WEB PROGRAMMING AND INTERNET TECHNOLOGIES

WEATHER APP PROJECT
REPORT

Ali Büyükberber - 041901011,
Ataman Fedai - 042101006,
Göktuğ Kocauşu - 042001019,
Bayram Yağcı - 041901044

Team

Ali Büyükberber: Design and Front-End
He contributed to the overall design and design of the site. He made arrangements such as background colors, box positioning, map positioning. As Front-End, he helped the project in HTML and CSS language, he designed the site in terms of design by making additions to the code in CSS language.

Ataman Fedai: Project Manager and Front-End
Throughout the project, he maintained a strong collaboration, exchanging feedback, discussing challenges, and finding solutions to ensure the successful completion of the weather forecast website. Their collective efforts resulted in the delivery of a high-quality, user-friendly, and functional website that met the project's expectations. Moreover, besides that he helped arrange the arrangement on map and temperature demonstration with CSS language.

Bayram Yağcı: Front-End and Tester ()
As Front-End, he helped the project in HTML and CSS language, he designed the site in terms of design by making additions to the code in CSS language. As a tester, random cities

in 10 different countries, which were entered as input, were tested. After the map was added, places were randomly selected on the map and their responses were tested. Support was received from other people in the team and the testing and front-end part of the project was completed.

Göktuğ Kocauşu: Back-End and Design

The goal of this project is to create an interactive weather application that collects real-time meteorological data based on user-provided city names using the OpenWeatherMap API. The location of the user-selected city is also visually represented using the LeafletJS mapping package.

To provide trustworthy, accurate weather data, the OpenWeatherMap API was used. Using JavaScript fetch operations, this data was obtained, parsed, and then presented to the user.

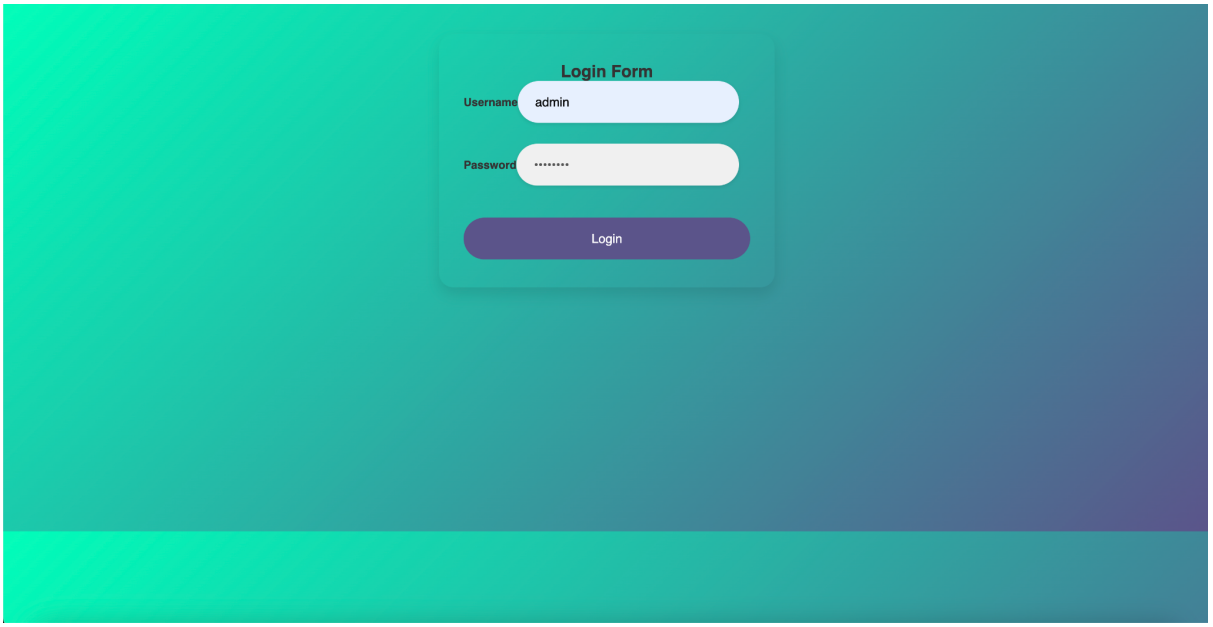
An appealing map was incorporated into the program using LeafletJS, and it was modified using the OpenWeatherMap API's geographic coordinates in response to the user's city input.

