My Web Application
Albums Vault

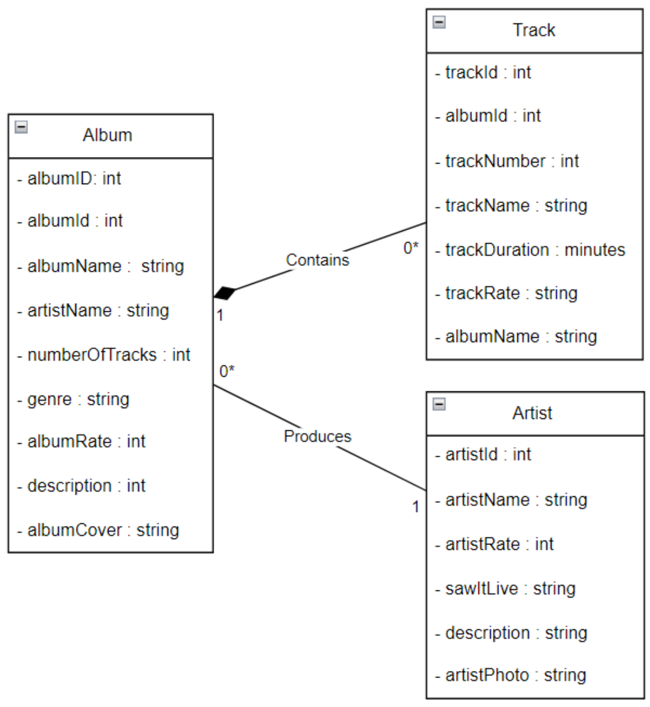
Technical Document

Contents

Backend Use

The main purpose of the back end of this web application is to store albums, artists, and tracks in a database. It handles requests from the frontend, ensuring that each request is processed correctly and returns the necessary data based on the request parameters.

Class Diagram



This is my class diagram with all my 3 entities (album as the main entity, artist and tracks)

Relationships: An Artist can have multiple Albums (1..0*). An Album belongs to only one Artist (1). This relationship ensures that each album is associated with a single artist. An Album can have multiple Tracks (1..0*). A Track belongs to only one Album (1). This relationship ensures that each track is associated with a album.

API Specification

GET Request

GET	**/albums/**		
Retrieve all albums with rating			
Parameters:	**Name**	**Type**	**Description**
** required*			
Responses:	**Code**	**Description / example if successful**	
	200	Returns a list of all albums with rating	
	404	List is empty	

GET	**/albums/tolistening**		
Retrieve all albums with no rating			
Parameters:	**Name**	**Type**	**Description**
** required*			
Responses:	**Code**	**Description / example if successful**	
	200	Returns a list of all albums with no rating	
	404	List is empty	

GET	**/albums/{albumID}**		
Retrieve the album by ID			
Parameters:	**Name**	**Type**	**Description**
** required*	UserID	query	Unique ID of the album object.
Responses:	**Code**	**Description / example if successful**	
	200	Returns the album with the ID that was provided.	
	404	User ID is incorrect	

GET	**/artist/{artistID}**		
Retrieve the artist by ID			
Parameters:	**Name**	**Type**	**Description**
** required*	ArtistID	query	Unique ID of the artist object.
Responses:	**Code**	**Description / example if successful**	
	200	Returns the artist with the ID that was provided.	
	404	Album ID is incorrect	

POST Request

POST	**/albums**		
Add a new album to the library.			
Parameters:	**Name**	**Type**	**Description**
** required*	Album	body	The album that will be added needs to be in a JSON format.
Responses:	**Code**	**Description / example if successful**	
	200	Album created successfully.	
	404	Invalid input data.	

POST	/artist		
Add a new artist to the library.			
Parameters:	**Name**	**Type**	**Description**
** required*	Artist	body	The artist that will be added needs to be in a JSON format.
Responses:	**Code**	**Description / example if successful**	
	200	Artist created successfully.	
	404	Invalid input data.	

