Fitflex: your personal fitness companian

1.Introduction:

- **Project Title:** FitFlex:Your Personal Fitness Companion
- **Team ID :** NM2025TMID46892

Team leader: Kaviya M kaviyakaviya5447@gmail.com

S.NO	TEAM MEMBERS	MAIL ID
1.	Nabeesha A	ansarabulwahab3344@gmail.com
2.	Nivetha S	rajapan715@gmail.com
3.	Parameshwari S	parameshwari7221@gmail.com
4.	Preethi M	shobisuha1607@gmail.com

2. Project Overview:

- **Purpose:**

To develop FitFlex, an innovative fitness platform that offers users flexible workout plans, real-time progress tracking, and personalized coaching, all tailored to individual fitness levels, goals, and schedules.

Goals:

-Deliver a user-friendly mobile app (iOS & Android)

- Provide AI-driven personalized workout and nutrition plans

-Enable real-time fitness tracking and analytics

-Integrate with wearables (Apple Watch, Fitbit, etc.)

-Build a community space for motivation and accountability

Target Audience:

- Busy professionals

- Fitness enthusiasts

- Beginners looking for guided, flexible routines

- Remote workers and home-gym users

1. Personalized Workout Plans

- AI-generated based on user goals (weight loss, muscle gain, flexibility, etc.)
- Adjustable duration, intensity, and equipment availability
- Daily/weekly scheduling

2. Nutrition & Meal Planning

- Custom meal suggestions based on dietary preferences
- Calorie and macro tracking
- Grocery list generator

3. Real-Time Progress Tracking

- Weight, reps, sets, and performance logging
- Body measurements and progress photos
- Integration with Apple Health, Fitbit, Garmin, etc.

4. Virtual Coaching

- AI coach for guidance and adjustments
- Option to book live sessions with certified trainers
- Voice-guided workouts

5. On-Demand Workout Library

- HIIT, strength, yoga, cardio, Pilates, etc.
- Filter by time, difficulty, body focus
- Video tutorials with form cues

6. Community & Challenges

-Group challenges, leaderboards, and badges

- Social feed for sharing progress and tips

- Accountability partners and team goals

7. Flex Mode

- Automatically adapts your workout if you miss a day

- Smart reshuffling of your weekly plan

- Notifications & motivation boosts

3.*Architecture*

- **Frontend:**

1. Platforms:

- Mobile App (React Native / Flutter for cross-platform)

- Web App (React.js / Angular / Vue.js)

1. Onboarding Screens:

- User sign-up/login (email, Google, Apple, etc.)

- Initial fitness goal selection (weight loss, muscle gain, etc.)

- Basic info input (age, height, weight, preferences)

2. Dashboard (Home):

- Daily workout summary

- Progress tracker (calories, steps, completed workouts)

- Motivational quote or tip of the day

- Quick access to "Start Workout", "Meal Plan", etc

3.Workout Player:

- Video demo, timer, and exercise instructions
- Pause, skip, next controls
- Real-time feedback (if AI coaching is involved)
- Progress bar for workout completion

4. Workout Library:

- Filter by type, duration, difficulty, equipment
- Search bar and categories
- Save favorites or add to routine

5. Meal/Nutrition Planner:

- Daily meal suggestions
- Macro and calorie breakdown
- Add/substitute meals

6. Profile Page:

- User stats and achievements
- Goal settings- Subscription plan & preferences

7. Community Section:

- Feed with user posts, likes, comments
- Join challenges or teams
- Leaderboards

8. Notifications Panel:

- Reminders for workouts, meals, hydration

- Progress alerts and achievements

9. Settings Page:

- Account settings
- App theme (light/dark mode)
- Privacy and notification settings

React.js with Bootstrap and Material UI

- **Backend:**

1. Tech Stack (suggested):

- Language: Node.js / Python (FastAPI or Django)
- Database: PostgreSQL (relational) + MongoDB (if needed for flexible data)
- Authentication: Firebase Auth / JWT
- Cloud Services: AWS / Google Cloud / Azure
- APIs: RESTful or GraphQL
- Real-time: WebSockets (for live sessions / chat)

Core Backend Components:

