

EZPCBuilder

Web Frameworks final ca

Kyle Carey, James Bellew | Web Frameworks | 03.05.2021

# Video URL for Demo

[Web Frameworks Demo](https://youtu.be/3V8039H1suI)



# Background

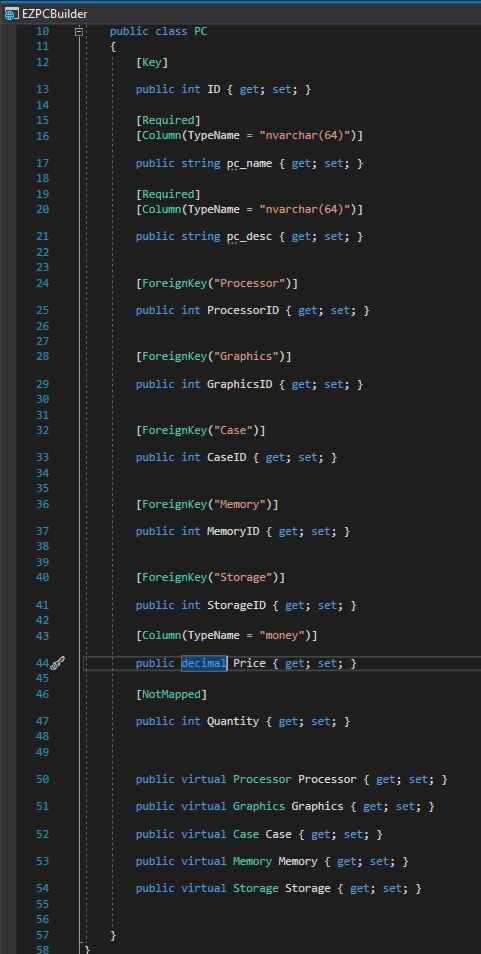
This website was made by us as part of our Web Frameworks module for our course. We were required to create a project which was a version of our project for our Software Project module. We had made a gaming pc store which was aimed at young adults with an interest in gaming. This is reflected in the style of our website. As part of the descriptor for this module we were tasked with the following:

* Using ASP.NET Core to create the website as an MVC website.
* It must reflect on our project for our software project.
* We must add in a self-learned part to the project each.

