**FitFlex:Your Personal Fitness Companion**

1. **Introduction**

**1.1 Project Tittle:**

**FitFlex:** YourPersonal Fitness Companion

**1.2 Team Detail:**

**Team ID:**

|  |  |  |
| --- | --- | --- |
| **Team Leader** | **PRIYANKA K** | **kramam2616@gmail.com** |

|  |  |  |
| --- | --- | --- |
| **Team member 1** | **PRIYANKA K** | kramam2616@gmail.com |
| **Team member 2** | **VIDHYA V** | vvidhya142006@gmail.com |
| **Team member 3** | **SUPRIYA N** | supriyaraju9707@gmail.com |
| **Team member 4** | **RASHIGA R** | rashigarajendranrashigarajendr@gmail.com |
| **Team member 5** | **YAMINI SRI A** | yaminisri2007@gmail.com |

**2.Project Overview**

**Purpose:**

The purpose of this project is to develop a fitness application that enables users to track workouts, monitor health metrics, and follow personalized fitness plans, thereby supporting them in achieving their health and fitness goals effectively.

**Features:**

The key features and functionalities of the frontend application, such as:

* **User Profile & Goals:** personal details,fitness goals,activity leavel& experience,target metrics,
* **Workouts Planning:** pre-designed workout plans,custom workout builder,video demonstration& animations,daily/weekly schedule,rest day management.
* **Exercise Logging:** Track reps,sets,weights,time,distance,log cardio,strength,flexibility workouts,voice inputs or bar code scanning.
* **Progress Tracking:** Weight tracker,body measurements,before/after photos,performance trends
* **Integration:** Sync with apple health,google fit,fitbil,garmin,etc.

sync with smartwatches or fitness bands,calender sync for workout

Schedule.

**3.Architecture:**

* **Component Structure:** Outline the structure of major react components and how they interact.
* **State Management:** Describe the state management approach used.
* **Routing:** Explain the routing structure if using react-router or another routing library.

**4.Setup Instructions:**

## **Prerequisites:**

**Node.js** (v14 or later)

**npm** or **yarn**

**Git**

**MongoDB** (if using a local database)

**Python 3+** (if the backend uses Python – optional)

**Android Studio / Xcode** (if building a mobile app

**React Native CLI / Expo CLI** (for React Native apps)

**.env configuration file** (see below)

1. **Installation Steps:**

### 1. Clone the Repository

git clone https://github.com/yourusername/FITFLEX.git

cd FITFLEX

2. Set Up Environment Variables

Create a .env file in the root directory with your environment settings. Example:

REACT\_APP\_API\_URL=http://localhost:5000/api

MONGO\_URI=mongodb://localhost:27017/fitflex

JWT\_SECRET=your\_jwt\_secret\_key

1. **Install Dependencies:**

#### For the frontend (React / React Native)

cd frontend

npm install

#### For the backend (Node.js / Express)

cd backend

npm install

If using Python (e.g., Django/Flask), run:

pip install -r requirements.txt

1. **Run the Application:**

#### **Start Backend Server**

cd backend

npm start

# or

nodemon index.js

#### Start Frontend Development Server

cd frontend

npm start

#### For React Native (Mobile)

Using Expo:

npm install -g expo-cli

expo start

**✅ Verification**

Once everything is up and running:

* Frontend: [http://localhost:3000](http://localhost:3000/)
* Backend API: <http://localhost:5000/api>

You should see the FITFLEX welcome screen or homepage.

**🧪 Running Tests (Optional)**

To run unit or integration tests:

npm test

# or for backend

cd backend && npm test

**🐛 Troubleshooting**

**Port already in use?** Kill the process using that port or change the port in .env.

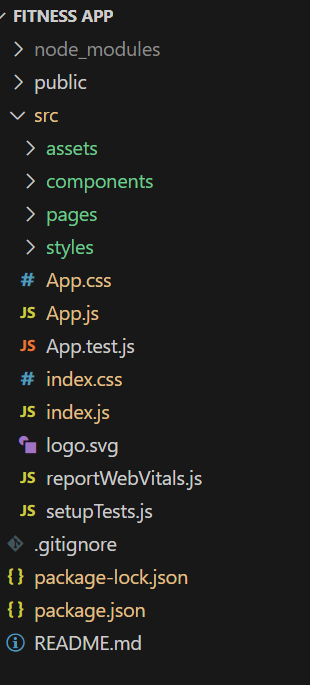
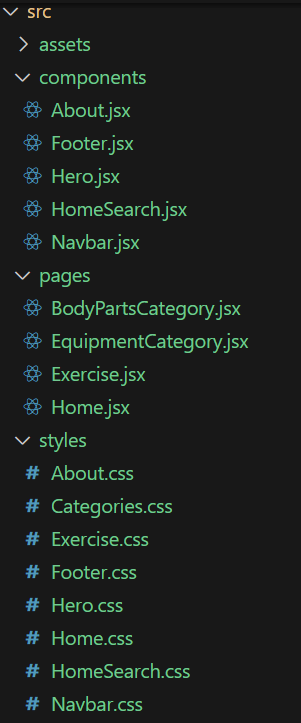
**MongoDB not connecting?** Make sure MongoDB is running locally or provide a cloud URI.

