Phantom Power RoadMap 0.2 Step By Step

Topic phantomPower

Tag code planning project

Links

Backlinks 👻 Phantom Power

Week 1: Core Infrastructure & Setup

Frontend (Next.js)

Day 1:

- Initialize Next.js project (TypeScript, Tailwind CSS, ESLint).
- Setup monorepo structure: `/frontend`, `/backend`, `/audio-service`.

Day 2:

- Configure global styles (color palette, typography).
- Integrate design system from Figma prototype (buttons, cards, inputs).

Day 3:

- Create static landing page structure and initial content placeholders.

- Develop basic navigation components.

Day 4:

- Build "browse feed" static page layout.
- Develop profile page template components.

Day 5:

- Responsive design testing (mobile, tablet, desktop).
- Basic accessibility checks and HTML semantic validation.

Backend (Spring Boot)

Day 1:

- Scaffold Spring Boot project in IntelliJ (Spring Boot 3.5.4, Java 21, Maven).
- Select dependencies: Web, Data JPA, Security, Validation, DevTools, Lombok, PostgreSQL driver, JWT (`jjwt`).

Day 2:

- Configure PostgreSQL connection properties in `application.properties`.
- Initial database setup via Docker Compose or local PostgreSQL instance.

Day 3:

- Define initial JPA entities: User, Profile, Project.
- Lombok integration (`@Getter` , `@Setter` , `@Builder` annotations).

Day 4:

- Create JPA repositories for core entities (User, Profile, Project).
- Verify repository functionality with basic unit tests (JUnit).

Day 5:

- JWT preliminary setup: Utility class and configuration preparation.
- Initial Spring Security setup (password encoding strategy, minimal filter chain).

Audio Microservice (FastAPI)

Day 1:

- Scaffold FastAPI project locally, structure project directories.

Day 2:

- Set up virtual environment, dependency installation (`librosa` , `madmom` , `uvicorn`).

Day 3:

- Implement basic FastAPI endpoint (`/analyze`) skeleton.

Day 4:

- Dockerize FastAPI service for consistent local environment.

Day 5:

- Preliminary audio analysis code: basic tempo and key extraction tests.

Week 2: Authentication, API Development & Integration

Frontend (Next.js)

Day 1:

- Install Axios for HTTP requests.
- Design login and registration screens (Figma → Next.js components).

Day 2:

- Implement login/signup forms with controlled form components.
- Client-side validation & error handling (Yup or Zod schemas).

Day 3:

- Set up authentication context/provider for managing JWT tokens.
- Secure route navigation (middleware for protected routes).

Day 4:

- Integrate API calls to backend endpoints for authentication.
- Test end-to-end authentication flow (registration/login/logout).

Day 5:

- Polish UI/UX for authentication-related screens.
- Unit testing for auth components (Jest/React Testing Library).

Backend (Spring Boot)

Day 1:

- Implement JWT Utility class (generate & validate JWT tokens).
- Create authentication payload classes (LoginRequest, SignupRequest, JwtResponse).

Day 2:

- Develop `AuthController` (login, registration endpoints).
- Secure endpoints with Spring Security configuration.

Day 3:

- Set up `UserDetailsService` implementation for Spring Security.
- Password encoder setup (BCrypt recommended).

Day 4:

- Integrate user registration flow with database persistence.
- Test authentication endpoints with Postman or similar tool.

Day 5:

- Implement token refresh logic endpoint.
- Write integration/unit tests for authentication components.

Audio Microservice (FastAPI)

Day 1:

- Create FastAPI endpoint `/analyze` accepting audio uploads.

Day 2:

- Integrate file upload handling & audio preprocessing (librosa, madmom).

Day 3:

- Perform basic audio analysis (extract tempo, key features).
- Return JSON response with audio metadata.

Day 4:

- Test and debug the `/analyze` endpoint locally.
- Add basic error handling and validations.

Day 5:

- Deploy to Render (initial testing deployment).
- Confirm communication with Spring Boot backend via REST API.

Week 3: Audio Handling, Real-Time Communication & DB Integration

Frontend (Next.js)

Day 1:

- UI for audio upload (Drag & Drop, File Input).

Day 2:

- Integrate audio upload API call (Axios → backend endpoint).

Day 3:

- Add playback/preview functionality for uploaded audio.

