SW Engineering CSC648/848 Fall 2023 Section 2 M4

“CalAwareNow - Digital Shield for California”

Team Lead / Backend Support: Ameen Safi (asafi1@mail.sfsu.edu)

Github Administrator/Frontend Support: Dev Patel

Backend Lead: Himal Shrestha

Frontend Lead: Chih Lin Chien (Brian)

Database Lead: Fasika Abera

Backend Lead: Trina Haque

11/7/2023

1. Product summary - Dev

Product name: **CalAwareNow**

CalAwareNow is a powerful online platform that acts as a protective shield for Californians. It gives them the latest updates on COVID-19, natural disasters, and security threats all in real-time, ensuring they stay informed and safe wherever they are.

Our Commitment to Functionality

- User Registration: Users can sign up with email verification for secure access.
- User Login : Existing users can log in using their username and password
- User Profiles: Allowing users to customize profiles and notification settings.
- County Data Entry: County department directors can input COVID-19, wildfire, security, and weather metrics
- Alerting System: Immediate notifications based on state guidelines and user preferences.
- Alert Preferences: Users can set preferences for how and what alerts they receive.
- Search Functionality: Users can use the search functionality to search for incidents.
- Filter: Users can use the filter to find county-specific information
- Messaging: Registered users can comment under a particular post, which can be responded to by who made the post or others.

Deployment URL: http://34.66.193.152

2. Usability Test Plan for CalAwareNow - Version 1 - Himal

1) Test Objectives:

The test aims to evaluate the usability of the "Search" function on our website. Specifically, we are testing this function to assess how effectively users can get information about natural disasters in California. The primary objective is to measure user satisfaction with the search functionality.

2) Test Background and Setup:

- **System Setup:** The test will be conducted on a laptop or ipad using the latest version of a popular web browser. The website will be in its development environment.
- **Starting Point:** Participants will begin at the website's homepage.
- **Intended Users:** The intended users are individuals residing in or interested in California or visiting California who want to access information about natural disasters.
- **URL:** http://34.66.193.152
- **Measurement:** User satisfaction will be the primary focus of the evaluation. We will use a Likert scale questionnaire to assess user satisfaction with the search experience.

3) Usability Task Description:

Participants will be instructed to perform the following task: Imagine you are concerned about the current status of various natural disasters in California, including COVID-19, rain, weather conditions, wildfires, and earthquakes. Please use the search bar on the website to find the most recent information about these natural disasters in California. After completing the task, you will be asked to assess your satisfaction with the search experience using the provided Likert questionnaire.

- **Measuring Effectiveness:** Effectiveness will be measured based on the participant's ability to successfully find information about earthquakes in California using the search functionality. Successful task completion indicates high effectiveness.
- **Measuring Efficiency:** Efficiency will be measured by the time taken by participants to complete the task. A shorter time to complete the task indicates higher efficiency.

4) Likert Subjective Test:

Participants will be asked to rate their satisfaction using a 5-point Likert scale for the following three questions:

1. "How satisfied are you with the search functionality's ability to help you find natural disaster information in California?"

- ☐ Very Dissatisfied
- ☐ Dissatisfied
- ☐ Neutral
- ☐ Satisfied
- ☐ Very Satisfied

2. "How easy was it to locate the information about a natural disaster in California using the search?"

- ☐ Very Difficult
- ☐ Difficult
- ☐ Neutral

- ☐ Easy
- ☐ Very Easy

3. "Overall, how satisfied are you with your experience using the website's search function?"

- ☐ Very Dissatisfied
- ☐ Dissatisfied
- ☐ Neutral
- ☐ Satisfied
- ☐ Very Satisfied

3. Test Plan for CalAwareNow - Version 1 - Fasika

Objective

The objective of this test plan is to test the functionality of CalAwareNow version 1 final commitment for product features (p1 list).

Features to be Tested

As part of the version 1 commitment, we are testing priority one functionalities. These include user registration/login, user profile, incident data entry, alerting, search/filter, and messaging functionalities.

