SW Engineering CSC648/848 Spring 2023

CrisisConnect

Team Details:

Class Section: 2
Team: 4

Contributors:

Team Lead: Mo Moses (mmoses3@mail.sfsu.edu)
Frontend Lead: Marc Castro
Backend Lead: Axel Biehler
FrontEnd Dev: Nicholas Chan
FrontEnd Dev: Virel Patel
BackEnd Dev: Raquel Lutges
BackEnd Dev: Lokesh Telaprolu

Milestone 2: March 14th, 2023

Revisions	
3/14/2023	Initial Submission

1. Functional Requirements - prioritized

1.1 Must have

Users

- Users shall be able to search
- Users shall have the ability to see a live feed of current and past incidents.
- Users shall have access to a dashboard that has real-time data in their area (weather, air-quality, etc.).
- Users shall have access to information and resources in case of an emergency.
- Users shall be able to look at a map interface and see current events.
- Users shall have the ability to filter out certain categories to only display certain metrics.
- Users shall be able to search county status.
- Users shall have access to registration capabilities.

Registered Users

- Registered users shall have the ability to filter their notification settings for what type of alerts they want to receive.
- Registered users shall have the ability to report incidents or suspicious activity to admins.
- Registered users shall have access to alerts sent by admins regarding current concerns regarding the admin's area and department (weather and fire, health, security).

Admin Users

- Admin users shall have the privilege of inputting data via an interface only accessible to them.
- Admin users shall have the privilege to trigger alerts to users.
- Admin users shall have access to data to update the data they previously inputted for areas of concern.

1.2 Desired

Registered Users

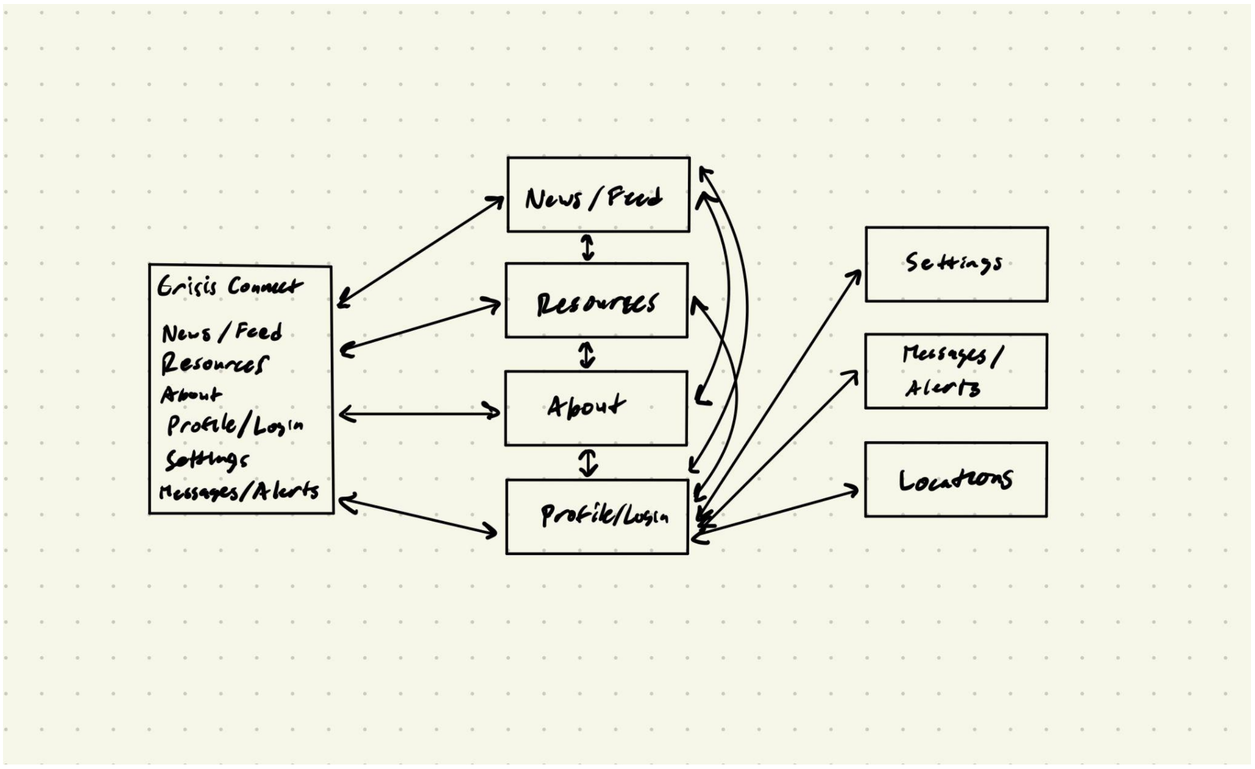
- Registered users shall be able to upload pictures/video of an incident.
- Registered users shall be able to add friends and family to monitor/track their loved ones.
- Registered users shall be able to form community groups for added public safety (i.e., neighborhood watch groups).
- Registered users shall be able to send messages to other registered users to alert them on possible dangers.
- Registered users shall have capabilities to input their current residence to search via their area.
- Registered users shall have capabilities to keep a list of areas they would like to keep an eye on.

1.3 Opportunistic

Registered Users

- Registered users shall be able to provide feedback to make the app better.

