**Introduction:**

**Project Title: FitFlex**

**Team Members:**

**Sneha D : Team Leader**

**Mail:snehaselvi71013@gmail.com**

**Dhivya M: Frontend Developer**

**Mail:** **m.dhivya0531@gmail.com**

**Gayathri V: Frontend Developer**

**Mail:gayathrichithra08@gmail.com**

**Birundha B: Rapid Api passkey connector**

**Mail:** **birunda2005@gmail.com**

**Keerthika E:Document creator**

**Mail:** **ekeerthika226@gmail.com**

**Project Overview: FitFlex –** Your Personal Fitness Companion

**Purpose of FitFlex:**

FitFlex aims to provide an accessible, user-friendly, and comprehensive fitness solution that enhances workout experiences through technology. The primary goals include:  
✔ User-Friendly Experience – A clean and modern UI that enables effortless navigation.  
✔ Comprehensive Exercise Management – Tools to explore, save, and customize workout plans.  
✔ Advanced Search and Discovery – A powerful search feature that helps users find specific exercises based on fitness preferences.  
✔ Seamless Integration with APIs – Fetch exercises from RapidAPI and potentially integrate fitness trackers, nutrition data, or workout tracking.  
✔ Engagement and Motivation – A community-driven approach where users can explore trending workouts and fitness challenges.

**Key Features of FitFlex:**

✔ Extensive Exercise Library – Access a diverse range of workouts from external fitness APIs.  
✔ Dynamic Visual Exploration – Users can browse workout routines through images and videos.  
✔ Advanced Search Functionality – Search exercises by category, muscle group, fitness level, and equipment needs.  
✔ Personalized Recommendations – Tailored workout plans based on user preferences.  
✔ Workout Details Page – In-depth descriptions, visual guides, and related YouTube videos for each exercise.  
✔ Community Engagement – Support for collaboration, sharing workout plans, and interacting with other fitness enthusiasts.

**📌 Architecture Overview of FitFlex:**

FitFlex is built using **React.js** and follows a **component-based architecture** with structured routing and state management to ensure an efficient and seamless user experience.

**1️⃣ Component Structure:**

FitFlex follows a **modular approach** by dividing the application into three main folders:

bash

CopyEdit

/fitness-app-react

│── /src

│ ├── /components

│ │ ├── Navbar.js

│ │ ├── Hero.js

│ │ ├── SearchBar.js

│ │ ├── CategoryCard.js

│ │ ├── ExerciseCard.js

│ │ ├── ExerciseDetails.js

│ │ ├── Footer.js

│ │ ├── Subscribe.js

│ │

│ ├── /pages

│ │ ├── Home.js

│ │ ├── CategoryPage.js

│ │ ├── ExercisePage.js

│ │ ├── NotFound.js

│ │

│ ├── /styles

│ │ ├── global.css

│ │ ├── components.css

│ │ ├── responsive.css

│ │

│ ├── /context

│ │ ├── ExerciseContext.js

│ │

│ ├── /api

│ │ ├── fetchExercises.js

│ │ ├── fetchCategories.js

│ │ ├── fetchExerciseDetails.js

│ │

│ ├── App.js

│ ├── index.js

│

│── /public

│── package.json

│── README.md

**🔹 Core Components:**

* **Navbar.js** → Handles navigation between pages.
* **Hero.js** → Displays a welcome banner with trending workouts.
* **SearchBar.js** → Allows users to search for exercises dynamically.
* **CategoryCard.js** → Displays different workout categories.
* **ExerciseCard.js** → Shows details of individual exercises.
* **ExerciseDetails.js** → Provides instructions, related videos, and other details.
* **Footer.js** → App footer with links and information.
* **Subscribe.js** → Allows users to subscribe to newsletters.

**2️⃣ State Management (Using Context API):**

FitFlex uses **React Context API** for state management. It helps manage **exercise data, user selections, and search queries** globally across the app.

