**PROJECT DOCUMENTATION**

FITFLEX : YOUR PERSONAL FITNESS COMPANSION

**Introduction**

* PROJECT TITLE : FITFLEX : YOUR PERSONAL FITNESS COMPANSION
* TEAM ID : NM2025TMID37136
* TEAM LEADER : S.PRIYA DHARSHINI & MAIL ID : [priyasuresh2608@gmail.com](mailto:priyasuresh2608@gmail.com)
* TEAM MEMBERS :
* SHRUTHI.K & [shruthi20061995@gmail.com](mailto:shruthi20061995@gmail.com)
* POORNIMA.P & [poornimagg2007@gmail.com](mailto:poornimagg2007@gmail.com)
* PRIYA DHARSHINI.B & pd73355832@gmail.com
* MEENA.T & [gunaguna2454@gmail.com](mailto:gunaguna2454@gmail.com)

Frontend Architecture (React.js with Bootstrap & Material UI)

The frontend of FitFlex is responsible for providing an engaging, responsive, and user-friendly interface. Since fitness applications are highly interactive, the frontend is designed to handle:

User Dashboard: Displays workout plans, diet suggestions, and progress charts.

Workout Player: Shows step-by-step exercise tutorials with embedded videos or GIFs.

Progress Visualization: Graphs and analytics powered by chart libraries to track calories, workouts, and BMI.

Community Interaction: Pages for leaderboards, group challenges, and chat support.

Admin Panel (UI): Secure login for admins to manage workouts, diet plans, and users.

Key Frontend Components:

React Components – Modular UI blocks for workout cards, diet lists, charts, and reminders.

State Management – Redux or Context API to manage user data consistently across pages.

**SOURCE CODE**

import { Route, Routes } from 'react-router-dom';

import './App.css';

import Home from './pages/Home';

import Exercise from './pages/Exercise';

import Navbar from './components/Navbar';

import Footer from './components/Footer';

import BodyPartsCategory from './pages/BodyPartsCategory';

import EquipmentCategory from './pages/EquipmentCategory';

function App() {

  return (

    <div className="App">

      <Navbar />

      <Routes>

        <Route path="/" element={<Home />} />

        <Route path="/bodyPart/:id" element={<BodyPartsCategory />} />

        <Route path="/equipment/:id" element={<EquipmentCategory />} />

        <Route path="/exercise/:id" element={<Exercise />} />

      </Routes>

      <Footer />

    </div>

  );

}

export default

2. Backend Architecture (Node.js + Express.js)

The backend acts as the brain of the system, managing user authentication, business logic, and communication between the frontend and database.

Authentication & Authorization:

JWT (JSON Web Token) used for secure login and session management.

Role-based access

3. Database Architecture (MongoDB)

The system uses MongoDB, a NoSQL database, for storing structured and unstructured data efficiently.

Collections:

1. Users – user details, goals, subscription plans.

2. Workouts – exercise library, workout categories, difficulty levels.

3. Diets – meal plans, recipes, nutritional data.

4. Progress – logs of workouts completed, calories burned, BMI updates.

5. Community – chat messages, challenge details, leaderboard scores.

6. Admin Data – admin users, activity logs, updates to workouts/diets.

 SETUP INSTRUCTION

Prerequisites

Before setting up and running the FitFlex project, the following tools, frameworks, and environments must be installed and properly configured:

1. Node.js

Required to run the backend server (Express.js) and manage dependencies.

Version: v16 or above recommended.

Download from: https://nodejs.org

Verify installation:

node -v

npm -v

2. MongoDB

Used as the primary database to store user profiles, workouts, diets, progress logs, and chat messages.

Can be installed locally or connected via MongoDB Atlas (cloud-based).

Version: MongoDB 5.0 or above recommended.

Verify installation:

mongod --version

3. Git

Required to clone the project repository and manage version control.

