### fITFLEX

1. Introduction

* Project Tittle: Fitflex
* Team Id: NM2025TMID40323
* Team Leader Name: Nazeema khathun .I ([naseem1272a17@gmail.com](mailto:naseem1272a17@gmail.com))
* Team Members:

-Aruna.A ([arunaachu1517@gmail.com](mailto:arunaachu1517@gmail.com))

-Jennifer Durand .D (nasreenafreen.[10@gmail.com](mailto:10@gmail.com))

-Abirami . H ([abi775733@gmail.com](mailto:abi775733@gmail.com))

-Prathigya Jaiswal . D ([prathigyajaiswal04@gmail.com](mailto:prathigyajaiswal04@gmail.com))

1. Project overview

* Purpose:

1. To get a structured fitness plan rather than random workouts.
2. To save time (you know what to do each day).
3. To monitor both exercise and diet in one place.
4. To adapt plans to your ability and progress (so you improve over time).
5. To enable people to workout from home or without fancy equipment.
6. To maintain motivation via tracking and visible progress

* Features:

1. Personalized Fitness Plans

Goal-based workout plans tailored to your fitness level (beginner, intermediate, advanced) and your objectives (weight loss, muscle gain, toning, etc.).

Ability to select workouts by type (strength, HIIT, abs, cardio, yoga, Pilates) and by difficulty, duration.

2. Flexible & Convenient Workouts

Home workouts with no equipment needed, for users who can’t access gym/gear.

Focused body part workouts (arms, legs, back, chest, abs) when you want to target specific zones.

Shorter or quick workouts for busy schedules.

3. Nutrition & Meal Tracking

Calorie counter, macro and micro nutrient tracking.

Meal planner (multi-day plans) that align with your goals.

Food logging, possibly with AI/food scanner (taking pictures to recognize foods) to simplify tracking.

4. Progress Monitoring

Tracking your workouts: reps, intensity, workout-history etc.

Tracking health metrics such as weight, hydration, calories burned etc.

5. Motivation & Reminders

Workout reminders, push notifications to stay on track.

Trending/new workouts to keep the routines fresh / prevent boredom.

6. Freemium + Subscription Model

Basic free version with limited access; premium/subscription unlocks more content, features.

Offers free trial periods.

7. Customisation and Flexibility

Users can filter workouts (by difficulty / duration / type).

Adjust plans as you progress. (The plans adapt based on your performance or feedback.)

8. Hydration Tracking

Water tracker to ensure daily hydration goals.

1. Architecture:

* Frontend:

I .FitFlex Workouts & Fitness App (iOS / Android app)

* + The UI includes personalized workouts, filtering by type/duration/difficulty.
  + Interfaces for tracking nutrition, meal plans, water intake.
  + Notifications / reminders.
  + Clean layout, with mobile UI design suited for fitness, health metrics, fitness plan dashboards etc. (multiple screenshots shown in app store).

2. FitFlex WordPress Theme for Gym / Sports Store

* + Built as a website theme, frontend is responsive (mobile-friendly, retina images) using HTML5 / CSS3.
  + Uses sliders (Slider Revolution, Swiper Slider) for hero banners or galleries.
  + Uses dynamic WordPress features: Elementor page builder for customization of layouts (header, footer), Ajax search, gallery / post layouts etc.

3. Some UI / UX Design Case Studies

* + There are Behance / Dribbble designs showing how the mobile app or web app might look: clean dashboards, fitness plan displays, modern typography and icons.

4.A documentation snippet (for one version of FitFlex) mentions frontend technology stack: React.js, state management (Hooks, Redux), API integrations (RapidAPI), user authentication (Firebase).

* Backend :

From one GitHub project (“RioGuglielmelli/FitFlex”) :

Backend built with Node.js + Express.

Database: Azure Cosmos DB.

Authentication: Azure Active Directory.

API integration: Uses external exercise‐data API (e.g. Wger API) for exercise info.

