**Server-Side Development Design Document**

Student number – 570421

Submission date – 9th February 2024

Table of Contents

[Introduction 3](#_Toc158337404)

[Requirements list 4](#_Toc158337405)

[Non-functional requirements list 4](#_Toc158337406)

[Functional requirements list 5](#_Toc158337407)

[Design choices 7](#_Toc158337408)

[Minimalism & white space 7](#_Toc158337409)

[Icons 8](#_Toc158337410)

[Error handling & feedback 8](#_Toc158337411)

[Security 8](#_Toc158337412)

[Wire frames 8](#_Toc158337413)

[Lo-Fi 9](#_Toc158337414)

[Hi-Fi 15](#_Toc158337415)

[Web structure 20](#_Toc158337416)

[API structure 21](#_Toc158337417)

[Data dictionary 22](#_Toc158337418)

[Tbl\_product 22](#_Toc158337419)

[Tbl\_users 23](#_Toc158337420)

[Tbl\_order 24](#_Toc158337421)

[Tbl\_productimg 24](#_Toc158337422)

[tbl\_orderproducts 25](#_Toc158337423)

[Tbl\_status 25](#_Toc158337424)

[Tbl\_orderstatus 26](#_Toc158337425)

[Tbl\_Category 26](#_Toc158337426)

[Tbl\_Cart 26](#_Toc158337427)

[Tbl\_Admin 26](#_Toc158337428)

[Entity Relationship Diagram 28](#_Toc158337429)

[Testing log 29](#_Toc158337430)

# Introduction

Grimsby and Clee Sell (GACS) have requested a web app allowing users to sell products and buy products from others. GACS has asked for the use of an admin account in this system to verify user’s products so users can request support.

# Requirements list

A set of requirements lists has been developed to group some clients' 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.

A set of requirements lists has been developed to group some clients' nonfunctional requirements into the most important and least important priorities when developing the solution.

All requirements other than requirements three and six have been given a high priority, as all other requirements are necessary to allow all users/customers 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 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 stages of development.

Requirements one, two, four and five have been given a high priority as they affect the overall ability to use the project or affect a user's/customer's security while using the project. If the client were 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 the 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 is reliable by using good standards of code, as well as thoroughly testing the code to ensure users will avoid 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 signing up, logging in, buying, and selling products. Requirements that would also significantly improve the user experience and user interaction have also been given a high priority; Examples of some of these requirements are being able to stop the sale of a product, having an approved system and the ability to upload images with a product. All the high-priority requirements will be completed by the end of the projects development.

Medium requirements have been organised to features that won't have a huge impact on user experience and won't 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 still need to be met at the end of development. However, they could come in updates later in the solution's lifetime.

Finally, the low requirements consist of requirements that won't make a significant 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, other requirements have been placed in low priority, such as the accessibility page and Cookies integration; these requirements are low because they wouldn't significantly impact the project if they weren't developed.

# Design choices

## Minimalism & white space

The UI is to be minimalistic, meaning elements will be spaced and designed in a way where the UI does not show off too much reading-heavy content and does not look too 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 elements in the UI in a way that is readable and understandable 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 project 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 of a user to navigate a website and improve the 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 will send a pop to the UI to tell the user what went wrong. The pops will have helpful 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 an error without sending data to the API; this validation will also have feedback to the user; however, it will not be displayed through a pop. JavaScript validation and feedback will appear primarily under the user's inputs.

