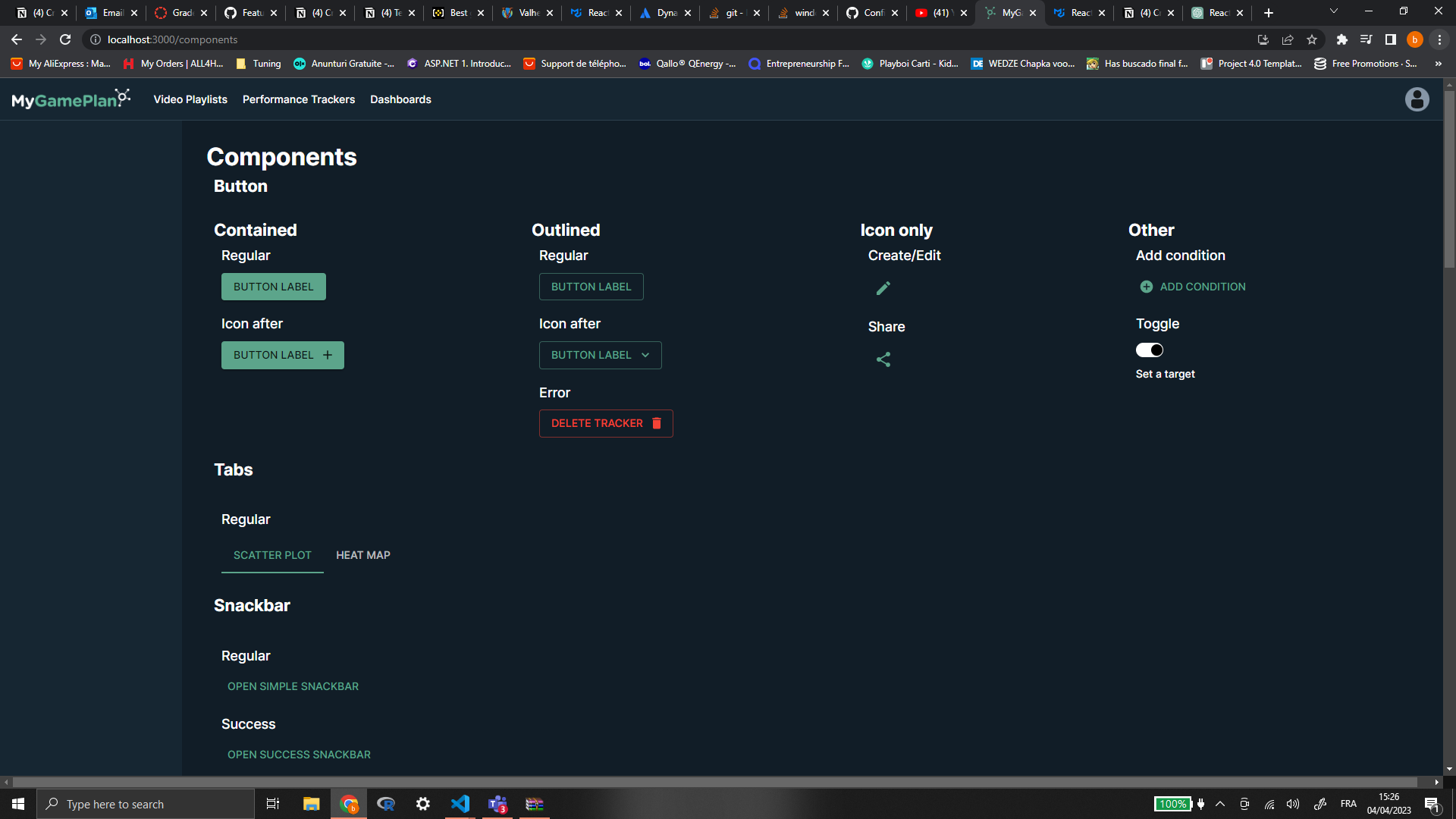
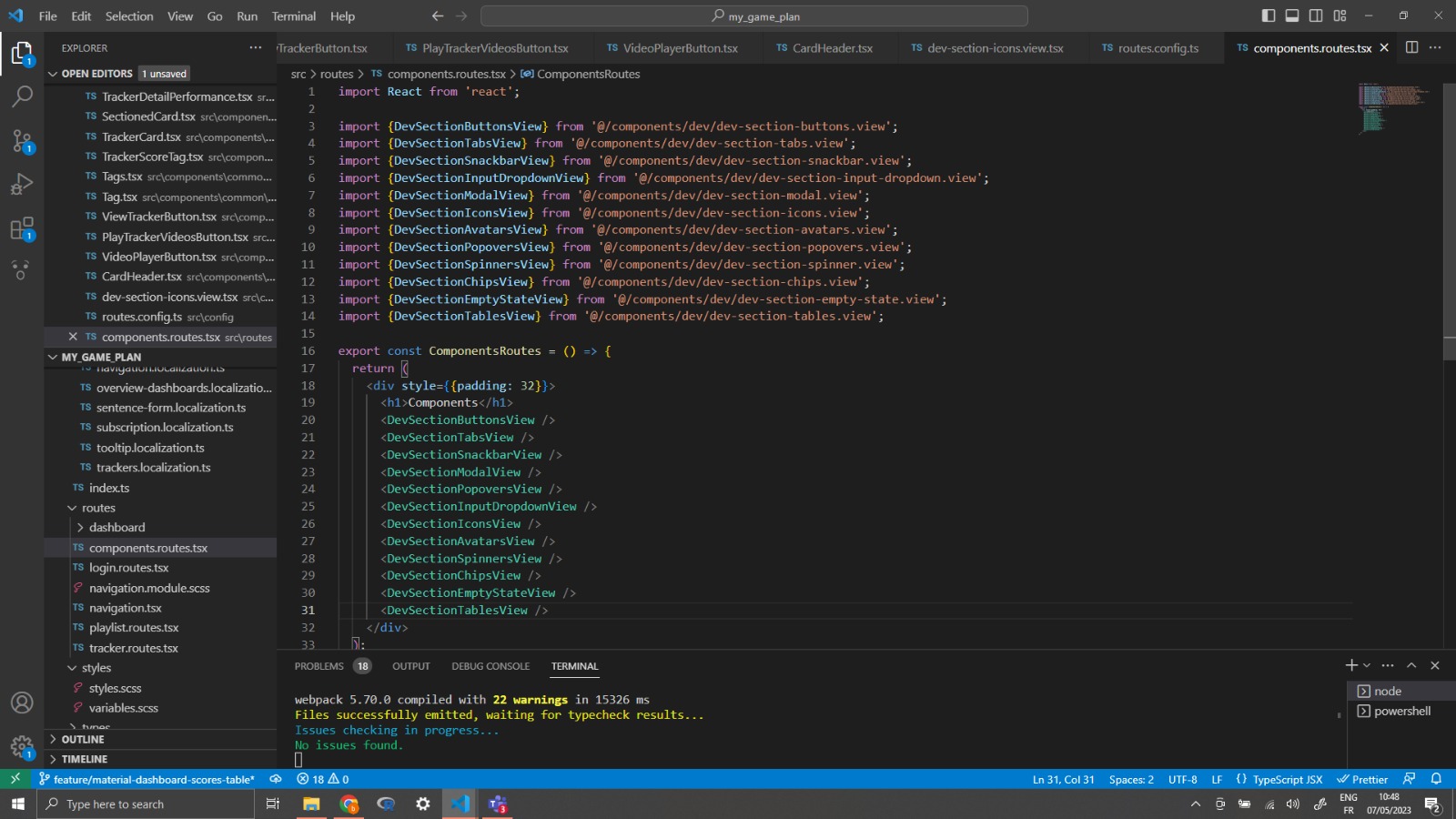
**Evidence for realization: MyGamePlan Internship**

