**Fitness Tracker Web Application**

**Introduction:**

This project is a React-based frontend application designed to help users track their fitness activities, including workouts, steps, calorie intake, and other health metrics. It uses Vite as the build tool for fast development and optimized production builds. The project also integrates Tailwind CSS for styling and leverages ESLint and Prettier to maintain code quality and consistent formatting.

**Features Overview:**

### **User Goals:**

* Set daily goals for various health metrics such as:
  + **Workouts**
  + **Calories Intake/Burned**
  + **Water Intake**
  + **Steps**
  + **Sleep Duration**
* These goals are fetched from the server and displayed in interactive forms. Users can edit, update, or delete goals.

### **Progress Visualization:**

* View daily, weekly, and monthly progress for fitness metrics through:
  + **Bar Charts**
  + **Line Charts**
  + **Progress Cards**

### **Responsive Design:**

* Fully responsive layout using Tailwind CSS, ensuring compatibility across devices.

### **Theming:**

* Switch between **Light** and **Dark** themes with the click of a button using the ThemeSwitcher component.

**Project Structure Details:**

**1. Main Application Structure**

**- `App.jsx`**: Root component encapsulating the app's structure, routing, and global state management.

**- `main.jsx`**: Entry point where the React DOM rendering occurs, rendering the App component.

**2. Goals Module**

- Components such as `CalBurned.jsx`, `CalIntake.jsx`, `Sleep.jsx`, `Steps.jsx`, `Water.jsx`, and `TargetWeight.jsx` handle specific fitness goals. They receive data from a higher-level component like `GoalsPage.jsx` and trigger events like updating a goal.

- `GoalsPage.jsx`: Parent component that manages all individual goal components.

* Src\components\goals\CalBurned.jsx :

Page for setting and saving daily burn goal. Fetches the goal from the DB

when mounted and updates the DB when the form is submitted.

@returns JSX for the page

* Src\components\goals\CalIntake.jsx :

Component for setting and saving daily calorie intake, goal Fetches the existing intake goal from the server and populates the form with the data If the goal doesn't exist, it creates a new one Handles saving the goal to the server and navigating to the Goals page after saving

@returns A JSX element representing the component

* Src\components\goals\GoalsPage.jsx :

The GoalsPage component renders a page with a list of user's goals.

The user can delete any of the goals by clicking on the delete button and confirming the deletion in the modal window.

The component fetches the goals data from the backend API on mount.

The data is stored in the component state and re-rendered when the user deletes a goal.

* Src\components\goals\Sleep.jsx :

Sleep component allows users to set and save their daily sleep goal.

It retrieves the existing sleep goal data from the server and populates the form if available.

Users can input the number of sleep hours and start date for their goal.

On saving, the goal is either updated or created on the server, and the user is navigated to the Goals page.

Utilizes react-router-dom for navigation and axios for server communication.

* Src\components\goals\Steps.jsx :

Component for setting and saving daily steps goal.

Fetches the existing steps goal from the server and populates the form with the data.

If the goal doesn't exist, it creates a new one.

Handles saving the goal to the server and navigating to the Goals page after saving.

@returns A JSX element representing the component

* Src\components\goals\Water.jsx :

Component for setting and saving daily water intake goal.

It retrieves the existing water goal data from the server and populates the form if available.

Users can input the number of liters and start date for their goal.

On saving, the goal is either updated or created on the server, and the user is navigated to the Goals page.

Utilizes react-router-dom for navigation and axios for server communication.

* Src\components\goals\WorkoutForm.jsx :

This component renders a form for setting a workout goal. It fetches the existing goal if there is one, and allows the user to update it. If not, it allows the user to create a new goal. The form includes select fields for the goal period, quantity, and duration, as well as a start date field. When the user submits the form, the component sends a PUT or POST request to the server to update or create the goal, respectively. If the request is successful, the component redirects the user to the goals page.

@returns {JSX.Element} The rendered form component.

