**FITFLEX**

**Project Documentation**

**1.INTRODUCTION:**

Project Title: FITFLEX-Fitness partner

* TEAM ID:
* Team member: VARSHINI G (Leader|)
* Team member: BHARATHI S S
* Team member: SARAVANAKUMAR V
* Team member: MOHAMMAD ASKAR k

**2. Project Overview**

* **Name**: FitFlex NM
* **Purpose**: To provide users with personalized fitness and nutrition management (“NM” likely stands for Nutrition & Management / or maybe “New Module” etc.)
* **Target Users**: People wanting to track workouts, lose or maintain weight, gain muscle, plan meals, etc.
* **Main Features**
  1. Daily/weekly workout plans tailored by user goals (weight loss, muscle gain, general fitness)
  2. Nutrition / diet recommendation system
  3. Progress tracking (workouts completed, calories consumed / burned, weight changes, etc.)
  4. User authentication, profiles, perhaps different subscription plans
  5. Possibly cross‑location or schedule flexibility for workouts (if that’s part of “FitFlex”)

**3. Architecture & Technology Stack**

* **Frontend**: React.js (or similar), or mobile app if applicable
* **Backend**: Node.js + Express, or Python / Flask / Django (if diet‑recommendation API, etc.)
* **Database**: MongoDB or relational DB depending on user / plan model
* **APIs**: For workout data, nutrition info, perhaps third‑party integration (food databases, etc.)
* **Hosting / Deployment**: e.g. cloud servers, Docker, maybe Netlify / Vercel / AWS etc.

**4. System Design**

* Components and Modules
  + **User Module**: Registration, authentication, profile management
  + **Workout Module**: Plan creation, schedule, tracking
  + **Nutrition Module**: Meal / diet plans, calorie tracking, food database
  + **Progress Module**: Charts, metrics
  + **Admin / Subscription Module**: Manage subscriptions, offers, pricing
* Data Models / Database Schema: Users, Plans, Meals, Exercises, Logs etc.
* API endpoints: e.g.
* POST /signup
* POST /login
* GET /workouts
* POST /workouts/log
* GET /meals /recommendations
* GET /progress
* Security: JWT tokens, password hashing, validation, data privacy

**5. UX / UI Design**

* Wireframes or mockups of major screens/pages: Signup, Dashboard, Workout plan, Meal planner, Progress graphs
* Navigation structure

**6. Project Structure (Code Organization)**

* Folder / file organization
* Key dependencies / modules
* How to run locally, prerequisites

**7. Challenges & Solutions**

* Any tricky parts (e.g. matching diet plans to user preferences / allergies, integrating third party APIs, handling offline data or state, etc.)
* How they were / will be solved

**8. Testing & Quality Assurance**

* Unit tests, integration tests
* Edge cases (e.g. invalid input, network failures)
* Performance (e.g. for large food/exercise databases)

**9. Deployment & Maintenance**

* CI/CD setup
* Versioning
* Monitoring & logs
* Updates / maintenance plan

**10. Roadmap & Future Enhancements**

* Features planned in future (e.g. social sharing, AI‑based coaching, custom meal upload, wearable integration)
* Scalability improvements

**11. Documentation & Resources**

* API documentation (endpoints, request/response formats)
* User manual / onboarding materials
* Developer guide (how to contribute, setup, coding standards)

**1. Project Overview**

* **Name**: FitFlex NM
* **Purpose**: To provide users with personalized fitness and nutrition management (“NM” likely stands for Nutrition & Management / or maybe “New Module” etc.)
* **Target Users**: People wanting to track workouts, lose or maintain weight, gain muscle, plan meals, etc.
* **Main Features**
  1. Daily/weekly workout plans tailored by user goals (weight loss, muscle gain, general fitness)
  2. Nutrition / diet recommendation system
  3. Progress tracking (workouts completed, calories consumed / burned, weight changes, etc.)
  4. User authentication, profiles, perhaps different subscription plans
  5. Possibly cross‑location or schedule flexibility for workouts (if that’s part of “FitFlex”)

**2. Architecture & Technology Stack**

* **Frontend**: React.js (or similar), or mobile app if applicable
* **Backend**: Node.js + Express, or Python / Flask / Django (if diet‑recommendation API, etc.)
* **Database**: MongoDB or relational DB depending on user / plan model
* **APIs**: For workout data, nutrition info, perhaps third‑party integration (food databases, etc.)
* **Hosting / Deployment**: e.g. cloud servers, Docker, maybe Netlify / Vercel / AWS etc.

**3. System Design**

* Components and Modules
  + **User Module**: Registration, authentication, profile management
  + **Workout Module**: Plan creation, schedule, tracking
  + **Nutrition Module**: Meal / diet plans, calorie tracking, food database
  + **Progress Module**: Charts, metrics
  + **Admin / Subscription Module**: Manage subscriptions, offers, pricing
* Data Models / Database Schema: Users, Plans, Meals, Exercises, Logs etc.
* API endpoints: e.g.
* POST /signup
* POST /login
* GET /workouts
* POST /workouts/log
* GET /meals /recommendations
* GET /progress
* Security: JWT tokens, password hashing, validation, data privacy

**4. UX / UI Design**

* Wireframes or mockups of major screens/pages: Signup, Dashboard, Workout plan, Meal planner, Progress graphs
* Navigation structure

**5. Project Structure (Code Organization)**

* Folder / file organization
* Key dependencies / modules
* How to run locally, prerequisites

**6. Challenges & Solutions**

* Any tricky parts (e.g. matching diet plans to user preferences / allergies, integrating third party APIs, handling offline data or state, etc.)
* How they were / will be solved

**7. Testing & Quality Assurance**

* Unit tests, integration tests
* Edge cases (e.g. invalid input, network failures)
* Performance (e.g. for large food/exercise databases)

**8. Deployment & Maintenance**

* CI/CD setup
* Versioning
* Monitoring & logs
* Updates / maintenance plan

**9. Roadmap & Future Enhancements**

* Features planned in future (e.g. social sharing, AI‑based coaching, custom meal upload, wearable integration)
* Scalability improvements

**10. Documentation & Resources**

* API documentation (endpoints, request/response formats)
* User manual / onboarding materials
* Developer guide (how to contribute, setup, coding standards)