No.	Category	Description	Test Input	Expected	Pass /

				Output	**Fail**
1	User Registration	New users can register by providing their full name, unique username, valid email address, password, and user type	- Enter full name - Enter a unique username - Enter valid email - Enter password - Select from the user type	The user is able to successfully register	Pass
2	Log In	Existing users can log in using their username and password	Enter valid credentials	The user can successfully login	Pass
3	User profile	Registered users can create and manage their profile, including contact information and notification preferences	- Login - Navigate to Profile	The user is able to view/edit their information and notification preferences	Not tested (not implemented yet)
4	County Data Entry	Admins and county department directors	Enter report metrics in the	The report is successfully	Pass

		can input COVID-19, wildfire, security, and weather metrics	dashboard	added	
5	Alerting System	Registered users can receive alerts based on state guidelines and user preferences.	- As Admin/county director, navigate to the dashboard - upload metrics with high severity	- Alert for the high severity disaster will be created and shown in the notification bell	Not Tested (not implemented yet)
6	Alert Preference	Registered users can set their alert preference	- Registered users can set their notification preference in their profile	-Based on the notification preference, users will get alert for related disaster and/or county	Not Tested (not implemented yet)
7	Search	Users can use the search functionality to search for incidents	Enter “covid” in the search bar	Search results containing county name,	Pass

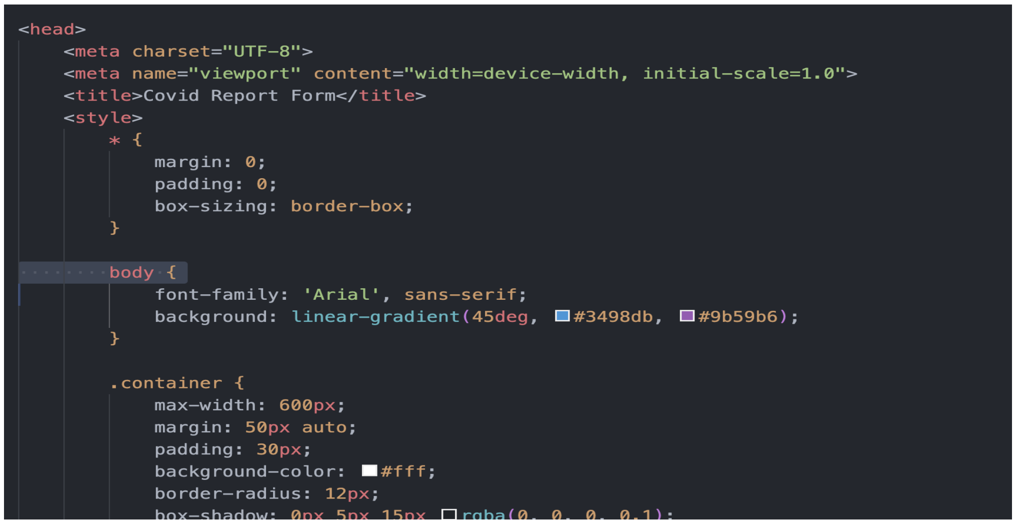
				the disaster type, recommended action, as well as information about the county, will appear on the screen	
8	Filter	Users can use the filter to find county-specific information	From the dropdown, select Alameda	Results with all disaster types that are recorded in Alameda county will show	Pass
9	Messaging	Registered users can comment under a particular post	Enter comment on the text field under the post	Any user will be able to view comment and reply to it	Not Tested (not implemen ted yet)

4. Code review - Brian

a) Our team chose the **Google HTML/CSS Style Guide** for its established best practices and industry standards. It ensures code consistency, readability, and maintainability, ultimately enhancing collaboration among team members. Google's expertise and widespread use of HTML/CSS made this style guide a trusted and logical choice for our development team.

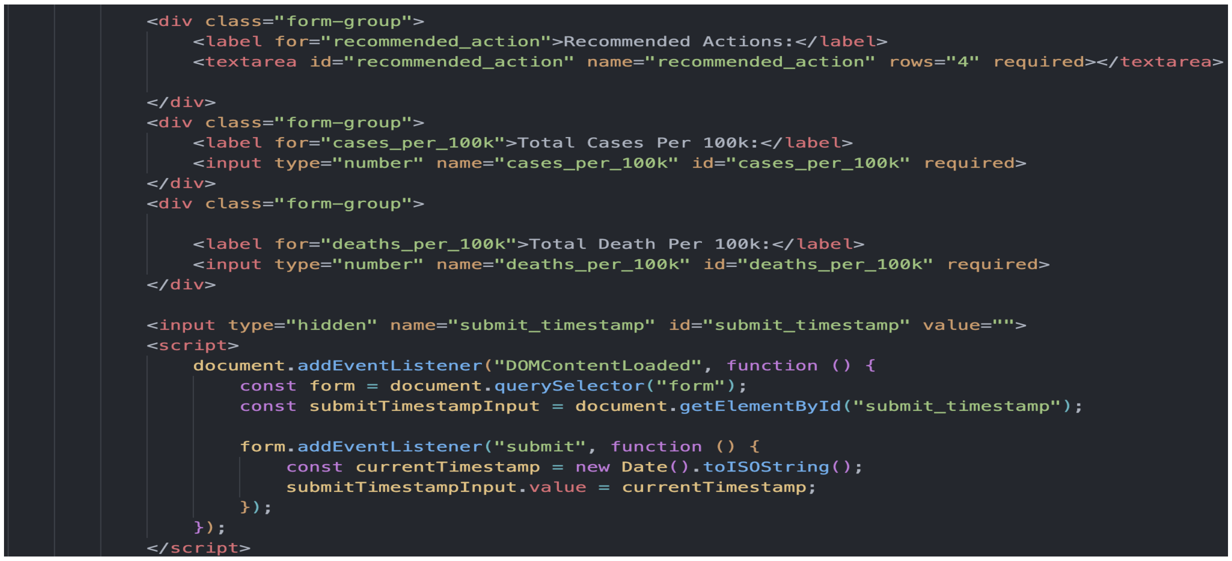
b) The codes I chose are the upload covid form feature and the backend.

