**OrderLine**

A Project Submitted in Partial Fulfillment of the Requirements for   
CSC 423 - Software Engineering

Department of Computer Science

Faculty of Natural and Applied Sciences  
Notre Dame University – Louaize  
Zouk Mosbeh, Lebanon

Mohamed Ali

Toni Habib

Elio Mourad

Spring 2019

Abstract

In this paper we are presenting an application called OrderLine where customers can find restaurants online, place order and pay online. In addition, the customers will be able to order food from restaurants with no delivery system. The business vision we are seeking in this project is to offer people the chance to order food from any restaurant, any area and specifically from restaurants with no delivery system. In order to solve this issue, the application has its own delivery members that are going to deal with restaurants with no delivery systems by picking up orders for the customers. With this new system we are trying to give the customers a better experience with online food delivery.

**Table of Contents**

[Chapter 1 : Business Vision, Glossary, and Risks 7](#_Toc2612825)

[1.1 Introduction 7](#_Toc2612826)

[1.2 Definitions, Acronyms, and Abbreviations 7](#_Toc2612827)

[1.2.1 <SnapShot> 7](#_Toc2612828)

[1.2.2 <LetsDeliver> 7](#_Toc2612829)

[1.2.3 <MyArea> 7](#_Toc2612830)

[1.3 Positioning 7](#_Toc2612831)

[1.3.1 Business Opportunity 7](#_Toc2612832)

[1.3.2 Problem Statement 7](#_Toc2612833)

[1.3.3 Product Position Statement 8](#_Toc2612834)

[1.4 User Summary and Competition 8](#_Toc2612835)

[1.4.1 User Summary 8](#_Toc2612836)

[1.4.2 Alternatives and Competition 9](#_Toc2612837)

[1.5 Constraints 9](#_Toc2612838)

[1.6 Other Requirements 9](#_Toc2612839)

[1.6.1 Applicable Standards 9](#_Toc2612840)

[1.6.2 System Requirements 10](#_Toc2612841)

[1.6.3 Performance Requirements 10](#_Toc2612842)

[1.6.4 Environmental Requirements 10](#_Toc2612843)

[1.7 Risks 10](#_Toc2612844)

[1.7.1 < Application system crash> 10](#_Toc2612845)

[1.7.2 <Delivery> 11](#_Toc2612846)

[Chapter 2 : Requirements Elicitation 12](#_Toc2612847)

[2.1 Adopted Technique(s) 12](#_Toc2612848)

[2.2 Questions 12](#_Toc2612849)

[2.3 Stakeholders’ General Statement 12](#_Toc2612850)

[Chapter 3 : Enumerated Requirements and Increments 13](#_Toc2612851)

[3.1 Functional Requirements 13](#_Toc2612852)

[3.2 Non-Functional Requirements 14](#_Toc2612853)

[3.4 Increments 15](#_Toc2612854)

[3.4.1 Release Backlogs 15](#_Toc2612855)

[3.4.3 Sprints 16](#_Toc2612856)

[Chapter 4 : Functional Requirements Specification 17](#_Toc2612857)

[4.1 Use Case Diagram 17](#_Toc2612858)

[4.3 Use Case Specification for <Use Case Name> 18](#_Toc2612859)

[4.3.1 Brief Description 18](#_Toc2612860)

[4.3.2 Actors 18](#_Toc2612861)

[4.3.3 Basic Flow of Events 18](#_Toc2612862)

[4.3.4 Alternative Flows of Events 18](#_Toc2612863)

[4.3.5 Special Requirements 19](#_Toc2612864)

[4.3.6 Preconditions 19](#_Toc2612865)

[4.3.7 Postconditions 19](#_Toc2612866)

[4.3.8 Extension Points 19](#_Toc2612867)

[Chapter 5 : Domain Analysis 21](#_Toc2612868)

[5.1 Domain Model Class Diagram 21](#_Toc2612869)

[5.2 Traceability Matrix 23](#_Toc2612870)

[Chapter 6 : Interaction and Data Flow Diagrams 25](#_Toc2612871)

[6.1 Sequence Diagrams 25](#_Toc2612872)

[6.3 Data Flow Diagram 26](#_Toc2612873)

[6.3.1 Context Diagram (Level 0) 26](#_Toc2612874)

