



Best Plan Advisor

FINAL PROJECT REPORT

GAVIN COLLINS

Abstract

The continuing growth of online retail shopping, the high number of mobile operators in Ireland and the complexity of mobile products means a consumer may not know where to look when searching for the best suited mobile plan to their needs. The range of products is vast, and the differences in terms of value may not be very apparent to a consumer that would usually be more comfortable discussing options with a sales agent. This project attempts to bridge the gap, allowing a consumer look across the entire market place on a single page web application. This application will perform some of the duties completed by an agent, and is also easy to use and understand, so it can be used by almost everyone. What I have developed is a single page application, using .NET core with a Blazor server framework, to deliver an interactive UI, supported by code written in C# and html which renders data to the UI. The data which relates to mobile plans is easy to add, change or update and will use rendering logic as defined in the Blazor components or razor pages. The future development is dependent on the data, which is stored in text format in a json file. Additional products may be incorporated into the data, allowing a further analysis to take place into bundled products and additional services, with associated discounts. The application could be integrated with a CRM platform, or simply used by an agent to assist with a sale. Web forms could also be developed to allow product owners maintain their own products in the data store.

Acknowledgements

I would like to thank Dublin Business School for allowing me to complete my studies in Higher Diploma in Science in Computing.

I would also like to thank my project supervisor Harnaik Dhoot for his support, encouragement and guidance during the course of my final project.

Contents

1.	Introduction	3
1.1	<i>Aim of the project</i>	4
1.2	<i>Scope of the project</i>	6
1.3	<i>Approach used in carrying out the project</i>	7
1.4	<i>Assumptions (if any) on which work is based</i>	8
2.	Background/Literature Review	9
3.	Specification and Design	11
3.1	<i>Overview</i>	11
3.2	<i>Functional and non-functional requirements</i>	12
3.3	<i>Software Architecture</i>	13
3.4	<i>UI Design</i>	13
3.5	<i>Input/Output</i>	15
3.6	<i>Tools and Platforms</i>	16
4.	Implementation	17
5.	Testing & Results	26
6.	Conclusion & Future Work	36
	<i>Appendix A</i>	38

1. Introduction

With a high number of mobile network and virtual operators, the Irish mobile market place can be a confusing place for consumers. Vodafone, Three and eir Mobile are the three main licenced mobile operators in Ireland. In addition, there are a number of MVNOs (Mobile Network Virtual Operators) 48, GoMo, Lycamobile, Post Mobile, Tesco Mobile and Virgin Mobile that offer a competitive product to consumers.

Each of the aforementioned mobile operators offer a choice of SIM only plans with varying benefits in the form of allowances and monthly costs often subject to contract from 30 day to 12 months.

By consolidating some of the key values and features relating to their usage and behaviours, a prospect or consumer can identify the most suitable brand and mobile plan on a SIM only plan that will suit their needs and provide the best value.

My project is a Best Plan Advisor application for mobile. This single page web application offers a consumer an easy to use user interface to allow them to search for an informed choice of most suitable mobile plan based on customers own feature requirement relating to voice, text & data usage. The consumer may also search for a plan on other key value added features such as discounts, special offers, cheap international calls or purchasable service add ons to help them make the correct choice when committing to a contract.

This web application may also be used by a retail or telesales agent while speaking to a customer, by taking their requirements while assisting a customer to make a purchase.

This web application could be further developed to include other service considerations such as including handset prices, total cost of contract, bundling other services such as landline, TV and broadband.

1.1 Aim of the project

This project aims to remove or reduce the confusion when selecting the mobile plan. The single page web application will in practice, enhance an online purchase journey. Taking the role of the agent, a prospect can search features across multiple network brands and find the most suitable mobile plan that fits with their requirement.

With the use of searchable 'Tags' on each mobile plan, a user will be capable of searching a plan based on their preferences, whether that be the most cost effective, or by searching for features such as unlimited data, 5G, add-ons for cheaper international calls and so on.

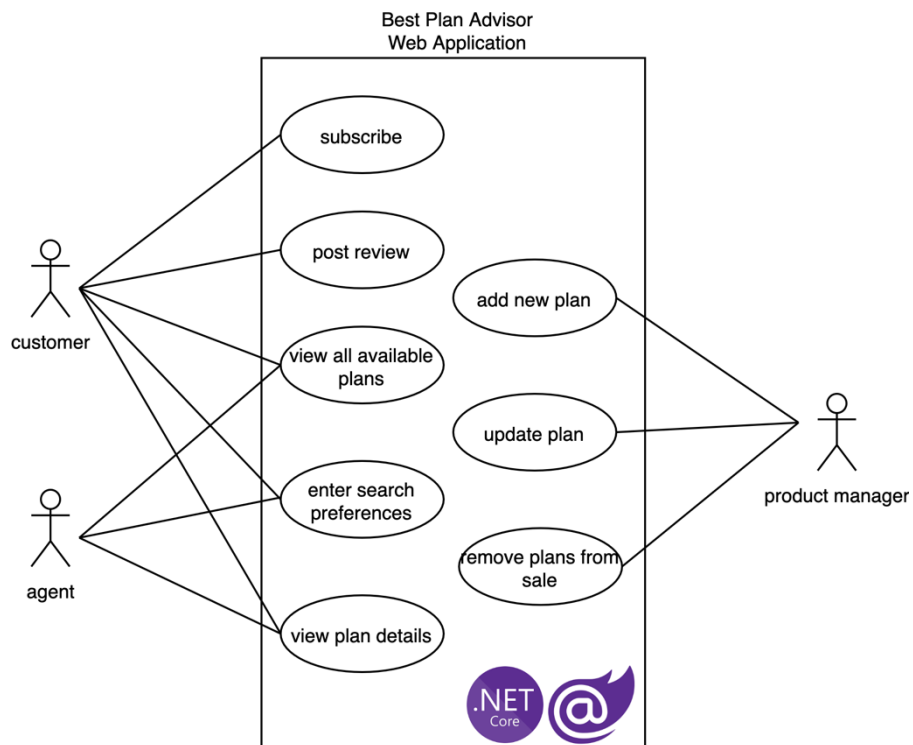


Fig.1: Best Plan Advisor Use Cases

The project will deliver a .NET Web Application that will present a simple user interface that will filter mobile plan recommendations based on user input via the search field. This method is preferred to the initial proposal of a submit behaviours model. As mobile plans become very similar and most offer unlimited allowances, it is specific user requirements or features that will separate and enable a personal recommendation to the end user.

The MoSCoW model below illustrates the required functionality of the Best Plan Advisor Web Application (MoSCoW Method, 2020)

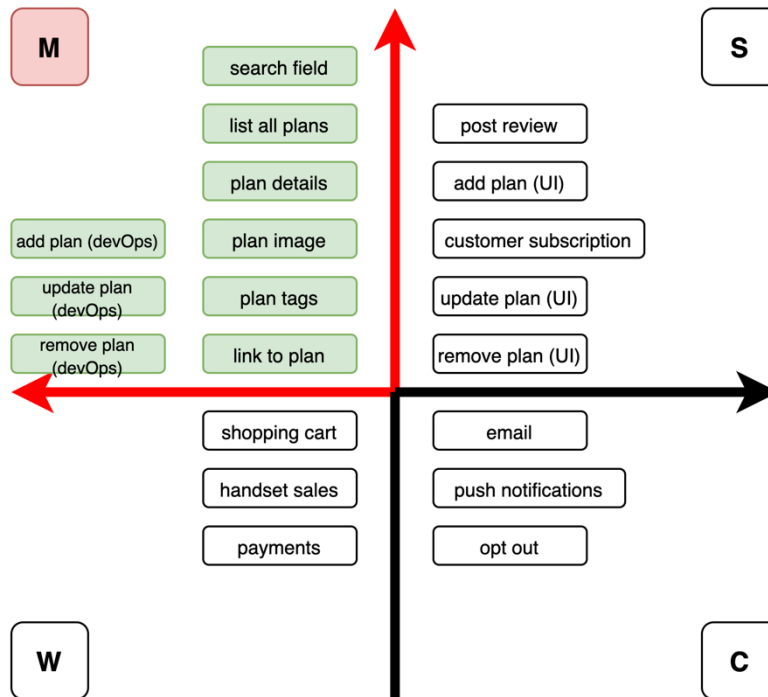


Fig.2: Project Requirements MoSCoW method

1.2 Scope of the project

My project will deliver a web application to present a consolidated mobile plan market place, where the end user will have the capability of searching across the Irish market and filter their preferences by using a search tool. This application will provide details that may not be evident to a prospect searching the market. This will allow for an informed decision to be made before a contract is signed by the prospect.

