Project document

Fitflex:your personal fitness companion

1.Introduction:

Project title:Fitflex:your personal fitness companion

- Team ID:NM2025TMID42786
- Team Leader:Amrutha.G&amruthagovindappa@gmail.com
- Team Members:
    - Deepika S&deepikareddy@gmail.com
    - Devaki M&mdevaki2002@gmail.com
    - Dhatchiya V&dhatchu1820@gmail.com

2.Project overview

Purpose: The purpose of **FitFlex** is to provide a **personalized fitness companion application** that helps individuals track, manage, and improve their health and wellness journey. It aims to:

1. **Promote Healthy Lifestyle** – Encourage users to stay consistent with workouts, nutrition, and mindfulness.
2. **Personalized Fitness Plans** – Offer customized workout routines and diet recommendations based on user goals (weight loss, muscle gain, endurance, etc.).
3. **Progress Tracking** – Monitor daily activity, calories, and overall fitness improvements.
4. **Motivation & Engagement** – Provide reminders, challenges, and goal-setting features to keep users engaged.
5. **Accessibility** – Make fitness guidance available anytime, anywhere without the need for a personal trainer.

1. Features: **User Profile & Personalization**
    - Create and manage personal fitness profiles.
    - Set fitness goals (weight loss, muscle gain, strength, flexibility).
    - AI-driven customized workout and diet plans.
2. **Workout Plans**
    - Predefined workout routines (yoga, cardio, strength training, HIIT).
    - Step-by-step video demonstrations.
    - Adaptive difficulty levels (beginner, intermediate, advanced).
3. **Diet & Nutrition Tracking**
    - Personalized diet suggestions.
    - Calorie and nutrient tracking.

- Healthy recipe recommendations.

4. **Activity & Progress Tracking**
   - Daily steps, calories burned, and distance covered.
   - Weight and BMI monitoring.
   - Visual progress reports (charts, graphs).
5. **Reminders & Notifications**
   - Workout reminders.
   - Water intake notifications.
   - Meal-time alerts.
6. **Gamification & Motivation**
   - Daily/weekly challenges.
   - Badges and rewards for achievements.
   - Community leaderboards.
7. **Social & Community Support**
   - Connect with friends and other fitness enthusiasts.
   - Share achievements on social media.
   - Join fitness groups or challenges.
8. **Integration with Wearables**
   - Sync with smartwatches/fitness bands.
   - Heart rate and sleep tracking.
9. **Virtual Trainer Support**
   - AI-powered guidance during workouts.
   - Real-time posture correction (if integrated with sensors/camera).
10. **Offline & Online Accessibility**
    - Access workout videos offline.
    - Cloud sync for data backup

3.Architecture: FitFlex follows a **3-tier architecture** (Presentation Layer, Application Layer, and Data Layer) to ensure scalability, security, and ease of use.

1. Presentation Layer (Frontend / User Interface)

- Mobile App (Android/iOS) or Web Application.
- Built with **React Native / Flutter** (mobile) or **React.js / Angular** (web).
- Provides features like:
  - User registration/login.
  - Workout dashboards.
  - Diet/nutrition plans.
  - Progress charts.
  - Notifications/reminders.

2. Application Layer (Backend / Business Logic)

- Responsible for **processing requests and applying fitness algorithms**.
- Built with **Node.js / Django / Spring Boot** (any suitable framework).
- Key functionalities:

    - User authentication & session management.
    - Workout & diet plan customization (AI/ML recommendations).
    - Progress tracking calculations.
    - API integration with wearables (Google Fit, Apple Health).
    - Push notifications & reminders.

3. Data Layer (Database & Storage)

- Stores all user and system data securely.
- Possible technologies: **MySQL / PostgreSQL / MongoDB**.
- Data handled:
    - User profiles & goals.
    - Workout & nutrition data.
    - Progress history (weight, BMI, activity).
    - Community & challenge records.

4. External Services & Integrations (optional)

- **Wearables & IoT Integration** – Sync with smartwatches, fitness trackers.
- **Payment Gateway** – For premium subscriptions.
- **Cloud Hosting** – AWS, Firebase, or Azure for scalability.

