****

Agile Products LTD

Group A1

**Declan Monaghan – P2432774**

**Nikhil Jagatia – P2422743**

**Daniel O’Hara – P2435725**

**Mishal Nanalal – P2434110**

**Farhaan Mohammad – P2432609**

**Table of contents:**

* **Page 2: Intro to the system**
* **Page 3: Logo**
* **Page 4,5: System ERD**
* **Page 6-10: Customer specification (Declan)**
* **Page 11-17: Staff specification (Nikhil)**
* **Page 18-21: Hardware specification (Daniel)**
* **Page 22-26: Order specification (Mishal)**
* **Page 27-32: Software specification (Farhaan)**

****

**Agile Products LTD**

“Our system will sell pc hardware and software”.

**Description/overview:**

Our company will sell all ranges of hardware and software to the general public, as well as the potentially bigger clients such as small companies. Through a web based front end, clients will be able to create an account of they wish and browse through all of our stock, selecting what they want and adding it to their basket and proceed with potentially buying said items. Also, at the front end we will have a FAQ section and provide a helpline, in case they need to discuss their order with us, or speak further about our products.

We will also have a staff facing back end. Where we will store information and it will containing customer details, order details, product details and staff details which we will gain once and order is completed and payment has been received, a conformation email will be sent to the client. But also, from here we can manage returns. The back-end database which will be used to store many things.

**Roles:**

staff– Nikhil

* Designing and building of the staff table in the database and implementation of staff details on the front end also. Allowing adding, editing, deleting and filtering of staff records/details.

Order – Mishal

* Designing and building of the order table and implementation of the ordering form on the front end. Allowing, adding, editing, deleting and filtering of orders and their respective details.

Customer, payment – Declan

* Designing and building the customer and payment table of the database. Allowing, adding, editing, deleting and filtering of customer records linking this to their payment details.

Products software – Farhaan

* Designing and building of the products table to provide levels of stock and allow orders to take place. Also, potential front-end presentation of products. Allowing, adding, editing, deleting and filtering of products and their respective details.

Products Hardware – Daniel

* Implementation/building of hardware tables to show levels of hardware stock and to maintain levels. Also, potential front-end presentation of products. While also allowing adding, editing, deleting and filtering of products and their respective details.

**Logo:**

****

Our logo is very attractive to the eye and simple which is very useful for business purposes. This is because it stands out and grabs a potential client’s attention and is simple and colourful, reflection how an agile system should be.

**Group ERD**

1…

1

Made by

Software

Hardware

Supplier

Made by

1

1

1…

1…

Contains

Contains

1

1

OrderLine

0…

0…

Has

1

Order

Customer

Staff

Makes

Handles

0…

1

0…

Review

Leaves

1

0…

Key:

0… = 0,1 or many

1… = 1 or many



clsStaff

StaffID: Int

Department: String

Find(): void

Validate(): void

StaffFirstnameAndSurname:

String

DateOfBirth: Date

StreetAddress: String

PermissionToChangeFields:

Boolean



clsHardware

Find(): void

Validate(): void

HardwareID : Int

HardwareName : String

HardwareDescription : String

Price : Decimal

DateOfArrival: Date

InStock : Boolean

QuantityInStock: Int



clsSoftware

Find(): void

Validate(): void

SoftwareID : Int

SoftwareName : String

SoftwareDescription : String

Price : Decimal

DateOfArrival: Date

InStock : Boolean

QuantityInStock: Int



clsCustomer

CustomerID : Int

CustomerFirstnameAndSurna

me : String

Member: Boolean

DateOfBirth : Date

Address: String

Email : String

CardNumber : Int

Sortcode : String

Find(): void

Validate(): void



clsOrderLine

HardwareID: Int

SoftwareID: Int

Find(): void

Validate(): void



clsOrderLineCollection

Count: Int

OrderLines: List<OrderLine>

ThisOrderLine: Int

Add(): void

Delete(): void

Edit(): void

Filter(): void



clsOrder

OrderID: Int

Find(): void

OrderDate: DateTime

Shipped: Boolean

ShippingVia: String

OrderLines:

clsOrderLineCollection

Validate(): void



clsOrderCollection

CustomerID: Int

Add(): void

Count: Int

OrderList: List<clsOrder>

ThisOrder: clsOrder

Delete(): void

Edit(): void

Filter(): void

OrderLineID: Int

OrderLineID: Int

OrderLineCollectionID: Int

OrderLineCollectionID: Int

OrderCollectionID: Int

OrderID: Int

CustomerID: Int



clsSupplier

Find(): void

Validate(): void

SupplierID: Int

HardwareID: Int

SoftwareID: Int

Name: String

Price: Decimal

QuantityinStock: Int



clsStaffCollection

Count: Int

ThisStaff: clsStaff

StaffList: List<clsStaff>

Add(): void

Delete(): void

Edit(): void

Filter(): void



clsHardwareCollection

Count: Int

HardwareList: List<clsHardware>

ThisHardware: clsWidget

Add(): void

Delete(): void

Edit(): void

Filter(): void



clsSoftwareCollection

Count: Int

SoftwareList: List<clsSoftware>

ThisSoftware: clsSoftware

Add(): void

Delete(): void

Edit(): void

Filter(): void



clsCustomerCollection

Count: Int

CustomerList: List<clsCustomer>

ThisCustomer: clsCustomer

Add(): void

Delete(): void

Edit(): void

Filter(): void



clsSupplierCollection

Count: Int

SupplierList: List<clsSupplier>

ThisSuplier: clsSupplier

Add(): void

Delete(): void

Edit(): void

Filter(): void

**Individual Specification (Customer – Declan):**

I will be designing and building the customer table or this system. For this I will need a table which holds the customers personal details such as name and address and much more and payment information all held under one ID, unique to the customer, this being the primary key.