Day 4:

- WebSocket client setup (socket.io-client or native WebSocket).

Day 5:

- Implement basic chat UI connected via WebSocket.

Backend (Spring Boot)

Day 1:

- REST controller for audio file uploads.

Day 2:

- Backend file storage & metadata persistence logic.

Day 3:

- Setup Spring WebSocket (STOMP).

Day 4:

- Implement chat message persistence to PostgreSQL.

Day 5:

- Test WebSocket & file upload flows end-to-end.

Audio Microservice (FastAPI)

Day 1:

- Enhance analysis with MFCC and chroma features.

Day 2:

- Add genre tagging logic (basic heuristic or lightweight ML model).

Day 3:

- Async handling & response optimization.

Day 4:

- Implement caching mechanism.

Day 5:

- Confirm stability & performance (local benchmarks).

Week 4: Matchmaking Logic & Initial Recommendations

Frontend (Next.js)

Day 1:

- Develop UI screens for recommended profiles/projects.

Day 2:

- Fetch matched recommendations from backend API.

Day 3:

- Display dynamic user profiles enhanced by audio metadata.

Day 4:

- Polish recommendation and profile UIs.

Day 5:

- Frontend tests for matchmaking components.

Backend (Spring Boot)

Day 1:

- Implement match scoring algorithm based on audio metadata.

Day 2:

- REST endpoint for fetching recommended matches.

Day 3:

- Logic for matching users by genre, tempo, etc.

Day 4:

- Integration tests for recommendation endpoints.

Day 5:

- Performance and correctness checks of matchmaking logic.

Audio Microservice (FastAPI)

Day 1:

- Setup LightGBM/XGBoost model for audio classification.

Day 2:

- Train model on available labeled dataset.

Day 3:

- Integrate trained ML model into `/analyze` endpoint.

Day 4:

- Improve prediction accuracy via iterative training.

Day 5:

- Benchmark and optimize model inference performance.

Week 5: User Interaction, Notifications & Enhancements

Frontend (Next.js)

Day 1:

- Booking request UI (collaboration requests).

Day 2:

- Notification display UI (pop-ups, notification center).

Day 3:

- WebSocket push notifications integration.

Day 4:

- Finalize interaction flow (accept/decline requests).

Day 5:

- Unit & integration testing for booking/notifications.

Backend (Spring Boot)

Day 1:

- Booking endpoint logic (request, accept, decline).

Day 2:

- Notification system via WebSocket (push notifications).

Day 3:

- Store notification history in database.

Day 4:

- Integration testing for booking/notification flows.

Day 5:

- Audit performance, stress-test notification system.

Audio Microservice (FastAPI)

Day 1:

- Enhance audio feature extraction (rhythmic, dynamic features).

Day 2:

- Validate extended feature extraction on diverse audio set.

Day 3:

- Stability checks, response optimization.

Day 4:

- Prepare documentation for API use.

Day 5:

- Conduct end-to-end performance benchmarking.

Week 6: Quality Assurance, Documentation & Final Deployment

Frontend (Next.js)

Day 1:

- Run frontend end-to-end tests, fix issues.

Day 2:

- Accessibility & responsive design verification.

Day 3:

- Lighthouse audits, optimize for web performance.

Day 4:

- Complete user documentation (frontend README).

Day 5:

- Prepare CI/CD scripts for deployment.

Backend (Spring Boot)

Day 1:

- Integration & unit tests verification.

Day 2:

- Security audit and performance tuning.

Day 3:

- Logging/Monitoring setup with Spring Boot Actuator.

Day 4:

- Dockerize backend application for Render deployment.

Day 5:

- Write deployment documentation.

Audio Microservice (FastAPI)

Day 1:

- Finalize unit/integration testing of audio APIs.

Day 2:

- Performance load testing.

Day 3:

- Prepare Dockerfile for Render deployment.

Day 4:

- Logging and error handling refinement.

Day 5:

- Complete deployment documentation.

Launch Preparation

- Deploy Next.js to Vercel.
- Deploy Backend & Audio microservice to Render.
- Announce via GitHub/LinkedIn, gather initial user feedback.

This comprehensive step-by-step guide aligns with the most recent architecture (version 0.8) and focuses on achievable, incremental progress toward your MVP launch. Let me know when you'd like to focus in-depth on a specific day's task!