**🔹 Exercise Context API (ExerciseContext.js)**

jsx

CopyEdit

import { createContext, useState, useEffect } from 'react';

import { fetchExercises } from '../api/fetchExercises';

export const ExerciseContext = createContext();

export const ExerciseProvider = ({ children }) => {

const [exercises, setExercises] = useState([]);

const [loading, setLoading] = useState(true);

useEffect(() => {

fetchExercises()

.then((data) => {

setExercises(data);

setLoading(false);

})

.catch((error) => console.error(error));

}, []);

return (

<ExerciseContext.Provider value={{ exercises, loading }}>

{children}

</ExerciseContext.Provider>

);

};

**🔹 How It’s Used in a Component**

jsx

CopyEdit

import { useContext } from 'react';

import { ExerciseContext } from '../context/ExerciseContext';

const ExerciseList = () => {

const { exercises, loading } = useContext(ExerciseContext);

if (loading) return <p>Loading...</p>;

return (

<div>

{exercises.map((exercise) => (

<p key={exercise.id}>{exercise.name}</p>

))}

</div>

);

};

**3️⃣ Routing Structure (React Router)**

FitFlex uses **React Router** for navigation between different pages.

**🔹 Setup Routing in App.js**

jsx

CopyEdit

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

import Home from './pages/Home';

import CategoryPage from './pages/CategoryPage';

import ExercisePage from './pages/ExercisePage';

import NotFound from './pages/NotFound';

import Navbar from './components/Navbar';

import Footer from './components/Footer';

function App() {

return (

<Router>

<Navbar />

<Routes>

<Route path="/" element={<Home />} />

<Route path="/category/:id" element={<CategoryPage />} />

<Route path="/exercise/:id" element={<ExercisePage />} />

<Route path="\*" element={<NotFound />} />

</Routes>

<Footer />

</Router>

);

}

export default App;

**🔹 Explanation of Routes**

| **Path** | **Component** | **Purpose** |
| --- | --- | --- |
| / | Home.js | Displays homepage with search and categories |
| /category/:id | CategoryPage.js | Shows exercises in a specific category |
| /exercise/:id | ExercisePage.js | Displays detailed exercise instructions |
| \* | NotFound.js | Handles invalid URLs |

**4️⃣ API Integration & Data Fetching**

FitFlex fetches data from the **RapidAPI Exercise Database**.

**🔹 Fetch Exercises (fetchExercises.js)**

jsx

CopyEdit

import axios from 'axios';

const API\_URL = 'https://exercisedb.p.rapidapi.com/exercises';

const API\_KEY = 'YOUR\_API\_KEY';

export const fetchExercises = async () => {

try {

const response = await axios.get(API\_URL, {

headers: {

'X-RapidAPI-Key': API\_KEY,

'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com'

}

});

return response.data;

} catch (error) {

console.error('Error fetching exercises:', error);

}

};

**🔹 Fetch Exercise Details (fetchExerciseDetails.js)**

jsx

CopyEdit

export const fetchExerciseDetails = async (id) => {

try {

const response = await axios.get(`${API\_URL}/${id}`, {

headers: { 'X-RapidAPI-Key': API\_KEY }

});

return response.data;

} catch (error) {

console.error('Error fetching exercise details:', error);

}

};

**🎯 Project Flow**

1️⃣ **User lands on Home Page (Home.js)**

* Sees **Hero section** with trending exercises.
* Uses **SearchBar** to look for exercises.
* Clicks on a **CategoryCard** to explore workouts.

2️⃣ **User visits a Category (CategoryPage.js)**

* Displays a list of exercises under that category.
* Clicks on an **ExerciseCard** for details.

3️⃣ **User views an Exercise (ExercisePage.js)**

* Sees instructions, related videos, and exercise details.
* Can **bookmark** or **save** workouts for later.

**🚀 Summary**

✔ **Component-Based Architecture** → Organized into reusable UI components.  
✔ **React Context API for State Management** → Stores exercise data globally.  
✔ **React Router for Navigation** → Handles dynamic routing and page transitions.  
✔ **API Integration** → Fetches exercises dynamically from **RapidAPI**.  
✔ **Modular File Structure** → Well-organized codebase for scalability.

