VALLIAMMAL COLLGE FOR WOMEN (1363)

DEPARTMENT OF COMPUTER SCIENCE WITH DATA SCIENCE

FITFLEX (OUT PERSONAL FITNESS)

(TEAM ID : 155764)

1. ABITHA V (TEAM LEADER)

2. ARCHANA V

3. NANDHINI M

4. NANDHITHA MV

5. PRIYADHARSHINI D

* **GITHUB LINK (CODING AND DOCUMENTATION)**

  https://github.com/abithav1/Fitflex_Abitha.git

* **GOOGLE DRIVE LINK (DEMO VIDEO LINK)**

1.INTRODUCTION

PROJECT TITLE : FITFLEX

TEAM ID : 155764

TEAM MEMBERS :

1. ABITHA V (TEAM LEADER)

2. ARCHANA V

3. NANDHINI M

4. NANDHITHA MV

5. PRIYADHARSHINI D

FitFlex is a revolutionary fitness app designed to transform your workout experience. It offers an intuitive interface, dynamic search, and a vast library of exercises for all fitness levels. Join FitFlex to embark on a personalized fitness journey and achieve our wellness goals.

Features of FitFlex:

- Exercises from Fitness API: Access a diverse array of exercises from reputable fitness APIs, covering a broad spectrum of workout categories and catering to various fitness goals.
- Visual Exercise Exploration: Engage with workout routines through curated image galleries, allowing users to explore different exercise categories and discover new fitness challenges visually.

2. PROJECT OVERVIEW

PURPOSE: The project is a React-based frontend application designed to provide a modern, responsive user interface with efficient routing and state management. It integrates essential libraries for testing and HTTP requests and ensures compatibility with multiple browsers.

FEATURES: Key features of the frontend include responsive UI components, dynamic page routing using react-router-dom, integration with external APIs using axios, and a testing suite with @testing-library/react and jest. The UI is designed to be modular and reusable.

3.ARCHITECTURE

COMPONENT STRUCTURE: The application follows a modular structure where major components are organized into reusable, functional components. These components are responsible for rendering specific parts of the user interface, such as forms, buttons, and data display areas.

STATE MANAGEMENT: The project primarily uses local state within React components. For managing state across multiple components, React's Context API or other state management solutions like Redux could be implemented in the future.

ROUTING: Routing is handled using react-router-dom, allowing users to navigate between different pages without refreshing the browser. The routes are dynamically configured to match various URL paths and render corresponding components.

4. SETUP INSTRUCTIONS

PRE-REQUISITES:

The project requires Node.js installed on the system.

INSTALLATION:

1. Clone the repository: git clone <repository-url>
2. Navigate to the project directory: cd client
3. Install dependencies: npm install
4. Configure any environment variables if needed.

5. FOLDER STRUCTURE

CLIENT: The React application is organized with the main folders such as components (for reusable UI components), pages (for individual route components), assets (for static files like images), and styles (for CSS or styled-components).

UTILITIES: Any helper functions, utility classes, or custom hooks used in the project are stored in the utils folder. These can include functions for API requests, data processing, or custom hooks for managing component states.

6.RUNNING THE APPLICATION

To start the frontend server locally, run the following command in the client directory:

- **npm start :**

  This will launch the application in development mode and open it in a browser.

7.COMPONENT DOCUMENTATION

KEY COMPONENTS: Major components include Navbar (for navigation), Footer (for the footer section), HomePage (for the landing page), and Login (for user authentication). Each component receives props such as data, user authentication status, or event handlers.

REUSABLE COMPONENTS: Components like Button, Input, and Modal are designed to be reusable throughout the application with configurable props to manage their appearance and behavior.