Head



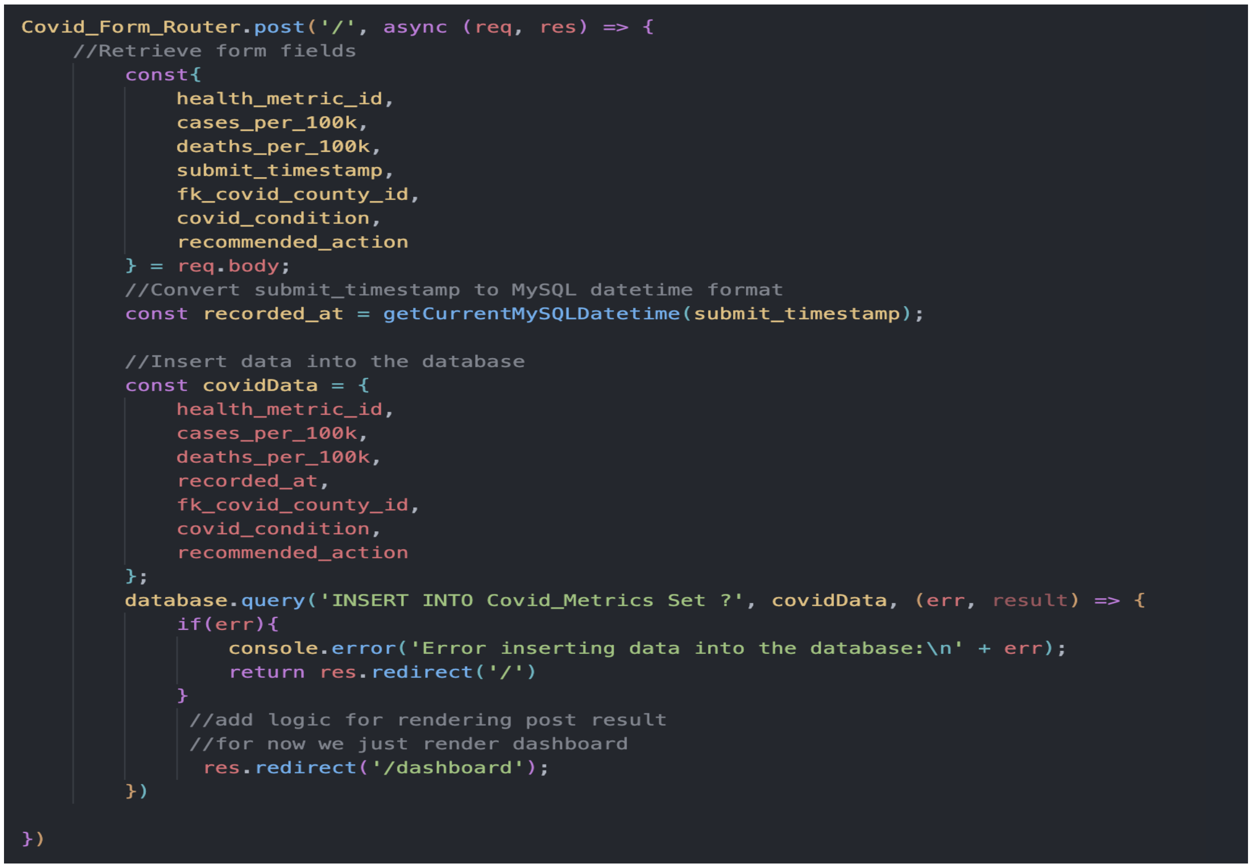
```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Covid Report Form</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Arial', sans-serif;
            background: linear-gradient(45deg, #3498db, #9b59b6);
        }

        .container {
            max-width: 600px;
            margin: 50px auto;
            padding: 30px;
            background-color: #fff;
            border-radius: 12px;
            box-shadow: 0px 5px 15px rgba(0, 0, 0, 0.1);
```