**📌 FitFlex Installation Guide**

**1️⃣ Prerequisites:**

Ensure the following tools are installed before setting up FitFlex:  
✔ **Node.js & npm** → [Download Here](https://nodejs.org/en/download/)  
✔ **Git** → [Download Here](https://git-scm.com/downloads)  
✔ **Code Editor** (VS Code Recommended) → [Download Here](https://code.visualstudio.com/download)  
✔ **Basic Knowledge** of HTML, CSS, JavaScript, and React.js

**2️⃣ Installation Steps:**

**🔹 Step 1: Clone the Repository**

bash

CopyEdit

git clone https://github.com/your-repo/fitness-app-react.git

cd fitness-app-react

**🔹 Step 2: Install Dependencies**

bash

CopyEdit

npm install

**🔹 Step 3: Start the Development Server**

bash

CopyEdit

npm start

**Access the App** → Open <http://localhost:3000> in your browser.

**📁 FitFlex Folder Structure:**

FitFlex follows a **structured and modular** approach with separate directories for components, pages, utilities, API calls, and state management.

bash

CopyEdit

/fitness-app-react

│── /client # Frontend (React.js)

│ ├── /src

│ │ ├── /components # Reusable UI Components

│ │ │ ├── Navbar.js

│ │ │ ├── Hero.js

│ │ │ ├── SearchBar.js

│ │ │ ├── CategoryCard.js

│ │ │ ├── ExerciseCard.js

│ │ │ ├── ExerciseDetails.js

│ │ │ ├── Footer.js

│ │ │ ├── Subscribe.js

│ │ │

│ │ ├── /pages # Page Components

│ │ │ ├── Home.js

│ │ │ ├── CategoryPage.js

│ │ │ ├── ExercisePage.js

│ │ │ ├── NotFound.js

│ │ │

│ │ ├── /context # State Management (React Context API)

│ │ │ ├── ExerciseContext.js

│ │ │

│ │ ├── /api # API Calls

│ │ │ ├── fetchExercises.js

│ │ │ ├── fetchCategories.js

│ │ │ ├── fetchExerciseDetails.js

│ │ │

│ │ ├── /utilities # Helper Functions & Configurations

│ │ │ ├── helpers.js

│ │ │ ├── constants.js

│ │ │

│ │ ├── /styles # CSS Styling

│ │ │ ├── global.css

│ │ │ ├── components.css

│ │ │ ├── responsive.css

│ │ │

│ │ ├── App.js # Main Application Component

│ │ ├── index.js # Entry Point

│ │

│ ├── /public # Static Files

│ ├── package.json # Dependencies

│ ├── README.md # Project Documentation

│

│── /server (Optional) # Backend (if needed in the future)

**📌 Folder Breakdown**

✔ **/client** → React frontend files.  
✔ **/components** → Reusable UI elements (Navbar, SearchBar, Cards).  
✔ **/pages** → Different pages (Home, Category, Exercise Details).  
✔ **/context** → Manages global state with Context API.  
✔ **/api** → Handles API calls (Fetching exercises, categories, details).  
✔ **/utilities** → Helper functions, constants, and configurations.  
✔ **/styles** → CSS files for global styling and responsiveness.

Would you like any specific additions like **Redux, authentication, or backend** integration? 🚀

**🚀 Running the FitFlex Application (Frontend)**

Follow these steps to set up and run the **FitFlex frontend (React.js)** application.

**1️⃣ Prerequisites**

Ensure you have the following installed before proceeding:  
✔ **Node.js & npm** → [Download Here](https://nodejs.org/en/download/)  
✔ **Git** → [Download Here](https://git-scm.com/downloads)  
✔ **Code Editor (VS Code Recommended)** → [Download Here](https://code.visualstudio.com/download)

**2️⃣ Installation & Setup**

**🔹 Step 1: Clone the Repository**

bash

CopyEdit

git clone https://github.com/your-repo/fitness-app-react.git

cd fitness-app-react/client

**🔹 Step 2: Install Dependencies**

bash

CopyEdit

npm install

**3️⃣ Running the Application**

**🔹 Step 3: Start the Development Server**

bash

CopyEdit

npm start

🔹 The app will automatically open in your browser at:  
👉 [**http://localhost:3000**](http://localhost:3000)

