**General**

In this project you are to develop a single page that makes playlists available.

All data will be saved on a RESTful server which you will install; there must be options to add, edit, and remove all of the data.

The songs will be represented as URLs to external MP3 files, played via an HTML5 player.

The project must implement the following subjects:

* HTML + CSS:
  + New HTML5 tags
  + CSS3 media queries and advanced selectors
  + Dynamic page layouts
  + Bootstrap & Font Awesome
* JavaScript
  + Objects
  + Function Closures
  + jQuery
  + **S**ingle **P**age **A**pplication foundations
  + Events
  + Ajax (RESTful API)
  + Documentation

**Project details**

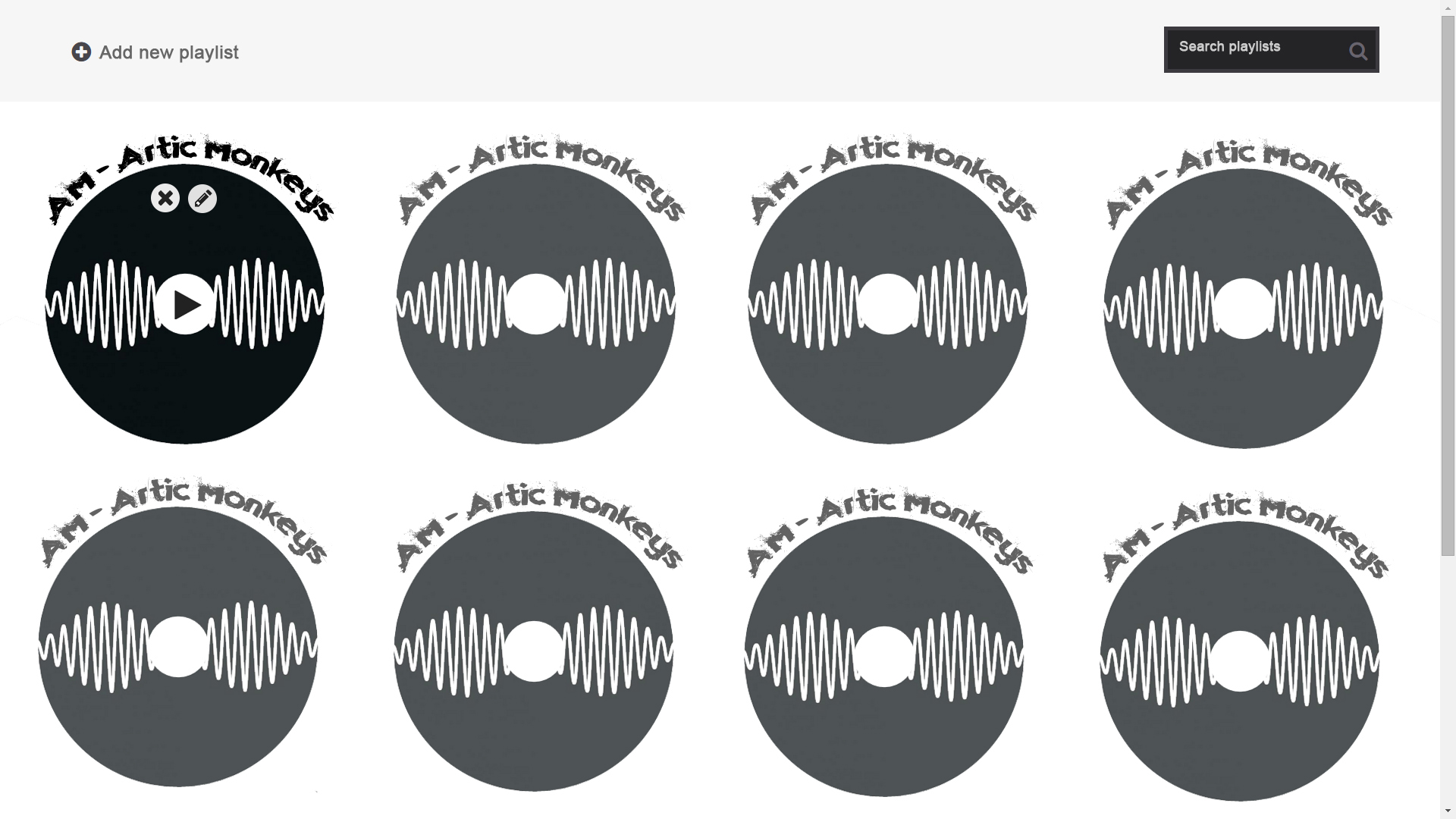
1. The page shows playlists in the shape of discs.
2. Every playlist has a name and a background image which can be edited. The image is an external URL to some image file.
3. Every playlist includes a list of songs which can be edited. The list will actually be an array of objects, each of which includes an external URL to an MP3 file and the name of the song.
4. When a playlist is clicked, the songs within it will be played using a simple HTML5 player; the background picture will spin like a disc. Once a song is finished, the next song will play.
5. There will also be an option to search for playlists by title through a search box. When more than two letters are typed into the search box, only the playlists with matching names will be displayed.
6. All of the data is stored on a RESTful server; all access, retrieval and editing of data will be done via an API. The API’s full documentation is given below.
7. The icons described are not images but characters made available via the Bootstrap framework, or using Font Awesome.
8. Use the jQuery framework as much as possible.
9. All user input must be validated using RegEx.

