Frontend Development with React.js

Project Documentation format

---

1. Introduction

- **Project Title**: FitFlx
- **Team Members**:
  - Janakiraman R (Team Leader)
  - Vignesh J
  - Gowtham P
  - Mukesh K

---

2. Project Overview

- **Purpose**:
  The Fitness Application is designed to help users monitor their fitness activities, track progress, and set fitness goals. The application provides a user-friendly interface for logging workouts, tracking calories, and visualizing progress over time.
- **Features**:
  - User authentication (login/signup)
  - Dashboard for tracking daily activities
  - Workout logging and history
  - Calorie tracker
  - Progress charts and analytics
  - Responsive design for mobile and desktop

---

3. Architecture

- **Component Structure**:
  - **App Component**: The root component that manages routing and global state.
  - **Dashboard Component**: Displays user's daily fitness metrics and progress.
  - **WorkoutLog Component**: Allows users to log and view their workout history.
  - **Auth Component**: Handles user authentication (login/signup).
- **State Management**:
  - **Redux**: Used for global state management to handle user authentication, workout data, and calorie tracking.
  - **Local State**: Managed within individual components using React's useState and useEffect hooks.

- **Routing**:
    - **React Router**: Used for navigation between different pages (e.g., Dashboard, Workout Log, Calorie Tracker).

---

4. Setup Instructions

- **Prerequisites**:
    - Node.js (v16 or higher)
    - npm (v8 or higher)
    - Git (for cloning the repository)
- **Installation**:
    1. Clone the repository: `git clone https://github.com/your-repo/fitness-tracker.git`
    2. Navigate to the client directory: `cd fitness-tracker/client`
    3. Install dependencies: `npm install`
    4. Configure environment variables: Create a `.env` file in the client directory and add the necessary variables (e.g., API keys).
    5. Start the development server: `npm start`

---

5. Folder Structure

- **Client**:
    - **src/components**: Contains all React components (e.g., Dashboard, WorkoutLog, CalorieTracker).
    - **src/pages**: Contains page components that are rendered based on the route.
    - **src/assets**: Stores static assets like images, icons, and styles.
    - **src/redux**: Contains Redux store, actions, and reducers.
    - **src/utils**: Utility functions and custom hooks.
- **Utilities**:
    - **useFetch**: Custom hook for making API requests.
    - **formatDate**: Utility function for formatting dates.
    - **calculateCalories**: Helper function for calculating calorie intake and expenditure.

---

6. Running the Application

- **Frontend**:
    - Navigate to the client directory: `cd client`
    - Start the development server: `npm start`
    - The application will be available at `http://localhost:3000`

---

7. Component Documentation

- **Key Components**:
  - **Dashboard Component**: Displays user's daily fitness metrics (e.g., steps taken, calories burned). Receives props like userData and progressData.
  - **WorkoutLog Component**: Allows users to log workouts and view their history. Receives props like workouts and onLogWorkout.
  - **CalorieTracker Component**: Tracks daily calorie intake and expenditure. Receives props like calorieData and onUpdateCalories.
  - **ProgressChart Component**: Visualizes user progress using charts. Receives props like progressData and chartType.
- **Reusable Components**:
  - **Button**: A reusable button component with customizable styles and onClick handlers.
  - **InputField**: A reusable input field component for forms, with validation support.
  - **Modal**: A reusable modal component for displaying pop-ups or alerts.