This information will be captured from the customer when signing up to an account or when placing an order (also creates account). When filling the form in, all the important fields will be marked and will be mandatory so cannot be null. The customer will also be given a randomly generated ID upon creation. This information is then stored in the database and can be used for future purchases saving time on data input at the checkout. Also, it can be used for advertising purposes and contact in relation to orders/issues and offers. A customer will ADD data upon signing up by inputting it into a form. It will be VALIDATED upon entry. They will be able to EDIT and VIEW their customer information should they wish from inside their account settings once they have logged in as well as DELETE their account if they so wish in their settings also. From the back-end staff will also be able to filter through customers details to find customers and also should they need, delete a customer’s account.

**Customer Table fields: Customer table data types:**

|  |  |
| --- | --- |
| Field | Data type |
| Customer ID (PRIMARY KEY) | Integer |
| Full Name | String |
| Member | Boolean |
| DOB | Date |
| Address | String |
| Email address | String |
| Card number | Integer |
| Sortcode | String |

|  |  |
| --- | --- |
| clsCustomer | |
|  | |
| Customer ID  Full Name  DOB  Address  Email address  Card number  Sortcode  Member | Int  varchar(40)  Date  varchar(60)  varchar(64)  Int  varchar(8)  Bool(bit) |
| Add();  View();  Edit();  Find();  Delete();  Filter();  Validate(); | |

**Collection table/table types**

|  |
| --- |
| clsCustomerCollection |
| ThisCustomer | | Int |
| customerList | | varchar(30) |
| Add();  View();  Delete();  Filter; | | |

**Use case diagram (Customer)**

Extends

Includes

includes

includes

includes

**Use Case descriptions (Declan - Customer)**

|  |  |
| --- | --- |
| **Use case name** | CREATE account |
| **Primary actor** | Customer |
| **Business goal** | To ADD a customer to the system and be able to edit |
| **Pre-condition** | Must be over 18, Must not be a bot |
| **Successes condition** | Customer has now registered an account |
| **Main path** | 1. Nonregistered customer clicks ‘Create account’ on the Index page or upon trying to make a purchase 2. System will display Register page. 3. Customer then enters information requested into the fields and presses register button. 4. System redirects the newly Registered customer to the Log in page or proceeds with purchase |
| **Variants path** | 1a. User enters invalid data  1.System will display validations and request information to be re-entered.  3b. username already exists in the database  1.System will display message telling user they already have an account and re-direct them to login  4a. Server error   1. System displays error message and tells user to try again later |

|  |  |
| --- | --- |
| **Use case name** | EDIT account |
| **Primary actor** | Customer |
| **Business goal** | To EDIT a customer to the system and be able to edit |
| **Pre-condition** | Have an account |
| **Successes condition** | Customer details updated |
| **Main path** | 1. Customer logs into account and access their data in the settings 2. Details page displayed 3. Change required fields and submit 4. System shows update successful notification |
| **Variants path** | 1a. Password forgotten  1.System will display message telling user to reset password if forgot  3a. User enters invalid data  1.System will display validations and request information to be re-entered.  4a. Server error  1.System displays error message and tells user to try again later |

|  |  |
| --- | --- |
| ***Use Case Name:*** | DELETE account |
| ***Primary Actor:*** | Customer |
| ***Business Goal:*** | *To DELETE a customer on the system* |
| ***Precondition*** | *Must have an account* |
| ***Success Condition*** | *A customer has now removed their account from the database* |
| ***Main Path*** |  |
|  | 1. Registered Customer logs into account and accesses settings 2. In settings will be a delete account button 3. Customer then enters password required for confirmation to delete 4. System redirects the main page logged out, with confirmation |
| ***Variant Paths*** |  |
|  | 1a. Password forgotten   1. System will display message telling user to reset password if forgot   4a. Server error   1. System displays error message and tells user to try again later |

|  |  |
| --- | --- |
| ***Use Case Name:*** | FILTER/SEARCH account details |
| ***Primary Actor:*** | Staff |
| ***Business Goal:*** | *To FILTER/SEARCH customer on the system* |
| ***Precondition*** | *Must be staff/ have access authority* |
| ***Success Condition*** | *Particular records found* |
| ***Main Path*** |  |
|  | 1. Staff member logs in with details 2. Enters search criteria into back end 3. Gets results and selects record/details to view 4. Staff can do whatever is necessary from there |
| ***Variant Paths*** |  |
|  | 1a. Password forgotten  1.System will display message telling staff member to reset password if forgot  2a. No results found   1. Error message displayed that no results meet the criteria and staff is returned to search form   3a. Server error   1. System displays error message and tells user to try again later |

