# Software Requirement Specification Document

Yousef Rady, Omar Sameh, Amr Ahmed, Ziad Ahmed

1 Introduction

1.1 Purpose of this document

Is to identify the project boundaries and describe functionality of project and to the intended audiences and anyone who wants to participate in the future development.

1.2 Scope of this document

Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals.

1.3 overview

Provide the user with fastest way to order online food from the nearest restaurants, according to the desired food cuisine and receive the food by delivery.

1.4 business context

Advertisements

Yearly subscription of restaurants

2 General Description

2.1 Product Functions

Our project (JetDelivery) helps people to order food in the fastest way possible, help restaurants to find more customers by connecting as many as restaurants to our database server, help the customer to find the nearest restaurant that has the desired food cuisine and find more info about the cuisine and menus of those restaurants.

2.2 Similar System Information

Standalone project

2.3 User Characteristics

Normal people , restaurants and companies(normal users) developer( we will provide an API that allows the developers to create applications that it compatible with our servers for future developments)

2.4 User Problem Statement

Lack of restaurants, high search time for food, different languages compatibility.

2.5 User Objectives

Order food from nearest restaurant with high precision with least effort and find possible alternatives.

2.6 General Constraints

Provide easy user interface with least complexity   
provide API compatible with multiple platforms

3 Functional Requirements

## The Web Ordering System

Users of the web ordering system, namely restaurant customers, must be provided the following functionality:

1-create an account .

2-manage their account.

3- log in to the system.

4- social log in .

5- navigate the restaurant menu.

6-select an item from the menu.

7-customize options for a selected item.

8 – add an item to their current order.

9- deals/ offers filter .

10-deals notifications.

11-review their current order

Main Functional Requirements

12- remove an /all item(s) in their current order

13-provide delivery and payment details.

14-place an order.

15- receive confirmation that the order has been accepted

16- provide an order number .

17-order from several restaurant and checking out at end.

18- generate discount codes (promo codes).

19 – provide payment options (cash , online payment).

20-wallet ( can be charged by promo codes ,order cash back).

21-“Jet” cash {have an option to receive money as remaining

cash on it ,and to use it as a payment method when it is

charged}.

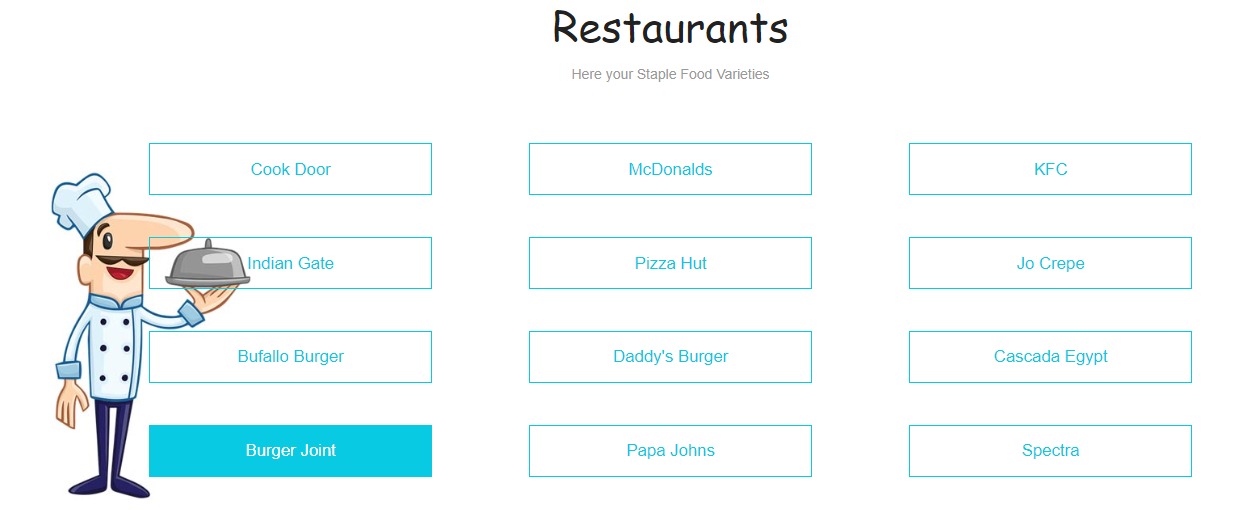
As the goal of the system is to make the process of placing an order as simple as possible for the customer, the functionality provided through the web ordering system is restricted to that.

