**Software Requirements Specification**

**Version 1.0**

**May 16, 2021**

**Online Pizza Store**

**Omran Taljeh**

**Kheder Taleb**

**Anas Razouq**

**Yazn Alsahyone**

Table of Contents

**Table of Contents …………………………………………………………………………………………………..……..02**

**List of Figures ………………………………………………………………………………………………………………..02**

**1. Introduction …………………………………………………………………………………………..…………………03**

1.1. Purpose ……………………….…………………………………………………………………………………….………………………………03

1.2. Scope …………….. …………………………….………………………………………………………………………………………………..…03

1.3. Glossary …….……………………………………………………………………………………………………………………………………….03

1.4. Overview of Document ………………………………………………………………………………………………………………………04

**2. Overall Description …………………………………………………..……………………………………………….05**

2.1. System environment ………………………………………………………………………………………………………………………….05

2.2. Functional Requirements Specification ………………………………………………………………………………………………06

2.2.1. Customer use case ………………………………………………………………………………………………………………………………………….06

2.2.2. Administrator use cases ………………………………………………………………………………………………………………………………….07

2.2.3. Manager use cases …………………………………………………………………………………………………………………………………………08

2.2.4. Chef use case ………………………………………………………………………………………………………………………………………………….09

2.2.5. Deliverer use case …………………………………………………………………………………………………………………………………………..10

2.3. User Characteristics ……………………………………………………………………………………………………………………………10

2.4. Non-Functional Requirements ……………………………………………………………………………………………………………11

**3. Requirements Specification ……………………………………………………………………………………….12**

3.1. External Interface Requirements ……………………………………………………………………………………………………..…12

3.1.1. GUI …………………………………………………………………………………………………………………………………………………………………12

3.1.2. Software Interface ………………………………………………………………………………………………………………………………………….12

3.2. Functional Requirements ……………………………………………………………………………………………………………………12

3.2.1. Create Order …………………………………………………………………………………………………………………………………………………..12

3.2.2. Update Order …………………………………………………………………………………………………………………………………………………13

3.2.3. Update User ……………………………………………………………………………………………………………………………………………………13

3.2.4. Accept Order ………………………………………………………………………………………………………………………………………………….14

3.2.5. Review Order ………………………………………………………………………………………………………………………………………………….14

3.2.6. Prepare Order …………………………………………………………………………………………………………………………………………………15

3.2.7. Deliver Order ……………………………………………………………………………………………………………………………………………….…15

3.3. Detailed Non-Functional Requirements ……………………………………………………………………………………………..16

3.3.1. Logical Structure of the Data ………………………………………………………………………………………………………………………….16

3.3.2. Security …………………………………………………………………………………………………………………………………………………………..18

3.3.3. Reliability ……………………………………………………………………………………………………………………………………………………….18

3.3.4. Maintainability ……………………………………………………………………………………………………………………………………………….18

3.3.5. Reusability ………………………………………………………………………………………………………………………………………………………18

List of Figures

Figure 1 Use Case Diagram ……………………………………………………………………………………………………………………….05

Figure 2 Logical Structure of the System …………………………………………………………………………………………………..16

1

**Introduction**

*1.1. Purpose*

This document gives detailed requirements, functional and non-functional for a Pizza Store, an online pizza ordering system

The software developer will look at this document so he could understand the requirements of this store then implement it in the system, and that is the purpose of this document

*1.2. Scope*

Going all the way to the store and ordering pizza is getting old, so after the software developer implement the system, this could be an online store where people could order their delicious pizzas from anywhere

With the help of this system customer should able to order their favorite pizzas by click of a button

