Project Documentation
FITFLEX

1.INTRODUCTION

- **Project Title:** FitFlex:Your Personal Fitness Companion
- **Team ID** : NM2025TMID39345
- **Team Leader: Meganathan.M /** meganathan01032007@gamail.com
- **Team Member:**
  - **NAME**: Tharun kumar.R **GMAIL**:tharuntharun9600@gamil.com
  - **NAME**: Mohammed mujahith.M **GMAIL**: **mohammedmujahith78@gmail.com**
  - **NAME**:. Thivakar.T **GMAIL**: thivakar975@gamil.com

2. Project Overview

The frontend project is a fitness-related application built with React. It provides a modern user interface where users can search for exercises by body part or equipment category and view exercise details. It uses a modular component structure for scalability and maintainability.

3. Architecture

- **Frontend**: Built with React.js, leveraging Bootstrap and custom CSS for UI.
- **Routing**: Handled with React Router for seamless page navigation.
- **Components**: Reusable components such as `Navbar`, `Hero`, `Footer`, and search forms.
- **Assets**: Includes images and video used for UI presentation.
- **State Management**: Uses React state hooks and component-level state for managing UI data.

4. Setup Instructions

- **Prerequisites**:
  - Node.js and npm installed
  - Git installed
- **Installation Steps**:

1. Clone the repository:
   `git clone <repo-url>`
2. Navigate to project directory:
   `cd code`
3. Install dependencies:
   `npm install`
4. Start development server:
   `npm start`
5. Open `http://localhost:3000` in a browser.

5. Folder Structure

```
code/
├── src/
│   ├── components/    # Reusable UI components (Navbar, Hero, Footer, etc.)
│   ├── pages/         # Pages like Home, Exercise, Categories
│   ├── styles/        # CSS files for component styling
│   └── assets/        # Images and video files
├── public/            # Static public files like index.html
├── package.json       # Project metadata and dependencies
├── README.md          # Project overview and instructions
```

6. Running the Application

- Start frontend development server:
  `npm start`
- Access the application at:
  `http://localhost:3000`

7. API Documentation

The frontend communicates with external APIs to fetch exercise data dynamically.
Key interactions:

- Uses `axios` to send HTTP GET requests to RapidAPI's ExerciseDB API.
- Example endpoint used:
  `https://exercisedb.p.rapidapi.com/exercises/exercise/{id}` – retrieves detailed exercise data by ID.
- Related videos are fetched using another API call based on the exercise name.
- API key and headers are set in the requests for authentication with the external service.

---

8. Authentication

Currently, the frontend does not implement user authentication features.
However, code suggests planned functionality:

- Navigation relies on `useNavigate` and React Router to manage page routing.

- Conditional rendering in the `Navbar.jsx` is based on the current URL and scroll position, indicating intention for route-based access control or dynamic menus.
- Future plans may include implementing login forms, JWT token storage, and protected routes.

---

9. User Interface

Implemented UI components (based on inspected code):

- **Hero Component**: Likely displays introductory content with visuals.
- **About Component**: Provides information about the app purpose.
- **HomeSearch Component**: Search bar to find exercises by body part or equipment category.
- **Navbar Component**:
  - Dynamically applies CSS class based on scroll position.
  - Provides navigation to Home and other pages.
- **Exercise Component**:
  - Displays detailed exercise info by fetching data from an external API based on exercise ID from URL parameters.
  - Displays related exercise videos.
- Responsive behavior is partly managed by CSS and conditional rendering logic in components.

---

10. Testing

Currently, no automated tests are implemented in the code base.
Observed files:

- `setupTests.js` exists, suggesting readiness to integrate Jest and React Testing Library.
Planned testing strategy:
- Add unit tests for React components like `Navbar`, `Hero`, and `Exercise`.
- Integration tests to verify API interaction.
- Manual testing is used during development to check navigation, API data fetching, and UI rendering.

11Known Issues

- The application relies on static data; no backend API connection implemented yet.
- Responsive design is basic and may have issues on very small screen sizes.

12.Future Enhancements

- Integrate a backend API to serve exercise data dynamically.
- Add authentication for user login and personalized features.
- Improve responsiveness for mobile devices.
- Implement unit testing for critical components.