Software Design Document
myAudiofiler

Course: Software Engineering 01:198:431

Group 12

Vasanth Banumurthy | Samantha Ames | Henry Chau

March 20, 2023

Table of Contents

1. Introduction

1.1 Overview

This document describes the design of myAudiofiler. This project is a part of ***SOFTWARE ENGINEERING 01:198:431 with Instructor: Juan Zhai***

Architecture Design describes the architectural styles used and diagrams specific to the project. It also reviews interface specifications and the data design.

Component Design showcases the class and sequence diagrams.

User Interface Design highlights the state machine diagram and offers a limited preview of the UI.

2. Architecture Design

2.1 Style

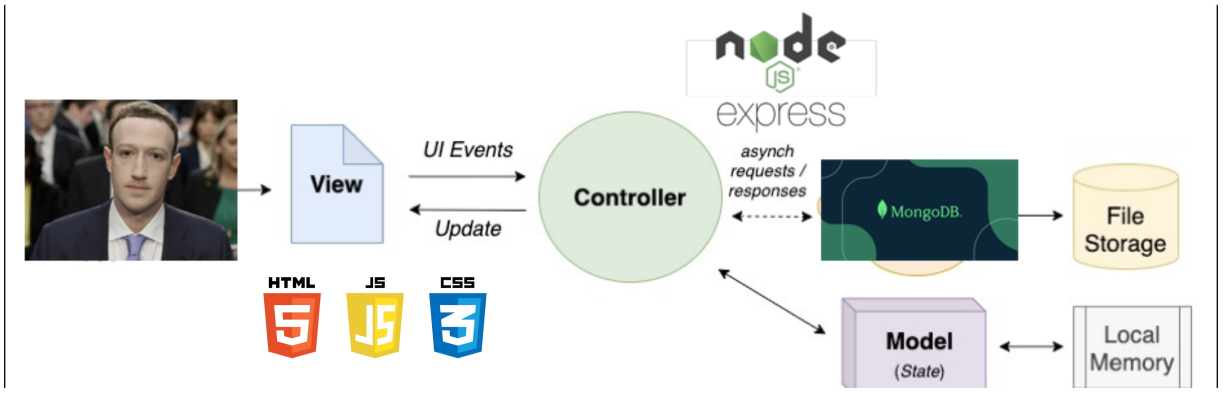
The Model-View-Controller (MVC) style was used because of its prominence in web development. myAudiofiler will launch as a web app.

Moreover, MVC provides separation of concerns by functionality, logic, and interface. This makes it simpler to organize programming among multiple developers.

1. ***Model:*** Includes the data for the vinyl records, such as album title, artist, release date, and rating. It would also include methods for adding, updating, deleting, and searching for vinyl records.

2. ***View:*** Includes the different screens and components that allow the user to interact with the data, such as the home screen, vinyl record details, add/edit vinyl record forms, search bar, user account page, and error/confirmation messages.
3. ***Controller:*** Handles actions such as adding, updating, deleting, and searching for vinyl records, as well as user authentication and account management.

2.2 Diagram



The view represents the UI. It's what our user sees and interacts with. These interactions are relayed to the controller. The controller processes the requests and fetches the relevant data from the model which is responsible for all the logic. The controller uses the data to help update the view.

Model: The model is implemented as a MongoDB collection that stores documents representing vinyl records, with fields for album title, artist, release date, and rating. Also included is a Node.js module that defines functions for adding, updating, deleting, and searching for records using the MongoDB driver for Node.js.

View: The view is implemented using HTML templates and a templating engine, which would render the dynamic data from the model.

Controller: The controller is implemented as an Express router that handles HTTP requests and responses using a combination of middleware functions and route handlers. The controller would receive user input from the view, invoke the appropriate methods on the model to handle the request, and then render the response back to the view.

The diagram above also features the tools used to accomplish this.

HTML:
https://developer.mozilla.org/en-US/docs/Web/HTML

CSS:
https://developer.mozilla.org/en-US/docs/Web/CSS

JavaScript:
https://developer.mozilla.org/en-US/docs/Web/JavaScript

Node and Express:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

MongoDB:
https://www.mongodb.com/docs/

2.3 Interface Specifications

1. ***Home screen:*** A landing page that displays a list of vinyl records with album title, artist, release date, and rating. Users can sort and filter the list by artist, release date, and rating.
2. ***Vinyl record details:*** A page that displays detailed information about a specific vinyl record, including the album title, artist, release date, rating, and tracklist.

