Group 11
Blazin_Seven

Milestone 4: Requirements and Preliminary Design for the Food Ordering System

1. <u>Summary Use Cases</u>

1) Increase Restaurant Exposure

Level: Summery
Actor: User - Restaurant
Goal: Increase exposure through advertisement
Activities: A ‘mom and pop’ style restaurant might wish to increase exposure and can do so by using this desktop application. This could be accomplished by offering new and unique food items that users might desire, or even by announcing their location through the sort by distance field.
Quality: This is a helpful feature for restaurants but is not the main use case. It should be working at a usable standard.
Version: October 5th, 2017

2) Reviewing Customer Feedback

Level: Summery
Actor: User - Restaurant
Goal: Improve quality of restaurant service
Activities: A participating restaurant can use user provided feedback to improve their menu options and restaurant experience. A restaurant can go to their account page and to a convenient customer comments section. From reading the comments, the restaurant can make the necessary changes. For example, if a particular item has been repeatedly deemed “not spicy enough”, changes could be made.
Quality: This is a helpful feature for restaurants but is not the main use case. It should be working at a usable standard.
Version: October 5th, 2017

3) FOS Restaurant Notification

Level: Summary
Actor: User - Restaurant
Goal: To inform participating restaurants of incoming orders
Activities: A customer ordering food from a participating restaurant will have their order and its information sent in real time the applicable restaurant. Expected wait times are provided to the user, and once the food is prepared, another notification is sent to the user.
Quality: Food Ordering is the main function of this program. This use case should be working to a very high standard.
Version: Oct 3rd , 2017

4) Distance Search

Distance Search
Level: Summary
Actor: User - Customer
Goal: To show the customer the closest restaurants along with their desired food item.
Activities: A customer is wishing to order a food item but does not want to travel far to pick up an item. The customer can browse for the nearest restaurants or they can enter their desired food item into the search bar and choose the metric option of distance. The search will return a list of restaurants with the desired food item sorted from closest to farthest.
Quality: This is a main use case in the system and should be working to a high standard.
Version: October 5th, 2017

5) Price Search

Level: Summary
Actor: User - Customer
Goal: To show the customer the cheapest prices of their desired food item.
Activities: A customer who is conscientious about money wants to search for the restaurants that are within their spending limit or they can search for the cheapest price of their desired food item. The customer will enter their desired food item into the search bar and choose the metric option of price. The search will return a list of restaurants with the desired food item sorted from cheapest to most expensive.
Quality: This is a main use case in the system and should be working to a high standard.
Version: October 5th, 2017

6) Rating Search

Rating Search
Level: Summary
Actor: User - Customer
Goal: To show the customer the best rated restaurants offering their desired food item.
Activities: A customer wants to find the best restaurants in the city or want to find the best version of their desired food item in the city. The customer will enter their desired food item into the search bar and choose the metric option of highest rated. The search will return a list of restaurants with the desired food item sorted from highest rated to lowest rated.
Quality: This is a main use case in the system and should be working to a high standard.
Version: Oct 5th, 2017

7) Wait/Prep Time Search

Wait Time Search
Level: Summary
Actor: User - Customer
Goal: To show the customer the restaurants with the lowest waiting time that has their desired food item.

Activities: A customer does not want to wait long to for their desired food item. The customer will enter their desired food item into the search bar and choose the metric of the lowest prep/wait time. The search will return a list of restaurants sorted from lowest wait time to highest wait time that have the customer's desired food item.
Quality: This is a main use case in the system and should be working to a high standard.
Version: Oct 5th, 2017

8) Adding Reviews

Adding Customer Reviews
Level: Summary
Actors: User - Customer
Goal: To allow user to add a review about a food item
Activities: A customer, who recently ordered and finished eating a food item wishes to add a review about the food quality. They can open the program, go to their recently ordered food items and easily add a review including a rating out of 5 stars and a comment accompanying their rating.
Quality: Ratings are one of the metrics that the system uses to help the customer to search for food. Thus, while the comment section is not a critical component, the five star rating is and thus should work to a high standard.
Version: October 5th, 2017

9) Placing Orders

Level: Summary
Actor: User - Customer
Goal: To avoid queuing at the restaurant and make ordering food hassle-free.
Activities: A busy and hungry customer does not want to wait for their ordered food at a restaurant because of time constraints. The customer can search for a restaurant or their desired food item with their choice of a metric option and place an order. The order can be placed to prepare the food right away or to have it ready by an allotted time later on during the day.
Quality: This is a main use case in the system and should be working to a high standard.
Version: October 2, 2017

10) Payment Option

Level: Summary
Actor: User - Customer
Goal: To allow the customer to pay their bill through the program
Activities: When the customer finishes ordering their food, the customer may want to save time and pay through the program. They will find two options that will allow them to pay when the food is picked up or through their account. The customer will choose to pay through the program and they can save/unsave their credit card information to their account. Now, the user can easily pay for their order.

