



## **CPSC 471: Final Report**

Group 17: Gil Abinal, Maria Mamacalay, Sadat Islam  
December 11th, 2017

<b>Project Proposal</b>	<b>3</b>
Introduction	3
Problem Definition	3
Proposed Solution	3
Motivation	4
Conclusion/Timeline	4
References	4
<b>Entity-Relationship Design</b>	<b>5</b>
Diagram	5
Assumptions	5
Changes from Implementation	6
<b>Relational (Logical) Database Design</b>	<b>7</b>
Diagram	7
Changes from Implementation	8
<b>General Application Hierarchal Structure</b>	<b>9</b>
HIPO Diagram	9
HIPO Functions	10
DFD Diagram	15
<b>User Manual</b>	<b>17</b>
General User	17
Main Layout	17
How to Order	17
Navigating the Menu	18
Specials	20
About	21
Contact	21
Administration Access	22
How to Login as Admin	22
Homepage for Admins	23
How to Manipulate the Menu	23
How to Create Promotions	24
How to Manage Orders	25
<b>Appendix A: Initial Database State</b>	<b>27</b>

# Project Proposal

## Introduction

A local Calgary family restaurant, *Bangla Bazaar Food Court*, currently does not have a website. While the restaurant has a Facebook page, the lack of an official website means potential customers do not have the option to view specific menu items offered, the price of each menu item, or additional information about the restaurant online. To solve this issue, our group will develop a web application so that the manager can advertise his restaurant on the web. Our motivation is to help his small business gain as much traction as possible.

This report will elaborate on our problem, our proposed solution, as well as our motivation for conducting this project. We will end the report with an estimated timeline of deliverables.

## Problem Definition

A website is usually the first place a customer goes to browse a company's products. Browsing the web is simple and efficient. Without a website, a restaurant risks losing potential customers who are interested in their cuisine, but have questions that they need answered. For instance, what is the average cost to dine? What does the restaurant offer in its menu? Is it family friendly? What hours are they open? If a customer has to arrive at the venue to get the information, they may choose it's not worth the effort.

In our case, a local Calgary restaurant *Bangla Bazaar Food Court* has no website to promote their business. As of now, the restaurant does have a Facebook page as an alternative solution, but it does not feature their menu or their prices. Having their own website would be an improvement, as the manager can organize essential information and advertise their restaurant.

## Proposed Solution

Our goal is to develop a website addressing all the questions listed in the problem definition, and then some. Our website will have a general description of the cuisine offered at *Bangla Bazaar Food Court*, as well as contact information, location, and so forth. We will also have an interactive menu where customers can browse dishes, view the prices, and read reviews.

With an admin login, we will allow the manager to update the homepage with new specials, update the menu items, and update employee information. We may choose to relocate the managerial tasks to an Android application to save activity on the web domain.

## Motivation

Running a small business takes a lot out of a manager's day. We want to alleviate some of the stress by advertising on his behalf. In this day and age, having your business online is a necessity to produce revenue and to attract potential customers - especially if your business is small and local. Although our project is simple, it is important and essential since it not only lets us apply what we have learned into real-life situations but it also lets us deploy an application to an actual client. We wanted to have an impact in the real world, and this is a real world project.

## Conclusion/Timeline

Overall, our group strives to develop a website for the *Bangla Bazaar Food Court* so the restaurant can promote their business and gain more customers. Our first step to achieve our goal is to create a ER diagram of our database. We will implement it on ASP.Net using Visual Studio 2017 and populate it with some test data. Following the schedule posted in the project information sheet on D2L, we will then work on the front-end and implement the required webpages and empty tables. Finally, we will write functions to interact with our database. Our group will continue to meet with the manager throughout this project and ask for their input on our progress - if our designs meets their requirements, if they would like to add any features, and so forth. The bulk of our time should be spent on the web application. If time permits, we will create a managerial phone application.

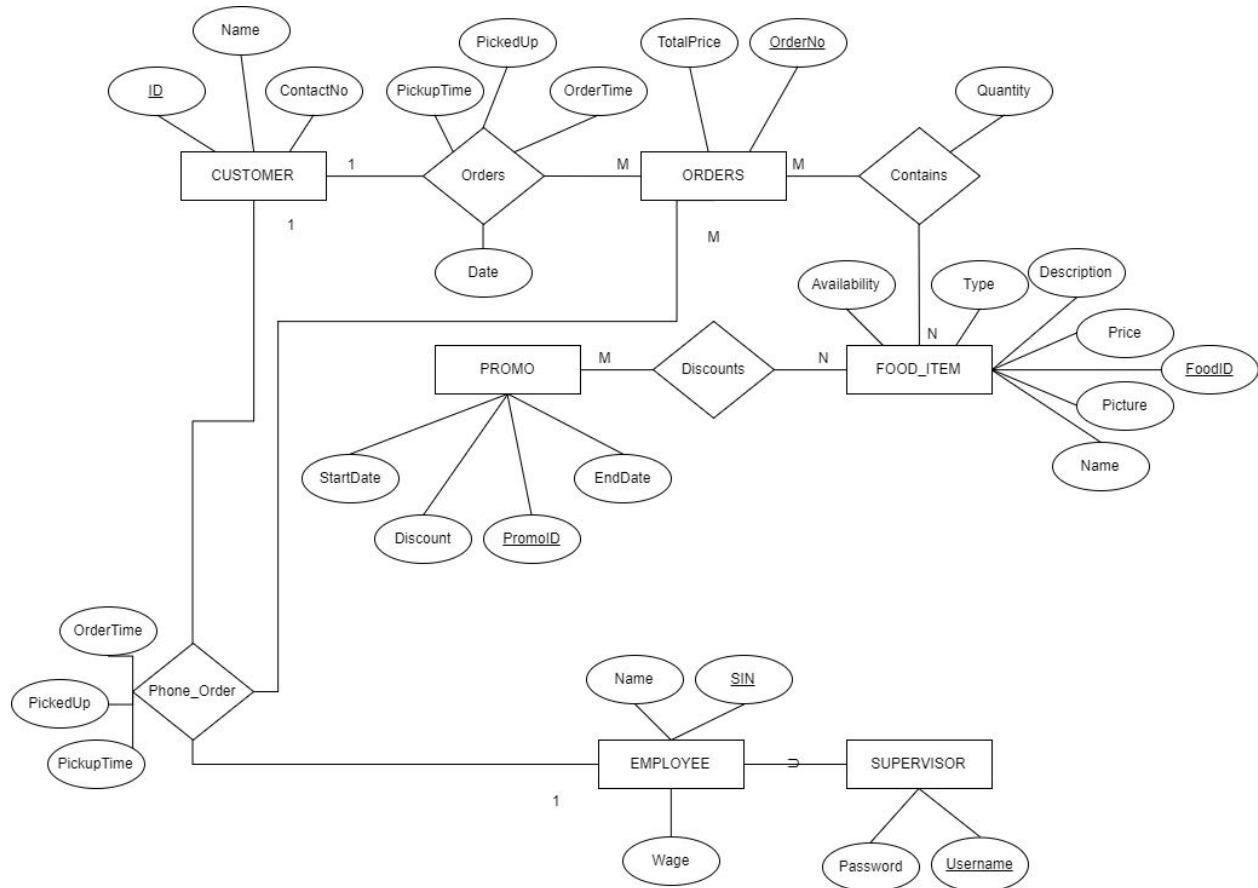
From the time given, our implementation resulted in a prototype of a [website](#) for the *Bangla Bazaar Food Court*. We have implemented the majority of the backend and database functionalities for the website, such as adding, editing, and deleting food items from the menu, special promotions, and orders. We also achieved in developing a front-end layout of the website and started on a few of its functionalities, which includes dynamically adding menu items on a customer's cart. Our group is motivated to refactor and refine the front-end functionalities in future iterations. Our website's functionalities and database system will be further explained in the sections below and in our User Manual.