2. UI Mockups and Storyboards

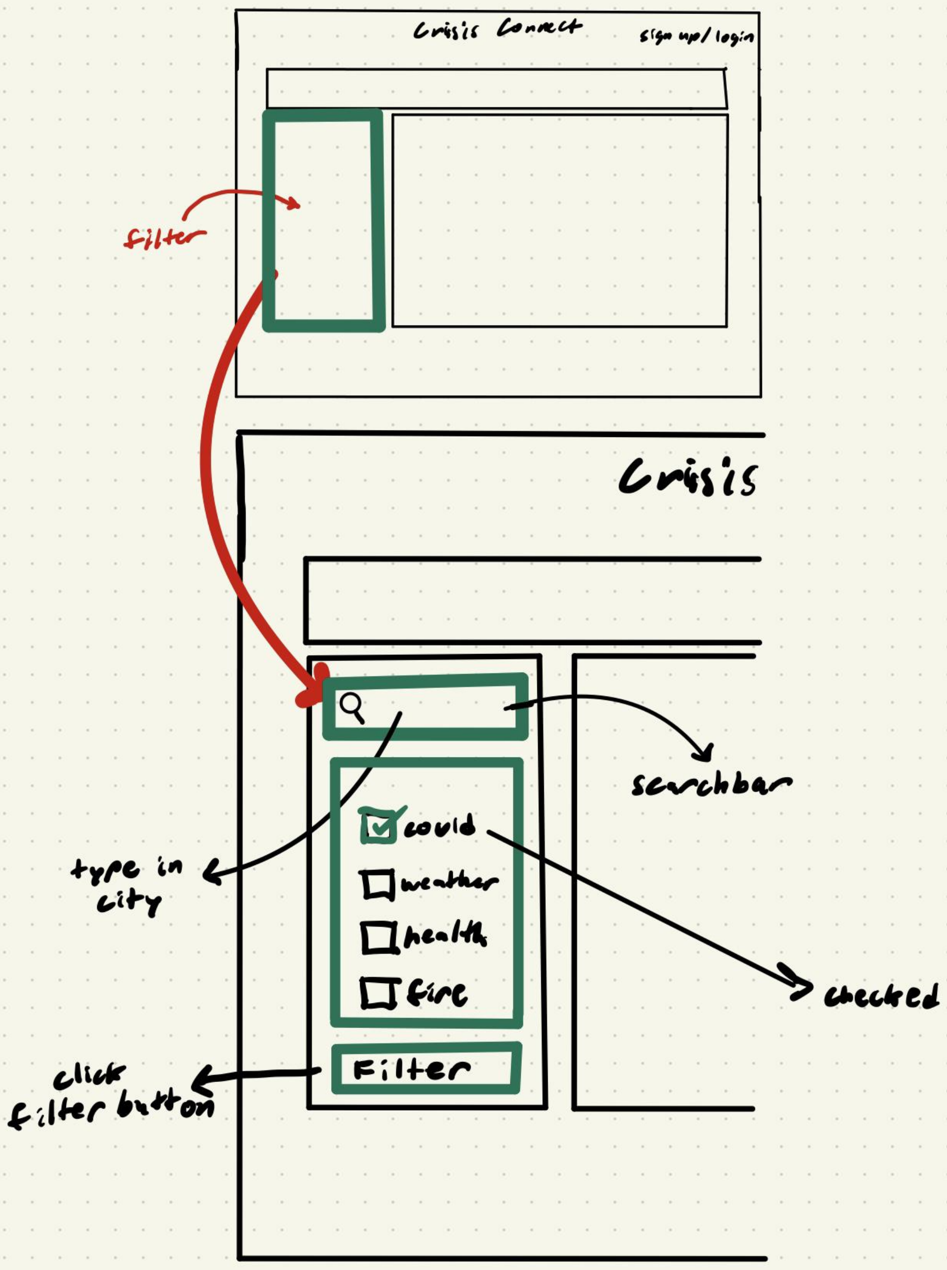


2.1 City Search

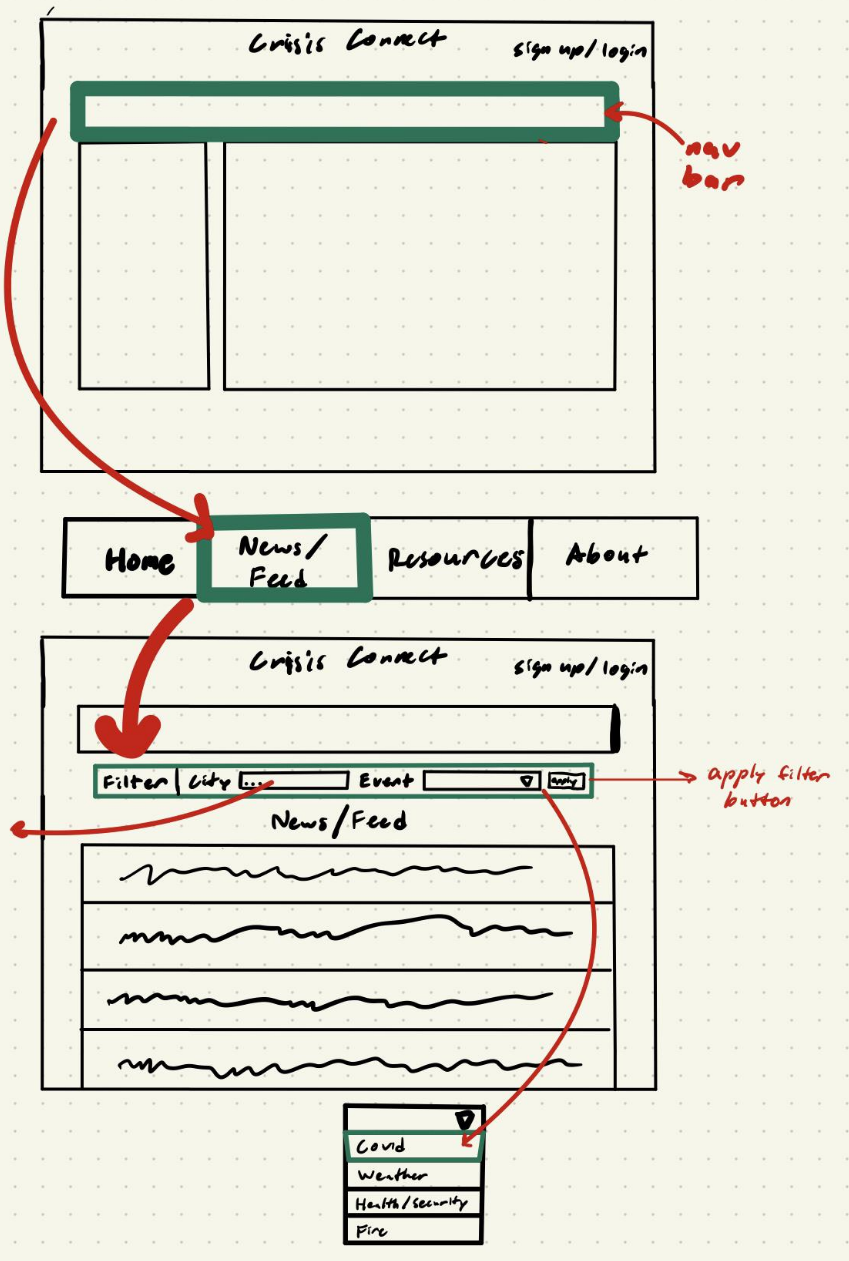
Use case: Joseph is a college student who just moved out to live on campus at his university. Joseph wants to look up the Covid rates in his university's city. Joseph has two options to look for information on his city regarding Covid cases. In his first option, if he wanted to see Covid rates in his city on the map, Joseph would find the filter box on the left hand side of the home page and would look at the top of it to find the search bar. In the search bar, he would type in the city that his university is in, check the "Covid" checkbox, and press the "filter" button. The map shall update to hover display his city and area surrounding it, allowing him to see covid cases around and in his city. In his second option, if Joseph would like to read updates, he would navigate to the "News/Feed" option in the navigation bar. Upon clicking it, Joseph would be brought to the "News/Feed" section of the website and be presented with a filter that Joseph can use to type in his city, select "Covid" in the drop down menu for "Events", and press the "apply" button. This will display all the news updates regarding Covid cases in a readable format.

