LiveLong Meal Delivery System

Rakshit Viswanatham (rakshitm@buffalo.edu, 50012274)
Deepanshu Yadav (dyadav@buffalo.edu, 50321285)
Pooja Sharma (poojaomp@buffalo.edu, 50322095)

CSE562

1 Problem Statement

The project is aimed to serve as a database for an online Food Ordering platform. The main purpose is to provide customers with the facility of ordering meals for a week or so, or even meals for one time from top restaurants or any specialized home cook without visiting multiple websites.
The project has mainly two different sets of users identified as customers and vendors. Vendor can either be a restaurant or a home cook. Vendor creates and uploads the menu on the system. Customers can login into the system, view the menu and place orders after looking into the menu. Delivery will be generated based on order. Customers can also win loyalty reward points for each order placed through the website.
There will be an option to choose delivery mode which can be either self pickup or delivery through the delivery executive. There will also be an option where customers can pre-order meals from various restaurants and can have it delivered later at their convenience.

Choosing database over excel for this project:
While spreadsheets are useful for numeric and text values in relatively low volume, databases serve better purpose for numeric and text values as well as bigger files like images and documents. In this problem, we need to store menu images as well as other images like logos.
Compared to databases, spreadsheets might require a large amount of hard-drive space for data storage and finding required information among millions of rows would be much more accessible and fast using querying tools in a database.
Databases are more secure as they provide centralized data storage and offer better security. User permissions can be assigned to view data, edit data, and restrict access to privileged information.

2 Target Users

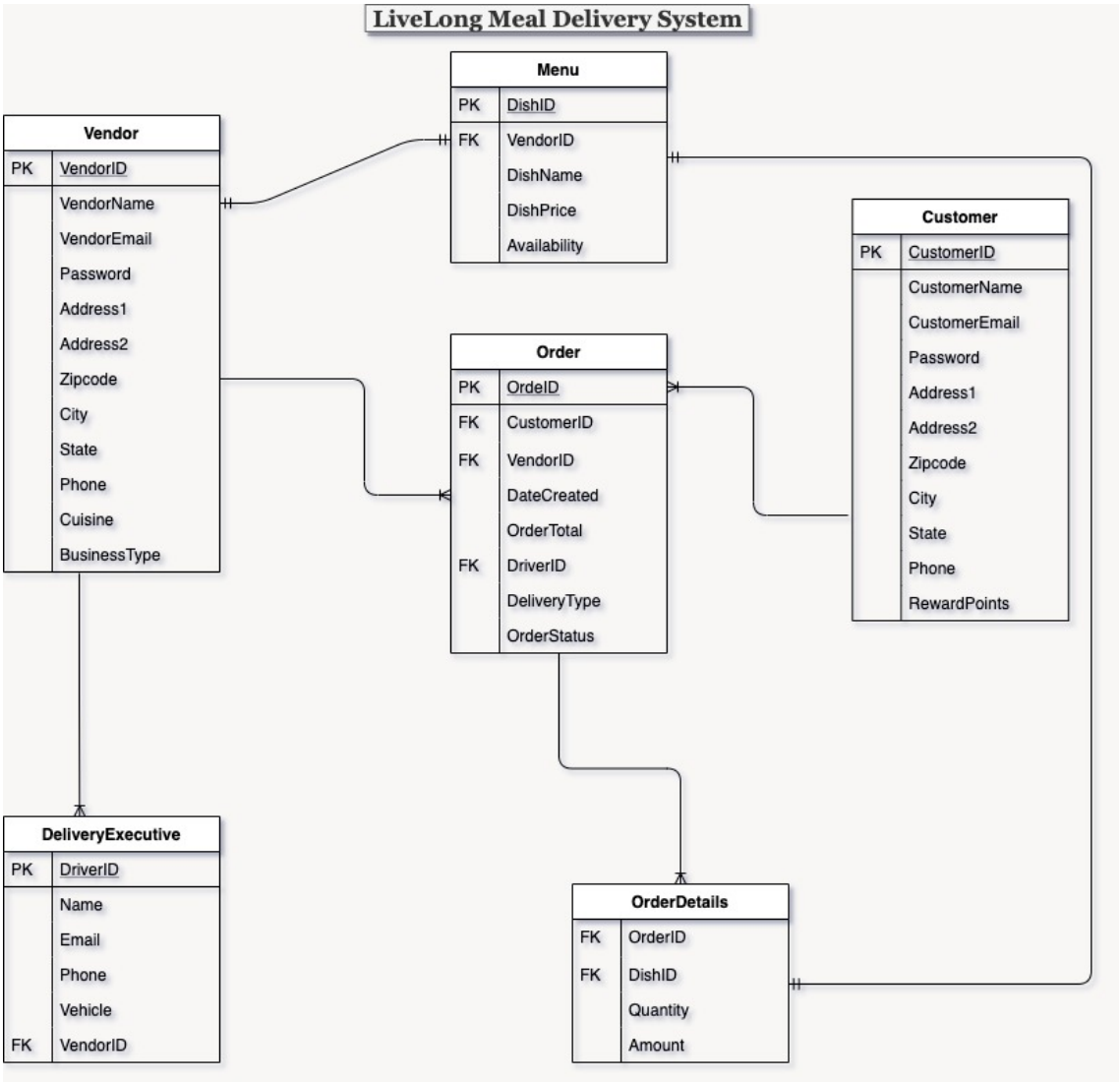
- End users
  - Business owners: who will have their menus, their delivery/pickup options, delivery executives, and orders.
  - Customers: who can order online, see their previous orders, access to coupons and their loyalty rewards program.
  - Delivery executives: who will need address, drop off date and time and the order details for the order to be delivered.
- Administrator: The company might appoint a team for managing the database. For scope of this project, we will be the administrators.

