Introduction

1. **Project Title**: FitFlex
2. **Team Members**: List all team members and their roles (e.g., Frontend Developer, Backend Developer, UI/UX Designer, etc.).
3. **Project Overview**: Briefly describe the purpose of the project — a fitness tracking application.
4. **Target Audience**: People interested in improving their fitness levels and monitoring progress.
5. **Technologies Used**: Technologies like React, Node.js, Redux, Tailwind CSS, and others are involved.

Project Overview

Purpose:

1. **Fitness Tracking**: Helps users track their workouts and monitor progress.
2. **Goal Setting**: Allows users to set and track fitness goals.
3. **Workout Logs**: Enables users to log exercises with sets, reps, and time.
4. **Analytics**: Provides insights into workout performance and progress.
5. **Community Engagement**: Social features to connect with other fitness enthusiasts.

Features:

1. **User Authentication**: Users can sign up and log in securely.
2. **Workout Tracking**: Users can log different types of exercises.
3. **Dashboard Analytics**: Visualizes progress and workout data.
4. **Community Feed**: Allows interaction with other users.
5. **API Integration**: Integrates fitness data via third-party APIs.

Architecture

Component Structure:

1. **Authentication Components**: Handles login, registration, and session management.
2. **Dashboard**: Displays user progress, stats, and workout history.
3. **Workout Tracker**: Log workouts including exercises, sets, reps, and duration.
4. **Community Feed**: Users can post and interact with others.
5. **Profile**: Displays user information, preferences, and settings.

State Management:

1. **Approach Used**: React Context API/Redux for centralized state management.
2. **State Flow**: Data fetched from APIs updates global state, accessed by multiple components.
3. **Component Communication**: Different components consume and update global state as required.
4. **Performance**: State management is optimized for efficient data flow.
5. **Persistence**: Data persists between sessions using local storage or session storage.

Routing:

1. **Library Used**: React Router for navigating between pages.
2. **Routes Defined**:
    - `/login`: For user authentication.
    - `/dashboard`: For user progress and workout history.
    - `/workout-log`: To log new workouts.
    - `/community`: For user interaction and posts.
    - `/profile`: For user settings and profile management.
3. **Dynamic Routing**: Routes adapt based on user authentication state.
4. **Protected Routes**: Certain pages require users to be logged in.
5. **Route Parameters**: Some pages (e.g., workout details) might use dynamic parameters.

Setup Instructions

Prerequisites:

1. **Node.js**: Ensure that Node.js (latest LTS version) is installed.

2. **npm/yarn**: Either npm or yarn package manager should be available.
3. **Git**: Git should be installed for cloning the repository.
4. **Text Editor**: Recommended editors: VS Code or Sublime Text.
5. **Environment Setup**: Set up environment variables (e.g., API keys) if necessary.

Installation:

1. Clone the project repository:
   `git clone https://github.com/Balaji300405/Fitflex.git`
2. Navigate to the project directory:
   `cd Fitflex`
3. Install project dependencies:
   `npm install`
4. Configure any required environment variables (e.g., API URLs).
5. Run the development server:
   `npm run dev`

Folder Structure

Client:

1. `/src/components/`: Contains React components used throughout the app.
2. `/src/pages/`: Pages that correspond to different routes.
3. `/src/services/`: API calls and data-fetching logic.
4. `/src/utils/`: Helper functions, utility functions, and constants.
5. `/public/`: Static assets like images, favicon, and the HTML template.

Configuration:

1. `package.json`: Manages dependencies and project metadata.
2. `vite.config.js`: Configuration file for Vite (bundler).
3. `README.md`: Project documentation file with setup instructions.

Utilities

Helper Functions:

1. Handles API calls for fitness data.
2. Formats data before sending it to the components.
3. Reusable logic for common tasks (e.g., authentication).
4. Centralized location for state management helpers.
5. Utility functions for local storage management.

