FitFlex: Your Personal Fitness Companion (React Application)

Team Leader: Fathima.M

Team Members: Divya.S, Harini.S, Elakkiya.M

INTRODUCTION:

FitFlex is a cutting-edge fitness app that's here to revolutionize your workout experience. With a seamless interface, powerful search capabilities, and an extensive collection of exercises for every fitness level, FitFlex provides the tools you need to succeed. Whether you're just starting or are already a fitness enthusiast, FitFlex customizes your journey to help you reach your health and wellness goals. Start your transformation with FitFlex today!

DESCRIPTION:

- Welcome to the cutting edge of fitness with FitFlex! Our state-of-the-art fitness app is carefully crafted to transform how you interact with your workout routines, serving the needs of everyone from casual fitness lovers to experienced athletes. Designed with an intuitive user interface and an all-encompassing set of features, FitFlex is here to completely reshape the way you discover and engage with fitness exercises.
- FitFlex is designed with a focus on sleek, user-friendly aesthetics, providing an unmatched fitness experience. Users can seamlessly explore a variety of exercise categories, with features like dynamic search that deliver the newest and most effective workouts from the fitness industry.
- FitFlex caters to a wide range of users, from beginners to experienced fitness enthusiasts, creating a diverse and dynamic community united by a passion for a healthy lifestyle. Our platform goes beyond just offering effective workouts; it aims to reshape the way users interact with fitness. By providing personalized exercise routines and encouraging collaboration, FitFlex fosters a sense of support and camaraderie within the fitness community. Whether you're just

starting or are already a fitness pro, FitFlex is designed to help you achieve your goals and stay motivated.

- Join us on a fitness journey where innovation blends with traditional exercise methods. Each tap in FitFlex brings you closer to a wide range of workouts and wellness insights. Experience the future of fitness engagement with features designed to support your health goals. FitFlex offers a glimpse into a healthier, more active you. Start your transformation today!
- Take your fitness journey to the next level with FitFlex, where each exercise opens the door to a world of health and wellness ready for you to explore. Let FitFlex be your trusted partner in maintaining a fit, active lifestyle, helping you stay connected and motivated every step of the way.

SCENARIO BASED INTRO:

You stand in front of your gym bag, ready to tackle your fitness goals, but unsure where to begin. That's when you think of FitFlex, the cutting-edge app designed to transform your workouts. With a single tap, the app opens to a dynamic, sleek interface—customized workout plans, a variety of exercise options, and a community eager to support you. This isn't just another fitness app; it feels like something more. Curious, you pick a workout and prepare to dive into a fitness experience like no other.

PROJECT GOALS AND OBJECTIVES:

The main goal of FitFlex is to create an easy-to-use platform designed for individuals who are dedicated to fitness, exercise, and overall well-being. We aim to offer an experience that supports users in their pursuit of a healthier lifestyle.

Our key objectives include:

✓ **User-Friendly Interface:** Build a simple, intuitive design that allows users to quickly browse, save, and share their favorite workout routines without hassle.

✓ **Effective Exercise Management:** Provide users with powerful tools to organize and track their exercise routines, with smart search capabilities to help tailor workouts to individual needs.

✓ **Technology Stack:** Utilize cutting-edge web development technologies, specifically React.js, to ensure the platform runs smoothly and delivers an enjoyable user experience.

FEATURES OF FITFLEX:

✓ **Exercise Library from Trusted Fitness APIs:** Access a wide variety of exercises pulled from reliable fitness APIs, covering different workout types and tailored to meet various fitness objectives.