## References

- [Facebook page of Bangla Bazaar Food Court](#)
- [Our group's database implementation of Bangla Bazaar Food Court](#)
- [Our GitHub repository and source code for Bangla Bazaar Food Court](#)

# Entity-Relationship Design

## Diagram



## Assumptions

- We do not need a MENU entity in our database. This is because we are assuming that the menu will be displayed in the front-end, with the user interface and interaction. For the back-end and the actual database system, the ORDERS and FOOD\_ITEM entities will suffice for the food information.
- The Type attribute in the FOOD\_ITEM entity refers to if a FOOD\_ITEM is an appetizer, entrée, dessert, drink, etc. The Description attribute is a text-based description of the FOOD\_ITEM.
- CUSTOMERS can order multiple times, but each ORDER will be assigned its own OrderNo and will be treated as a different transaction. CUSTOMERS will be assigned an ID, which will appear as an incrementing AutoNumber data type in the database. This is

due to the assumption that the restaurant does not require customers to be a member to order online.

- It is possible that EMPLOYEES may place an ORDER for a CUSTOMER. This may occur if a customer orders through the phone. To do this, we created a "Phone\_Order" relationship between the EMPLOYEE, CUSTOMER, and ORDER. We chose to use a different relationship compared to our regular ORDER in order to reduce null values, since we expect most of the orders to be in person or online.
- A SUPERVISOR has access to modify the website or the database, so we need to store their Username and Password, where the Password will be properly encrypted through hashing to enforce security.

## **Changes from Implementation**

During our implementation of the website application, we have encountered some changes to the Entity-Relationship Design. We decided to omit the Phone\_Order relation, as we found its attributes redundant to the orders relationship in our implementation. In the case a customer may order through the phone, we decided that the employee will fill out their information for them through the database, and then edit its PickUpTime when it is picked up. Moreover, we also decided to remove the PickedUp attribute from the Orders relationship, as it was also redundant to our implementation. We saw that the PickedUp attribute can be inferred from the PickUpTime attribute, since this attribute will be set to NULL until it is edited with a Date value - thus indicated that an order was picked up.

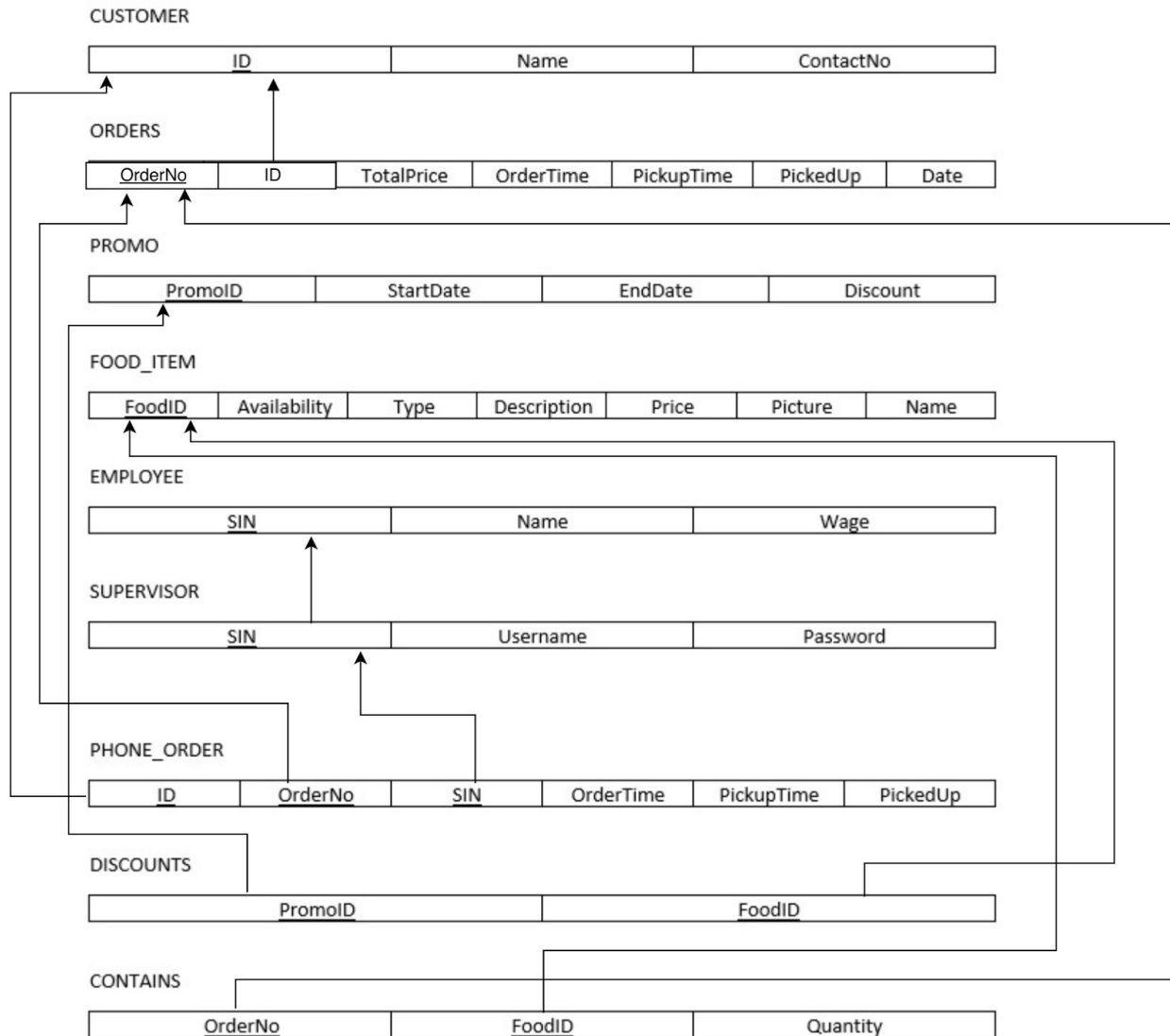
Our group also did not include the SUPERVISOR entity in our database implementation. Instead, we used ASP.Net's provided interface and methods to generate logins for the supervisor to create, edit, and delete items from a selected database. This way, we can ensure that the security of the supervisor's username and password is safely encrypted -- rather than us encrypting the information ourselves.

