Deployment Manual

Deployment Manual

1. Introduction

This document provides deployment instructions for the Gymnetics application developed by ByteSquad (Summer 2025). It includes environment setup, backend and frontend deployment, and integration with third-party services (Stripe for payments, AWS S3 for storage, and AI APIs for recommendations).

The goal is to enable any developer or system administrator to deploy the app locally for development or to production servers with minimal friction.

2. System Overview – How the App Works

The Gymnetics application is a full-stack fitness platform with the following core components:

Backend (Java Spring Boot)

Handles authentication (JWT-based).

Manages users, supplements, workout routines.

Integrates with Stripe for payment processing.

Integrates with AWS S3 for file uploads (e.g., profile images, workout plans).

Frontend (React + TypeScript, Vite)

Provides the user interface.

Key modules: Auth (login, register), Cart (shopping), Home, Profile & Settings, Workout programs, Supplements.

Integrates with backend APIs and uses OpenAI/Deepseek services for AI workout recommendations.

Database

SQL schema defined in schema.sql, with migrations for user, supplement, product, and purchase data.

AI Services

openaiService.ts and deepseekService.ts provide personalized workout recommendations and supplement guidance.

User Flow Example

A user registers and logs in (AuthController).

They complete an onboarding quiz.

AI recommends personalized workouts and supplements.

Users can add supplements to their cart and checkout (StripeController).

Purchase details are stored in the database (Purchase entity).

Uploaded content (e.g., workout plans) is stored in AWS S3 (S3Controller).

3. System Requirements

Hardware

CPU: Intel i5/i7 or AMD equivalent, 2.0 GHz+.

RAM: 8 GB minimum, 16 GB recommended.

Storage: 10–20 GB free space.

Software

Backend: JDK 17+, Maven 3.8+.

Frontend: Node.js 18+, NPM 8+.

Database: MySQL/PostgreSQL (configurable).

Other: Git 2.4+, modern browser (Chrome/Firefox/Edge).

4. Deployment Steps

4.1 Clone Repository

git clone https://github.com/htmw/2025Su-ByteSquad.git

cd 2025Su-ByteSquad

4.2 Backend Setup (Spring Boot)

Navigate to backend:

cd backend

Copy configuration template:

cp src/main/resources/application.properties.template src/main/resources/application.properties

Edit application.properties:

Set DB URL, username, password.

Configure AWS S3 credentials.

Configure Stripe API key.

Set JWT secret key.

Run the backend:

mvn spring-boot:run

By default, backend runs on: http://localhost:8080

4.3 Frontend Setup (React + Vite)

Navigate to frontend:

cd ../frontend

Install dependencies:

npm install

Configure environment variables in .env:

VITE_API_URL=http://localhost:8080

VITE_STRIPE_KEY=<your_stripe_key>

VITE_S3_BUCKET=<your_bucket>

Run frontend:

npm run dev

Access UI at: http://localhost:5173

5. Testing & Validation

Manual Testing

Authentication: Register, login, logout, password change.

Supplements: List, search, add to cart.

Cart & Checkout: Add/remove items, complete Stripe test payment.

Profile: Update details, upload image → check S3.

AI Features: Generate workout plan with OpenAI/Deepseek integration.

Automated Testing

Backend unit tests run with:

mvn test

Frontend tests (if configured):

npm test

6. Deployment to Production (Optional)

Backend: Deploy with Docker or on cloud (AWS EC2, Azure, GCP).

Frontend: Build optimized bundle:

npm run build

Deploy via Netlify, Vercel, or S3 static hosting.

Database: Use managed MySQL/Postgres.

Environment: Set all keys via environment variables (never hardcode).

7. Troubleshooting

Port conflicts: Change backend (8080) or frontend (5173) ports.

Stripe payment errors: Ensure correct test keys in .env.

S3 upload failures: Check AWS credentials and bucket policy.

CORS issues: Verify application.properties has correct allowed origins.