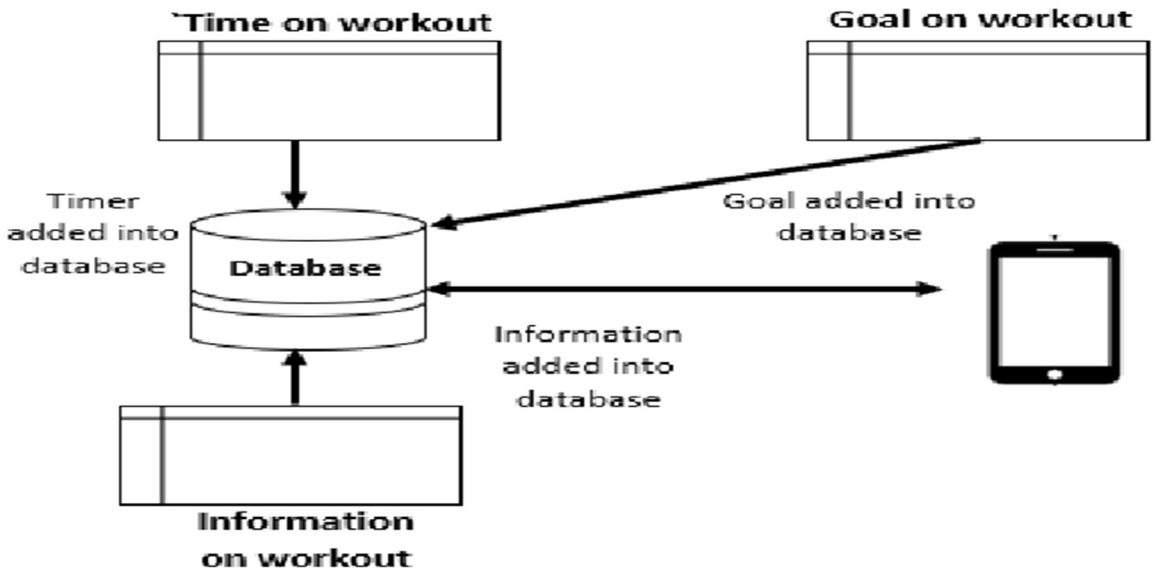
✓ **Visual Exercise Discovery:** Explore workout routines through curated image galleries that allow users to visually navigate through different exercise categories and find new fitness challenges.

✓ **Simple and Intuitive Interface:** Enjoy a smooth, modern design that makes it easy to use the app, with a straightforward layout for quick and efficient exercise selection.

✓ **Enhanced Search Functionality:** Quickly find specific exercises or workout plans using an advanced search feature, making it simple for users with different fitness preferences to navigate and personalize their experience.

TECHNICAL ARCHITECTURE:

FitFlex takes a user-first approach, ensuring a seamless and intuitive experience right from the start. The sleek and interactive user interface (UI), likely built using a framework like React Native, makes navigation smooth and easy. On the backend, FitFlex uses a specially designed API client that connects to external services through Rapid API. This platform enables the integration of features like fitness trackers, nutrition information, and workout tracking, without the need to develop these components from scratch. This strategy allows FitFlex to offer a rich set of features while keeping the focus on its core functionalities.



PRE-REQUISITES

Before starting with the development of the FitFlex frontend application using React.js, ensure that you have the following tools and technologies installed:

1. Node.js and npm

Node.js is a JavaScript runtime environment that enables the execution of JavaScript code on the server side. npm (Node Package Manager) comes bundled with Node.js and is used to install dependencies for your project.

- **Download Node.js**: Node.js Download
- **Installation Instructions**: Node.js Installation Guide

2. React.js

React.js is a JavaScript library used for building user interfaces, especially for creating reusable components and handling dynamic data.

- **To create a new React app**:
  - Run the following command in your terminal:

```
npx create-react-app my-react-app
```

Replace my-react-app with your preferred project name.

- **Navigate to your project directory**:

```
cd my-react-app
```

- **To run the development server**:
  - Start the development server by executing:

```
npm start
```

  - This will launch the React application, and you can view it by navigating to http://localhost:3000 in your web browser.

3. HTML, CSS, and JavaScript Basics

Basic knowledge of **HTML**, **CSS**, and **JavaScript** is required to create the structure, style, and functionality of your React app. These technologies are essential for building the user interface and client-side interactivity.

4. Version Control with Git

To track changes and collaborate with others, it is recommended to use **Git** for version control. Platforms such as **GitHub** or **Bitbucket** can be used to host your repositories.

- **Download Git**: Git Download

5. Development Environment

Choose an IDE or code editor that you prefer to work with. Some recommended editors include:

- **Visual Studio Code**: Download Visual Studio Code
- **Sublime Text**: Download Sublime Text
- **WebStorm**: Download WebStorm

Setting Up the FitFlex Application

Follow the steps below to set up the FitFlex frontend application on your local machine:

1. Clone the Project

Download the project files from the provided link:

- https://drive.google.com/file/d/1xdB7aMWIi0ryh6wipn-cm0Y2Dv7IkPu2/view?usp=drive_link

2. Install Dependencies

After downloading and extracting the project, navigate to the project directory using the terminal:

```
cd fitness-app-react
```

Install the necessary libraries and dependencies by running the following command:

```
npm install
```

3. Start the Development Server

Once the dependencies are installed, you can start the development server by running:

```
npm start
```

This will launch the application, and you can access it in your web browser at:

```
http://localhost:3000
```