 Installation Steps

Follow these steps to successfully install and run the FitFlex application on your system:

Step 1: Clone the Repository

Open a terminal or command prompt and clone the project repository using Git:

git clone <repository-url>

Move into the project directory:

cd fitflex

Step 2: Install Backend Dependencies

Navigate to the server (backend) folder and install the required packages:

cd server

npm install

This will install dependencies such as:

express – web framework

mongoose – MongoDB connection

bcrypt – password hashing

jsonwebtoken – authentication

cors, dotenv, body-parser – middleware and configurations

Step 3: Install Frontend Dependencies

Navigate to the client (frontend) folder and install the required packages:

cd ../client

npm install

This will install dependencies such as:

react – frontend framework

react-router-dom – navigation between pages

redux / context API – state management

bootstrap / material-ui – UI components

axios – HTTP client for API requests

Step 4: Configure Environment Variables

Create a .env file in the server folder with the following configuration:

PORT=5000

MONGO\_URI=mongodb://localhost:27017/fitflex

JWT\_SECRET=your\_secret\_key

PORT – defines which port the backend server runs on.

MONGO\_URI – connection string for MongoDB (local or cloud).

JWT\_SECRET – secret key for authentication tokens.

If you are using MongoDB Atlas (cloud), replace mongodb://localhost:27017/fitflex with your cluster connection string.

Step 5: Start the Backend Server

In the server folder, start the backend server:

npm run dev

You should see a message like:

Server running on http://localhost:5000

MongoDB connected..

Step 6: Start the Frontend Client

Open a new terminal, go to the client folder, and start the React app:

npm start

By default, the frontend will run on:

http://localhost:3000

Step 7: Test the Application

1. Open http://localhost:3000 in your browser.

2. Register a new user and log in.

3. Explore features such as workout plans, diet suggestions, and progress tracking.

4. Test API endpoints (optional) using Postman/Insomnia.

 FOLDER STRUCTURE

The FitFlex project is organized into a modular folder structure to separate concerns between frontend, backend, and configuration files. This ensures scalability, maintainability, and ease of collaboration.

fitflex/

│

├── client/ # Frontend (React.js)

│ ├── public/ # Static assets (index.html, icons, images)

│ ├── src/ # React source code

│ │ ├── assets/ # Images, fonts, stylesheets

│ │ ├── components/ # Reusable UI components (Navbar, Footer, Cards)

│ │ ├── pages/ # Main pages (Login, Dashboard, Workouts, Diets, Community)

│ │ ├── redux/ # State management (actions, reducers, store)

│ │ ├── services/ # API calls (axios services for workouts, diets, users)

│ │ ├── utils/ # Helper functions (validation, constants)

│ │ ├── App.js # Main React app component

│ │ ├── index.js # Entry point for React

│ │ └── styles.css # Global styles

│ └── package.json # Frontend dependencies

│

├── server/ # Backend (Node.js + Express)

│ ├── config/ # Database connection, environment configs

│ ├── controllers/ # Route logic (userController, workoutController, dietController)

│ ├── middleware/ # Authentication, error handling, validation

│ ├── models/ # Mongoose schemas (User, Workout, Diet, Progress, Community)

│ ├── routes/ # API routes (userRoutes, workoutRoutes, dietRoutes)

│ ├── utils/ # Utility functions (token generation, logging)

│ ├── server.js # Entry point for backend server

│ └── package.json # Backend dependencies

│

├── .env # Environment variables (PORT, MONGO\_URI, JWT\_SECRET)

├── .gitignore # Ignored files for Git

├── README.md # Project overview and setup guide

└── package.json # Root-level config (if using combined scripts)

 Running the Application

Frontend (React.js)

Once the installation and setup are completed, the frontend (client) can be started using the following steps:

Step 1: Navigate to the Client Folder

Open your terminal or command prompt and go into the client directory:

cd client

Step 2: Start the Development Server

Run the following command to start the React application:

npm start

This will start the development server on http://localhost:3000/ by default.