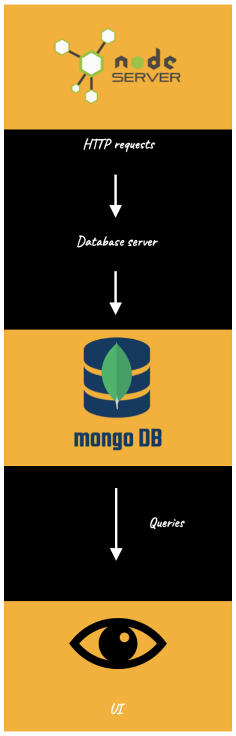
3. ***Add new vinyl record*:** A form that allows users to add a new vinyl record to the system by entering the album title, artist, release date, and rating.
4. ***Edit vinyl record:*** A form that allows users to edit an existing vinyl record by updating the album title, artist, release date, and rating.
5. **Delete vinyl record:** A button that allows users to delete an existing vinyl record from the system.
6. ***Search:*** A search bar that allows users to search for vinyl records by album title, artist, or release date.
7. ***User account:*** A page that allows users to view their account details, including their name, email, and password. Users can also update their account information and change their password.
8. ***Login/logout:*** A login page that allows users to log in to the system using their email and password. A logout button that allows users to log out of the system.
9. ***Error messages:*** Error messages that appear when a user tries to perform an invalid action, such as deleting a vinyl record that doesn't exist or entering an invalid rating.
10. ***Confirmation messages:*** Confirmation messages that appear when a user successfully performs an action, such as adding a new vinyl record or updating their account information.

2.4 Data Design

With MVC, data needs to be passed between components. This data must be specified and preferably human readable. Specifically, myAudiofiler must transfer details about vinyl records. Since we are using MongoDB (noSQL) as our data management solution, the following represents a sample json file featuring 70s rock albums.

```
{
  "vinyl_records": [
    {
      "id": 1,
      "album_title": "Led Zeppelin IV",
      "artist": "Led Zeppelin",
      "release_date": "November 8, 1971",
      "rating": 4.5
    },
    {
      "id": 2,
      "album_title": "The Dark Side of the Moon",
      "artist": "Pink Floyd",
      "release_date": "March 1, 1973",
      "rating": 5
    },
    {
      "id": 3,
      "album_title": "Rumours",
      "artist": "Fleetwood Mac",
      "release_date": "February 4, 1977",
      "rating": 4
    },
    {
      "id": 4,
      "album_title": "Boston",
      "artist": "Boston",
      "release_date": "August 25, 1976",
      "rating": 4.5
    },
    {
      "id": 5,
      "album_title": "Hotel California",
      "artist": "Eagles",
      "release_date": "December 8, 1976",
      "rating": 4
    }
  ]
}
```

2.5 Physical View



The physical view describes the system's hardware components and their interactions. Breaking myAudiofiler into three parts...

- A web server running Node.js and Express to handle HTTP requests and responses.
- A MongoDB database server to store and retrieve data for the application.
- A client-side web browser to render the user interface.