## Creating the project

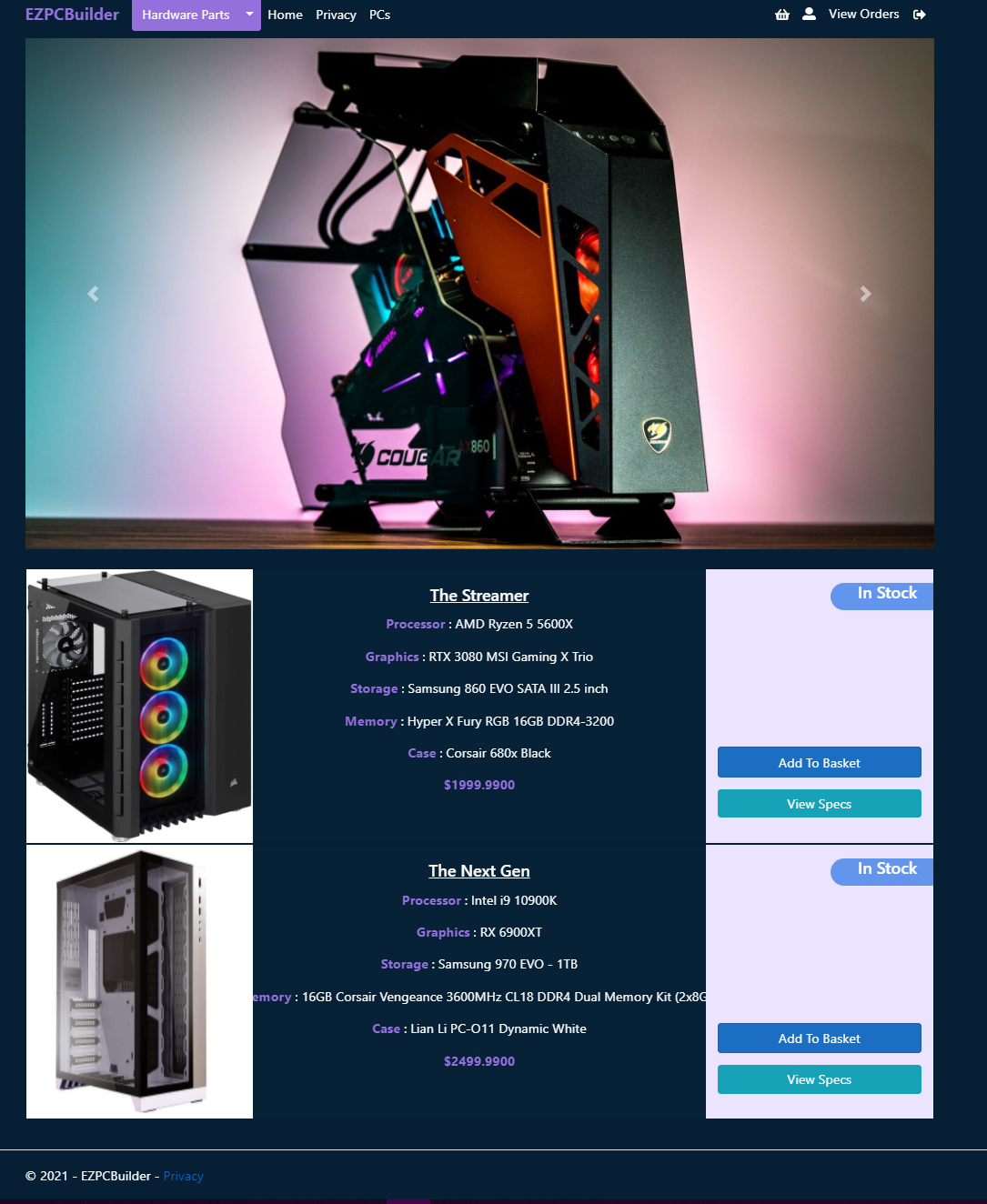
As our software project was originally produced in Java and was using 20 tables across 2 different databases. So due to time constraints we set a realistic standard for our C# project. We took a code first approach to speed up a lot of the work with the databases and took out the more unnecessary parts of our project.

We created models for the PC and it was populated with other models such as a Case, Processor, Graphics, etc.

