

---

**FETA**

---

**Greek Cafe Website  
Design Report Specification  
For Web Application**

**Version 1.1**

FETA	Version: 1.1
Design Report Specification	Date: 04/19/2024
FETA Phase II Report	

## Revision History

Date	Version	Description	Author
03/26/2024	1.0	Initial draft of the Phase I report	Arbab Husain Daniel Rappaport
04/19/2024	1.1	Created design reports	Arbab Husain Daniel Rappaport Joshua Henry Kareem Virani

FETA	Version: 1.1
Design Report Specification	Date: 04/19/2024
FETA Phase II Report	

## Table of Contents

1. Introduction	4
2. Use Case Analysis	5
3. Entity-Relation Diagram Analysis	7
4. Detailed Design	10
5. System Screens	13
6. Group Memo	16
7. Git Repo	16

FETA	Version: 1.1
Design Report Specification	Date: 04/19/2024
FETA Phase II Report	

# Design Report Specification

## 1. Introduction

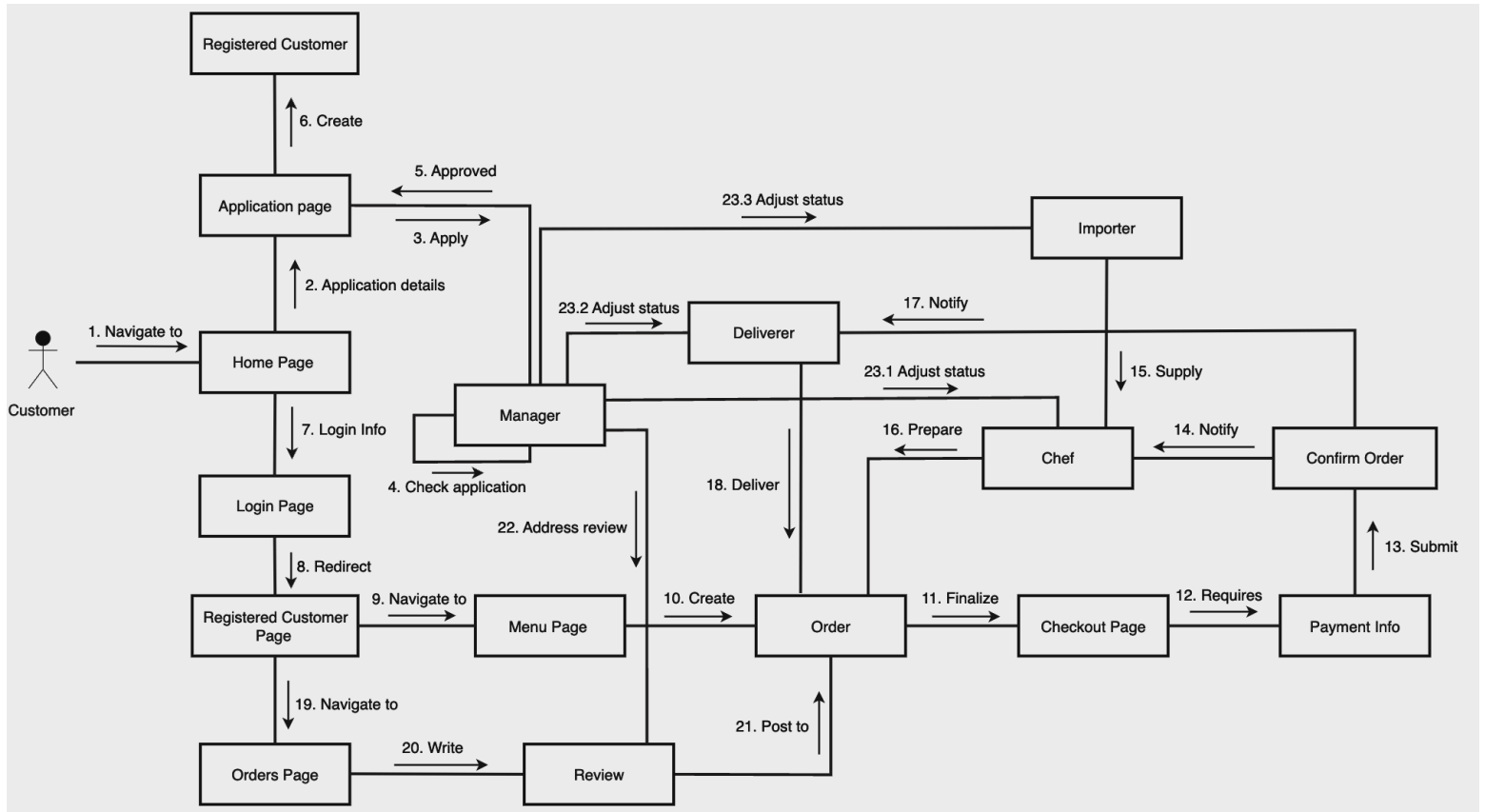
This document will give an overview of the entire Greek Cafe Website, Feta. It will include collaboration, petri net, use case, and E-R diagrams. It will include descriptions for all diagrams, features, and functions discussed.