Deployment / Hosting / Operations: Hosted on Azure; uses services like AKS (Azure Kubernetes Service), App Service, API Management. Containerization with Docker. Monitoring via Application Insights / Azure Monitor.

From other documentation / student projects (“FitFlex: Your Personal Fitness Companion” on Scribd etc.) :

Backend: Node.js.

Endpoints / API: Exposes endpoints to support user login/auth, workout plan retrieval / creation, progress tracking, etc.

Data storage: Storing user profiles, goals, history of workouts, etc.

* Database:

Here’s a hypothetical / likely database schema for a fitness app like FitFlex, based on the observed features and what’s known. You can adapt this depending on relational (SQL) or NoSQL (MongoDB) as preferred.

Database of fitflex

1. Setup Instructions :

* Prerequisites:

1. For Users (prerequisites to use FitFlex app)

If you just want to install and use the app:

📱 Smartphone (Android or iOS).

🌐 Internet connection (needed for syncing data, fetching workout/food databases, AI features).

👤 Account setup (email, Google, Apple login, or Firebase authentication).

🏋️ Basic health data (age, height, weight, gender, goals) — required to personalize workouts.

✅ Free or paid subscription (premium unlocks more workouts, nutrition features).

* Installation Steps :

1. For Users (Mobile App Installation)

On Android (Google Play Store / APK)

1. Open Google Play Store on your phone.

2. Search for “FitFlex: Workouts & Fitness”.

3. Tap Install.

4. Once installed, open the app.

5. Sign up / log in using email, Google, or Apple ID.

6. Enter your fitness details (age, weight, height, goals).

7. Start using workouts, meal plans, and trackers.

On iPhone (App Store)

1. Open the App Store.

2. Search for “FitFlex: Fitness & Workout AI”.

3. Tap Get → Install.

4. Open the app after installation.

5. Create an account / login.

6. Set up your profile (fitness goals, preferences).

7. You’re ready to start!

1. Fonder Structure :

fitflex/

│

├── backend/ # Server-side code

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

│ ├── controllers/ # Business logic (user, workout, meal controllers)

│ ├── models/ # Mongoose models (User, Workout, Food, Plan, etc.)

│ ├── routes/ # Express routes (auth, workouts, nutrition, logs)

│ ├── middleware/ # Auth middleware, error handling

│ ├── utils/ # Helper functions (validators, JWT, API calls)

│ ├── server.js # Entry point for Express app

│ └── package.json # Backend dependencies

│

├── frontend/ # Client-side code (React or React Native)

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

│ ├── src/

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

│ │ ├── components/ # Reusable UI components (Navbar, Button, Card)

│ │ ├── pages/ # Screens (Login, Dashboard, Workouts, Nutrition)

│ │ ├── context/ # React Context API (Auth context, Theme context)

│ │ ├── hooks/ # Custom hooks (useAuth, useWorkout)

│ │ ├── services/ # API calls (axios/fetch wrappers)

│ │ ├── styles/ # CSS / Tailwind / SCSS files

│ │ ├── App.js # Main app component

│ │ └── index.js # Entry point for React app

│ └── package.json # Frontend dependencies

│

├── docs/ # Documentation (API docs, usage guides)

│

├── .env # Environment variables (DB URI, API keys, secrets)

├── .gitignore # Git ignore rules

├── README.md # Project description

└── package.json # Root config (if monorepo setup)

1. Running the Application

* Frontrnd:

1. Prerequisites

Install Node.js (LTS version recommended).

Install npm (comes with Node) or yarn.

Code editor (e.g., VS Code).

Backend server running (if the frontend depends on APIs).

---

2. Navigate to the frontend folder

If your project has a structure like:

fitflex/

backend/

frontend/

Open a terminal and move into the frontend directory:

cd fitflex/frontend

---

3. Install dependencies

Run:

npm install

(or if you use Yarn)

yarn install

This installs React, Axios, Tailwind, and other required packages.