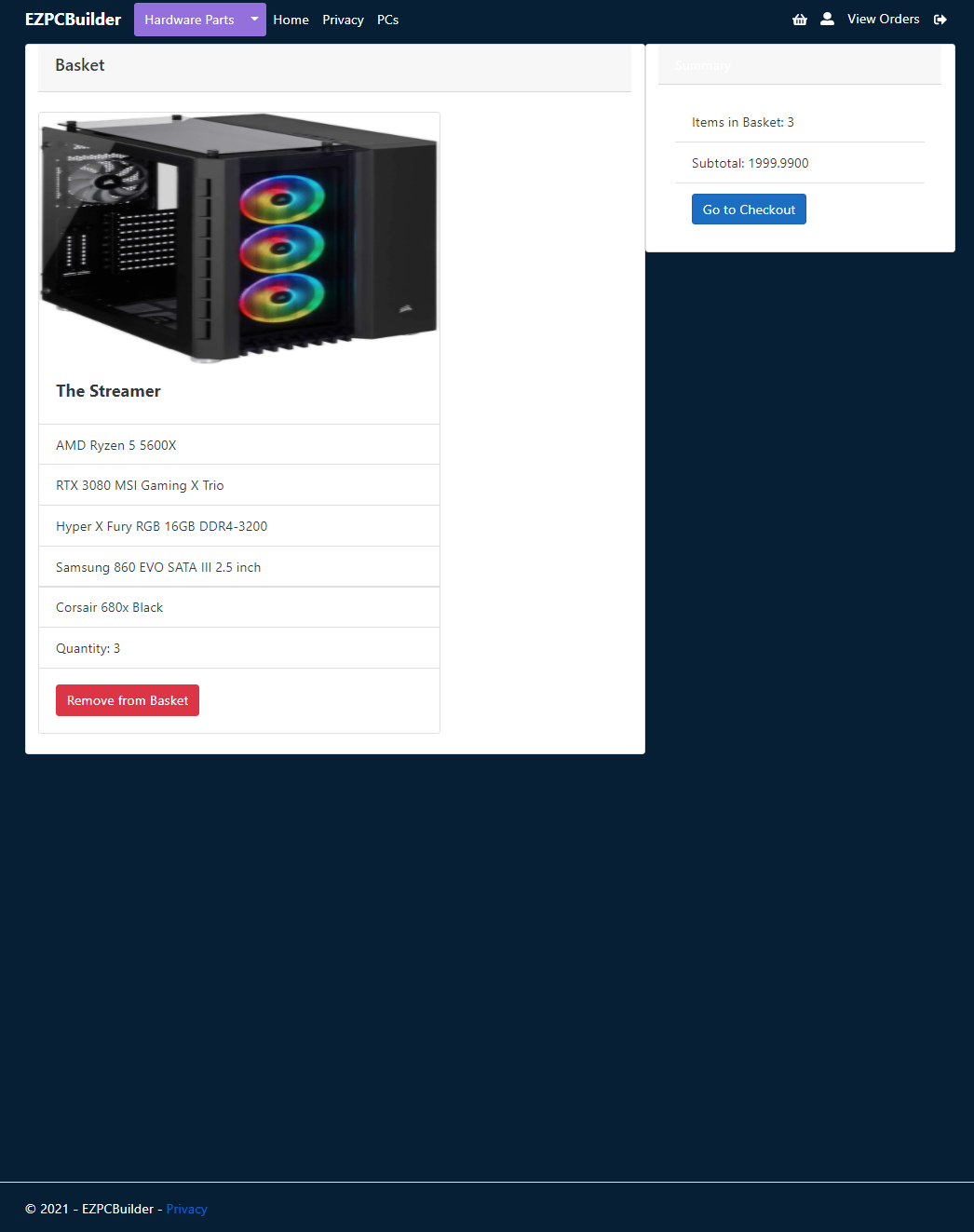


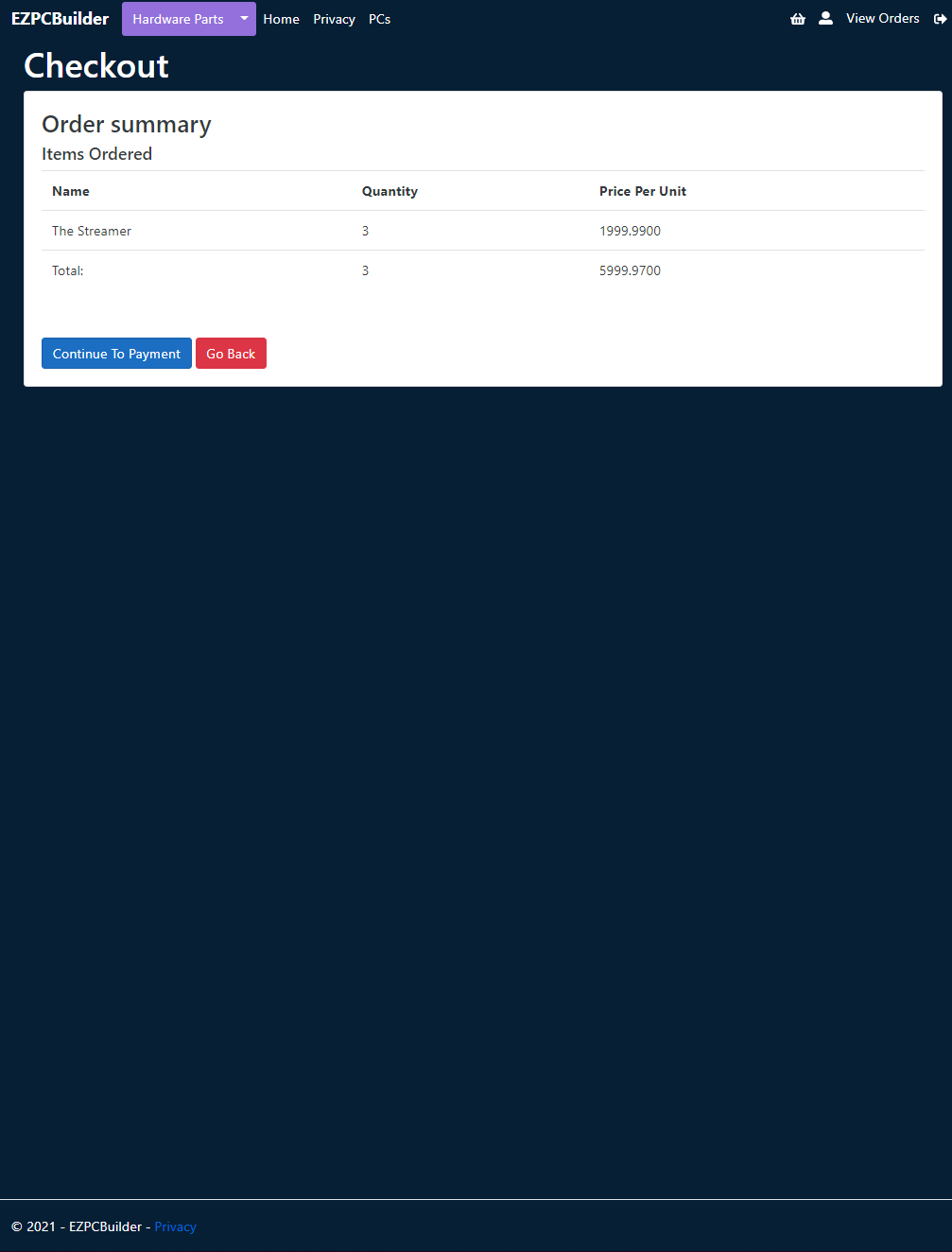
Each of the models were created and then scaffolded using the built-in framework.

