**Server-Side Development Design Document**

Student number – 570421

Submission date – 9th February 2024

Table of Contents

[Introduction 3](#_Toc158151281)

[Requirements list 4](#_Toc158151282)

[Non-functional requirements list 4](#_Toc158151283)

[Functional requirements list 5](#_Toc158151284)

[Design choices 7](#_Toc158151285)

[Minimalism & white space 7](#_Toc158151286)

[Icons 8](#_Toc158151287)

[Error handling & feedback 8](#_Toc158151288)

[Security 8](#_Toc158151289)

[Wire frames 8](#_Toc158151290)

[Lo-Fi 9](#_Toc158151291)

[Hi-Fi 13](#_Toc158151292)

[Web structure 15](#_Toc158151293)

[API structure 16](#_Toc158151294)

[Data dictionary 17](#_Toc158151295)

[Tbl\_product 17](#_Toc158151296)

[Tbl\_users 18](#_Toc158151297)

[Tbl\_order 19](#_Toc158151298)

[Tbl\_productimg 19](#_Toc158151299)

[tbl\_orderproducts 20](#_Toc158151300)

[Entity Relationship Diagram 21](#_Toc158151301)

[Testing log & Testing plan 22](#_Toc158151302)

# Introduction

Grimsby and Clee sells (GACS) has request a web app that allows user to sell products, as well as buy products from other users. GACS has requested the use of admin account in this system to verify user’s products as well as so users can request support.

# Requirements list

A set of requirements lists have been developed to group some of the clients and nonfunctional requirements into different priorities, of what is most important and least important when developing the solution.

## Non-functional requirements list

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement number | Requirement name | Description | Priority |
| 1 | Usability | The user will be able to use the website with ease and the user experience and interaction should be to the highest standard. | High |
| 2 | Functionality | The solution should work as intended with no unplanned errors. | High |
| 3 | Accessibility | The project should have features (eg, overlays and text settings) to allow for users who have accessibility needs to use the website with ease. | Medium |
| 4 | Reliability | The project must have an up time of 99.99% | High |
| 5 | Security | The project must be secure all data such as uses data, by encrypting all sensitive user data eg. Passwords and banking information. | High |
| 6 | Scalability & compatibility | The project must scale to different screen sizes depending on the screen size or device. It should work on all devices. | Medium |

The priority system is a follows:

**High** – The up most importance

**Medium** – May be implanted or developed depending on the solutions development.

**Low** – isn’t important and most likely would be developed.

All requirements other than requirements three and six have been given a high priority, as all other requirements all necessary to allow all uses/customer to be able to use the client's system. Requirements three and six impact a small number of uses and do not impact the majority of the client's target audience, therefore these requirements can be given a lower priority. Features relating to the requirements three and six can be implemented late in an update after the project has been released, that way requirements with more significance can be focused on while the project is in its early sates of development.

Requirements one, two, four and five have been given a high priority as they effect the overall ability to use the project or effect a user's/ customer's security while using the project. If the client was to release the project while not considering these requirements, users would have a bad experience when using the project or the client may be in break with general data protection regulations or data protection act 2018.

Requirement four will be partly the client’s respectability to maintain once the project has been released to the public, the requirement consists of the project being useable majority of the time as well as the project being reliable. The web developer can ensure that the project reliable by using good standards of code, as well as thoroughly testing the code to ensure uses will not encounter any unplanned errors. In addition to the web developer using a good standard of code, the client will have to host the project in such a way that it remains reliable. The client would have to buy a reliable server to host the project or source an external company to host the project.