The following is a scope statement for this project:

Project Name		Best Plan Advisor	
Scope Description	<p>IN SCOPE</p> <ul style="list-style-type: none"> - Web application to support customers and sales agents for products across multiple brands - Search Tool filtering functionality to match key values to product tags - Data storage file to store and exchange data between server and web application - Support for updates to data storage on behalf of mobile plan product owners or managers - Selectable link to allow redirection to operator online stores <p>OUT OF SCOPE</p> <ul style="list-style-type: none"> - Sales supported on web application - Taking of payments for goods - Handset sales 		
Project Deliverables	<ul style="list-style-type: none"> - Customer facing web application - Mobile plan data across multiple operators - Project report & artifacts 		
Acceptance Criteria	A functional web application allowing customers to browse products with the aid of a search tool to filter products.		
Constraints			
Assumptions	<ul style="list-style-type: none"> - No subscription will be required for browsing - Subscription would only be required for promotional and marketing notification opt in - Discounts based on mobile plan only, no service bundle discounts applied for multi-play services i.e. mobile and fixed service discount 		

Table 1: Project Scope Statement

1.3 Approach used in carrying out the project

1.3.1 Planning Phase

The project plan was composed by highlighting the key dates as provided by DBS for project delivery, followed by identifying the key tasks of the project and working them into a project plan. The plan was then illustrated into a basis Gantt chart and submitted as part of my proposal. I used this chart to track progress of my project and assist with the timely delivery of the project.

See below:

Week ending	17-May	24-May	31-May	07-Jun	14-Jun	21-Jun	28-Jun	05-Jul	12-Jul	19-Jul	26-Jul	02-Aug	09-Aug	16-Aug	23-Aug
Sprint 0															
Proposal															
Supervisor Allocation															
Meeting with Supervisor			★												
Sprint 1															
Document Project Requirements & Scope															
Preliminary Research and Design															
Design and Build															
Sprint 2															
Update Interim Report															
UI Design and CSS Build															
Test Design & Execution															
Sprint 3															
Interim Report Submission															
Demonstration															
E2E testing and Product Verification															
Sprint 4															
E2E testing and Product Verification															
Final Test & Regression															
Sprint 5															
Update and Complete Project Report															
Presentation															

Table 2: Project Gantt Chart

1.3.2 Proposal & high level concept

This phase included outlining of the high level project proposal and scope. This included a high level specification of the project design and proposed front end UI design. I also highlighted my project learning objectives.

1.3.3 Review with supervisor

Met with supervisor, Harnaik Dhoot. Meeting agenda included a presentation of my proposal. Supervisor recommendation was to study MS .NET Core Framework, ASP.NET and Blazor Server Model and consider these methods when developing my web application.

1.3.4 High Level Requirements

Documented high level requirement following MoSCoW model (see Fig.2)

1.3.5 Research

Completed research of .NET Core, ASP.NET and Blazor Server Application Mode running on .NET Core

1.3.6 High level design

Designed data model and code build

1.3.7 Application build and test

Completed build and unit test of web application

1.4 Assumptions (if any) on which work is based

- No subscription will be required for browsing
- Subscription would only be required for promotional and marketing notification opt in
- Discounts based on mobile plan only, no service bundle discounts applied for multi-play services i.e. mobile and fixed service discount

2. Background/Literature Review

The context of the project is to develop an online web application which allows the user to search and compare mobile plans offered by Irish mobile operators all in the one place, without the need for visiting different sites. The user can browse the entire rendered list of mobile plans and products. In addition, the search feature allows a user to search for key features based on a tag associated with the mobile plan.

Since most mobile plans are quite similar in that they offer unlimited minutes, texts and data, this platform would be suitable for a prospect searching for key features such as 5G, international calls etc. These features may not always be clearly evident from their online or media advertising.

Users of this web application would typically be a customer, out of contract and looking for a SIM Only option for their mobile to reduce their costs and potentially improve their services and allowances. The user profile would typically be looking to reduce the cost of their monthly bill to their current provider by signing up to a new plan with their provider, or switching to a new provider. Otherwise, users may be a new customer, looking for a generous plan at a reasonable cost.

This web application would also be useful to retail agents dealing with customers in a store or telesales, promoting the best mobile plan to suit a customer’s mobile requirements. It could be used with or integrated to a CRM system, allowing agents to have an additional feature available to them with assisting a customer.

It provides a quick response to key customer requirements and will reduce the time spent searching for options online.

The Software Development method to develop this Web Application will follow Agile methodology (Agile Alliance, 2019). The software classes and components are built in incremental iterations. This allows for the testing and detecting of defects early in the process, followed by frequent incremental improvements.

url	Description
https://channel9.msdn.com/Series/ASPNET-Core-101	This was the starting point for me as recommended. Includes a easy to follow walk through of developing web applications using ASP.NET framework, developing an API, use of razor pages (c# and html) and an introduction to Blazor. Tutorial by Lesley Richardson & Scott Hanselman
https://dotnet.microsoft.com/learn/aspnet/blazor-tutorial/intro	Getting started with Blazor, including installation and web application demos
https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/Blazor-Tips-and-Tricks?term=blazor&lang-en=true	Good explanation of some of the features of Blazor. Presented by Robert Green & Ed Charbeneau
https://channel9.msdn.com/events/dotnetConf/Focus-on-Blazor/Welcome-to-Blazor?term=blazor&lang-en=true	Welcome to Blazor. Introduction to some of the key features of Blazor framework for web application development. Includes a description of Blazor with some practical applications to demonstrate. Presented by Dan Roth

https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/Blazor-Part-1?term=blazor&lang-en=true	Walkthrough of Blazor, good demonstration also showing local storage using json binding. Presented by Sam Basu & Ed Charbeneau
https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/Blazor-Part-2	Part 2 of above, using WebAssembly not Server project. Presented by Sam Basu & Ed Charbeneau
https://channel9.msdn.com/events/Visual-Studio/Visual-Studio-for-Mac-Refresh/VS02?term=blazor&lang-en=true&pageSize=15&skip=15	Building Blazor applications on a Mac. I am using a MacBook, but using windows version of VS using Parallels Desktop. I had tried the VS Mac edition however the interface is slightly different, may take me a while to get used to, however this video again covered some of the features and useful demonstrations. Presented by Dan Roth & Kendra Havens.

Table 3: Literature Review

3. Specification and Design

3.1 Overview

The solution delivers a web application that presents the user with a simple user interface with a full list of products, those being mobile plans. The mobile plans will be presented in a rendered tile format on the home page which when selected, will redirect to a page presenting a more detailed description of the mobile plan. This includes features, discounts, allowances and other details which may not be clear when viewing an advertisement. This could be developed further to include data that is usually buried in the terms and conditions, but presented in bullet format so it is more clear to the user.

A search tool will also be provided to allow a user to search for their plan, based on their expected behaviours and required features. By using key words in the search, as would be used if interacting with a retail or telesales agent, a user will be assisted in making an order and directed to an online order of a mobile product or plan. The result of using key words will narrow the search by tag association, resulting in the presentation of fewer mobile plans to the user. This could be developed further by displaying selectable tags on the home page which perform filtering functions for users who are not familiar with the tags.

3.2 Functional and non-functional requirements

The functional requirements for Best Plan Advisor are documented in the table below.

Functional Requirements		
Ref	Requirement	MoSCoW
1	As a consumer, I would like to view all available mobile plans on the market across multiple mobile networks	M
2	As a consumer, I would like to select and view detailed features of a mobile plan	M
3	As a consumer, I would like to view the image provided by operator relating to a mobile plan	M
4	As a consumer, I would like to search for key feature to reduce the full list and narrow the search for my chosen plan	M
5	As a consumer, I would like to post a comment relating to a mobile plan	S
6	As a consumer, I would like to subscribe to Best Plan Advisor and avail of marketing communications and special offers	S
7	As a consumer, I would like to complete my purchase selecting a link to online store	M
8	As a subscribed consumer, I would like to add my contact details and my preferred contact method so that I can receive details of new mobile plans, updates to mobile plans, special offers and discounts	C
9	As a subscribed consumer, I would like to update my contact details and my preferred contact method so that I can continue to receive information if my contact details have changed	C
10	As a subscribed consumer, I would like to remove my contact details and my preferred contact method so that I no longer will receive information from Best Plan Advisor	C
11	As a subscribed consumer, I would like to opt in for push notifications so that I can receive information on mobile plans	C
12	As a subscribed consumer, I would like to opt out of push notifications so that I no longer receive information on mobile plans from Best Plan Advisor	C
13	As an agent, I would like to view all mobile plans available to offer to a prospect	M
14	As an agent, I would like to search for a suitable mobile plan for a prospect based on their preference and behaviours so that I can make a recommendation	M
15	As an agent, I would like to view mobile plan details so that I can recommend to a prospect	M
16	As an agent, I would like to view customer reviews so that I can recommend to a prospect and feedback to network provider	S
17	As a product manager, I would like to add a new mobile plan to Best Plan Advisor so that consumers and agents can view them	M
18	As a product manager, I would like to update details of my existing mobile plans so that consumers and agents can view them	M
19	As a product manager, I would like to remove mobile plans from Best Plan Advisor so that retired mobile plans will no longer be recommended to a consumer or a prospect	M
Non-Functional Requirements		
Ref	Requirement	MoSCoW
NFR1	Best Plan Advisor web application user interface must be easy to use	M

NFR2	Best Plan Advisor user interface must render depending on screen size and presented content	M
NFR3	Best Plan Advisor must respond within less than 1 sec following user input	M
NFR4	Best Plan Advisor must support multiple concurrent users	M

Table 4: Functional & Non Functional Requirements

3.3 Software Architecture

There were a number of architecture styles considered when choosing the overall structure of the software system to deliver this as a solution. Blazor is a single page application framework (SPA) that uses C# with html, or razor pages instead of requiring JavaScript (Blazor | Build client web apps with C# | .NET, 2019). Using the Blazor Server Model the application can run on the server and render to the UI. The program is constructed loosely as an MVC, where the Model contains data related to the mobile plans. This data is stored in a json file (JavaScript Object Notation) in readable format. The data objects consisting of attribute value pairs and array data types are serialized, transmitted and deserialized and presented on the user interface. The View consists of a full list of mobile plans retrieved from the Model. The user may interact with the view by inserting key words as associated with a mobile plan. The Controller will process the transmission of the key words to the Model and search for tags to present back to the user. The Model will alert the View when an update has occurred. The View will query the Model and render the updated attribute value pairs as associated with the search query change as entered in the search tool on the user interface. Blazor is designed to work together with MVC and razor pages for server rendering needs while using Blazor for client side UI interactions.