---

8. State Management

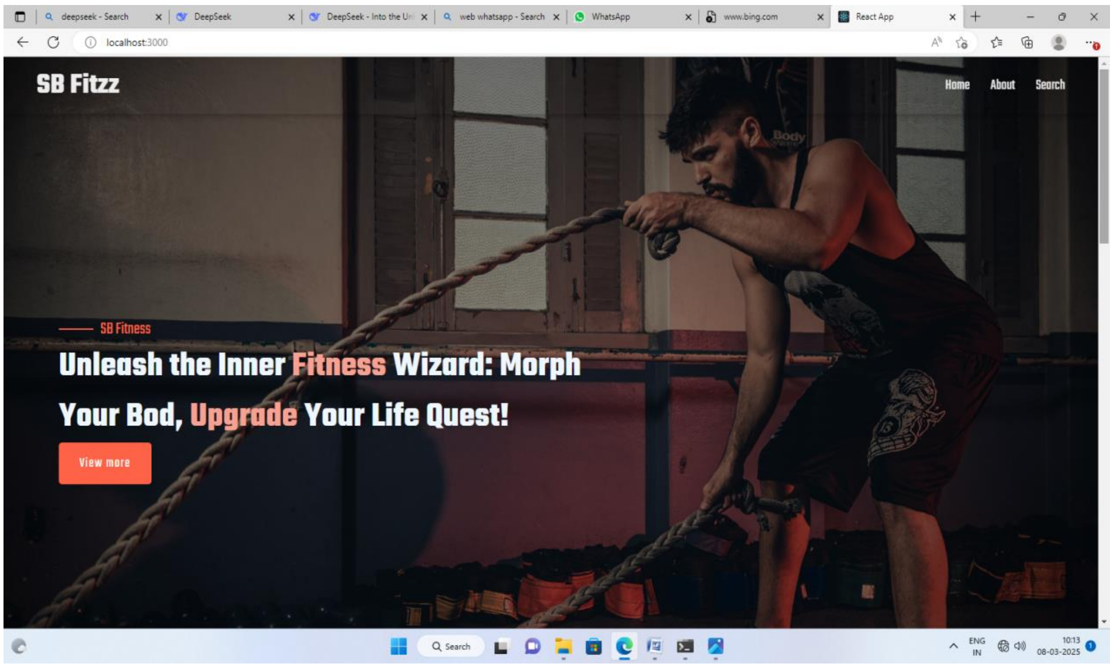
- **Global State**:
  - **Redux Store**: Manages global state for user authentication, workout data, and calorie tracking. Actions like LOGIN_USER, ADD_WORKOUT, and UPDATE_CALORIES are dispatched to update the state.
  - **State Flow**: The state flows from the Redux store to components via useSelector and is updated using useDispatch.
- **Local State**:
  - Managed within components using React's useState and useEffect hooks. For example, the WorkoutLog component uses local state to manage the form inputs for logging workouts.

---

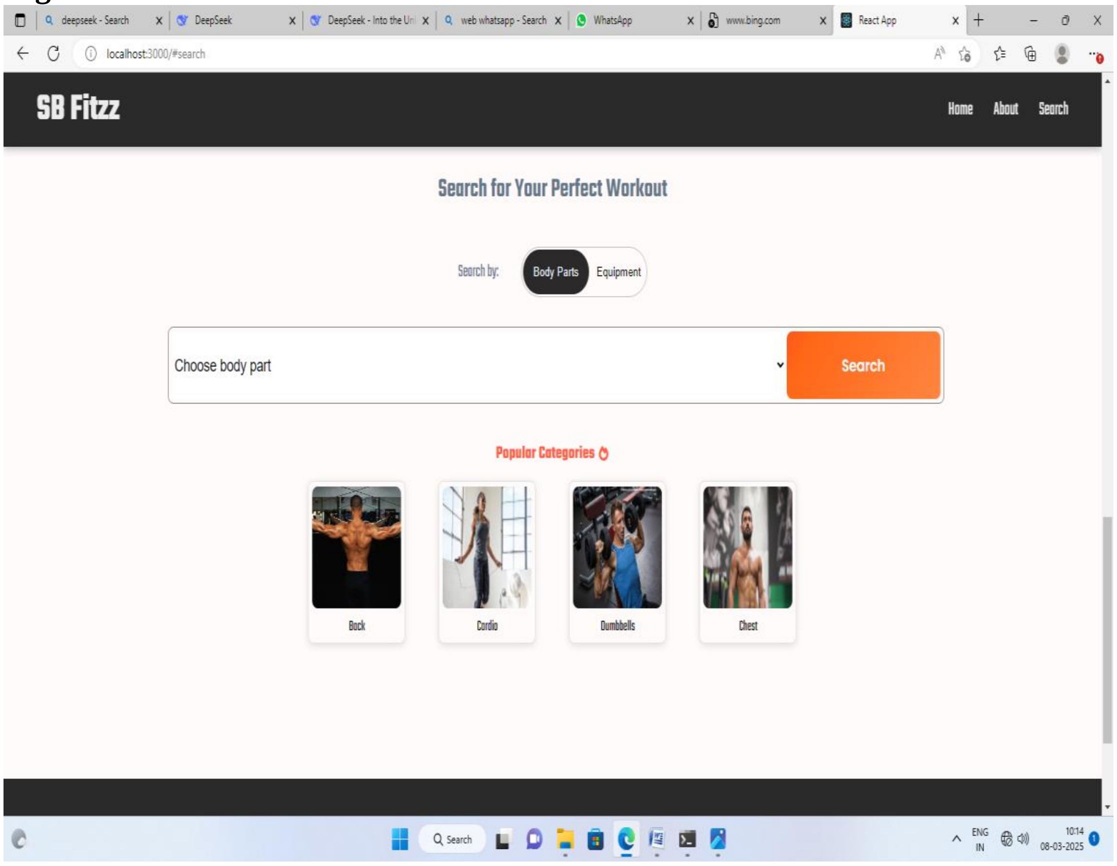
9. User Interface

- **Screenshots**:
- **Screenshots are below** :