UI Mockup → Use Case 1: Viewing Data

Option 1

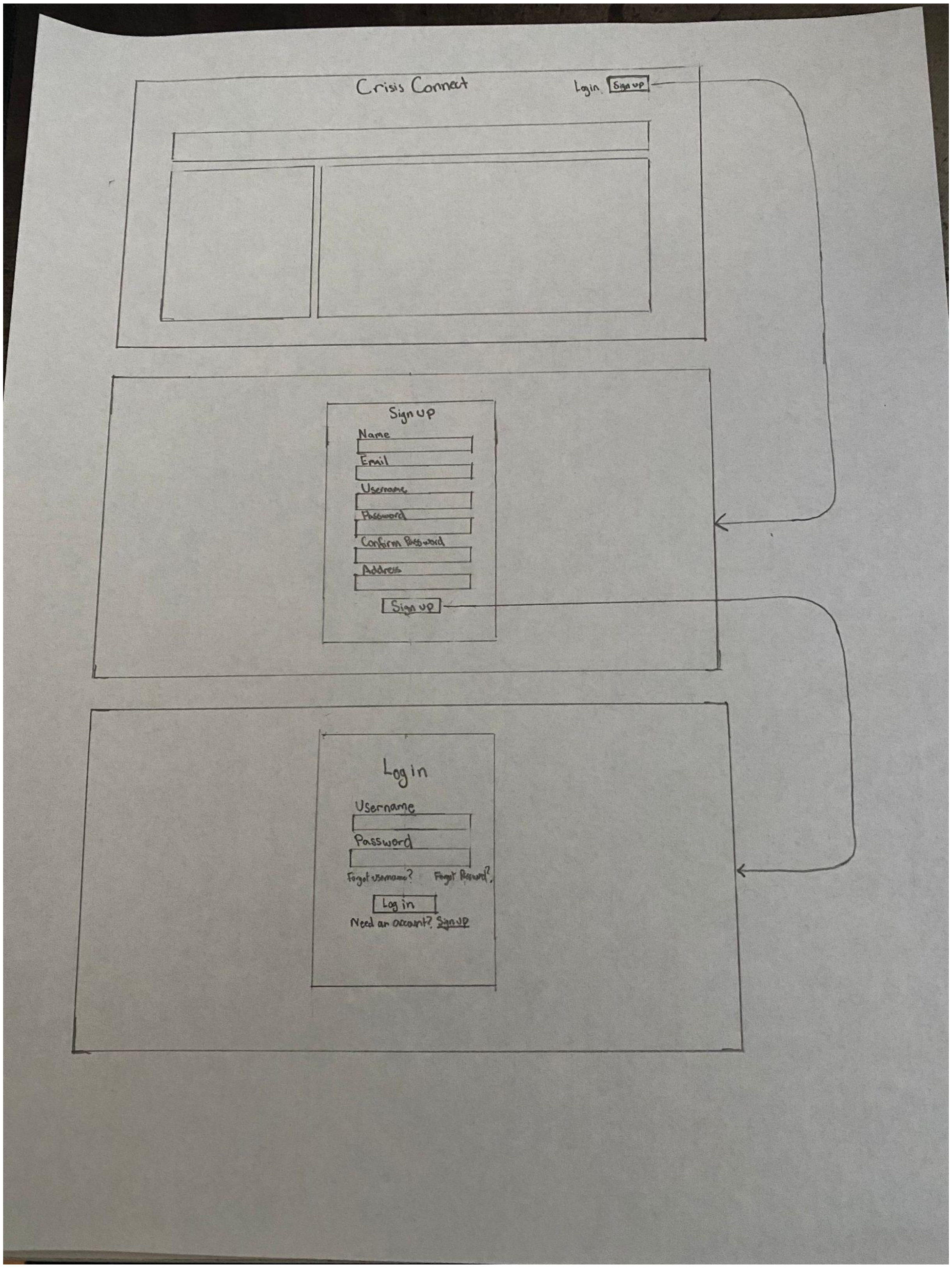


Option 2


Crisis Connect
sign up/login
nav bar
Home
News/Feed
Resources
About
Crisis Connect
sign up/login
Filter
City
...
Event
apply
apply filter button
type in city
News/Feed
Covid
Weather
Health/Security
Fire