### 1.1 Purpose

The purpose of this is to document the progress made as well as outline the overall structure and specific functions of the system. This will provide a detailed description of the system and its functionalities.

### 1.2 Collaboration Class Diagram

A collaboration class diagram shows how the different objects, users, and classes interact within the system to achieve the purpose of the system. The diagram below provides an overall picture of our system and includes all of the interactions between the different users and objects.



FETA	Version: 1.1
Design Report Specification	Date: 04/19/2024
FETA Phase II Report	

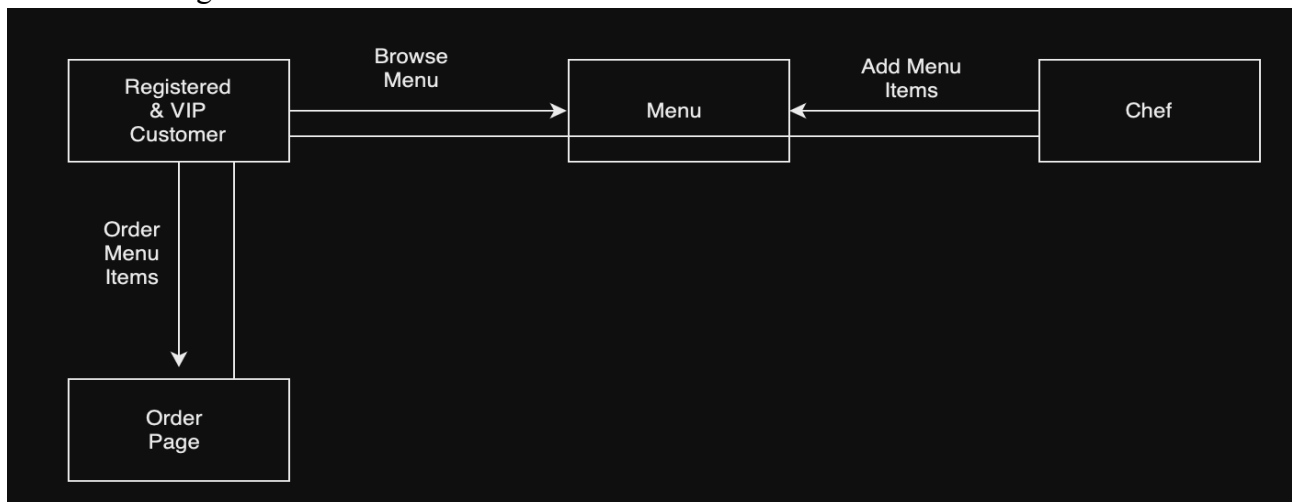
## 2. Use Case Analysis

In this section, a more detailed overview of each of the main use cases mentioned in our specification report. For each of the main use cases, there is 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.

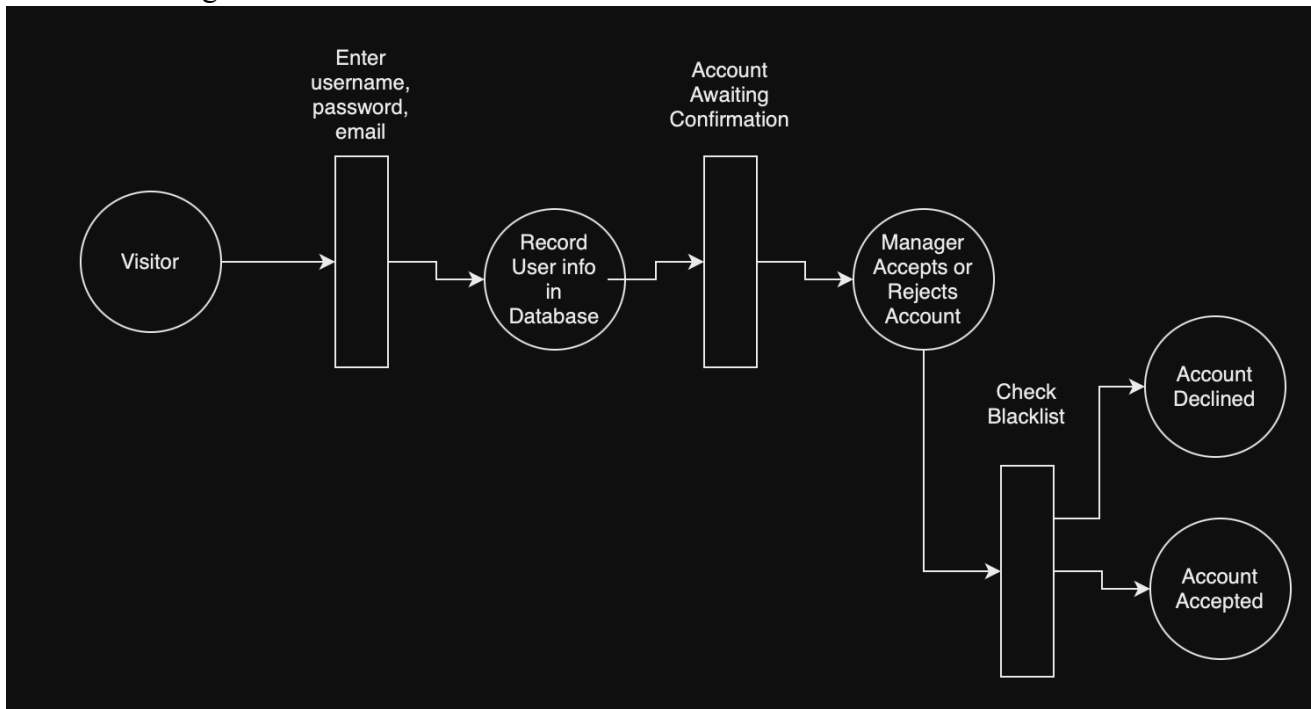
Use Cases:

- 2.1 Menu Navigation
- 2.2 Account Registration
- 2.3 Login
- 2.4 Order Placement
- 2.5 Menu Item Rating System
- 2.6 Customer Rating System
- 2.7 Price Options

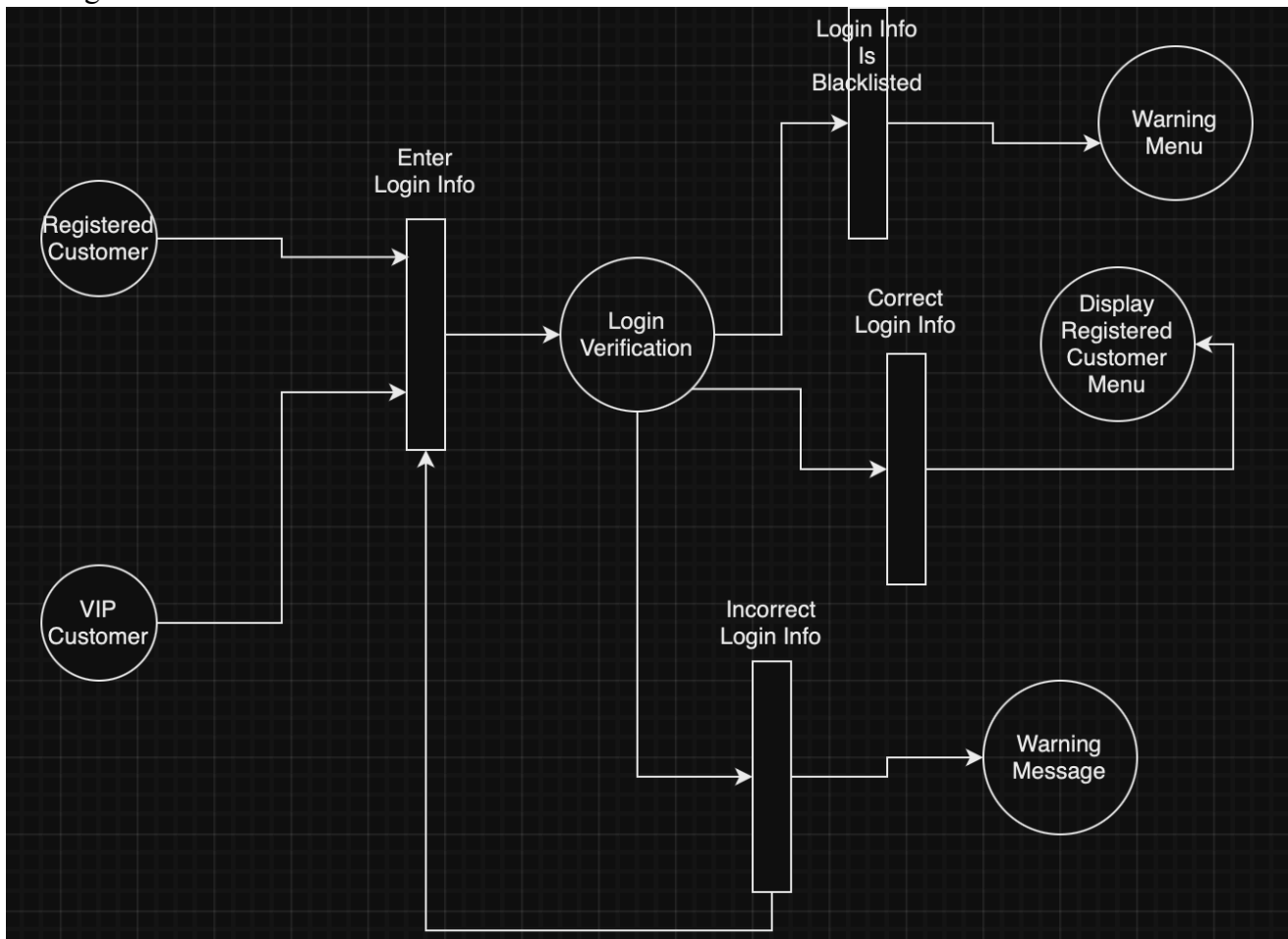
### 2.1 Menu Navigation:



