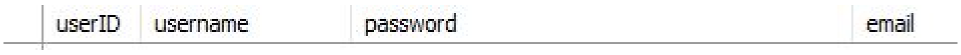
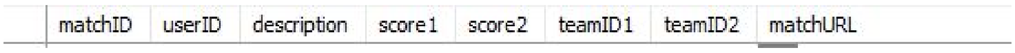
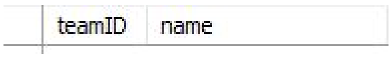
Dylan Clarry 160718680
Peter Mykulak 160635980

Football Library Application(FLA)
Documentation

Functionalities and Implementation

- The app consists of the user logging in or signing up. The user will then have the opportunity to view, add, edit and delete information.
- The app connects to a MYSQL database that is connected to Heroku's ClearDB MySQL platform. Heroku hosts all of FLA's source code as well.
    - users table:
        - 

userID	username	password	email
    - matches table:
        - 

matchID	userID	description	score1	score2	teamID1	teamID2	matchURL
    - teams table
        - 

teamID	name

Functionality	**Implementation**
Single Page Routing	Building the website as a Single Page Application (SPA) Gave our app a very snappy, desktop app-like feel that also integrated well with the REST API we built on the backend. To implement this feature we took advantage of the pound sign (#) dead link. Any link with a pound sign following its url will ignore whatever follows it. By adding a hashchange listener to the page to detect when a link with a pound sign has been clicked we were able to find the correct path from the url and load the corresponding page without making a new HTTP request.
JSON Web Tokens	The backend checks authorization using JWT's. In order to keep track of this, JWT's were stored in local storage and retrieved whenever an API call was made. All JWT's are time stamped and given a time limit before they need to be reset.
AJAX	AJAX was implemented using the JQuery library. AJAX made it possible to make asynchronous calls to our API and create, edit,

	read, and delete data dynamically without page reloads.
Main page	The client will make a call to the backend using https://footlib-backend.herokuapp.com/matches This returns a JSON list that contains information on all matches. The client will then populate the collection of match information on the main page.These matches are matches that the users have created.
Sign Up page	When the information is entered, the client will make a call to https://footlib-backend.herokuapp.com/signup with the credentials in the relevant request body. The information must be unique and not match with anything in the user database. If this is true the user will be added to the database with the provided information as his credentials. The password provided by the user will then be hashed and stored in the **users** table. The password hashing is done with BCRYPT.
Login Page	When the information is entered, the client will make a call to https://footlib-backend.herokuapp.com/login with the credentials in the relevant request body. If the username exists within the **users** table the provided password is then compared to the hashed password with BCRYPT. If the comparison is successful then a bearer token is created for future JWT authentication
Dashboard Page create	Client has the option to create a match entry by calling https://footlib-backend.herokuapp.com/matches/:userID/create The request body will contain all relevant information to create said match and the match record will be added to the matches table.
Dashboard Page edit	Client has the option to update a match record https://footlib-backend.herokuapp.com/matches/:userID/update/:matchID The request body will contain the name of the column we want to update along with the replacement value for said column. A matches table row with the matching matchID and

	userID will be edited.
Dashboard Page delete	The client has the option to delete a record using https://footlib-backend.herokuapp.com/matches/:userID/delete/:matchID This will delete the entire row that contains matchID in the matches table.

