**Food Delivery Application**

* 1. **Background :**

Food delivery application is an online food ordering management system is designed to ease and enrich customer's delivery experience. Sometimes when we don't feel to cook or either want to go to the restaurants, food delivery application comes to the rescue. Our Re-engineered food delivery application serves as a forum between consumers and restaurants. It provides a platform where restaurants can register and upload the menus, payment receiving options and not think about operations, management and making the deliveries. This will help restaurants reach a broader audience and meet their profit goals. Restaurants are monitored from time to time based on user/customer feedback or ratings. users can use the application to order from the restaurant of their preference. The vendors (restaurant), the clients (customers or users) who would be ordering on the application and the delivery agent who would be hired to deliver the order are the three parties involved. The users can explore menus, ratings and prices of the restaurants and get to choose what they want. Different payment options such as cash, credit card, debit card and pay pal are incorporated to make the payment process flexible. Once the order is placed, the information is passed on to the restaurant and the delivery agent would be assigned to deliver the order to the customer. The status of the order can be tracked by the user right from the order being placed to delivery. This model offers a lot of ease and support to the restaurants and customers who do not have the resources to have their own delivery system in place.

* 1. **Problems with current system :**

We are re-engineering some part of the existing apps like UberEats, Yelp, etc to understand the business domains.We did not find any potential problems as they are optimized and have good operational efficiency.

* 1. **Data Requirements :**

Food delivery app that we are re-engineering greatly simplify the ordering process for both the customers as well as restaurants.We would like to store details like Customer information which stores details like Name,Address,Phone,Email and order details like Restaurant name,Food items,Total amount,Delivery agent ID etc. Restaurant details like Name,Address,Menu,Rating etc.Delivery agent details consists of Name ,Contact Number etc.

**Business rules :**

This food delivery applicationprovides delivery services from the restaurants which

are delivered with the delivery agents. The dated requirements are stated below.

* Information about customer that needs to be stored includes customers unique Id, Name, current address, phone and email. Every customer must provide their address for delivery. Each customer is uniquely identified by customer id. Two customers cannot not have the same customer id. Each customer can make many orders.Customer can pay with either credit card or debit card or cash.
* Database needs to store information about each order.Orders can consist of many items in a menu.Each order has a unique order id. Database stores orderid,orderdate and Total amount.
* Database needs to store information about each payment. Each payment is assigned a unique payment number. Database records the issue date and amount for each payment.
* Information about restaurant that needs to be stored such as RestaurantId, Restaurant name, Restaurant phone,Restaurant location
* The menu lists the items of food and its price.Database needs to store information about Menu which stores details like Menu Id, Menu description , Unitprice. Menu Id is unique to each menu.
* Database needs to store information about Delivery agent.Each agent has a unique AgentID.Delivery agent stores details like AgentID,Name,Phone and vehicle number.
* Database needs to record status of the order.The order feedback can be given by the customer on each order.
* Customer places orders to Restaurant. Database needs to store which customer placed each order. Each order is placed by exactly one customer. Each customer may place many orders.
* Each order is always paid for with one payment. Database needs to store which order each payment is for.Each payment can be paid by many payment types like cash,cards,digital cash etc.
* Only one type of restaurant can be picked for each order. Database needs to store which restaurant is served for each order.Restaurants offers many menu types to choose from each restaurant.
* The database needs to store what feedback is obtained for the order.One feedback is given on each order.
  1. **Templates**
* T1: Database needs to store information about each customer.
* T2: Database needs to store a unique Id for each customer(Primary key)
* T2: Database needs to store Name, Address, phone number,Email address of each customer.
* T1: Database needs to store information about each restaurant
* T2: Database needs to store a unique restaurant id for each restaurant  (Primary key)
* T2: Database needs to store Name, phone Number, address, area for each restaurant
* T1: Database needs to store information about each order
* T2: Database needs to store a unique  order Id of each order (Primary key)
* T3: Database needs to store  the orders placed by each customer
* cardinality relationship between customer and order is one to many .Each customer can place many orders and each order can be placed by only one customer.
* T3: Database needs to store the orders served by each restaurant
* cardinality relationship between restaurant and order is one to many .Each restaurant can serve many orders and each order can be served by only one restaurant.
* T2: Database needs to store  order date, total amount of each order.
* T1: Database needs to store information of the menu item
* T2: Database needs to store a unique Id for each menu item (Primary key)
* T2: Database needs to store Unit price and other information for each menu item
* T3: Database needs to store the the orders  placed from the each menu item
* cardinality relationship between menu and order is many to many .Each item can have many orders and each order can have many items in the menu.
* T4:Database needs to store the current price and Quantity of each order placed from each menu
* T1: Database needs to store information of each delivery agent
* T2: Database needs to store a unique Id for each agent(Primary key)
* T2: Database needs to store Name, Vehicle number, phone number for each agent
* T3: Database needs to store which order is assigned to the delivery agent
* cardinality relationship between delivery agent and order is one to many