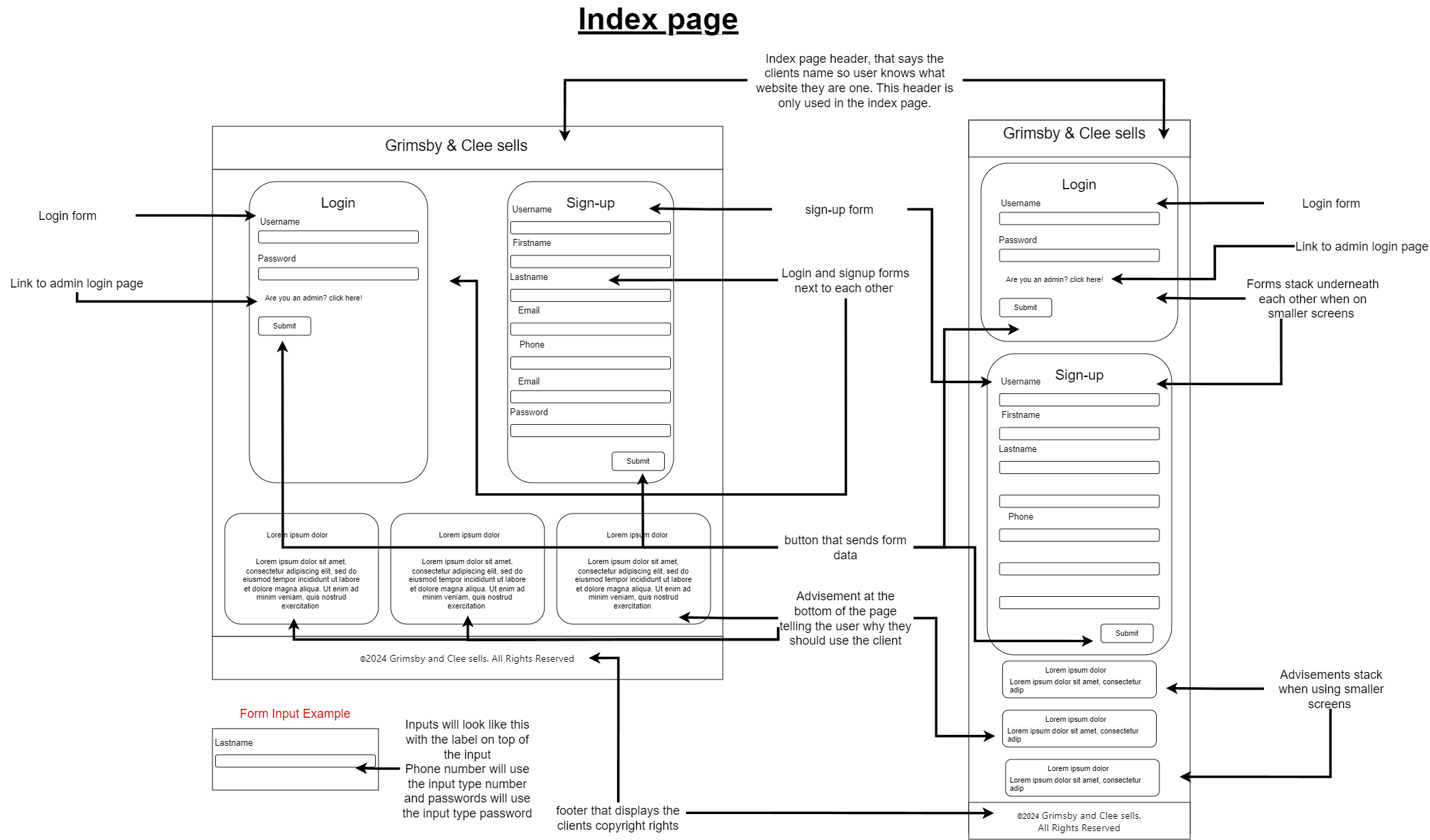
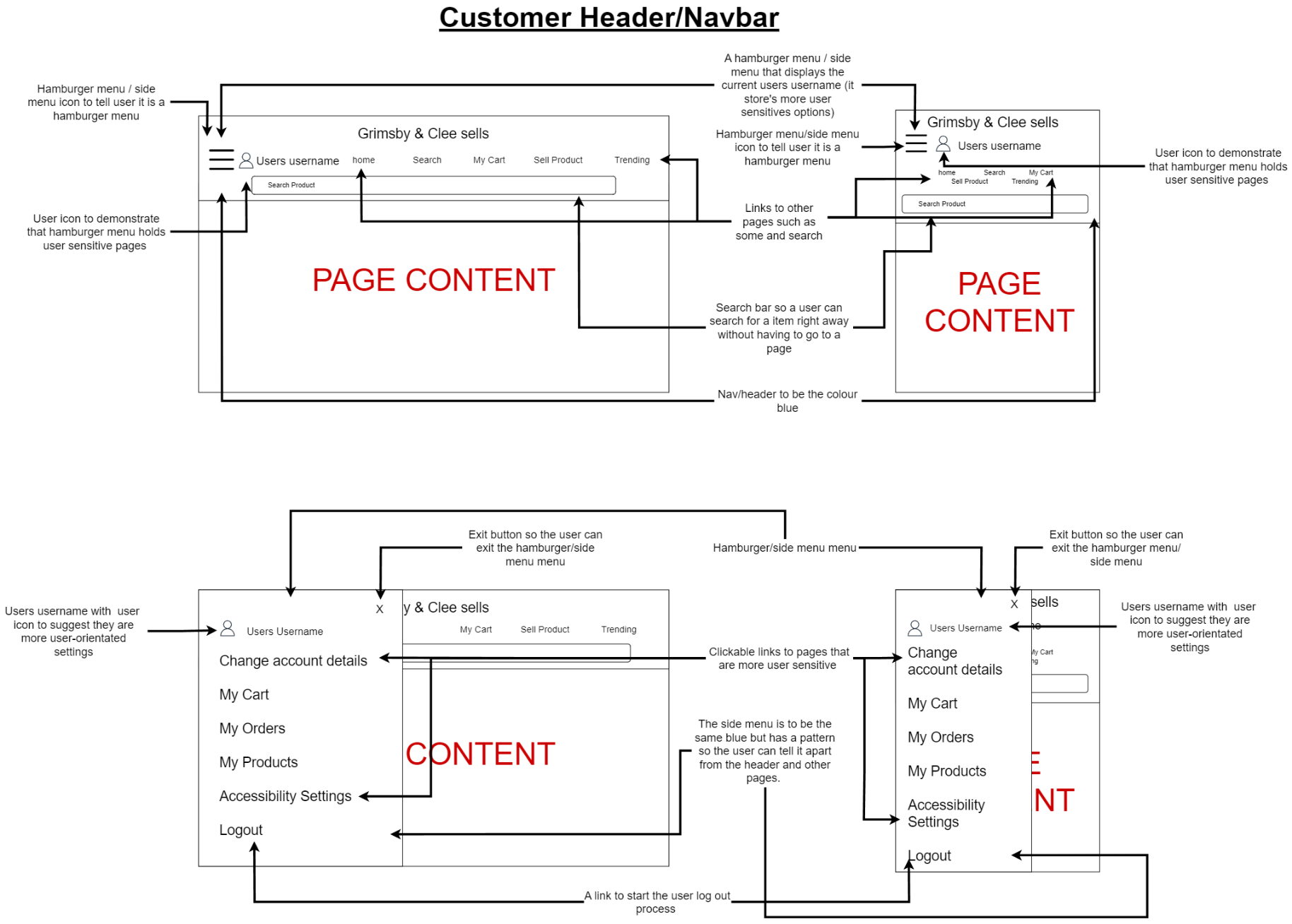
## Security