**Screens**

Mockups of the screens can be found in the Docs folder.

**Main screen**

The main screen is in fact a list of all the playlists in the system.



On top of the screen there is a menu including an “add new playlist” button and a search box which enables the filtering of playlists by title. This top section is fixed, so that scrolling the screen down does not affect it.

The playlists are displayed with their title over them (find a solution for writing the title in a curved fashion by using either CSS3, jQuery Plugin or both), and the linked image will be displayed as a circle (CSS3 border-radius).

All of the albums are at 0.85 opacity. On hover, the following will happen:

The transparency will be removed and the image will become fully opaque;

A play button will appear in the middle;

In the top part of the image, “edit” and “delete” buttons will appear.

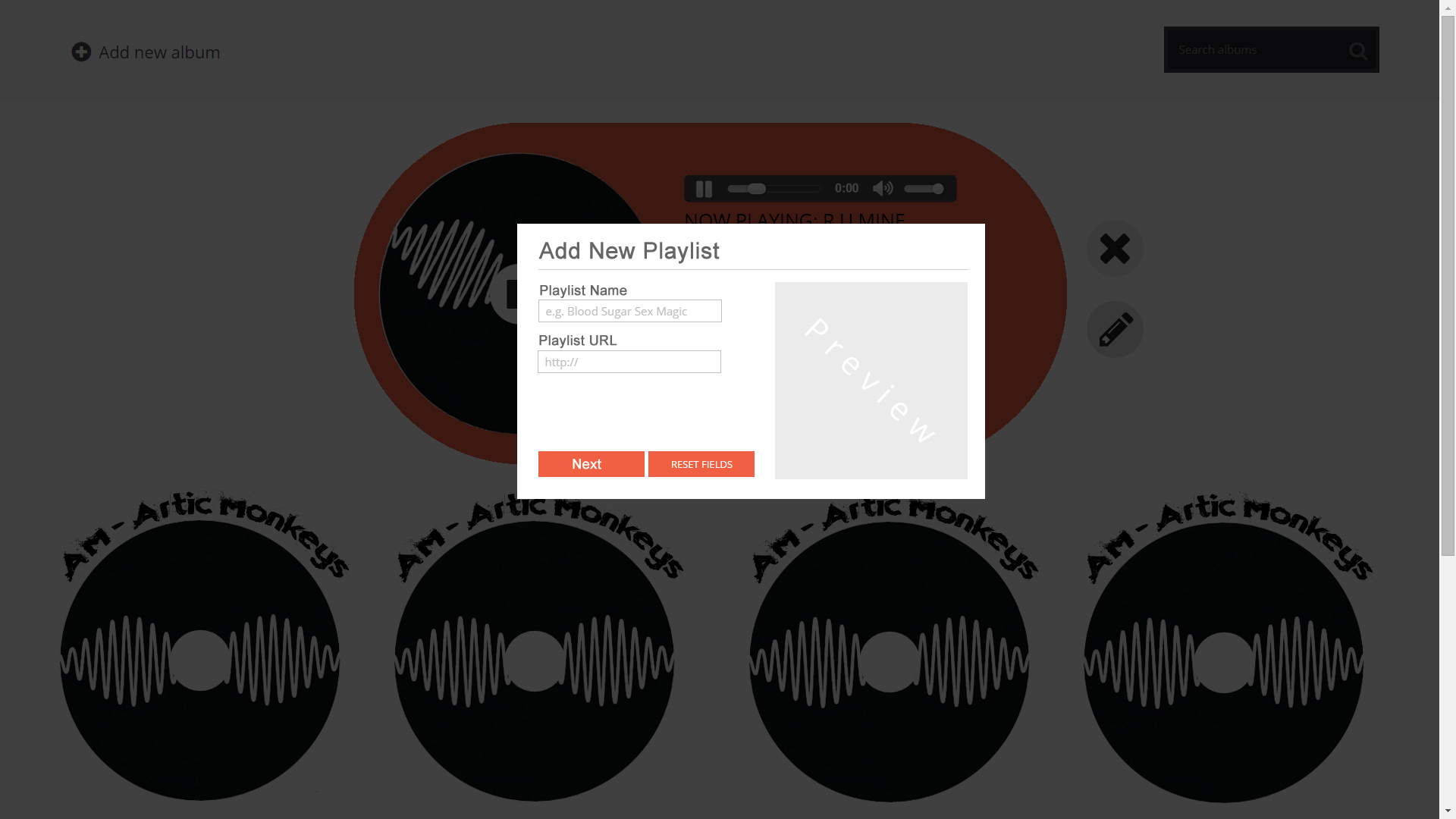
These changes will be implemented not with JavaScript but with CSS. In addition, as mentioned above, the buttons must be created using an external framework and not images.

When the “delete” button is pressed, the question “Are you sure?” will pop up in a modal window, as described further below, and after confirmation, the playlist will be deleted.

The next screens describe the actions of playing, editing, and adding playlists.