Quality: Payment options will be made available. However, it is not the main use case of the program and will be working to a usable standard.
Version: October 1st, 2017

11) Save favourite foods

Level: Summary
Actor: User - Customer
Goal: To allow the customer to save their favourite foods
Activities: A customer has a regular food item that they frequently order. Instead of having to search for the item and then placing an order, the user can save their favourite foods to their account. Then in the future, they will simply have to open their account page, go to their saved items, and place an order.
Quality:
Version: October 7th, 2017

12) Browse Restaurant Options

Level: Summary
Actor: User - Customer
Goal: Allow a user to peruse different food options in their city.
Activities: A user may not wish to order food, but they may still want to know their options. They would use the search function with their desired metric to display different orderable options so that next time they do want to order food, they have a better understanding of what's available.
Quality:
Version: October 5th 2017

13) Search Restaurants Information

Level: Summary
Actor: User - Customer
Goal: Allow a user to search for a specific restaurant in the database
Activities: A user may want to look up information about a specific restaurant. Instead of searching for food items of the restaurant or finding the restaurant through the browser, the user can enter the restaurant's name in the search bar and quickly find the information that they are looking for. This information may include location, menu, open/close times, or ratings.
Quality: This use case will include the base function that all other searches will build from. Thus, it should be a very high quality.
Version: October 12th 2017

14) User Account Creation and Log In

Level: Summary
Actor: User – Customer & Restaurant

Goal: Allow customers to create accounts to save personal information and restaurants to join the program's database of restaurants.
Activities: A customer does not want to repeatedly enter their information each time an order is placed. Creating an account will allow them to save information such as name, location, credit card information, and favourite foods to make the process of ordering go faster. A restaurant wants to increase their exposure to new customers and will create an account to be included in the searches by the customers. In their account, information such as menu, location, open/close times will be added for searches.
Quality: Restaurant account creation is important for this program to exist. Thus, this should be working at a high quality.
Version: October 12th 2017

2. Fully-Dressed Use Cases

Use Case 1: Search for Distance

Scope: Food Ordering System Application

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Wants an easily readable display of restaurants or restaurants with their desired food item organized from closest to farthest to themselves.
- Restaurant: Wants to ensure that if a customer is within close distance to their restaurant, then their restaurant is appearing in the distance search results and their address is accurately listed.

Preconditions: The user should select the distance metric through the search bar for a food item or through the browser for restaurants. The customer's location is authenticated through their account information.

Success Guarantee and Post conditions: Search correctly returns a list of restaurants ordered from closest to farthest displaying the measure of distance in kilometers.

Main Success Scenarios:

1. Customer opens and logs into the application
2. Customer enters their desired food item into the search bar.
3. Customer chooses the metric of distance.
4. Program authenticates the customer's location.
5. Program will search the database for restaurants containing the customer's desired food item and return a list.
6. Program will organize the list of restaurants from closest to farthest by comparing the restaurant's location and the customer's location.

7. The program returns a list of restaurants containing the customer's desired food item ordered from closest to farthest to the customer in the UI.

Extensions:

2a. The customer is not looking for a specific food item and only wants to browse for restaurants closest to them:

1. Customer chooses the browsing option.
2. Program will search through the database for the restaurants that are closest to the customer
3. The program will sort the list from the closest to the farthest restaurants using the customer's location
4. The program returns the list of restaurants to the customer in the UI.

2b. Customer's desired food item is not in database

1. Return a statement saying item cannot be found

4a. The customer's location cannot be authenticated.

1. Customer's address location from account is used.

*a. At anytime the application crashes.

1. Customer closes and reopens the application.

Special Requirements:

- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:

- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: Since it is the main function of the application, the distance search will be used often.

Open Issues:

- Will there be a limit or a max distance that the restaurants will be chosen from?
- Can the customer choose for the number of closest restaurants?
- Can the customer choose what the max distance should be?
- Can this metric be combined with other metrics?

Use Case 2: Search for Price

Scope: Food Ordering System Application

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Wants an easily readable display of restaurants or restaurants with their desired food item organized from cheapest to most expensive

- Restaurant: Wants to ensure that if the restaurant contains the customer's desired food item, then they are included in the returned list to the customer along with their accurate information.

Preconditions: The user should select the price metric through the search bar for a food item or through the browser for restaurants.

Success Guarantee and Post conditions: Search correctly returns a list of restaurants ordered from cheapest to most expensive.

Main Success Scenarios:
1. Customer opens and logs into the application
2. Customer enters their desired food item into the search bar.
3. Customer chooses the metric of price
4. Program will search the database for restaurants containing the customer's desired food item and return a list.
5. Program will sort the list of restaurants from cheapest to most expensive
6. The program returns a list of restaurants containing the customer's desired food item ordered from cheapest to most expensive to the customer in the UI.

Extensions:

2a. The customer is not looking for a specific food item and only wants to search for cheap restaurants in the city.
   1. Customer chooses the browsing option.
   2. Program will search through the database for a list of restaurants
   3. The program will sort the list from the cheapest to the most expensive restaurants
   4. The program returns the list of restaurants to the customer in the UI.

2b. Customer's desired food item is not in database
   1. Return a statement saying item cannot be found

*a. At anytime the application crashes.
   1. Customer closes and reopens the application.

Special Requirements:
- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:
- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: Since it is the main function of the application, the price search will be used often.