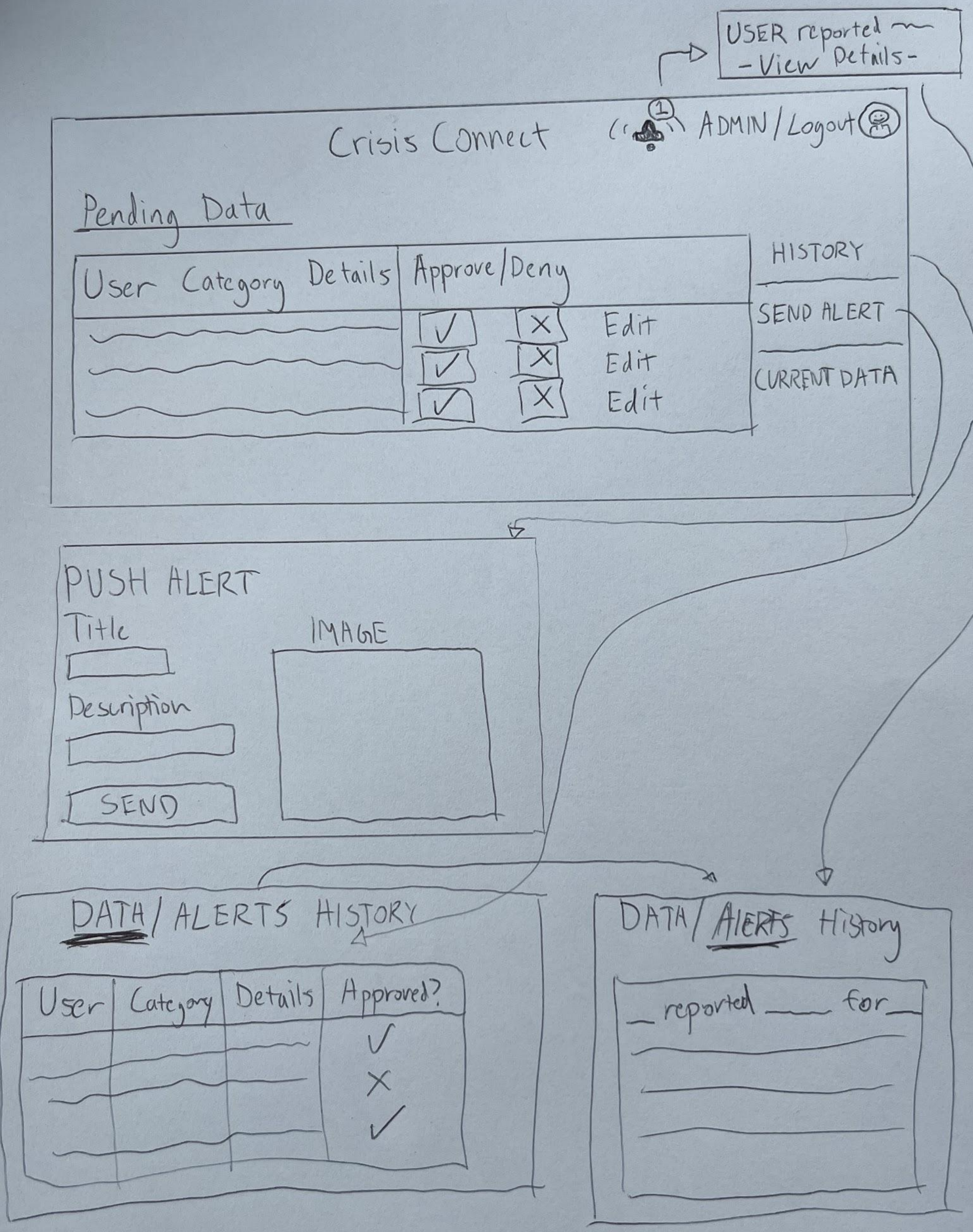
2.2 Sign up / Login

Use case: Mary is an old woman who lives in a very dangerous/hazardous area. Mary decides to register for CrisisConnect so that she can receive alerts regarding safety hazards. First, when she goes onto the CrisisConnect page she clicks on the “Sign Up” button in the top right corner. This takes her to the sign up page where she enters her Name, Email, Username, Password, and her address. By clicking the Sign Up button, her account is created and she is automatically taken to the Login page where she can sign in with her new account. From here, all she has to do in order to log in is enter her username and password and click the Login button.


Crisis Connect
Login
Sign up
Sign up
Name
Email
Username
Password
Confirm Password
Address
Sign up
Log in
Username
Password
Forgot username?
Forgot Password?
Log in
Need an account? Sign up

2.3 Administrative Tools

Use Case: Nicholas is an administrator that needs to review the data that is sent in by County Directors and ensure that it is categorized and inputted correctly. Once he logs into his account, he is automatically taken to the administrative tools page where he can approve/deny/edit data that has been submitted by users/County Directors. On the sidebar, he can click on a link that will direct him to the website's history of data and alerts. This defaults to a screen that has a history of approved/denied data that has been submitted but by clicking Alerts, he will be taken to a screen that shows all the current and past alerts in which he can investigate. Back to the administrative tools page, the top right corner has a button that indicates how many notifications he has and clicking on it will show them. By clicking "View Details" on a notification, it will take him to the previously mentioned "Alerts" page. Back to the administrative tools page, the sidebar has a button called "Send Alert" that will take him to a page where he can write an alert to push to all registered users. In this alert he can add a title, description, and image.


USER reported ~
- View Details -
1
Crisis Connect
ADMIN / Logout
Pending Data
User Category Details
Approve/Deny
✓ ✗ Edit
✓ ✗ Edit
✓ ✗ Edit
HISTORY
SEND ALERT
CURRENT DATA
PUSH ALERT
Title
Description
SEND
IMAGE
DATA / ALERTS HISTORY
User
Category
Details
Approved?
✓
✗
