E-Shop System

Author1 (Yad Ghazi Othman, y\_othman@utb.cz)

Author2 (Mohammed Khalid Jamal, [m\_jamal@utb.cz](mailto:m_jamal@utb.cz))

Author3 (Milad Sulaiman Safari Barzani, m\_barzani@utb.cz)

Author4 (Ahmed Hamid Taha Al-Doori, a\_al\_doori@utb.cz)

Content

[1. Introduction 4](#_Toc120040744)

[1.1. Overview of the proposed systems 4](#_Toc120040745)

[1.1.1. System architecture and design 4](#_Toc120040746)

[1.1.2. Clean architecutre pros 5](#_Toc120040747)

[1.1.3. Clean archirecutre cons 5](#_Toc120040748)

[1.2. System limitations 5](#_Toc120040749)

[1.2.1. Genral problem architecutre 5](#_Toc120040750)

[2. Functional analysis 6](#_Toc120040751)

[2.1. Functional requirements 6](#_Toc120040752)

[2.2. Non-functional requirements 6](#_Toc120040753)

[2.3. Actors 7](#_Toc120040754)

[2.4. Use case model 7](#_Toc120040755)

[2.5. Specification of use cases 7](#_Toc120040756)

[2.6. Implementation of requirements 8](#_Toc120040757)

[3. System architecture 9](#_Toc120040758)

[3.1. Class model 9](#_Toc120040759)

[3.2. Data model 9](#_Toc120040760)

[3.3. Realization of UC 10](#_Toc120040761)

[4. Description of the proposed app 11](#_Toc120040762)

[4.1. Wireframe 11](#_Toc120040763)

[4.2. Application description (if app exists) 11](#_Toc120040764)

# Introduction

## Overview of the proposed systems

* The system will be used as an online shop, the system should not be aimed to one specific category, the system can contain one or more category such as (Electronics, clothes, headsets and more)
* The system can make huge benefits for people that want to start their own business or company, they can use the system as their online E-shop to show their available products, the system should be adaptable to any product categories, the system also can be used in all different fields (Pharmacy, Food and supplements, Electronics, …etc.)

### System architecture and design

The system should use the Server-side architecture where the whole system is divided into

* Client side
  + Client side can be any front-end framwork or vanila javascript with HTML, CSS
* Server side
  + While in the server side we will be using the **clean architecture design (currently the application uses the MVC structure but in the future we are planning to move to clean architecture)**, where we have several layers (Presentation layer, Domain layer, Application layer, Infrastructure layer)
  + The main beneift of the clean architecture desing is that the system later can adapt to any client application, in other word By employing Clean Architecture, you can design applications with very low connection and independent of technical implementation details, such as databases and frameworks. That way, the application becomes easy to maintain and flexible to change.
  + The clean architecutre design flow works as nested refrences where the outside can access the inside (meaning we can only go forward, but not backward), for example the domain layer can not access the outside layer, while the outside layers can access the inner layers
* Database side:
  + The system should be adaptable to any database type, since our system in the server side we will use object relational mapper as a layer between the server side and database side

### Clean architecutre pros

Clean architecture has a number of principles that can be summarized here:

* Independent of Frameworks
* Testable
* Independent of UI
* Independent of Database
* Independent of any external agency
* Greate for long live projects
* Parllel teams
* Greate for big teams
* Scalability
* Maintainability

### Clean archirecutre cons

Clean architecutre also has some down sides but they are too little compared to the pros of this architecture:

* Boilerplate code
  + sections of code that are repeated in multiple places with little to no variation.
* Mutiple ways to implement

Diagram

Description automatically generated

## System limitations

### Genral problem architecutre

The only main problem in this general architecutre is the when the application grows the folder architecutre and folder nesting grows hugly and navigation between the folders of the system will become harder, otherwise this architecture with the use of MeditoR and CQRS the application and adopt to many changes with no problem.