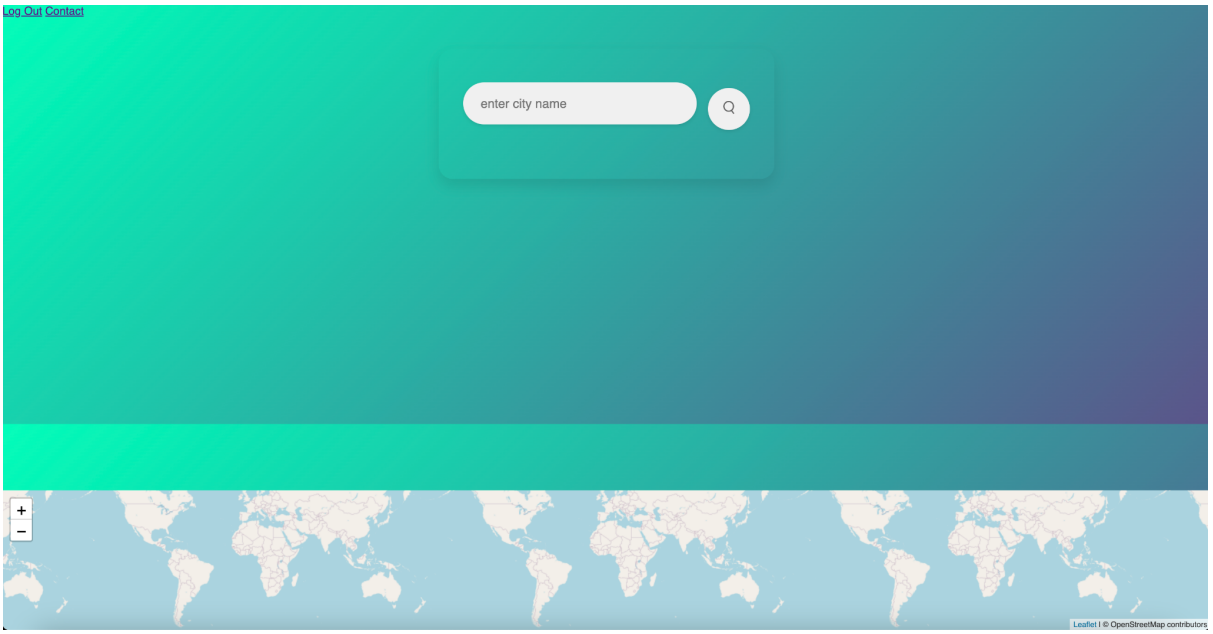
The application's layout was crafted to be intuitive and adaptive, ensuring a superior user experience on various device types. The weather information and map were displayed concurrently for ease of understanding.

Finally, this project underscores the practical application of APIs in web development, the management of asynchronous JavaScript, and the significance of adaptable design.

Client Side

The user opens the site, after opening the username and password section for the log in, and "admin" for the user section, "password" for the password section is entered. After logging in, the search button appears at the top and we enter the name of the city where we want to get weather information and click the search button. In addition to the temperature information, wind and humidity information are also included in the opening. When searched, the city written on the map is displayed. If the user clicks on a city or a district on the map, he gets the weather information of that place. If incorrect or incomplete information is entered, it will give an error and cannot provide any information. With the Contact button at the top left, the name, surname, mail and message sections are displayed, and when the submit button is pressed, an alert is issued and a thank you message is published. Clicking the LogOut button in the upper left will exit the application and request a login again.