[6.3.2 Level 1 26](#_Toc2612875)

[Chapter 7 : Collaboration and Individual Contribution 27](#_Toc2612876)

[7.1 Collaboration Approach 27](#_Toc2612877)

[7.2 Problems that Occurred 27](#_Toc2612878)

[7.3 Individual Contribution Breakdown 27](#_Toc2612879)

[List of References 28](#_Toc2612880)

[Appendix A: Presentation 29](#_Toc2612881)

List of Tables

[Table 1.1: Problem Statement 8](#_Toc465016241)

[Table 1.2: Product Positioning 8](#_Toc465016242)

[Table 1.3: User Summary 9](#_Toc465016243)

[Table 3.1: List of Functional Requirements 15](#_Toc465016244)

[Table 3.2: List of Functional Requirements 15](#_Toc465016245)

[Table 3.2: Core Features Release Backlog 16](#_Toc465016246)

[Table 3.2: Release Backlog 1 16](#_Toc465016247)

[Table 3.5: Core Features Release Backlog Sprints 17](#_Toc465016248)

[Table 5.1: Domain Concept / Use Case Traceability Matrix 25](#_Toc465016249)

[Table 7.1: Individual Contribution 29](#_Toc465016250)

List of Figures

[Figure 4.1: Bank ATM Use Case Diagram 19](#_Toc465016251)

[Figure 5.1: Student Enrollment in Seminar Class Diagram 24](#_Toc465016252)

[Figure 6.1: Basic Sequence Diagram for Login 27](#_Toc465016253)

# : Business Vision, Glossary, and Risks

## Introduction

In this chapter we will be identifying some definitions and abbreviations so the user have better understanding about the features. Also, we will be setting our business vision of this project and what problem it will be solving. In addition, a future competitor application may occur, so we will discuss their major strength and weaknesses. At the end, we will be listing some major risks that may affect the project.

## Definitions, Acronyms, and Abbreviations

### <SnapShot>

SnapShot is a feature that allow restaurant owners using the software to take a small video no more than 15 seconds to show the atmosphere of the restaurant.

### <LetsDeliver>

LetsDeliver is a feature that allow the users to order from restaurants that doesn’t have a delivery system.

### <MyArea>

MyArea is a feature that allow users to show restaurants on their area.

## Positioning

### Business Opportunity

Orderline software lists the restaurants in a sorted way by categories. The software gives the opportunity for people to have a Full/Part time job as Delivering orders position. In addition, it gives opportunity to restaurants with no delivery system to benefit from the system.

### Problem Statement

The software gives the users the opportunity to order food from restaurants with no delivery system.

Table 1.1: Problem Statement

|  |  |
| --- | --- |
| The problem of | Implementing a Delivery system for restaurants that doesn’t own a delivery system. |
| Affects | Users |
| the impact of which is | Users cant order from some restaurants because these restaurants doesn’t have delivery system. |
| a successful solution would be | Creating a software that have his own delivery guys. Users can order from restaurants with no delivery system and the software delivery guys will pick up the order and deliver it. |

### Product Position Statement

The product tends to provide a service which is an own delivery system so that restaurants with no delivery system can benefit from it.

Table 1.2: Product Positioning

|  |  |
| --- | --- |
| For | People |
| Who | Online Ordering. |
| The Orderline | is a Food & Drink category. |
| That | Takes percentage from the order facture still cheaper than paying salary for a delivery guy for small restaurants. |
| Unlike | Online Ordering Food Software |
| Our product | Has its own delivery system. |

## User Summary and Competition

### User Summary

Application users, delivery guys, moderators and support team.

Table 1.3: User Summary

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Responsibilities** |
| Application Users | The user who orders from the software. | * Rate and give feedback to the software. |
| Delivery Guys | The guy who pick up the order from a restaurant and delivers it. | * **Pick up the right order** * **Delivering to the right location** * **Check if the payment is done through the software or not.** |
| Moderators | The guy who keep in touch with the customers. | * Handles Reviews and Feedbacks. * **Helps guiding people using the app** * **Raise all technical issues to the support team.** |
| Support Team | They are the developers of the software. | * They handle all technical issues that are related to the software (bugs, database, design) * **Testing and keep the software updated.** |
| Restaurant Managers | The restaurant managers handle the restaurant profile | * Handles the restaurant profiles |

### Alternatives and Competition

A future competitive application that could include a delivery system where a user adds a destination such as a restaurant, grocery shop, pharmacy..etc, and gets his items delivered. The major strength would be that they get deliveries from all types of shops. Major weakness would be that if a lot of users are using the app the orders might take more time to be delivered.

