|  |  |
| --- | --- |
| User Story | Acceptance Criteria |
| As a customer I want to be able to order online | User can create order using the User Dialogue.  User dialogue navigates user through different fields.  User dialogue loops (does not exit) upon completion. |
| As a customer, I want to be able to see the pizza I am going to order to know the toppings and prices. | User can call function and view full menu of pizza and drinks.  Menu data is stored within the system |
| As a customer, I want to be able to choose what toppings I want on the pizza. | User can order pizza with extra toppings.  User can change selected extra toppings. |
| As a customer, I want to be able to login and edit my profile. | User can login to access their information.  User can change their information.  User information is stored within the system. |
| As a customer I want to be able to request for delivery. | User can request for delivery in order page. |
| As a customer I want to be able to see my orders | User can view their order.  System display only specific user order. |
| As a customer, I want to be able to pay with card or cash. | Payment can be chosen as credit card or cash. |
| As owner I want customers to be able to choose which branch to order from. | User can choose which store to order from.  Store location displayed in order detail. |
| As a customer I want to be able to manage my order(shopping card) so I can add and remove/edit items. | User can perform CRUD operations on shopping cart.  User cannot submit an empty shopping cart. |
| As the shopkeeper owner I want orders to be able to delivered directly to customers. | Employee can be assigned to specific delivery orders. |
| As a customer. I want to be able to register and apply for a club membership | User can perform registration.  Discount is applied to checkout |
| As a shopkeeper I want to be able to get the customer who made a specific order. | Search can be performed to find customer who made specific order. |
| As a shopkeeper I want to be able to view all orders. | User can view all orders made by all users. |

|  |
| --- |
|  |

User stories and aceptance criteria added after 1st Sprint.

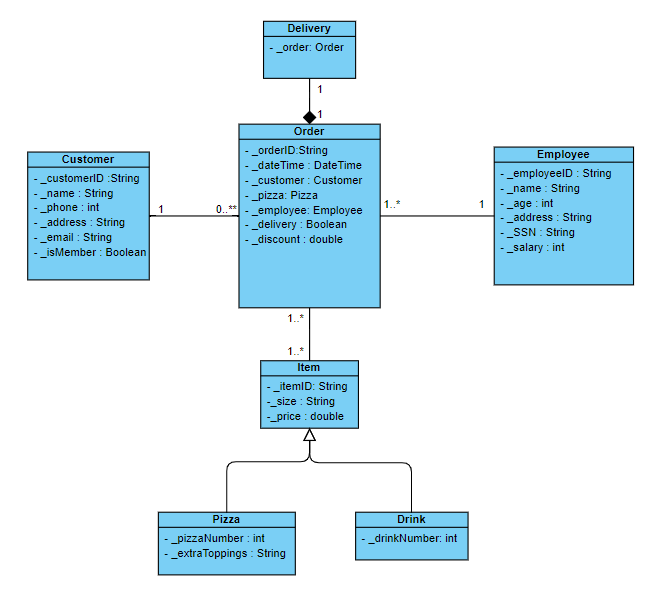
|  |  |
| --- | --- |
| Functional Requirements | Non Functional Requirements |
| R1.0 User order is taken. | R1.1 Good response Time of system  R1.2 Able to handle multiple-users at multiple levels  R1.3 Usability of User Dialogue |
| R2.0 User account is created | R2.1 Transparency of User information  R2.2 Security of user information |
| R3.0 User credentials validated and user logs in. | R3.1 Login system security  R3.2 User access level validated.  R3.3 Error prevention |
| R4.0 Menu Catalog displays. | R4.1 Correct information valid to user access is displayed.  R.4.2 Understandability of User dialogue. |
| R5.0 CRUD operations performable on Order. | R5.0 Affordance of user operations.  R5.1 Accurate and Correct reflection of operations. |

