

National College of Ireland

Midpoint Submission

Computing (BSHCSD4)
Software Development

Academic Year 2021/2022

Shane Haesy

x16395821

x16395821@student.ncirl.ie

Mobile Direct

Technical Report

Contents

Computing (BSHCSD4)	0
Executive Summary	1
1.0 Introduction	1
1.1. Background	1
1.2. Aims.....	1
1.3. Technology.....	2

1.4.	Structure	2
2.0	System.....	2
2.1.1.	Functional Requirements.....	3
2.1.1.1.	Use Case Diagram	5
2.2.	Graphical User Interface (GUI).....	42
2.3.	Testing.....	45
2.4.	Evaluation	48
3.0	Conclusions	49
4.0	Further Development or Research	50
5.0	References	50

Executive Summary

Max 300 words. Summarise the key points of the report. Restate the purpose of the report, highlight the major points of the report, and describe any results, conclusions, or recommendations from the report[COME BACK TO THIS AFTER YOU FINISH]t.

1.0 Introduction

1.1. Background

The Reasons for me Undertaking this project was first of all I felt like it would be most beneficial for me and my future goals and where I want to be as a software developer this project puts a heavy focus on Frontend Technologies and API Configurations, My Goal as a Software Developer is to be an expert in Frontend Developing and User Interface designing. So, When I read the project's description, honesty for me it was an easy choice to make. I wanted to showcase my learnings and Skill and put on show what I can offer as a Software Developer this project is ultimately what the 4 Years of this Program is leading up to and I want to make sure I leave a good impression.

1.2. Aims

Mobile Direct aims to provide the User with an easy experience all the way through we want to offer the latest in all technologies including User-Security, Front-End Display, Speed and Performance we do this by

- Providing a very clean and user-friendly User Interface with ease of use where the User can easily access all of Mobile Directs functionality
- Providing an easy and straight-forward flow from User Register -> User Checkout where the user is directed and navigated
- Since Mobile Direct is a Store, we want to User to be able to find their Chosen product by offering lots of Filtering and Search options to help them make their choice
- Ensuring the Application runs smoothly and doesn't contain any bugs or bottlenecks or cause any security risks to the User

1.3. Technology

Angular – This is a Framework created by the team at Google that runs via the Node.JS environment first released in 2016 its primary use is for developing full-fledged web applications, Angular is described as a Single-Page application it's known for being ultra-fast and responsive and for ultimate scalability and is constantly getting updated and new features added to it all the most it's component based with each component being very usable and easy Integratable with other technologies , the coding language Angular uses is Typescript which is a flavour of JavaScript developed by Microsoft it's the exact same as JavaScript but offers strict syntactical code which results in code that's easier to read and better formatted the Angular compiler will convert Typescript code into vanilla JavaScript when its being compiled for the browser to interpret.

Angular Material – This is a User Interface Component Library that is built especially for Angular it offers a wide range of U.I Components that are simplistic and aesthetic in design and very customizable and programmable to integrate into your system

ASP.Net Core - This is an open-source web framework by Microsoft that covers all facets of web technologies I will be using this to build my backend/API, .NET Applications are written in C#, which is a language almost identical to JAVA, .Net also comes with loads of documentation on their official site and a huge community of developers so it makes your life as a developer a lot easier

Auth0 – This is an Authentication and Data-Storage platform that is created to integrate with websites for User Creation and Authentication it offers high level Authentication and Integration

MySQL – This is Oracle's take on the Relational Database it's open source and offers a very powerful and robust platform for Data Storing and Manipulating that is widely used across the web and comes with lots of integration and cross platform features

Sendinblue – This is an online tool for sending automated emails since my project is an online store, I'll need to send E-mails for when the User completes an Order it's also comes with a great dashboard that makes configuring it very intuitive

Local Storage – These technologies were used to save User data I wanted it so when a User would enter products inside their cart that the cart wouldn't reset every time the User refreshed or clicked on a different website

1.4. Structure

This report will be templated as follows

- System
- Conclusions
- Further Research and Development
- References
- Appendices

2.0 System

2.1 Requirements Specification

Before going into great technical detail on functional & non-functional requirements for Mobile Direct the Overall hierarchy of the System are as follows

1. The System must be flawless in its navigation meaning that the end user will not keep getting lost in the process and committing errors for this the System will need to boast and clean and open User Interface . There should not exists uncertainty with the End-user in the Systems capabilities and what the System is designed for it should take no more than a few minutes for a User to become fluent in Navigating **Mobile Direct**
2. The System must be content rich as always be displaying product related content in all its views with an option for a User to easily purchase these products
3. The System must be dynamic in what content it displays to the End-User meaning a User with Administrative rights must be viewing different content then a regular User with standard privileges
4. The System display data based on querying and searching by the User to help the User to find the product they desire they can add these products to the basket and a secure payment terminal will complete their order
5. The System must track the Users usage and shopping habits and be able to recommend the user a product based on thee shopping habits
6. The system will send the user confirmation email and is able to view their orders in the profile section of the System

2.1.1.Functional Requirements

Functional Requirements are the Actions a System is able to perform consistently in order to be defined as Functional, For Mobile Direct as explained before there are two type of Users these are Regular Users which I'll be referring to them as just Users and Users that have more privileges and are able to perform more actions that regular users these are known as Admins

So here is a List of all these actions that the System is capable of performing

User Requirements:

1. User should be able to Create Account
2. User should receive email to confirm their account
3. User should be able to login to their account
4. User should be able to logout of their account
5. User should be able to search for a product using the search bar
6. User should be able to filter for a product using the filter options
7. User should be able to add products to their basket
8. User should be able to view the product in a separate view
9. User should be to clear products from basket
10. User should be able to create order
11. User should receive email confirming their order
12. User should be able to enter a promo code in the basket

Admin Requirements

1. Admin should be able to navigate to the Panel on the system
2. Admin should be able to view All Orders by all users
3. Admins should be able to create products
4. Admins should be able to delete products
5. Admins should be able to update the price of product

Requirements Explained

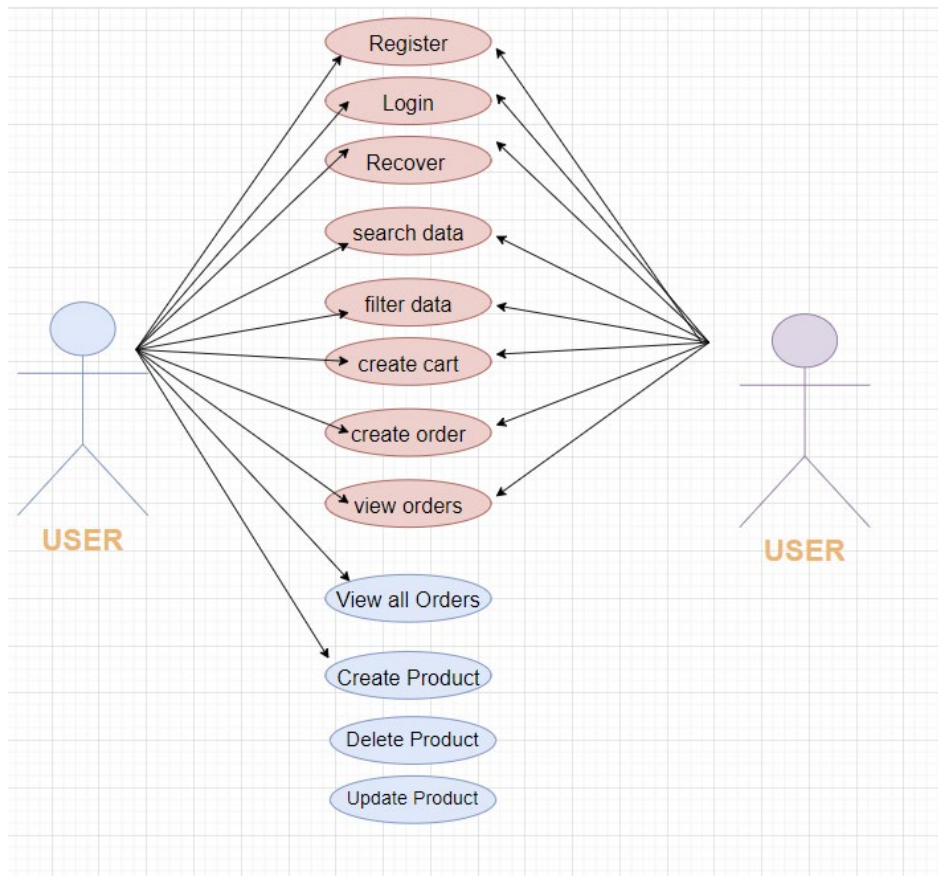
User Requirements

1. Refers to when a User lands on the Webpage for the first time they will be greeted with a Sign-up button that will allow to them enter their relevant details to create an account with Mobile Direct
2. Once the User enters their details, they will be authenticated and stored in an Auth0 database and sent an email to confirm these details once they confirm their account this will be displayed on their User Profile
3. Users should have the option to Log In to their account at any time once the details are correct
4. Users should be allowed to log out of there account at any time with a button displayed on the home section of the System
5. User should be able to recover in the case of them forgetting their password
6. The products are all stored in a database and fetched via an API call these products should be made visible to the user when they navigate to the Store section of the website
7. Refers to the process of a User being able to Search for a product they desire by entering text into a text field and hitting enter
8. Refers to the process of a User being able to filter their options via Brand in a dropdown menu the results displayed in the Store will be instantly updated
9. Refers to a User should be able to click a button and the cart containing the products is emptied
10. Refers to the process of a User clicking on the Add button on a Product so its stored in their basket in the Cart section of the website the User will then be able to view these products they entered into their Baset
11. This refers to User should be able to create an Order with the Items from there Cart and their details are stored inside this order
12. User should receive an Email confirming their Email once they fully complete the checkout process which includes entering their Credit Card details and there Address the emails are dynamic and will contain Details specific to that User

Admin

1. Refers to when an Admin account is logged in the System will recognise this and display an extra navigation button in the Navigation Bar titled Amin from here, they will be able to access their special privileges are Admin
2. Refers to regular Users are only able to view orders associated with them while Admins are able to view All orders by All Users

3. Refers to Admins having the ability to being able to create new Products from the Admin Panel, A form will be displayed that will POST into the Products Database
4. Refers to Admins having the ability to delete a product from the Catalogue this is done in the Panel for Admins, the Admin will be asked to input the ID of the Product they want deleted and my Backend will take care of the rest
5. Refers to Admins being able to Update the Price of a Product, they will be greeted with a Dropdown menu of all the Products and an input field beside it once they choose the product and enter the price beside it, my API will communicate with the Back end to update the Product in the SQL Database



2.1.1.1. Use Case Diagram

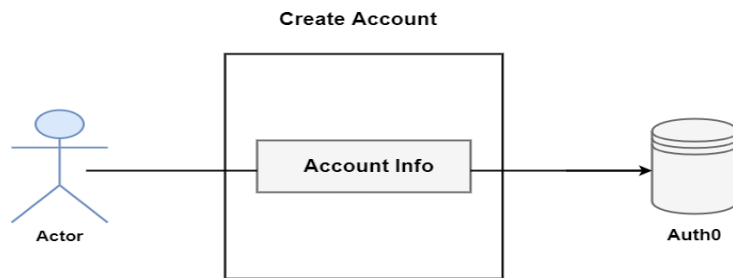
Use Case 1 – Create Account (UC_1)

DESCRIPTION

Refers to when a User lands on the Webpage for the first time they will be greeted with a Sign-up button that will allow to them enter their relevant details to create an account with Mobile Direct

ACTORS - User, Admin

USE CASE DIAGRAM



Activation

Commences when User click Log in button on the home page then further click the sign up

Preconditions

- System is Turned On with a valid internet connection
- The Auth0 API Is integrated with the System
- The User is not yet registered with the Auth0 database

Post Conditions

- User will receive an indication on the display that they have completed the process
- User will see their details on the screen
- User will receive an Email confirming the process has been completed