# Functional analysis

## Functional requirements

A picture containing diagram

Description automatically generated

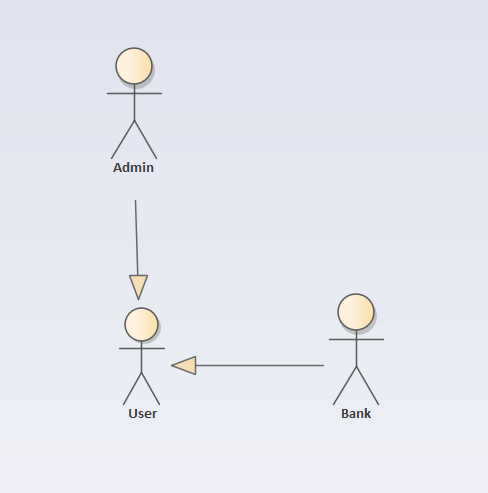
## Non-functional requirements

Graphical user interface, text, application

Description automatically generated

## Actors

We can see that we have 3 actors. The Admin of the website and an User which both can be describes as internal users. There is a generilization relationship between the user and the admin which means the admin can take the role of the user with all its attributes. There is also a bank actor which is an external system that generalizes the user by doing everything the user does.



Obrázek 1 Actors

## Use case model

## 

Obrázek 2 Use Case Diagram

## Specification of use cases

Firstly we have the admin, which can control the each aspect of the webpage. He can control the products that are on the website and can decide to add or remove items. Further more, the Admin can view all the website orders and edit website services. In addition, he can vew all the orders and create discount coupons for all the users to use. Secondly we have the actual user itself. The user can the the most common steps including, creating a profile, logging in, viewing their order, logging out, signing in, viewng their items, checking out... and many more. All the things that are done by the user are also able to be done by the admin and the bank which is considered as an external system for confirming the payments done by any user.

Table 1: Template for User Creation

|  |  |  |
| --- | --- | --- |
| Title: User Creation | | |
| ID: UC01 | | |
| Characteristics:  Create a profile for the new user | | |
| Primary actor:  User | | |
| Secondary actors:  Admin | | |
| Entry conditions:  Before making a purchase(mandatory), or can be done on earlier stages(optional) | | |
| Output conditions:  A valid profile is created | | |
| Main scenario: | | |
| Step | Actor/System | Description |
| 1 | User | Fill personal information |
| 2 | User | Confirm Information |
| 3 |  |  |
|  | | |
| Alternative scenarios: | | |

Table 2: Template for editing items in the Cart

|  |  |  |
| --- | --- | --- |
| Title: Edit Cart | | |
| ID: UC02 | | |
| Characteristics:  Remove or add any item to the cart | | |
| Primary actor:  User | | |
| Secondary actors:  Admin | | |
| Entry conditions:  No conditions are necessary | | |
| Output conditions:  An update to the cart | | |
| Main scenario: | | |
| Step | Actor/System | Description |
| 1 | User | Click on prefered item |
| 2 | User | Add/Remove Item from the cart |
| 3 |  |  |
|  | | |
| Alternative scenarios: | | |

Table 3: Template for Product Control

|  |  |  |
| --- | --- | --- |
| Title: Product Control | | |
| ID: UC03 | | |
| Characteristics:  Make any changes to products in the webpage | | |
| Primary actor:  Admin | | |
| Secondary actors: | | |
| Entry conditions:  Admin should have the proper access | | |
| Output conditions:  Item Could have been added/edited/deleted | | |
| Main scenario: | | |
| Step | Actor/System | Description |
| 1 | Admin | The Admin can make chages to the products in the website |
| 2 |  |  |
| 3 |  |  |
|  | | |
| Alternative scenarios: | | |

Table 4: Template for Checkout

|  |  |  |
| --- | --- | --- |
| Title: Checkout | | |
| ID: UC04 | | |
| Characteristics:  Filling in the billing information | | |
| Primary actor:  User | | |
| Secondary actors: | | |
| Entry conditions:  The user must have a valid profile and have items in the cart | | |
| Output conditions:  Confirmation of payment must be accepted by the bank | | |
| Main scenario: | | |
| Step | Actor/System | Description |
| 1 | User | Checking out |
| 2 | Bank | Confirm payment |
| 3 | Admin | Export the data |
|  | | |
| Alternative scenarios: | | |