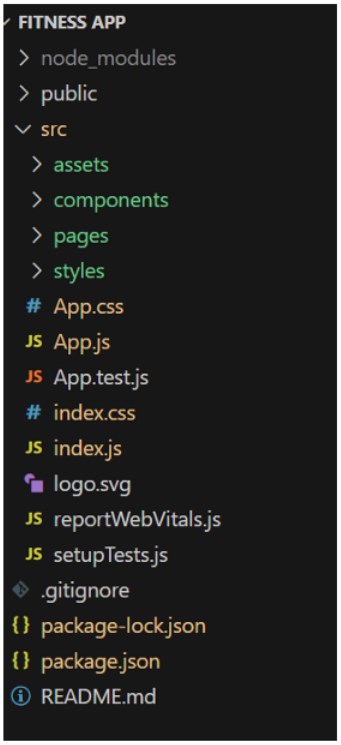
8.STATE MANAGEMENT

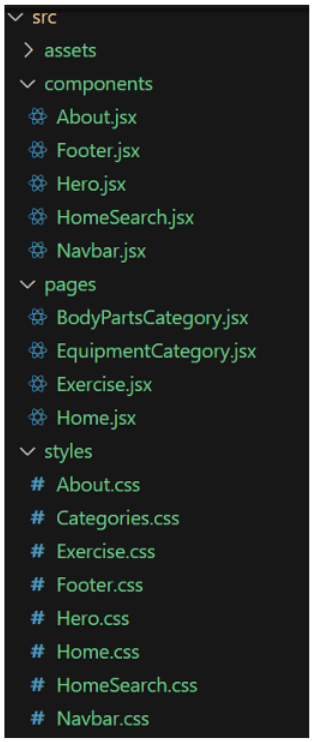
Global State: Global state management could be implemented with Context API or Redux in the future, ensuring that state like user authentication or global settings are accessible across different components.

Local State: Local state is handled using React's built-in useState hook, which is used to store data that is specific to individual components, such as form input values or UI visibility toggles.

9.USER INTERFACE

The UI includes various interactive elements like forms, buttons, and lists, designed for a smooth user experience. Screenshots or GIFs of key UI features could demonstrate the interface's responsiveness and interaction design.





10.STYLING

CSS Frameworks/Libraries: The project may use CSS frameworks like Bootstrap or Material-UI for predefined components and responsive design. Alternatively, custom CSS or pre-processors like Sass could be used for styling.

Theming: The project can implement a custom theme or design system for consistent styling across all components, allowing easy changes to the color palette, typography, and layout.

11.TESTING

Testing Strategy: Unit tests for individual components are written using Jest and React Testing Library. Integration tests ensure that components work together as expected, and end-to-end tests could be implemented using tools like Cypress.

Code Coverage: Code coverage tools such as Jest's built-in coverage feature are used to ensure the tests adequately cover all major code paths.