We then were able to create the store front once we added in some data. Below is the final result of the store front. This code is seen in PCs/Index.



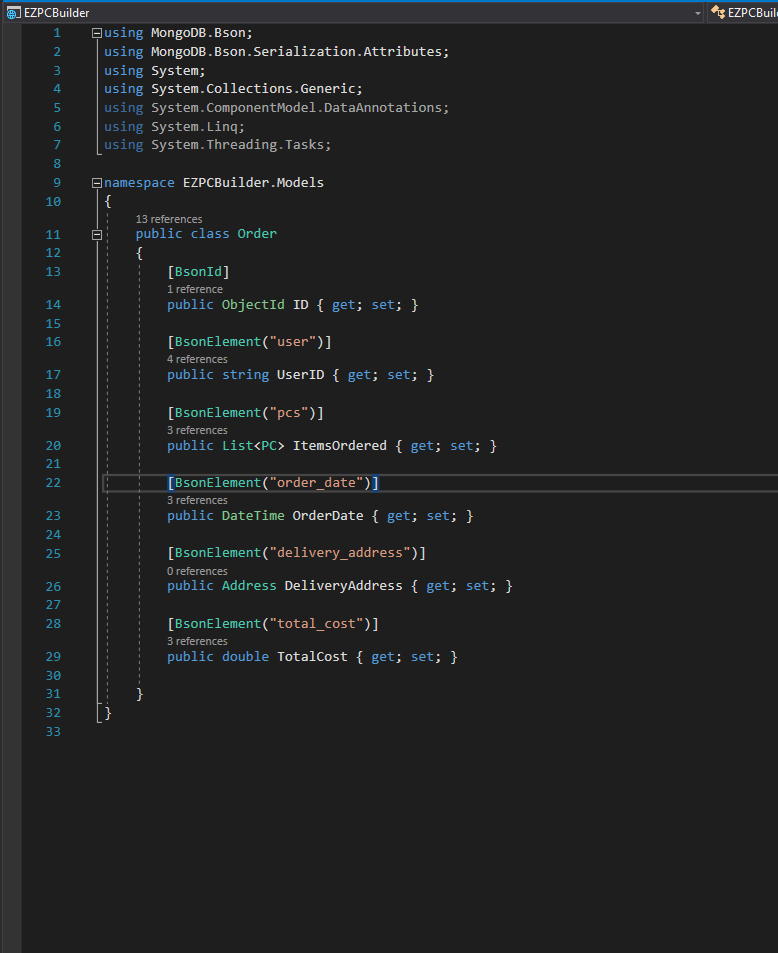
We also implemented a functional basket. This was done with a Basket model that accepted a PC id and quantity. The subtotal and items in basket were calculated as the page loaded. The code here is used in the Baskets/Index.cshtml file.