Main Flow

- User opens up website
- User clicks Sign Up
- User enters all relevant information in the form
- Information is then sent to Auth0 service to be authenticated
- Data is then sent back to the User indicating that the Data is valid
- User is sent to back to home screen system displays users details

Alternative Flow

Passwords don't match

- User opens up website
- User clicks Sign Up
- User enters all relevant information in the form
- An Error message is displayed detailing that passwords don't match
- User remains on Create Account form

Email has been used before

- User opens up website
- User clicks Sign Up
- User enters all relevant information in the form

- An Error message is displayed detailing Email has been used before
- User remains on Create Account form

Exceptional Flow

Termination

Once the Authentication returns with a success the process has been fully completed and therefore terminate

Post-Condition

The User is navigated to the Home Screen

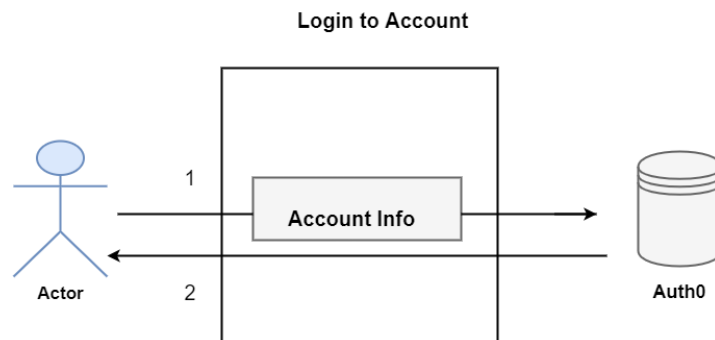
Use Case 2 – Login to Account (UC_2)

DESCRIPTION

Refers to Users being able to gain access to an already created account once they obtain the correct login information this is applicable to all type of users

ACTORS - User, Admin

USE CASE DIAGRAM



Activation

Commences when the User clicks the Log in button

Preconditions

- User has internet connection
- The Auth0 API is correctly configured
- The User is already stored inside the Auth0 Database
- The User is on the correct webpage
- The User is not already logged in

Post Conditions

- User will see a visual representation indicating that they logged in
- The Navigation Bar will change

- The User's Name will appear on the welcome screen

Main Flow

- User opens up website
- User clicks Login
- User enters all relevant information in the form
- Information is then sent to Auth0 service to be authenticated
- Data is then sent back to the User indicating that the Data is valid
- User is sent to back to home screen system displays users details

Alternative Flow

Incorrect Credentials

The Login Page will give a warning message

If the User enters the wrong details a number of times the account will become locked

User must submit email request to gain access to account

Exceptional Flow

The User clicks off the website while the login process is happening therefore cancelling the flow

Termination

When the User has successfully logged in with no problems the use case becomes terminated

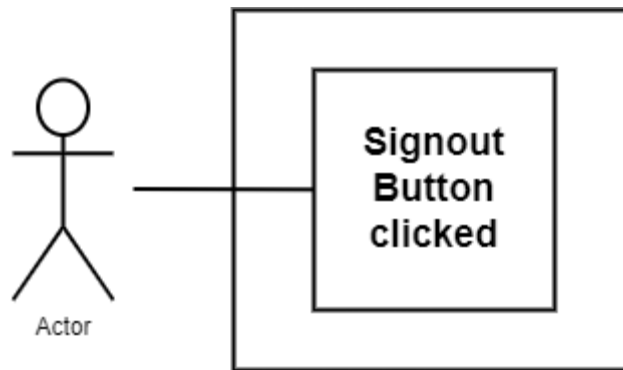
Use Case 3 – Logout of Account (UC_3)

DESCRIPTION

This is referring to the process of a User of being able to safely sign out of their account and their current activities and sessions has ended if User wants to continue, they will need to complete Use Case 2 again

ACTORS - User, Admin

USE CASE DIAGRAM



Activation

User clicks the Logout Button

Main Flow

- The User clicks logout which is visible from the home Page component
- The System will then terminate the session
- The System will refresh the page and the User interface will be different

Alternative Flow

- If The Auth0 system detects and suspicious activity relating the account, they will be forced logged out

Exceptional Flow

Termination

Use case terminates when user has successfully logged out and returned to the home screen

Preconditions

User is logged in already

Post Conditions

- User logged out and can't make any changes to their account
- The Navigation Bar and User Interface will change

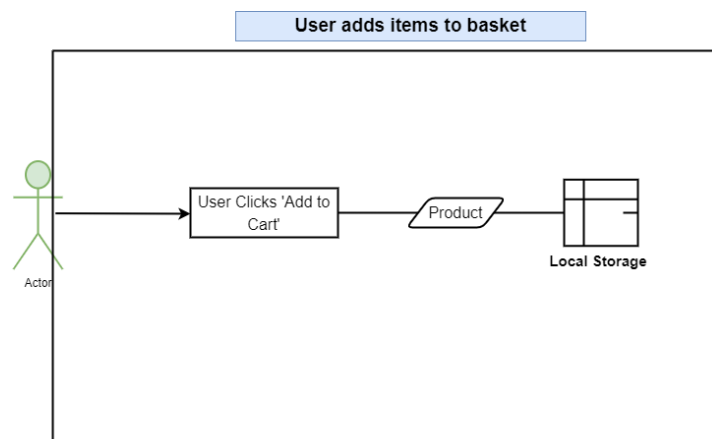
Use Case 4 – Add to Item to Basket (UC_4)

DESCRIPTION

This refers to when a User is browsing the products section of the website and adds a product to their Shopping Cart

ACTORS - User, Admin

USE CASE DIAGRAM



Activation

When a User clicks the add button displayed on a Product Card

Preconditions

- The User is using the system with a valid internet connection
- The User has enabled cookies on the website

Post Conditions

The User will see in the navigation Bar a visual representation of how many items they have in their shopping Basket and this number will change when they click Add on a product

Main Flow

- User Navigates to Products section of the website
- User clicks Add on an Item
- A Display will popup indication that they added the item to their basket along with a recommendation for products similar to the one they just added
- The number representation of their basket quantity will increment
- The User can view the Item they added in the Cart section of the website

Alternative Flow

Exceptional Flow

Termination

Process is terminated when the Item has been successfully added to the basket

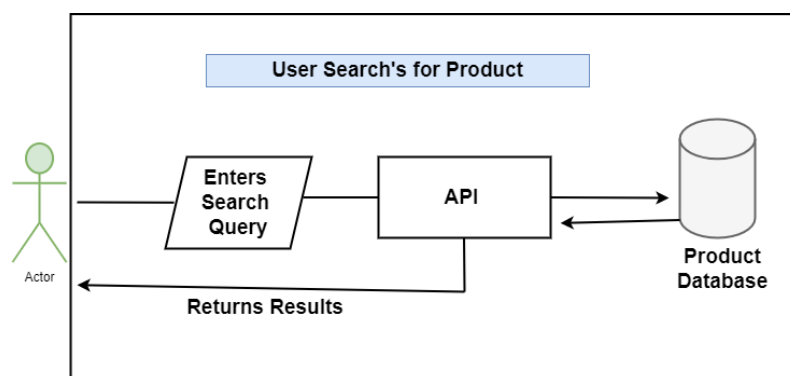
Use Case 5 – User Search's for Product(UC_5)

DESCRIPTION

When a User enters Text in the Search bar to search for a specific product rather than just scrolling

ACTORS - User, Admin

USE CASE DIAGRAM



Activation

When a User Enters their Search string in the search box and click's the search button

Preconditions

- The User is using the system with a valid internet connection
- The User is on the store page

Post Conditions

The System will return products to the display relating to their search query

Main Flow

- User navigates to the Products section of the website
- User clicks the Filter icon top right of the webpage
- The User clicks on the search text field
- User enters their search string
- User clicks search

Alternative Flow

Exceptional Flow

Termination

When user clicks enter, and the system returns the results from the API

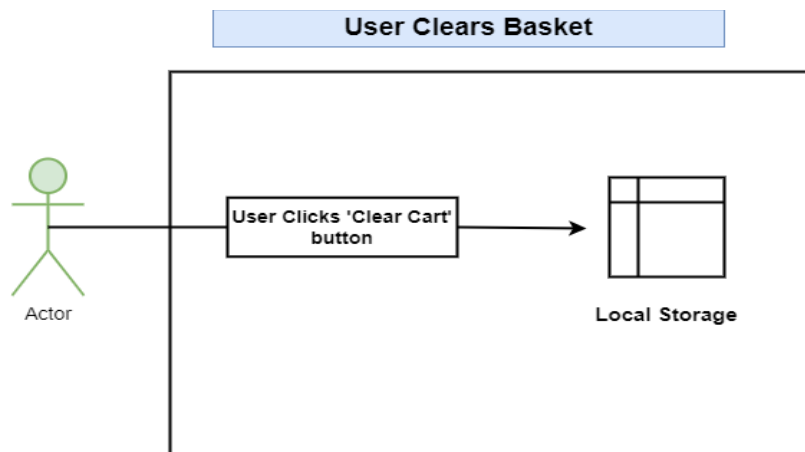
Use Case 6 – User Clears Basket (UC_6)

DESCRIPTION

This refers to when a User has items inside their basket and wants to empty the basket so it's empty again

ACTORS - User, Admin

USE CASE DIAGRAM



Activation

When user wants to empty the basket

Preconditions

- The user is using the website with a valid internet connection
- The user has cookies enabled
- The user has items inside their basket

Post Conditions

- The Navigation bar indicator will return to zero
- The Basket will no longer contain any items
- Visual representation saying the basket is empty

Main Flow

- Users adds items to basket
- User navigates to cart section of the website
- User has items inside there basket
- User clicks empty basket inside
- User’s basket will be empty
- User will be greeted with a ‘Your basket is empty’ screen

Alternative Flow

Exceptional Flow

Termination

When the User has clicked the button and the basket is empty

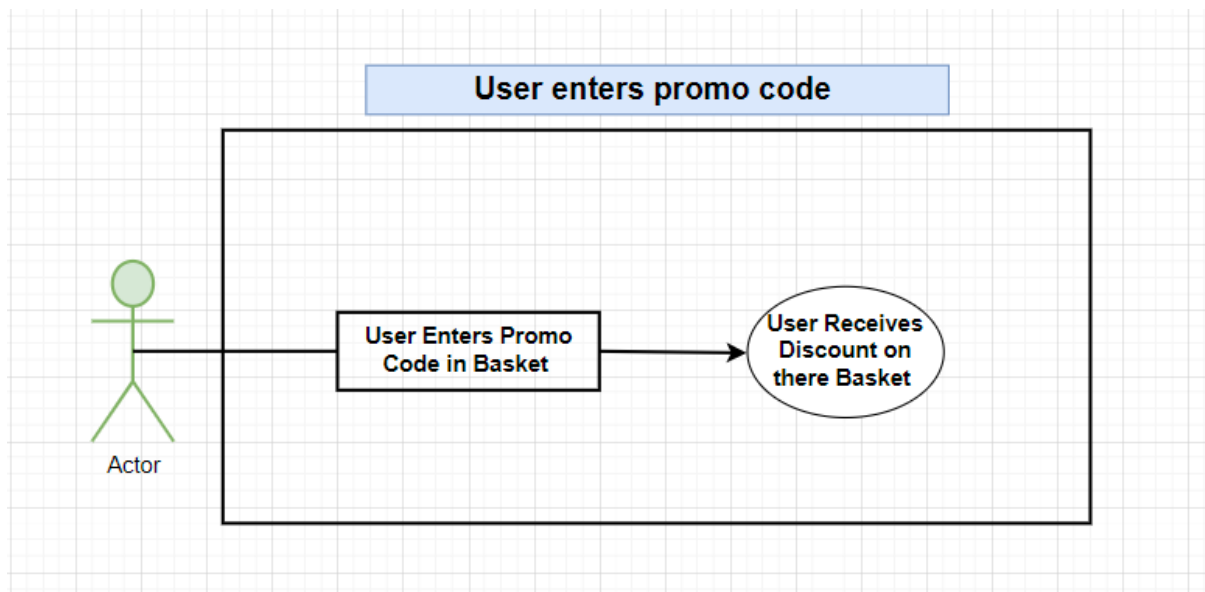
Use Case 8 – User Enters Promo Code

DESCRIPTION

This refers to a User heads over to the Basket Component and enters in a Promo code in the Promo section and in return they receive a discount on their total amount