4. Access the Application

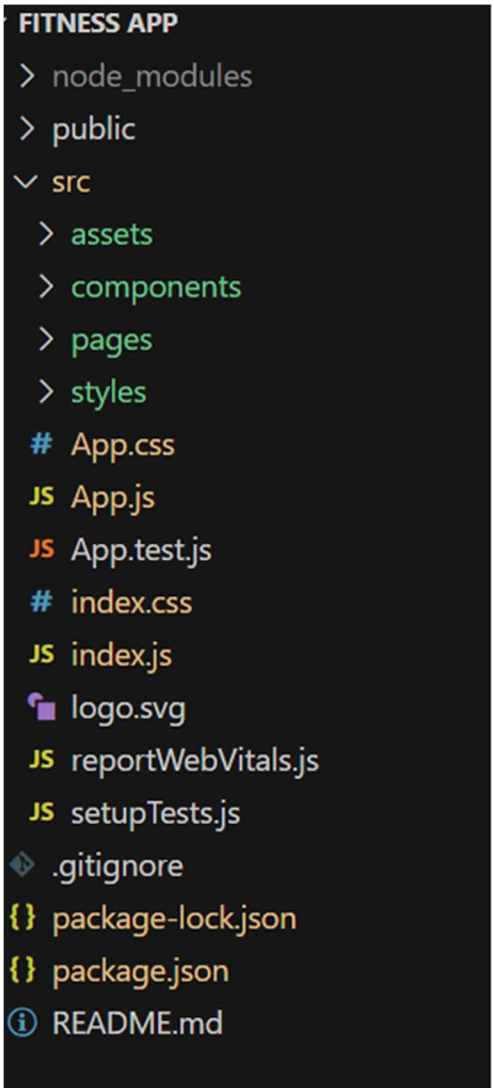
Upon successful setup, you should see the homepage of the FitFlex application in your browser. If the application loads properly, the installation and setup are complete.

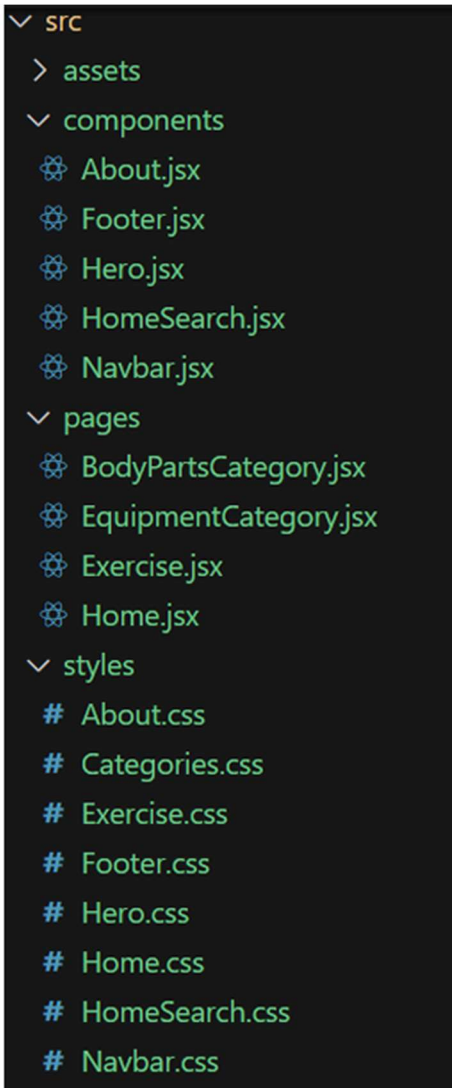
Next Steps

You have now successfully set up the FitFlex application on your local machine. You can proceed with further tasks such as:

- **Customization**: Modify the application to meet your specific requirements.
- **Development**: Continue developing new features, improving the UI/UX, or integrating additional functionality.
- **Testing**: Run tests to ensure the application is working as expected and make any necessary adjustments.

PROJECT STRUCTURE:





In this project, we've split the files into 3 major folders, Components, Pages and Styles. In the pages folder, we store the files that acts as pages at different URLs in the application. The components folder stores all the files, that returns the small components in the application. All the styling css files will be stored in the styles folder.

Project Flow:

Project demo:

starting to work on this project, let's see the demo.

Demo link:

https://drive.google.com/file/d/1gRut2q44CQunMjS0qUH26LaAdnO2Jis-/view?usp=drive_link

Use the code in:

https://drive.google.com/file/d/1xdB7aMWli0ryh6wipn-cm0Y2Dv7IkPu2/view?usp=drive_link

Milestone 1: Project setup and configuration.