**3. Home Module**

**- `GoalsTab.jsx`**: Displays user goals in a tabbed format.

**- `Progress.jsx`**: Visualizes overall fitness progress, aggregating data displayed through `BarChart.jsx`, `LineChart.jsx`,ProgressCard.jsx`, and `ProgressPreview.jsx`.

* Src\components\home\GoalsTab.jsx :

GoalsTab component renders a tab for the Goals page on the home page.

It initially renders a message saying "You did not set any goals yet!" and a button that navigates to the goals page to set goals.

If the user already has goals, the goals will be displayed instead of the message saying "You did not set any goals yet!"

The component also includes a button to "act as there are no goals" and "act as there are goals" which can be used for testing purposes.

* Src\components\home\Progress.jsx :

A page component for displaying progress and goals.

Renders a tabbed component with two tabs, "goals" and "progress". The "goals" tab displays the GoalsTab component.

The "progress" tab displays a select component for selecting a period ("daily", "weekly", "monthly", "yearly"), and

a list of ProgressCard components that display the logged data for that period.

The period state is stored in the component state and is used to conditionally render the ProgressCard components.

@returns {JSX.Element} A JSX element representing the component.

* Src\components\home\progress\ProgressCard.jsx:

A card component that renders a progress card for the given card props.

It fetches the data based on the period and renders the data in a card format.

It also renders a preview of the data in a given period.

@param {Object} card - The card data object which includes id, title, desc, icon, and unit.

@param {String} period - The period for which the data should be fetched.

@returns {ReactElement} A ReactElement that renders a progress card.

* Src\components\home\progress\BarChart.jsx:

A bar chart component for displaying logged data by period.

@param {string} period The period of data to display. One of "weekly", "monthly", or "yearly".

@param {string} id The id of the log to display.

@param {Object} dataSets An object containing the logged data. The keys are the log ids and the values are arrays of numbers.

@returns {JSX.Element} A bar chart component.

* Src\components\home\progress\LineChart.jsx:

A line chart component for displaying logged data by period.

@param {string} period The period of data to display. One of "weekly", "monthly", or "yearly".

@param {string} id The id of the log to display.

@param {Object} dataSets An object containing the logged data. The keys are the log ids and the values are arrays of numbers.

@returns {JSX.Element} A line chart component.

* Src\components\home\progress\ProgressPreview.jsx:

A component that displays a preview of a log's progress in a dialog.

It displays a bar chart and a line chart for the given period and id.

It also displays the given date range and a title that is the given period and id.

The component is a trigger for a dialog that wraps the content.

@param {string} period The period of data to display. One of "weekly", "monthly", or "yearly".

@param {string} id The id of the log to display.

@param {Object} dateRange An object containing the start and end dates of the period.

@param {Object} dataSets An object containing the logged data. The keys are the log ids and the values are arrays of numbers.

@returns {JSX.Element} A component that displays a preview of a log's progress in a dialog.

**4. Logs Module**

Components that display logged user data, fetching and showing calorie intake, workouts, and more.

* Src\components\logs\CaloriesLog.jsx :

Component to log calories intake or burned.

It fetches the existing calories logs from the server and populates the form with the data.

If the log doesn't exist, it creates a new one.

Handles saving the log to the server and navigating to the previous page after saving.

Utilizes react-router-dom for navigation and axios for server communication.

@returns A JSX element representing the component

* Src\components\logs\SelectWorkout.jsx :

SelectWorkout component allows users to search and filter workouts.

It fetches workout data from an API and provides functionality to filter workouts

based on user input. The component also handles loading and error states and

displays the filtered or complete list of workouts using WorkoutCard components.

@returns {JSX.Element} The rendered SelectWorkout component.

* Src\components\logs\SearchComponent.jsx :

SearchComponent component for searching workouts.

@param {object} props Component props.

@param {string} [props.placeholder] Placeholder text for the search input.

@returns {ReactElement} The rendered SearchComponent.

* Src\components\logs\LogMenu.jsx :

A menu component for logs, which allows user to select an activity to log (workout, sleep, weight, water, steps, calories).