ACTORS - User/Admin

USE CASE DIAGRAM



Activation

When a User clicks Enter Promo Code button in the Promo Code section of the Basket Component

Preconditions

The User must have at least one Product inside their Basket

The API must be turned on

The Promo Code must be valid

Post Conditions

The Total Amount for that basket will update

Main Flow

- User Adds an Item(s) to their Basket
- User clicks on Basket component
- User scrolls down to the Enter Promo Code section
- User enters a Valid Promo code and clicks the button

Alternative Flow

- Flow the Same as Main Flow

Exceptional Flow

The same as Main Flow

Termination

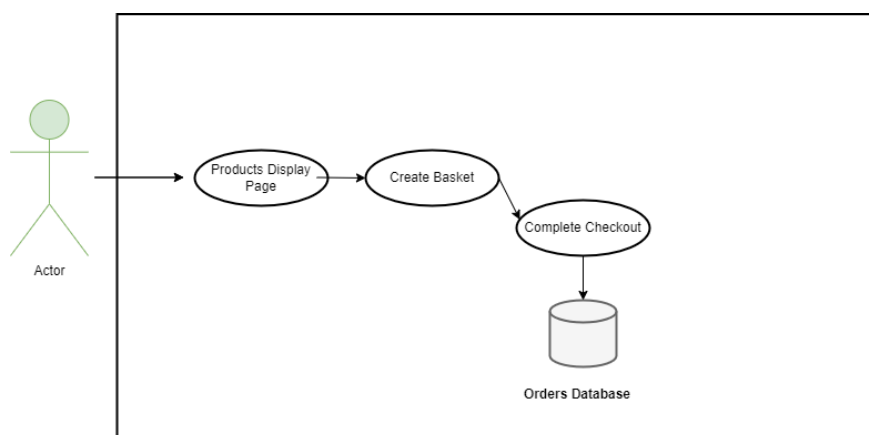
The use case is terminated once the User has entered a Promo Code that is valid and it the Basket

Use Case 7 – User Create Order (UC_7)

DESCRIPTION

ACTORS - User, Admin

USE CASE DIAGRAM



Activation

When a user completes an order, and the payment process has been completed

Preconditions

- The User has items inside the basket
- The User is logged in
- The User has completed the checkout process
- The User has cookies enabled
- The User has completed the Payment process

Post Conditions

The user will receive a confirmation email completing the order

The Users basket will be empty again

The user will be returned to the cart page

Main Flow

- User adds items to basket
- User sign's in
- User adds items to basket
- User navigated to cart page
- User clicks on purchase items
- User enters payment details
- User then receive email of confirmation

Alternative Flow

Payment Details incorrect

- The system identifies that the credit card information is incorrect
- They will then be returned to the cart page
- They can try again

Exceptional Flow

Termination

Once the order has been added to the Database

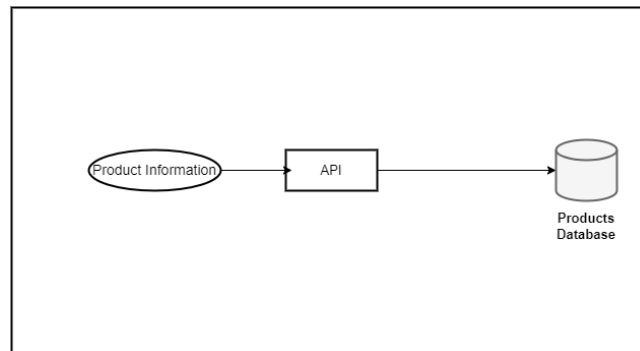
Use Case 8 – Admin Create Product (UC_8)

DESCRIPTION

This refers to an admin being able to products to the Products Database from the Admin panel the product will then be added to the store section of the website

ACTORS - Admin

USE CASE DIAGRAM



Activation

When an Admin navigates to the admin section of the website and enters in the details of a product and click add product at the bottom of the form

Preconditions

The User must be an Admin

The API must be turned on

Post Conditions

The Product will be added to the Database

The Product will be instantly available to buy in the products section of the website

Main Flow

- User logs in as Admin
- Admin then navigates to the Admin Panel
- Admin scrolls down to the add product form
- Admin enters correct Product information
- Admin Clicks Add Product
- Product is added to Database and is available on the website

Alternative Flow

Product details incorrect - The Product ID must be Unique

- Admin enters Details in Product form
- Admin enters Product ID that already exists
- Error will be displayed in the Console and on the Display
- Admin can try again

Exceptional Flow

Termination

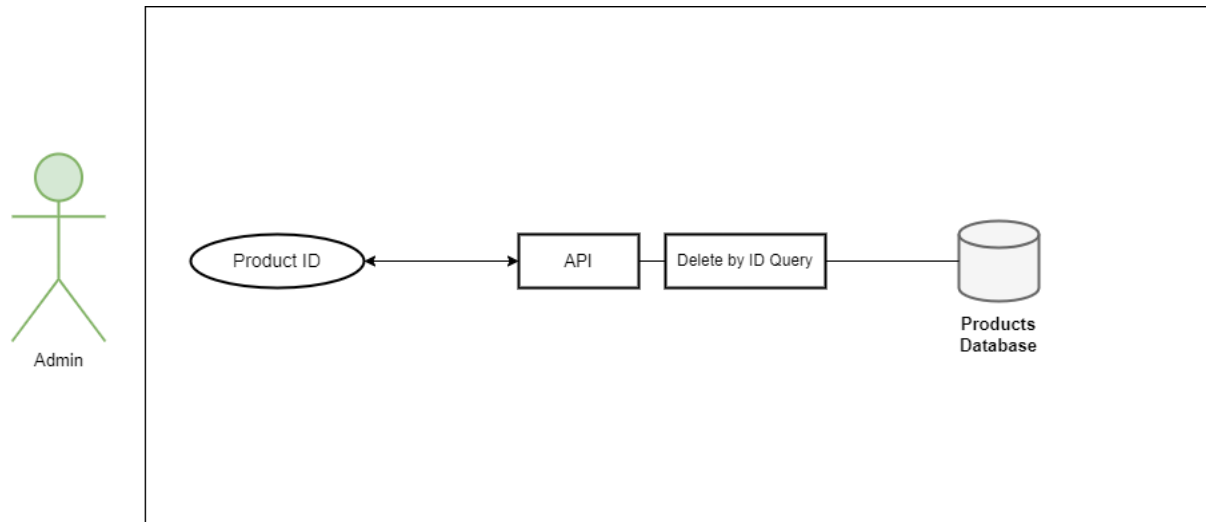
Use Case 9 – Admin delete Product (UC_1)

DESCRIPTION

An Admin can delete a Product if they want, they must enter the Product ID they wish to delete inside the admin panel

ACTORS - Admin

USE CASE DIAGRAM



Activation

When user enters Product ID and clicks Submit

Preconditions

- User is logged in as Admin
- API is turned on

Post Conditions

Product is removed from Database

Main Flow

- User logs in as admin
- User navigates to Admin section of the website
- User enters the ID For the Product they want to delete
- User clicks Submit

Alternative Flow

ID does not Exists

- User enters Product ID that does not exists
- Error message is displayed

Exceptional Flow

Termination

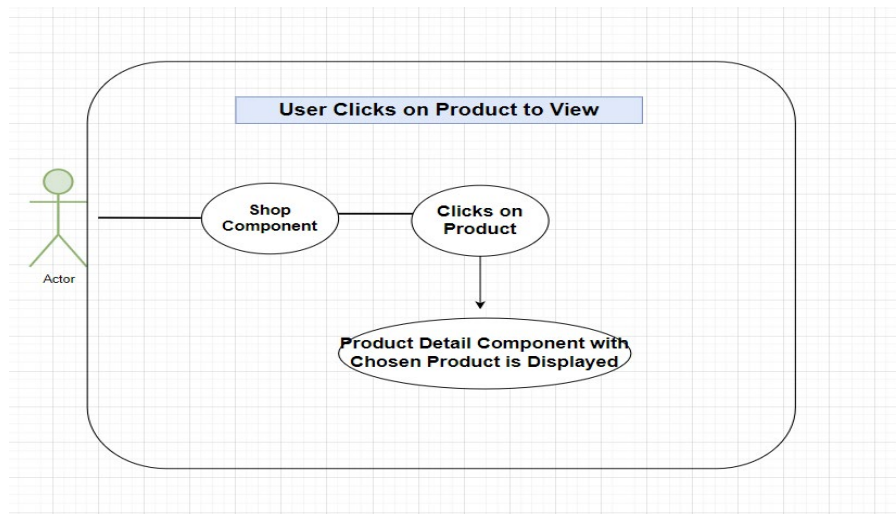
Use Case 8 – Admin Create Product (UC_8)

DESCRIPTION

This refers to a User when browsing the products in the Shop Component can view a Product in greater detail by clicking on it

ACTORS - Admin

USE CASE DIAGRAM



Activation

When a User Navigates to the Shop Component and Clicks the Image of a Product

Preconditions

The User must be an Admin

The API must be turned on with the Products being Displayed in the Component

Post Conditions

The Application to route to another URL with the Product ID as the URL

The Product will then be displayed for the User

Main Flow

- User logs in
- User Clicks on 'Shop' in the Navigation Bar
- User Browses the Products and Clicks on the Image of a Product
-

Alternative Flow

User Uses the URL to Navigate the product display page

- User Enters the URL of the Products ID along with the String 'Products/{id}'
- Error will be displayed in the Console and on the Display

Exceptional Flow

Exceptional flow follows the same suit as the main flow

Termination

The use case is terminated once the User is happy with the Content, they are looking for they may add the Product to their Basket or simply click back on the browser to go back to the Shop Component and view other products

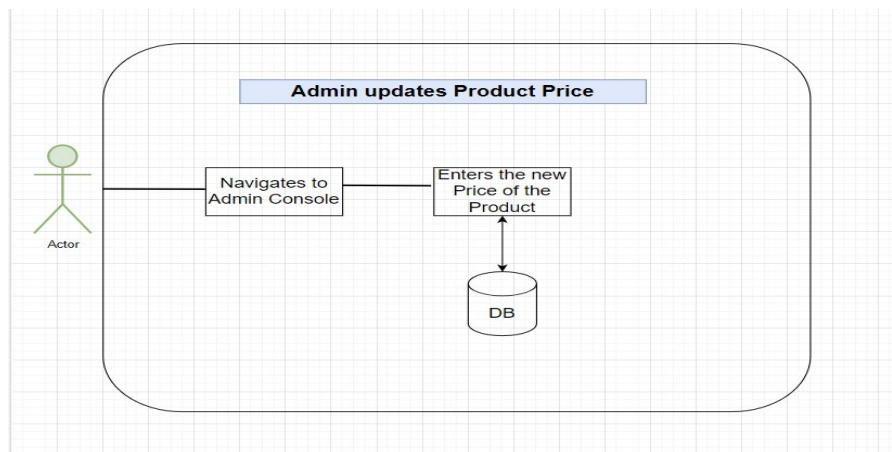
Use Case 8 – Admin updates product price (UC_8)

DESCRIPTION

This refers to a when an Admin uses their Administration privileges to Update the Price of a Product within the Catalogue the Admin will go to the admin console, and this will in effect POST a request to my API to change the product price using the ID as an identifier

ACTORS - Admin

USE CASE DIAGRAM



Activation

When an Admin is signed in and the Admin Console appears in the Navigation Bar, and they click on it and scroll down to the Update Product Price section

Preconditions

The User must be an Admin

The API must be turned on

Post Conditions

The Product Price will update to what the admin inputted

The Product will then be displayed with the new price for all users to view

Main Flow

- Admin
- Admin Clicks on 'Admin' in the Navigation Bar
- Admin scrolls down to the Update Price section
- They Choose from the dropdown menu the Product they want to alter
- They Input the corrected price in the input field and click the 'Update Button'

Alternative Flow

- Flow the Same as Main Flow

Exceptional Flow

The same as Main Flow

Termination

The use case is terminated once the admin is happy with the changes they have made

Non-Functional Requirements

Security

This was one of the most important requirements for me when I was designing the website since Mobile Direct is an online shop security must be a most upheld feature with lots of attention paid to this facet, since there is going to multiple users using the system at once Multitenancy is something I was going to have to incorporate, this basically means not being to view other's Users private information that why when a User uses logs in with Auth0 they are issues a Token that is unique to them an Authenticates the user and secures there information with encryption

Maintainability

In modern Software development we release updates rather than new full versions of a Software product so this is something I needed to consider a lot when creating this project, with Angular being component based it's very easy to develop new features and integrate them, everything is very modular and re-usable and removable so if a big change needs to be made it will be very straight forward. This is one of the reasons as to why I chose Angular.