- **Installation of required tools**:

To build the FitFlex app, we'll need a developer's toolkit. We'll leverage React.js for the interactive interface, React Router Dom for seamless navigation, and Axios to fetch fitness data. To style the app, we'll choose either Bootstrap or Tailwind CSS for pre-built components and a sleek look. Open the project folder to install necessary tools. In this project, we use:

  - React Js
  - React Router Dom
  - React Icons
  - Bootstrap/tailwind css
  - Axios

- For further reference, use the following resources
  - React Installation Guide
  - React Bootstrap v4 - Getting Started
  - Axios Documentation - Introduction
  - React Router - Tutorial

Milestone 2: Project Development

❖ Setup the Routing paths Setup the clear routing paths to access various files in the application.

```jsx
<div className="App">

  <Navbar />

  <Routes>

    <Route path="/" element={<Home />} />
    <Route path="/bodyPart/:id" element={<BodyPartsCategory />} />
    <Route path="/equipment/:id" element={<EquipmentCategory />} />
    <Route path="/exercise/:id" element={<Exercise />} />
  </Routes>

  <Footer />

</div>
```

- Develop the Navbar and Hero components
- Code the popular search/categories components and fetch the categories from rapid Api.
- Additionally, we can add the component to subscribe for the newsletter and the footer.
- Now, develop the category page to display various exercises under the category.
- Finally, code the exercise page, where the instructions, other details along with related videos from the YouTube will be displayed.

Important Code snips:

- **Fetching available Equipment list & Body parts list**

From the Rapid API hub, we fetch available equipment and list of body parts with an API request.

```
const bodyPartsOptions = {
  method: 'GET',
  url: 'https://exercisedb.p.rapidapi.com/exercises/bodyPartList',
  headers: {
    'X-RapidAPI-Key': 'place your api key',
    'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com'
  }
};

const equipmentOptions = {
  method: 'GET',
  url: 'https://exercisedb.p.rapidapi.com/exercises/equipmentList',
  headers: {
    'X-RapidAPI-Key': 'place your api key',
    'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com'
  }
};

useEffect(() => {
  fetchData();
}, [])

const fetchData = async () =>{
  try {
    const bodyPartsData = await axios.request(bodyPartsOptions);
    setBodyParts(bodyPartsData.data);

    const equipmentData = await axios.request(equipmentOptions);
    setEquipment(equipmentData.data);
  } catch (error) {
    console.error(error);
  }
}
```

Here's a breakdown of the code:

Dependencies:

The code utilizes the following libraries:

Axios: A popular promise-based HTTP client for JavaScript. You can add a link to the official documentation for Axios https://axios-http.com/

API Key:

Replace 'place your api key' with a placeholder mentioning that the user needs to replace it with their own RapidAPI key. You can mention how to acquire an API key from RapidAPI.

Body Parts Options and equipment Options:

These variables hold configuration options for fetching data from the RapidAPI exercise database.

- method: The HTTP method used in the request. In this case, it's set to GET as the code is fetching data from the API.
- url: The URL of the API endpoint to fetch data from. Here, it's set to https://exercisedb.p.rapidapi.com/exercises/bodyPartList for fetching a list of body parts and https://exercisedb.p.rapidapi.com/exercises/equipmentList for fetching a list of equipment.
- headers: This section contains headers required for making the API request. Here it includes the X-RapidAPI-Key header to provide your API key and the X-RapidAPI-Host header specifying the host of the API.

fetchData function:

This function is responsible for fetching data from the API. It makes use of async/await syntax to handle asynchronous operations. First it fetches data for body parts using axios.request(body Parts Options). Then it stores the fetched data in the bodyParts state variable using setBodyParts.

Similarly, it fetches data for equipment using axios.request(equipmentOptions). Then it stores the fetched data in the equipment state variable using setEquipment. In case of any errors during the API request, the catch block logs the error to the console using console.error.

useEffect Hook:

The useEffect hook is used to call the fetchData function whenever the component mounts. This ensures that the data is fetched as soon as the component loads.

Overall, the code snippet demonstrates how to fetch data from a RapidAPI exercise database using JavaScript's Axios library.

