WELLSY*(health and selfcare website)*

Overview:

WELLSY is a React-based web focused on health and self-care.
The website helps users take care of their physical and mental health by providing different features such as healthy meal suggestions, mood tracking, and a helpful chatbot.

The pages in the project are:

*Home Page

*Mood Page

*Meals Page

*Chatbot Page

HOME PAGE

The Home page layout is simple and responsive — created using functional components, CSS Flexbox, and React hooks.
The page includes a navigation bar with links to “Home,” “Mood,” “Meals,” and “Chatbot.”
In the Home Page of WELLSY, I used HTML and CSS to design a clean and welcoming layout.
At the top, there is the website title “WELLSY” and a short welcoming message introducing the purpose of the site.

Below that, I created three main sections – Mood, Meals, and Chatbot.
Each section has a short description inside a box, styled using CSS for spacing, borders, shadows, and alignment to make the layout organized and visually appealing.

I chose light and calming colors to match the theme of health and self-care.
All design decisions, including font style, color palette, and layout spacing, were made using CSS to keep the page simple and easy to read.

MOOD PAGE

The Mood Page is where users can easily track how they feel and receive instant wellness tips.
Built with React, it uses the useState hook to manage all form data smoothly and dynamically.
Users select their mood, sleep quality, hydration, exercise, and stress levels then get personalized advice designed to help them improve their day.

All logic runs locally, ensuring fast performance without relying on any external APIs.
The design is powered by a dedicated CSS Module, giving the page a clean, calm, and modern look.
Soft colors and rounded elements create a relaxing experience that matches the page's purpose.

Navigation between sections is made simple with React Router Links, allowing users to move easily between Mood, Meals, Chatbot, and Home.
The Mood Page delivers a seamless blend of functionality and wellness in one elegant interface.

MEALS PAGE

Its designed to be helpful for the user as it recommends healthy meals for breakfast ,lunch and dinner .healthy and easy meals to make if your aim is a healthy lifestyle
(Functions Used)
usestate (React Hook)
usestate was used to store the state and details of the selected meal.
When you click on a meal image the function updates the state with that meal's information (ingredients and recipe)This allows the page to show the details of the clicked meal
(Buttons and interactions)
Each meal image is clickable (that was able by using "onlclick")
The" onclick" event changes the state to show the details of the selected meal
There's also a" Back" button that returns to the main meals page
(CSS Styling)
All meal images have the same width and height to keep the design organized.
When the user hovers over a meal image with the mouse a zoom effect appears using the CSS (hoover effect)
(Main Page)
The main meals page contains three sections
Breakfast, Lunch,Dinner
Each section displays several meal images with titles
When you click the " Back" button the website returns to this main meals page

CHATBOT PAGE

(function of chatbot on the website)

The Chatbot exists to help the user interact with the website naturally — asking about mood, food, or self-care tips. Its main idea is to act like a friend who helps you take care of yourself, not a tracking or monitoring tool.
(language and libraries used)
We used JavaScript through the React.js library because it makes the page interactive and fast. We also used Framer Motion for animations and Lucide React for icons.
(using react hooks)

We used “usestate” to store messages and loading states, and “useeffect” to execute certain actions when the chat opens or its state changes. This helps organize the code and makes handling states easier.

(source of the API key)

The API Key comes from the Gemini API service by Google, which generates smart AI-based responses.

(storing API key)

It’s stored in a file called “.env” for security reasons, so it doesn’t appear in the public code.

(calling the API key)

It’s called from the app’s environment variables, allowing access to the .env value without exposing it.

(API url)

The connection is made through the Gemini API endpoint responsible for text generation.

(type of request used)

The type used is POST because we send data from the user and expect a response from the server.

(headers)

They include the data type (JSON) so the server can understand and process it correctly.

(request body)

It contains the text written by the user, following a specific structure that Gemini reads to generate a suitable response.

(handling the response)

After receiving the response from the server, it’s converted to text and the chatbot’s reply is extracted.

(extracting the text response)

Only the final reply part is extracted so that it appears clearly and simply to the user.

(styling)

All the styles are in one CSS file dedicated to the Chatbot, keeping the design organized and separate.

(icon)

The bot has its own icon, and the user has a send icon, which makes the interface interactive and appealing.

(runtime issues)

At first, there was a CORS security issue when the API key was visible, but it was fixed after moving the key to the .env file.