# Modify Product (Add, remove, update in information)-scenario

# Normal:

The employee / manager enters the site and logs into the account. He wants to introduce a new type of meat. Go to the meat page. Click the add button. A special page is presented to him to add, then he puts the information about the type of meat such as the name of the meat, its price and quantity, and then confirms the listing for display to customers.

Alternative:

The employee / manager enters the site and enters the account. He wants to adjust the price of meat. Goes to the meat page, clicks on the row inside the table that contains information about the meat, then on the edit button, the site goes to a special interface in the meat modification, the information appears in text fields as it is, and then chooses the item to be modified, then sets the price, then Confirm the modification.

Alternative:

The employee/manager enters the site and logs into the account after login, he wants to delete an item, he goes to the meat page and then chooses the row within the table by clicking on it and then on the delete button of the item he wants to delete, then confirms the deletion.

Error:

The employee / manager enters the site and enters the account after logging in, he wants to update the name of the type of meat, he goes to the meat page of the meat, then he chooses the meat that he wants to update from then to edit the page, he wants to update the name of the type, but when he updates it, the Displays an error because there is also an element with the same name, now the weft type is (unique) within the databases.

Modify Product (Add, remove, update in information)-Use Case

|  |  |
| --- | --- |
| Actors | Customers |
| Description | The employee/manager amends, adds, and deletes within the databases on meat and its types and knows who did this process within the system |
| Pre-conditions | 1. The employee/manager login in website. 2. Go to the special table in the meat inside the site.  3. And do what he wants to modify inside the site on the meat |
| Sequence/Flow of Events | 1. Make a login 2. He clicks on the special weft button to modify 3. Shows him a table containing all the woof information within the site 4. If he wants to add, he clicks on the add button, then an interface appears containing a text field to write the types he wants to add. He writes what he wants and presses the green add button 5. And if he wants to modify, he clicks on a row inside the table that contains the data and presses “Modify” by moving it to an interface that contains text fields with the information inside the information of the row he clicked on. You can edit what he wants and press the edit button 6. And if he wants to delete, he clicks on a row inside the table that contains the data and presses “Delete.” An interface appears to him containing the text “Are you sure what you want to do?” Then you press the Yes button, if you want, if you do not want to, press the No button 7. After completion, you will exit the page with an exit button that will take you back to the employee and manager information page |
| Data | Addition in the event of a new type, for example, and deletion such as corruption of the weft and modification when changing the price |
| Stimulus/Trigger | The employee and manager can use the meat shop use the product and type modification |
| Post-conditions/ Response | Everything that was used is saved on the special databases of the meat store's website |
| Comments | There are no limits on the number of updates |

start

employee/manager login in website

Button Modify Product

No

No

Add item

Close page

Remove item

Edit item

unique)

Done

End

yes

yes

yes

Add item

Add item

Add item

The id number is correct

The id number is correct

No

Type is unique

ACTORS analysis and their description

|  |  |  |
| --- | --- | --- |
| ACTORS | Description | USE-CASE |
| Customer | This actor represents someone who is a customer of the meat shop, that can search and order meat online. | 1-Register  2-login  3-Modify Order  4-Search Product |
| Employee | This actor represents someone who takes  The customer's request, and verifies it, updates and adds to the types of meat | 1-login  2- Search Product  3-Modify Product  4-Accept an Order |
| manager | This representative represents the person who updates and adds types of meat, creates accounts for employees, and accepts the customer upon registration  It can also be modified according to the customer's request  The decision-maker is on the site | 1-Register  2-login  3-Modify Order  4-Search Product  5-Modify Product  6-display report |

1.8-

1)

In my view of evaluating the project, I see that it is a successful project because of the ease of the site and its handling. It is possible to enter more than one customer and provide convenience to him and no fraud or fraud will be placed.

It also provides a payment method that suits the user.

2) We entered into a lot of discussions, such as the distribution of roles in the project. The first thing that was discussed was the requirements of the program.

We arranged the requirements, collected them, and tried them,

I also discussed (time estimation) with the team and wrote down the times,

I discussed with the team (actors analysis) and I wrote, and we discussed the use case with my colleague Saleh.

I also made plans (Overall architecture diagram)

The diagram has three layers, one for the graphical user interface and one for the application and infrastructure,

User interface classes are set as components of the application, and we set common technical elements

Categories for infrastructure components.

A couple between two classes means that these cases frequently cooperate.

To make the component diagram less complex, and each component serves one function,

I also wrote and edited the (General design goal).

General design goal:

1- High cohesion: The system must have high cohesion so it must cooperate to do one function to make the component schema less complex, components perform logically related tasks, so they may use the same function in certain places, using the same data, this is every component we designed in our system performs well a particular function, that is, we define two components: the “customer”, the “employee” who will lead

For high cohesion because we separate each component to serve only one function. While that

From having only one component called "USER" that contains both employee and

A customer that will result in a low cohesion system, as we can see from our components

Diagram the relationship between the components is simple and not complicated.

2- Low coupling:

Couples between two classes mean that these issues frequently cooperate.

Customer categories, meat and order list frequently cooperate which means they should

be in the same domain class, so we grouped these classes into the same component (order

database) and deals with it simply

3-User-Friendliness: System should be easy to use and fast to learn, both customer and employee don't have to waste their time learning about the system

|  |  |  |  |
| --- | --- | --- | --- |
| UR | Estimated Effort | Estimated No of Developers | Total Effort |
| UR1 | 2 pw | 1 | 2 |
| UR2 | 1 pw | 1 | 1 |
| UR3 | 1 pw | 2 | 2 |
| UR4 | 1 pw | 1 | 1 |
| UR5 | 1 pw | 1 | 1 |
| UR6 | 4 pw | 2 | 8 |
| UR7 | 1 pw | 1 | 1 |
| UR8 | 2pw | 1 | 2 |
| UR9 | 1 pw | 1 | 1 |
| UR10 | 6pw | 2 | 12 |
| UR11 | 5 pw | 1 | 5 |
| Total effort/avgas | 25pw | 14/11=1.27 | 36pw |
| Schedule time 30% | 25\*1.30=33w |  | 36\*1.30=  468w |
| Cost |  | avgas salary = 200$ | 200$\*22W=  4400$ |
| Profit margin(Min 10% and Max 30% |  | Min Cost: | 4400\*1.10  =4840 |
|  |  | Max Cost | 4400\*1.30  =5720 |

If the work is divided in half between the two  
UR1+UR4+UR5=2W

UR3 =1W

UR2+UR7=1W

UR6=4W

UR10=6W

UR11+UR8+UR9=5W

19 PW

we make them 19 pw the Work with 19 weeks if we use two devs at the same Time 19\*1.30=24.7w