A few attributes were also added or modified in our implementation. For instance, we included a "Name" attribute so administrators can easily distinguish which Promotion they may be editing or deleting, thus increasing the user-friendliness of our website.

These changes should also be noted for our Relational(Logical) Database Design below.

# Relational (Logical) Database Design

## Diagram



**Note:** The primary keys are the first element in every table. They are underlined, and it distinguishable through the heavier line weight underneath each primary key.

## **Changes from Implementation**

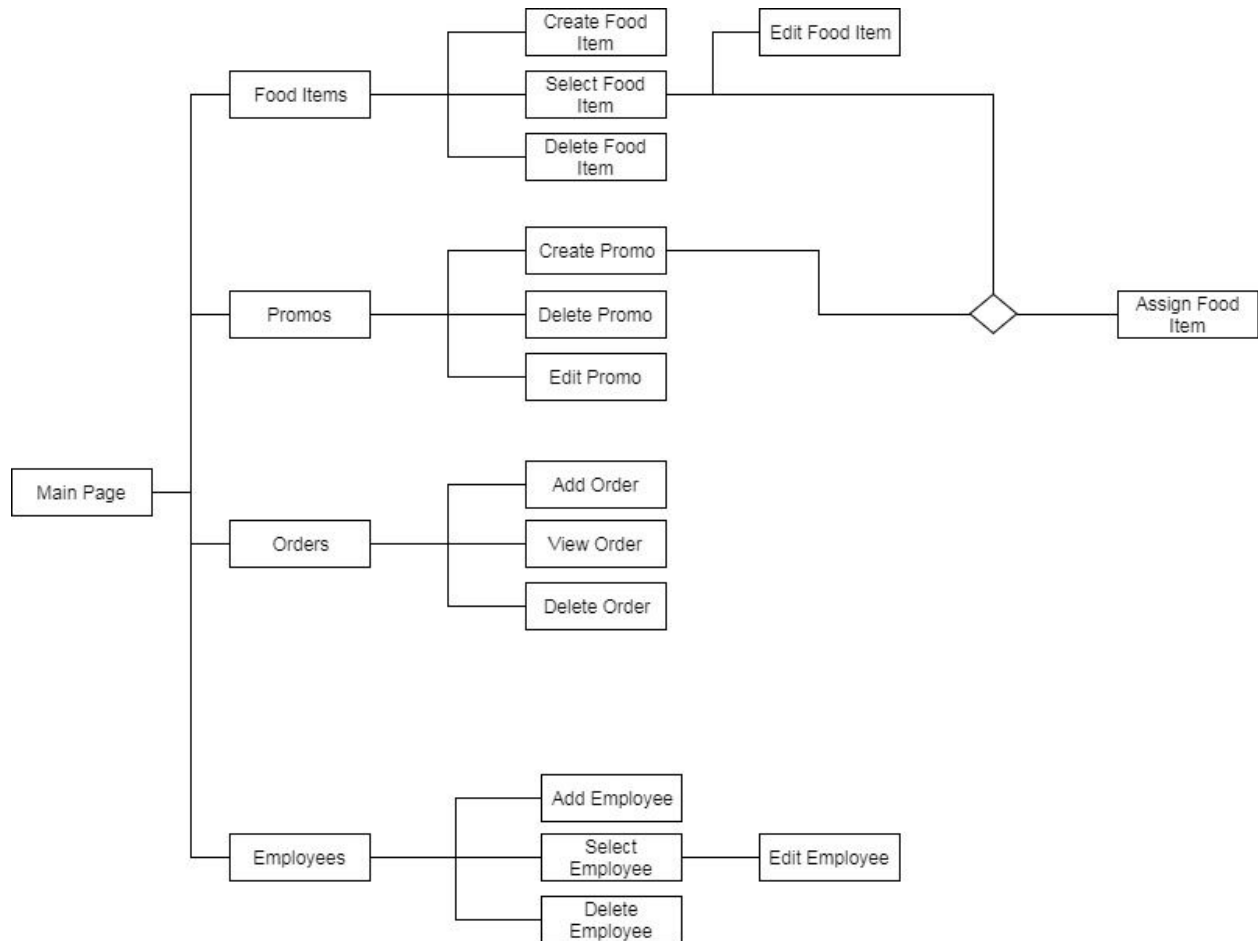
As mentioned in the previous section, our group implemented changes to our Entity-Relationship and consequently our Relational (Logical) Database Design. Our changes in our relational model is similar to the changes in our Entity-Relationship design: the PHONE\_ORDER and SUPERVISOR relations and the PickedUp attribute of the ORDERS entity were omitted, and we included a Name attribute to our PROMO relation.

To justify our modifications to our initial database design, we believed that these modifications will reduce the redundancy and the increase the user-friendliness and security of our database implementation. The PHONE\_ORDER relation may take up unnecessary overhead since most of its attributes are the same as the ORDERS relation, and including a Name attributes for the Promotions will make each promotion distinguishable to the administration - rather than having to memorize a promotion's ID number or its start and end dates.



# General Application Hierarchical Structure

## HIPO Diagram



## HIPO Functions

**Function:** Create Food Item

**Inputs:** @Food\_Item x

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = INSERT INTO Food\_Items( Availability, Type, Description, Price, Picture, Name )  
VALUES (x.availability, x.type, x.description, x.price, x.picture, x.name);

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else false

**Function:** Select Food Item

**Inputs:** @int id

**Outputs:** Food\_Item

**Pseudocode:**

Connect to database

Query = SELECT \* FROM Food\_Items WHERE id = FoodID;

Parse Query

Execute Query

Close Connection to Database

Return the Food\_Item object

**Function:** Delete Food Item

**Inputs:** @int id

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = DELETE FROM Food\_Items WHERE 'id' = 'Food\_Id';

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else false

**Function:** Edit Food Item

**Inputs:** @Food\_Item x

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = UPDATE Food\_Item SET Availability = x.availability, Type = x.type, Description = x.description, Price = x.price, Picture = x.picture, Name = x.name WHERE x.id = F.id;  
Parse Query  
Execute Query  
Close Connection to Database  
Return true if it worked, else return false

**Function:** Assign Food Item

**Inputs:** @Food\_Item x @Promo y

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = INSERT INTO Discounts (PromoID, FoodID) VALUES (y.promoID, x.foodID);

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else false

**Function:** Create Promo

**Inputs:** @PromoID, @StartDate, @EndDate, @Discount

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = INSERT INTO Promo

VALUES (@PromoID, @StartDate, @EndDate, @Discount);