**React Native issues?** Try clearing the cache: expo start -c.

**📬 Contact**

**For any setup issues, please contact**:  
📧 support@fitflexapp.com  
💬 Or open an issue on GitHub

**5.Folder 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.

1. **Running the Application:**

Start the development server:

npm start

This command launches the development server, and you can access your React app at [http://localhost:3000](https://naanmudhalvan.smartinternz.com/Student/guided_project_workspace/about:blank) in your web browser.

1. **Component Documentation**

* User Interface (UI)

Provides a simple and responsive design.

Includes login, dashboard, workout, diet, and progress screens.

* Authentication & User Management

Secure user login and registration.

Stores personal details like age, weight, goals.

* Workout Planner

Personalized workout routines.

Daily/weekly schedules with exercise guides.

* Nutrition & Diet

Custom diet plans and calorie tracking.

Logs meals and water intake.

* Progress Tracker

Tracks weight, BMI, workouts, and diet history.

Provides reports and charts for user progress.

* Notifications & Reminders

Sends workout, diet, and hydration reminders.

Motivational alerts for consistency.

* Database

Stores user data, workouts, diet, and progress securely.

Acts as the backbone for all components.

Analytic s

* Analytics & Recommendations

Analyzes progress and suggests improvements.

Provides AI-driven workout and diet adjustmens.

**8.State Management**

* Global State

FITFLEX uses global state management to handle data that needs to be accessed across multiple components.

User Authentication & Profile: Stores login status, user details, and fitness preferences.

Workout Plans & Progress: Manages workout routines, progress tracking, and completion history.

Nutrition & Goals: Tracks calorie intake, meal plans, and user goals.

Theme & App Settings: Maintains app-wide preferences such as theme (dark/light mode) and notification settings.

The state flows from a central store (e.g., Context API/Redux/Zustand) to components through context providers and selectors. Components can both read and update the global state, ensuring consistency across the application.

**Example flow:**

User updates daily calorie intake → Global state updates → Nutrition Dashboard, Reports, and Profile automatically reflect changes.

* **Local State**

Local state is used for data that is specific to a single component and does not affect the entire application.

Form Inputs: Handling login/signup form values or workout plan creation fields.

UI Controls: Managing toggles, dropdowns, modals, and step-based navigation.

Temporary Data: Storing values like search queries, current tab selection, or in-progress workout timers.

Local state ensures components remain self-contained and performant, without unnecessary re-renders across the app.

**Example flow:**

User opens workout filter modal → Local state manages selected filter options → On apply, relevant workouts are displayed → Only global state updates if filters need to persist across screens.

**9.User Interface**

1. On boarding / Welcome Screens

* Screen 1: App logo + motivational tagline (“Your Fitness Journey
* Screen 2: Select goals (e.g., weight loss, muscle gain, stamina, wellness).
* Screen 3: User details (age, gender, weight, height).

Call to Action: “Get Started” button.

2. Home Dashboard

* Top Bar: Greeting (“Hi, Alex 👋”), profile picture.
* Main Widgets:

Daily progress ring (calories, steps, workout minutes).

Quick action buttons:

Start Workout

Log Nutrition

Track Sleep

* Daily Tips Section: Motivational quotes / health advice.
* Bottom Navigation Bar:

Home | Workouts | Progress | Community | Profile

3. Workout Screen

* Categories Tabs: Strength | Cardio | Yoga | HIIT
* Workout Cards: Each with image, duration, calories burn.
* Start Workout Button: Big, floating CTA.
* Progress Tracking: Live workout timer, reps counter.

1. Progress / Analytic

* Charts & Graphs:
* Weekly workout time
* Calories burned
* Weight tracking
* Badges/Achievements: Gamification (“7-Day Streak”, “1000 Calories Burned”).