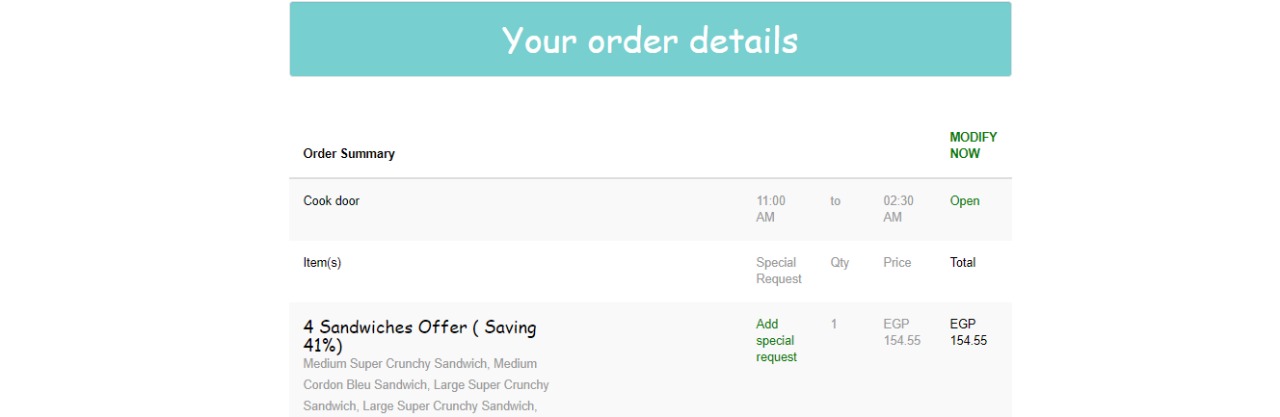
4 Interface Requirements

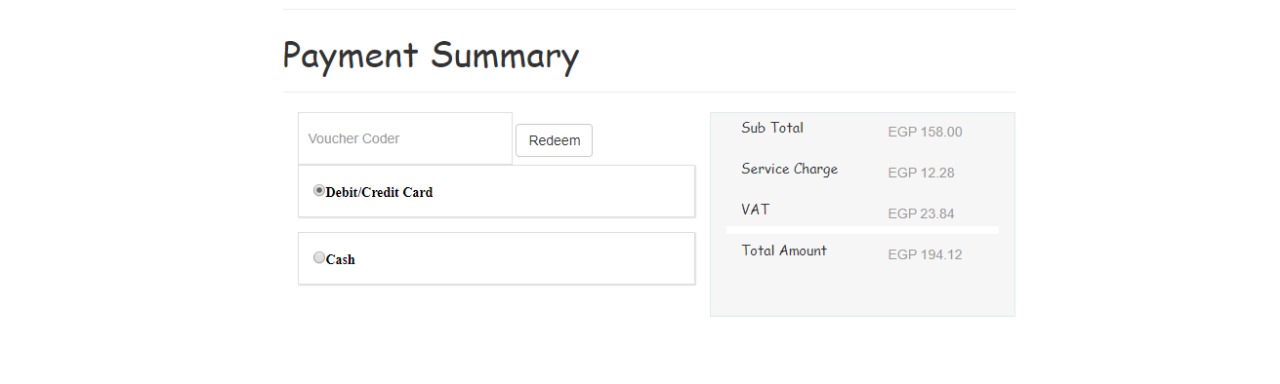
4.1 User Interfaces

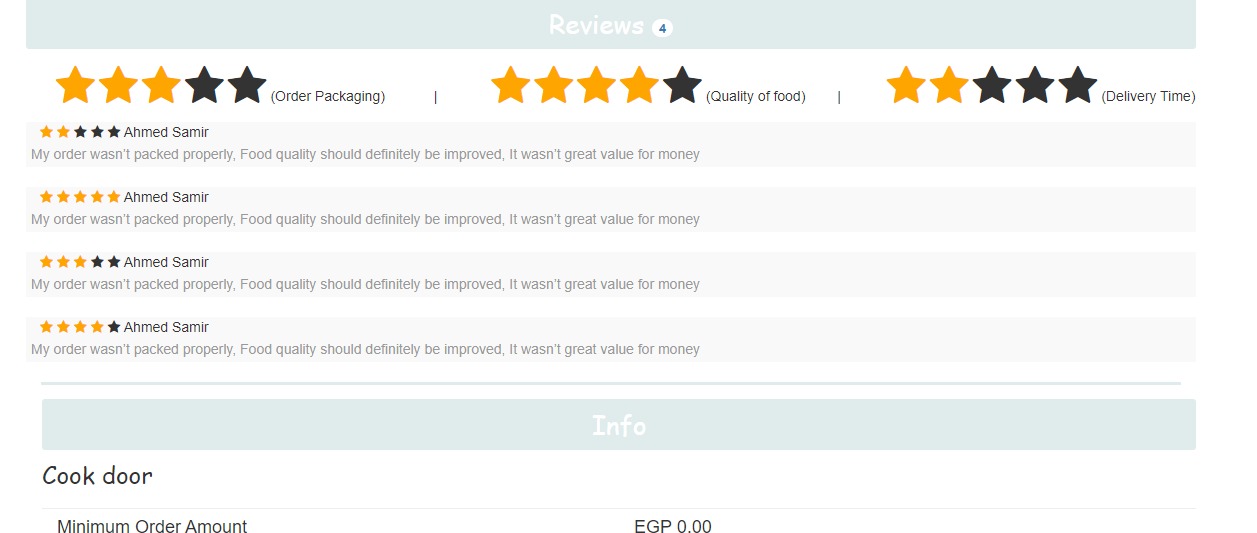
The interface with user will be easy and simple to ensure that he will find the desired food.