3.4 UI Design

Microsoft state the following:

“Blazor lets you build interactive web UIs using C# instead of JavaScript. Blazor apps are composed of reusable web UI components implemented using C#, HTML, and CSS. Both client and server code is written in C#, allowing you to share code and libraries.” (<https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor#:~:text=Blazor%20lets%20you%20build%20interactive,to%20share%20code%20and%20libraries.>)

Using Blazor framework, I have developed a Single Page Application (SPA) using C# and html and added styling using css. The Index hosts the Home page which consist of a branded header & footer. The body of the home page is a rendered tile presentation of the data extracted from the mobilePlan.json file, including some images stored within the Best Plan Advisor project folders. Each of the tiles are selectable and anchor to a PlanDetail page for the selected plan. Above the tiled presentation is a Search tool, which will allow the user to enter their required features. These will link to tags associated with each of the mobile plans, and filter and render down as a new key word is entered into the Search box.

The home page follows the principle design of that presented in the interim report, as shown in the mockups below.

When the web application is running, the data will be presented to the user on the UI. A search field on the UI will allow the user to filter according to their preference, for example, user will specify that their preference is for a 'postpay' plan which allows for 'unlimited' data and '5G'.

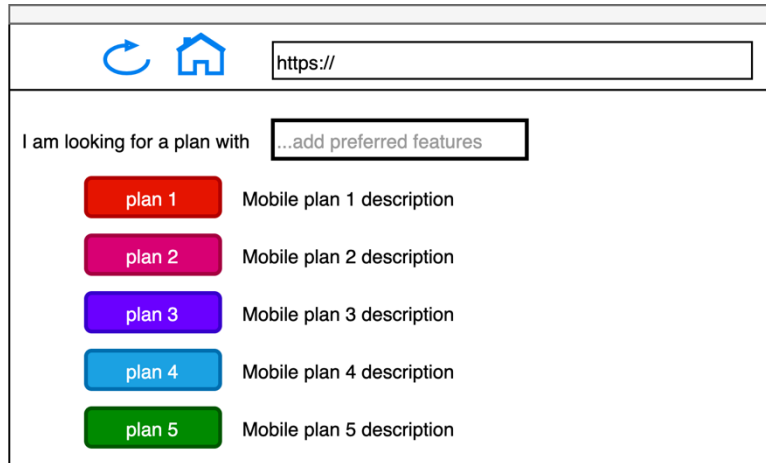


Fig.3: User presented with ListAllPlans

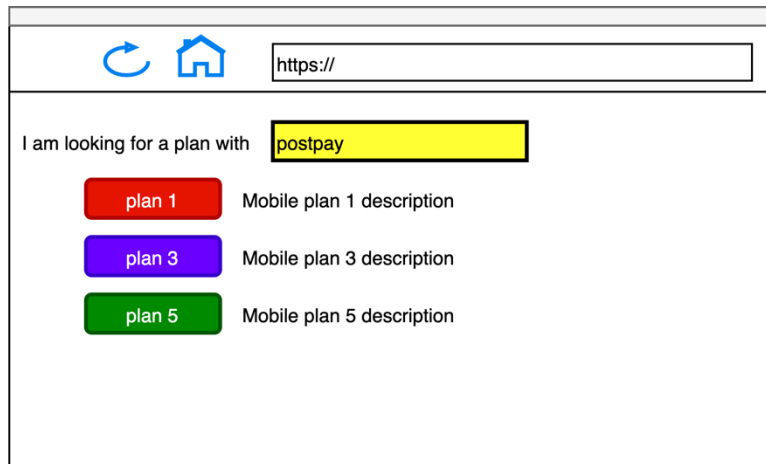


Fig.4: User has entered first preference

In Fig.4 (above), user has entered their first preference. The search field will have a timer where if there is no activity in the search field, data will be filtered based on the search criteria entered.

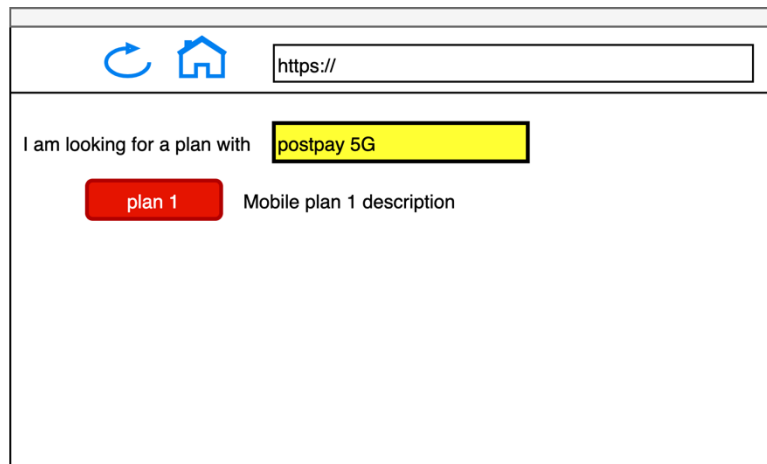


Fig.5: User enters second preference

Fig.5 illustrates the user entering a further preference, further narrowing the search down to one possible option. The plan description will contain a url link to online store where sale can be completed. The url (or in this case a button with a html anchor element) will link out to operator online store.

Instead of presenting the user with a Search button, a search will happen on input and will access the data model over a websocket using SignalR connectivity. This is detailed in the Section 4 under Implementation as functionality within the SearchTool.razor component.

3.5 Input/Output

Input: Mobile plan data stored in a text-based json format, representing structured data based on JavaScript object syntax. This file is used for transmitting data between the web application and the server.

The user will input a key value which associates with a tag which is then associated with a product.

Output: The json file content is rendered and displayed on the web application UI. This will be filtered down in association with input tags entered using the search tool.

3.6 Tools and Platforms

Tools	Visual Studio 2019
Language	C# and html
Platform	.NET Core
Programming Model	Blazor Server

Table 5: Tools & Platforms

.NET is the recommended platform to build the single page application functionality of Best Plan Advisor project as it is generally a fast and secure way of building web applications. .NET core is a cross platform, open source implementation of .NET. Blazor is a framework that caters for full stack web development with C# and Razor.

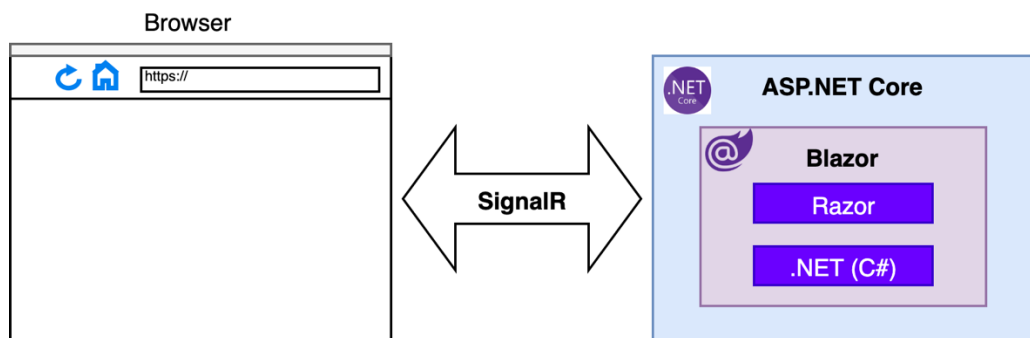


Fig.6: Blazor Server side

With Blazor server applications, all components will be running on the server on .NET core. There is a real time connection established with the browser over a web socket using SignalR. UI events that happen in the browser are sent serially to the server and handled by components which then render. Blazor keeps track of any changes made on the UI using an diff algorithm. Any changes will be sent to the browser and applied to the live model on the browser.

Blazor server apps simplify the overall architecture by removing the need for standing up an API between client and server. All code stays on the server, so as more code is added to a class or component, the web application size stays the same and does not grow.

4. Implementation

Blazor is a full stack solution for building web applications using .NET. It is a Client side web UI framework which allows the development of an entire web application using only .NET. It also allows you to write reusable web UI components with C# and html which result in responsive, single page applications without having to write any JavaScript. .NET code is shared with both the client and the server

In my project, I have implemented my .NET code using Blazor server model. This means that my components are on the server on top of .NET Core. Blazor server sets up a real time connection with the browser typically over a websocket using SignalR. Any UI events that happen in the browser are sent to the server and handled by the components which then render.

