Daily\_Vitals

To explain your project in a way that’s clear and technical but still simple, let’s go over each key part of the project, breaking down its purpose and structure. You can use this outline to communicate your plan to your lecturer:

---

### Project Overview: Daily Vitals Tracker

The \*\*Daily Vitals Tracker\*\* is a health logging app focused on tracking daily health information. Users can record their mood, sleep, and any symptoms each day. Over time, they’ll have a record of their health stats, and they can view, edit, or delete these records as needed.

---

### Project Structure and Technology Choices

1. \*\*Overall Structure\*\*:

- The app is divided into a \*\*frontend\*\* and a \*\*backend\*\*.

- The \*\*frontend\*\* is where the user interacts, adding health data or viewing their history.

- The \*\*backend\*\* is responsible for handling data requests, managing data storage, and ensuring that data is saved securely.

2. \*\*Frontend Details\*\*:

- \*\*HTML\*\*: The frontend will have a simple HTML layout with input fields (like text boxes) and buttons. This is where the user will enter information about their mood, sleep, and symptoms.

- \*\*JavaScript\*\*: JavaScript will allow the frontend to communicate with the backend without needing to refresh the page. When the user adds or views data, the frontend sends requests to the backend and displays the responses immediately.

3. \*\*Backend Details\*\*:

- \*\*Node.js and Express\*\*: We’re using Node.js (a JavaScript runtime) with Express (a framework) for the backend. Together, they make it possible to write backend code in JavaScript, keeping the whole project in one language.

- \*\*API Endpoints\*\*: The backend is structured around \*\*API endpoints\*\*, which are like individual data handlers:

- \*\*POST endpoint\*\* for creating a new health log.

- \*\*GET endpoint\*\* for reading all previous health logs.

- \*\*PUT endpoint\*\* for updating an existing log.

- \*\*DELETE endpoint\*\* for removing a log that’s no longer needed.

- \*\*Data Storage\*\*: For simplicity, we’ll start by storing health logs in a \*\*JSON file\*\*. JSON is a lightweight way to save data, and Node.js can read from and write to this file efficiently. In the future, we could switch to a more advanced database if needed.

4. \*\*Data Flow and CRUD Operations\*\*:

- CRUD stands for \*\*Create, Read, Update, and Delete\*\*. These operations form the backbone of our app:

- \*\*Create\*\*: When the user adds a new health entry, the frontend sends a “Create” request to the backend.

- \*\*Read\*\*: When the user wants to see their logs, the frontend sends a “Read” request, and the backend responds with the stored data.

- \*\*Update\*\*: If the user wants to edit a past log, the frontend sends an “Update” request with the updated details.

- \*\*Delete\*\*: If the user no longer needs a log, they can delete it, and the frontend sends a “Delete” request to the backend to remove it from storage.

5. \*\*Testing\*\*:

- We will test the backend’s functionality with unit tests, ensuring each CRUD operation works as expected. We’ll also do integration testing, confirming the frontend and backend work together smoothly.

6. \*\*Version Control and Documentation\*\*:

- \*\*Git and GitHub\*\* will track every change, so the development process is transparent. Each step, from setting up the server to adding features, will be recorded in GitHub.

- \*\*Documentation\*\* will be created in Google Docs, detailing each part of the project, how it works, and any external resources used.

---

### Why This Structure Works Well

- \*\*API Architecture\*\*: This setup allows a modern approach to app building, where the frontend and backend are independent but communicate through a well-defined API. This makes the system flexible and scalable.

- \*\*Simple Data Management\*\*: Using JSON for storage keeps things straightforward. JSON is readable by both humans and code, making it a good choice for initial development.

- \*\*Easily Testable\*\*: By building the backend with clearly defined functions for each CRUD operation, we can test each part separately. This improves reliability and makes sure everything works together.

---

This explanation shows your understanding of the project structure and why each part of it is essential to the overall functionality. Let me know if you need more details or examples for any specific section!