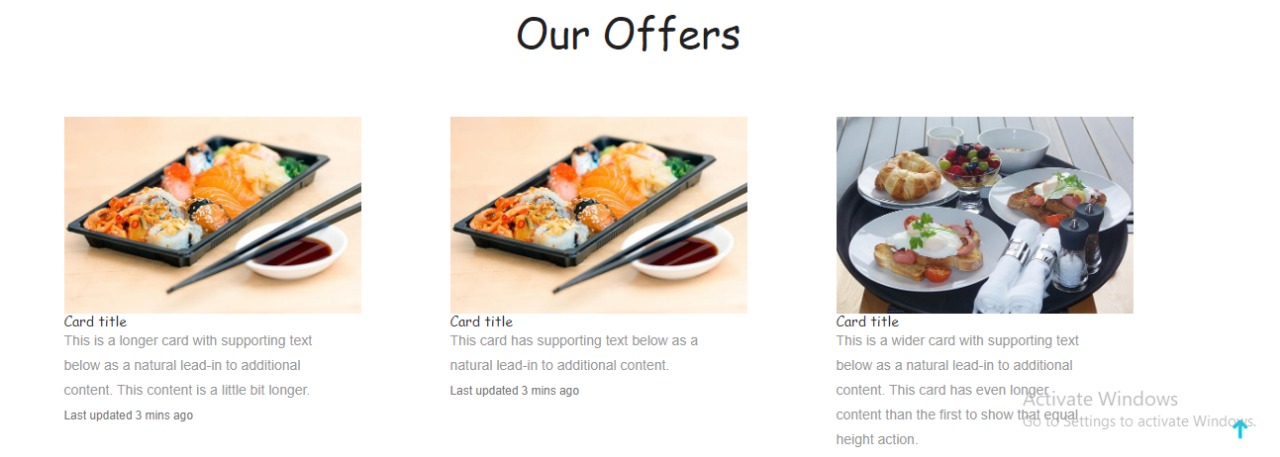
4.1.1 GUI

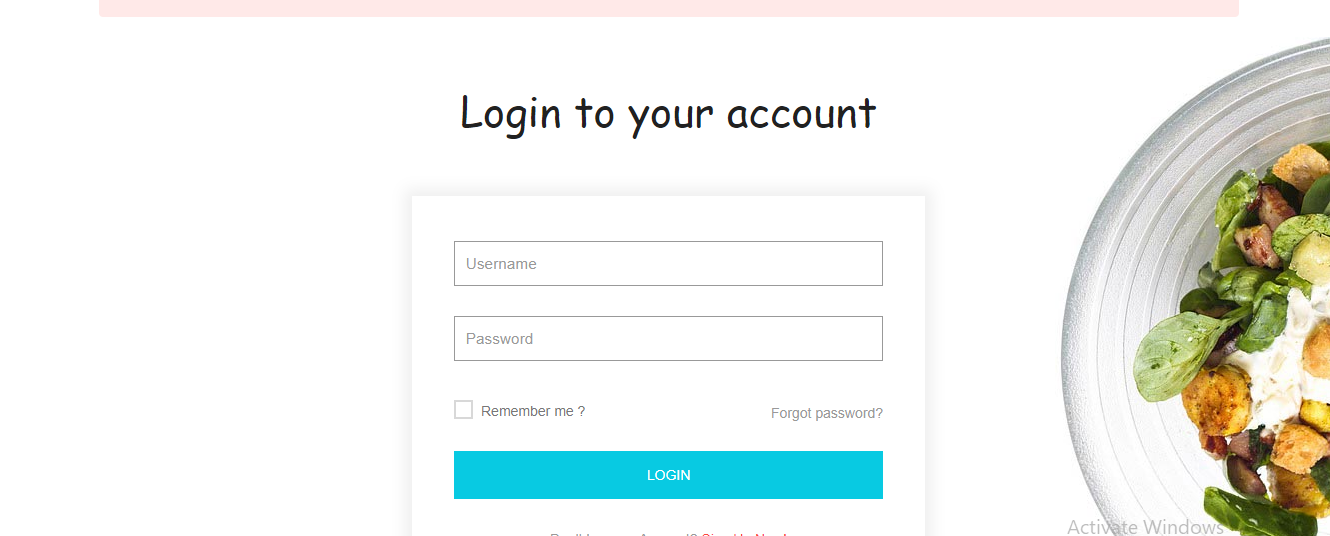
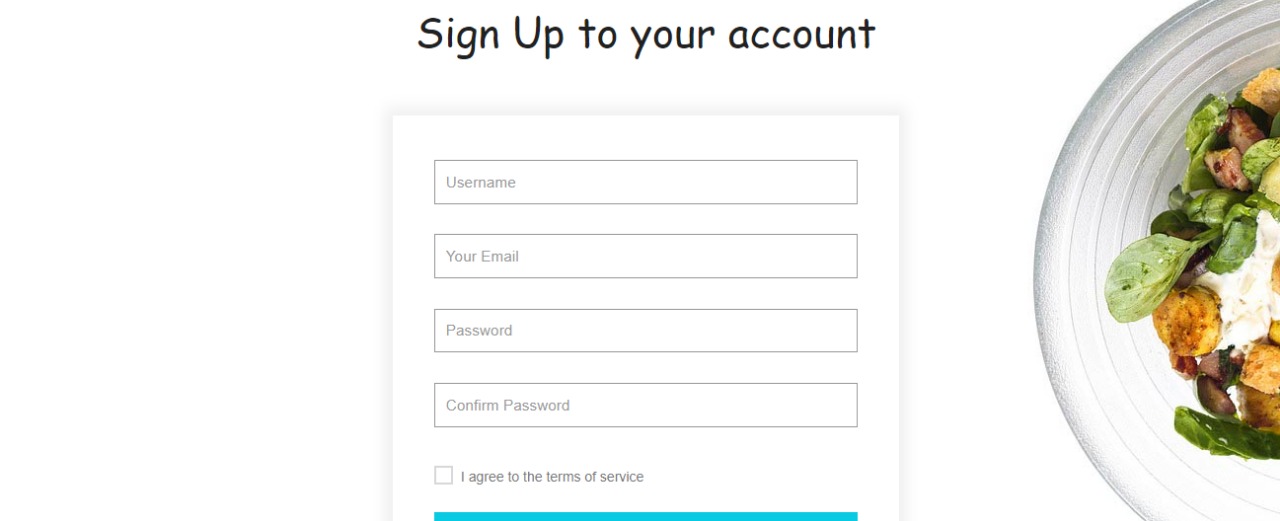












