Weirdoughs

Pizza Ordering System
Design Report
for Web & Mobile Application

Version <1.1>

Revision History

Date	Version	Description	Author
March 21st, 2018	1.0	Create software requirement specification	Gerry Xu Melvin Tham Ayushya Amitabh Tobias He
April 27th, 2018	1.1	Create design report	Melvin Tham Tobias He Gerry Xu Ayushya Amitabh

Pizza Ordering System	Version:	<1.1>
Design Report	Date:	April 27th, 2018
Weirdoughs Phase 2 Report.docx		

Table of Contents

Design Report

1. Introduction

This design report will give an overview of the design and functionality of the entire Pizza Order System by our company Weirdoughs. The purpose along with key definitions will better clarify and help understand our system.

1.1 Purpose

This document will detail the functionalities, which will be carried by our company Weirdoughs. This section will begin to introduce how the system work and function.

1.2 Collaboration Class Diagram

Collaboration diagrams are models that describe how a group of objects collaborate in some behavior. The diagrams show a number of objects and the messages that are passed between these objects. The collaboration class diagram below provides an overview of our Pizza Order System. This overview outlines the interactions of the various users with the system and the overall functionality of the system under typical use.
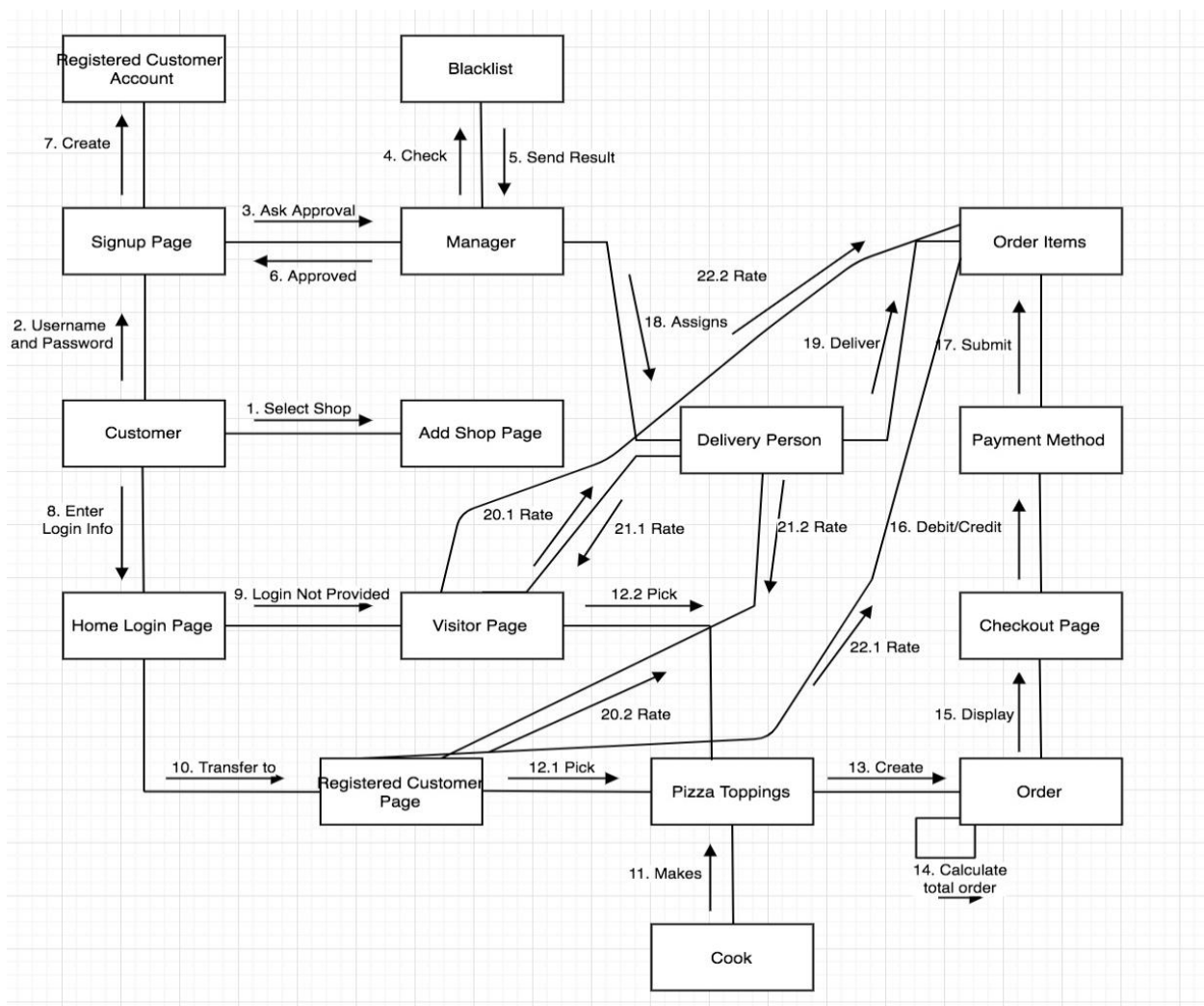
2. Use Case Analysis

In this section, we provide a more detailed overview of each of the main use cases mentioned in our specification report. For each of the main use case listed below, we provide a collaboration class diagram detailing the interaction between classes/objects in the system and a petri-net detailing the various processes involved with each use case.

Main Use-Case Overview:

1. Pick Pizza Store
2. Register Account
3. Login
4. Pick Pizza Choices
5. Level of Prices
6. Submit Order
7. Rating System

2.1 Pick Pizza Store

Normal Scenario

The customer must first select their current location in midtown Manhattan on the map that is provided. The system then generates the three nearest pizza stores. The shop ID along with the name of the shop will be saved to the database once you confirm the selected pizza store.

Exceptional Scenarios:

If the selected shop's name and ID are already registered to our database system then a message will be display saying "the shop is already registered".
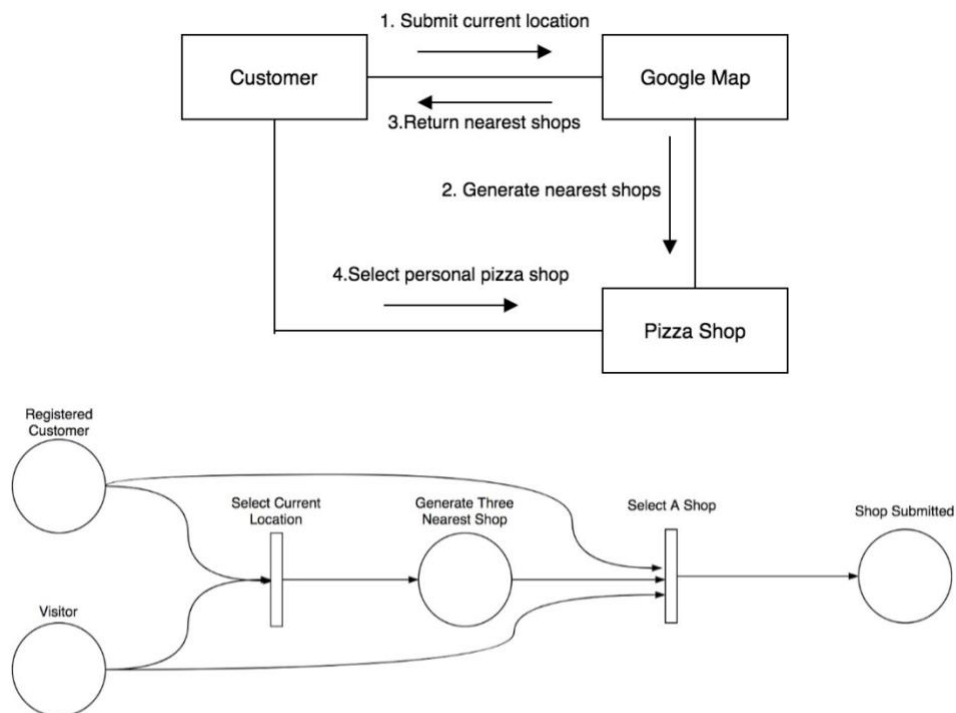
2.2 Register Account