✓
DATA / ALERTS History
_ reported ___ for _

3. High level Architecture, Database Organization

3.1 DB organization

```
CREATE TABLE User (
        User_id INT (PK)
        User_name VARCHAR(50)
        User_county VARCHAR(50) (FK REFERENCES County),
        Director_id INT (FK REFERENCES User(user_id))
        Director_name VARCHAR(50) INT (FK REFERENCES User(user_name)
        Admin_id INT (FK REFERENCES User(user_id))
        User_email VARCHAR(50)
        Department_id INT UNIQUE (FK REFERENCES)
        Department(department_id)
 );

CREATE TABLE Events (
        Event_id INT (PK)
        Department_id (FK REFERENCES Department(department_id)
        Event_type VARCHAR(50) NOT NULL
        Event_date VARCHAR(50) NOT NULL
        Start_time DATETIME NOT NULL
        Severity_level VARCHAR(50)
);

CREATE TABLE Department (
        Department_id INT (PK)
        Department_name VARCHAR(50)
        County_id INT (FK)
        County_name (FK)
        Director_id INT
        Director_name VARCHAR(50)
);

CREATE TABLE County (
        County_id  INT (PK)
        County_name VARCHAR(50)
);
```

```
CREATE TABLE  Alert (
        Alert_id INT (PK)
        User_id INT (FK)
        Event_type VARCHAR(50)
        Wildfire_id INT (FK)
        Weather_id INT (FK)
        Covid_id INT (FK)
        County_id (FK)
        Severity_level CHAR
        Alert_message VARCHAR(50)
        Alert_date DATETIME
);

CREATE TABLE  Covid_Event (
        Covid_event_id INT (PK)
        Department_id INT (FK)
        Shelter_in_place (FK REFERENCES event_type)
        Covid_cases INT
        Covid_deaths INT
        Date_time DATETIME
);

CREATE TABLE Weather_Event (
        Weather_id INT (PK)
        Department_id INT (FK)
        Blocked_roads VARCHAR(50) (FK to event_type)
        Temperature INT
        Wind_speed INT
        Precipitation INT
        Uv_index INT
);

CREATE TABLE Wildfire_Event (
        Wildfire_event_id INT (PK)
        Department_id INT
        Wildfire_evac_level INT (FK to severity level)
        Wildfire_count INT
):
```

```
CREATE TABLE  Security_Event (
        Security_event_id INT (PK)
        Department_id INT
        Reported_incident (FK to event type)
        Date_time DATETIME
);
```

3.2 Media storage

Videos and images will be kept in the file system, in a folder by a special API route to upload them.