Reliability

If I decide to Host Mobile Direct, I will need to choose a service which offers an impressive uptime and offers scalability and good deployment options I will need to Host the Front End and the Backend and SQL Database that's 3 different applications so I'll need to choose one that can tend to all my needs

Collaboration

Mobile Direct's stack and development has been designed in such a way that it will be easy for other software engineers to implement changes to the Project its very modular in its design as its purely component based and API is the Same, I also have comment all over the Codebase to help other viewing the Code to understand what's going on better

Data Requirements

This section will give a run through of the data structure that mobile direct runs on it will describe in detail how Mobile Direct stores its data and how the data is linked to each other

Mobile Direct uses two types of Data storage, for products and orders it uses a traditional relational database to store this information it uses MySQL both and a JSON based database MySQL is a platform developed by Oracle which is table based, The Database can be split into two declarations one being Product data this will mean all the data associated with the Products for instance the Table that holds the Products Information (as seen as below) and the Order table that will hold the information associated with Orders that will contain product data inside the table. The second declaration will be User data since we want to be able to track the User's behaviour, we need to store this information somewhere, so for a short explanation the system works as follows when a User Creates an Account and Clicks on a Product to View the System will create a Table with their Username as the Table Name and Log their Click to that Table.

The other type of Data Storage used is a collection-based storage which is based on JSON Objects rather than Traditional Tables to collect information this is done via Auth0 when a user signs up to Mobile Direct their email and password will be stored in a Auth0 JSON Collection,

<i>User</i>
user_id
email
password
admin (true or false)

<i>Product</i>
prouduct_id
name
price
brand
desc
type
img

<i>Order</i>
order_id
email
date
products
total

{{ Username }}
brand
clicks

User Requirements

These are the Requirements necessary based for each user for Mobile Direct to be fully functional

Admin Requirements

- Admin must be able to view all orders
- Admins must be able to create a new product
- Admin must be able to delete a product
- Admin must be able to access the admin panel

All Users Requirements

- Users must be able to create an account given that they have to provide the correct details and information
- Users must be able to access their profile from the navigation bar
- Users must be able to view all products
- Users must be able to add items to their basket
- Users must be able to purchase a Product
- Users must be able to receive an email confirming their order

Environmental Requirements

Angular can be quite an intensive application to run on a system it uses the Ivy Compiler that is always recompiling the data and that is being presented to the User so boasting a strong machine would be an Environment Requirement

Usability Requirements

Learning

When creating a system that is meant to be used by all types of users ease of use and should be a goal for any developer, Mobile Direct is designed with a low learning curve to a User can fluently navigate the system in just a short space of time even for Users that aren't good at using computers

The Systems features need to be easily accessed and self-explained with the User having to getting frustrated

Presentation

- The User Interface should be designed in such a way that it adheres to good U.I Designed practises and principles, Clarity and Simplicity should be the main theme of Presentation that the system presents to the User, this can be achieved with a clever use of Colour scheming, Font Sizing, and spacing
- The Presentation should be fun and welcoming, and the User should feel excited when using the system, the colour scheme should be consistent throughout the whole the experience
- The system should make use of components such as text box's buttons to navigate through the system it should be easy and clear to what each buttons does everything needs to be obvious and leave the user without any feelings of ambiguity
- Everything needs to be conveniently located for the User to access with the User having to search through the system to access something

Maintainability

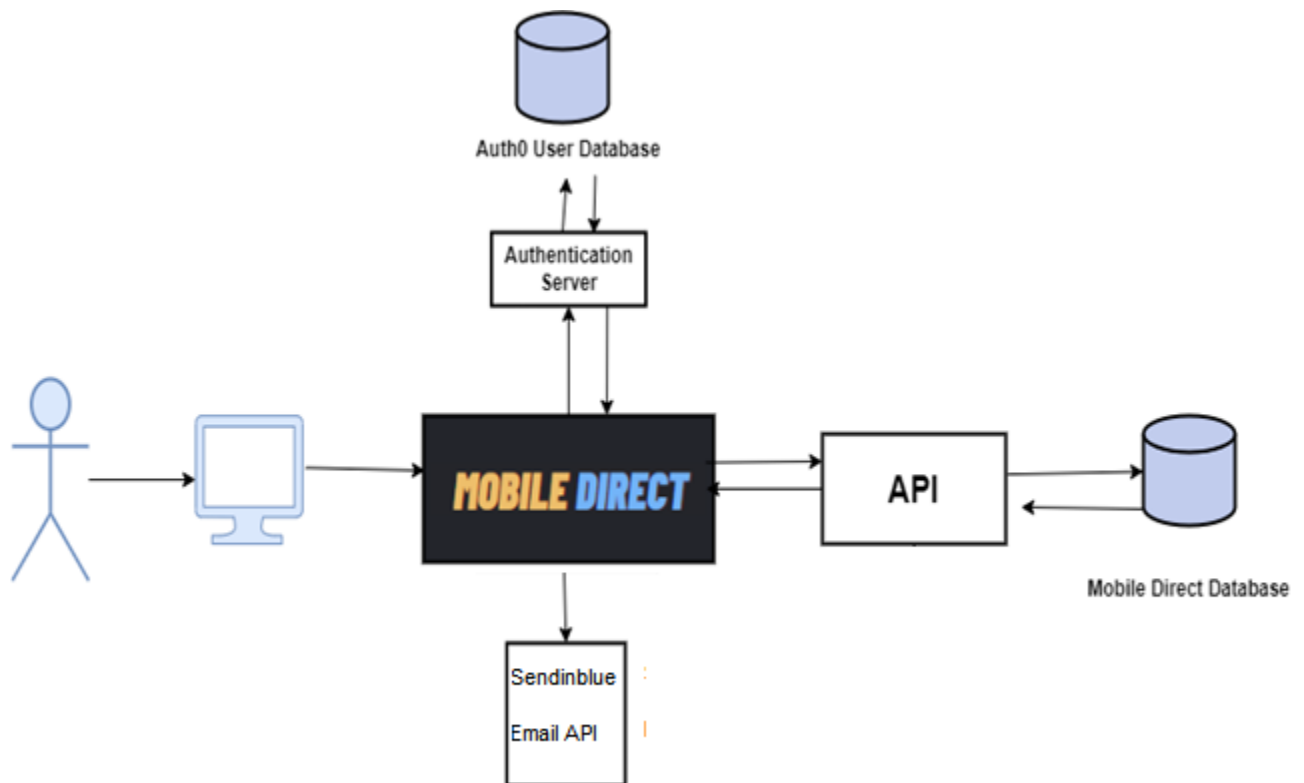
Users should feel awarded and fulfilled to maintain their usage of the system

Design & Architecture

The Architecture of the System works as following when a User signups to the system there User details will be stored in a Auth0 cloud database that uses high end consumer grade encryption once they sign up they will be able to purchase products which creates orders, once they are successfully logged in they are the API that I created Handles the rest of the operations when they create an Order the API will execute a command in the SQL Database whit there Auth0 credentials used as a identifier the API will then communicate with Sendinblue to send the User an email confirming there order

This is a Simple Overview Diagram of the Architecture

S.O.L.I.D Software Design is on show here as each component in the Architecture only has single responsibility and its modular



More in depth Overview of the Architecture

Implementation

I'm not going to discuss every single line of code as there are 1000's of lines of code and talk about every single aspect of implementing the System, but I will give a detailed explanation the main parts of the system

Auth0

To implement Auth0 into my system I needed to create an account with Auth0 and use their Dashboard on their website to configure it correctly with my project this was including such thing as 'Rules' and 'User Types' for instance Admin rights and so on once this was done, I had to configure Auth0 with my System and import the Auth0 libraries into my Angular Application this was done as following lucky Auth0 make it simple to integrate with Node.JS Platforms

I had to install the Auth0 Node Package via the Node terminal into my projects directory once this is done, I can reference the library as following in my project

```
import { AuthService } from '@auth0/auth0-angular';
```

Once the service is imported into my project, I can be using the Service and the API's that Auth0 has to offer let's say for example I want to retrieve all the User Information related with my Account this can be done as follows :


```
//Check for what Role the User is
this.auth.user$.subscribe(s => {
  if(s['role'] == 'admin'){
    this.admin = true;
  }
})
```

Here I'm checking the Role of my Account and seeing if it is an Admin or Not as I said before this account are stored in the Auth0 JSON Database to view and monitor the account this is done via there User Management Portal it looks like this

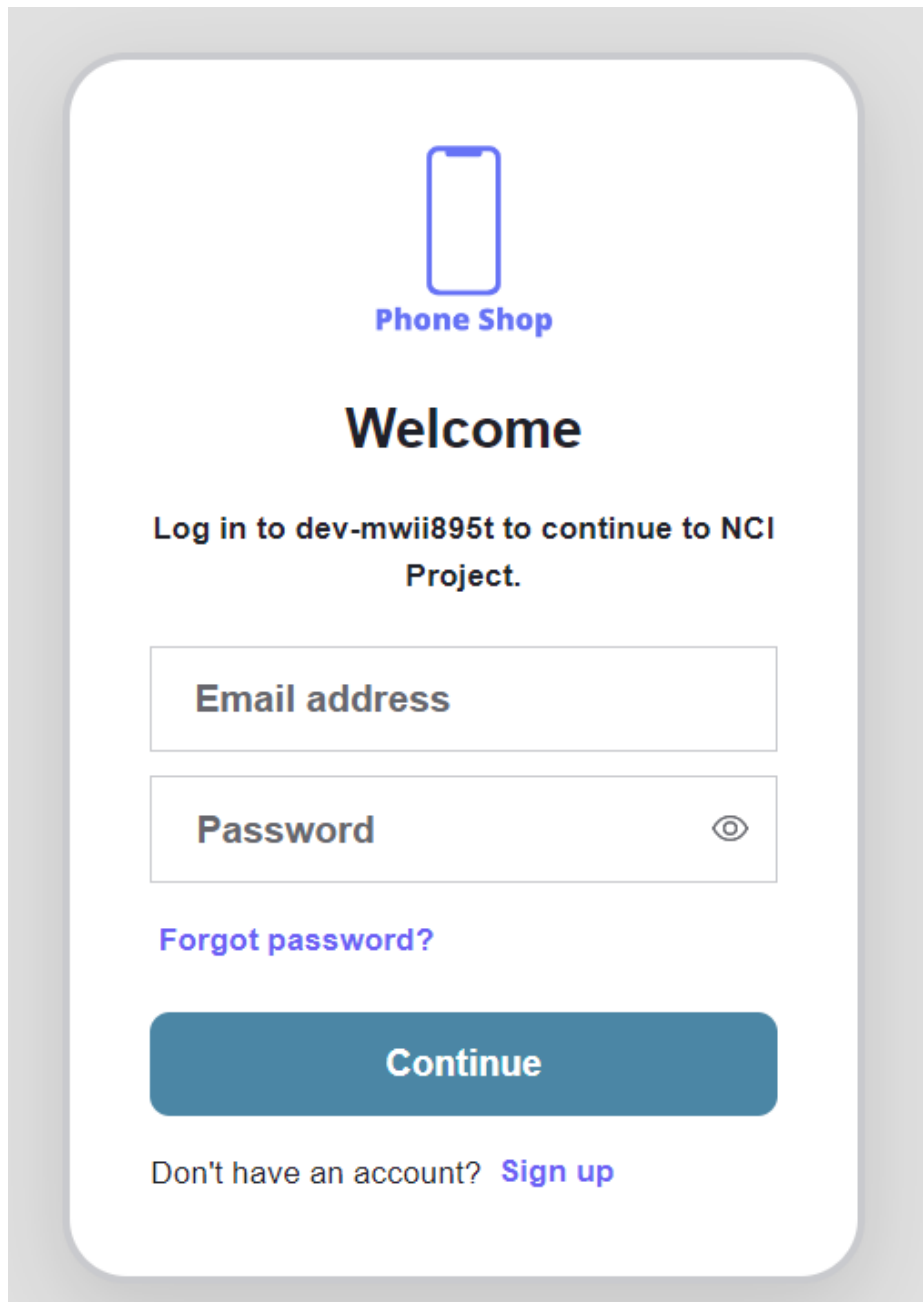
Users + Create User

An easy to use UI to help administrators manage user identities including password resets, creating and provisioning, blocking and deleting users. [Learn more →](#)