Blazor keeps track of any changes that are made to the UI using a diff algorithm. Any changes detected are then sent back to the browser and are applied to the live DOM (Document Object Model). Blazor server uses a thin client with will provide good performance on most devices and browsers.

.NET code runs on a full .NET Core runtime allowing access to all .NET features and debugging. Blazor server apps simplify the overall application architecture by eliminating the need to stand up additional APIs or http endpoints.

Blazor Server was launched with .NET Core 3.1 LTS (Long Term Support).

The following is a description of the components that have been developed to support Best Plan Advisor with a high level description of the functionality of each component.

Best Plan Advisor Project
<p>Best Plan Advisor Blazor Server Project</p> <p>This is an ASP.NET web service application created by the Blazor Server Framework running on .NET Core, in this instance targeting netcoreapp3.1.</p> <pre><PropertyGroup> <TargetFramework>netcoreapp3.1</TargetFramework> <RootNamespace>Best_Plan_Advisor</RootNamespace> </PropertyGroup></pre>
Startup.cs
<p>Setting up of Blazor server and addition of Blazor Server services</p> <pre>public void ConfigureServices(IServiceCollection services) { services.AddRazorPages(); services.AddServerSideBlazor(); //IPlanDisplay and JsonPlanDisplay services services.AddSingleton<IPlanDisplay, JsonPlanDisplay>(); }</pre> <p>Endpoint is also added to manage the real time connections with the browser.</p>

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapBlazorHub();
    endpoints.MapFallbackToPage("/_Host");
});
```

_Host.cshtml

Single razor page created by Blazor Server where a blazor server script will be added which will establish a real time connection back to the MapBlazorHub

```
<script src="_framework/blazor.server.js"></script>
```

Rendering the root component of the app using the following component tag

```
<app>
    <component type="typeof(App)" render-mode="ServerPrerendered" />
</app>
```

MainLayout.razor

This is the main layout component for the web application.

This project is using a header, main and footer tab for the single page layout.

Inherits from LayoutComponentBase

```
@inherits LayoutComponentBase

<header>
    <h1>Best Plan Advisor</h1>
</header>
<main>
    @Body
</main>
<footer>
    Best Plan Advisor
</footer>
```

The @Body property specifies where the content of the routable components should be rendered.

App.razor

Built in router component, which will find all routable components in the application and navigations to those components.

Also specifies that default layout should be MainLayout

```
<Router AppAssembly="@typeof(Program).Assembly">
    <Found Context="routeData">
        <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
    </Found>
    <NotFound>
        <LayoutView Layout="@typeof(MainLayout)">
            <p>Sorry, there's nothing at this address.</p>
        </LayoutView>
    </NotFound>
</Router>
```

Index.razor

The Index razor component contains the Home page for this web application. This component renders at the root of the app.

This page uses dependency injection to inject the PlanDisplay class into the index component.

```
@page "/"
@page "/tag/{tag}"
@inject Best_Plan_Advisor.Data.IPlanDisplay PlanDisplay
```

SearchTool a customised component (see SearchTool.razor) providing a search feature to the user which uses html attributes to allow the user 'Search mobile plans...' and will also allow for a call back if there is a change to the user search attribute, calling the search method.

```
<SearchTool placeholder="Search mobile plans..." SearchQueryChanged="Search" />
```

An unordered list of mobile plans is rendered on the home pages as plan-list and plan-list items detailing some features to be displayed as summary to the user.

MobilePlan is a customised component (see MobilePlan.razor) that will display and render each of the mobile plans in the .json file (see mobilePlan.json)

```
@if (plans == null)
{
  <p>Loading plans...</p>
}
else
{
  <ul class="plan-list">
    @foreach (var plan in plans)
    {
      <li class="plan-list-items">
        <a href="@plan.Id">
          <MobilePlan Plan="plan" />
        </a>
      </li>
    }
  </ul>
}
```

OnInitializedAsync is a Blazor component lifecycle event, which when initialized will call to the PlanDisplay to retrieve the full list of plans.

When a Search occurs, a query will be sent to PlanDisplay and retrieve a filtered list of plans to display on the home page.

```

@code {
    IEnumerable<Best_Plan_Advisor.Data.Plan> plans;

    [Parameter]
    public string Tag { get; set; }

    protected override async Task OnInitializedAsync()
    {
        //pass Tag into
        plans = await PlanDisplay.GetAllPlans(Tag ?? string.Empty);
    }

    async Task Search(string query)
    {
        plans = await PlanDisplay.GetAllPlans(query);
    }
}

```

PlanDetails.razor

The PlanDetails razor component renders the selected details of the plan or product from the home page, and is routed by value, that being the planId. A dependency injection of the PlanDisplay is required to pull values.

```

@page "{planId}"
@inject Best_Plan_Advisor.Data.IPlanDisplay PlanDisplay

```

This razor component is configured to render out details relating to the plan, including a planname, a graphic of the plan, some details of features of the mobile plan, a url that the user can follow to complete a purchase, or sale in the case of agent.

```

<div class="plan">
    @if (plan == null)
    {
        <h1>Sorry, no plan found!</h1>
    }
    else
    {
        <h1>@plan.PlanName</h1>
        <div class="operator-link">
            <h3>Website: @plan.Website</h3>
            <h3>Operator: @plan.Operator</h3>
        </div>

        <div class="plan-features">
            <h3>Features</h3>
            <ul>
                @for (int i = 0; i < plan.Features.Length; i++)
                {
                    string id = $"features{i}";
                    <li>
                        <label for="@id">@plan.Features[i]</label>
                    </li>
                }
            </ul>
        </div>
        <*/a class="continue" href="@plan.Website"> Continue to purchase this plan</a*/>
        <div class="proceedtoorder">
            <a href="@plan.Website"><i type="button" class="btn btn-primary">Select to Order</i></a>
        </div>
    }

```

A list of tags will be rendered on the bottom of the page, and these will become the searchable items that render down the home page to the plan of your choosing.

```

<div class="tags">
    <h4>Tags</h4>
    @foreach (var tag in plan.Tags)
    {
        <a class="tag" href="@($"tag/{tag}")">@tag</a>
    }
</div>

```

Plan.cs

Plan.cs defines properties that will be used throughout the Best Plan Advisor project. These properties align with the data stored in the mobilePlan.json file.

Plan.cs is a c sharp class that represents the data.

```
public class Plan
{
    2 references
    public string Id { get; set; }
    2 references
    public string Operator { get; set; }
    0 references
    public string Segment { get; set; }
    4 references
    public string PlanName { get; set; }
    1 reference
    public string PlanCost { get; set; }
    2 references
    public string[] Features { get; set; }
    0 references
    public string Discount { get; set; }
    1 reference
    public string Voice { get; set; }
    1 reference
    public string Text { get; set; }
    1 reference
    public string Data { get; set; }
    0 references
    public string Source { get; set; }
    2 references
    public string Website { get; set; }
    1 reference
    public Uri Image => new Uri($"images/card/{PlanName}.png", UriKind.Relative);
    1 reference
    public Uri NetworkBrand => new Uri($"images/brand/{Operator}.png", UriKind.Relative);
    2 references
    public string[] Tags { get; set; }
}
```

JsonPlanDisplay.cs

The class implements the IPlanDisplay interface to call method GetAllPlans.

```
public class JsonPlanDisplay : IPlanDisplay
```

The main function of this class is to takes in all of the data from the mobilePlan.json file and converts using Serializer and Deserializing of the object.

```
IDictionary<string, Plan> plans;
InMemorySearch searchProvider;
0 references
public JsonPlanDisplay()
{
    //Read json file, case insensitive
    //serialize and deserialize
    var json = File.ReadAllText("mobilePlan.json");
    var jsonOptions = new JsonSerializerOptions
    {
        PropertyNameCaseInsensitive = true,
        AllowTrailingCommas = true
    };
    plans = JsonSerializer.Deserialize<Dictionary<string, Plan>>(json, jsonOptions);
    searchProvider = new InMemorySearch(plans);
}
```

Once the list of objects is retrieved from the json file, a list is created where IE numerable is preferred to list.

This allows a for each type logic to apply to the list.

```
//List of objects deserialized and retrieved from json file, making a list
//for each over items in list by using IEnumerable
3 references
public Task<IEnumerable<Plan>> GetAllPlans(string query)
{
    if (string.IsNullOrWhiteSpace(query))
    {
        return Task.FromResult<IEnumerable<Plan>>(plans.Values);
    }
    return Task.FromResult(searchProvider.Search(query));
}
```

iPlanDisplay.cs

IPlanDisplay in an interface class which is implemented in the JsonPlanDisplay class.

```
public interface IPlanDisplay
{
    3 references
    Task<IEnumerable<Plan>> GetAllPlans(string query = "");
    2 references
    Task<Plan> GetPlan(int planId);
}
```

InMemorySearch.cs

Responsible primarily for search query indexing

```
public class InMemorySearch
{
    readonly IDictionary<string, Plan> plans;
    IDictionary<string, ICollection<(string PlanId, int Count)>> searchIndex;

    1 reference
    public InMemorySearch(IDictionary<string, Plan> plans)
    {
        this.plans = plans;
        BuildSearchIndex();
    }
}
```

Once the list is indexed, a number of foreach statements are used to enable the search function with term being matched and added to searchIndex value.

