**Updated Report**

By HASsan nisar, haroon sadiq, mauro nunes

2022

# **Contents**

[**Contents** 1](#_Toc94179069)

[Updated Background 2](#_Toc94179070)

[Updated Feature List 3](#_Toc94179071)

[Updated Use Case Diagram(s) 4](#_Toc94179072)

[Updated ERD Diagram(s) 5](#_Toc94179073)

[Project Testing 6](#_Toc94179074)

[Technology Evaluation 7](#_Toc94179075)

[Razor Pages versus MVC 7](#_Toc94179076)

[The strengths and weaknesses of the EntityFramework and Scaffolding 7](#_Toc94179077)

[LINQ Queries 8](#_Toc94179078)

[Other technologies such as Blazor or PHP 8](#_Toc94179079)

[Any technical difficulties with Visual Studio and ASP.NET 8](#_Toc94179080)

[Application Evaluation 10](#_Toc94179081)

# Updated Background

The background will mention any changes in the proposed features of the project such as if we added features, removed features, and why. The team had to remove a lot of the proposed features, this was due to the lack of experience the team had in coding as the team wanted to make some aspects in which they did not know how to code and was in the features, so those characteristics had to be ridden of. New features were added into the list as the team wanted to add as well as take away features so everything will be balanced out.  
  
The features that had been dispersed was everything after viewing a movie such as adding movie to basket, paying for the movie and so on. We had to get rid of adding movie to the basket because the team had no experience of coding a basket feature and they had no extra time in which they could go and figure it out, in addition of having no basket the Customer could not pay for anything so paying for a movie feature had to be removed from the features list furthermore we decided on that our application will revolve around being a streaming website and not a movie buying website. Many of the Staff features we could not include in our application such as adding a movie, removing a movie, or editing a movie, by the end that only thing the staff could do was sign up/login and access the staff portal. The reason for there not being any staff features is that the team spent majority of their time fixating on the Customer features which they thought would be the main priority of the application but when they finished the Customer features and realised the staff had a lot to include as well it became a big issue.  
  
The added features include a carousel and slide on the home page of the application, this was made as the team wanted to add something in which they thought would balance out all the features that were taken away and it would be something in their application in which would outstand from the other teams. The purpose of the Carousel and slide is to show movies to the Customer on the home page, this makes it easier to view movies which have been newly added or the in which the GUI suggests to the Customer they should watch based on their viewed movies.  
  
The team did not add any new pages to the proposed features or pages that they had decided on as they knew they had already met the features they wanted or needed, so adding more pages will be just be more work on top of what they already had.

Updated Feature List  
The Customer will be able to Register an Account  
The Customer will be able to Sign In with their own unique username and password  
The Customer will be able to View Movies  
The Customer will be able to View the Movie Descriptions  
The Customer will be able to Use a carousel   
The Customer will be able to Watch Trailers Via Carousel  
The Customer will be able to view movies by a slide  
The Customer will be able to Search Movies  
The Customer will be able to update/edit their details  
  
The Staff can Register an Account  
The Staff can Login  
The Staff can View the Movies  
The Staff can create another staff account for additional users  
The Staff can edit Reviews  
The Staff can edit Customer Details  
The Staff can Allow Authorization  
The Staff can handle modernization of the CRUD Pages

Updated Use Case Diagram(s)Diagram

Description automatically generatedDiagram

Description automatically generated

# Updated ERD Diagram(s)

# Project Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST | EXPECTATION | SCREENSHOT | COMPLETED | IMPROVEMENT |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Technology Evaluation  
  
The advantages of ASP.NET Core versus legacy ASP.NET

As ASP.NET Core is the newer and more improved version of the .NET framework, there are several advantages that come with it. Some of the key advantages of using ASP.NET Core versus legacy ASP.NET are that the former is open-source as well as cross-platform, broadening the spectrum of what the framework can be used for. One other advantage lies in the fact that Core has robust Cloud support, and supports modular architecture better than the legacy ASP.NET. One reason for teams to use ASP.NET instead of Core could be that a given project might have been in development before its introduction, consequently allowing teams the choice of not using Core as it would take time to learn and deadlines might have to be met soon. On the other hand, ASP.NET Core might be used instead for the fact that the interface looks better, as well as an unconscious need to follow the market updates, which means using a newer platform. Technologically, .NET Core has more advantages, such as: more precise dependency control, faster performance, and a better interface. Ultimately, ASP.NET Core is “better” because it is more recent, therefore it will receive more updates and feel more up-to-date.

