Fitflex - Fitness App

1 . INTRODUCTION

Project Title: Fitflex - Fitness App

Team Details:

Team ID : SWTID1741179768146377
Team Size : 5

Team Leader : MUGILAN G	lanmugi88@gmail.com
Team member : BALAMURUGAN G	gbalamurugan01072004@gmail.com
Team member : BHARATHRAJ V	bharathrajv2005@gmail.com
Team member : RAVIKUMAR M	ravikumarmohan2005@gmail.com
Team member : YOGANANDHAN T K	yoganandhan2005@gmail.com

FitFlex is a smart fitness app designed to enhance your workout experience. It offers an easy-to-use interface, quick search options, and a wide range of exercises for all fitness levels. Start your fitness journey with FitFlex and reach your goals with ease.

2 . PROJECT OVERVIEW

Purpose :

FitFlex is designed to provide a seamless and personalized fitness experience for users of all fitness levels. The app simplifies workout planning with an intuitive interface, dynamic search, and a diverse exercise library. It also offers customized recommendations, progress tracking, and an engaging user experience to help individuals stay motivated and achieve their fitness goals efficiently.

Goals:

- Enhance Accessibility: Offer an easy-to-use platform for fitness enthusiasts.
- Personalized Experience: Provide tailored workout recommendations.
- Comprehensive Exercise Library: Include a variety of exercises for different fitness levels.
- User Engagement: Encourage consistency and motivation in workouts.

FitFlex Frontend Features and Functionality :

- Visually Appealing Hero Section – Captures user attention with a sleek design.
- Informative About Section – Provides a brief overview of FitFlex and its benefits.
- Interactive Search Bar – Enables users to find exercises efficiently using the DBExercise API.
- API-Powered Exercise Data – Fetches and displays workout details dynamically.
- Well-Structured Footer – Includes essential links and information.
- Responsive and Modern Design – Ensures a smooth user experience across devices.

3 . ARCHITECTURE

Component Structure :

The FitFlex website follows a component-based architecture, which breaks the UI into reusable and manageable parts. This approach improves code readability and reusability. The major components and their interactions are outlined below:

1. App Component
   - The root component of the application.
   - Renders the main structure of the website, including the header, hero section, about section, search bar, and footer.
   - Manages routing (if implemented).

2. HeroSection Component

- Displays the main introduction of FitFlex with a visually appealing design.
- Provides a call-to-action (e.g., encouraging users to explore exercises).
- May contain buttons or links to direct users to the search feature.

3. AboutSection Component

- Offers information about FitFlex, its purpose, and key features.
- Helps users understand the benefits of using the platform.
- Typically includes static content but may have interactive elements.

4. SearchBar Component

- Allows users to input exercise names or keywords.
- Sends requests to the DBExercise API to fetch exercise data.
- Calls a function to update the displayed results dynamically.

5. ExerciseResults Component

- Displays the search results retrieved from the API.
- Each result includes an exercise name, description, and possibly an image or video.
- Uses React state to update and show results in real-time.

6. Footer Component

- Contains essential links, contact information, and credits.
- Ensures a complete and professional website structure.

Component Structure :

The App component renders the layout and includes all the major sections.

The SearchBar component accepts user input and triggers an API call.

The ExerciseResults component receives the API response and updates the UI.

The HeroSection, AboutSection, and Footer components provide static but well-structured content.

This modular approach ensures that each component has a single responsibility, making the project easier to manage and scale.

State Management :

State management is a crucial aspect of any React application, including FitFlex. It helps maintain and update data across components efficiently. FitFlex uses both global state and local state to handle different types of data.

Global State Management in FitFlex

Global state refers to data that needs to be accessed by multiple components throughout the application. Managing global state effectively ensures that the application remains scalable and responsive.

Local State Management in FitFlex

Local state is data that is only relevant to a specific component. It does not need to be shared across multiple components and is typically handled using useState or useEffect.