11.1 CODING:

```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
```

```
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.6.2",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.12.0",
    "react-router-dom": "^6.21.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
```

```
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

13.SCREENSHOTS OR DEMO

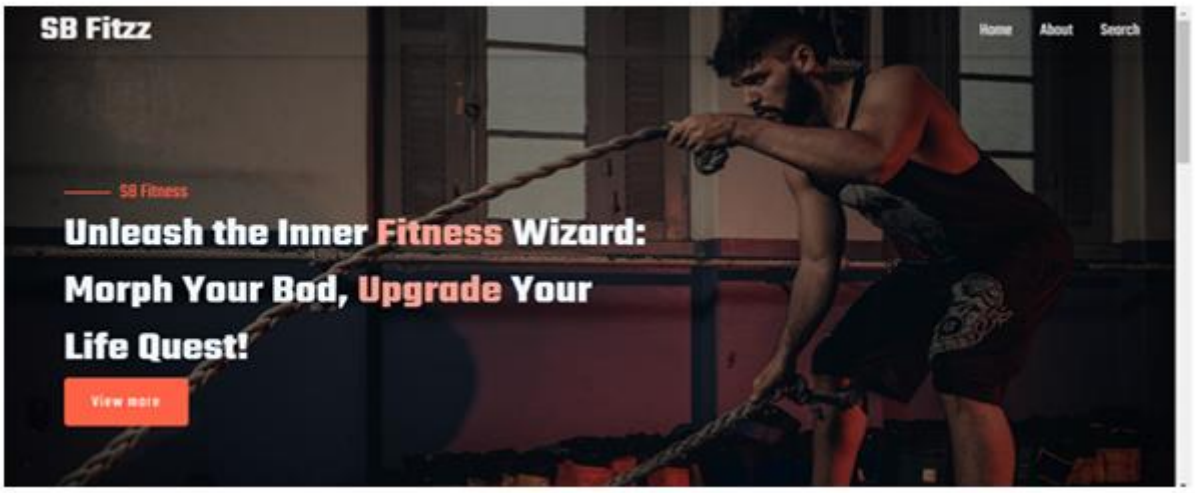


Fig 13.1 Home screen

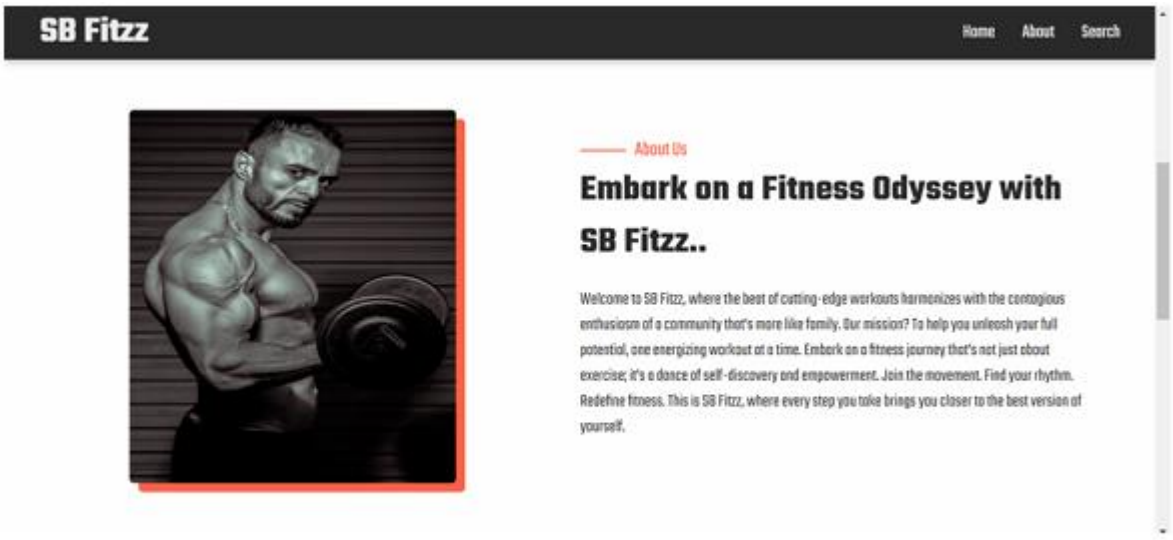


Fig 13.2 About screen

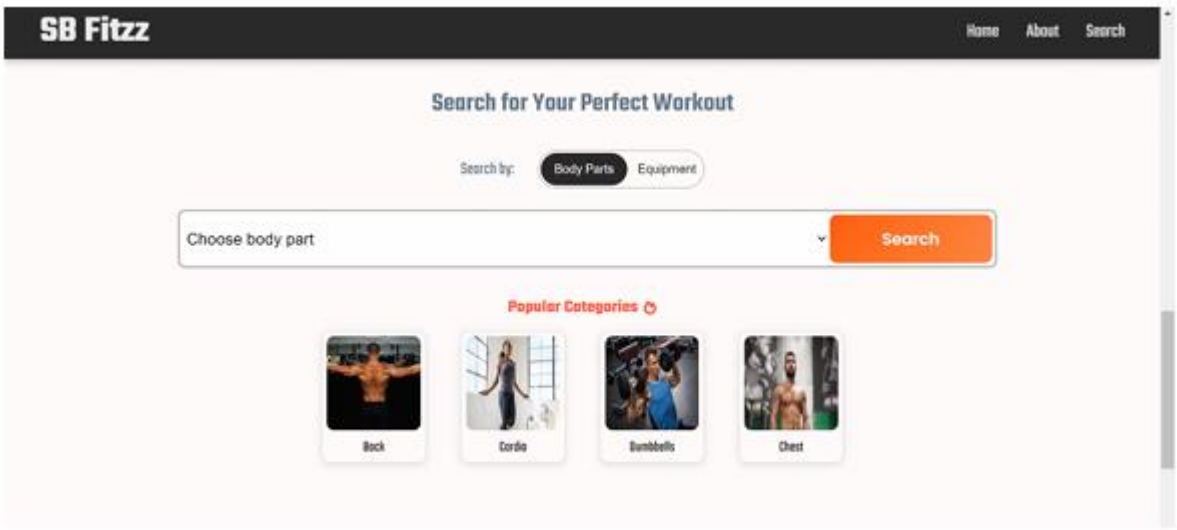


Fig 13.3 Search screen

14. KNOWN ISSUES

Any known issues or bugs in the application are documented here for both users and developers. These might include UI glitches, compatibility issues, or unhandled edge cases.

15. FUTURE ENHANCEMENTS

Future improvements could include adding new components like a dynamic dashboard, improving animations or transitions, optimizing performance, or integrating new API features for enhanced functionality.