5 Performance Requirements

Provide search results with max delay 0.2 sec and provide the restaurant location

with the highest precision with radius of default value 2.5km.

6 Design Constraints

Must be easy to use interface with simple ads included.

7 Non-functional Requirement

7.1 Security

Encrypted passwords:

All passwords will be hashed in the DB.

7.2 Binary Compatibility

Compatible Devices:

The system will be compatible with all web browsers.

7.3 Reliability

Availability:

System will be available all the time with no crash.

7.4 Maintainability

Service:

System will be maintainable, any known defect will be able to be corrected.

7.5 Portability

Portable System:

System will run on all computers with any browser.

7.6 Re-usability

Familiar Interface:

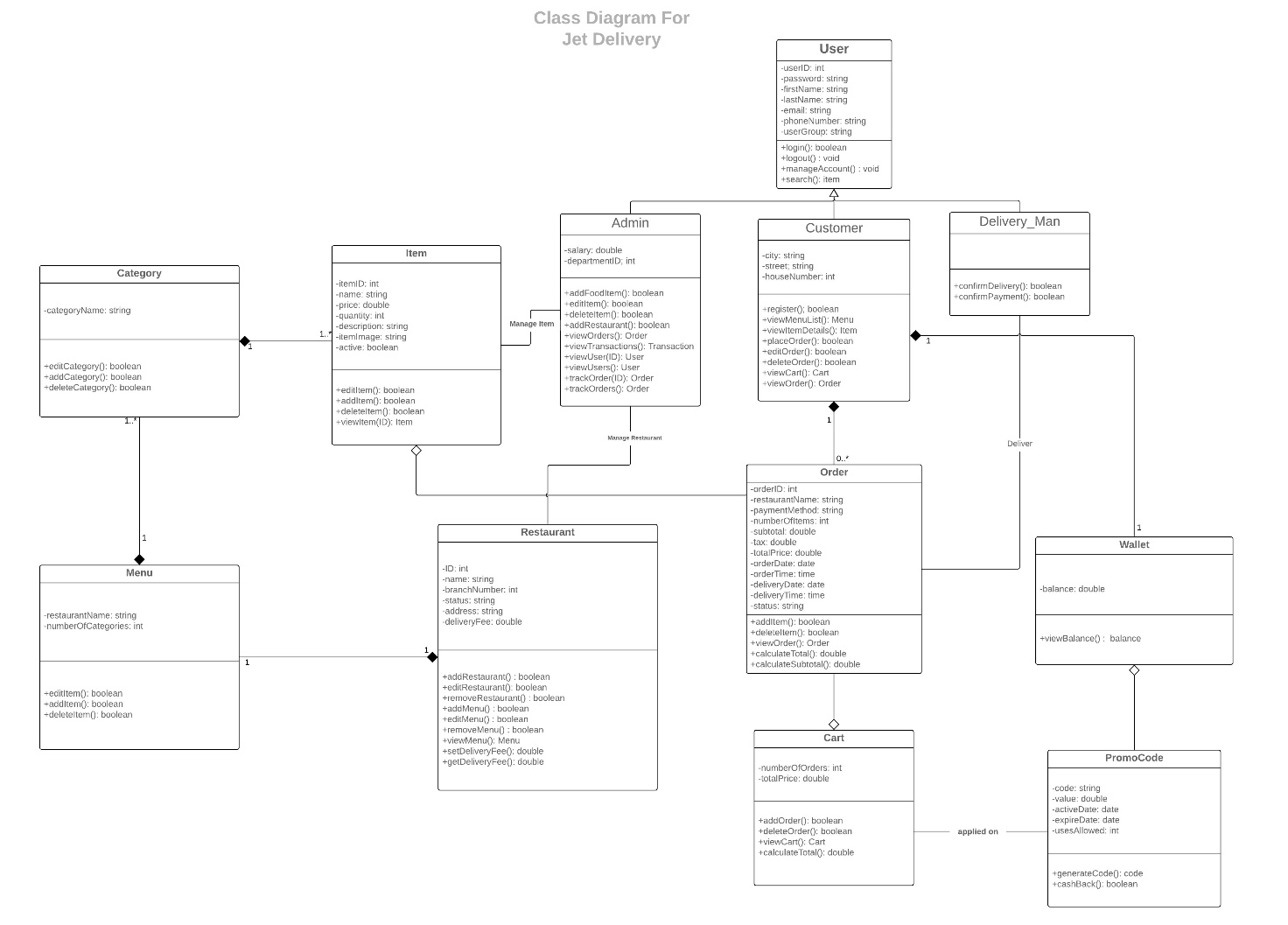
UI must be user friendly.