Open Issues:
- Can the customer choose the highest price to search under?
- Can the customer choose how many restaurants to get returned to them?
- Can the customer choose the range of prices to check for?

- Can this metric be combined with other metrics?

Use Case 3: Search for Ratings

Scope: Food Ordering System Application

Primary Actor: Customer

Stakeholders and Interests:
- Customer: Wants an easily readable display of restaurants or restaurants with their desired food item organized from the most highly rated to the lowest rated
- Restaurant: Wants to ensure that if the restaurant is positively rated, then they are included in the returned list to the customer along with their accurate information.

Preconditions: The user should select the rating metric through the search bar for a food item or through the browser for restaurants.

Success Guarantee and Post conditions: Search correctly returns a list of restaurants ordered from the highly rated to the lowest rated.

Main Success Scenarios:
1. Customer opens and logs into the application
2. Customer enters their desired food item into the search bar.
3. Customer chooses the metric of ratings
4. Program will search the database for restaurants containing the customer's desired food item and return a list.
5. Program will organize the list of restaurants from highest rating to the lowest ratings
6. The program returns a list of restaurants containing the customer's desired food item ordered from the highest rated to the lowest rated to the customer in the UI.

Extensions:

2a. The customer is not looking for a specific food item and only wants to search for the best restaurants in the city.
1. Customer chooses the browsing option.
2. Program will search through the database for a list of restaurants
3. The program will sort the list from the highest rated to the lowest restaurants
4. The program returns the list of restaurants to the customer in the UI.

2b. Customer's desired food item is not in database

5. Return a statement saying item cannot be found

*a. At anytime the application crashes.
1. Customer closes and reopens the application.

Special Requirements:
- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:
- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: Since it is the main function of the application, the rating search will be used often.

Open Issues:
- Can the customer choose how many restaurants to get returned to them?
- Can the customer choose the range of ratings of restaurants to search from?
- Can the customer combine this metric with other metrics?

Use Case 4: Search for Waiting Time

Scope: Food Ordering System Application

Primary Actor: Customer

Stakeholders and Interests:
- Customer: Wants an easily readable display of restaurants or restaurants with their desired food item organized from lowest waiting time to highest waiting time
- Restaurant: Wants to ensure that if the restaurant contains the customer's desired food item and have a relatively low wait time, then they are included in the returned list to the customer along with their accurate information.

Preconditions: The user should select the wait time metric through the search bar for a food item or through the browser for restaurants.

Success Guarantee and Post conditions: Search correctly returns a list of restaurants ordered from lowest wait time to the highest wait time.

Main Success Scenarios:
1. Customer opens and logs into the application
2. Customer enters their desired food item into the search bar.
3. Customer chooses the optional metric of wait time
4. Program will search the database for restaurants containing the customer's desired food item and return a list.
5. Program will sort the list of restaurants from lowest wait time to highest wait time
6. The program returns a list of restaurants containing the customer's desired food item ordered from lowest wait time to highest wait time to the customer in the UI.

Extensions:

2a. The customer is not looking for a specific food item and only wants to search the restaurants with the lowest wait time in the city.

1. Customer chooses the browsing option.

2. Program will search through the database for a list of restaurants
3. The program will sort the list of restaurants from the lowest wait time to the highest wait time
4. The program returns the list of restaurants to the customer in the UI.

2b. Customer's desired food item is not in database

5. Return a statement saying item cannot be found

*a. At anytime the application crashes.

1. Customer closes and reopens the application.

Special Requirements:

- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:

- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: Since it is the main function of the application, the wait time search will be used often.

Open Issues:

- Can the customer choose how many restaurants they want to be returned to them?
- Can this metric be combined with another metric?
- Will the metric use average prep time or will they have the restaurant's wait time have real time updates?

Use Case 5: Search Restaurant Information

Scope: Food Ordering System Application

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Wants to generate information about the restaurant their interested in
- Restaurant: Wants their information, such as their location and menu, to be accurate and to be easily accessible to the customer.

Preconditions: Customers input the restaurant name in the search bar

Success Guarantee and Post conditions: Related restaurant name and information will appear in the UI if the restaurant is in the database.

Main Success Scenarios:

1. Customers input the name or some related words into the search bar and press "go"
2. Restaurants that fit the situation will show up at the center of the main interface.
3. Customers press the name of the restaurant to generate more information.

Extensions:

2a. The restaurant is not exist in our database

1. The customer will be notified that there is no restaurant matching their description in the database.

2b. The customers use different upper and lower cases which may return no or unwanted results

Special Requirements:

- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:

- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: Many times, a customer just wants to look at the restaurants information. This use case will be used frequently because it is inconvenient to search for a food item of the restaurant or navigate through browsing options to find a restaurant's information.

Use Case 6: Placing an Order

Scope: Food Ordering System Application

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Wants to place an order of their desired food item of a restaurant and wants confirmation that the order is confirmed. Paying customers want to be ensured that the correct amount of money has been received by the restaurant.
- Restaurant: Wants to be notified of the incoming order in real time. Details such as which food items, the time when they should be prepared by, and the customer's information (name, order number, payment information) should be also be received and be accurate.
- Wants to ensure that money has been received by customers who paid through the application.

