**FitFlex: Your Personal Fitness Companion**

**1.Introduction**

**Project Title:** FitFlex: Your Personal Fitness Companion **Team Members:**

**Team ID :** SWTID1741339542146169

* **Team Leader :** Syedabudhagir .S(syedabu9003257797@gmail.com)
* Praveen Kumar.J (praveenkumar16072004@gmail.com)
* Vijay Prasanna.K (vijayprasanna035@gmail.com)
* Raghul.S(raghulsrinivasan2004@gmail.com)
* Dhanush.P (dhanush1862005@gmail.com)

**2.Project Overview**  
**Purpose:** The Fitness Exercise App is a web application designed to help users explore exercises based on body parts, view detailed exercise instructions, and discover popular workouts. The goal is to provide an intuitive, visually appealing frontend powered by React.js, integrated with the ExerciseDB API from RapidAPI.

**Features:**

* Browse exercise categories by body part (e.g., back, chest).
* View detailed exercise information, including GIF demonstrations.
* Responsive design with a modern, fitness-themed UI.
* Navigation between home, category, and exercise detail pages.

**3.Architecture**

**Component Structure:**

* **App.js:** Root component with routing setup using react-router-dom.
* **Home.js:** Displays the hero section, body part categories, and popular exercises.
* **Exercises.js:** Lists exercises for a selected body part category.
* **ExerciseDetail.js:** Shows detailed info for a specific exercise.
* **ExerciseCard.js:** Reusable card component for displaying exercise summaries.
* **Navbar.js:** Navigation bar (assumed, not provided).
* **Footer.js:** Footer section (assumed, not provided).
* **Hero.js:** Hero banner on the home page (assumed, not provided).

**State Management:**

* **Local State:** Managed with React’s useState and useEffect hooks in components like Home.js, Exercises.js, and ExerciseDetail.js for fetching and storing API data.
* **Global State:** Not implemented (no Redux or Context API observed). Each page fetches its own data independently.

**Routing:**

* Uses react-router-dom for client-side routing.
* Routes:
  + / → Home
  + /exercises/:category → Exercises
  + /exercise/:id → ExerciseDetail

**4.Setup Instructions**

**Prerequisites:**

* Node.js (v14 or higher)
* npm (v6 or higher)
* RapidAPI account and ExerciseDB API key

**Installation:**

1. Clone the repository:

git clone https://github.com/Rubeshvirat08/FitFlex.git

cd Fitflex

1. Install dependencies:

npm install

1. Create a .env file in the root directory and add your RapidAPI key:

REACT\_APP\_API\_KEY = 6beb41637bmsha87b9b11f2fceebp1efdeejsn4de831e59759

1. Ensure additional dependencies are installed (e.g., Bootstrap, Axios):

npm install react-router-dom axios bootstrap

**5.Folder Structure:**

* **src/**
  + **components/**: Reusable components (Navbar.js, Footer.js, Hero.js, ExerciseCard.js).
  + **pages/**: Page components (Home.js, Exercises.js, ExerciseDetail.js).
  + **styles/**: CSS files (App.css, index.css).
  + **App.js**: Main app component with routing.
  + **index.js**: Entry point for React rendering.
  + **reportWebVitals.js**: Performance metrics utility.

**Utilities:**

* Axios for API requests.
* No custom hooks or helper functions explicitly provided, but useParams and useEffect are used for dynamic routing and data fetching.

**6.Running the Application:**

* Start the frontend server:

npm start

* Open http://localhost:3000 in your browser.

**7.Component Documentation**

**Key Components:**

* **App.js:**
  + Purpose: Sets up routing and layout with Navbar and Footer.
  + Props: None.
* **Home.js:**
  + Purpose: Landing page with categories and popular exercises.
  + Props: None (fetches data internally).
* **Exercises.js:**
  + Purpose: Displays exercises for a selected category.
  + Props: None (uses useParams for category).
* **ExerciseDetail.js:**
  + Purpose: Shows detailed exercise info with GIF.
  + Props: None (uses useParams for exercise ID).

**Reusable Components:**

* **ExerciseCard.js:**
  + Purpose: Displays a summary of an exercise with a link to its details.
  + Props: exercise (object with id, name, target).
  + Configuration: Links to /exercise/:id using react-router-dom’s Link.