3 List of Relations

3.1 Relations/Tables

- Vendor
- Menu
- Customer
- Orders
- OrderDetails
- Driver

3.2 Schema


LiveLong Meal Delivery System
Menu
PK DishID
FK VendorID
DishName
DishPrice
Availability
Vendor
PK VendorID
VendorName
VendorEmail
Password
Address1
Address2
Zipcode
City
State
Phone
Cuisine
BusinessType
Customer
PK CustomerID
CustomerName
CustomerEmail
Password
Address1
Address2
Zipcode
City
State
Phone
RewardPoints
Order
PK OrdeID
FK CustomerID
FK VendorID
DateCreated
OrderTotal
FK DriverID
DeliveryType
OrderStatus
DeliveryExecutive
PK DriverID
Name
Email
Phone
Vehicle
FK VendorID
OrderDetails
FK OrderID
FK DishID
Quantity
Amount

Vendor

Attribute	DataType	Purpose
VendorID (PK)	Integer (Not null)	To uniquely identify vendors
VendorName	varchar (Not null)	Name of the business
VendorEmail	varchar (Not null)	To serve as username for login and authentication
Password	varchar (Not null)	For authentication when logging in
Address1	varchar (Not null)	Street Address
Address2	varchar	Optional
City	varchar (Not null)	City
State	varchar (Not null)	State
Zip	Integer (Not null)	Zip code of address
Phone	Integer (Not null)	Contact number of the business
Cuisines	varchar	Type of cuisine available
BusinessType	varchar (Not null)	Whether it's restaurant or a home based business

Customer

Attribute	DataType	Purpose
CustomerID (PK)	Integer (Not null)	To uniquely identify the customers
CustomerName	Varchar (Not null)	Name of the customer
Email	Varchar (Not null)	To serve as username for login and authentication
Password	Varchar (Not null)	For authentication when logging in
Address1	Varchar (Not null)	Street Address
Address2	varchar	Optional
City	Varchar (Not null)	City
State	Varchar (Not null)	State
Zip	Integer (Not null)	Zip code of address
Phone	Integer (Not null)	Contact number of the customer
RewardPoints	Integer (default 0)	Reward points earned by the customer

Menu

Attribute	DataType	Purpose
DishID (PK)	Integer (Not null)	To uniquely identify the dishes
VendorID (FK)	Integer (Not null)	To link the dish with the vendor
DishName	varchar (Not null)	Name of the dish
DishPrice	Float (Not null)	Price charged by the vendor for the dish
Availability	Boolean (Not null)	Whether the dish is available right now