Preconditions: Customer must choose one or more food items to order

Success Guarantee and Post conditions: The correct order is placed. Money transacted through the application is validated. Both the customer and the restaurant receive their respective confirmation notifications. Restaurant begins to prepare the customer's desired food item.

Main Success Scenarios:

1. Customer opens and logs into the application
2. Customer enters their desired food item into the search bar and their choice of a metric option.

3. Program will search the database for restaurants containing the customer's desired food item and return a sorted list to the customer.
4. The customer can choose their food item and click a button to place an order on it
5. The customer will choose the option of having the food prepared right away
6. The restaurant will be notified of the incoming food order in real time
7. The customer will choose the option of paying through the application
8. The payment will be validated.
9. The customer will get a confirmation notification of their order being processed.

Extensions:

4a. Customer may want to pick up their order at a later time in the day instead of being picked up right away.

1. The customer will choose the option of picking up their order at a later time.
2. The customer will select the exact time at which they would like their order to be ready by.

5a. Customer may decide to pay at the restaurant

1. The customer will choose the option of paying at the restaurant
2. When the order is picked up, the restaurant will receive the payment in person.

6a. The order does not go through and the restaurant does not receive the notification

1. If the customer does not receive a confirmation notification, this indicates that the order did not go through.
2. The customer can try to place the order again
3. If problem persists, the customer can contact the system administrators.

8a. The payment from the customer does not successfully get transacted to the restaurant

1. If the customer does not receive a confirmation notification, this indicates that the order did not go through.
2. The customer can try to place the order again
3. If problem persists, the customer can contact the system administrators.

9a. The order goes through, both the customer and restaurant get confirmation notifications, but the restaurant does not prepare the food in time.

1. Customer should relay this information in the review section
2. System administrators can implement a strike system to discourage restaurants from preparing orders late.

*a. At anytime the application crashes.

1. Customer closes and reopens the application

Special Requirements:

- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:

- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: Placing an order is one of the main functions of the application and will be used frequently.

Open Issues:
- How many strikes can a restaurant have?
- Will there be a compensation for the customer?

Use Case 7: Account Creation and Logging into Account

Scope: Food Ordering System Application

Primary Actor: Customer and Restaurant

Stakeholders and Interests:
- Customer: Wants program to save their information including name, address, credit card information, and favourite foods.
- Restaurant: Wants to include their restaurant in the searches to increase exposure to their restaurant. Wants address, menu, opening/closing hours, and pictures to be added to their account information.

Preconditions: Users without an account should use the "sign up" feature and users with the account should use the "log in" feature.

Success Guarantee and Post conditions: Customers and Restaurant users should be able to successfully log into their account with correct password. Their account information is saved will continue to appear each time users log into their accounts.

Main Success Scenarios:
1. Customer without an account will create one by pressing the "sign up" button on the landing page.
2. Customer will register by entering their username, password, and the, "Sign Up as a Customer" feature will be chosen.
3. The customer can complete their registration by clicking the button that says "Sign Up Right Now!"
4. Customer will receive a notification that the registration was a success.
5. Customer can now enter their personal information such as name, address, credit card information and save their favourite foods.
6. The customer's information will be saved to the application's database.

Extensions:

1a. The customer has already created an account.
1. The customer can log in to their created account
2. Customer can add/change their personal information
3. Customer can log out.

1b. A restaurant without an account will create a new account.
1. The restaurant should press the "sign up" button on the landing page.
2. The restaurant should register with username, password, restaurant ID and the "Sign Up as a Restaurant" feature will be chosen

3. The restaurant will receive a notification that the registration was a success
4. The restaurant can proceed to add information to their account such as address, open/close times, menu, pictures etc.
5. The restaurant's information will be saved to the application's database.

1c. The restaurant has already created an account.

1. The restaurant can log in to their created account
2. Restaurant can add/change their personal information
3. Restaurant can log out.

2a. The customer username has already been used

1. Customer will get a notification indicating that the entered username has already been used.
2. The customer will enter a different username and can continue to do so until they create a unique username.

2b. The restaurant username and ID has already been used

1. Restaurant will get a notification indicating that the entered username or ID has already been used.
2. The restaurant will enter a different username and can continue to do so until they create a unique username. Or the restaurant must recheck their restaurant ID.

Special Requirements:
- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:
- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: Customers will prefer to log-in into their account to skip entering their information such as their address and phone number and to save time. Thus, the log-in feature will be used often.

Open Issues:
- How will the program deal with customers registering as restaurants and vice versa?
- What if the customers/restaurants forget their username and/or password?

Use Case 8: Giving Reviews

Scope: Food Ordering System Application

Primary Actor: Restaurant

Stakeholders and Interests:
- Customer: Wants to give a review that includes a rating out of five stars and a comment that relays information about the restaurant such as the restaurant's food, staff, atmosphere, cleanliness, actual wait time, etc.

- Restaurant: Will want to their customers to give reviews of their service to improve their rating and get more exposure through the Rating Search and/or to see how they can further improve their service to their customers.

Preconditions: Customer must have ordered, picked up, and eaten their food from a restaurant available in this application

Success Guarantee and Post conditions: Order is placed and picked up by customer. Customer eats their food item and will give a review of the food through their account page under recent orders. The review information from the customer will be integrated with the current information of the restaurant and the ratings will be recalculated.