## Constraints

If the server host is down the application wont function.

## Other Requirements

### Applicable Standards

The software follows several standards. First, for legal regulations we are using UCC standard. Second, for communication standards we are using TCP/IP standard. Third, for platform compliance we are using IOS and Android. Last, for safety standards we are ISO.

### System Requirements

The application will be using Android and IOS as a host operating systems. The application needs about 15 MB storage. There are no peripherals. In addition, we are using Google maps as a companion software.

### Performance Requirements

None

### Environmental Requirements

None

## Risks

### < Application system crash>

This is considered the highest risk

Description

The application is running on a specific software version, and a new operating system is upgraded, the system might crash.

Impacts

May prevent the users to use the application which could make the application unreliable to use.

Indicators

Whenever the application crash, the system asks the user to report the crash.

Mitigation Strategy

Releasing updates in increments in order to detect technical issues.

Contingency Plan

Releasing a beta version to be tested by specified users and technical support team.

### <Delivery>

Late delivery.

Description

the delivery may be delayed due to accident or traffic.

Impacts

The food would not be delivered on time which results in a customer being unsatisfied.

Indicators

Some indications could include the customers giving bad ratings due to delivery being late. Other indication is when a customer or the delivery guy calls in due to late delivery.

Mitigation Strategy

Having specific number of delivery guys for each area to maintain the orders.

Contingency Plan

Having more than one delivery guy in case of an accident.

# : Requirements Elicitation

## Adopted Technique(s)

An interview was made with a restaurant manager for an hour with and without delivery system how can they benefit from the application.

## Questions

1. What payment methods do you accept?
2. How many categories do you offer?
3. How much time does it take for food to be ready on average?
4. How many takeaways do you serve per week?
5. Do you mind adding a delivery charge on your total bill?
6. Do you often offer daily platters or special offers?

## Stakeholders’ General Statement

Stakeholders general statement was some people would prefer online payment and other cash on delivery. His restaurant provides five categories including appetizers, platters, sandwiches, desserts, and beverages. Food gets done usually between ten to twenty minutes. We usually serve about 30 to 35 takeaways per week. I would add a delivery charge as long it doesn’t exceed 1.5$. His restaurant usually offers different types of daily platters, except Sunday the restaurant offers open buffet.

# : Enumerated Requirements and Increments

## Functional Requirements

Table 3.1: List of Functional Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Priority** | **Description** |
| REQ1 | 2 | The system shall provide a history log for all order attempts for later review. |
| REQ2 | 1 | The system shall be able to facilitate the accurate payment process. |
| REQ3 | 2 | The system shall keep track of the delivery order. |
| REQ4 | 1 | The system shall provide security regarding users personal information. |
| REQ5 | 3 | The user shall be able to customize his/her selected order. |
| REQ6 | 1 | The user shall be able to place an order. |
| REQ7 | 3 | The user shall be able to use the live chat if any technical issue happened. |
| REQ8 | 2 | the user shall be able to review the restaurant at any time. |
| REQ9 | 1 | The restaurant managers should be able to customize his items. |

## Non-Functional Requirements

Table 3.2: List of Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Priority** | **Description** |
| REQ1 | 1 | The system shall be available 99.999 percent of the time. |
| REQ2 | 3 | The system shall store the orders automatically as a Draft in case of system failure (eg. System crash , Power Loss). |
| REQ3 | 2 | The user shall be able to change password or account information due to security issues. |
| REQ4 | 2 | The user shall be in touch with the support team in less than 15 minute. |
| REQ5 | 1 | Any feature in the application should take no longer than 7 seconds to respond. |
| REQ6 | 1 | The system shall be capable of supporting an arbitrary number of active user payments, that is no payment shall be lost. |

## Increments

### Release Backlogs

Table 3.2: Core Features Release Backlog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement IDs** | | | | | |
| REQ2 | REQ6 | REQ9 |  |  |  |
| Payment method | Placing order | Restaurant customization |  |  |  |

Payment method as a core feature because the user should be able to make safe payments by either using their credit cards or cash on delivery. Placing order is another important core feature that should be included in the application, because the customer should be able to place his desired order. In addition restaurant customization is a necessity in the application, since it enables the restaurant managers to add, edit or remove items from their main restaurant page.