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else return false

**Function:** Delete Promo

**Inputs:** @PromoID

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = DELETE FROM Promo

WHERE (PromoID = @PromoID);

Parse Query

Execute Query

Close Connection to Database

Return true if deletion worked; else, return false

**Function:** Edit Promo

**Inputs:** @PromoID

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = UPDATE Promo

SET "attribute"

WHERE (PromoID = @PromoID);

Parse Query

Execute Query

Close Connection to Database

Return true if update worked; else, return false

**Function:** Add Order

**Inputs:** @OrderNo, @ID, @TotalPrice, @OrderTime, @PickupTime, @PickedUp, @Date, @SIN, @FoodID, @Quantity

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = INSERT INTO Orders

VALUES (@OrderNo, @ID, @TotalPrice, @OrderTime, @PickupTime, @PickedUp, @Date);

If (Order came through Phone) then do {

INSERT INTO Phone\_Order

SELECT c.ID, o.OrderNo, s.SIN, o.OrderTime, o.PickupTime, o.PickedUp

FROM Customer as C, Order as o, Supervisor as s

WHERE (c.ID = @ID) AND (o.OrderNo = @OrderNo) AND (s.SIN = @SIN); }

// inserting what food items were ordered by customer

INSERT INTO Contains(OrderNo, FoodID)

SELECT o.OrderNo, f.FoodID

FROM Order as o, Food\_Item as f

WHERE o.OrderNo = @OrderNo AND f.FoodID = @FoodID;

// adding quantity to the food items order

UPDATE Contains

SET Quantity = @Quantity

WHERE OrderNo = @OrderNo AND FoodID = @FoodID;

Parse Query

Execute Query

Close Connection to Database

Return true if addition is successful; else, return false

**Function:** View Order

**Inputs:** @OrderNo

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = if (the order was a Phone\_Order), then do {

```
    SELECT *
    FROM Phone_Order as p
    WHERE p.OrderNo = @OrderNo AND
        ( SELECT *
          FROM Contains as c
          WHERE c.OrderNo = p.OrderNo; )}
```

else do {

```
    SELECT *
    FROM Orders as o
    WHERE o.OrderNo = @OrderNo AND
        ( SELECT *
          FROM Contains as c
          WHERE c.Orderno = o.OrderNo; )}
```

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else return false

**Function:** Delete Order

**Inputs:** @int orderNo

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = // first deleting any records that contain OrderNo as a foreign key

```
    DELETE FROM Contains as c WHERE c.OrderNo = orderNo;
    If (the order was a Phone_Order), then do {
        DELETE FROM Phone_Order as p WHERE p.OrderNo = orderNo;
    }
```

// deleting record

```
    DELETE FROM Orders as o WHERE o.OrderNo = orderNo;
```

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else return false

**Function:** Add Employee

**Inputs:** @Employee x

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = INSERT INTO Employee (SIN, Name, Wage) VALUES (x.sin, x.name, x.wage);

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else false

**Function:** Select Employee

**Inputs:** @int sin

**Outputs:** Employee

**Pseudocode:**

Connect to database

Query = SELECT \* FROM Employee WHERE SIN = sin;

Parse Query

Execute Query

Close Connection to Database

Return Employee if found, else return null

**Function:** Edit Employee

**Inputs:** @Employee x

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = UPDATE Employee SET Name = x.name, Wage = x.wage WHERE SIN = x.sin;

Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else return false

**Function:** Delete Employee

**Inputs:** @int sin

**Outputs:** Boolean

**Pseudocode:**

Connect to database

Query = DELETE FROM Employee WHERE SIN = sin;