**Sprint 1 – Backlog. (27/9/2020 – 3/10/2020)**

|  |  |  |  |
| --- | --- | --- | --- |
| User Story | To Do | Doing | Done |
| As a customer I want to be able to order online | User dialogue loops (does not exit) upon completion. | User can create order using the User Dialogue.  User dialogue navigates user through different fields. |  |
| As a customer, I want to be able to choose what toppings I want on the pizza. |  |  | User can order pizza with extra toppings.  User can change selected extra toppings. |
| As a customer I want to be able to manage my order(shopping card) so I can add and remove/edit items. | User cannot submit an empty shopping cart |  | User can perform CRUD operations on shopping cart. |
| As a customer. I want to be able to register and apply for a club membership |  |  | User can perform registration.  Discount is applied to checkout |
| As a customer I want to be able to see my orders | System display only specific user order. |  | User can view their order. |

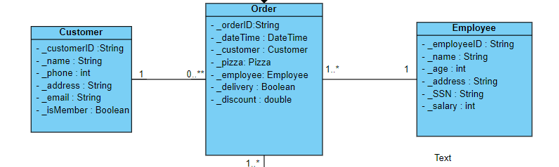
**Sprint 2 – Backlog (20/10/2020 – 24/10/2020)**

|  |  |  |  |
| --- | --- | --- | --- |
| User Story | To Do | Doing | Done |
| As a customer I want to be able to order online | User dialogue loops (does not exit) upon completion. | User can create order using the User Dialogue.  User dialogue navigates user through different fields. |  |
| As a customer I want to be able to manage my order(shopping card) so I can add and remove/edit items. | User cannot submit an empty shopping cart |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Domain Model

*Figure 1.1 Pizzeria Domain Model*

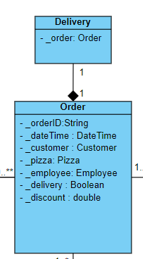
Domain model, done at the domain level to provide a brief overview of how the classes interact with each other.



*Figure 1.2 Customer Interaction with Employee via Order Class*

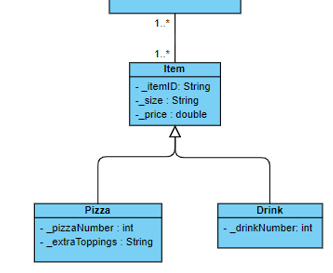
The domain model of the pizzeria shows how Customers interact with Employees within the application which is through the order system. There is no direct connection between Customers and Employees via ordering online. Similarly with the Item class which is unable to be directly interacted by a customer or employee without an order.

Composition - is done as if an instance of class Order that is tied to delivery is deleted, the instance of class delivery is deleted as well. This is in preparation for the chance that customer cancels delivery or deliver is canceled store side.



*Figure 1.2 Delivery Composition class*

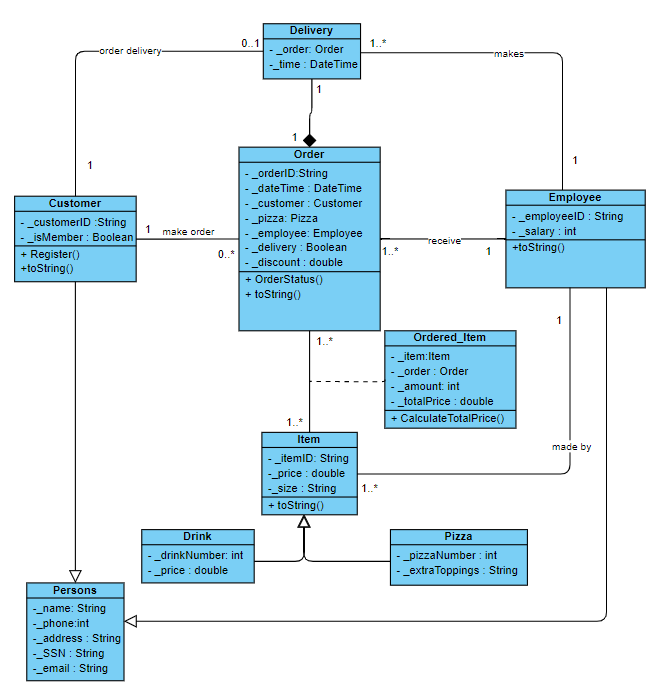
Generalisation – is done so the similar main attributes of the products in the menu can be inherited. It would be easier and more readable for the items to be generalised in the Domain Model. Should any extra variable be added unique to the Pizza or Drink child classes it would not affect the main class.