|  |  |
| --- | --- |
| ***Use Case Name:*** | VIEW account details |
| ***Primary Actor:*** | Customer |
| ***Business Goal:*** | *To VIEW a customer on the system* |
| ***Precondition*** | *Must have an account* |
| ***Success Condition*** | *A customer has now removed their account from the database* |
| ***Main Path*** |  |
|  | 1. Registered Customer logs into account and accesses settings   1. In settings will be a button for view account details, they press this. 2. This opens and the customer is presented with their details with further functions if needed. 3. Once done user logs out. System redirects the main page logged out, with confirmation |
| ***Variant Paths*** |  |
|  | 1a. Password forgotten   1. System will display message telling user to reset password if forgot   4a. Server error   1. System displays error message and tells user to try again later |

|  |  |
| --- | --- |
| ***Use Case Name:*** | Validate customer details |
| ***Primary Actor:*** | Customer |
| ***Business Goal:*** | *To Validate a customer’s information on the system* |
| ***Precondition*** | *Must be signing up for an account* |
| ***Success Condition*** | *A customer will have successful met the criteria for a account and created one* |
| ***Main Path*** |  |
|  | 1. Nonregistered customer clicks ‘Create account’ on the Index page or upon trying to make a purchase 2. System will display Register page. 3. Customer then enters information requested into the fields and presses register button. 4. System validates details, if they meet criteria then customer proceeds to purchase, if not era is displayed |
| ***Variant Paths*** |  |
|  | 1a. Incorrect data/fields missing   1. System will display message telling user inputted data is incorrect for field type, user must re-enter correct data, or data for fields.   4a. Server error   1. System displays error message and tells user to try again later |

**Individual Specification (Staff - Nikhil):**

**Allocated Component:** Staff Table

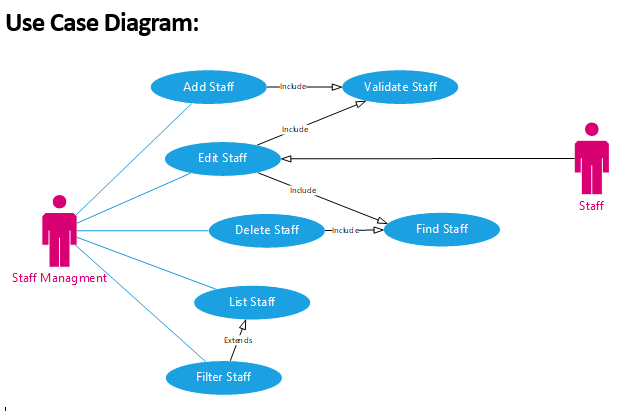
**Developer:** Nikhil

**Overview:**

My individual requirement is to create the staff management table for the database. The staff table will be used to enable the management to be able to see the staff details such as phone number if the management department needs to contact them. Staff is a vital table to have in a database as it contains what person is responsible for what job role (department). Management should control the whole table to ensure it has the fields and data inserted is correct.

My task is also to do the front end of the table. I have to a create the fields plus allow certain departments to add, edit, delete and filter the staff records. This means that I should add the staffs details such as their names and personal information e.g. address. Staff management will have to filtering the staff records. Filtering would help certain department too see what’s relevant to them. An example would be for a manager department, they might not want to see the ‘Permission to change/ add fields’ field whilst a Staff Management department will want to see it. Edit and delete in a sense of changing details such as updating email addresses if needs to be. Validate should also be done to ensure it has the correct format such as for database it should be dd/mm/yyyy, possibly format phone number by the region code e.g. +44…. When looking for staff from a certain authorised department, find will allow them to list them all.

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Key** |
| StaffID | AutoNumber (Integer) | Primary Key |
| Staff Firstname and Surname | Text |  |
| Department | Text |  |
| Date of Birth | Date |  |
| Street Address | Text |  |
| Phone Number | Integer |  |
| Permission to change/ add fields | Boolean |  |



**Use Case Descriptions:**

|  |  |
| --- | --- |
| Use Case Name (Short two or three word name) | Listing of staff. |
| Use Case Description (Short description) | The staff management views a list of staff in the system. |
| Use Case Author(s) (Who wrote this) | Nikhil Jagatia. |
| Actor(s) (Who does this) | Staff Management. |
| Locations (Where does this happen) | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | List staff.  The management should first enter the system and a list of staff names should appear. |
| Alternate pathways (What other paths are there that are not the “happy path”?) | If there is no data in the system – a message is displayed saying “No Staff data to be shown”.  Management will apply filters that can hide the staff.  A filtered list will make the user have specific search criteria e.g. look for staff in Leicester and no staff is listed. |
| Exception pathways (What could possibly go wrong?) | Database connection does not connect.  Added staff not saved. |

|  |  |
| --- | --- |
| Use Case Name (Short two or three word name) | Filtering of staff. |
| Use Case Description (Short description) | Filtering staff should only show relevant information for the staff e.g. name, id and department and ignoring addresses |
| Use Case Author(s) (Who wrote this) | Nikhil Jagatia. |
| Actor(s) (Who does this) | Staff Management. |
| Locations (Where does this happen) | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | The filter staff should show the relevant information that relates to the query. E.g. who’s in department Advertiser, this should show only the people who is in advertising department |
| Alternate pathways (What other paths are there that are not the “happy path”?) | If there is no data in the system – a message is displayed saying “No Staff data to be shown”.  Case sensitive meaning if you don’t put “Advertiser” compared to “advertiser” it will not show the result. |
| Exception pathways (What could possibly go wrong?) | Field is left as null meaning doesn’t need the specific information of the staff |