Orders

Attribute	DataType	Purpose
OrderID	Integer (Not null)	To uniquely identify the orders
CustomerID (FK)	Integer (Not null)	To link the order with the customer
VendorID (FK)	Integer (Not null)	To link the order with the vendor
DateCreated	Date (Not null)	When the order is created
OrderTotal	Float (Not null)	Total amount in the order
CouponID (FK)	Integer	Which coupon is applied with the order
DriverID (FK)	Integer	To which delivery executive the order is assigned
DeliveryType	Char (Not null)	Whether it's delivery (D) or pickup (P)
OrderStatus	Boolean (Not null)	Is the order delivered or not

OrderDetails

Attribute	DataType	Purpose
OrderID (FK)	Integer (Not null)	To link this particular food item with the order
DishID (FK)	Integer (Not null)	To link this particular food item with the order
Quantity	Integer (Not null)	Quantity of this particular dish ordered
Amount	Float (Not null)	Total amount for this dish

Driver

Attribute	DataType	Purpose
DriverID (PK)	Integer (Not null)	To uniquely identify the delivery executive
Name	Varchar (Not null)	Name of the delivery executive
Email	Varchar (Not null)	As a part of contact info of the delivery executive
Phone	Integer (Not null)	As a part of contact info of the delivery executive
Vehicle	Varchar (Not null)	The vehicle used for delivery
VendorID (FK)	Integer (Not null)	To link the delivery executive with the vendor

3.3 Primary and Foreign Keys

3.3.1 Primary Keys

- Vendor: VendorId (To uniquely identify vendors)
- Menu: DishId (To uniquely identify dishes)
- Customer: CustomerId (To uniquely identify customers)
- Orders: OrderId (To uniquely identify orders)
- OrderDetails: (OrderID, DishID) will together form Primary key as a particular dish will be unique in an order
- Driver: DriverId (To uniquely identify delivery executives)

3.3.2 Foreign Keys

- Vendor: None
- Menu:

  - VendorID: On Delete Cascade
    (To link the dishes with the vendors)

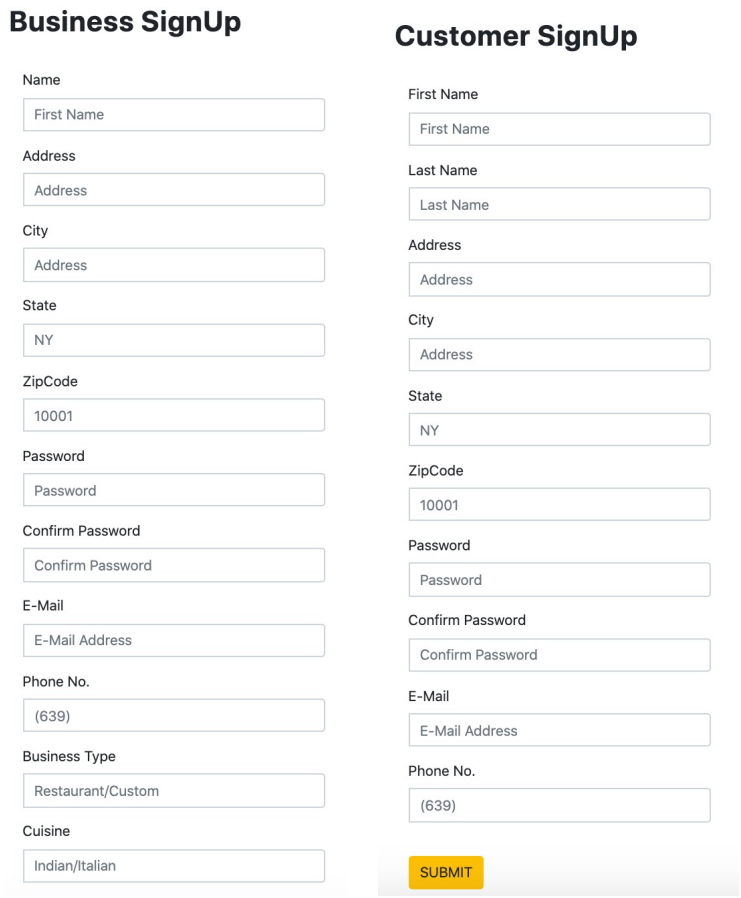
- Customer: None
- OrderHeader:
  - CustomerID: no action
    (To link the order with customer)
  - VendorID: On Delete Cascade
    (To link the order with vendor)
  - DriverID: On Delete set null
    (To link the order with Delivery executive it is assigned to)
- OrderDetails:
  - OrderID: On Delete Cascade
    (To link the order details with the order)
  - DishID: On Delete no action
    (To link the order details with the dish)
- Driver:
  - VendorID: On Delete Cascade
    (To link the delivery executive with the business they are associated with)