## Functional requirements list

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement number | Requirement name | Description | Priority |
| 1 | Sign-up | A user should be able to make an account to use the project. | High |
| 2 | Login | A user should be able to login to their account to use the project. | High |
| 3 | Home page | The project should have a home page that is displayed when they login-in or sign up. The page will show the most commonly should products. | High |
| 4 | Product pages | The project should be able to show products through a variety of pages. | High |
| 5 | Product Search | The project will have the ability to search for certain products. | High |
| 6 | Edit account | The project should allow a user to edit their details they created their account in. | High |
| 7 | Placing products for sale | A user will be able to place an item for sale. | High |
| 8 | Admin accounts | The project will have administrator accounts to monitor the usage of the project. | High |
| 9 | Administration viewing all your items for sale. | Administrator accounts should be able to view items that they have put for sale | High |
| 10 | Administration views all items that are for sale. | Administrator accounts should be able to view all items that are for sale. | High |
| 11 | Approved system | All Items need to be approved by an administrator account before they can be post for sale. | High |
| 12 | Administrator to remove items | An administrator account can remove any item for sale. | High |
| 13 | Administrator to edit items, | An administrator account can edit any item that is for sale. | High |
| 14 | Administrators view all users accounts. | An administrator account should be able to view anyone’s accounts | High |
| 15 | Administrators to delete a user’s account. | An administrator account should be able to delete a person’s account. | Medium |
| 16 | Administrators to edit a user’s account. | An administrator account should be able to edit a person’s account. | Medium |
| 17 | Upload image with item. | A user should be able to upload an image with an item | High |
| 18 | Rating system | A user will be able to give a product a rating | Medium |
| 19 | Average rating | A product will have an average based on the ratings uses gave that product. | Medium |
| 20 | Descriptive reviews | A user will be able to leave descriptive reviews on a product. | Medium |
| 21 | Messaging between users | Users may be able to message each other via the project | Low |
| 22 | Auction system (sell) | A user will be able to put a product up for auction. | Low |
| 23 | Auction system (buy) | A user should be able to take part in a auction. | Low |
| 24 | Basket system | A user should have access to a basket system, that stores the items they would like to buy. | High |
| 25 | Watch items system | A system where a user can watch an item and get notification when an item has changes made to it. | Medium |
| 26 | Stop sale of product | A user should be able to stop the sale of their products. | High |
| 27 | Delete account | A user will be able to delete their account. | High |
| 28 | Accessibility page | A user should have access to an accessibility page that allows them to turn on accessibility settings | Low |
| 29 | Cookies | The project will use to store the users preferences. | Low |
| 30 | User to view all their items for sale | The project will allow users to see all their products for sale. | High |
| 31 | View items that have been sold | The user will be able to see their products that have been sold | High |
| 32 | User to see their products that have yet been approved | The user will be able to see all their items that have not been approved. | High |
| 33 | Edit the users product | The user will be able to change the products they are selling | High |
| 34 | Delete a users product | The user will be able to delete their products when they are up for sale. | High |
| 35 | Creating admin account | An account for admins allow allows admins to do administrative tasks should be able to be made. | High |
| 36 | Admin Login | An admin should be able to login to an admin account to do administrative tasks. | High |
| 37 | Buy Product | A user should be able to buy products | High |
| 38 | Order system | After buying a product a user should be able to see a history of orders. | High |

The priority system is a follows:

**High** – The up most importance

**Medium** – May be implanted or developed depending on the solutions development.

**Low** – isn’t important and most likely would be developed.

The functional requirements have been organised so features that the project will need to be usable are a high priority, this includes requirements like being able to sign-up, login, buy products and sell products. Requirements that would also greatly improve the user experience and user interaction have also give a high priority, an example of some of these requirements are being able to stop the sale of a product, approved system and the ability to upload images with a product. All the high priority requirements will be completed by the end of development of the project.

Medium requirements have been organised to features that won’t have a huge impact on user experience and wont impact the “main system”. Some medium requirements are watching items system, messaging between users, average rating and administrators to edit a user’s account. Some of these requirements may not be met at the end of development, however they could come in updates later in the solutions lifetime.

Finally, the low requirements consist of requirements that won’t make great difference to the user experience or are not reasonable when considering time-constraints as they may take a while to develop. Some of the requirements that have been given a low priority due to time constraints are messaging between users, auction system (sell) and auction system (buy). In addition to the following requirements, there are other requirements that have been placed in the low priority such as accessibility page and Cookies integration, these requirements are low because they wouldn’t make a huge impact on the project if they weren’t developed.