- **Fetching exercises under particular category**

To fetch the exercises under a particular category, we use the below code.

```
const fetchData = async (id) => {
    const options = {
        method: 'GET',
        url: `https://exercisedb.p.rapidapi.com/exercises/equipment/${id}`,
        params: {limit: '50'},
        headers: {
          'X-RapidAPI-Key': 'your api key',
          'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com'
        }
    };

    try {
        const response = await axios.request(options);
        console.log(response.data);
        setExercises(response.data);
    } catch (error) {
        console.error(error);
    }

}
```

It defines a function called fetchData that fetches data from an exercise database API. Here's a breakdown of the code:

const options = {...}:

This line creates a constant variable named options and assigns it an object literal. The object literal contains properties that configure the API request, including:

- method: Set to 'GET', indicating that the API request is a GET request to retrieve data from the server.
- url: Set to https://exercisedb.p.rapidapi.com/exercises/equipment/${id}, which is the URL of the API endpoint for fetching exercise equipment data. The ${id} placeholder will likely be replaced with a specific equipment ID when the function is called.
- params: An object literal with a property limit: '50'. This specifies that you want to retrieve a maximum of 50 exercise equipment results.
- headers: An object literal containing two headers required for making the API request:
- 'X-RapidAPI-Key': Your RapidAPI key, which is used for authentication. You should replace 'your api key' with a placeholder instructing users to replace it with their own API key.
- 'X-RapidAPI-Host': The host of the API, which is 'exercisedb.p.rapidapi.com' in this case.

const fetchData = async (id) => {...}:

This line defines an asynchronous function named fetchData that takes an id parameter. This id parameter is likely used to specify the equipment ID for which data needs to be fetched from the API.

try...catch block:

- The try...catch block is used to handle the API request.
- The try block contains the code that attempts to fetch data from the API using axios.request(options).
- The await keyword is used before axios.request(options) because the function is asynchronous and waits for the API request to complete before proceeding.
- If the API request is successful, the response data is stored in the response constant variable.
- The console.log(response.data) line logs the fetched data to the console.
- The.then method (not shown in the image) is likely used to process the fetched data after a successful API request.
- The catch block handles any errors that might occur during the API request. If there's an error, it's logged to the console using console.error(error).

- **Fetching Exercise details**

Now, with the help of the Exercise ID, we fetch the details of a particular exercise with API request.

```
useEffect(()=>{
    if (id){
        fetchData(id)
    }
},[])

const fetchData = async (id) => {
    const options = {
      method: 'GET',
      url: `https://exercisedb.p.rapidapi.com/exercises/exercise/${id}`,
      headers: {
        'X-RapidAPI-Key': 'ae40549393msh0c35372c617b281p103ddcjsn0f4a9ee43ff0',
        'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com'
      }
    };

    try {
        const response = await axios.request(options);
        console.log(response.data);
        setExercise(response.data);

        fetchRelatedVideos(response.data.name)
    } catch (error) {
        console.error(error);
    }

}
```

The code snippet demonstrates how to fetch exercise data from an exercise database API using JavaScript's fetch API. Here's a breakdown of the code: API Endpoint and Key:

- Replace 'https://example.com/exercise' with the actual URL of the API endpoint you want to use.
- Replace 'YOUR_API_KEY' with a placeholder instructing users to replace it with their own API key obtained from the API provider.

async function:

The code defines an asynchronous function named fetchData that likely takes an id parameter as input. This id parameter might be used to specify the ID of a particular exercise or category of exercises to fetch.

fetch request:

Inside the fetchData function, the fetch API is used to make an HTTP GET request to the API endpoint. The function creates a fetch request with the following details:

- Method: GET (to retrieve data from the server)
- URL: The API endpoint URL where exercise data resides.

Handling the Response:

