Project Milestone 4

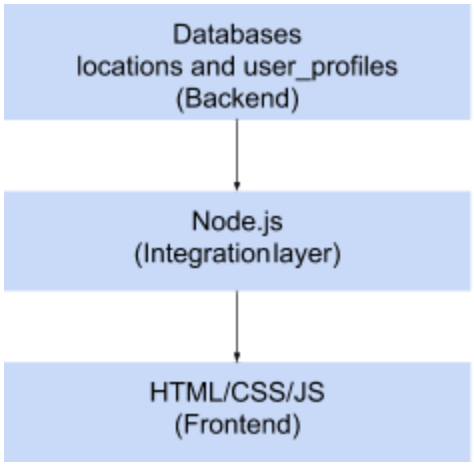
Group number: 4

Ziyue Guo
Robert Dumitrescu
Sarah Lundell
Kevin Xu
Tongxin Zhu

Revised List of Features (Ordered by Priority):

- Interactive Map
    - Interactive weather of Colorado, detailing general weather of state; reaches more detail when user selects region
    - Notifies user of significant events (rain, snow, storms, etc.) when area selected
        - Also houses general safety info regarding storms
- Realtime data reception
    - Weather API will be used in order to collect/store weather data and update the map in real time
- Dropdown Icon Menu
    - Allows user to filter weather type(s) shown on weather map
- Search Bar
    - Leverages user input, and returns desired region/weather type
    - Automatic detection/correction for misspelled/mistyped input
- Pins on major landscape
    - Ski resorts, national parks, 14ers…
- User Login
    - Login using a username and password
    - Users can save locations, subscribe to alerts

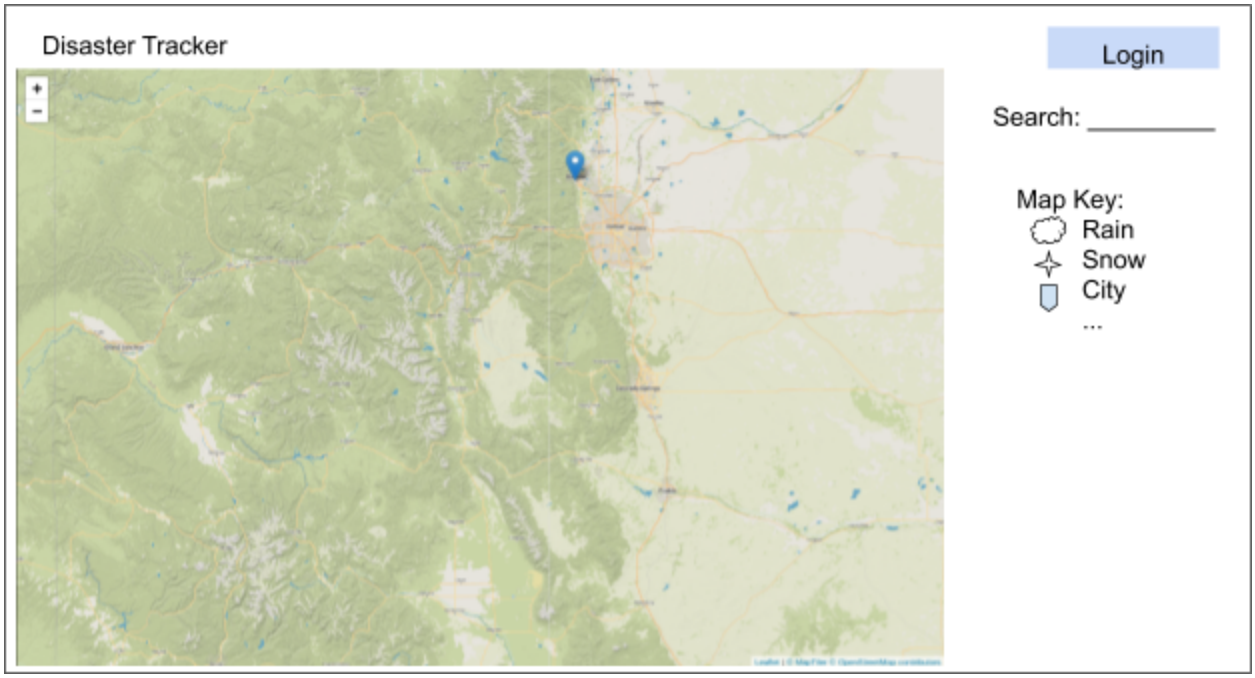
Architecture Diagram:



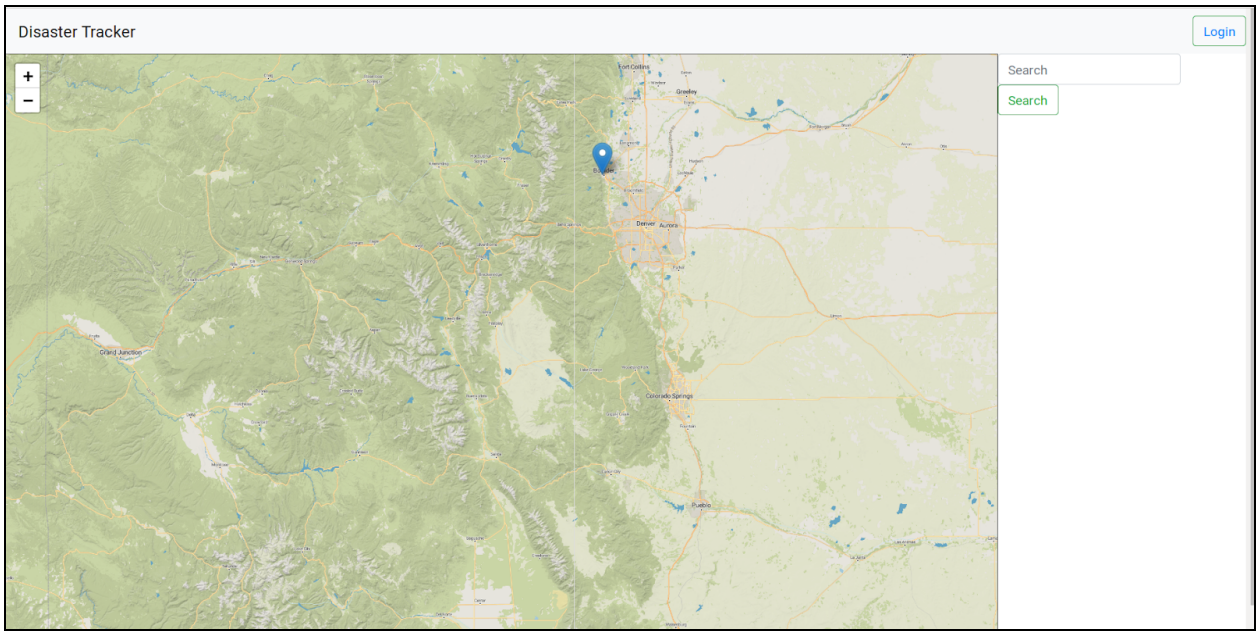
Front End Design:

- Home page：search bar/ log in bar/ map
- User profile page: user information
    - Updates from initial idea - added more fields (like location, phone number, and email)
    - Just made it a sign up sheet for the notifications, may add login modal later...

