Plishe Project Documentation

1. Project Overview

1.1 Project Name

Plishe

- Portmanteau of "Playlist" + "Share"
- Phonetically similar to "cliché" for easy recall

1.2 Project Purpose

- Large-scale project for development portfolio
- Build a music playlist-centered social media platform

1.3 Core Concept

A **music playlist-centered social platform** differentiated from existing photo/text-based social media

- Solve playlist sharing issues across different music streaming platforms
- Enable cross-platform playlist conversion and sharing

2. Core Features

2.1 Playlist Creation and Sharing

- Platform-independent playlist creation
- Unified music search system
- Cross-platform playlist conversion

2.2 Streaming Service Integration

- **Integration Method**: Optional (not mandatory)
- **Supported Platforms**: Spotify, Apple Music, YouTube Music, etc.
- **Playback Method**: No direct playback; only playlist creation via platform APIs
- **Copyright Solution**: Music playback only available through user's connected streaming service

2.3 Social Interactions

1. **Collaborative Playlist**
   - Multiple users co-creating playlists
   - Invitation-based collaboration system
2. **Taste Match**
   - User matching based on music preferences
   - Similarity calculation through playlist analysis
3. **Playlist Challenges**
   - Theme-based playlist challenges
   - Community participation events

3. Technical Architecture

3.1 Music Database Structure

On-Demand Approach + Popular Track Pre-loading

- **Initial Setup**: Pre-load TOP 10,000 popular tracks/artists
- **Expansion Method**: Store only tracks searched/added by users
- **Real-time Search**: API calls to streaming services for tracks not in DB
- **Track Matching**: ISRC code and metadata-based identification

3.2 Stored Data

Track Metadata (no actual audio files stored)

- Track title, artist, album
- ISRC code
- Platform-specific IDs (Spotify ID, Apple Music ID, YouTube Music ID, etc.)
- Additional info: genre, release date, etc.

Estimated DB Size

- Initial (10K tracks): ~3.4 MB
- Mid-stage (100K users, 500K tracks): ~460 MB
- Large-scale (1M users, 3M tracks): ~1.5 GB

3.3 Platform

- **Web Application**
- **Mobile App** (iOS/Android)
- Cross-platform development consideration (React + React Native or Flutter)

4. User Experience Design

4.1 Onboarding Process

1. Sign up (social login or email)
2. Music taste profiling (optional, for recommendation algorithm)
3. Platform exploration (recommended playlists, trending challenges)
4. Streaming service integration (later, when needed for playback/export)

4.2 User Profile

- Created playlist gallery
- Follow/follower list
- Music taste summary (preferred genres, artists)
- Challenge participation and activity history
- Badges/statistics

4.3 Feed Structure

Dual Feed System

1. **Main Feed (Discover/Explore)**
   - Algorithm-based recommended playlists
   - Trending challenges
   - Popular playlists
   - Taste Match-based recommendations
   - Category/mood-based browsing
2. **Following Feed (Friends/Following)**
   - Activities from followed users
   - Intimate sharing space

4.4 Core Action Flow

Playlist Creation and Sharing

1. Start creating playlist
2. Search tracks (Plishe unified search: DB + real-time API)
3. Add tracks
4. Set title, description, cover image, tags
5. Set visibility (public/followers only/private)
6. Publish → Feed exposure

5. Community Structure

5.1 Social Structure

- **Individual Profile** based
- **Follow/Follower** system
- **Hashtags** for playlist categorization and discovery
- **Taste Match** for connecting similar taste users

5.2 Group/Community Features

- Excluded from initial version
- Start simple with individual-centered structure
- Keep future expansion possibilities open

6. Business Model

6.1 Monetization Strategy

- **Current Stage**: No monetization consideration
- **Initial Goal**: User acquisition and portfolio building
- **Future Options**: Premium subscription, streaming service partnerships, advertising, etc.

7. Key Differentiators

1. **Cross-Platform Compatibility**: Playlist sharing across different streaming services
2. **Music-Centered Social**: Music playlists as core content, not photos/text
3. **Broad Community + Intimate Sharing**: Balance between discovery and personal sharing
4. **Minimal Entry Barrier**: Playlist creation possible without streaming service integration

8. Development Priorities (To be detailed)

Phase 1: MVP

- Basic playlist creation/sharing
- Integration with 1-2 major streaming services
- Simple profile and follow system

Phase 2: Social Feature Enhancement

- Taste Match
- Collaborative Playlist
- Challenge system

Phase 3: Platform Expansion

- Additional streaming service integrations
- Mobile app optimization
- Advanced recommendation algorithm

9. Technical Considerations

9.1 Database Schema (High-Level)

Tables:

- users: User profiles and authentication
- playlists: Playlist metadata
- tracks: Track metadata with multi-platform IDs
- playlist_tracks: Many-to-many relationship
- follows: User follow relationships
- challenges: Challenge information
- playlist_collaborators: Collaborative playlist permissions

9.2 API Integration Requirements

- Spotify Web API
- Apple Music API
- YouTube Music API
- Amazon Music API
- TIDAL API

9.3 Tech Stack Considerations

Frontend:

- Web: React, Next.js
- Mobile: React Native/Flutter

Backend:

- Node.js/Express
- RESTful API or GraphQL

Database:

- PostgreSQL (relational data)
- Redis (caching)

Infrastructure:

- Cloud hosting (AWS, Google Cloud, or similar)
- CDN for cover images

10. Next Steps

Items requiring detailed design:

- Detailed UI/UX design (wireframes)
- Database schema design
- API endpoint definitions
- Final tech stack decision
- Development timeline
- User flow diagrams
- Component architecture

11. Risk Mitigation

11.1 Technical Risks

- **API Rate Limits**: Implement caching and request optimization
- **Platform API Changes**: Build abstraction layer for easy adaptation
- **Track Matching Accuracy**: Implement fallback mechanisms and manual correction

11.2 UX Risks

- **Search Speed**: Progressive loading and caching strategy
- **Empty State**: Seed initial database with popular content
- **Learning Curve**: Intuitive onboarding and tooltips

Document Version

- Version: 1.0
- Date: January 8, 2026
- Status: Initial Planning Phase