```
void BuildSearchIndex()
{
    //Search based on Tags
    searchIndex = new Dictionary<string, ICollection<(string, int)>>();
    foreach (var plan in plans.Values)
    {
        var terms = plan.PlanName.ToLower().Split()
            .Concat(plan.Tags.Select(tag => tag.ToLower()))
            .GroupBy(term => term)
            .SelectIGrouping<string, string>, (string Term, int TermCount)>(termGroup => (termGroup.Key, termGroup.Count()));
        foreach (var term in terms)
        {
            if (!searchIndex.ContainsKey(term.Term))
            {
                searchIndex[term.Term] = new List<(string, int)>();
            }
            searchIndex[term.Term].Add((plan.Id, term.TermCount));
        }
    }
}

1 reference
public IEnumerable<Plan> Search(string query)
{
    return query.ToLower().Split()
        .Where(term => !string.IsNullOrWhiteSpace(term))
        .SelectMany(term => searchIndex.Keys)
        .Where(key => key.StartsWith(term))
        .SelectMany(key => searchIndex[key])
        .GroupBy(termCount => termCount.PlanId, termCount => termCount.Count)
        .OrderByDescending(termCounts => termCounts.Sum())
        .Select(termCounts => plans[termCounts.Key]);
}
```

MobilePlan.razor

MobilePlan razor component contains html which renders out some static detail relating to each mobile plan

Passing a parameter into the component, this is the Plan that we render in the plan detail page.

```

<div class="mobile-plan">
  
  <div class="mobile-plan-body">
    <h3 class="mobile-plan-name">Plan Name: @Plan.PlanName</h3>
    <h2 class="mobile-plan-cost">@Plan.PlanCost per month</h2>
    <h3 class="mobile-plan-voice">Voice: @Plan.Voice</h3>
    <h3 class="mobile-plan-text">Text: @Plan.Text</h3>
    <h3 class="mobile-plan-data">Data: @Plan.Data</h3>
  </div>
</div>

@code {
  [Parameter]
  public Best_Plan_Advisor.Data.Plan Plan { get; set; }
}

```

SearchTool.razor

SearchTool razor component contains a html input.

The attributes object allows for any addition attributes added to the search which are then rendered on the input tag. The extra attributes are captured by the AddAttributes parameter in the code and are rendered to the search component.

The bind is configured to create a bind between the search input value and the SearchQuery parameter as defined in the code. This is known as two way data binding. The SearchQuery property is set with a new value each time a new input value is entered. When the value is re rendered, the value of the input is initialized using the value of the SearchQuery property. Changes will occur on input event of the value, oninput event meaning each time a new value in input.

```

using System.Timers;
implements IDisposable

<div class="search">
  <input placeholder="Search..." @Attributes="AddAttributes" @bind="SearchQuery" @bind:event="oninput" />
</div>

@code {
  Timer debounceTimer;
  string searchQuery;

  [Parameter]
  public string SearchQuery
  {
    get => searchQuery;
    set
    {
      searchQuery = value;
      debounceTimer?.Stop();
      debounceTimer?.Start();
    }
  }
}

```

The bind will be triggered on each keyboard entry in the Search box.

Debouncing is configured in order not to trigger the search event with each change to the Search input, but instead after a change and a certain period of inactivity.

I have configured a Timer class into the razor component using System.Timers .

Each time SearchQuery property gets set, the timer that is running will stop, followed by start and initialize.

This calls the OnInitialized method in the code, where the value of the interval = Debounce.

The Debounce parameter is defined in the code as being of value 300 mS.

Once the Debounce Timer elapses, a call will be made to Search event handler.

When the Search event handler is called, it will then invoke SearchQueryChanged event, which is another defined parameter in the SearchTool code of the razor component. This event is specified on the Index.razor component which manages the home page.

```
[Parameter(CaptureUnmatchedValues = true)]
public IDictionary<string, object> AddAttributes { get; set; }

[Parameter]
public int Debounce { get; set; } = 300;

[Parameter]
public EventCallback<string> SearchQueryChanged { get; set; }

protected override void OnInitialized()
{
    debounceTimer = new Timer();
    debounceTimer.Interval = Debounce;
    debounceTimer.AutoReset = false;
    debounceTimer.Elapsed += Search;
}

async void Search(Object source, ElapsedEventArgs e)
{
    await InvokeAsync(() => SearchQueryChanged.InvokeAsync(SearchQuery));
}
```

The timer may require disposal after the value is changed.

```
public void Dispose()
{
    debounceTimer?.Dispose();
}
```

mobilePlan.json

mobilePlan.json file holds all collateral relating to the core products. This data in this file will be extracted and converted into objects that can work with the code, similar to a database. For this project, this file takes the place of a database.

A collection of objects.