Login Form
Username
admin
Password
Login


Log Out Contact
enter city name
+
−
Leaflet | © OpenStreetMap contributors

Log Out Contact

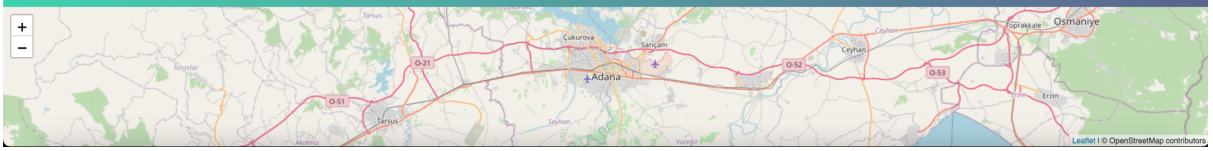
enter city name

23°C

Adana Province

88%
Humidity

2.06km/h
Wind Speed



BACK

Contact Form

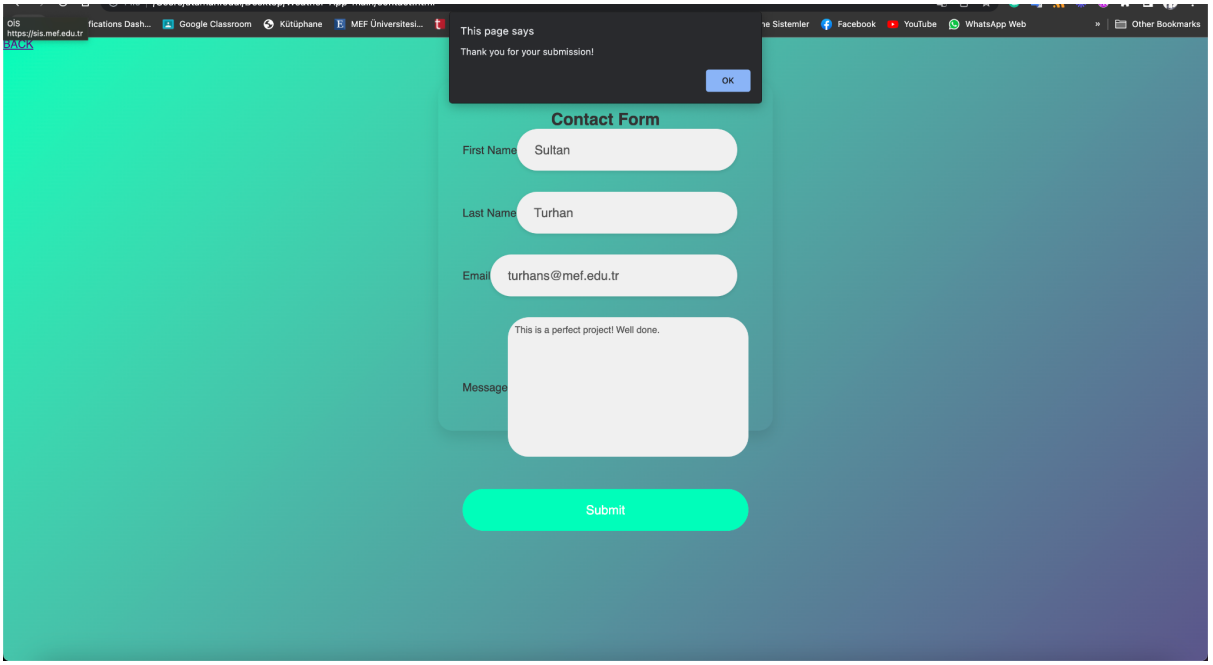
First Name: Sultan

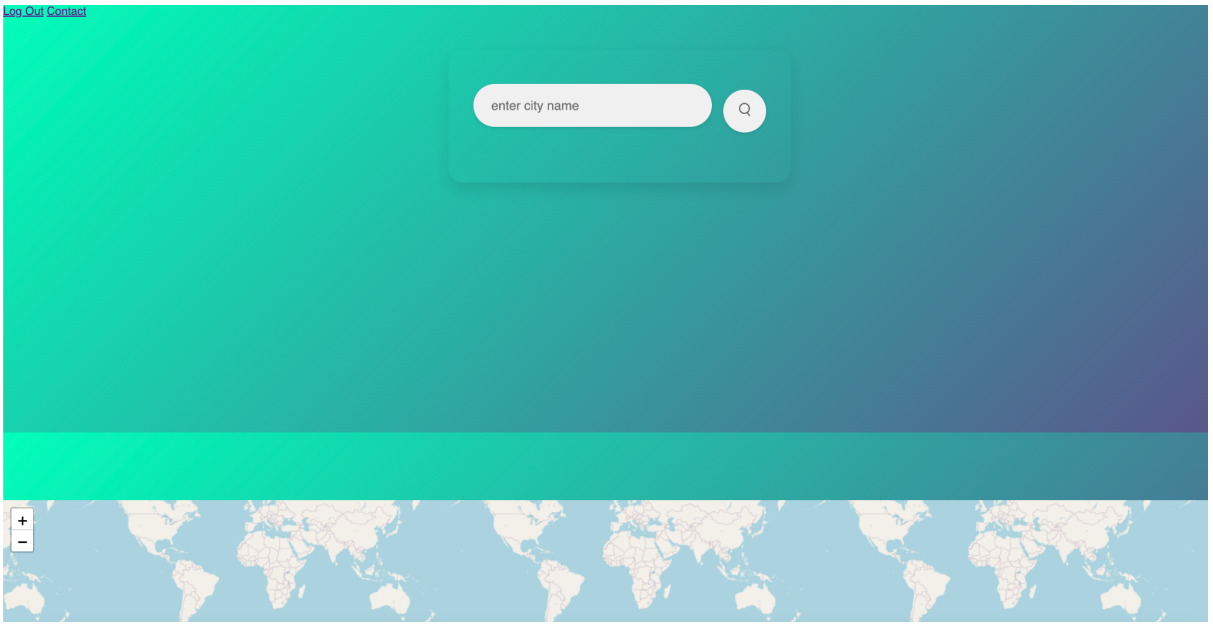
Last Name: Turhan

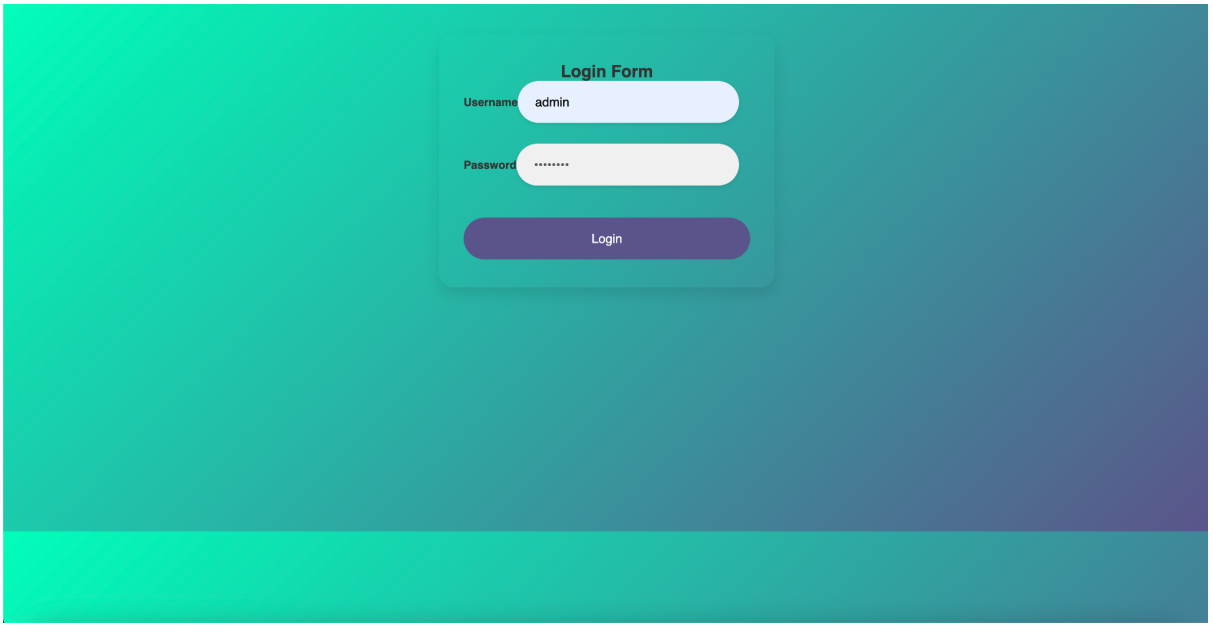
Email: turhans@mef.edu.tr

Message: This is a perfect project! Well done.

Submit


OIS
https://sis.mef.edu.tr
Google Classroom
Kütüphane
MEF Üniversitesi...
Facebook
YouTube
WhatsApp Web
Other Bookmarks
This page says
Thank you for your submission!
OK
Contact Form
First Name
Sultan
Last Name
Turhan
Email
turhans@mef.edu.tr
Message
This is a perfect project! Well done.
Submit


Log Out
Contact
enter city name



Enhancements

Transitioning the Interactive Weather App to a serverless structure can provide several improvements:

Adaptable Scaling: Serverless architecture stands out due to its automatic scalability. If your weather app witnesses an unexpected surge in user activity, the serverless framework will instantaneously adjust to manage this increase, ensuring a seamless user experience.

Decreased Response Time: Serverless architecture can execute code near the end-users by deploying in various geographical areas. This shortens the response time, ensuring faster weather updates for users, enhancing their experience.

