Frontend development with react.js

Fitness app

1.Introduction

- **Project title: Fitflex**
- TeamI'd:SWTID1741165341150867
- Team Members:
- U.Nandhini
- M.Deepika
- S.Mega
- A. Joy stella

2.Project Overview

Purpose

- FitFlex is a React-based fitness companion application that helps users track workouts, set goals, and monitor progress through an intuitive interface.

Features

- User authentication and profile management
- Customizable workout plans
- Progress tracking with interactive graphs
- Integration with fitness APIs (e.g., step counters, heart rate monitors)
- Dark mode and theme customization.

3.Architecture

Component Structure

- App.js – Main application wrapper
- Workout/ – Manages workout creation and tracking
- Settings/ – Allows customization of user preferences

State Management

Using Redux Toolkit for global state management, handling user authentication, workout data, and theme settings.

Routing

Implemented with React Router for seamless navigation:

- / – Landing Page
- /workouts – Workout plan customization
- /settings – Personalization options

4.Setup Instructions

Prerequisites

- Node.js (>= 16.x)
- npm or yarn
- React (>= 18.x)

Installation

1. Install dependencies:

npm install

2. Configure environment variables in .env file

3. Start the development server:

npm start

5.Folder Structure

>Public
__src
 >assets
>Components
>Page
>Styles
#App.css
JS App.js
JS App.test.js
#index.css
JS index.js
__logo.svg
JS reportWebVitals.js
JS setupTest.js
__ . gitignore

{} package.lockjson
{} package.json
__README.md

6.Running the application

To run the app locally:

npm start

7.Component Documentation

Key Components

Dashboard – Displays user stats

WorkoutTracker – Allows users to log workouts

ProfileSettings – Handles account and theme settings

Reusable Components

Button – Customizable buttons

Modal – Pop-up UI component

InputField – Standardized input elements

8.State Management

Global State

Redux Toolkit used for managing authentication and workout data.

Actions and reducers handle API calls and state updates.

Local State

Component-level states handled with useState() and useEffect().

9.User Interface

Screenshots

(Include images showcasing the app’s UI, dashboard, workout tracking page, etc.)

10.Styling

CSS Frameworks/Libraries

Styled Components for theme-based styling

Material-UI for UI components

Theming

Dark mode/light mode toggle

Customizable UI themes based on user preference

11.Testing

Testing Strategy

Jest & React Testing Library for unit and integration tests

Cypress for end-to-end testing

Code Coverage

Coverage reports generated using Jest's built-in coverage tool.

12.Screenshots or Demo

(Include a hosted link or GIF demonstrating app features)

13.Known Issues

API call latency in fetching real-time workout stats

Some UI elements may not be fully responsive on smaller devices

14.Future Enhancements

AI-based workout recommendations

Integration with Apple Health and Google Fit

Community features like workout sharing