ROS

Restaurant Ordering System

Restaurant Ordering System
Final Report

12 December, 2015

Submitted to:
Dr. Marlene M. de Leon

Submitted by:
Breginald Jino Basilio
Zachary Ivan Martinez
Kryzl Deanne Pascual
Jenina Isabel Sanchez

Group E

CS123 A

Context of the Project

The Restaurant Ordering System (ROS) is a menu application that aims to make the ordering process easier for the administrator, customer, and the kitchen staff.

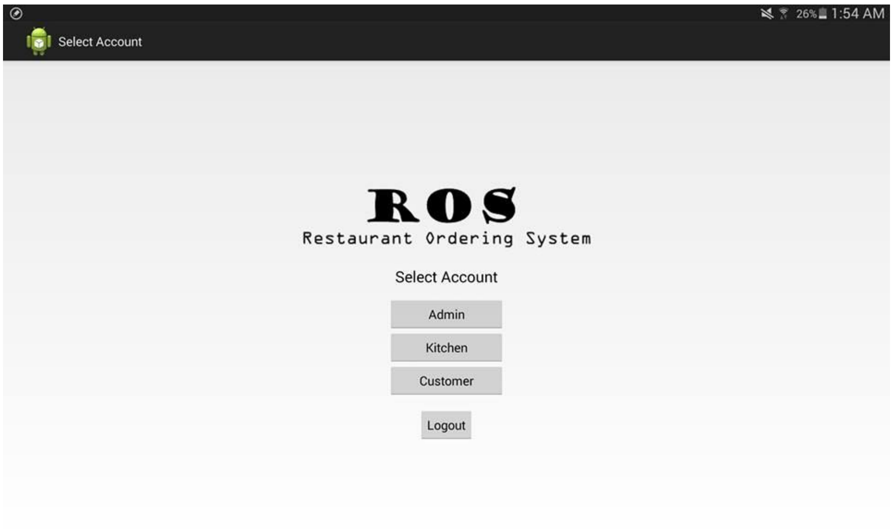
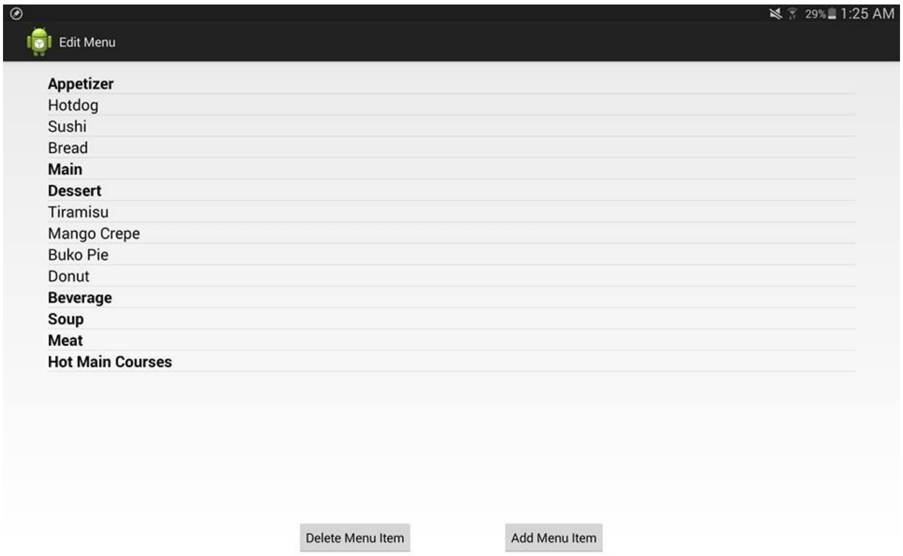


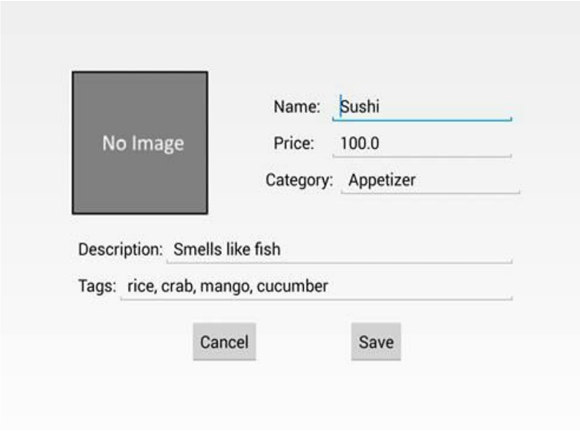
Figure 1. The account role choosing screen.

The administrator can view and edit the menu. This menu is divided into categories, and each category can expand to reveal the menu items. The administrator can view the profile of the menu item by clicking on its name. From this screen, the administrator may also choose to edit any detail. From the edit menu screen, they may choose to add new menu items.