|  |  |
| --- | --- |
| Use Case Name (Short two or three word name) | Deleting of staff. |
| Use Case Description (Short description) | The staff management should be able to delete staff out the system. |
| Use Case Author(s) (Who wrote this) | Nikhil Jagatia. |
| Actor(s) (Who does this) | Staff Management. |
| Locations (Where does this happen) | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | Staff management should be able to see the data table and delete the staff |
| Alternate pathways (What other paths are there that are not the “happy path”?) | If there is no data in the system so there is nobody to delete off  Deleting specific information of the staff but not the whole field |
| Exception pathways (What could possibly go wrong?) | If the management does not delete all of the certain staff records and leave some fields still in it will cause issues to filtering as it will show in queries that a person is fitting the criteria however the staff is not active.  The management accidentally deletes a staff record |

|  |  |
| --- | --- |
| Use Case Name (Short two or three word name) | editing of staff. |
| Use Case Description (Short description) | The staff management should be able to edit staff records |
| Use Case Author(s) (Who wrote this) | Nikhil Jagatia. |
| Actor(s) (Who does this) | Staff Management. |
| Locations (Where does this happen) | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | Staff is shown on the data table and the management will be able to overwrite the information |
| Alternate pathways (What other paths are there that are not the “happy path”?) | If there is no data in the system, so their wont be anybody to edit. |
| Exception pathways (What could possibly go wrong?) | The management accidentally edits the wrong information  The management does not understand the specific formats or validation. |

|  |  |
| --- | --- |
| Use Case Name (Short two or three word name) | Adding of staff. |
| Use Case Description (Short description) | The staff management should be able to add staffs and their record. |
| Use Case Author(s) (Who wrote this) | Nikhil Jagatia. |
| Actor(s) (Who does this) | Staff Management. |
| Locations (Where does this happen) | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | Staff should be able to be added with an autogenerated ID.  Staff records should be easily entered |
| Alternate pathways (What other paths are there that are not the “happy path”?) | Write partial staff information but not enter all the details |
| Exception pathways (What could possibly go wrong?) | The management accidentally adds a new field of staff which will make a random ID generate and that field will not be able to be deleted. Meaning there is a random StaffID with no staff to fill its place. |

******Individual Specification:**

**Products hardware – Daniel**

My individual component will hold and maintain information on the stock levels of products that are in the product table. It will do this by monitoring the current stock levels constantly and if needed instruct the employees to purchase more stock if it gets too low through the “Stock Required” field. The table will also allow for several back-end commands these are: Adding, Removing, Modifying Products, Checking and returning a product with its fields if it exists, Listing products either with or without a filter. For the front-end commands these are: Searching, Filtering, listing products, when searching for or in any way interacting with products there will be a small validation method to ensure the product exists or has been removed. This allows the website to return “No products match”, if no products exist for the customer as well as stop an employee adding several products with the same key as it will check if the key exists beforehand.

For the backend I will create a table with several fields for each product being represented in the hardware table, these are:

|  |  |  |
| --- | --- | --- |
| **clsTableHardware** | **Data Type** | **Description** |
| Product ID | Integer | Product unique ID key. |
| Name | String | Name of the product. |
| Description | String | Description of the product. |
| Price | Integer | Price of the product. |
| Quantity in Stock | Integer | Displays the quantity in stock. |
| Stock Required? | Boolean | Tells employees if stock is needed. |
| Date | Date | Specifies when the stock will go out of stock. |

The hardware table will keep track of hardware stock levels and help maintain them it also provides some basic info on a product, it does this by keeping track of the current stock quantity and displaying if stock is required as the levels has got too low. If the stock levels get too low, it will swap “Stock required” to Yes telling employees that it is likely the product will go out of stock if not resupplied soon. The primary key is used to identify a product in the table as each key is unique, each record also includes a name so employees can easily identify a product they are trying to retrieve, the design of each record allows for improvements such as addition fields to be added later on. The table will also store the date the product is likely to go out of stock.

**Use Case Diagram:**

<<Includes>>

Extends

**Use Case Diagram:**

|  |  |  |
| --- | --- | --- |
| ***Use Case Name:*** | *SearchForProduct* | |
| ***Primary Actor:*** | *Customer* | |
| *Business Goal:* | *Display a product(s) to the customer* | |
| *Precondition* | *The product must exist* | |
| *Success Condition* | *A product(s) is found and displayed to the customer* | |
| *Main Path* | 1. Customer enters a Hardwareid number or product name | 2.Hardwareid is checked to see if it matches any Hardwareid in the table. If a name is entered each Name on each row is checked to see if there is a match |
|  | 3. Results returned if it does match a product, with the product: Name, Description, Price, Amount in Stock, if more stock is needed or on the way, and the date it was added. |  |
| *Variant Paths* | 3a. User enters incorrect search bar input data type(E.g. Non string data type).   1. Page will display “No products match”.   3b. Server is down, or not working.   1. Page will display an error.   3c. Hardwareid doesn’t exist   1. Page will inform the customer the Hardwareid or product doesn’t exist.   3d. The customer wants to filter out products   1. The page will display products that have a valid Hardwareid and fit the filter applied, for example a price below £150 only products with a price of £149.99 and below will show. | |

|  |  |  |
| --- | --- | --- |
| ***Use Case Diagram:*** | *RemoveProduct(s)* | |
| ***Primary Actor:*** | *Employee* | |
| Business Goal: | *Remove Product from hardware table* | |
| Precondition | *The product must exist to be removed* | |
| Success Condition | *Product exists and successfully removed* | |
| Main Path | 1. Find row matching to the Hardwareid provided | 2. Remove all columns that are in the row with the Hardwareid provided then remove the Hardwareid |
| Variant Path | 2a.Matching Hardwareid not found   1. Inform the employee that they cannot remove a product which doesn’t exist as the Hardwareid doesn’t exist. | |