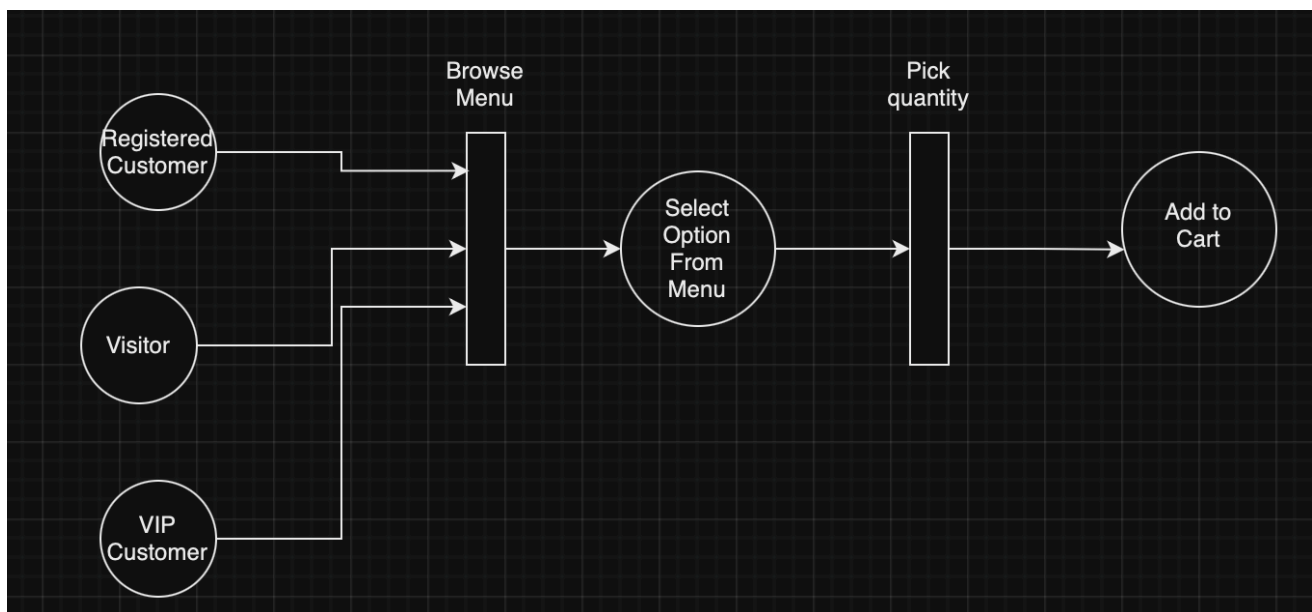
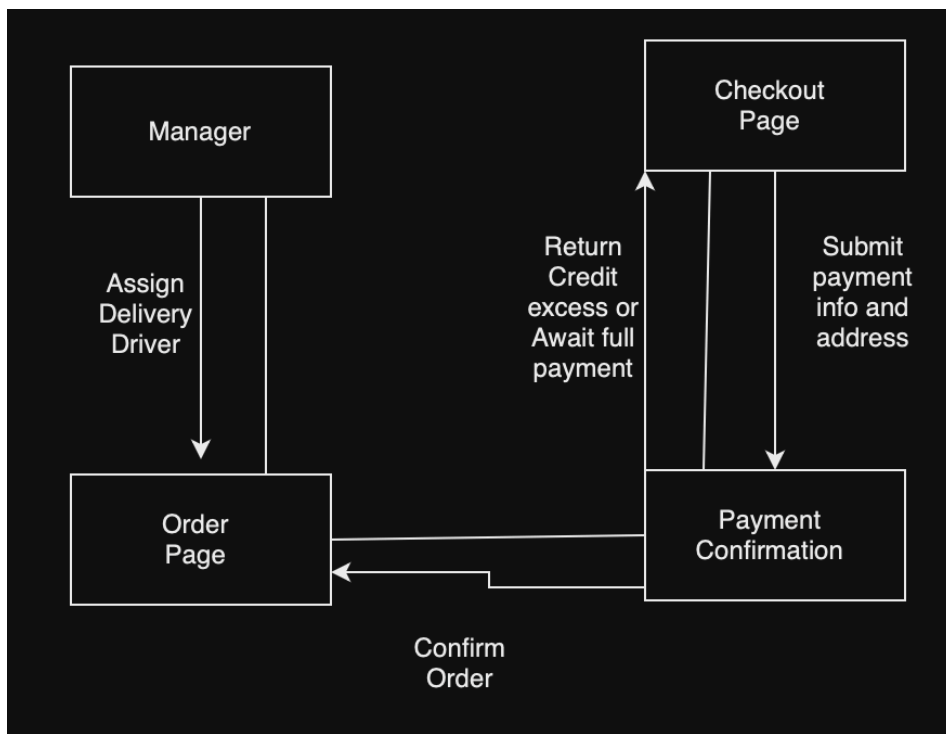
## 2.2 Account Registration:



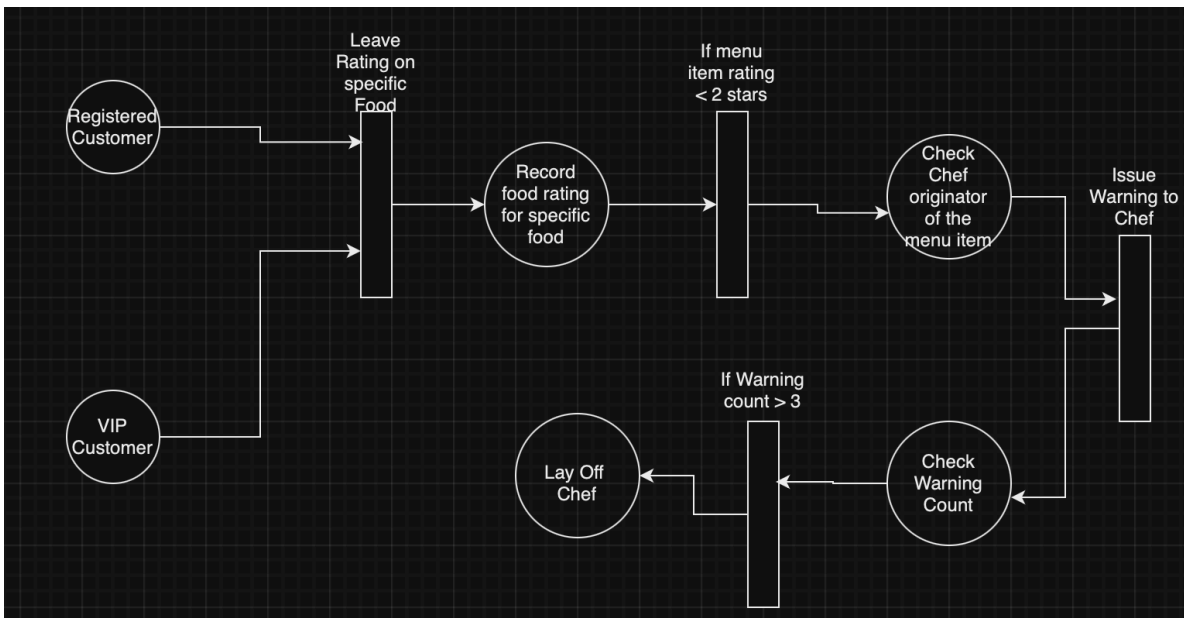
## 2.3 Login:



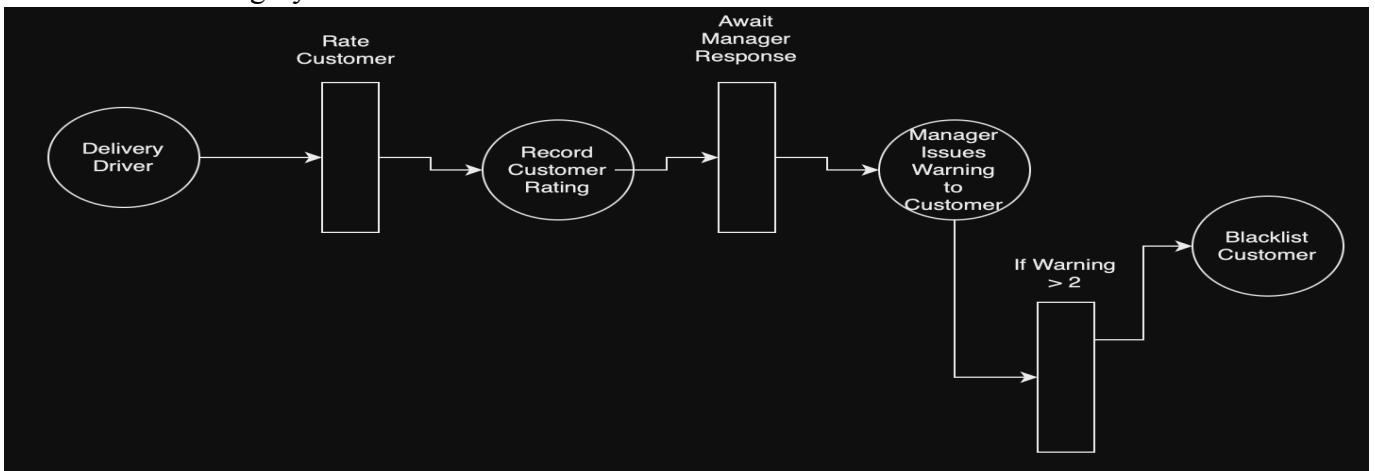
## 2.4 Order Placement:



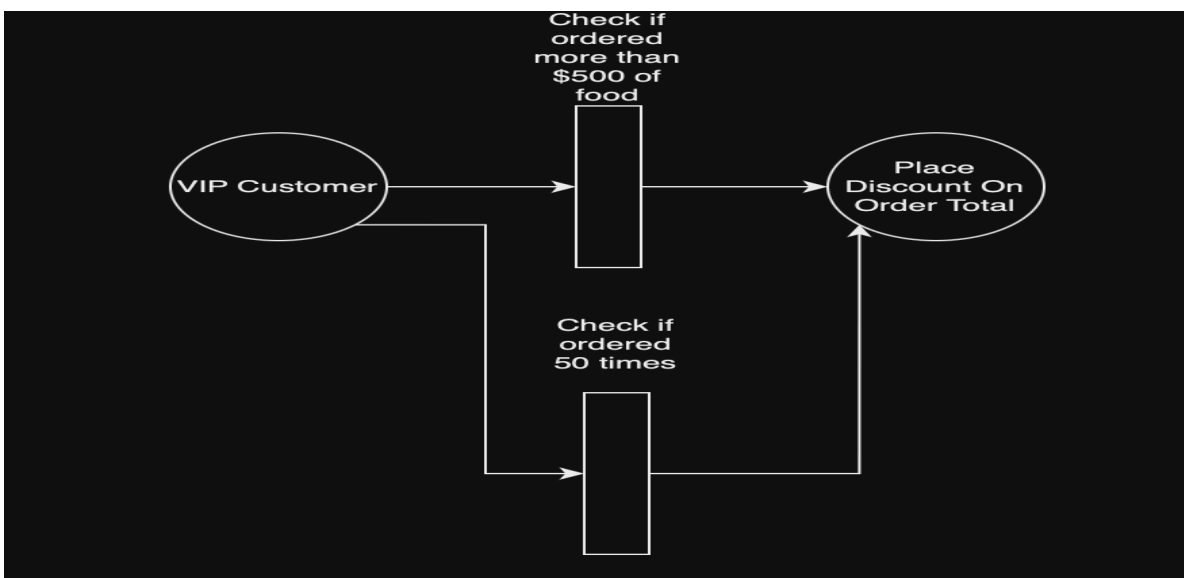
2.5 Menu Item Rating System:



## 2.6 Customer Rating System:



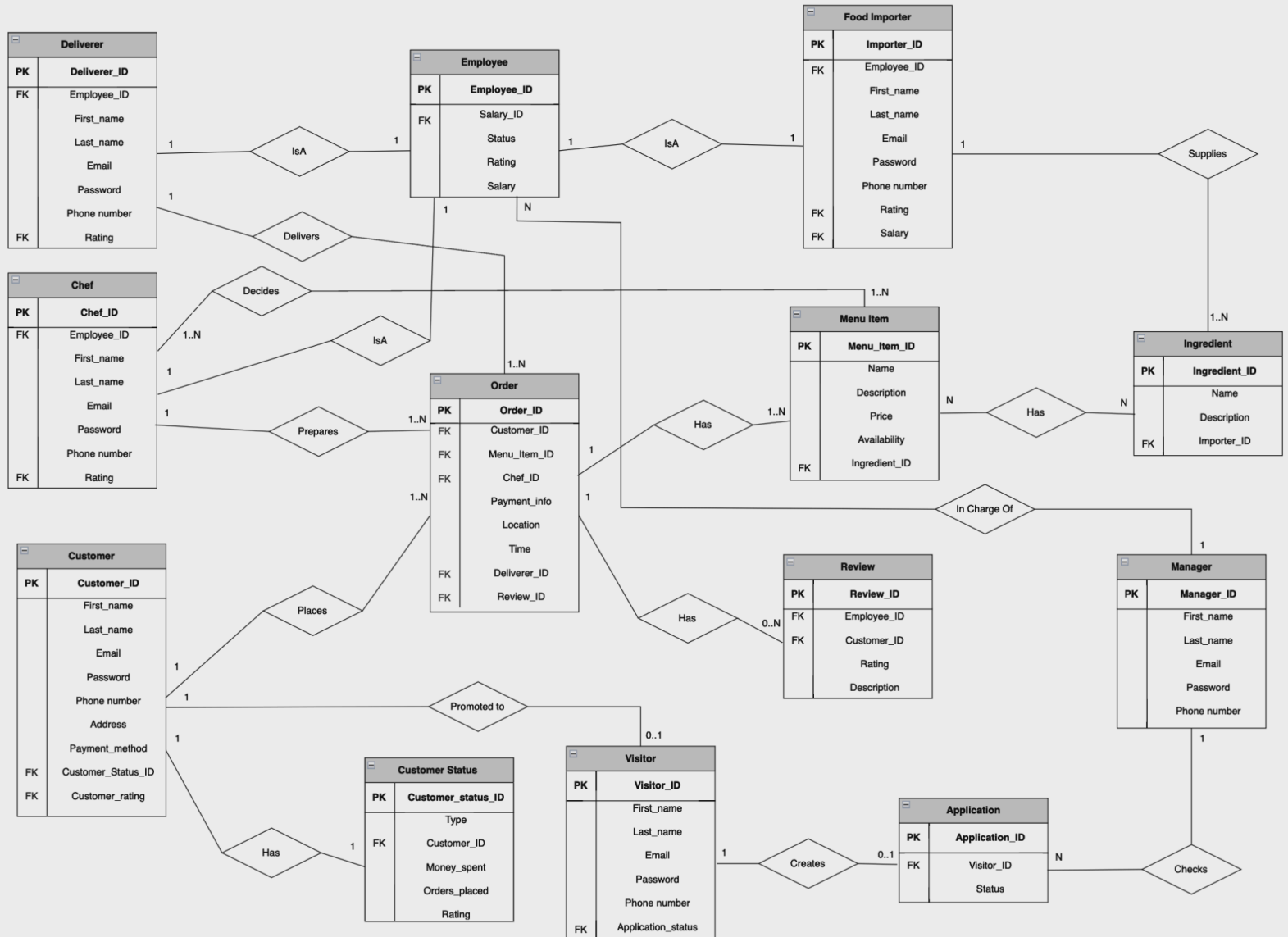
## 2.7 Price Options:





### 3. E-R Diagram

An E-R Diagram shows the structure of the database for a system. The E-R Diagram below shows all the entities and their attributes for our restaurant website system. It also shows the cardinalities and natures of the relationships that exist between them.



## 4. Detailed Design

This section consists of a detailed technical (backend) design of the main functionalities of our application. It outlines the methods available to each user group and specifies its API route, input, output, and pseudocode.

### 4.1. Public (Surfer) Methods (No auth required)

#### 4.1.1. Get Menu

HTTP GET /api/menu

Input: None

Output: List of all dishes with details (name, price, description, average rating)

Pseudocode:

```
function getMenu() {  
    menu = []  
    items = database.query("menu_items")  
    for each item in items:  
        menu.add({item.name, item.description, item.price, item.rating})  
    return menu  
}
```

#### 4.1.2. Get Item Reviews

HTTP GET /api/menu/{itemId}/reviews

Input: itemId (integer/uuid)

Output: List of all reviews for the given dish/item

Pseudocode:

```
function getItemReviews(itemId) {  
    return database.query("menu_items", itemId).reviews  
}
```

### 4.2. Member Methods

#### 4.2.1. Register Member

HTTP POST /api/members/register

Input: name (string), email (string), phone (long), password (string)

Output: Status representing whether or not registration was successful

Pseudocode:

```
function registerMember(name, email, phone, password) {  
    if database.exists(email, "members")  
        return Error("Email already registered")  
    database.insert("members", new Member(name, email, phone,  
encrypt(password))  
    return {"success": true}  
}
```

#### 4.2.2. Member Login

HTTP POST /api/members/login

Input: email (string), password (string)

Output: Authentication token and member id

Pseudocode:

```
function memberLogin(email, password) {  
    member = database.query("members", email, encrypt(password))  
    if member is null  
        return ("Invalid email or password")  
    return {"token": createToken(member), "id": member.id}  
}
```

