Project Documentation

Fit Flex - Your Personal Fitness Companion

**1. Introduction**

**•Project Title: Fit Flex - Your Personal Fitness Companion**

**•Teams ID: NM2025TMID30003**

**•Team Leader: PRIYANGA B -** [**202400442@sigc.edu**](mailto:202400442@sigc.edu)

**• Team Members:**

**• USHALINI R -** [**202400788@sigc.edu**](mailto:202400788@sigc.edu)

**• NIVETHA K -** [**202400333@sigc.edu**](mailto:202400333@sigc.edu)

**• MONIKA R -** [**202400129@sigc.edu**](mailto:202400129@sigc.edu)

**2. Project Overview**

**· Purpose: Fit Flex is a modern, responsive web application designed to help users achieve their fitness goals. It allows users to browse workout routines, track their progress, save favorite exercises, and manage their personal fitness calendar.**

**Features:**

**· User authentication (Login/Logout):**

**• The application includes a secure authentication system that allows users to create a personal profile. This feature is crucial for saving progress, favorite workouts, and personalized settings.**

**• Login: Users can log in with their credentials to access their dashboard and private data.**

**• Logout: A logout function securely ends the user's session, protecting their personal information.**

**· Browse and filter workout routines by category (e.g., Cardio, Strength, Yoga)**

**• Fit flex offers an extensive library of workouts that users can easily navigate.**

**• Categories: Workouts are organized into clear categories such as Cardio, Strength Training, and Yoga, allowing users to quickly find routines that match their fitness goals.**

**• Filtering: Users can apply filters to narrow down their search based on intensity level, equipment needed, or duration.**

**· View detailed exercise instructions with animations**

**• Each workout routine includes detailed guidance for proper execution.**

**• Instructions: Every exercise comes with clear, step-by-step text instructions to ensure correct form and technique.**

**• Animations: To provide a better visual guide, each exercise description is accompanied by a GIF or short animation demonstrating the movement. This helps users perform exercises safely and effectively.**

**· Save favorite workouts to a personal list**

**• Users can build their own custom library of preferred workouts.**

**• Personal List: With a simple click, users can save any workout routine they like to their personal "Favorites" list.**

**• Easy Access: This feature allows for quick access to go-to workouts without having to search for them repeatedly.**

**· Interactive calendar to schedule and track workout sessions**

**• The application includes a progress tracking tool to help users stay motivated.**

**• Scheduling: Users can use the interactive calendar to schedule their workout sessions in advance.**

**• Tracking: After completing a workout, they can mark it as finished on the calendar. This feature helps users visualize their progress over time and maintain a consistent routine.**

**· Responsive design for desktop, tablet, and mobile**

**• Fit flex is built with a responsive design approach, ensuring a seamless user experience across different devices.**

**• Desktop: The interface is optimized for larger screens, providing a spacious and detailed view of all features.**

**• Tablet: The layout automatically adjusts to tablet dimensions, making it easy to use on devices like iPads.**

**• Mobile: The application's design adapts to smaller mobile screens, with a clean and intuitive interface that allows users to access all features on the go. This ensures that users can work out and track their progress from anywhere.**

**3. Architecture**

**• Component Structure: The app uses a functional component structure with a hierarchy starting from App.js. Major page-level components (e.g., Home Page, Workouts Page) render reusable presentational components (e.g., Workout Card, Button).**

**• State Management: Global state (user authentication, favorites, workout data) is managed using Context API (Auth Context, Workout Context). Complex form state and component-specific state are handled with the use State and use Reducer hooks.**

**• Routing: Client-side routing is implemented using React Router v6. Protected routes redirect unauthenticated users to the login page.**

**• The Fit flex application follows a client-server architecture, specifically a Single-Page Application (SPA) model. The front-end, built with React.js, and the back-end, which serves data via an API, are separate components that communicate with each other.**

**4. Setup Instructions**

**· Prerequisites:**

**• Node.js: Version 18 or higher is required.**

**• npm or yarn: A package manager to install the project dependencies.**

**• Git: Essential for cloning the repository and for version control.**

**• Web Browser: A modern browser like Chrome, Firefox, or Edge for testing and usage.**

**• A text editor or IDE: An environment for working with the code (e.g., VS Code).**

**• API Key: An API key for a fitness API is needed and must be configured in a .env file.**

**Key Technologies:**

**•The application's front-end is built with the following technologies, so familiarity with them is beneficial:**

**• React.js: The primary framework for the user interface.**

**• React Libraries:**

**• react-router-dom: For handling navigation and routing.**

**• react-icons: For adding icons to the UI.**

**• Tailwind CSS: Used for styling the application with a utility-first approach.**

**· Node.js (v16 or higher)**

**• JavaScript Runtime: Fit flex is a web application built using JavaScript, and Node.js is the environment that allows that JavaScript code to be executed outside of a web browser.**

**• Installing Node.js also installs npm, which is essential for managing the project's dependencies. The Fit flex code relies on various libraries and packages (e.g., React, Tailwind CSS), and npm is used to download and install them.**

**· npm or yarn**

**• npm is the default package manager that comes with Node.js. It's widely used and has a large ecosystem.**

**• yarn is an alternative package manager developed by Facebook. It's known for its speed and reliability.**

**• Regardless of which one you choose, the steps to install the project dependencies are straightforward:**

**• If using npm: Run npm install in the project's root directory.**

**• If using yarn: Run yarn install in the project's root directory.**

**Both commands will read the package. jSon file and download all the required packages to a node \_modules folder, allowing you to run the application.**