Table 3.2: Release Backlog 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement IDs** | | | | | |
| REQ1 | REQ3 | REQ4 | REQ5 | REQ7 | REQ8 |
| History backlog | Tracking order | System security | Customizing order | Live chat support | Reviewing restaurants |

### Sprints

Table 3.5: Core Features Release Backlog Sprints

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement IDs** | | | | | |
| Sprint 1 | | Sprint 2 | | Sprint 3 | |
| REQ2 |  | REQ6 |  | REQ9 |  |
| 10 days |  | 14 days |  | 22 days |  |

# : Functional Requirements Specification

## Use Case Diagram

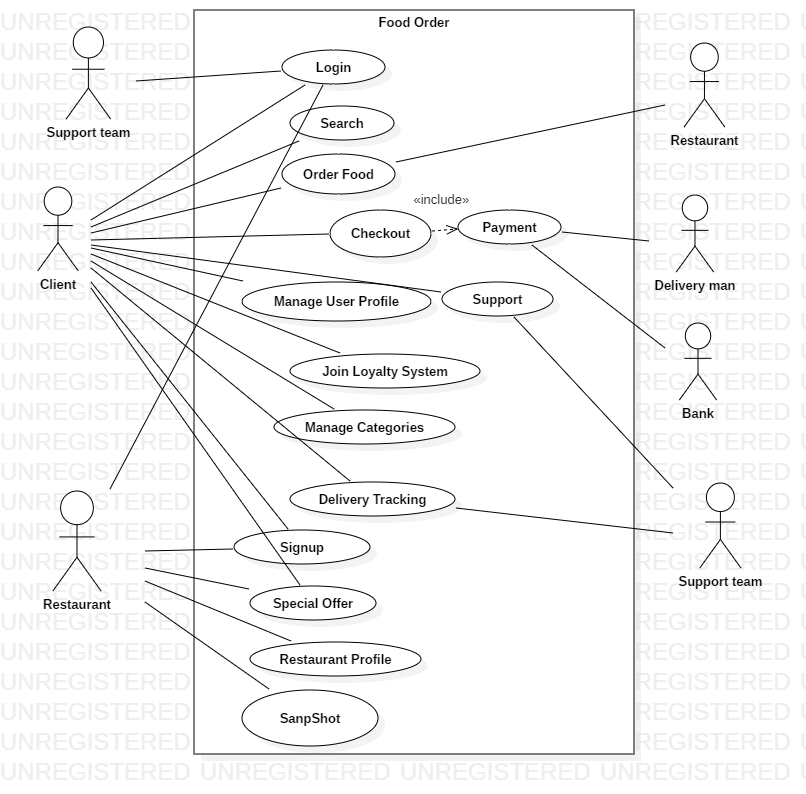


Figure 4.1: OrderLine Use Case Diagram

## Use Case Specification for <Search>

### Brief Description

The search use case allow users to search for restaurants by categories

### Actors

Primary actors: Users.

### Basic Flow of Events

1. The user triggers the search button.
2. The user search for restaurants by name, location or categories.
3. The system distributes the restaurants based on user search.
4. Repeat step b until the user finds the desired restaurant.

### Alternative Flows of Events

< Wrong Information >

1. The user enters a wrong restaurant name or category.
2. The system asks the user to search again.

### Special Requirements

None

### Preconditions

None

### Postconditions

< Desired Restaurant >

The system will show the user the desired restaurant based on user search.

### Extension Points

None

## Use Case Specification for < Order Food >

### Brief Description

The order food use case should allow users to add orders to his/her cart and make special orders.

### Actors

Primary actors: Users.

### Basic Flow of Events

1. The user checks the menu of the restaurant.
2. The system shows the menu of the restaurant with available food categories.
3. The user picks orders that he/she want.
4. The system asks the user for the quantity of each order.
5. The user choose the quantity for his orders.
6. Repeat steps b-e until the user is satisfied.
7. The system add the orders to the user cart.
8. The system saves the orders on the cart until the user conforms.