Covid Form Function



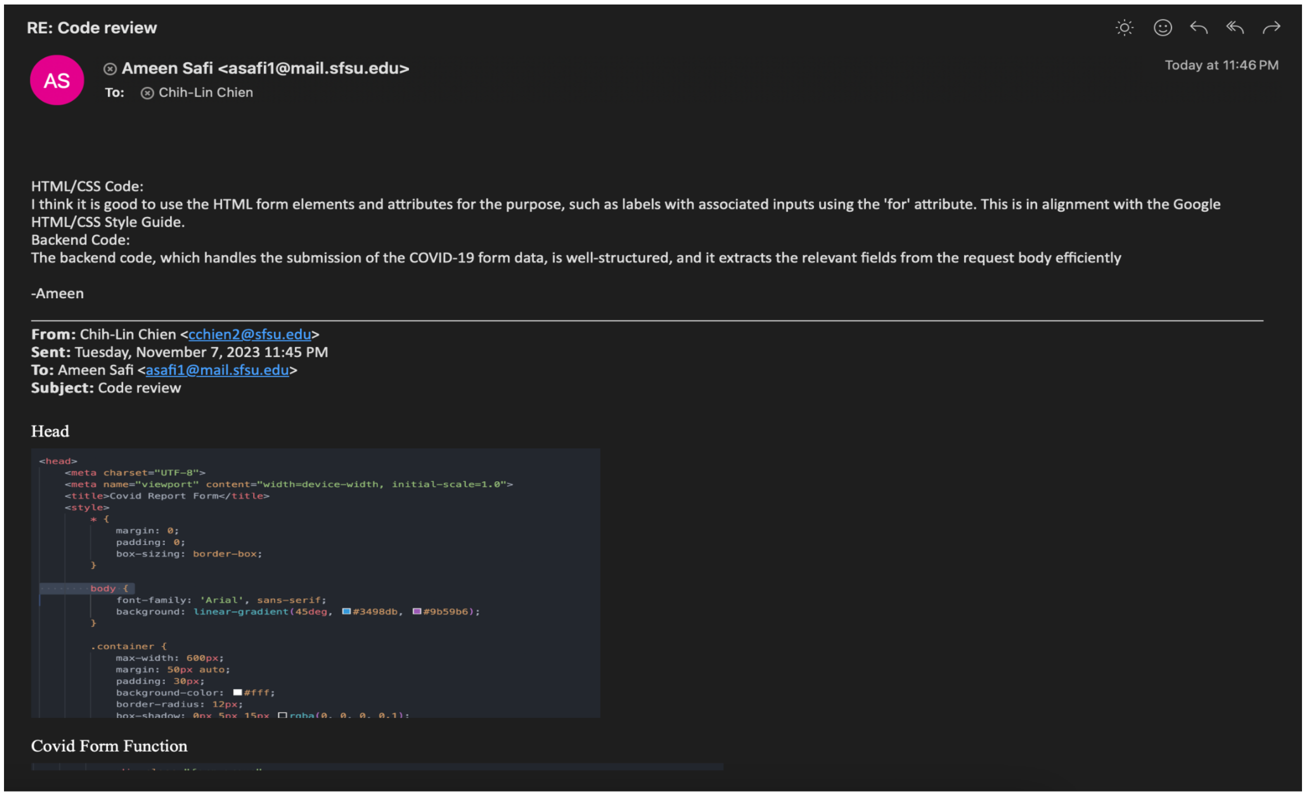
```
            <div class="form-group">
                <label for="recommended_action">Recommended Actions:</label>
                <textarea id="recommended_action" name="recommended_action" rows="4" required></textarea>

            </div>
            <div class="form-group">
                <label for="cases_per_100k">Total Cases Per 100k:</label>
                <input type="number" name="cases_per_100k" id="cases_per_100k" required>
            </div>
            <div class="form-group">

                <label for="deaths_per_100k">Total Death Per 100k:</label>
                <input type="number" name="deaths_per_100k" id="deaths_per_100k" required>
            </div>

            <input type="hidden" name="submit_timestamp" id="submit_timestamp" value="">
            <script>
                document.addEventListener("DOMContentLoaded", function () {
                    const form = document.querySelector("form");
                    const submitTimestampInput = document.getElementById("submit_timestamp");

                    form.addEventListener("submit", function () {
                        const currentTimestamp = new Date().toISOString();
                        submitTimestampInput.value = currentTimestamp;
                    });
                });
            </script>
```