1. User Management:

- Sign-up/login with OAuth (Google, Apple, etc.)
- Role-based access (user, trainer, admin)
- Profile data (goals, preferences, history)

2. Workout Management:

- Store workouts, exercises, sets/reps, categories
- Schedule planner per user- AI-generated or trainer

-created plans

3. Meal & Nutrition System:

- Meal database with macros and ingredients
- Custom meal plans per user
- Grocery list generation

4. Progress Tracking & Analytics:

- Store metrics: weight, photos, completed workouts
- Graphs and dashboards for visual analytics
- Achievements and goal tracking

5. AI/Recommendation Engine (Optional):

- Suggests workouts based on past activity
- Adapts plans using user feedback or inactivity
- Nutrition tweaks based on goals

6. Notification System:

- Push notifications (via Firebase or OneSignal)
- Email/SMS reminders and alerts

7. Community & Challenges:

- Post, comment, like, follow features
- Challenge creation, leaderboard management
- Group workout tracking

8. Media Handling:

- Upload & stream videos (workout demos)
- Profile images, progress photos
- CDN for performance (e.g., Cloudflare)

9. Admin Panel (Backend Dashboard):

- User and content moderation
- Analytics and usage reports
-Manage trainers, workouts, and challenges

10. Integrations:

- APIs for wearable sync (Apple Health, Fitbit, etc.)
- Payment gateways (Stripe, Razorpay, etc.)

Node.js and Express.js managing server logic and API endpoints

- **Database:**

1. Users

- user_id (PK)
- name
- email (unique)
- password_hash- age, gender, height, weight
- goal (e.g. lose weight, build muscle)
- created_at, last_login

2. Workouts

- workout_id (PK)
- title
- category (e.g. HIIT, Yoga)
- difficulty
- duration
- created_by (FK → users or admin)
- is_public (bool)

3. Exercises

- exercise_id (PK)
- name
- description
- video_url- equipment_required

4. Workout_Exercises

(many-to-many: connects workouts to exercises)

- id (PK)
- workout_id (FK)
- exercise_id (FK)
- sets, reps, rest_time

5. User_Workout_Logs- log_id (PK)

- user_id (FK)
- workout_id (FK)
- completed_at
- duration
- calories_burned

6. Meal_Plans

- feedback
- meal_id (PK)
- name
- meal_type (breakfast/lunch/dinner/snack)
- calories, protein, carbs, fat
- ingredients
- created_by (admin/nutritionist)

7. User_Meal_Logs

- log_id (PK)
- user_id (FK)
- meal_id (FK)
- logged_at
- quantity

8. Progress_Tracking

- entry_id (PK)
- user_id (FK)
- date- weight
- waist, chest, biceps (optional)
- progress_photo_url

9. Challenges

- challenge_id (PK)

- name

- description

- start_date, end_date

10. User_Challenge

- id (PK)

- user_id (FK)

- challenge_id (FK)

- progress, statuds

4. Setup Instructions

- **Prerequisites:**

1. Business Prerequisites

- Clear Vision & Goals – Define what FitFlex offers (e.g., flexibility, personalization, hybrid coaching).

- Target Audience – Identify user segments (e.g., beginners, busy professionals, home workout users).

- Monetization Plan – Subscriptions, freemium model, in-app purchases, or ads.

- Legal & Compliance – Terms of service, privacy policy, GDPR compliance, health disclaimers.

2. Technical Prerequisites

- Tech Stack Selection

- Frontend: React Native / Flutter (mobile), React.js (web)

- Backend: Node.js / Django / FastAPI

- Database: PostgreSQL, MongoDB (if needed

- Cloud & Hosting

- AWS / Google Cloud / Azure

- Firebase for auth, push notifications (optional)

- APIs & Integrations

- Fitness wearables (Apple Health, Fitbit, etc.)