Architecture Diagram (Textual)

```
  [ User (Mobile/Web App) ]
             |
      Presentation Layer
             |
   --------------------------
   |    Application Layer    |
   |  (APIs & Business Logic)|
   --------------------------
             |
      Data Layer (DB)
             |
 [External Services/Integrations]
```

4.Setup Instruction:.Prerequisites

Before setting up the project, ensure the following are installed:

- **Node.js** (for backend & frontend build)
- **npm / yarn** (package manager)
- **Database**: MySQL / MongoDB (based on project choice)
- **Git** (for version control)
- **Android Studio / Xcode** (if running on mobile)

2. Clone the Repository

```
git clone https://github.com/your-username/fitflex.git
cd fitflex
```

3. Install Dependencies

For frontend:

```
cd frontend
npm install
```

For backend:

```
cd backend
npm install
```

4. Configure Environment Variables

Create a `.env` file in the **backend** folder with:

```
PORT=5000
DB_URI=mongodb://localhost:27017/fitflex
JWT_SECRET=your_secret_key
CLOUD_URL=your_cloud_storage_url
```

5. Database Setup

- Start MongoDB or MySQL server.
- Create a database named **fitflex**.
- Run initial migration/seed scripts (if provided).

6. Run the Application

Start backend server:

```
cd backend
npm start
```

Start frontend application:

```
cd frontend
npm start
```

7. Access the Application

- Web App: http://localhost:3000
- Mobile App: Run via emulator (Android Studio/Xcode) or scan QR code if using **Expo**.

8. (Optional) Build for Production

Frontend build:

```
cd frontend
npm run build
```

Backend deployment:

- Use **Heroku / AWS / Firebase / Vercel** for hosting.

5.Folder structure: **FitFlex Folder Structure**

```
fitflex/
│
├── frontend/                 # Frontend (React / React Native / Flutter)
│   ├── public/               # Static assets (icons, images, index.html)
│   ├── src/                  # Main source code
│   │   ├── assets/           # Images, fonts, icons
│   │   ├── components/       # Reusable UI components (Navbar, Buttons, 
Cards)
│   │   ├── pages/            # Screens/Pages (Login, Dashboard, Workout, 
Profile)
│   │   ├── services/         # API service calls (Axios/Fetch)
│   │   ├── context/          # State management (Redux / Context API)
│   │   ├── utils/            # Helper functions
│   │   ├── App.js            # Main app entry
│   │   └── index.js          # React entry point
│   └── package.json          # Frontend dependencies
│
├── backend/                  # Backend (Node.js / Django / Spring Boot)
│   ├── config/               # DB & environment configurations
│   ├── controllers/          # Business logic (WorkoutController, 
UserController)
│   ├── models/               # Database schemas (User, Workout, Diet, 
Progress)
│   ├── routes/               # API routes (userRoutes, workoutRoutes)
│   ├── middlewares/          # Authentication, error handling
│   ├── utils/                # Helper functions (JWT, validations)
│   ├── server.js             # Entry point of backend server
│   └── package.json          # Backend dependencies
│
├── database/                 # Database related scripts
│   ├── migrations/           # Migration files
│   ├── seeds/                # Initial dummy data
│   └── schema.sql            # DB schema (if SQL)
│
├── docs/                     # Documentation
│   ├── project-report.docx
│   ├── architecture-diagram.png
│   └── setup-instructions.md
│
├── tests/                    # Unit & Integration tests
│   ├── frontend-tests/
│   └── backend-tests/
│
```

```
│
├── .env                          # Environment variables
├── .gitignore                    # Ignored files for git
├── README.md                     # Project overview
└── package.json                  # Root configuration (if monorepo style)
```

6.Running the Application: Step 1: Start the Backend Server

1. Navigate to the backend folder:
2. `cd backend`
3. Start the server:
4. `npm start`
   - The backend will run on **http://localhost:5000** (or the port defined in `.env`).

Step 2: Start the Frontend Application

1. Open a new terminal and go to the frontend folder:
2. `cd frontend`
3. Run the application:
4. `npm start`
   - The frontend will run on **http://localhost:3000**.