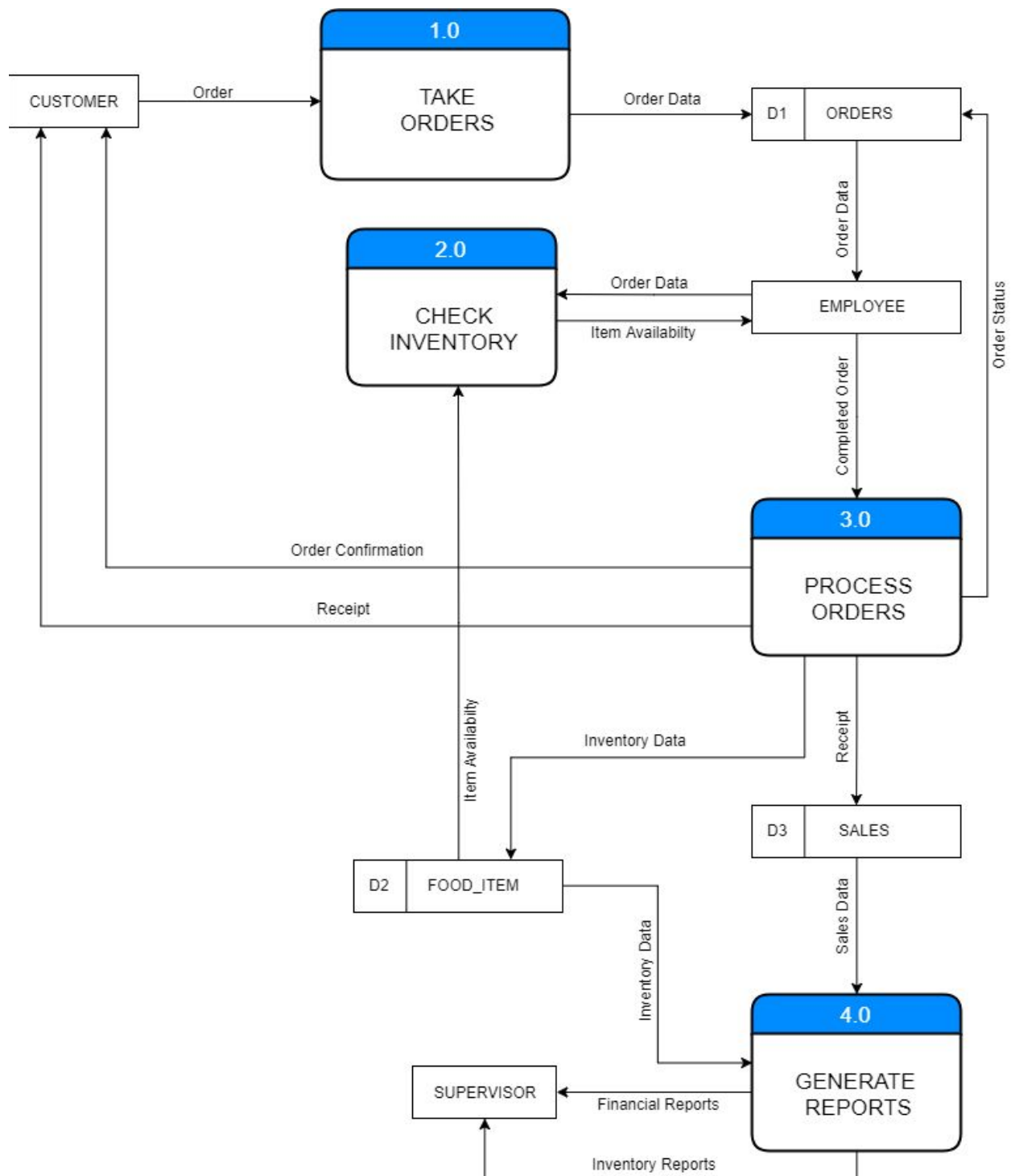
Parse Query

Execute Query

Close Connection to Database

Return true if it worked, else return false

## DFD Diagram



**Note:** We included the GENERATE REPORTS process in the DFD model but not in the HIPO model because we are planning to include it into our bonus implementation (the Android implementation) if time allows it. For our main implementation, we are focusing on creating a website for the customers to order food items from the database and for employees to add food items, employees, and promotions to the database. Our planned bonus implementation focuses on the back-end, staff, and employees. Here, the staff administrators will be able to add and edit employees as well as see monthly reports on their inventory and finances. Thus, our HIPO diagram is solely on the website implementation, whereas our DFD diagram shows our overall flow of data in our project, which includes generating monthly reports for the staff and employees to view in the Android application.



# User Manual

## General User

### Main Layout

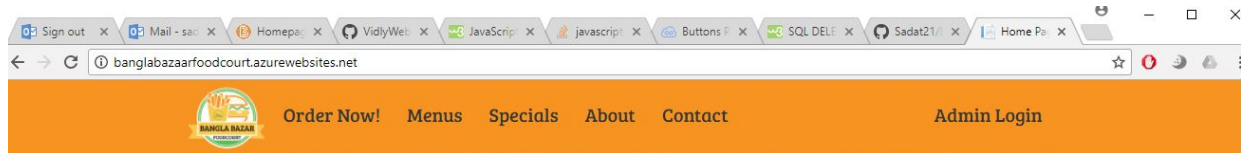


Figure 1: Navigation Bar

Our main layout will primarily consist of a navigation bar that is fixed on top of the page. The navigation bar contains links for customers to navigate around the page. Below is a list of the pages and what the customer can do among these pages:

- **Logo** Homepage consisting of facebook news feed and google reviews
- **Order Now!** Order selected food items online through the website
- **Menus** View the online menu, and can order a selected item while on the menu
- **Specials** View special promotions, events, combos, and discounted prices.
- **About** View additional information about the website, their mission, and so forth.
- **Contact** View the contact information of the Bangla Bazar Food Court.

### How to Order

A screenshot of the 'Order Now!' page on the Bangla Bazar Food Court website. The page features an orange navigation bar at the top with the logo, 'Order Now!' link, and links for 'Menus', 'Specials', 'About', 'Contact', and 'Admin Login'. Below the navigation bar, the page is titled 'Order'. On the left, there is a section titled 'Combos & Promotions' with two items: 'Mutton Biryani Combo' for \$9.99 and 'Beef Burger Combo' for \$6.99. Below this are four yellow buttons labeled 'Appetizers', 'Entrées', 'Desserts', and 'Drinks'. On the right, there is a 'Your Order' section. It contains a table with the following data:

Name	Quantity	Price
Shawarma	- 12 +	\$20

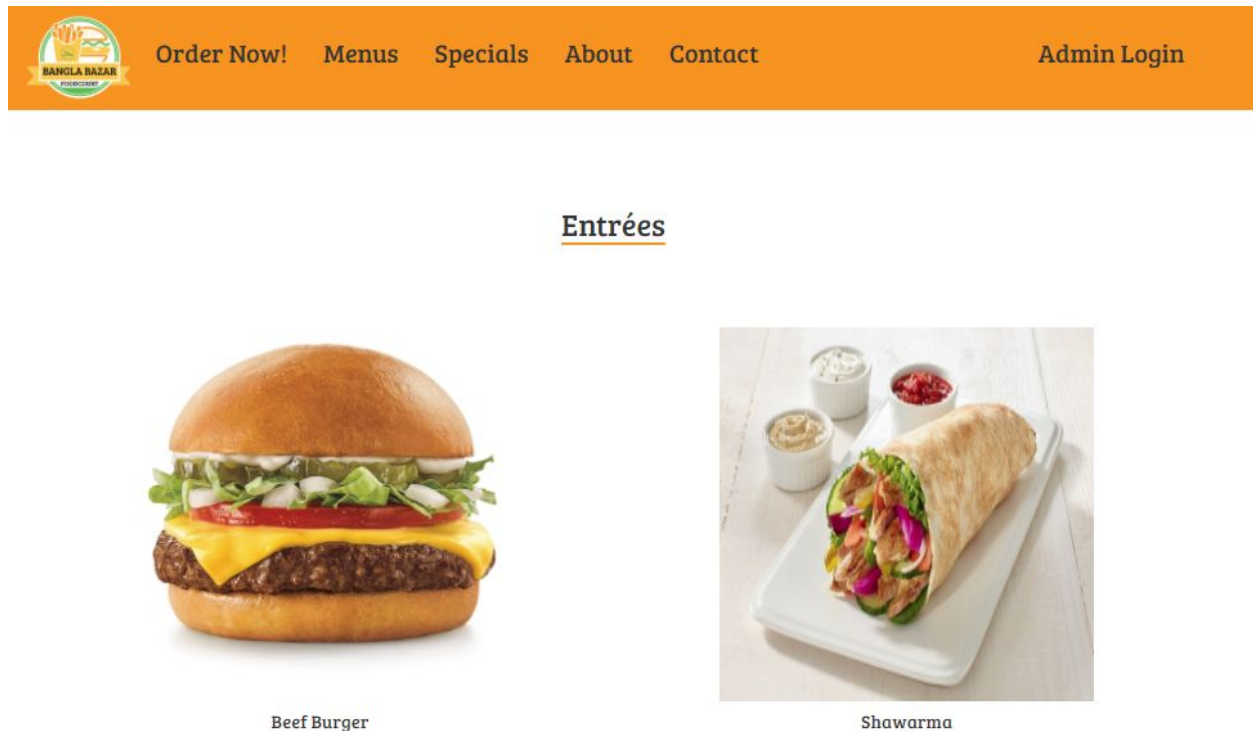
Below the table, there is a 'Total Price' label and a yellow 'Submit' button. At the bottom of the page, there is a copyright notice: '© 2017 - Bangla Bazaar Food Court - f'.

Figure 2: Order Now! page

In our **Order Now!** page, we implemented an order menu using two elements: a collapsible list that is grouped by the food type and a table that dynamically shows a person's selected order. The collapsible list is grouped by food type, and underneath each food type contains the food items of that food type. From there, the customer will be able to select how much of each food type they would want to order. This list will also contain any combos or specials available to order.