**8.State Management**

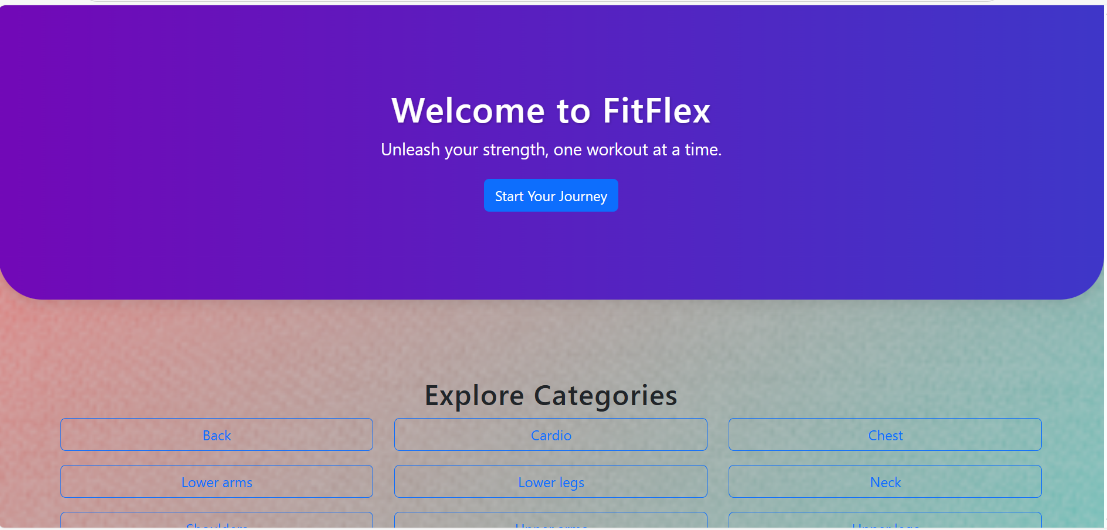
**Global State:**

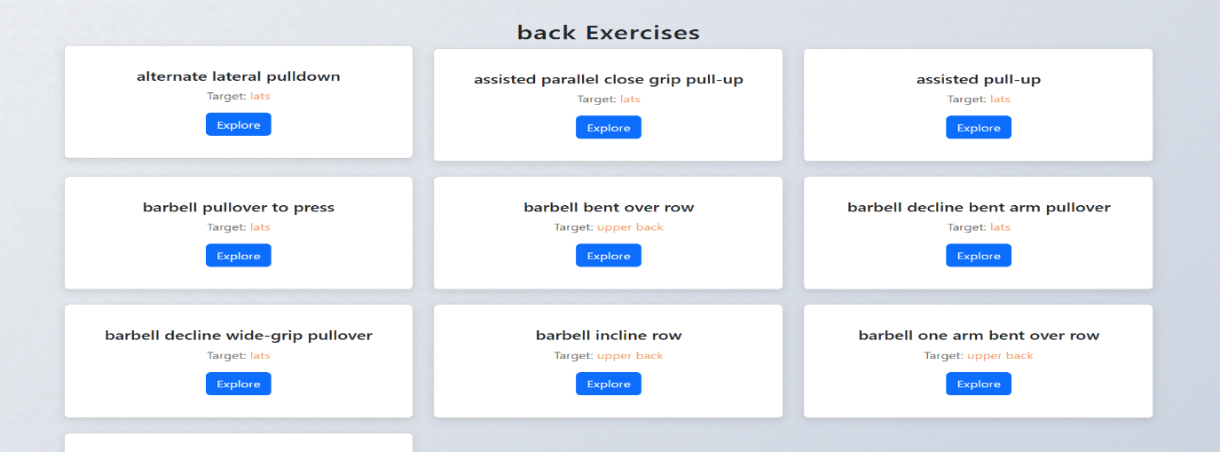
* None implemented. Each component independently fetches data from the ExerciseDB API.