**4️⃣ Project Structure Overview**

bash

CopyEdit

/client

│── /src

│ ├── /components # Reusable UI components

│ ├── /pages # Page-based components

│ ├── /context # State management (Context API)

│ ├── /api # API calls

│ ├── /utilities # Helper functions

│ ├── /styles # CSS styling

│ ├── App.js # Main app file

│ ├── index.js # React entry point

│── /public # Static files

│── package.json # Project dependencies

│── README.md # Documentation

**5️⃣ Additional Commands**

✔ **Stop the server:**

bash

CopyEdit

CTRL + C

✔ **Run in production mode (optional):**

bash

CopyEdit

npm run build

🔹 This generates an optimized **/build** folder for deployment.

**📖 FitFlex Component Documentation**

This document provides an overview of the **key components** and **reusable components** in the **FitFlex** fitness app.

**📌 1️⃣ Key Components**

These are the main components that define the **core functionality** of the application.

**🔹 Navbar.js *(Navigation Bar)***

**Purpose:** Provides site-wide navigation for users.  
**Location:** /src/components/Navbar.js

jsx

CopyEdit

import { Link } from 'react-router-dom';

const Navbar = () => {

return (

<nav>

<h2>FitFlex</h2>

<ul>

<li><Link to="/">Home</Link></li>

<li><Link to="/categories">Categories</Link></li>

</ul>

</nav>

);

};

export default Navbar;

✅ **Features:**  
✔ Navigation to Home & Categories  
✔ Responsive & mobile-friendly

**🔹 Hero.js *(Homepage Banner)***

**Purpose:** Displays a featured workout section on the homepage.  
**Location:** /src/components/Hero.js

jsx

CopyEdit

const Hero = () => {

return (

<div className="hero">

<h1>Transform Your Fitness Journey</h1>

<p>Find the best workouts tailored to your needs.</p>

</div>

);

};

export default Hero;

✅ **Features:**  
✔ Motivational Call-To-Action  
✔ Eye-catching UI

**🔹 SearchBar.js *(Exercise Search Functionality)***

**Purpose:** Allows users to search for exercises.  
**Location:** /src/components/SearchBar.js

jsx

CopyEdit

import { useState } from 'react';

const SearchBar = ({ onSearch }) => {

const [query, setQuery] = useState('');

const handleSearch = () => {

onSearch(query);

};

return (

<div className="search-bar">

<input

type="text"

placeholder="Search exercises..."

value={query}

onChange={(e) => setQuery(e.target.value)}

/>

<button onClick={handleSearch}>Search</button>

</div>

);

};

export default SearchBar;

✅ **Features:**  
✔ Real-time user input handling  
✔ Calls onSearch function passed as a prop

**🔹 CategoryPage.js *(Workout Categories Page)***

**Purpose:** Displays a list of exercise categories.  
**Location:** /src/pages/CategoryPage.js

jsx

CopyEdit

import { useEffect, useState } from 'react';

import { fetchCategories } from '../api/fetchCategories';

import CategoryCard from '../components/CategoryCard';

const CategoryPage = () => {

const [categories, setCategories] = useState([]);

useEffect(() => {

fetchCategories().then(data => setCategories(data));

}, []);

return (

<div className="categories">

{categories.map((category) => (

<CategoryCard key={category.id} category={category} />

))}

</div>

);

};

export default CategoryPage;

✅ **Features:**  
✔ Fetches and displays exercise categories  
✔ Uses CategoryCard.js for individual categories

**🔹 ExercisePage.js *(Detailed Exercise Information)***

**Purpose:** Displays detailed exercise information, including **videos** and **instructions**.  
**Location:** /src/pages/ExercisePage.js

jsx

CopyEdit

import { useParams } from 'react-router-dom';

import { useEffect, useState } from 'react';

import { fetchExerciseDetails } from '../api/fetchExerciseDetails';

const ExercisePage = () => {

const { id } = useParams();

const [exercise, setExercise] = useState(null);

useEffect(() => {

fetchExerciseDetails(id).then(data => setExercise(data));

