Juzzt - Technical Specifications Document

1. Project Overview

Name: Juzzt
Description: Juzzt is an online store for jazz records featuring AI-powered recommendations, mood-based playlists, and a personalized "Record of the Day" feature. The backend is built using **Spring Boot**, with **PostgreSQL** as the primary database and **Redis** for caching. The system will later integrate **Kafka** for event-driven processes and deploy using **Docker**.

2. Tech Stack

Backend

- Spring Boot (Java)
- Spring Data JPA (PostgreSQL)
- Spring Cache (Redis)
- Spring Security (for authentication, future implementation)
- Kafka (for real-time event streaming, future implementation)
- Docker (for containerization)

Database

- PostgreSQL (Primary Data Storage)
- Redis (Caching for fast recommendations and queries)

AI & Recommendations

- TensorFlow/Keras (Collaborative Filtering Model)
- Surprise Library (Lightweight ML for user-item recommendations)
- Spotify API (Mood-Based Playlists using user history)

DevOps & Deployment

- Docker & Docker Compose
- CI/CD Pipeline (GitHub Actions or Jenkins)
- Cloud Hosting (AWS/GCP/Azure, TBD)

---

3. System Architecture

High-Level Architecture

- **User requests** go through the API Gateway (Spring Boot Controllers).
- **Business logic** is handled by Service Layer.
- **Data is fetched** from PostgreSQL and cached in Redis.
- **AI-powered recommendations** use ML models to suggest records.
- **Kafka (future implementation)** for real-time notifications and events.

Backend Architecture Layers

1. **Controller Layer**: Handles HTTP requests (e.g., /records/recommendations).
2. **Service Layer**: Implements business logic (e.g., fetching recommendations, filtering by mood).
3. **Repository Layer**: Communicates with PostgreSQL via JPA.
4. **Cache Layer (Redis)**: Caches AI recommendations & frequently accessed records.

---

4. Database Schema

Entities & Relationships

- **User** *(id, name, email, password, preferences, purchase history)*
- **Record** *(id, title, artist, genre, mood_tags, rating, price, stock)*
- **Recommendation** *(id, user_id, record_id, score, type [AI-based, mood-based])*
- **Order** *(id, user_id, order_date, total_price, status)*
- **OrderItem** *(id, order_id, record_id, quantity, price)*

---

5. Features & APIs

Core Features

✅ **AI-Powered Personalized Recommendations** (User preferences & purchase history)
✅ **Mood-Based Playlist Generation** (Users describe a mood, system suggests records)
✅ **Record of the Day** (Daily unique recommendations based on trending & user history)
✅ **Standard E-commerce Functionalities** (Cart, checkout, order history)

API Endpoints (Planned)

User Management

- POST /users/register → Register new user
- POST /users/login → Authenticate user
- GET /users/{id}/recommendations → Get AI-powered recommendations

Record Store

- GET /records/ → List all records
- GET /records/{id} → Get record details
- GET /records/mood/{mood_tag} → Get records based on mood
- GET /records/daily → Get "Record of the Day"

Orders

- POST /orders/ → Place an order
- GET /orders/{id} → Get order details

AI & Caching

- GET /recommendations/user/{id} → Fetch recommendations from AI model
- GET /cache/clear → Clear Redis cache

---

6. Conclusion

Juzzt aims to be more than just a regular online record store by integrating AI-driven recommendations and mood-based music discovery. The project follows a structured approach with well-defined backend architecture, caching strategies, and a long-term vision to integrate Kafka for real-time features and blockchain for record authentication. 🚀