Group 3 - Third Iteration

yAMUN, a course planning system for MUN students

Jack Harrhy, 201732922
(jaharrhy@mun.ca)
Genadi Valyeyev, 201921376
(gvalyeyev@mun.ca)
Aidan Langer, 201735677
(aplanger@mun.ca)

1 Functionalities

Our team set out to design and create a dashboard that would make planning and constructing course schedules more ergonomic and comfortable for Memorial University students; and we'd say we've reached that goal. - Created three scraping tools to crawl the required pages (University Calendar, Banner, People) and extract the data using things such as regular expressions and HTML parsing.

- Wrote the logic to take in the unstructured data from above into well formatted data, that is inserted into MongoDB.
- Enriched the course offerings data with course descriptions from the Memorial University calendar and instructor contact information from the "People Search" API.
- Built a well-designed REST API around all of the above mentioned data, which can be consumed easily by any other project that wants to utilize this data in the future.
- Wrote over 70+ tests to ensure the structure of our application is sound.
- Built a modern frontend using React, Tailwind CSS, and Redux, that easily blows what MUN has to offer out of the water.

On top of all the functionalities already discussed, the functionalities that we could not implement but had made note of in our previous iteration have, of course, been properly integrated into our project. This includes interfacing with the University Calendar to collect further course information, and fleshing out the calendar exporting system to include multiple events and accessibility with Google Calendar.

As for changes from our previous iteration, we changed how we identify courses. During the development of the frontend, we found out the course CRNs are not unique when you consider multiple semesters. This meant that we had no way to uniquely identify each course offering. To fix that, we decided to create a composite key of the course semester, year, and crn; we call it SID (short for section id).

We have also implemented a new People model, which scrapes information from the People API about the various staff members at MUN and allows

our users access to even more helpful information, including professor details, email addresses, office location as well as hours, among other things.

The filters implemented were not as robust as we had originally planned, some that were mentioned in our first iteration we were unable to include. However, we believe that the filters we have implemented are adequate for a smooth user experience.

2 Views

In our application there are four main views.

- **Find Courses:** this is the main view of our application. At the top of the page, you can find various fields to input data into to find the course you require. This includes year, semester, semester level (graduate, undergraduate), subject, and course number. After filling in these forms and clicking the "Submit" button, the courses in the database that match the search queries will appear seamlessly below, along with the acompanying information. The information provided includes the name of the course, the CRN, the professor name, and the times at which the course is scheduled. You can also find a star icon at the top right corner of each course, which will initially be empty. Upon clicking the star, the course is added to your bookmarks page, and the star is filled in with yellow. As well, an icon in the bottom of the course listing can be clicked to show a course description, pulled from the University Calendar.

- **Schedules:** here a user can find their personalized schedules, first being presented with a list of the schedules created, as well as a button to create an additional schedule at the top. They can also remove the schedule from this window. Upon clicking a schedule, they will be taken to a representation of their schedule in a weekly calendar format. Hovering over the courses will show the name, number, and subject of the course, in case it is cramped in the calendar view. Under this, the user will find all the course they currently have added to the schedule, and they are able to access all the options that are present on the "Find Courses" page; add or delete from bookmarks, and delete from schedule, which updates the page in real time. There is also the option to export a course to an .ics file by filling when they wish the calendar entries to begin and end.

- **Bookmarks:** here a user can see the courses they have bookmarked, presented in the same manner as the "Find Courses" page. They are of course, able to do all that they can in the previous two views in this one as well, such as adding or removing from schedule, and deleting the course from their bookmarks, which also updates the page in real time.
- **People:** this page uses the People API to show information about a MUN staff member requested through a search form at the top of the page, written as first letter of first name, followed by last name. When a person or persons are returned, they will be shown alongside their email, department, campus, room, and telephone number. If another staff member is searched for while one is on screen, the page will be updated to only show the new search query's results.
- **Login:** this is, of course, the login page. Users will be redirected here upon failing authentication, and will have to create an account on the same page if they do not have pre-existing login information. If the user is already logged in, the page will show the user the option to log out.

3 Frontend

Upon loading our application, you are first required to create and login with a personal account. This authorization is required to access most of our routes, or views, and if no authorization is presented when trying to access a page the user will be redirected to the Create Account page. The layout is plain and simple; a list of tabs at the top of the page that can be instantaneously switched between with a click. On top of this, the way we use local storage persistence ensures that your progress on a particular tab (search queries, filters, location on page) will not only be kept in between tabs on the site, but it will persist when closing and re-opening the tab as well. Our frontend also uses multiple outside libraries to ensure fluidity, as well as provide a smooth user interface. Here is a list of the main libraries we've used.

- **React:** a component-based JavaScript library for building user interfaces. React allows for simple, smooth UI construction, a clean structure that allows for easy code refactoring and debugging, as well as cohesive code reusing by importing components. We decided to use this over jQuery as React allows for much easier rendering and updating, being that React does not require you to define how these processes will be carried out.

- **Tailwind:** a framework with many pre-defined CSS utility classes to be used easily and quickly. As well, only the CSS classes currently used in the program are bundled for production, which means that most tailwind projects ship less than 10kb to the clients!
- **Redux:** a state container for JavaScript applications, Redux allows you to build apps that run consistently on any environment, and are easy to test. It also allows for real-time testing and extensive debugging.

As for animations, we decided to be very sparse with them, to reflect the minimalist nature of our application, we played around with adding them but we found none that really increased the quality / style. You can however see animations when adding or removing a course from your personal schedule or bookmarks, as well as when receiving error messages. When clicking the star icon next to a course to add it to your bookmarks, for example, a confirmation message will appear to "swoop" from the top of the screen ever so slightly, with the message text being relevant to the action performed. This is handled with react-hot-toast.

4 Running the Application

There is a document called README.md in the root of our source code which outlines how to run the application, since it is slightly more complicated than one single npm command. If there are any issues in doing so, feel free to reach out to us.