@returns The LogMenu component

* Src\components\logs\SetInput.jsx :

SetInput - A form component for logging workouts based on sets and reps.

@param {string} watchOption - The selected option (sets or duration)

@param {object} formSchema - The schema for the form

@returns {JSX.Element} The rendered form component

* Src\components\logs\SleepLog.jsx :

SleepLog is a component that logs the user's sleep hours on a given date.

It renders a form with input fields for sleep hours and log date.

When the form is submitted, it sends a POST request to the server with the log data.

After a successful submission, it displays a success message and resets the form fields.

If there is an error, it displays an error message.

The component also has a back button that navigates the user back to the previous page.

@returns {JSX.Element}

* Src\components\logs\StepsLog.jsx :

StepsLog is a component that logs the user's steps on a given date.

It renders a form with input fields for steps count and log date.

When the form is submitted, it sends a POST request to the server with the log data.

After a successful submission, it displays a success message and resets the form fields.

If there is an error, it displays an error message.

The component also has a back button that navigates the user back to the previous page.

@returns {JSX.Element}

* Src\components\logs\LogsTable.jsx :

LogsTable component is responsible for displaying logs for a specific activity (e.g., weight, sleep, workouts).

It displays the logs as cards and shows a message if no logs are found.

@param {Array} logs - Array of logs data to display

@param {String} tableHead1 - Type of the activity (e.g., weight, steps, sleep)

@param {Boolean} loading - Flag to show if data is being loaded

@param {Boolean} error - Error flag or message if there's an issue loading logs

* Src\components\logs\WorkoutCard.jsx :

WorkoutCard displays information for a specific workout.

@param {Object} workout - The workout data.

@param {String} userId - The current user's ID.

* Src\components\logs\WorkoutDetails.jsx :

WorkoutDetails component to display detailed information of a specific workout.

Fetches the workout details from the API using the workoutId.

Stores the fetched data in the state and sets loading to false.

If an error occurs, logs the error and sets the error state.

* Src\components\logs\WorkoutLogForm.jsx :

WorkoutLogForm - A form for logging a workout based on sets or duration.

@returns {ReactElement} A form with input fields for logging a workout.

* Src\components\logs\WeightLog.jsx :

Component to log weight in kgs.

It fetches the existing weight logs from the server and populates the form with the data.

If the log doesn't exist, it creates a new one.

Handles saving the log to the server and navigating to the previous page after saving.

Utilizes react-router-dom for navigation and axios for server communication.

@returns A JSX element representing the component

* Src\components\logs\WaterLog.jsx:

WaterLog is a component that logs the user's water intake on a given date.

It renders a form with input fields for water liters and log date.

When the form is submitted, it sends a POST request to the server with the log data.

After a successful submission, it displays a success message and resets the form fields.

If there is an error, it displays an error message.

The component also has a back button that navigates the user back to the previous page.

@returns {JSX.Element}

* src\components\logs\workoutloggingForms\DurationBasedForm.jsx:

Duration-based form component for logging a workout based on duration.

* Src\components\logs\workoutloggingForms\SetsBasedForm.jsx:

SetsBasedForm - A form to log a workout based on sets and reps.

* src\components\logs\workoutloggingForms\WorkoutLogForm.jsx

WorkoutLogForm - Main form to log workout based on either duration or sets/reps

**5. Layout Components**

- Components like `BusinessLogo.jsx`, `Header.jsx`, `Layout.jsx`, `Sidebar.jsx`, and `UserMenu.jsx` manage the structure and layout.

- `ThemeSwitcher.jsx` and `ToggleSideMenu.jsx` provide user interaction elements for toggling themes or expanding/collapsing the sidebar.

* Src\components\layout\Header.jsx :

Renders the header component of the application.

This component includes a link to the Dashboard, a theme switcher, and a user menu. It is styled with a border, shadow, and spacing to ensure proper alignment and presentation.

* Src\components\layout\Layout.jsx :

The main layout component for the app.

This component is responsible for rendering the main layout of the app, which includes the sidebar, header, and main content area. The main content area is where the routes are rendered.

