# FitFlex: **Your Personal Fitness Companion**(React Application)

**Introduction:**

FitFlex is a revolutionary fitness app designed to transform your workout experience. It offers an intuitive interface, dynamic search, and a vast library of exercises for all fitness levels. Join FitFlex to embark on a personalized fitness journey and achieve your wellness goals.

**Description:**

Welcome to the forefront of fitness exploration with FitFlex! Our innovative fitness app is meticulously designed to revolutionize the way you engage with exercise routines, catering to the diverse interests of both fitness enthusiasts and seasoned workout professionals. With a focus on an intuitive user interface and a comprehensive feature set, FitFlex is set to redefine the entire fitness discovery and exercise experience.

Crafted with a commitment to user-friendly aesthetics, 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 effective workouts from the fitness world.

From those embarking on their fitness journey to seasoned workout aficionados, FitFlex embraces a diverse audience, fostering a dynamic community united by a shared passion for a healthy lifestyle. Our vision is to reshape how users interact with fitness, presenting a platform that not only provides effective exercise routines but also encourages collaboration and sharing within the vibrant fitness community.

Embark on this fitness adventure with us, where innovation seamlessly intertwines with established exercise principles. Every tap within FitFlex propels you closer to a realm of diverse workouts and wellness perspectives. Join us and experience the evolution of fitness engagement, where each feature is meticulously crafted to offer a glimpse into the future of a healthier you.

Elevate your fitness exploration with FitFlex, where every exercise becomes a gateway to a world of wellness waiting to be discovered and embraced. Trust FitFlex to be your reliable companion on the journey to staying connected with a fit and active lifestyle. ♂

**Scenario based Intro:**

You lace up your sneakers, determined to get serious about your fitness. But where do you start? Suddenly, you remember FitFlex, the innovative app that promised to revolutionize your workouts.

Workout plans, diverse exercise categories, and a supportive community. This isn't your typical fitness app. FitFlex feels...different. Intrigued, you select a workout and get ready to experience the future of fitness.

**Project Goals and Objectives:**

The overarching aim of FitFlex is to offer an accessible platform tailored for individuals passionate about fitness, exercise, and holistic well-being.

Our key objectives are as follows:

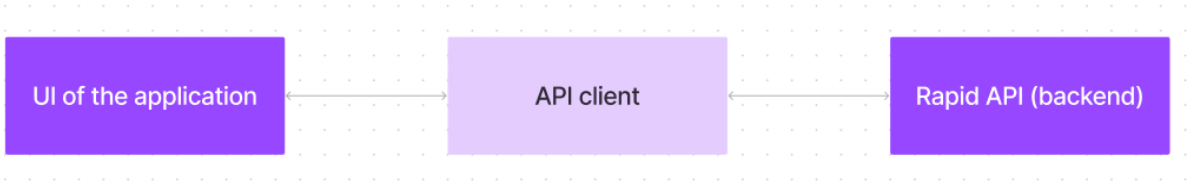
- ✔ **User-Friendly Experience:** Develop an intuitive interface that facilitates easy navigation, enabling users to effortlessly discover, save, and share their preferred workout routines.
- ✔ **Comprehensive Exercise Management:** Provide robust features for organizing and managing exercise routines, incorporating advanced search options 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:**

- ✔ **Exercises from Fitness API:** Access a diverse array of exercises from reputable fitness APIs, covering a broad spectrum of workout categories and catering to various fitness goals.
- ✔ **Visual Exercise Exploration:** Engage with workout routines through curated image galleries, allowing users to explore different exercise categories and discover new fitness challenges visually.

- ✔ **Intuitive and User-Friendly Design:** Navigate the app seamlessly with a clean, modern interface designed for optimal user experience and clear exercise selection.
- ✔ **Advanced Search Feature:** Easily find specific exercises or workout plans through a powerful search feature, enhancing the app's usability for users with varied fitness preferences.

**Technical Architecture:**




FitFlex prioritizes a user-centric approach from the ground up. The engaging user interface (UI), likely built with a framework like React Native, keeps interaction smooth and intuitive. An API client specifically designed for FitFlex communicates with the backend, but with a

twist: it leverages Rapid API. This platform grants access to various external APIs, allowing FitFlex to potentially integrate features like fitness trackers, nutrition data, or workout tracking functionalities 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 application using React.js:

- ✔ **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.

✔ **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.

- Create a new React app:

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

- Navigate to the project directory:

- Running the React App:

  With the React app created, you can now start the development server and see your React application in action.

- Start the development server:

  This command launches the development server, and you can access your React app in your web browser.