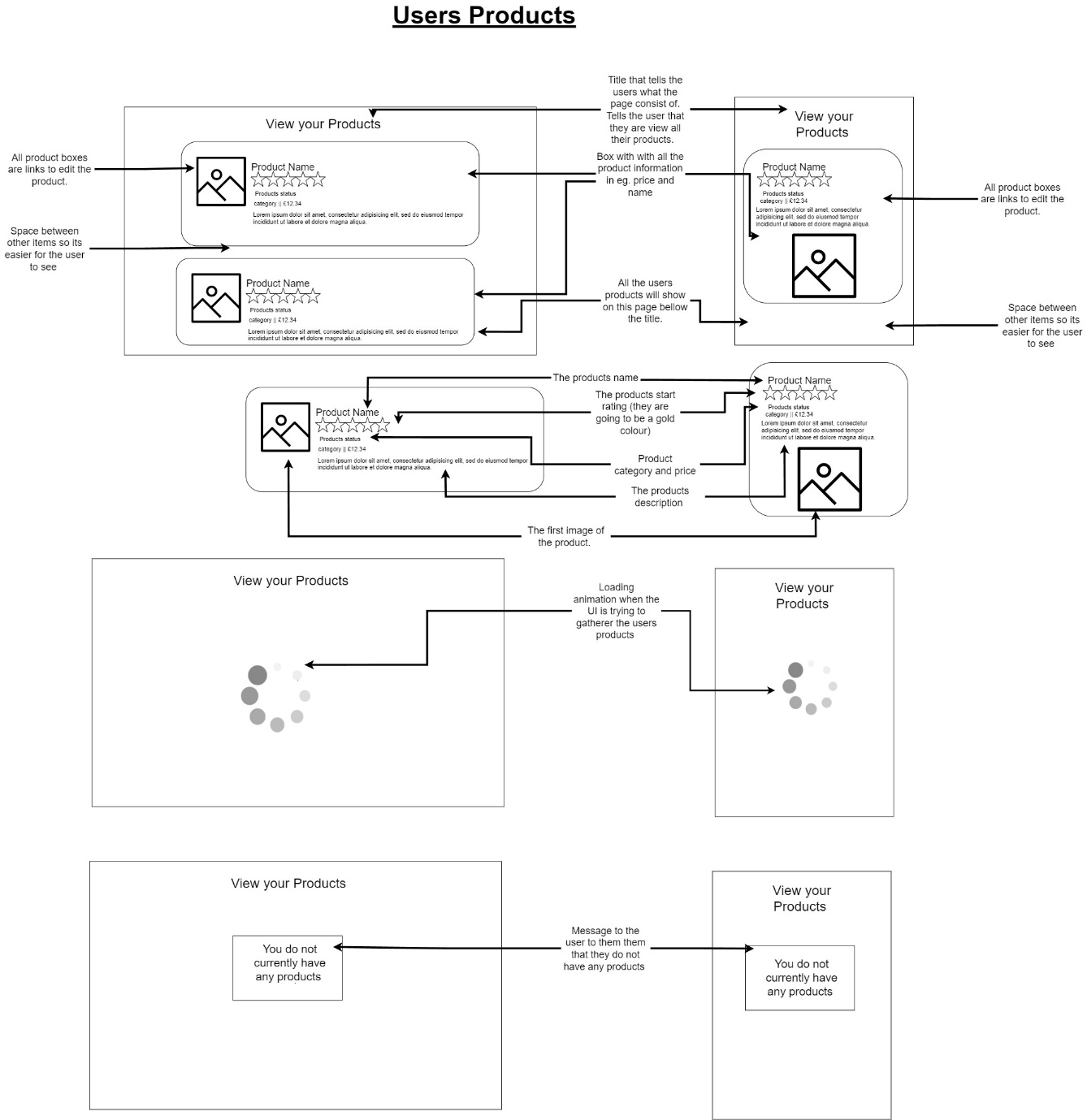
The project will use the ASP.NET framework to create an API. Using this approach means, unlike standard PHP, the user wouldn't send the data from their machine straight to the database. Instead, the data will be processed by the API, which can validate the data, and then the API can process data to the database. Making an API send data to the database rather than the user's machine means it's 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.