4 . SETUP INSTRUCTIONS

Prerequisites :

Before setting up FitFlex, ensure that your system meets the necessary requirements. The following software dependencies are required:

1.Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications. Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

- Download: https://nodejs.org/en/download/
- Installation instructions: https://nodejs.org/en/download/package-manager/

2.React.js :

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications. Install React.js, a JavaScript library for building user interfaces.

Step 1: Install Vite globally (optional) => npm install -g create-vite

Step 2: Create a new project => npm create vite@latest my-project

Step 3: Navigate to our project folder and install dependencies

=> cd my-project

=> npm install

Step 4: Start the development server => npm run dev

Installation :

1. Clone the Repository

Open the terminal and run:

>git clone https://github.com/yourusername/FitFlex.git

>cd FitFlex

2. Install Dependencies

Use npm or yarn to install required packages:

>npm install or >yarn install

3.Configure Environment Variables

Create a .env file in the project root.

Add your DBExercise API key:

REACT_APP_DBEXERCISE_API_KEY=”your_api_key_here”

4. Start the Application

Run the project locally:

> npm start

The app will open at http://localhost:3000/.

5 . FOLDER STRUCTURE

FitFlex follows a structured approach to organizing its React frontend to maintain scalability and readability. The project is divided into various directories, each serving a specific purpose. Below is the breakdown of the folder structure:

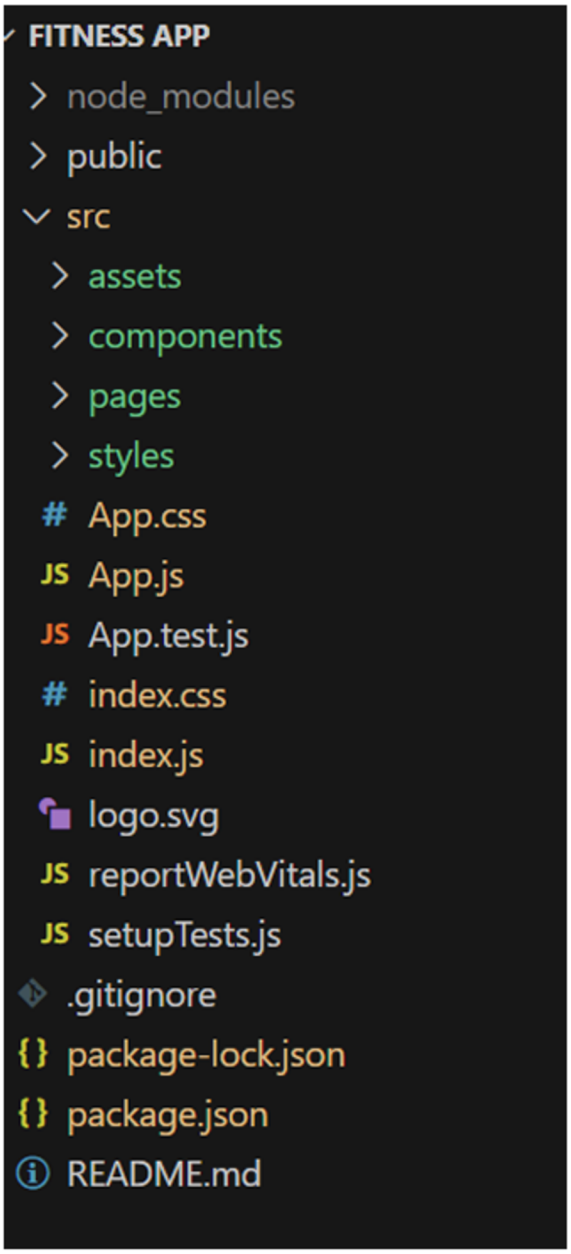
Client Directory (Frontend Application) :