The information about store and every pizza is also provided

*1.3. Glossary*

|  |  |
| --- | --- |
| **Definition** | **Term** |
| The Person who enters the Online pizza store website and create the order, so it could be delivered to this person | Customer |
| The Owner or Owners of the online store, they have special accounts and full control over orders and employees, they don't have a specific job and their existence isn't essential for the flow of work, but they could perform the manager's job | Administrator |
| The Person who is most essential to the store, who get in contact with the system the most, has an account, accept orders, change state of orders and assign tasks to other employees | Manager |
| The Person who lead the kitchen and observe the work of preparing pizza up close, update the list of cooks in the kitchen and confirm preparing of orders | Chef |
| The Person who is his only job is to deliver orders to customers, then report the result of this delivery, successful or unsuccessful | Deliverer |
| The Person who prepare the pizzas, doesn't get in contact with the system, only prepare pizza then report to the chef | Cook |
| It could be the manager or the chef | User |
| The list of selected pizzas by the customer, created by customer, accepted by manager, prepared by chef, then delivered by the deliverer | Order |
| The food or snack that this store provide for customers online without visiting the store itself, there are many types of it where the customer could choose from when creating an order | Pizza |

*1.4. Overview of Document*

This system provide easy solution to the customer to order the item he/she wants directly through internet without visiting the store

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product, it describes the informal requirements

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product

**2**

**Overall Description**

*2.1. System Environment*

Administrator

Customer

Manager

Chef

**Figure 1 – Use Case Diagram**

Deliverer

The online pizza store has three active actors and one cooperating system

The Customer, the Database Administrator and the Delivery

*2.2. Functional Requirements Specification*

*This section outlines the use cases for each of the active actors separately*

*The Customer does not need an account and has one use case*

*The Administrator need an account and has two use cases*

*The Manager who need an account and got two use cases*

*The Chef of the kitchen who need an account and got one use case*

*The Deliverer does not need an account and has one use case*

2.2.1. Customer Use Case:

**Use Case: Create Order**

**Brief Description**

The Customer access the online store website to order a pizza, and to do that he/she needs to create an order then confirm it

**Initial Step-By-Step Description**

Before this use case can be initiated, The Customer has already accessed the Online Pizza Store

1. The Customer select 'Create New Order'

2. The System opens a empty form for the Customer

3. The Customer should enter his/hers name, phone number, address and delivery data, the delivery date can't be less than an hour, and it could be up to two weeks ahead

4. The System check the validation of the inputs by customer

5. The System opens the pizza menu

6. The Customer selects his/hers pizza from the list, more than one pizza could be selected

7. The System counts the price of all selected pizzas and show it to the customer

8. The System provide the customer with a list of all available ingredients in the store

9. The Customer could add or delete ingredients to a specific pizza

9.1. The Customer add ingredients by selecting it from the provided list of ingredients

9.2. The Customer delete ingredients from the options of the selected pizza

10. The System provide the customer with a photo of the selected pizza, and update the photo depending on the added or deleted ingredients

11. The Customer confirm the order after selection

12. The System create the order, and send it to the list of orders

**Xref:** Section 3.2.1, Create Order

2.2.2. Administrator Use Cases:

**Use Case: Update Order**

**Brief Description**

The Administrator has full control over orders, so he/she could add, edit and delete them

**Initial Step-By-Step Description**

Before this use case can be initiated, The Administrator has already accessed his/hers account and gain Administrators features

1. The System show the Administrator a list containing every order and it's state

2. The System send a notification to the Administrator every time a customer send an order, along with the details of every order

3. The Administrator could add, edit or delete any order:

3.1 The Administrator add an order and select pizza(s) without entering any inputs

3.2. The Administrator edit any order by selecting it from the list then make the changes

3.3. The Administrator delete any order by selecting it from the list then deleting it, and could enter order and delete any pizza

4. The Administrator must be able to view an order log containing all details of previous orders

**Xref:** Section 3.2.2, Update Order

**Use Case: Update User**

**Brief Description**

The Administrator has full control over users, so he/she could add, edit or delete them

**Initial Step-By-Step Description**

Before this use case can be initiated, The Administrator has already accessed his/hers account and gain Administrators features

1. The System provide the administrator with a list containing all employees in the store, names and state of work ('available' or 'unavailable') by updating the list

2. The Administrator could assign a specific task to any employee

3. The Administrator could add a new employee account, or edit/delete any existing one:

3.1. The Administrator add a new employee by creating a new account with the information of this new employee, then gives him/her the email and password