4 Web Interface

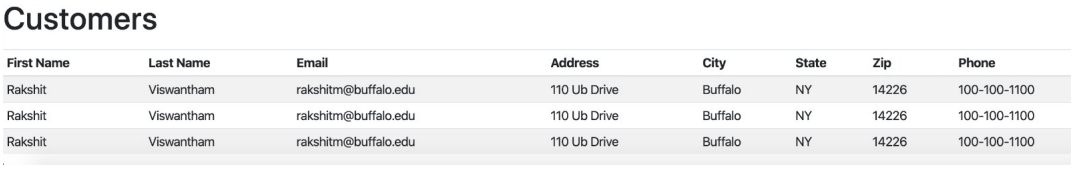
This section contains the rough sketches of the UI interface that will be used for login, creating a new customer or for a business to sign up. In addition, we have added few sketches as to how the results will be displayed.

The following image is the Login Screen, where a customer or a business can use their email and saved password to login and view their respective screens. The next UI Screen is the one where the Business can sign up and be part of the system, they will enter all the required details and then they will be included in the list of results by the customer later on when they are searching

in their zipcode. Next to it is the Customer Sign up where the customer will sign up with their details and then can therefore access the different places to order from.



The following image shows the list of customers, that can viewed by a business as to who their customers are or an administrator who want customers in a city.



The following image shows the places a customer can order from the relevant details that could be used to make the decision as to where to order from.

Places To Order

Name	Type	Address	City	State	Phone
Dosa Palace Click To Order	Indian	110 Ub Drive	Buffalo	NY	100-100-1100
Olive Garden Click To Order	Italian	110 Ub Drive	Buffalo	NY	100-100-1100
Eat Rite Foods Click To Order	American	110 Ub Drive	Buffalo	NY	100-100-1100

The following image shows all the orders a customer as placed in a specific period of time. They can from where the order was and when was it ordered. In addition, they can see the delivery/pickup details and everything.

Orders

Place	Date	Total	Type	Date of Delivery/Pickup	Time
Dosa Palace	1/13/2020	$45.67	Delivery	1/14/2020	3:00PM
Olive Garden	1/14/2020	$30.26e	Pickup	1/20/2020	1:00PM
Eat Rite Foods	1/15/2020	$55.66	Delivery	1/15/2020	1:00PM

Order Details

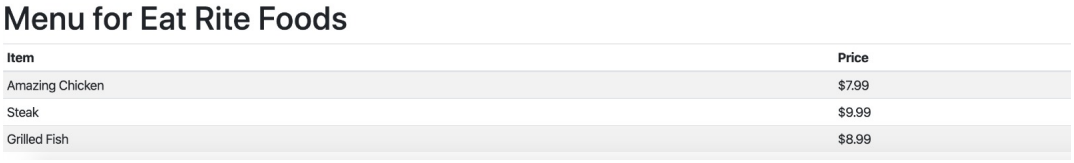
Order ID: 289329
Order Date: 1/13/2020
Order Total: $55.66
Order Payment Method: MasterCard 9797
Delivery Method: Delivery
Delivery Date/Time 1/15/2020 1:00PM

Order Items

Description	Quantity	Unit Price	Total Price
Grilled Fish	4	8.99	35.96
Amazing Chicken	4	7.99	31.96
Steak	4	10.99	35.96

The image above can be used to show what the specific order contained, like further details with the items ordered and their specific prices and quantity ordered.

The last image below is a sketch of how the menu would look and how they can see what they want and add that to the order.



Menu for Eat Rite Foods

Item	Price
Amazing Chicken	$7.99
Steak	$9.99
Grilled Fish	$8.99

All the above images are used as a foundation and rough sketch for how the UI will look like for views users and situations. This can potentially change based on the permissions and what can a single person view.