Normal Scenario:

> After picking the pizza store, the customer is giving an option to register for an account. The customer must provide their first and last name, email as well as a password to the signup section. They are also required to upload a profile picture as well an ID for verification.

Exceptional Scenarios:

> The customer's background is check and if it turns out that the customer is listed in their blacklist, their account registration will be denied. A message will appear and notify them that their registration is cancelled. Also, if no sign-up information is provided, the customer is considered a visitor.
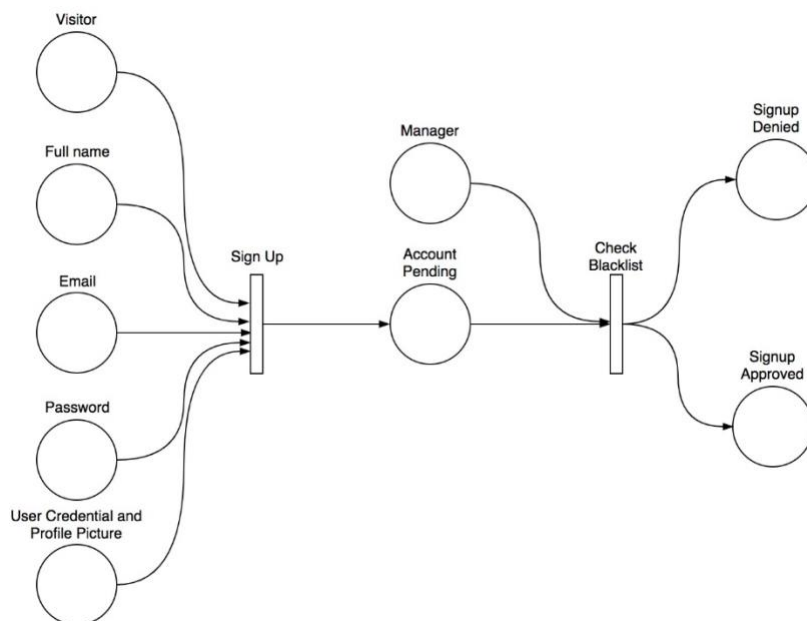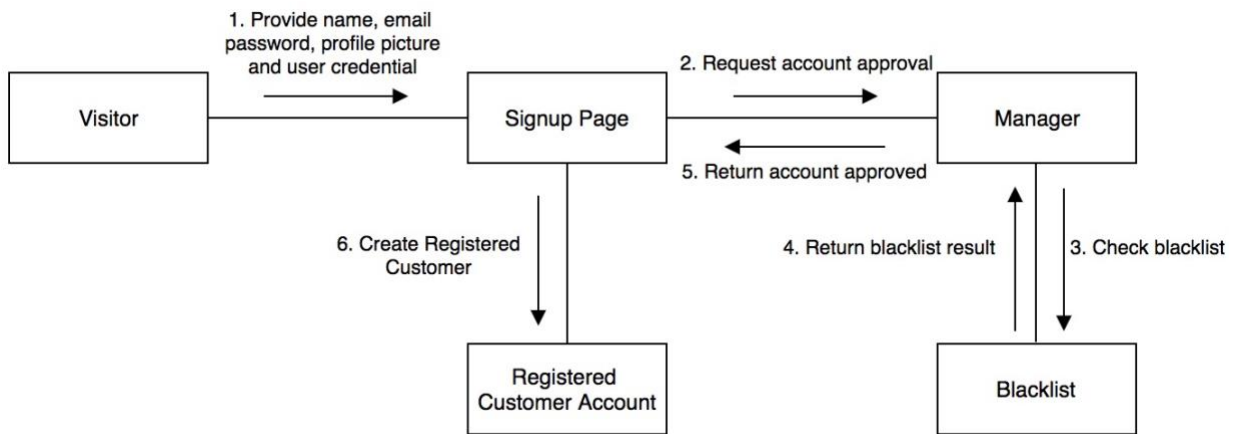
2.3 Login

Normal Scenario:

The customer will be prompted to input their User ID and their password. Once a customer logs in, he/she will have access to view their account. In the account, the customer is able to see his/her rating and also will be able to order a pizza.

Exceptional Scenario:

If the customer has a history of having a lot of bad ratings, the manager has the option of blacklisting that customer. When the customer logs in, the customer will see a warning screen and will be denied going further into the website. Furthermore, if a customer enters the wrong User ID and Password, they will be able to re-enter the correct information.
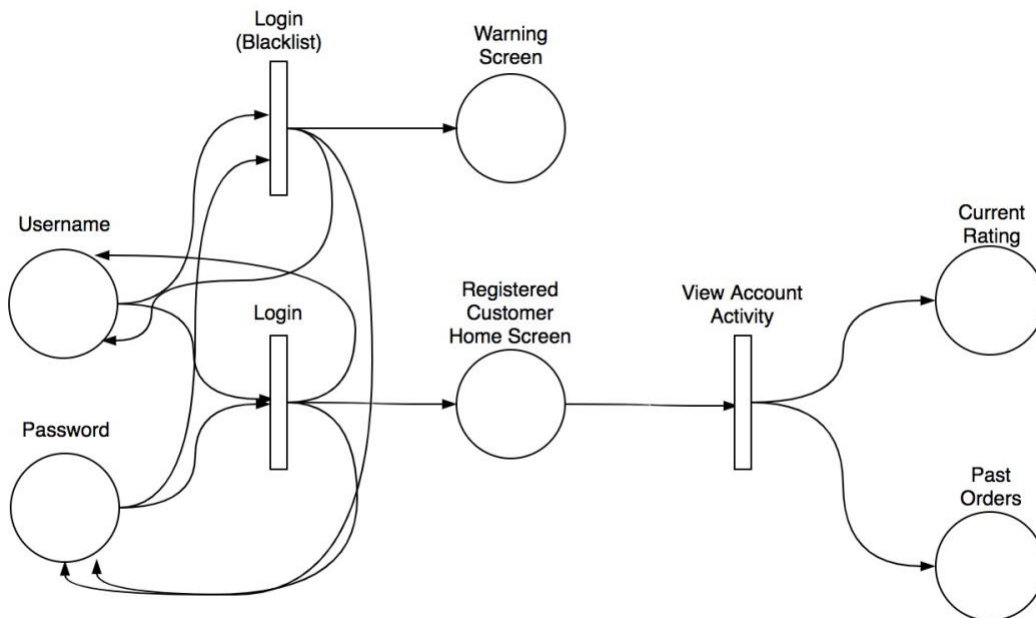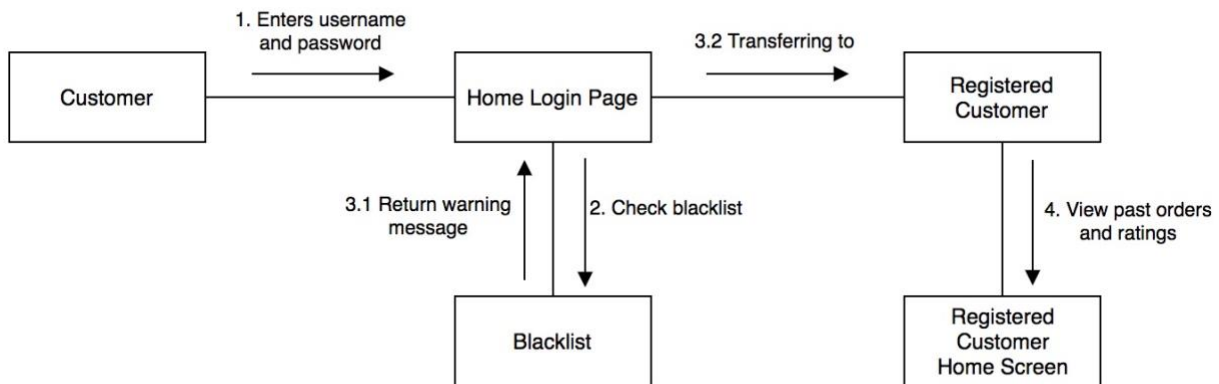
2.4 Pick Pizza Choices

Normal Scenario:

Registered customers along with visitors are greeted with a screen that allows them to select from a variety of pizza toppings along with the quantity of each pizza. They are also able to pick one cook that will make the specific pizza toppings.

Exceptional Scenario:

If the registered customers or visitors are not satisfied with the listed variety of pizza toppings, they are given an option to design their own unique pizza with any topping combinations they want. The price of the unique pizza will be priced higher than the predetermined pizza toppings.
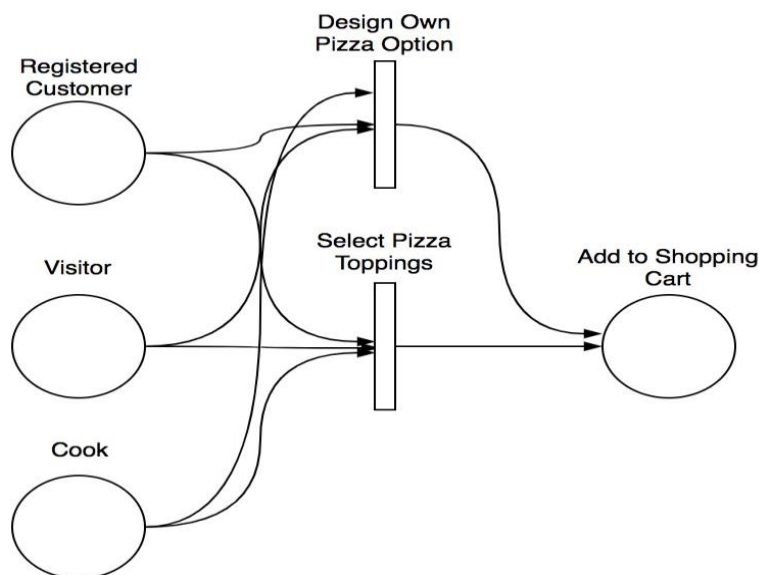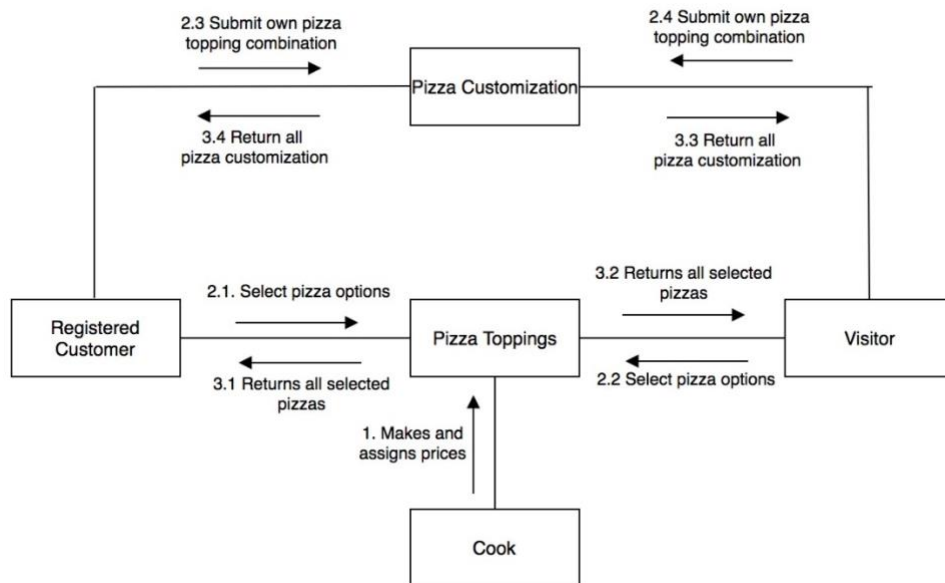
2.5 Level of Prices

Normal Scenario:

After the customer is content with their order, they will be able to checkout their pizza. The prices are based off the choice of the cook.