The terminal will display something like:

Compiled successfully!

You can now view fitflex in the browser.

Local: http://localhost:3000

Backend (Node.js + Express.js)

The backend server powers all business logic, authentication, and database operations for FitFlex. After completing the setup and installation, follow these steps to run the backend:

Step 1: Navigate to the Server Folder

Open your terminal and move into the server directory:

cd server

Step 2: Start the Backend Server

Run the backend using the following command:

npm run dev

This starts the server in development mode using nodemon (if configured).

Alternatively, to run without nodemon:

node server.js

Step 3: Verify the Server is Running

If everything is configured correctly, the terminal will display messages such as:

Server running on http://localhost:5000

MongoDB connected successfully...

http://localhost:5000 is the base URL for backend APIs.

If using a different PORT in your .env file, replace 5000 with that number.

Step 4: Test API Endpoints

Use Postman or Insomnia to test backend endpoints. Examples:

User Registration

POST http://localhost:5000/api/users/register

User Login

POST http://localhost:5000/api/users/login

Get Workouts

 GET http://localhost:5000/api/workouts

Log Progress

POST http://localhost:5000/api/progress

If these return JSON responses, the backend is working correctly.

Step 5: Stop the Backend Server

To stop the backend server, press:

CTRL + C

in the terminal where the server is running.

 APL DOCUMENTATION

User

The User API manages registration, authentication, and profile-related operations for FitFlex users. All endpoints follow RESTful principles and return data in JSON format.

---

1. Register User

Endpoint:

POST /api/users/register

Description:

Creates a new user account in the system.

Request Body (JSON):

{

"name": "John Doe",

"email": "johndoe@example.com",

"password": "securePassword123"

}

Response (Success 201):

{

"message": "User registered successfully",

"user": {

"\_id": "64f9cbd92f7a3a10b9d8e8c5",

"name": "John Doe",

"email": "johndoe@example.com"

},

"token": "jwt-authentication-token"

}

Response (Error 400 – Email Exists):

{

"error": "User already exists with this email"

}

---

2. User Login

Endpoint:

POST /api/users/login

Description:

Authenticates the user and returns a JWT token for further requests.

Request Body (JSON):

{

"email": "johndoe@example.com",

"password": "securePassword123"

}

Response (Success 200):

{

"message": "Login successful",

"user": {

"\_id": "64f9cbd92f7a3a10b9d8e8c5",

"name": "John Doe",

"email": "johndoe@example.com"

},

"token": "jwt-authentication-token"

}

Response (Error 401 – Invalid Credentials):

{

"error": "Invalid email or password"

}

**ARCHITECTURE**

Frontend Architecture (React.js with Bootstrap & Material UI)

The frontend of FitFlex is responsible for providing an engaging, responsive, and user-friendly interface. Since fitness applications are highly interactive, the frontend is designed to handle:

User Dashboard: Displays workout plans, diet suggestions, and progress charts.

Workout Player: Shows step-by-step exercise tutorials with embedded videos or GIFs.

Progress Visualization: Graphs and analytics powered by chart libraries to track calories, workouts, and BMI.

Community Interaction: Pages for leaderboards, group challenges, and chat support.

Admin Panel (UI): Secure login for admins to manage workouts, diet plans, and users.

Key Frontend Components:

React Components – Modular UI blocks for workout cards, diet lists, charts, and reminders.

State Management – Redux or Context API to manage user data consistently across pages.

UI Frameworks – Bootstrap and Material UI for responsive layouts.

Routing – React Router for smooth navigation between pages.

By separating UI into reusable components, scalability and maintainability are ensured.

2. Backend Architecture (Node.js + Express.js)

The backend acts as the brain of the system, managing user authentication, business logic, and communication between the frontend and database.

Authentication & Authorization:

JWT (JSON Web Token) used for secure login and session management.