Technologies

Technologies	**Reasons and implementations**
Node.JS	● Allows for event driven development ● Easily scalable ● Utilizes javascript, therefore both project members are very familiar with the language ● Allows for easy generation of dynamic content ● Large amount of modules to help your application
Express.JS	● Modular web framework and very lightweight ● Easy templating ● Effective error handling ● Flexible routing system
Javascript	● Supported across all browsers ● JSDOM makes it possible to build single page application routing without an external framework ● ES6+ features help with modularization of code and creating pure functions whenever possible
JQuery	● Provides easy to use solutions without the need of a full framework ● Used for AJAX features ● First time using JQuery (outside of class)
Sass	● Used to modularize CSS code for different pages and components ● Gives more structure than normal CSS ● Easier to find and troubleshoot broken code ● Makes it easier to work with

	Javascript ● Minimizes all CSS code into one CSS file ● Was very familiar with Sass from prior experience
Browserify	● Used to modularize Javascript for different pages and components ● Gives more structure to normal Javascript ● Easier to build pages and components ● Provides error mapping to troubleshoot errors ● Makes it easier to work with Sass/CSS ● Browserify was the first time using a bundler

Dylan Clarry's Take on the Technologies:

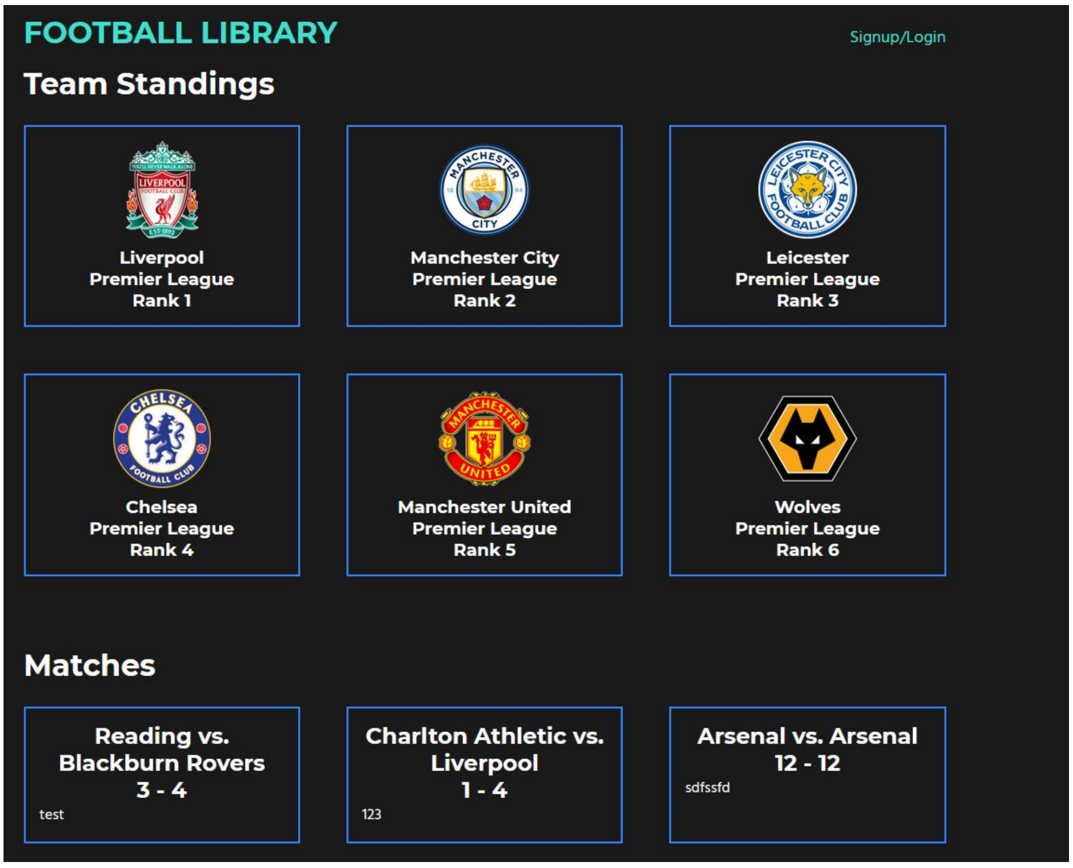
The main reason and justification for choosing the technologies that we did was to keep our code modularized and decoupled. By keeping these principles in mind it made it much easier to build and maintain our code as well as working together remotely. Even though the Covid-19 pandemic was not anticipated and was expected to create major problems with our workflow as a team it actually did not have much impact. The reason for this is that because we chose to completely separate the frontend from the backend we were able to both work at the same time on separate components. For example, the frontend of the site was able to be built independently from the REST API and was tested using fake data from a .json file. Then, once the backend was ready to be hosted it was as simple as getting data from the API instead of the .json file.