A table is also shown on this page, which dynamically shows the customer's currently selected items that they want to order and the total price of their order. Customers may select an item on this table to edit or delete the item from their order. There is a submit button next to this table, which allows the restaurant to save the customer's order into the database. After submitting an order, we ask the customer for their name and contact number.

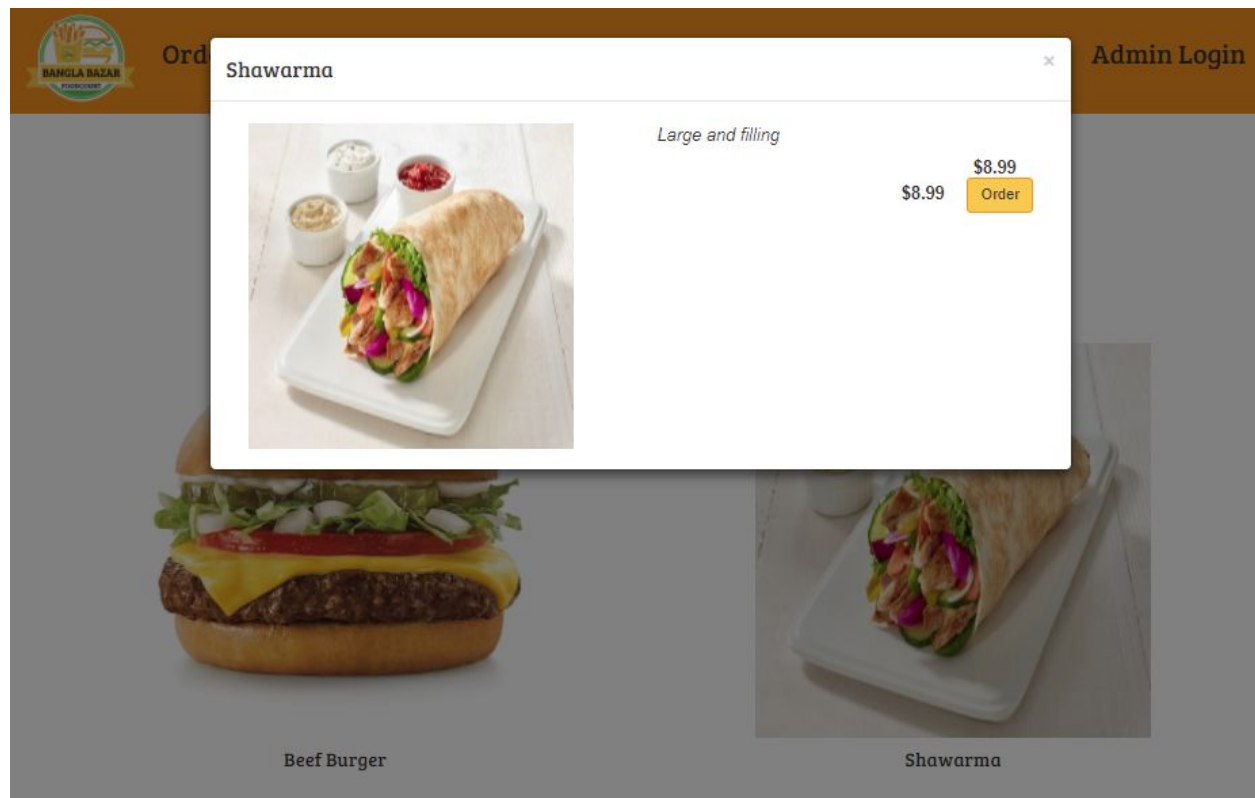
## Navigating the Menu



*Figure 3: Menu page*

To view the menu, customers can navigate to the **Menus** tab on the navigation bar. From there, they will see a gridview of images of the available food items in Bangla Bazaar's menu. A

customer can click on a given image, and a modal will appear on the screen, giving the name of the food item, its description, its price, and the option to order the said item.



*Figure 4: Greater Description of a Menu Item*


We also have have hyperlinks at the top of our menu to fast track to a certain portion of our menu.




*Figure 5: Hyperlinks to Sections of our Menu*

## Specials

In the **Specials** page, we have a carousel view that shows images of the current combos, discounts, and events currently taking place. We will included a list of these specials underneath the carousel. Customers add one of these orders to their cart.

[Order Now!](#) [Menus](#) [Specials](#) [About](#) [Contact](#) [Admin Login](#)

### Specials



**BOTI KEBAB & DAAL COMBO**

HALAL

**\$9.99 ONLY**

○○○○○●○○

**Mutton Biryani Combo**  
(Biryani + Desi Salad + Raita + POP)  
\$9.99 [Order](#)

**Beef Burger Combo**  
(Burger + Fries + a POP)  
\$6.99 [Order](#)


Figure 6: Specials Page

## About

The **About** page will be fairly simple, as it will include a description of the Bangla Bazar Food Court, its mission, and who the founders of the restaurant are. This section is currently blank as we are still in the process of getting this information.

## Contact

The **Contact** page displays Bangla Bazar Food Court's contact information. Here, customers can see the restaurant's phone number, email address, as well as the location of the restaurant. They can also choose to leave a message for the manager.

[Order Now!](#) [Menus](#) [Specials](#) [About](#) [Contact](#) [Admin Login](#)