Role-based access

3. Database Architecture (MongoDB)

The system uses MongoDB, a NoSQL database, for storing structured and unstructured data efficiently.

Collections:

1. Users – user details, goals, subscription plans.

2. Workouts – exercise library, workout categories, difficulty levels.

3. Diets – meal plans, recipes, nutritional data.

4. Progress – logs of workouts completed, calories burned, BMI updates.

5. Community – chat messages, challenge details, leaderboard scores.

6. Admin Data – admin users, activity logs, updates to workouts/diets.

* SETUP INSTRUCTION

Prerequisites

Before setting up and running the FitFlex project, the following tools, frameworks, and environments must be installed and properly configured:

1. Node.js

Required to run the backend server (Express.js) and manage dependencies.

Version: v16 or above recommended.

Download from: https://nodejs.org

Verify installation:

node -v

npm -v

2. MongoDB

Used as the primary database to store user profiles, workouts, diets, progress logs, and chat messages.

Can be installed locally or connected via MongoDB Atlas (cloud-based).

Version: MongoDB 5.0 or above recommended.

Verify installation:

mongod --version

3. Git

Required to clone the project repository and manage version control.

* **FOLDER STRUCTURE**

The FitFlex project is organized into a modular folder structure to separate concerns between frontend, backend, and configuration files. This ensures scalability, maintainability, and ease of collaboration.

fitflex/

│

├── client/ # Frontend (React.js)

│ ├── public/ # Static assets (index.html, icons, images)

│ ├── src/ # React source code

│ │ ├── assets/ # Images, fonts, stylesheets

│ │ ├── components/ # Reusable UI components (Navbar, Footer, Cards)

│ │ ├── pages/ # Main pages (Login, Dashboard, Workouts, Diets, Community)

│ │ ├── redux/ # State management (actions, reducers, store)

│ │ ├── services/ # API calls (axios services for workouts, diets, users)

│ │ ├── utils/ # Helper functions (validation, constants)

│ │ ├── App.js # Main React app component

│ │ ├── index.js # Entry point for React

│ │ └── styles.css # Global styles

│ └── package.json # Frontend dependencies

│

├── server/ # Backend (Node.js + Express)

│ ├── config/ # Database connection, environment configs

│ ├── controllers/ # Route logic (userController, workoutController, dietController)

│ ├── middleware/ # Authentication, error handling, validation

│ ├── models/ # Mongoose schemas (User, Workout, Diet, Progress, Community)

│ ├── routes/ # API routes (userRoutes, workoutRoutes, dietRoutes)

│ ├── utils/ # Utility functions (token generation, logging)

