**Frontend Development with React.js**

**Project Documentation**

**1. Introduction**

* **Project Title:** FitFlex
* **Team ID:** NM2025TMID40306
* **Team Leader:** danielraj&(rajdaniel480@gmail.com)
* **Team Members:** [**aruledvish&aruledvish@gmail.com**](mailto:aruledvish&aruledvish@gmail.com)
* [Seravin&seravinsera1@gmail.com](mailto:Seravin&seravinsera1@gmail.com)
* Surendar&surendarsudha23@gmail.com

**2. Project Overview**

* **Purpose:**  
  Kelwin’s Team is a React-based frontend application designed to provide an interactive fitness/workout experience. The app aims to deliver body-part focused workout plans, progress tracking, and community engagement features.
* **Features:**
  + Select workouts by specific body parts
  + View detailed exercise instructions with videos
  + Schedule workouts on a calendar
  + Track progress and achievements
  + Nutrition and recovery guidance
  + Social and community interaction (planned)

**3. Architecture**

* **Component Structure:**  
  The app is modularly structured into core components such as:
  + App (root component)
  + Navbar and Footer (common layout)
  + Workouts section (with BodyPartSelector, WorkoutList, WorkoutDetail)
  + Planner (calendar and workout scheduler)
  + ProgressTracker (user progress overview)
  + Additional pages: Nutrition, Recovery, Community
* **State Management:**  
  Uses **React Context API** for managing global state including:
  + Current user session and preferences
  + Selected body part filter
  + Workouts data
  + Scheduled workouts and progress
* **Routing:**  
  Implements **React Router v6** for SPA navigation with routes such as:
  + / → Home page
  + /workouts and /workouts/:bodyPart → Workouts by body part
  + /workout/:id → Specific workout details
  + /planner → Workout calendar
  + /progress → Progress tracking
  + /nutrition, /recovery, /community → Related pages

**4. Setup Instructions**

* **Prerequisites:**
  + Node.js (version 18 or higher recommended)
  + npm (comes with Node.js)
  + Git (optional but recommended)
* **Installation:**
  + Clone the repository:
  + git clone https://github.com/daniel-hast/dls-project
  + cd dls-project
  + Install dependencies:
  + npm install
  + Start the development server:
  + npm start
  + (Optional) Build for production:
  + npm run build

**5. Folder Structure**

* **Client (React App):**
* Kelwin-s-Team/
* ├── public/ # Static assets and HTML template
* ├── src/
* │ ├── components/ # Reusable UI components (buttons, cards, selectors)
* │ ├── pages/ # Route-level components (Home, Workouts, Planner)
* │ ├── assets/ # Images, icons, stylesheets
* │ ├── context/ # React Context providers and hooks
* │ ├── utils/ # Helper functions, custom hooks
* │ ├── App.js # Root app component
* │ └── index.js # Entry point
* ├── package.json
* └── README.md

**6. Running the Application**

* To run the frontend locally, execute:
* npm start
* This runs the app in development mode at http://localhost:3000.
* The app supports hot-reloading on code changes.

**7. Component Documentation**

* **Key Components:**
  + BodyPartSelector: Allows users to select body parts for workouts.
  + WorkoutList: Displays a list of workouts filtered by the selected body part.
  + WorkoutDetail: Shows detailed instructions and videos for a specific workout.
  + CalendarView: Displays the user’s workout schedule.
  + ProgressTracker: Visualizes user progress and history.
* **Reusable Components:**
  + Button, Card, Modal, Loader — common UI elements configurable through props.

**8. State Management**

* **Global State:**  
  Managed using React Context to handle:
  + User authentication and preferences
  + Current workout filters
  + Scheduled workouts
  + Progress data
* **Local State:**  
  Individual components manage local UI states like form inputs, modal visibility, and animations using React useState or useReducer.

**9. User Interface**

(Add screenshots or GIFs here showcasing key pages, like the workout selection screen, calendar planner, and progress tracker.)