### Alternative Flow of Events

< Customization >

1. User picks a specific order.
2. The system asks user to customize the order.
3. Use case flow continue with basic flow step b.

< Delete >

1. The user delete orders from the cart.
2. Use case flow continue with basic flow step b.

### Special Requirements

None

### Preconditions

The system must be configured with username and password.

### Postconditions

None

### Extension Points

None

## Use Case Specification for<Join Loyalty System>

### Brief Description

The join loyalty system use case allow users to benefit from special offers from restaurants and discounts. The users can join the

### Actors

Primary users: Clients.

### Basic Flow of Events

1. The client access the loyalty system service.
2. The client clicks on join loyalty system.
3. The system asks the client for payment.
4. The client completes payment.
5. The client confirms.
6. The client joins the loyalty system.

### Alternative Flow of Events

None

### Special Requirements

Having a credit card to be able to pay online.

### Preconditions

None

### Postconditions

The client will have access to special offers only accessible by loyalty members.

### Extension Points

None

## Use Case Specification for<Delivery Tracking>

### Brief Description

The delivery tracking use case allow clients to track orders from the moment of ordering until the order reaches the client place.

### Actors

Primary actors: clients.

Secondary actors: delivery guy and restaurant.

### Basic Flow of Events

1. The client order food and the system immediately start tracking order.
2. The client tracks the order while the food is being prepared by the restaurant.
3. The delivery guy pick up the order.
4. The client tracks the order that is picked up by the delivery guy.
5. The order reaches the client.
6. The system immediately stop tracking the order.

### Alternative Flow of Events

None

### Special Requirements

None

### Preconditions

The user must enable his location to be able to track the order.

### Postconditions

None

### Extension Points

None

## Use Case Specification for <Login>

### Brief Description

The login use case allows users to login into their already created accounts.

### Actors

Primary actor: User, restaurant manager, moderators and support team.

### Basic Flow of Events

1. The user triggers the login button
2. The system shows the login page
3. The user enters his username and password and clicks login
4. The system checks if the entered username and password match
5. If the username and password match, the system welcomes the user and takes them to their specified page.

### Alternative Flows of Events

1. If the username was not found the system will take the user to a sign up page
2. The user will enter a new username and password and click on create
3. The system will check if the username is not already in use and creates the account
4. Usecase flow continues in the basic flow from point.

### Special Requirements

None

### Preconditions

A user must have already signed up with a username and password prior the login step.

### Postconditions

None

### Extension Points

Username or password incorrect

## Use Case Specification for <User profile>

### Brief Description

this use case allows the users to setup their information, which includes their full name, address, picture and phone number. This way each delivery will have the users information included within it.

### Actor

Primary actor: User

### Brief Description

1. The user selects to add or edit his profile
2. The system provides blank spaces where the user can add or edit his information
3. The user fills the required spaces and clicks the save button
4. The system saves and closes the page

### Alternative Flows of Events

None

### Special Requirements

None

### Preconditions

### the user should be already logged in into his account.

1. The user should have some saved photos on his phone.

### Postconditions

None

### Extension points

1. If the address and phone number were invalid the user should not be able to save his information.

## Use Case Specification for <Checkout>

### Brief Description

Checkout is a feature that goes through all the order payments.

### Actors

Primary: User, Credit Payment Service.

Secondary: Delivery Guy

### Basic Flow of Events

* + 1. After the user finishing his ordering and adding to card, the user triggers the checkout feature.
    2. The system displays the final order and the amount should be paid.
    3. The system waits for the user to confirm the order.
    4. The user confirms the order.
    5. The system asks the user for the payment method: Cash on delivery or Credit Card.
    6. If the user chooses the Cash on delivery method.
    7. The system notifies the user with the estimated time of delivery.
    8. The system assigns the Delivery Guy member to deliver the order with the user member details.
    9. The system waits for the Delivery Guy to confirm the payment of the user.
    10. If the user chooses the Credit Card method.
    11. The system asks the user to put the credit card information.
    12. The user puts all the information.
    13. The system sends all the info to the Credit Payment Service.
    14. The Credit Payment Service checks for the availability of the money amount.
    15. The Credit Payment Service send request successful to the system.
    16. The system notifies the user that the payment is successful.
    17. Repeat step g & h.

