Design Portfolio

**Robotics Systems**

**UFMFJA-30-2**

**2019/2020**

**Dennis Thomas ID:18023925**

**Ntani Kokkinos ID:18027660**

**Nikoleta Ivanova ID:18006402**

Contents

[1. Percentage of Contribution 3](#_Toc36819895)

[2. Introduction 3](#_Toc36819896)

[3. Full Specification of the Application 3](#_Toc36819897)

[4. Assumptions 3](#_Toc36819898)

[5. Functional Requirements 3](#_Toc36819899)

[6. Non-Functional Requirements 4](#_Toc36819900)

[7. Use Case Model 4](#_Toc36819901)

[7.1 Use Case Diagram 4](#_Toc36819902)

[7.2 Use Case Documentation 5](#_Toc36819903)

[7.2.1 Call Waiter 5](#_Toc36819904)

[7.2.2 Start Order 5](#_Toc36819905)

[7.2.3 Edit Item 6](#_Toc36819906)

[7.2.4 Remove Item 6](#_Toc36819907)

[7.2.5 Add Item 6](#_Toc36819908)

[7.2.6 Close Order 7](#_Toc36819909)

[7.2.7 Request Menu 7](#_Toc36819910)

[7.2.8 Request Bill 7](#_Toc36819911)

[7.2.9 Make Payment 8](#_Toc36819912)

[7.2.10 Deliver Order 9](#_Toc36819913)

[7.2.11 Check Status 9](#_Toc36819914)

[7.2.12 Request Taxi 9](#_Toc36819915)

[7.2.13 Directions to WC 10](#_Toc36819916)

[7.2.14 Music Request 10](#_Toc36819917)

[7.2.15 Call Manager 10](#_Toc36819918)

[7.2.16 Complaint 11](#_Toc36819919)

[8. Class Diagram 12](#_Toc36819920)

[9. Activity Diagrams 13](#_Toc36819921)

[10. Sequence Diagrams 16](#_Toc36819922)

# 1. Percentage of Contribution

For this assignment all members of the team contributed equally.

Dennis Thomas – 33.3%

Ntani Kokkinos – 33.3%

Nikoleta Ivanova – 33.3%

# 2. Introduction

The aim of this group project is to develop a robot software design. The system is going to be a robot waiter operating in a bar. The objective is to create a design portfolio containing a use case model, requirements documentation, class diagram, activity diagrams and the sequence diagrams. The requirements document includes a full specification of the application including functional requirements, non-functional requirements, assumptions and supporting documentation. The class diagram is a representation of the interacting entities required to build the system. Individual diagrams (activity diagrams and sequence diagrams) are included with everyone’s name and student id number.

# 3. Full Specification of the Application

The system is in the form of a robot that is operating as a waiter in a bar. The robot is moving on 4 wheels and has a humanoid upper body. The system is capable of (but not limited to) taking orders, handing out menus, delivering orders, arranging payments and completing a variety of special requests. Special requests include ordering a taxi, taking music requests, etc.

# 4. Assumptions

Global assumptions made about the environment in which the system operates.

* The bar floor is levelled.
* Security checks ID of customers, prior to entering the bar, to make sure customers are above age.
* The floor can sustain the weight of the robot
* The bar operates only with card payments
* The spacing between the tables is large enough for the system to move between tables
* The bar staff have a basic knowledge of the maintenance for the system

# 5. Functional Requirements

Global functional requirements of the system.

* The system is battery powered and has a charging station.
* The system moves on a base with wheels and has a humanoid upper body.
* The system cannot climb stairs; thus, any regions with stairs are inaccessible by the robot.
* The system can detect and avoid obstacles, whether they are stationary or moving
* The system is as tall as an average adult.
* The system cannot exit the bar.
* The system cannot enter behind the bar nor the kitchen area.
* The system is aware of the floor plan of the bar and all facilities within it.
* The user must call the system before attempting to request any other service