**10. Styling**

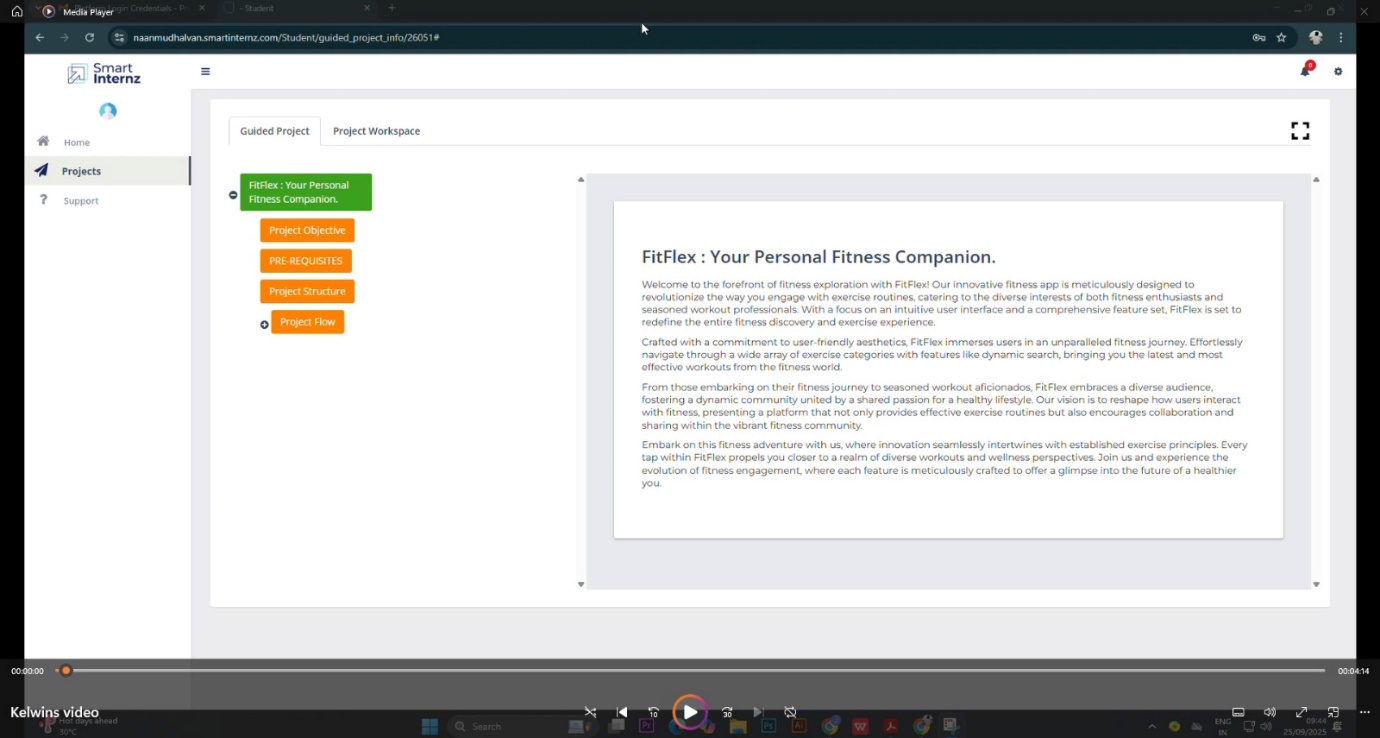
* Uses standard **CSS** with modular component styles.
* No external CSS frameworks currently, but the design allows easy integration with libraries like **Styled-Components** or **Sass**.
* Theming support can be added in future releases.