7.7 Serviceability

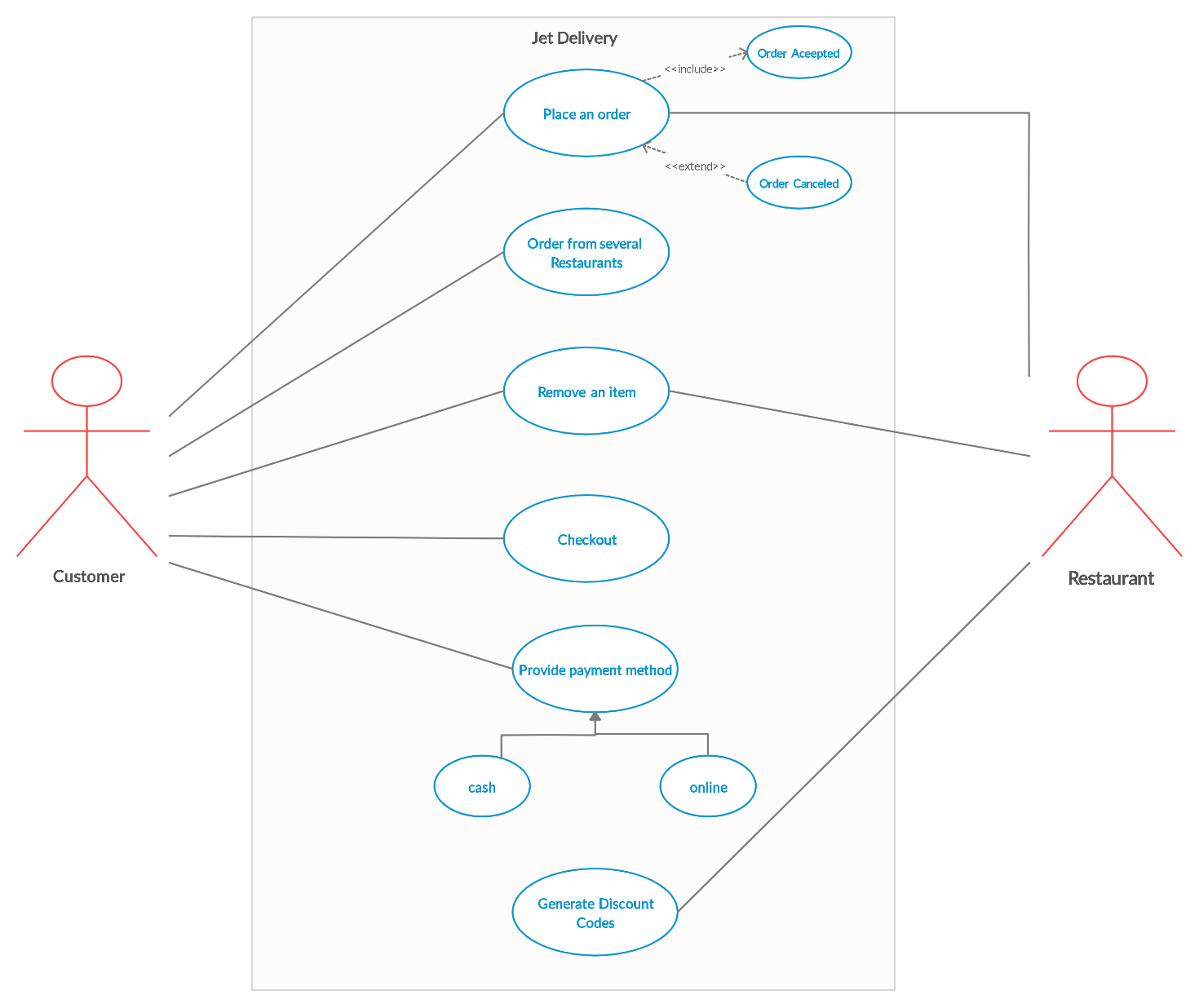