### Alternative Flows of Events

None

### Special Requirements

Credit Payment Service

### Preconditions

Valid Credit Card for Credit Card Payment method.

### Postconditions

System waiting for user interaction. Checkout order saved in the order history.

### Extension Points

Credit Card Payment Error:

1. Invalid Credit Card info.
2. More than 3 attempts of Invalid info.

## Use case for <Restaurant Profile>

### Brief Description

Restaurant Profile feature allows the restaurant user to manage his restaurant information and menu.

### Actors

Primary: Restaurant User.

Secondary: Customer User.

### Basic Flow of Events

* + 1. The restaurant user triggers the Restaurant Profile feature
    2. The system asks the restaurant user to put or change his Restaurant Profile information (Name, Location, Opening Dates and Hours)
    3. The restaurant user puts all the information.
    4. The system asks the restaurant user to manage his order menu.
    5. The restaurant user has the choice to manage his order by Categories, Items and Subitems.
    6. Repeat step d & c until the restaurant user is satisfied.
    7. The restaurant user has the choice to make his restaurant visible to the customer user.

### Alternative Flow of Events

None

### Special Requirements

None

### Preconditions

System must be configured with Email Address and Password.

### Postconditions

None

### Extension Points

None

## Use Case Specification for <SnapShot>

### Brief Description

SnapShot feature allows the restaurant user to take a short video of 15 seconds to advertise his restaurant.

### Actors

Primary: Restaurant User

Secondary: Customer User

### Basic Flow of Events

* + 1. The restaurant user triggers the SnapShot feature.
    2. The restaurant triggers the Add Video feature.
    3. The system opens phone camera and waits for the user to start recording a video.
    4. The restaurant user starts recording a video of maximum 15 seconds
    5. The system asks the restaurant user to confirm the video before posting.
    6. If the restaurant user decline, repeat step b to d until the restaurant user confirms.
    7. The system uploads the video through the restaurant profile.
    8. The system forbids the restaurant user to add another video to his profile.
    9. The system opens review section for the customer user to put their opinions

### Alternative Flows of Events

* + 1. The restaurant has the feature to remove the video
    2. Continue from step b in Basic Flow of Events.

### System Requirements

Mobile Camera in order to take the SnapShot.

### Preconditions

The restaurant user should fill the restaurant profile information before taking any SnapShot.

### Postconditions

A video of 15 seconds will be stored in the Restaurant Profile.

### Extension Points

* + 1. If the mobile phone is running on a low battery, the system doesn’t allow the restaurant user to take a SnapShot.
    2. If the mobile phone has no camera or a broken camera, the system doesn’t allow the restaurant user to take a SnapShot.

## Use Case Specification for <Manage Categories>

### Brief Description

This feature allows the customer user to sort out the restaurants by Its type, by average pricing, or by its offers.

### Actors

Primary: Customer User.

### Basic Flow of Events

* + 1. After or Before searching for any restaurant, the user has a list of to sort out the restaurants by.
    2. The user has the choice to choose how he wants to sort out the restaurants.
    3. Repeat step b until the use is satisfied.
    4. The system sorts out the restaurants by user preferences.

### Alternative Flow of Events

None

### System Requirements

None

### Preconditions

None

### Postconditions

None

### Extension Points

None

## Use Case Specification for <Sign Up>

### Brief Description

Sign Up feature allows the Restaurant User or Customer User to create an account.

### Actors

Primary: Restaurant User, Customer User.

### Basic Flow of Events

* + 1. The user triggers the Sign Up feature
    2. The system asks the users to specify if he is a Restaurant User or a Customer User.
    3. The user chooses what user he is.
    4. The system asks the user to enter the Username, Password and Date of Birth.
    5. The user enters all the information.
    6. The system opens a profile for the user.

### Alternative Flow of Events

None

### System Requirements

None

### Preconditions

None

### Postconditions

* + 1. Email Address and Password will be stored for specified user
    2. Whether the user is a Restaurant or a Customer, a profile will be opened to be managed.

### Extension Points

* + 1. If the user doesn’t enter a valid email address, the system asks the user again to enter a valid email address.
    2. If the user doesn’t enter a password that is more than 6 characters, the system asks the user again to enter a valid password
    3. If the user skips the Date of Birth, the system asks the user again to enter his Date of Birth.