Table 5: Template for Viewing All Orders

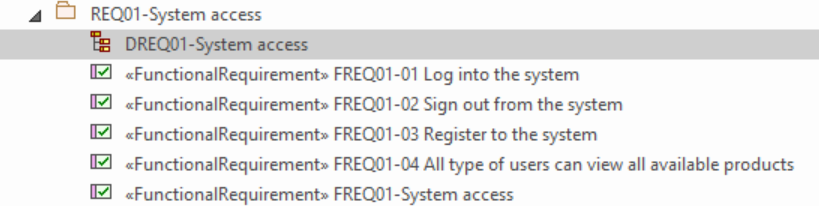
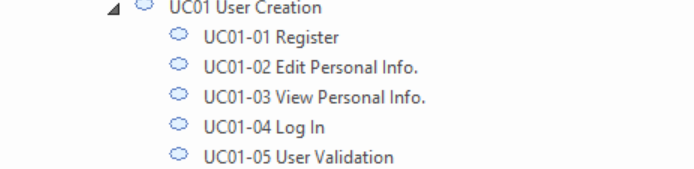
|  |  |  |
| --- | --- | --- |
| Title: View All Orders | | |
| ID: UC05 | | |
| Characteristics:  Create discount coupons | | |
| Primary actor:  Admin | | |
| Secondary actors: | | |
| Entry conditions:  The product selection | | |
| Output conditions:  Price will be updated | | |
| Main scenario: | | |
| Step | Actor/System | Description |
| 1 | Admin | Select the item |
| 2 | Admin | Create a discount coupon |
| 3 | Admin | Update the product |
|  | | |
| Alternative scenarios: | | |

Table 6: Template for Website Control

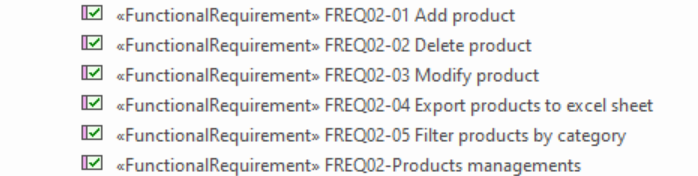
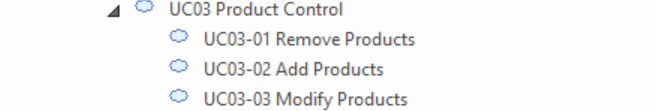
|  |  |  |
| --- | --- | --- |
| Title: Website Control | | |
| ID: UC06 | | |
| Characteristics:  View website Users | | |
| Primary actor:  Admin | | |
| Secondary actors: | | |
| Entry conditions:  Should have correct access | | |
| Output conditions:  List of Users should be viewed | | |
| Main scenario: | | |
| Step | Actor/System | Description |
| 1 | Admin | Show all users and a list of all users should be viewed |
| 2 |  |  |
| 3 |  |  |
|  | | |
| Alternative scenarios: | | |

## Implementation of requirements

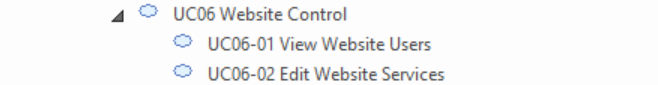
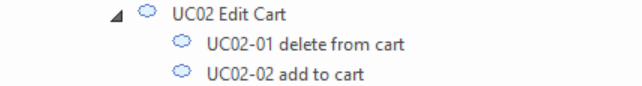
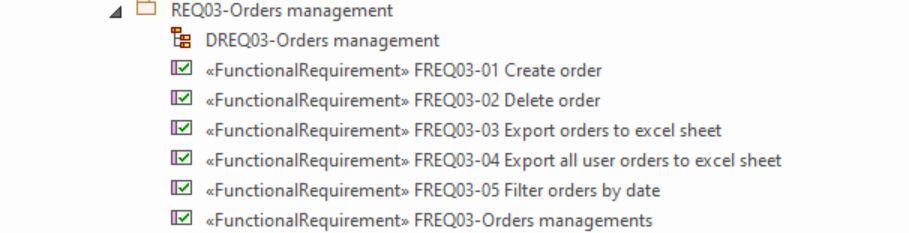
The User or any other actor that has a generalized relationship with the user is able to do any of the following requirements that can be seen in both requirements and the use cases

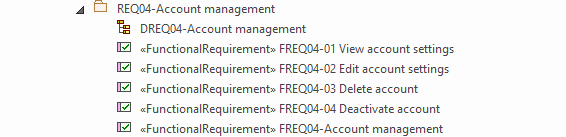
-----------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------

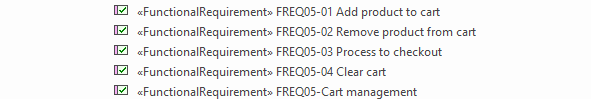


-----------------------------------------------------------------------------------------------------------------