POST	/artist/{artistID}		
Add a new album to the artist.			
Parameters:	**Name**	**Type**	**Description**
** required*	artistID	path	Needs the artistID to be added
	album	body	The album that will be added needs to be in a JSON format.
Responses:	**Code**	**Description / example if successful**	
	200	Album created successfully.	
	404	Invalid input data.	

POST	/tracks/{albumID}		
Add a new track to the album.			
Parameters:	**Name**	**Type**	**Description**
** required*	albumID	path	Needs the albumID to be added
	track	body	The track that will be added needs to be in a JSON format.
Responses:	**Code**	**Description / example if successful**	
	200	Track created successfully.	
	404	Invalid input data.	

PUT Request

PUT	/albums/{album_id}		
Update details of the existing album			
Parameters:	**Name**	**Type**	**Description**
** required*	album_id	path	Required. ID of the album to be edit.
	album	body	The album that will be added needs to be in a JSON format.
Responses:	**Code**	**Description / example if successful**	
	200	Album updated successfully.	
	404	Album not found.	

PUT	/artists/{artist_id}		
Update details of the existing artist			
Parameters:	**Name**	**Type**	**Description**
** required*	artist_id	path	Required. ID of the artist to be edit.
	artist	body	The artist that will be added needs to be in a JSON format.
Responses:	**Code**	**Description / example if successful**	
	200	Artist updated successfully.	
	404	Album not found.	

DELETE Request

DELETE	artist/{artist_id}		
Delete an artist from the library.			
Parameters:	**Name**	**Type**	**Description**
** required*	artist_id	path	Required. ID of the artist.
Responses:	**Code**	**Description / example if successful**	
	200	Artist deleted successfully.	
	404	Album not found.	

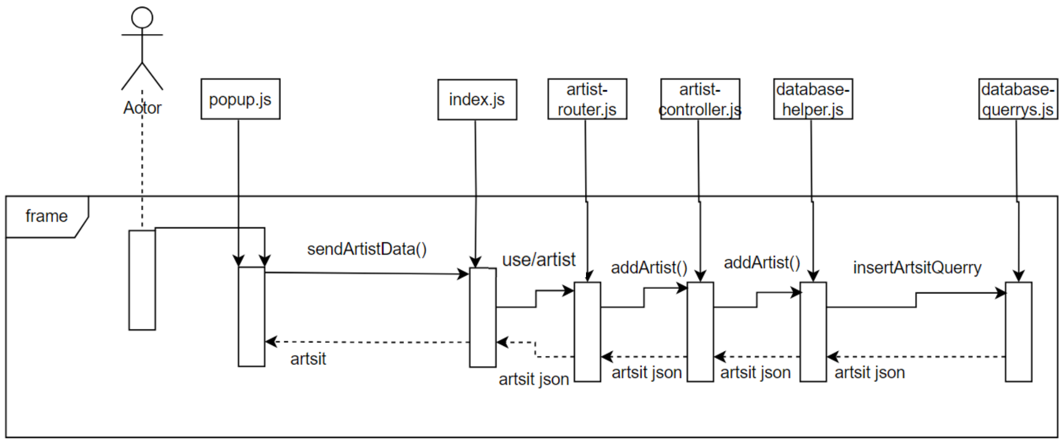
DELETE	album/{album_id}		
Delete an album from the library.			
Parameters:	**Name**	**Type**	**Description**
** required*	album_id	path	Required. ID of the artist.
Responses:	**Code**	**Description / example if successful**	
	200	Album deleted successfully.	
	404	Album not found.	

DELETE	tracks/{album_id}		
Delete an track from the album.			
Parameters:	**Name**	**Type**	**Description**
** required*	artist_id	path	Required. ID of the artist.
Responses:	**Code**	**Description / example if successful**	
	200	Track deleted successfully.	
	404	Track not found.	