## Use Case Specification for <Special Offers>

### Brief Description

Special Offers is a feature that allows the restaurant user to put his offers and special events.

### Actors

Primary: Restaurant

Secondary: Customer

### Basic Flow of Events

* + 1. The restaurant user triggers the Special Offers feature
    2. The system asks for the special time of the event or offers.
    3. The restaurant user enters the required time.
    4. The system asks if it is an event or an order.
    5. The restaurant user enters the option.
    6. The system asks the restaurant user for a description if it is an event and an add item if it is an order.
    7. The restaurant user adds all the required information
    8. The system stores the info in restaurant ordering to be available for customers or in the user profile for events.
    9. The system removes the item or the event automatically after the time has ended.

### Alternative Flow of Events

None

### System Requirements

None

### Preconditions

The Restaurant user should be logged in

### Postconditions

An Event will be stored in the Restaurant Profile, or the Item will be stored in the Restaurant menu.

### Extension Points

None

## 

# : Domain Analysis

## Domain Model Class Diagram

[Represent your domain model as a class diagram and write a brief explanation underneath it. If your domain model is large, you can split it into multiple diagrams to maintain clarity and readability. If large figures do not fit on a portrait page layout, you can use section breaks and place them on a landscape page layout to maintain readability.]

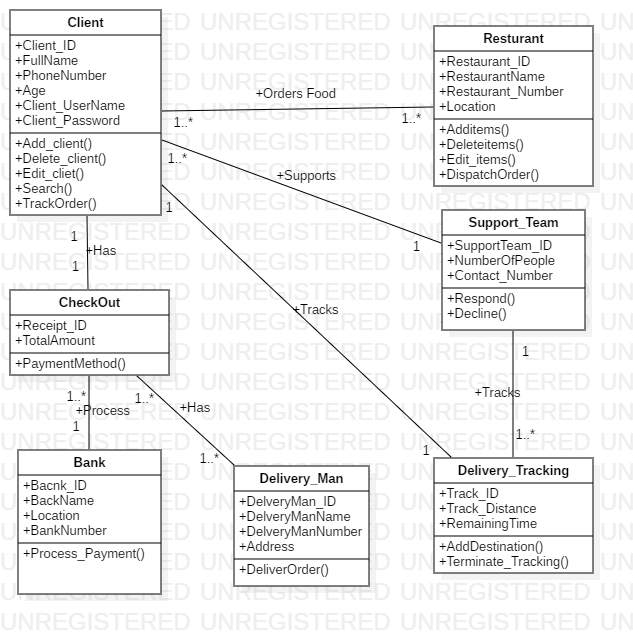


Figure 5.1: Ordering Food Class Diagram

In this class diagram we represented each of our actors and their relationships. A client can order food from one than restaurant at the same time and the restaurant should be able to receive orders from more than one client at the same time. A client can chat with a support team at any time, and the support team is supposed to answer to as many clients as possible at same time. Each client is also able to track his/her order at any requested time, to check how much time is still needed to deliver the order or to check if there is any problem with the order. The client checks out after ordering, each client is provided with one check out that includes the total amount and the ordered items. If the payment method selected was by using a credit card, the clients information is directly sent to the bank and the payment will be processed. If the payment method selected was cash on delivery, the delivery man will pick up the money from the client.

## Traceability Matrix

[Indicate in the following table which domain concepts are used in which use cases.]

Table 5.1: Domain Concept / Use Case Traceability Matrix

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Domain Concept / Use Case** | Login | Search | Order Food | Checkout | Support | Manage user Profile | Join Loyalty System | Manage Categories |
| Client | **🗸** | **🗸** | **🗸** | **🗸** | **🗸** | **🗸** | **🗸** | **🗸** |
| Restaurant | **🗸** |  |  |  | **🗸** |  |  | **🗸** |
| SupportTeam | **🗸** |  |  |  | **🗸** |  |  |  |
| Bank |  |  |  | **🗸** |  |  |  |  |
| Delivery Man |  |  |  | **🗸** |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Domain Concept / Use Case** | Delivery Tracking | Signup | Special Offer | Restaurant Profile | SnapShot |
| Client | **🗸** | **🗸** | **🗸** |  |  |
| Restaurant |  | **🗸** |  | **🗸** | **🗸** |
| SupportTeam | **🗸** |  |  |  |  |
| Bank |  |  |  |  |  |
| Delivery Man |  |  |  |  |  |

