MUSICAPP - PROJECT FEATURES

PROJECT OVERVIEW

MusicApp is an Angular-based music streaming application that allows users to browse, search, and play music with mood-based filtering, playlist management, and personalized recommendations.

CORE FEATURES

MUSIC BROWSING & DISCOVERY

- Browse all available songs in the music library
- Filter songs by mood categories (Chill, Focus, Energy, Classic)
- Search functionality across songs, artists, and albums
- Display song count and metadata (title, artist, album, duration)
- Recently played songs on home page

PLAYER FUNCTIONALITY

- Integrated YouTube player for music playback
- Play/pause controls
- Select and play any song from the library
- Real-time player status tracking
- YouTube IFrame API integration

MOOD-BASED FILTERING

- Four mood categories: Chill, Focus, Energy, Classic
- Filter songs by selected mood
- View mood recommendations on home page
- Quick access to mood-based playlists

SEARCH & DISCOVERY

- Full-text search across songs, artists, and albums
- Real-time search results filtering
- Combined search with mood filtering
- Empty state handling for no results

LIKED SONGS / FAVORITES

- Toggle like/unlike on individual songs
- Track liked songs for user

- Persist likes in database
- Access to liked songs library

PLAYLIST MANAGEMENT

- View user playlists
- Browse playlist details with songs
- Create and manage personal playlists
- Public/private playlist options

PLAYBACK TRACKING

- Record when users play a song
- Track play count per song
- Store playback history in database
- Fetch play statistics

USER PROFILE

- User profile toggle in navigation
- Display user information
- Quick access profile menu

TECHNICAL COMPONENTS

FRONTEND COMPONENTS

- Home Component: Welcome page, recently played, mood suggestions, quick actions
- Browse Component: Main song browsing with search and mood filters
- Library Component: User's liked songs and personal collection
- Playlist Detail Component: Detailed view of individual playlists

CUSTOM PIPES

- Duration Pipe: Format song duration (seconds to MM:SS format)
- Highlight Pipe: Highlight search terms in results
- Time-Ago Pipe: Display relative time (e.g., "2 hours ago")

SERVICES

- Music Service: Central service for all music-related API calls and state management
    - Load songs with filtering
    - Toggle like status
    - Record playback
    - Manage selected song state
    - Fetch liked songs and playlists

API ENDPOINTS

- GET /api/songs - Fetch songs with optional mood and search filters
- GET /api/songs/:id - Get specific song details
- POST /api/likes/:songId - Toggle like status for a song
- POST /api/play/:songId - Record a song play event
- GET /api/liked-songs - Get user's liked songs
- GET /api/playlists - Get user's playlists
- GET /api/playlists/:id - Get playlist details with songs

DATABASE FEATURES

- Songs table with mood categorization
- Artists and albums data
- User likes tracking
- Playlist management
- Play history/statistics
- YouTube URL integration for each song

USER EXPERIENCE

NAVIGATION

- Responsive navigation bar with app branding
- Router links to Browse, Home, Library pages
- User profile menu toggle
- Quick navigation to all major sections

VISUAL DESIGN

- Grid layout for song cards
- Card-based UI components
- Mood-based visual indicators
- Loading states with spinner animation
- Error handling with retry functionality
- Empty state messaging

INTERACTIONS

- Click to play songs
- Hover effects on interactive elements
- Filter pills for mood selection
- Real-time search input
- Smooth transitions between views

ANGULAR FEATURES USED

- Angular Signals for reactive state management
- Standalone components
- Router and RouterLink for navigation
- HTTP Client for API calls
- Computed signals for derived state
- Effects for side effects
- OnInit lifecycle hook
- ViewChild decorators
- Dependency Injection
- Custom pipes
- Responsive templating with @for and @if

DATABASE SCHEMA

Key Tables

- songs: Store song metadata (title, artist_id, album_id, mood, duration, youtube_id)
- artists: Artist information
- albums: Album information
- user_likes: Track user's liked songs
- playlists: User playlists
- play_history: Track song plays

SERVER SETUP

- Express.js backend running on port 3000
- SQLite/MySQL database for data persistence
- CORS enabled for cross-origin requests
- JSON middleware for API communication
- Database initialization and seeding scripts included

DEPLOYMENT READY

- The application includes:

- Production-ready Angular build configuration
- API server with proper error handling
- Database setup scripts
- CORS configuration for deployment
- Comprehensive component structure
- Reusable service architecture
- State management with signals