---

4. Set up environment variables

Create a .env file in the frontend/ folder (if not already present). Example:

REACT\_APP\_API\_URL=http://localhost:5000/api

REACT\_APP\_GOOGLE\_KEY=your\_google\_api\_key

⚠️ Make sure this matches your backend URL.

---

5. Run the development server

Start the React frontend:

npm start

or

yarn start

By default, React runs at http://localhost:3000

You should see the FitFlex login screen or dashboard page.

---

6. Build for production (optional)

If you want a production build:

npm run build

This will create an optimized version in the /build folder, ready to be deployed.

* Backend:

1. Prerequisites

Install Node.js (LTS recommended).

Install npm (comes with Node) or yarn.

Install MongoDB (local or use MongoDB Atlas cloud).

Code editor (VS Code preferred).

---

2. Navigate to the backend folder

If your project has:

fitflex/

backend/

frontend/

Go into backend:

cd fitflex/backend

---

3. Install dependencies

Run:

npm install

(or if using Yarn)

yarn install

This installs Express, Mongoose, CORS, JWT, dotenv, etc.

---

4. Set up environment variables

Create a .env file inside backend/ folder. Example:

PORT=5000

MONGO\_URI=mongodb://localhost:27017/fitflex # or Atlas connection string

JWT\_SECRET=your\_jwt\_secret

NODE\_ENV=development

👉 If using MongoDB Atlas, replace MONGO\_URI with the connection string from Atlas.

---

5. Start MongoDB

If local MongoDB:

mongod

If MongoDB Atlas: just ensure your cluster is active.

---

6. Run the backend server

Start the server:

npm start

or (if set with Nodemon for auto-restart during dev)

npm run dev

✅ By default, it runs at http://localhost:5000

---

7. Test the API

Open Postman or browser and hit:

http://localhost:5000/api/users → for user routes

http://localhost:5000/api/workouts → for workout routes

http://localhost:5000/api/nutrition → for food/nutrition

---

8. Common Issues

MongoDB connection error → Check MONGO\_URI in .env.

CORS error (when frontend connects) → Add:

const cors = require('cors');

app.use(cors());

Port already in use → Change PORT in .env (e.g., 5001).

* Access :

1. Backend Setup for Access

Ensure your backend is already running (npm run dev in /backend).

Routes for access usually exist in:

backend/routes/authRoutes.js

backend/controllers/authController.js

Typical routes:

POST /api/auth/register → create account

POST /api/auth/login → login and get JWT token

GET /api/auth/profile → access protected user profile

---

2. Frontend Setup for Access

In /frontend/src/pages/ you’ll have pages like:

Login.js

Register.js

In /frontend/src/services/ an API service like:

import axios from "axios";

const API\_URL = process.env.REACT\_APP\_API\_URL + "/auth/";

export const login = async (email, password) => {

const res = await axios.post(API\_URL + "login", { email, password });

localStorage.setItem("token", res.data.token);

return res.data;

};

export const register = async (userData) => {

const res = await axios.post(API\_URL + "register", userData);

return res.data;

};

When you run the frontend (npm start in /frontend), you can access login/register forms.

---

3. Access Flow (How it Runs)

1. User opens frontend → http://localhost:3000/login.

2. User enters email & password.

3. Frontend sends request to backend /api/auth/login.

4. Backend validates user → generates JWT token.

5. Frontend saves token (usually in localStorage or cookies).

6. Now user can access protected routes like workouts, nutrition, progress.

---

4. Testing Access

Use Postman to manually test:

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

Content-Type: application/json

{

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

"password": "123456"

}

Should return:

{

"token": "jwt\_token\_here",

"user": { "id": "123", "name": "John Doe" }

}

1. API Documentation

* User:

1. For Developers (Building the App)

Helps frontend devs connect to backend properly.

Shows how to send and receive data (e.g., login, workouts, meals).

Prevents miscommunication between frontend & backend teams.

Example:

If frontend wants to get workouts → they check API docs:

GET /api/workouts

Headers: Authorization: Bearer <token>

Response: { workouts: [...] }

---

2. For Testing & Debugging

QA/testers can use Postman or Swagger UI to test endpoints.