DELETE	artist/{artist_id}		
Delete an album from the artist			
Parameters:	**Name**	**Type**	**Description**
** required*	artist_id	path	Required. ID of the artist.
Responses:	**Code**	**Description / example if successful**	
	200	Album deleted successfully.	
	404	Album not found.	

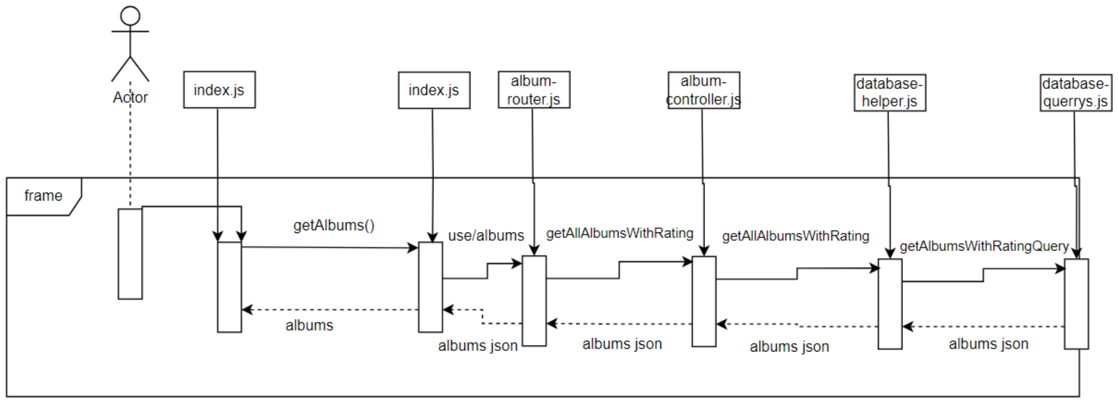
Sequence diagrams

User add new artist



The path for adding an artist is http://localhost:3000/albums and the user can go from any page add an artis or an album that will be the same process

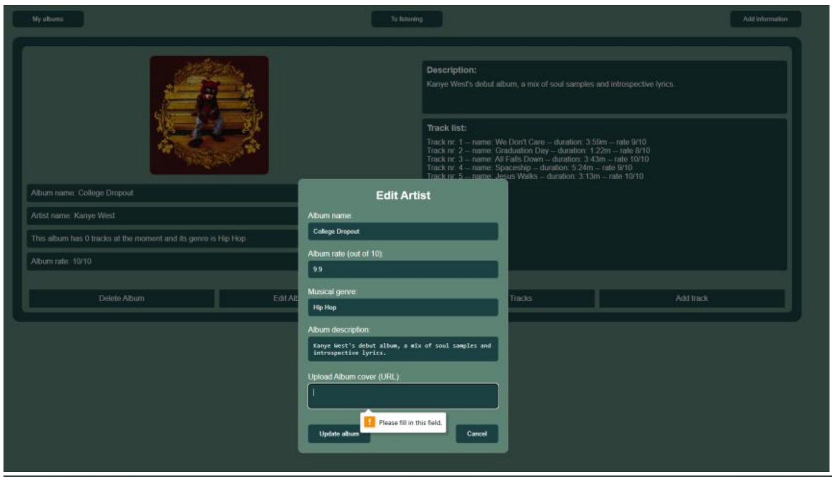
User gel all albums with rating



The path for adding an artist is http://localhost:3000/albums because this is the main page where the user gets all albums with a rating

Test report

Front-End



Description:
Kanye West's debut album, a mix of soul samples and introspective lyrics.