- Payment gateway (Stripe, Razorpay)

- CDN for media (e.g., Cloudflare)

- Development Tools- GitHub / GitLab for version control

- CI/CD pipelines

- Postman for API testing

3. Design & Content

- UI/UX Design Tools– Figma / Adobe XD for prototyping

- Workout & Meal Content – Curated plans, videos, articles

- Branding – Logo, color palette, typography, tone

4. Team Requirements

- Product Manager

- UI/UX Designer

- Frontend & Backend Developers

- Fitness/Nutrition Experts (for content)

- QA/Testers

- Marketing & Growth Lead

5. MVP Planning

- Decide core features for launch (e.g., workout plans, tracking, basic dashboard

• Installation Steps:

- Node.js (for frontend or backend with Express/React)

- Python (if backend uses Django or FastAPI)
- PostgreSQL or MongoDB
- Git
- npm / yarn (for JS projects)
- Virtualenv (for Python projects)

2. Clone the Repository

Bash

git clone https://github.com/your-org/fitflex.git

cd fitflex

3. Frontend Setup

Bash

cd frontend

npm install # or yarn

npm start # Runs on localhost:3000

bash npm install

4. Backend Setup

For Node.js:

bash

cd backend

npm install npm run dev # Runs backend server (e.g., on port 5000)

For Django:

bash

cd backend

python -m venv venv

source venv/bin/activate

pip install -r requirements.txt

python manage.py migrate

python manage.py runserver

5. Database Setup

- Create a PostgreSQL DB (e.g., fitflex_db)

- Update .env file with DB credentials

Env

DB_NAME=fitflex_db

DB_USER=youruser

DB_PASS=yourpass

6. Environment Variable

Set up a .env file in both frontend and backend folders (API URLs, secrets, keys)

7. Run the App

- Frontend: localhost:3000

- Backend API: localhost:5000 (or 8000 for Django)

- Mobile App: via Android/iOS emulator or Expo

8. Optional: Docker Setup

If using Docker:

bash docker-compose up –build

Clone the repository git clone

Install client dependencies cd

client npm install

Install server dependencies cd

/server npm install

5. Folder Structure

```
FitFlex/
├── frontend/          #Mobile or web client
│ ├── src/
│ │ ├── assets/              # Images, fonts, icons
│ │ ├── components/    # Reusable UI components
│ │ ├── screens/       # Pages (Home, Workout, Profile, etc.)
│ │ ├── navigation/     # Stack/tab navigation (React Navigation)
│ │ ├── services/      # API calls (axios, fetch)
│ │ ├── context/       # Auth, theme, user context
│ │ ├── utils/          #Helpers,constants
│ │ └── App.js
│ └── package.json
├── database/          #SQL scripts,seeders,migrations
│ ├── schema.sql
│ ├── seed.js
│ └── migrations/
├── .env                  #Environment variables
├── .gitignore
├── README.md
└── docker-compose.yml          # Optional: containerized setup
```

6. 1. Clone the Project Running the Application

7. bash

git clone https://github.com/your-org/fitflex.git

cd fitflex

2. Start the Backend (Node.js Example)

Bash

cd

npm install Create .env file with DB, PORT, etc.

npm run dev # or node app.js

Runs backend server at http://localhost:5000

3. Start the Frontend (React Native)

bash

cd ../frontend

npm install npx expo start # or react-native run-android / run-ios

Use Expo Go or emulator to view the app.

4. Setup Database (PostgreSQL Example)

- Create DB: fitflex_db

- Update .env in backend:

env

DB_HOST=localhost

DB_PORT=5432

DB_USER=your_user

DB_PASS=your_pass

DB_NAME=fitflex_db

- Run migrations/seed (if available)

5. Environment Setup

Both frontend and backend should have their own .env files for API URLs, keys, etc.

6. Optional Docker Setup

Bash

docker-compose up --build

8. API Documentation

1. Auth APIs

POST /auth/register

- Create a new user

- Body:

{ "name": "John", "email": "john@example.com", "password": "123456" }

- Response:

{ "token": "JWT_TOKEN", "user": { ... } }

POST /auth/login

- Log in user

- Body:

{ "email": "john@example.com", "password": "123456" }

- Response:

{ "token": "JWT_TOKEN", "user": { ... } }

2. User Profile

GET /user/profile

- Headers: Authorization: Bearer <token>

- Returns user details

PUT /user/profile

- Update user info/goals

- Body:

{ "weight": 70, "goal": "muscle_gain" }

3. Workouts

GET /workouts

- List all workouts

- Query: ?category=HIIT&difficulty=easy

GET /workouts/:id

- Get workout by ID

POST /workouts/log

- Log a completed workout

- Body:

{ "workout_id": 123, "duration": 30, "calories": 250 }

4. Exercises

GET /exercises

- All available exercises

GET /exercises/:id

- Details of a single exercise

5. Meal Plans

GET /meals- Get daily meal suggestions

- Query: ?day=Monday

POST /meals/log

- Log a meal

- Body:

{ "meal_id": 456, "quantity": 1 }

6. Progress Tracking

GET /progress

- View progress entries

POST /progress

- Add a new entry

- Body:

{ "weight": 70, "waist": 30, "photo_url": "..." }

7. Challenges

GET /challenges

- List available challenges

POST /challenges/join

- Join a challenge

- Body: { "challenge_id": 12 }

8. Notifications

GET /notifications

- List recent alerts, reminders

8. Authentication

1. User Registration

- Endpoint: POST /auth/register

- Body:

json

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

- Response:

Json

```
 {
 "token": "JWT_TOKEN",
 "user": { "id": 1, "name": "John", "email": "john@example.com" }
 }
```

2. User Login

- Endpoint: POST /auth/login

- Body:

Json

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

- Response:

Json

```
 {
```

"token": "JWT_TOKEN",

"user": { ... }

3. Token Handling

- Token is stored on the client (e.g., AsyncStorage in React Native).

- Sent in headers with each request:

http

Authorization: Bearer JWT_TOKEN

4. Protected Routes (Backend)

- Middleware verifies the JWT:

Js

const token = req.headers.authorization?.split(" ")[1];

// Verify with jwt.verify(token, SECRET)

- If valid, user proceeds. If not, return 401 Unauthorized.

5. Token Expiry

- JWT typically expires in 1h or 24h.

- Use refresh tokens if long-lived sessions are needed.

6. Logout (Client-Side Only)

Optional Enhancements:

- OAuth login: Google, Apple, Facebook

- Password reset flow: with email verification

- 2FA: via email or SMS codes

10. Testing

• Manual testing during milestones

• Tools: Postman, Chrome Dev Tools

1. Identify Milestone Scope

E.g., Milestone 1: User Auth + Basic Dashboard

2. Test Scenarios (Examples)

A. User Authentication

Register with valid inputs

Register with existing email

Login with valid/invalid credentials

Logout and token invalidation

Password validation rules

B. Dashboard UI

-Welcome message after login

Display correct user name and goals

Navigate to workout and meal sections

3. Test Data

Prepare valid and invalid test inputs (emails, passwords, etc.)

4. Devices & Environments

- Mobile: Android (Pixel), iOS (iPhone SE/14)

- Web: Chrome, Safari, Firefox

- Test on different resolutions

5. Logging & Reporting

- Use a test case sheet or tools like Google Sheets / TestRail
- Include:
- Test case name
- Steps
- Expected result
- Actual result
- Status (Pass/Fail)
- Bug ID (if failed)

6. Bug Tracking

Log issues in:

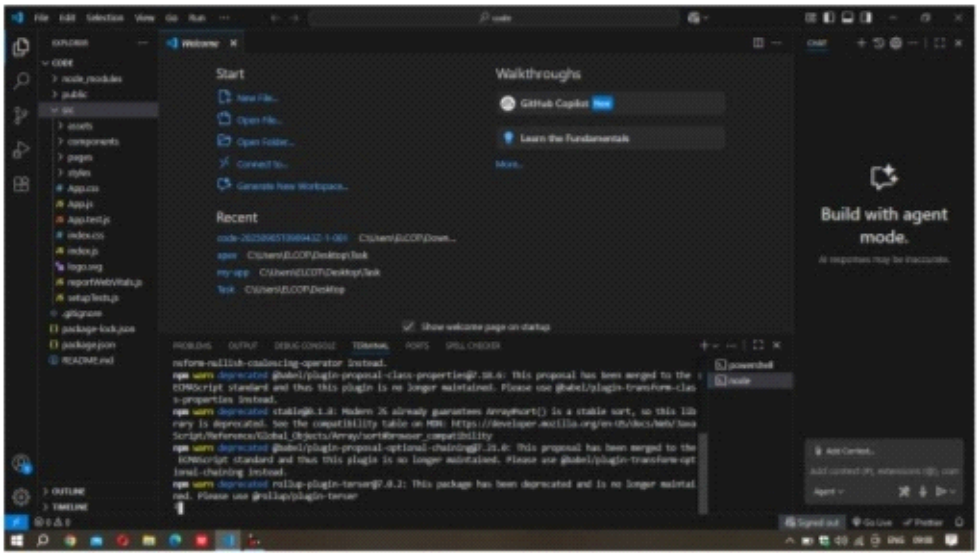
- GitHub Issues
- Jira / Trello board
- Tag them by milestone and priority

