Angular Javascript Frontend Frameworks(Angular)

Submitted By:

Anush Jowin A - 2462044

Duggempudi Praveen Kumar Reddy - 2462066

Gutha Nihitha - 2463021

Darain Brit A – 2462060

- **Institution**: *Christ University*
- **Date of Submission**: *23/01/2026*

Project 14:

Music Streaming and Playlist Management Application

Github: https://github.com/Darain-Brit-A/Music-Streaming_and_playlist_Mangement_application

Deployment Link: https://music-streaming-and-playlist-mangem.vercel.app/

1. Introduction

The Music Streaming and Playlist Management Application is a modern single-page web application (SPA) developed using Angular and TypeScript. The application allows users to browse songs, play audio tracks, manage playlists, and explore artists in an interactive and user-friendly interface. This project demonstrates the practical use of Angular's complete ecosystem including components, routing, services, dependency injection, forms, observables, and Angular Material for UI design.

The final output of the project, titled SerenityTunes, provides a dark-themed, responsive music streaming interface with core music player controls and playlist management features.

2. Objectives

The main objectives of this project are:

- To design and build a music streaming application using Angular and TypeScript
- To understand Angular component-based architecture
- To implement routing and navigation in a single-page application
- To use services and dependency injection for data sharing

- To apply reactive and template-driven forms with validation
- To use RxJS observables for audio playback state management
- To design a modern UI using Angular Material components

3.Scope of the Project

- Develop a single-page music streaming application using Angular and TypeScript
- Implement audio playback controls such as play, pause, next, and previous
- Provide playlist management including create, add, and delete songs
- Enable navigation and artist details using Angular routing
- Design a responsive user interface using Angular Material with mock data

4.Tools & Technologies Used

Tool/Technology	Purpose
Angular	Frontend framework
TypeScript	Strongly typed programming language
VS Code	Code editor
CSS	Styling and layout
JSON Server	Mock backend for songs, artists, and playlists
Node.js	Runtime environment
Angular CLI	Project scaffolding and build tool
Angular Material	UI components and theming

5.System Architecture

The application follows a modular Angular architecture:

- Components handle UI and user interactions

- Services manage data fetching and business logic
- Routing module controls navigation
- Models (interfaces) define structured data
- Angular Material ensures consistent UI design

6.Key Features

Feature	Description
Responsive UI Design	Adapts to different screen sizes using Angular Material
Music Playback Control	Supports play, pause, next, previous, and progress bar
Playlist Management	Allows users to create and manage playlists
Angular Routing	Enables smooth navigation without page reloads
Observable-Based State	Uses RxJS to manage audio playback and UI updates

7.Angular Components Design

The application is divided into reusable components:

Song List Component

- Displays all available songs
- Provides play, add-to-playlist, and favorite options

Song Player Component

- Controls play/pause functionality
- Displays progress bar and playback controls
- Shows current song status (e.g., "No song playing")

Playlist Manager Component

- Allows users to create, edit, and delete playlists
- Manages songs within playlists

Artist Detail Component

- Displays artist biography
- Shows top tracks by the selected artist

Navbar Component

- Provides navigation links (Home, Artists, Playlists)
- Uses Angular Material Toolbar

8.Services and Dependency Injection

MusicService

- Fetches songs and playlists from mock JSON data
- Handles song filtering and retrieval

AudioService

- Manages audio playback using HTML5 Audio API
- Controls play, pause, next, and previous actions

UserService

- Manages user preferences such as favorites

9.Angular Material and UI Design

The UI is designed using Angular Material components:

- **MatToolbar** – Navigation and branding
- **MatCard** – Song and album previews
- **MatSlider** – Volume and playback progress
- **MatDialog** – Playlist confirmation and song details
- **MatTabs** – Switching between songs, albums, and artists

Material theming is applied to support:

- Dark mode
- Responsive grid layouts

The final UI (**SerenityTunes**) provides a clean, modern, and visually appealing music streaming experience.

10.Challenges Faced & Solutions

Challenge	Solution
Managing audio playback state	Used centralized AudioService with RxJS Observables
Component communication issues	Implemented Angular services with Dependency Injection
Form validation errors	Used Reactive Forms with validators
UI responsiveness issues	Applied Angular Material responsive layouts
Handling async data	Used HttpClient with Observables

11.Outcome

- Complete Angular project with modular architecture
- Functional music player with playlist management
- Reactive and template-driven forms with validation
- Angular Material-based responsive UI with dark/light themes
- Mock backend integration using JSON data

- Project documentation including setup steps, architecture, and UI screenshots

12.Future Enhancements

- Integrate a real backend with database and user authentication to enable personalized accounts and persistent playlists
- Add advanced music features such as recommendations, search filters, and offline playback support