✔ **HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

✔ **Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

- Git: Download and installation instructions can be found at:

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

To get the Application project from drive:

Follow below steps:

✔ **Get the code:**

- Download the code from the drive link given below:

**Install Dependencies:**

- Navigate into the cloned repository directory and install libraries:

✔ **Start the Development Server**:

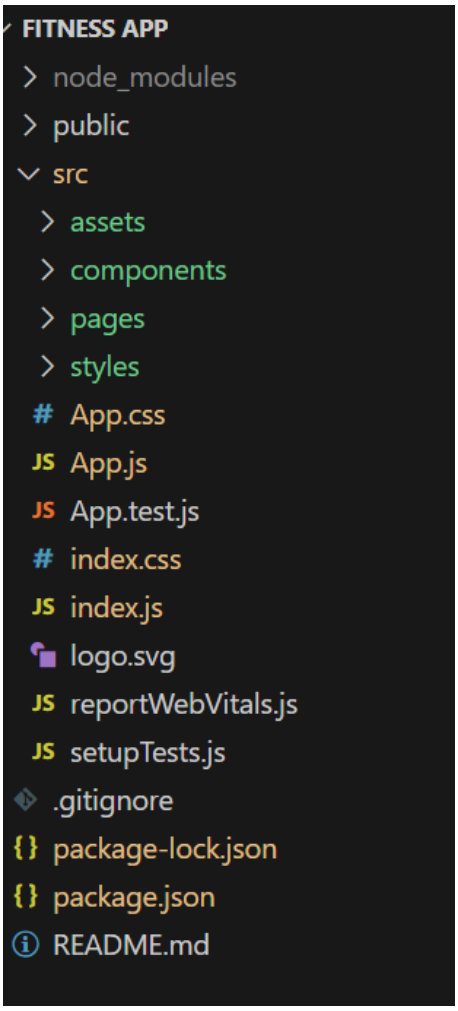
- To start the development server, execute the following command:

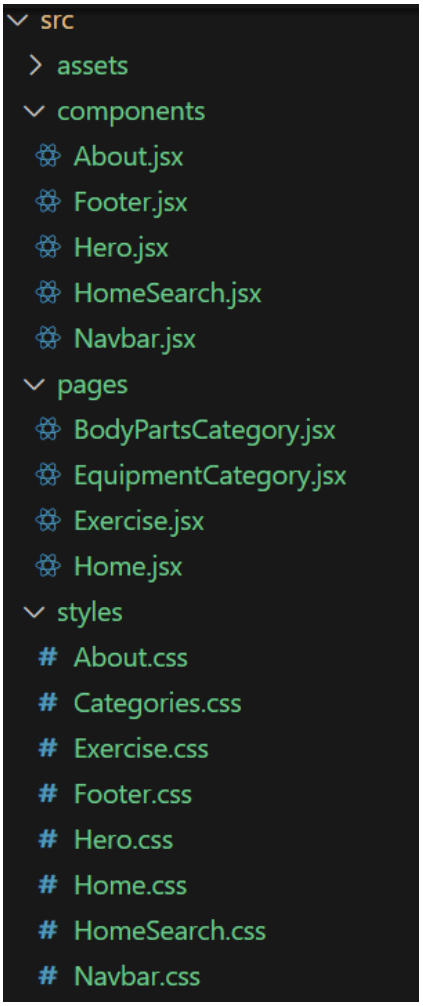
**Access the App:**

- Open your web browser and navigate.
- You should see the application's homepage, indicating that the installation and setup were successful.

You have successfully installed and set up the application on your local machine. You can now proceed with further customization, development, and testing 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 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 3Flow:**

**Project demo:**

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

## 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

## Milestone 2: Project Development

- Setup the Routing paths

  Setup the clear routing 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***.
- 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

*API Key:*

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

*Body Parts Options and equipment Options:*

These variables hold configuration options for fetching data from the Rapid API 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-Rapid API-Key header to provide your API key and the X-Rapid API-Host header specifying the host of the API.

*Fetch Data 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 body Parts state variable using set Body Parts.

Similarly, it fetches data for equipment using axios .request(equipment Options).

Then it stores the fetched data in the equipment state variable using set Equipment. In case of any errors during the API request, the catch block logs the error to the console using console error.

*Use Effect Hook:*

The use Effect hook is used to call the fetch Data 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 Rapid API exercise database using JavaScript's Axios library.

**Fetching exercises under particular category**

To fetch the exercises under a particular category, we

```
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 fetch Data 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.

- 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-Rapid API-Key': Your Rapid API 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-Rapid API-Host': The host of the API, which is 'exercisedb.p.rapidapi.com' in this case.

*const fetch Data = async (id) => {...}:*

This line defines an asynchronous function named fetch Data 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

```
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 details of a particular exercise with API request.

