SW Engineering CSC648/848 Spring 2023

Application Title: Dooms Day Alert

Team 06

Umid Muradli (umuradli@mail.sfsu.edu)

Khabibullo Khujamberdiev

North Wiriyachinnakarn

Matthew Marcos

Arin Ton

Edward Li

Milestone 4

Date: 04/25/2023

History Table

Date Submitted	04/25/2023
Date Revised	

1) Product summary

- **Name of the product**
    - Dooms Day Alert
- **List of major P1 functions**
    - Users have the option to create an account by entering their email, username, and choosing a password, also receive alerts based on their chosen county.
    - Users are able to opt-in as a government official, which allows them to add data to the metrics.
    - Users can search a specific city within the website's search features.
    - On the Covid-19 page, users can search for a specific county to access information such as the number of covid-19 cases reported in that area.
    - On the Weather page, users can search for a specific county to access information, such as the temperature, air quality, and warnings.
    - On the Wildfires page, users can search for a specific county to access information such as affected area.
    - On the Security page, users can search for a specific county to view incident types, dates, times and more.
    - Users can interact with maps to visualize the different counties within California.
    - Users can interact with the map and choose between Security and Wildfire to display on the map.
    - Users can search for the name of the fire.
    - Users can search for a specific incident type.
    - Users can access and view temperature, air quality, and warnings for weather.

    - Users can access and view incidents and law enforcement for security/crimes.
    - Users can access and view fire incidents, evacuation orders, or warnings for wildfires.
    - Users can access and view case counts, deaths and recoveries.
    - Users are required to provide information to become an official account.
- **Say what is unique in your product**
    - This product is unique because it offers a comprehensive and centralized platform that combines various critical metrics, such as covid-cases, weather, wildfires, and security and it is specifically designed for California. It also allows government officials to contribute data, ensuring accurate and up-to-date information. The interactive maps and advanced search features provide users with a user-friendly and visually engaging experience to stay informed about local conditions and events.
- **URL to your product accessible to instructors, on deployment server**
    - http://34.212.246.140/

2) Usability test plan

Test Objective

One major function that we have implemented onto our website is a search feature. This search feature has multiple characteristics. One of the key benefits of our search feature is the ability to filter results based on specific criteria. Users can refine their search by selecting different filters, such as covid, weather, security, or wildfire, which helps to narrow down the results to only those that are relevant to their needs. This feature is being tested due to its importance to our website. Our website, Dooms Day Alert, heavily focuses on the search to give users the information that they are looking for. The primary objective of this test is to evaluate the usability of the search feature on the website. We are currently testing the search feature to ensure that it meets the high standards that we have set for our website. Our team of developers and designers are constantly monitoring console logs and making adjustments to the search feature as needed. By doing so, we are able to identify any areas of improvement and make changes that will enhance the overall user experience.

Test Background and Setup

The Test Background and Setup is an essential aspect of conducting a successful evaluation of a website. It is critical to ensure that the testing environment is set up correctly to get accurate and valid results. The following is an elaboration of the various components of the test background and setup. The first aspect to consider is the system setup. In this test, the website's live version will be used without any modifications. This approach is crucial as it provides an accurate representation of the website's real-life performance. It is essential to ensure that the system is stable and functioning correctly before the test begins to avoid any technical glitches that could interfere with the test's results.

The second aspect of the test background and setup is the starting point. In this test, the testers will start at the homepage of the website. This approach is significant as it provides a baseline for user interaction with the website. It also enables testers to evaluate the website's ease of navigation, information architecture, and other critical elements.

The third aspect to consider is the intended users. In this test, the intended users are individuals who have basic computer and internet skills and are potential users of the website. Such potential users are community members and community leaders. This approach is significant as it provides a realistic representation of the website's target audience. It enables testers to evaluate the website's accessibility, usability, and user-friendliness from the perspective of the target audience.

The fourth aspect of the test background and setup is the URL of the system to be tested. In this test, the URL of the system to be tested is http://34.212.246.140/. This approach is significant as it provides a specific location for the website being tested, which ensures that testers are evaluating the correct website.

Finally, the fifth aspect to consider is what is to be measured. In this test, the focus will be on user satisfaction evaluation, which will be measured through a Likert test. This approach is significant as it provides a standardized method for measuring user satisfaction with the website. It enables testers to evaluate the website's effectiveness in meeting the needs and expectations of the users.

Usability Task description

1. The user task is to use the search feature within the home page and search for a specific county within California.

2. Then, the user needs to click on the filter and select Covid-19, Security, Weather, and Wildfire.
3. Once the user has selected all the filters, the user needs to click search which will show all the results of the metrics within that county.

Measure Effectiveness: The way we will evaluate the effectiveness of the search feature is when the user searches for a specific county. The results should show the information accurately to the specific county that the user searched.

