Phantom Power - Roadmap 0.1

Topic phantomPower

Tag code project planning

Links

Backlinks 👻 Phantom Power

Week 1: Core Features & Infrastructure

Day 1: Planning & Design Kickoff

- Define MVP scope: user profiles, gig postings, messaging, audio matching.
- Design tools: Use Figma (with AI features like Figma Make/Sites) or Framer for interactive prototyping.
- Design trends: Employ bento grid layouts, layering, and subtle 3D animations for immersive feel.
- Setup project repos with monorepo structure: /frontend, /backend, /audio-service.

Day 2: Next.js Setup & Design Integration

- Scaffold with Next.js + TypeScript.
- Install Tailwind CSS for utility-first styling and responsiveness.
- Integrate design components from Figma prototype: colors, buttons, typography.

- Create static screens: landing page, browse feed, profile template.

Day 3: Spring Boot Base & Database

- Generate Spring Boot app with Spring Web, Spring Security (JWT), Spring Data JPA, and PostgreSQL.
- Define entities: User, Profile, Project, Message, Booking.
- Secure endpoints with JWT-based authentication.

Day 4: REST API & Database Integration

- Build REST API for user registration, profile CRUD, and project listings.
- Connect Spring Boot to PostgreSQL using Docker.
- Implement basic tests (JUnit) for endpoints.

Day 5: Basic Messaging & Real-Time Setup

- Add WebSocket support in Spring Boot for real-time chat.
- On Next.js connect via next-websocket or simple socket.io-client with Next API routes.
- UI stub for messaging/chat interface.

Day 6: Python Audio Service Scaffold

- Create Python microservice (FastAPI) with endpoints for audio file upload and analysis.
- Integrate madmom or librosa for tempo, key, genre detection.
- Dockerize and test sample endpoint returning metadata.

Day 7: Feature Integration & Manual Testing

- Connect Next.js to Python service and display audio analysis results on profile pages.
- Begin implementing match scoring based on tags + audio metadata.
- Conduct end-to-end test: post project → upload sample → receive match suggestions.

Week 2: Polish, UX/UI Enhancements & Deployment

Day 8: UI Polishing & Animation

- Add layering, subtle micro-interactions, and 3D cues (e.g., on audio cards) ().
- Use CSS transitions or Framer Motion for smooth effects.
- Ensure responsive and mobile-friendly design.

Day 9: Notifications & Booking Flow

- Implement booking requests: user requests a collaborator.
- Spring Boot manages bookings & statuses.
- Next.js push notifications (via WebSocket) and Python handles calendar reminders.

Day 10: Deployment + CI/CD Setup

- Frontend: Deploy Next.js to Vercel, with automatic preview builds.
- Backend: Deploy Spring Boot to Heroku or AWS Elastic Beanstalk (Docker).
- Audio service: Deploy Python on AWS Lambda (Zappa) or GCP Cloud Run.
- Implement CI/CD via GitHub Actions for tests & deploys.

Day 11: Testing & Quality Assurance

- Add unit tests to frontend (Jest/React Testing Library).
- Add integration tests to backend.
- Run Lighthouse audits and optimize asset loading & performance.

Day 12: Final Refinements & Logging

- Implement basic logging/monitoring (Spring Boot Actuator, Python logs).
- Finalize UI touches, ensure accessibility and polish.
- User documentation: Setup README with architecture diagram, tech choices, endpoints, design notes.

Day 13: Recording & Portfolio Prep

- Capture screencast walkthrough highlighting unique features (audio matching, chat).
- Prepare architecture diagram and design rationale.
- Export design assets from Figma or Framer to include in portfolio.

Day 14: Launch & Reflection

- Announce demo on LinkedIn/GitHub with live demo link and code repo.
- Collect informal feedback from peers.
- Outline next-phase enhancements (e.g. escrow flow, video chat, subscription tiers).

Sprint 1 Data Models

(Comprehensive Models: Phantom Power - Data Models)

Core Models

- **User**: Needed for registration, authentication (JWT), and ownership of all content.
- **Profile**: Displays public user info; enriched with audio analysis, bios, etc.
- **AudioFile**: Even if audio uploading isn't built yet, we need this for AI analysis and profile display.

Media & Content

- **AIAnalysisResult**: Required for matching logic and profile enrichment from Python service.
- **Tag**: Needed for music metadata categorization and future matching algorithms.

Social & Interaction

- **Message**: Required for Day 5 (real-time chat).
- **Match**: Needed to store basic match scores returned from the Python audio service.

Support & System

- **Session**: Implicitly handled by Spring Security + JWT, but worth noting.
- **Role**: Keep minimal support to distinguish admins/devs from standard users (optional but helpful).

Relationship Tables

- **AudioFileTag**: If tag system is integrated into AI or user UI in Week 2.
- **UserFollow**: Optional in Sprint 1, but consider early setup if browsing feeds or follows are mocked in UI.