Main Success Scenarios:

1. Customer finds their desired food item and places the order.
2. Customer picks up the food item, experiences the restaurant and their customer service.
3. Customer eats the food item and wants to give a review of what they ordered and of their experience of the restaurant.
4. Customer goes to their account page, to recent orders, and chooses the review option.
5. Customers give a rating out of 5 stars and gives a comment explaining their choice.
6. Rating and comment information will be taken and the Restaurant's current rating will be recalculated

Extensions:

2a. Order may be placed but food is not picked up

1. Customers will be given a strike system that will discourage them from not picking up orders and wasting food. Too many strikes could ban the customer from placing orders.

5a. Customers may give a review that is offensive, derogatory, or unrelated to the restaurant's service.

1. Customer's reviews will quickly be screened checking for specific words that could prevent such comments.
2. If a customer's review contains these words, the customer's review will not be posted and may get a strike. Too many strikes could ban the customer from giving reviews.

Special Requirements:

- Any working operating system with the FOS desktop application downloaded

Technology and Data Variations List:

- Search on Mac OS using the .jar of FOS
- Search on Windows using the .exe of FOS

Frequency of Occurrence: The Rating Search is completely dependent on customers giving their review of restaurants. Without the customers giving reviews, this search would not work well. Thus this use case should occur often, hopefully as often as the number of orders placed.

Open Issues:

- Is there a way to help encourage customers to place reviews (coupons, point system, some rewards)
- Will there be a way to show if the restaurant that has been ordered from is on the customer's preference list with the review.
- Could orders that weren't picked up be donated/distributed to homeless shelters?

3. Supplementary Specification

1. Security

- Our system is set up in a way that browsing and searching through restaurants and their menus do not require user authentication but placing orders does require user authentication. This means that our system users do not need an account to look up restaurants but do need an account on our system to make an order and save the food preferences.

2. Human Factor

- Restaurant/Customer will be using the FOS (Food Ordering System) software on different displays. Therefore our system text and windows design will be proportional and legible for all screens on all different display sizes.

- We will also be avoiding colors associated with color blindness and all other common form of eye defects.

- Speed, easy usability and error-free processing are also important for our system.

- Our system will have an appealing design including pictures and graphics. These visuals are important in facilitating our system users to make their food ordering decisions.

3. Implementation Constrains

- We are using Java, JavaFx and SQL technologies to develop our system as these are the most widely known languages known by our group members. We are predicting it will help us improve our systems in the long term with porting, supportability, and also to ease further development in future.

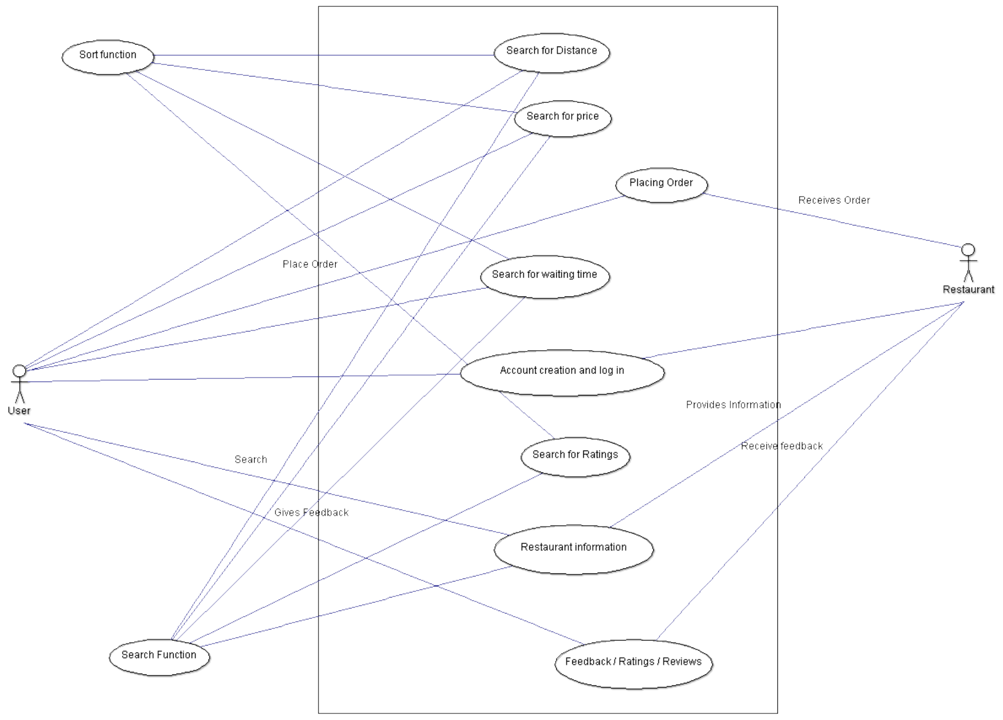
4. Free Open Source Component

- We will also be using some open source java technology components on this project. Though we have not completely decided all the open source components we will be using, one definite component that we will be using is the integration of Google Maps into our system to generate information about the distance between our system user and the restaurants they are browsing.

