Audio Hive

Comprehensive Project Documentation

# 1. Project Overview

AudioHive is a real-time collaborative music streaming Android application that allows multiple users to share and control Spotify playback sessions simultaneously across devices. The app leverages Firebase Realtime Database for instantaneous data synchronization and Spotify Web API for music search and playback control.

# 2. Technical Implementation

• Language: Kotlin  
• Architecture: MVVM with LiveData & Coroutines  
• Backend: Firebase Realtime Database  
• Music Service: Spotify Web API (OAuth 2.0)  
• UI: Material Design 3 with custom animations  
• Build: Gradle 8 (Android Gradle Plugin)

# 3. Setup Guides

## 3.1 Firebase Setup

Follow FIREBASE\_SETUP\_GUIDE.md for creating a Firebase project, adding the Android app, and configuring Realtime Database.

## 3.2 Spotify API Setup

Follow SPOTIFY\_SETUP\_GUIDE.md for registering a Spotify developer application, setting redirect URIs, and configuring credentials in SpotifyService.kt.

## 3.3 Quick Start

See GETTING\_STARTED.md for a 15-minute quick-start checklist that covers Firebase, Spotify, building, and testing.

# 4. Architecture & Data Flow

The application uses MVVM architecture. UI events are handled in Activities which update ViewModels; ViewModels interact with FirebaseService and SpotifyService. Real-time data flows from host devices to Firebase and back to all connected clients, while SpotifyService controls music playback on the host device.

# 5. Project Structure

• activities/ – UI screens (CreateSessionActivity, JoinSessionActivity, MusicPlayerActivity)  
• services/ – FirebaseService (real-time sync) & SpotifyService (API integration)  
• models/ – Data classes (Session, Track, PlaybackState, User)  
• adapters/ – RecyclerView adapters for lists  
• utils/ – UIAnimationHelper & UIOptimizationHelper  
• res/ – Layouts, drawables, animations, themes

# 6. Key Features

• Real-time multi-user session management  
• Collaborative queue with instant synchronization  
• Full Spotify search, playback, and control  
• Modern Material Design 3 user interface with smooth animations  
• Robust error handling and offline support

# 7. Build & Run Instructions

1. Clone the repository and open in Android Studio.  
2. Add google-services.json to the app/ folder.  
3. Add your Spotify CLIENT\_ID and CLIENT\_SECRET in SpotifyService.kt.  
4. Sync Gradle and run on an Android device (API 24+).

# 8. Testing Guide

• Single-device test: Authenticate with Spotify, create a session, add songs, and verify playback.  
• Multi-device test: One device hosts, others join with the room code and verify real-time sync.  
• Real-world test: Connect host to a Bluetooth speaker and enjoy collaborative playback.

# 9. Change Log Summary

Key milestones include crash fixes, complete playback state overhaul, multi-user collaboration enhancements, UI/UX polish with custom animations, and extensive debugging improvements. See MusicPlayerActivity\_Readme.md for the full chronological log.

# 10. Future Enhancements

• Album art loading with Glide  
• Drag-and-drop queue reordering  
• 30-second track previews  
• Vote-to-skip functionality  
• User profiles and session analytics