5 Data

Our data will be self populating. We will create test businesses with test menus. It will also have test customers, delivery executives that will be created. We will use `https://www.mockaroo.com/` to mock create as much data as possible and rest will be self-created.

6 Task 2

6.1 Functional Dependencies

- Vendor
  - vendorid → vendorname, vendoremail, vendorpassword, vendorAddress1, vendorAddress2, zipcode, vendorPhone, cuisine, businessType
  - zipcode → city, state
- Menu
  - dishid → dishstatus, dishName, dishPrice
- Customer
  - customerId → customername, customeremail, customerpassword, customerAddress1, customerAddress2, zipcode, customerPhone, rewardPoints
  - zipcode → city, state
- Orders
  - orderId → dateCreated, orderTotal, deliveryType, orderStatus
- OrderDetails
  - orderId, dishId → quantity, amount
- Driver
  - driverId → drivername, driveremail, driverPhone, driverVehicle

6.2 Transforming to BCNF

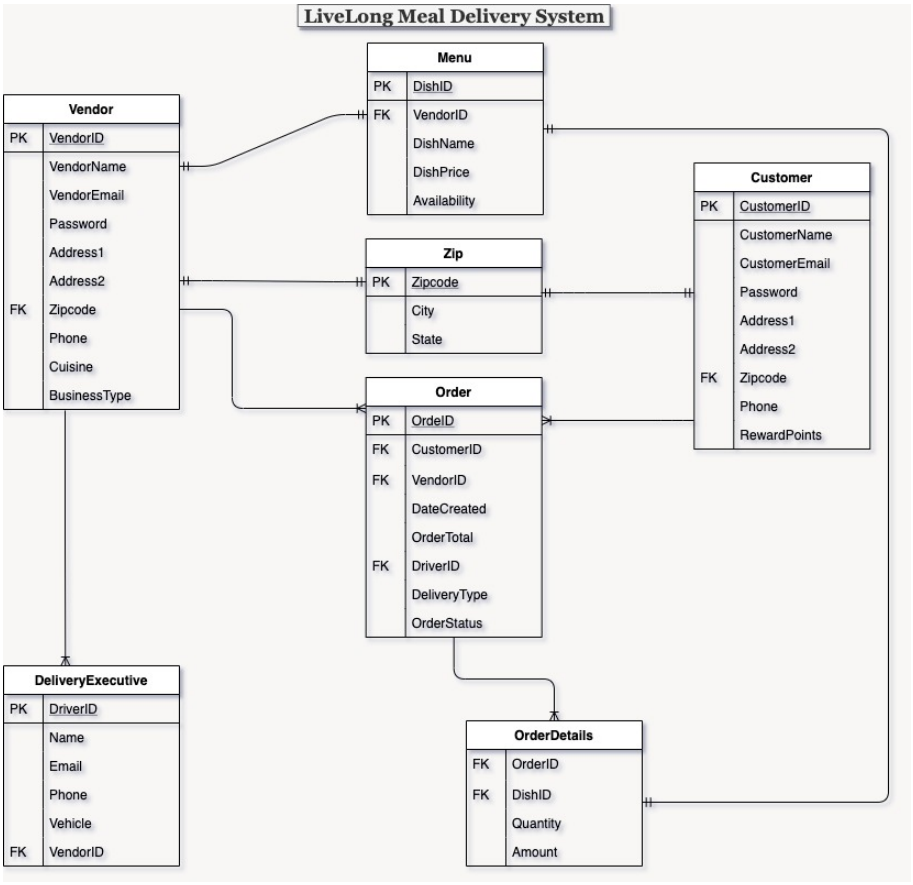
Tables Customer and Vendor are not in BCNF since they contain FDs zipcode → city, state. Thus, we will create another table ZIP with columns Zipcode, city and state where Zipcode is the primary key. This Zipcode will be a foreign key to Vendor and Customer table.

All the other tables are in BCNF as all the attributes are dependent on the primary key.