```
{
  "1": {
    "id": "1",
    "operator": "vodafone",
    "segment": "Postpay",
    "planname": "RED Unlimited",
    "plancost": "€35",
    "discount": "€25 for first 6 months",
    "voice": "unlimited",
    "text": "unlimited",
    "data": "unlimited",
    "features": [
      "12 month contract\r",
      "Unlimited data\r",
      "Max speed 10 Mbps\r",
      "Unlimited calls and texts to any Irish mobile or landline\r",
      "3 month free trial of Secure Net\r",
      "Weekly rewards courtesy of Fantastic Days"
    ],
    "website": "https://n.vodafone.ie/shop/bill-pay-plans.html",
    "tags": [
      "unlimited data",
      "vodafone",
      "discount",
      "12 month contract",
      "unlimited calls",
      "unlimited texts"
    ]
  },
}
```

5. Testing & Results

Unit black box testing was executed to complete the overall product verifications. These test cases were derived from the requirements that were baselined at an earlier phase in the project. All functional and non-functional requirements must relate to a identified business need of the product. However, just as important is that requirements must be measurable and testable. With that in mind, as part of the requirement analysis, I have compiled and documented test cases to verify the various functions developed in the project. To accompany each test case, I have also documented the acceptance criteria or expected result of each test case. This allows for each test case to be categorised as passed or failed and signed off once verified.

The testing methodology used to complete the test phase of the project is black-box unit testing. It was not possible to complete test driven development (TDD) testing or behaviour driven development (BDD) unit testing as test development had not been kept up to date with the development of the program. On reflection, this would have been of great benefit as these test cases would be beneficial when verifying class or components, and could be repeated if a change was to be made to the code, or run to demonstrate the functionality of the web application.

The test case report is detailed below

Test Case Ref	Test Case Title	Run	Result (P/F)	Complete
TC01	View All	Yes	P	Yes
TC02	Select & view plan	Yes	P	Yes
TC03	View plan image	Yes	P	Yes
TC04	Home Button	Yes	P	Yes
TC05	Search Query	Yes	P	Yes
TC06	Complete Order	Yes	P	Yes
TC07	Add new plan	Yes	P	Yes
TC08	Update/ Modify plan	Yes	P	Yes
TC09	Remove plan	Yes	P	Yes
Progress				100%

Test exit report summary

TC01**Test Case Title:** View all Plans**Description:** View all available mobile plans on the market across multiple mobile networks**Execution Steps:**

1. Go to Best Plan Advice website/Start program
2. All mobile plans appear in the browser in tiled format on home page
3. Compare result to input i.e. json file

Expected Result: All mobile plans display on BPA user interface

The screenshot displays the Best Plan Advisor website interface. At the top, the title "Best Plan Advisor" is shown in green. Below the title is a search bar labeled "Search mobile plans...". The main content area features a grid of mobile plans, each represented by a card with the network logo, plan name, price, and details. The plans shown include:

- Vodafone:**
 - Plan Name: RED Unlimited, €35 per month, Voice: unlimited, Text: unlimited, Data: unlimited.
 - Plan Name: RED Unlimited Max, €45 per month, Voice: unlimited, Text: unlimited, Data: unlimited.
- 3:**
 - Plan Name: Unlimited Flex Max, €30 per month, Voice: 10000 minutes, Text: 10000, Data: unlimited.
- eir:**
 - Plan Name: connect, €29.99 per month, Voice: 45000 minutes, Text: 10000, Data: 10GB.
- 4G:**
 - Plan Name: 20GB, €9.99 per month, Voice: 300 minutes, Text: unlimited, Data: 20GB.
 - Plan Name: 40GB, €14.99 per month, Voice: 300 minutes, Text: unlimited, Data: 40GB.
- TESCO mobile:**
 - Plan Name: 20month, €20 per month, Voice: unlimited, Text: unlimited, Data: 6GB.
 - Plan Name: 30month, €10 per month, Voice: 100 minutes, Text: 100, Data: 1GB.
- Lyca mobile:** (Four identical cards shown)
- Virgin media:** (Card partially visible)

Result: Passed – All plans displayed

TC02

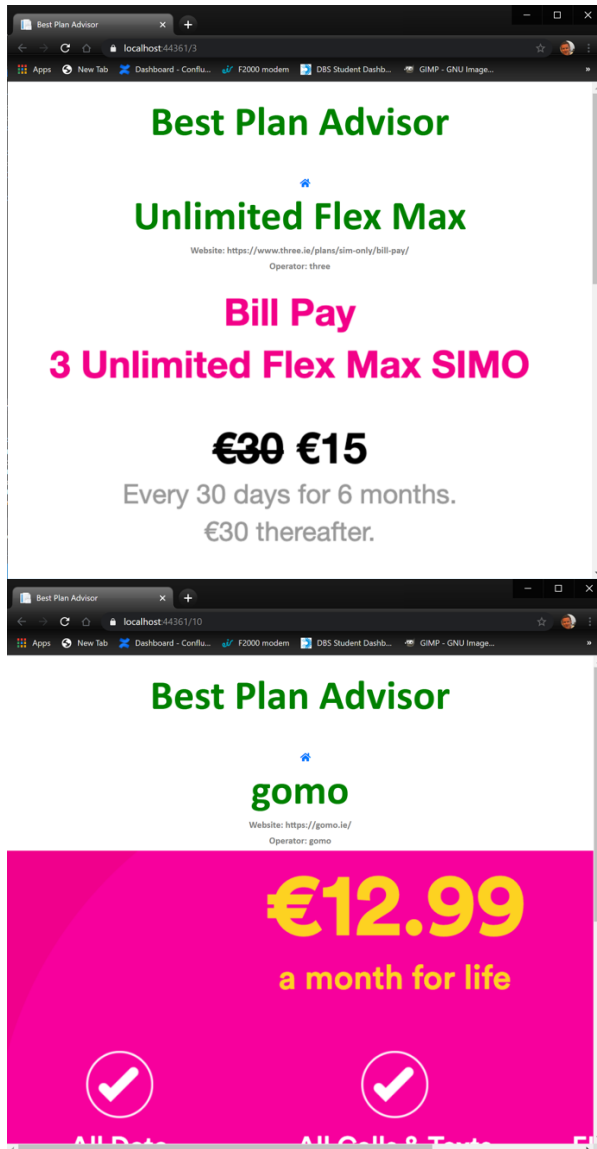
Test Case Title: Select and view plan

Description: Select and view detailed features of a mobile plan

Execution Steps:

1. From BPA home page, select a mobile plan tile
2. Routed to plan detail page

Expected Result: Plan detail rendered to browser



Result: Passed – Examples shown, 3 & GoMo

TC03

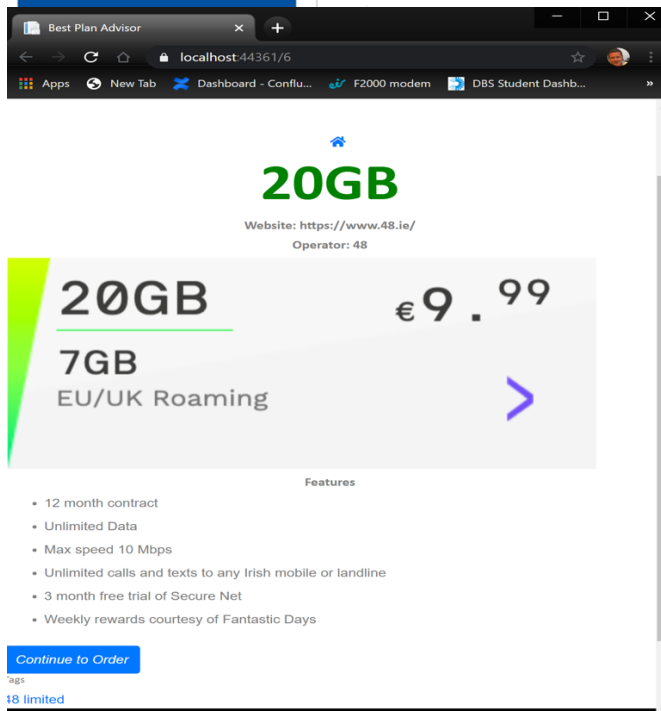
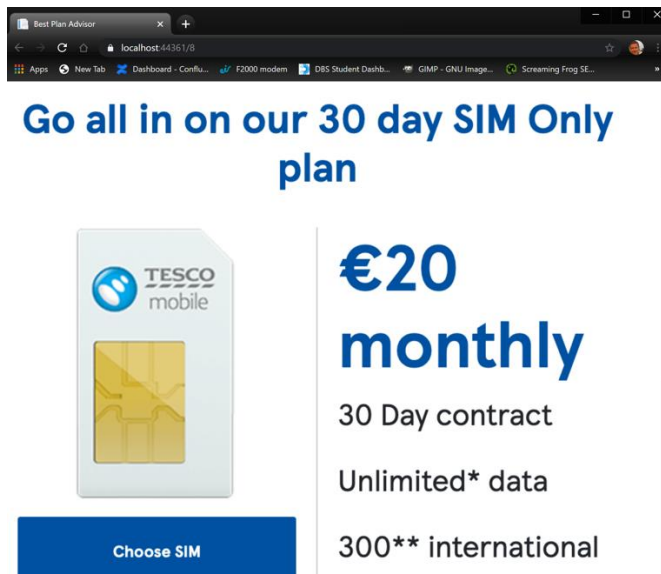
Test Case Title: View plan image

Description: View the image provided by operator relating to a mobile plan

Execution Steps:

1. From BPA home page, select a mobile plan tile
2. Routed to plan detail page
3. Verify image

