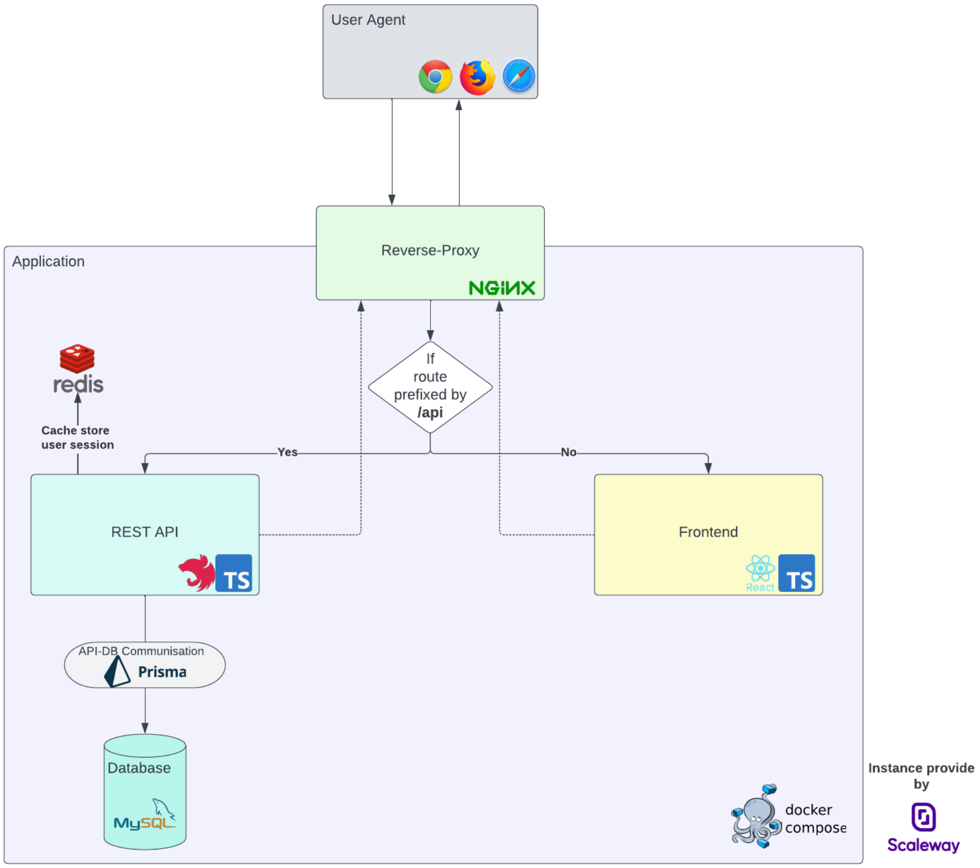
This architecture scheme is bound to evolve and all updates will be available on the db diagrams link above.

We will use MySQL to create the database because it is required of us and because we can use MySQL Workbench to check the database remotely via SSH.

Search/Filter Architecture and Implementation:

The search algorithm uses the SQL %LIKE% operator. The search algorithm will match all alerts whose type or location contains the search term entered.

We will use MySQL to query the database, NestJS and Typescript to send the results to the front-end, and ReactJS to display the results to the user. As shown in the architecture diagram below:



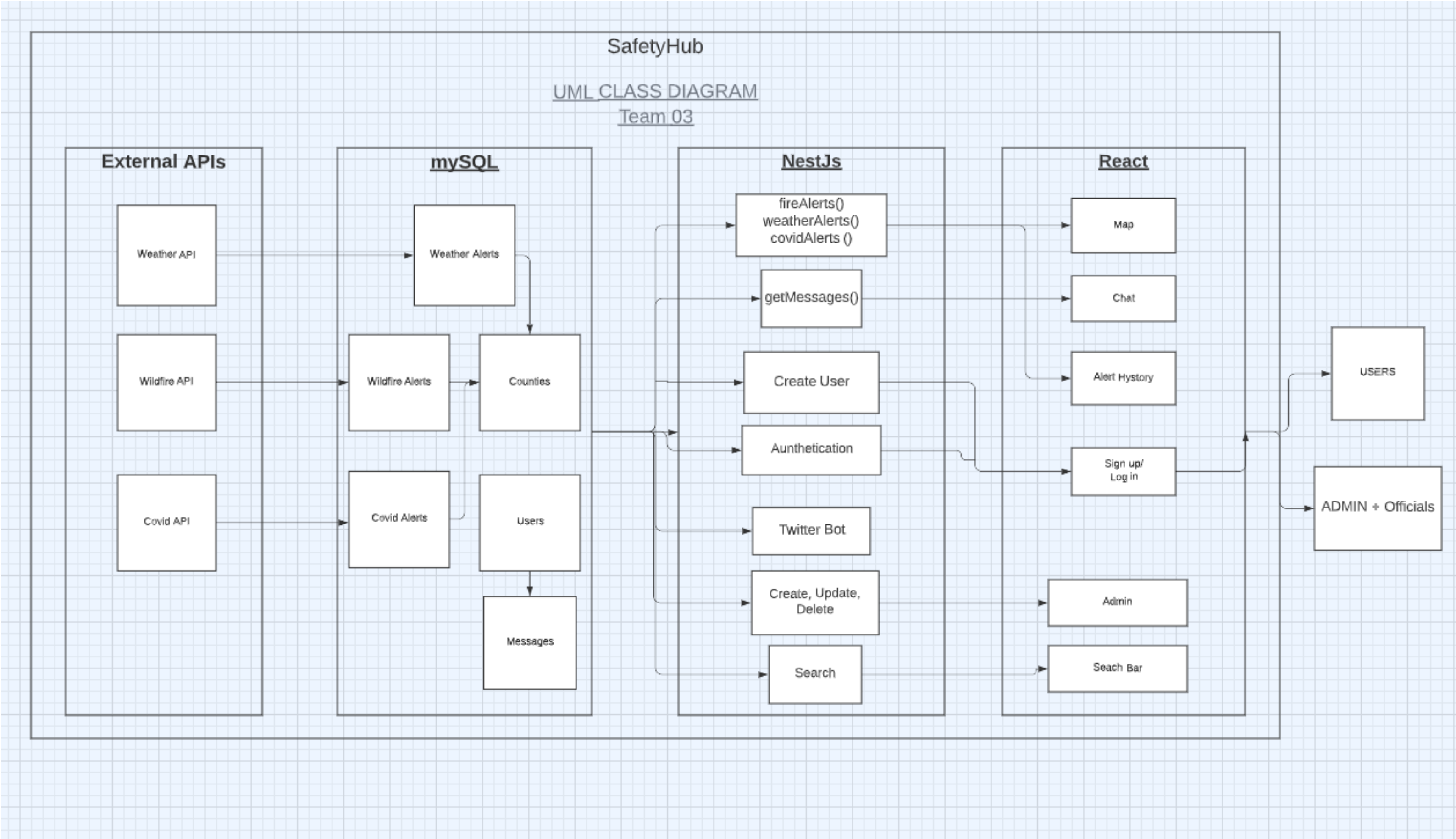
In addition, to facilitate communication with the db, we will use Prisma which is an ORM that will allow us to avoid writing SQL queries by hand.
ORMs are used to simplify and facilitate access to database data by providing an object-oriented programming interface.
For specific needs we will probably need to write sql queries