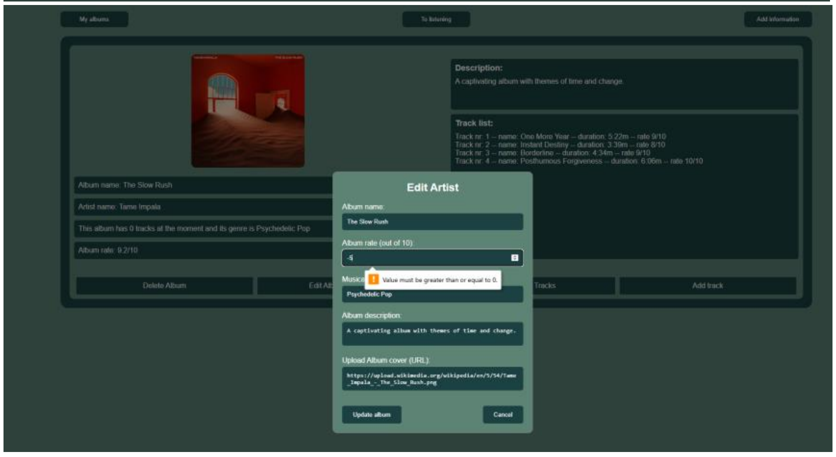
Track list:
Track nr: 1 -- name: We Don't Care -- duration: 3:59m -- rate 9/10
Track nr: 2 -- name: Graduation Day -- duration: 1:22m -- rate 8/10
Track nr: 3 -- name: All Falls Down -- duration: 3:43m -- rate 10/10
Track nr: 4 -- name: Spaceship -- duration: 5:24m -- rate 9/10
Track nr: 5 -- name: Jesus Walks -- duration: 3:13m -- rate 10/10

Album name: College Dropout
Artist name: Kanye West
This album has 0 tracks at the moment and its genre is Hip Hop
Album rate: 10/10

Edit Artist

Album name: College Dropout
Album rate (out of 10): 9.9
Musical genre: Hip Hop
Album description: Kanye West's debut album, a mix of soul samples and introspective lyrics.
Upload Album cover (URL):

Please fill in this field.



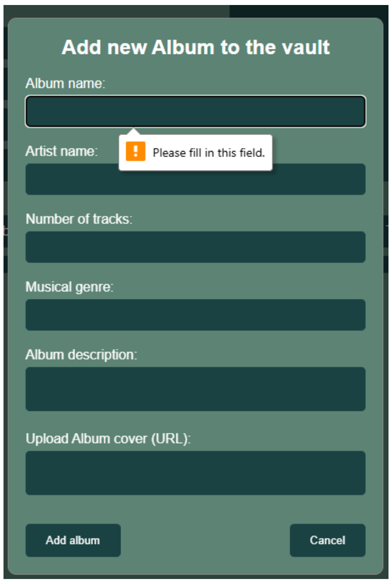
Description:
A captivating album with themes of time and change.

Track list:
Track nr: 1 -- name: One More Year -- duration: 5:22m -- rate 9/10
Track nr: 2 -- name: Instant Destiny -- duration: 3:39m -- rate 8/10
Track nr: 3 -- name: Borderline -- duration: 4:34m -- rate 9/10
Track nr: 4 -- name: Posthumous Forgiveness -- duration: 6:06m -- rate 10/10

Album name: The Slow Rush
Artist name: Tame Impala
This album has 0 tracks at the moment and its genre is Psychedelic Pop
Album rate: 9.2/10

Edit Artist

Album name: The Slow Rush
Album rate (out of 10): -5
Musica
Value must be greater than or equal to 0.
Psychedelic Pop
Album description: A captivating album with themes of time and change.
Upload Album cover (URL): https://upload.wikimedia.org/wikipedia/en/5/54/Tame_Impala_-_The_Slow_Rush.png



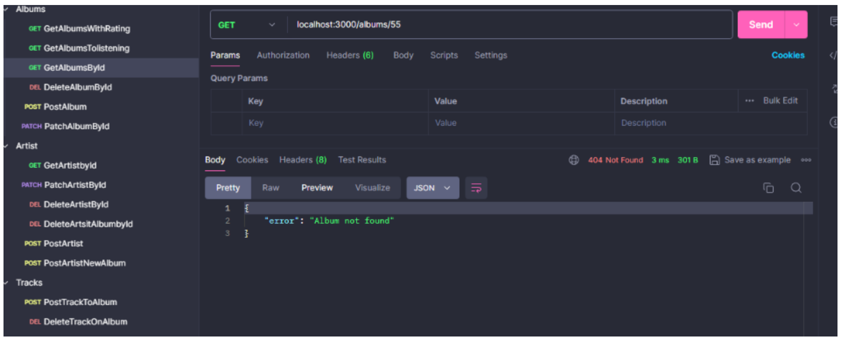
Add new Album to the vault

Album name:
Artist name:
Please fill in this field.
Number of tracks:
Musical genre:
Album description:
Upload Album cover (URL):

