**Online Food Ordering**

**Design Document**

November 26, 2017

By

Connor Twohey - 1001177282

Nicholas Truong - 1000985550

# **Table of Contents**

[**Table of Contents**](#_di4jf83nliyh) **1**

[**Introduction**](#_l4wd5cuq91mq) **2**

[**Requirements**](#_fbmi02ytvubh) **2**

[Complete Requirements](#_r14xgh4wow15) 2

[Missing Requirements](#_vsstiv2le3sg) 3

[**Assumptions**](#_y7459os53nrg) **3**

[**EER Diagrams**](#_iu7ulxukfmvc) **4**

[**Relational Database Schema**](#_ls1qiquscv08) **6**

[**Tables and Choices**](#_y4crmwtuu7ht) **7**

[Tables](#_2pguc5n4fq6d) 7

[Choices](#_mtaarpzc85cb) 10

# **Introduction**

This document gives detailed information on the design of an “**online food ordering**” system. The document includes requirements, assumptions, and diagrams about the Enhanced Entity Relationship (EER) schema of the database, which is then used for a relational database schema. The exact breakdown of the document can be seen from the Table of Contents from the previous page.

# **Requirements**

This section cover the requirements for the design of the DBMS. The section is split into Document Requirements and Missing Requirements. Document Requirements lists requirements from the Project document on BlackBoard, while Missing Requirement are requirements that are not listed or derived from the document. The following requirements are listed in tables with a unique ID (UID) basic Software Engineering principles for clarity.

## **Complete Requirements**

|  |  |
| --- | --- |
| **UID** | **Requirement** |
| FD00 | Database will keep track of CUSTOMERs. |
| FD01 | Each CUSTOMER shall have a unique IdNo (a unique integer generated by the system), a Name(first initial and last name only ), a PhoneNo (a string of 12 characters), an Address (a string of MAX 256 characters), an Email (a unique string with a max of 100 characters), a UserName (a unique string of between 8‐12 characters), a Password (a string of between 8‐12 characters), and CreatedDate (Datetime). |
| FD02 | The database will keep track of PRODUCTs(food) available for ordering, which shall be categorized based on their type. There shall be five main types: APPETIZER, SALAD, BEVERAGE, MAINDISH and DESSERT. |
| FD03 | Each PRODUCT (FOOD) shall have a ProductID (a unique number for each product – assume it is a number 1001, 1002, 1003, ...), Description ( a description for each food), Product\_Image (an image for each food) |
| FD04 | The database will keep track of the current (active) ORDERS. |
| FD05 | The database will keep track of available PRODUCTS. |
| FD06 | For each ORDER, the information kept shall include the specific PRODUCT and CUSTOMER as well as the CUSTOMER\_PAYMENTS. OrderDate, and total price, which is a derived value that can be calculated from the other information. |

## **Missing Requirements**

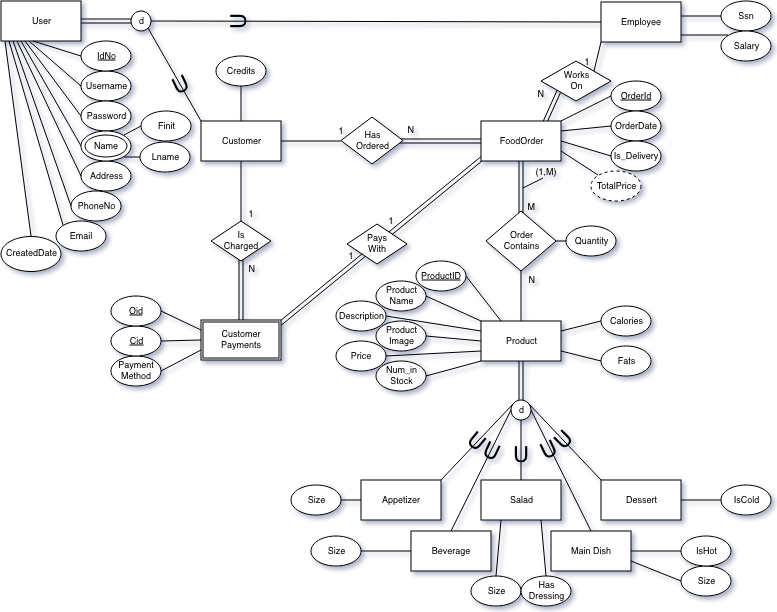
|  |  |  |
| --- | --- | --- |
| **UID** | **Requirement** | **Assumption** |
| FD07 | The database will keep track of CUSTOMER\_PAYMENTS, which is specific to both CUSTOMER and ORDERs. | There can only be one payment for each order. A bill cannot be split with cash and credit or amongst multiple customers. |
| FD08 | The database keeps track of the number in stock of a product. | Derived from FD05. The number of items of a specific product is kept track of in the DBMS. |
| FD09 | Payment information like credit/debit card numbers are not stored in the database. | Protecting sensitive customer information like credit card numbers is outside the scope of the project. |
| FD10 | Orders shall be pickup or delivery. | There is a storefront or restaurant where orders are picked up or delivered from. |
| FD11 | EMPLOYEEs will handle ORDERS and maintain the database. | An employee of the business associated with the DBMS handles individual orders. |
| FD12 | Total price will be a derived value from all PRODUCTs listed in order. | PRODUCT contains price value. |
| FD13 | Inactive orders will be removed from the database order is delivered. | Derived from FD04. Only active orders are tracked in the database. |
| FD14 | There can only be one customer listed for each order. | Only one customer can be listed for an order. |
| FD15 | Different types of products are treated differently. | Different categories of products have attributes that differentiate themselves from each other. |
| FD16 | First initial and last name are stored separately. | Name is a composite value with first initial and last name. |