#### 4.2.3. Place Order

HTTP POST /api/members/order

Input: Auth token, list (the user's "cart") of menu item ids and quantities  
+ extra options/specs

Output: Order confirmation, order number

Pseudocode:

```
function placeOrder(token, cart) {  
    member = getMember(token)  
    if token is null or member is null:  
        return Error("Unauthorized")  
    order = new Order(cart)  
    database.insert("orders", order)  
    return {"orderId": order.id, "message": "Thank you! Your order is  
confirmed."}  
}
```

### 4.3. Chef Methods

#### 4.3.1. Update Menu Item

HTTP PUT /api/menu/{itemId}

Input: Auth token, itemId and a new menu item object with the updated  
fields (price not accepted)

Output: None

Pseudocode:

```
function updateMenuItem(token, itemId, newDetails) {  
    if token is null or getChef(token) is null:  
        return Error("Unauthorized")  
    if newDetails.hasKey("price")  
        return Error("Price update forbidden")  
    item = database.query("menu_items", itemId)  
    for each element in newDetails:  
        item.update(element.key, element.value)  
}
```

### 4.4. Manager Methods

#### 4.4.1. Update Menu Item (Admin)

HTTP PUT /api/menu/{itemId}

Input: Auth token, itemId and a new menu item object with the updated fields

Output: None

Pseudocode:

```
function updateMenuItemAdmin(token, itemId, newDetails) {  
    if token is null or getManager(token) is null:  
        return Error("Unauthorized")  
    item = database.query("menu_items", itemId)  
    for each element in newDetails:  
        item.update(element.key, element.value)  
}
```

#### 4.4.2. Add Menu Item

HTTP POST /api/menu

Input: Auth token and a new menu item object

Output: Status representing success or error

Pseudocode:

```
function addMenuItem(token, item) {  
    if token is null or getManager(token) is null:  
        return Error("Unauthorized")  
    database.insert(item, "menu_items")  
    return {"success": true}  
}
```

### 4.5. Delivery Driver Methods

#### 4.5.1. Get Deliveries

HTTP GET /api/drivers/deliveries

Input: Authentication token

Output: List of assigned deliveries

Pseudocode:

```
function getDeliveries(token) {  
    driver = getDeliveryDriver(token)  
    if token is null or driver is null:  
        return Error("Unauthorized")  
    return database.query("deliveries", driver.id)  
}
```

## 5. System Screens

Here we show some of the GUI features of our project such as the homepage, menu, and customer login/application.

### 5.1 Homepage



*Figure 1: Display shows the initial landing page*


## 5.2 Menu

Home	Menu	Login/apply
<div><h1>MENU:</h1><hr/><h3>Starters</h3><hr/><div><div><b>Homemade Rice Dolmades</b> homemade rice dolmades vine leaves stuffed with rice and herbs in lemon sauce</div><div>\$12.95</div></div><div><div><b>Zucchini Fritz</b> zucchini and feta pancakes</div><div>\$13.95</div></div></div> <hr/> <h3>Soups</h3> <hr/> <div><div><b>Chicken Avgolemono</b> lemon chicken soup with rice</div><div>\$7.00</div></div> <div><div><b>Soup of the day</b> soup of the day</div><div>\$7.00</div></div>		
<hr/> <h3>Salads</h3> <hr/> <div><div><b>Greek Salad</b> lettuce, tomato, onion, cucumber, peppers, olives, feta cheese &amp; olive oil</div><div>\$12.95</div></div> <div><div><b>Maroulosalata</b> romaine lettuce, scallions, dill, olive oil &amp; balsamic</div><div>\$13.95</div></div>		
<hr/> <h3>Sandwiches</h3> <hr/> <div><div><b>Pork Gyro</b> sliced marinated pork cooked slowly on rotisserie</div><div>\$12.95</div></div> <div><div><b>Sliced steak in a pit</b> sliced marinated steak cooked to perfection</div><div>\$16.95</div></div>		
<hr/> <h3>Desserts &amp; Beverages</h3> <hr/> <div><div><b>Baklava</b> filo pastry with nuts</div><div>\$8.50</div></div> <div><div><b>Galaktoboureko</b> rice pudding</div><div>\$9.95</div></div>		

Figure 2: User can look over the menu items

### 5.3 Apply/login page

Home	Menu	Login/apply
------	------	-------------



Login:

Email:

Password:

Apply Now:

Email:

Password:

*Figure 3: User can login to an existing registered customer account or apply to become a registered customer*

## 6. Minute of Group Meeting

Meeting #	Discussion
1	Report, Use Cases, Roles
2	Layout of basic functionality, frameworks for development, setting up repository
3	Begin implementing some backend code and front end UI creation

## 7. Github Repo Link

Github Link: <https://github.com/ArbabsLab/Feta>