**· Installation:**

**1. Clone the repository: git clone https://github.com/your-username/fitflex-app.git**

**2. Navigate to the project directory: cd fit flex-app**

**3. Install dependencies: npm install**

**4. Create a .env file in the root directory and add necessary environment variables (e.g., REACT\_APP\_API\_KEY=your key).**

**5. Start the development server: npm start**

**5. Folder Structure**

**```**

**src/**

**├── components/ # Reusable UI components**

**│ ├── ui/ # Button, Card, Modal, Layout**

**│ ├── workout/ # Workout Card, Exercise Detail**

**│ └── common/ # Header, Footer, Loading Spinner**

**├── pages/ # Page-level components**

**│ ├── Home.js**

**│ ├── Workouts.js**

**│ └── Profile.js**

**├── context/ # React Context for state management**

**│ ├── AuthContext.js**

**│ └── WorkoutContext.js**

**├── hooks/ # Custom React hooks**

**│ ├── useLocalStorage.js**

**│ └── useFetch.js**

**├── utils/ # Helper functions & constants**

**├── assets/ # Images, icons, styles**

**├── styles/ # Global & module CSS files**

**└── App.js**

**```**

**6. Running the Application**

**· Frontend: From the project root directory, run npm start. The application will open in your browser on http://localhost:3000.**

**7. Component Documentation**

**· Key Components:**

**1. <Workout Card />: Displays a preview of a workout. Receives workout (object), on Favorites (function), and is Favorite (boolean).**

**Props:**

**• workout (object, required): An object containing the workout data to be displayed, including properties such as name, description, and image.**

**• on Favorite (function, required): A callback function that is triggered when the user clicks the favorite button on the card.**

**• is Favorite (boolean, required): A boolean flag that determines the visual state of the favorite button (e.g., filled or outlined icon).**

**2. <Exercise Detail Modal />: A modal that shows detailed instructions and a GIF for a selected exercise. Controlled by is Open (boolean) and on Close (function).**

**Props:**

**• is Open (Boolean, required): A boolean that controls whether the modal is visible (true) or hidden (false).**

**• on Close (function, required): A callback function that closes the modal. This is triggered when the user clicks the close button or the modal's overlay.**

**• exercise (object, required): An object containing the data for the exercise to be displayed, including a gif url, instructions, and name.**

**3. Reusable Components:**

**<Button />: A styled button component. Accepts variant (e.g., 'primary', 'secondary'), on Click, and children.**

**Props:**

**• variant (string, optional): A string that defines the button's style. Common variants include 'primary' (for main actions) and 'secondary' (for alternative actions).**

**• on Click (function, required): A function to be executed when the button is clicked.**

**• children (node, required): The content to be displayed inside the button (e.g., text, icons).**

**· <Protected Route />: A wrapper component that checks user authentication before rendering child components.**

**Props:**

**• children (node, required): The components to be rendered if the authentication check passes.**

**8. State Management**

**· Global Component State:**

**• The Workout Context provides workout data and user favorites to any component in the tree, preventing prop drilling. State is updated via functions defined in the context.**

**• React Context API: This is a built-in feature of React that allows data to be passed down through the component tree without having to manually pass props at every level. It's often used for managing global data like authentication status or theming. The application likely uses the Context API to make the current user's information available to components like <Header /> and <Profile />.**

**• Local Component State:**

**• Component-specific data (e.g., whether a modal is open or the value of an input field), fit flex uses Reacts use State hook. This approach is sufficient for managing state that only affects a single component and doesn't need to be shared with its siblings or a distant parent.**

**9. User Interface**

**(This section would contain screenshots or GIFs of the main pages: Home, Workout list, Exercise detail modal, and User Profile.)**

**10. Styling**

**· CSS Frameworks/Libraries: The application is styled using CSS Modules for component-scoped styles, combined with Sass/SCSS for variables and mixins.**

**· Theming: A basic light/dark theme is implemented using CSS custom properties (variables) that are toggled via a context.**

**•Tailwind Works:**

**• Instead of writing a CSS rule for a class like card-container, you add pre-defined utility classes directly to your HTML elements. For example, to style a workout card, you might use classes like:**

**• bg-white: Sets the background color to white.**

**• p-6: Adds padding of size 6.**

**• rounded-lg: Applies a large border-radius to round the corners.**

**• shadow-md: Adds a medium-sized box shadow.**

**• flex and flex-col: Apply flexbox properties for layout.**

**11. Testing**

**· Testing Strategy: Component unit testing is done with Jest and React Testing Library. Tests focus on user interactions and ensuring components render correctly with given props.**

**· Code Coverage: The goal is to maintain above 80% test coverage for utility functions and core components. Coverage is tracked using the built-in coverage tool in Jest.**

**12. Screenshots or Demo**

**· Live Demo:** [**https://drive.google.com/file/d/1AYob51\_Pj6LN9-RETVJCLmXB4egH\_bQF/view?usp=drivesdk**](https://drive.google.com/file/d/1AYob51_Pj6LN9-RETVJCLmXB4egH_bQF/view?usp=drivesdk)

**· Screenshot Gallery:**



**13. Know Issues**

**· Nil**

**14. Future Enhancements**

**· Integrate a backend API for persistent data storage.**

**· Add social features like sharing workouts and following friends.**

**· Implement a Pomodoro timer for timed workouts (HIIT, AMRAP).**

**· Develop a progressive web app (PWA) for offline functionality.**

**· Add video demonstr**ations for exercises alongside GIFs.