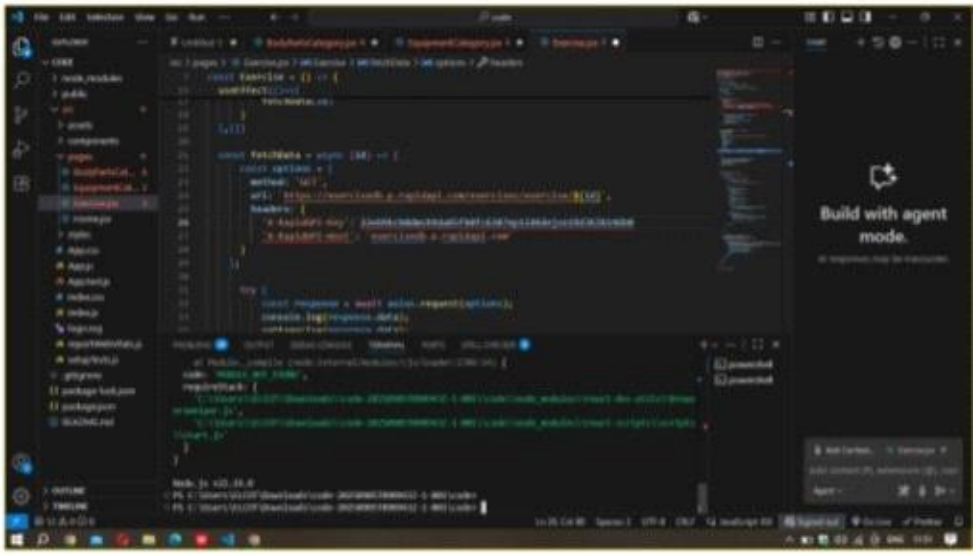
7. Retest & Regression

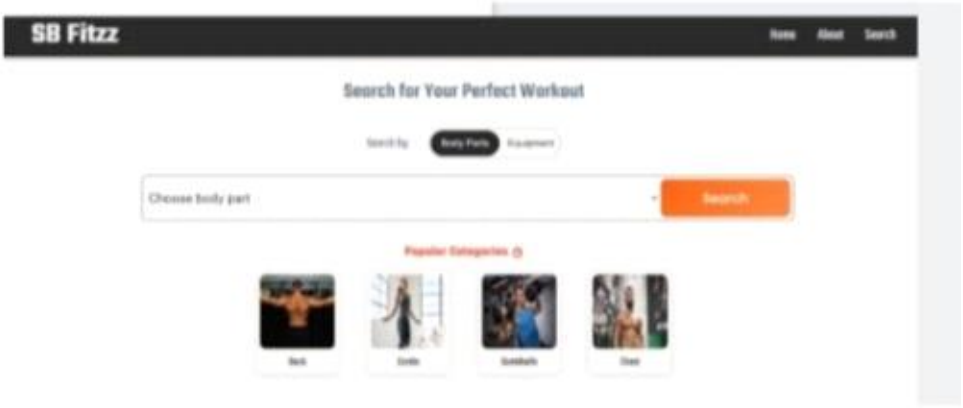
After fixes:

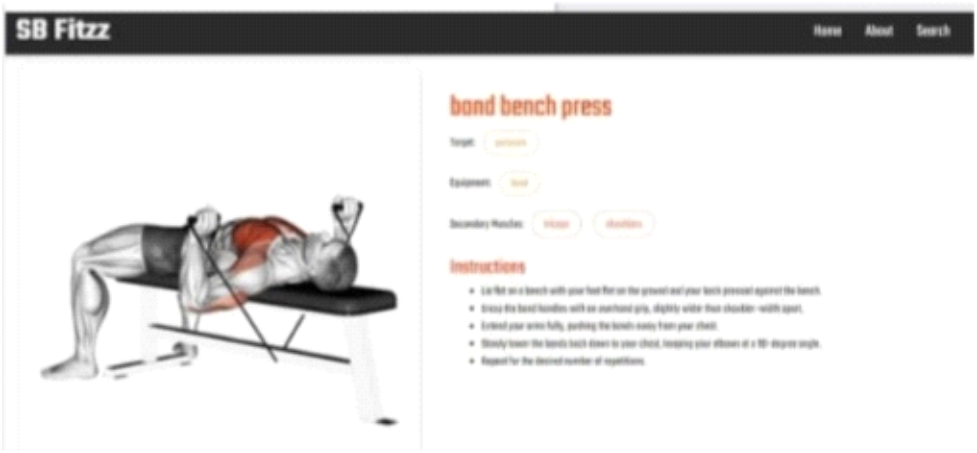
- Retest failed cases
- Run quick regression on previously working modules

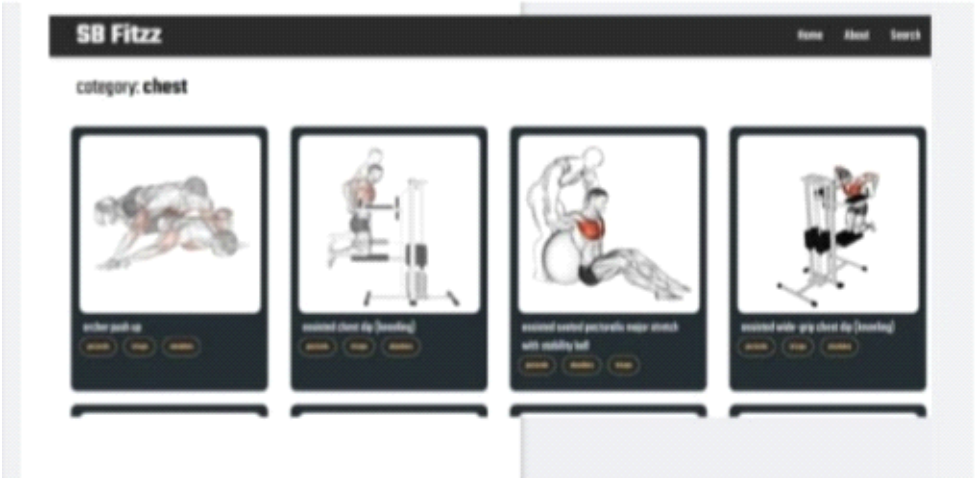
11. Screenshots or Demo


Start
New File...
Open File...
Open Folder...
Connect to...
Generate New Workspace...
Walkthroughs
GitHub Copilot
Learn the Fundamentals
More...
Recent
Build with agent mode.
AI responses may be inaccurate.


Build with agent mode.


SB Fitzz
Home
About
Search
Search for Your Perfect Workout
Search by
Body Parts
Equipment
Choose body part
Search





12.known Issues

1. Authentication

elayed response or timeout during login on slow networks

Token expiration not handled gracefully (user stays on expired session)

Social login (Google/Apple) inconsistent across platforms

2. Workout Player

Workout video freezes on low-end devices

Audio instructions out of sync in some workouts

"Pause" not saving exact progress in longer sessions

3. Meal Plans

Calorie/macro mismatch on meal log summary

Some meals show “undefined” in ingredient list

Meal substitutions not reflected in daily plan

4. Progress Tracking

Image upload for progress photos occasionally fails (esp. on slow networks)

Graphs not updating in real-time after new entry

5. Notifications

Push reminders not triggering on Android 13+

Duplicate notifications in some edge cases