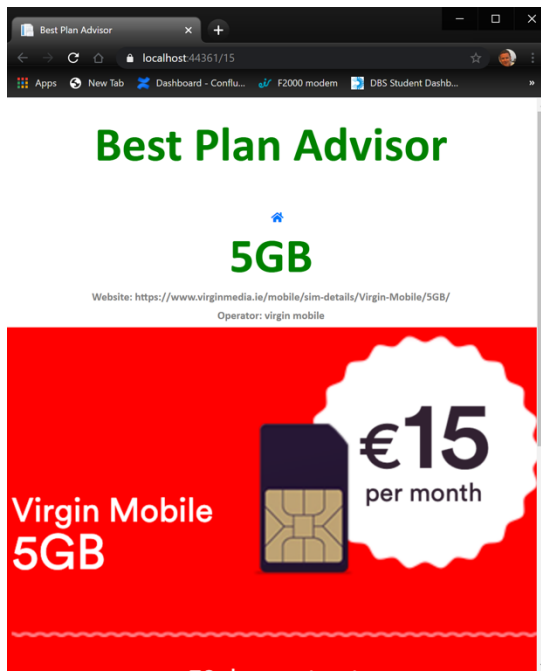
Expected Result: Image included in Plan Detail page



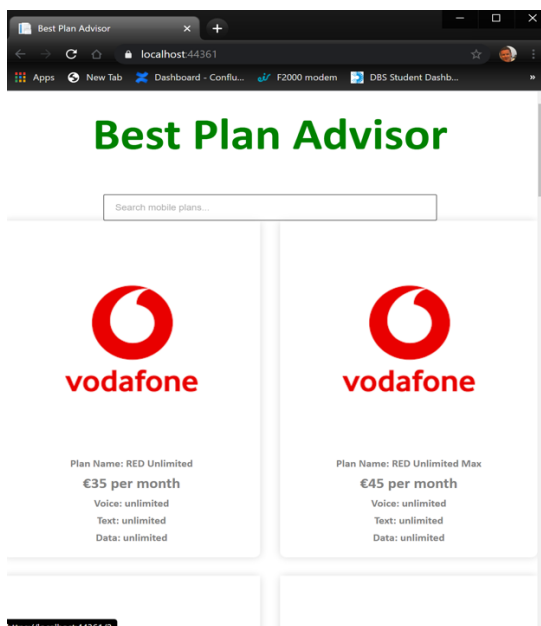
Result: Passed – Examples shown Tesco & 48

TC04**Test Case Title:** Home button**Description:** Test Home button icon**Execution Steps:**

1. From BPA home page, select a mobile plan tile
2. Routed to plan detail page
3. Select Home icon on Plan Detail page

Expected Result: Home icon returns to home page in browser

Home button appears over the plan name 5GB

**Result:** Passed – Home button returns to home page

TC05

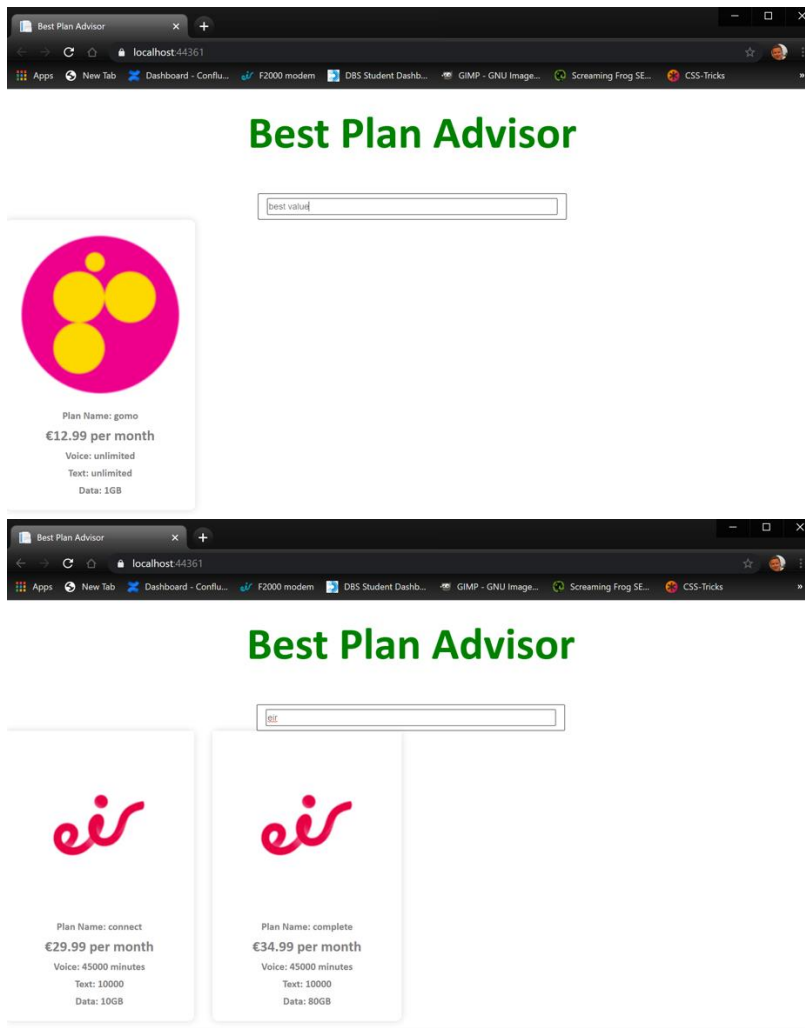
Test Case Title: Search Query

Description: Search for key feature to reduce the full list and narrow the search for my chosen plan

Execution Steps:

1. From BPA home page, enter keyword in search tool
2. Options reduce in associated with plan tag

Expected Result: Reduced number of mobile plans displayed



Result: Passed – 2 examples shown

TC06

Test Case Title: Complete Order

Description: Complete my purchase selecting a link to online store

Execution Steps:

1. From Plan Detail page scroll down
2. Select Box 'Select to Order'
3. Button activation will link out to operator online shop

Expected Result: Redirect to operator product order page

The first screenshot shows a browser window at localhost:44361/5 displaying a mobile plan page. The plan is priced at €34.99 a month thereafter. Key features include:

- No Limits Data
- Unlimited texts
- Unlimited Irish calls
- 200 international minutes and texts
- 50GB of your data to use in the EU
- Free eir sport pack on your mobile

Additional features listed include:

- 12 month contract
- Unlimited Data
- Max speed 10 Mbps
- Unlimited calls and texts to any Irish mobile or landline
- 3 month free trial of Secure Net
- Weekly rewards courtesy of Fantastic Days

A blue button labeled "Select to Order" is visible. Below it, a link for "unlimited eir discount complete sports" is shown.

The second screenshot shows the eir mobile website. It features a navigation menu with options for Broadband, Mobile, TV, Sport, and Need some help?. A purple banner asks if the user is an existing eir residential broadband or landline customer, with "NO" and "YES" buttons. Below this, the "SIM only Plans" section is displayed, offering two options:

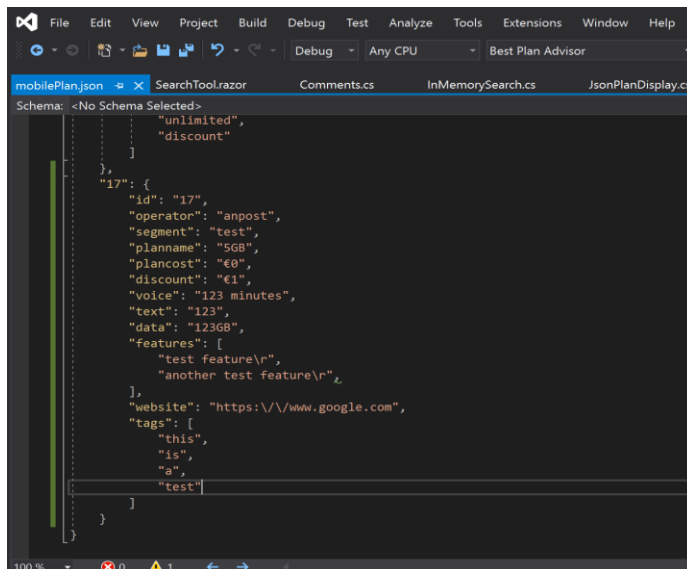
- eir mobile connect - 30 Day: €14.99 a month for 4 months, €24.99 a month thereafter. Features: No Limits Data, Unlimited texts.
- eir mobile complete - 30 Day: €19.99 a month for 4 months, €24.99 a month thereafter. Features: No Limits Data, Unlimited texts.

A "Chat Now" button is visible on the right side of the page.

Result: Passed – Redirected to eir online shop

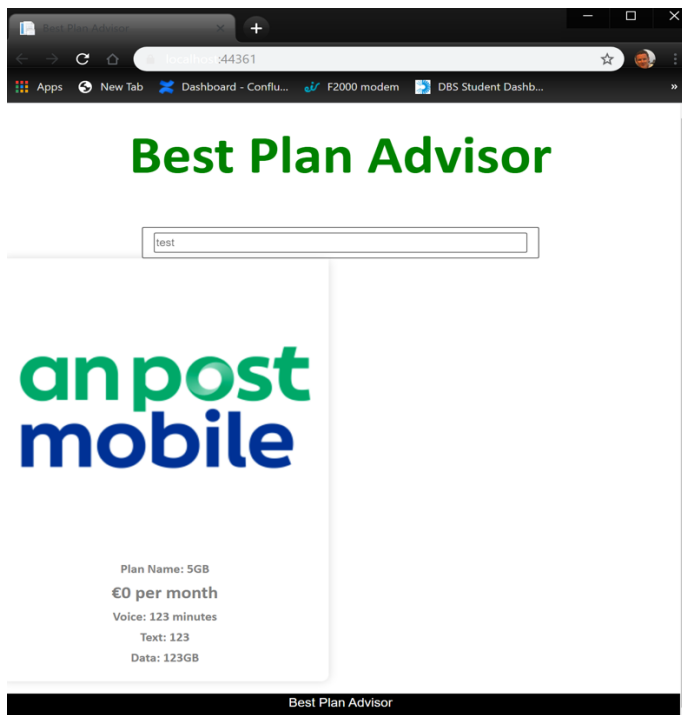
TC07**Test Case Title:** Add new plan**Description:** Add new plan**Execution Steps:**

1. Edit mobilePlan.json (using text editor or edit using VS2019)
2. Create new id and configure new plan attribute values
3. Save and rebuild program

Expected Result: New plan will render to browser

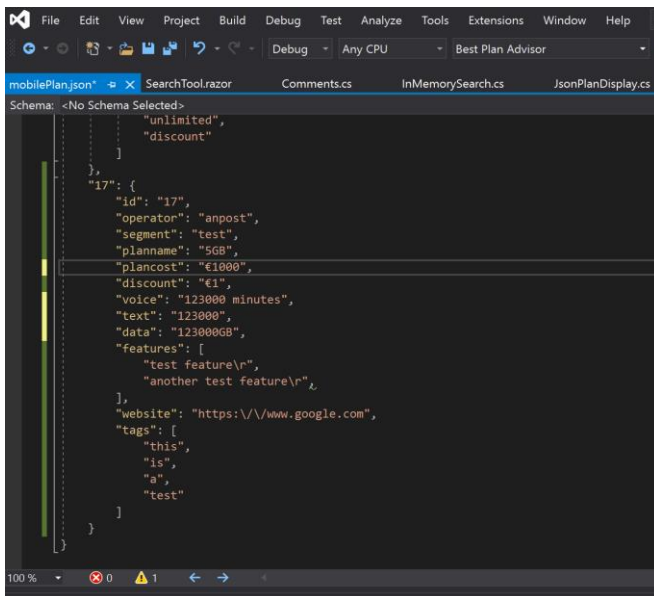
```
Schema: <No Schema Selected>
{
  "unlimited",
  "discount"
},
"17": {
  "id": "17",
  "operator": "anpost",
  "segment": "test",
  "planname": "5GB",
  "plancost": "€0",
  "discount": "€1",
  "voice": "123 minutes",
  "text": "123",
  "data": "123GB",
  "features": [
    "test feature\r\n",
    "another test feature\r\n",
  ],
  "website": "https://www.google.com",
  "tags": [
    "this",
    "is",
    "a",
    "test"
  ]
}
```

New plan created with dummy values

Result: **Passed – Test Plan Created**

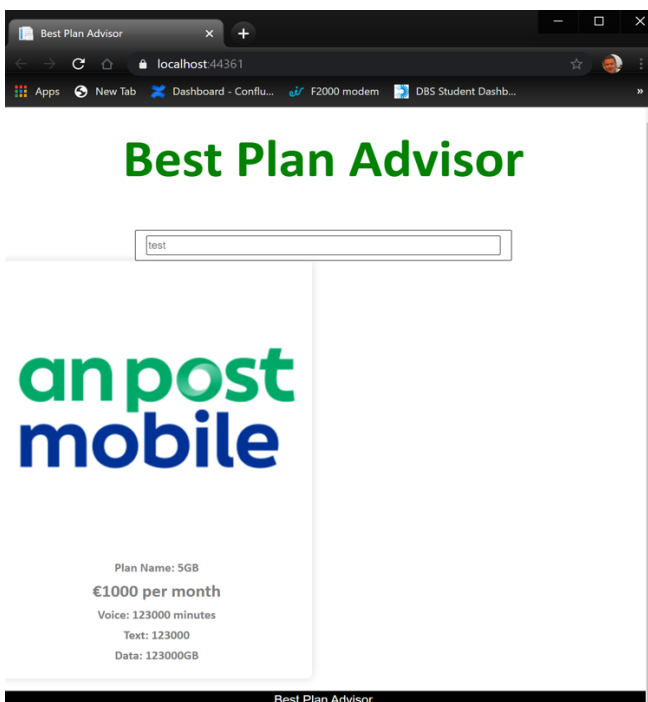
TC08**Test Case Title:** Update/ Modify plan**Description:** Update/ Modify plan**Execution Steps:**

1. Edit mobilePlan.json (using text editor or edit using VS2019)
2. Change values to plan created in TC07
3. Save and rebuild program

Expected Result: New plan will render to browser with updated values

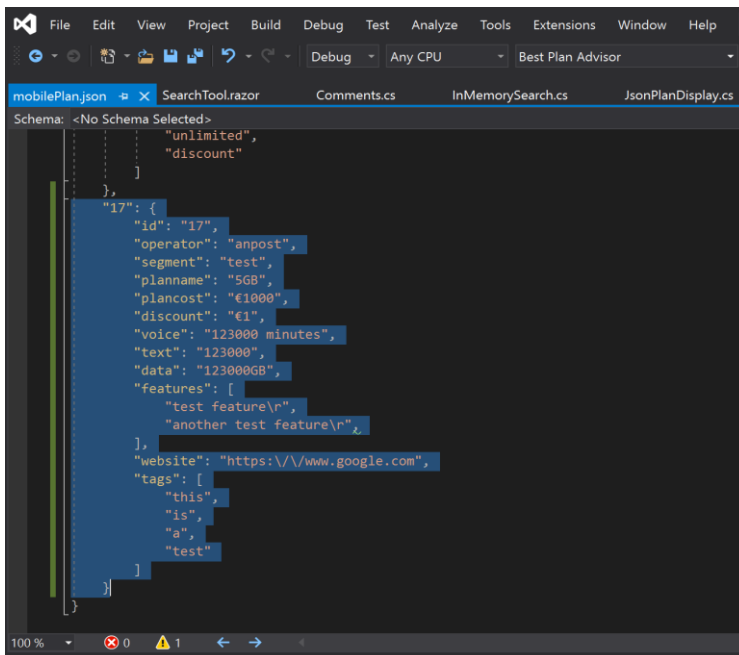
```
Schema: <No Schema Selected>
{
  "id": "17",
  "operator": "anpost",
  "segment": "test",
  "planname": "5GB",
  "plancost": "€1000",
  "discount": "€1",
  "voice": "123000 minutes",
  "text": "123000",
  "data": "123000GB",
  "features": [
    "test feature\r\n",
    "another test feature\r\n"
  ],
  "website": "https://www.google.com",
  "tags": [
    "this",
    "is",
    "a",
    "test"
  ]
}
```

Previous plan updated with modified values

Result: **Passed – Plan values modified**

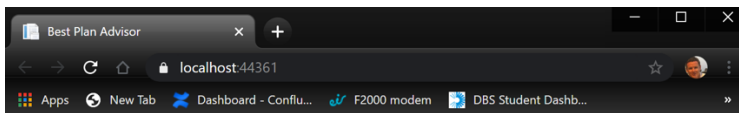
TC09**Test Case Title:** Remove plan**Description:** Remove plan**Execution Steps:**

1. Edit mobilePlan.json (using text editor or edit using VS2019)
2. Remove values to plan created in TC07
3. Save and rebuild program

Expected Result: New plan will not render to browser