13.Sample Code

```
import { ApplicationConfig, provideBrowserGlobalErrorListeners } from '@angular/core';

export const appConfig: ApplicationConfig = {
  providers: [
    provideBrowserGlobalErrorListeners(),

  ]
};
```

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { NavbarComponent } from './components/navbar/navbar.component';
import { SongListComponent } from './components/song-list/song-list.component';
import { SongPlayerComponent } from './components/song-player/song-player.component';
import { PlaylistManagerComponent } from './components/playlist-manager/playlist-manager.component';
import { ArtistsListComponent } from './components/artists-list/artists-list.component';

@Component({
  selector: 'app-root',
  imports: [
    CommonModule,
    NavbarComponent,
    SongListComponent,
    SongPlayerComponent,
    PlaylistManagerComponent,
    ArtistsListComponent
  ],
  template: `
    <div class="app">
      <app-navbar (viewChange)="currentView = $event"></app-navbar>

      <main class="content">
        <div *ngIf="currentView === 'home'">
          <app-song-list></app-song-list>
        </div>

        <div *ngIf="currentView === 'artists'">
          <app-artists-list></app-artists-list>
        </div>
```

Components:

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ArtistModel } from '../../models/artist.model';
import { SongModel } from '../../models/song.model';
import { MusicService } from '../../services/music.service';

@Component({
  selector: 'app-artist-detail',
  standalone: true,
  imports: [CommonModule],
  template: `
    <div class="artist-detail" *ngIf="selectedArtist">
      <div class="artist-header">
        <img
          *ngIf="selectedArtist.imageUrl"
          [src]="selectedArtist.imageUrl"
          [alt]="selectedArtist.name">
        <div
          *ngIf="!selectedArtist.imageUrl"
          class="placeholder-image">
          {{selectedArtist.name.charAt(0)}}
        </div>
        <h2>{{selectedArtist.name}}</h2>
      </div>

      <div class="artist-bio">
        <h3>Biography</h3>
        <p>{{selectedArtist.bio}}</p>
```