Step 3: Access the Application

- Open a web browser and visit:
  □ **http://localhost:3000**
- If using a mobile version (React Native / Expo):
  - Run `npm start` in frontend.
  - Scan the QR code with Expo Go app on mobile, or run on **Android Studio/Xcode emulator**.

Step 4: Testing the Application

- Create a new user account.
- Log in and set fitness goals.
- Explore features: workout plans, diet tracking, progress charts, and reminders.

Step 5: Stop the Application

- Press `CTRL + C` in both backend and frontend terminals to stop servers.

7.API Documentation: Authentication

1. Register User

- **Endpoint:** `/auth/register`

- **Method:** `POST`
- **Description:** Creates a new user account.
- **Request Body:**

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "mypassword"
}
```

- **Response:**

```
{
  "message": "User registered successfully",
  "userId": "64aef01abc23"
}
```

2. Login User

- **Endpoint:** `/auth/login`
- **Method:** `POST`
- **Description:** Logs in the user and returns a JWT token.
- **Request Body:**

```
{
  "email": "john@example.com",
  "password": "mypassword"
}
```

- **Response:**

```
{
  "token": "jwt_token_here",
  "userId": "64aef01abc23"
}
```

User Profile

3. Get User Profile

- **Endpoint:** `/users/:id`
- **Method:** `GET`
- **Headers:**
    - `Authorization: Bearer <token>`
- **Response:**

```
{
  "id": "64aef01abc23",
  "name": "John Doe",
  "email": "john@example.com",
  "goals": "Weight Loss",
  "age": 25,
  "weight": 72,
  "height": 175
}
```

4. Update User Profile

- **Endpoint:** `/users/:id`
- **Method:** `PUT`
- **Headers:**
  - `Authorization: Bearer <token>`
- **Request Body:**

```
{
  "weight": 70,
  "height": 175,
  "goals": "Muscle Gain"
}
```

- **Response:**

```
{
  "message": "Profile updated successfully"
}
```

Workout & Fitness

5. Get All Workouts

- **Endpoint:** `/workouts`
- **Method:** `GET`
- **Headers:**
  - `Authorization: Bearer <token>`
- **Response:**

```
[
  {
    "id": "w101",
    "name": "Push Ups",
    "type": "Strength",
    "duration": "15 min",
    "caloriesBurned": 50
  },
  {
    "id": "w102",
    "name": "Jogging",
    "type": "Cardio",
    "duration": "30 min",
    "caloriesBurned": 150
  }
]
```

6. Log Workout Progress

- **Endpoint:** `/progress`
- **Method:** `POST`
- **Headers:**
    - `Authorization: Bearer <token>`
- **Request Body:**

```
{
  "userId": "64aef01abc23",
  "workoutId": "w101",
  "duration": "20 min",
  "caloriesBurned": 60
}
```

- **Response:**

```
{
  "message": "Workout progress logged successfully"
}
```

Diet & Nutrition

7. Get Diet Plan

- **Endpoint:** `/diet/:userId`
- **Method:** `GET`
- **Headers:**
    - `Authorization: Bearer <token>`
- **Response:**

```
{
  "userId": "64aef01abc23",
  "calorieIntake": 2000,
  "meals": [
    { "meal": "Breakfast", "food": "Oats with milk", "calories": 300 },
    { "meal": "Lunch", "food": "Brown rice, dal, salad", "calories": 600 },
    { "meal": "Dinner", "food": "Grilled chicken with veggies", "calories":
500 }
  ]
}
```

Reminders & Notifications

8. Set Reminder

- **Endpoint:** `/reminders`
- **Method:** `POST`
- **Headers:**
    - `Authorization: Bearer <token>`

- **Request Body:**

```
{
  "userId": "64aef01abc23",
  "type": "Workout",
  "time": "07:00 AM"
}
```

- **Response:**