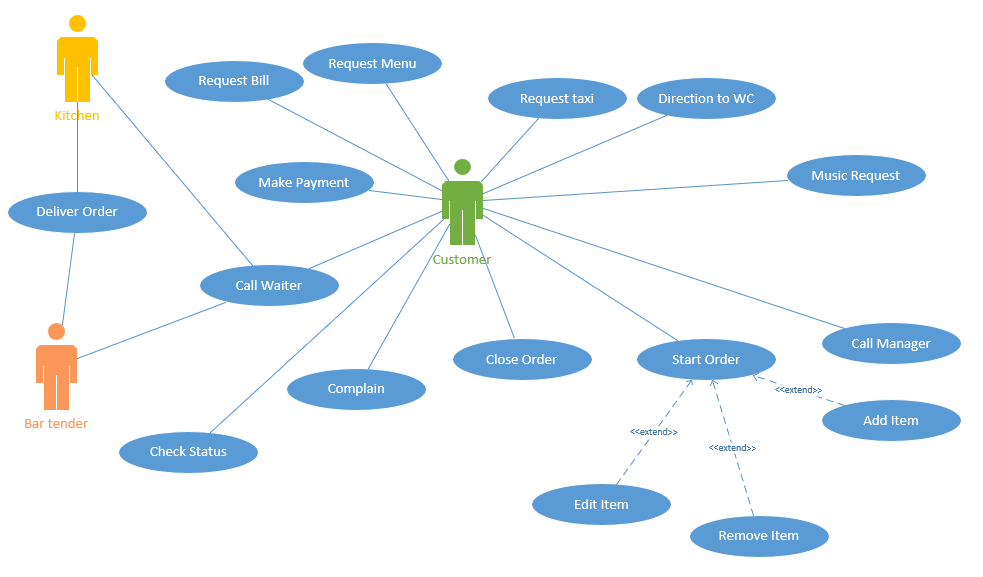
# 6. Non-Functional Requirements

Global non-functional requirements of the system.

* <Speed> The system moves at average speed of 1.5 m/s
* <Capacity> The system moves on 4 wheels
* <Performance> The system has 2 humanoid arms with grippers as end-effectors
* <Capacity> Obstacles are objects larger than 4cmx4cmx4cm
* <Reliability> The system cannot avoid obstacles moving towards it faster than 2 m/s or 3m/s in any other direction
* <Capacity> The system is 1.7m tall
* <Performance> The system requires a maximum time of 20 seconds to reach any part of the bar.
* <Security> The system does not collect any personal data about the customer
* <Security> The system prioritises the user's health and safety over anything else.
* <Security> The system has ultrasound sensors to detect obstacles
* <Security> The system has bump sensors to detect collisions
* <Security> The system stops moving when it collides with an obstacle

# 7. Use Case Model

## 7.1 Use Case Diagram



## 7.2 Use Case Documentation

### 7.2.1 Call Waiter

|  |  |
| --- | --- |
| Use Case | Call Waiter  ID - 01  History modified 14/02/2020 |
| Description | The system is alerted by the user or bar tender and will go to them. |
| Actors | Customer  Bar tender  Kitchen |
| Assumptions | * All tables have a call button. |
| Steps | Customer:   1. The system goes to the user’s table.   Bar tender:   1. System goes to bar tender’s location.   Kitchen:   1. System goes to Kitchen’s location. |
| Variations | * The user double taps the button to cancel the call |
| Non-Functional | <Priority> Calls issued by bar tender and kitchen hold higher priority than calls issued by customer  <Performance> Calls are executed in a chronological order  <Reliability> The system knows the location of all tables in the bar and the location of the bartender |

### 7.2.2 Start Order

|  |  |
| --- | --- |
| Use Case | Start Order  ID - 02  History modified 14/02/2020 |
| Description | The user starts the order. |
| Actors | Customer |
| Assumptions | * The user already knowns what they want to order. * The user has settled down. |
| Steps | 1.The system adds a new order to the order list and gives it an ID   2.The system awaits further instructions from the user. |
| Variations | * The user has regretted starting the order, so the system removes the order from the order list. * The user has decided to add, remove or edit an item (see use cases ID 2.1,2.2,2.3) |
| Non-Functional | <Speed>: The system should arrive to the customer in a timely manner.  <Security> The system shall not disclose any information about the order, except to the bar staff.  <Performance> The system has access to the order list.  <Capacity>The system can store up to 100 orders in the order list. |

### 7.2.3 Edit Item

|  |  |
| --- | --- |
| Use Case | Edit Item  ID – 02.1  History modified 14/02/2020 |
| Description | The user requests to customise a specific item on the menu. |
| Preconditions | <1> <Start Order> <2> <Edit Item> |
| Actors | Customer |
| Assumptions | * The user knowns what variations can be made to the order. |
| Steps | 1. The system edits the item. 2. System awaits further instructions from the user. |
| Variations | * The user regrets editing the item, so the system cancels the edit. * User’s customisation is not possible, so the system notifies the user |
| Non-Functional | <Performance> The system has access to the order list.  <Performance> The system knows which edits are available for each item. |