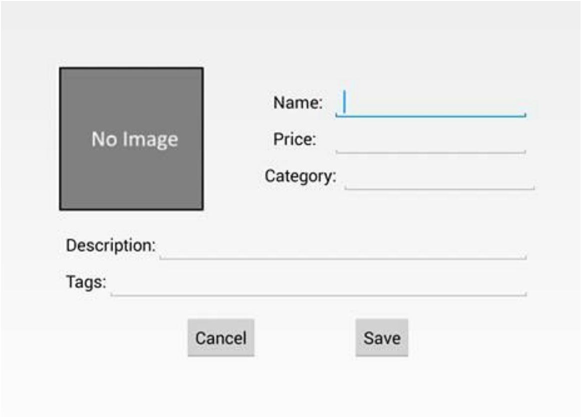


Appetizer
Hotdog
Sushi
Bread
Main
Dessert
Tiramisu
Mango Crepe
Buko Pie
Donut
Beverage
Soup
Meat
Hot Main Courses

Figure 2. Viewing the menu as administrator



Name: Sushi
Price: 100.0
Category: Appetizer
Description: Smells like fish
Tags: rice, crab, mango, cucumber

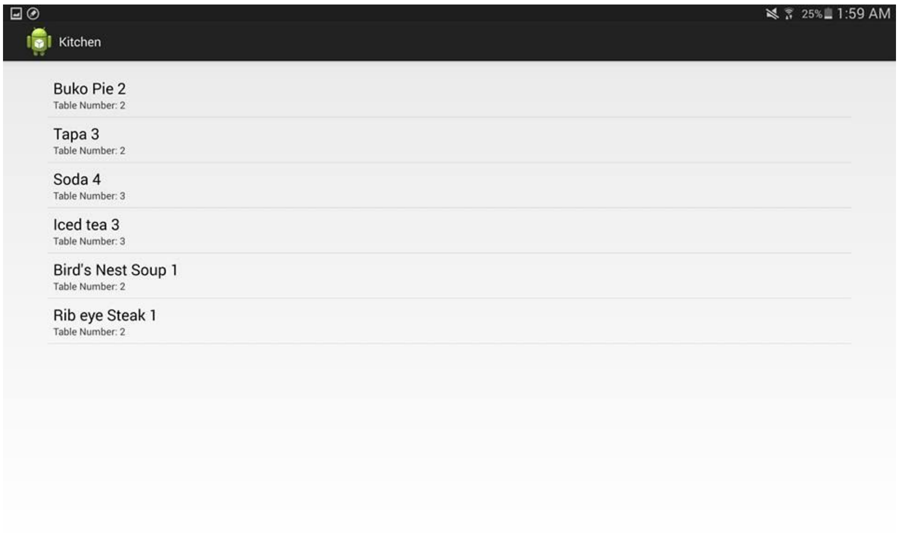


Name:
Price:
Category:
Description:
Tags:

Figure 3. Viewing a menu item profile

Figure 4. Adding a new menu item

The kitchen staff can view the list of orders. The order contains the menu item, quantity, and the table who ordered it. Clicking an order will remove it from the list.



Buko Pie 2
Table Number: 2

Tapa 3
Table Number: 2

Soda 4
Table Number: 3

Iced tea 3
Table Number: 3

Bird's Nest Soup 1
Table Number: 2

Rib eye Steak 1
Table Number: 2

Figure 5. Viewing the order list as kitchen

The customer can view the menu and search for menu items. The search bar checks for any item containing the query. Clicking Search replaces the menu list with a list of items that fit the criteria. Clicking the Return button will return the item list to the menu.

Clicking the menu item will bring up the item order profile. The customer may select the quantity they wish to add to their cart. After clicking Add to Cart, the customer may click View Cart, then click Finalize to submit their orders to the kitchen.

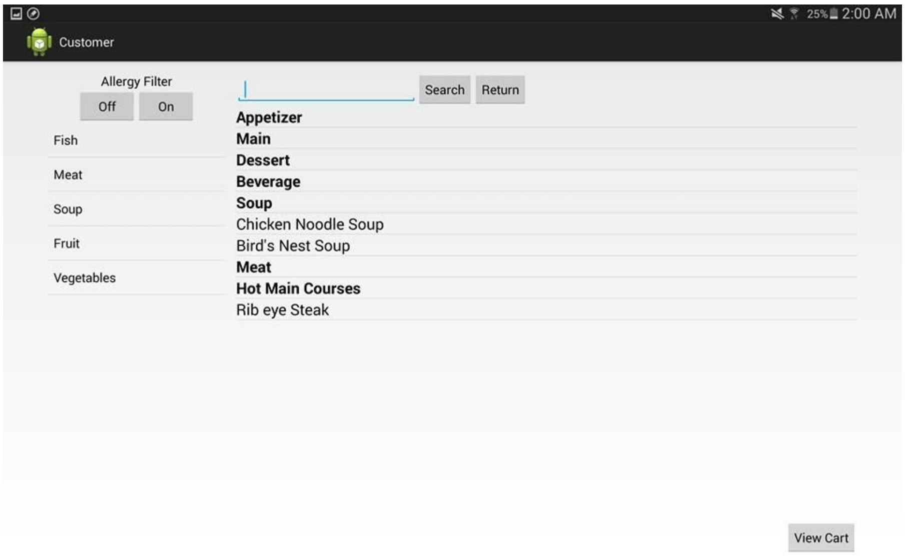
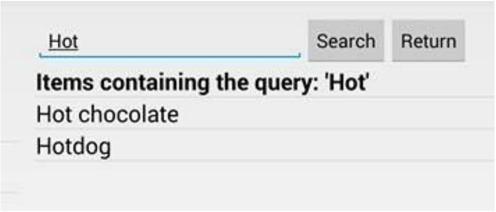