**Local State:**

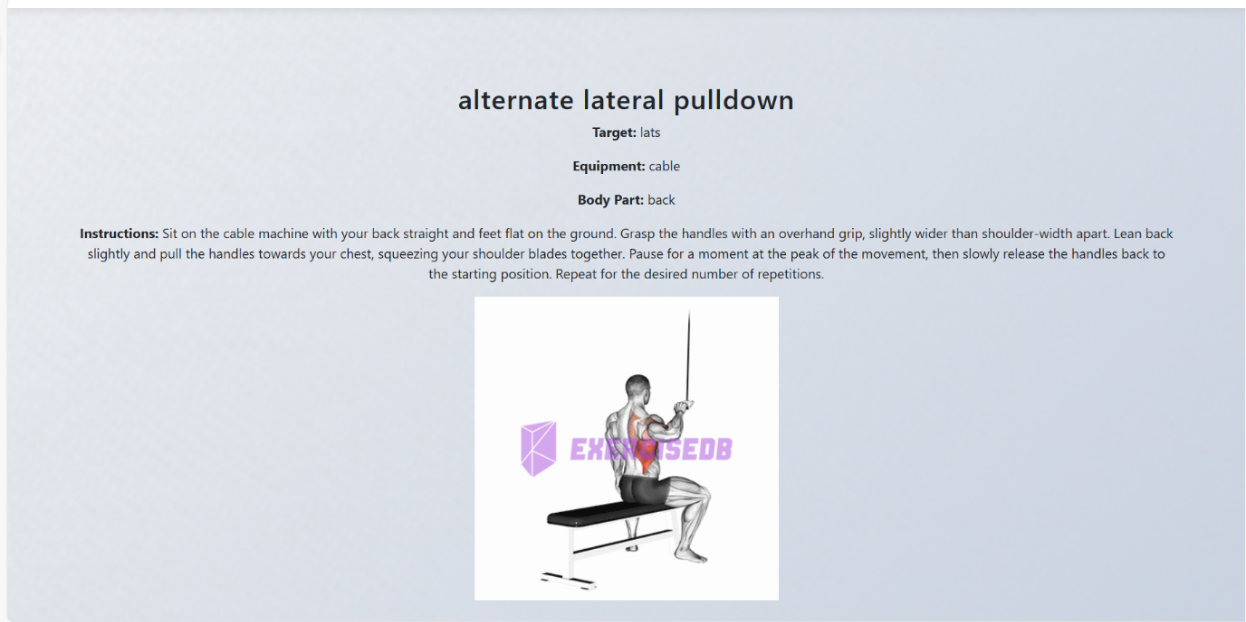
* Managed with useState for API responses (e.g., bodyParts, exercises, exercise).
* useEffect triggers API calls when component mounts or dependencies (e.g., id, category) change.

**9.User Interface**

* **Home Page:** Categories as buttons, popular exercise cards, and a hero section.
* 
* **Exercises Page:** Grid of exercise cards for a selected body part.



* **Exercise Detail Page:** Exercise name, target, equipment, instructions, and GIF.



**10.Styling**

**CSS Frameworks/Libraries:**

* Bootstrap (bootstrap.min.css) for grid layout and basic styling.
* Custom CSS in App.css for detailed theming and component styles.

**Theming:**

* Fitness-inspired design with:
  + Deep green (#2d6a4f) for headings and accents.
  + Warm orange (#f4a261) for buttons and highlights.
  + Soft gradient background (#f5f7fa to #c3cfe2).
* Subtle fabric texture overlay on the app background.
* Responsive design with media queries for smaller screens.

**11.Testing**

**Testing Strategy:**

* Basic unit test provided in App.test.js using React Testing Library and Jest.
* Current test checks for a "learn react" link (mismatched with app content).
* Suggested: Add tests for ExerciseCard, Home, and API fetch success/failure.

**Code Coverage:**

* No explicit coverage tools mentioned (e.g., Istanbul).
* Recommendation: Use npm test -- --coverage with Jest to measure coverage.

**12.Screenshots or Demo**

**Demo link** : **http://drive.google.com/file/d/1lU1F5WTEZqGH2wt14ZO9DZ9to0vya5rF/view**

**13.Known Issues**

* **API Key Exposure:** Hardcoded in source files; should use environment variables.
* **Limited Error Handling:** API errors logged to console but not displayed to users.
* **No Loading States:** Home.js and Exercises.js lack loading indicators.

**14.Future Enhancements**

* **Search Functionality:** Add a search bar to filter exercises by name or target.
* **Global State:** Implement Context API or Redux for caching API data.
* **Animations:** Add transitions (e.g., fade-ins) using Framer Motion.
* **User Accounts:** Allow saving favorite exercises with a backend integration.
* **Improved Testing:** Expand test suite for components and API interactions.