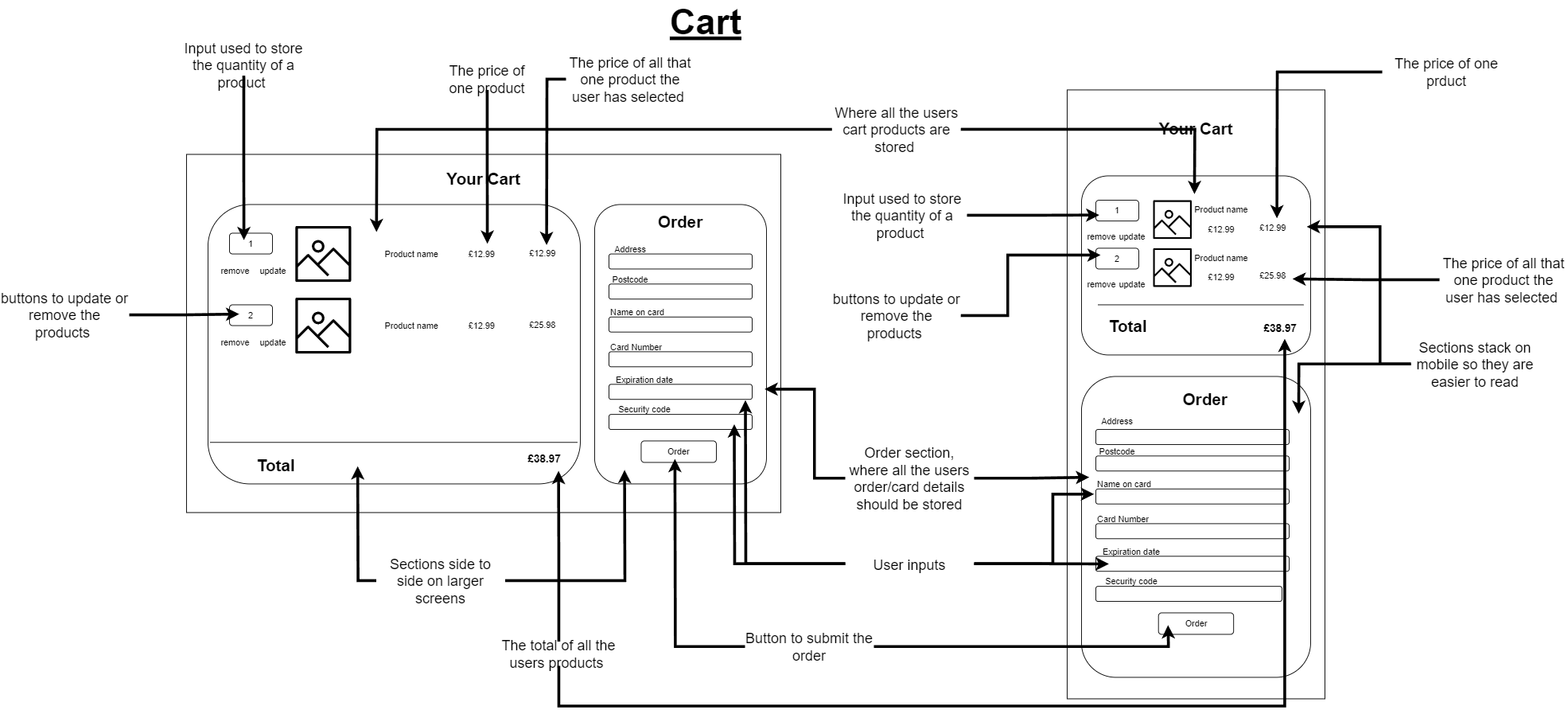
The API will not get the data via SQL queries, which means that the API will not be susceptible to SQL injection attacks. The API will use entity framework core this means prevents SQL injection due to the way the framework is structured as one repository and controller can execute one SQL query.