Also, because we both had a much deeper understanding of our ends of the stack it was very easy to troubleshoot problems. I found that whenever I got a response from the API that did not seem right I could simply describe the issue to my partner and he could always solve the problem immediately. I feel as though this would not have been the case if we took a different approach and chose different technologies.

Peter Mykulak's Take on the Technologies:

Application development with regards to using node.js and the express.js framework was a new endeavour for me. Other than some practice for university using javascript i have never really worked with node.js in such an intimate manner before. I decided to use node and express because my partner in this project recommended it to me. Learning it was quite easy though as there are a ton of educational resources on youtube. One major issue I had was getting the MySQL database to run reliably with our hosting platform. Our hosting platform is Heroku.

Instructions for application use:

1. To access our application, please visit (https://footballlibrary.herokuapp.com)
2. On the main page, you will be able to see a variety of matches that the app is currently tracking.
    a. 

FOOTBALL LIBRARY

Signup/Login

Team Standings

Liverpool
Premier League
Rank 1

Manchester City
Premier League
Rank 2

Leicester
Premier League
Rank 3

Chelsea
Premier League
Rank 4

Manchester United
Premier League
Rank 5

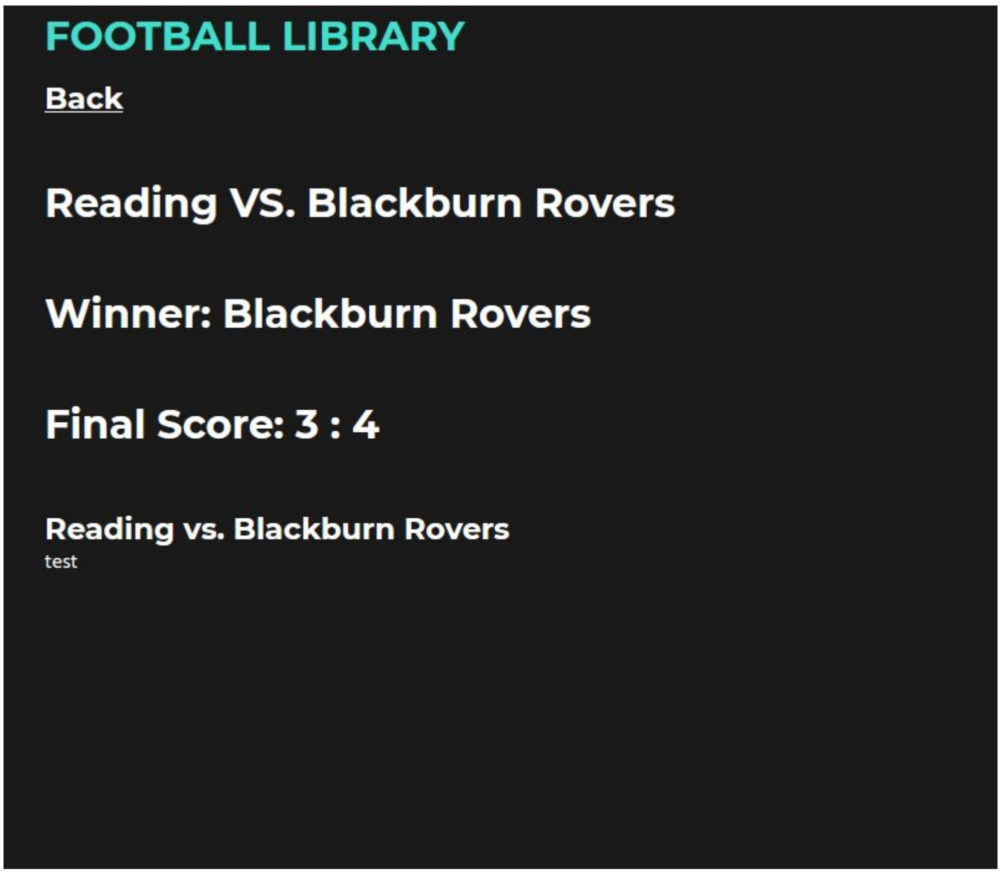
Wolves
Premier League
Rank 6

Matches

Reading vs.
Blackburn Rovers
3 - 4
test

Charlton Athletic vs.
Liverpool
1 - 4
123

Arsenal vs. Arsenal
12 - 12
sdfssfd

3. You can click on any one of these matches to get more information on them.
   a. 
      FOOTBALL LIBRARY

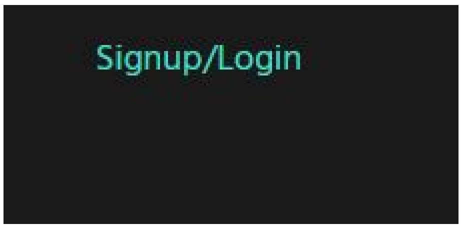
      Back

      Reading VS. Blackburn Rovers

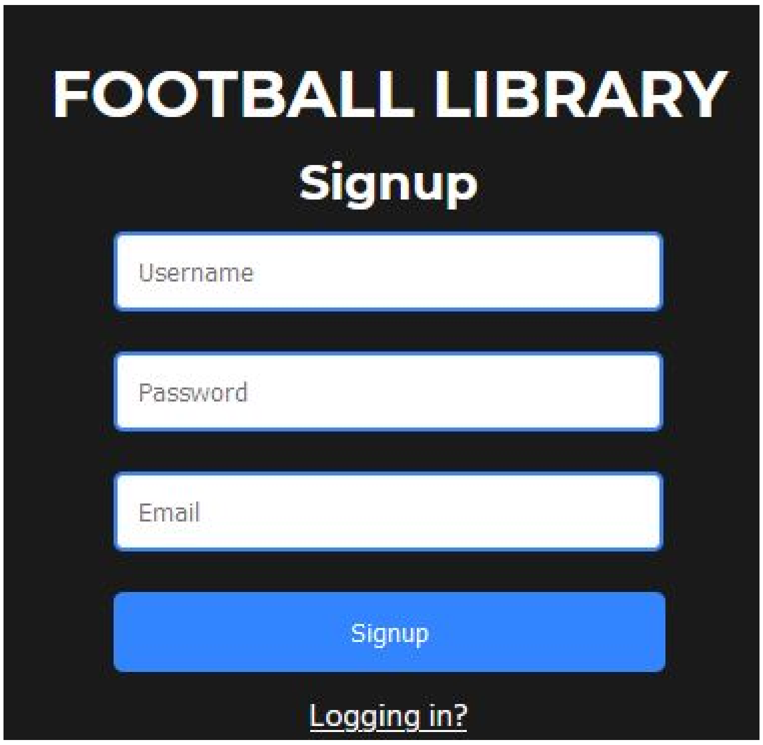
      Winner: Blackburn Rovers

      Final Score: 3 : 4

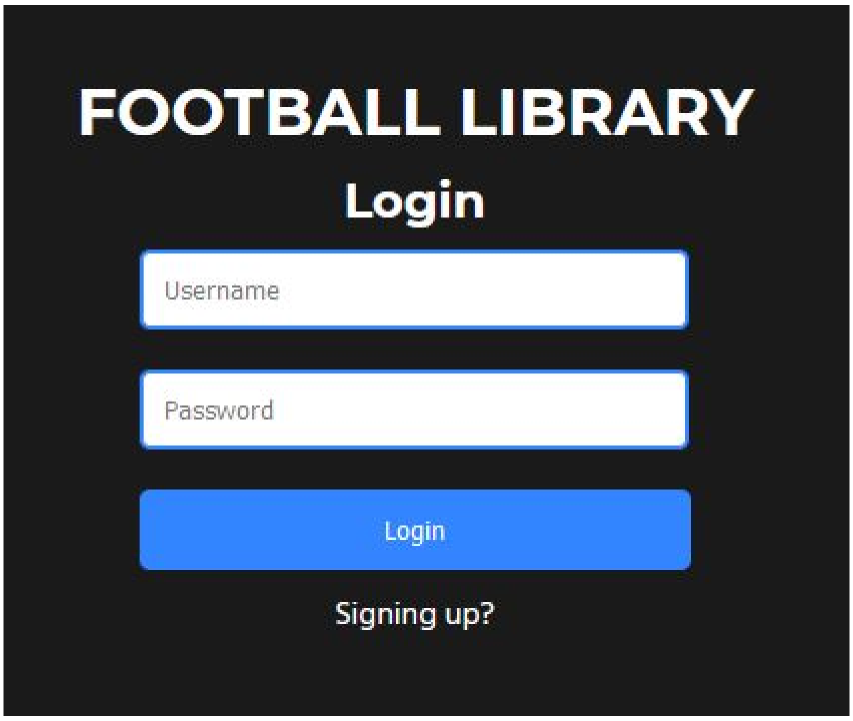