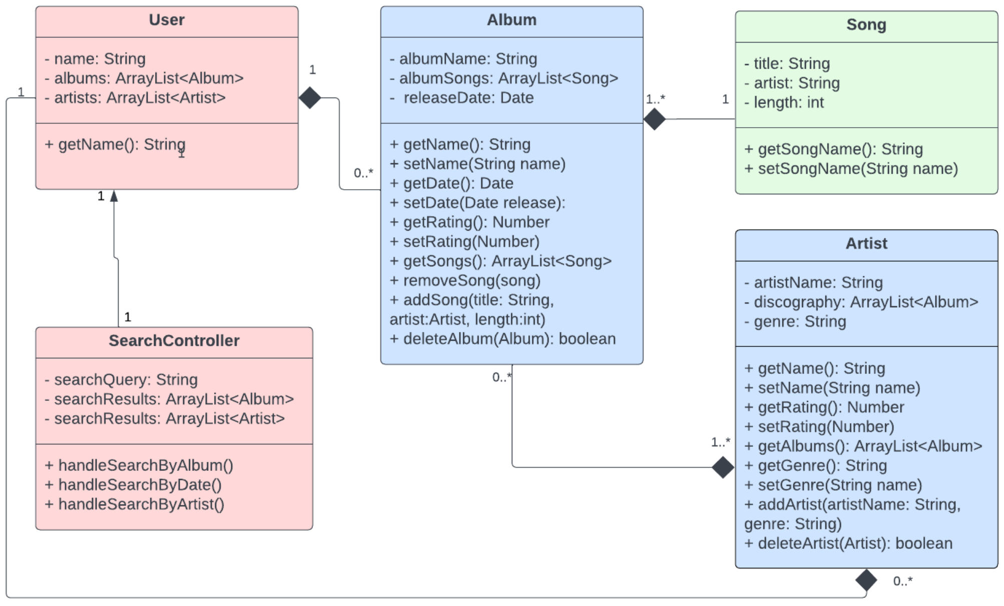
2.6 Process View

The process view describes the system's processes and how they interact with each other...

- The user initiates a search request by entering keywords into a search box on the application's user interface.
- The client-side JavaScript code sends the search query to the Node.js server using an HTTP request.
- The Node.js server handles the request by querying the MongoDB database for albums that match the search query.
- The server returns the search results to the client-side JavaScript code, which updates the user interface to display the search results.
- When the user selects an album, the client-side JavaScript code sends an HTTP request to the Node.js server to retrieve the album details from the MongoDB database.
- The server retrieves the album details from the database and returns them to the client-side JavaScript code, which updates the user interface to display the album details.

3. Component Design

3.1 Class Diagram



myAudiofiler

Figure 3.1 illustrates the UML diagram for myAudiofiler. The main class in the diagram is ‘Album’, which has attributes for the album name, album songs, and release date. The class, ‘Artist’, has the attributes for artist name, discography, and genre. The ‘Song’ class contains attributes for song title, artist name, and length (duration). The ‘User’ class has attributes name, album list, and artist list.

The relationships between the classes are as follows: ‘User’ and ‘Album’ have a composite relationship with ‘Artist’, ‘Album’ has a composite relationship with ‘User’, and ‘Song’ has a composite relationship with ‘Album’. ‘SearchController’ has a generalization relationship with ‘User’.