# **Assumptions**

Assumptions:

1. Each product has unique product name.
2. Sales tax is not accounted for in DBMS design. Maybe accounted for in the website.
3. One payment per order. Cannot split payments with cash and credit.
4. One customer per order.
5. Showing available product is done by tracking the number of product in stock.
6. Important payment information like credit/debit card number are not stored in the database.
7. There is a storefront or restaurant that this database is representing, thus order can be pickup or delivery.
8. Products are edible and have a nutritional value.

# **EER Diagrams**

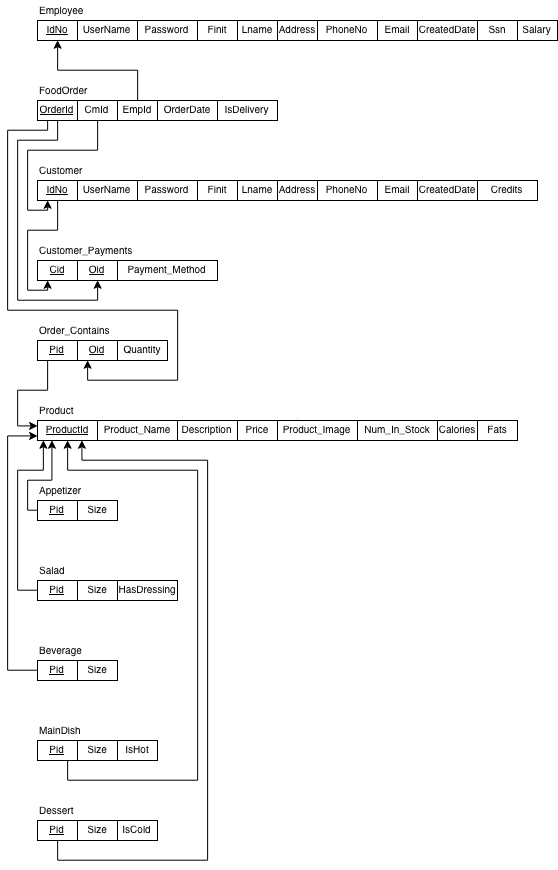


As per suggestion, Employee and Customer entities are subclasses to the User subclass. Customer has a 1:N relationship with FoodOrder to represent for one customer ordering multiple orders. Customer\_Payments have a 1:1 relationship with FoodOrder as only one payment can be made for one order and cannot be split. Customer\_Payments is a weak entity that has a 1:N relationship Customer with foreign keys referencing Customer’s and FoodOrder’s primary keys.

Employee entity tracks Ssn and salary. FoodOrder instances are headed by an Employee, which is represented in the 1:N Works\_On relationship with FoodOrder. FoodOrder has total participation as all orders must have an employee heading it. Employees can serve multiple FoodOrders.

All FoodOrders must have at least one Product, which is represented by the M:N Order\_Contains relationship. An order must have at least one product in the relationship represented by the structural constraint (1,M). An order can have multiples of the same product, which is represented by the Quantity attribute. Product must have unique names as customers identify products by name and not by ProductID. It also contains Num\_In\_Stock attribute so that the database can track if the product is available. Product has multiple disjoint subclasses called Appetizer, Beverage, Salad, Main\_Dish, and Dessert each with at least one attribute to differentiate one another. All products must be listed as one of the subclasses.