Measure Efficiency: The way we will evaluate the efficiency of the search feature is by the amount of time that is required for the user to complete the task and the number of clicks required to navigate and show the results that the user desires.

Lickert subjective test

How easy was it to use the search feature on the website?

Very Difficult	Difficult	Neutral	Easy	Very Easy
				✓

How satisfied were you with the information provided when the results show?

Very Unsatisfied	Unsatisfied	Neutral	Satisfied	Very Satisfied
			✓	

Overall, how satisfied were you with your experience using the search feature?

Very Unsatisfied	Unsatisfied	Neutral	Satisfied	Very Satisfied
			✓	

3) QA test plan

QA Test Plan #1

1. Test Objectives:
    - Searching for a county, city or zip code shall return events in the respective category/categories in the searched location.
2. HW and SW setup (including URL):
    - Hardware:
        i. Dell Laptop, Intel Core i7 CPU, 16GB RAM
    - Software:
        i. Firefox Browser (latest version)
        ii. Chrome Browser (latest version)
    - URL:
        i. http://34.212.246.140/
3. Feature to be tested
    - Search
4. QA Test plan #1 (Firefox Browser):

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Pass/Fail
1	County search	Searching county should show results for that county in all tables	“Alpine”	List items for Alpine County for Covid-19, Security, Weather, Wildfire	PASS
2	City search	Searching city should return results for the city’s respective county	“Pleasant Hill”	List items for Contra Costa County for Covid-19, Security, Weather, Wildfire	PASS

3	Zip code search	Search should return results for the county containing the zip code	“94102”	List items for San Francisco County for Covid-19, Security, Weather, Wildfire	PASS

QA Test plan #2 (Chrome Browser):

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Pass/Fail
1	County search	Searching county should show results for that county in all tables	“Alpine”	List items for Alpine County for Covid-19, Security, Weather, Wildfire	PASS
2	City search	Searching city should return results for the city’s respective county	“Pleasant Hill”	List items for Contra Costa County for Covid-19, Security, Weather, Wildfire	PASS
3	Zip code search	Search should return results for the county containing the zip code	“94102”	List items for San Francisco County for Covid-19, Security, Weather, Wildfire	PASS

4) Code Review

Naming Conventions:

1. Variables: camelCase (i.e. `homeCSS` and `selectedCounty`)

2. Component Names: PascalCase (i.e. `Card` and `CardGroup`)

3. CSS class names: kebab-case

File Naming and Organization:

For the file names we used a combination of PascalCase, camelCase, and kebab-case. PascalCase examples are “About.jsx”, “AddData.jsx”, “Alerts.jsx”, “ChangeAccountType.jsx”. In camelCase the file names would be “about.jsx”, “addData.jsx”, “alerts.jsx”, and “changeAccount.jsx”. In kebab-case we have files like “add-data.jsx”, “change-account.jsx”, and “change-password.jsx”.

Code Formatting:

The code uses the standard JavaScript formatting style with an indentation of 2 spaces.

Code Comments:

The code comments are used to describe basic information about the purpose of the function or method. The code includes comments that use both single-line and multi-line formats.

Re: Request for peer review of search function code

Arin Ton <aton@mail.sfsu.edu>
Mon 4/24/2023 8:39 PM
To: Edward Li <eli9@sfsu.edu>

Hi Edward,

I think your code looks good. I like the consistent use of camel case for variable names and consistent use of indentation, especially when there are multiple attributes, for example in the `<input>` tags.

I also like the proper use of indentation for different levels of elements in the code as well as line breaks between groups of code to make the code more readable and easy to understand.

However, I did notice that there are some unnecessary comments on the code, for example the comment `` `/* <NewHeader loggedIn={loggedIn} /> */` ``, the comments for Covid-19 and Security filters, and the comment` `` `// Updating the state with the received data` ``.

Other than that, I am satisfied with the efficiency, readabaility and effectiveness of the code.

Feel free to email me back if you have other questions regarding your code or coding style.

-Arin Ton

From: Edward Li <eli9@sfsu.edu>
Sent: Monday, April 24, 2023 8:09 PM
To: Arin Ton <aton@mail.sfsu.edu>
Cc: Matthew Mercado Marcos <mmarcos4@mail.sfsu.edu>; Umid Muradli <umuradli@mail.sfsu.edu>; North Wiriyachinnakarn <swiriyachinnakarn@mail.sfsu.edu>; Khabibullo Khujamberdiev <kkhujamberdiev@sfsu.edu>
Subject: Request for peer review of search function code

Dear Arin,

I am reaching out to you because I would like to request your assistance with a peer review of our search function code. We believe that we have made some significant improvements to the code, but we would appreciate your feedback and input on how we can further optimize it. We would like to request that you review our code and provide us with your thoughts and recommendations. Specifically, we would be interested in your thoughts on the efficiency, readability, and overall effectiveness of the code. We would be happy to provide you with more information about the project, including any necessary background information or context.