3.3 Search algorithm

For the search algorithm we will use the alert_message field to make a search by text input and the second parameters will be the type of the Alerts using a filter with an enumeration.

3.4 API

/auth/register - POST

User can create his account on the application
Body example:

```
{
   "username": "Johndoe",
   "password": "dummy_password",
   "name": "John Doe",
   "email": "johndoe@gmail.com",
   "2378 41st Ave, San Francisco, CA 94116"
}
```

Response:

```
{
  "status": true,
  "data": {
     "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9",
     "Role": "admin",
  }
}
```

/auth/login - POST

User enter username and password to get the bearer token to be authenticated
Body example:

```
{
   "password": "dummy_password",
   "email": "johndoe@gmail.com",
}
```

Response:

```
{
  "status": true,
  "data": {
     "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9",
     "Role": "admin",
  }
}
```

/alerts?search=${params}&type=${params2} - GET

Query to retrieve all the alerts with 2 query params for the search and filter by type
Response:

```
{
  "status": true,
  "data": [
    {
       ...objectValue,
    },
    {
       ...objectValue,
    },
  ],
```

/alerts - POST

Add an alerts on the database
Body example:

```
{
  ...objectValue,
}
```

/alerts/:id - DELETE

Delete an alert by id

/media - POST

Add images/videos/audios
Body example:

```
{
   "data": "image/audio/videos",
}
```

header:

```
{
   "Content-Type": "multipart/form-data"
}
```

/media/:id - GET

Get the image/video/audio

/media/:id - DELETE

Delete the image/video/audio
Response:

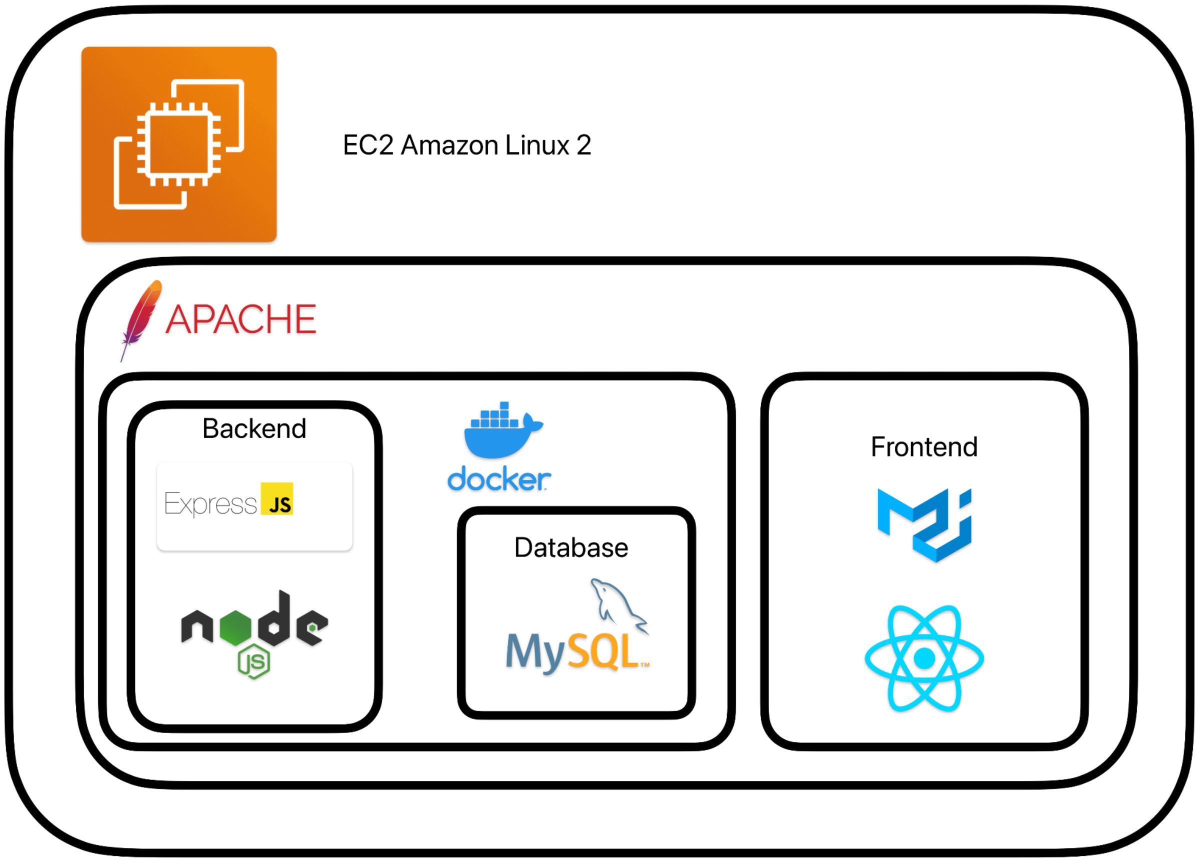
```
{
   "status": true
}
```

Some routes need for the user to be authenticated using a Bearer token header example:

```
"Header": {
  "Accept": "application/json",
  "Authorization": "Bearer ${token}"
}
```
Error response format:
```
 {
   "status": false,
   "error": "Error message"
}
```

These API routes may change in the future depending on project needs.