|  |  |  |
| --- | --- | --- |
| ***Use Case Diagram:*** | *AddProduct(s)* | |
| ***Primary Actor:*** | *Employee* | |
| Business Goal: | *Add Product to the hardware table* | |
| Precondition | *The product must not exist to be added* | |
| Success Condition | *Product does not exist and is added* | |
| Main Path | 1.Check the Hardwareid to ensure it doesn’t already exist to an existing product | 2. Create an empty row with the Hardwareid, with each column being Null until information is entered. |
| 3. Once the row is created the values Null are replaced with the information provided by the employee |  |
| Variant Path | 3a. Matching Hardwareid found   1. Inform the employee the Hardwareid already exists   3b. Incorrect data entered   1. Inform the employee that the information they’ve entered is either in the incorrect format or data type, E.g. Date: 12091993, Price: Fifty Pounds. | |

|  |  |  |
| --- | --- | --- |
| ***Use Case Diagram:*** | *Edit/UpdateProduct(s)* | |
| ***Primary Actor:*** | *Employee* | |
| Business Goal: | *Edit or update a product in the table* | |
| Precondition | *The product must exist to be modified* | |
| Success Condition | *Product exists and is modified.* | |
| Main Path | 1.Validate that the Hardwareid exists from the selected product | 2. Display the data attached to the Hardwareid, such as the name, etc. |
| 3.Make changes to the product such as price, or amount in stock. | 4. Update the database overwriting and replacing any modified data. |
| Variant Path | 4a. Matching Hardwareid not found   1. Inform the employee the product does not exist so it cannot be edited or updated   4b. Incorrect data entered   1. Inform the employee that the information they’ve entered is either in the incorrect format or data type, E.g. Date: 12091993, Price: Fifty Pounds. | |

|  |  |  |
| --- | --- | --- |
| ***Use Case Diagram:*** | *ValidateProduct(s)* | |
| ***Primary Actor:*** | *Employee/Customer* | |
| Business Goal: | *Validate the product exists* | |
| Precondition | *There must be at least one product in the database.* | |
| Success Condition | *Validation returns if the product exists or not.* | |
| Main Path | 1.Retrieve the Hardwareid and attempt to match it to a product(s). | 2. Continue with the next commands (List, Add, Remove, etc). |
| Variant Path | 2a. The Hardwareid doesn’t exist.   1. Display “No Products Match” on the page | |

|  |  |  |
| --- | --- | --- |
| ***Use Case Diagram:*** | *FilterProduct(s)* | |
| ***Primary Actor:*** | *Customer* | |
| Business Goal: | *Filter out products based on filter entered* | |
| Precondition | *There must be at least one product in the database.* | |
| Success Condition | *Products are displayed based on the filter.* | |
| Main Path | 1. Select the product(s) with the entered Hardwareid. | 2. Remove all products that do not match the filter used. |
| 3.Display the remaining products onto the page. |  |
| Variant Path | 3a. The Hardwareid doesn’t exist.   1. Display “No Products Match” on the page   3b. No products to display   1. Tell the customer that the filter doesn’t return any matching products | |

|  |  |  |
| --- | --- | --- |
| ***Use Case Diagram:*** | *ListProduct(s)* | |
| ***Primary Actor:*** | *Customer.* | |
| Business Goal: | *List Products in database.* | |
| Precondition | *There must be at least one product in the database.* | |
| Success Condition | *One or more products are listed depending on the amount in the database.* | |
| Main Path | 1. Find the product with the matching Hardwareid. | 2. Validate the product and ensure it exists in the database |
| 3.Once validated display the product(s) on the page. |  |
| Variant Path | 3a. The Hardwareid doesn’t exist.   1. Display “No Products Match” on the page | |

**Individual Specification (Mishal):**

**Allocated Component:** Order

**Developer:** Mishal

**Overview:**

My particular component will be focusing on the outgoing orders which the customers will have placed. To achieve this an order summary will be required to be made using the right pieces of information being collected throughout each of the earlier stages. An order ID will be assigned where the given customers’ details as well as the product(s) they have chosen will be added to. As soon as the order is payed for the order date and time would be recorded as reference. According to the processing times the rest of the details, whether the product(s) have been shipped and who by, will be added to the overall summary for the convenience of both the customer and supplier.

The front end will enable the customer to see the order details as a virtual receipt, whilst the back end will allow for the maintenance of the ordering process. This by either allowing the order to proceed through each of the stages or cancel it completely, which can be affected by both the customer themselves or by staff.