# **Relational Database Schema**



# **Tables and Choices**

## **Tables**

The following is code generated by phpMyAdmin’s SQL. Also provided with this document is an SQL file of the database.

CREATE TABLE `Appetizer` (

`Pid` int(11) NOT NULL COMMENT 'References Product',

`Size` char(1) NOT NULL DEFAULT 'S' COMMENT 'Can only be ''S'', ''M'', or ''L'''

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Beverage` (

`Pid` int(11) NOT NULL COMMENT 'References Product',

`Size` char(1) NOT NULL DEFAULT 'S' COMMENT 'Can only be ''S'', ''M'', or ''L'''

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Customer` (

`IdNo` int(11) NOT NULL,

`UserName` varchar(12) NOT NULL COMMENT '8 to 12 charcters',

`Password` varchar(12) NOT NULL COMMENT '8 to 12 characters',

`Finit` char(1) NOT NULL COMMENT 'First initial',

`Lname` varchar(32) NOT NULL COMMENT 'Last name',

`Address` varchar(256) NOT NULL,

`PhoneNo` char(12) NOT NULL COMMENT 'Dashes are stored. US standard phone number.',

`Email` varchar(100) NOT NULL,

`CreatedDate` datetime NOT NULL COMMENT 'Date of creation for customer.',

`Credits` int(11) NOT NULL COMMENT 'Tracks customer loyalty'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Customer\_Payments` (

`Oid` int(11) NOT NULL COMMENT 'References Order',

`Cid` int(11) NOT NULL COMMENT 'References Customer',

`Payment\_Method` varchar(100) NOT NULL COMMENT 'Can be credit, debit, or cash.'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Dessert` (

`Pid` int(11) NOT NULL COMMENT 'References Product',

`IsCold` tinyint(1) NOT NULL COMMENT 'Cold if true'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Employee` (

`IdNo` int(11) NOT NULL,

`UserName` varchar(12) NOT NULL COMMENT '8 to 12 charcters',

`Password` varchar(12) NOT NULL COMMENT '8 to 12 characters',

`Finit` char(1) NOT NULL COMMENT 'First initial',

`Lname` varchar(32) NOT NULL COMMENT 'Last name',

`Address` varchar(256) NOT NULL,

`PhoneNo` char(12) NOT NULL COMMENT 'Dashes are stored. US standard phone number.',

`Email` varchar(100) NOT NULL,

`CreatedDate` datetime NOT NULL COMMENT 'Date of creation for customer.',

`Ssn` char(9) NOT NULL COMMENT 'Social security number',

`Salary` int(11) NOT NULL COMMENT 'Annual salary'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `FoodOrder` (

`OrderId` int(11) NOT NULL,

`CmId` int(11) NOT NULL COMMENT 'References Customer',

`EmpId` int(11) NOT NULL COMMENT 'References Employee',

`OrderDate` datetime NOT NULL COMMENT 'Time order was made.',

`Is\_Delivery` tinyint(1) NOT NULL COMMENT 'Delivery if true, pickup if false'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `MainDish` (

`Pid` int(11) NOT NULL COMMENT 'References Product',

`Size` char(1) NOT NULL DEFAULT 'S' COMMENT 'Can only be ''S'', ''M'', or ''L''',

`IsHot` int(11) NOT NULL COMMENT 'Hot if true. Not if false.'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Order\_Contains` (

`Pid` int(11) NOT NULL COMMENT 'References Product',

`Oid` int(11) NOT NULL COMMENT 'References Order',

`Quantity` int(11) NOT NULL DEFAULT '1' COMMENT 'At least one'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Product` (

`ProductId` int(11) NOT NULL,

`Product\_Name` varchar(32) NOT NULL,

`Description` varchar(256) NOT NULL,

`Price` decimal(10,2) NOT NULL COMMENT 'In Dollars',

`Product\_Image` blob NOT NULL,

`Num\_In\_Stock` int(11) NOT NULL COMMENT 'Tracks stock of product',

`Calories` int(11) NOT NULL,

`Fats` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Salad` (

`Pid` int(11) NOT NULL COMMENT 'References Product',

`Size` char(1) NOT NULL DEFAULT 'S' COMMENT 'Can only be ''S'', ''M'', or ''L''',

`HasDressing` tinyint(1) NOT NULL COMMENT 'Dressing if true. Else false.'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `Appetizer`

ADD PRIMARY KEY (`Pid`);

ALTER TABLE `Beverage`

ADD PRIMARY KEY (`Pid`);