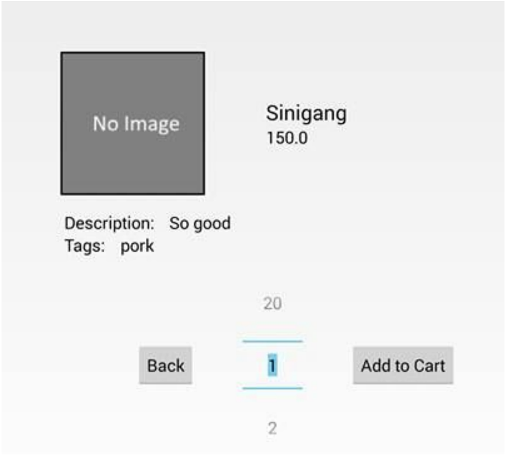
Figure 6. Viewing the menu as customer



Items containing the query: 'Hot'

Hot chocolate

Hotdog



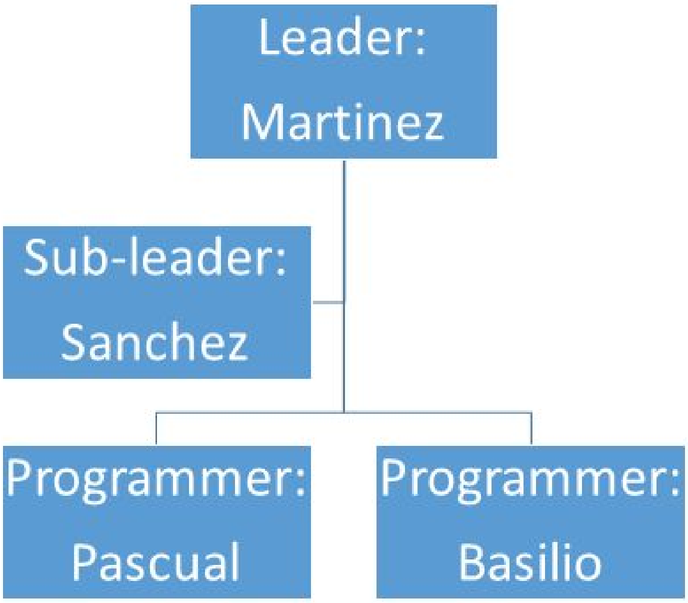
Sinigang
150.0

Description: So good
Tags: pork

Figure 7. Viewing the search results

Figure 8. Viewing the item order profile

Team Organization



Our team decided to adopt the Modified Hierarchical Team organization. We chose this because we realized that it is easier to work when there is a leader dividing the work and telling us what to do next. We decided to have a sub-leader, as well, to assist the leader in their work. The remaining team members worked mainly as programmers. All members of the team were tasked with programming their respective parts of the application, as well as working on the documentation.