Example: Test if /api/auth/register really creates a new user.

---

3. For Integration with Third-Party Apps

If FitFlex integrates with fitness devices (Fitbit, Apple Health, etc.) or external nutrition APIs → docs explain how external apps can safely use FitFlex APIs.

---

4. For Security & Access Control

API docs define which endpoints require JWT token (protected routes).

Example: /api/auth/login → no token needed, but /api/workouts → token required.

---

5. For Future Development

When new features (AI coach, diet tracking, progress analytics) are added, API docs are updated.

Ensures scalability and easier onboarding of new developers.

* Projects :

1. Authentication & User Management

Purpose: Handles login, signup, authentication, and user profiles.

Endpoints in docs:

POST /api/auth/register → Register new user

POST /api/auth/login → Authenticate user and return JWT

GET /api/auth/profile → Get user details (protected)

---

2. Workout Management

Purpose: Manages workouts, routines, and training plans.

Endpoints in docs:

GET /api/workouts → Fetch all workouts

POST /api/workouts → Add a new workout

PUT /api/workouts/:id → Update a workout

DELETE /api/workouts/:id → Delete a workout

---

3. Nutrition & Diet Tracking

Purpose: Manage food logs, calorie tracking, and nutrition database.

Endpoints in docs:

GET /api/nutrition → Get food/nutrition database

POST /api/nutrition/log → Log a meal

GET /api/nutrition/logs → Get user’s logged meals

---

4. Progress & Analytics

Purpose: Track user progress (weight, BMI, performance).

Endpoints in docs:

GET /api/progress → Fetch progress logs

POST /api/progress → Add new progress entry

GET /api/progress/stats → Get analytics (charts, insights)

---

5. Admin & Access Control

Purpose: Manage users, roles, and system-wide monitoring.

Endpoints in docs:

GET /api/admin/users → List all users

DELETE /api/admin/users/:id → Remove user

GET /api/admin/stats → Platform usage statistics

---

6. Integration Projects (Optional / Advanced)

Purpose: Integrate with third-party APIs like Fitbit, Apple Health, or food databases.

Endpoints in docs:

GET /api/integrations/fitbit

GET /api/integrations/apple-health

* Chats :

1. Chat Authentication

POST /api/chats/connect

Headers: Authorization: Bearer <token>

🔑 Ensures only registered users can access chat.

---

2. Send Message

POST /api/chats/send

Body: {

"to": "trainerId123",

"message": "What’s my workout plan today?"

}

📤 Used to send text from a user to trainer (or AI bot).

---

3. Fetch Messages

GET /api/chats/:conversationId

Response: {

"messages": [

{ "from": "userId", "text": "Hello", "time": "2025-09-17T12:00Z" },

{ "from": "trainerId", "text": "Hi, let’s start your workout", "time": "2025-09-17T12:01Z" }

]

}

📥 Used to load full chat history.

---

4. Real-Time Messaging (Sockets)

API docs may also explain WebSocket/Socket.IO events:

chat:join → Join a room.

chat:message → Send/receive real-time messages.

chat:typing → Notify typing status.

---

5. Notifications / Reminders

GET /api/chats/notifications

1. Authentication:

1. User Registers

User provides name, email, password, and fitness details.

Password is hashed (using bcrypt).

User data is saved in MongoDB / Firebase DB.

2. User Logs In

User provides email + password.

Backend verifies credentials.

If valid → issues a JWT token.

3. Token Storage

Token is stored on the frontend (localStorage or cookies).

Used in Authorization headers for protected requests.

4. Protected Routes

Example: /api/workouts, /api/progress require valid JWT.

Middleware (authMiddleware.js) checks token before allowing access.

1. User Interface:

1. Authentication Screens

Login Screen → Email, Password, "Forgot Password", "Login with Google/Apple".

Signup Screen → Name, Email, Password, Fitness Goals.