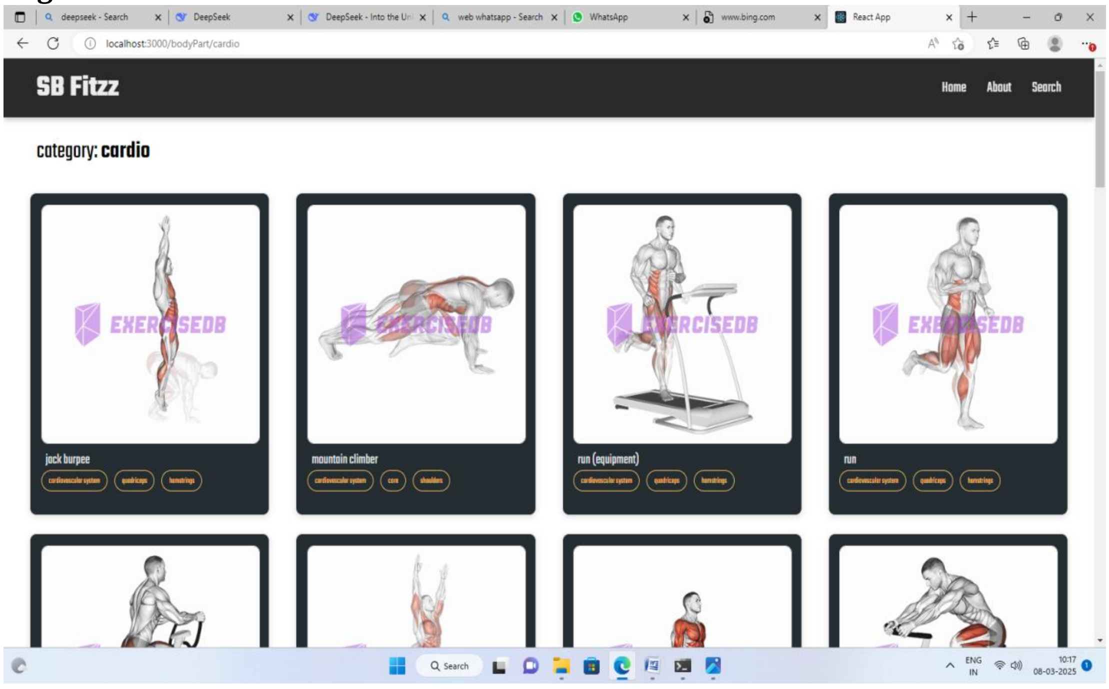
- **Dashboard**:


SB Fitzz
Home
About
Search
SB Fitness
Unleash the Inner Fitness Wizard: Morph Your Bod, Upgrade Your Life Quest!
View more

- **Search Page**:



- **Work Log**:



---

10. Styling

- **CSS Frameworks/Libraries**:
  - **Styled-Components**: Used for component-level styling.
  - **Bootstrap**: Used for responsive grid layouts and pre-built components.
- **Theming**:
  - A custom theme is implemented using Styled-Components, with support for light and dark modes.

---

11. Testing

- **Testing Strategy**:
  - **Unit Testing**: Jest and React Testing Library are used for unit testing individual components.
  - **Integration Testing**: Ensures that components work together as expected.
  - **End-to-End Testing**: Cypress is used for end-to-end testing of user flows (e.g., logging in, logging workouts).
- **Code Coverage**:
  - Code coverage is monitored using Jest's built-in coverage tool. The current coverage is 85%.

---

12. Screenshots or Demo

- **Demo Link**: React App
- **Screenshots**: See section 9 for UI screenshots.

---

13. Known Issues

- **Issue 1**: The calorie tracker sometimes fails to update in real-time when the user logs a meal.
- **Issue 2**: The progress chart may not render correctly on older browsers.
- **Issue 3**: The mobile navigation menu occasionally overlaps with content on smaller screens.

---

14. Future Enhancements

- **New Features**:
  - Integration with wearable devices (e.g., Fitbit, Apple Watch).
  - Social features to share progress with friends.
  - Gamification (e.g., badges, rewards for achieving fitness goals).
- **UI/UX Improvements**:
  - Add animations for a more engaging user experience.
  - Improve the mobile navigation menu for better usability.
- **Performance Optimization**:
  - Optimize the rendering of charts for better performance on low-end devices.
  - Implement lazy loading for components to reduce initial load time.

This documentation provides a comprehensive overview of the Fitness Tracker Application, including its architecture, setup instructions, and future enhancements.

New chat