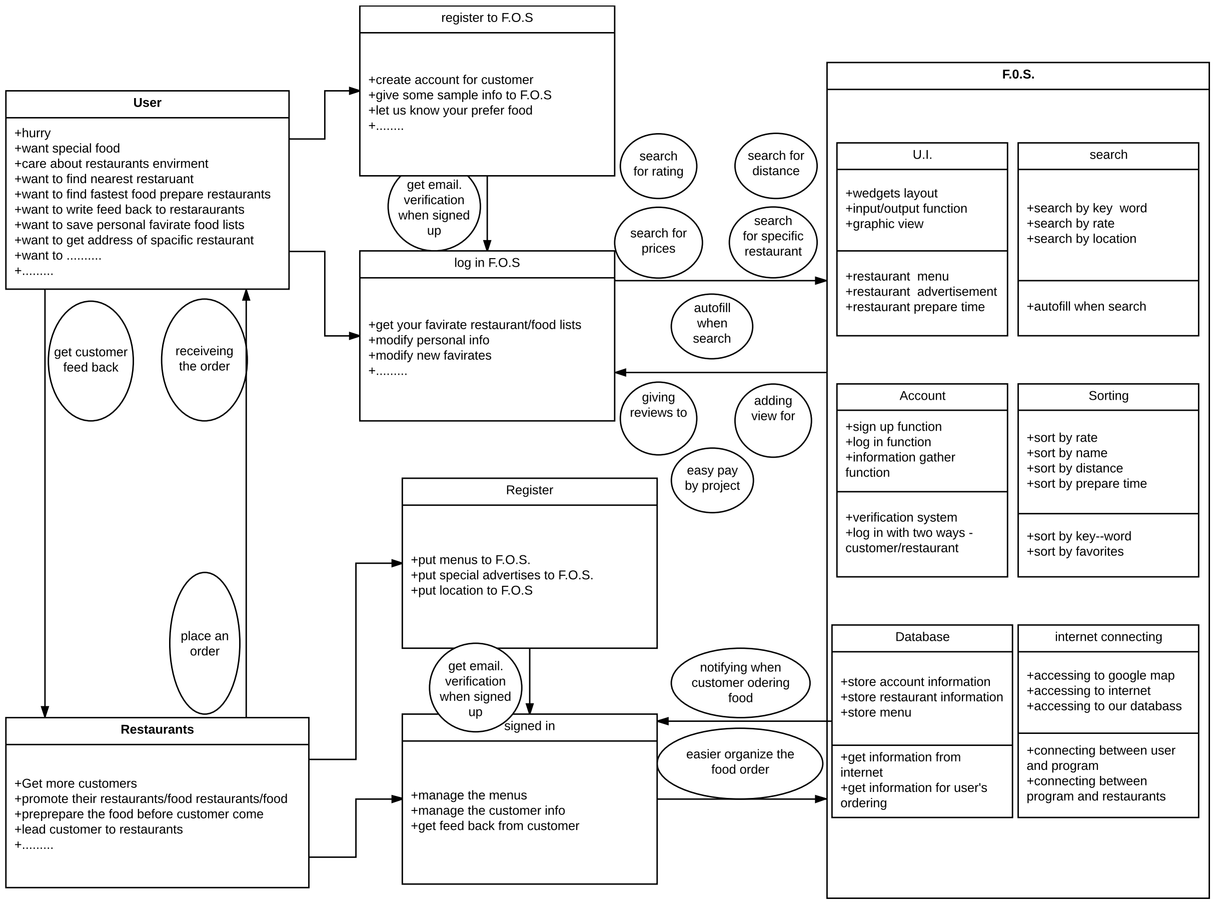
5. Software Interface

- Since we will be incorporating software like Google Maps and SQL Database into our system, we will be implementing our interface in such a way that all the stated above incorporation will be simple and hassle free.

4. Use Case Diagram



5. Domain Model


register to F.O.S
+create account for customer
+give some sample info to F.O.S
+let us know your prefer food
+........
User
+hurry
+want special food
+care about restaurants envirment
+want to find nearest restaurant
+want to find fastest food prepare restaurants
+want to write feed back to restaraunts
+want to save personal favirate food lists
+want to get address of spacific restaurant
+want to ....
+.........
get email. verification when signed up
log in F.O.S
+get your favirate restaurant/food lists
+modify personal info
+modify new favirates
+.........
search for rating
search for distance
search for prices
search for specific restaurant
autofill when search
giving reviews to
adding view for
easy pay by project
F.0.S.
U.I.
+wedgets layout
+input/output function
+graphic view
+restaurant menu
+restaurant advertisement
+restaurant prepare time
search
+search by key word
+search by rate
+search by location
+autofill when search
Account
+sign up function
+log in function
+information gather function
+verification system
+log in with two ways - customer/restaurant
Sorting
+sort by rate
+sort by name
+sort by distance
+sort by prepare time
+sort by key--word
+sort by favorites
Database
+store account information
+store restaurant information
+store menu
+get information from internet
+get information for user's ordering
internet connecting
+accessing to google map
+accessing to internet
+accessing to our databass
+connecting between user and program
+connecting between program and restaurants
get customer feed back
receiveing the order
place an order
Register
+put menus to F.O.S.
+put special advertises to F.O.S.
+put location to F.O.S
get email. verification when signed up
signed in
+manage the menus
+manage the customer info
+get feed back from customer
notifying when customer odering food
easier organize the food order
Restaurants
+Get more customers
+promote their restaurants/food restaurants/food
+preprepare the food before customer come
+lead customer to restaurants
+.........