Onboarding → Simple intro screens showing app features.

---

2. Dashboard (Home Screen)

Personalized welcome message (e.g., “Good Morning, John 👋”).

Quick stats → Calories burned, Steps, Workouts completed.

Daily workout plan preview.

Notifications/reminders (like “Drink Water”, “Leg Day workout at 6PM”).

---

3. Workout Module UI

Workout Categories → Cardio, Strength, Yoga, HIIT.

Workout List with thumbnails, duration, difficulty.

Workout Detail Page → Video demo, reps/sets, calories burned.

Start Workout Button with progress tracking timer.

---

4. Nutrition Module UI

Meal Logging Screen → Add Breakfast, Lunch, Dinner, Snacks.

Food Search → Search API (calories, macros).

Meal Detail Page → Nutritional info (protein, carbs, fats).

Daily Calorie Tracker → Circular progress bar showing consumed vs goal.

---

5. Progress & Analytics UI

Graphs/Charts → Weight over time, BMI, calories burnt.

Weekly & Monthly performance summaries.

Achievements → Badges for milestones.

---

6. Chat / Community UI (if enabled)

Chat screen with trainers or AI coach.

Group discussions (fitness communities).

Typing indicators, notifications.

---

7. Profile & Settings UI

User photo, name, email, fitness goal.

Settings → Notifications, Dark/Light theme, Subscription details.

Logout button.

1. Testing:

1. Unit Testing

👉 Tests small, isolated parts of the app.

Backend:

Test functions like password hashing, JWT token generation, BMI calculation.

Example (Jest / Mocha):

test("should hash password correctly", async () => {

const hashed = await hashPassword("123456");

expect(hashed).not.toBe("123456");

});

Frontend:

Test React components (LoginForm renders correctly).

---

2. Integration Testing

👉 Ensures frontend + backend + database work together.

Example:

User registers → data stored in MongoDB.

User logs in → JWT returned and validated.

Tools: Postman, Jest, Supertest.

---

3. API Testing

👉 Tests all API endpoints defined in documentation.

Example:

POST /api/auth/register → returns 201 Created.

POST /api/auth/login → returns JWT.

GET /api/workouts → requires valid token.

Tools: Postman, Newman, Swagger UI.

---

4. UI / Functional Testing

👉 Ensures the user interface works correctly.

Login form validation (empty fields, wrong password).

Dashboard shows correct calories & progress.

Workout starts, pauses, and logs completion.

Tools: Selenium, Cypress, React Testing Library.

---

5. Performance Testing

👉 Checks how FitFlex handles heavy load.

Many users logging workouts at the same time.

Load test API /api/workouts.

Tools: JMeter, Locust.

---

6. Security Testing

👉 Since FitFlex handles user data, security is key.

Test JWT authentication (cannot access protected routes without token).

Test for SQL/NoSQL injection, XSS, CSRF.

Ensure passwords are hashed and not stored as plain text.

---

7. User Acceptance Testing (UAT)

👉 Real users test the app to ensure it meets fitness tracking needs.

Example: A trainer logs workouts for clients and checks if data is correct.

Feedback loop → UI/UX improvements.

---

8. Automation & CI/CD

👉 Automated tests run whenever code is updated.

GitHub Actions / Jenkins used for continuous testing.

Ensures new features don’t break existing ones.

---

1. Screenshots or Demo:

📱 FitFlex Demo (Screens & Flow)

1. Authentication (Access)

Login Screen

Fields → Email, Password.

Buttons → Login, Sign up, Forgot Password.

Social logins → Google / Apple.

Signup Screen

Fields → Name, Email, Password, Fitness Goals.

---

2. Dashboard (Home)

Greeting: “Good Morning, John 👋”.

Quick stats → Calories burned, Steps, Current weight.

Daily workout plan preview.

Notifications/reminders panel.

---

3. Workouts

Workout List Screen

Categories: Cardio, Strength, Yoga, HIIT.

Thumbnails + duration + calories burned.