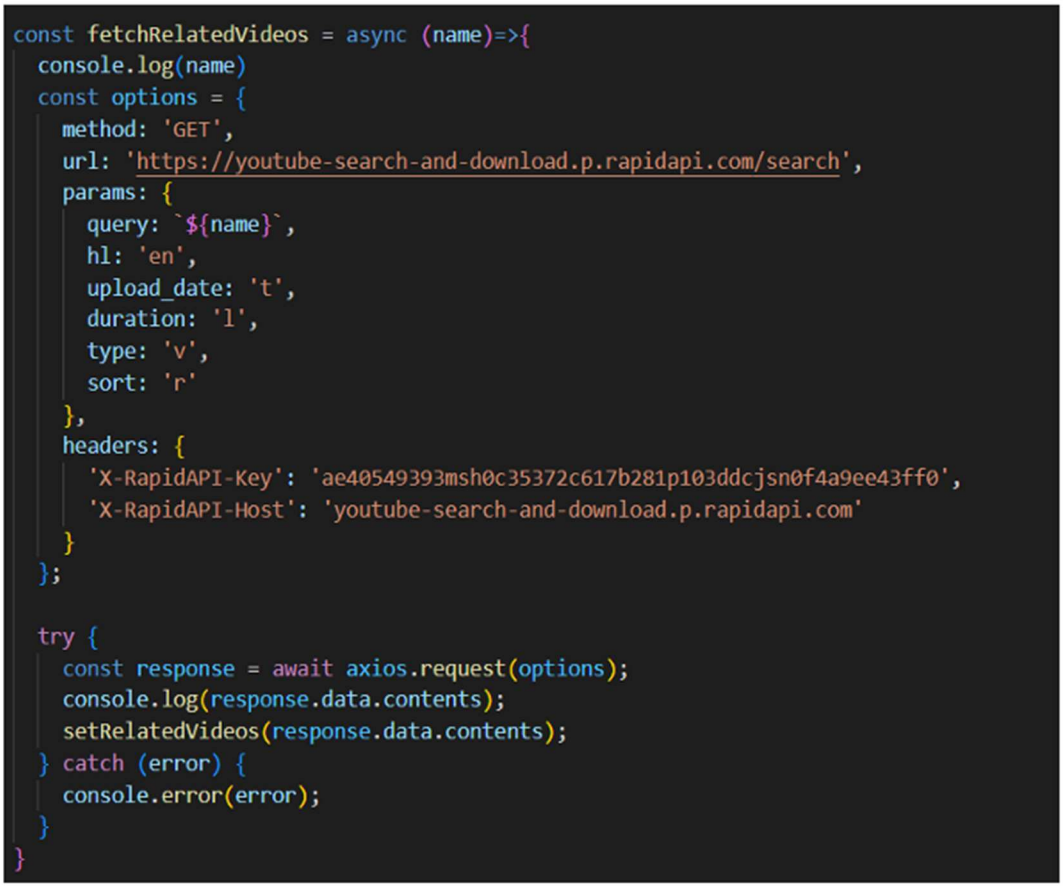
- The then method is used to handle the response from the API request. If the request is successful (i.e., status code is 200), the response is converted to JSON format using response.json().
- The then method then likely processes the fetched exercise data, which might involve storing it in a state variable or using it to populate a user interface.

Error Handling:

The catch method is used to handle any errors that might occur during the API request. If there's an error, it's logged to the console using console.error.

- **Fetching related videos from YouTube**

Now, with the API, we also fetch the videos related to a particular exercise with code given below.



```
const fetchRelatedVideos = async (name)=>{
  console.log(name)
  const options = {
    method: 'GET',
    url: 'https://youtube-search-and-download.p.rapidapi.com/search',
    params: {
      query: `${name}`,
      hl: 'en',
      upload_date: 't',
      duration: 'l',
      type: 'v',
      sort: 'r'
    },
    headers: {
      'X-RapidAPI-Key': 'ae40549393msh0c35372c617b281p103ddcjsn0f4a9ee43ff0',
      'X-RapidAPI-Host': 'youtube-search-and-download.p.rapidapi.com'
    }
  };

  try {
    const response = await axios.request(options);
    console.log(response.data.contents);
    setRelatedVideos(response.data.contents);
  } catch (error) {
    console.error(error);
  }
}
```

The code snippet shows a function called fetchRelatedVideos that fetches data from YouTube using the RapidAPI service. Here's a breakdown of the code:

fetchRelatedVideos function:

This function takes a name parameter as input, which is likely the name of a video or a search query.

API configuration:

The code creates a constant variable named options and assigns it an object literal containing configuration details for the API request:

- method: Set to 'GET', indicating a GET request to retrieve data from the server.
- url: Set to 'https://youtube-search-and-download.p.rapidapi.com/search', which is the base URL of the RapidAPI endpoint for YouTube search.
- params: An object literal containing parameters for the YouTube search query:
- query: Set to \${name}, a template literal that likely gets replaced with the actual name argument passed to the function at runtime. This specifies the search query for YouTube videos.
- Other parameters like hl (language), sort (sorting criteria), and type (video type) are included but their values are not shown in the snippet.
- headers: An object literal containing headers required for making the API request:
- 'X-RapidAPI-Key': Your RapidAPI key, which is used for authentication. You should replace 'YOUR_API_KEY' with a placeholder instructing users to replace it with their own API key.
- 'X-RapidAPI-Host': The host of the API, 'youtube-search-and-download.p.rapidapi.com' in this case.

Fetching Data (try...catch block):

- The try...catch block is used to handle the API request.
- The try block contains the code that attempts to fetch data from the API using axios.request(options).
- axios is an external JavaScript library for making HTTP requests. If you don't already use Axios in your project, you'll need to install it using a package manager like npm or yarn.
- The then method (not shown in the code snippet) is likely used to process the fetched data after a successful API request.