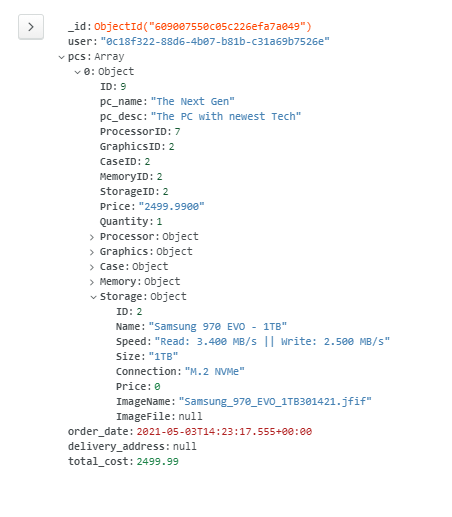
In the checkout there is a summary of items that were in the basket.

# Kyle Self Learned

Kyle implemented the use of a NoSQL database as part of the project to take orders. The reason a NoSQL database was used here was because the data for an order was dynamic and needed the flexibility, so a single query to the database would return a single document with the order details. The database used was MongoAtlas, which is a cloud-based MongoDB server. The Order model was then designed code-first for easy implementation.



When the data goes to the database it looks like the image below. It stores the ids of each of the parts that make up the computer model, and then saves each part as an object, this is useful in retrieving data as it doesn’t require too many queries to the database.



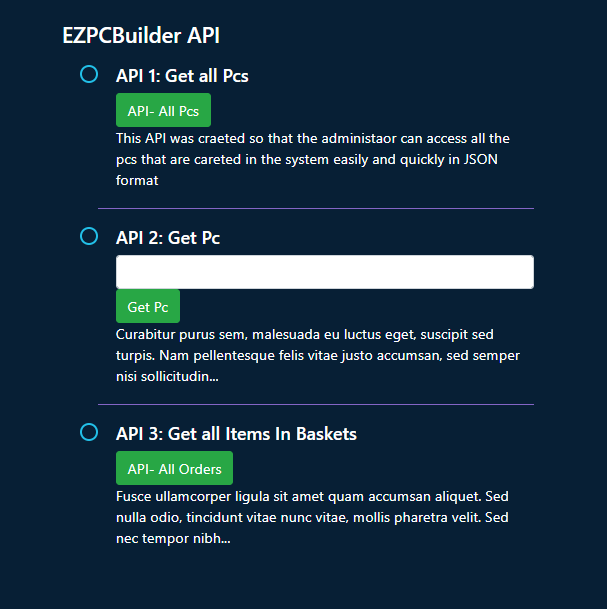
As MongoAtlas is a cloud-based database it was easily accessible by the both of us. There are many tiers available for MongoAtlas that gives more storage, ram and processing power, meaning that it is scalable. This was good practice to get started with a database like this seeing as it was not covered in the course.

# James Self Learned

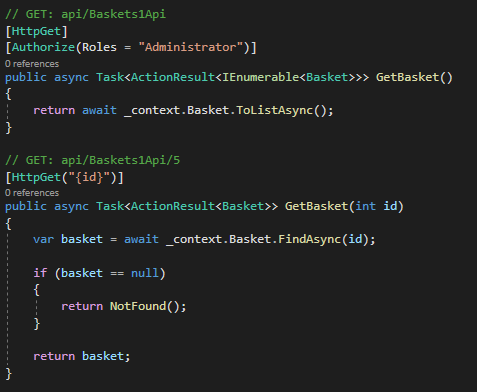
James’ self-learned was in the area of API , authorization and deployment. There needs to be a certain level of security on the site and implementing user roles is a must to maintain a cites functionality and security. For this James used the ManagerRole entity from the dotnet Framework to allow James to create different roles within the site and to permit certain users to have access to different tools and features like CRUD features.