By Bogdan Lazar (R0779031 - 3AI01)

# Task 1: Strategy for Replacing AtlasKit with MaterialUI

MyGamePlan wanted to replace its long lasting AtlasKit component library for something smoother and more subtle. MaterialUI was the best choice and in order to start refactoring, a special page would have to be created within the app in order to not only guide the average developer when looking for how to implement a certain component but also to provide a solid strategy on what to replace and with what.





In order to set the foundation for this feature, due to the structure of the app, a page had to be set up with its respective route. After creating the components.routes.tsx file, this page would be found via /components in the app. To also lean out the code it was a much better idea to then create a view for each set of components that would be shown.

In order for the actual component bit to fall into place, a scalable React component had to be made. This is the DevSection component which can be told by the naming convention of each view. Each view basically takes this component, binding the right data to the appropriate props of DevSection and returning it as JSX code. Here is the DevSection code with an example of a view:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

In order for the view to work properly, a way to store the data for each component was needed. This was done in the end via JSON with an array of objects where each object contained an array of variants of each component where the components would be stored by type. To fix any potential compilation issues, special interfaces were made to instruct the app that a property of the object was of React.Node type. After putting all of these in place, for each DevSection it was simply rinse and repeat with certain caveats of logical functions to make them work properly for a basic demo in each case.

The only problem that remained was that some components were either not available in the base MaterialUI library and either required experimental code or paying for the X library. There was also an issue with finding some components to replace AtlasKit ones with like tables that the company was using. It was a good learning experience. It really put the developer life into perspective.

# Task 2: Replacing AtlasKit with MaterialUI

Now it was time to actually replace the AtlasKit components with MaterialUI ones. This started small, with buttons that could be swapped out immediately. It just took the code and imports for the Material UI items that would be used instead. Then the right bindings had to be made to the props of the Material UI component and it would be ready to go. Here’s an example:

Text

Description automatically generated

In this simple case anything that was still AtlasKit component code was commented out and the MaterialUI code was dropped in. Commenting helped in case errors were encountered and made it easy to simply switch the codebases in order to visualize which one was working.

Everything worked well until the components such as modals were encountered which were working with logic that was programmed at way deeper levels and required more experience with the overall code of the application in order to be done. The team then decided that this task was better escalated to more experienced personnel. After seeing that there were no more simple components to replace, it was time to take on another task. It turned out that the tech lead managed to find a way to replace the data tables with regards to shots and crosses of the players in a team.

# Task 3: Replacement for shots and crosses tables (strategy)

This task would briefly revisit the components page. The point was to replace the AtlasKit table that was being used for shots and crosses data in the dashboards of a team. After discussions with the tech lead, it was decided that using a dummy json object array we could have a proof of concept for such a replacement table. In this case the idea was to recreate this table with basic React and TypeScript. It was required of the user to be able to select columns, rows and cells. With enough research, it turned out that this could actually be done with a Material basic table but with a bit more vanilla React. Here is the code with an explanation:

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

The basic idea is that with a bit of logic to enable the required selection, Material’s basic table could easily be used for this. This task has been completed about 75%. The table was displaying the data as required and rows and columns were selectable. The only problem was that cells were not selectable and rows could be selected by clicking any cell within a row. The idea was only to enable selection for clicking the first cell of a row. Now this problem was an easy fix within the logic by simply only enabling selection when clicking the first cell of a row. Column selection was already a bit more extensive with the code. After consultation with the rest of the team, it was decided another task could be taken on instead of ruminating over cell selection and wasting time.

# Task 4: Easier access to User manual from app

This task entailed a bit more than it let on. Not only was it required to provide easier access to the user manual from the user settings popup menu but also to try refactoring the main navigation of the app to Material UI. This is how it was done:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

This was done by first creating a view for this navbar to follow the new convention the dev team agreed upon. In this view the navigation itself with all of its respective logic was coded. It included Material’s AppBar with a Tabs component entailed within. Lots of logic function programming was required therefore. This was then declared as a React component in JSX code in the file that displayed the navigation itself. This was a lot of developer fun. The whole set of problems encountered as a developer on the regular were faced during this task but alas it was carried out quite well. Of course, feedback followed that was therefore resolved by fixing the issues within the code and committing once more.