Both Order Line Collection and Order Collection will be allowed to add, delete, edit and filter their corresponding lists, this is so that whatever the alterations that need to be made can be possible e.g. if an order is to be transferred for the customer to collect then it would be possible when it gets added to the Order Collection list. Filter and find would enable the department in question to search first narrow results and then locate any piece of information in a given table, this would be greatly beneficial as it saves time and effort. For example, if a certain hardware component is to be found then a simple reduction of all software components would simplify the results from which the component would be easily found from. Validation should be implemented so that the correct format of information is inputted into the database, for example it would be possible to enter a date as dd/mm/yyyy with the right formatting in place.

tblOrder - OLD

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Key |
| OrderID | Int | Primary |
| CustomerID | Int | Foreign |
| HardwareID | Int | Foreign |
| SoftwareID | Int | Foreign |
| OrderDate | Date/Time |  |
| Shipped | Boolean |  |
| ShippingVia | Text |  |

tblOrder

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Key |
| OrderID | Int | Primary |
| OrderLineCollectionID | Int | Foreign |
| CustomerID | Int | Foreign |
| OrderDate | Date/Time |  |
| Shipped | Boolean |  |
| ShippingVia | Text |  |

tblOrderLine

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Key |
| OrderLineID | Int | Primary |
| HardwareID | Int | Foreign |
| SoftwareID | Int | Foreign |

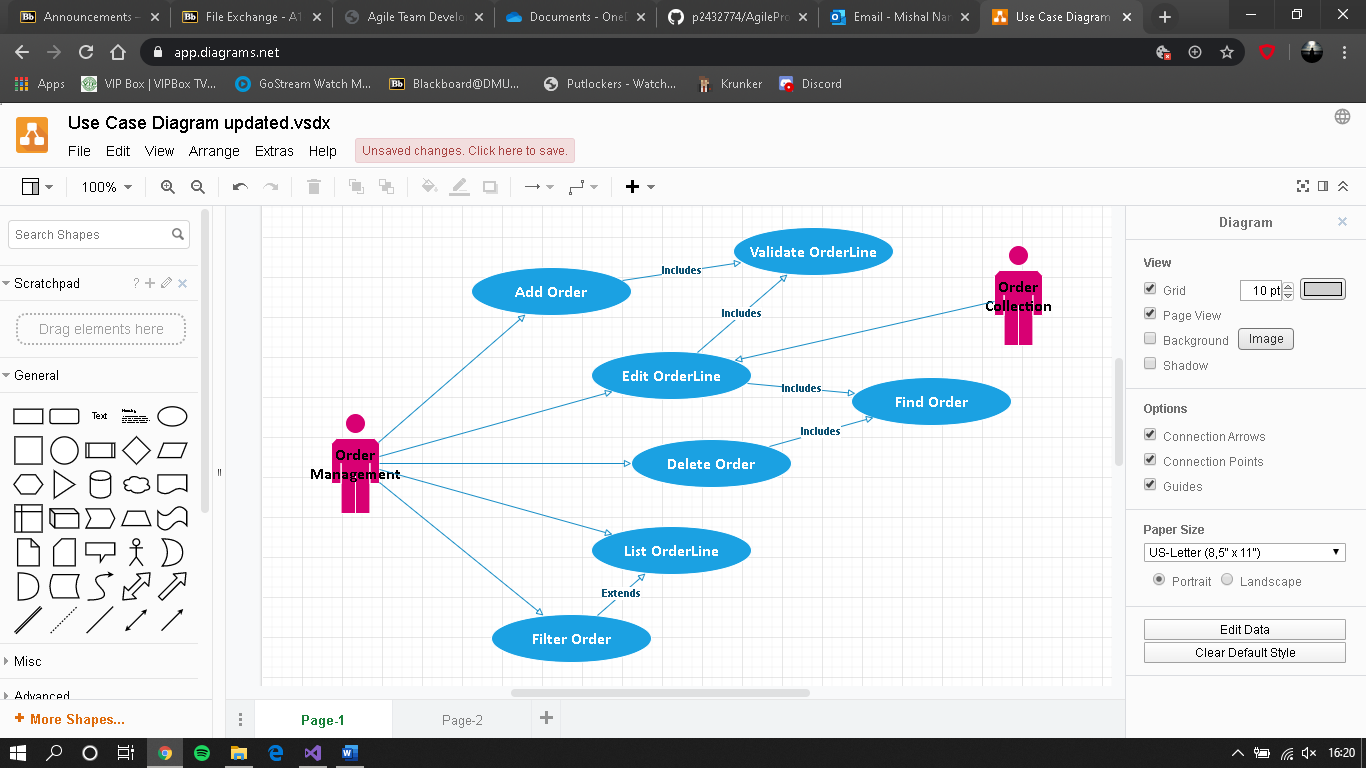
tblOrderLineCollection

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Key |
| OrderLineCollectionID | Int | Primary |
| OrderLineID | Int | Foreign |

tblOrderCollection

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Key |
| OrderCollectionID | Int | Primary |
| OrderID | Int | Foreign |
| CustomerID | Int | Foreign |

**Use Case Diagram:**



**Use Case Descriptions**

|  |  |
| --- | --- |
| Use Case Name | AddOrder |
| Use Case Description | The Order Management adds an Order to the order line. |
| Use Case Author(s) | Mishal Nanalal. |
| Actor(s) | Order Management. |
| Locations | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | Adds an Order to the order line.  User submits an order; the system adds said order onto the existing order line to be processed. |
| Alternate pathways (What other paths are there that are not the “happy path”?) | Invalid Order is placed – a message is displayed informing them “Order could not be placed”.  User applies filter; a filtered list of the active order line collection is displayed to the user.  User clears filter; full list of the correct matches is displayed to the user. |
| Exception pathways (What could possibly go wrong?) | Database connection fails; error displayed to the user advising of connection problem. To prevent any of these problems both the cloud database and server need to be running correctly. |

