WrittenDocumentation

What is the purpose of your application? What does it do?

My application is a simple, old-school styled champion composition creator/crafter for the game *League of Legends*. It allows the user to create and save team compositions consisting of one champion of their choosing for each of the five roles of the game (top, jungle, mid, bot, support). They can view and delete these comps from their account screen.

How are you using React?

I am using react for all of the views within the app in order to fuel the UI.

What components do you have?

I have the following components across different .JSX files:

- TeamsList (account.jsx)
- PremiumSubscriptionWindow (account.jsx)
- CancelPremiumSubscriptionWindow (account.jsx)
- ChangePasswordWindow (account.jsx)
- TeamsViewer (account.jsx)
- PremiumViewer (account.jsx)
- LoginWindow (login.jsx)
- SignupWindow (login.jsx)
- TeamCompViewerNavigation (app.jsx)
- TeamCreationForm (app.jsx)
- App (app.jsx)

What data are you storing in MongoDB?

I am storing account information (username, password, id, whether or not they have premium) and team information (name, top champion, jungle champion, mid champion, bot champion, support champion).

What went right in the development of this project?

To be honest, not much. I think the thing that really went ‘right’ was the speed at which I was able to complete it, which I will get into more in the what went wrong section. I understood the concepts and code very well, at least I think I do, you can be the judge of that. I was able to easily solve any errors I stumbled upon and overall spent most of my time in active development, not bug fixing.

What went wrong in the development of this project?

That is about where the good ends and the bad begins. I messed up with this project, big time. I believe I have reached a point of no return when it comes to procrastination. I literally submitted this at 11:59pm, the same 60 second window in which it was due. I thought it would be easy and not a big deal, so I kept pushing it off more and more and more, until it was due tomorrow. Even then, I only worked on it somewhat and didn’t fully commit until the afternoon of the day it was due. Clearly, this was not a good choice. I have struggled with this problem of procrastination for about my entire life, especially in school. I don’t know why I can’t learn from the endless number of times I have done basically this exact same thing, but I don’t. However, this time is different. This time, I really messed up. This time I had to miss my friends going away celebration, before she moves to NYC for our last semester of school, because I procrastinated this project to the point where I *had* to work on it during that time or I wouldn't get it done. I was seriously questioning just not doing it and taking the D in this class, but I couldn't let my mom down like that. I need to seriously rethink how I prioritize tasks, work, and really any kind of time management. I learned how poor my time management and relative work estimation really is with this experience, and I will not let it continue that way. To be honest, I don't think this project is very good. I originally wanted it to be a portfolio piece, which I may still go back and improve it to do so, but in its current state it has no business being on a portfolio. It works without errors and the core functionality is technically there, but the UI, feedback, look and feel, and just overall user experience are heavily lacking. It's ugly, confusing, and barely feels like a final project. I am embarrassed to be submitting this to you, I know I can do better. Other than that, pretty much everything went smoothly, I just ran out of time because I am a moron.

What did you learn while developing this project?

I learned a lot while developing this, actually. I won't go over the life lessons and high level things that I learned about myself and my time management, because I kind of already did. Instead, I'll talk about how much I learned about React, Node, MongoDB, and MVC server structure. To be honest, I didn't really understand these concepts before this project. I know how they worked and the general idea behind them, but I never really felt comfortable coding with them in mind nor did I think I could build an app with these tools from scratch. Now I think I can. I understand the different functions and components required for an MVC to function, I actually know how React works now instead of just trying to guesstimate the syntax to a video, and I feel confident in my skills to do so. I really took the time to understand and learn what the code pieces in Domomaker actually did and how to use them in my project. I am very glad I did this project, even if I am not happy with the end result, because I think it forced me to learn the concepts that this class was really trying to teach. It helped to solidify those concepts in my mind instead of me just using my short term memory to complete smaller assignments. I feel that if someone asked me questions about any of the concepts within this project, I would be able to answer them with confidence and ease.