# Wire frames

## Lo-Fi







A diagram of a diagram

Description automatically generatedA diagram of a website

Description automatically generated

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

A screenshot of a cart page

Description automatically generated

A diagram of a website

Description automatically generated

A screenshot of a website

Description automatically generated

A screenshot of a login page

Description automatically generated

# Web structure

A diagram of a company

Description automatically generatedThe solution will use jQuery. jQuery allows a webpage to use other HTML and add them to the webpage. Doing this improves the efficiency of the html as things such as loading animations and pop ups don’t have to be coded on each page. All that is done instead id a div tag is made with an unique id to tell the java script to put the html inside in. As shown from the diagram there are pages that customer can go on and some that admins can go on, and two pages that everyone can access. Some pages have access to java script files to retrieve data from the API or do other features, but not all are required for each page. Lastly all pages will have access to the styling sheet, that will also link assets such as fonts to the pages as well.

# A computer screen shot of a computer diagram Description automatically generatedAPI structure

The solution will use ASP.Net as it’s a very secure, as its not susceptible to SQL injection. The security is due to its file structure as it uses multiple files to execute a request. The API will act as a Application layer also, this stops users from having a direct connection to the database, unlike PHP where the user needs to run the process from the their computer. ASP.net typically run a fetch from the controller file then, interface repository the SQL repository file. The SQL Repository file then goes to, the database file witch then connects to the database.