ZIP

Attribute	DataType	Purpose
Zipcode (PK)	Integer (Not null)	All the zipcodes
City	Varchar (Not null)	Name of the city corresponding to Zipcode
State	Varchar (Not null)	Name of the State corresponding to Zipcode

6.3 Updated Schema



7 Query Execution Analysis

The query which is taking the most time to execute is as follows:

```
SELECT * FROM (
      SELECT v.vendorid as id, v.vendoremail as email,
      v.vendorpassword as password, 'vendor' as type
      FROM vendor v
      UNION
      SELECT c.customerid as id, c.customeremail as email,
      c.customerpassword as password, 'customer' as type
      FROM customer c) AS t
WHERE t.email = 'deep123@gmail.com';
```

Here, we are trying to check if the email ID entered by user on Login Page is a customer or vendor. We scan the vendor table first searching for the email ID and if we don't find it in Vendor table, we scan the Customer table for the same. Below is the execution plan of this query.

	QUERY PLAN text
1	Unique (cost=13.88..13.90 rows=2 width=232) (actual time=1.987..1.988 rows=1 loops=1)
2	-> Sort (cost=13.88..13.88 rows=2 width=232) (actual time=1.985..1.986 rows=1 loops=1)
3	Sort Key: v.vendorid, v.vendoremail, v.vendorpassword, ('vendor'::text)
4	Sort Method: quicksort Memory: 25kB
5	-> Append (cost=0.00..13.87 rows=2 width=232) (actual time=1.954..1.955 rows=1 loops=1)
6	-> Seq Scan on vendor v (cost=0.00..6.50 rows=1 width=68) (actual time=0.134..0.134 rows=0 loops=1)
7	Filter: ((vendoremail)::text = 'deep123@gmail.com'::text)
8	Rows Removed by Filter: 200
9	-> Seq Scan on customer c (cost=0.00..7.34 rows=1 width=67) (actual time=1.818..1.819 rows=1 loops=1)
10	Filter: ((customeremail)::text = 'deep123@gmail.com'::text)
11	Rows Removed by Filter: 200
12	Planning Time: 2.751 ms
13	Execution Time: 2.692 ms

As we can see in the execution plan, we are performing a sequential scan on Vendor table followed by a sequential scan on Customer table.

7.1 Improvisation

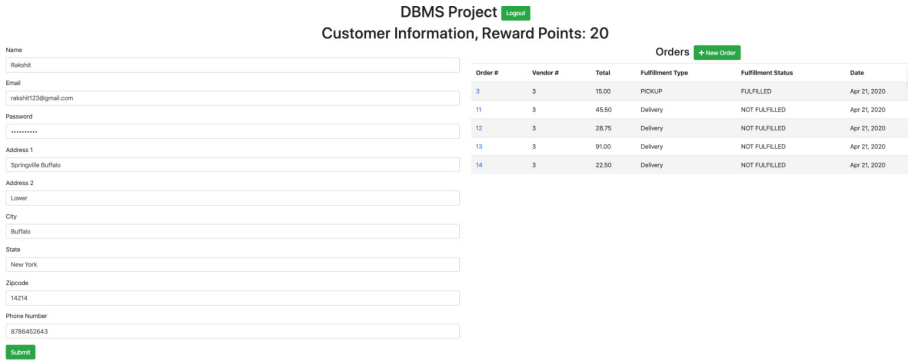
We can improve the performance of the query by creating a separate table which consists of all the email IDs and passwords and a type column which will define if the user is a Vendor or Customer. The table can be defined as:

ID, EmailId, Password, Type.

The ID will be a foreign key in both Vendor and Customer table. By doing this, we eliminate the need of joining 2 tables and scanning each table separately. We just need to lookup in a single table and get the type of the user, i.e. Customer or Vendor. This will reduce the overall cost, as well as save the planning and execution time.