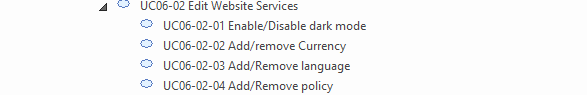
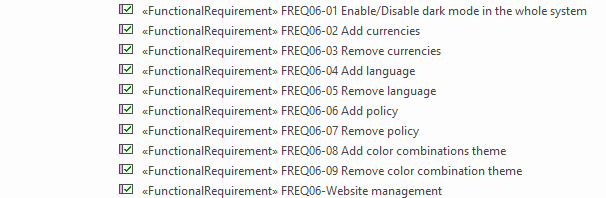
All the other requirments can be shown elsewhere

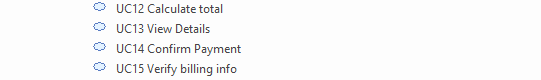
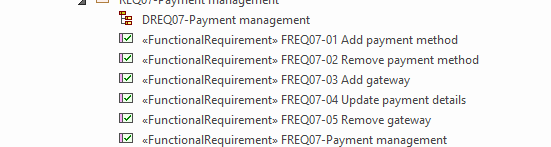
-----------------------------------------------------------------------------------------------------------------



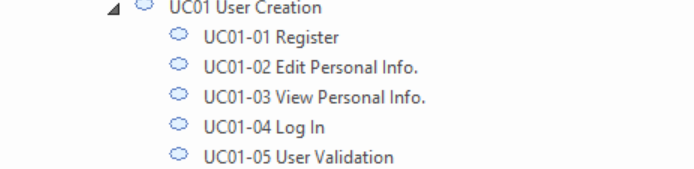


-----------------------------------------------------------------------------------------------------------------





-----------------------------------------------------------------------------------------------------------------



-----------------------------------------------------------------------------------------------------------------

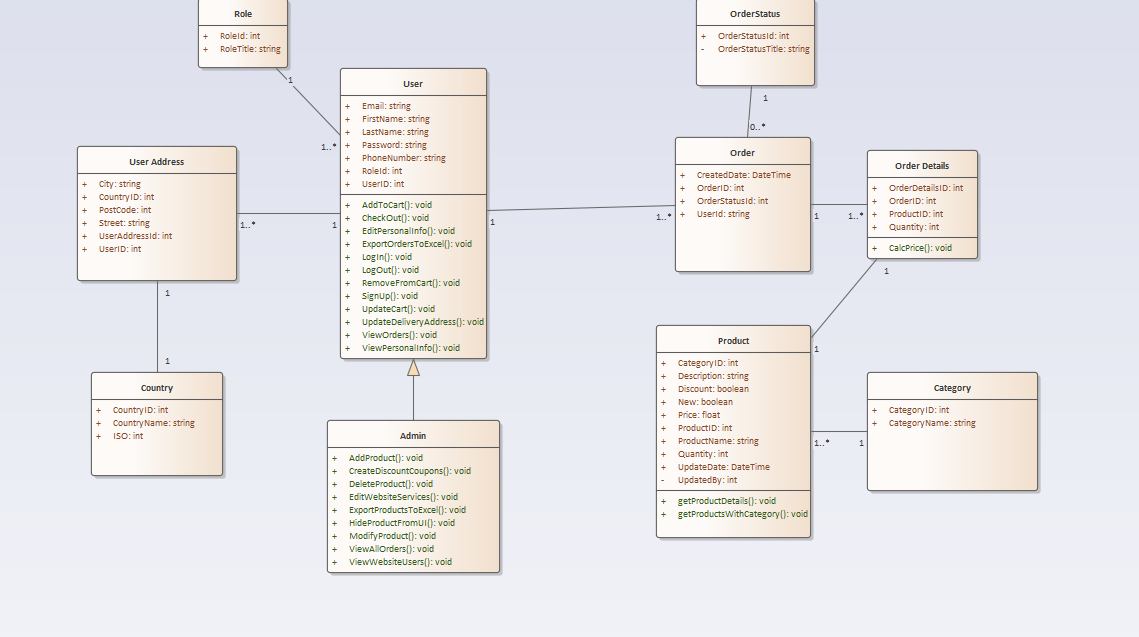
Each requirment has been fullfiled with each use case with some tweeks depending on how you want to handle some parts of the application. We can access each part of the requiments from small details to product managment to more diffecult parts like confirming payments.