# : Interaction and Data Flow Diagrams

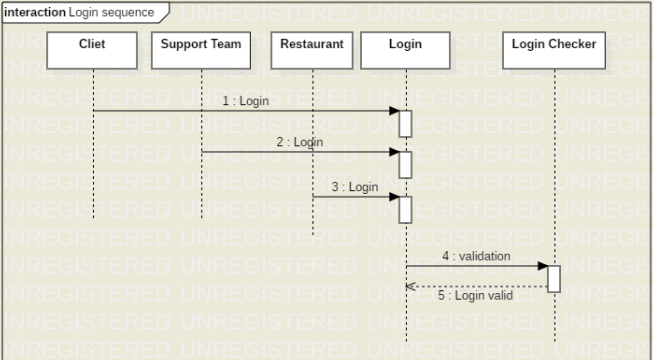


Figure 6.1: Basic Sequence Diagram for Login

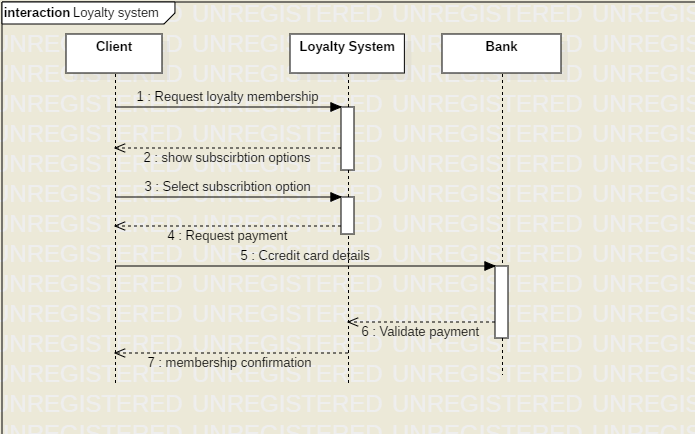


Figure 6.2: Basic Sequence Diagram for Loyalty system



Figure 6.3: Basic Sequence Diagram for Loyalty system



Figure 6.4: Basic Sequence Diagram for Checkout

## Data Flow Diagram

### Context Diagram (Level 0)

  
Figure 6.5: Data Flow Diagram level 0

### Level 1



Figure 6.5: Data Flow Diagram level 0

# : Collaboration and Individual Contribution

## Collaboration Approach

Deliverable – one:

For this deliverable we worked together on all the questions. We met for six times for each meeting for about 2 hours. The meetings were set at the NDU dorms cafeteria. We went together to the restaurant manager to ask our questions.

Deliverable – two:

For this deliverable we met two times for 5 hours where we divided the work equally between us. We refined our previous deliverable as well. We finished our second deliverable with lots of time to spare.

Deliverable – three:

For this deliverable we met one time for 4 hours where we divided or work equally. Two of us worked on on the sequence diagram and the other worked on the data flow diagram.

## Problems that Occurred

Deliverable – one:

No problems occurred, everything was well organized.

Deliverable – two:

We postponed two meetings due to personal problems. We also had to work extra time for this deliverable because it was more bulky that the previous one.

Deliverable – three:

We met on the last deliverable day due to exams.

## Individual Contribution Breakdown

Table 7.1: Individual Contribution

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Mohamed Ali** | **Toni Habib** | **Elio Mourad** |
| Task 1 Description | 33.3% | 33.3% | 33.3% |
| Task 2 Description | 33.3% | 33.3% | 33.3% |
| Task 3 Description | 33.3% | 33.3% | 33.3% |
| Task 4 Description | 25% | 25% | 25% |
| … |  |  |  |

# List of References

[Add a list of references, e.g. books, websites, etc.]

[You can use a reference management tool like Zotero to organize your references.]

Smith, M. (2001). Writing a successful paper. *The Trey Research Monthly*, *53*, 149-150.

# Appendix A: Presentation

[Convert your presentation slides into figures and include them in this appendix.]