3.2. The Administrator edit any employee by editing his/her account, in case of the employee asked to, or by the decision of the administrator

3.3. The Administrator delete any employee by deleting his/hers account

**Xref:** Section 3.2.3, Update User

*Next to the Administrator use cases, the administrator could perform anything the manager could, like accepting orders and reviewing them*

2.2.3. Manager Use Cases:

**Use Case: Accept Order**

**Brief Description**

Mostly the Employees who get in contact with the system are the managers, who accept orders that have been sent by customers

**Initial Step-By-Step Description**

Before this use case can be initiated, The Manager has already accessed his/hers account and gain Managers features

1. The System send a notification to the Manager every time a customer send an order, along with the details of every order

2. The System provide the Manager with a list of available cooks who is ready to prepare orders, and list of available deliverers who is ready to deliver orders

3. The Manager select an order marked as 'waiting' and assign the task of preparing it to the cooks, then mark this order as 'preparing'

3. The System gives a second notification to the Manager if the state of specific order remain 'waiting' for more than half an hour

4. The Manager search for 'prepared' orders and assign the task of delivering them to available deliverers, then mark these orders as 'delivering' and the deliverer as 'unavailable'

**Xref:** Section 3.2.4, Accept Order

**Use Case: Review Order**

**Brief Description**

The Manager need to review old orders to confirm that it has been delivered

**Initial Step-By-Step Description**

Before this use case can be initiated, The Manager has already accessed his/hers account and gain Managers features

1. The Manager mark the state of an order as 'delivered' or 'failed to deliver' after the return of the deliverer of that order, and mark the deliverer as 'available'

2. The Manager check the orders that marked as 'failed to deliver', to find the reason of failure and trying to solve it:

2.1 The Manager could confirm if the deliverer destination and timing matches the right inputs of the customer

2.2 The Manager could contact the Customer in case of false entered inputs

2.3 The Manager could contact the Chef in case of wrong order

3. The Manager assign the task of delivering the order after solving the problem

**Xref:** Section 3.2.5, Review Order

2.2.4. Chef Use Case:

**Use Case: Prepare Order**

**Brief Description**

The cooks in the kitchens work to prepare orders that assigned to them, the one who lead the cooks in the kitchens is the chef

**Initial Step-By-Step Description**

Before this use case can be initiated, The Chef has already accessed his/hers account

1. The System provide a list of names of all cooks in the kitchens for the Chef

2. The Chef update the list of cooks after receiving a new order, or finishing preparing one, by changing state between 'available' and 'unavailable'

3. The System send a new order from the Manager to the Chef whenever there is available cooks ready to work

4. The Chef mark an order as 'prepared' after done preparing it

**Xref:** Section 3.2.6, Prepare Order

2.2.5. Deliverer Use Case:

**Use Case: Deliver Order**

**Brief Description**

The Deliverer is the Employee who deliver orders to customer addresses after being prepared

**Initial Step-By-Step Description**

Before this use case can be initiated, The Deliverer state should be 'available'

1. The Deliverer receive the task of delivering an order from the Manager

2. The Deliverer pick up the order and deliver it to the customer's address

3. The Deliverer inform the manager about the state of order after delivering it or failing to deliver it

**Xref:** Section 3.2.7, Deliver Order

*2.3. User Characteristics*

The Customer is expected to be Internet literate and be able to use a search engine and web browser

The Chef is expected to be Internet literate and to be able to use email with attachments The Administrator and Manager are expected to be Windows literate and to be able to use button, menus, and similar tools

The Deliverer just need to be good with delivering orders  
The main page of the Online Pizza Store Website will a navbar on the top with few buttons, the rest of the page are just pictures of pizza products from the store, and commercials for the store, the navbar buttons are:

1. Log In button: this button for administrators, managers, chef and deliverers to enter their accounts

2. Create Order: this button for customers to create a new order

*2.4. Non-Functional Requirements*

As an operational requirement, the system will run as a database with a website as user interface

As performance requirement the system must be accessible 24 hours a day, 7 days a week

Due to the nature of the system as an ordering website, the system must have a low response time, preferably shorter than second, with a maximum of five seconds

