**The Discography Client Interface**

1. Search for the albums released in 1977:
   1. How does the interface change when you select Search?

It adds a “Search by” selection input, a “Search Term” input, and a “Search” button.

* 1. How many items are in the table after the search completes?

10

* 1. What happens to the table if your search field is empty?

Shows all discography and says “Nothing matches your search criteria”.

1. In src\public\discography.css, explain what the following blocks of code are doing AND on what elements they are applied to:
   1. The table thead tr th block

It is applying the specified styles (“top: 0” and “position: sticky”) to the applicable elements, which are the table headers (“Album”, “Artist”, and “Year”).

* 1. The .height-limit block

It is applying the specified styles (“height: 400px” and “overflow-y: auto”) to the applicable elements, which is the “div” element that contains the table.

* 1. The .header block

It is applying the specified styles (“display: flex” and “align-items: center”) to the applicable elements, which is the “div” element that contains the image and heading.

1. Notice how the table header behaves as you scroll up and down:
   1. What CSS property causes this to happen?

The “position: sticky” property applied to the table headers.

* 1. How does the property work?

Per [W3Schools](https://www.w3schools.com/css/css_positioning.asp): “A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).”

**The Discography Client Code**

1. In src\public\discography.js:
   1. Why do we need to assign a function to window.onload?

To make sure the document has fully loaded before adding the event handlers.

* 1. What the function doing?

Adding event handlers to the various inputs.

* 1. What would happen if the function was not there?

The event handlers could potentially be attempted to be added to elements that haven’t yet loaded, which would cause an error.

1. In src\public\discography.js, in the addAlbum function, from where is the script getting the values for the new entry?

From hidden input fields.

1. In src\public\discography.js, in the search function:
   1. Explain what document.getElementById("searchType").value; does. How does it know which value to return?

It gets the value from the “Search by” selection input. It knows which value to return based on which option was selected.

* 1. When we searched for albums released in 1977, what was the value stored in requestData?

“year=1977”

* 1. What is done with that value when the GET request is called?

It is added as a query parameter to the fetch request.

* 1. How is the appropriate server end-point selected?

It uses the “searchType” value, such as “year” in the 1977 example.

**The Discography Server Code**

In src\directoryServer.mjs:

1. **Explain what the code is doing on lines 10-13.**

On line 10, we are initializing a new instance of the express library as a variable called app. Next, we are using that app to listen for any HTTP requests on our local computer on port 3000. This will set up a server that is running locally on our computer and can only be accessed by us. Once a connection is found, we print out to the console that we are connected.

1. **Explain what the code is doing on lines 14-15.**

On line 14, we are looking into the directory root "public" and getting the "discography.html" to use. On line 15, we allow the interpretation of a json file that we were able to retrieve.

1. **Explain what the code is doing on lines 17-25.**

On line 19, we are connecting to the MongoDB server database that is labeled "discography". If the connection between the client and the server fails, we catch the error and print out the error to the server's console. After the error is logged, an exit command is used to terminate any running code or any code that will run in the future. On line 24, we create a new schema with a title, an artist, and a year. On line 25, we create a model, which takes in the schema we made before and labels it as 'albums'.

**Error Handling and Security**

1. **From what you discovered provide a set of steps that a malicious user could do to store (add) an album to the database that when loaded on another user's computer could execute a cross site scripting attack.**
   1. A screenshot of a computer

      Description automatically generatedAn XSS attack that is able to be performed on the website is a simple JavaScript injection. In this example, an alert will pop up when a user hovers over the label.
   2. In replace for the alert pop up, a script could be injected into the database that could redirect the user to a site that could inject/download something malicious onto the user’s computer.

**Suggestions and/or Notes**

I enjoyed figuring out how the code worked.

Overall, I enjoyed this assignment, as it gave a great introduction to MongoDB along with managing databases. One main thing I enjoyed was learning the inner workings of MongoDB and using Mongosh to add/delete items in the database.