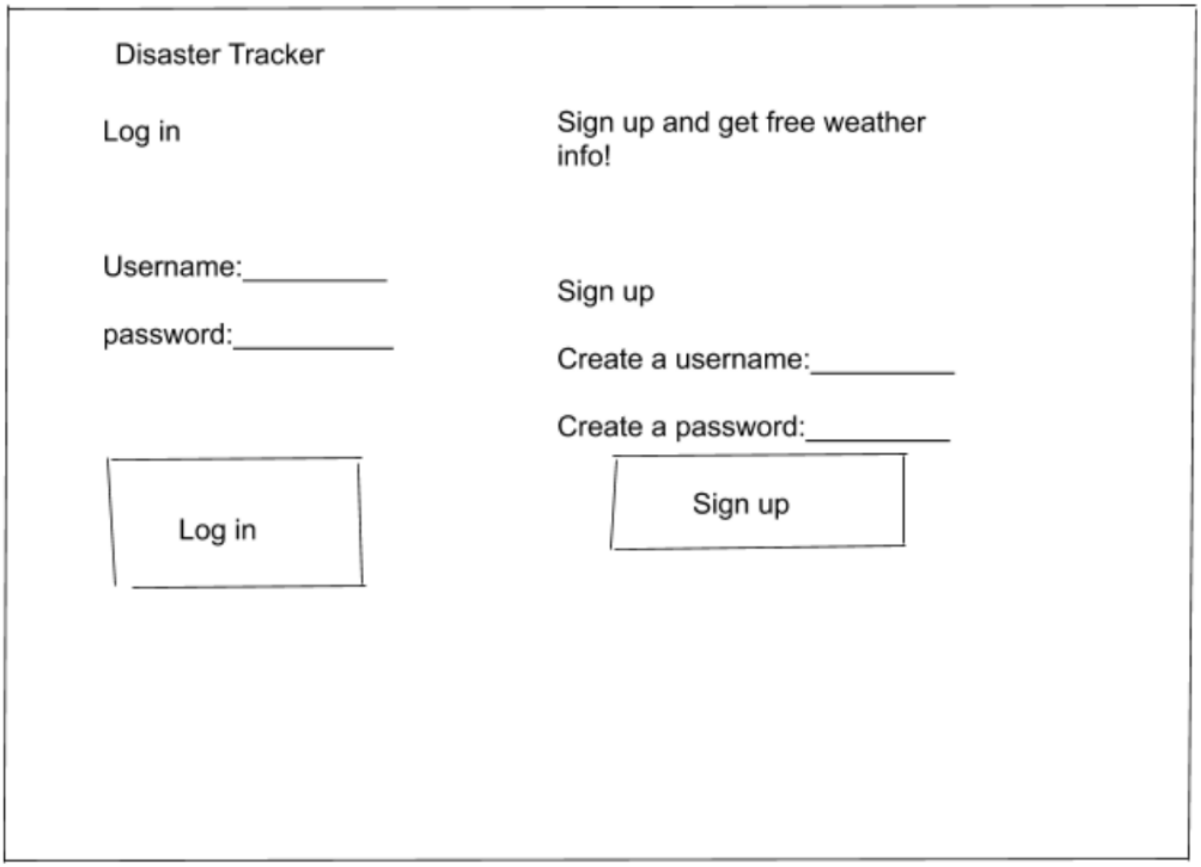
Home page -



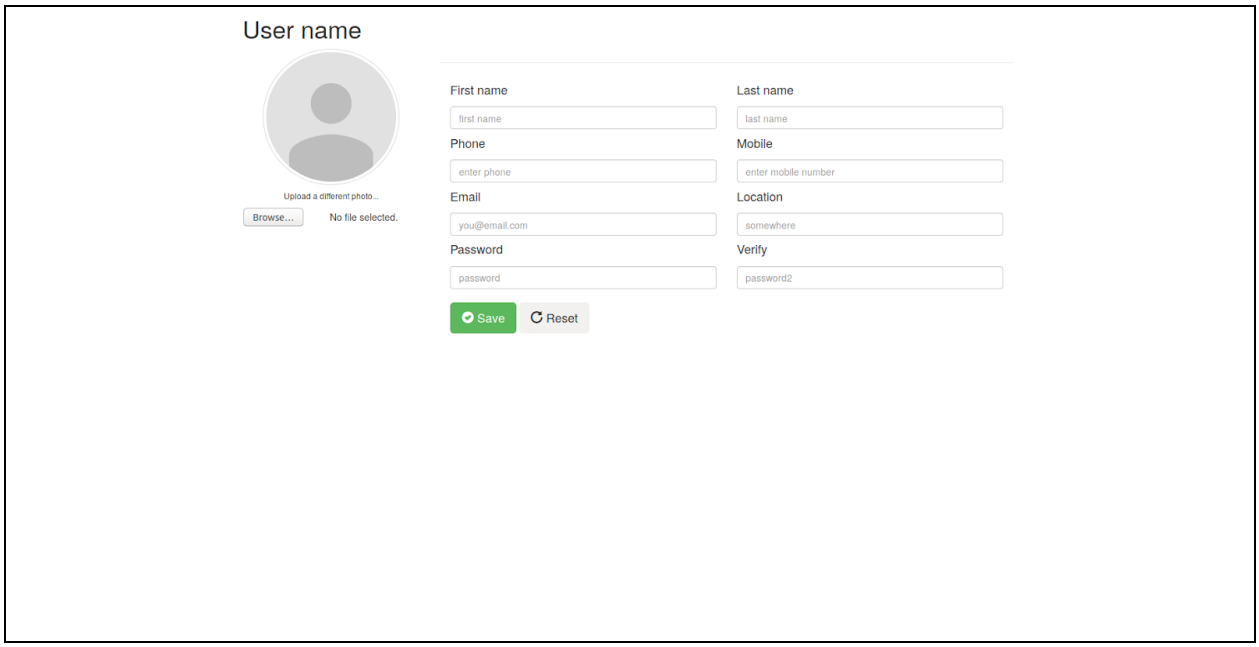

Current:




Login Page -


Disaster Tracker
Log in
Username:_____
password:_____
Log in
Sign up and get free weather info!
Sign up
Create a username:_____
Create a password:_____
Sign up

Current:

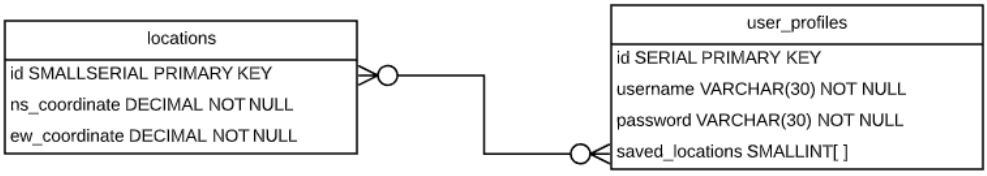

User name
Upload a different photo...
Browse...
No file selected.
First name
last name
Last name
last name
Phone
enter phone
Mobile
enter mobile number
Email
you@email.com
Location
somewhere
Password
password
Verify
password2
Save
Reset

Web Service Design:

- APIS:
    - Weather.gov Weather API
        - Through input of latitude and longitude(via user input or function call), the API returns a parsable JSON file containing weather information from the nearest weather center.
    - Leaflet (Map API)
        - A Javascript based API allowing for the creation of a dynamic map component. Using online MapTiles, the API creates a displayable map with interactivity through the creation of markers and icons. Marker creation/interactivity can be created using specific latitude and longitude values within the javascript file. After parsing information from the aforementioned weather API, markers can be populated and styled using the parsed information, and then placed on the MapTile. This will allow for real time updating of the markers and map through the Weather API.

Database :

For our database, we will be using PostgreSQL. We have two tables, locations, and user_profile which are linked by locations. Each location has an id in the locations table and the user_profile table has a list of saved_locations that contains the corresponding location ids.




TA Demo Notes:

Features we showed in the demo:

Home page: (**Worked in demo**)

- The home page shows the map (from the map api).
- Search bar to look for specific locations
- Button to get user get to profile page

User profile page: (**Worked in demo**)

- Show a profile picture
- Upload new pictures
- Type in user information

Map API: (**Worked in demo**)

- Leaflet API
- Markers created (latitude, longitude), containing fields for populating weather info
- Map itself- features such as zooming in/out, and bounds for zoom out.

Weather API: (**Talk about, but not shown in demo,** but it does work)

- The weather data comes from weather.gov. It takes a latitude and longitude and returns information for what the weather is like near that area.

Database (locations and user profile data): (**Worked in demo**)

- Locations:
    - The locations (latitude, longitude) of 455 cities in Colorado come from simplemaps.com
    - The locations (latitude, longitude) of 58 14ers in Colorado come from kaggle.
    - Extracting the locations from csv files by using pandas data frame in python.
- Profile:
    - Holds locations that the users saved on the profile page
    - Also holds username and password
    - All information is populated by the user profile page

Issues during development/demo:

There is no make file for us all to get copies for the database locally.
Took a while to pick our APIs, but now we have a good setup.
We have not connected the database to the frontend yet.
We initially wanted to host the project on git, but it only lets you host static pages and we need something with a database so we will need to pick something else.
We don't have input checking yet so a user could put an invalid input in and it would fail.

Things to think about after the demo:

What if someone tries to find a location that isn't in our database?
What are we doing for the middle layer?
Where will this be hosted? Must be on a cloud server

Suggestions by TA:

Documentation of ‘Create’ statements for database- enables for easier local testing
Hosting app on heroku instead of git- allows us to host database and handle server calls, while also keeping real time updates through git
Figure out a way to handle misspelled/mistyped input- handle these user errors through a function