So the user will be able to search for specific

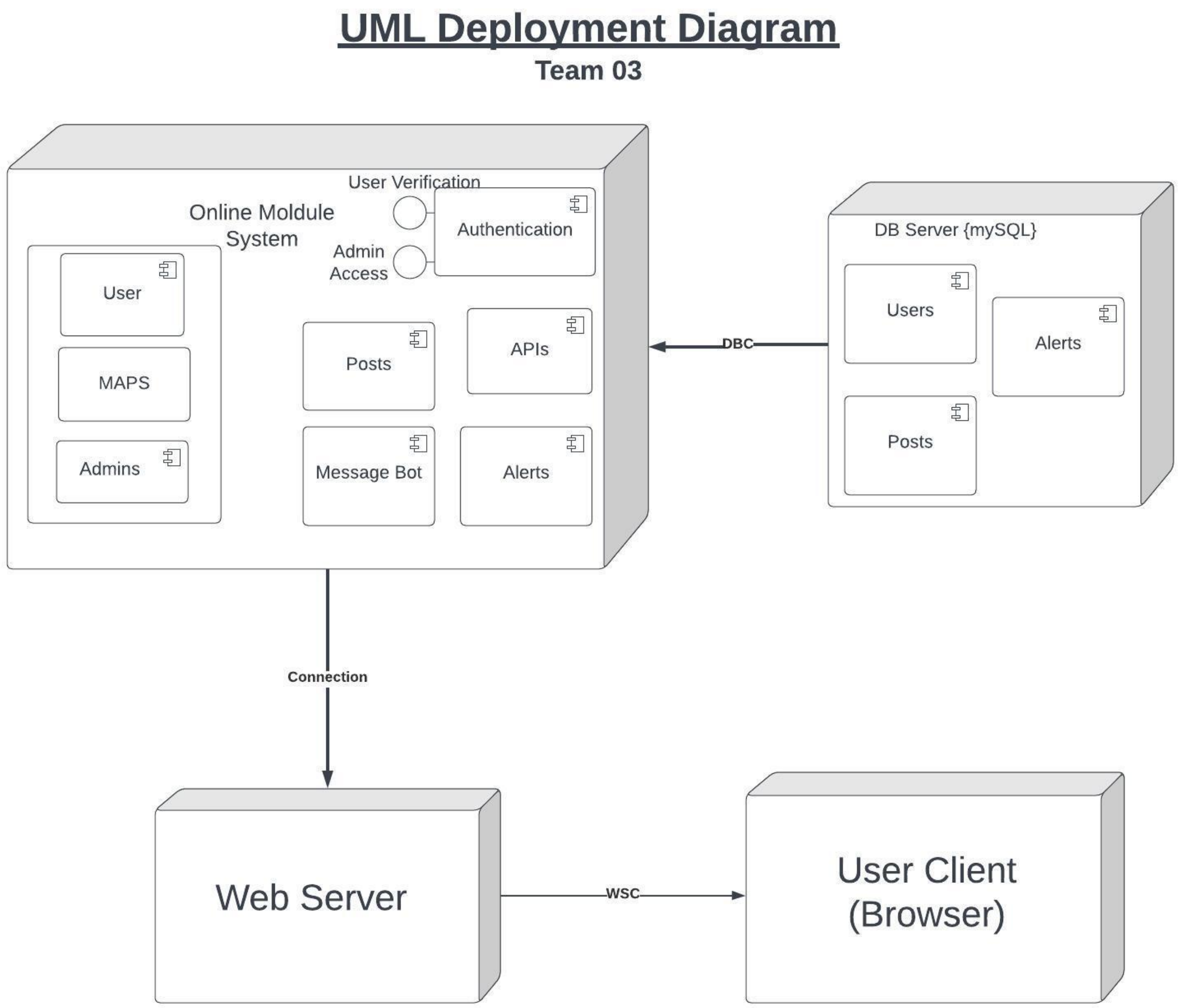
- specific alerts (e.g. covid or wildfire).
- alerts by geographical area.

4. High Level UML Diagrams:

Class Diagram:



Component and Deployment Diagram:



5. Identify risks for your project at this time:

Every team has its strong and weak points. We have identified the following to be our major risk factors:

- **Skills**: Most of our team is not familiar with the frameworks we are going to be using. Additionally, although TypeScript is a superset of Javascript, it will still take time to adjust to it.
- **Schedules:** Because of the number of team members, we find it troublesome to find times to meet that are not during class. We all have different schedules, so having meetings in which we are all present can be difficult.
- **Communication:** A lack of communication can lead our product to not be delivered in time.

Our approach to risk management:

- **Skills:** The team will work very hard to get comfortable with the different technologies we are using. This can happen through individual practice, going through the documentation, etc.
- **Schedules:** We will use websites like When2Meet to find the best meeting times when they become more important for our project.
- **Communication:** As of now, Discord is our main communication channel and has been enough to be up to milestones. We have different channels on our server so it's easier to keep track of conversations. Although we do not have many meetings, we have encouraged communication on the server. Group members can share their questions, concerns, and ideas through the server. However, working on arranging more meetings is our next step.

6. **Project Management:**

As of now, the way we have attempted to manage the team is by having each team member be in a fixed position and then assigning work to that position. Since this class already gives us an outline of what subgroups look like (front-end, back-end, database master, etc.). The process involves going through milestone requirements, and then breaking it down into sections. We then identify what subgroups are concerned with what section and assign work. Now, each section can have a subsection, but the lead of each subgroup can decide how to manage them. Once subgroups complete their tasks, the work done is put together, reviewed, and shipped. It's a cascade-like approach to team management.

We believe this method will continue to work for future assignments. It will definitely involve more communication between subgroups. That is when team meetings will become more crucial. Currently, every group member has a delegated section from the milestone, and if their section involves team decisions, we come together as a team and complete the section. In the future, there will be a stronger back-and-forth between group members, subgroups, and the group. Additionally, if milestones involve more work, we can set goals and due dates within the milestones.

We will ensure good communication by arranging more team meetings and additional involvement in the Discord server. Moreover, using websites like Trello will be helpful for the team.

I am aware that coding proficiency does not mean anything without good teamwork.