Album & Retro Game Pairing App

Product Requirements Document

Executive Summary

A web application that eliminates choice fatigue by intelligently pairing albums from the user's music collection with retro games from their gaming collection. The app serves as a smart recommendation engine that suggests complementary entertainment experiences while also providing discovery opportunities for expanding collections.

Problem Statement

Music and gaming enthusiasts with large collections face two primary challenges:

1. **Choice Paralysis**: Too many options make it difficult to decide what to listen to or play
2. **Discovery Stagnation**: Difficulty finding new content that aligns with existing preferences

Solution Overview

A simple, elegant pairing interface that:

- Instantly suggests album + retro game combinations from user's existing collections
- Provides intelligent discovery recommendations for collection expansion
- Learns from user preferences to improve suggestions over time

---

Core Features

1. Collection-Based Pairing Engine

Priority: P0

- Single-click pairing generation from user's existing Discogs and Gameye collections
- Smart matching algorithm based on:
  - Mood compatibility (ambient music → exploration games)
  - Era synergy (80s synthwave → retro arcade)
  - Energy levels (high-tempo → action games, mellow → puzzle games)
  - Aesthetic alignment (pixel art games → chiptune music)
  - Session time matching (album length ≈ typical game session)

2. Discovery Mode

Priority: P1

- "Expand Your World" feature offering hybrid pairings:
  - One item from collection + one recommended item
  - Suggestions based on collection analysis and compatibility scoring
- Integration with collection APIs to add recommended items
- Availability and pricing checks for suggested items

3. Learning System

Priority: P1

- Simple thumbs up/down rating system
- Pairing history tracking
- Preference learning to improve future suggestions
- User behavior analysis (most-played genres, preferred session lengths)

4. User Interface

Priority: P0

- Minimalist design focused on reducing decision fatigue
- Primary "Get Pairing" button
- Clean pairing display cards with album art and game screenshots
- Quick rating and "Try Another" options
- Optional mood/preference selectors

Technical Architecture

Data Layer

User Collections

- Sync from Discogs API (albums, artists, genres, release years)
- Sync from Gameye API (games, platforms, genres, release years)
- Local caching with incremental sync capability

Pairing History

- User ratings and feedback
- Session data and usage patterns
- Preference profiles

Recommendation Engine

- Compatibility scoring algorithms
- Machine learning for preference adaptation
- Discovery suggestion logic

API Integration

Discogs API

- Collection synchronization
- Album metadata enrichment
- Related artist discovery
- Marketplace integration for purchase suggestions

Gameye API

- Game collection management
- Metadata and categorization
- Platform and genre information

Backend Services

Express.js Application

- RESTful API endpoints
- Authentication and user management
- Rate limiting and caching
- Background sync processes

Database

- SQLite for development/small scale
- PostgreSQL for production scaling
- Schemas for users, collections, pairings, ratings

Frontend

Technology Stack

- Vanilla JavaScript or lightweight React
- Responsive design for desktop and mobile
- Progressive Web App capabilities
- Offline functionality for cached pairings

User Experience Flow

Primary Flow: Quick Pairing

1. User clicks "Get Pairing"
2. Algorithm selects compatible album + game from collections
3. Display pairing with cover art, brief descriptions, and estimated time
4. User rates pairing (thumbs up/down) or tries another
5. System learns from feedback

Discovery Flow: Collection Expansion

1. User toggles "Discover Mode"
2. System suggests pairing with one new item
3. Display includes "Add to Collection" option with availability info
4. User can add items or rate suggestions
5. New additions influence future recommendations

Preference Customization

1. Optional mood selector (energetic, chill, nostalgic, experimental)
2. Session length preferences
3. Genre weighting adjustments
4. Platform/era filtering

Success Metrics

Engagement Metrics

- Daily active pairings generated

- User return rate and session frequency
- Rating participation rate
- Discovery mode usage

Effectiveness Metrics

- Positive rating percentage
- Time to decision (pairing acceptance)
- Collection growth rate from recommendations
- User satisfaction surveys

Technical Metrics

- API response times
- Sync success rates
- App performance and load times
- Error rates and system stability

Development Phases

Phase 1: MVP Foundation (4-6 weeks)

- Basic API integration with Discogs and Gameye
- Simple pairing algorithm based on genre/era matching
- Minimal UI with core pairing functionality
- Local data storage and user profiles

Phase 2: Intelligence Layer (3-4 weeks)

- Enhanced pairing algorithm with mood and energy matching
- Rating system and basic learning capabilities
- Pairing history and preference tracking
- UI improvements and responsive design

Phase 3: Discovery Features (3-4 weeks)

- Discovery mode implementation
- Collection expansion recommendations

- "Add to Collection" integration
- Advanced filtering and customization options

Phase 4: Polish & Scale (2-3 weeks)

- Performance optimization
- Advanced learning algorithms
- Social features (optional sharing)
- Production deployment and monitoring

Technical Considerations

Scalability

- Design for single-user initially, with multi-user architecture consideration
- Efficient caching strategy for API data
- Batch processing for collection syncs

Privacy & Data

- User collection data remains private
- Local processing of preferences where possible
- Transparent data usage policies

Performance

- Lazy loading of collection data
- Precomputed compatibility scores
- Offline capability for core functionality

Extensibility

- Plugin architecture for additional collection sources
- Configurable pairing algorithms
- API endpoints for potential mobile app

Risk Mitigation

API Dependencies

- Fallback mechanisms for API outages
- Local data caching to reduce dependency
- Rate limiting compliance and monitoring

User Adoption

- Simple onboarding process
- Clear value proposition demonstration
- Progressive feature introduction

Technical Complexity

- Start with simple algorithms, iterate based on feedback
- Modular architecture for easier debugging and updates
- Comprehensive testing strategy

Future Enhancements

Advanced Features

- Social sharing of favorite pairings
- Community-driven pairing suggestions
- Integration with streaming services for immediate playback
- Mobile app with notification-based pairing suggestions

AI/ML Improvements

- Natural language processing for mood detection
- Computer vision for game aesthetic analysis
- Collaborative filtering based on similar user preferences
- Seasonal and contextual pairing adjustments

Ecosystem Integration

- Support for additional collection platforms
- Integration with media players and game launchers
- Calendar integration for scheduled pairing sessions
- Smart home integration for ambient pairing experiences