```
Schema: <No Schema Selected>
    "unlimited",
    "discount"
  ],
  "17": {
    "id": "17",
    "operator": "anpost",
    "segment": "test",
    "planname": "5GB",
    "plancost": "€1000",
    "discount": "€1",
    "voice": "123000 minutes",
    "text": "123000",
    "data": "123000GB",
    "features": [
      "test feature\r\n",
      "another test feature\r\n",
    ],
    "website": "https://www.google.com",
    "tags": [
      "this",
      "is",
      "a",
      "test"
    ]
  }
}
```

Selected data for removal



Best Plan Advisor

Best Plan Advisor

Result: Passed – Search for test no longer renders a result to browser

6. Conclusion & Future Work

The web application could be expanded further to incorporate other considerations when choosing the most suitable mobile plan, such as consideration for handset costs, bundling of other services such as fixed or mobile broadband, TV and more. The solution has diversified slightly from that originally specified in the project proposal. This search feature has replaced a web form in allowing a prospect to search for key features based on a tag associated with the mobile plan. The justification for this is that most mobile plans are quite similar in that they offer unlimited minutes, texts and data, which were originally planned to be search criteria of the form. Instead, this web application would be suitable for a prospect searching for key features such as 5G, international calls, best value etc. These features may not be clearly evident from their online advertising.

This project performs some of the duties of the retail or telesales agent, and will promote the online sales journey which is standard and in the case of some operators, the sole source of product sales. The application removes the legwork that a customer faces by removing the need to visit multiple online stores to compare different products. From an agents viewpoint, Best Plan Advisor would be a useful sales tool whether used as a web application with a front end UI, or integrated to a CRM platform.

In a recent interview, Carolan Lennon CEO of eir said that eir “closed retail overnight, field sales overnight, everything went online and now that retail is back open we’ve seen footfall drop by 27% but also seen people spending 40% less time in the store, they used to be there for 13 – 14 minutes, now they are there for 8.....we have seen that big shift to our online channels and I expect we will see this continue post COVID as people have changed their habits.” (RTE, 2020). Online stores are progressively becoming the main source of sales for an organisation, and the simplification of the journey through the introduction of a web application like Best Plan Advisor would be a welcome platform to simplify the online customer experience and make the journey more accessible to the public.

Mobile products by nature are complex, and for a consumer, they may be difficult to comprehend. As all mobile plans are quite similar on the surface, a consumer may be tempted to go for the best value plan with the cheapest monthly charge, without understanding the additional charges that will apply once service thresholds are breached. Out of bundle charges are costly and without understanding the implications exceeding an allowance, a 12 month contract may have been signed without understanding a more suitable product which would have cost less in the long run.

Best plan advisor aims to make hidden charges, usually buried in the terms & conditions, made more transparent for each product available. Data can be provided to better serve the consumer and provide a better understanding of their behaviours in relation to amount of data consumed monthly, minutes used etc. As the json data file is enriched with additional information relating to the plan, it becomes a more useful source of information to assist with the sales journey. I would plan in future to create a simple interface to enable easy modifications or additions to the data store.

Consumers will also be better informed with a comments section relation to each mobile plan, allowing customers relay their experiences, good or bad and rating the product for suitability. This would be useful for both prospects, consumers and the product managers who design the plans for each of the mobile networks.

A product manager launching a new plan to the market will find Best Plan Advisor easy to update, and a useful tool to reach their prospective customers.

As the data model evolves, it will be possible to incorporate other products and services that are typically bundles with mobile, for example fixed services such as broadband and TV.

A subscription service could easily be added as a new project class to Best Plan Advisor, allowing clients to register, provide contact details and sign up for marketing and promotional messages that will trigger when launched or when contacts are approaching renewal.

Appendix A

List of References

'Agile Essentials | Agile Alliance' (2019), 26 September. Available at:
<https://www.agilealliance.org/agile-essentials/> (Accessed: 13 August 2020).

Blazor | Build client web apps with C# | .NET (no date) Microsoft. Available at:
<https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor> (Accessed: 14 August 2020).

Blazor Tutorial | Build your first app (no date) Microsoft. Available at:
<https://dotnet.microsoft.com/learn/aspnet/blazor-tutorial/intro> (Accessed: 14 August 2020).

Building Blazor applications on a Mac (no date). Available at:
<https://channel9.msdn.com/Events/Visual-Studio/Visual-Studio-for-Mac-Refresh/VS02> (Accessed: 14 August 2020).

'Eir's third quarter earnings rise by 5%, revenues dip' (2020). Available at:
<https://www.rte.ie/news/business/2020/0520/1139484-eir-quarterly-results/>
(Accessed: 14 August 2020).

MoSCoW Method (no date) Project Smart. Available at:
<https://www.projectsmart.co.uk/moscow-method.php> (Accessed: 14 August 2020).

Welcome to Blazor (no date). Available at:
<https://channel9.msdn.com/Events/dotnetConf/Focus-on-Blazor/Welcome-to-Blazor> (Accessed: 14 August 2020).

What is ASP.NET? [1 of 13] (no date). Available at:
<https://channel9.msdn.com/Series/ASPNET-Core-101/What-is-ASPNET-1-of-13>
(Accessed: 14 August 2020).