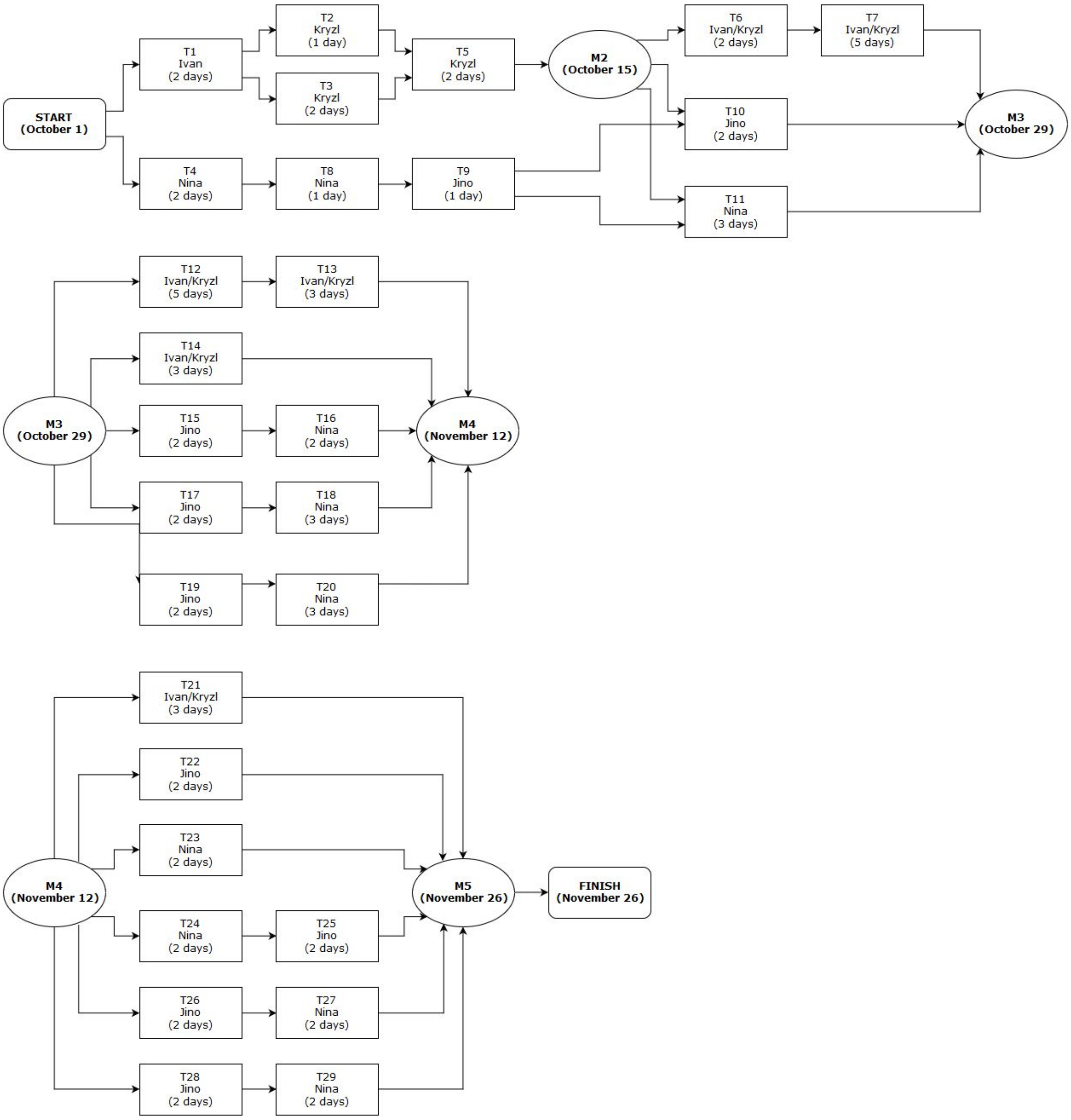
Our team communicated in person, through Facebook, and through Messenger. We used GitHub as our version control and online repository. The leader divided the tasks based on the team member's available time and expertise.

Sprint Plans

Sprint	User Stories	Tasks
2 October 15	The user should be able to log-in.	Front-end: T1: Design all the screens for the Android application. T2: Implement the log-in screen. T3: Implement the “choose-user-type” screen Back-end: T4: Design the ERD. T5: Access the choose-user menu after logging in
3 October 29	The admin should be able to log-in. The admin should be able to view the interface.	Front End: T6: Develop admin’s menu screen T7: Develop admin menu add/edit item screen Back End: T8: Create the database tables. T9: Implement the database. T10: Implement the log-in function for administrator. T11: Implement log-in authentication for administrator.
4 November 12	The customer should be able to view the menu. The customer should be able to add menu items to cart.	Front End: T12: Develop customer menu T13: Develop customer cart screens T14: Develop admin bills screens Back End: T15: Implement function code of the cart menu screen T16: Implement function queries of the cart menu screen T17: Implement add function code for admin user. T18: Implement add function query for admin user.