9. Implementation

9.1 - A running prototype that has a skeleton implementation, and implements two primary success cases has been attached in the same folder as this document. However, this prototype does not include the search function and will be merged in the near future.

9.4 - ReadMe.txt (Also included as a separate file in the same folder)

8.1 Running Prototype:

Prototype name: F.O.S.jar and Food-Ordering-System.zip
F.O.S.jar can run on any machine that has Java VM. It is opened when a user double clicks it. There is also F.O.S.exe that can be run on Windows machines, it is run the same way (double click). Food-Ordering-System files need to be opened in a text editor or IDE.

If anyone wants to run the code in InteliJ (An IDE), they need to install jdbc postgresql driver first. Then run the file and the main interface will comes out.
Part of the registration function and Search function is completed.
The project uses Postgresql Database. However, users do not need to install the Postgresql driver because the prototype jar already contains the driver. Users only need to open the jar file.

8.2 Implementation of two Primary Success Scenarios:

The first success scenario is the Sign-in and Sign-up use case (Use case 1).

Step 1: A user can open our application and press the "Sign up" button to sign up.

Step 2: The user enters a username (E.g. "abc123") and password (E.g. "qwerty"). The user then checks "Sign up as customer" and clicks "Sign Up right Now!". Now the user is successfully registered, and the OptionPanel "You have been successfully Signed in" will be displayed. Note, that the password and the username will be uploaded to our postgresql. Registering with a previously existing username is not allowed.

Step 3: Now the user needs to enter their information. Since the registration system is not totally complete, we only need their name, phone number and e-mail address. The use can also chose to save their preferred food. Now press save button and quit.

Step 4: Use the "Log in" feature, and enter the username and password the user previously registered and press submit. After a short wait the OptionPanel "Okay, Login" will show up to let the customer know they're successfully logged in. All the information is saved in our database, so if the customer inputs the wrong username or password, they will not be able to log in. Instead, an OptionPanel "Wrong username or password" will be displayed

The second scenario is the "Search Restaurant" use case (use case 2). Since the search function is not totally complete, only the "search restaurant" function is implemented.

Step 1: The user opens the application and inputs the name of the restaurant they want to order from. They then press "Go".

Step 2: If the restaurant searched exists, the result will show up in the middle of the main interface. However, if there is no matched result, ErrorOptionPanel "Sorry we can't find that in our system" will be displayed.

Step 3: Users can click the restaurant name showed in the middle of the interface to generate more information about the restaurant.

8.3 Skeleton Implementation:

We are using java so the skeleton should be in the “java class” and “package” form.

Package database:
(This package contains the classes necessary for connecting to the database postgresql)

```
—- AddCustomers.java              // Adding customer information into database
—- AddRestaurants.java            // Adding restaurant information into database
—- AddUser.java          // Adding User (customer or restaurant) info into database
—- DisplayRestaurant.java         // Display the restaurant information (for developer use)
—- GoConnection.java              // Main function connecting to database
—- Login.java                     // Check if the user input the correct username or not
—- AddFeedback.java               // Adding feedback to the restaurant
-- SearchRestaurants.java // Search restaurant name in the database
```

Package Interface:
(This package contains the main interface)

```
—- Main.java             // The main interface (Panel). Combines Left, Centre and Right
—- Centre.java                    // The centre panel (part) of the main interface.
—- Left.java             // The left part of the main interface
—- Right.java            // The right part of the main interface
-- Registration.java    // The user registration and login part of the interface
```

Package postgresql:
(This package contains the postgresql.jar so user can user our file directly)

```
-- postgresql-42.1.4.jar  // the driver of postgresql
```

Package Map:
(This package contains the classes necessary for connecting to the Google map if we decide to use Google map later, so nothing in the package for now)

Package Order:
(This package contains the classes necessary for placing orders. We are not sure how to do it now so nothing in the package for now)

10. Project Plan

Project Plan

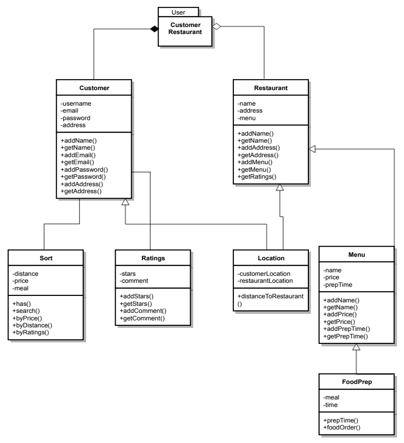
Activity	Duration (hr)	Member(s)
Presentation	7	Emmanuel, Duke, Aparna, Josh, Ridwan, Eric, Yinsheng (14% each)
Search	15	Duke (100%)
User Login Information (Database)	20	Yinsheng(100%)
Interface	18	Eric (100%)
UML Diagram	4	Emmanuel (100%)
Sorting	5	Duke (100%)
Use Case Diagram, Project and Group Logo	5	Ridwan (100%)
Milestone 2, 3	6	Emmanuel, Duke, Aparna, Josh, Ridwan, Eric, Yinsheng (14% each)
Project Plan	2	Emmanuel (100%)
Meeting Minutes	2	Josh (100%)
Milestone 2, 3 Write-up	8	Josh (100%)
Summary and Fully Dressed Use Cases	4	Aparna (70%), Josh (30%)
Supplementary Specification	2	Ridwan(100%)
Readme.txt and 2 Fully Dressed Use Cases	1	Duke (100%)
Domain Model	3	Eric(100%)
Milestone 4 Compilation	6	Josh (80%), Aparna (20%)