|  |  |
| --- | --- |
| Use Case Name | EditOrderLine |
| Use Case Description | The Order Management edits an Order on the order line. |
| Use Case Author(s) | Mishal Nanalal. |
| Actor(s) | Order Management. |
| Locations | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | Allows editing of the Order in the order line.  Submitted order can be edited before it reaches the next stage; updating the existing order line. |
| Alternate pathways (What other paths are there that are not the “happy path”?) | Particular order cannot be edited at current stage – a message is displayed showing this “Order cannot be altered”.  Listed order line will not be altered, and original order will be shown instead. |
| Exception pathways (What could possibly go wrong?) | Database connection fails; error displayed to the user advising of connection problem. To prevent any of these problems both the cloud database and server need to be running correctly. |

|  |  |
| --- | --- |
| Use Case Name | DeleteOrder |
| Use Case Description | The Order Management views a list of order line collection. |
| Use Case Author(s) | Mishal Nanalal. |
| Actor(s) | Order Management. |
| Locations | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | List Order collections.  User enters the system; a list of all the active order line collection is displayed to the user at system start. |
| Alternate pathways (What other paths are there that are not the “happy path”?) | There is no data in the system – a message is displayed saying “No order line data available”.  User applies filter; a filtered list of the active order line collection is displayed to the user.  User clears filter; full list of the correct matches is displayed to the user. |
| Exception pathways (What could possibly go wrong?) | Database connection fails; error displayed to the user advising of connection problem. To prevent any of these problems both the cloud database and server need to be running correctly. |

|  |  |
| --- | --- |
| Use Case Name | FilterOrder |
| Use Case Description | The Order Management filters valid results to pinpoint correct Order(s). |
| Use Case Author(s) | Mishal Nanalal. |
| Actor(s) | Order Management. |
| Locations | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | List Order(s) which match ID.  User enters the system; the relevant order is displayed to the user at system start. |
| Alternate pathways (What other paths are there that are not the “happy path”?) | There is no data in the system – a message is displayed saying “No order data available”.  User applies filter; a filtered list of active order line collections is displayed to the user.  User clears filter; full list of the correct matches is displayed to the user. |
| Exception pathways (What could possibly go wrong?) | Database connection fails; error displayed to the user advising of connection problem. To prevent any of these problems both the cloud database and server need to be running correctly. |

|  |  |
| --- | --- |
| Use Case Name | ListOrderLine |
| Use Case Description | The Order Management views a list of order line collection. |
| Use Case Author(s) | Mishal Nanalal. |
| Actor(s) | Order Management. |
| Locations | Part of the back-end functionality. |
| Primary pathway (What is the normal “happy path” for this use case?) | List Order collections.  User enters the system; a list of all the active order line collection is displayed to the user at system start. |
| Alternate pathways (What other paths are there that are not the “happy path”?) | There is no data in the system – a message is displayed saying “No order line data available”.  User applies filter; a filtered list of the active order line collection is displayed to the user.  User clears filter; full list of the correct matches is displayed to the user. |
| Exception pathways (What could possibly go wrong?) | Database connection fails; error displayed to the user advising of connection problem. To prevent any of these problems both the cloud database and server need to be running correctly. |

**Individual Specification (Farhaan):**

**Products software – Farhaan-**

Allowing, adding, editing, deleting, listing and filtering products data from our records**.**

This table will be used for software Products where I will be categorising different types of software such as Internet security software, Office Software, creative and editing software in my table. In this table will have one primary key which is product ID. As this where customers place an order from it has various foreign key from various table When the customer places an order it will pull the product information from Product ID in order to get place an order.

**Product Table-** Product ID, Product Name, Product Description, Price, In stock

**Data Types**

**Product ID – (Primary Key) -**This will generate unique number in order to find the Product by using product ID from the database. (Auto number)

**Product Name –** Name of the Products. (Text)

**Product Description**- I will be adding product specification in this table. (Text)

**Price** –This will view the price of the products. (Decimal)

**Date of arrival** - Date

**In Stock** -Boolean (Yes/No)- This will check the stock level of the products. If we have stock customers can place an order otherwise it won’t let the customer to make an order until the company have products in stocks.

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Types** | **Key** |
| Product ID | AutoNumber | Primary Key |
| Product Name | Text |  |
| Product Description | Text |  |
| Price | Decimal |  |
| Date of arrival | Date |  |
| In Stock | Boolean |  |

Developer: Software Management System

Overview:

A customer visits the web site to buy some Software. They have the option to add themselves as a customer to the system and edit their details. Also, they are able to cancel their account on-line. Upon finding out that they are not a customer they follow the sign-up process entering their details and selecting a user name and password. On the back end of the system staff have the option to manage the data for all customers (add, edit delete) and run reports on customers. I will have access to Software Table where I will be dealing with software categories and full responsible for keeping the data up to date.