### Contact.

#### Leave a Message

**Name**

**Email**

**Comment**

Submit

#### Contact Details

**Address**  
109 4851 Westwinds Dr, NE  
Calgary, Alberta  
T3J 4L4

**Phone Number:** (587) 356-6000

**Email:** [banglabazar.canadabangladeshhl@gmail.com](mailto:banglabazar.canadabangladeshhl@gmail.com)

### Find Us!

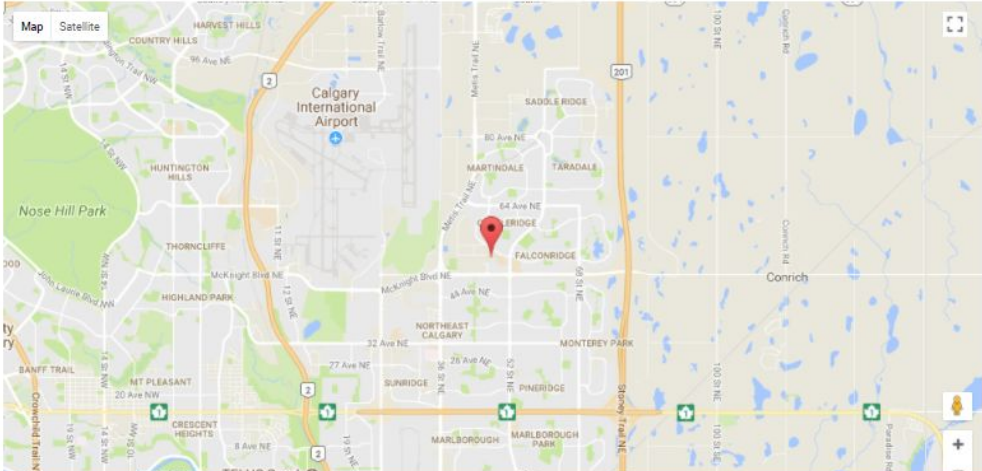


Figure 7: Contact Page



## Administration Access

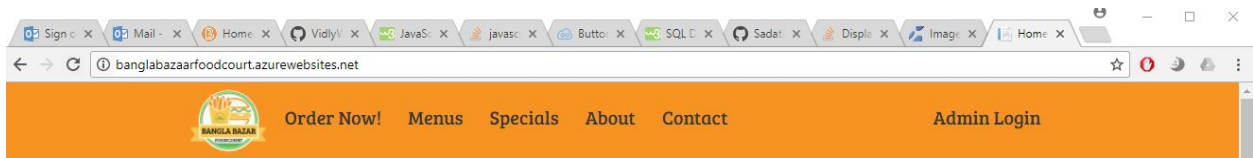


Figure 8: Navigation Bar

As seen in the image above, the navigation bar also allows for employee administrators to interact with the website, and they will have to click the **Admin Login** tab to login and access the administrative pages, where you can manipulate the menu, create promotions, and manage orders.

## How to Login as Admin

To log in as an administrator, the admin will have to click the **Admin Login** tab and fill in their username and password, which will be stored in our database. Once they fill in their log-in information, we will check if the values match and grant them access if they do. Once logged in, they will be able to:

- **Manipulate the Menu** add or remove food items to the menu
- **Create Promotions** add new promotions listed in the database into the website
- **Manage Orders** view, edit, delete orders that have been placed through the online website

A screenshot of the login page for the Bangla Bazaar Food Court website. The page has an orange header with the 'Bangla Bazaar' logo and navigation links: 'Order Now!', 'Menus', 'Specials', 'About', 'Contact', and 'Admin Login'. Below the header, the page is titled 'Log in.' and has two sections: 'Use a local account to log in.' and 'Use another service to log in.' The 'Use a local account to log in.' section contains a form with fields for 'Email' (containing 'admin@bbfc.com') and 'Password' (containing '\*\*\*\*\*'). There is a 'Remember me?' checkbox and a 'Log in' button. Below the form is a link to 'Register as a new user'. The 'Use another service to log in.' section contains a message: 'There are no external authentication services configured. See [this article](#) for details on setting up this ASP.NET application to support logging in via external services.' At the bottom of the page is a copyright notice: '© 2017 - Bangla Bazaar Food Court - 

Figure 9: Login Page

## Homepage for Admins

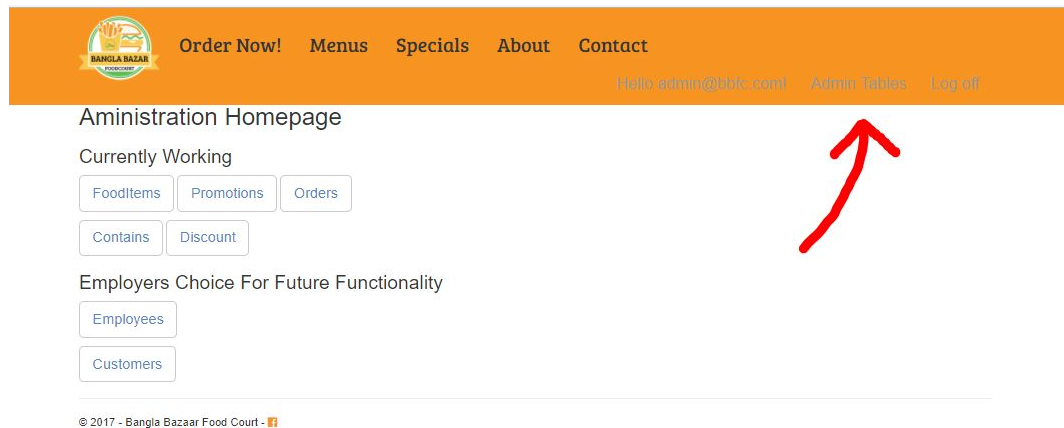


Figure 10: Admin Homepage

Clicking **Admin Tables** will lead the manger to the admin homepage. Here they can view all the database entries.

## How to Manipulate the Menu

From the homepage, the manager will want to click on the **FoodItems** button. This will lead them to the following webpage below.

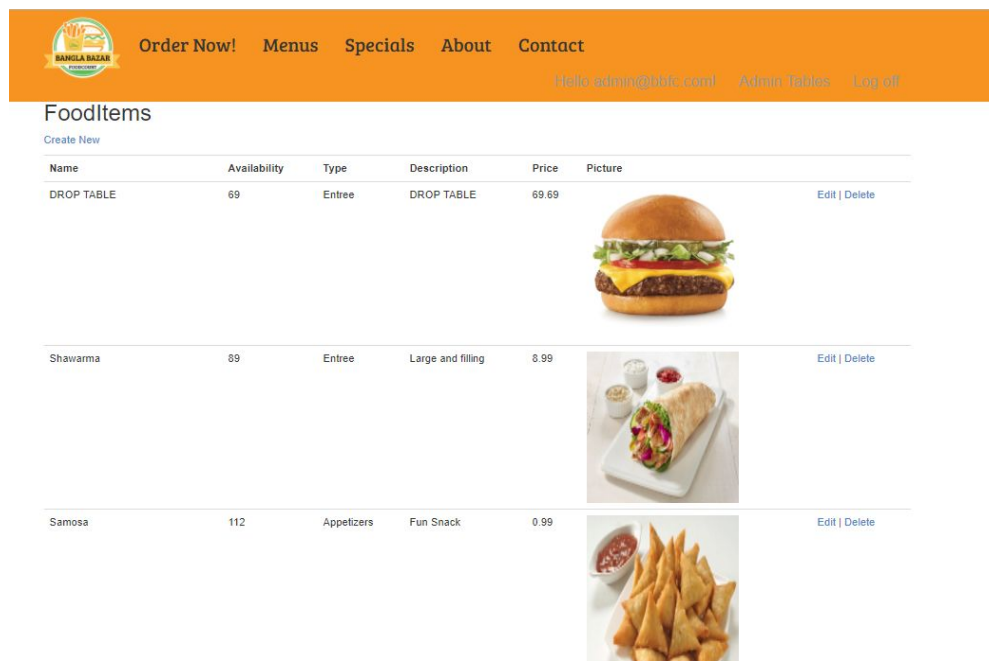


Figure 11: Food Item Homepage

Here the manager will have a view of all the current menu items in the system. Each item gives the choice to edit or delete said item on the far right hand side. Underneath the heading, there is an hyperlink that allows you to create new menu items.

**CreateFoodItem**

Food\_Item

Name

Availability

Type

Description

Price

Picture  No file chosen

[Back to List](#)

© 2017 - Bangla Bazaar Food Court -

*Figure 12: Create New Food Item*

Here you fill out the information for the new menu item and upload a picture. The picture is mandatory as it is required to be displayed in the menu. Once you fill out the form, hit submit. You will now see your newly added menu item.