**Add and edit playlist screen**

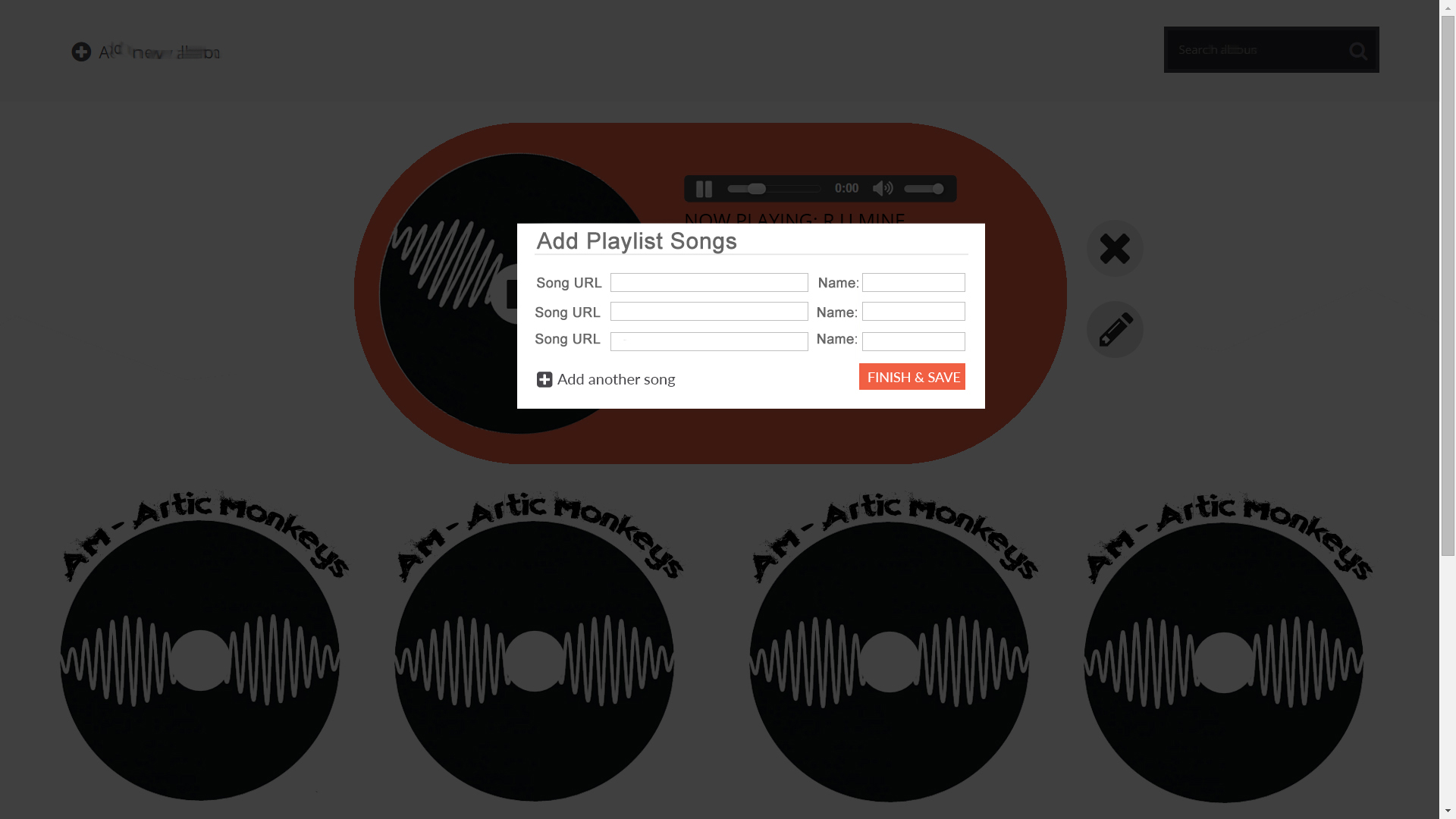
This screen will enable the user to add a new album to the collection and edit it. The title is shown here as “Add”, yet for editing, all of the existing data must be loaded and the notifications adjusted (“Edit” in the title, “Update” in the button etc.)



The entire screen is darkened (use position and opacity or a modal window from an external framework), and in the middle a dialog appears. In it, there are the following fields: album name and cover image (link to external image). This input must be validated using RegEx.

Immediately as the link to the image is typed in (when valid), the image must be displayed in its slot (where it says “Preview”).

When “Next” is clicked, if all data is valid, the dialog will be replaced by a new one, through which songs can be added to the playlist.



Every line represents one song: an external link to an MP3 file, and the song title. The link must end in “.mp3” and this must be validated using a regular expression. If the URL is invalid, then as soon as the user exits the field, it will be colored red (or a warning icon will appear, or any notification you choose), and it will not be possible to save the album.

Pressing the “+” icon will add fields for more songs per the user’s needs.

Optional: an Ajax test can be run on the file URL in order to ascertain that the header with the content-type indeed shows a sound file (audio/mpeg).