      Reading vs. Blackburn Rovers

      test
4. On the top right of the screen you will see a login/signup button
   a. 

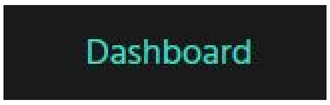
5. After you click said button, you will be brought to the signup screen.
   a. If you have an account you can log in with your current credentials

i. 

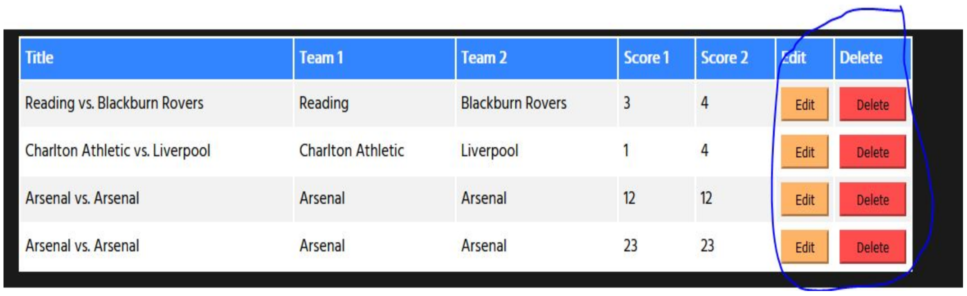
b. If you do not have an account please choose unique credentials

i. 

6. After you are logged in you can see a dashboard button on the top right of your screen, click this button

a. 

7. On the dashboard menu you will see a table of your current matches tied to your account.
    a. You can update, delete, or add new matches.



Title	Team 1	Team 2	Score 1	Score 2	Edit	Delete
Reading vs. Blackburn Rovers	Reading	Blackburn Rovers	3	4	Edit	Delete
Charlton Athletic vs. Liverpool	Charlton Athletic	Liverpool	1	4	Edit	Delete
Arsenal vs. Arsenal	Arsenal	Arsenal	12	12	Edit	Delete
Arsenal vs. Arsenal	Arsenal	Arsenal	23	23	Edit	Delete