If you were to continue, what would you do to improve your application?

The main thing to improve is the user experience. Not just the UI, which is atrociously ugly (although that was intentional in a way, because I wanted it to be reminiscent of ancient League websites that were hideous), but the general feel of the site. There are moments where the user has no feedback (mainly when creating a team comp), that need to be fixed. There isn't much overall juice to the site when navigating and it's a bit confusing to know where everything is or what it does. Certain navigation buttons do not go away when you are on their 'tab'. That is generally what I would improve, but the one thing I specifically want to add is the ability to 'compare' team comps to each other. This was originally what I wanted to do for 'above and beyond' work, but clearly I did not get to it in time. This is the main feature that any League player would actually want to use if this were a real website because it would give them a generalized probability of their chances to win as either team comp. I would base this algorithm on a lot of information surrounding the game that I can pull from Riot Games' API (champion win rates in specific lanes, match-up win rates, etc.). I think this feature would be really cool and legitimately helpful to League players.

If you went above and beyond, how did you do so?

I definitely did not go above and beyond, sorry :(.

If you used any borrowed code or code fragments, where did you get them from?

I didn't borrow any code from anywhere other than Domomaker.

What do the code fragments do? Where are they in your code?

N/A

Endpoint Documentation

/getChampionData
Supported Methods: GET, HEAD
Middleware: Requires Login
Query Params: None
Description: Retrieves all the champion data contained in server/controllers/champion.json
Return Type: JSON

/getTeams
Supported Methods: GET, HEAD
Middleware: Requires Login
Query Params: owner (the id of the owner of the requested team)
Description: Retrieves all the docs of the teams collection for the specified user
Return Type: JSON

/login
Supported Methods:
Middleware: Requires Secure, Requires Logout
Query Params: GET, HEAD
Description: Renders the login page
Return Type: HTML

/
Supported Methods: GET, HEAD
Middleware: Requires Secure, Requires Logout
Query Params: None
Description: Renders the login page (exact same as the GET method /login)
Return Type: HTML

/login
Supported Methods: POST, HEAD
Middleware: Requires Secure, Requires Logout
Body Params: username (the user's username), password (users password)
Description: Logs the user into their account
Return Type: JSON

/signup
Supported Methods: POST, HEAD
Middleware: Requires Secure, Requires Logout
Body Params: username, password, password2 (the users password retyped)
Description: Signs the user up for a new account
Return Type: JSON

/logout

Supported Methods: GET, HEAD
Middleware: Requires Login
Query Params: None
Description: Logs the user out of their account
Return Type: Redirect

/teamMaker
Supported Methods: GET, HEAD
Middleware: Requires Login
Query Params: None
Description: Renders the team maker page for the user
Return Type: HTML

/makeTeam
Supported Methods: POST, HEAD
Middleware: Requires Login
Body Params: name (name of the team), top, jungle, mid, bot, support (the name each champion designated for each role in the team, respectively)
Description: Creates a new team in the users account
Return Type: JSON

/accountPage
Supported Methods: GET, HEAD
Middleware: Requires Secure, Requires Login
Query Params: None
Description: Renders the account page
Return Type: HTML

/checkPremium
Supported Methods: GET, HEAD
Middleware: Requires Secure, Requires Login
Query Params: None
Description: Returns whether the user has premium enabled or not
Return Type: JSON

/getPremium
Supported Methods: PUT, HEAD
Middleware: Requires Secure, Requires Login
Query Params: None
Description: Enables premium mode for the user’s account
Return Type: JSON

/cancelPremium
Supported Methods: PUT, HEAD
Middleware: Requires Secure, Requires Login
Query Params: None
Description: Disables premium mode for the user's account
Return Type: JSON

/changePassword
Supported Methods: PUT, HEAD
Middleware: Requires Secure, Requires Login
Body Params: username (user's username), oldPassword (user's old password), newPassword (user's new password), newPassword2 (user's new password retyped)
Description: Changes the users password
Return Type: JSON