## How to Create Promotions

From the *Admin Homepage*, click the Promotions button. You should see the following webpage below.

**Promotions**

[Create New](#)

Name	Start Date	End Date	Discount	
Mutton Biryani Combo	12/11/2017 12:00:00 AM	12/31/2017 12:00:00 AM	9.99	<a href="#">Edit</a>   <a href="#">Add Food</a>   <a href="#">Delete</a>
Beef Burger Combo	12/11/2017 12:00:00 AM	12/31/2017 12:00:00 AM	6.99	<a href="#">Edit</a>   <a href="#">Add Food</a>   <a href="#">Delete</a>

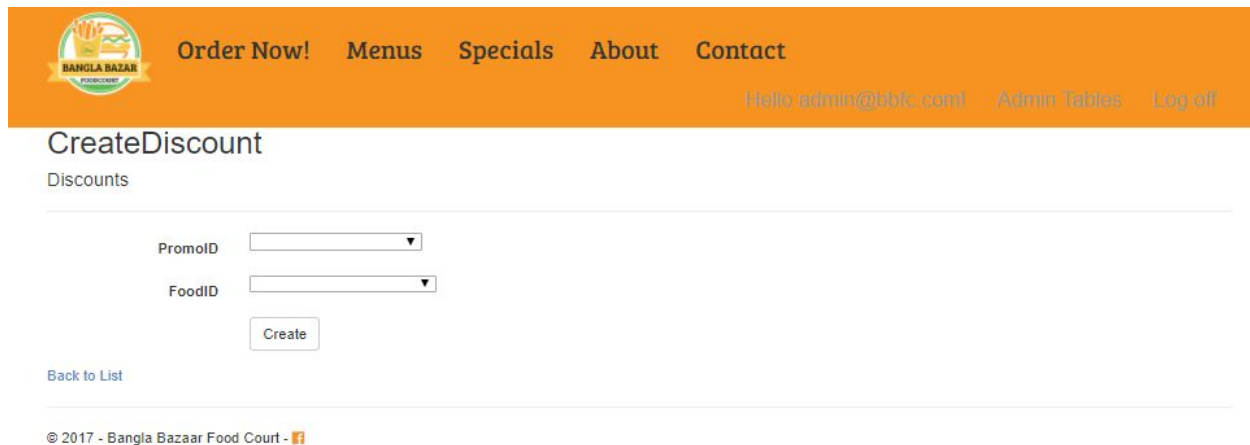
© 2017 - Bangla Bazaar Food Court -

*Figure 13: Promotions Homepage*

Here you can view all the current Specials that are listed on the website. Here you can create, edit, and delete a promotion. A few new features on this page include the ability to add food

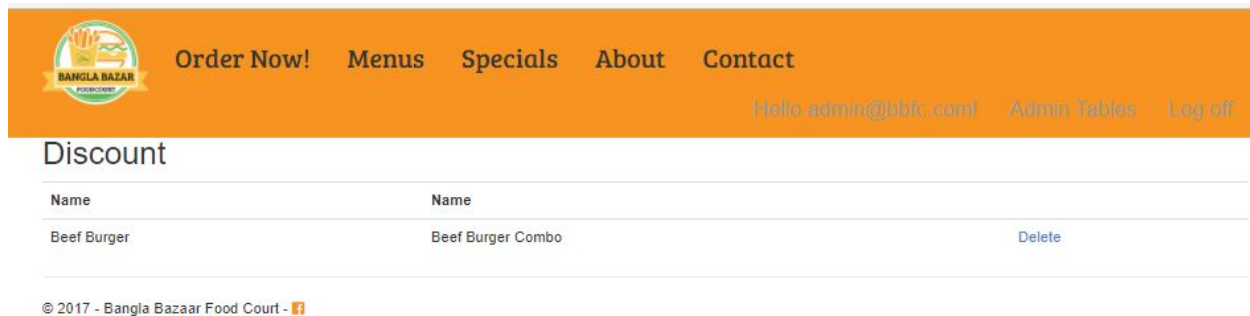


items to each promotion. If you click the *Add Food* button, you should arrive at the webpage below.



*Figure 14: Create Discount Page*

The *PromoID* drop down menu will consist of the promotion that you wish to add more food items too. The *FoodID* drop down menu will list all the food items currently in the menu. After making your selections, hit submit. You will be taken back to the *Promotions Homepage*. Now if you click the name of the promotion, you will be taken to the following page below.




Name	Name
Beef Burger	Beef Burger Combo
	<a href="#">Delete</a>

*Figure 15: Discount Page for Selected Promotion*

Here you will be able to see the newly added food item to the promotion. You can delete any of these food items from this menu as well.

## How to Manage Orders

From the *Admin Homepage*, click the button that says *Orders*. You'll arrive at the page shown below.

[Order Now!](#)[Menus](#)[Specials](#)[About](#)[Contact](#)

Hello admin@bbfc.com! [Admin Tables](#) [Log off](#)

## Orders

[Create New](#)

Name	orderNo	Total Price	Order Time	Pickup Time	Date	
	1	99	12:20	13:00	5/5/2005 12:00:00 AM	<a href="#">Edit</a>   <a href="#">Add Food Item</a>   <a href="#">Delete</a>

*Figure 16: Orders Homepage*

You are able to create, edit, and delete any order. You can also view what food elements an order contains by clicking on the *orderNo*. You can also manually add food items to each order the same way we did for promotions.

## Appendix A: Initial Database State

### CUSTOMER

<u>ID</u>	Name	ContactNo
NULL	NULL	NULL

### ORDERS

ID	<u>OrderNo</u>	TotalPrice	OrderTime	PickupTime	Date
1	1	99.00	12:20	13:00	5/5/2005 12:00:00AM

### PROMO

<u>Name</u>	StartDate	EndDate	Discount
Mutton Biryani Combo	12/11/2017 12:00:00AM	12/31/2017 12:00:00AM	9.99
Beef Burger Combo	12/11/2017 12:00:00AM	12/31/2017 12:00:00AM	6.99

### FOOD\_ITEM

<u>FoodID</u>	Name	Availability	Type	Description	Price	Picture
1	Beef Burger	69	Entree	DROP TABLE	69.69	...
2	Shawarma	89	Entree	Large and filling	8.99	...
3	Samosa	112	Appetizers	Fun snack	0.99	...
4	Chocolate Cheese Cake	56	Dessert	Creamy	4.99	...

## EMPLOYEE

<u>SIN</u>	Name	Wage
NULL	NULL	NULL

## DISCOUNTS

<u>PromoID</u>	<u>FoodID</u>
2	1
1	4

## CONTAINS

<u>OrderNo</u>	<u>FoodID</u>	Quantity
1	1	2
1	2	2