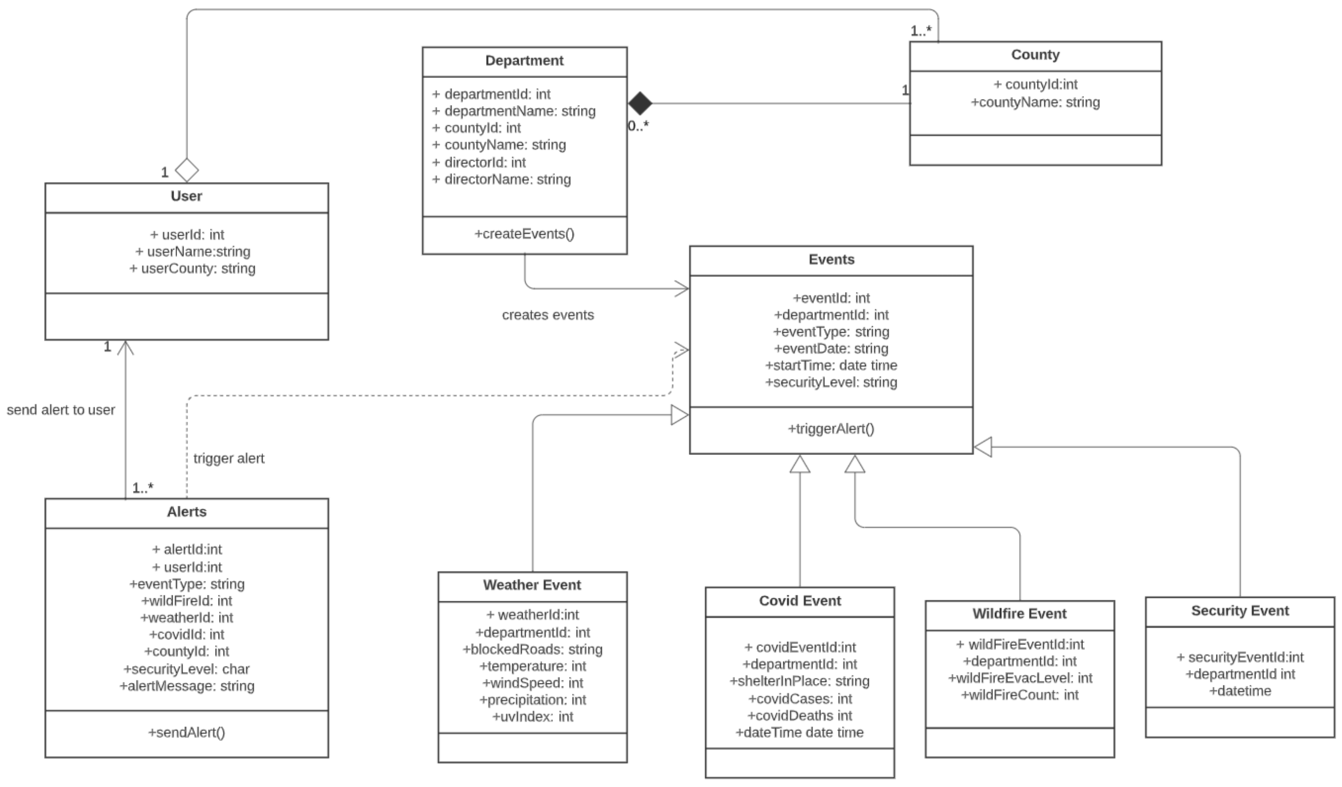
3.5 Software tools and framework



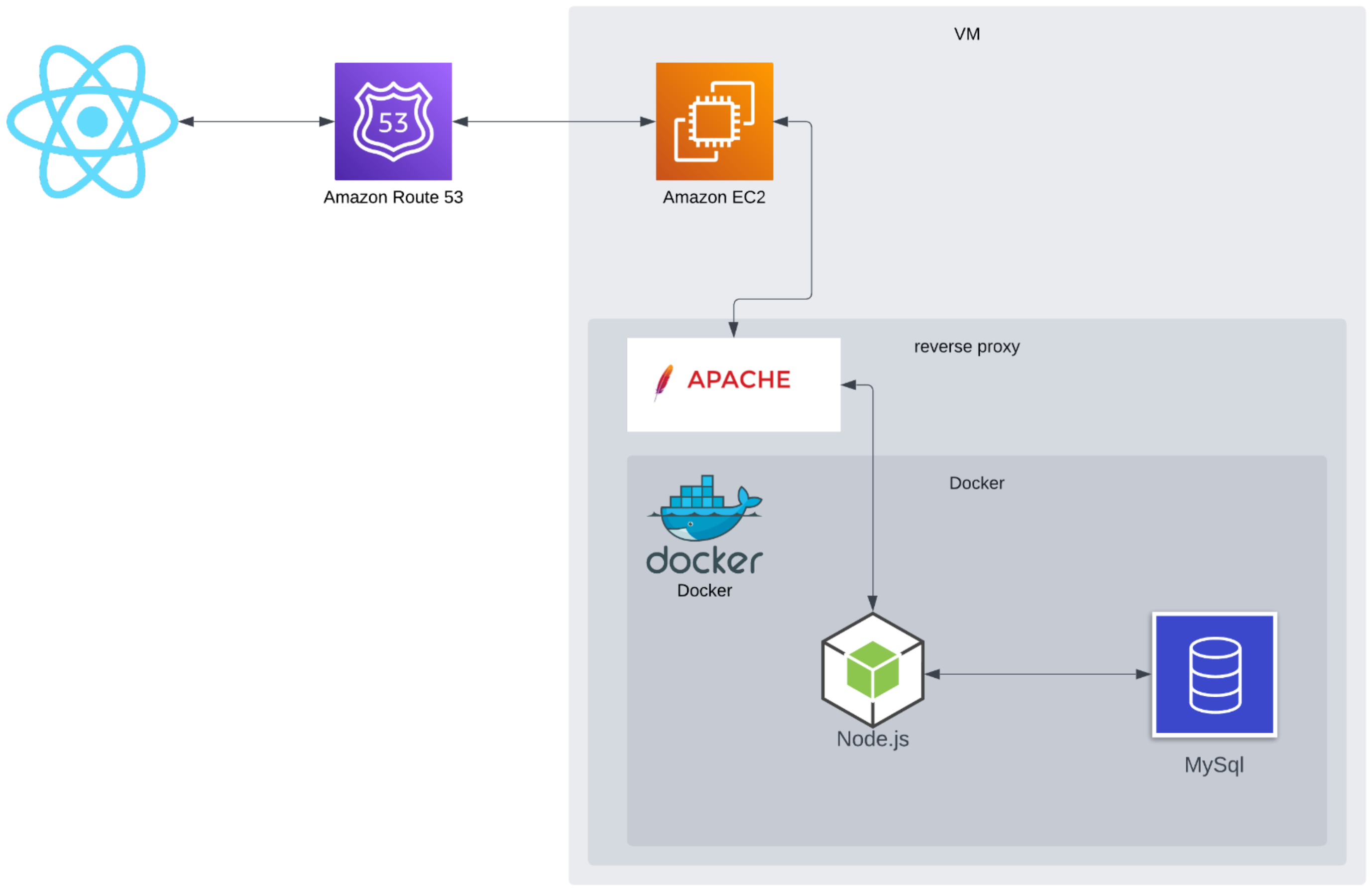
The main change in infrastructure is the node js application inside docker now because it makes sense for deployment which is easier and also the communication with the database inside docker network.

4. Database UML and deployment

4.1 UML



4.2 Deployment



5. Key risks of the project

5.1 Skills Risks

Our stack is, for the most part, well supported by our skill set as we have team members that have good familiarity with all of the components that make it up. The risk is that the level is unbalanced and thus could result in heroic actions to attempt to make our product work where distribution of responsibilities would allow all to contribute and none to burn out. A good solution here would be to encourage a culture of knowledge sharing between the more knowledgeable team members in their domains of understanding. We will have SME (subject matter experts) but our culture should be that the SMEs share and guide where possible as opposed to just do it themselves. This will allow the work to be even distributed and no one gets burned out.

5.2 Schedule Risks

Availability will be a risk in timely delivery simply because we all have other obligations to attend to outside of this project, as well as, the school as a whole (i.e. work, family, etc.). The risk, of course, when you distribute the work, is if one person can't get it completed in a timely fashion due to external factors, it can side track the process. The answer to this is also sharing. All members of - at least the sub section - should have an understanding of the work each person is doing and should be able to receive parts when one person doesn't have the time to complete. The addition to this is that project work should be broken up into small enough and clear enough segments that pieces can be handed off to another person to continue to work.

5.3 Technical Risks