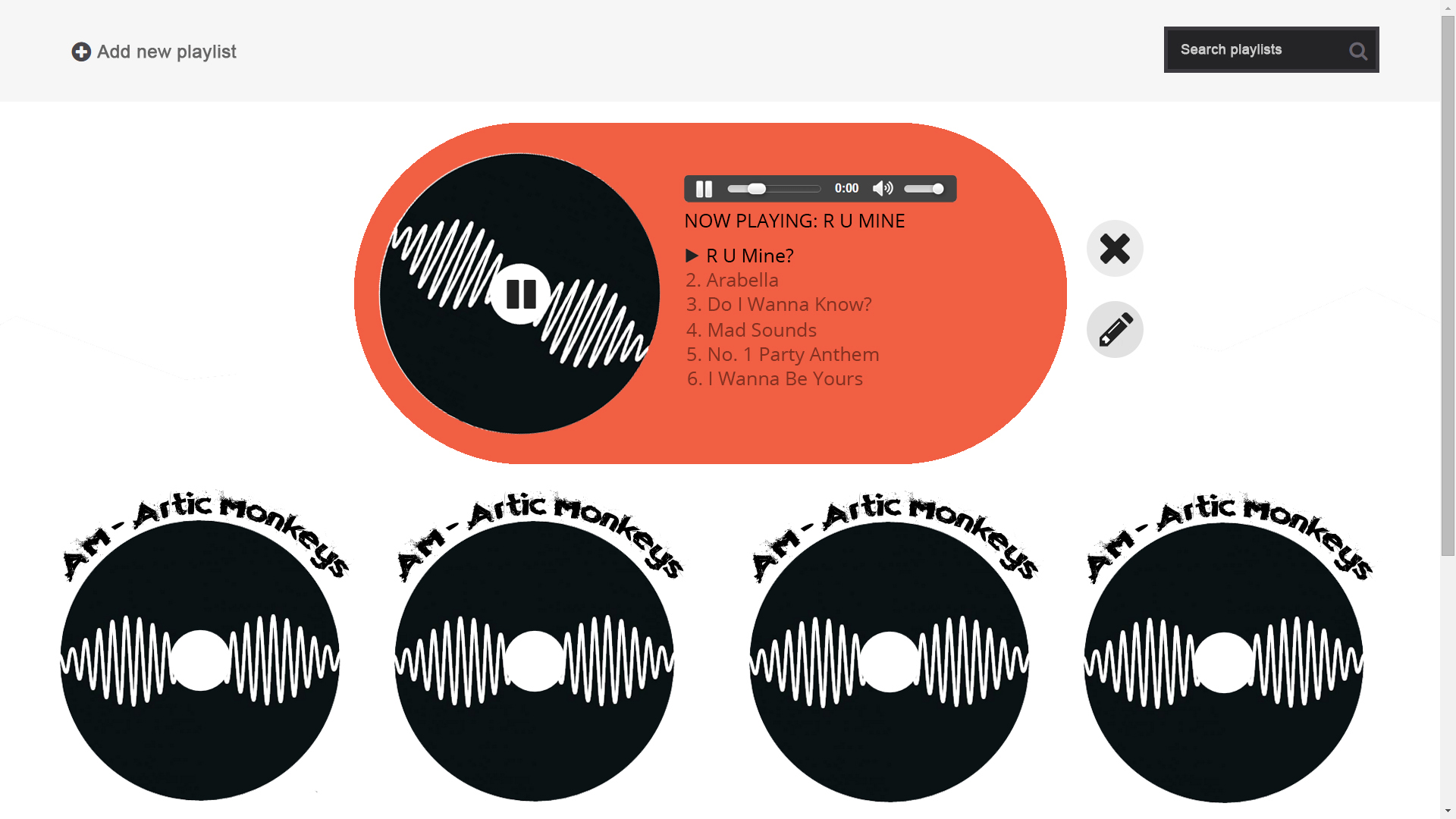
When “Save and finish” is pressed, the album will be saved on the server and once the server confirms it and gives it an ID, it will be added to the DOM.