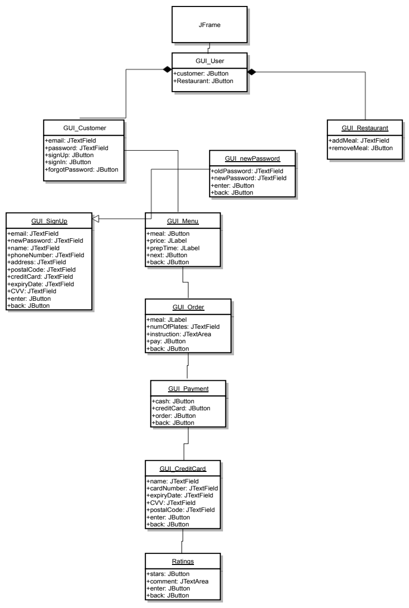
New Tasks

Activity	Duration(hr)	Member(s)
Email Verification	20	Yinsheng, Emmanu
Autofill	10	Eric
Sort by Distance, and Wait time	1	Duke
Access Google Maps	10	Duke, Aparna, Josh
User Interface	30	Eric, Ridwan
Place Order	15	Yinsheng, Emmanuel
Payment System	15	Josh, Ridwan

Notify Restaurant of Order	10	Yinsheng, Aparna, Eric, Emmanuel, Duke, Josh, Ridwan
System and User Testing	10	Yinsheng, Aparna, Eric, Emmanuel, Duke, Josh, Ridwan

11. UML Diagram





12. Meetings

Meeting September 11th
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013),
Duration: 60 minutes
Summary: This was our first meeting and it mostly consisted of introductions. We told everyone a little about ourselves including our names. This meeting lasted around an hour.

Meeting September 12th
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013),
Duration: 120 minutes
Summary: This meeting we planned and talked about possible project ideas. As a group, we decided that the highest mark from milestone 0 would be the project that we'd go with. We also used this time to pick our group name.

Meeting September 19th
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 120 minutes
Summary: This meeting we decided on our project idea. Eric got the highest score on Milestone 0, so we decided to go with his idea. Our project will be a type of "Food ordering System". We also worked out some basic functionality as well as very rough idea for design.

Meeting September 21st
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 120 minutes
Summary: This meeting was mostly spent working on the third milestone. As a group, we talked about possible use case scenarios. We also spent time thinking about work decomposition and we delegated tasks.

Meeting September 26th
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 120 minutes

Summary: This meeting we the finished planning for milestone 3 and we then started to think about the project proposal presentation. After we finished our ideas for the milestone, we talked about future work, and potential problems. We worked out possible ways to work around our lack of hardware; this in particular was in relation to informing restaurants about incoming orders.

Meeting September 28th
Attendance: This meeting was attended by Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 120 minutes
Summary: This meeting was all about the project proposal presentation. Aparna made a rough outline of some slides beforehand and we spent the majority of the meeting going through them, making revisions to suite our project. Our slides were mostly ready to be presented after that.

Meeting October 3rd
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 60 minutes
Summary: This was a short meeting. As a group we went through our slides and everyone picked a couple they wished to talk about during the presentation. We then decided that we should all supplement our slides verbally; we planned to do that individually over the weekend. We also decided we would to show up early to the presentation to practice.

Meeting October 5th
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 120 minutes
Summary: This meeting had a split purpose. Firstly, we spent a little time revising our presentation. We felt some of our slides missed the mark, so we made a few changes. After that, we began talking about Milestone 4. We all brought a couple of different use cases to the meeting and we typed them up into a single document. We also planned out a division of labor to make sure the rest of the Milestone 4 tasks will get completed.

Meeting October 10th
Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 120 minutes
Summary: This meeting was spent talking about where we're at in regard to Milestone 4. We also spent some time talking about looking forward with the project, what we need to work on after Milestone 4. We talked about the programming components we still have left, as well as who wants to work on them.

Meeting October 12th

Attendance: This meeting was attended by Aparna Angu (ara561), Josh Kocur (jak472), Emmanuel Oriade (eoo983), Hao Li (hal356), Ridawn Raji (ror716), Yinsheng Dong (yid164) and Duke Rong (yur013)
Duration: 120 minutes
Summary: This meeting was spent putting the finishing touches on Milestone 4, as well as putting all of the independent components together. This was done as a group. We also spent a little bit of time looking at the next Milestone.