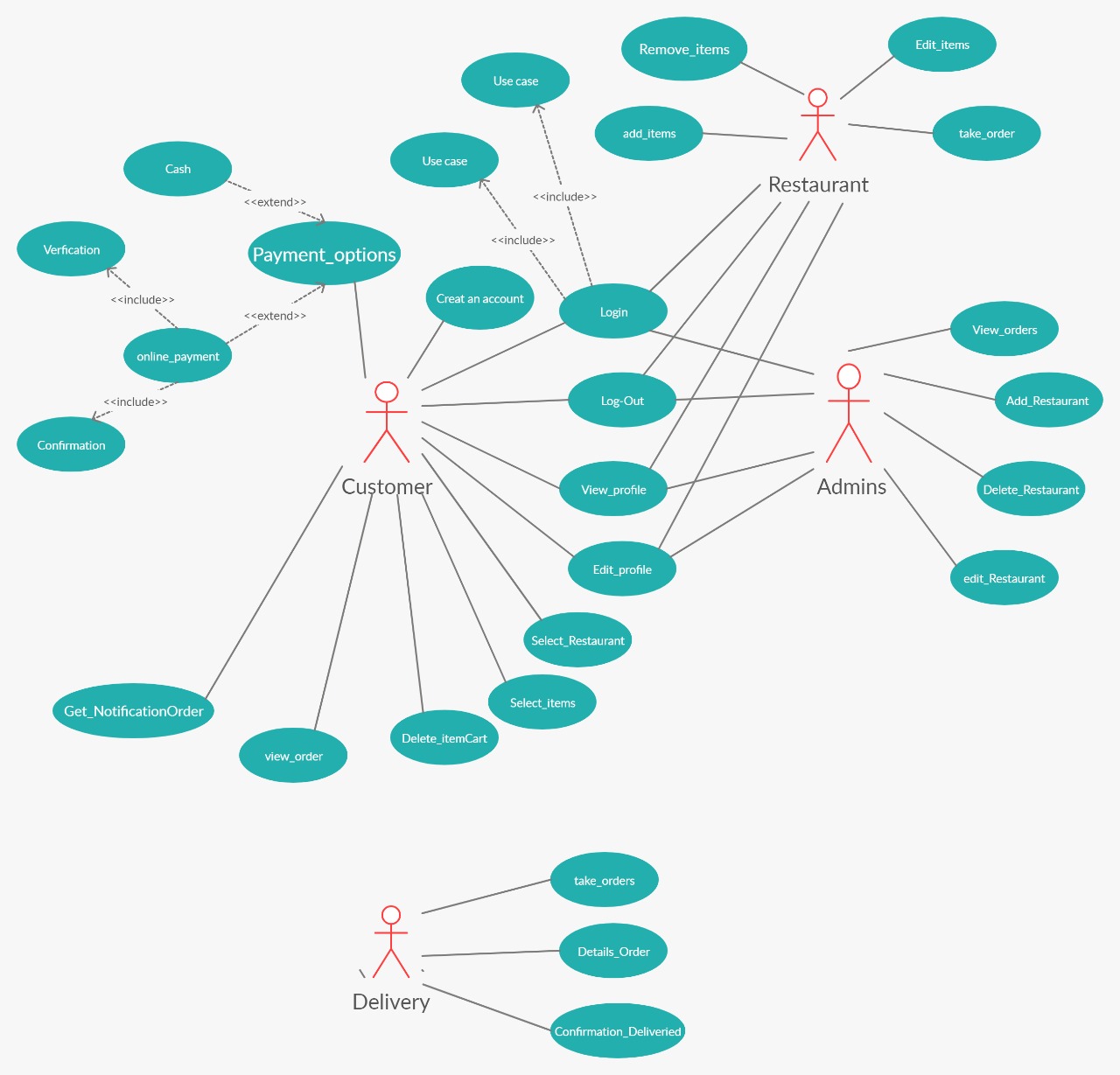
Test use case:

All major use cases must be tested well.