The exception is viewing order logs which could have a higher response time (of seconds) as the log increases in size over time

Due to the low complexity of the system, no problems with response time are expected Customers who visit the website to order will get a session ID for their visit, which is used to identify them while using the system

For every action they take, a timestamp is stored

From time to time a service on the server will scan session ID’s and timestamps  
Session ID’s which have not been active for more than three hours will be deleted along with the corresponding ordering information

**3**

**Requirements Specification**

*3.1. External Interface Requirements*

3.1.1. GUI:  
The user interface must be highly interactive so that all users may be able to operate the system as easily and fluently, GUI would be used for creating home page with navigation bar and other pages for respective categories

3.1.2. Software Interface:

The software interfaces required for this system are:  
1. Internet connection

2. The PHP and MYSQL must be installed  
3. Any browser

*3.2. Functional Requirements*

3.2.1. Create Order

|  |  |
| --- | --- |
| Create Order | Use Case Name |
| Section 2.2.1, Create Order | Xref |
| The Customer accesses the Online Pizza Store Website | Trigger |
| The Web is displayed with a navbar and buttons | Precondition |
| 1. The Customer select 'Create New Order'  2. The System opens a empty form for the Customer  3. The Customer should enter his/hers information  4. The System check the validation of the inputs by customer  5. The System opens the pizza menu  6. The Customer selects his/hers pizza from the list, more than one pizza could be selected  7. The System counts the price of all selected pizzas and show it to the customer  8. The System provide the customer with a list of all available ingredients in the store  9. The Customer could add or delete ingredients to a specific pizza  9.1. The Customer add ingredients by selecting it from the provided list of ingredients  9.2. The Customer delete ingredients from the options of the selected pizza  10. The System provide the customer with a photo of the selected pizza, and update the photo depending on the added or deleted ingredients  11. The Customer confirm the order after selection  12. The System create the order, and send it to the list of orders | Basic Path |
| In Step 4, if the inputs of customer isn't valid, the system show an alert message and ask the customer to enter his/hers inputs correctly | Alternative Paths |
| The Order is created and get added to the orders list | Postcondition |
| The Customer could cancel the create order operation any time and return to the main page | Exception Paths |
| The inputs/information of a customer is name, phone number, address and delivery date, the delivery date can't be less than an hour, and it could be up to two weeks ahead | Other |

3.2.2. Update Order

|  |  |
| --- | --- |
| Update Order | Use Case Name |
| Section 2.2.2, Update Order | Xref |
| The Administrator accesses his/hers account on the online store website | Trigger |
| The Administrator has entered his/hers account and gained administrators features | Precondition |
| 1. The System show the Administrator a list containing every order and it's state  2. The System send a notification to the Administrator every time a customer send an order, along with the details of every order  3. The Administrator could add, edit or delete any order:  3.1 The Administrator add an order and select pizza(s) without entering any inputs  3.2. The Administrator edit any order by selecting it from the list then make the changes  3.3. The Administrator delete any order by selecting it from the list then deleting it, and could enter order and delete any pizza  4. The Administrator must be able to view an order log containing all details of previous orders | Basic Path |
| In Step 2, The Administrator could turn off the notifications | Alternative Paths |
| The Order get updated on the orders list and the changes become visible to all managers | Postcondition |
| The Administrator could cancel his/hers account anytime without updating anything | Exception Paths |
| The state of any order could be 'waiting', 'preparing', 'prepared', 'delivering', 'delivered' or 'failed to deliver' | Other |