Exceptional Scenario:

There are different price points based on a customer's merit. For example, if you are a VIP, you will get a VIP discount when purchasing a pizza. If you are a registered customer, you will get a registered customer discount when purchasing a pizza, and if you are a visitor, you will pay the standard price.
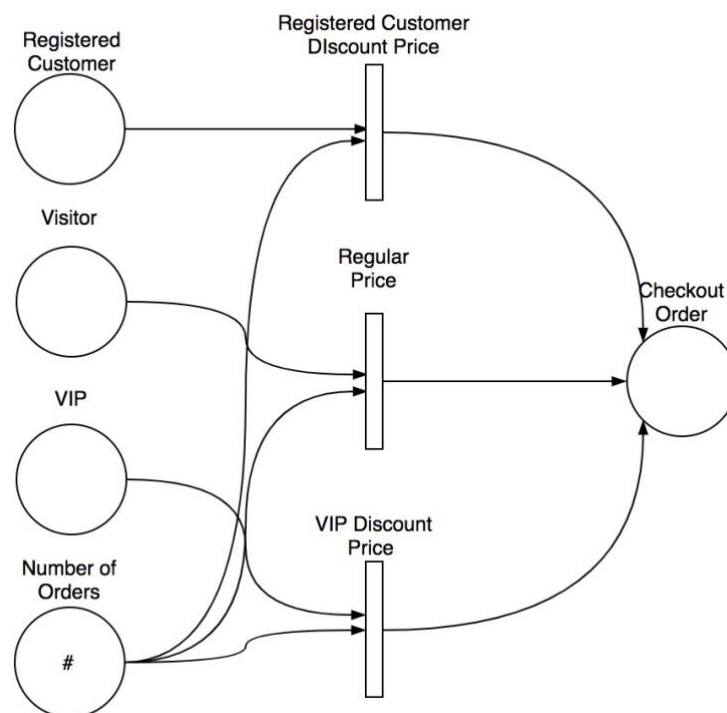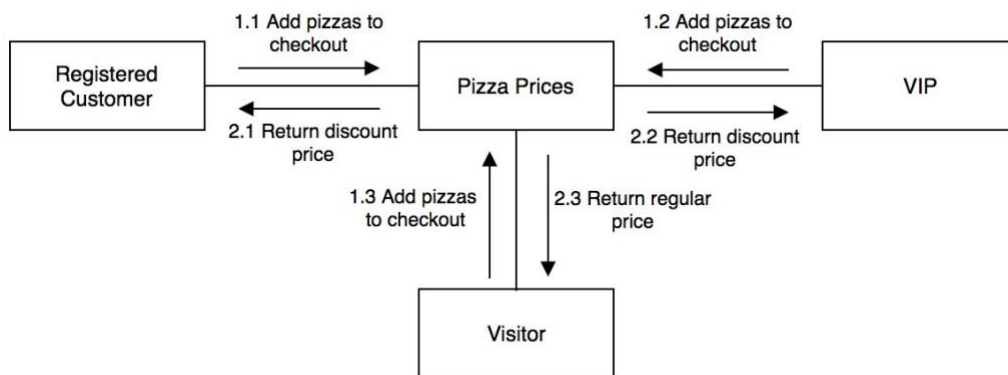
2.6 Submit Order

Normal Scenario

Once the customer is happy with their order, they can proceed to the checkout page where they are greeted with an option to pay with either credit or debit card. After the payment information is entered, they can submit their order. This will notify the manager to choose a deliverer to deliver the pizza. The deliverer will choose the best route based on traffic towards the customer's location.