```
{
  "message": "Reminder set successfully"
}
```

8.Authentication:

1. Authentication Type

- **JWT (JSON Web Token) Based Authentication**
- Ensures secure, stateless communication between frontend and backend.

2. Authentication Flow

1. **User Registration (Sign Up)**
   - Endpoint: `/auth/register`
   - A new user creates an account by providing `name`, `email`, `password`, `age`, `weight`, `height`, `goals`.
   - Password is **hashed** using `bcrypt` before storing in the database.
2. **User Login**
   - Endpoint: `/auth/login`
   - User enters `email` and `password`.
   - If valid, the backend generates a **JWT token** with user ID and role.
3. **Token Storage**
   - The JWT token is stored in the **frontend (localStorage / AsyncStorage)**.
   - Token is attached to every **authorized API request** in the `Authorization` header:
4. `Authorization: Bearer <jwt_token>`
5. **Access Protected Routes**
   - Routes like `/users/:id`, `/workouts`, `/diet/:userId` require authentication.
   - Middleware verifies the token before allowing access.
6. **Logout**
   - Frontend clears the stored JWT token.
   - User session ends.

3. Example API Requests

Register User

```
POST /api/auth/register
Content-Type: application/json
{
  "name": "Mamatha",
  "email": "mamatha@example.com",
  "password": "mypassword",
  "age": 18,
  "weight": 55,
  "height": 160,
  "goals": "Weight Loss"
}
```

Login User

```
POST /api/auth/login
Content-Type: application/json
{
  "email": "mamatha@example.com",
  "password": "mypassword"
}
```

Response:

```
{
  "token": "jwt_token_here",
  "userId": "64af123xyz",
  "message": "Login successful"
}
```

9.User Interface:

The **FitFlex User Interface** is designed to be **simple, interactive, and user-friendly** so that users of all ages can easily manage their fitness journey. The UI follows a **mobile-first responsive design** to support both web and mobile platforms.

1. Login & Registration Page

- Clean and minimal design.
- Fields for email, password, and new user signup.
- “Forgot Password” option.
- Social logins (optional: Google/Facebook).

2. Dashboard (Home Screen)

- Personalized greeting (e.g., *"Hello, Mamatha □"*).
- Quick overview of:
  - Today's steps, calories burned, and workout progress.
  - Water intake reminder.
  - Daily motivational quote.
- Navigation menu for **Workouts, Diet, Progress, Profile, Settings**.

3. Workout Screen

- List of workouts (Cardio, Strength, Yoga, HIIT).
- Each workout card shows: name, duration, calories burnt.
- Play button to start guided video tutorials.
- Option to log progress after completion.

4. Diet & Nutrition Screen

- Displays daily meal plan (Breakfast, Lunch, Dinner, Snacks).
- Calorie intake tracker with progress bar.
- Suggested healthy recipes with nutrition info.
- Add custom food items manually.

5. Progress Tracking Screen

- Visual graphs/charts showing:
  - Weight changes over time.
  - BMI trends.
  - Calories burnt weekly/monthly.
- Comparison of set goals vs actual progress.

6. Reminder & Notification Screen

- Set reminders for:
  - Workout times.
  - Water intake.
  - Meal times.
- Push notifications on mobile.

7. User Profile Screen

- View & edit profile (age, weight, height, goals).
- Update password and preferences.
- Track subscription (Free / Premium).

8. Community & Challenges (Optional Feature)

- Join fitness challenges.
- Share achievements with friends.
- Leaderboard to encourage motivation.

UI Style Highlights

- **Colors**: Soft gradients (green, blue, white) for a healthy & calm feel.
- **Typography**: Clean, readable fonts.
- **Navigation**: Bottom navigation bar for quick access.
- **Charts/Graphs**: Simple and interactive visuals.

10.Testing:

Testing is an essential phase in the **FitFlex project** to ensure that all features work correctly, securely, and efficiently. Multiple levels of testing are applied to validate both the **frontend** and **backend**.

1. Unit Testing

- Tests individual components and modules.
- Example:
  - Checking if **User Registration API** creates users correctly.
  - Verifying if the **Workout Calculator** returns correct calories burned.
- Tools: **Jest, Mocha, Jasmine**.

2. Integration Testing

- Tests how different modules work together.
- Example:
  - Ensure **login system** passes a valid token to protected routes.
  - Check if **frontend workout tracker** fetches correct data from backend API.
- Tools: **Supertest, Postman, Cypress**.

3. Functional Testing

- Validates that features work as expected based on requirements.
- Example Test Cases:
  - Register → Login → View Dashboard.
  - Add workout → Log progress → Check updated progress chart.
  - Set reminder → Receive notification at correct time.

4. User Interface (UI) Testing

- Ensures the application looks and behaves correctly across devices.
- Example:
  - Buttons, forms, navigation menus are responsive.
  - Charts and progress bars display correctly.
- Tools: **Selenium, Cypress**.

5. Security Testing

- Ensures user data is protected.
- Example:
  - Passwords stored using **bcrypt hashing**.
  - Only authenticated users can access protected APIs.
  - JWT tokens expire after set duration.
- Tools: **OWASP ZAP, Postman**.

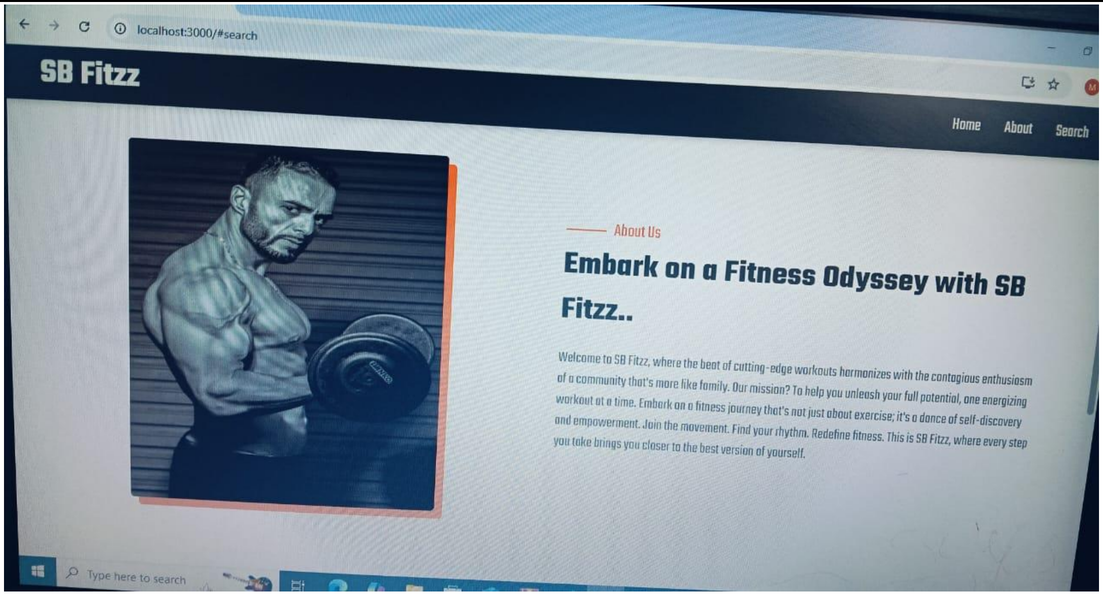
6. Performance Testing

- Checks app performance under load.
- Example:
  - Backend handles multiple API requests at once.
  - Frontend loads workout data within seconds.
- Tools: **Apache JMeter, Locust**.

7. User Acceptance Testing (UAT)

- Final testing with real users.
- Ensures the app meets expectations for **usability, simplicity, and fitness tracking accuracy**.

11.Screenshots or Demo:



Embark on a Fitness Odyssey with SB Fitzz..

Welcome to SB Fitzz, where the beat of cutting-edge workouts harmonizes with the contagious enthusiasm of a community that's more like family. Our mission? To help you unleash your full potential, one energizing workout at a time. Embark on a fitness journey that's not just about exercise; it's a dance of self-discovery and empowerment. Join the movement. Find your rhythm. Redefine fitness. This is SB Fitzz, where every step you take brings you closer to the best version of yourself.

12.known issues:

Limited support for offline mode

Requires manual input for some food items

13.Futures enhancements:

Integration with wearable devices (smartwatches,fitness bands)

Ai chatbot for instant fitness guidance

Voice –assited work out mode

Enhanced gamification features (badges,leader boards)