3.2.3. Update User

|  |  |
| --- | --- |
| Update User | Use Case Name |
| Section 2.2.3, Update User | Xref |
| The Administrator accesses his/hers account on the online store website | Trigger |
| The Administrator has entered his/hers account and gained administrators features | Precondition |
| 1. The System provide the administrator with a list containing all employees in the store, names and state of work ('available' or 'unavailable') by updating the list  2. The Administrator could assign a specific task to any employee  3. The Administrator could add a new employee account, edit/delete any existing one  3.1. The Administrator add a new employee by creating a new account with the information of this new employee, then gives him/her the email and password  3.2. The Administrator edit any employee by editing his/her account, in case of the employee asked to, or by the decision of the administrator  3.3. The Administrator delete any employee by deleting his/hers account | Basic Path |
| In Step 2, it is not the administrator job to assign tasks, it is the manager job, but the administrator could do it anyway | Alternative Paths |
| The edited information on employees accounts get updated on the employees list | Postcondition |
| The Administrator could cancel his/hers account anytime without updating anything | Exception Paths |
| The employees accounts that could be edited by administrator are the managers and the chef | Other |

3.2.4. Accept Order

|  |  |
| --- | --- |
| Accept Order | Use Case Name |
| Section 2.2.4, Accept Order | Xref |
| The Manager accesses his/hers account on the online store website | Trigger |
| The Manager has entered his/hers account and gained managers features | Precondition |
| 1. The System send a notification to the Manager every time a customer send an order, along with the details of every order  2. The System provide the Manager with a list of available cooks who is ready to prepare orders, and list of available deliverers who is ready to deliver orders  3. The Manager select an order marked as 'waiting' and assign the task of preparing it to the cooks, then mark this order as 'preparing'  3. The System gives a second notification to the Manager if the state of specific order remain 'waiting' for more than half an hour  4. The Manager search for 'prepared' orders and assign the task of delivering them to available deliverers, then mark these orders as 'delivering' and the deliverer as 'unavailable' | Basic Path |
| None | Alternative Paths |
| A specific order get accepted then prepared, and after that get sent to deliver | Postcondition |
| The Manager could cancel his/hers account anytime and handle the job to another manager | Exception Paths |
| The Manager is the one responsible for updating states all the time, so every employee could do his/hers job | Other |

3.2.5. Review Order

|  |  |
| --- | --- |
| Review Order | Use Case Name |
| Section 2.2.5, Review Order | Xref |
| The Manager accesses his/hers account on the online store website | Trigger |
| The Manager has entered his/hers account and gained managers features | Precondition |
| 1. The Manager mark the state of an order as 'delivered' or 'failed to deliver' after the return of the deliverer of that order, and mark the deliverer as 'available'  2. The Manager check the orders that marked as 'failed to deliver', to find the reason of failure and trying to solve it:  2.1 The Manager could confirm if the deliverer destination and timing matches the right inputs of the customer  2.2 The Manager could contact the Customer in case of false entered inputs  2.3 The Manager could contact the Chef in case of wrong order  3. The Manager assign the task of delivering the order after solving the problem | Basic Path |
| In Step 2.2, in case the customer had entered false inputs, the manager rewrite this inputs to match the customer's information  In Step 2.3, in case of wrong order (wrong pizza or ingredients) the manager ask the chef to prepare the order again | Alternative Paths |
| Confirm that all orders are delivered to their customers | Postcondition |
| The Manager could cancel his/hers account anytime and handle the job to another manager | Exception Paths |
| The Manager is the one responsible for reviewing orders, otherwise some customers may never receive their orders | Other |

3.2.6. Prepare Order

|  |  |
| --- | --- |
| Prepare Order | Use Case Name |
| Section 2.2.6, Prepare Order | Xref |
| The Manager accesses his/hers account on the online store website | Trigger |
| The Manager has entered his/hers account and gained managers features | Precondition |
| 1. The System provide a list of names of all cooks in the kitchens for the Chef  2. The Chef update the list of cooks after receiving a new order, or finishing preparing one, by changing state between 'available' and 'unavailable'  3. The System send a new order from the Manager to the Chef whenever there is available cooks ready to work  4. The Chef mark an order as 'prepared' after done preparing it | Basic Path |
| None | Alternative Paths |
| Prepare all orders in the store, making them ready to picked up | Postcondition |
| The Chef could handle the job to his/hers right-hand assistance | Exception Paths |
| The Chef is the one who lead every cook in the kitchen, and the only one who get in contact with the system | Other |