Q Search for users Search by User X Reset

Name	Connection	Logins	Latest Login
 hahesyshane@gmail.com hahesyshane@gmail.com	Username-Password-Authenti...	48	4 days ago

When a User clicks Signup or Login they will be greeted with the following screen



I configured the User Data via the Auth0 dashboard where I added an attribute to the User JSON File called admin which is a Boolean value when a user sign in, I check if the User is an Admin or not by the following piece of code

```
//Check for what Role the User is
this.auth.user$.subscribe(user_object => {
  if(user_object['role'] == 'admin'){
    this.admin = true;
  }
});
```

Auth0 will create a JSON Object of each user in its Cloud database this is what the objects looks like

```
email: "hahesyshane@gmail.com"
email_verified: true
name: "hahesyshane@gmail.com"
nickname: "hahesyshane"
picture: "https://s.gravatar.com/avatar/1e936cf3cd69637b5bb6f346b147d387?s=480&r=pg&d=ht
role: "admin"
sub: "auth0|62c8ef179160dacc2311ba27"
updated_at: "2022-08-20T16:39:30.094Z"
```

As you can see this object holds a lot of useful information about the User, I can do a lot with this JSON Object as I can display information the User and make HTML Changes based on their JSON Object for example if role is set to admin, I can create a component based of this called the admin panel that will display admin features based on the JSON Property ,

API

So once the User is successfully signed up on the system the API I created in C# Will hands the rest of the Systems functionality first off all when a User navigated to the Products section of the Website they will be greeted with a lot of products they can purchase these products are stored in a MySQL Database so how it works is Angular will first make a Http GET Call to the API this looks like this I wanted the frontend to do the least amount of work for performance reasons I wanted the backend API to most of the heavy lifting so that the frontend was littered with Code and Bugs so this API Call is very simple as Angular's built in HTTP Service makes it easy to use

```
//Get Products from API
this.http
  .get<any>("https://localhost:7005/Products/All")
  .subscribe((data) => {
    this.products = data;
  });
```

I then Store the Products in Object Array the Array will only accept Objects that match the 'Item' Interface this Interface or Model is created as a standard .ts with setters and getters notice how the Item Model matches the exact same fields as the Product Object from the SQL Database

Here is what the 'Item.ts' class looks like

```
export class Item{  
  
    name: string;  
    price: number;  
    brand: string;  
    desc: string;  
    type: string;  
    img: string;  
    id: number;  
}
```

```
public products: Array<Item>;
```

So, when the API Receives this HTTP Call from the frontend it will communicate with the MySQL Database this is done as following

```

public IActionResult GetProducts()
{
    string select3 = "select * from mobile_direct.products";
    string server = "server=127.0.0.1;port=3306;database=ang_test;uid=root;password=1";
    //Open the SQL Connection
    MySqlConnection conn = new MySqlConnection(server);
    //This ArrayList will hold the User Objects as defined in the User.cs class
    List<Product> ProductList = new List<Product> { };
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(select3, conn);
        Console.WriteLine("SQL Command Executed");
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine(reader["id"]);
            Console.WriteLine(reader["name"]);
            Console.WriteLine();
            Console.WriteLine("----- \n");

            Product p = new Product(
                (int)reader["id"],
                (string)reader["name"],
                (float)reader["price"],
                (string)reader["brand"],
                (string)reader["desc"],
                (string)reader["type"],
                (string)reader["img"]
            );

            ProductList.Add(p);
        }
        conn.Close();
        return Ok(ProductList);
    }
    catch (SqlException ex)
    {
        conn.Close();
        Console.WriteLine("there was an issue!", ex);
        return Ok(ex);
    }
}

```

This block of code retrieves the Products from Database and stores them in an array list so the user from the front end can retrieve this array its just a simple Query String used to Collect all the Products Data 'Select * from db.products'

When creating an Order

the API will take information from the Auth0 users JSON Object and extract the Users Email and use this as a Field to be stored in the Database this way I don't have to use any normalization techniques in my Database and create extra columns I found using Data Separation between the Users and the Products and Orders made my application simpler and lighter and easier to code with. So once the User is finished purchasing and their checkout process is complete, they will be sent to the Completed component that will display to the user a Thanks for Shopping with Mobile Direct page

```

var Payload_Order = {
    "email" : this.AuthEmail,
    "products" : this.arr.toString(),
    "total" : this.Total,
}

this.http.post<JSON>("https://localhost:7005/Orders/New",
Payload_Order).subscribe(data => {
    console.log(data);
});

```

The Backend API will then execute a SQL Command to store the User Order information in the Database the User can then navigate to the profile section of the website and view their orders

```

// This Method is for Creating Orders
[HttpPost("~/Orders/New")]
public async Task<IActionResult> CreateOrder(JObject Payload)
{
    var email = Payload["email"];
    var products = Payload["products"];
    var total = Payload["total"];

    string server = "server=127.0.0.1;port=3306;database=mobile_direct;uid=root;password=";
    string query = $"INSERT INTO 'mobile_direct'. 'orders'('email','date','products','total')VALUES('{email}',now(),'{products}', {total});";

    //Open the SQL Connection
    MySqlConnection conn = new MySqlConnection(server);

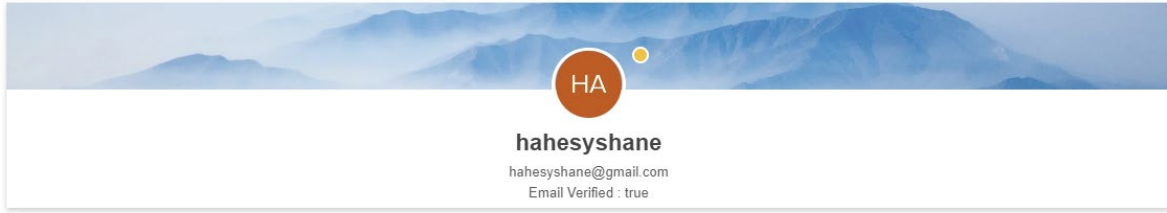
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        Console.WriteLine("SQL Command Executed");
        MySqlDataReader reader = cmd.ExecuteReader();
    }
    catch (SqlException ex)
    {
        conn.Close();
        Console.WriteLine("there was an issue!", ex);
        return Ok(ex);
    }

    return Ok("Order Created");
}

```

This is the Backend that runs when the User creates a new email it takes information from the JSON is retrieved from the User and it uses this as parameters to execute a Query into the Database the API Method only needs 3 pieces of information the Email associated with order, The Products inside the Order and total order amount the Database takes an extra field and this is to tell when the user created the order this is done using the now() MySQL Operator

I wanted to keep the Code within the API very uniform this way it will be easy to expand on and make changes



haesyshane
 haesyshane@gmail.com
 Email Verified : true

Past Orders

Order ID	Products in the Order	Total Amount of Order	Date of Order
13	Galaxy s22	1199.99	2022-08-06T04:53:30
16	Galaxy A53,iPhone 13 PRO	1939.98	2022-08-06T05:13:35

As you can see the Users orders are displayed to them, if the User is an Admin, they can view all orders created by all Users this looks like this



Total Amount : €35582.61
 Total Orders : 17

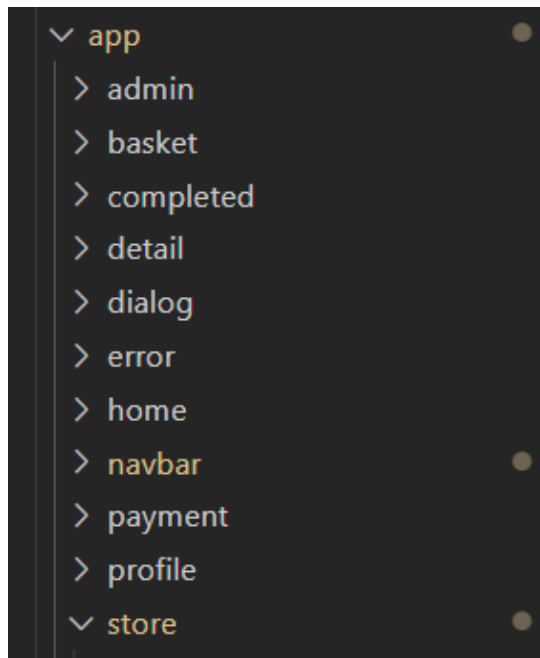
Order ID	Email	Products in the Order	Total Amount of Order	Date of Order
9	LucyyRedmond@yahoo.com	Airpods 2, iPhone 13 Pro	€1129.99	2022-08-06T04:51:40
10	JackDarcy@yahoo.com	Airpods 2	€119.99	2022-08-06T04:52:02

GET	/Test	▼
POST	/Orders/Create	▼
GET	/Products/All	▼
POST	/Products/id	▼
GET	/Products/Brands	▼
POST	/Mail	▼
POST	/EMail	▼
POST	/Search	▼
POST	/Orders/Email	▼
POST	/Products/Reccomend	▼
POST	/Orders/New	▼
POST	/Products/UpdatePrice	▼
GET	/Orders/All	▼
GET	/Orders/All/Amount	▼
GET	/Orders/All/Quantity	▼
POST	/Products/Create	▼
POST	/Product/Delete	▼
POST	/User/CreateDB	▼
POST	/User/CreateBrand	▼
POST	/User/UpdateBrand	▼
POST	/User/DecreaseBrand	▼
POST	/User/Data	▼

This is the API I created With all its HTTP Methods

Front End

The Front is probably the most complex component of the Project the Angular application is broken down into component and these are as follows



These components are listed in alphabetical order each component is placed within the Project's main wireframe the app.component.html the navbar component is what routes between each component via a click on the navigation bar this is achievable thanks to Angular Routing this what Angular routes look like

```
const routes: Routes = [  
  { path: '', component: HomeComponent },  
  { path: 'store', component: StoreComponent },  
  { path: 'admin', component: AdminComponent },  
  { path: 'profile', component: ProfileComponent },  
  { path: 'basket', component: BasketComponent },  
  { path: 'payment', component: PaymentComponent },  
  { path: 'product/:id', component: DetailComponent },  
  { path: 'completed', component: CompletedComponent },  
  { path: '**', component: ErrorComponent }  
];
```