Thank you for your time and consideration.

Best regards,

Edward Li

The Code:

```
import React, { useState } from "react";
import homeCSS from "../css/home.module.css";
import Card from "react-bootstrap/Card";
import CardGroup from "react-bootstrap/CardGroup";
import Axios from "axios";

function Home() {
  // Using state hooks to manage component state
  const [selectedCounty, setSelectedCounty] = useState("");
  const [countyData, setCountyData] = useState([]);
  const [covid, setCovid] = useState(true);
  const [selectedFilters, setSelectedFilters] = useState({
    "Covid-19": true,
    Security: true,
    Weather: true,
    Wildfire: true,
  });

  const submitData = (event) => {
    event.preventDefault();
    Axios.get("http://34.212.246.140:3001/check/", {
      params: {
        selectedCounty: selectedCounty,
      },
    })
      .then((response) => {
        if (response.data.length === 0) {
          setCountyData([]);
          alert("No data found for the selected county.");
        } else {
          setCountyData(response.data);
        } // Updating the state with the received data
      })
      .catch((error) => {
        console.error(error);
        setCountyData([]);
        alert("Failed to fetch data.");
      });
  };

  // Function to handle changes in the filter checkboxes
  const handleFilterChange = (event) => {
    const filterName = event.target.value;
    setSelectedFilters({
      ...selectedFilters, // Copying the current state
      [filterName]: event.target.checked, // Updating the value of the selected filter
    });
  };

  // Filtering county data based on the selected filters
  // eslint-disable-next-line
  let filteredCountyData = [];
  if (Array.isArray(countyData)) {
    filteredCountyData = countyData.filter((data) => {
      if (
        (selectedFilters["Covid-19"] && data.category === "Covid-19") ||
        (selectedFilters["Security"] && data.category === "Security") ||
        (selectedFilters["Weather"] && data.category === "Weather") ||
        (selectedFilters["Wildfire"] && data.category === "Wildfire")
      ) {
        return true;
      }
      return false;
    });
  }
  const cardComponents = [];
  if (selectedFilters["Covid-19"]) {
    cardComponents.push(
      <Card
        bg={"light"}
        key={"light"}
        text={"dark"}
        style={{ width: "18rem" }}
        className="mb-2"
        id="covid"
      >
```

```
          Array.isArray(countyData) && countyData.length > 0 ? (
            countyData.map((data, index) => (
              <Card.Text key={index}>
                {"Searched Location: " + data.county}
                <br />
                {"Confirmed Cases: " + data.confirmedCases}
                <br />
                {"Deaths: " + data.deaths}
                <br />
                {"Recoveries: " + data.recoveries}
              </Card.Text>
            ))
          ) : countyData.length === 0 ? (
            <Card.Text> Search for a County... </Card.Text>
          ) : (
            <p>{"No results found."}</p>
          )
        }
      </Card.Body>
    </Card>
  );
}

/* Security */

if (selectedFilters["Security"]) {
  cardComponents.push(
    <Card
      bg={"light"}
      key={"light"}
      text={"dark"}
      style={{ width: "18rem" }}
      className="mb-2"
    >
      <Card.Header>Security</Card.Header>
      <Card.Body>
        {
          // If Security filter is selected and countyData is not empty, display data
          selectedFilters["Security"] &&
          Array.isArray(countyData) &&
          countyData.length > 0 ? (
            countyData.map((data, index) => (
              <Card.Text key={index}>
                {"Searched Location: " + data.county}
                <br />
                {"Type: " + data.incidentType}
                <br />
                {"Description: " + data.incidentDescription}
                <br />
                {"Date: " + data.date}
                <br />
                {"Time: " + data.time}
                <br />
                {"Address: " + data.address}
              </Card.Text>
            ))
          ) : countyData.length === 0 ? (
            <Card.Text> Search for a County... </Card.Text>
          ) : (
            <p>{selectedFilters["Security"] ? "No results found." : ""}</p>
          )
        }
      </Card.Body>
    </Card>
  );
}

if (selectedFilters["Weather"]) {
  cardComponents.push(
    <Card
      bg={"light"}
      key={"light"}
      text={"dark"}
      style={{ width: "18rem" }}
      className="mb-2"
    >
```

```
      <Card.Header>Weather</Card.Header>
      <Card.Body>
        {
          // If Weather filter is selected and countyData is not empty, display data
          selectedFilters["Weather"] &&
          Array.isArray(countyData) &&
          countyData.length > 0 ? (
            countyData.map((data, index) => (
              <Card.Text key={index}>
                {"Searched Location: " + data.county}
                <br />
                {"Farenheight: " + data.temperature}
                <br />
                {"AQI: " + data.AQI}
                <br />
                {"Warnings: " + data.weather_warnings}
              </Card.Text>
            ))
          ) : countyData.length === 0 ? (
            <Card.Text> Search for a County... </Card.Text>
          ) : (
            <p> {selectedFilters["Weather"] ? "No results found." : ""}</p>
          )
        }
      </Card.Body>
    </Card>
  );
}

if (selectedFilters["Wildfire"]) {
  cardComponents.push(
    <Card
      bg={"light"}
      key={"light"}
      text={"dark"}
      style={{ width: "18rem" }}
      className="mb-2"
    >
      <Card.Header>Wildfire</Card.Header>
      <Card.Body>
        {
          // If Wildfire filter is selected and countyData is not empty, display data
          selectedFilters["Wildfire"] &&
          Array.isArray(countyData) &&
          countyData.length > 0 ? (
            countyData.map((data, index) => (
              <Card.Text key={index}>
                {"Searched Location: " + data.county}
                <br />
                {"Wildfires: " + data.name}
                <br />
                {"Date Start: " + data.dateStart}
                <br />
                {"Date End: " + data.dateEnd}
                <br />
                {"Warnings: " + data.wildfire_warnings}
                <br />
                {"Casualties: " + data.casualties}
              </Card.Text>
            ))
          ) : countyData.length === 0 ? (
            <Card.Text> Search for a County... </Card.Text>
          ) : (
            <p>{selectedFilters["Wildfire"] ? "No results found." : ""}</p>
          )
        }
      </Card.Body>
    </Card>
  );
}
return (
  <div>
    {/* <NewHeader loggedIn={loggedIn} /> */}
    <section class="text-center">
      <div>
        <div class="col-lg-6 col-md-8 mx-auto">
          <h1 class="fw-light">Dooms Day Alert</h1>
        </div>
```