Custom Hooks:

1. `useAuth`: Manages authentication state (login/logout).
2. `useWorkoutLog`: Fetches, adds, and edits workout logs.
3. `useProgress`: Tracks user fitness progress over time.
4. `useCommunity`: Handles community posts and interactions.
5. `useAnalytics`: Gathers and displays workout statistics.

Running the Application

1. **Run the frontend server**:
   - Start with `npm run dev`.
2. **Development Mode**: The app runs on `localhost` and automatically reloads on file changes.
3. **Error Handling**: Errors are logged in the terminal for debugging.
4. **Build for Production**: Run `npm run build` for production deployment.
5. **Deployment**: Deploy on services like Vercel, Netlify, or any hosting platform of choice.

Component Documentation

Key Components:

1. **WorkoutTracker**: Component to log workouts (sets, reps, duration).
2. **Dashboard**: Shows user progress and workout history.
3. **Community**: Allows interaction, commenting, and post creation.
4. **Profile**: User settings and profile data management.
5. **Navigation**: Header and footer components for navigation.

Reusable Components:

1. **Button**: Action buttons across the app.

2. **Card**: Displays workouts or community posts.
3. **Input**: Reusable input fields for forms.
4. **Modal**: Reusable modal for displaying details.
5. **Loader**: A loading spinner for data fetching.

State Management

Global State:

1. **Authentication State**: Manages current user login status.
2. **Workout Data**: Stores user's workout history.
3. **Community Feed**: Stores posts, comments, and interactions.
4. **Progress Analytics**: Holds user's progress data over time.
5. **Theme State**: Tracks the selected theme (light/dark mode).

Local State:

1. **Form Handling**: Managed with `useState` for user input.
2. **Modal Visibility**: Controls showing/hiding modals.
3. **Notifications**: Stores temporary notifications or alerts.
4. **Loading States**: Manages loading spinners during API calls.
5. **UI State**: Manages active/inactive states of UI elements.

User Interface

UI Features:

1. **Responsive Design**: The app is optimized for all screen sizes.
2. **Dark Mode**: Allows users to toggle between light and dark themes.
3. **Intuitive Dashboard**: Easy-to-read workout statistics.
4. **Interactive Charts**: Visualizes progress with bar and line charts.
5. **Mobile-First**: Ensures the app is fully functional on mobile devices.

Styling

CSS Frameworks/Libraries:

1. **Tailwind CSS**: Used for fast and responsive styling.
2. **Custom Styles**: Custom CSS for specific design elements.

3. **Styled Components**: For scoped and modular component styling.
4. **CSS Variables**: For theming (light/dark mode).
5. **Responsive Design**: Tailored styles for mobile, tablet, and desktop views.

Testing

Testing Strategy:

1. **Unit Testing**: Ensures individual components function as expected.
2. **Integration Testing**: Verifies that components work together correctly.
3. **End-to-End Testing**: Simulates user interactions to ensure the entire app works.
4. **Test Coverage**: Monitored with Jest coverage tools.
5. **Continuous Integration**: Runs tests automatically on code push.

Screenshots or Demo

1. **Login Screen**: Displays user authentication form.
2. **Dashboard**: Shows user workout history and analytics.
3. **Workout Log**: UI for logging new exercises.
4. **Community Feed**: User interactions within the app.
5. **Profile Settings**: Allows users to update their settings.

Known Issues

1. **Mobile View Bugs**: Some minor layout issues on mobile devices.
2. **Slow API Response**: Delays in fetching workout data from the API.
3. **Login Timeout**: Occasional timeout issues during authentication.
4. **Form Validation**: Validation inconsistencies on certain forms.
5. **Cross-Browser Compatibility**: Some issues with older browsers.

Future Enhancements

1. **AI-Based Fitness Recommendations**: Personalize user workout suggestions.
2. **Real-Time Chat**: Implement chat functionality for user interaction.
3. **Enhanced Analytics**: Detailed workout progress with custom charts.
4. **Voice Assistance**: Integrate voice commands for hands-free tracking.
5. **Multi-Language Support**: Allow users to choose their preferred language.