*Figure 1.3 Generalisation of Pizza and Drink Classl*

Note that the relationship here of Many to Many is difficult to implement and will be resolved in the Design Class Diagram.

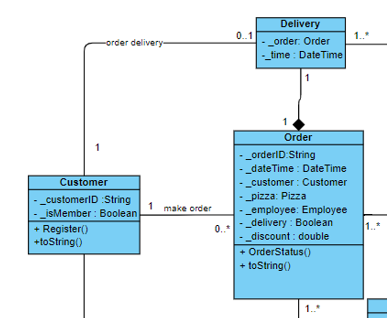
2.0 Design Class Diagram



*Figure 2.1 Pizzeria Design Class Diagram*

Relationships are explored more in depth in this Design Class Diagram. Methods are added to each class and extra associations are added to map the classes as how they would be in real life scenarios. Specifically, the 4 main classes of Customer, Order, Employee and Item will be explored.

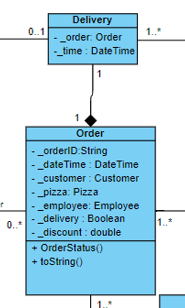
An update to the previous Domain Model, Customer and Employees are now child classes to Persons class. This is to avoid redundancy as Customer and Employee share the same attributes in some cases so it would be better to inheret these attributes from a parent class and force some similar attributes to the classes that they would otherwise not have in the domain model.



*Figure 2.2 Customer Relationships*

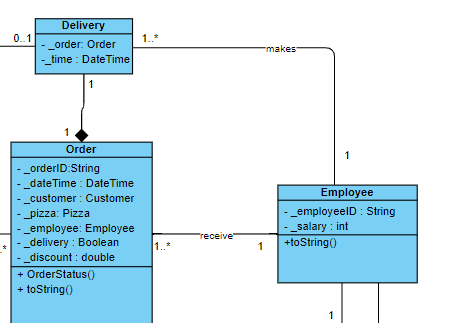
Relations of Customer class – 1 Customer can make 0 to many orders, and many orders can be made by only 1 customer. What this implies is that Each unique order can only belong to 1 customer. No 2 customers can have the same instance of an order. At the same time a Customer can exist within the system and not make any orders at all.

1 Customer can have 0 to 1 deliveries. This means thatmuch like an order, only 1 delivery can be made to any 1 customer at a time. Multiple customers cannot accept the same delivery and customers may not make a delivery at all in some cases.



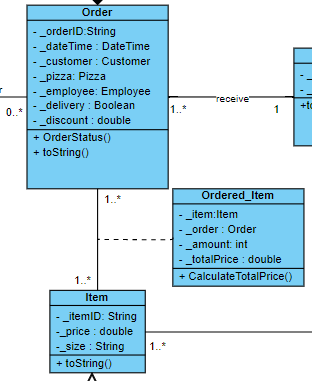
*Figure 2.2 Delivery and Order Classes*

Order class is in composition with Delivery class because as previously eplained , delivery class cannot exist without an instance of order class. The relationship of these two classes are one to one because only one order can be tied to one delivery at any given time.



*Figure 2.3 Employee Relationships*

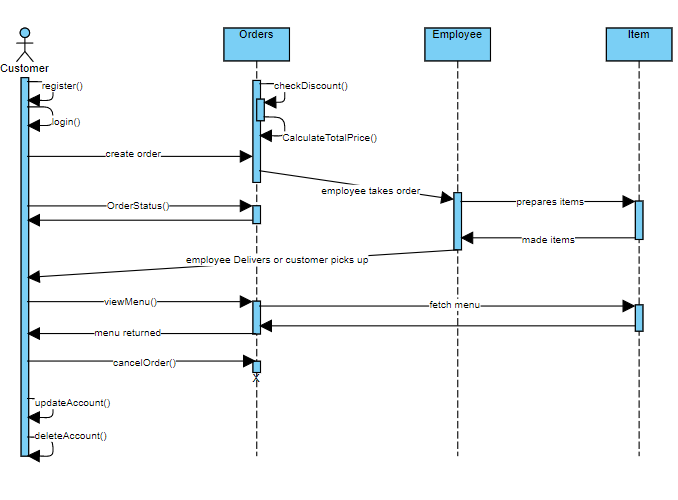
Order is in relationship with employee of 1..Many to 1 because Order must be completed by an employee. There is no employee that does not complete an order. At the same time , an employee can complete many orders and no same orders are given to more than 1 employee. Employee also is connected to Delivery with a 1..many to one in the same manner as order in that an employee must complete a delivery and an employee is able to complete more than 1 delivery at a time.



