Frontend Development with React.js

Project Documentation format

**1.Introduction**

**Project Title:FitFlex \_Your Personal Fitness Companion**

**Team Members:1) S.SATHISH (Project Manager)**

**2)S.RAMASAMY (Frontend Developer)**

**3)D.JEEVANKUMAR (Backend Developer)**

**4)R.PRAVEENKUMAR (UI Desingner)**

**5)D.NITHESH (Project Video Creater)**

**2.Project Overview**

Purpose: Fitness And Workouts

Features: User Authentication – Secure login and signup system.

Personalized Dashboard – Displays workouts, daily goals, and stats.

Exercise Library – Collection of exercises with images, videos, and instructions.

Progress Tracking – Charts and graphs to monitor fitness progress.

Goal Setting & Reminders – Helps users set fitness goals and get timely reminders.

Responsive Design – Works seamlessly on both mobile and desktop.

Profile Management – Users can update their details and preferences.

**3.Architecture**

Component Structure: Component Structure

Navbar → Navigation bar with links (Dashboard, Workouts, Progress, Profile).

Footer → App footer with copyright and links.

Auth Components → Login, Signup, Forgot Password pages.

Dashboard → Main page showing workout summary, goals, and progress highlights.

WorkoutPlans → Displays workout routines (daily/weekly plans).

ProgressTracker → Shows user progress using charts and graphs.

ExerciseLibrary → Searchable list of exercises with images and descriptions.

Profile → User details, preferences, and settings.

Reusable Components →

Button

InputField

Card

Modal

State Management: 1. Global State (handled with Redux Toolkit):

User Data: Login status, profile information.

Workouts: User’s workout plans, completed exercises.

Progress: Progress stats and chart data.

Goals: Fitness goals and reminders.

2. Local State (handled with React useState / useReducer):

Form inputs (e.g., login form, add workout form).

UI states (modal open/close, dropdowns, toggles).

Temporary values like search filters in Exercise Library

Global State → Redux Toolkit for data shared across the app.

Local State → useState/useReducer for small, component-specific valu

Routing: / → Home / Landing Page

/login → Login Page

/signup → Signup / Registration Page

/dashboard → User Dashboard (summary of workouts & goals)

/workouts → Workout Plans Page

/progress → Progress Tracker (charts & stats)

/exercises → Exercise Library (search & filter exercises)

/profile → User Profile & Settings

**4.Setup Instructions**

Prerequisites: 1. Node.js – Version 16 or above (for running React app).

2. npm (Node Package Manager) or yarn – To install dependencies.

3. Git – For cloning the repository and version control.

4. Code Editor – (Recommended: Visual Studio Code).

5. Web Browser – (Recommended: Google Chrome / Edge) to test the app.

Installation: git clone

<https://github.com/s9440786-arch/Fitflex-_your-personal-fitness-companion.git>

cd fitflex

npm install

npm start

**5.Folder Structure**

Client: src/

│

├── components/ # Reusable UI components

│ ├── Navbar/

│ ├── Footer/

│ ├── WorkoutCard/

│ ├── ProgressChart/

│ └── Common/ # Buttons, Inputs, Modals

│

├── pages/ # Main pages of the app

│ ├── Dashboard/

│ ├── Workouts/

│ ├── Progress/

│ ├── Exercises/

│ └── Auth/ # Login, Signup, Forgot Password

│

├── redux/ # State management

│ ├── slices/ # Redux Toolkit slices (user, workouts, progress)

│ └── store.js

│

├── assets/ # Static files

│ ├── images/

│ └── icons/

│

├── utils/ # Helper functions & custom hooks

│

├── App.js # Root component

├── index.js # Entry point

└── routes.js # Application routes

Utilities: 1) Common Utilities

dateFormatter.js → Formats dates for workouts and progress logs.

apiHelper.js → Handles API calls (GET, POST, PUT, DELETE).

storageHelper.js → Saves and retrieves data from localStorage (e.g., JWT tokens, user preferences).

validation.js → Form validations (email format, password strength, required fields).

2)Custom Hooks

useAuth.js → Custom hook for handling authentication (login, logout, session check).

useFetch.js → Custom hook for fetching data from APIs with loading/error states.

useTheme.js → Handles Light/Dark mode switching.