3.2 Sequence Diagram

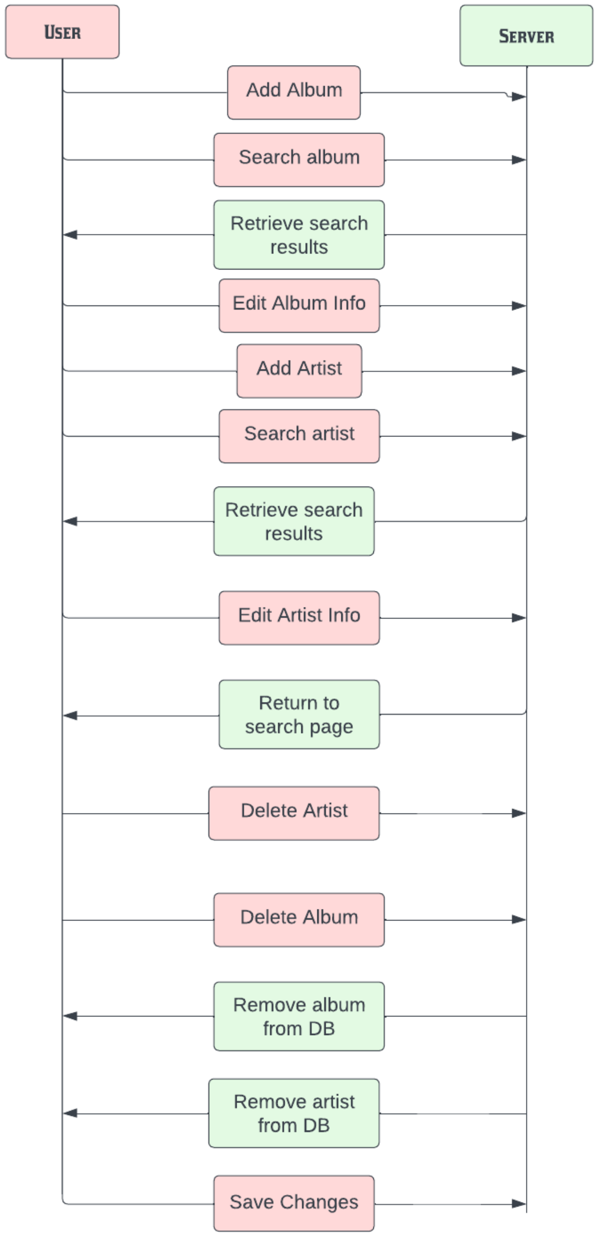


Figure 3.2 illustrates the Sequence diagram for myAudiofiler. The user begins by adding an album or an artist to their personal catalog. After an album or artist has been inducted into the catalog, the user has the option to search for the artist or album within their catalog, or to edit the information associated with an artist or

album within their catalog. The server will subsequently retrieve the search results for the user. Additionally, the user can delete an album or an artist from their catalog. The server will subsequently remove the artist or album from the database. Finally, the user is able to save any changes made to their catalog.

4. User Interface Design

4.1 State Machine Diagram

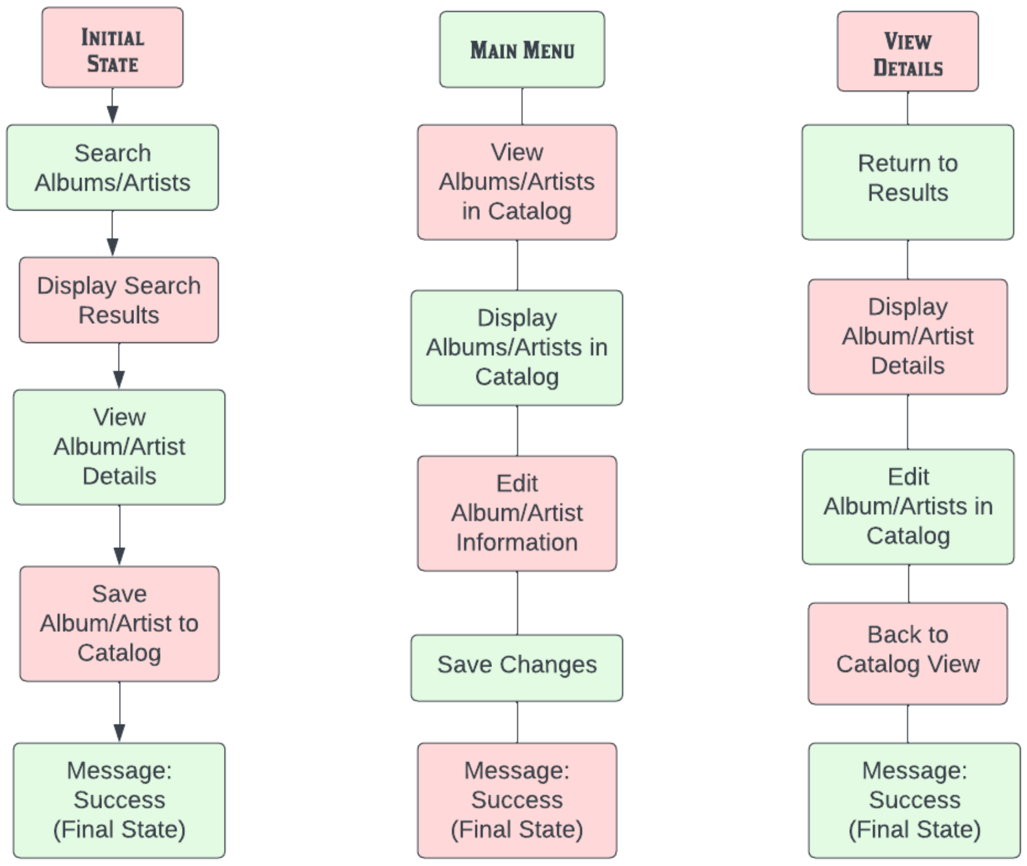
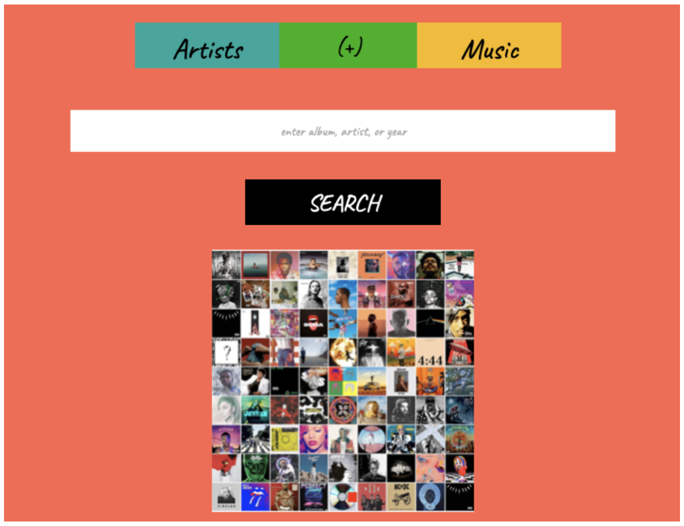
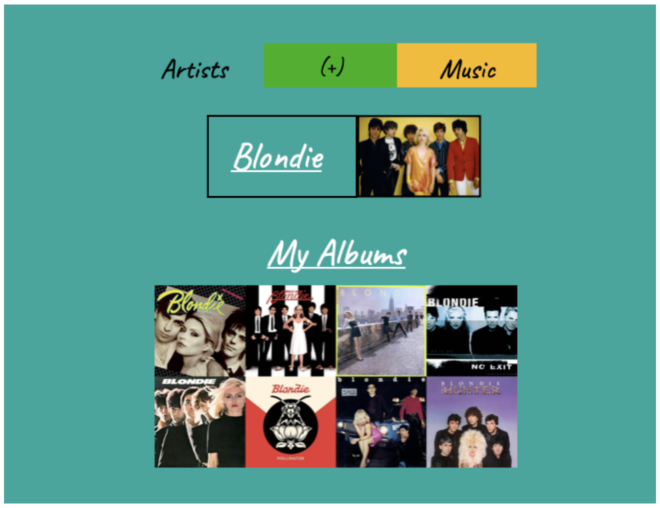
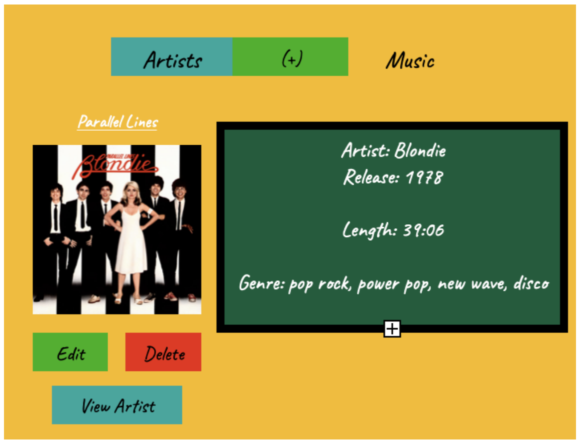