- The catch block handles any errors that might occur during the API request. If there's an error, it's logged to the console using console.error(error).

PROJECT EXECUTION:

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

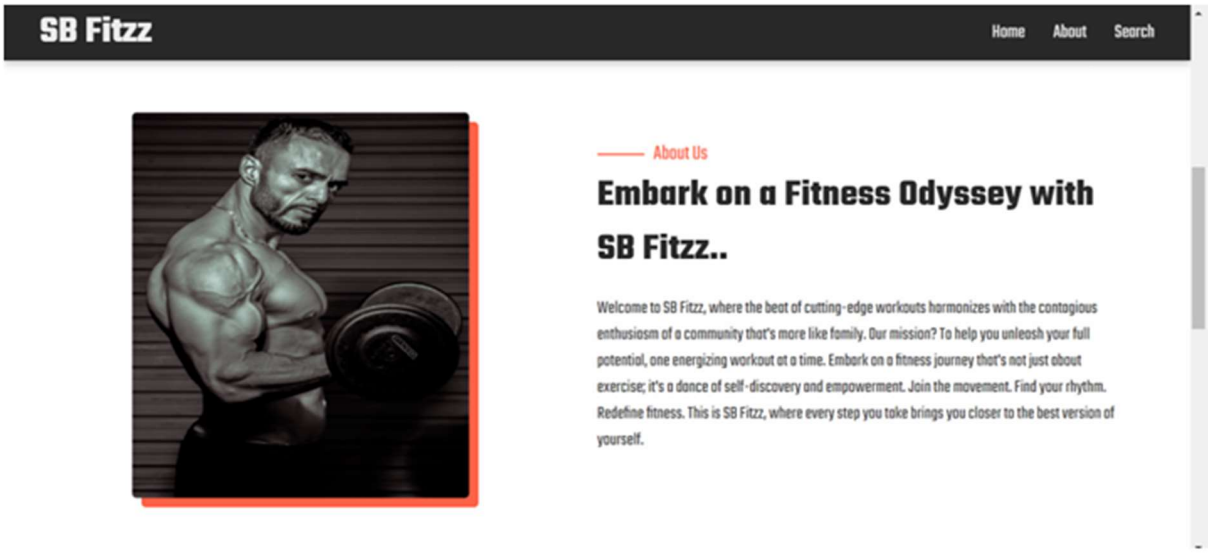
Here are some of the screenshots of the application.