5) Self-check on best practices for security

List of major assets to protect and how we are protecting them:

- Accounts
    1. Hashed passwords make it difficult for hackers to find your password.
- Events in data tables
    1. Requiring verification for accounts to be labeled official & be able to add information

Confirm password encrypted in DB:

#id	username	email	password
1	arin	aton@mail.sfsu.edu	$2b$10$jjiujqHGxe9oenKFUG0DZOwLnWAbZ4G1QmrFAqXB9Bq9HijOtR77e
2	umid	umuradli@mail.sfsu.edu	$2b$10$BF.UTsYnxPa2E/l4ISzPP.hIn.U5KCPSAiZ7T0032Cff.4PryxURi
3	north	swiriyachinnakarn@mail.sfsu.edu	$2b$10$eL6yFWtQ1UCF4r.zlK9lE.xYr.3O4EmKs9QCD7GTT4ePCbB2kaop.
4	matthew	mmarcos4@mail.sfsu.edu	$2b$10$xCW0M8Ftad5I/bVCwEJ2Z.KLVPX6PEZMT7jBlQ9LNqaKqLEbqZ2hW
5	edward	eli9@sfsu.edu	$2b$10$YccFwmIS7Zwmb1ekvPsWPuY.Z50/QrtNbNN2ojZQqGiMbedxGz6A2
6	khabibullo	kkhujamberdiev@sfsu.edu	$2b$10$bNi713jwRzkBfClSKeKcG.qv5fo9N0Q545xBQGwG/2yzfIJdLDANm

Confirm input validation:

- Input is validated so the user can only input a search term of up to 40 alphanumeric characters.
- ```
  // validate search input
  if (!/^[a-zA-Z0-9]{1,40}$/.test(searchTerm)) {
    console.log("Invalid search term. Please enter up to 40 alphanumeric characters.");
    res.status(400).send("Invalid search term. Please enter up to 40 alphanumeric characters.");
  }
  ```
  ```

6) Self-check: Adherence to original Non-functional specs

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). - DONE
2. Application shall be optimized for standard desktop/laptop browsers e.g., must render correctly on the two latest versions of two major browsers. - DONE
3. Selected application functions must render well on mobile devices (this is a plus) - DONE
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. - DONE
6. The language used shall be English.- DONE
7. Application shall be very easy to use and intuitive.- ON TRACK - Suggestion is that mobile render should not show the search bar after hamburge menu is clicked. It can confuse the user.
8. Google maps and analytics shall be added - DONE
9. No email clients shall be allowed. You shall use webmail. - DONE
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) - DONE
12. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development - DONE

13. The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2023. For Demonstration Only” at the top of the WWW page. (Important so not to confuse this with a real application). -DONE