Audio Library ASP.Net Core Project

Samuel Bailey

2025

Contents

[1 Project Overview 2](#_Toc210727506)

[2 Architecture Diagram 2](#_Toc210727507)

[3 Technology Stack 2](#_Toc210727508)

[4 Design Patterns 2](#_Toc210727509)

[4.1 MVC Architecture 2](#_Toc210727510)

[4.2 Data Model Details 2](#_Toc210727511)

[4.3 Wireframe Page (UI) Designs 2](#_Toc210727512)

[4.4 User Experience (UX) Workflows 3](#_Toc210727513)

[5 API Design 3](#_Toc210727514)

[6 Data Model 3](#_Toc210727515)

[7 Deployment Strategy 3](#_Toc210727516)

[8 Appendix 3](#_Toc210727517)

[8.1 Packages Used 3](#_Toc210727518)

[8.2 References 3](#_Toc210727519)

# Project Overview

This section provides a high-level overview of the ASP.NET Core solution, including its purpose, scope, and key features.

**Purpose:**

RecordShelf is a web-based application designed to allow users to manage and post their audio recordings, either for podcasts or music projects. The app supports CRUD operations (Create, Read, Update, Delete) on audio posts stored in a MongoDB NoSQL database. It also features categorization, search, and tagging capabilities. Users finally will be able to play their audio files from the browser.

**Features:**

* User registration and authentication
* Add, edit, delete, and view posts
* Categorize and tag recordings
* Search and filter by title, artist, or tags
* Play recordings

# Architecture Diagram

Insert architecture diagram here illustrating the ASP.NET Core API, Web projects, and MongoDB backend.

A diagram of a diagram

AI-generated content may be incorrect.

# Technology Stack

* ASP.NET Core
* MongoDB
* ASP.NET Core Web API
* ASP.NET Core MVC
* C#
* Visual Studio
* Docker (optional for deployment)

# Design Patterns

This section outlines the design patterns used in the solution, including MVC architecture, data modeling, UI wireframes, and UX workflows.

## MVC Architecture

The solution follows the Model-View-Controller (MVC) design pattern. The API project handles the controller logic and data access, while the Web project manages the views and user interactions.

* **Model**
* **View**
* **Controller**

## Data Model Details

The backend uses MongoDB to store data in collections. Each collection represents a domain entity. Relationships are modeled using embedded documents or references.

* Users Collection: Stores user profiles and authentication details.
* Products Collection: Contains product information and metadata.
* Orders Collection: Tracks user orders and payment status.
* Relationships: Orders reference Users and Products using ObjectId fields.

## Wireframe Page (UI) Designs

Wireframes represent the layout of key pages in the Web project. Placeholder descriptions:

* Home Page: Navigation bar, featured products, login/register buttons.
* Product Page: Product image, description, add to cart button.
* Checkout Page: Cart summary, payment form, order confirmation.

## **User Experience (UX) Workflows**

User experience workflows describe the journey users take through the application.

* Registration Workflow: User signs up, receives confirmation email, logs in.
* Shopping Workflow: User browses products, adds items to cart, checks out.
* Order Tracking Workflow: User views order history and status updates.

# API Design

Describe the RESTful API endpoints, request/response formats, authentication mechanisms, and error handling strategies.

# Data Model

Define the MongoDB collections and document structure. Include relationships and indexing strategies.

# Deployment Strategy

Outline the deployment process including environment setup, CI/CD pipeline, and hosting options (e.g., Azure, AWS, Docker).

# Appendix

## Packages Used

Details packages used in project. Include description of package and reasoning for including package.

## References

Using APA7 Referencing, include any relevant sources used to help develop/design your project.