Reliability: Serverless structures are inherently designed to be highly reliable and fault-tolerant. This ensures minimal downtime for the weather app, which translates into consistent user satisfaction.

Low Maintenance: The service provider manages server upkeep in a serverless framework, enabling you to focus on refining application features, like expanding weather data or improving the user interface.

Quick Deployment: A serverless architecture can expedite the application development and deployment process since there's no need for server management, giving you more time to focus on developing the application.

Real-time Updates and Event-Driven Processing: Serverless architecture supports event-driven and real-time processing, allowing real-time weather updates and instant alerts for severe weather events.

Integration Flexibility and Microservices: It's easy to integrate third-party services and APIs in a serverless environment. This allows for the application to be segmented into microservices that function independently, facilitating the addition of new features or updates without affecting the overall application.

In summary, the shift to a serverless architecture for the Interactive Weather App will not only improve scalability, reliability, and affordability but will also decrease response times, improve user experience, and simplify maintenance.

Presentation

https://alibberber.github.io/Weather-App/

Github

https://github.com/alibberber/Weather-App

What to do Next

A description of what you would do if you had three more weeks to work on the project.

1. Implement a geolocation function so that users may access weather data based on their current location. This can be accomplished by calling the 'getWeatherByCoordinates' function with the user's latitude and longitude after retrieving them using the Geolocation API of the browser. Additionally, zooming in on the map will show the location of the city that was searched for in the search field.

2. Enhance the user experience by taking into account the addition of new features. You could, for instance, add auto-complete city name recommendations as the user types them into the search input area. This can be done by using external APIs or locally holding a list of well-known city names. It is possible to establish a favorite list, add the appropriate locations to it, and then display the weather prediction from that list with just one click. Alternatively, the commonly searched cities can be displayed as a separate list with just one click.

3. Add unit conversion: Give users the option to choose between various temperature, wind speed, and humidity measurement units. You can add dropdown menus or buttons to let users switch between metric and imperial units.

4. Implement a forecast display: Enlarge the app to display weather forecasts for a number of days. You can adjust the user interface (UI) to display the new information after retrieving forecast data from the weather API. To enable customers to browse among several prediction periods, think about utilizing a carousel or tabs.