│ ├── server.js # Entry point for backend server

│ └── package.json # Backend dependencies

│

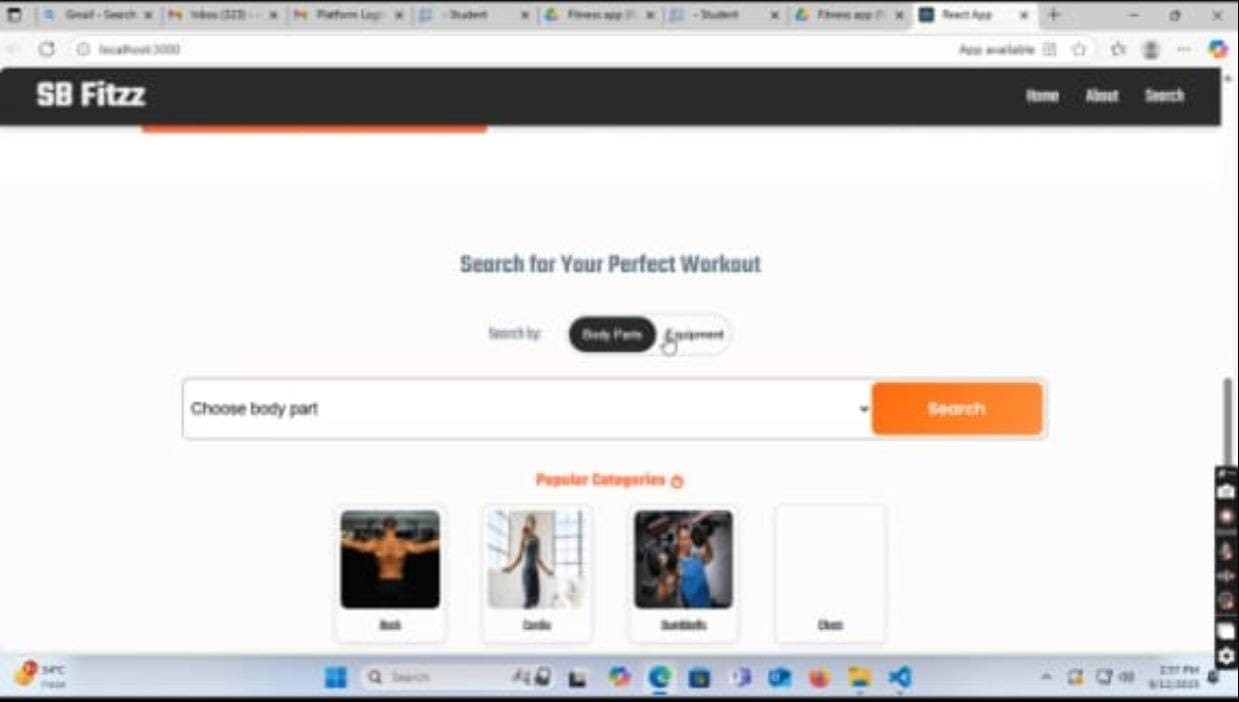
├── .env # Environment variables (PORT, MONGO\_URI, JWT\_SECRET)