*Figure 2.3 Resolving Many to Many*

Item and order have a relationship of many to many. This means that Many items can belong to the Many different unique orders. Many to many relationships in a UML Diagram must be resolved as it is not realistic to implement. The way it is resolved is by having another class of Ordered\_Item with unique variable(s) specific to the class in this instance the Total Price.

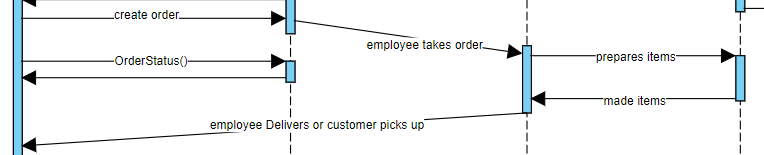
3.0 Sequence Diagram



*Figure 3.1 Customer Administration Sequence Diagram*

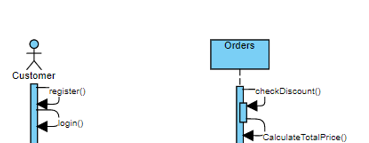
The purpose of sequence diagram is to view how processes run in unison with each other or show in sequence certain processes are handled between each other. Figure 3.1 shows how customers and the shopkeeper interact with the system in real time.

Some operations such as CRUD operations for Customers and Shopkeepers are mapped onto the sequence diagram.



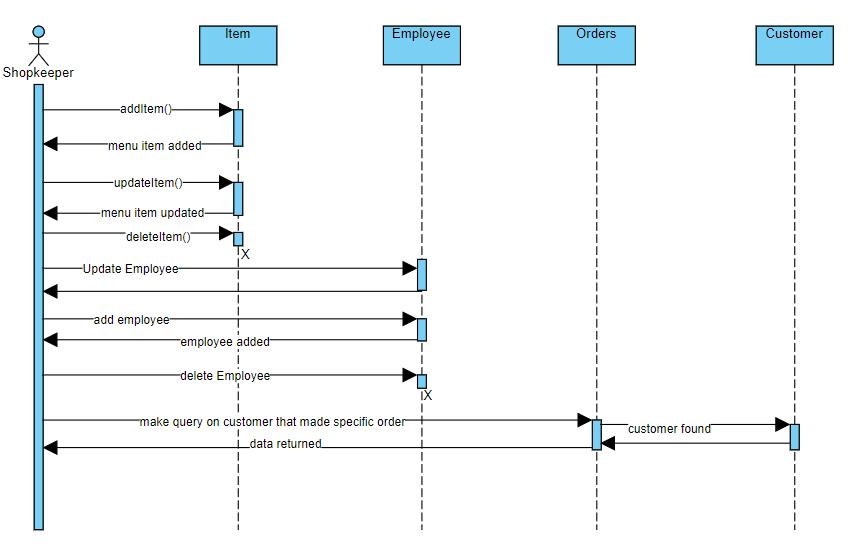
*Figure 3.1.1 Customer Pizza Ordering Sequence*

The sequence of customer placing order to when employee completing orders. Customer is able to check order status while message is being relayed. Message is also timed as employees are expected to receive the order and the order is supposed to be given to customer within a time frame.



*Figure 3.1.2 Self Returning Messages*

Some self returning messages and looped messages are used to imply that the system or the user will perform these processes and send the data back to themselves. For Check Discount(), system will be in constant loop checking to see if customer is eligible for discount and for Register() user is registering an account and sending the information back to itself.



*Figure 3.2 Pizza Menu Administration Sequence Diagram*