FitFlex: Your Personal Fitness

Companion**(React Application)**

Introduction:

FitFlex is an innova ve fitness applica on designed to enhance your workout experience. With its intui ve user interface, dynamic search func onality, and a comprehensive library of exercises suitable for all fitness levels, FitFlex provides a personalized approach to fitness. Start your fitness journey with FitFlex and work towards achieving your wellness goals with ease and efficiency.

Description:

Welcome to the forefront of fitness explora on with FitFlex! Our innova ve fitness app is me culously designed to revolu onize the way you engage with exercise rou nes, catering to the diverse interests of both fitness enthusiasts and seasoned workout professionals. With a focus on an intui ve user interface and a comprehensive feature set, FitFlex is set to redefine the en re fitness discovery and exercise experience.

Cra ed with a commitment to user-friendly aesthe cs, FitFlex immerses users in an unparalleled fitness journey. Effortlessly navigate through a wide array of exercise categories with features like dynamic search, bringing you the latest and most effec ve workouts from the fitness world.

From those embarking on their fitness journey to seasoned workout aficionados, FitFlex embraces a diverse but also encourages collabora on and sharing within the vibrant fitness community.

Embark on this fitness adventure with us, where innova on seamlessly intertwines with established exercise a glimpse into the future of a healthier you.

Elevate your fitness explora on with FitFlex, where every exercise becomes a gateway to a world of wellness

wai ng to be discovered and embraced. Trust FitFlex to be your reliable companion on the journey to staying connected with a fit and ac ve lifestyle.

Scenario Based intro:

You're ready to take your fitness journey seriously. As you lace up your sneakers, a ques on lingers—where do you begin? Then you remember FitFlex, the cu ng-edge app designed to transform your workouts. With a simple tap, the screen comes to life—vibrant visuals, personalized workout plans, diverse exercise categories, and a thriving community of fitness enthusiasts. This isn't just another fitness app—FitFlex feels different. Drawn in by its seamless experience, you select a workout, ready to step into the future of fitness.

Project Goals and Objectives:

The overarching aim of FitFlex is to offer an accessible pla orm tailored for individuals passionate about fitness, exercise, and holis c well-being.

Our key objec ves are as follows:

✓ **User-Friendly Experience:** Develop an intui ve interface that facilitates easy naviga on, enabling users to effortlessly discover, save, and share their preferred workout rou nes.

✓ **Comprehensive Exercise Management:** Provide robust features for organizing and managing exercise rou nes, incorpora ng advanced search op ons for a personalized fitness experience.

✓ **Technology Stack:** Harness contemporary web development technologies, with a focus on React.js, to ensure an efficient and enjoyable user experience.

Features of FitFlex:

1. Exercises from Fitness API

Gain access to a comprehensive selection of exercises sourced from reputable fitness APIs. FitFlex offers a diverse range of workout categories tailored to various fitness goals, ensuring users find routines that suit their needs.

2. Visual Exercise Exploration

Discover workouts through an engaging image gallery that allows users to visually explore different exercise categories. This feature enhances the fitness experience by making it easier to identify and select workouts based on preference and interest.

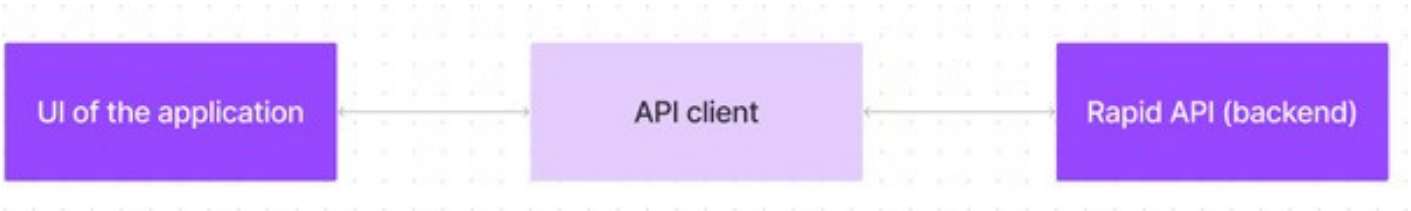
3. Intuitive and User-Friendly Design

FitFlex is designed with a modern, clean interface that ensures seamless navigation. The intuitive layout optimizes the user experience, making exercise selection effortless and engaging.

4. Advanced Search Functionality

Easily find specific exercises or workout plans using the app's powerful search feature. This functionality enhances accessibility, allowing users to quickly locate workouts that align with their fitness preferences and goals.

