Project Documentation

1.Project Title: FitFlex: Your Personal Fitness Companion

2. Team ID: NM2025TMID30047

3. Team Leader: MYTHILI A

Leader Mail ID : [202400534@sigc.edu](mailto:202400534@sigc.edu)

5. Team Members:

* NIGILESHWARI P - [202400789@sigc.edu](mailto:202400789@sigc.edu)
* SUBATHRA S - [202400913@sigc.edu](mailto:202400913@sigc.edu)
* POOSHITHA R - [202400791@sigc.edu](mailto:202400791@sigc.edu)

**2. Project Overview**

· Purpose: FitFlex is designed to be a user-friendly fitness tracking application. Its goal is to help users create, manage, and track their workout routines and personal fitness goals, providing a centralized hub for their fitness journey.

· Features:

· User authentication (Login/Logout)

· View and filter a library of exercises

· Create and manage personalized workout routines

· Track workout history and progress

· Responsive design for mobile and desktop.

**3. Architecture**

· Component Structure: The app uses a functional component structure with hooks. Major components are organized into pages (e.g., HomePage, Dashboard), which are composed of smaller, reusable UI components (e.g., Button, ExerciseCard).

· State Management: Global state is managed using the Context API (e.g., AuthContext for user state, WorkoutContext for workout data). Local component state is handled with the useState and useReducer hooks.

· Routing: React Router v6 is used for client-side routing, enabling navigation between pages without full page reloads.

**4. Setup Instructions**

· Prerequisites:

· Node.js (v16 or higher)

· npm or yarn

· Installation:

1. Clone the repository: git clone <https://github.com/Mythili534/Fitflex.git>

2. Navigate to the project directory: cd fitflex-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\_BASE\_URL).

5. Start the development server: npm start

**5. Folder Structure**

src/

├── components/ # Reusable UI components (Button, Card, Navbar)

│ ├── ui/

│ └── exercise/

├── pages/ # Top-level page components (Home, Dashboard, Login)

├── contexts/ # React Context providers (AuthContext, WorkoutContext)

├── hooks/ # Custom React hooks (useLocalStorage, useApi)

├── utils/ # Helper functions and constants (apiClient, formatters)

├── assets/ # Static files (images, icons)

├── styles/ # Global and 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:3001](http://localhost:3001/).

**7. Component Documentation**

· Key Components:

· <Dashboard />: The main hub after login. Fetches and displays the user's recent workouts and statistics. Receives no props.

· <WorkoutForm />: A form for creating and editing workouts. Receives onSubmit function and optional initialData props.

· Reusable Components:

· <ExerciseCard />: Displays an exercise's name, image, and muscle group. Props: exercise (object), onClick (function).

· <Modal />: A reusable modal dialog. Props: isOpen (boolean), onClose (function), title (string), children.

**8. State Management**

· Global State: The AuthContext manages the user's authentication token and profile data. The WorkoutContext manages the list of exercises and the user's created workouts, reducing prop drilling.

· Local State: Components like WorkoutForm use the useState hook to manage form input fields and validation errors locally.

**9. Styling**

· CSS Frameworks/Libraries: The application is styled using CSS Modules for component-scoped styles, avoiding naming conflicts.

· Theming: A basic CSS custom properties (variables) theme is implemented for consistent colors (--primary-color, --text-color) across the application, allowing for easy future theming.

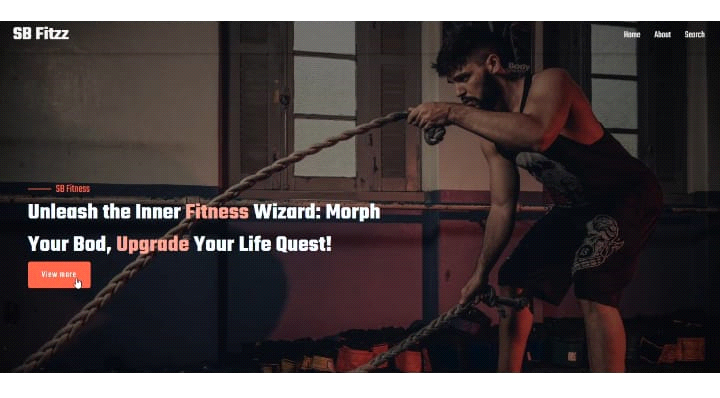
**10. Testing**

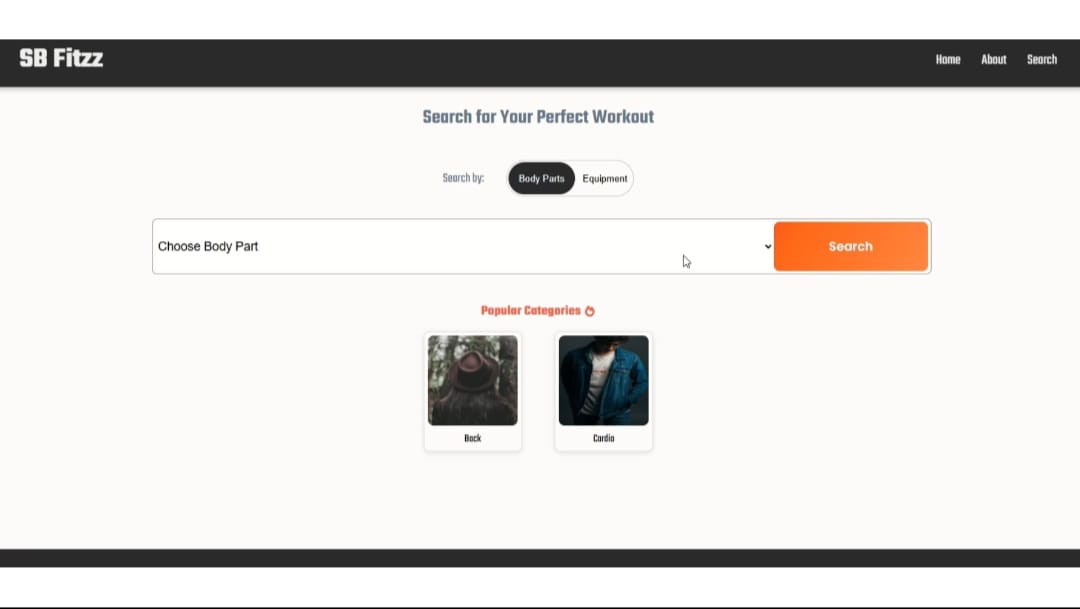
· Testing Strategy: Component unit testing is done with Jest and React Testing Library. Focus is on testing component rendering and user interactions.

· 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.

**11. Screenshots or Demo**

· Live Demo: <https://drive.google.com/file/d/1ZzMOoSg_ckvDXC3vVll3XACTSmLrTBTL/view?usp=drivesdkmn>





**12. Known Issues**

· On slow networks, the exercise list might load before images, causing a layout shift.

· The workout timer feature occasionally does not reset correctly when navigating away mid-workout.

**13. Future Enhancements**

· Social features: Ability to share workouts and follow friends.

· Integration with wearable devices for automatic activity tracking.

· Advanced data visualization for progress tracking (charts/graphs).

· A native mobile version using React Native.