Each agent can be assigned to deliver only one order and each order can be assigned

to many agents

* T1: Database needs to store information about order status
* T2: Database needs to store a unique Id for each order status (Primary key)
* T2: Database needs to store information about order status
* T3: Database needs to store the status of each order placed
* cardinality relationship between order status and order is many to many. Each order can be have many status(including the past and present status) and each status can have many orders.
* T4: Database needs to store the date and time of each order status for each order placed
* T1: Database needs to store information about each payment
* T2: Database needs to store a unique Id for each payment (Primary key)
* T2: Database needs to store status , Date of each payment
* T3: Database needs to store the payment for each order
* Cardinality relationship between payment and order is one to one .Each order can have only one payment each payment can have only one order.
* T1: Database needs to store information about payment type
* T2: Database needs to store a unique Id for each payment type (Primary key)
* T2: Database needs to store Additional infromation for each payment type
* T3: Database needs to store the paymeny type for each payment
* cardinality relationship between payment and payment type is one to many. Each payment can have only one payment type and each payment type  can have many payments.
* T1: Database needs to store information about orderfeedback
* T2: Database needs to store a unique Id for each orderfeedback (Primary key)
* T2: Database needs to store Feedback and rating of each orderfeedback
* T3: Database needs to store order feedback of each order.
* cardinality relationship order and order feedback is one to one. Each order can have only one feedback type and each feedback can be given for only one order.
  1. **Sample Data**

**Customer Table :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CustID** | **FName** | **Lname** | **Address** | **Phone** | **email** |
| 1001 | Frederich | Nietzsche | 345 VIEWRIDGE | 415-598-6609 | nietfred@gmail.com |
| 1002 | Jane | Austen | 490 BOLI RD. | 415-368-1223 | jane17@gmail.com |
| 1003 | Agatha | Christie | 9722 HAMILTON | 415-644-3341 | christ.agath@gmail.com |
| 1004 | Stephen | King | HILLVIEW MALL | 415-677-9312 | king.13@gmail.com |
| 1005 | Charles | Dickens | 574 SUYYYYY RD. | 408-996-2323 | charlesd@gmail.com |

**Restaurant table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RID** | **Name** | **Phone** | **Location** | **Address** |
| 101 | McDonald's | 720123145 | Highlands Ranch | 9235 S Broadway, Highlands Ranch, CO 80129 |
| 102 | Starbucks | 720157955 | Lucent & Town Center | 1519 Park Central Drive, Highlands Ranch, CO 80129 |
| 103 | Taco Bell | 720157956 | 7911 S Broadway | 7911 S Broadway, Littleton, CO 80122 |
| 104 | Subway | 720157957 | 9245 S Broadway | 9245 S Brdway, Highlands Ranch, CO 80129 |
| 105 | Corner Bakery | 720157958 | 2253 E Briarwood Ave | 2253 E Briarwood Ave, Centennial, CO 80122 |

**Payment table :**

|  |  |  |  |
| --- | --- | --- | --- |
| **PID** | **Status** | **Date** | **TypeId** |
| P1 | Pending | 5/2/2009 | 1 |
| P2 | Success | 5/6/2010 | 2 |
| P3 | Failed | 7/5/2009 | 4 |
| P4 | Payment processing | 4/4/2008 | 3 |

**Payment Type :**

|  |  |
| --- | --- |
| **TypeId** | **Description** |
| 1 | Online Payment |
| 2 | Credit Card |
| 3 | Debit Card |
| 4 | Wallet |

**Order table :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OrderId** | **Date** | **TotalAmount** | **CustId** | **RID** | **PID** | **AgentID** |
| 1234 | 5/6/2010 | $1,500 | 1001 | 103 | P2 | A1 |
| 4231 | 7/5/2009 | $320 | 1003 | 102 | P3 | A2 |
| 5512 | 4/4/2008 | $152 | 1004 | 105 | P4 | A3 |
| 6789 | 5/2/2009 | $722 | 1005 | 103 | P1 | A5 |
| 1158 | 5/1/2009 | $520 | 1005 | 104 | P1 | A2 |
| 3321 | 5/3/2010 | $333 | 1001 | 105 | P2 | A4 |