# Design choices

## Minimalism & white space

The UI of the is to be minimalistic, meaning elements will be spaced and designed in away where the UI doesn’t show off too much reading heavy content and doesn’t look to cramped with content. To achieve a minimalistic the UI will use white in combination with Fitts's Design Law to achieve an ease to navigate website. Fitts Design Law consists of the time it takes a user to find the elements they are looking for, so the project will white space to spread all the element in the UI in away that is readable and understand to improve the time it takes for a user to find what they are looking for. Considering both points will improve the user experience and user interaction.

## Icons

The projects UI will use Icons on elements such as links and buttons to indicate what they do, developing icons will also aid Fitt’s Design Law as users tend to view icons before reading the text beside them meaning they will find elements quicker granted the icons used make regarding the element they are used for. Icons will improve the ability for a user to navigate a website and improve a user experience.

## Error handling & feedback

The application programming interface (API) will have validation to ensure that a user enters the correct details/ information, if they did not send the current information the API then then send a pop to UI to tell the user what went wrong. The pops will have useful feedback to tell users why it went wrong (eg, a missing letter or number) to ensure the user will understand.

Along with API validation the project will include JavaScript validation that will tell the user if there is a error without sending data to the API, this validation will also have feedback to the user, however will not be displayed through an pop. JavaScript validation and feedback will appear primarily under the users inputs.

