**Detailed Design Document  
Orel Emuna – 208947036 & Yuval Greenberg - 211994082**

**Introduction**

The Guitar Teacher Recommendation System is designed to help users enhance their guitar learning by recommending songs based on chords they know and their preferred difficulty level. This document outlines the detailed design of the system, including its architecture, component interactions, and technical specifications.

**System Overview**

**System Objectives:**

* Recommend songs based on user-selected chords and preferences.
* Provide a scalable, efficient platform for guitar learners.
* Optimize recommendations using caching and advanced filtering algorithms.

**Components:**

* **User Interface (UI):** Developed using React Native, it displays song recommendations and progression dashboards, collects user input.
* **Backend:** Built with Java (e.g., Spring Boot) to process API requests, recommendation logic, and database interactions, ensuring compatibility with the React Native frontend and leveraging the team's familiarity with Java.
* **Database:** PostgreSQL stores user profiles, chord data, and song metadata.

**Design Considerations**

**Assumptions:**

* The system will primarily be used on iPhone and Android devices, leveraging a React Native frontend for a seamless mobile experience.
* Users will have a basic understanding of guitar chords and difficulty levels.
* A maximum of 500 concurrent users.

**Constraints:**

* The current system is designed to support both iPhone and Android mobile apps using React Native for the frontend, ensuring a seamless experience across platforms. While React Native allows for shared code between platforms, this approach may limit access to native device features compared to fully native solutions
* Focused on foundational features for proof-of-concept purposes.

**System Architecture**

**Architecture Diagram:**

A diagram of a computer network

Description automatically generated

**Frontend Layer (Mobile App)**

The frontend layer hosts the Guitar Teacher Recommendation System as a mobile application for iPhone and Android devices. It contains:

* **Song Recommendation View:**
  + Displays a list of recommended songs with chord diagrams and difficulty levels.
  + Displays suggested chords to learn next based on user`s level and previus known chords.
* **Progress Tracker:**
  + Visualizes user progress and mastery over time.
  + Includes a dashboard for tracking practiced songs and chords.
* **Playing Interface:**
  + Real-time chords interface for playing chosen song while using spotify`s API.
* **User`s song review:**
  + After playing a song the user can rate the lesson in order to update user`s profile and improve future recommendations.
* **Sign up:**
  + Upon registeration the user will provide his knoladge and level in order to update user`s profile and improve future recommendations.

**Backend Layer (Backend Server)**

* **Recommendation Engine:**
  + Executes the pruning algorithm and additional recommendation logic.
  + Queries the database for songs.
  + Recommendations based on users playlists on spotify.
* **User Manager:**
  + Handles user`s profile updates, learning history and goals based on his reviews and progression.
* **Song Manager:**
  + Fetches songs data from external sources and extracts useful information and updates Songs DB.

**Data-Base Layer**

* **Song:**
  + Stores useful and calculated features and metadata for each song.
* **User:**
  + Stores users preferences and level and other calculated features in order to provide best results for future recommendations.

**Data Design**

**Database Schema**

1. **Users:**
   * user\_id: INT
   * name: VARCHAR
   * email: VARCHAR
   * password: VARCHAR
   * created\_at: TIMESTAMP
   * level: INT
2. **Chords:**
   * chord\_id: INT
   * name: VARCHAR
   * complexity: INT
3. **Songs:**
   * song\_id: INT
   * title: VARCHAR
   * artist: VARCHAR
4. **Genres:** 
   * genre\_id: INT
   * A diagram of a computer

     Description automatically generatedname: VARCHAR

**A screenshot of a computer

Description automatically generatedClass Diagram**