Covid Form backend code



```
Covid_Form_Router.post('/', async (req, res) => {
    //Retrieve form fields
        const{
            health_metric_id,
            cases_per_100k,
            deaths_per_100k,
            submit_timestamp,
            fk_covid_county_id,
            covid_condition,
            recommended_action
        } = req.body;
        //Convert submit_timestamp to MySQL datetime format
        const recorded_at = getCurrentMySQLDatetime(submit_timestamp);

        //Insert data into the database
        const covidData = {
            health_metric_id,
            cases_per_100k,
            deaths_per_100k,
            recorded_at,
            fk_covid_county_id,
            covid_condition,
            recommended_action
        };
        database.query('INSERT INTO Covid_Metrics Set ?', covidData, (err, result) => {
            if(err){
                console.error('Error inserting data into the database:\n' + err);
                return res.redirect('/')
            }
             //add logic for rendering post result
             //for now we just render dashboard
              res.redirect('/dashboard');
        })

})
```

Peer review



RE: Code review

AS Ameen Safi <asafi1@mail.sfsu.edu>
To: Chih-Lin Chien

Today at 11:46 PM

HTML/CSS Code:
I think it is good to use the HTML form elements and attributes for the purpose, such as labels with associated inputs using the 'for' attribute. This is in alignment with the Google HTML/CSS Style Guide.
Backend Code:
The backend code, which handles the submission of the COVID-19 form data, is well-structured, and it extracts the relevant fields from the request body efficiently

-Ameen

From: Chih-Lin Chien <cchien2@sfsu.edu>
Sent: Tuesday, November 7, 2023 11:45 PM
To: Ameen Safi <asafi1@mail.sfsu.edu>
Subject: Code review

Head

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Covid Report Form</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Arial', sans-serif;
            background: linear-gradient(45deg, #3498db, #9b59b6);
        }

