Fitflex: Your Personal Fitness Companion — Development & Technical Documentation

1. Overview

This document provides a complete technical guide for developing Fitflex — a personal fitness companion app. It covers architecture, tech stack, data model, API design, sample code, deployment, CI/CD, testing, and security best practices.

2. Recommended Tech Stack

- Frontend: React (with Tailwind CSS) or React Native for mobile

- Backend: Node.js + Express (or NestJS for larger apps)

- Database: PostgreSQL for relational data, Redis for caching

- Authentication: JWT + OAuth 2.0 (Google/Apple sign-in)

- Storage: AWS S3 or equivalent for media (profile pics, videos)

- DevOps: Docker, Kubernetes (optional), CI (GitHub Actions/GitLab CI), IaC (Terraform)

- Monitoring: Prometheus + Grafana, or SaaS like Datadog

- Testing: Jest (unit), Supertest (API), Playwright/Detox (E2E)

3. Architecture (high-level)

Client (React/React Native) ↔ API Gateway/Backend (Node.js/Express) ↔ Database (Postgres) Optional components: - CDN for static assets - Media storage (S3) - Redis for sessions/caching - Worker queue (BullMQ/RabbitMQ) for background jobs (email, video processing)

4. Data Model (example tables)

```
-- users table
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(100),
  email VARCHAR(150) UNIQUE NOT NULL,
  password_hash VARCHAR(255),
  created_at TIMESTAMP DEFAULT now(),
  updated_at TIMESTAMP DEFAULT now()
);

-- profiles table
CREATE TABLE profiles (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  height_cm INTEGER,
  weight_kg NUMERIC(5,2),
  fitness_goal VARCHAR(255),
  activity_level VARCHAR(50),
  created_at TIMESTAMP DEFAULT now()
);

-- workouts table
CREATE TABLE workouts (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id),
```

```
  title VARCHAR(200),
  duration_minutes INTEGER,
  difficulty VARCHAR(50),
  burned_calories INTEGER,
  created_at TIMESTAMP DEFAULT now()
);
```

5. API Endpoints (REST examples)

```
GET /api/v1/auth/me
POST /api/v1/auth/register
POST /api/v1/auth/login
GET /api/v1/users/:id/profile
PUT /api/v1/users/:id/profile
GET /api/v1/workouts  -- list workouts (with filters)
POST /api/v1/workouts -- create workout (auth required)
GET /api/v1/workouts/:id
POST /api/v1/uploads   -- upload media (returns S3 signed URL)
```

6. Sample Backend Code (Node.js + Express)

```
// index.js (Express)
const express = require('express');
const helmet = require('helmet');
const cors = require('cors');
const authRoutes = require('./routes/auth');
const workoutRoutes = require('./routes/workouts');

const app = express();
app.use(helmet());
app.use(cors());
app.use(express.json());

app.use('/api/v1/auth', authRoutes);
app.use('/api/v1/workouts', workoutRoutes);

const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

7. Sample Frontend Code (React - fetching workouts)

```
// useWorkouts.js (React hook)
import { useEffect, useState } from 'react';

export function useWorkouts() {
  const [workouts, setWorkouts] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    async function fetchWorkouts() {
      const res = await fetch('/api/v1/workouts');
      const data = await res.json();
      setWorkouts(data);
      setLoading(false);
    }
    fetchWorkouts();
  }, []);

  return { workouts, loading };
```

```
}
```

8. Deployment (short checklist)

1. Containerize app using Docker.
2. Push images to registry (Docker Hub / ECR).
3. Use Kubernetes or a PaaS (Heroku, Render, Vercel) to deploy.
4. Setup environment variables and secrets (use Vault or platform secrets).
5. Configure a managed Postgres (AWS RDS / Supabase).
6. Setup CI to run tests and build images on merge to main.

9. CI/CD Example (GitHub Actions - build & test)

```
name: CI

on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Use Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'
      - run: npm ci
      - run: npm test
      - run: npm run build --if-present
```

10. Testing Strategy

- Unit tests for business logic (Jest).
- Integration tests for APIs (Supertest).
- E2E tests for critical flows (signup, create workout) with Playwright.
- Load testing for API endpoints using k6 or Artillery.

11. Security & Privacy

- Store password hashes using bcrypt or argon2.
- Enforce HTTPS and secure cookies.
- Validate and sanitize all incoming data.
- Implement rate limiting and monitoring.
- Be GDPR/PDPA aware: provide data export & deletion endpoints.

12. Next Steps / Roadmap

- MVP: Auth, Profiles, Basic Workouts, Simple Tracking.
- v1.1: Nutrition module, Social features, In-app onboarding.
- v1.2: Video-guided workouts, Personalization engine (ML).
- v2.0: Integrations with wearables, advanced analytics dashboard.

Contact & Credits

Prepared by: Fitflex Development Team

For questions, reach out to dev@fitflex.example (example address)