Exceptional Scenario

If the credit card exceeds the credit limit or the debit card balance is insufficient to pay for the order, the order will not process and the customer will have to re-enter another credit or debit card.
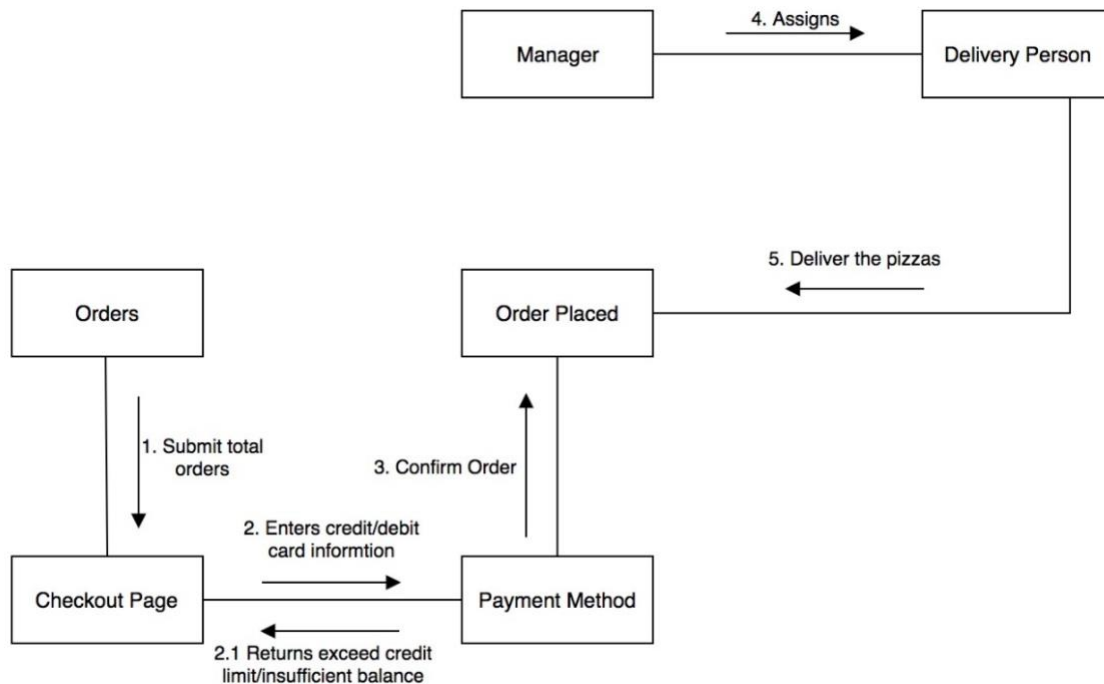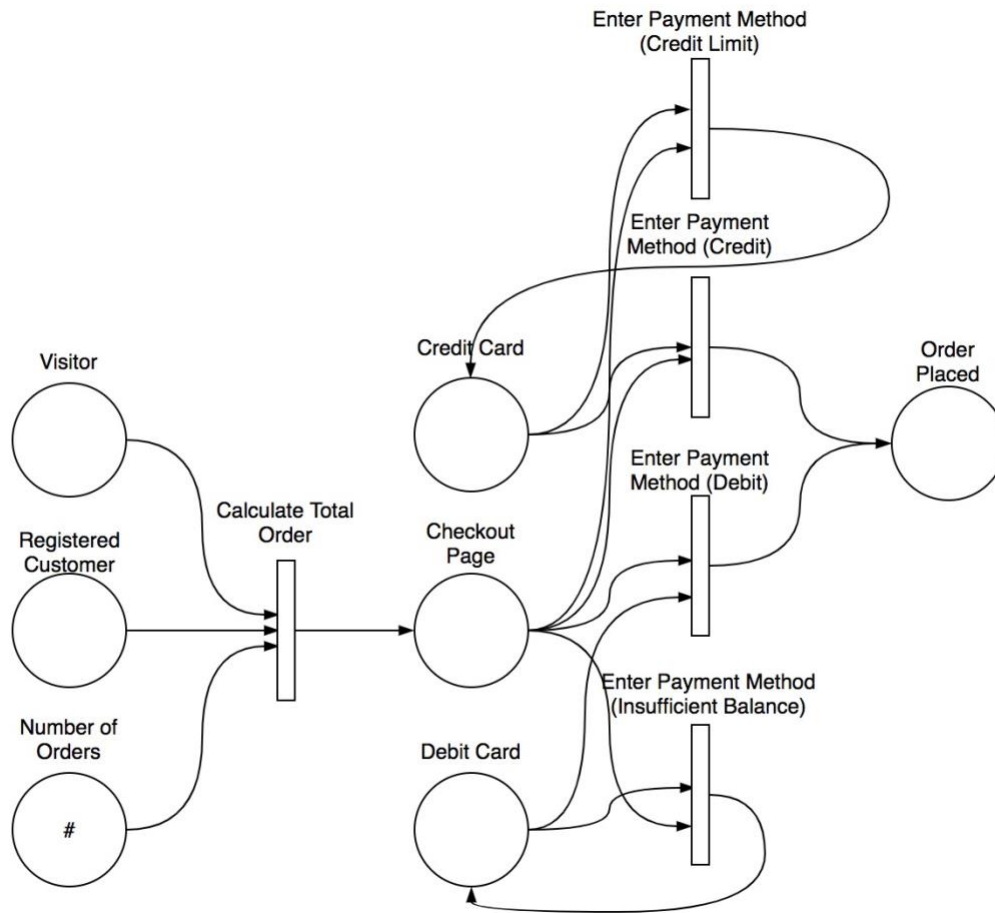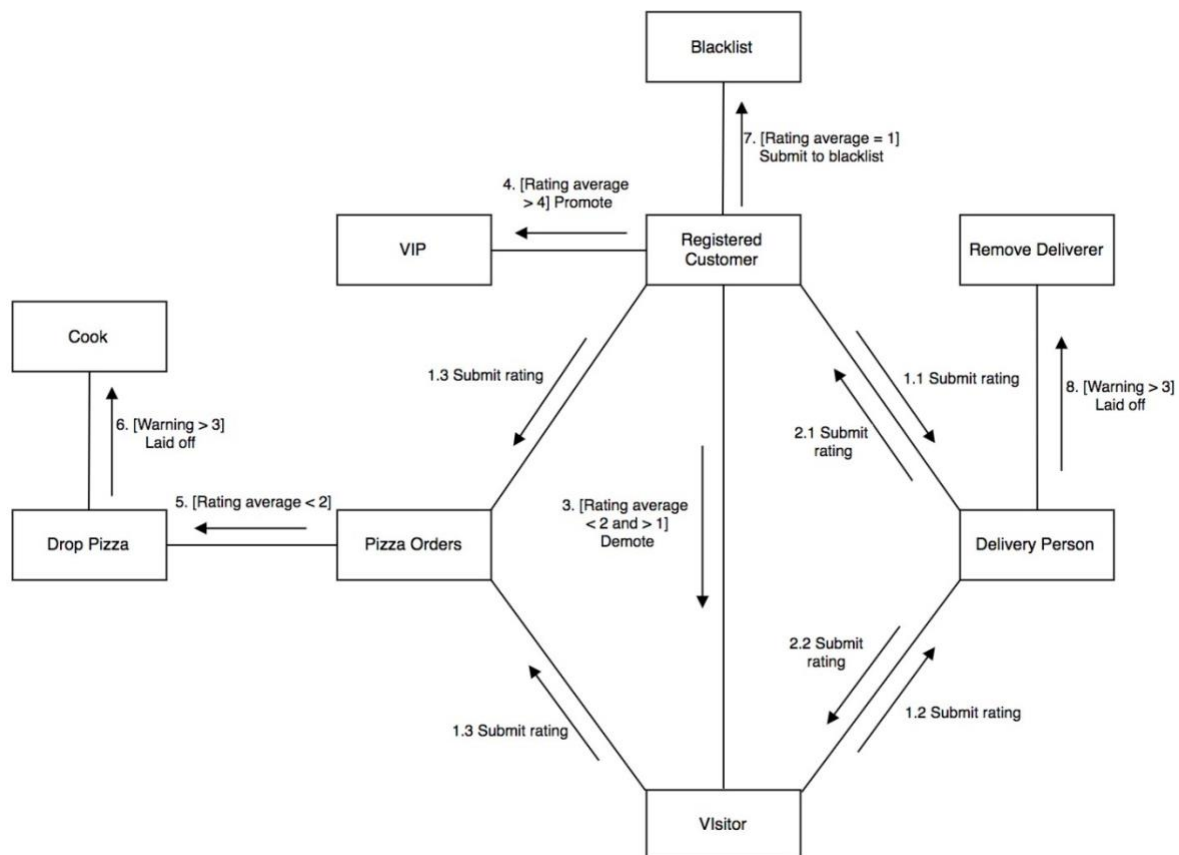
2.7 Rating System

Normal Scenario:

> After the customer has received the pizza from the deliverer, both customer and deliverer will be asked to rate each other based off service and mannerism. Furthermore, the customer will be asked to rate the pizza which will be considered a rating for the cook. Finally, the customer will also have to rate the store. These ratings are based on a 1 (worst - 5 (best) scale.

Exceptional Scenarios:

> Any ratings below an average of 3/5 will be viewed as a complaint and must provide a sentence defending their rating. A registered customer whose rating is below an average of 2/5, but greater than 1/5 will be demoted to a "visitor", while a customer rating with an average of 1/5 will be blacklisted. A customer with an average rating of above 4/5 will be made into a VIP. A deliverer whose average rating is less than 2/5 in the last three orders will be given a warning that can be erased. If given more than three warnings, the deliverer will be laid off. Any pizza with an average rating less than two in the last three orders will be dropped. A cook who made two dropped pizza's will be warned. After more than three warnings, the cook will be laid off.

If Average Rating
< 1

Put to
Blacklist

If Average Rating
< 3 (General)

Mark as
Complaint

Registered
Customer

Rate Registered
Customer/Visitor

If Average Rating < 3
but > 2 (Customer)

Demote to
Visitor

Visitor

Rate Delivery
Person

Record Rating

If Average Rating
> 4 (Customer)

Promote
to VIP

Laid Off

Delivery
Person

Rate
Pizza

If Average Rating
< 2 (Deliverer)

Receive
Warning

Cook

Pizza

If Average Rating
< 2 (Pizza)