        .container {
            max-width: 600px;
            margin: 50px auto;
            padding: 30px;
            background-color: #fff;
            border-radius: 12px;
            box-shadow: 0px 5px 15px rgba(0, 0, 0, 0.1);
```

Covid Form Function

5) Self check on best practices for security - Trina

I. List major assets you are protecting:

- Our google cloud and database credentials.
- Any information from our database. For example, User information such as their email, password and other sensitive information, admin/director privilege.

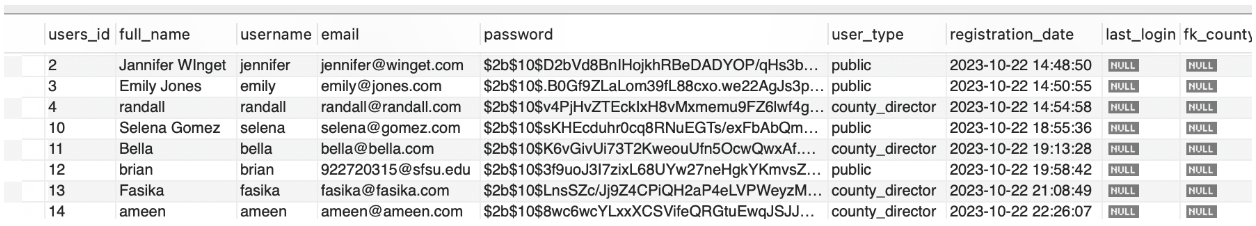
II. Say how you are protecting each asset (1- 2 lines of text per each):

We are using confidential login and password as well as github personal access token to anything related to credentials. For anything involving databases, we took multiple similar measures to protect them. For Frontend, Backend and Database work, we have at least two members working on the work and reviewing each other's code to make sure we are following proper coding practices In all our forms and user inputs, we marked them as "required" in the frontend code, specified required type, for example, email needs to be in email format. In that way, we made sure only we could enter valid information. We use bcrypt hashing in the controller to hash the password after the password input passes all the validation criteria. Then we store that encrypted password in our user table. When validating users and users type, we use a decrypting algorithm to check it with the db. We tried to use secure internet protocols when deploying it using google cloud. We checked buffer size by making sure that user input size was restricted to a certain size. We added several validation checks in the controller to prevent injection of SQL code into the forms before querying into the db

III. Confirm that you encrypt PW in the DB

We use bcrypt hashing in the controller to hash the password after the password input passes all the validation criteria. Then we store that encrypted password in our user table.