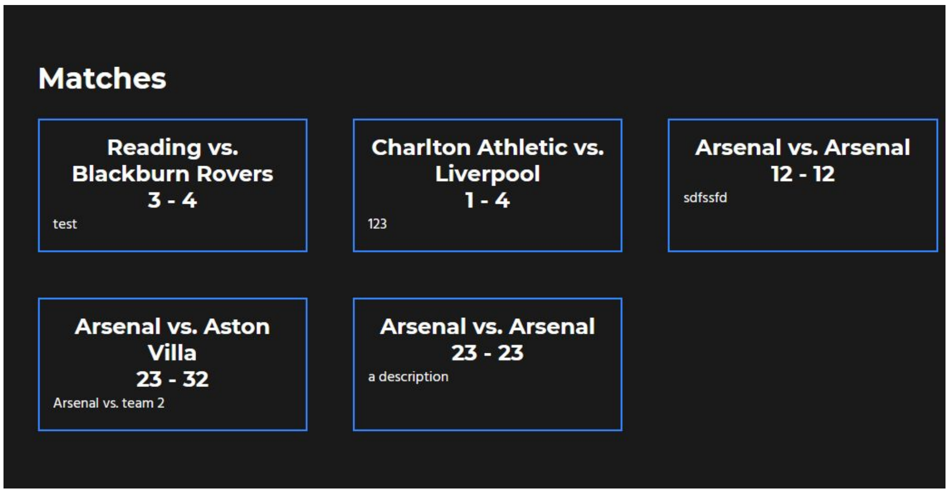
i.

ii.

8. To return to the main page please click on the Football Library Logo

a. FOOTBALL LIBRARY

9. This will take you back to the dashboard where you will see your newly added, or edited matches.

a. 

Matches

Reading vs. Blackburn Rovers
3 - 4
test

Charlton Athletic vs. Liverpool
1 - 4
123

Arsenal vs. Arsenal
12 - 12
sdfssfd

Arsenal vs. Aston Villa
23 - 32
Arsenal vs. team 2

Arsenal vs. Arsenal
23 - 23
a description

Challenges Faced and Lessons Learned

We learned several lessons during this project. Our main issue with this project was server reliability in conjunction with Heroku hosting. Since we chose MySQL for our database platform, we ran into some issues where the MySQL connection would timeout and crash the backend server every 30 seconds when connected to the heroku app. We decided to fix this by using pooled connections, opening up a connection as we return an api call and access the database, then close the connection as the response gets sent out. This allows us to be within the 30 second timeout window and allow our backend heroku server to become stable. This was easily the most time consuming part of the project when implementing the backend resources.

In terms of the frontend of the app, the biggest challenge faced was with JSON Web Tokens. The difficulty came from mismanaging the tokens as well as not initially setting up error handling from the database crashing and reissuing tokens. The work around for this was to set timestamps on tokens when they are created and to make sure the user's browser is only holding one token at a time. This was definitely the most time consuming part of the frontend. We learned that managing sessions and authentications is much more difficult in an environment where the frontend and backend are separated.

The final big issue we came across was that the Covid-19 pandemic affected our use of our initial third party API. The original intention with the API was to get data on upcoming events in the league and live scores to show to the user. Unfortunately, due to the pandemic there aren't any leagues currently in play so the API was no longer sending us data to use for these features. We ended up settling on using the API-Football API to display teams standing in the league before the pandemic. This was one of our main features so we needed to improvise a bit and find a new feature to build at the last minute.

A website that has a similar idea to ours but one that is much more enhanced is https://www.premierleague.com/stats
This application allows users to review live statistics on everything related to the premier league. This includes, match statistics, player statistics and team statistics. Another very amazing functionality this site has is the ability to do direct comparisons between different teams, players and game seasons. On the opposite spectrum we have our own website. This allows users to record their own matches they see themselves, not pull from a live database like the aforementioned website. We also allow users to create their own matches to add to their account so they can have direct access to track them in the future.

In terms of expanding our project and including more features, using a third party api to allow users to search and pick live matches would have been a great addition. Instead of having them create their own matches to track, they could have just clicked a save icon and tied a match that was provided by a third party API to their own account.