AI Music Recommender System
Architecture & User Flow Diagram

Deployed with Docker Compose

December 7, 2025

Abstract

This document presents a comprehensive architectural diagram of a containerized AI Music Recommender System. The system employs a microservices architecture orchestrated by Docker Compose, featuring a React.js frontend, Flask backend with machine learning recommendation engine, and multiple data stores (PostgreSQL, Redis, Qdrant). The diagram visualizes user interactions, API call flows, database queries, and inter-service communication across the deployed containers.

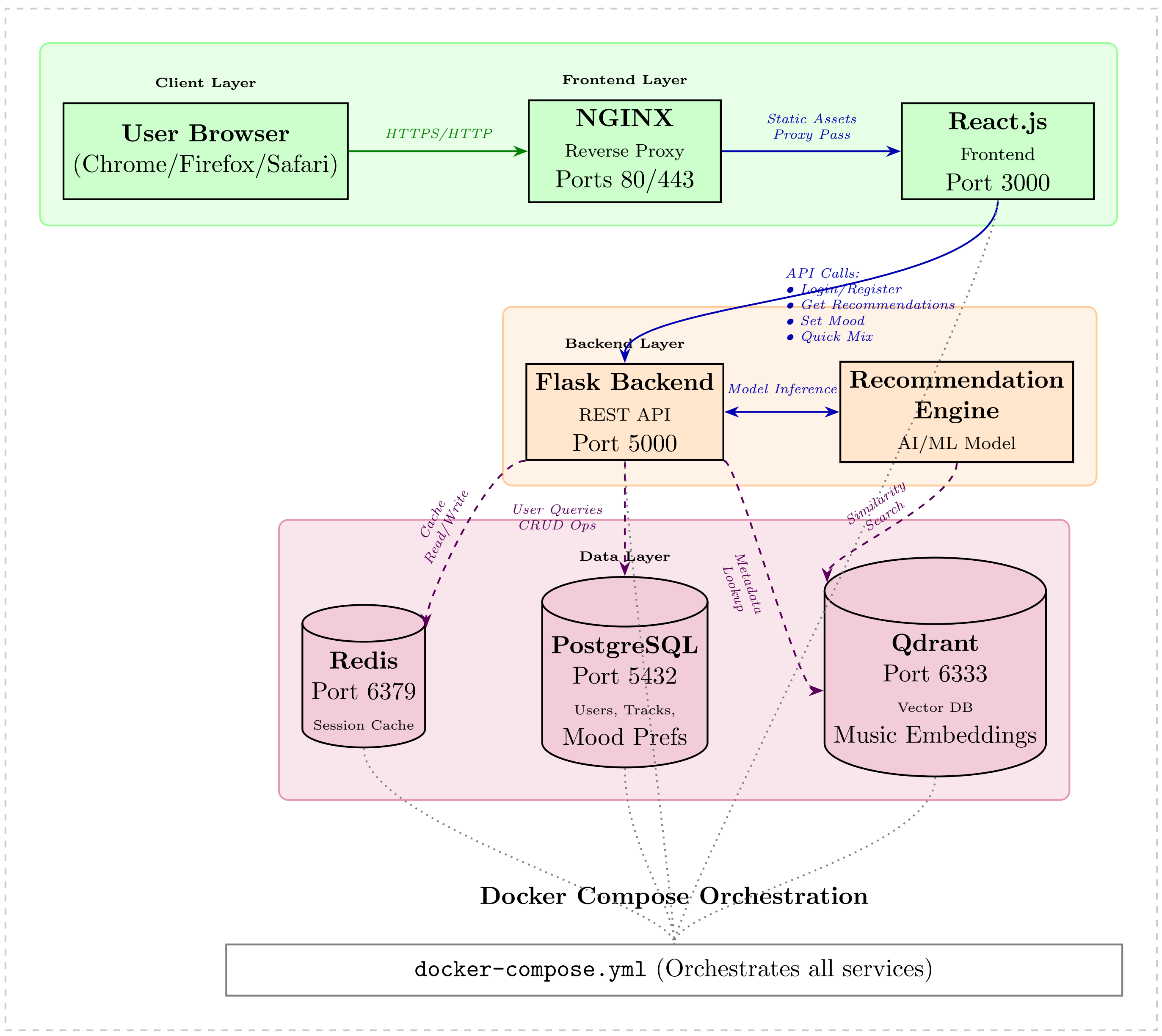
Architecture Summary

The system is designed as a three-tier application:

- **Presentation Layer (Green):** User-facing components including browser, NGINX reverse proxy, and React.js SPA.
- **Application Layer (Orange):** Business logic and AI components with Flask backend and Recommendation Engine.
- **Data Layer (Purple):** Persistent and ephemeral storage systems for user data, music metadata, and vector embeddings.

All services are containerized and managed via Docker Compose, ensuring environment consistency, simplified deployment, and service isolation. Communication follows RESTful API patterns between frontend/backend and dedicated protocols for database interactions.

System Architecture Diagram



Flow Legend:

⟶	REST API Calls
⇢	Database Queries
⟶	User Requests
⋯	Docker Network

Key Interactions:

- **Login:** Auth flow with Redis session
- **Mood Selection:** Updates user preferences in PostgreSQL
- **Recommendations:** Vector search in Qdrant + filtering
- **Quick Mix:** Cached playlist generation

Docker Services Legend:

Service	Image/Container	Port
Reverse Proxy	nginx:alpine	80:80 443:443
Frontend	node:18-alpine + react-app	3000:3000
Backend API	python:3.11-slim + flask	5000:5000
Database	postgres:15-alpine	5432:5432
Cache	redis:7-alpine	6379:6379
Vector DB	qdrant/qdrant	6333:6333

Data Persistence:

- PostgreSQL: `postgres_data`
- Redis: `redis_data`
- Qdrant: `qdrant_storage`
- NGINX SSL: `nginx_certs`

Architecture Notes: All services run in isolated Docker containers connected via a custom bridge network. NGINX handles SSL termination, static file serving, and routes API calls to Flask. The Recommendation Engine uses embedding vectors stored in Qdrant for semantic similarity matching. PostgreSQL stores relational data (users, tracks, preferences), Redis caches session data and frequent queries, and Qdrant performs nearest-neighbor search on music embeddings for recommendations.