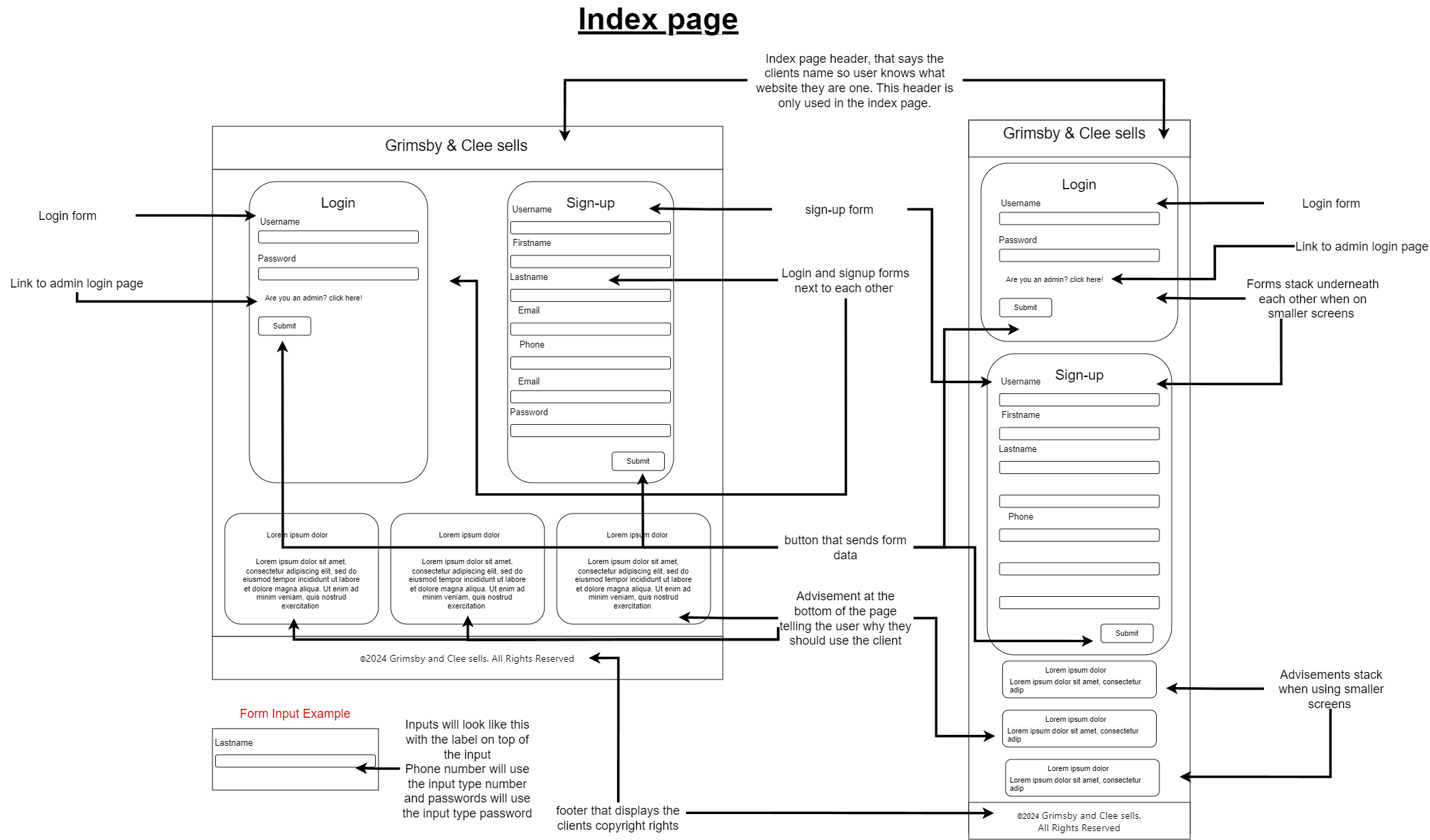
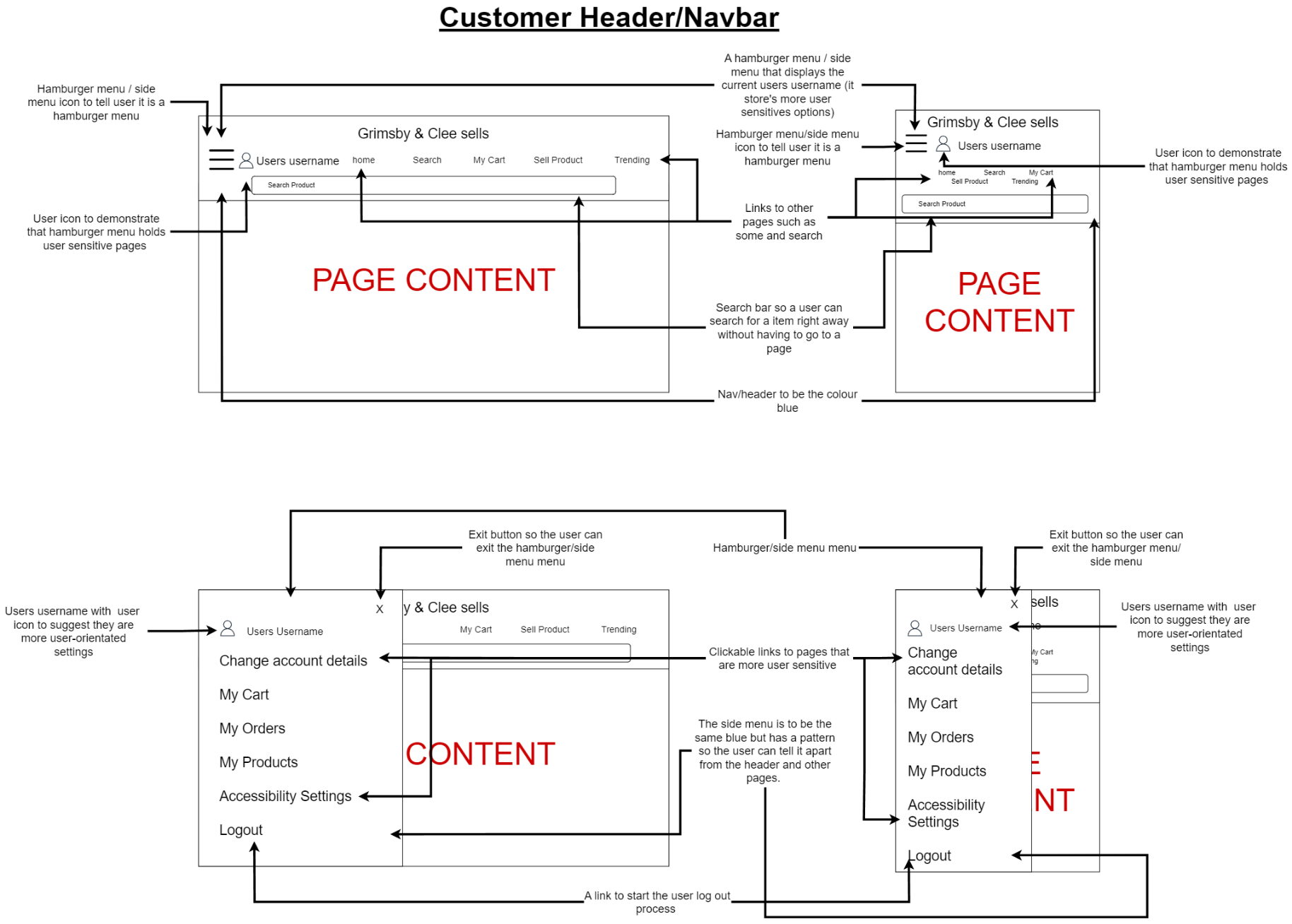
## Security