## Razor Pages versus MVC

MVC is action and entity-focused while Razor Pages are page-focused. This interesting fact makes for a specific crutch of using MVC, as entities in MVC applications mostly start with simple CRUD operations, making it typically short-lived and easy to have controllers become too full, as more actions are needed. Razor Pages on the other hand allows for each page to focus on one activity each, giving them space by making them smaller. Another significant advantage of Razor Pages over MVC is that they are simple, due to its architecture being condensed, page-focused, and intuitive. While there are certainly parallels to draw between the two, Razor Pages maintains a strict separation between the markup and page model. This lack of separation in Web Forms made unit testing difficult and, in many ways, violated the separation of concerns principle.

## The strengths and weaknesses of the EntityFramework and Scaffolding

EntityFramework is a tool that enables developers to work with data using objects of domain specific classes without focusing on the underlying database tables and columns where this data is stored. Some advantages of using it include: auto generation of code, reduction of development time and cost, allows multiple conceptual models to be mapped to a single storage schema, and it provides unique syntax for all object queries whether it is database or not. Some disadvantages of using EntityFramework are: lazy loading (loading items only when the user can see them), complicated syntax, It does not work if there is a need to change any schema of the database. An update would be needed to the schema on the solution, and that Its logical schema is not able to understand business entities and relation among each other. As for Scaffolding, its main advantage is that it saves time building applications, as it generates code that can be built on.

## LINQ Queries

Language-Integrated Query (LINQ) is the name for a set of technologies based on the integration of query capabilities directly into the C# language. It is used to retrieve data from different sources and formats. It is used in a similar way to SQL, although it is built into C# and VB.NET instead. LINQ queries return results as objects, and it allows the user to use and object-oriented approach and to not worry about turning different formats of results into objects. LINQ queries have many advantages, such as: offering a common syntax for querying any type of sources of data, closing the gap between relational and object-oriented approaches. Some disadvantages of LINQ include, but are not limited to: it is not complex enough to write complex queries like SQL, its performance is degraded if the LINQ query is not written correctly, and the fact that LINQ does not take full advantage of all SQL features, such as cached execution plan for stored procedures.

## Other technologies such as Blazor or PHP

Blazor is an amalgamation of the words Browser and Razor (mentioned earlier), and it is a SPA, or a Single Page Application development framework. It works by substituting the need to have to execute Razor views on the server in order to present HTML to the browser, but instead it executes these views on the client. Since Blazor offers access to different renderers, developers have an easier time developing UI not only for web applications but also for native mobile apps, for example. Its use also brings many benefits, among them are: Blazor runs in a memory-safe, sandboxed environment and is fast when it comes to execution, and Blazor-made applications can be deployed and executed like static files, where the machines don’t have .NET. When it comes to Microsoft Blazor, the development is conducted using C# which makes it an easier transition for the .NET developer.

## Any technical difficulties with Visual Studio and ASP.NET

With the database being managed by m  
  
  
Team Evaluation  
The Blockbuster project evaluation will contain information about everything that happened after the design aspect such as how well or bad did the implementation go. We will be evaluating the team and the project piece by piece.  
  
First I wanted to mention how we planned to implement each part of the web application. Each week we were set a task that we had to complete by the next such as CRUD Pages, seeding the databases and so on, so each week a task that was set by the teacher was completed and that was how we planned the tasks. The Order of the tasks was again set by the teacher but it was mainly having all the details of each product, customer and everything ready first then allowing us to implement things such as buttons, different pages such as my account, a staff portal and basket.  
  