Each component is given a path the path is base URL with an extension so for example Store component would be 'mobiledirect.com/store' AS you can see the Home Route is denoted with an empty string this is because this is the Component I want to be displayed when the URL does NOT have any extension the navigation bar code looks like this the Route with ** as its destination this is angular way of error handling if a User navigates to a URL that is un-routed it will display to the user a 404 Error Page, The Product/ID route takes in to account the Product ID that a user has clicked on and will show the Details of only that Product ID

```

<ul class="nav_links">
  <li><a routerLink="">Home</a></li>
  <li><a routerLink="store">Store</a></li>
  <!-- ng if for Cart Size data stored in Cookie Storage-->
  <li *ngIf="cartSize == 0" > <a routerLink="cart"><i class='fas fa-shopping-basket' ></i></a></li>
  <li *ngIf="cartSize > 0" > <a routerLink="cart"><i class='fas fa-shopping-basket'></i> {{cartSize}}</a></li>
  <li *ngIf="admin == true" > <a routerLink="admin">Admin</a></li>

```

As you can see each link has a reference to the Route Path from the Routes Array

So, the First Component is Home Component which is the first page a User will see when they land on the website there is not going much going on in backend wise with this component as its mostly just hardcoded HTML Content to make the website look nice but with Angular you can add dynamic functionality to your html code, I created a Login Authenticator without any JavaScript Code it looks like this

```

<div class="auth">
  <ul *ngIf="auth.user$ | async as user">
    Welcome Back {{ user.nickname}}
  </ul>

  <ng-container *ngIf="auth.isAuthenticated$ | async; else loggedIn">
    <button mat-raised-button color="primary" (click)="auth.logout({ returnTo: document.location.origin })"> Log out </button>
  </ng-container>

  <ng-template #loggedIn>
    <button mat-raised-button color="accent" (click)="auth.loginWithRedirect()">Log in </button>
  </ng-template>
</div>

```

Once the User is Logged in it should display their name with the welcome back string behind it



Welcome Back haheesyhane

Log out

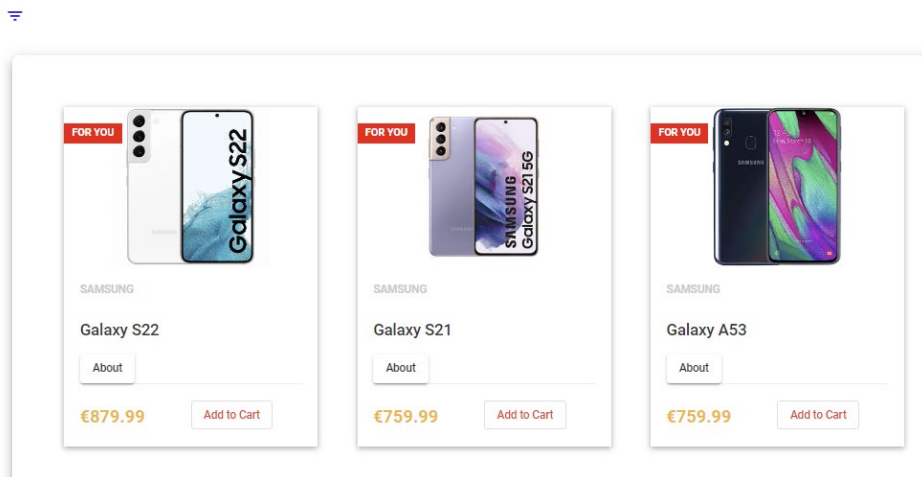
The next Component I will discuss is the Store component, so the Store component is what displays the Products to the user via a HTTP Call to the Backend as previously explained so the Products are displayed using Angular-HTML Code again I loop through my products array and display them as such

```

<div *ngFor="let i of products">
  <div class="product-card">
    <div class="badge">for you</div>
    <div class="product-tumb">
      
    </div>
    <div class="product-details">
      <span class="product-catagory">{{i.brand}}</span>
      <h2> {{i.name}}</h2>
      <button mat-raised-button matTooltip= "Description : {{i.desc}} Type: {{i.type}}"
        matTooltipClass="custom-tooltip"
        aria-label="Button that displays a tooltip when focused or hovered over">About</button>
      <div class="product-bottom-details">
        <button mat-stroked-button color="warn" (click)="addToCart(i)">Add to Cart</button>
        <div class="product-price">€{{i.price}}</div>
      </div>
    </div>
  </div>
</div>

```

As you can see, I can loop through the array and display the products as such this results in the following being rendered to the client



The User can add to cart when they click the button when this happens a recommended item based on the brand will be recommended to the client. The item recommended will always be an accessory, and it's always random on what item will be recommended. This is done by the following

```

//Once the User Picks a Phone
if(this.Recommend == true){
  //We Send the Phone Brand to the API i.e {'SAMSUNG'}
  let Payload = { brand : this.LatestItem.brand };
  this.http.post<any>("https://localhost:7005/Products/Reccmend", Payload).subscribe(data => {
    this.Recommended_Item = data;
  });
  //The API Will execute a SQL Query then return a List of Accesories that match the Users Phone Brand
  let x = this.Recommended_Item.length;
  let y = Math.floor(Math.random() * x);
  this.RandomItem = this.Recommended_Item[y];
  //We then display a Random Item from this Recommended List to the User
}

```

The Add to Cart button will load the Dialog Component which contains this code. It will make a call to the API which will return products based on the brand they chose, but the product they chose must be of type Phone as we only recommend accessories based on their phone they picked.

This is what the API Method looks like

```

[HttpPost("~/Products/Recommen*")]
0 references
public async Task<IActionResult> ProductsByRecommen(JObject Payload)
{
    var brand = Payload["brand"];
    string server = "server=127.0.0.1;port=3306;database=mobile_direct;uid=root;password=redking";
    string query = $"SELECT * FROM 'mobile_direct'.products'WHERE brand = '{brand}' AND type = 'ACCESSORY'";

    //Open the SQL Connection
    MySqlConnection conn = new MySqlConnection(server);

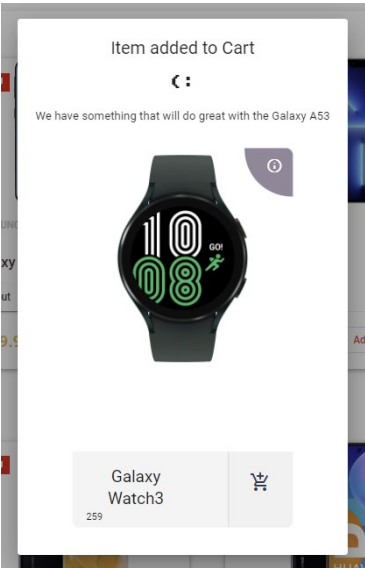
    //This ArrayList will hold the User Objects as defined in the User.cs class
    List<Product> ProductList = new List<Product> { };

    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        Console.WriteLine("SQL Command Executed");
        MySqlDataReader reader = cmd.ExecuteReader();

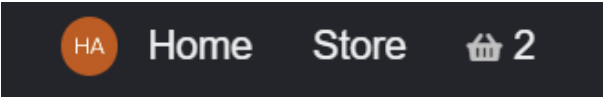
        while (reader.Read())
        {
            Product p = new Product(
                (int)reader["id"],
                (string)reader["name"],
                (float)reader["price"],
                (string)reader["brand"],
                (string)reader["desc"],
                (string)reader["type"],
                (string)reader["img"]
            );
            //CHANGES FINISHED;
            ProductList.Add(p);
        }
    }
}

```

In return this is what the finished article looks like displayed to the User



The User can have many items inside there Cart and with the more items they add the navigation bar will indicate the size of their cart



As you can see here the user has two items inside there cart, The Cart is held via Local storage to hold the Items data number of items inside the Cart this is done Local Storage can be tricky to work with as they can only hold flat strings in key: value pair style this is why it's best to convert you're JavaScript object that contains Items into a JSON Object then append that to the Storage Item then

you can destringify the Local Storage string back into JSON then back into a JavaScript Object which can then go into an array

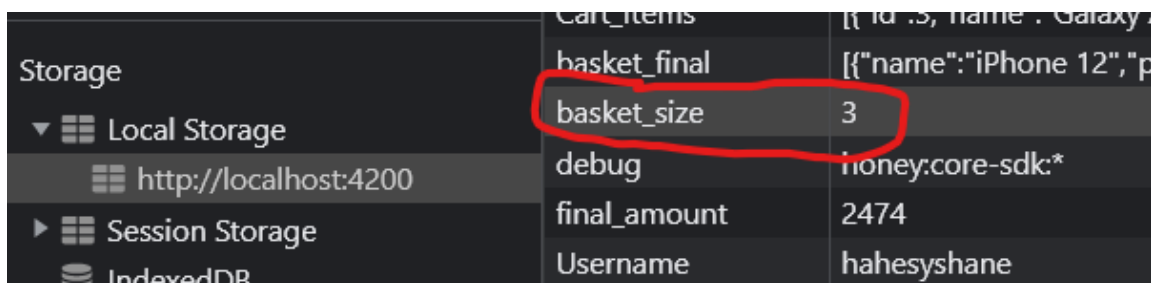
```

//Store the Total Amount in a Cookie
let CookieAmount = Number(this.cookie.get("TotalX"));
let newCookieAmount = CookieAmount + item.price;
this.cookie.set("TotalX", newCookieAmount);

//Store the Cart Size in a Cookie
let CartAmount = Number(this.cookie.get("CartX"));
let newCartAmount = CartAmount + 1;
this.cookie.set("CartX", newCartAmount.toString());

//Store Cart Items in LocalStorage
let myArray = [];
let y = JSON.parse(window.localStorage.getItem("Cart_Items"));

```



Since I want the Local Storage Value that is to be a number rather than a string, I can use JavaScript parsing to convert this value back to Number

The Basket Component then gets this information from Local storage and displays it the end user

```



this.CartT = Number(cookie.get('CartX'));
this.Total = Number(cookie.get('TotalX'));
let n = this.Total.toFixed(2);
this.DisplayTotal = Number(n);

let x = cookie.get('cart_items');

this.Items = JSON.parse(localStorage.getItem('Cart_Items'));

```

This is what is then displayed to the User

PRODUCT	PRICE	QUANTITY	CHANGE QUANTITY	SUB-TOTAL
 Galaxy A53 Category:PHONE	€779.99	4	+ -	€3119.96
 iPhone 13 PRO Category:PHONE	€1179.99	1	+ -	€1179.99

When a Checkout is completed a payment box will be displayed where the user once they confirm the payment and the authentication is successfully we will then USE the API To interact with Sendinblue's API this will create a Order Confirmation for the User in question, The Sendinblue API

works as follows you to send of a http POST request with a valid API-Key and the body of the request will contain the users data and basket, so to this I use the Front End to make the Http Call to the Sendinblue API with the Users Data, this Request is made once the User has Completed the Order and Entered valid details into the Payment Box

This is what it looks like :

```
let j = {
  "to": [
    {
      "email": user_email
    }
  ],
  "templateId": 1,
  "params": {
    "firstname": this.Name,
    "date": date_string,
    "address" : "21 Sea View Lawns, Swords",
    "total": total.toFixed(2),
    "items": JSON.parse(x)
  },
  "headers": {
    "charset": "iso-8859-1"
  }
};

//Options For the Email Service
const httpOptions = {
  headers: new HttpHeaders({
    'accept' : 'application/json',
    'Content-Type': 'application/json',
    'api-key': 'Enter Api Key Here !'
  })
};

//Sending all the Data to the Emailing API
this.http.post<any>('https://api.sendinblue.com/v3/smtp/email' ,j , httpOptions).subscribe(data => {
```

I Declare a JSON Object and fill it with the Users Details, the “Items” Value is the Array from Local Storage that holds the Users Basket Data once this is complete the User will then receive an Order Confirmation Email that look like this, but since Local Storage can hold Basic Strings, I must retrieve that String from the Local Storage and then Convert it into a JSON Object that why there is the ‘JSON.parse’ method being used

MOBILE DIRECT
Thank you **haesyshane** !