# System architecture

* In the division into modules and subsystems, it is possible to use the component model, deployment diagram, etc.

## Class model

Class diagram is a blueprint from our project. Here is our Class Diagram, which displays a class model for our E-Shop project. Our purpose of the Class model is to show our terms and classes as a diagram. Our class model consists of 10 classes (Role, User, User Address, Country, Admin, Order, Order Details, Order Status, Product and Category). Each class has it is own responsibilities, that describes attributes, operations, and its relations with other classes.



In (Role) Class we have a unique ID and a RoleTitle. A Role can belong to many Users and Admins, but Each User belongs to one Role.

A User have a unique ID, Email, First Name, Last Name, Password, Phone Number. The User have a right to (AddToCart, CheckOut, EditPersonalInfo, ExportOrdersToExccel, Login, LogOut, RemoveFromCart, SignUp, Update Cart, UpdateDeliveryAddress, ViewOrders, ViewPersonalInfo).

Admin is inhertis from User, Admin has the same attributes and functions as the User, but also it has own functions that the User does not have it (AddProduct,

CreateDiscountCoupons, DeleteProduct, EditWebsiteServices, ExportProductsToExcel(Its exports all the products to Excel Sheet), HideProductFromUI, ModifyProduct, ViewAllOrders, View WebsiteUsers).

User can have many Addresses, while User Address can belong to one User. User Address can have (City, CountryID,PosteCode,Street,UserAddressID, UserID).

Coutry can be belong to one User Address and User Adress belongs to one country.

Diagram

Description automatically generated

User with a unique ID can place many Orders, while Orders belongs to one User, Order can have (CreatedTime, OrderID, OrderStatusID, UserID).

OrderStatus has a unique ID which linked to Order, Order Status can belong to many Order.

OrderStatus can have (OrderStatusId, OrderStatusString).

Order can belong to many Order Details, which Order Details have(OrderDetailsID, OrderID, ProductID, Quantity). Order Details have a function (CalcPrice) for calculation price.

OrderDetails belong to one Product and Product can have one OrderDetails. Product can have (CategoryID, Description, Discount, New(If the product is new or not), Price, ProductID, ProductName, Quantity, UpdateDate, UpdatedBy). Product have a functionality which you can (getProductDetails, getProductWithCategory).

Each Category belongs to Many product and each product has it is own category. Category can have (CategoryID, CategoryName).

Diagram

Description automatically generated

## Data model

* ERD:

Diagram, schematic

Description automatically generated

## Realization of UC

Diagram

Description automatically generated

- Here, an actor plays the role of a user who opens our website, browses the products, and adds an item to their shopping cart.Diagram

Description automatically generated

- The user wishes to buy the item and go to the checkout.

If the user is already registered and selects the proper payment option, he receives a payment confirmation and the order is confirmed.Diagram