5. Community / Social

* Feed: Users sharing progress, photos, achievements.
* Challenges: Join fitness challenges (e.g., “30-day plank challenge”).
* Friends: Add friends, see their progress.

6. Profile

* Personal info
* Goals & preferences
* Connected devices (wearables)
* Settings (notifications, reminders, dark mode)

🖌️ Design Style

* Colors: Energetic (blue, green, orange gradients).
* Typography: Bold headings, clean sans-serif (Poppins, Inter, Roboto).
* UI Elements: Rounded cards, shadows, minimal icons.
* Animations: Smooth transitions, progress ring animation.

**10.Styling**

* **Styling**

CSS Frameworks/Libraries

FITFLEX uses a combination of modern styling tools to ensure a clean, responsive, and user-friendly interface:

Tailwind CSS: Provides utility-first classes for rapid UI development with consistent spacing, typography, and responsive design.

Styled-Components (Optional for custom cases): Enables writing CSS-in- JS for dynamic styling within React components.

CSS Modules: Used in specific areas to maintain modular, scoped styling and prevent class name conflicts.

This approach allows flexibility while maintaining scalability and readability of styles.

* **Theming**

FITFLEX implements a custom theming system to enhance user experience and maintain brand consistency:

Light & Dark Mode: Global theme switcher ensures accessibility and personal preference support.

Custom Color Palette: Fitness-inspired colors (energetic tones like blue, green, and orange) are defined in Tailwind configuration for a consistent look and feel.

Typography & Components: Unified font styles and reusable design tokens (buttons, cards, inputs) ensure cohesive branding across the application.

**11.Testing**

**Testing Strategy**

The FITFLEX application follows a structured testing approach to ensure reliability, maintainability, and performance across all features.

1. Unit Testing
2. Freamwork: Jest
3. Purpose: Validate individual React components, utility functions, and hooks to ensure they behave as expected.

Example: Testing input validation logic in login/sign-up forms, verifying state updates in workout tracking components.

1. Integration Testing
2. Library: React Testing Library (RTL)
3. Purpose: Ensure multiple components work together correctly.

Example: Testing interactions between the workout plan generator and the progress tracker, verifying data flow between context/state management and UI.

3. End-to-End (E2E) Testing

1. Tool: Cypress (or Playwright, depending on project preference)
2. Purpose: Simulate real user scenarios from start to finish.

Example: User registration, logging in, creating a fitness plan, and tracking workout sessions.

4. Manual Testing

Conducted during development for UI/UX validation, responsiveness checks, and cross-browser/device compatibility.

**Code Coverage**

Tool Used: Jest with coverage reports (--coverage flag).

Coverage Goals:

Functions: 80%+

Statements: 80%+

Branches: 70%+

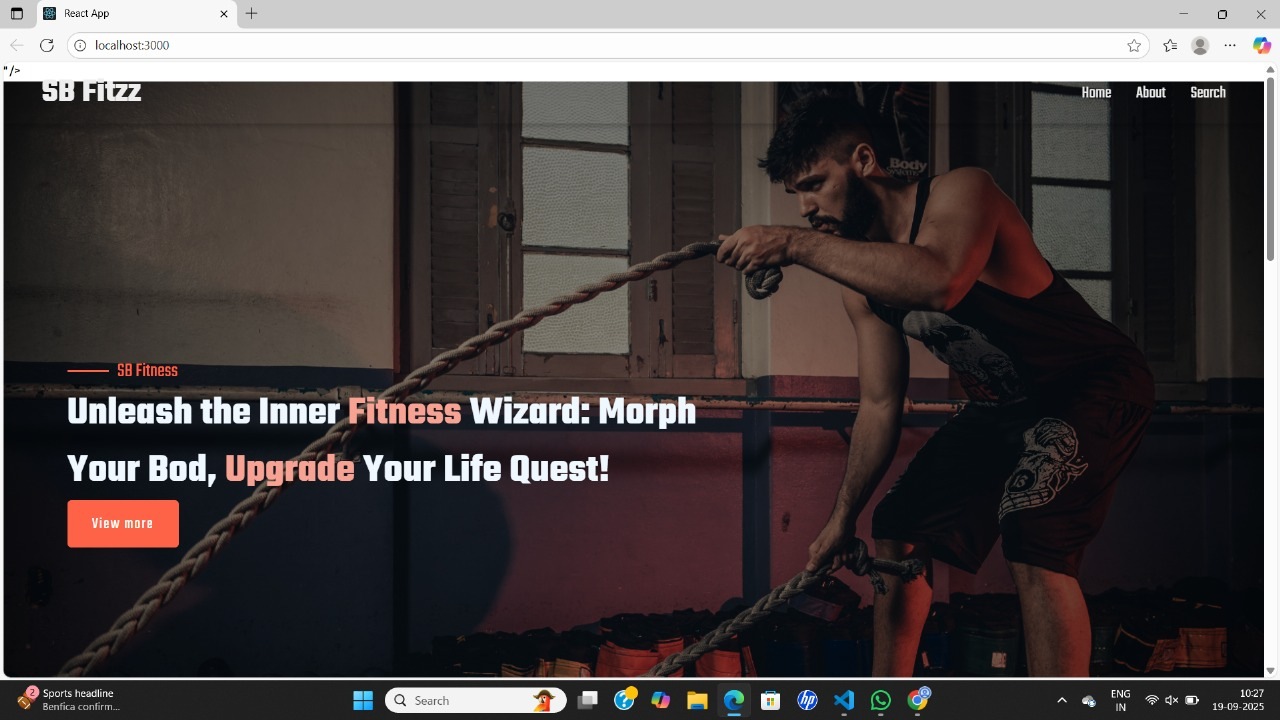
Lines: 80%+

Coverage reports are generated automatically and reviewed before merging pull requests.

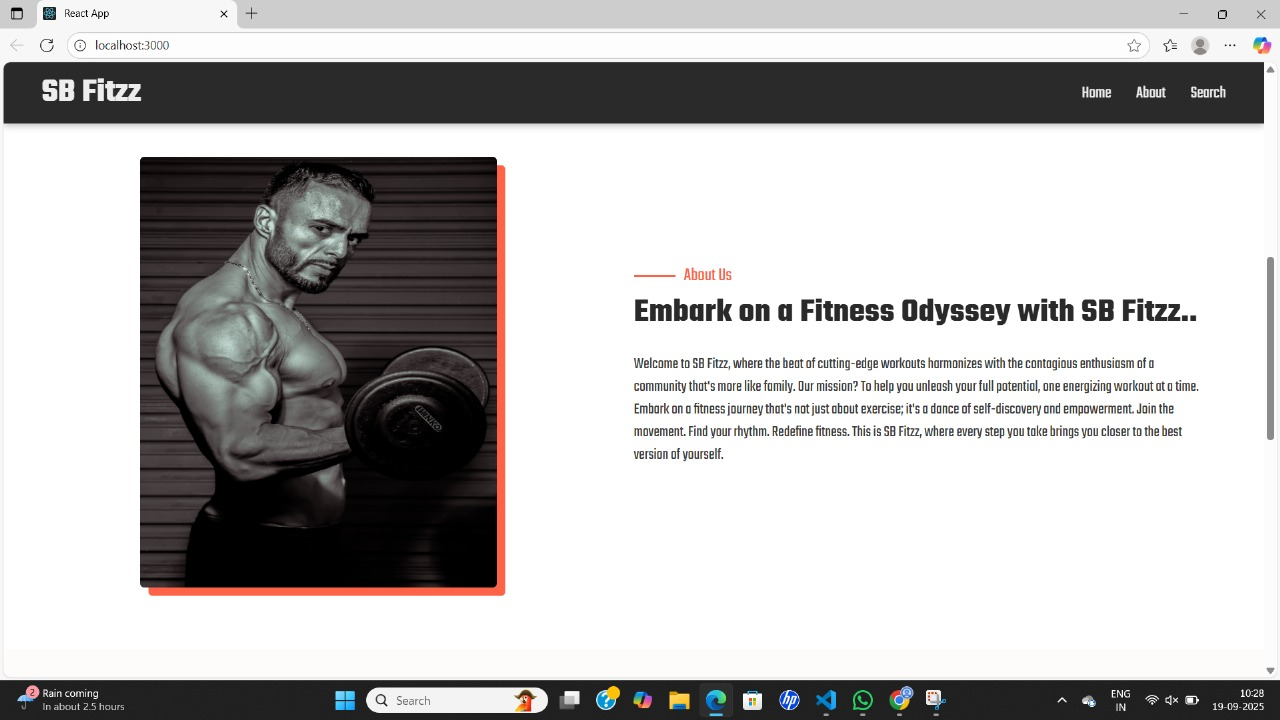
Additional monitoring through CI/CD pipelines ensures that coverage thresholds are consistently met.