Drop
Pizza

3. Entity-Relationship Diagram

An entity relationship diagram is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. Below is an Entity Relationship Diagram describing the overall Pizza Order System and outlining our database.
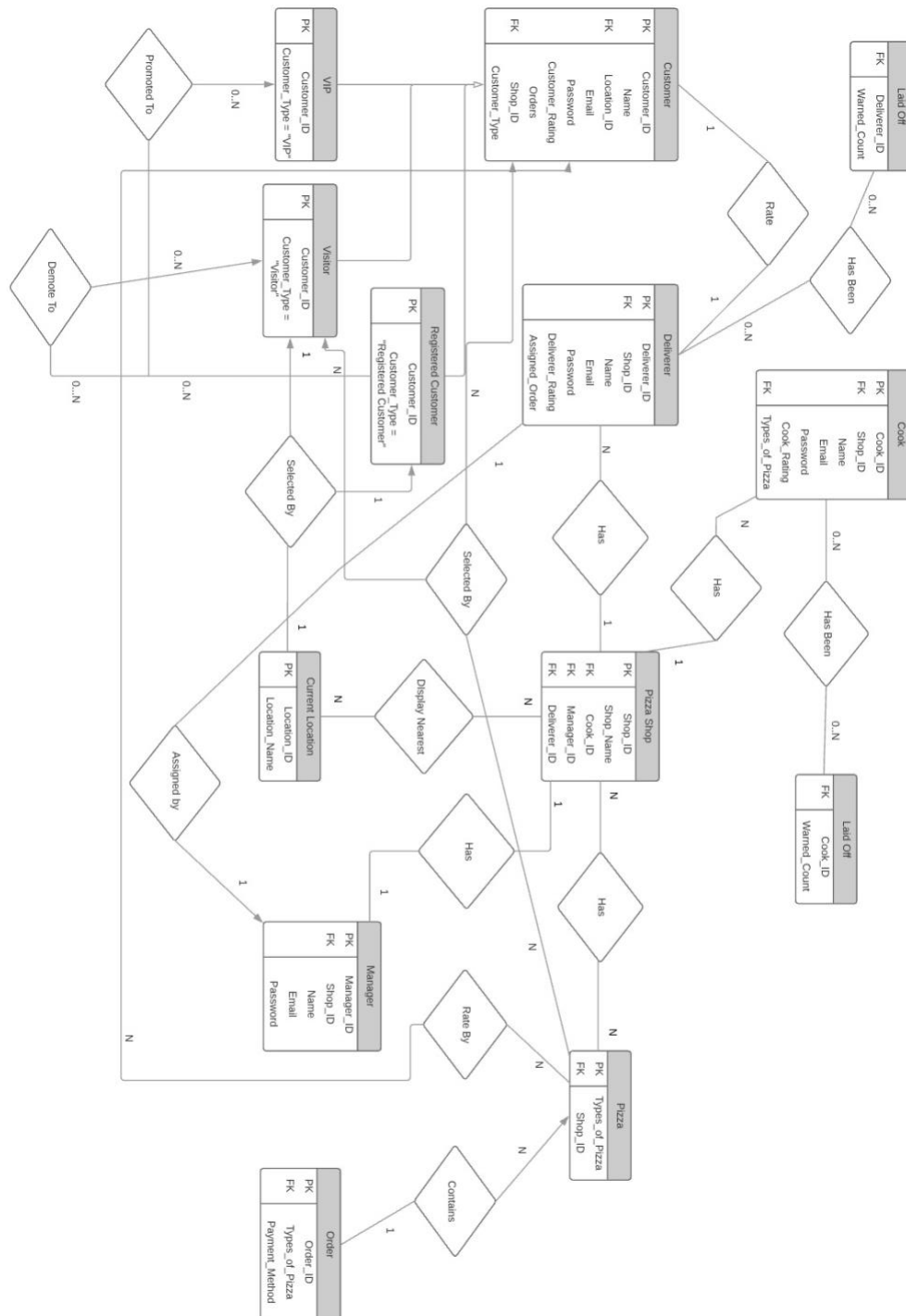
4. Detailed Design

4.1 addShop

HTTPS-POST

Received: Shop details – location, gmap_id, and name

Returned: Success or Failure

----

data = {

       location: request→body→location,

       gmap_id: request→body→gmap_id,

       name: request→body→name

};

Add {data} in /Shops/{request→body→uid}/

4.2 doesShopExist

HTTPS-POST

Received: Shop's gmap_id to check for

Returned: Found or Not Found

Get all Shops data from database:

       For each shop:

              If the gmap_id requested exists in database:

                     Send back *found*

              Else send back *not found*

4.3 getUserOrderData

HTTPS-POST

Received: Array of order IDs

Returned: Array of order details – matches order of IDs

For each requested OrderID:

       oData = Array of order details

       Get order details from database:

              Add details to oData

Send back oData

4.4 getClosestShops

HTTPS-POST

Received: Array of shops from Google Maps Places API

Returned: Array of shops that also exist in database

Get all shops from database:

       For each result in GoogleMapsAPI

              If GoogleMapsAPI.gmap_id exists in database:

                     Add shop detail to return array

Send back array of order details

5. System Screens

Here, we demonstrate major GUI screens of the system and showcase our Add Shop Functionality in our Pizza Order System.

