Frontend Development with React.js Project Documentation for Fit Flex

1. **Introduction**
   * **Project Title**: **fit flex**
   * **Team Members**:

 **Pooja S** (**Team Leader**) [Email Id:poojas22cs059@gmail.com]

 **Pradisha V** [Email Id: pradishav22cs060@gmail.com]

 **Prathiksha G V** [Email Id:prathikshagv22cs061@gmail.com]

 **Priya Dharshini T** [Email Id:priyadharshinit22cs061@gmail.com]

* **Team ID:** SWTID1741248974149812(Team-149812)

# Project Overview

* + **Purpose**:

Fit Flex is a versatile platform designed to enhance fitness experiences by providing personalized workout plans, tracking progress, and offering a community-based environment for users. Its primary purpose is to help individuals achieve their fitness goals through tailored exercise routines, progress monitoring, and access to a wide range of workouts and challenges. The platform promotes flexibility, enabling users to adjust their routines according to their personal fitness levels and preferences, while fostering a supportive community that encourages motivation and accountability.

# Features:

* + - Personalized Workout Plans.
    - Progress Tracking.
    - Workout Library.
    - Challenges and Goal.

# Architecture

* + **Component Structure**:

A compound structure for **Fit Flex** would consist of interconnected features, modules, and functionalities working together seamlessly to deliver a holistic fitness experience.

* + - User Profile Module.
    - Workout Library.
    - Challenges and goal Module.
    - Community Interaction Module.
    - Nutrition and Wellness Module.
    - Progressive Difficulty and Adaptability Module.

# State Management:

# State management for FitFlex involves tracking user data, workout progress, and preferences in real-time, ensuring seamless synchronization across devices. It ensures a dynamic, personalized experience by updating and storing information on user goals, routines, and fitness milestones.

# Routing:

* **Home** (/home): The main dashboard displaying personalized workout recommendations, progress, and goals.
* **Workouts** (/workouts): A page with the workout library, where users can browse exercises or select routines.
* **Profile** (/profile): Displays user information, progress tracking, and settings for customization.
* **Challenges** (/challenges): A section to view and join fitness challenges.
* **Community** (/community): A page for users to engage with others, share achievements, and join support groups.
* **Nutrition** (/nutrition): Offers meal plans, nutrition tips, and food tracking.

# **Settings** (/settings): Allows users to adjust preferences, integrate with devices, or manage account details.

# Setup Instructions

* + **Prerequisites**:
    - Node.js (v16 or higher)
    - npm (v8 or higher)
    - Git

# Installation:

1. Clone the repository: git clone [https://github.com/unm12912137/rhythmic-](https://github.com/unm12912137/rhythmic-tunes.git) [tunes.git](https://github.com/unm12912137/rhythmic-tunes.git)
2. Navigate to the client directory: cd rhythmic-tunes/client
3. Install dependencies: npm install
4. Configure environment variables: Create a .env file in the client directory and add the necessary variables (e.g., API keys).
5. Start the development server: npm start

# Folder Structure

* + **Client**:
* **src/**: Contains all the source code for the application.
* **api/**: Handles API requests for different modules like workout, nutrition, and user data.
* **assets/**: Stores static resources like images and global styles.
* **components/**: Reusable components like buttons, headers, cards, and other UI elements.
* **contexts/**: Manages global state using React Context for authentication, workouts, nutrition, etc.
* **pages/**: Each page of the application, such as the home page, user profile, workout page, etc.
* **utils/**: Contains helper functions for validation, data formatting, or any other utilities used across the app.
* **App.js**: The main app file that integrates everything together, including routing and global settings.
* **index.js**: The entry point for the app that initializes the application.
* **routes.js**: Defines the routing logic for navigating between different pages.

#  **package.json**: Defines the project's dependencies and scripts, like how to start the server or run tests.Utilities:

* + - **api.js**: Handles API requests to the backend.
    - **auth.js**: Manages user authentication and token storage.
    - **hooks/usePlayer.js**: Custom hook for managing the music player state.

# Running the Application Frontend:

* To start the frontend server, run the following command in the client directory: npm start
* npm install
* npx json-server ./db/db.json
* npm run dev
* The application will be available at [http://localhost:3000](http://localhost:3000/)

1. **Components Documentation**

* **Key Components:**
* **Personalized Fitness Plans**
* **User Data** (age, fitness level, goals, preferences)
* **Tailored Workout Routines** (strength, cardio, flexibility)
* **Adaptive Progression** (modifying routines as users improve)
* **Workout and Exercise Library**
* **Diverse Exercise Categories** (strength, yoga, HIIT, mobility, etc.)
* **Video Demonstrations** (for proper form and technique)
* **Search and Filter Functionality** (e.g., difficulty, body part focus, duration)
* **Progress Tracking and Analytics**
* **Fitness Metrics** (e.g., calories burned, steps taken, reps completed)
* **Visual Progress Charts** (graphs showing improvements in strength, endurance, etc.)
* **Milestone Achievements** (badges, streaks, etc.)