Technical Architecture:



FitFlex priori zes a user-centric approach from the ground up. The engaging user interface (UI), likely built with a framework like React Na ve, keeps interac on smooth and intui ve. An API client specifically designed for FitFlex communicates with the backend, but with a twist: it leverages Rapid API. This pla orm grants access to various external APIs, allowing FitFlex to poten ally integrate features like fitness trackers, nutri on data, or workout tracking func onali es without building everything from scratch. This approach ensures a feature-rich experience while focusing development efforts on the core FitFlex functionalities.

PRE-REQUISITES:

Here are the key prerequisites for developing a frontend applica on using React.js:

✓ **Node.js and npm**:

Node.js is a powerful JavaScript run me environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient pla orm for building network applica ons.
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/

✓ **React.js**:

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

Install React.js, a JavaScript library for building user interfaces.

- Create a new React app:

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

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

- Navigate to the project directory:

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

- Running the React App:

With the React app created, you can now start the development server and see your React applica on in ac on.

- Start the development server:

```
npm start
```

This command launches the development server, and you can access your React app at h p://localhost:3000 in your web browser.

✓ **HTML, CSS, and JavaScript**: Basic knowledge of HTML for crea ng the structure of your app, CSS f and JavaScript for client-side interac vity is essen al.

✓ **Version Control**: Use Git for version control, enabling collabora on and tracking chan development process. Pla orms like GitHub or Bitbucket can host your repository.

  - Git: Download and installa on instruc ons can be found at: https://git-scm.com/downloads

✓ **Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

- Visual Studio Code: Download from https://code.visualstudio.com/download
- Sublime Text: Download from https://www.sublimetext.com/download
- WebStorm: Download from https://www.jetbrains.com/webstorm/download

To get the Applica on project from drive:

Follow below steps:

✓ **Get the code:**

Download the code from the drive link given below:

https://drive.google.com/drive/folders/14f9eBQ5W7VrLdPhP2W6PzOU_HCy8UMex?usp=sharing

Install Dependencies:

Navigate into the cloned repository directory and install libraries:

```
cd fitness-app-react npm install
```

✓ **Start the Development Server**:

To start the development server, execute the following command:

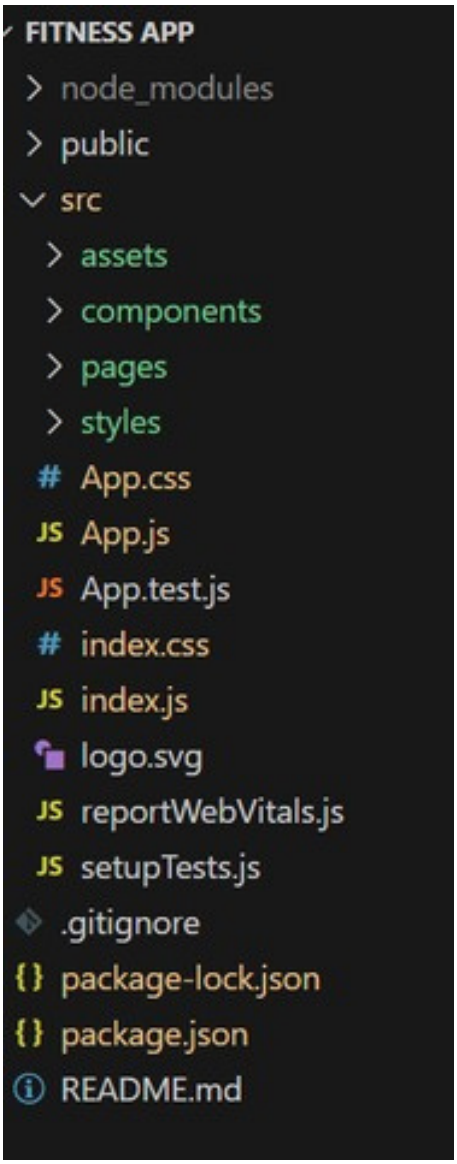
```
npm start
```

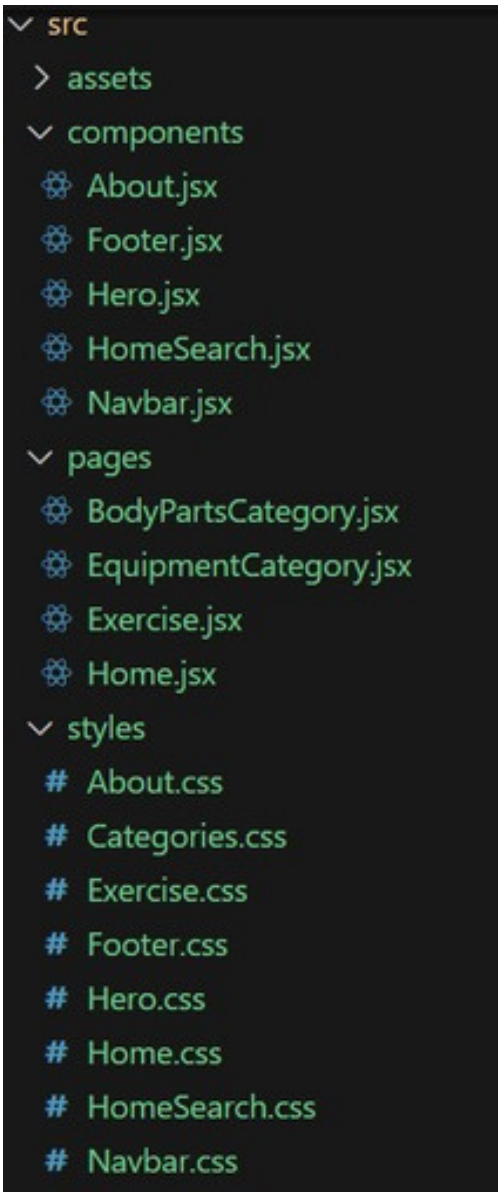
Access the App:

Open your web browser and navigate to http://localhost:3000.
You should see the applica on's homepage, indica ng that the installa on and setup were successful.

You have successfully installed and set up the applica on on your local machine. You can now proceed with further customiza on, development, and tes ng as needed.

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 applica on. The components folder stores all the files, that returns the small components in the applica on. All the styling css files will be stored in the styles folder.

Project Flow Project demo

Before star ng to work on this project, let's see the demo.

link:https://drive.google.com/file/d/1-wYDcG7pYo0x0io_1orJgoByvuW9kYNa/view?usp=drivesdk

Use the code in:

https://drive.google.com/drive/folders/14f9eBQ5W7VrLdPhP2W6PzOU_HCy8UMex?usp=sharing

Milestone 1: Project setup and configuration.

Installa on of required tools:

To build the FitFlex app, we'll need a developer's toolkit. We'll leverage React.js for the interac ve interface, React Router Dom for seamless naviga on, 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
  - https://react.dev/learn/installation
  - https://react-bootstrap-v4.netlify.app/getting-started/introduction/
  - https://axios-http.com/docs/intro
  - https://reactrouter.com/en/main/start/tutorial

Milestone 2: Project Development

- Setup the Rou ng paths

Setup the clear rou ng paths to access various files in the application.

```
<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***.
- Addi onally, we can add the component to subscribe for the newsle er and the footer.
- Now, develop the category page to display various exercises under the category.
- Finally, code the exercise page, where the instruc ons, 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 u lizes the following libraries:

Axios: A popular promise-based HTTP client for JavaScript. You can add a link to the official documenta on for Axios h ps://axios-h p.com/

API Key:

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

bodyPartsOp ons and equipmentOptions:

These variables hold configura on op ons 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 h ps://exercisedb.p.rapidapi.com/exercises/bodyPartList for fetching a list of body parts and h ps://exercisedb.p.rapidapi.com/exercises/equipmentList for fetching a list of equipment.

- *headers:* This sec on 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 func on is responsible for fetching data from the API. It makes use of async/await syntax to handle asynchronous opera ons. First it fetches data for body parts using axios.request(bodyPartsOp ons). Then it stores the fetched data in the bodyParts state variable using setBodyParts.

Similarly, it fetches data for equipment using axios.request(equipmentOp ons). 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 func on 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 par cular category

To fetch the exercises under a par cular 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 func on called fetchData that fetches data from an exercise database API. Here's a breakdown of the code:

const op ons = {...}:

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

- method: Set to 'GET', indica ng that the API request is a GET request to retrieve data from the server.

- url: Set to h ps://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 func on 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 authen ca on. You should replace 'your api key' with a placeholder instruc ng 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 func on 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 a empts to fetch data from the API using axios.request(options).
- The await keyword is used before axios.request(op ons) because the func on 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 a er 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