}, [id]);

if (!exercise) return <p>Loading...</p>;

return (

<div className="exercise-details">

<h2>{exercise.name}</h2>

<p>{exercise.instructions}</p>

</div>

);

};

export default ExercisePage;

✅ **Features:**  
✔ Fetches exercise details dynamically  
✔ Displays **name, description, and related videos**

**📌 2️⃣ Reusable Components**

These components are designed to be used across multiple pages for UI consistency.

**🔹 CategoryCard.js *(Reusable Exercise Category Card)***

**Purpose:** Displays a single workout category.  
**Location:** /src/components/CategoryCard.js

jsx

CopyEdit

import { Link } from 'react-router-dom';

const CategoryCard = ({ category }) => {

return (

<div className="category-card">

<h3>{category.name}</h3>

<Link to={`/category/${category.id}`}>View Exercises</Link>

</div>

);

};

export default CategoryCard;

✅ **Features:**  
✔ Displays category name  
✔ Links to category-specific exercise list

**🔹 ExerciseCard.js *(Reusable Exercise Display Card)***

**Purpose:** Displays an individual exercise in a grid or list view.  
**Location:** /src/components/ExerciseCard.js

jsx

CopyEdit

import { Link } from 'react-router-dom';

const ExerciseCard = ({ exercise }) => {

return (

<div className="exercise-card">

<h3>{exercise.name}</h3>

<Link to={`/exercise/${exercise.id}`}>View Details</Link>

</div>

);

};

export default ExerciseCard;

✅ **Features:**  
✔ Displays exercise name  
✔ Links to detailed exercise page

**🔹 Footer.js *(Application Footer)***

**Purpose:** Displays footer links and social media.  
**Location:** /src/components/Footer.js

jsx

CopyEdit

const Footer = () => {

return (

<footer>

<p>&copy; 2024 FitFlex. All rights reserved.</p>

</footer>

);

};

export default Footer;

✅ **Features:**  
✔ Provides copyright info  
✔ Keeps UI consistent

**🔹 Subscribe.js *(Newsletter Subscription Form)***

**Purpose:** Allows users to subscribe to a fitness newsletter.  
**Location:** /src/components/Subscribe.js

jsx

CopyEdit

const Subscribe = () => {

return (

<div className="subscribe">

<h3>Subscribe for Updates</h3>

<input type="email" placeholder="Enter your email" />

<button>Subscribe</button>

</div>

);

};

export default Subscribe;

✅ **Features:**  
✔ Simple **email input form**  
✔ Call-to-action for user engagement

**📌 Summary**

| **Component** | **Type** | **Purpose** |
| --- | --- | --- |
| Navbar.js | Key Component | Navigation across pages |
| Hero.js | Key Component | Motivational homepage banner |
| SearchBar.js | Key Component | Allows users to search exercises |
| CategoryPage.js | Key Component | Displays list of workout categories |
| ExercisePage.js | Key Component | Provides detailed exercise info |
| CategoryCard.js | Reusable | Displays a workout category |
| ExerciseCard.js | Reusable | Shows a single exercise |
| Footer.js | Reusable | Displays app footer |
| Subscribe.js | Reusable | Newsletter subscription form |

✅ **Modular & Reusable Design**  
✅ **Well-Structured for Scalability**  
✅ **Efficient API Calls & Dynamic Data Handling**

**🛠️ FitFlex State Management:**

FitFlex uses **Global State (Context API)** and **Local State (useState)** for efficient state handling.