8 Class Diagram



9 Operational Scenarios



Use case Description

**1-Place an order**

**Scenario** : Customer wants to make an order

**Main Success Scenario :** customer wants to order food and order has been accepted

**Unsuccessful Scenario :** restaurant cancel the order

**2- order from different restaurant**

**Scenario**: customer ordered from a restaurant and he wanted to get something else from different restaurant

**Main Success Scenario :** he will order from other restaurants and orders will be added successfully to the cart

**Unsuccessful Scenario:** cart doesn’t accept any additional order

**3- remove an item**

**Scenario:** customer wants to remove item from his order

**Main success Scenario :** he removed items from his orders successfully

**Unsuccessful Scenario :** order becomes in delivery stage and customer cant remove or change anything on his order

**4-Checkout**

**Scenario :** customer finished his order and wanted to see how much he will pay

**Main success scenario :** customer checkout his orders successfully

**Unsuccessful scenario :** he has entered wrong visa number so he couldn’t checkout

**5- provide payment method**

**Scenario :** customer choosing his payment method cash or online with visa card

**Main success scenario :** customer choose to pay in cash so when delivery arrived he gave his money

**Unsuccessful scenario :** customer choose to pay in cash but application didn’t accept and force him to pay online

**6- generate discount codes**

**Scenario :** customer after he used application several times application gives him a discount code

**Main success scenario :** application provide customer a promo code of 30% discount on healthy food

**Unsuccessful scenario :** customer order food without getting any discount from application

**7- order cash back**

**Scenario :** customer after using application to order food he gain cash back added to his wallet

**Main success scenario :** customer can use what he saved in his wallet as a cash back from his previous orders

**Unsuccessful scenario :** customer find his wallet empty

Use Case Specifications

|  |  |
| --- | --- |
| **Name** | Create Account |
| **Brief Description** | The Create Account use case allows the User to create a login and become a Registered User. |
| **Actor(s)** | Guest |
| **Flow of Events** | |
| **Basic Flow** | |
| This use case starts when the User accesses the system feature that enables him/her to create an account by entering information that is maintained in the User’s account.   1. The User enters the required *User Account* information values and requests that the system saves the entered values. 2. The system validates the entered *User Account* information. 3. The values for the *User Account* information are stored in the User’s account. The system notifies the User that the account has been created. 4. The use case ends | |
| **Alternate Flows** | |
| **Title** | **Description** |
| User Cancels Request | At any time, the User may choose to cancel the account creation. At which point, the processing is discontinued, the user account remains unchanged, and the user is notified that the account management request has been cancelled. |
| User Enters Invalid User Account Information | If during Create Account, the system determines that the User entered invalid *User Account* information, the following occurs:   1. The system describes which entered data was invalid and presents the User with suggestions for entering valid data. 2. The system prompts the User to re-enter the invalid information. 3. The User re-enters the information and the system re-validates it. 4. If valid information is entered, the User Account Information is stored. 5. If invalid information is entered, the Entered Information is Invalid alternative flow is executed again. This continues until the User enters valid information, or chooses Cancel (see the User Cancels Account Management Request alternative flow).   Invalid *User Account* information:   * Missing information items * Username already exists in the system * *User Account* information entered does not comply to its definition in the glossary * Not well formed e-mail address * Offending words in any part of the *User Account* information |
| **Pre-Conditions** | |
| **Title** | **Description** |
| None |  |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | The User entered data is stored in the user account. Confimation is sent to the appropriate email address. |
| The user account was not created | The User entered invalid data or chose to cancel the account creation request. In either case, no account will be created. |
| **Extension Points** | |
| None | |

|  |  |
| --- | --- |
| **Name** | Sign in |
| **Brief Description** |  |
| **Actor(s)** | User, Administrator |
| **Flow of Events** | |
| **Basic Flow** | |
| This use case starts when the User accesses the sign in feature of the system.   1. The system prompts the User for his/her username and password. 2. The User enters his/her username and password. 3. The system validates the entered information, making sure that the entered username and password are valid for one user account in the system, and that the required password is entered for the entered username. 4. The User is signed in. The system displays a message indicating that the user is signed in. 5. The use case ends. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| New User | If the User does not have an account, the System will give the User the opportunity to create an account. See the **Create Account** use case. Once the account is created, the User is considered signed in. |
| User Forgot User Name/Password | If the User forgot his/her user name or password, the System will prompt the user to answer his/her security question provided in the Create Account stage. Provided that it is answered correctly, the user name and password are emailed to the email address provided when the account was created. |
| User Fails Authentication | If the User entered an invalid username and/or password, the following occurs:   1. The system describes the reasons why the User failed authentication. 2. The system presents the User with suggestions for changes necessary to allow the User to pass authentication. 3. The system prompts the User to re-enter the valid information. 4. The Basic Flow continues where the User enters new information (see step 2 of the Basic Flow). |
| **Pre-Conditions** | |
| **Title** | **Description** |
| None |  |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | The User is authenticated and the system displays all features available for the role the user is associated with as defined in his/her user account. |
| User not signed in | This can occur because the User repeatedly entered invalid sign in information. The User has been notified of the reason why he/she was not signed in. The User is not authenticated and remains in the Anonymous User role. |