```
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);
    }

}
```

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

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

async function:

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

fetch request:

Inside the fetchData func on, the fetch API is used to make an HTTP GET request to the API endpoint. The func on 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 par cular 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 func on called *fetchRelatedVideos* that fetches data from YouTube using the RapidAPI service. Here's a breakdown of the code:

fetchRelatedVideos function:

This func on 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 op ons and assigns it an object literal containing configura on details for the API request:

method: Set to 'GET', indica ng a GET request to retrieve data from the server.

url: Set to 'h ps://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 func on at run me. This specifies the search query for YouTube videos.
Other parameters like hl (language), sort (sor ng 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 authen ca on. You should replace 'YOUR_API_KEY' with a placeholder instruc ng users to replace it with their own API key.
'X-RapidAPI-Host': The host of the API, which is '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 a empts 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 a er 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:

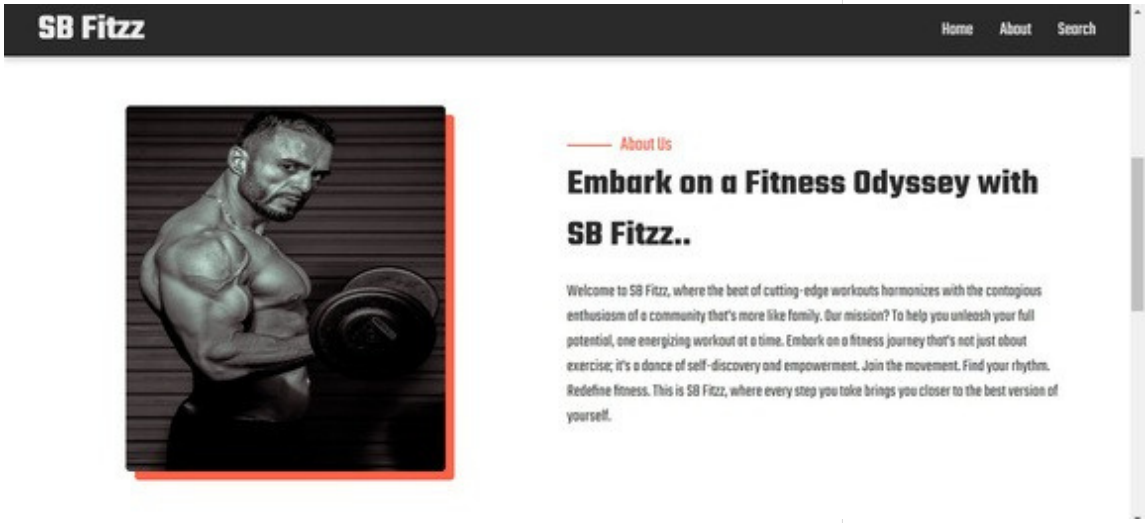
A er comple ng the code, run the react applica on by using the command “npm start” or “npm run dev” if you are using vite.js

Here are some of the screenshots of the application.

?Hero component

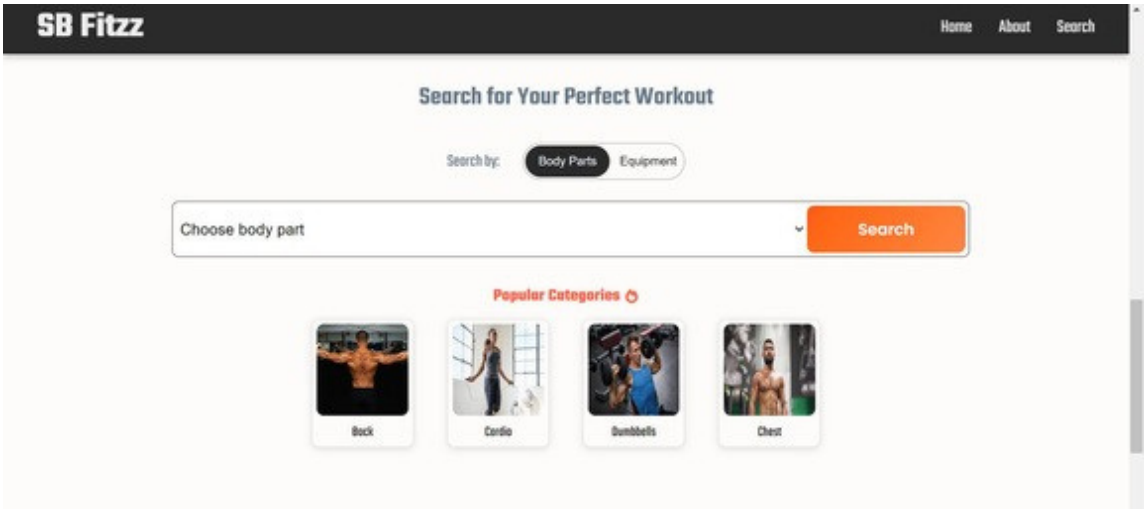
this sec on would showcase trending workouts or fitness challenges to grab users' a en on.

? About



FitFlex isn't just another fitness app. We're me culously designed to transform your workout experience, no ma er your fitness background or goals.

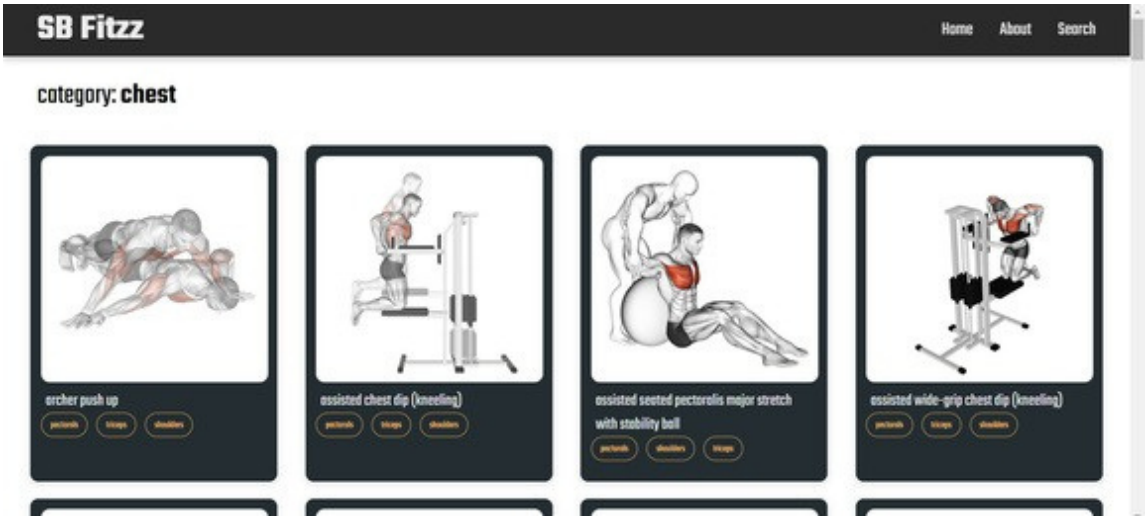
? Search



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.

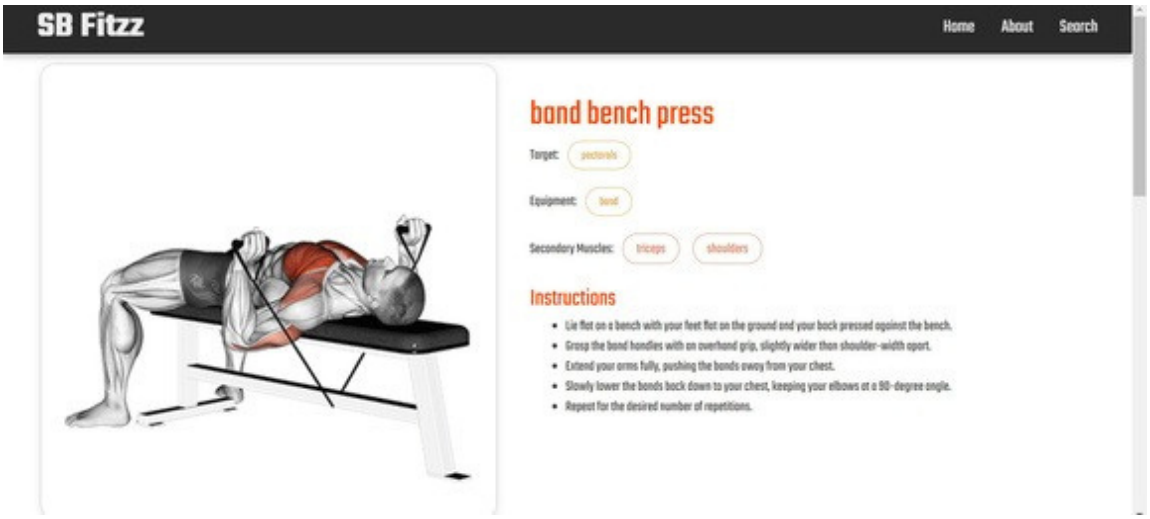
? Category page

FitFlex would offer a dedicated sec on for browsing various workout categories. This could be a grid layout with les showcasing different exercise types (e.g., cardio, strength training, yoga) with icons or short descrip ons 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 instruc ons, accompanied by high-quality visuals like photos or videos demonstra ng proper form. Addi onal details like targeted muscle groups, difficulty level, and equipment requirements (if any) will ensure you have all the informa on needed for a safe and effec ve workout.



Demo link: h ps://drive.google.com/file/d/1mMqMb41RtroiFbUQ-1ZfeYfWJZ6okSNb/view?usp=sharing