# API

James used the Scaffolding technique to allow him to create API controllers to allow others to call and access the information supplied form the API



Here is an example of one of the API services with a call to get all the items in the basket within the site, to allow an admin to get a good idea of the stock that might be going out of the inventory.

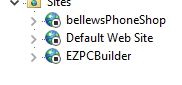


And here is the output of this call

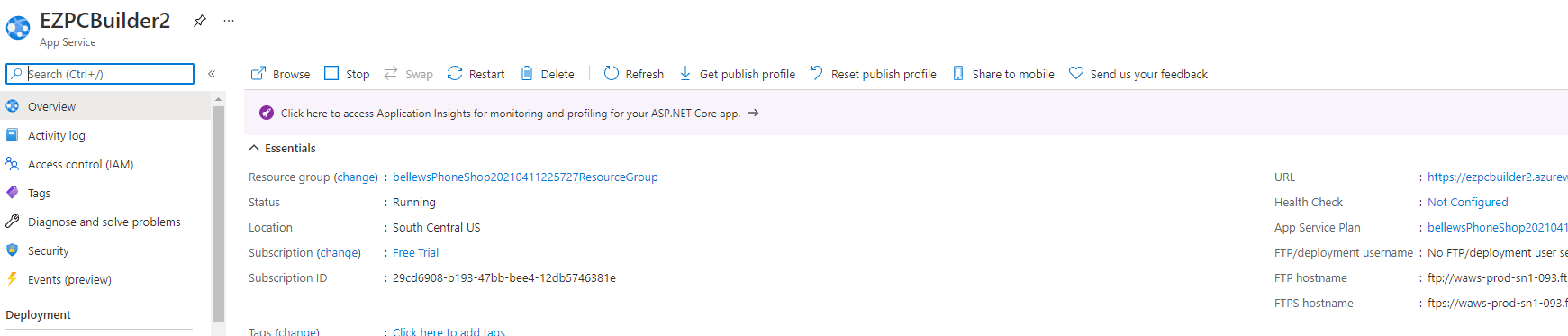
# Deployment

James originally deployed this site on the iis service which overlaps with his sys admin course in college , but James wanted to try and deploy this site on Microsoft Azure to just try out a different way of publishing it.

Here is the IIS deployment



Here is the Azure Deployment



# Authorization

James learned how to implement display certain data to certain users depending on their role within the application. James also learned how to prevent user form accessing some features with the URL by using the in app Microsoft.AspNetCore.Authorization tool.

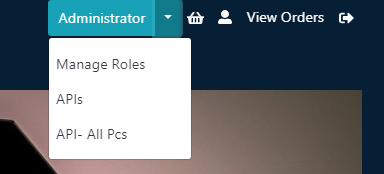
Regular User View



Admin View



Admin Tools



# What would we improve?

Although the project went smoothly there are several things that we would have liked to implement but with the time constraints and other projects to tend to there were things left out of this that could be implemented.