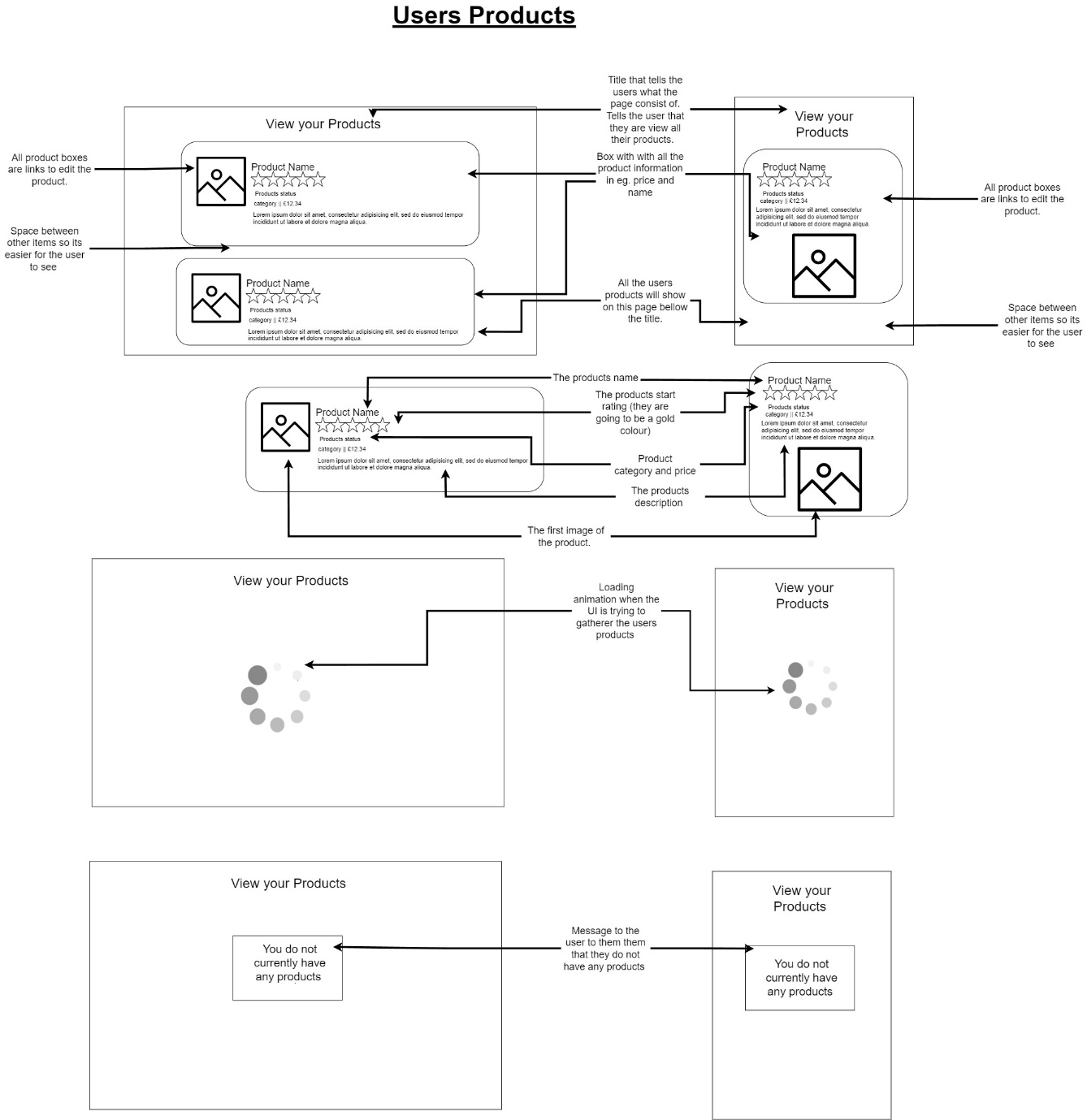
The project will be using the framework ASP.NET to create an API. Using this approach means unlike standard PHP, the user wouldn’t send the data from their machine straight to the database. Instead the data will be processed to the API then, it can validate the data then the API can process data to the database. Making an API send data to the database rather than users machine means its unlikely the data can be tampered with. If the API detects something wrong with data, then it will reject data sent by the user and send an error to the UI.

