

Module: Web Apps and Services

Lecturer: Dr. Zafar Khan, Barry Hebbron

Student no: C2644873   
Student name: Abdulrahman Al Hariri

ThAmCo App Report

Introduction:

Overview of the purpose of the report :

In this report I will talk about the ThAmCo evaluation and best practices followed, how the solution was approached and the security features that have been implemented or planned for the application.

The provided Visual Studio solution (ThAmCo) is a full-stack application was fully developed with ASP.net core 6. It serves events booking system, and allows the users to manage events, staff, guests, food booking and venue booking.

Database: for database, SQLite was used and entity framework to map the tables and columns.

Backend: The backend of the solution is depending on a ready provided endpoint (Venue) for booking venues. It also uses a project was developed to provide endpoint for managing and booking food (Catering). Finally, the application has its own backend to make call to those end points and to manage events, stuff and guests.

A screenshot of a computer

Description automatically generated

Frontend:

For front end couple of tools were used, mainly the app uses Model View Controller (MVC) this pattern helps to achieve separation of concerns, also Ajax, bootstrap, and sweetalert2.

A screenshot of a computer

Description automatically generated

Completeness of the Solution:

Catering Services (Catering):

Implemented Web API services for CRUD operations on food items and menus.

Developed functionality to add/remove food items from menus.

Enabled booking, editing, and cancellation of food for events, providing a confirmation identifier.

MVC Web App(Events):

Created interfaces for managing guests, events, and staff.

Implemented the ability to book guests onto events, register attendance, and display individual guest details.

Provided functionalities to create, list, and edit events, including editing except for date and type.

Venues Integration:

Integrated with the Venues web service for venue reservations, enhancing event planning capabilities.

Over all the solution meets the defined criteria and goals provided by the requirements document.

Own Working Practices:

Best Practices Followed:

Architecture and Design Patterns:

Utilized ASP.NET Core and Entity Framework Core features optimally.

Applied appropriate architectural and design patterns, promoting code maintainability, scalability, and separations of concern. That’s by using classes and Interfaces I used Dependency injections which helps improve the reusability of code, while also reducing the frequency of needing to change a class, a template of the methods or variables in an object.

A screenshot of a computer

Description automatically generated

Documentation:

I wrote a readme file, explains the solution. And I wrote comments (Where needed), it **really were needed,** so you will never find me writing this type of comments:

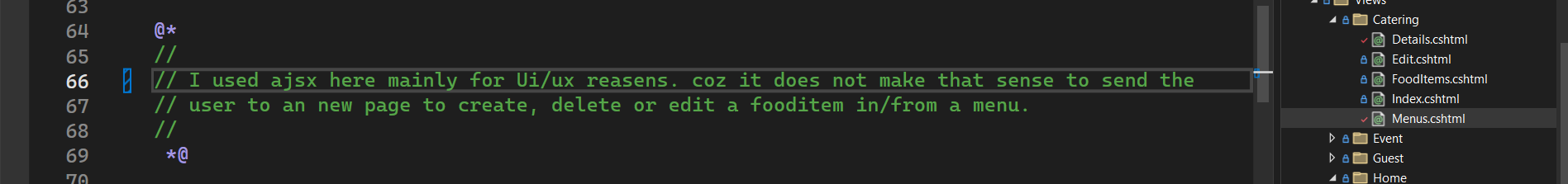
A stop sign on a fence

Description automatically generated

If that the type of comments you want! then, yup unfortunately you won’t find!

However, you will find comments if I used unusual logic, and usually you will find the method name or the variable name tells you what it does or it is about.

I also included comments or references to external resources like stackoverflow.com. or other resources of why I went that way or why I choose to do it this way.

One example might be, that I used JS and Ajax calls for Catering creating editing and deleting food items and menus. But you will find an explanation why as a comment:   
 

Also, because I believe JS is super important language and that it’s underrated in our course, so it was an opportunity for me to use JS and play with it.

Regarding an interesting issue I faced during writing JS code, that I was not able to write JS code in a separate file, I was getting complier error.  
A screenshot of a computer program

Description automatically generated

It was ESlint error, and that was only on my machine, that’s because of incompatible versions of Node, Npm and ESlint.   
My approach was changing the node version, I use NVM, deleting Npm and ESlint and reinstalling them.

That did not work, for an unknown reason, I might reinvestigate this if I got the chance to.

However, I could not even found documents for this problem. The last solution I left with was from VS itself turning ESlint off.