3.2.7. Deliver Order

|  |  |
| --- | --- |
| Deliver Order | Use Case Name |
| Section 2.2.7, Deliver Order | Xref |
| An order get prepared and ready to be delivered | Trigger |
| None | Precondition |
| 1. The Deliverer receive the task of delivering an order from the Manager  2. The Deliverer pick up the order and deliver it to the customer's address  3. The Deliverer inform the manager about the state of order after delivering it or failing to deliver it | Basic Path |
| None | Alternative Paths |
| Delivering all order, or failing to deliver them | Postcondition |
| There are multiple deliverers to handle the delivering | Exception Paths |
| None | Other |

Administrator

*3.3. Detailed Non-Functional Requirements*

Update

3.3.1. Logical Structure of the Data:

Customer

Manager

Chef

Create

Accept

Order

Include

Pizza

Prepare

Observer

Deliver

**Cook**

**Deliverer**

**Figure 2 – Logical Structure of the System**

The data descriptions of each of these data entities is as follows:

Customer Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of the customer | Real name or an alias |
| Phone Number | Integer | Phone Number of the customer |  |
| Address | Text | Address of the customer |  |
| Delivery date | Date | The Delivery date of the order |  |
| Order id | Integer | Order entity | The created order |

Pizza Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| ID | Integer | ID number of the pizza | Every pizza has an id |
| Name | Text | Name of the pizza |  |
| Size | Integer | Size of the pizza |  |
| Price | Double | Price of the pizza |  |
| Order id | Integer | Order entity | Every pizza belong to an order |
| Ingredients | Text | A list of all ingredients of the pizza | May be several |

Order Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| ID | Integer | ID number of the order | Every order has an id |
| State | Text | The state of the order | 'waiting', 'preparing' ... |
| Customer info | Pointer | Customer entity |  |
| Pizza id | Pointer | Pizza entity | May be several |
| Manager info | Pointer | Manager entity | Manager who accepted the order |
| Chef info | Pointer | Chef entity | Chef who prepare the order |
| Deliverer info | Pointer | Deliverer entity | Deliverer who deliver the order |
| Price | Double | Price of all selected pizzas |  |

Administrator Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Full Name | Text | Full name of the administrator | First name and last name |
| Email | Text | Email address of the administrator |  |
| Password | Integer | Password of the administrator |  |
| Order id | Pointer | Order entity | May be several |
| Manager id | Pointer | Manager entity | May be several |
| Chef id | Pointer | Chef entity | May be several |

Manager Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Full Name | Text | Full name of the manager | First name and last name |
| Email | Text | Email address of the manager |  |
| Password | Integer | Password of the manager |  |
| Address | Text | Address of the manager |  |
| Phone Number | Integer | Phone number of the manager | May be several |
| Order id | Pointer | Order entity | May be several |
| Administrator id | Pointer | Administrator | Administrator with full control |

Chef Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Full Name | Text | Full name of the chef | First name and last name |
| Email | Text | Email address of the chef |  |
| Password | Integer | Password of the chef |  |
| Address | Text | Address of the chef |  |
| Phone Number | Integer | Phone number of the chef | May be several |
| Order id | Pointer | Order entity | May be several |
| Administrator id | Pointer | Administrator entity | Administrator with full control |

Deliverer Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Full Name | Text | Full name of the deliverer | First name and last name |
| State | Text | State of work | 'available' or 'unavailable' |
| Address | Text | Address of the deliverer |  |
| Phone Number | Integer | Phone number of the deliverer | May be several |
| Order id | Pointer | Order entity |  |

Cook Data Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Full Name | Text | Full name of the cook | First name and last name |
| State | Text | State of work | 'available' or 'unavailable' |
| Address | Text | Address of the cook |  |
| Phone Number | Integer | Phone number of the cook | May be several |
| Chef id | Pointer | Chef entity | Chef who observe the cook |

3.3.2. Security:

The server-side security is needed to protect the system from hackers

3.3.3. Reliability:The system should be highly reliable and it should generate all the updated information in  
correct order

3.3.4. Maintainability:The system should be maintainable in such a manner that if any new requirement occurs then it should be easily incorporated in an individual module

3.3.5. Reusability:The system would be usable as long as people want to use it