### 7.2.4 Remove Item

|  |  |
| --- | --- |
| Use Case | Remove Item  ID – 02.2  History modified 14/02/2020 |
| Description | The user requests to remove an item from their order. |
| Preconditions | <1> <Start Order> <2> < Remove Item > |
| Actors | Customer |
| Assumptions |  |
| Steps | 1.The system removes the item from the order.  2. System awaits further instructions from user. |
| Variations | * The item does not exist on the ongoing order, so the system notifies the user. |
| Non-Functional | <Performance> The system has access to the order list.  <Fault tolerance> The system removes the correct item. |

### 7.2.5 Add Item

|  |  |
| --- | --- |
| Use Case | Add Item  ID – 02.3  History modified 14/02/2020 |
| Description | The user adds an item to the order. |
| Preconditions | <1> <Start Order> <2> < Add Item> |
| Actors | Customer |
| Steps | 1.The system adds the item to the order  2. System awaits further instructions from user |
| Variations | * The item does not exist on the menu, so the system notifies the user. |
| Non-Functional | <Performance> The system has access to the order list.  <Performance> The system knows the bars menu. |

### 7.2.6 Close Order

|  |  |
| --- | --- |
| Use Case | Close Order  ID - 03  History modified 14/02/2020 |
| Description | User is satisfied with their choice of items and wants to complete the order. |
| Actors | Customer |
| Assumptions | * User has an order to complete * The user knowns that he cannot change any of the items in the order after completing it |
| Steps | 1. System waits for customer to close order 2. System marks order as closed |
| Variations | 1. The user needs more items 2. The system starts a new order and amends the orders. |
| Non-Functional | <Performance Mean> The system must wait up to five minutes for the user to close order. The system closes the order on its own if no action is taken.  <Performance> The system has access to the order database to be able to amend items |

### 7.2.7 Request Menu

|  |  |
| --- | --- |
| Use Case | Request Menu  ID - 04  History modified 14/02/2020 |
| Description | The user has sat down and is ready to browse the menu, by requesting it from the system. |
| Actors | Customer |
| Assumptions | * The user does not need to request the menu to make an order. * Allergens are included in the menu; thus, the system has no responsibility if the customer orders food they might be allergic to. |
| Steps | 1. The system picks out as many menus are required from its pouch and gives them to the users |
| Variations | * The user asks for the vegetarian, kids, gluten free or vegan menu instead of the default menu. * When the system run out of menus, it refills the pouch from the bar. |
| Non-Functional | <Capacity> The system cannot take any order that are not on the menu  <Capacity> The system has a pouch that fits 20 menus. |

### 7.2.8 Request Bill

|  |  |
| --- | --- |
| Use Case | Request Bill  ID - 05  History modified 14/02/2020 |
| Description | The user has finished eating or drinking and is ready to request the bill to pay the order. |
| Actors | Customer |
| Assumptions | * The user has something to pay for, otherwise, the user cannot use this functionality of the system. * The user may make several orders. The bill has to request only when the user finished with all orders and wants to leave the bar. |
| Steps | The system prints the bill and gives it to the user. |
| Variations | * The users might request to split the bill; Thus, the system prints the amount of bills required. |
| Non-Functional | <Security> The system does not disclose the users bill to any other user.  <Reliability>The system gives the user the correct bill |

### 7.2.9 Make Payment

|  |  |
| --- | --- |
| Use Case | Make Payment  ID - 06  History modified 14/02/2020 |
| Description | The system accepts payment issued by the user for their placed orders. |
| Actors | Customer |
| Assumptions | * The user has something to pay for, otherwise, the user cannot use this functionality of the system. * The user is paying in the correct currency. |
| Steps | 1. The system displays the total amount on the card reader and waits for the user to make the payment. 2. The system checks if the payment was successful. 3. The system asks the user if they want a receipt. 4. The system prints out receipt and gives it to the user. |
| Variations | * User does not want the receipt, so the system completes the payment. * If the payment is successful, the system removes the order from the order list. * If the pay payment is unsuccessful the system asks the user to try again (up to 4 attempts) * If all attempts have been used, the system calls the manager over. |
| Non-Functional | <Performance> The system is equipped with card reader.  <Fault Tolerance> The system ensures the payment method is working (card reader working)  <Performance> The system can deal with card payments, apple pay, google pay.  <Security> The system does not share the payment details to anyone.  <Performance> The system has access to the order list. |