*async function:*

The code defines an asynchronous function named fetch Data 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 fetch Data 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 *fetch Related Videos* that fetches data from YouTube using the Rapid API service. Here's a breakdown of the code:

*Fetch Related Videos 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 Rapid API 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-Rapid API-Key': Your Rapid API 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-Rapid API-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 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:**

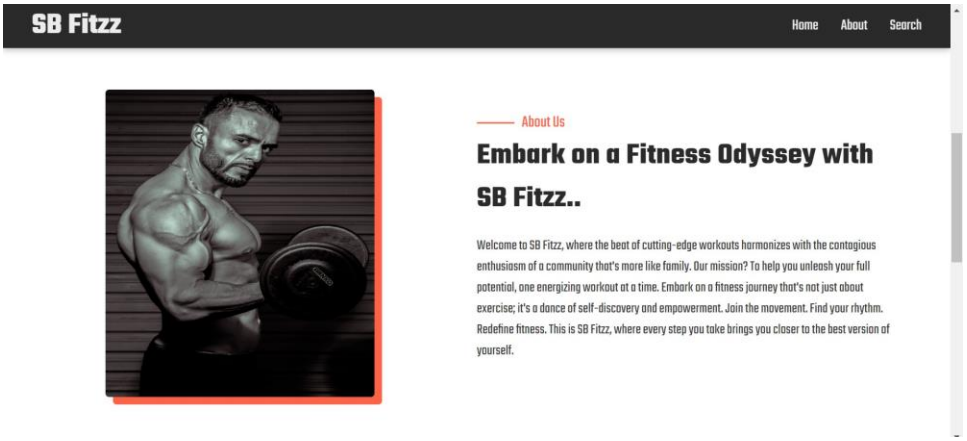
After completing the code, run the react application by using the command “npm start” or “npm run dev” if you are using vite.js

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

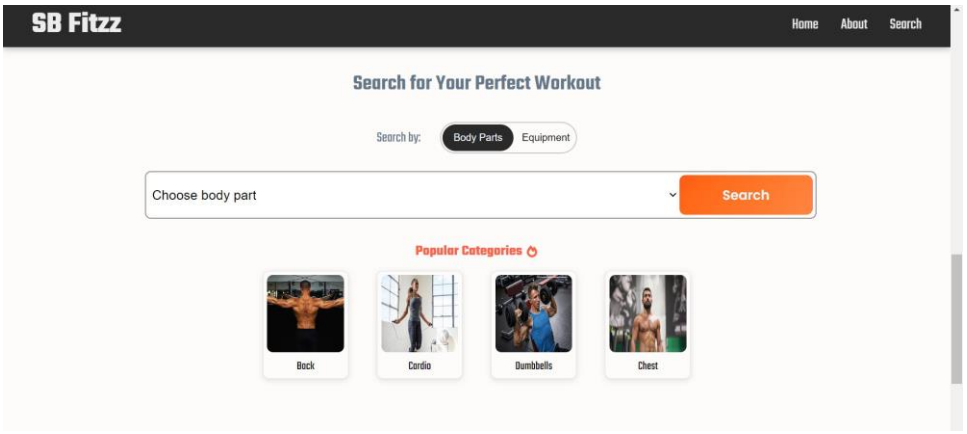
FitFlex isn't just another fitness app. We're




meticulously designed to transform your workout experience, no matter your fitness background or goals.

## Search

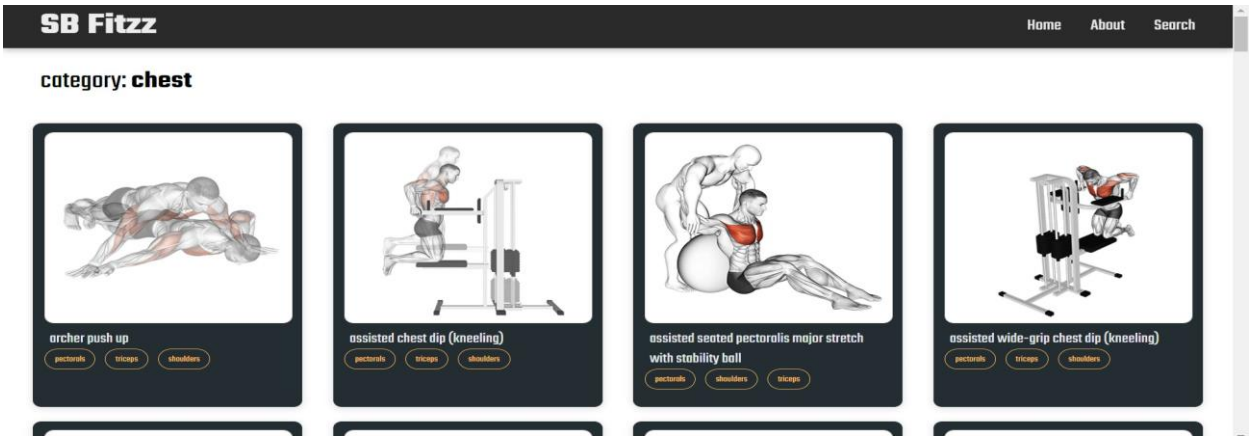
B Fitzz makes finding your perfect workout effortless. Our prominent search bar empowers you to explore exercises by keyword, targeted




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.

SB Fitzz

Home About Search

## bond bench press

Target: pectorals

Equipment: band

Secondary Muscles: triceps shoulders

### Instructions

- Lie flat on a bench with your feet flat on the ground and your back pressed against the bench.
- Grasp the band handles with an overhand grip, slightly wider than shoulder-width apart.
- Extend your arms fully, pushing the bands away from your chest.
- Slowly lower the bands back down to your chest, keeping your elbows at a 90-degree angle.
- Repeat for the desired number of repetitions.