I tested the front-end by adding, editing, and deleting albums, artists, and tracks. I checked that the website responded correctly to these actions visually. I also made sure the system showed the right error messages when I entered invalid data, helping users correct mistakes.

For every pop-up or form to add or edit albums, artists, or tracks, I carefully checked for any mistakes in the information entered. This helped ensure the website guided users to fix errors before submitting their changes.

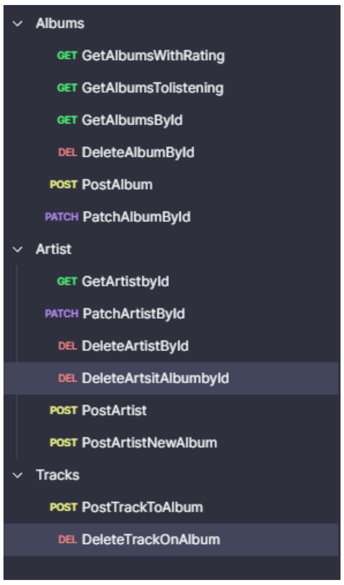
Back-End

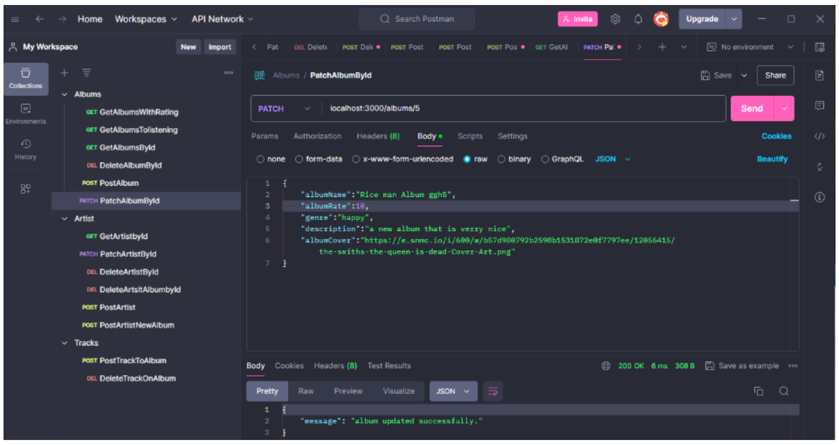


GET localhost:3000/albums/55

404 Not Found 3 ms 301 B

```
{
    "error": "Album not found"
}
```





PATCH localhost:3000/albums/5

```
{
    "albumName":"Rice man Album ggh5",
    "albumRate":10,
    "genre":"happy",
    "description":"a new album that is verry nice",
    "albumCover":"https://e.snmc.io/i/600/w/b57d980792b2598b1531872e0f7797ee/12056415/
        the-smiths-the-queen-is-dead-Cover-Art.png"
}
```

200 OK 6 ms 308 B

```
{
    "message": "album updated successfully."
}
```

I thoroughly tested my web application's backend using Postman. I sent HTTP requests to each API endpoint for managing albums, artists, and tracks. This included creating, updating, deleting, and retrieving data. For instance, endpoints like /api/albums, /api/artists, and /api/tracks were tested with different scenarios, both valid and invalid, to ensure everything worked as expected and errors were handled properly.

Justification of choices

The website was created using JavaScript, CSS, and HTML without using any frameworks. For the server-side operations, Express and NodeJS, and the database utilizes Better-SQLite3.