### 7.2.10 Deliver Order

|  |  |
| --- | --- |
| Use Case | Deliver Order  ID - 07  History modified 14/02/2020 |
| Description | The bar tender gives the order to the system and successfully delivers it to the user |
| Actors | Bar tender  Kitchen |
| Assumptions | * The bar tender or kitchen place the ordered items on a tray |
| Steps | 1. The system picks up the tray 2. The system delivers the tray to the users’ table |
| Variations |  |
| Non-Functional | <Reliability> The system can deliver the tray without damaging or spilling it.  <Performance> The system can carry one tray at a time.  <Performance Mean> The system delivers the tray within one minute after collection.  <Reliability> System knows who the tray should be delivered to and the location of the user |

### 7.2.11 Check Status

|  |  |
| --- | --- |
| Use Case | Check Status  ID - 08  History modified 14/02/2020 |
| Description | The user asks the system to check the status of the order. |
| Actors | Customer |
| Assumptions | * The bar has an external system that tracks the progress of the order. |
| Steps | 1. The system checks the external system for the order progress 2. The system notifies user of the order progress |
| Variations | <Reliability> The information passed to the user is accurate.  <Tolerance> The system reports status as soon as possible to prevent user’s dissatisfaction.  <Performance> The system can access the order ID. |
| Non-Functional |  |

### 7.2.12 Request Taxi

|  |  |
| --- | --- |
| Use Case | Request Taxi  ID - 09  History modified 14/02/2020 |
| Description | The user wants to leave the bar and requests a taxi from the system. |
| Actors | Customer |
| Assumptions | * The bar has a list of taxi services that they work with. |
| Steps | 1.The system asks the user for the destination.  2.The system notifies the taxi services to send a taxi.  3.The system adds the amount to the bill  4.The system pays the taxi services. |
| Variations | * The user specifies the number of taxies needed and the size of the taxi. |

### 7.2.13 Directions to WC

|  |  |
| --- | --- |
| Use Case | Directions to WC  ID - 10  History modified 14/02/2020 |
| Description | The user needs to use the bathroom; thus, asks the system for directions. |
| Actors | Customer |
| Assumptions | * The bathrooms are unisex |
| Steps | 1. The system directs the user verbally. |
| Variations | * The user disabled; thus, the system escorts them to the disabled bathroom. |
| Non-Functional | <Fault Tolerance> The system knowns whether the bathrooms are out of order. |

### 7.2.14 Music Request

|  |  |
| --- | --- |
| Use Case | Music Request  ID - 11  History modified 14/02/2020 |
| Description | The user asks the system to change a song or start playing some music if there is no music playing. |
| Actors | Customer |
| Assumptions | * The bar has a music system. The music system only supports Spotify, SoundCloud and YouTube. The bar has the rights to play music. |
| Steps | 1. The system asks the user for their music choice 2. The system adds the choice to the playlist. |
| Variations | * If there are any other music requests, the system adds the users request to a queue and notifies him about it. |
| Non-Functional | <Performance> The system communicates with the bars music system to change the music remotely |