So each week the plan would be to complete the task set by the teacher, if we had completed that we would move onto next week’s task in the order,  
  
The project planning tools we mainly used were social media, we used social media platforms to communicate with each other and have weekly meetings, this allowed a team leader to set tasks and deadlines in addition with being able to see if they were met and if not why and keep a record of who did what by when.  
  
Where We Went Wrong is When we seeded the data into the database we were confronted by a complication that some of the data would not seed, we found that the issue was brought up by unsuitable data in the DB initialiser and the data that was unfitting was Regular Expressions that had no reason or relation to anything in the solution and in many pieces of data there primary keys starting at 0 and the solution to fixing these issues was commenting out the regular expressions and changing the primary keys that were 0   
to 10, this allowed the data to be seeded.  
  
In addition, we had a made a User Interface using bootstrap and when implementing that UI onto the Visual Studio project, everything in the footer and header had been mixed up allowing headers and footers to clash, we fixed this by getting rid of the bootstrap project out of the Visual Studio project, working on it a while longer and when it was finished we then merged them together.  
  
What went well was everything else as we did not meet any other problems with anything else, as images were easily implemented with the movies, all the CRUD pages seemed to work, overall nothing else was an issue.  
  
If we could change something about our project it would be the products of the project, we tried making something better than Netflix or any other streaming website which was a bit too ambitious, if we had made a application based on a product which had not application for it we would not have any expectations to meet apart from our own.  
  
Haroon Sadiq took the task of making sure a good User Interface was generated as this was something which had not been required and could get us an outstanding grade, he also made sure the merge of both projects went well, furthermore he disputed any inconsistency with the project design and the features, as if there was something which we had done was not listed in the features he would list it down, or if there was something which we had not covered in the features list he would let the team know.  
  
Mauro had the task of filling in the inconsistencies, as if Haroon had notice that a piece of the code was missing he would let Mauro know, who would then fill in that piece and make sure it all matched up with the rest. Mauro also had the tasks of inputting the actual products of the application such as the name, review, description and so on.  
  
Hassan had the task of making sure all the weekly tasks for the code were generated and working, this meant watching each video and being able to implement the code into Visual Studio and have a running application with all the necessity features such as logging in/register, a products page, staff portal, seeding of the data and so

Application Evaluation  
The features that were met was a Customer would be able to login using an unique username and password, and they would be able to view the movies such as the review or description.  
  
All of the other features were not met, the reason for this being is because we tried making a project that was trying to exceed applications that had already been built as our thought was we needed to make something similar or better to Netflix so we took all of us Netflix’s’ requirements or what they had done and made a feature that would either be similar to them or better but our programming/coding experience had let us down. So, most requirements were not met due to them being tasks that were outside of our limit.  
  
The whole project is 60% complete as we have the basic features such as logging in or register a customer, viewing a product and then a stop. The next and final features would have been adding the product to a basket and then paying for the product but those features we had no time to complete, additionally the application did not allow for any staff features to be done, just the customer side.   
  
The features that needed more work are the basket and recommendation of movies. The team had planned that the application will recommend movies for the Customer based of what the Customer had liked or previously watched but they learnt when they started to make this feature, that it required code that they had not yet learnt and for them to learn it there needed to be time to which they did not have. The next feature which needed more work to be implemented was the basket, the team had planned on making an order basket in which the Customer would have been able to see which Movies they had added and they could remove but this feature was planned to be generated at the end and again the team had little to none experience on making a basket and no time in learning.  
  
Just as I mentioned before if we could extend or develop our application with the right time and experience, we would have generated something similar or better than any other streaming website but as we saw first-hand if it’s not broken don’t fix it. Hopefully if we ever come across another team project similar to this one in the future, we could extend and develop on this and actually complete our vision of this project.  
  
Our weakness was our coding experience if we had more experience on how to code more complex features, we could have gotten this project complete as being able to add a basket and the staff features.  
  
  
Our Strengths were being able to see a task was finished once it was started, this meant that once we had started a task such as seeding the database we would not stop until that task was completed before moving onto the other tasks, but sometimes this could be seen as a weakness as at sometimes two people would have been working on the same task which only needed the focus of one.