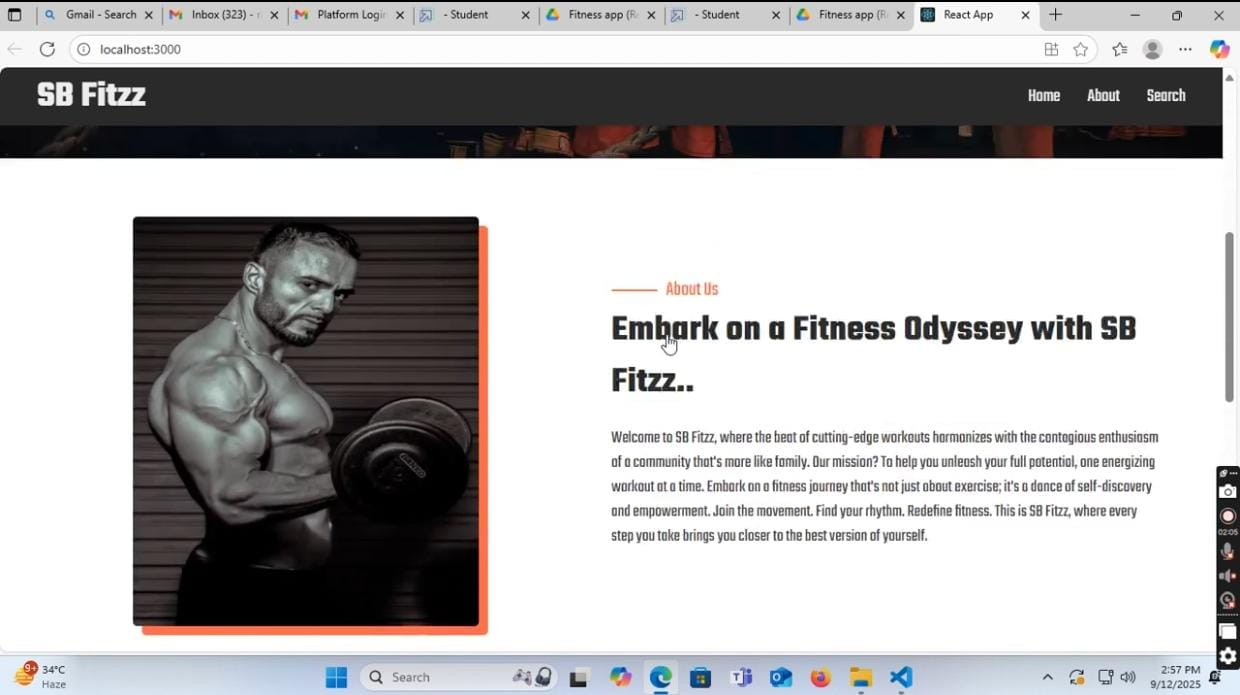
├── .gitignore # Ignored files for Git

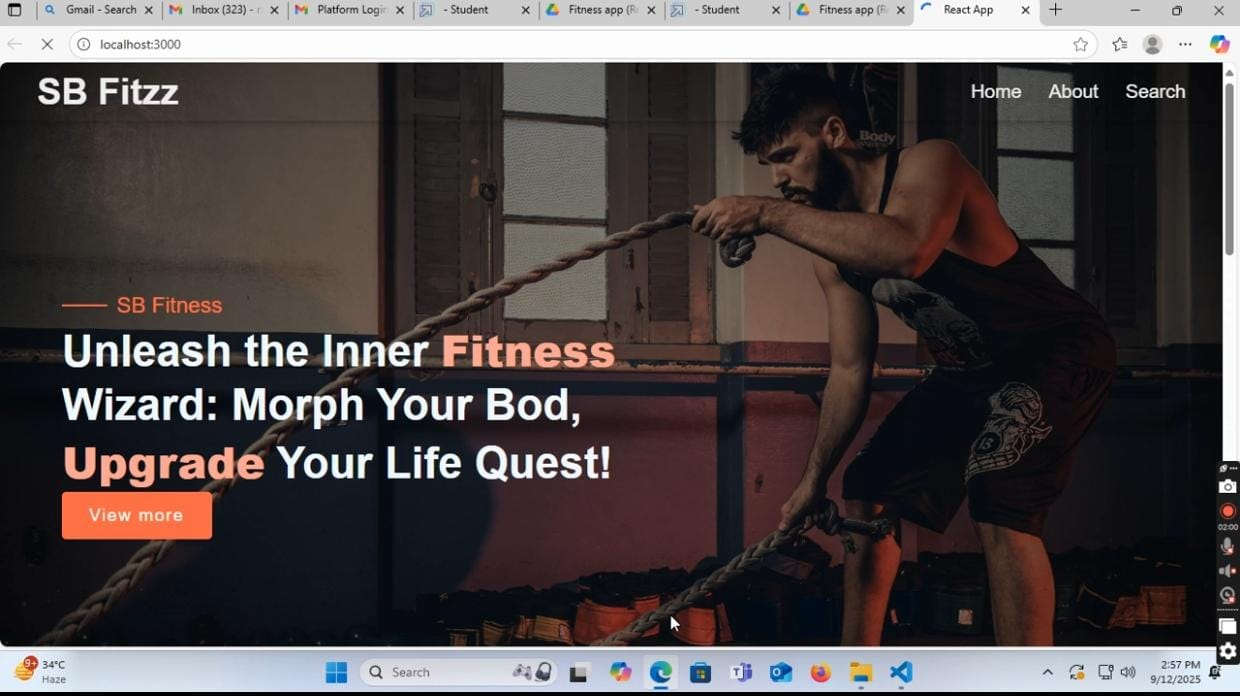
├── README.md # Project overview and setup guide

└── package.json # Root-level config (if using combined scripts)

OUTPUT SCREEN SHORT







**DEMO LINK**

https://drive.google.com/file/d/1eJGhVhO7osp4YK00dr\_qWgkFHVivs\_Ie/view?usp=sharing