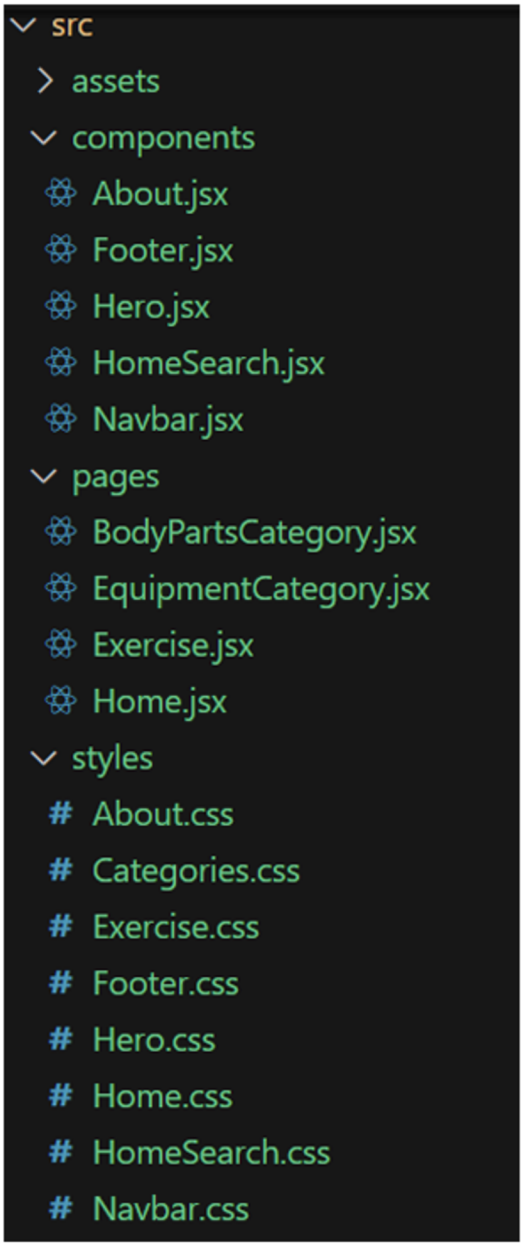
The frontend is located in the client/ directory, containing all source files necessary for building the user interface.

- src/ – This is the main source directory, containing all application logic and assets.
- components/ – This folder includes reusable UI components such as buttons, search bars, and headers.
- pages/ – Each major view of the application (like the homepage or exercise details page) is stored here.
- assets/ – Stores static resources such as images, icons, and fonts.
- utils/ – Contains utility functions and helper scripts.
- styles/ – Holds global CSS files and styling configurations.
- This well-organized structure ensures modularity, making it easy to maintain and scale the project.

Utilities :