A screenshot of a computer

Description automatically generated

VS Source: <https://learn.microsoft.com/en-us/visualstudio/javascript/linting-javascript?view=vs-2022>

NVM source : <https://www.freecodecamp.org/news/node-version-manager-nvm-install-guide/>

Finally, I Used Azure DevOps as the source/version control system.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

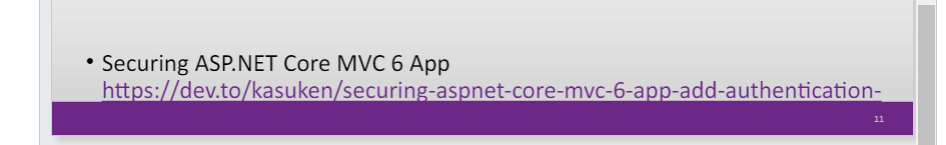
Description automatically generated

Security Features:

Built security features:

1. Used Auth0 to ensure logging in to system before use. Allowing the user to sign up with an email and a password or using his Google account.

Note: You have provided an article for Auth0 but this article is out of date and it uses rules. Auth0 are deprecating rules by actions by the 18th of Nov (please keep in mind for next year students).

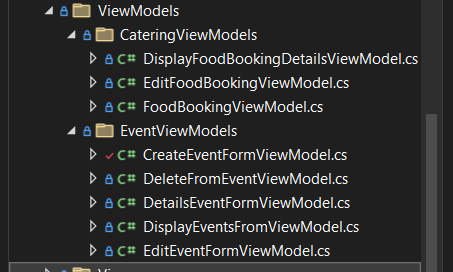


Auth0 has introduced migrating to Actions which is written in Node.js that execute at certain points within the Auth0 platform. Actions are used to customize and extend Auth0's capabilities with custom logic.

Source: <https://www.youtube.com/watch?v=B5pzpPqPbbM>

1. Input Validation:

Used View models and added extra validation for inputs, I did not implement all the security features on all the inputs but I demonstrated the use and the way to use it.



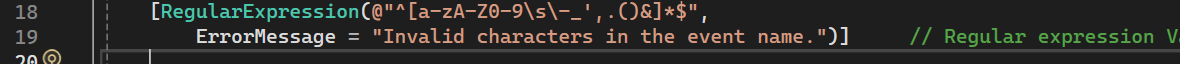
Source : <https://www.c-sharpcorner.com/article/best-practices-for-securing-your-asp-net-core-mvc-application/#:~:text=validate%20user%20input.-,Create%20a%20model%20class,-Create%20a%20model>

View models basically works as a DTO it works as a protection layer between user’s interaction and the system.

An deep example and a very interesting and good way to validate user input is by using RegularExpression also known as regex. Using regex in validation is an efficient way to make sure that the users of your system are entering an expected input.

In Event ThAmCo.Events.ViewModels.EventViewModels.CreateEventFormViewModel

You ‘ll see the following line of code



In fact I suppose to make this a s a custom validation attribute and use it for all input through the application.

However, I did not for time reasons.  
So, how this would be useful?

If the user enters something like this as event name:   
 ‘<script>alert("Hi from address ")</script>’



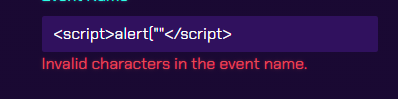
So the page will be rendered with this alert every time someone access this event or where ever the event name is rendered!

How and why?

It will try to render the name but the name would has a script tag?

In that case the browser will interbit that as JS code and I’ll try and run the code inside the tag. This known as Injection, the most fames ways of injecting is SQL injection, but that would be in the URL, that exactly what happened in our case here!

So basically, the regex validation added will prevent script injection!

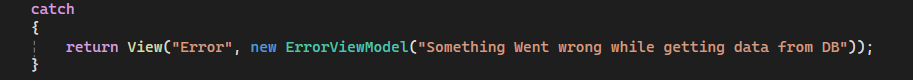


Source: <https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/security/preventing-javascript-injection-attacks-cs>

Error handling: the app handles the error quit in a sufficient way, it implements a custom error page, with multiple uses, it can be called with a customised error message, with a status code if there is need to inform the user of the status code.

A computer screen shot of a program

Description automatically generated

Use:  


Planned Security Measures:

Role and Claims Based Access Control Role.

Was planed to be done, but unfortunately did not have enough time, how ever I’m nearly there.

So I just skipped this and jumped to write this report. :(

Testing and Quality Assurance:

The application has been fully tested end-to-end.

I included a way to start the application and how to set it up on the readme file.

Conclusion:

In conclusion, the is fully working serving as an event management system.

Finally, I would like to say that I enjoyed the Module and would like to thank Afar, Shatha, and Barry.

And my apology for missing some of the lectures! :(

Resources:

<https://learn.microsoft.com/en-us/training/modules/persist-data-ef-core/>

<https://learn.microsoft.com/en-gb/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website&tabs=visual-studio>

<https://www.w3schools.com/js/js_ajax_intro.asp>

<https://getbootstrap.com/>