**6.Running the Application**.

Frontend: npm start

**7.Component Documentation**

Key Components: 1. Navbar

Displays navigation links (Dashboard, Workouts, Progress, Profile).

Visible on all pages after login.

2. Footer

Contains copyright info and quick links.

3. Auth Components

Login → User authentication.

Signup → New user registration.

Forgot Password → Password recovery option.

4. Dashboard

Shows workout summary, goals, and quick stats.

Acts as the main home page for logged-in users.

5. WorkoutCard

Displays workout details (exercise name, sets, reps, duration).

Reusable Components: 1. Button

Common styled button (primary, secondary, outline).

Used in forms, modals, and navigation actions.

2. InputField

Reusable text input with label and validation support.

Used in login, signup, profile forms, and search bar.

3. Card

Container for displaying information neatly (workout, exercise, progress stats).

Helps maintain a uniform layout.

4. Modal

Popup component used for adding/editing workouts, reminders, or confirming actions.

5. Loader / Spinner

Displays loading animation while fetching data from API.

6. Alert / Toast

For showing success, error, or warning messages.

**8.State Management**

Global State: Main Global States

1. User Slice

Stores login status, user profile, authentication token.

Example: { isLoggedIn: true, name: "Sathish", token: "abc123" }

2. Workout Slice

Stores user’s workout plans and completed exercises.

Example: { workouts: [ { name: "Push-ups", sets: 3, reps: 15 } ] }

3. Progress Slice

Stores progress tracking data for charts (weight, calories burned, workout stats).

Example: { progress: [ { date: "2025-09-01", calories: 500 } ]

4. Goals Slice

Stores fitness goals and reminders.

Example: { goals: [ { type: "Weight Loss", target: "5kg in 2 months" } ] }

Local State: Examples of Local State in FitFlex

1. Form Inputs

Login / Signup form fields (email, password).

Profile update inputs.

Example:

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

2. UI Controls

Modal open/close state.

Dropdown selections.

Toggle for Light/Dark mode.

Example:

const [isModalOpen, setIsModalOpen] = useState(false);

3. Temporary Filters & Search

Exercise Library search bar input.

Workout filter (e.g., “Cardio”, “Strength”).

Example:

const [searchTerm, setSearchTerm] = useState("");

4. Small Component States

Active tab in Dashboard.

Checkbox selections for workout preferences.

**9.User Interface**

Dashboard → Shows workout summary, daily goals, and quick stats.

Progress Tracker → Graphs and charts (Recharts) to display fitness progress.

Exercise Library → Searchable and filterable list of exercises with images and details.

Profile Page → User details, preferences, and goal settings.

Responsive Design → Works smoothly on mobile, tablet, and desktop.

Light/Dark Mode → Better user experience with theme switching.

**10.Styling**

CSS Frameworks/Libraries: 1. Tailwind CSS

Utility-first CSS framework.

Used for responsive layouts, spacing, colors, and typography.

Example: className="p-4 bg-green-500 text-white rounded-xl"

2. Styled Components: CSS-in-JS library for component-level styling.

Used for custom UI parts (buttons, modals, cards).

Example:

const Button = styled.button`

background: #4CAF50;

color: white;

padding: 10px 20px;

border-radius: 8px;

`;

3. Recharts (for UI charts)

Used to display progress tracking in charts and graphs.

Theming: FitFlex uses a custom theming system to improve user experience and branding:

1. Light Mode & Dark Mode

Toggle option for better accessibility.

Light mode → clean and minimal.

Dark mode → eye-friendly for night usage.

2. Color Palette (Fitness Branding)

Primary Colors: Green (energy, health), Blue (trust, calmness).

Secondary Colors: Black & White (contrast, readability).

Accent Colors: Orange/Yellow for highlights (progress, alerts).

3. Typography

Modern sans-serif font for readability.

Font weights (bold for headings, medium for subheadings, regular for body text).

4. Consistency

Buttons, cards, inputs, and modals follow the same theme.

Colors and shadows kept uniform across components.

**11.Testing**

Testing Strategy:

1. Unit Testing

Tests for individual components (e.g., Button, InputField, WorkoutCard).