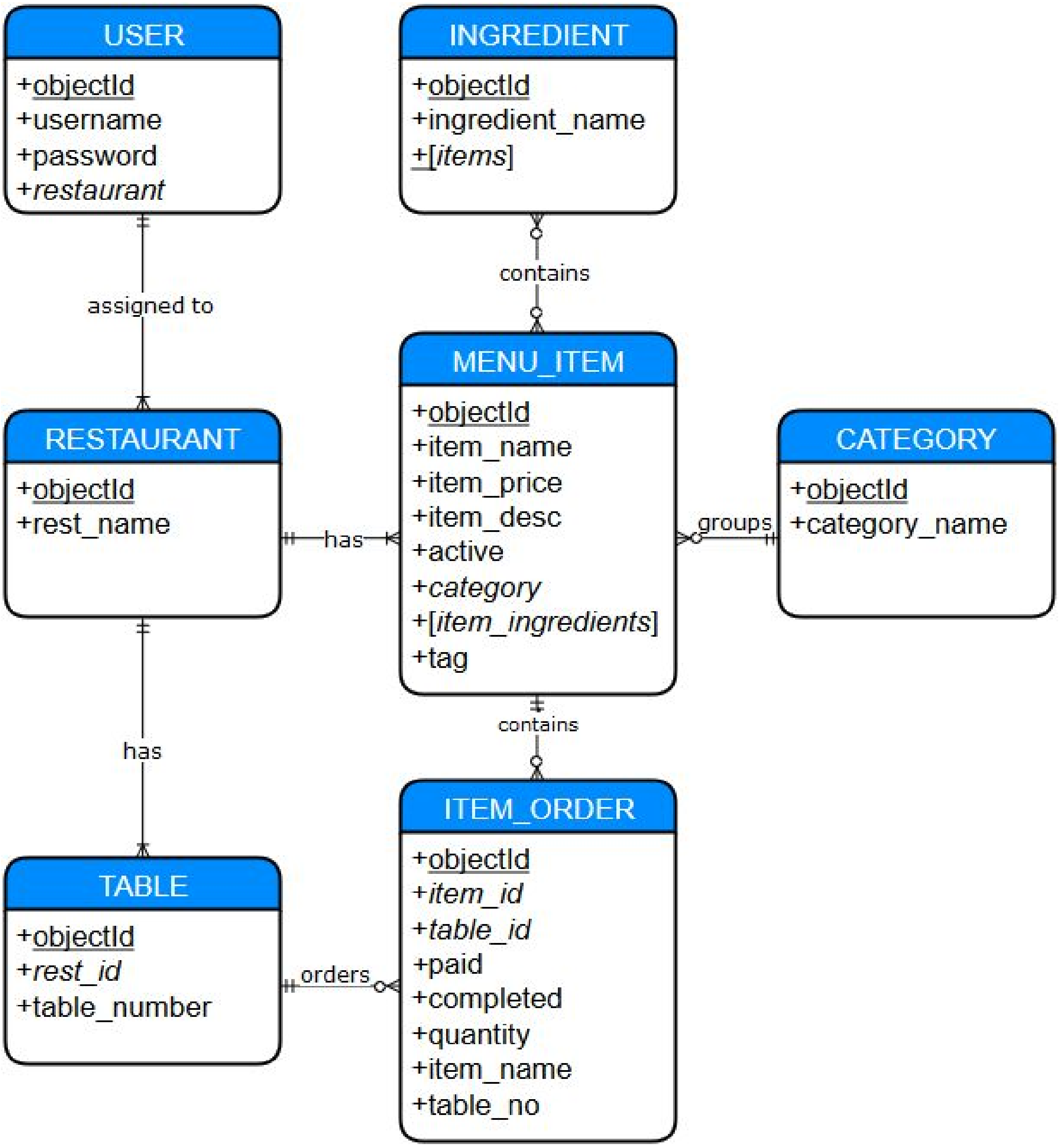
		T19: Implement edit function code for admin user. T20: Implement edit function query for admin user.
5 November 26	The customers should be able to send their orders to the admin and the kitchen. The kitchen account should be able to view queued orders.	Front End: T21: Develop Kitchen order queue Back End: T22: Connect cart menu to the database. T23: Connect customer user to database T24: Develop kitchen order queue function T25: Connect kitchen user to database T26: Implement function code for customer filter T27: Implement function query for customer filter T28: Implement add function code for customer. T29: Implement add function query for customer.

Activity Network Diagram



Entity-Relationship Diagram and Data Dictionary

I. Entity-Relationship Diagram



II. Data Dictionary

TABLE NAME: USER

FIELD NAME	**DATA TYPE**	**LENGTH**	**PK?**	**FK**
objectId	String	10	Yes	No
username	String	10	No	No
password	String	15	No	No
restaurant	Pointer<Restaurant>	10	No	Yes

TABLE NAME: RESTAURANT

FIELD NAME	**DATA TYPE**	**LENGTH**	**PK?**	**FK?**
objectId	String	10	Yes	No
rest_name	String	25	No	No

TABLE NAME: TABLE

FIELD NAME	**DATA TYPE**	**LENGTH**	**PK?**	**FK?**
objectId	String	10	Yes	No
rest_id	Pointer<Restaurant>	10	No	Yes
table_number	Int	2	No	No

TABLE NAME: MENU_ITEM

FIELD NAME	**DATA TYPE**	**LENGTH**	**PK?**	**FK?**
objectId	String	10	Yes	No
item_name	String	15	No	No
item_price	Double	-	No	No
item_desc	String	40	No	No
active	Boolean	-	No	No
category	Pointer<Category>	10	No	Yes

item_ingredients	Array<Pointer<Category>>	-	No	No
tag	String	30	No	No

TABLE NAME: INGREDIENT

FIELD NAME	DATA TYPE	LENGTH	PK?	FK?
objectId	String	10	Yes	No
ingredient_name	String	20	No	No
items	Array<Pointer<Menu_Item>>	-	No	No