# 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. | Auto Incremented |
| 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 | N/A | 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. | Auto Incremented |
| user\_username | VARCHAR | 25 | N/A | NO | The user’s unique username. |  |
| users\_firstname | VARCHAR | 25 | N/A | NO | The users first name. |  |
| users\_lastname | VARCHAR | 25 | N/A | NO | The user’s 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 and SQL Server 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 user’s password. | The user’s password will have 255 characters of length, so it helps with the encryption process. |
| users\_balance | FLOAT | N/A | N/A | NO | The user’s balance | Float uses a predefined length in SQL Server and each user’s balance starts at 10000 |

## 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. | Auto Incremented |
| Order\_userid | INT | N/A | FK | NO | 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 | NO | The address of where the order is going to be delivered. |  |
| Order\_date | DATE | N/A | N/A | NO | The date the order was placed. |  |
| Order\_recipientname | VARCHAR | 70 | N/A | NO | The name of who order the order. |  |
| Order\_statusid | INT | N/A | FK | NO | 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. |
| Order\_detail1 | VARCHAR | 255 | N/A | NO | The user’s card number | The column is named Order\_detail1 to aid in the security of the bank details |
| Order\_detail2 | VARCHAR | 255 | N/A | NO | The user’s expiry date on their bank card | The column is named Order\_detail2 to aid in the security of the bank details |
| Order\_detail3 | VARCHAR | 255 | N/A | NO | The user’s security number for their bank card (CVV) | The column is named Order\_detail3 to aid in the security of the bank details |

## 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. | Auto Incremented |
| 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 unique 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 unique 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. |  |

## Tbl\_status

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| status\_id | INT | N/A | PK | NO | The unique ID of the approval stat us | Auto Incremented |
| status\_name | VARCHAR | 20 | N/A | NO | The name of the approval status |  |

## Tbl\_orderstatus

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| orderstatus\_id | INT | N/A | PK | NO | The unique id of the order status | Auto Incremented |
| orderstatus\_status | VARCHAR | 20 | N/A | NO | The name of the order status |  |

## Tbl\_Category

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| category\_id | INT | N/A | PK | NO | The unique id of the category | Auto Incremented |
| category\_name | VARCHAR | 50 | N/A | NO | The name of the category |  |

## Tbl\_Cart

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute name | Data type | Length | Key | Nullable | Description | Additional details |
| cart\_userid | INT | N/A | PK FK | NO | The unique identifier for the user | Used as a composite key to let a user add a product to their cart |
| cart\_productid | INT | N/A | PK FK | NO | The unique identifier for the product the user has chosen | Used as a composite key to let a user add a product to their cart |
| cart\_quantity | INT | N/A | N/A | NO | The quantity of the chosen product |  |