# Reusable Components:

* + - **Button**: A customizable button component.
      * Props: text, onClick, disabled.
    - **Input**: A reusable input field for forms and search.
      * Props: type, placeholder, value, onChange.

# State Management

* + **Global State**:

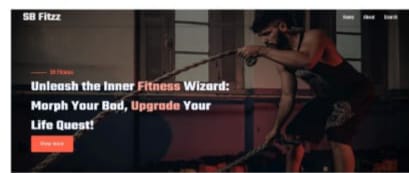
The global state for **Fit Flex** can be managed using React's Context API or another state management solution like Redux, to ensure seamless data flow across the application. The global state would include several key contexts, such as **AuthContext** for user authentication and profile management, **Workout Context** to track workout routines, progress, and selected exercises, and **Nutrition Context** to manage meal plans, calorie tracking, and nutrition goals. These contexts allow various components to access and update shared data, such as user login status, workout completion, and diet plans, without the need for prop drilling. This centralized state management enhances user experience by providing real-time updates and ensuring consistency across different sections of the app, such as workouts, challenges, and profile settings.

# Local State:

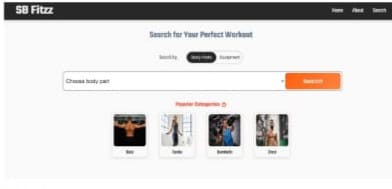
Local state in **Fit Flex** is used for managing component-specific data, such as form inputs, toggle states, or UI interactions like button clicks and modal visibility. It ensures that each component has its own isolated state, independent of the global context, for a more responsive and dynamic user interface.

# User Interface

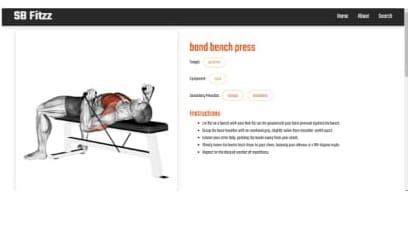
* + **Screenshots:**
* **Home Page:**



* **Search Page:**

****

* **Sample Page:**

****

# Styling

* + **CSS Frameworks/Libraries**:

The application uses **Styled-Components** for styling. This allows for modular and scoped CSS within components.

# Theming:

A custom theme is implemented using Styled-Components, with support for light and dark modes.

# Testing

* + **Testing Strategy**:
    - **Unit Testing:** Using **Jest** and **React Testing Library**.
    - **Integration Testing**: Is performed to ensure that components work together as expected.
    - **End-to-End Testing: Cypress** is used for end-to-end testing of user flows.

# Code Coverage:

* + - Code coverage is monitored using Jest’s built in coverage tool. The current coverage is 85%.

# Screenshots or Demo

* + **Demo Link:**
  + <https://drive.google.com/file/d/1MGfd-d2iG8HTVV44nUGlzFqKn6IO4qX3/view?usp=sharing>
  + **Screenshots:** See section 9 for UI screenshots.

# Known Issues

* **Performance Issues with Large Workouts**: As the app scales with more complex workout routines, managing and rendering large sets of exercises, progress data, or media (like videos) may lead to slower performance or lag, particularly on lower-end devices.
* **Synchronization of User Data**: Ensuring that user data (like workout progress, nutrition logs, or settings) syncs seamlessly across different devices or browsers can be challenging, leading to potential inconsistencies or delays in data updates.

# Future Enhancements

* **Future Features**:
  + **Integration with Wearable Devices**: Enhancing FitFlex with deeper integration for fitness trackers (like Fitbit, Apple Watch, or Garmin) would allow for real-time tracking of heart rate, steps, calories burned, and sleep patterns, providing a more personalized fitness experience.
  + **AI-Driven Workout Personalization**: Incorporating AI to analyze user progress and adapt workout plans automatically based on performance, goals, and feedback could offer more dynamic and individualized routines, enhancing engagement and results.
  + **Social and Community Features**: Expanding social features, such as allowing users to create groups, share progress, participate in live challenges, or interact through fitness forums, could increase motivation and foster a sense of community within the app.