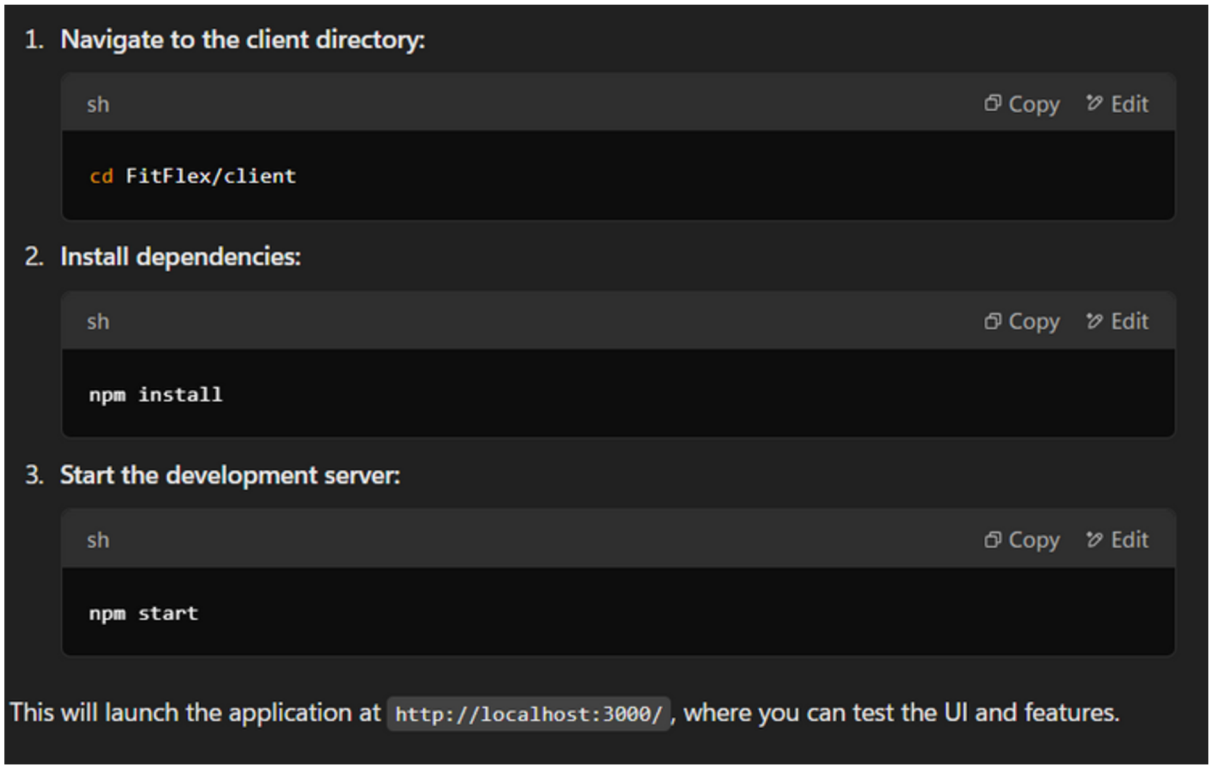
- The utils/ directory plays a vital role in the project by storing utility functions and custom hooks that enhance code reusability and maintainability.
- Key Utilities
- fetchExercises.js – This file contains functions that handle API requests to fetch exercise data dynamically.
- useFetch.js – A custom React hook for fetching data asynchronously, handling loading states, and managing errors.
- These utility functions reduce code duplication and ensure that API interactions are handled efficiently.


FITNESS APP
node_modules
public
src
assets
components
pages
styles
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md


src
assets
components
About.jsx
Footer.jsx
Hero.jsx
HomeSearch.jsx
Navbar.jsx
pages
BodyPartsCategory.jsx
EquipmentCategory.jsx
Exercise.jsx
Home.jsx
styles
About.css
Categories.css
Exercise.css
Footer.css
Hero.css
Home.css
HomeSearch.css
Navbar.css

6 . RUNNING THE APPLICATION

- The utils/ directory plays a vital role in the project by storing utility functions and custom hooks that enhance code reusability and maintainability.
- Key Utilities
- fetchExercises.js – This file contains functions that handle API requests to fetch exercise data dynamically.
- useFetch.js – A custom React hook for fetching data asynchronously, handling loading states, and managing errors.
- These utility functions reduce code duplication and ensure that API interactions are handled efficiently.



7 . COMPONENT DOCUMENTATION

The App component renders the layout and includes all the major sections. The SearchBar component accepts user input and triggers an API call. The ExerciseResults component receives the API response and updates the UI.The HeroSection, AboutSection, and Footer components provide static but well-structured content. This modular approach ensures that each component has a single responsibility, making the project easier to manage and scale.

Key Components:

- HeroSection – Displays the homepage banner and introductory content.
- SearchBar – Allows users to input queries to search for exercises.
- ExerciseResults – Dynamically fetches and displays exercise details.
- Footer – Contains essential site links and credits.

Reusable Components :

- Button – A customizable button component used across multiple pages.
- Card – Displays exercise details in a structured format.

8 . STATE MANAGEMENT

State Management :

State management is a crucial aspect of any React application, including FitFlex. It helps maintain and update data across components efficiently. FitFlex uses both global state and local state to handle different types of data.

Global State Management in FitFlex

Global state refers to data that needs to be accessed by multiple components throughout the application. Managing global state effectively ensures that the application remains scalable and responsive.

Why Use Global State?

In FitFlex, certain pieces of data, such as user preferences, search results, and API responses, might be needed in different parts of the application. Instead of passing this data through props, which can lead to "prop drilling," a centralized state management approach is more efficient.
React Context API for Global State

FitFlex utilizes the React Context API for managing global state. The Context API allows data to be shared between components without manually passing it down through each level of the component tree.