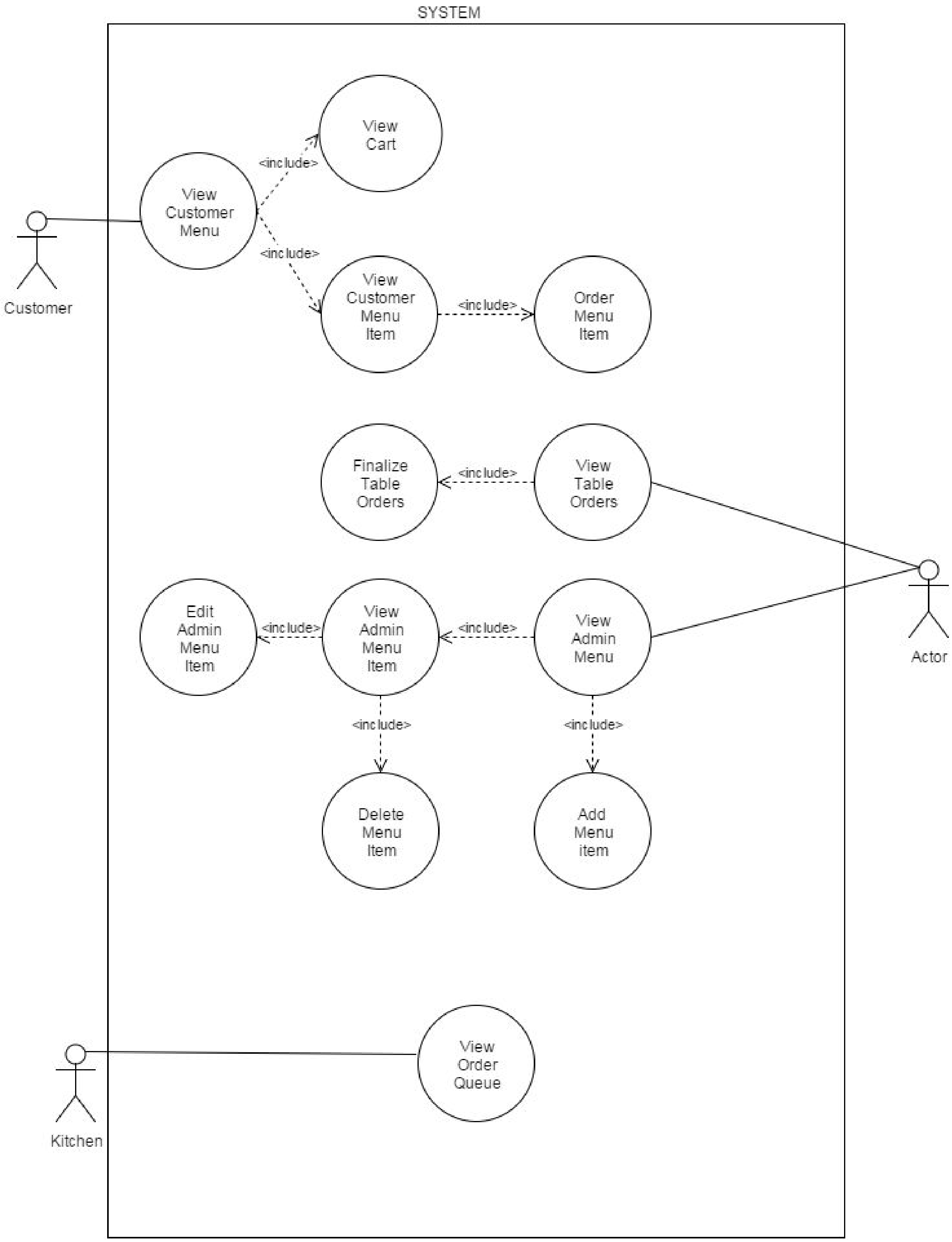
TABLE NAME: CATEGORY

FIELD NAME	DATA TYPE	LENGTH	PK?	FK?
objectId	String	10	Yes	No
category_name	String	20	No	No

TABLE NAME: ITEM_ORDER

FIELD NAME	DATA TYPE	LENGTH	PK?	FK?
objectId	String	10	Yes	No
item_id	String	10	No	Yes
table_id	String	10	No	Yes
paid	Boolean	-	No	No
completed	Boolean	-	No	No
quantity	Int	2	No	No
item_name	String	15	No	No
table_no	Int	2	No	No

Use Case Diagram and Description



Use Case: View Customer Menu
Author: Group E
Date: SEPT-22-2015
Purpose:

- View the current menu of the restaurant

Overview:

- The admin will log on to the app, and will choose the Customer user type. The system will load the menu items from the database into the app for the customers to see.

