Dave Centeno
CS5610 Web Dev
December 16th 2022

Project Documentation

Executive Summary

My application is called “Festival Conductor”. I created this application which is still in progress because of a problem the Maine Music Educators Association (MMEA) has. Festival conductor allows, educators, judges, and festival managers to register with my application and have centralized access to key information to running their festivals, this includes student registration, user registration, role assignments, a centralized area for announcements, a contact list of active participants, and eventually scheduling and some way to perform emailing/invoicing.

Project Background

Every year MMEA holds festivals that require a lot of organization, communication, scheduling, contact information, student registration, scoresheets, and other moving parts. MMEA’s process is currently, being managed by loose leaf papers, an assortment of google sheets, word documents, and monthly meetings. They have expressed that this is a major pain point for them and they have continually looked for different ways to resolve these problems. Being a non-profit and school organization, their budget is limited and some decent solutions that they look at are out of reach to keep year-to-year realistically.

Still having ties with my music education background I found this an interesting opportunity to try and create something for a community that I still care about. Creating this application is still in its infancy and certainly a rough draft but I think the core ideas are there, that is to centralize a lot of information in one place securely.

Given the timeline of this project, I believe there was a lot that I was ambitious about but I took on a bit more than I could handle with the time given. My goals with the time given were to create the homepage, figure out an authentication system, a dashboard that provided some type of infographics and announcements, a student registration page, a user registration page, an events page that drew from Google Calendar API, a contacts page, and a scoresheet page. I feel like those objectives were a good start to the core of the application. I was able to get a decent amount of the mentioned objectives implemented but not to the refinement that I would have liked. This will most likely be an ongoing project for me.

Project Development

I. Login

Working with the login was something I should have looked into a bit sooner as I was more focused on some of the application's core functionality first than working with authentication. For context, I developed my application using the Next.JS framework (https://nextjs.org/) because it offered Server-side rendering and other features that I believed would in the end provide a secure and performant application. The learning curve for it was a little bit on the steep side but after a bit of learning started to feel pretty natural developing.

Having to manage authentication manually which is something I had tried to develop in the past I remember not having a lot of good luck with that so I looked to an API that could assist with authentication. I found Auth0 (https://auth0.com/) to be an excellent fit for the application. It had many tutorials on what the product was, outstanding documentation, and easy integration. Their documentation for NextJS was excellent and easy to implement. What made Auth0 a step above was its ease of use to manage users and add different api callbacks.

With that being said authentication is working and working well within my application but introduced an interesting problem on my end. I am currently using MongoDB to store all user information and other important information. Users can be created with Auth0 but they do not have any relation to the user profile created in MongoDB. My current workaround, for now, is to use the Auth0 ID with the user registration process to have that associated with the newly registered user. Auth0 does offer a way to connect a database with their service but ran out of time to look at this documentation and test how this works. In the future, I'm looking to link Auth0 to my MongoDB atlas database.

II. Home Page

The homepage is where I have taken a different direction from the initial specification of the final. My application is very reliant that a user being able to sign in and access protected pages and eventually role-based permissions as well. The anonymous user web page is where someone would be able to see a very basic page where they can sign in, log out, sign up for an account, and navigate to the about and privacy pages to see what this application can do. The true home page for an authenticated user is the dashboard page. This is where a user will be ideally led to after authentication and see important information. Currently, when logging in to the application the user will be met at the dashboard page with a card-like item that has announcements. The announcement card is dynamic providing CRUD operations to the mongodb of holding announcements. This will be useful to show important events or deadlines for users of the application. Future plans for the dashboard are to include an infographic of registered students, how many registered users are attending a festival or audition process based on roles, and other important information.

III. Profile Page

The profile page is one of the more polished areas where I felt I met the majority of the requirements here. Users navigating to their profile page can edit any information that they choose and send a PUT request to the database to update their profile information. Due to my current login situation, I have hard-coded a userID to pull in to prove that the profile page and its updating and saving work. Once I have my relationship with Auth0 and MongoDB figured out this should work without issue and present a pretty nice perk. The nice perk would be that password reset is absent from this screen because it is being handled by Auth0 which handles the user credentials. A password can be reset two ways for the user, they can either click on the forgot password helper in the login page or they can request an Auth0 admin (myself the developer, or a designated user) to log in and reset their password for them. Through the profile page, there are no other items to be saved at the moment other than the personal information that the user would like to update.