The internet is a wicked and wild place. Service stability will always be an unknown to some degree since we can only control some aspects of the total chain of systems that deliver the product to an end user. Data integrity is a key risk since we're storing data in a RDBMS. The possibility of accidental mass data changes (someone forgot to use a 'where' clause) are pretty high in these systems as SQL especially when delivered from command line, doesn't do - "are you sure you want to make this change" checks. As much as possible it would be imperative we make tools that allow us to visualize and manipulate database data in a controlled way. With simple scripts or even stored procedures, we can program in protections to ensure critical data isn’t incorrectly manipulated or corrupted.

5.4 Teamwork Risks

So far we have worked together well. The team is very cordial and helpful. The key risk in this space is failing to communicate at a reasonable interval. I think the solution here is to identify a time to report on progress. We’ve also set up a means to communicate asynchronously which allows people to request help or feedback pretty easily and the team has shown excellent responsiveness there. This communication will help even in repository issues as well.

5.5 Legal/Content Risks

Looking good isn’t easy especially if you’re trying to get a great icon and images to give some flavor to the design. It is in this space that I believe we could incur the biggest risk. I think the solution is to be very attentive to the source of the things we decorate our page with and ensure that we have legal rights to use it.

6. Project management

Our team operates in 2 teams of 3 persons with lead and devs filling the roles for frontend and back end teams and the lead who should float and serve where can be of best use in the forwarding of the project as well as ensuring all deliverables are complete prior to release/submission for review. We initially divide the tasks between front end and back end teams and deliver the tasks to the section leads. It’s up to the frontend leads in negotiation with their sections to decide who will be responsible for each task assigned. The role of the lead, then, is to ensure the parts come together and the final project is cohesive. Going forward this approach seems to be optimal with the addition of using some Project Management tooling. In our case we have decided to settle on managing project work through Trello. In summary, our team uses a divide and conquer approach and our efficiency will be improved as we lean into using Trello to track and report on the work being done.