- **Hero component**

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

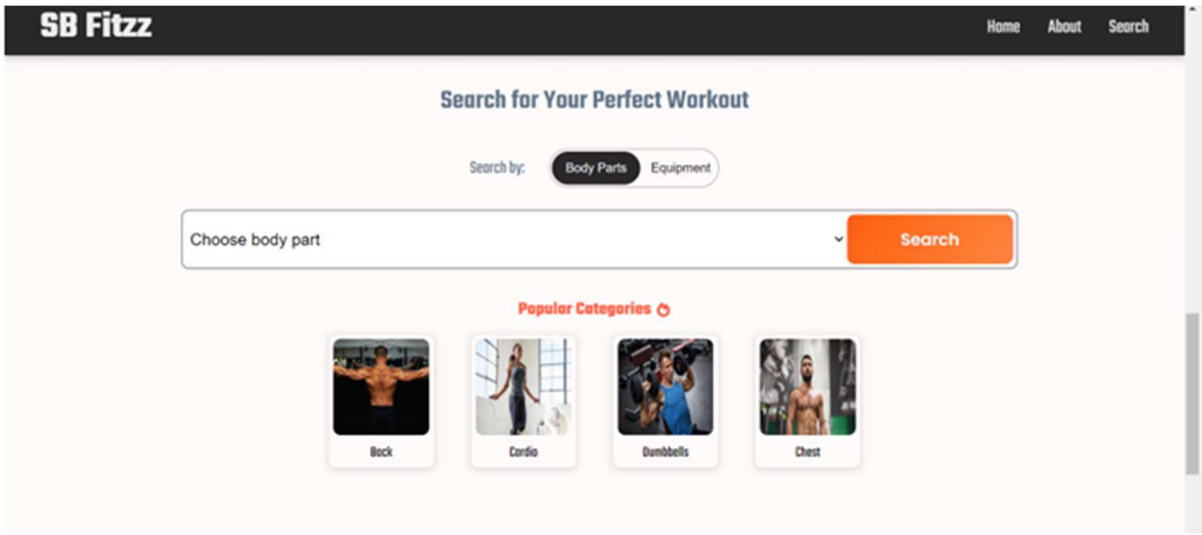
- **About**

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



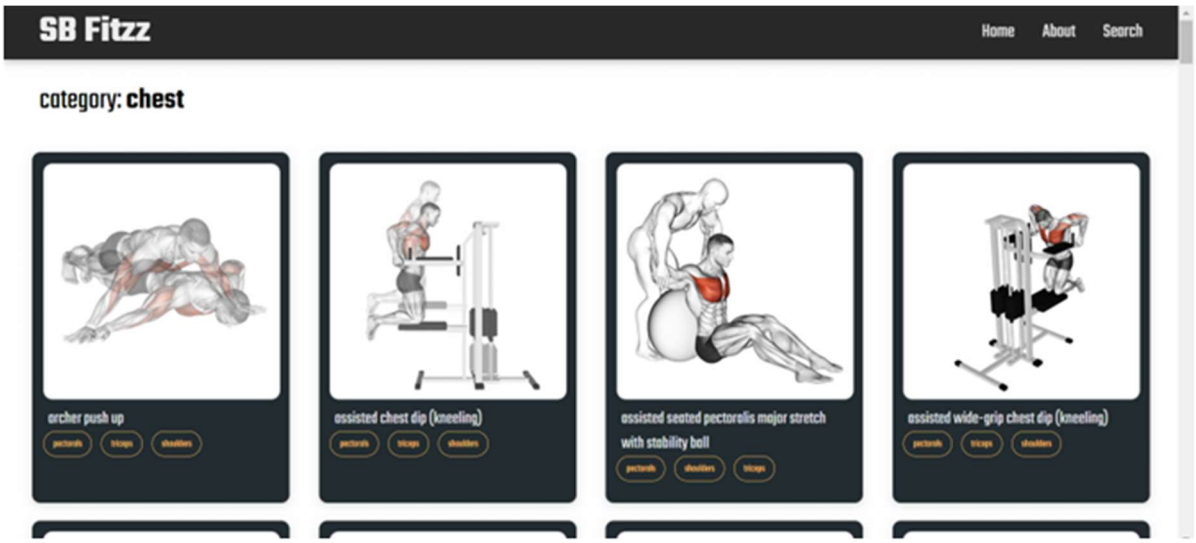
- **Search**

B Fitzz 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



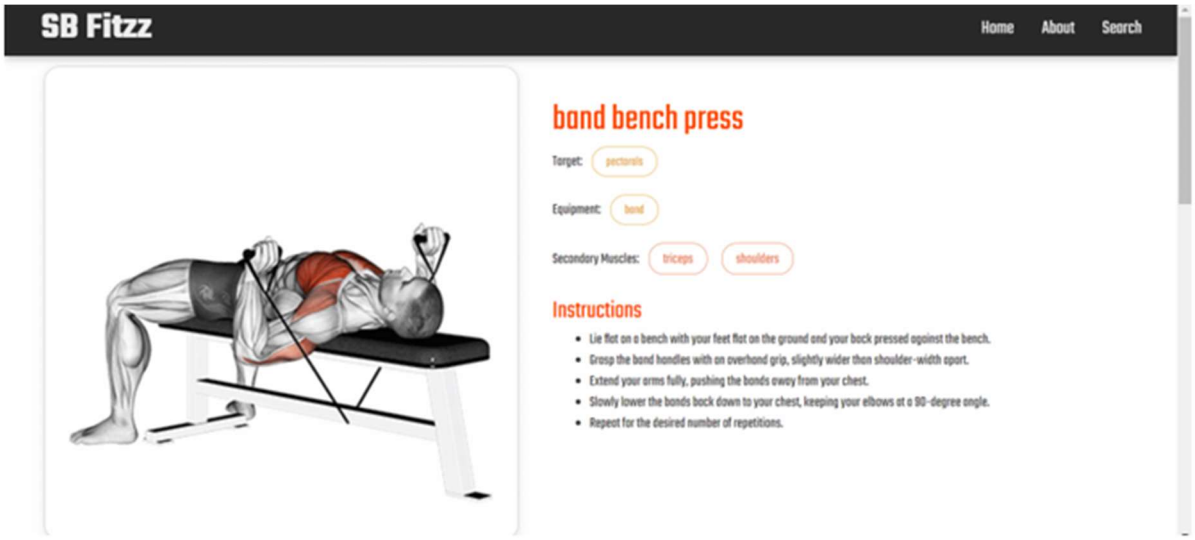
- **Category page**

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 page**

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/file/d/1gRut2q44CQunMjS0qUH26LaAdnO2Jis-/view?usp=drive_link