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-artists-list',
  standalone: true,
  imports: [CommonModule],
  template: `
    <div class="artists-list">
      <h2>Artists</h2>
      <div class="artists">
        <div
          *ngFor="let artist of artists"
          class="artist-card">
          <div class="artist-image" *ngIf="!artist.imageUrl">
            {{artist.name.charAt(0)}}
          </div>
          <img *ngIf="artist.imageUrl" [src]="artist.imageUrl" [alt]="artist.name">
          <div class="artist-info">
            <h3>{{artist.name}}</h3>
            <p class="genre">{{artist.genre}}</p>
            <p class="song-count">{{artist.songCount}} songs</p>
            <p class="bio">{{artist.bio}}</p>
          </div>
        </div>
      </div>
    </div>
  `,
  styles: [`
```

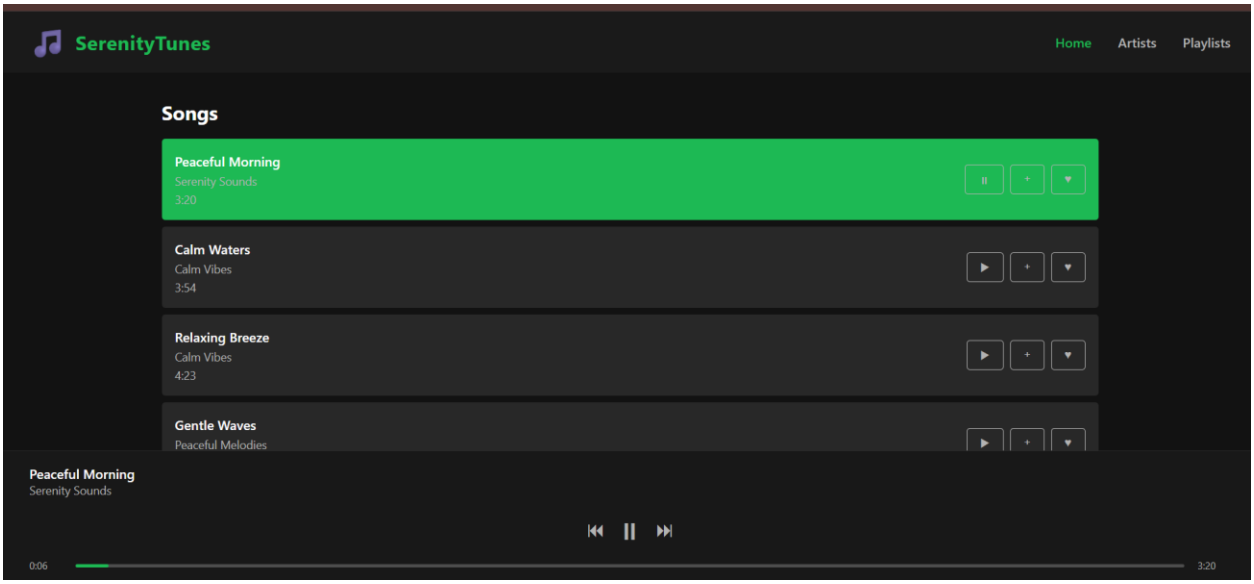
```
import { Component, Output, EventEmitter } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-navbar',
  standalone: true,
  imports: [CommonModule],
  template: `
    <nav class="navbar">
      <div class="logo-section">
        <div class="logo">🎵</div>
        <span class="app-name">SerenityTunes</span>
      </div>
      <div class="nav-items">
        <a
          *ngFor="let item of navItems"
          [class.active]="activeView === item.view"
          (click)="setActiveView(item.view)">
          {{item.label}}
        </a>
      </div>
    </nav>
  `,
  styles: [`
    .navbar {
      background: #1a1a1a;
      padding: 1rem 2rem;
      display: flex;
      align-items: center;
      justify-content: space-between;
```

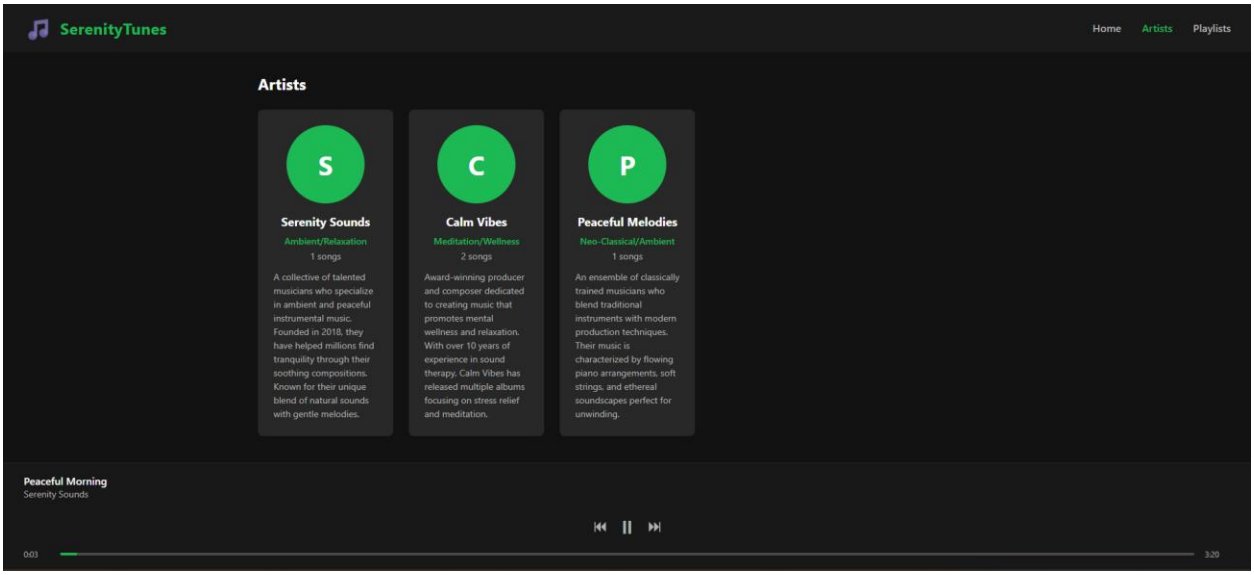
```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { PlaylistModel } from '../../models/playlist.model';
import { PlaylistService } from '../../services/playlist.service';

@Component({
  selector: 'app-playlist-manager',
  standalone: true,
  imports: [CommonModule, FormsModule],
  template: `
    <div class="playlist-manager">
      <h2>Playlists</h2>

      <div class="create-form" *ngIf="showCreateForm">
        <input
          [(ngModel)]="newPlaylistName"
          placeholder="Playlist name"
          (keyup.enter)="createPlaylist()">
        <input
          [(ngModel)]="newPlaylistDesc"
          placeholder="Description"
          (keyup.enter)="createPlaylist()">
        <button (click)="createPlaylist()">Create</button>
        <button (click)="showCreateForm = false">Cancel</button>
      </div>