**OrderItem :**

|  |  |  |  |
| --- | --- | --- | --- |
| **MID** | **OrderId** | **CurrentPrice** | **Qty** |
| 1 | 1234 | $315 | 2 |
| 2 | 1158 | $3,500 | 3 |
| 3 | 5512 | $780 | 1 |
| 4 | 6789 | $310 | 5 |
| 5 | 1158 | $300 | 1 |
| 6 | 3321 | $589 | 4 |
| 7 | 4231 | $414 | 7 |
| 8 | 6789 | $115 | 2 |
| 9 | 4231 | $414 | 19 |
| 10 | 5512 | $3,570 | 52 |

**OrderStatus :**

|  |  |
| --- | --- |
| **StatusId** | **Description** |
| S1 | Waiting for order confirmation |
| S2 | Confirmed |
| S3 | Preparing Order |
| S4 | Waiting For delivery agent |
| S5 | On the way to pick up |
| S6 | Picked Up |
| S7 | Delivered |

**OrderCurrentStatus :**

|  |  |  |
| --- | --- | --- |
| **StatusId** | **OrderId** | **DateTime** |
| S1 | 1234 | 5/6/2010 |
| S2 | 4231 | 7/5/2009 |
| S4 | 5512 | 4/4/2008 |
| S4 | 6789 | 5/2/2009 |
| S5 | 1158 | 5/1/2009 |
| S6 | 3321 | 5/3/2010 |
| S7 | 1158 | 5/1/2009 |
| S3 | 1234 | 5/6/2010 |

**MenuItem :**

|  |  |  |  |
| --- | --- | --- | --- |
| **MID** | **Description** | **UnitPrice** | **RestuarantId** |
| 1 | Chicken Pomodori | 10.49 | 105 |
| 2 | Bagels | 1.29 | 105 |
| 3 | Chicken & Bacon Ranch Footlong Regular Sub | 9.29 | 104 |
| 4 | American Club Footlong Regular Sub | 9.99 | 104 |
| 5 | Nacho Fries Box | 6 | 103 |
| 6 | Supreme Taco Party Pack | 23.99 | 103 |
| 7 | Iced Caramel Macchiato | 4.95 | 102 |
| 8 | Caffè Latte | 3.85 | 102 |
| 9 | Spicy Chicken Sandwich Meal | 8.59 | 101 |
| 10 | Double Bacon Quarter Pounder with Cheese Meal | 11.03 | 101 |