✨


Thanks for choosing Mobile Direct you are a legend

Order No: 007345142
Order date: 05/12/2022

Track your order

Continue shopping

Shipping address:
mala



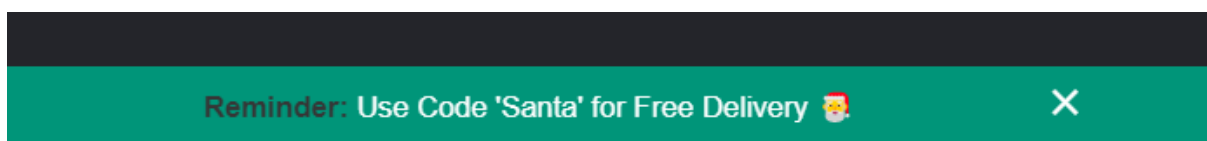
Galaxy A53
PHONE
Article No:
0987654321
inclusive 19.9% VAT

x2 **\$759.9 €**

Discount	-120€
Subtotal	1432€
Tax	143,2€
Shipping fee	50€
TOTAL	1575,2€

Promo Codes

The shop accepts Promo Codes and will deduct the code from the Total in the basket area for this to work the User must have at least One Item in their Basket Mobile Direct advertises a Free Delivery Promo code in the Basket Component in the form of an overlay display banner here is a look at the Banner



When the Customer Inserts this String into the input field at the bottom of the Basket View if its correct, they will be greeted with a Message confirming this

PROMO CODE	ORDER SUMMARY								
<i>If you have a promo code, please enter it in the box below</i>	<i>Shipping and additional costs are calculated based on values you have entered.</i>								
<input type="text" value="santa"/> <input type="button" value="📄 Apply coupon"/>	<table border="1"> <tr> <td>Promo Deductions</td> <td>-€10</td> </tr> <tr> <td>Shipping and handling</td> <td>€0</td> </tr> <tr> <td>Tax 10%</td> <td>€196</td> </tr> <tr> <td>Total</td> <td>€2156</td> </tr> </table>	Promo Deductions	-€10	Shipping and handling	€0	Tax 10%	€196	Total	€2156
Promo Deductions	-€10								
Shipping and handling	€0								
Tax 10%	€196								
Total	€2156								
Nice, Free shipping on us 📄									

But if the Customer enters an Incorrect Promo Code a different message will display the Input Field the

```

<!--This is the HTML Logic for Promo Codes -->
<!--Depending on the Code they entered a certain block will
run-->

<div *ngIf="promo_active == true">
  <h2>Nice, Free shipping on us 📄</h2>
</div>

```

There is a Boolean stored in the Typescript component and if the Input Field entry matches the stored Promo code it will run the block of code

Data Tracking

I will now begin to discuss how the Data Tracking works and how the System recommends Items to the User based on their viewing habits the Database will hold these viewing habit records; a User must have an Account with Mobile Direct for this to work. When a User clicks on a Products Image in the Shopping component as explained before this opens up this detail component for that Product, so if it's for the First Time it will create a Table in the Database for the User, the Table will be named after the User. The Name is their Email minus the '@EmailProvider' so for instance since my email address is haesyshane@gmail.com it will create a table just called 'haesyshane' in the Database this is done as follow's

1. In the Front-End we will send a request to my API as follow

```

//Create a DB with the User's name if there isnt one already created
http.post<any>("https://localhost:7005/User/CreateDB", i).subscribe();

```

2. Then in the Backend it will insert a Query into the Database

```

string query1 = $"CREATE TABLE `mobile_direct`.{email}` (`Brand` VARCHAR(255) NOT NULL, `Clicks` INT NULL, PRIMARY KEY(`Brand`))";

MySQLConnection conn = new MySQLConnection(server);

//QUERY One: Create new Table for User

```

This Query will create a Table for that User if one doesn't already exist in the Database,

The Design of the Table I Wanted to keep as simple as possible the Table consists of 2 Columns 'Brand' and 'Clicks' when the User clicks on for example a Samsung Galaxy s22 it

will insert Samsung into the Table and Increment the Clicks column for that brand by 1 this is done as follows

```
//Insert the Brand into the DB if its not already there
http.post<any>("https://localhost:7005/User/CreateBrand", i).subscribe();

//Increment the Value associated with the Brand
http.post<any>("https://localhost:7005/User/UpdateBrand", i).subscribe();
```

As we can see if we click on a Samsung product for the first time it will insert that brand into the Table

```
//This will Insert a new brand into the Table
string query2 = $"INSERT INTO `mobile_direct`.`{email}`(`Brand`,`Clicks`) VALUES ('{brand}' , 0)";
```

This will increment the Value of that Brand by 1

```
string query3 = $"UPDATE {email} SET Clicks = Clicks + 1 WHERE Brand = '{brand}' ";
```

This is how the Table will look after the User has clicked on a few products

	Brand	Clicks
▶	APPLE	3
	HUAWEI	4
	LG	3
	SAMSUNG	28
*	NULL	NULL

As we can see from the shoppers habits, they are more than likely going to buy a Samsung product rather than anything else we then want the Front-End to retrieve this Information in JSON Format this is done as follows

First, we make a Request to the API

```
//Let get the Latest JSON File that has the Users Data tracked
http.post<any>("https://localhost:7005/User/Data" , p).subscribe(data => {
```

The API Will then Run this Query into the Database

```
//Get the User Data and Order there Most Clicked Brands by Descending
string select3 = $"select * from mobile_direct.{email} ORDER BY Clicks DESC";
```

In return we will receive a JSON Object which will look like

```
▼ (4) [{...}, {...}, {...}, {...}] ⓘ
  ▶ 0: {brand: 'SAMSUNG', clicks: 28}
  ▶ 1: {brand: 'HUAWEI', clicks: 4}
  ▶ 2: {brand: 'APPLE', clicks: 3}
  ▶ 3: {brand: 'LG', clicks: 3}
  length: 4
```

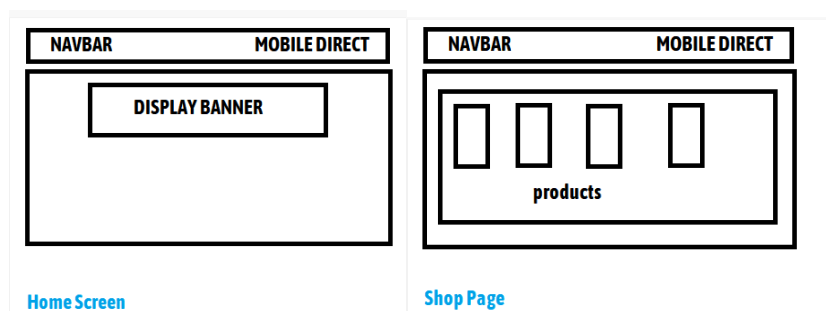
this when the User hits a Brand with over 8 Clicks this means they are showing Interest in that brand of products so we will further entice them by offering them a Promo Code to be used for that Brand of Products the User Interface will display this banner as such

```
for(let i = 0; i < this.user_data.length; i++){
  console.log(this.user_data[i]);
  //This will Prove the Customer is liking the Product
  if(this.user_data[i].clicks > 10)
  {
    //Show the Banner with the Promo Code
    this.showPromo(this.user_data[i].brand);
  }
  //This API Method will Set there Clicks back to 5
  if (this.user_data[i].clicks > 15){
    http.post<any>("https://localhost:7005/User/DecreaseBrand",
i).subscribe();
  }
}})
});
```

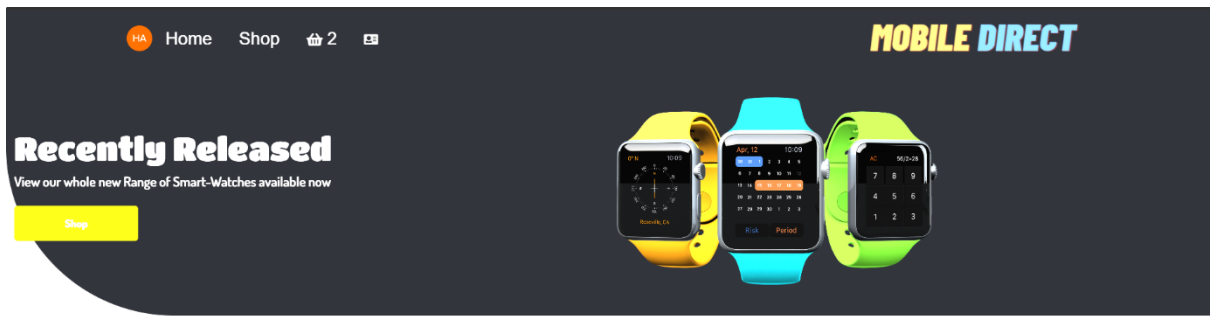
If the User Reaches over 15 clicks and still doesn't Purchase that Item we will reset the Clicks back to 5

2.2. Graphical User Interface (GUI)

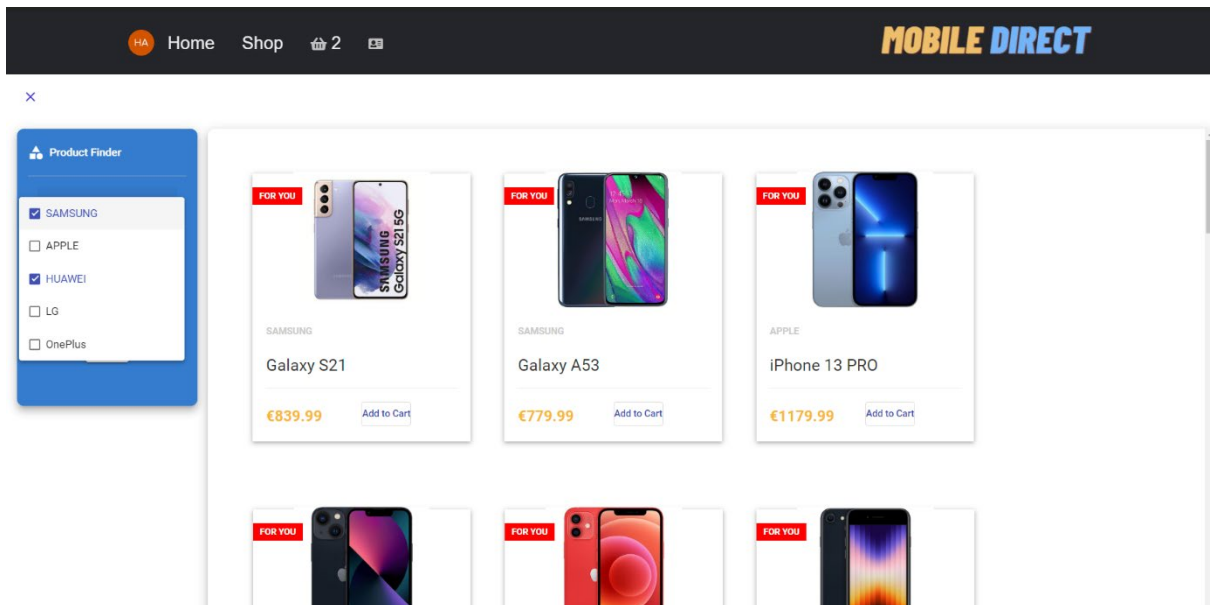
In this Section of the Report, I will be showcasing some of the User Interface for my project before I done any Coding, I began drawing some Wireframes on how I wanted the Application to look these are as follows





This was the original wireframe for the Home Screen



This is the Home page of the Website



The Shop Component where the User can Filter, Search and View there Products

PRODUCT	PRICE	QUANTITY	CHANGE QUANTITY	SUB-TOTAL
 Galaxy A53 Category:PHONE	€779.99	1	+ -	€779.99 ✕
 iPhone 13 Pro Category:PHONE	€1179.99	1	+ -	€1179.99 ✕