IV. Search

This part of the project was probably the toughest part to implement and admittedly was only to get this part working due to the difficulty of dealing with Google's API. I chose to make use of Google Calendar's API which in theory would sound easy to use but was probably the most difficult API I have used to date. Their documentation was incredibly lacking in steps on how to set up their connection and spent multiple days trying to troubleshoot why I could not "GET" request to my own calendars. Earlier this week I was able to figure out the reasons why I could not get my requests to work, 1) You needed to perform these requests on a public calendar, 2) A more advanced GET request header was required one requiring a Bearer token that expires every hour, and other parameters. After much troubleshooting, I was able to get a successful GET request which allowed me to produce the Events page. This page works intermittently at the moment due to Google's security settings. One also has to log in to perform any requests. When working though, the page lists all available events and shows dates and times in a human-readable fashion. The search bar on the top will allow the user to edit the fetch request url to pull in an event and display it on the page.

V. Operations.

For this project, I chose to work with MongoDB Atlas. This has been a platform tinkering on in a previous class and continued to learn about it in this class here. The document model is great to work with because it allowed me to update my schemas relatively easily. The pages Student Registration, Registration, Profile, and Announcements, all feature CRUD operations in some capacity. NextJS makes use of talking to the database through its API folder which is invoked by making any of the common requests like GET and POST. Having this happening on the server side also reduced the amount of external packages needed to help make these requests to the database. In some cases I used Axios to deal with a bit more involved requests as I found it a bit better to handle the requests.

Project Completeness

My project I think fundamentally meets the majority of the requirements of the project but definitely needs some refinement to be a clean working website. So I would say I'm about roughly around the 90% marker of completing my initial objectives. I was able to build out the login page, profile page, build out a database, figure out user authentication, and create pages using CRUD operations against the database. My overal design of the website I was pretty happy about as I spent sometime earlier this sketching out some visuals of the application and trying to re-create with vanilla CSS and a bit later with BootStrap.

The biggest items I need to address is making the relation of the MongoDB user with Auth0 database, being able to test if saving an event works for the user, update some CSS to be more responsive, and finding an alternative to the Google Calendar API. The creating of the relation of the Auth0 user and the MongoDB just takes time and understanding how I can get the two databases to talk to each other in a way that makes sense. There's quite a bit of programming to write there to securely to get this to happen. My CSS for 90% of the project has been hand written vanilla CSS. I wanted to make sure I understood the CSS concepts of the class and be able to execute on them without relying on libraries such as bootstrap to make these items for me. Near the end of the project I realized there is a time and place for an item like bootstrap especially when youre trying to demo your application idea. In the later part of my project I did implement bootstrap t help with some responsive design and speed up getting place holders for my logic to take place. Lastly, I'm really disappointed about the Google API for Calendars, for such a well known product the developing experience was less than fantastic and being able to integrate with it was probably one of the biggest challenges of this entire project. It cost me a lot of time troubleshooting and trying to get work appropriately.

For independently working on this project I think I made some good progress and learned a lot by building this website. There are a lot more elements to making a website than I initially thought and some smaller intricacies that were very time consuming which I did not estimate correctly and put me behind my schedule. With that said there are some elements in my program that statically assigned to show proof that the application can perform the operations correctly such as the Google Calendar API and the profile page pulling in user information and updating user information.

Project Reconsiderations

I would reconsider my project choice to better align with the specification of the final. I felt like I was ambitious to get this application going but ended up being a larger piece of work than I initially thought and led to less polished final project than I would have liked. I would in the future pick something smaller that focused around one particular concept such as a simple web app to pull in the stats of a popular video game and have the stats save to a user. I think something like that would have highlighted the final project better than what I was trying to accomplish.

If I was going to do the same project again I would have a more focused approach on a simpler part of the app like just make a scoresheet app where a registered user could sign in and save a scoresheet to their name. This would effectively showcased the “Judge” role of my application. The user would sign in and submit a scoresheet for a student and have that scoresheet saved off to them to show proof of scoring the students audition and save off a copy to the student as well.

Another thing I would have done differently was research APIs prior to starting the project. Having APIs that are accessible and easy to work with would have made developing the web page easier.