Note: when editing, updates must be sent to the server at every stage using the correct request. Meanwhile when creating a new playlist, all data must be submitted via a single request.

**Player screen**

When the “Play” button in the album (main) screen is clicked, a player will appear in the upper section of the screen. The first song will begin playing. The player is also fixed so that it is not affected by scrolling.

In addition, the title of the page will change (that is, the <title> tag within the page <head>) to feature the name of the playlist and the title of the currently playing song.



The player shows the image chosen for the playlist on the left. When the songs are playing, the image spins like a disc (use CSS3 keyframes and transform. The icon in the middle will be replaced by a “Pause” icon. On the right of the disc the songs on the playlist will be listed; the first song will begin playing.

The player is very simple: using the <audio> tag in HTML5 enables an MP3 song to be played and you must only replace the “src” attribute of the tag. Find out how to create a response to the event of the song ending in order to go to the next song.

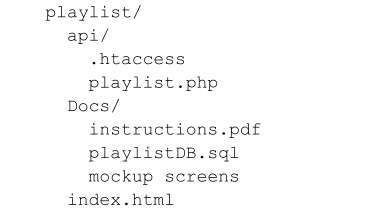
All song titles will be transparent (opacity by default), except the currently playing song. On hover over any song, a “Play” icon will be displayed next to it, and if that is clicked, the song will start playing and become opaque, not transparent.

On hover over the currently playing song, the “Play” icon will be replaced with “Pause”.s

The name of the currently playing song will appear underneath the player. Next to the player, two icons will be displayed: “Edit” and “Delete”. When clicked, they will bring up the editing and deleting screens, as described above, and if the change is applied, the player will reflect it.

**Installation**

In order to run the project, install an environment that supports PHP and MySQL, like XAMPP or WAMP. The project’s folder will be organized as follows:



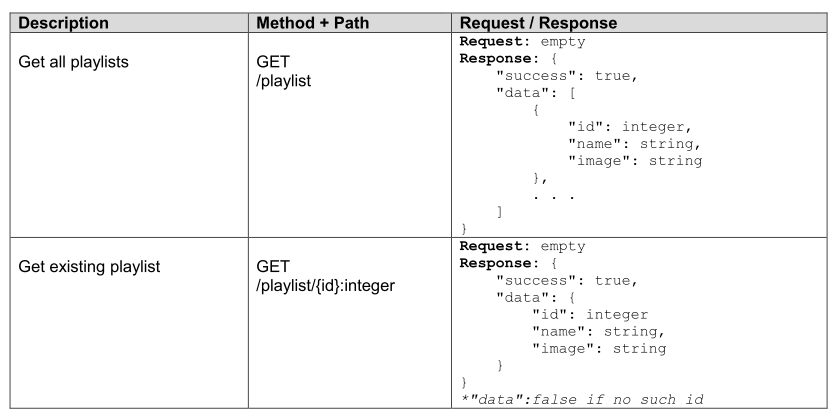
The “playlist” folder must be placed directly in the root folder, like “htdocs”.

Create a new DB schema using the playlistDB.sql file found in the Docs folder.

The main screen of the project will be index.html, found directly in the “playlist” folder.

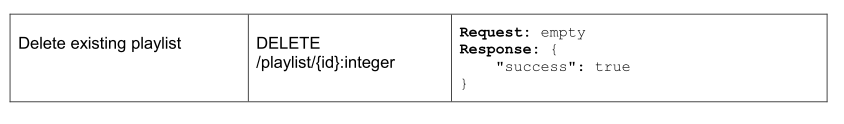
**API Docs**

Endpoint: [**http://localhost/playlist/api/**](http://localhost/playlist/api/)









**Response header codes:**

* 200: Success
* 400: Bad request
* 404: Bad URL / Page not found
* 422: DB Error

**Links**

* Icons for buttons from Font Awesome <http://fontawesome.io/icons>
* <audio> tag in HTML5 <http://www.w3schools.com/tags/ref_av_dom.asp>