**1️⃣ Global State (Context API) – For Shared Data**

🔹 Used for data **shared across multiple components** (e.g., exercises, categories).  
🔹 **Stored in:** /src/context/ExerciseContext.js

**🔹 Example: Managing Exercise Data Globally**

jsx

CopyEdit

import { createContext, useState, useEffect } from 'react';

import { fetchExercises } from '../api/fetchExercises';

export const ExerciseContext = createContext();

export const ExerciseProvider = ({ children }) => {

const [exercises, setExercises] = useState([]);

useEffect(() => {

fetchExercises().then(data => setExercises(data));

}, []);

return (

<ExerciseContext.Provider value={{ exercises }}>

{children}

</ExerciseContext.Provider>

);

};

✔ **Used in App.js**

jsx

CopyEdit

import { ExerciseProvider } from './context/ExerciseContext';

<ExerciseProvider>

<App />

</ExerciseProvider>;

✔ **Access in any component**

jsx

CopyEdit

import { useContext } from 'react';

import { ExerciseContext } from '../context/ExerciseContext';

const ExerciseList = () => {

const { exercises } = useContext(ExerciseContext);

return exercises.map((ex) => <p key={ex.id}>{ex.name}</p>);

};

**2️⃣ Local State (useState) – For Component-Specific Data**

🔹 Used for **temporary states** like input fields, toggles, and UI controls.  
🔹 **Managed inside individual components**

**🔹 Example: Handling Search Input (Local State)**

jsx

CopyEdit

import { useState } from 'react';

const SearchBar = ({ onSearch }) => {

const [query, setQuery] = useState('');

return (

<input

type="text"

value={query}

onChange={(e) => setQuery(e.target.value)}

/>

);

};

**📌 Summary**

| **State Type** | **Use Case** | **Example** |
| --- | --- | --- |
| **Global State (Context API)** | Shared data (Exercises, Categories) | ExerciseContext.js |
| **Local State (useState)** | Component-specific UI (Inputs, Toggles) | SearchBar.js |

✅ **Use Global State for app-wide data**  
✅ **Use Local State for UI interactions**

**🎨 FitFlex User Interface (UI) Overview:**

FitFlex features a **clean, modern, and responsive UI** designed for an intuitive fitness experience.

**1️⃣ Main Pages & Layout**

✔ **🏠 Home Page (Landing Page)**  
🔹 Hero section with motivational text & featured workouts.  
🔹 Search bar for exercises.  
🔹 Categories section (e.g., Cardio, Strength, Yoga).

✔ **📂 Categories Page**  
🔹 Grid layout of workout categories.  
🔹 Clickable **Category Cards** leading to exercises.

✔ **🏋️‍♂️ Exercise Details Page**  
🔹 Exercise name, difficulty, and muscle group.  
🔹 **Step-by-step instructions** & **YouTube workout videos**.

✔ **🔍 Search Results Page**  
🔹 Displays exercises **matching the search query**.

✔ **📩 Subscription Section**  
🔹 Newsletter signup for fitness updates.

✔ **🔻 Footer**  
🔹 Social media links & copyright info.

**2️⃣ UI Components & Design**

✔ **🌟 Navbar** → Sticky navigation with Home & Categories links.  
✔ **🎨 Cards (Categories & Exercises)** → Clickable **image-based** exercise previews.  
✔ **🔄 Responsive Design** → Mobile-friendly using **CSS Grid & Flexbox**.  
✔ **🖥️ Dark Mode (Optional)** → Can be added for better UX.

**:**

**🎨 FitFlex Styling & CSS Frameworks**

FitFlex uses **modern and responsive styling** for a sleek UI.

**1️⃣ CSS Frameworks Used**

✔ **🎭 Tailwind CSS** → Utility-first styling for fast development.  
✔ **🅱 Bootstrap** (Optional) → Pre-built components for buttons, grids, and forms.

**Installation:**

bash

CopyEdit

npm install tailwindcss

or

bash

CopyEdit

npm install bootstrap

**2️⃣ Global Styling (Tailwind Example)**

**📁 /src/styles/global.css**

css

CopyEdit

@tailwind base;

@tailwind components;

@tailwind utilities;

✔ **Applying Tailwind in Components**

jsx

CopyEdit

<div className="bg-blue-500 text-white p-4 rounded-lg">

Welcome to FitFlex!