**12.Screenshots or Demo**

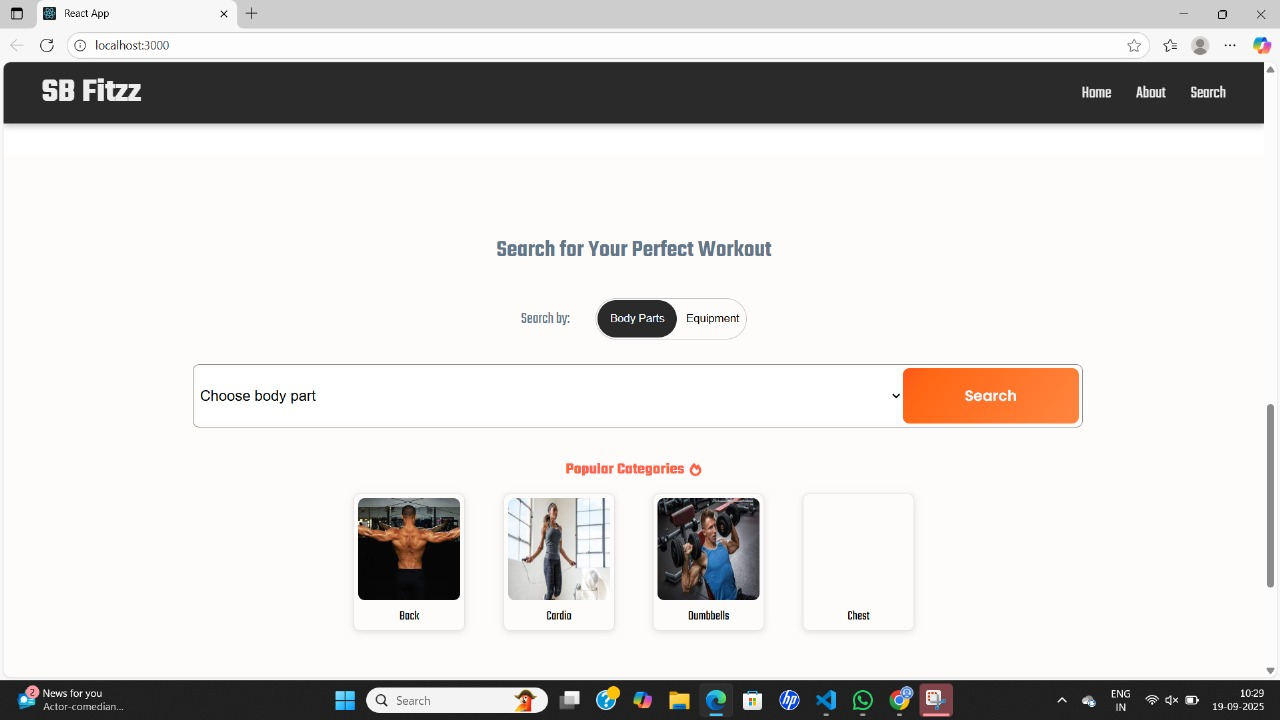
HOME PAGE:



ABOUT:



SEARCH:



**13. Known Issues**

While FITFLEX is fully functional, the following issues are known and are being monitored for future improvements:

1. Cross-Browser Compatibility

Minor UI inconsistencies may appear in older versions of Internet Explorer and Safari.

Some animations may not render smoothly on low-performance devices.

2. Performance with Large Data Sets

When handling a large number of workout logs or nutrition entries, the app may experience slow rendering in certain views.

3. Offline Mode Limitations

Limited offline support: workout plans and history may not fully sync if the user loses connectivity during data updates.

4. Push Notifications

Notifications may be delayed or inconsistent on some Android devices due to background restrictions.

5. Accessibility Gaps

Some components may lack complete screen reader support.

Color contrast in dark mode could be improved for better readability.

6. Testing Coverage

While most core features are covered, some edge cases in integration and end-to-end testing still require expansion.

**14.Future Enhancement**

FITFLEX plans to expand with the following improvements:

1. AI-Powered Workout & Diet Plans – Smarter recommendations based on user progress.
2. Gamification Features – Badges, streaks, and leaderboards to boost motivation.
3. Enhanced UI & Animations – Smooth transitions, customizable themes, and modern styling.
4. New Components-Water tracker,meal planner,and sleep monitoring dashboard.
5. Wearable Integration – Sync with smartwatches for real-time fitness data.
6. Offline Support – Better offline mode with auto-sync on reconnection.
7. Community & Social Features – Group challenges, friend tracking, and social sharing.
8. Accessibility & Multilingual Support – Improved screen reader support and regional languages.
9. Advanced Analytics Dashboard – Detailed insights into workouts, nutrition, and overall progress.