Workout Detail Screen

Video demo, sets/reps, trainer notes.

“Start Workout” button with live tracker.

---

4. Nutrition

Meal logging cards: Breakfast, Lunch, Dinner, Snacks.

Food search bar (powered by nutrition API).

Circular calorie progress bar (e.g., 1,500 / 2,000 kcal).

---

5. Progress & Analytics

Graphs:

Weight vs Time (line graph).

Calories consumed vs burned (bar chart).

Badges for achievements (e.g., “10 Workouts Completed”).

---

6. Chat / Trainer Support

Chat list → Trainer, AI Coach, Community.

Real-time chat bubbles with text/images.

Notifications for new messages.

---

7. Profile & Settings

Profile photo + details.

Edit goals (e.g., Lose Weight, Build Muscle).

Switch theme (Light/Dark).

Subscription info & logout.

---

🎥 Demo Flow Example (User Journey)

1. User signs up → sets fitness goal.

2. Lands on Dashboard → sees today’s workout.

3. Clicks Workout → follows exercise with video guidance.

4. Logs meals in Nutrition → calorie tracker updates.

5. Checks Progress → sees weight drop graph.

6. Chats with trainer in Chat Module for guidance.

1. Known Issues

⚠️ Known Issues in FitFlex

1. Authentication / Login Issues

Problem: Users sometimes can’t log in or get “Invalid token” errors.

Cause:

JWT expiration or misconfigured secret key.

Email/password mismatch.

Workaround: Re-login or reset password; backend config review.

---

2. Workout / Exercise Loading

Problem: Workout videos or exercises fail to load.

Cause:

API request to exercise database fails.

Network or server timeout.

Workaround: Retry; ensure internet connection; check API keys.

---

3. Nutrition / Meal Logging Errors

Problem: Food search or logging doesn’t update calories.

Cause:

External food database API downtime.

Incorrect frontend-to-backend request format.

Workaround: Manual logging; check API request payload.

---

4. Progress Tracking Glitches

Problem: Charts do not update in real-time.

Cause:

Backend response delay.

Frontend state management issue (React context or Redux).

Workaround: Refresh dashboard; backend optimization.

1. Future Enhancements

🚀 Future Enhancements in FitFlex

1. AI-Powered Personalized Workouts

Use AI to create adaptive workout plans based on user goals, fitness level, and progress.

Suggest new exercises if user performance improves or plateaus.

Integrate computer vision for posture correction using the device camera.

---

2. Nutrition & Meal Planning

AI-based meal recommendations based on calorie target, macronutrients, and allergies.

Integration with grocery delivery APIs to automatically create shopping lists.

Barcode scanning for logging packaged foods.

---

3. Wearable / Device Integration

Connect with Fitbit, Apple Watch, Garmin, or Google Fit.

Real-time syncing of steps, heart rate, calories, and sleep data.

Use wearable data to adjust workout intensity automatically.

---

4. Gamification & Social Features

Add achievement badges, leaderboards, and challenges to increase motivation.

Create a community feed for users to share progress, workouts, or meal tips.

Trainer-led group workouts or competitions.

---

5. Advanced Analytics & Progress Tracking

Predictive insights → predict weight loss/gain trends.

Heatmaps for body areas trained most/least.

Personalized weekly or monthly reports with charts and recommendations.

---

6. Enhanced Chat / Support

AI-powered fitness coach chatbot for instant guidance.

Video call integration with personal trainers.

Multi-language support for global users.

---

7. Offline Mode

Allow workout videos and meal plans offline.

Sync progress once the device is online.

---

8. Improved UI/UX

Dark mode and custom themes.

Voice commands for hands-free workout navigation.

Customizable dashboard widgets.

---

9. Subscription & Payment Enhancements

Tiered subscription plans with extra features.

Integration with more payment gateways.

Promo codes and referral programs.

---

10. Security & Privacy Enhancements

End-to-end encryption for chat messages and personal data.

Two-factor authentication (2FA).

GDPR/CCPA compliance for global users.