|  |  |
| --- | --- |
| **Name** | Manage Account |
| **Brief Description** | The Manage Account use case allows the User to update the User Account Information maintained in the User's account. |
| **Actor(s)** | Registered User |
| **Flow of Events** | |
| **Main Success Scenario** | |
| This use case starts when the User accesses the system feature that enables him/her to update the information that is maintained in the User's account.   1. The system displays the *User Account* information currently stored for the User. 2. The User enters the desired *User Account* information values and requests that the system saves the entered values. 3. The system validates the entered *User Account* information. 4. The values for the *User Account* information are stored in the User’s account. The system notifies the User that the account has been updated. 5. The use case ends | |
| **Alternate Flows** | |
| **Title** | **Description** |
| User Cancels Request | At any time, the User may choose to cancel the account update/deactivation. At which point, the processing is discontinued, the user account remains unchanged, and the user is notified that the account management request has been cancelled. |
| User Enters Invalid User Account Information | If during Modify Account, the system determines that the User entered invalid *User Account* information, the following occurs:   1. The system describes which entered data was invalid and presents the User with suggestions for entering valid data. 2. The system prompts the User to re-enter the invalid information. 3. The User re-enters the information and the system re-validates it. 4. If valid information is entered, the User Account Information is stored. 5. If invalid information is entered, the Entered Information is Invalid alternative flow is executed again. This continues until the User enters valid information, or chooses Cancel (see the User Cancels Account Management Request alternative flow).   Invalid *User Account* information:   * Missing information items * Username already exists in the system * *User Account* information entered does not comply to its definition in the glossary * Not well formed e-mail address * Offending words in any part of the *User Account* information |
| **Pre-Conditions** | |
| **Title** | **Description** |
| User is signed in | The User must be signed in before the User can edit or deactivate his/her account. See the Sign In use case |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | The User entered data is stored in the user account. |
| The user account was not updated | The User entered invalid data or chose to cancel the account management request. In either case, there is no change to the user account. |
| **Extension Points** | |
| None | |

|  |  |
| --- | --- |
| **Name** | View Items |
| **Brief Description** | The View Items use case allows the User to view current selection of Textbooks available |
| **Actor(s)** | Registered User, Administrator |
| **Flow of Events** | |
| **Basic Flow** | |
| This use case starts when the User accesses one of the following system features:   1. Display all textbooks available. 2. Use of drop down menus to filter the results based on the criteria selected. 3. Direct search by distinct criteria. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| None |  |
| **Pre-Conditions** | |
| **Title** | **Description** |
| None |  |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Textbooks would be displayed on the current page. |
| No entries | No textbooks available from the given criterion. |
| **Extension Points** | |
| None | |

|  |  |
| --- | --- |
| **Name** | Place Order |
| **Brief Description** | This Use Case describes the process by which orders are entered into the order processing system. |
| **Actor(s)** | Registered User, |
| **Flow of Events** | |
| **Basic Flow** | |
| 1. The Use Case starts when the customer selects Place Order.  2. The system displays the Place Order Screen  3. The customer enters their name and address.  4. The customer will enter product codes for products they wish to order 5. For each product code entered  4.1 Use Give Product Information  4.2 The system will add the price of the item to the total.  End  6. The customer will enter credit card payment information.  7. The customer will select submit.  8. The system will verify the information and save the order as pending  9. Use Update Account  10.The order is marked confirmed, an order id is returned to the customer, and the use case ends.. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| User cancel | At any time before selecting submit, the customer can select cancel. The order is not saved and the use case ends. |
| **Pre-Conditions** | |
| **Title** | **Description** |
| None |  |
| **Post-Conditions** | |
| **Title** | **Description** |
|  |  |
|  |  |
| **Extension Points** | |
| None | |

|  |  |
| --- | --- |
| **Name** | Update Account |
| **Brief Description** | This Use Case interacts with the Accounting System to apply charges or credits to an account.. |
| **Actor(s)** | Accounting System |
| **Flow of Events** | |
| **Basic Flow** | |
| 1. The Use Case begins when a request to update an account is received.  2. The system will send credit card info and amount of credit or debit to the Accounting System.  3. The Accounting System will send a status of OK.  4. The Use Case ends.. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| User cancel | At any time before selecting submit, the customer can select cancel. The order is not saved and the use case ends. |
| **Pre-Conditions** | |
| **Title** | **Description** |
| None |  |
| **Post-Conditions** | |
| **Title** | **Description** |
|  |  |
|  |  |
| **Extension Points** | |
| Account overdrawn ■ Account doesn’t exist ■ Accounting System not available | |

|  |  |
| --- | --- |
| **Name** | Get Status on Order |
| **Brief Description** | This Use Case describes the process by which a customer can get the current status on any of their orders... |
| **Actor(s)** | Customer ■ Customer Rep |
| **Flow of Events** | |
| **Basic Flow** | |
| 1. The Use Case starts when the user selects review order.  2. Use Search for Order.  3. The system will display the order status screen and the Use Case ends. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| ……. | ……….. |
| **Pre-Conditions** | |
| **Title** | **Description** |
| None |  |
| **Post-Conditions** | |
| **Title** | **Description** |
|  |  |
|  |  |
| **Extension Points** | |
| Order not found | |

|  |  |
| --- | --- |
| **Name** | Log Out |
| **Brief Description** | Disable Customer access to the system |
| **Actor(s)** | Customer |
| **Flow of Events** | |
| **Basic Flow** | |
| 1) The customer selects ‘Log Out’ from the navbar  2) The customer disables access to the system | |
| **Alternate Flows** | |
| **Title** | **Description** |
| ……. | ……….. |
| **Pre-Conditions** | |
| **Title** | **Description** |
| Logged in | The customer is already logged in |
| **Post-Conditions** | |
| **Title** | **Description** |
|  |  |
| Logged out | The customer no longer requires access to the system to perform their order |
| **Extension Points** | |
| Order not found | |