Local State Management

Local state is data that is only relevant to a specific component. It does not need to be shared across multiple components and is typically handled using useState or useEffect.

Why Use Local State?

- To manage UI changes like button clicks, input field values, or loading states.
- To store temporary data that doesn't need to persist globally.
- To keep components self-contained and independent.

```
import React, { useState } from "react";

const SearchBar = () => {
  const [query, setQuery] = useState(""); // Local state

  const handleSearch = (e) => {
    setQuery(e.target.value);
  };

  return (
    <input
      type="text"
      placeholder="Search exercises..."
      value={query}
      onChange={handleSearch}
    />
  );
};
```

Handling API Responses with useEffect

FitFlex fetches exercise data from the DBExercise API. Since API responses depend on external servers and user interactions, useEffect is used to handle API calls dynamically.

Example: Fetching Data from API

How It Works

1. useState([]) initializes exercises as an empty array.
2. useEffect(() => {...}, []) runs once when the component mounts, triggering the API call.
3. The API response updates the state using setExercises(data).
4. The UI re-renders to display the fetched exercise data.

Choosing Between Global and Local State

Feature	Global State (Context API)	Local State (`useState`)
Scope	Shared across multiple components	Limited to a single component
Best Used For	Search results, user preferences, API responses	UI interactions, form inputs, component-specific data
Performance Impact	Can trigger multiple re-renders if not optimized	Re-renders only the component using it
Complexity	Requires setup (Provider, Consumer)	Simple and easy to use

9 . USER INTERFACE

The UI of FitFlex is designed to be simple yet functional. Key features include:

- A search bar to find exercises.
- A results section displaying fetched exercises.
- A clean and responsive layout with a structured hero section and footer.

After completing the code, run the react application by using the command “npm start” or “npm run dev” if you are using vite.js

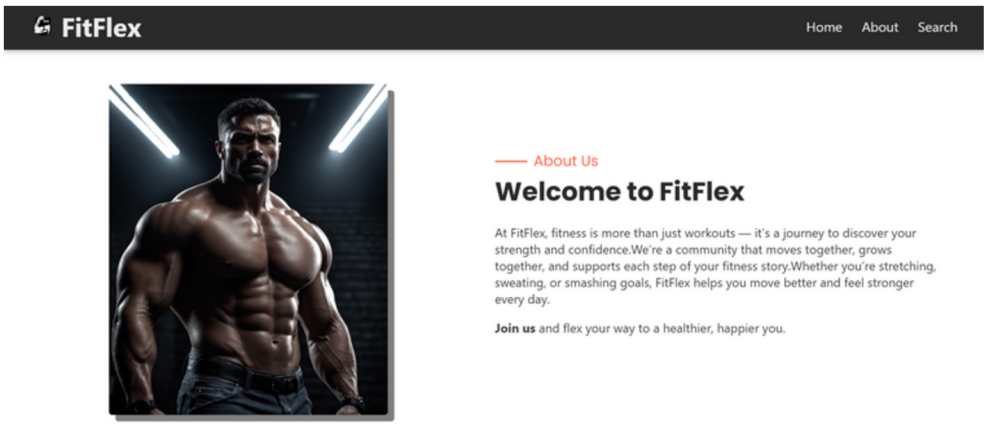
Here are some of the screenshots of the application.

Hero component :

this section would showcase trending workouts or fitness challenges to grab users' attention.