</div>

**3️⃣ Key Styling Elements**

✔ **📌 Navbar & Buttons**  
🔹 Uses **fixed positioning** with flex for alignment.

✔ **📌 Grid-Based Layouts**  
🔹 Tailwind **grid-cols-3** or Bootstrap **row-cols-md-3** for categories & exercises.

✔ **📌 Responsive Design**  
🔹 Mobile-friendly with md:flex (Tailwind) or d-md-flex (Bootstrap).

**✅ FitFlex Testing & Strategies**

FitFlex uses **unit, integration, and end-to-end (E2E) testing** to ensure a bug-free experience.

**1️⃣ Testing Strategies**

✔ **🧪 Unit Testing (Jest + React Testing Library)**  
🔹 Tests individual components (e.g., SearchBar, ExerciseCard).

bash

CopyEdit

npm install --save-dev jest @testing-library/react

jsx

CopyEdit

import { render, screen } from "@testing-library/react";

import SearchBar from "../components/SearchBar";

test("renders search input", () => {

render(<SearchBar />);

expect(screen.getByPlaceholderText("Search exercises...")).toBeInTheDocument();

});

✔ **🔄 Integration Testing**  
🔹 Ensures **data flows correctly** between components.  
🔹 Example: ExercisePage correctly **fetches & displays exercise details**.

✔ **🌐 End-to-End (E2E) Testing (Cypress or Playwright)**  
🔹 Simulates **real user interactions** (e.g., searching for workouts).

bash

CopyEdit

npm install --save-dev cypress

js

CopyEdit

describe("Search Feature", () => {

it("allows users to search for an exercise", () => {

cy.visit("/");

cy.get("input").type("Push-ups");

cy.contains("Push-ups").should("exist");

});

});

DEMO LINK:

**🚨 Known Issues in FitFlex:**

Here are some potential **challenges and known issues** in the FitFlex project:

**1️⃣ API Limitations**

🔹 **Issue:** Free-tier APIs (RapidAPI) may have request limits.  
🔹 **Fix:** Implement **caching** or upgrade to a higher-tier API plan.

**2️⃣ Performance Bottlenecks**

🔹 **Issue:** Slow data fetching affects **exercise details page**.  
🔹 **Fix:** Use **lazy loading** and **pagination** for API calls.

**3️⃣ UI Responsiveness Issues**

🔹 **Issue:** Some layouts may break on **smaller screens**.  
🔹 **Fix:** Ensure **Tailwind’s responsive utilities** (md:grid, sm:flex) are used properly.

**4️⃣ State Management Complexity**

🔹 **Issue:** Context API may become inefficient with **large state updates**.  
🔹 **Fix:** Consider **Redux Toolkit** for better scalability.

**5️⃣ Search Functionality Limitations**

🔹 **Issue:** Search may not always return **relevant results**.  
🔹 **Fix:** Implement **fuzzy search** or **improve query handling**.

**🚀 Future Enhancements for FitFlex:**

**1️⃣ AI-Powered Personalized Workouts**

🔹 Use **machine learning** to recommend workouts based on user preferences.

**2️⃣ User Authentication & Profiles**

🔹 Allow users to **sign up, save workouts, and track progress**.

**3️⃣ Workout Tracking & Progress Analytics**

🔹 Implement a **dashboard** to track **completed workouts & calories burned**.

**4️⃣ Video Tutorials & Live Coaching**

🔹 Integrate **YouTube API** for guided video tutorials.  
🔹 Add **live coaching sessions** via WebRTC.

**5️⃣ Social Features & Community Engagement**

🔹 Enable **workout sharing, leaderboards, and challenges** with friends.

**6️⃣ Dark Mode & Custom Themes**

🔹 Add a **toggle for dark/light mode** for better UI flexibility.

**7️⃣ Mobile App (React Native)**

🔹 Convert FitFlex into a **cross-platform mobile app** for iOS & Android.

**Demo Link**:

https://drive.google.com/file/d/1POiDv8h8Z0z2GN5OBynnT6trZf4RNiKE/view?usp=sharing