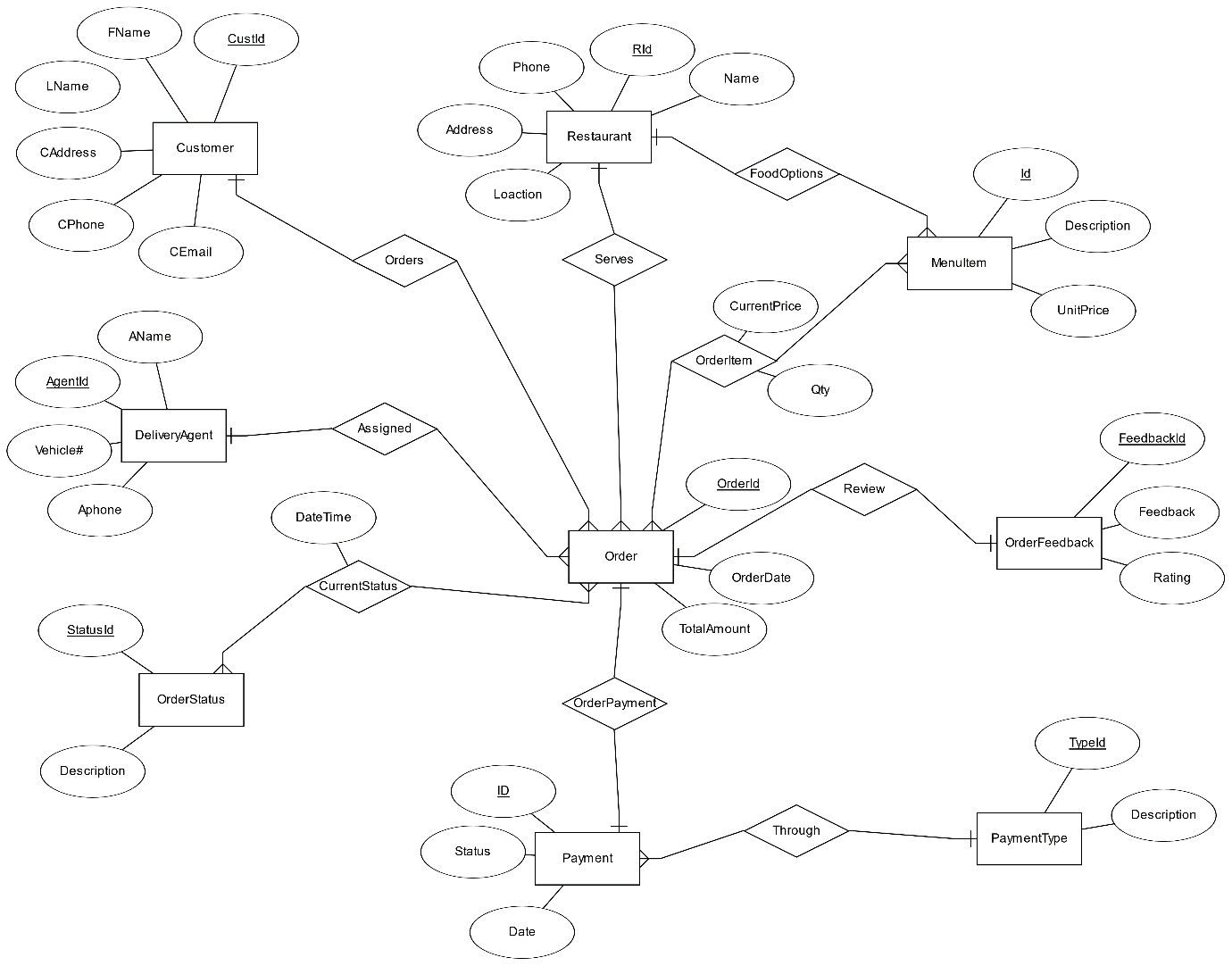
**OrderFeedback :**

|  |  |  |  |
| --- | --- | --- | --- |
| **FID** | **Feedback** | **Rating** | **OrderId** |
| F1 | Good food | 5 | 1234 |
| F2 | Crispy and hot | 4 | 4231 |
| F3 | Very spicy | 3 | 5512 |
| F4 | Food tastes like old stock | 2 | 6789 |
| F5 | Wrong food delivered | 1 | 1158 |
| F6 | Amazing food | 6 | 3321 |

**Delivery Agent :**

|  |  |  |  |
| --- | --- | --- | --- |
| **AgentId** | **Aname** | **Vehicle#** | **Aphone** |
| A1 | Michael | TA136 | 874-968-7432 |
| A2 | Steve | GH179 | 984-788-5521 |
| A3 | Rory | FF181 | 312-767-2314 |
| A4 | Stephen | CH785 | 726-662-1234 |
| A5 | George | BB321 | 333-222-1514 |

**ERD Diagram :**



**Explanation of Entities and Relationships:**

Customer: customer is the user that places an order. They are our clients.

Order: We place an order to the restaurant when customers places an order,

Restaurant : Stores information about each restaurant that serves. They are our clients.

Payment: Stores information about each payment we made.

Payment Type: Stores information about type of payment done.

Menu Item: Store  Stores information about the items in the menu.

Delivery agent: Stores information about each assigned delivery agent.

OrderStatus: Stores information about status of each order.

OrderFeedback: Stores information about feedback given for each order.

Status-of-order: when customer places an order.This records the status for which order until the order is delivered

Order-menuitem: This records what items from the menu are in eachorder.

**a) Converting into tables :**

Customer: (CustID, CFName, CLName,Caddress, CPhone, CEmail)

Restaurant: (RID, RPhone, Raddress, Rlocation, RName)

MenuItem: (MID, MenuDescription, UnitPrice, RID)

DeliveryAgent: ( AgentID, AName, Vehicle#, Aphone, OrderId)

OrderStatus: (StatusId, OrderDescription)

Payment: (PID, Status, Date, TypeId)

PaymentType: (TypeId, PayDesc)

OrderFeedback: (FeedbackId, Feedback, Rating, OrderId)

Order: (OrderId, OrderDate, TotalAmount, CustID, RID, PID)

StatusOfOrder: (StatusId, OrderId, DateTime)

OrderItem: (MID, OrderId, CurrentPrice, Qty)

**b) Convert all relationships**

**Customer - orders - order relationship : It is 1-n relationship.Adding the primary key (custid) of customer in order table.we get**

**Order : (**OrderId, OrderDate, TotalAmount, CustID)

Restaurant - serves - order relationship : This is a 1-n relationship. Add the primary key of restaurant entity to the many side (Order) as a foreign key.