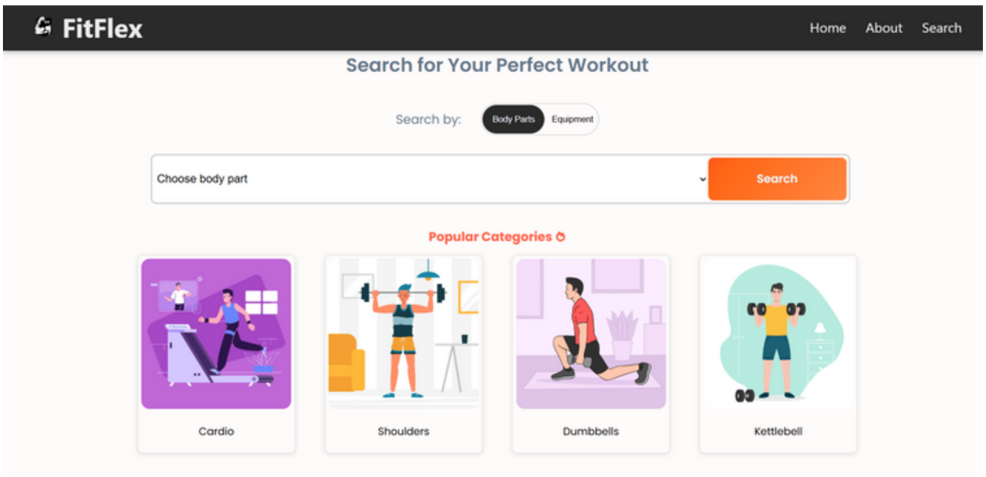
About component :

FitFlex isn't just another fitness app. We're meticulously designed to transform your workout experience, no matter your fitness background or goals.



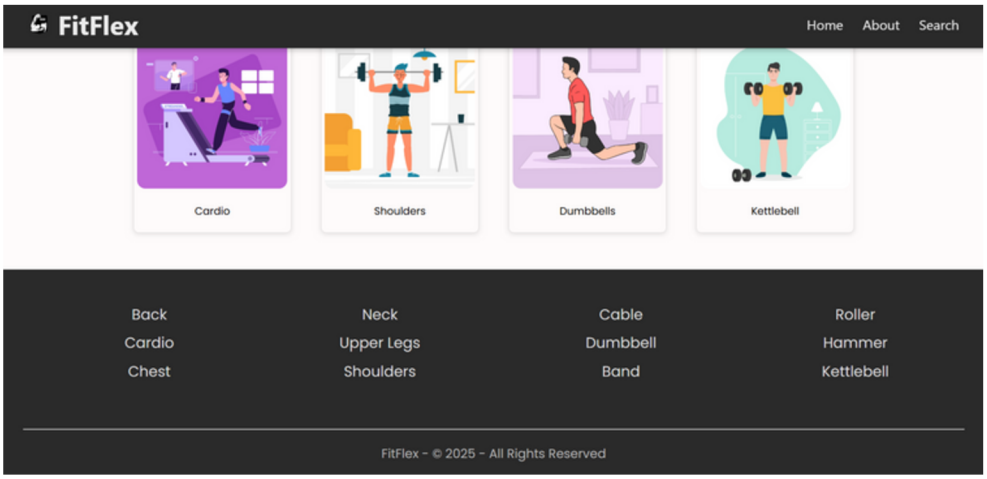
SearchBar :

Fitflex makes finding your perfect workout effortless. Our prominent search bar empowers you to explore exercises by keyword, targeted muscle group, fitness level, equipment needs, or any other relevant criteria you have in mind. Simply type in your search term and let FitFlex guide you to the ideal workout for your goals.



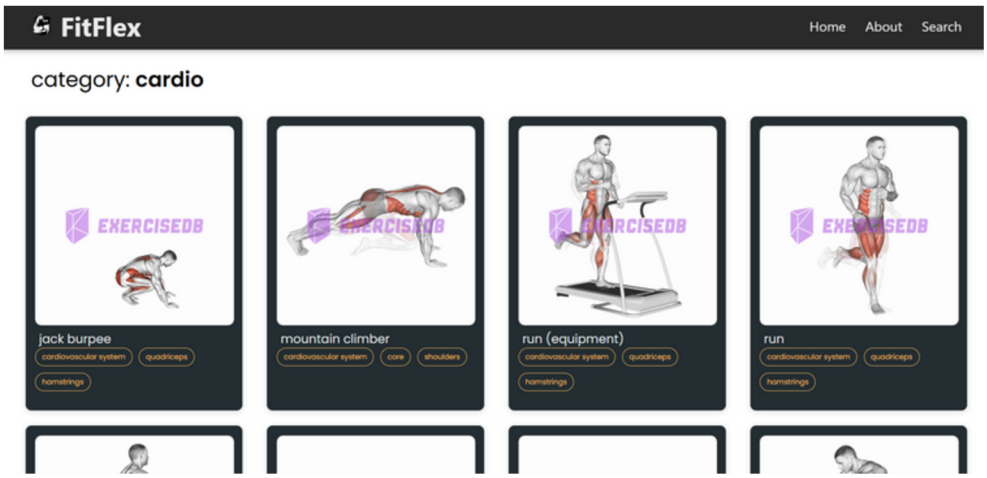
Footer :