### 7.2.15 Call Manager

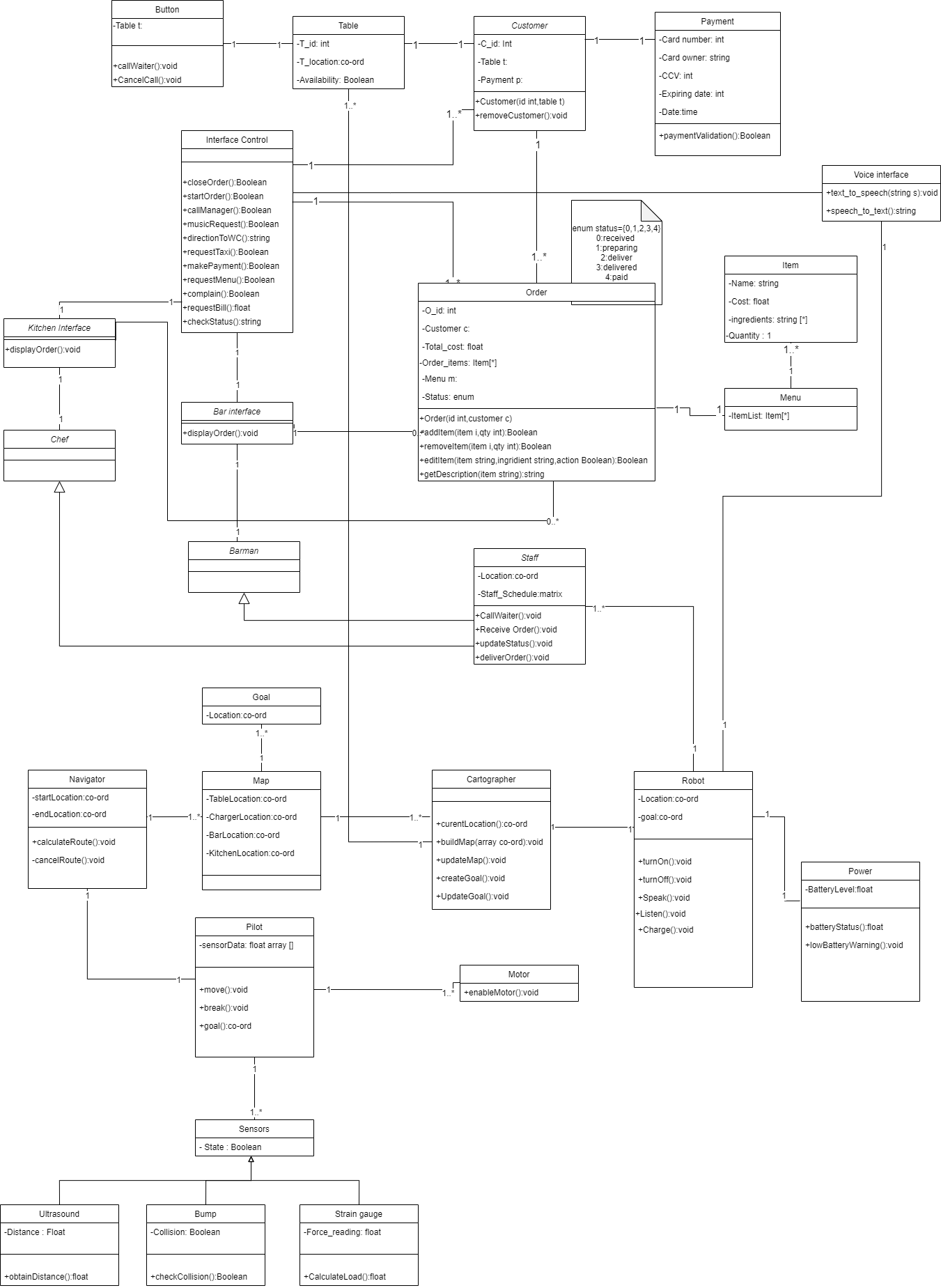
|  |  |
| --- | --- |
| Use Case | Call Manager  ID - 12  History modified 14/02/2020 |
| Description | The user wants to speak with the manager. The user requests the system to call the manager. |
| Actors | Customer |
| Assumptions | * The bar has a manager |
| Steps | 1.The system notifies the manager which table needs assistance. |
| Variations |  |
| Non-Functional | <Performance> The system knows who the on-shift manager is.  <Performance> The system has access to the bar’s shift schedule. |