Framework: Jest + React Testing Library.

2. Integration Testing

Tests how multiple components work together.

Example: Dashboard fetching data from Redux + displaying in ProgressChart.

3. End-to-End (E2E) Testing

Simulates real user interactions.

Tools: Cypress / Playwright.

Example: User login → add workout → view progress.

4. UI/UX Testing

Cross-browser compatibility (Chrome, Edge, Firefox).

Mobile responsiveness (using Chrome DevTools).

5. Performance Testing

Ensures fast load times and smooth navigation.

Tools: Lighthouse / WebPageTest.

6. Security Testing

Checks for authentication, authorization, and data protection.

Code Coverage: Main Coverage Areas:

1. Components Coverage

Navbar, Footer, WorkoutCard, ProgressChart, ExerciseList.

Tests ensure correct rendering and props handling.

2. Redux State Coverage

User Slice, Workout Slice, Progress Slice, Goals Slice.

Tests verify reducers and actions.

3. Utilities Coverage

dateFormatter, apiHelper, validation, storageHelper.

Tests ensure correct outputs for given inputs.

4. Routing Coverage

Routes (/login, /dashboard, /progress, etc.) tested for proper navigation.

5. Authentication Coverage

Login, Signup, Logout flows tested for correctness.

Tools Used:

Jest → Unit tests.

React Testing Library → Component rendering.

Istanbul/nyc → Generates coverage reports (line, function, branch coverage).

Expected Coverage Target:

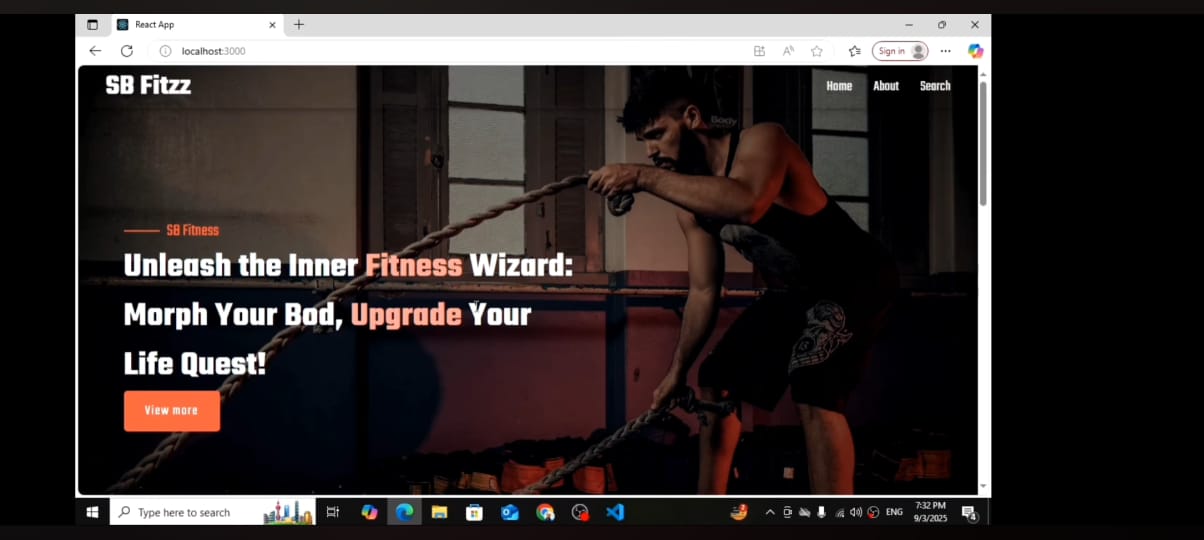
Lines: 80%+

Functions: 75%+

Branches: 70%+

Statements: 80%+

**12.Screenshot or Demo**



**13.Known Issues**

\*On slower networks, image-heavy pages may load with a delay.

\*Dark mode toggle may briefly flash default light theme during reload.

\*Some mobile devices experience layout shift in dashboard charts.

**14.Future Enhancements**

\*Add offline support with service workers.

\*Introduce animations using Framer Motion for smoother UI transitions.

\*Expand internationalization (i18n) for multilingual support.

\*Implement role-based access control for admin and user dashboards.