Cross References: View Cart (Use Case), View Customer Menu Item (Use Case)
Actors: Customer
Normal flow of events:

Actor Actions	System Actions
1. Admin accesses Customer user type	
2. The customer is granted access to the app Customer screen	
	3. The system loads the menu item from the database
	4. The system will load the menu in list form

Exceptional flow of events:

- Step 3 will not succeed if there is no steady Internet connection.

Use Case: View Cart
Author: Group E
Date: SEPT-22-2015
Purpose:

- To see what has the customer ordered so far, including prices of each item and the total.

Overview:

- The customer will see all of the items ordered and added to the cart so far. It includes the menu item names, quantity, and also the computer prices per food ordered and the total cost.

Cross References: View Customer Menu (Use Case)
Actors: Customer
Normal flow of events:

Actor Actions	System Actions
1.Admin accesses Customer user type	
2. The customer is granted access to the apps Customer user screen	
	3. The system loads the menu item from the database
	4. The system will load the menu in list form
5.The customer selects the “View Cart” button	
	6. The system will load the list of menu items the customer has chosen
	7. The system will compute the food prices with the quantity per food, and compute the total price.

Exceptional flow of events:

- Step 3 will not succeed if there is no steady Internet connection.

Use Case: View Customer Menu Item
Author: Group E
Date: SEPT-22-2015
Purpose:

- Enables the customer to look at the details of a certain menu item

Overview:

- Whenever the customer will click a certain menu item, data needed is sent over to another window and will load the necessary details to the corresponding food item

Cross References: View Customer Menu (Use Case), Order Menu Item (Use Case)
Actors: Customer
Normal flow of events:

Actor Actions	System Actions
1.Admin accesses Customer user type	
2. The customer is granted access to the apps Customer user screen	
	3. The system loads the menu item from the database
	4. The system will load the menu in list form
5. The customer selects a menu item from the list	
	6. The system will call a new window
	7. The system will check from the menu item from the database
	8. The system will load the details from the corresponding menu item

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: Order Menu Item
Author: Group E
Date: SEPT-22-2015
Purpose:

- To allow the customer to add the menu item to the cart, alongside the quantity of said menu item

Overview:

- When the customer selects a menu item, they are directed to a profile screen, and will be able to add the menu item from there, alongside the quantity of the menu item for ordering

Cross References: View Customer Menu Item (Use Case)
Actors: Customer
Post Conditions:

- Cart database will be updated.

Normal flow of events:

Actor Actions	System Actions
1.Admin accesses Customer user type	
2. The customer is granted access to the apps Customer user screen	
	3. The system loads the menu item from Parse
	4. The system will load the menu in list form
5. The customer selects a menu item from the list	
	6. The system will call a new window
	7. The system will check from the menu item from the Parse database
	8. The system will load the details from the corresponding menu item
	9. The system will ask the quantity of food .
10. The customer will decide on the quantity and adds to the cart.	
	11. The system will add the information to the cart

	12. The system will add the order to the Parse database

Alternative flow of events:

- In case the Back or Cancel button was pressed in Step 10, the system will not proceed to Step 11.

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: View Order Queue
Author: Group E
Date: SEPT-22-2015
Purpose:

- Check the incoming orders and pull them out if they are done.

Overview:

- The kitchen will see the food that will be incoming, and from which table, and in a queue fashion. The kitchen only needs to press one order and it will be erased off the list, signaling its done.

Cross References: None
Actors: Kitchen
Normal flow of events:

Actor Actions	System Actions
1. The admin will access the Kitchen user type.	
2. The kitchen is granted access to the Kitchen user type	
	3. The system will load the orders from the database in list form

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: View Table Orders
Author: Group E
Date: SEPT-22-2015
Purpose:

- To see the orders of a table, using its table number as index.

Overview:

- Usually, the table number is set right before the customer is presented with menu. The admin will be able to see the tables and their orders.

Cross References: Finalize Table Orders (Use Case)
Actors: Admin
Normal flow of events:

Actor Actions	System Actions
1.The admin is logged in to the Admin user type.	
2. The admin chooses to view the table orders	
	3. The system will load the current orders from the database
	4. The system will show two lists; one for the table lists and the other for the lists of a chosen table.

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: Finalize Table Orders
Author: Group E
Date: SEPT-22-2015
Purpose:

- Finalize the orders of a certain table.

Overview:

- Basically if a table is finalized, all of the orders are to be uploaded to the database, and the kitchen will load these new orders.

Cross References: View Table Orders (Use Case)
Actors: Admin
Post Conditions:

- The Order Queue database will be updated.

Normal flow of events:

Actor Actions	System Actions
1.The admin is logged in to the Admin user type.	
2. The admin chooses to view the table orders	
	3. The system will load the current orders from the database
	4. The system will show two lists; one for the table lists and the other for the lists of a chosen table.
5. The admin decides to finalize the orders from the table	
	6. The system will upload the orders to the database.