@param {React.ReactNode} children The children of the component, which should be the routes.

@returns {React.ReactElement} The main layout component.

* Src\components\layout\Sidebar.jsx:

Sidebar component responsible for rendering the sidebar with toggle functionality and side menu buttons.

* Src\components\layout\BusinessLogo.jsx:

Renders the business logo with an avatar, image, and fallback content.

* Src\components\layout\ToggleSideMenu.jsx:

ToggleSideMenu

@description Toggle button to open/close side menu

@param {boolean} isOpen - Whether the side menu is open or not

@param {(isOpen: boolean) => void} setIsOpen - Function to set `isOpen` value

@returns {JSX.Element} A toggle button

* Src\components\layout\ThemeSwitcher.jsx:

A button to toggle between light and dark theme.

@returns {ReactElement} A button with a moon or sun icon to toggle theme.

* Src\components\layout\SideBarNav.jsx :

A component that renders a sidebar navigation with buttons.

@param {{ sideMenuBtns: { title: string, href: string, icon: ReactNode }[], isOpen: boolean }} props

@returns {JSX.Element}

* Src\components\layout\UserMenu.jsx:

A component that shows a user menu when you click on the avatar.

it will show user controls like sign out and profile page if we will do one.

**6. Shared Library**

`ThemeContext.jsx`: Manages the theme context of the application, providing methods to toggle between light and dark modes.

* Src\components\goals\shared\GoalCard.jsx:

A component that shows a user menu when you click on the avatar.

it will show user controls like sign out and profile page if we will do one.

* Src\components\goals\shared\GoalCard.jsx:

GoalCard component displays a card for each goal with detailed information such as goal type, quantity, duration, and target values.

It allows users to edit, delete, and view details of each goal.

* Src\components\goals\shared\ConfirmModal.jsx:

A modal that prompts the user to confirm a destructive action.

@param {{ showModal: boolean, onConfirm: Function, onCancel: Function }}

showModal - Whether the modal should be shown.

onConfirm - The function to call when the user confirms the action.

onCancel - The function to call when the user cancels the action.

* Src\components\goals\shared\GoalsPreview.jsx:

This component renders a preview of the goal details. It takes the goalType and goal as props from the route state.

It renders a card with the goal details in a format that depends on the goalType.

If no goal data is available (i.e., the user has not set any goals yet), it will render a message saying so.

It also includes a button to go back to the goals page.

* Src\components\goals\shared\NavLinks.jsx:

Renders a navigation menu with links to all goal setting pages.

@returns The navigation menu component.

**7. UI Components**

- Reusable utility components such as `dialog.jsx`, `drawer.jsx`, `dropdown- menu.jsx`, `input.jsx`, `label.jsx`, and `textarea.jsx` offer common UI functionalities.

**Installation:**

1. **Clone the repository:**

**Git clone** [**https://github.com/Depi-Graduation-Project/Project-FrontEnd.git**](https://github.com/Depi-Graduation-Project/Project-FrontEnd.git)

cd Project-FrontEnd

1. **Install dependencies**:

npm install

1. **Running the Application**

To start the development server, run the following command:

npm run dev

The app will be accessible at <http://localhost:3000>.

1. **Building the Application**

To build the project for production, use the following command:

npm run build

This will generate a dist folder with the optimized production build.

1. **Testing the Application**

The project includes some mock data files to simulate user activities. You can modify the mock data in the following files:

* fakeData.jsx
* fakeLoggedData.jsx
* fakeLoggedWorkouts.jsx

This allows for development and testing without the need for a live backend.

**Configuration Files**

* **vite.config.js**: Configuration for Vite's development and build setup.
* **eslint.config.js**: Enforces coding standards using ESLint.
* **tailwind.config.js**: Configuration file for Tailwind CSS.
* **.prettierrc**: Configures Prettier for consistent code formatting.

**Contribution**

Feel free to fork this project, open issues, or submit pull requests. Contributions are welcome to improve the app!

**License**

This project is licensed under the MIT License.