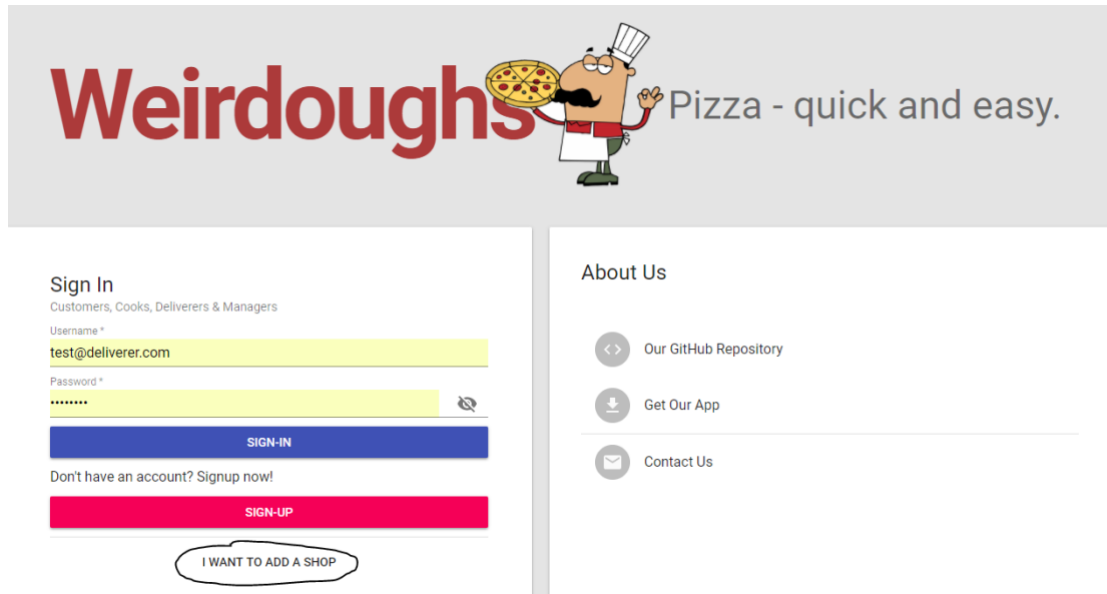## 5.1 Showing the Add Shop Function

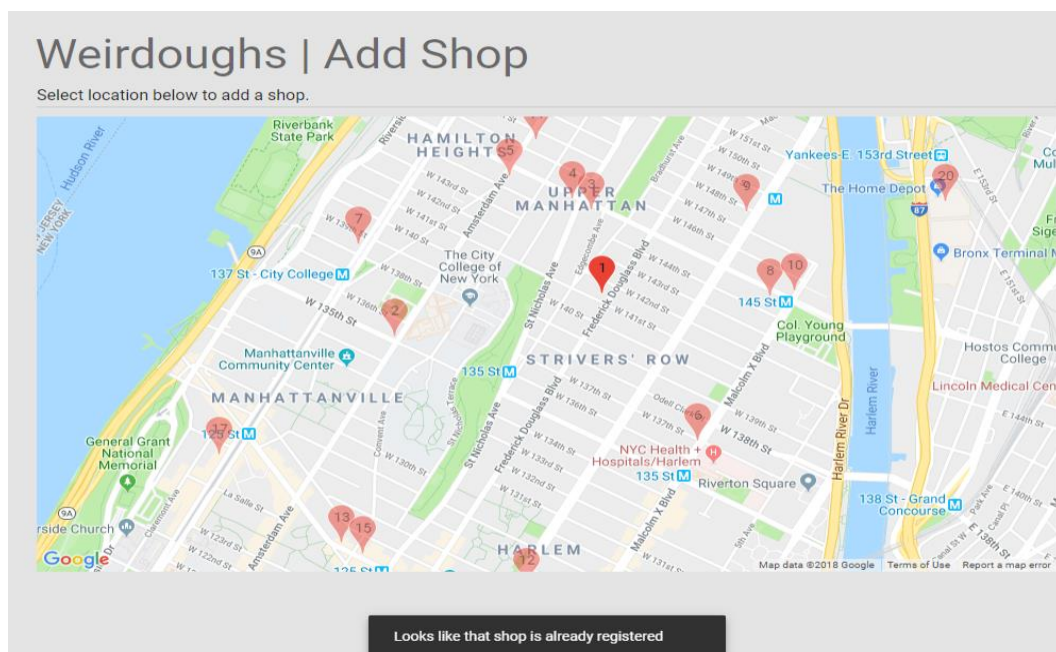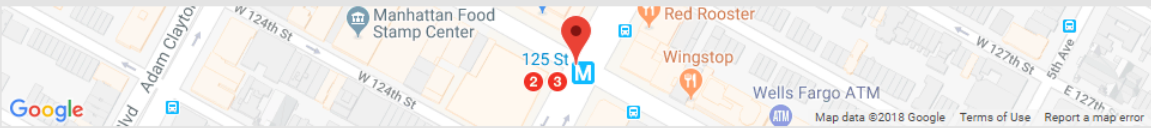*Figure 1: User can add a shop*

Figure 2: User will be able to pick a pizza store from multiple locations in Manhattan. If the shop is already registered, it will give a prompt below

Figure 3: If the pizza store is not registered, the user is able to add that shop to the list.

6. Minutes of Group Meeting

| Meeting #	Discussion
1	Report and Diagram
2	Layout our basic functionalities of each users along with how our database will look like
3	Begin implementing our systems and the user functionalities

