# Spotify MVP - Local Deployment Guide (Without Docker)

This guide will help you deploy the Spotify MVP locally on your machine without using Docker. Follow these step-by-step instructions to get your music streaming platform running.

## 📋 Prerequisites

Before starting, make sure you have the following installed on your machine:

### Required Software

* **Node.js 18+** - [Download here](https://nodejs.org/)
* **PostgreSQL 14+** - [Download here](https://www.postgresql.org/download/)
* **Git** - [Download here](https://git-scm.com/downloads)
* **npm or yarn** - Comes with Node.js

### Check Your Installations

# Check Node.js version (should be 18+)  
node --version  
  
# Check npm version  
npm --version  
  
# Check PostgreSQL version (should be 14+)  
psql --version  
  
# Check Git version  
git --version

## 🗄️ Step 1: Database Setup

### 1.1 Start PostgreSQL Service

#### On Windows:

* Start PostgreSQL from the Start Menu or Services panel
* Or use Command Prompt: net start postgresql-x64-14 (adjust version as needed)

#### On macOS:

# If installed via Homebrew:  
brew services start postgresql  
  
# Or manually:  
pg\_ctl -D /usr/local/var/postgres start

#### On Linux (Ubuntu/Debian):

sudo systemctl start postgresql  
sudo systemctl enable postgresql

### 1.2 Create Database and User

# Connect to PostgreSQL as superuser  
sudo -u postgres psql  
  
# Or on Windows/macOS:  
psql -U postgres

In the PostgreSQL shell, run these commands:

-- Create database  
CREATE DATABASE spotify\_mvp;  
  
-- Create user (optional, but recommended)  
CREATE USER spotify\_user WITH ENCRYPTED PASSWORD 'spotify\_password';  
  
-- Grant privileges  
GRANT ALL PRIVILEGES ON DATABASE spotify\_mvp TO spotify\_user;  
  
-- Connect to the new database  
\c spotify\_mvp  
  
-- Enable UUID extension  
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";  
  
-- Exit PostgreSQL shell  
\q

### 1.3 Import Database Schema

Navigate to your project directory and import the schema:

# Navigate to the spotify-mvp directory  
cd spotify-mvp  
  
# Import the database schema  
psql -U spotify\_user -d spotify\_mvp -f database/schema.sql  
  
# Or if using postgres user:  
psql -U postgres -d spotify\_mvp -f database/schema.sql

## 🔧 Step 2: Backend Setup

### 2.1 Navigate to Backend Directory

cd backend

### 2.2 Install Backend Dependencies

# Install all backend dependencies  
npm install

### 2.3 Create Environment Configuration

Create a .env file in the backend directory:

# Create .env file  
touch .env  
  
# Or on Windows:  
type nul > .env

Add the following configuration to the .env file:

# Database Configuration  
DATABASE\_URL=postgresql://spotify\_user:spotify\_password@localhost:5432/spotify\_mvp  
  
# JWT Configuration  
JWT\_SECRET=your-super-secret-jwt-key-change-this-in-production  
JWT\_EXPIRES\_IN=24h  
JWT\_REFRESH\_EXPIRES\_IN=7d  
  
# Server Configuration  
NODE\_ENV=development  
PORT=3001  
CORS\_ORIGIN=http://localhost:3000  
  
# Rate Limiting  
RATE\_LIMIT\_WINDOW\_MS=900000  
RATE\_LIMIT\_MAX\_REQUESTS=100  
  
# File Upload Configuration  
UPLOAD\_PATH=./uploads  
MAX\_FILE\_SIZE=50MB  
  
# Security  
TRUST\_PROXY=false

### 2.4 Create Uploads Directory

# Create directory for audio file uploads  
mkdir -p uploads

### 2.5 Run Database Migrations and Seed Data

# Run database migrations (if migration scripts exist)  
npm run db:migrate  
  
# Seed the database with sample data  
npm run db:seed

If these scripts don’t exist, you can manually insert some test data by connecting to PostgreSQL:

-- Connect to database  
psql -U spotify\_user -d spotify\_mvp  
  
-- Insert sample artist  
INSERT INTO artists (name, bio, genre) VALUES   
('Demo Artist', 'This is a demo artist for testing', 'Pop');  
  
-- Insert sample album  
INSERT INTO albums (artist\_id, title, release\_date, genre) VALUES   
(1, 'Demo Album', '2024-01-01', 'Pop');  
  
-- Insert sample track (you'll need to add actual audio files)  
INSERT INTO tracks (artist\_id, album\_id, title, duration\_seconds, file\_url, file\_path, genre) VALUES   
(1, 1, 'Demo Song', 180, '/uploads/demo-song.mp3', './uploads/demo-song.mp3', 'Pop');

## 🎵 Step 3: Add Sample Music (Optional)

### 3.1 Copy Sample Music Files

# Go back to project root  
cd ..  
  
# Copy sample music to backend uploads folder  
cp sample-music/\* backend/uploads/  
  
# Make sure files have correct permissions  
chmod 644 backend/uploads/\*

### 3.2 Run Sample Music Setup Script

# Make the script executable  
chmod +x setup-sample-music.sh  
  
# Run the script to populate database with sample music  
./setup-sample-music.sh

## ⚛️ Step 4: Frontend Setup

### 4.1 Navigate to Frontend Directory

cd frontend

### 4.2 Install Frontend Dependencies

# Install all frontend dependencies  
npm install  
  
# Or if you prefer pnpm (faster):  
npm install -g pnpm  
pnpm install

### 4.3 Create Frontend Environment Configuration

Create a .env file in the frontend directory:

# Create .env file  
touch .env

Add the following configuration:

# API Configuration  
REACT\_APP\_API\_URL=http://localhost:3001/api  
REACT\_APP\_STREAM\_URL=http://localhost:3001/api/stream  
  
# App Configuration  
REACT\_APP\_APP\_NAME=Spotify MVP  
REACT\_APP\_VERSION=1.0.0  
  
# Development  
GENERATE\_SOURCEMAP=false

## 🚀 Step 5: Start the Application

### 5.1 Start Backend Server

Open a new terminal window and navigate to the backend directory:

cd spotify-mvp/backend  
  
# Start the backend development server  
npm run dev  
  
# Or for production:  
npm start

You should see output like:

Server is running on port 3001  
✅ Database connected successfully  
✅ Server is ready to accept connections

### 5.2 Start Frontend Server

Open another terminal window and navigate to the frontend directory:

cd spotify-mvp/frontend  
  
# Start the frontend development server  
npm run dev  
  
# Or if using pnpm:  
pnpm run dev

You should see output like:

✅ Local: http://localhost:3000/  
✅ Network: http://192.168.1.xxx:3000/

## 🌐 Step 6: Access Your Application

### 6.1 Open Your Browser

Navigate to: **http://localhost:3000**

### 6.2 Test the Application

1. **Health Check**: Visit http://localhost:3001/health
2. **API Test**: Visit http://localhost:3001/api/tracks
3. **Frontend**: Visit http://localhost:3000

### 6.3 Create Test Account

1. Click “Sign Up” on the homepage
2. Create an account with:
   * Email: test@example.com
   * Password: password123
   * Full Name: Test User
3. Or use pre-seeded accounts (if available):
   * Admin: admin@spotify-mvp.com / password123
   * Demo: demo@spotify-mvp.com / password123

## 🔧 Step 7: Troubleshooting

### Common Issues and Solutions

#### Database Connection Issues

# Check if PostgreSQL is running  
sudo systemctl status postgresql # Linux  
brew services list | grep postgres # macOS  
  
# Test database connection  
psql -U spotify\_user -d spotify\_mvp -c "SELECT 1;"

#### Backend Port Issues

# Check what's running on port 3001  
lsof -i :3001 # macOS/Linux  
netstat -ano | findstr :3001 # Windows  
  
# Kill process if needed  
kill -9 [PID] # macOS/Linux

#### Frontend Build Issues

# Clear npm cache  
npm cache clean --force  
  
# Delete node\_modules and reinstall  
rm -rf node\_modules  
npm install

#### Audio File Issues

# Check file permissions  
ls -la backend/uploads/  
  
# Set correct permissions  
chmod 644 backend/uploads/\*

### Environment Variable Issues

If you get environment variable errors, check:

1. **Backend .env file exists** in spotify-mvp/backend/.env
2. **Frontend .env file exists** in spotify-mvp/frontend/.env
3. **Correct DATABASE\_URL format**: postgresql://user:password@host:port/database

### Database Schema Issues

If you get database table errors:

# Recreate database  
psql -U postgres -c "DROP DATABASE IF EXISTS spotify\_mvp;"  
psql -U postgres -c "CREATE DATABASE spotify\_mvp;"  
psql -U postgres -d spotify\_mvp -f database/schema.sql

## 📊 Step 8: Verify Everything Works

### Backend API Endpoints

Test these URLs in your browser or with curl:

# Health check  
curl http://localhost:3001/health  
  
# Get tracks  
curl http://localhost:3001/api/tracks  
  
# Get artists  
curl http://localhost:3001/api/artists

### Frontend Features

1. **Login/Registration** - Create and login with accounts
2. **Music Browse** - View tracks, artists, albums
3. **Search** - Search for music content
4. **Music Player** - Play audio tracks (if sample music is loaded)
5. **Playlists** - Create and manage playlists

## 🎯 Step 9: Production Considerations

For production deployment, consider:

### Security

* Change all default passwords and secrets
* Use environment variables for sensitive data
* Enable HTTPS
* Configure proper CORS settings

### Performance

* Set up connection pooling for database
* Configure CDN for audio files
* Enable compression and caching
* Set up load balancing

### Monitoring

* Set up logging (Winston, etc.)
* Monitor database performance
* Set up error tracking (Sentry, etc.)
* Configure health checks

## 🎉 Success!

Your Spotify MVP should now be running locally! You can:

* Browse music at http://localhost:3000
* Create user accounts
* Play music (if sample tracks are loaded)
* Create and manage playlists
* Search for content
* Test the full streaming experience

## 📞 Need Help?

If you encounter issues:

1. Check the troubleshooting section above
2. Verify all prerequisites are correctly installed
3. Ensure all environment variables are set
4. Check that PostgreSQL is running and accessible
5. Verify that both frontend and backend servers are running

The application logs will show detailed error messages to help diagnose any issues.