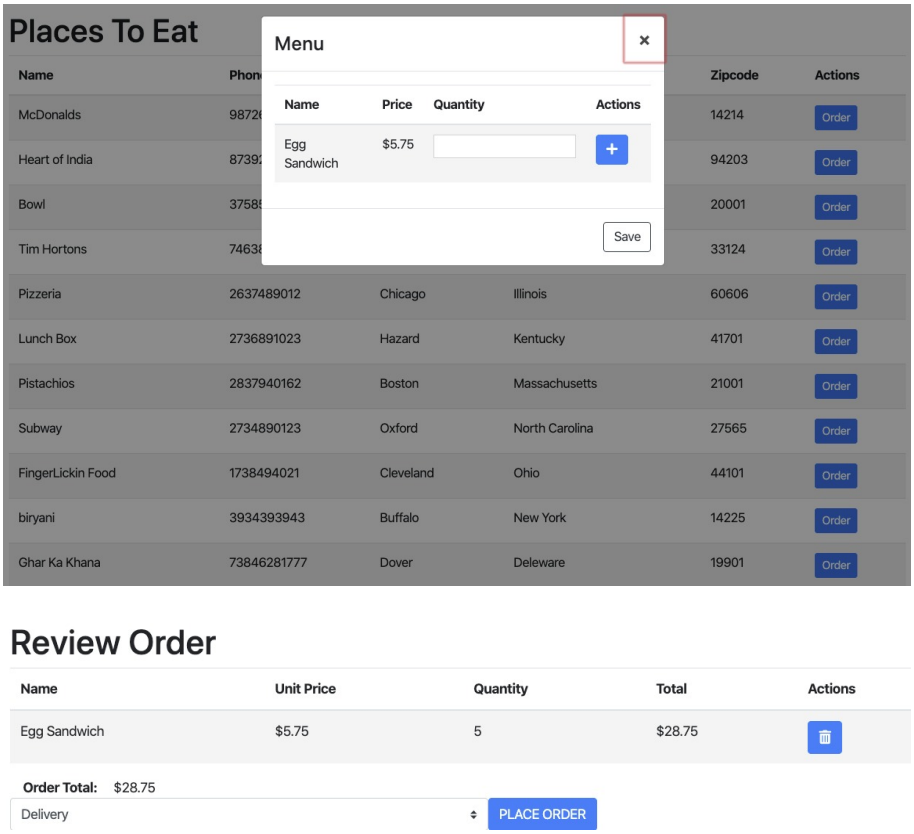
8 Updated UI

8.1 Customer Dashboard



This is the Customer Dashboard where they can see their info on the left and click submit to save any changes they have made and on the right they can see their past orders as well and when they click "New Order" Button then they can go to the Placing order screen and place order from their favorite place.

8.2 Order Placing Screen



This is the order placing screen where they can see the list of vendors from where they can order and when clicking on the blue order button they can see the popup with the menu and add the quantity needed and then press the "+" button to add the items to order, Once done they can scroll all the way down to place order, which is shown above.

8.3 Confirmed Order Screen

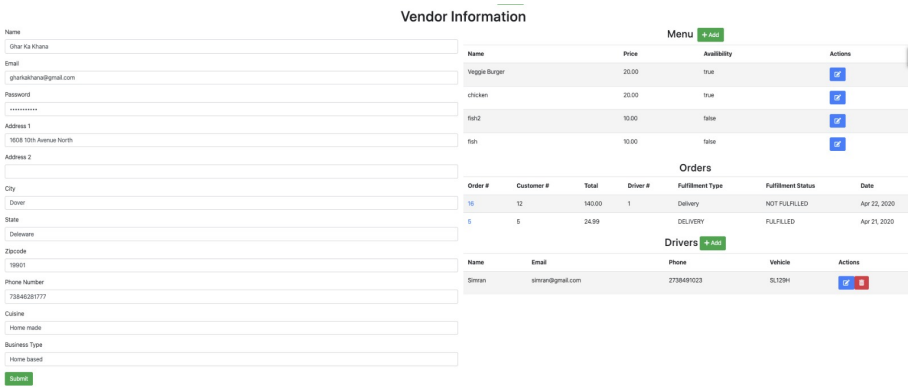
Order ID: 17
Order Date: Apr 22, 2020
Order Total: $28.75
Order Status: NOT FULFILLED
Delivery Method: Delivery

Order Items

Dish Name	Quantity	Dish Price	Total Price
Egg Sandwich	5	$5.75	$28.75

Here we can see the confirmed order screen which shows the important information about the order and what the items were.

8.4 Vendor Dashboard



This is the vendor dashboard which is similar to the customer dashboard as they can see and edit their info on the left and on the right they can see the menu, past orders, and drivers on hand. They can then click the edit or add button to edit/add menu items or drivers and then click on the blue link on the orderid to see more info on the past order.

9 GitHub Repository Link

https://github.com/rakishii13/cse562_dbms