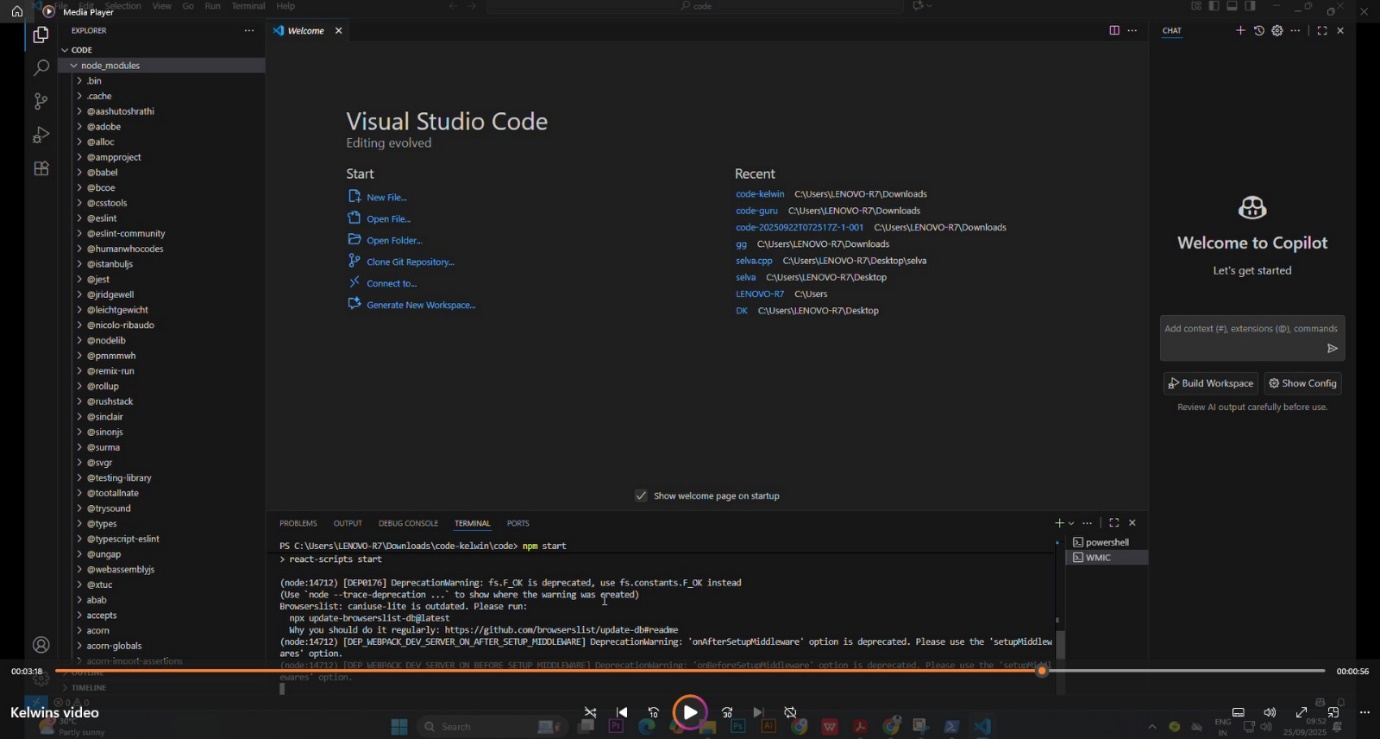
**11. Testing**

* No automated tests included yet.
* Future testing plans:
  + Unit and integration tests using **Jest** and **React Testing Library**.
  + End-to-end tests using **Cypress**.

**12. Screenshots or Demo**

**Video**

****

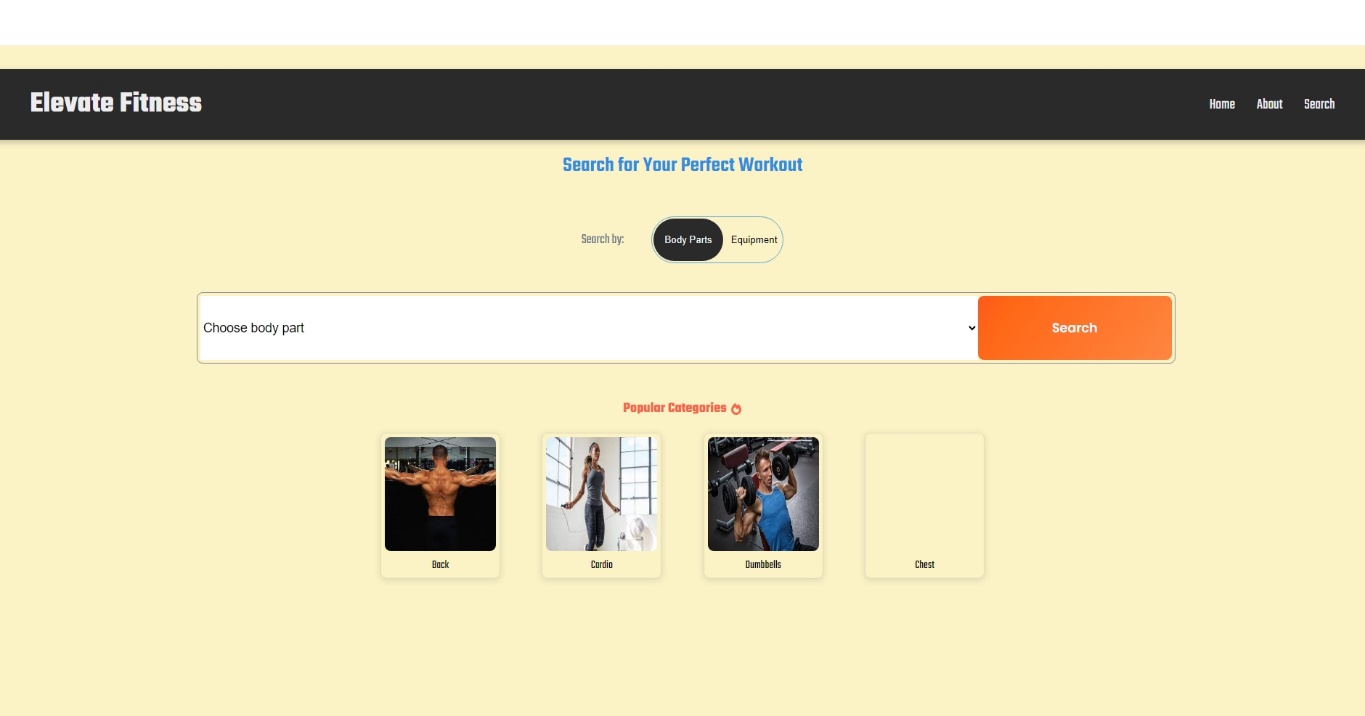
****

**Website**

**13. Known Issues**

* Some UI responsiveness issues on smaller screens.
* Lack of user authentication and backend integration currently.
* No offline support or caching.





**14. Future Enhancements**

* Add user authentication and profile management.
* Backend API integration for persistent data storage.
* Improved UI/UX with animations and dark mode support.
* Expand community features with real-time chat or forums.
* Add automated testing for reliability.
* Mobile app version or PWA support.