Below is a screenshot from our user table where the password is stored in its encrypted person:



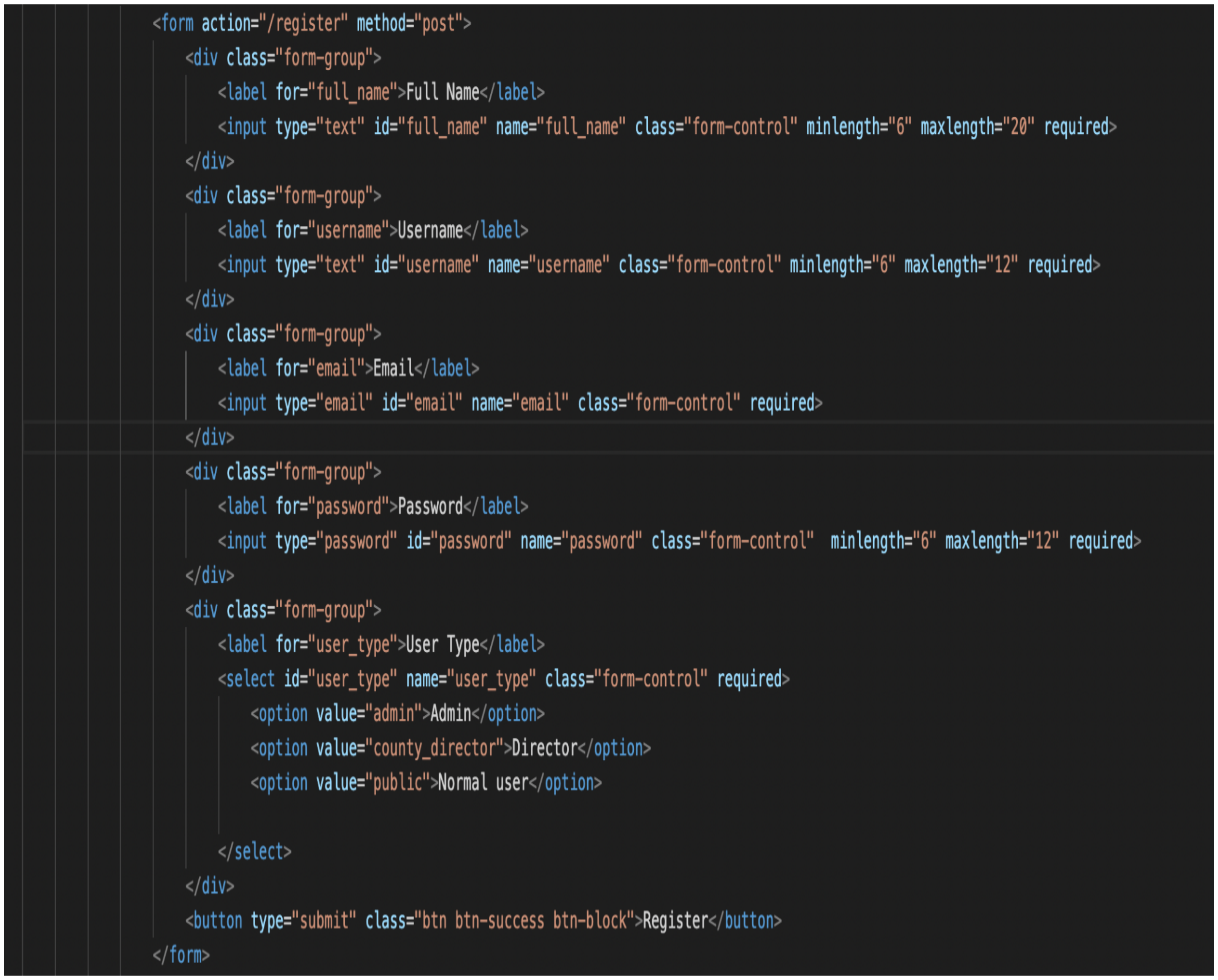
users_id	full_name	username	email	password	user_type	registration_date	last_login	fk_county
2	Jannifer WInget	jennifer	jennifer@winget.com	$2b$10$D2bVd8BnIHojkhRBeDADYOP/qHs3b…	public	2023-10-22 14:48:50	NULL	NULL
3	Emily Jones	emily	emily@jones.com	$2b$10$.B0Gf9ZLaLom39fL88cxo.we22AgJs3p…	public	2023-10-22 14:50:55	NULL	NULL
4	randall	randall	randall@randall.com	$2b$10$v4PjHvZTEckIxH8vMxmemu9FZ6lwf4g…	county_director	2023-10-22 14:54:58	NULL	NULL
10	Selena Gomez	selena	selena@gomez.com	$2b$10$sKHEcduhr0cq8RNuEGTs/exFbAbQm…	public	2023-10-22 18:55:36	NULL	NULL
11	Bella	bella	bella@bella.com	$2b$10$K6vGivUi73T2KweouUfn5OcwQwxAf.…	county_director	2023-10-22 19:13:28	NULL	NULL
12	brian	brian	922720315@sfsu.edu	$2b$10$3f9uoJ3I7zixL68UYw27neHgkYKmvsZ…	public	2023-10-22 19:58:42	NULL	NULL
13	Fasika	fasika	fasika@fasika.com	$2b$10$LnsSZc/Jj9Z4CPiQH2aP4eLVPWeyzM…	county_director	2023-10-22 21:08:49	NULL	NULL
14	ameen	ameen	ameen@ameen.com	$2b$10$8wc6wcYLxxXCSVifeQRGtuEwqJSJJ…	county_director	2023-10-22 22:26:07	NULL	NULL

IV. Confirm Input data validation (list what is being validated and what code you used) (we request you validate search bar input for up to 40 alphanumeric characters)

Currently, we are validating all our forms and user inputs. Below is the list:

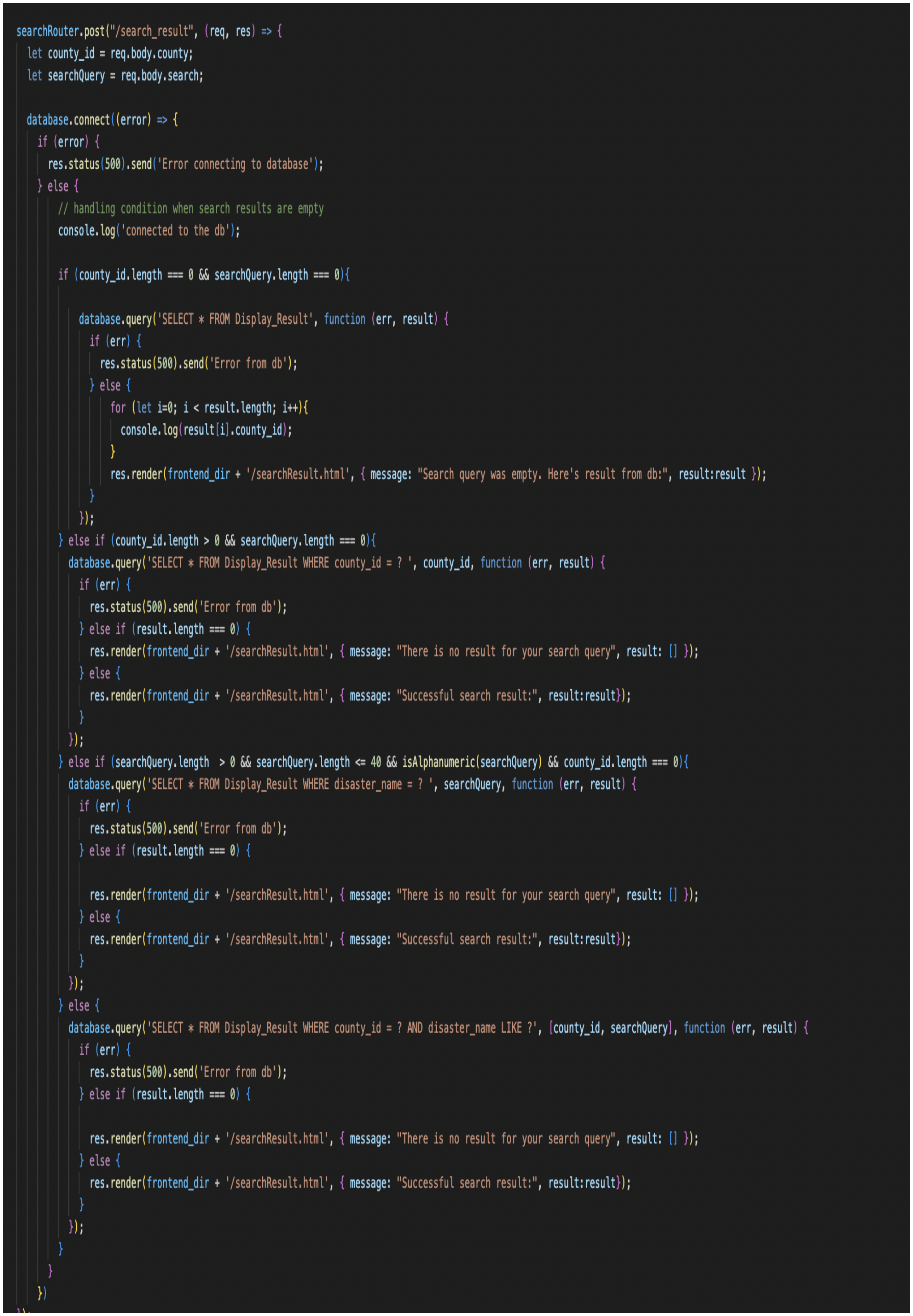
1. All fields in Registration form such as username, email, password
2. All fields in Login form such as username and password
3. Inputs for search and filter and making sure the values are alphanumeric and up to 40 characters
4. All the data entries from admins and directors

We added code for input validation in all our forms both in frontend and the backend. we marked them as “required” in the frontend code, specified required type, for example, email needs to be in email format. In that way, we made sure only we could enter valid information. Below is an example HTML code of how we are validating our Registration input form:



```html
<form action="/register" method="post">
    <div class="form-group">
        <label for="full_name">Full Name</label>
        <input type="text" id="full_name" name="full_name" class="form-control" minlength="6" maxlength="20" required>
    </div>
    <div class="form-group">
        <label for="username">Username</label>
        <input type="text" id="username" name="username" class="form-control" minlength="6" maxlength="12" required>
    </div>
    <div class="form-group">
        <label for="email">Email</label>
        <input type="email" id="email" name="email" class="form-control" required>
    </div>
    <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" name="password" class="form-control"  minlength="6" maxlength="12" required>
    </div>
    <div class="form-group">
        <label for="user_type">User Type</label>
        <select id="user_type" name="user_type" class="form-control" required>
            <option value="admin">Admin</option>
            <option value="county_director">Director</option>
            <option value="public">Normal user</option>

        </select>
    </div>
    <button type="submit" class="btn btn-success btn-block">Register</button>
</form>
```

In the backend, we added various JavaScript code to validate our inputs, its length. We also added validation when we query from the database and only send responses back to the client side after they all pass. Below is an example of our validation code in the backend for our search:



```
searchRouter.post("/search_result", (req, res) => {
  let county_id = req.body.county;
  let searchQuery = req.body.search;

  database.connect((error) => {
    if (error) {
      res.status(500).send('Error connecting to database');
    } else {
      // handling condition when search results are empty
      console.log('connected to the db');

      if (county_id.length === 0 && searchQuery.length === 0){

            database.query('SELECT * FROM Display_Result', function (err, result) {
              if (err) {
                res.status(500).send('Error from db');
              } else {
                for (let i=0; i < result.length; i++){
                  console.log(result[i].county_id);
                }
                res.render(frontend_dir + '/searchResult.html', { message: "Search query was empty. Here's result from db:", result:result });
              }
            });
      } else if (county_id.length > 0 && searchQuery.length === 0){
        database.query('SELECT * FROM Display_Result WHERE county_id = ? ', county_id, function (err, result) {
          if (err) {
            res.status(500).send('Error from db');
          } else if (result.length === 0) {
            res.render(frontend_dir + '/searchResult.html', { message: "There is no result for your search query", result: [] });
          } else {
            res.render(frontend_dir + '/searchResult.html', { message: "Successful search result:", result:result});
          }
        });
      } else if (searchQuery.length  > 0 && searchQuery.length <= 40 && isAlphanumeric(searchQuery) && county_id.length === 0){
        database.query('SELECT * FROM Display_Result WHERE disaster_name = ? ', searchQuery, function (err, result) {
          if (err) {
            res.status(500).send('Error from db');
          } else if (result.length === 0) {

            res.render(frontend_dir + '/searchResult.html', { message: "There is no result for your search query", result: [] });
          } else {
            res.render(frontend_dir + '/searchResult.html', { message: "Successful search result:", result:result});
          }
        });
      } else {
        database.query('SELECT * FROM Display_Result WHERE county_id = ? AND disaster_name LIKE ?', [county_id, searchQuery], function (err, result) {
          if (err) {
            res.status(500).send('Error from db');
          } else if (result.length === 0) {

            res.render(frontend_dir + '/searchResult.html', { message: "There is no result for your search query", result: [] });
          } else {
            res.render(frontend_dir + '/searchResult.html', { message: "Successful search result:", result:result});
          }
        });
      }
    }
  })
});
```

6. Self Check: Adherence to Non-Functional specs - Ameen

1. 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).

ON TRACK

2. Application shall be optimized for standard desktop/laptop browsers e.g., must render correctly on the two latest versions of two major browsers

ON TRACK

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

ON TRACK

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

DONE

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

ON TRACK

6. The language used shall be English.

DONE

7. Application shall be very easy to use and intuitive.

ON TRACK

8. Google maps and analytics shall be added

ON TRACK

9. No e-mail clients shall be allowed. You shall use webmail.

ON TRACK

10. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.

DONE

11. Site security: basic best practices shall be applied (as covered in the class)

ON TRACK

12. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development

ON TRACK

13. 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).

ON TRACK