Order: (orderid, Orderdate,totalamount, Rid)

Delivery agent - assigned - order : It is in 1-n relationship.Therfore adding the primary key (AgentId) of delivery agent table in order table

Order: (OrderId, OrderDate, TotalAmount, CustID, RID,AgentID)

Order - current status – order status relationship : It is in m-n relationship.Therefore creating an other table called OrderCurrentStatus

OrderCurrentStatus: (StatusId, OrderId, DateTime)

Order -OrderPayment- payment relationship : It is in 1-1 relationship.Therefore adding primary key of payment (PID) in ordertable

Order: (OrderId, OrderDate, TotalAmount, CustID, RID, PID, AgentID)

Payment through payment type : It is a n-1 relationship.Therefore adding primary key of payment type in payment table.we get

Payment: (PID, Status, Date, TypeId)

Order – Review – Feedback Relationship: It is in 1-1 relationship.Adding the primary key of order (orderid) in OrderFeedback table

OrderFeedback: (FeedbackId, Feedback, Rating, OrderId)

Order – OrderType - MenuItem relationship : It is in m-n relationship.Therefore creating an othertable called OrderItem

OrderItem: (MID, OrderId, CurrentPrice, Qty)

Restaurant – Foodoptions - Menu relationship : This is a 1-n relationship.Therefore adding the primary key of restaurant in MenuItem table.

MenuItem: (MID, MenuDescription, UnitPrice, RID)

**c) Final Conversion Result**

Customer: (CustID, CFName, CLName, Caddress, CPhone, CEmail)

FD: CustID -> CFName, CLName, Caddress, Cphone, CEmail

Restaurant: (RID, RPhone, Raddress, Rlocation, RName)

FD: RID -> Rphone, Raddress, Rlocation, RName

MenuItem: (MID, MenuDescription, UnitPrice, RID)

FD : MID -> MenuDescription, UnitPrice, RID

DeliveryAgent: ( AgentID, AName, Vehicle#, Aphone)

FD: AgentID -> Aname, Vehicle#, Aphone

OrderStatus: (StatusId, OrderDescription)

FD : StatusId -> OrderDescription

Payment: (PID, Status, Date, TypeId)

FD : PID -> Status, Date, TypeId

PaymentType: (TypeId, PayDesc)

FD : TypeId -> PayDesc

OrderFeedback: (FeedbackId, Feedback, Rating, OrderId)

FD : FeedbackId -> Feedback, Rating, OrderId

Order: (OrderId, OrderDate, TotalAmount, CustID, RID, PID, AgentID)

FD : OrderId -> OrderDate, TotalAmount, CustID, RID, PID, AgentID

OrderCurrentStatus: (StatusId, OrderId, DateTime)

FD : StatusId + OrderId -> DateTime

OrderItem: (MID, OrderId, CurrentPrice, Qty)

FD : MID + OrderId -> CurrentPrice, Qty

**d) All the above tables are normalized.**

**e) Final Result after normalization**

Customer: (CustID, CFName, CLName, Caddress, CPhone, CEmail)

Restaurant: (RID, RPhone, Raddress, Rlocation, RName)

MenuItem: (MID, MenuDescription, UnitPrice, RID) Foreign keys : RID

DeliveryAgent: ( AgentID, AName, Vehicle#, Aphone)

OrderStatus: (StatusId, OrderDescription)

Payment: (PID, Status, Date, TypeId) Foreign keys : TypeId

PaymentType: (TypeId, PayDesc)

OrderFeedback: (FeedbackId, Feedback, Rating, OrderId) Foreign keys : OrderId

Order: (OrderId, OrderDate, TotalAmount, CustID, RID, PID, AgentID) Foreign keys : CustID, RID, PID, AgentID

OrderCurrentStatus: (StatusId, OrderId, DateTime) Relationship attribute : DateTime

OrderItem: (MID, OrderId, CurrentPrice, Qty) Relationship attribute : CurrentPrice,Qty

**f). Check foreign keys**

There are eleven tables with ten foreign keys. Since number of foreign keys = number of tables -1, this satisfies the minimum number of foreign keys.

**Foreign keys :**

RID, TypeId, OrderId, CustID, RID, PID, AgentID, DateTime, CurrentPrice,Qty