Discord Music Bot – Complete Project Documentation

1. Project Overview

This project is a feature-rich, scalable, and user-friendly music bot for Discord.
It supports streaming music from multiple platforms, user-controlled queues, and both "request" and "radio" modes.

The bot can:
- Run 24/7 in one or more guilds.
- Handle multiple concurrent queues (per server).
- Allow interactive song control with slash commands and buttons.
- Be deployed on Docker, VPS, or cloud platforms.

2. Objectives

- Play music in voice channels with high reliability.
- Allow users to choose what to play.
- Offer an optional radio mode for continuous playback.
- Keep architecture modular for future features.
- Be easy to deploy and maintain.

3. Features

Core

- Multi-source streaming: YouTube, Spotify, SoundCloud, Direct URLs
- Queue Management: Add songs, Skip, Pause, Resume, View queue, Auto-play next track
- Auto Behavior: Leave when channel empty, Stay if 24/7 mode enabled
- Audio Controls: Volume adjustment, Stop & clear queue
- Error Handling: Auto-retry, Graceful error messages
- Multi-server support

User Experience

- Slash commands with autocomplete
- Rich embeds for song info & queue
- Button controls for quick actions
- Configurable DJ role permissions

Future Features

- Persistent queues
- Web dashboard
- Playlist saving/loading
- Song search with interactive menu

4. Architecture

Component Flow:
User Command → Command Handler → Queue Manager → Audio Player → Discord Voice

Modules:
1. Command Handler – Registers slash commands, executes logic
2. Queue Manager – Maintains per-server queues, triggers playback
3. Audio Player – Streams audio, handles events
4. Event Handlers – ready, interactionCreate, voiceStateUpdate
5. Utilities – Search Utility, Embed Utility, Logger

5. Data Structures
Queue Object Example:
{
textChannel: ,
voiceChannel: ,
connection: ,
player: ,
songs: [
{ title, url, duration, requestedBy }
],
playing: Boolean
}

6. Command List
/play – Search or play from URL
/skip – Skip current track
/queue – Show current song list
/pause – Pause playback
/resume – Resume playback
/stop – Stop playback & clear queue
/volume – Set volume (0–100)
/mode – Switch between radio & request modes

7. Setup Instructions
Prerequisites:
- Node.js v18+
- FFmpeg
- Docker (optional)
- Discord Bot Token

Installation:
git clone
cd discord-music-bot
npm install

Environment Variables (.env):
BOT_TOKEN=your_bot_token_here
VC_ID=voice_channel_id
YOUTUBE_URL=default_radio_url
DEFAULT_PREFIX=!

Run:
node index.js

8. Deployment
Dockerfile:
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
CMD ["node", "index.js"]

Docker Run:
docker build -t discord-music-bot .
docker run -d --name music-bot --env-file .env discord-music-bot

Hosting Options:
- VPS
- Railway.app
- Render.com
- Heroku

9. Error Handling & Stability
- Retry Logic
- Graceful Disconnect
- Logging
- Uncaught Error Handlers

10. Maintenance
- Keep play-dl updated
- Monitor bot via logs
- Backup configs
- Restart periodically

11. Future Scalability
- Sharding
- Database Integration (MongoDB, Redis)
- Web Dashboard