<p>PROMO CODE</p> <p><i>If you have a promo code, please enter it in the box below</i></p> <div style="border: 1px solid #ccc; padding: 5px; display: flex; align-items: center;"> <input style="width: 80%;" type="text" value="santa"/> <div style="background-color: #333; color: white; padding: 2px 10px; margin-left: 5px;"> Apply coupon </div> </div> <p>Nice, Free shipping on us 📦</p>	<p>ORDER SUMMARY</p> <p><i>Shipping and additional costs are calculated based on values you have entered.</i></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>Promo Deductions</td> <td style="text-align: right;">-€10</td> </tr> <tr> <td>Shipping and handling</td> <td style="text-align: right;">€0</td> </tr> </table>	Promo Deductions	-€10	Shipping and handling	€0
Promo Deductions	-€10				
Shipping and handling	€0				

Here is Mobile Direct's Basket Component you can see here this is where the User will view their basket


Home Shop 2 📧
MOBILE DIRECT

←

SAMSUNG

GALAXY S21

€839.99



Samsung Galaxy

In stock

★★★★☆ (54 reviews)

PRODUCT TYPE
PHONE

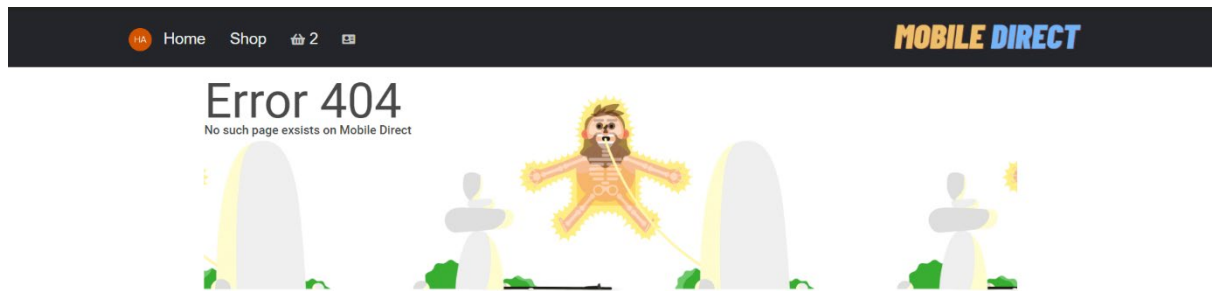
Add to cart

25% OFF

SAMSUNG Products

CODE: PROMO25

Detail Page this is where the User will view the Product in Detail



The Error page will load if the User navigates to a Route this isn't listed in Mobile Direct Routing Application

2.3. Testing

Describe any testing tools, test plans and test specifications used in the project. Provide evidence for and results of all Unit, Integration and End User testing that is carried out.

Testing is a Vital Part for any Software Application test Driven Development is the number one way to prevent Bugs and Glitches with your Software application we need to also be testing our Application throughout the development process in order to maximise the Quality and Integrity of our application, in this Section I will discuss some of the Testing I have done with this Project

Angular is really good for testing as it comes pre-installed with some pretty nice and in-depth Testing Tools, As I discussed before Angular is component Based, we generate Components when developing an Angular Component when we generate a Component it comes included a file especially for testing that component this is called the 'spec.ts' file, It's a typescript File that we use to Write Tests for that Component Especially for example if we want to write a test to make sure the Title of the Page is Mobile Direct we would write

```
it('should have as title 'mobile direct'', () => {  
  const fixture = TestBed.createComponent(AppComponent);  
  const app = fixture.componentInstance;  
  expect(app.title).toEqual('Mobile Direct');  
});
```

Here is an example of some the Test Methods I wrote to test my application

To Test to make sure the Http Request to my Backend that retrieves all the products information was secure I done this :

```
it('should make a secure HTTP request to retrieve products',  
inject([HttpTestingController], (httpMock: HttpTestingController) => {  
  // Arrange
```

```

    const component =
TestBed.createComponent(StoreComponent).componentInstance;

    // Act
    component.getProducts();

    // Assert
    const req = httpMock.expectOne('/Products/All');
    expect(req.request.method).toEqual('GET');
    expect(req.request.headers.get('Authorization')).toBeTruthy();
  }));
});

```

Here is another Test to make sure that the Promocode Form in the Basket Component is working properly

```

it('should mark the promocode form group as invalid if the code is empty', ()
=> {
  // Arrange
  const fixture = TestBed.createComponent(BasketComponent);
  const component = fixture.componentInstance;
  fixture.detectChanges();

  // Act
  component.promocode.get('code').setValue('');

  // Assert
  expect(component.promocode.valid).toBeFalsy();
});

it('should mark the promocode form group as valid if the code is not empty',
() => {
  // Arrange
  const fixture = TestBed.createComponent(BasketComponent);
  const component = fixture.componentInstance;
  fixture.detectChanges();

  // Act
  component.promocode.get('code').setValue('ABC123');

  // Assert
  expect(component.promocode.valid).toBeTruthy();
});
});

```

The Testing Software Angular comes with is Karma which is specially designed to Test JavaScript based web applications, once you write the Test Code you have to open a new terminal and run

The **Ng Test** command and this will generate a full report on your application

End User Testing

I performed some trunk testing on Mobile Direct, this will give me a really good idea on where I'm at in terms of easy of usability, so to perform this test I asked the person being tested a few questions and recorded there responses and times

Here are the questions for this truck test :

- What is the name of this Website
- What section of the page do you think you're on
- Can you create an account
- What is the website about
- Where can view the products on sale
- Where do you logout

Here are the results for Tester Participant number 1 :

User Num	Task	Task Goal	Start Time	End Time	Expected Behaviour	Actual Behaviour
1	What is the name of this Website	User tells me the Name of website	0:15	0:19	User tells me apps name	User tells me the name of the app
1	What section of the page do you think you're on	User must tell me their location in the app	0:23	0:31	User tells me there on the home page	User tells me there on the signup page
1	Can you create an account	User must click the register button	0:35	0:42	User navigates to sign up section	User navigated to signup
1	What is the website about	User must tell the main goal of app	0:46	0:51	User tells me it's an online retailer selling mobiles	User told me it's a shop for selling phones
1	Where can view the products on sale	User must locate the Shop section	0:56	1:05	User navigates to shop section	User navigated to shop section
1	Where do you logout	User must click logout	1:08	1:20	User clicks logout	User clicked logout

User Num	Task	Task Goal	Start Time	End Time	Expected Behaviour	Actual Behaviour
2	What is the name of this Website	User tells me the Name of website	0:15	0:22	User tells me apps name	User tells me the name of the app
2	What section of the page do you think you're on	User must tell me their location in the app	0:27	0:36	User tells me there on the home page	User tells me there on Main Page
2	Can you create an account	User must click the register button	0:38	0:49	User navigates to sign up section	User navigated to signup
2	What is the website about	User must tell the main goal of app	0:52	0:57	User tells me it's an online retailer selling mobiles	User told me it's a selling platform for Tech
2	Where can view the products on sale	User must locate the Shop section	1:06	1:12	User navigates to shop section	User navigated to shop section
2	Where do you logout	User must click logout	1:15	1:20	User clicks logout	User clicked logout

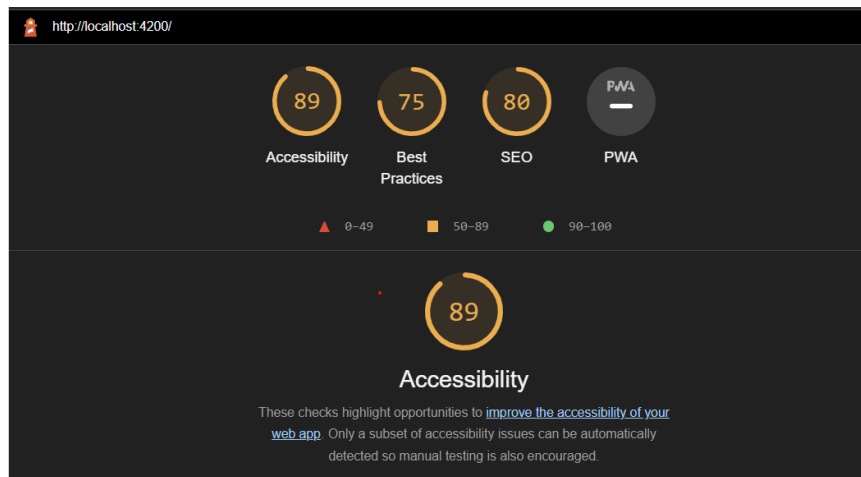
2.4. Evaluation

How was the system evaluated and what are the results? This may consist of usage data. It may also include performance evaluations, scalability, correctness, etc. depending on the focus of the project. Quantitative results may be reported in tables or figures.

Memory – The first thing I will do is Open Google Chromes built in Task Manager and measure the Size of the Website's load Chrome measure this in memory footprint this shows how much RAM each process is taking up as you can see below its 111.972k which is just a little over 100mb to give you an idea YouTube memory footprint is usually around 500mb



Here is the Google Chrome lighthouse Report



3.0 Conclusions

Describe the advantages/disadvantages, strengths, and limitations of the project

Advantages

Tech Stack : Since my Project is Built using Angular and ASP.Net C# These are two technologies that still very well kept and maintained by there creators Google and Microsoft. This means they will always be adding new features and coming out with new versions which means my application will stay up to the date with the latest tech trends and not become stale and slow

The API was Built by me from scratch from the ground up and it features Full CRUD Functionality and principles having a Strong API that was custom made for a Application is a very good thing to have,

Built on JavaScript : JavaScript is the most updated coding language in the world which means that my application will always be avail of new tricks and features that JavaScript comes out with every version

Disadvantages

Mobile Direct could be faster and more lightweight this is something I'll definitely look into it could run faster I do make lots of Http Requests in the applications maybe I Could figure out a way to make these requests less computer intensive and save memory in return boosting performance,

The App may be expensive to Host and Deploy as it will require three different Servers to host all of it One for the Frontend Angular Code, another for the Backend C# Code and then a third for the SQL Database this could create scaling and cost issues when hosting the Application, so this is another disadvantage in my opinion

On looking back at the Goals of this Project and the requirements I set out to the do from the start I can honestly be happy with my efforts I built a fully functionality S.O.L.I.D System that does what I need it to do,

When I Set out my goal, I wanted an E-Commerce platform with a slick and clean User Interface the ability to be able create Accounts, The use of Local Storage so the User can keep can there Data after

their browsing session has ended, the User of Backend that will do all the middle man work between the Frontend and Database, The ability to send the Users Emails

4.0 Further Development or Research

With additional time and resources, which direction would this project take?

If I had more time to work on this Project I think I'd migrate all my Data from the SQL Database to No SQL Database that is JSON Based for example I'd use such services such as Firebase or MongoDB to hold all my Data because using JSON Databases is a lot quicker and lighter and if I was to host the Database the it would cost a lot less if I used a JSON Database than using a traditional SQL Database I'd also integrate PayPal into my project and create a test environment for application and link the PayPal API with the User's profile, I'd also like to give a theming option for the application like a Dark/Light Mode theme where the user can select between a Dark and Light theme this would give my project a more completed and professional look in my eyes,

I'd also use a different User Interface Component Library than Angular Material for the Styling of my website I feel like there's some better options out there for U.I Component's the reason why I used Angular Material is because it's the one that Google recommends the most

I'm defiantly going to keep updating and working on this Application I feel like I want to develop it to be where a User with very little tech-literacy who wants to sell Products Online can use this Platform to sell there Products, rather than being T-shirts, Jewellery anything I want it to be really robust and have a powerful and easy to use Admin Panel that they can do anything and configure the platform to there needs, I'd also like to keep building on the API as I had lots of fun working on the API and I want to keep adding features to it, I could then Charge people a Monthly fee to sell their products using my Platform and do all the bug fixing myself while they worry about selling there products I'd also like to build into it a full fledged analytics tool that will send daily report and use More A.I Algorithms to boost the Intelligence of the Application to give the System more Data so they can make better decisions

5.0 References

Please include references throughout your document where appropriate. See [here](#) for a guide on referencing from the NCI library.