ALTER TABLE `Customer`

ADD PRIMARY KEY (`IdNo`);

ALTER TABLE `Customer\_Payments`

ADD PRIMARY KEY (`Oid`,`Cid`),

ADD KEY `cm\_pay` (`Cid`);

ALTER TABLE `Dessert`

ADD PRIMARY KEY (`Pid`);

ALTER TABLE `Employee`

ADD PRIMARY KEY (`IdNo`);

ALTER TABLE `FoodOrder`

ADD PRIMARY KEY (`OrderId`),

ADD KEY `cm\_order` (`CmId`),

ADD KEY `emp\_order` (`EmpId`);

ALTER TABLE `MainDish`

ADD PRIMARY KEY (`Pid`);

ALTER TABLE `Order\_Contains`

ADD PRIMARY KEY (`Pid`,`Oid`),

ADD KEY `od\_cd` (`Oid`);

ALTER TABLE `Product`

ADD PRIMARY KEY (`ProductId`),

ADD UNIQUE KEY `Product\_Name` (`Product\_Name`);

ALTER TABLE `Salad`

ADD PRIMARY KEY (`Pid`);

ALTER TABLE `Customer`

MODIFY `IdNo` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `Employee`

MODIFY `IdNo` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `FoodOrder`

MODIFY `OrderId` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `Product`

MODIFY `ProductId` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `Appetizer`

ADD CONSTRAINT `app\_prod\_key` FOREIGN KEY (`Pid`) REFERENCES `Product` (`ProductId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `Beverage`

ADD CONSTRAINT `bev\_prod` FOREIGN KEY (`Pid`) REFERENCES `Product` (`ProductId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `Customer\_Payments`

ADD CONSTRAINT `cm\_pay` FOREIGN KEY (`Cid`) REFERENCES `Customer` (`IdNo`),

ADD CONSTRAINT `od\_pay` FOREIGN KEY (`Oid`) REFERENCES `FoodOrder` (`OrderId`);

ALTER TABLE `Dessert`

ADD CONSTRAINT `ds\_prod` FOREIGN KEY (`Pid`) REFERENCES `Product` (`ProductId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `FoodOrder`

ADD CONSTRAINT `cm\_order` FOREIGN KEY (`CmId`) REFERENCES `Customer` (`IdNo`),

ADD CONSTRAINT `emp\_order` FOREIGN KEY (`EmpId`) REFERENCES `Employee` (`IdNo`);

ALTER TABLE `MainDish`

ADD CONSTRAINT `md\_prod` FOREIGN KEY (`Pid`) REFERENCES `Product` (`ProductId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `Order\_Contains`

ADD CONSTRAINT `od\_cd` FOREIGN KEY (`Oid`) REFERENCES `FoodOrder` (`OrderId`),

ADD CONSTRAINT `pd\_cd` FOREIGN KEY (`Pid`) REFERENCES `Product` (`ProductId`);

ALTER TABLE `Salad`

ADD CONSTRAINT `salad\_prod` FOREIGN KEY (`Pid`) REFERENCES `Product` (`ProductId`) ON DELETE CASCADE ON UPDATE CASCADE;

## **Choices**

Customer and Employee were made into separate tables, as it is expected for Employees and Customer to be treated differently by the system. Since Employees are expected to deliver or lead orders in the system, Employee logins will be onsite. Offsite would be through company websites and not customer website.

FoodOrder has foreign keys from Employee and Customer due to the 1:N relationships Has\_Ordered and Works\_On.

Order\_Contains was a M:N relationship that was made into a table that references primary keys from FoodOrder and Product. This was done following instructions given by the book and slides.

Product was made into its own table while all subclasses references the ProductId with additional attributes specific to each subclass. This was done so that Order\_Contains can reference a single Product from the Product table. This is also done so that ProductId’s will be unique across all subclasses and not unique for each subclasses.

## **Addendum**

As of November 21st, there have been some changes made to the original table and schema. These changes were made due to time constraints and they were also made in response to new information about certain requirements of the functionalities of the application. We found out that an Order only needed to contain one product. This means that the Product-Order relationship becomes a 1:N relationship. This is the only change in the actual EER diagram. However, the Order\_Contains table (which was originally meant to represent a M:N relationship between Product and Order) was dropped, as it was no longer needed. Also, we added an attribute to Order, which contains a Product ID referencing the Product relation. Also, we added another attribute to Customer\_Payments, TotalPrice, which is just a value derived from the Product relation. This was done in order to simplify certain aspects of the application.