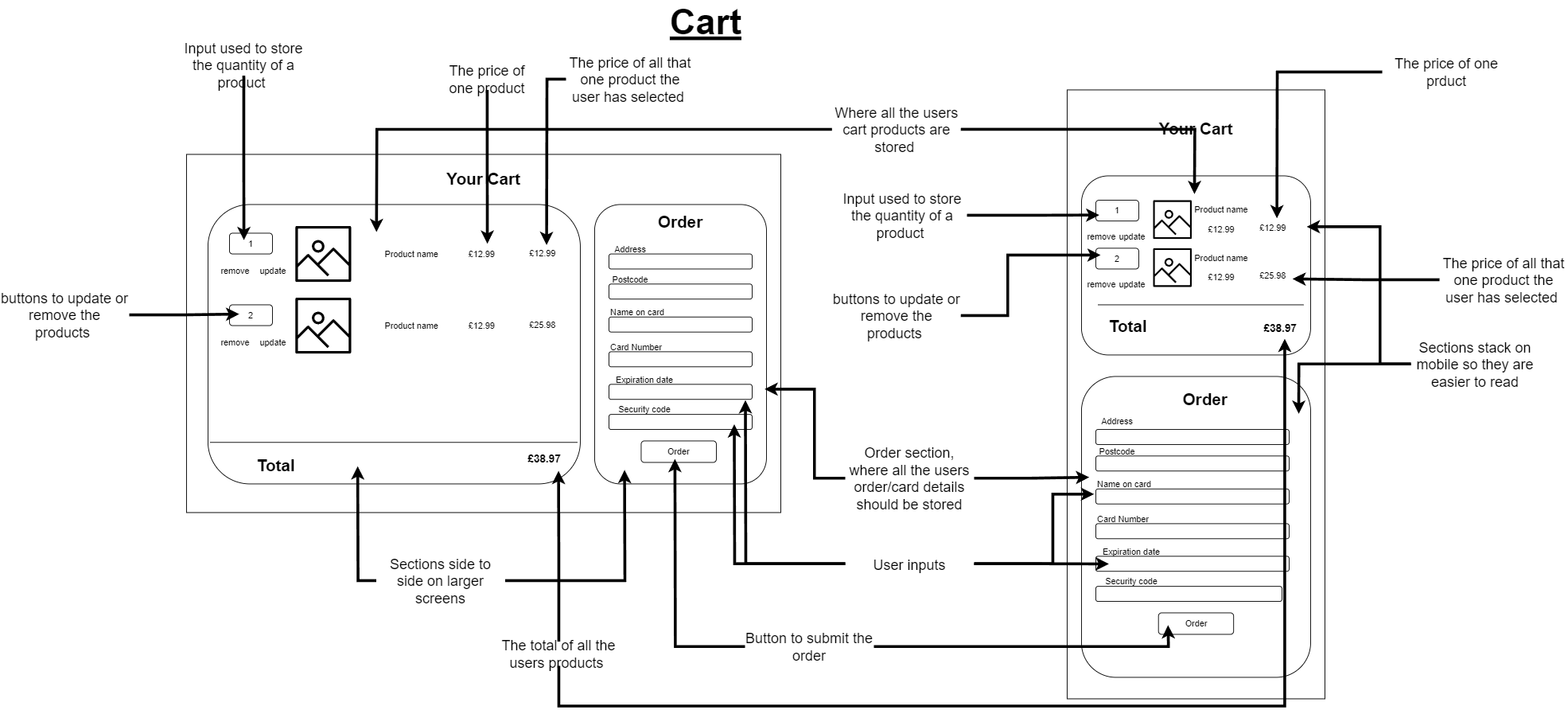
The API will not get the data via SQL queries, this means that the API will not be susceptible to SQL Injection attacks. The API will use entity framework core this means prevents SQL injection due to the way the framework is structured as one repository and controller can execute one SQL query.