Description automatically generated

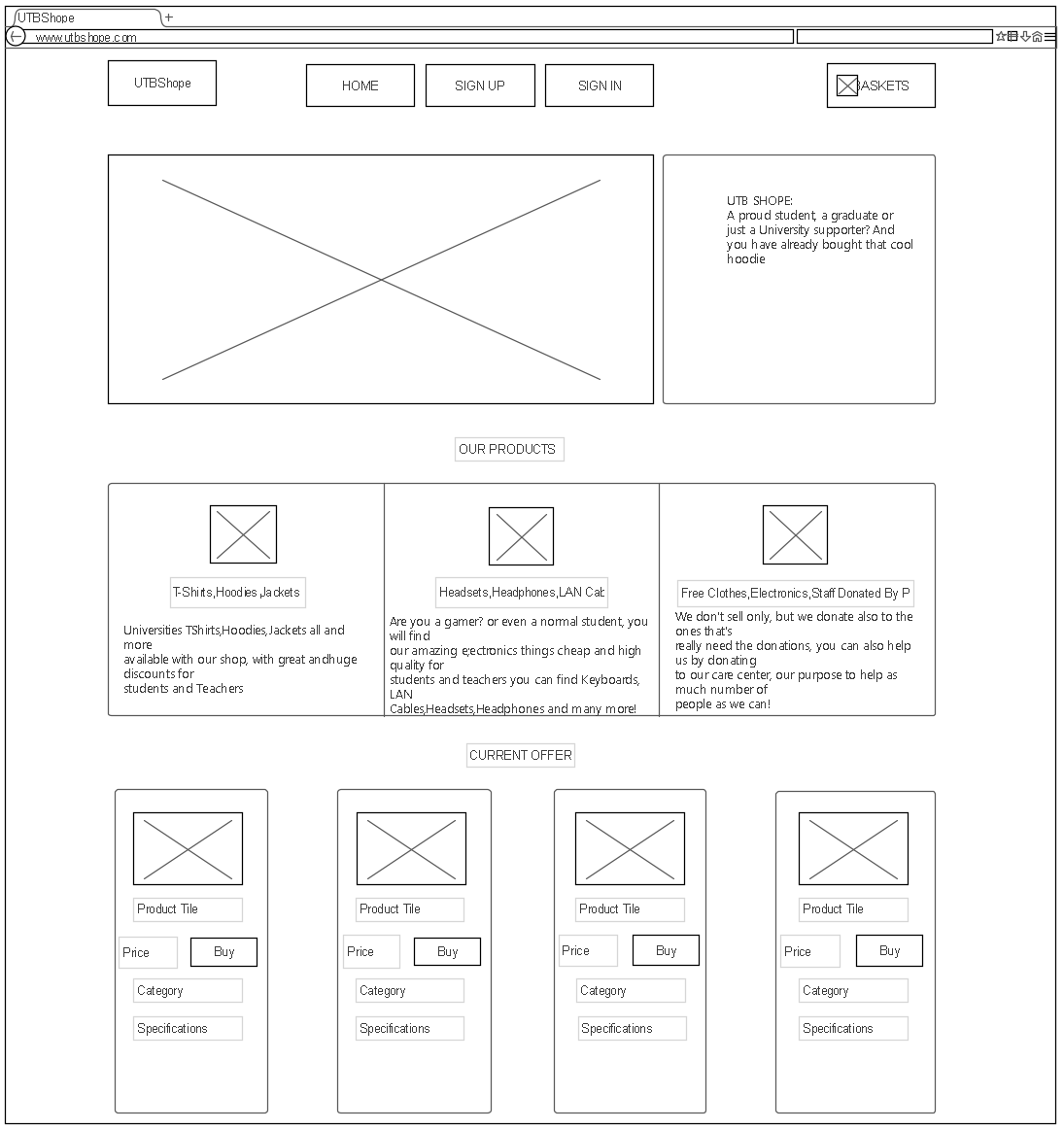
-To receive payment confirmation and have the order completed, the user must enter correct and valid information if the payment method they used was incorrect.

If the user is not already registered, he will not be able to checkout until he does so. Once registered, he will be able to check out using the proper payment option, receive payment confirmation, and confirmation of the order.

# Description of the proposed app

## Wireframe

- Here is a sketch of the system that will be built.

The drawing will be used later by the web developers to determine exactly what the customer had in mind, and the wireframe offers them an idea of the scale of various elements.

## 

## Application description (if app exists)

### Application technolgoies

For the front-end part the application will be build using the following techs:

* HTML5
* CSS3
* Jquery
* Vanila JavaScript

For the back-end the application will be build using the following techs:

* C#
  + ASP.NET MVC
  + EF Core as object relational mapper
* MS SQL Server DB

### Application Roles

* Admin:
  + An admin will be able to modify products and website services as describe in the documentation
  + An admin also able modify website services
* User:
  + A verfied registered user that can order items
* Database adminstrator (outside the application):
  + This role is not inside the application but it is important to mention it, since he will be able to modify users (delete, add...etc)

### Authetication and authorization

The authetication and authorization will be done from the usermanger ASP.NET Core Identity package.

The process of persistent user login will be doing using Http only secure cookie

### Current and future system architecutre

* Current system architecture:

The current system architecture is MVC (Model-View controller) where models are the entities of the application and controller are the endpoint to return the view when they are called (views are just pages or components).

* Future system architecture:

The future system architecture as we mentioned will be **clean architecture.** We will try to update the system in the future, so that the system can adapt to any changes

### Cart and orders

The application will have the ability to let the user add items to his carts and it will be shown how much items inside the cart and what is the total price, all of this is done using session storage

### Responsivity:

The website is designed in way that is mobile friendly, means the website is reponsive on all different type of devices.

### Snapshot of the real application

Graphical user interface, website

Description automatically generated