      <button
        *ngIf="!showCreateForm"
```

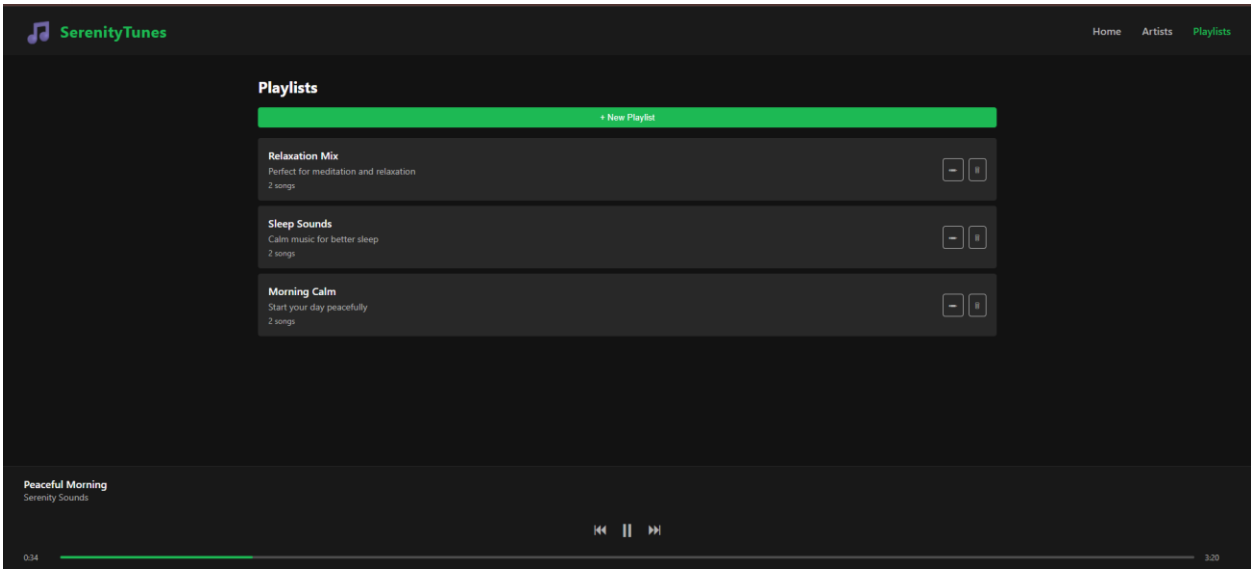
```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { SongModel } from '../../models/song.model';
import { AudioService } from '../../services/audio.service';
import { MusicService } from '../../services/music.service';
import { PlaylistService } from '../../services/playlist.service';
import { PlaylistModel } from '../../models/playlist.model';

@Component({
  selector: 'app-song-list',
  standalone: true,
  imports: [CommonModule],
  template: `
    <div class="song-list">
      <h2>Songs</h2>
      <div class="songs">
        <div
          *ngFor="let song of songs"
          class="song-item"
          [ngClass]="{'playing': currentSongId === song.id}">
          <div class="song-info">
            <span class="title">{{song.title}}</span>
            <span class="artist">{{getArtistName(song.artistId)}}</span>
            <span class="duration">{{song.getDurationFormatted()}}</span>
          </div>
          <div class="song-actions">
            <button (click)="playSong(song.id)">
              {{currentSongId === song.id ? '⏸' : '▶'}}
            </button>
            <button (click)="addToPlaylist(song.id)">+</button>
            <button
```

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { CommonModule } from '@angular/common';
import { SongStatus } from '../../models/song.model';
import { AudioService } from '../../services/audio.service';
import { MusicService } from '../../services/music.service';

@Component({
  selector: 'app-song-player',
  standalone: true,
  imports: [CommonModule],
  template: `
    <div class="player" *ngIf="currentSong">
      <div class="song-details">
        <span class="song-title">{{currentSong.title}}</span>
        <span class="song-artist">{{currentSong.artist}}</span>
      </div>

      <div class="controls">
        <button (click)="previous()">⏮</button>
        <button (click)="togglePlay()" class="play-btn">
          {{status === songStatus.Playing ? '⏸' : '▶'}}
        </button>
        <button (click)="next()">⏭</button>
      </div>

      <div class="progress">
        <span class="time">{{formatTime(currentTime)}}</span>
        <div class="progress-bar" (click)="seek($event)">
          <div
            class="progress-fill"
```

14.Screenshots of Final Output







15.Conclusion

The Music Streaming and Playlist Management Application successfully demonstrates the use of Angular and TypeScript to build a scalable, maintainable, and visually appealing single-page application. Through this project, key Angular concepts such as components, routing, services, dependency injection, forms, observables, and Material UI were effectively implemented, making it an excellent learning experience and a strong foundation for real-world Angular applications.

16. References

- L&T LMS : https://learn.lntedutech.com/Landing/MyCourse
- Angular Official Documentation – https://angular.io/docs
- Angular Material Documentation – https://material.angular.io
- TypeScript Documentation – https://www.typescriptlang.org/docs