# Wire frames

## Lo-Fi







## A screenshot of a web page Description automatically generatedHi-Fi

A screenshot of a cart page

Description automatically generated

# Web structure

# API structure

A computer screen shot of a computer diagram

Description automatically generated

# Data dictionary

## Tbl\_product

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| product\_id | INT | N/A | PK | NO | The unique identifier for the product. |  |
| product\_name | VARCHAR | 50 | N/A | NO | The name of the product. |  |
| product\_description | VARCHAR | 200 | N/A | NO | A description of the product. |  |
| product\_category | INT | N/A | FK | NO | The category the product falls under. | The name of category falls under the tbl\_category table as this table uses a foreign key so the data can link. |
| product\_userid | INT | N/A | FK | NO | The ID of the who has posted the product. |  |
| product\_status | INT | N/A | FK | NO | The waiting to be approved status for product. |  |
| product\_price | FLOAT | 7 | N/A | NO | The price of the product. |  |

## Tbl\_users

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| users\_id | INT | N/A | PK | NO | The unique identifier for the user. |  |
| user\_username | VARCHAR | 25 | N/A | NO | The users unique username. |  |
| users\_firstname | VARCHAR | 25 | N/A | NO | The users first name. |  |
| users\_lastname | VARCHAR | 25 | N/A | NO | The user last name. |  |
| users\_email | VARCHAR | 50 | N/A | NO | The users email. |  |
| users\_phone | VARCHAR | 11 | N/A | NO | The users phone number. | The users phone number does not use CHAR as it is not compatible with ASP.NET so validation will be made in the API. |
| users\_dob | DATE | N/A | N/A | NO | The users date of birth. |  |
| users\_password | VARCHAR | 255 | N/A | NO | The users password. | The users password will have 255 characters of length so it helps with the encryption process. |

## Tbl\_order

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| Order\_id | INT | N/A | PK | NO | The unique identifier for the order. |  |
| Order\_userid | INT | N/A | FK |  | The id of the user who ordered the order. | The name of user falls under the tbl\_users table as this table uses a foreign key so the data can link. |
| Order\_address | VARCHAR | 100 | N/A |  | The address of where the order is going to be delivered. |  |
| Order\_date | DATE | N/A | N/A |  | The date the order was placed. |  |
| Order\_recipientname | VARCHAR | 70 | N/A |  | The name of who order the order. |  |
| Order\_status | INT | N/A | FK |  | The status of the order. | The name of status falls under the tbl\_status table as this table uses a foreign key so the data can link. |

## Tbl\_productimg

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| productimg\_id | INT | N/A | PK | NO | The unique identifier for the image. |  |
| productimg\_productid | INT | N/A | FK | NO | The product id of product the images are for. | The name of product falls under the tbl\_product table as this table uses a foreign key so the data can link. |
| productimg\_img | IMAGE | N/A | N/A | NO | The image of the product. |  |
| productimg\_thumbnail | IMAGE | N/A | N/A | NO | A lower quality image of the main image. | A lower quality image is made in the database so when loading the them in the UI its much faster loading. |

## tbl\_orderproducts

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| orderproducts\_productid | INT | N/A | PK FK | NO | The ID of the product that is in the order. | Is used with a composite key to fetch all products for an order. |
| orderproducts\_orderid | INT | N/A | PK FK | NO | The ID of the order that is in the order. | Is used with a composite key to fetch all products for an order. |
| orderproducts\_quantity | INT | 2 | N/A | NO | Stores the quantity of how many of a product has been ordered. |  |

# A computer screen shot of a computer Description automatically generatedEntity Relationship Diagram

# Testing log & Testing plan