Fitflex have the interactive footer that make us easy to access the exercises



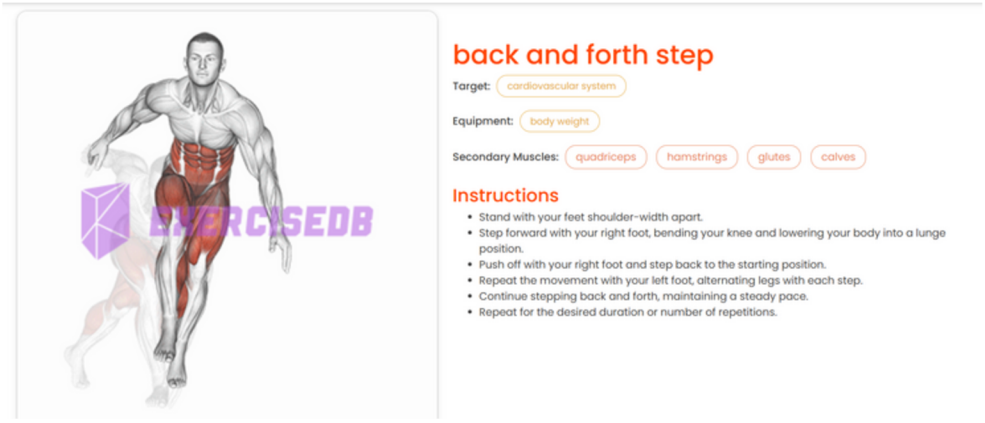
Category page :

FitFlex would offer a dedicated section for browsing various workout categories.This could be a grid layout with tiles showcasing different exercise types (e.g., cardio, strength training, yoga) with icons or short descriptions for easy identification.



Exercise:

This is where the magic happens! Each exercise page on FitFlex provides a comprehensive overview of the chosen workout.Expect clear and concise instructions, accompanied by high-quality visuals like photos or videos demonstrating proper form. Additional details like targeted muscle groups, difficulty level, and equipment requirements (if any) will ensure you have all the information needed for a safe and effective workout.



Demo link:
https://drive.google.com/drive/folders/1j8P_8jXMNTWN2_l21VHJjU0bnyMOGwnl

10 . STYLING

CSS Frameworks/Libraries

FitFlex primarily uses CSS for styling but also incorporates Styled Components for modular styles.

Theming :

A global theme provider can be implemented to introduce dark and light mode support in the future.

11 . TESTING

Testing Strategy :

FitFlex primarily uses CSS for styling but also incorporates Styled Components for modular styles.

Code Coverage :

A global theme provider can be implemented to introduce dark and light mode support in the future.

12 . DEMO

Project demo:

Before starting to work on this project, let’s see the demo.
Demolink:
https://drive.google.com/drive/folders/1j8P_8jXMNTWN2_l21VHJjU0bnyMOGwnl

Use the code in:

https://drive.google.com/drive/folders/1j8P_8jXMNTWN2_l21VHJjU0bnyMOGwnl

13 . KNOWN ISSUES

Some API requests may take longer than expected due to external server response time.

Mobile responsiveness can be further improved in some sections.

14 . FUTURE ENCHANTMENTS

Implement dark mode for better accessibility.

Add filtering options for exercises based on difficulty level.

Improve animations and transitions for a smoother user experience.

THANK YOU