## Tbl\_Admin

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

# Entity Relationship Diagram

A computer screen shot of a diagram

Description automatically generated

# Testing log

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test name | Data | Expected outcome | Outcome | Success? | Actions |
| 1 | API connection |  | API should connect to the UI and Database. | The API connects to the UI and Database. | Yes | N/A |
| 2 | Swagger create customer. | Username firstname lastname  [Email@email.com](mailto:Email@email.com)  07671621512  01/01/2000  Apassword@1 | The API should make an account. | The API makes a user account. | Yes | N/A |
| 3 | Swagger validate the an customer account. | Username  Apassword@1 | The API should validate the user details. | The API validates an existing users details. | Yes | N/A |
| 4 | Swagger create a users product | Red Toy Car,  This is a red toy car, it has four wheels and lights, 4,  9,  1,  5.50,  4, | The API should create a product. | The API makes a product and stores it in the database. | Yes | N/A |
| 5 | UI create account. | Username1 firstname lastname  [Email@email.com](mailto:Email@email.com)  07671621512  01/01/2000  Apassword@1 | The UI should send a create account request to the UI then take them to the home page. | Does not send an a request but instead sends error code 400. | No | Removed user\_id from the domain user model and DTOs. |
| 5.1 | UI create account. | Username1 firstname lastname  [Email@email.com](mailto:Email@email.com)  07671621512  01/01/2000  Apassword@1 | The UI should send a create account request to the UI then take them to the home page. | The sends a create account request to the UI then take them to the home page. | Yes | N/A |
| 6 | UI login | Username1  Apassword@1 | The UI should send a login request, then get the data then use the data to go to the home page. | The UI sends a login request, then get the data then use the data to go to the home page. | Yes | N/A |
| 7 | UI create product | Red Toy Car,  This is a red toy car, it has four wheels and lights, 4,  9,  1,  5.50,  4, | The UI should send create product, then take them to a success page. | The UI sends create product, then take them to a success page. | Yes | N/A |
| 8 | Create product name validation API | Red | The API should send an error say that name is to small | The API sends an error say that name is to small | Yes | N/A |
| 9 | Create product description validation API | aaaa | The API should send an error say that description is to small. | The API sends an error say that description is to small. | Yes | N/A |
| 10 | UI validation for create product | WRONG Inputs for each field. | The UI should show that the inputs a not correct, as well as showing messages that tell the user why they are not correct. | The UI shows that the inputs a not correct, as well as messages that tell the user why they are not correct. | Yes | N/A |
| 11 | UI validation for create user | WRONG Inputs for each field. | The UI should show that the inputs a not correct, as well as showing messages that tell the user why they are not correct. | The UI should shows that the inputs a not correct, as well as showing messages that tell the user why they are not correct. | Yes |  |
| 12 | Add product to cart |  | When adding product to cart using the UI a pop up should be shown and it should be added to the cart table. | Adding product to cart will enable a pop up and it should be added to the cart table. | Yes |  |
| 13 | Admin login | Testing1  Apassword@1 | The user should be able to login, then be directed to the admin home page. | The user is logged in and directed to the admin home page. | Yes |  |
| 14 | Create Admin | Testing22  Apassword@1 | A admin should be able to create an admin and then login with that account. | A admin can create an admin and then login with that account. | Yes |  |
| 15 | Search product | Panda | A product should appear when the user searches for its name. | A product appears when the user searches for its name. | Yes |  |
| 16 | Show top products (home page) |  | A list of products sorted by how many sold should be shown on the home page. | Too many products was show and it was unreadable/ not user friendly. | No | In the API added the function .take(5) so the user would only see 5 |
| 16.1 | Show top products (home page) |  | A list of products sorted by how many sold should be shown on the home page. | A list of products sorted by how many sold is shown and is readable. | Yes |  |
| 17 | Admins approve products |  | An admin should be able to approve a product waiting to be approved. | Am admin can approve a product waiting to be approved. | Yes |  |
| 18 | Admins to reject products |  | An Admin should be able to reject an product. | An Admin can reject an product | Yes |  |
| 19 | Rejected products do not show up for base users (other than showing a user their own products) |  | Rejects products should not shown for base users. | Rejected products do not show for base users. | Yes |  |
| 20 | Products awaiting review don’t show for base users (other than showing a user their own products) |  | Rejected products should not shown for base users. | Rejected products are not shown for base users. | Yes |  |
| 21 | Admin show user products |  | System must show the a users products to admins. | System shows users products to admins. | Yes |  |
| 22 | Admin show users’ products with users that have no products |  | The system must display a pop up that tells the admin that the user doesn’t have any products. | The system shows the popup however the page is distorted as the user data has not been found. | No | Made a new fetch In the JavaScript file that gets this information. The fetch will be there just in case the system cannot get any products. |
| 22.1 | Admin show users’ products with users that have no products |  | The system must display a pop up that tells the admin that the user doesn’t have any products. | The system shows display a pop up that tells the admin that the user doesn’t have any products, and the page is not distorted due to lack of user data. | Yes |  |
| 23 | Create admin validation | WRONG Inputs for each field. | When an admin creates a new admin if wrong inputs are inputted an error message should show up | When an admin creates a new admin if wrong inputs are inputted an error message shows up | Yes |  |
| 24 | Create admin validation API | WRONG Inputs for each field. | When wrong data is entered the api sends an error pop to the UI | When wrong data is entered the api sends an error pop to the UI | Yes |  |
| 25 | Index page scalability |  | Index page should be suitable for mobile screens | Index page are suitable for mobile screens | Yes |  |
| 26 | Adminlogin page scalability |  | Adminlogin page should be suitable for mobile screens | Adminlogin page are suitable for mobile | Yes |  |
| 27 | User headers scalability |  | All user headers should be scale for mobile. | All user headers scale for mobile. | Yes |  |
| 28 | Edit account | Baseuser1 | The page should reload then show the new details as well as changing them in the database. | Changes are made on the database but are not made in the UI | No | Add set session function to the JavaScript file that deals with changing the details/ |
| 28.1 | Edit account | Baseuser1 | The page should reload then show the new details as well as changing them in the database. | Changes are made on the database but are not made in the UI | No | Add the generate jwt to the API method that edits the accounts. |
| 28.2 | Edit account | Baseuser1 | The page should reload then show the new details as well as changing them in the database. | All edits are made when the user changes the details. | Yes |  |