Phantom Power RoadMap 0.2 Overview

Topic Select tags

Tag Select tags

Links

Backlinks 👻 Phantom Power

Strategic Recommendation:

- **Frontend** → Vercel (Best-in-class for Next.js apps, Edge/CDN integration)
- **Spring Boot Backend + PostgreSQL DB** → Render.com (single provider simplifies integration)
- **Python Audio Analysis Microservice** → Render.com (tight integration with your backend, simplifies communication and service orchestration)

Phantom Power – Integrated Development Roadmap (6 Weeks MVP)

Week 1: Initial Setup & UI Foundation

- **Frontend (Next.js + Vercel)**
    - Scaffold Next.js, Tailwind CSS, TypeScript
    - Setup basic screens: Landing, Browse, Profiles
- **Backend (Spring Boot + Render)**
    - Scaffold Spring Boot App (Spring Security, JWT, JPA)
    - Database setup (PostgreSQL via Render)
    - Define core entities (`User`, `Profile`, `Project`)
- **AI Microservice (FastAPI + Render)**
    - Basic FastAPI setup, Docker container on Render
    - Proof-of-concept: analyze uploaded audio files locally (tempo, key)

Week 2: Core Integration & Authentication

- **Frontend**
    - Connect frontend to backend REST APIs
    - Authentication: JWT handling, login/signup screens
- **Backend**
    - JWT Authentication endpoints (Spring Security)
    - REST endpoints: User CRUD, Profile CRUD
- **AI Microservice**
    - Deploy basic audio analysis API (`POST /analyze`)
    - Integrate backend ↔ microservice via REST calls

Week 3: Audio Upload & Initial Audio Analysis

- **Frontend**
    - UI for audio uploads (drag & drop, playback)
- **Backend**

    - Handle audio file uploads, store metadata
    - Asynchronous processing: backend → microservice → DB
- **AI Microservice**
    - MVP features: tempo, key, energy/dynamics
    - Store & cache results via backend DB

Week 4: Messaging & Real-Time Communication

- **Frontend**
    - Real-time chat UI (WebSocket-based messaging)
- **Backend**
    - WebSocket server for messaging (Spring Boot + STOMP)
    - Message persistence (PostgreSQL)
- **AI Microservice**
    - Improve accuracy of audio feature extraction (add MFCCs, chroma, basic genre estimation)

Week 5: Matchmaking & Recommendations

- **Frontend**
    - Matchmaking/recommendation UI (suggest musicians/projects)
- **Backend**
    - Build recommendation logic based on audio analysis results
    - Tagging, genres, and matching engine logic
- **AI Microservice**
    - Lightweight ML model (LightGBM) for genre classification (optional CNN exploration)

Week 6: Polish, Testing & Deployment Prep

- **Frontend**
    - UI polish, accessibility improvements, responsive design
    - End-to-end testing (Jest, React Testing Library)
- **Backend**
    - Integration & endpoint tests (JUnit, Postman)

    - Security enhancements, logging & monitoring setup
- **AI Microservice**
    - Optimize API speed and caching
    - Load testing, finalize ML model performance & accuracy

Deployment Topology (Recommended):

```
User Browser
    |
    ├──> Next.js Frontend (Vercel)
            | (REST API, JWT)
            ▼
      Spring Boot Backend (Render)
            | PostgreSQL
            | REST APIs (audio data)
            ▼
     FastAPI Audio Analysis (Render)
            | librosa/madmom ML analysis
            └──▶ Cached Audio Reports
```

Benefits of Recommended Architecture

- **Cost Control**: Predictable monthly costs (no hidden AWS-style expenses).
- **Simpler Networking**: Backend ↔ AI microservice closely integrated on Render.
- **Development Velocity**: Easy deployments, no ops overhead.
- **Scalability**: Clear paths to scaling both backend and microservices.