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: View Admin Menu
Author: Group E
Date: SEPT-22-2015
Purpose:

- To see the current implemented menu item.

Overview:

- The admin logs in to the app, and accesses the Admin user type, and goes the currently implemented menu.

Cross References: View Admin Menu Item (Use Case), Add Menu Item (Use Case)
Actors: Admin
Normal flow of events:

Actor Actions	System Actions
1. The admin logs in to the app and accesses the Admin user type.	
2. The admin will access the current menu items.	
	3. The system load the current menu items on the database
	4. The system will load the menu items in list form

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: Add Menu Item
Author: Group E
Date: SEPT-22-2015
Purpose:

- The admin will be able to add a new item on the menu.

Overview:

- The admin will be able to access the currently implemented menu and be able to add any menu items. The screen will refresh the contents of the new list to accommodate the latest entry.

Cross References: View Admin Menu (Use Case)
Actors: Admin
Post Conditions:

- The Menu Item database will be updated.

Normal flow of events:

Actor Actions	System Actions
1.The admin logs in to the app and accesses the Admin user type.	
2. The admin will access the current menu items.	
	3. The system load the current menu items on the database
	4. The system will load the menu items in list form
5. The admin will add a new menu item	
	6. The system will now ask the details of the new menu item
7. The admin has finished putting details and adds the menu item	
	8. The system will upload the new menu item to the database.
	9. The system will refresh the list to see the new implemented items.

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: View Admin Menu Item

Author: Group E

Date: SEPT-22-2015

Purpose:

- To view the profile containing the price, description and tags of a menu item.

Overview:

- Whenever the admin will click a certain menu item, data needed is sent over to another window and will load the necessary details to the corresponding food item

Cross References: View Admin Menu (Use Case), Delete Admin Menu Item (Use Case), Edit Admin Menu Item (Use Case)

Actors: Admin

Normal flow of events:

Actor Actions	System Actions
1.The admin logs in to the app and accesses the Admin user type.	
2. The admin will access the current menu items.	
	3. The system load the current menu items on the database
	4. The system will load the menu items in list form
5. The admin selects one of the menu items in the list	
	6. The system will call a new window
	7. The system will check from the menu item from the database
	8. The system will load the details from the corresponding menu item

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: Delete Admin Menu Item
Author: Group E
Date: SEPT-22-2015
Purpose:

- The admin can delete the a menu item

Overview:

- The admin can access the profile from the menu item list., a button will allow the admin to delete said item.

Cross References: View Admin Menu Item (Use Case)
Actors: Admin
Post Conditions:

- The Menu Item database will be updated.

Normal flow of events:

Actor Actions	System Actions
1.The admin logs in to the app and accesses the Admin user type.	
2. The admin will access the current menu items.	
	3. The system load the current menu items on the database
	4. The system will load the menu items in list form
5. The admin selects one of the menu items in the list	
6. The admin will delete the menu item.	
	7. The system will remove the entry from the database.
	8. The system will refresh the list to accommodate the data entries.

Alternative flow of events:

- If the Back or Cancel button is pressed in Step 6, the system will not proceed to Step 7.

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Use Case: Edit Admin Menu Item
Author: Group E
Date: SEPT-22-2015
Purpose:

- The admin can edit any details regarding a menu item.

Overview:

- The admin can select any item in the menu, and with a button, can edit the details of the menu by having the profile screen pop-up, and change any details regarding the item.

Cross References: View Admin Menu Item (Use Case)
Actors: Admin
Post Conditions:

- The Menu Item database will be updated.

Normal flow of events:

Actor Actions	System Actions
1.The admin logs in to the app and accesses the Admin user type.	
2. The admin will access the current menu items.	
	3. The system load the current menu items on the database
	4. The system will load the menu items in list form
5. The admin selects one of the menu items in the list	
6. The admin will delete the menu item.	
	7. The system will get the details regarding the item.
	8. The system will ask for the details
9. The admin will save any changes	
	10. The system will refresh the list to accommodate the data entries.

Exceptional flow of events:

- Step 3 will not succeed if there is no steady internet connection.

Lessons Learned

When we planned on how the team organization will work, we decided to think not only what would be best on how would the team function together, but on how would we be able to finish the project, without sacrificing too much time or resources. This allowed the team to greatly improve the product and at the same time fix the errors, for most of the parts. Communication also played a big part of the project.

During the project development, we encountered lots of difficulty in managing time and learning new skills. It was very stressful, but we all came out of it with new knowledge and a better understanding of creating software and applications. We learned that managing a team is extremely difficult, especially when said team is also busy with other things.

We designed the project itself by separating the front end and the back end of the project. That way, all of the members are building up the product alongside both ends, so as to save a little bit of time. There were times, however, when it was difficult for the back end to test their code when there was no front end yet (especially at the early stages).