Add -

Edit-

Update -

Delete -

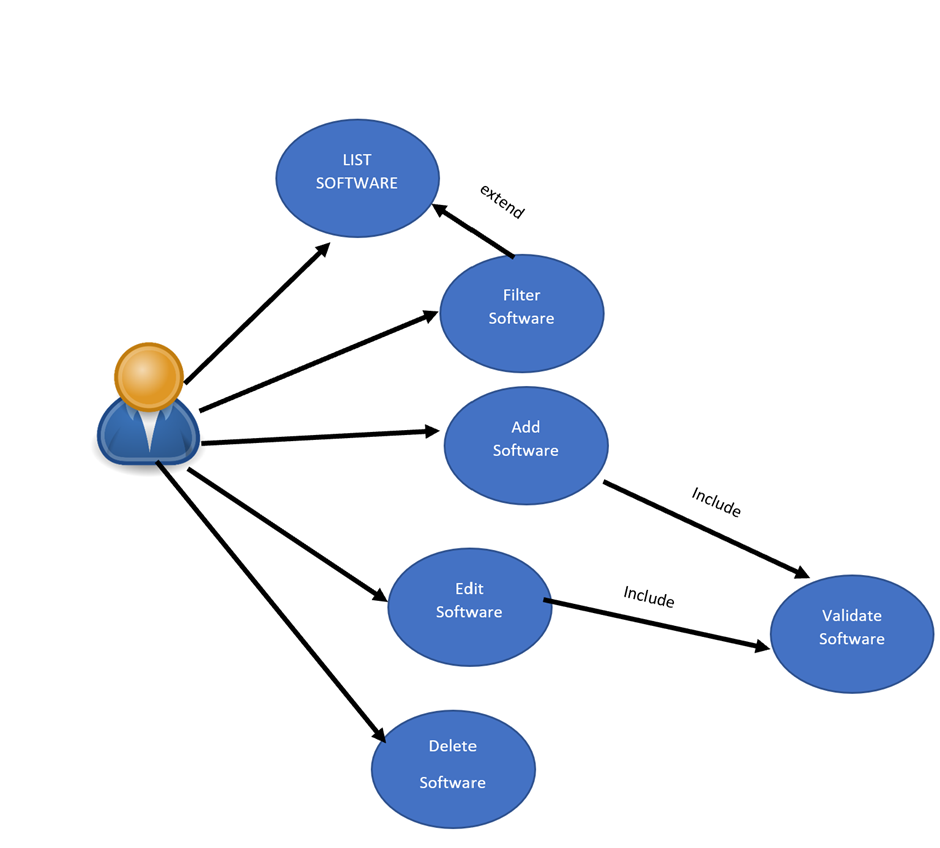
List -

Find -

Filter -

Validate

**Case Diagram**

****

**Case Description**

|  |  |
| --- | --- |
| Use Case Name: | List Software |
| Primary Actor | Customer |
| Business Goal | Display the products to the customers |
| Precondition | The product must exist |
| Success Condition | A product is found and displayed to the customer |
| Main Path | * Customer will enter a product name on the website search bar. * New window will appear with lists of software.   The customer will click on the product and able to get access to the software information.   * If product does not exist it will come up with an error “No Product Found” |
| Validate Path | * Database connection fails/ Use the filter incorrectly/ * Error displayed to the user advising of connection problem |

|  |  |
| --- | --- |
| Use Case Name: | Add Software |
| Primary Actor | Employee |
| Business Goal | Add product to Software Table |
| Precondition | The product must exist to be added |
| Success Condition | Product is successfully added to the software table |
| Main Path | * User must enter a product information to the correct field. * User must fill all the fields * User must insert data with correct field size. * User will press OK. * Once the Ok button has been hit then it will add the product to the software Table. * The user will get the message “The user has successfully added the table”. * It will validate and will be added automatically. |
| Validate Path | * The User must stay with specific field size otherwise user won’t be able to add a product. * If user does not use all the field , All filed must be entered otherwise it won’t work, * Database connection fails/ Use the filter incorrectly/ * Error displayed to the user advising of connection problem |

|  |  |
| --- | --- |
| Use Case Name: | Edit Software |
| Primary Actor | Employee |
| Business Goal | Edit product from Software Table |
| Precondition | The product must exist to be edited |
| Success Condition | Product is successfully edited to the software table |
| Main Path | * User will have to press find button. * New window will pop up * Then user will have to enter specific Software ID in order to find a product. * User will have to enter some data in order to find the data then user needs to edit it. * User will make changes and click on saved button after making changes. * Example: The user can change prices or Name * The user will get warning message “Are you sure you want to save these changes” then you click ok * It will validate automatically and appear in the software list. |
| Validate Path | * Product does not exist. * User is not using correct Field size. * User is putting wrong data to wrong field such price field 299 you can only put numbers not Text. * Database connection fails * Error displayed to the user advising of connection problem |
| Use Case Name: | Delete Software |
| Primary Actor | Employee |
| Business Goal | Delete to Software Table |
| Precondition | The product must exist to be Deleted |
| Success Condition | Product exists and successfully removed |
| Main Path | * The user will press Find Button which will allow user to find. * The user will find a product by using specific name such as Unique ID or Name. * Then New list will appear and then user can delete the field. * Before you delete any record, it will come up with conformation “Are you sure you want to delete”. When you click OK * When you hit ok the data will be removed |
| Validate Path | Database connection fails/ Use the filter incorrectly/  Error displayed to the user advising of connection problem |

|  |  |
| --- | --- |
| Use Case Name: | Filter Software |
| Primary Actor | Employee/Customer |
| Business Goal | Filter |
| Precondition | The product must exist to be Filter |
| Success Condition | The products should display with filters |
| Main Path | * List Software * User enters the system * A list is displayed to the user at system start. * There is no data in the system – a message is displayed saying so * User applies filter * A filtered list is displayed to the user new window will pop up with new list. * User clears filter Full list is displayed to the user |
| Validate Path | Database connection fails/ Use the filter incorrectly/  Error displayed to the user advising of connection problem |