### 7.2.16 Complaint

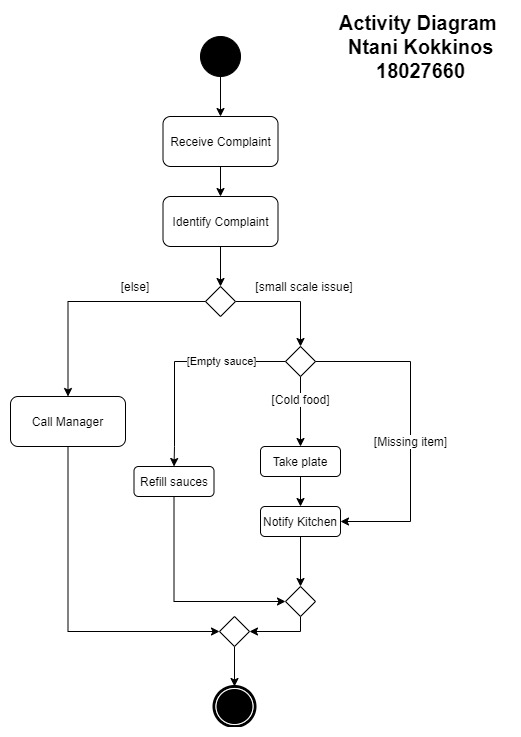
|  |  |
| --- | --- |
| Use Case | Complaint  ID - 13  History modified 15/02/2020 |
| Description | The user has an issue with his order. The system helps the user resolve the issue by either dealing with the issue itself or by calling over the manager. |
| Actors | Customer |
| Steps | 1. The user issues complaint to the system. 2. The system identifies the issue. 3. The system resolves the issue. |
| Variations | 1. The system deems that the issue is outside of its scope. 2. The system alerts the manager. 3. The manager resolves the issue. |
| Non-Functional | <Reliability> If the order is faulty, the system ensures that the fault is removed without adding further faults.  <Performance> The system contains knowledge on how to solve small scale issues:   1. Refill sauces – The system brings new sauces to the users table. 2. Cold food – Take plate away from user and notify kitchen to remake the faulty item in the order. 3. Missing item in order – Notify kitchen to prepare missing item |

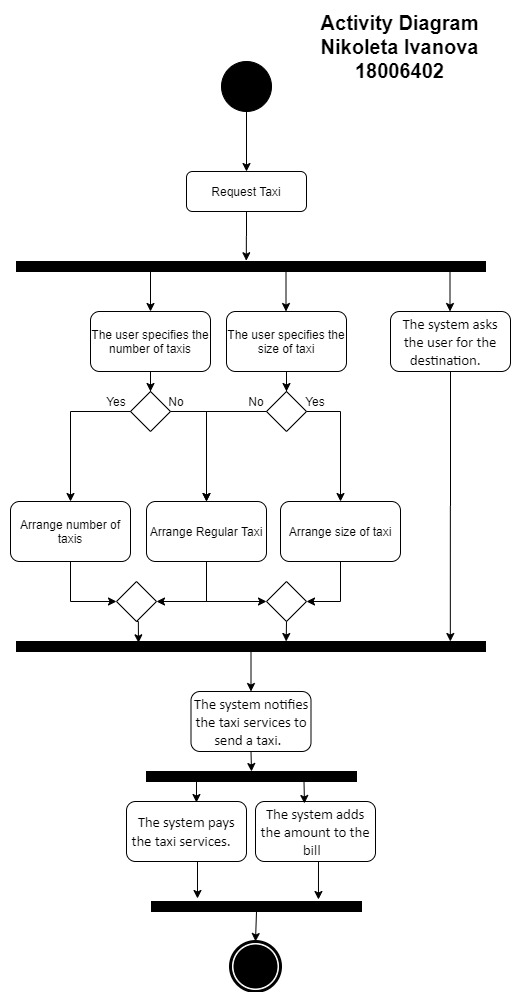
# 8. Class Diagram

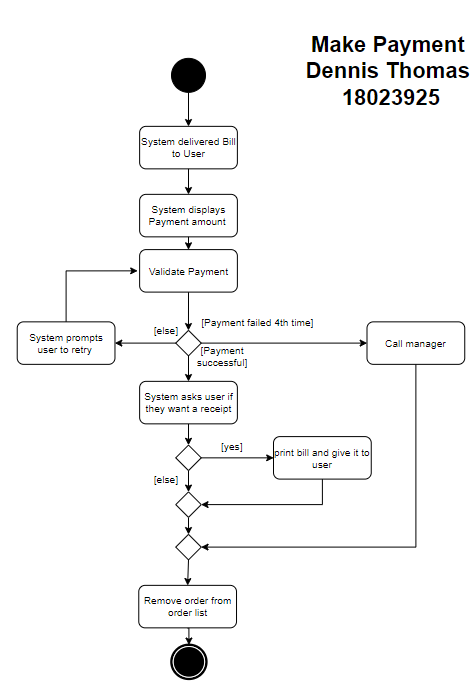
The class diagram is a representation of the interacting entities required to build the system.



# 9. Activity Diagrams







# 10. Sequence Diagrams