7. Miscellaneous

Fixed several cases within our diagram where the <include> and <extends> were labeled incorrectly. The original phase 1, software requirements specification report is also attached.
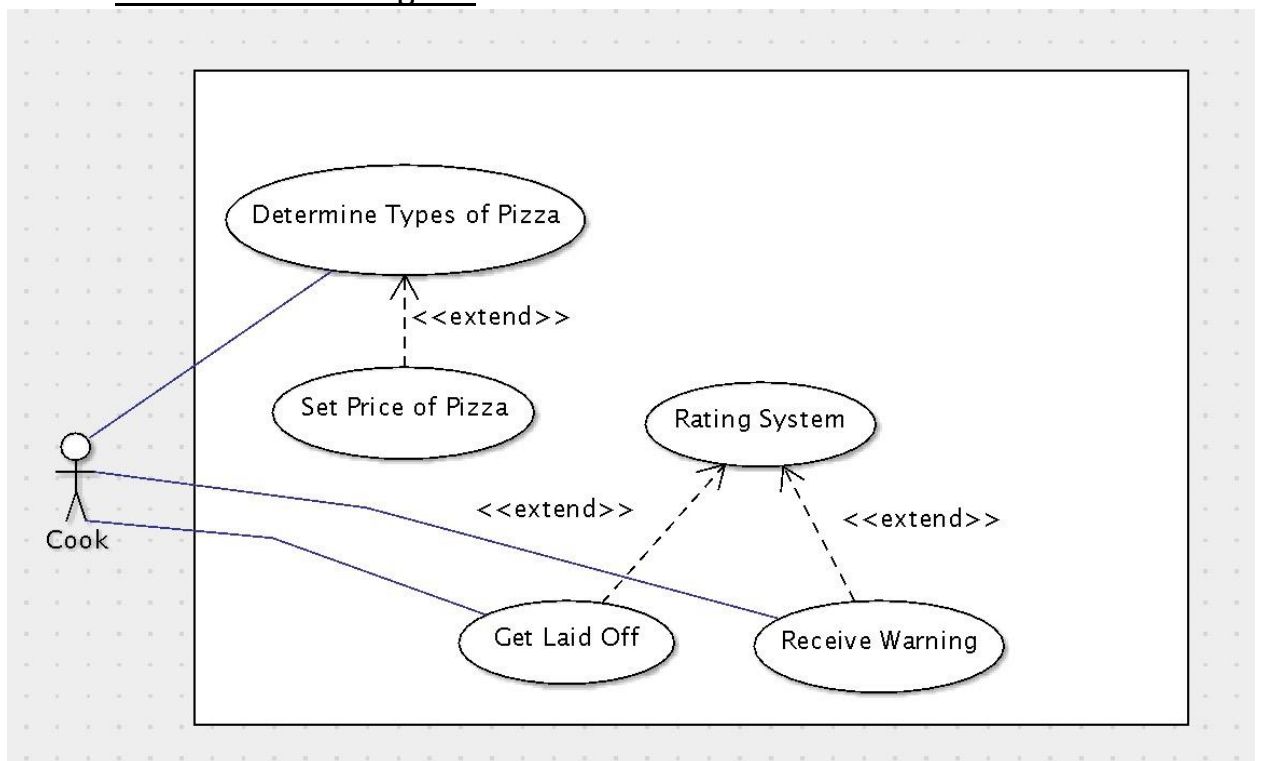
7.1 Overall Use-Case Diagram
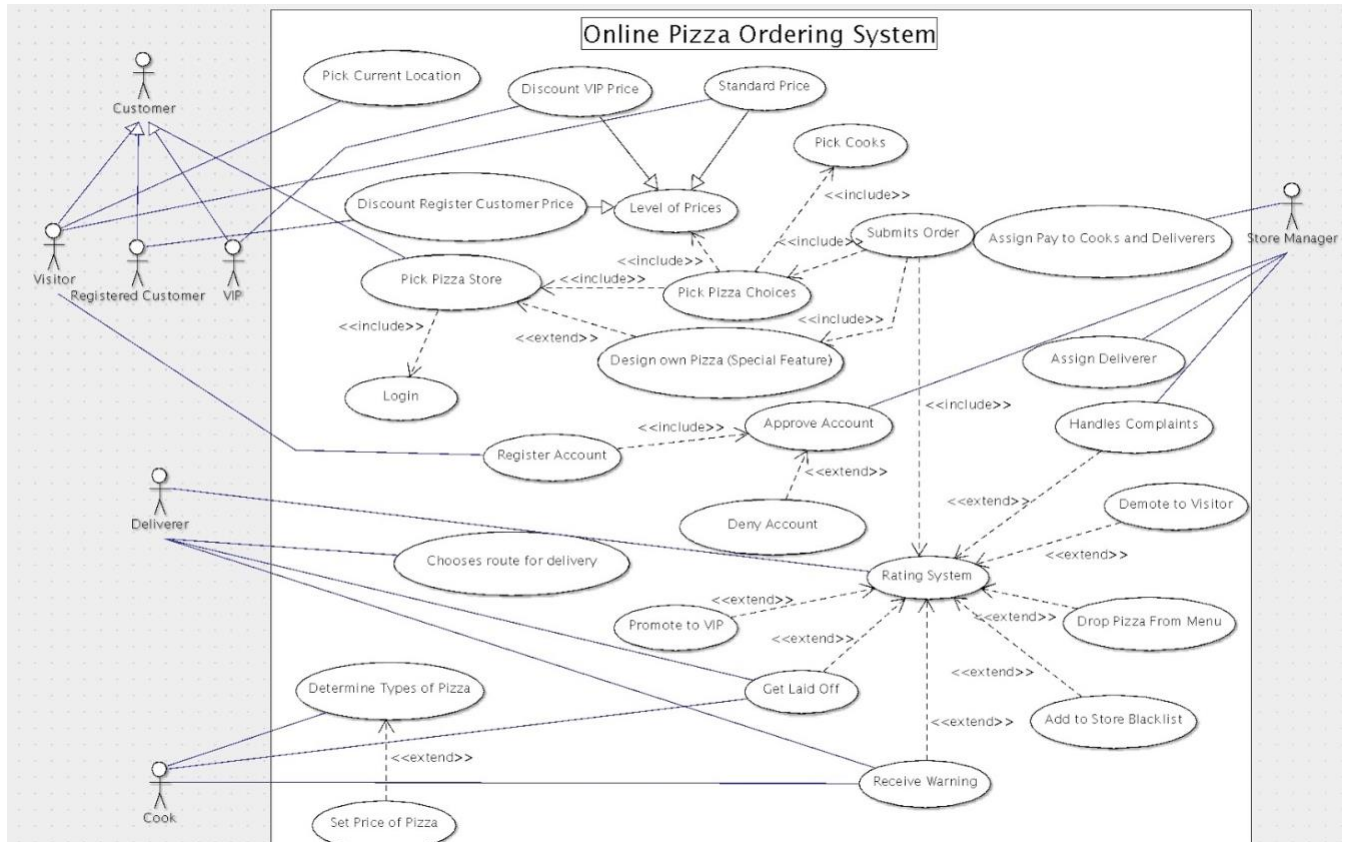
7.2 Customer Use-Case Diagram
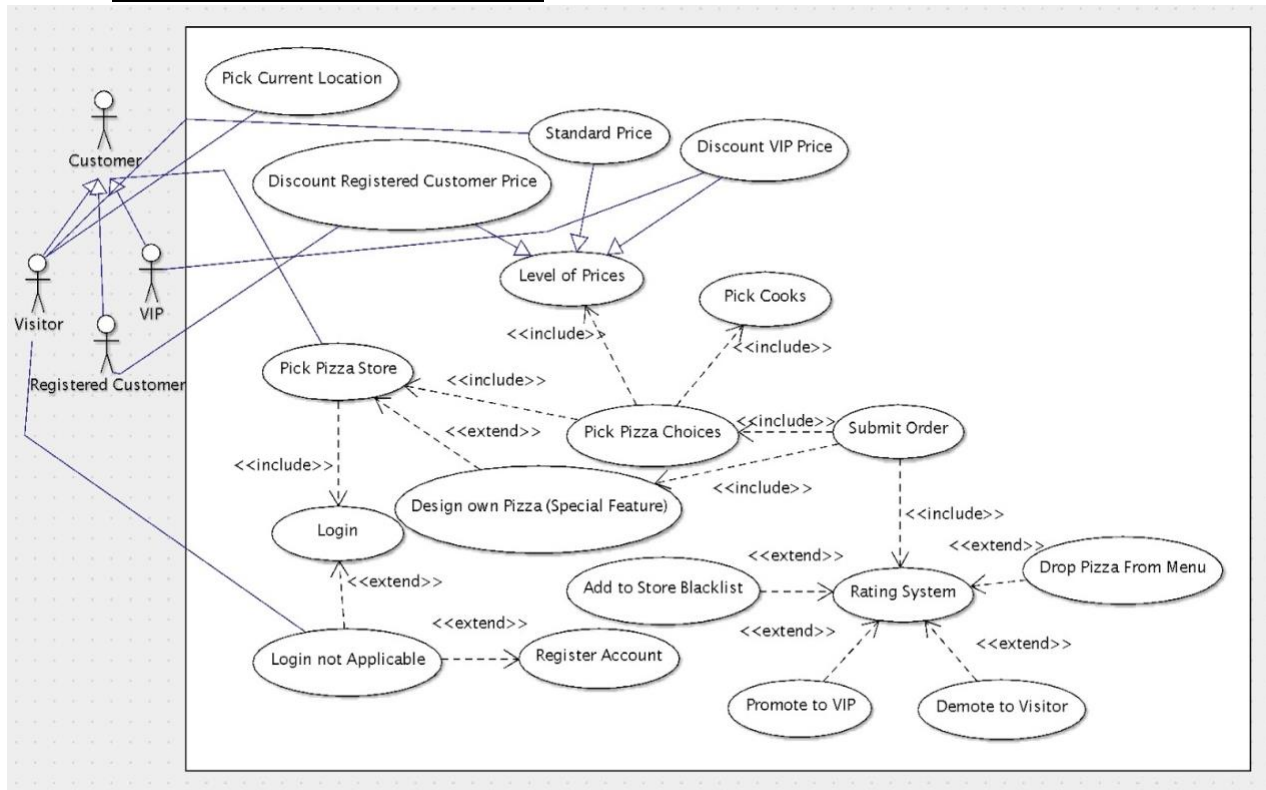
7.3 Cook Use-Case Diagram

8. Supporting Information

The Pizza Ordering System described above is being developed by the Weirdoughs ™ team – Gerry Xu, Melvin Tham, Tobias He and Ayushya Amitabh.

Contact our team at tagmhaxt@gmail.com.

Our complete code can be found at: https://github.com/ayushyamitabh/PizzaOrderSystem
Our work distribution can be found at: https://goo.gl/4U1JqU
Our database structure is shown at: https://goo.gl/ky93yD

Our project is being hosted at: https://pos-tagmhaxt.firebaseapp.com