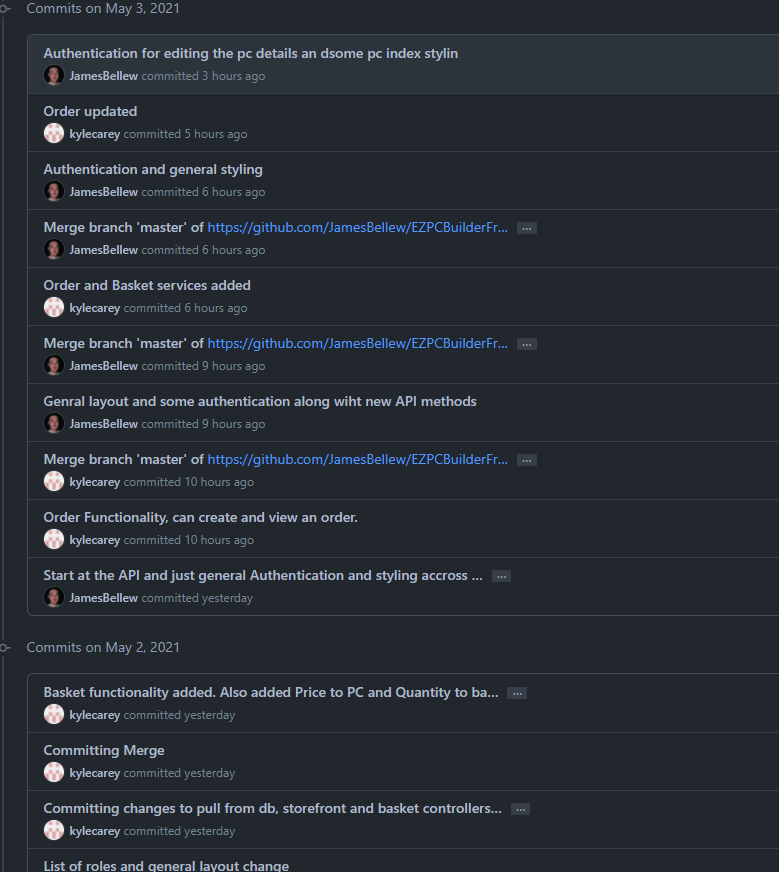
* In our Java project we had a page for customers that would allow them to pick all the parts they wanted for a computer. Unfortunately, there is a lot of validation and work involved with this and we decided to leave it out as it would require many more tables and models.
* Addresses. We had aimed to have the user enter a delivery address however we left it out to tend to more important features so this was left out in the end. Given another day this would have been implemented. Most of the code was there, however, it was simply left out.
* Styling. We had spent very little time styling and it would have made the project style linear across the whole website.
* API. We would have liked to add a client API that would consume the data we had and made it useable.
* Deployment. We had some issues getting the project running on Azure. It is hosted there but there are some issues with accessing the database. You can view the website but not access some pages.

# Teamwork

In order for James and Kyle to be able to complete this project they had to use a variety of tools in order to connect and communicate to each other and to allow them to work simultaneously on the same project.

For this they both used a number of tools like

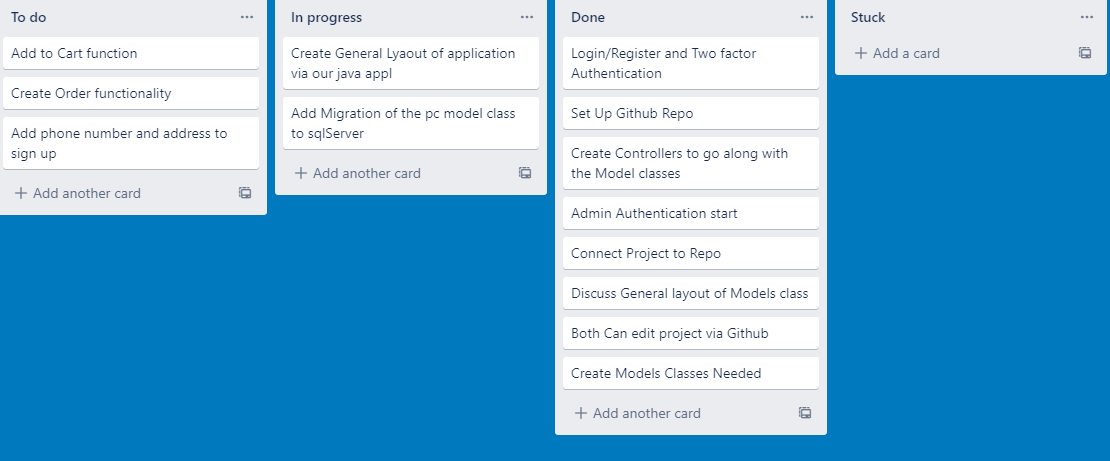
**GitHub**



Github was the clear choice for us when it came to both James and Kyle needing to work on the same project without having to worry about sending each other files and what not.

**Trello**

They used [www.trello.com](http://www.trello.com) to keep track of all the tasks that we needed to complete and any issues they were having. Trello was a simple solution to keep track of tasks and organizing the work ahead



**Communication**

To communicate with one another both James and Kyle used Microsoft Teams and whats-app to communicate with any immediate issues or just for calls talking about anything relating to the project.

