COMP3000

Full Stack Development

2020/2021

Sports Collection Search – Tayla Harding

**Source code**: https://github.com/ProTJ98/Sports-Card-Application

**Whom is my Application is for?**

The Sportscard industry Is enormous in the USA and has now started catching on to other countries. For years baseball, soccer, football cards have taken the states by storm, and now the rest of the world is catching on. Within his collecting world, there is no application in which sports collectors have searched and created their own virtual personal collections. There are currently many forums, discords, and selling sites out there, but there are many complications within these. One of these complications includes sports card scammers. The application is designed to make the sports card collecting world a safer and better place to do business within an easier way to story cards. Having all of the projected features for the application will make the application the first of its kind and pave the industry's footsteps.

Currently, there are no applications/websites that include some of the main features needed for this sort of industry. There are many applications out there in which you can buy/sell/trade, but this also ties into the security side in which scammers are still out there. There are also other websites for investing and virtual collection books. The is not an application that incorporates all of the features together.

**Project initiation and features:**

**Main Features:**

Database tool for sports cards:

The application's central concept is for all sports cards collectors to use, although with there being so many different sports with many different cards, most of my data in my database are soccer. The database has a base set of players from Liverpool with some variants of the cards. For example, I have set up an Alisson page that shows all of his Topps Finest cards.

**Checklist database function:**

When a database has been made, users will be able to implement their current collections by adding to their own "virtual binder". The virtual binder feature allows users to access their collections wherever they are, but also users will be able to upload pictures of their cards to their virtual binder. Other applications allow you to view a stock photo of the card and what it looks like. The idea of users uploading their card gives a more detailed look at the card. For example, if a user wanted to sell that card, they can instantly list it with the pictures already on the application. Many different cards come in different shapes and sizes, and most importantly, the condition. Some cards might have better corners, surfaces than other cards, for a collector this can be a massive requirement for a card for their collection. Within the database checklist, users can search for cards they are missing, and the database will check if that card is currently on the market. The Checklist will be linked to MongoDB. The user can request information from the cloud database and pull all of that information for that card already stored. As there are so many different Variants of the cards each card has a default set of information, then using routing will grab each card set. Using an eBay API, this was done by linking 130 point data and requesting it into my server. 130 point is a site that tracks all eBay sales. A Below Image can see if being worked within my application:

**Price comparison:**

Sportscard go up and down all of the time, so if I am a user of the application and I want to buy a card, I want at the lowest price and also make sure I get the best for my money. This is where the price comparison tool comes in. Users can search for previous sales of cards to ensure they are getting the right price for their card.

**Personal binders:**

Users of the application have a personal profile. The Database stores that individual's collection in a text file. The text File has the basis of the cards and then links back to the database to retrieve that information and shows the user linking to the EJS file.

**How is my application structured?**

The First Diagram shows how the Username and registration page works. When a user signs up to the website, it will send the data to the mongo dB with the user credentials and update it. It will also use Bcrypt to change the password from a text file into hash. Using that same data, the user can use the login page. The login page goes through the same process in which the user enters there credentials while MongoDB validates them and if they are correct, the user will be logged to there dashboard. If the validation check fails, it will ask the user to re-enter the details and try again.

The Second Diagram is a complex but simple design. When the user wants to search for a card, they can search for different subsets. The Card search database uses the following subsets for the search:

* Player Attributes
* Publisher
* Set
* Variation

Within these subsets, the user can search for specific terms. All of these terms are linked together to once specific card. The Search results keep getting narrowed down until the user has found the card they want. If the search matches the database, it will be pulled, and the card page will be loaded to the user. If there is no such card, the database will return a message saying no card is found.

My code is structured in a way in which everything links together but different variables. For example, using a basic route for the Topps chrome set and adding the attributes from the database for that player's specific card. This made the development process so much easier to do. Request a set of data that is already implemented and then using that data in a different subset of a card makes the code more efficient and simplistic.

Diagram

Description automatically generated

**Testing:**

Test Table:

Please Refer to Testing.xlsx file for the test sheet stored in the program files.

Testing my application at different stages was vital. When developing the EJS and HTML side of my application, I had to test my front end more than usual. Because I was using a bootstrap template but also developing it from scratch. For example, the Profile page was built entirely from scratch and only using the circular template for the profile. Regarding the back end, I was using the following command:

Npm run dev

This command made it a lot easier to test and run. Instead of starting my application by restarting it every time this automatically kept all of the files up to date before starting the server. This made it a lot easier to keep up to date with any errors that might have appeared in the code.

**When I tested:**

Testing the application was difficult due to the complexity of my program. I imputed error checking into my application, for example:

value="<%= typeof password2 != 'undefined' ? password2 : '' %>"

(Register.ejs Harding,2020)

Splitting my code into different sectors made it a lot easier to manage as well. If I was block coding, meaning that I was updating or writing a block of code simultaneously, using comments to split it down made it easier to track a problem should one occur.

Using built-in error checking makes the application easier to program and a lot more user friendly. So the above code checkers whether the user has entered data into all of the subjects. If not, it will throw in error.

**DevOps pipeline:**

I started my application by building the front end. This would give me a basis to make it a lot easier to develop the back end. I started by using a simple bootstrap layout in order to speed things up a little bit. This made me a simplistic HMTL boilerplate with some CSS that I can change and adapt to my project. Using this, it took me around one month to get to a stage where the back end could be started. The struggled mentioned before it was hard to work within a simplistic pipeline, due to not having access to MongoDB in the library, which is why I started with the application's HTML side. The website was pushed to GitHub every hour and automatically backed up on the NAS server. Due to the library and not being able to access GitHub desktop I had to upload the HTML folders. This is a terrible way to manage a project, but this was my only option due to lack of resources. The link to Github can be found here of the progress-

Once the front end was done, I start on the back end so installing mongo, NodeJS and VS code. I used brackets for my HTML and Visual studio code for my back end development. There were a few combability issues along the way as well, which were solved with a VM. The backend was pushed to Git every 30 minutes due to the nature of the code. I started with the login page and using localhost on port 5000. Once the login page was fully working, I moved to atlass and using postman. I then recycled some of the code used for the login page for the Database of the cards. The Database had the following Values for each card.

Player Name

Publisher

Set

Year

Position

Card Variant

Team

I also added some mini features such as the database search for Soccer Players and Basketball players as you can see below. This was done with some JS but was intended to link to mongo to give it a more user-friendly interface. I was continuously testing along the way to make sure that bugs did not occur.

Personal reflections:

Overall the project went well until the recently when my program corrupted on my NAS storage. Within Covid-19, I struggled to work from home due to having an older iMac which causes combability issues and spending time on workarounds. When possible, I went to the university library, but due to firewall/installation restrictions, this limited me a lot. This means that I had to write the code pre and then test it when at home, even when I could and there weren't any combability issues. It was a roller coast with NAS storage getting corrupted, not having the right hardware, University Closures due to COVID. My project was definitely more about adapting and overcoming finding solutions that would not have been there if COVID was not a factor.

I enjoyed learning about MongoDB as it opened my eyes to a more simple form factor if it wasn't for the issues I had it would have been nice to explore more of it. Before the issues, I managed to do my research in getting my Database on the cloud, but mongo opens up so much more possibilities to the database structure.

**Video of my application:**

<https://www.youtube.com/watch?v=lpyYagzrHUE> (Private can only be opened with a link)