Initial State
Search Albums/Artists
Display Search Results
View Album/Artist Details
Save Album/Artist to Catalog
Message: Success (Final State)
Main Menu
View Albums/Artists in Catalog
Display Albums/Artists in Catalog
Edit Album/Artist Information
Save Changes
Message: Success (Final State)
View Details
Return to Results
Display Album/Artist Details
Edit Album/Artists in Catalog
Back to Catalog View
Message: Success (Final State)

Figure 4.1 illustrates the State Machine diagram for myAudiofiler. The 'Initial State' is the start state, in which the user begins the sequence by accessing the web application. The user is then presented with the 'Main Menu', where they can either choose to add an album/artist, search for an album/artist, or edit the associated information for an album/artist. If the user chooses to add an album or artist, they are taken to a page which prompts them to input the respective information. If the user chooses to search for an album/artist, similarly, they are taken to a page which prompts them to input their search criteria. If the user chooses to edit the information for an album/artist, they are taken to a page specific to their respective inquiry, and are prompted to edit the information accordingly. Subsequently, the user is able to save any changes and is returned to the main page, which displays their updated catalog. The final state for the user after any of these options is the 'Message of Success'.

4.2 Sketches


Artists
(+)
Music
enter album, artist, or year
SEARCH


Artists
(+)
Music
Blondie
My Albums


Artists
(+)
Music
Parallel Lines
Artist: Blondie
Release: 1978
Length: 39:06
Genre: pop rock, power pop, new wave, disco
Edit
Delete
View Artist