**1 – Introduction**

*1.1 – Purpose*

This document represents the Requirements Analysis and Specification Document (RASD).

The document goals are to analyse the customer real needs and to show the solution the software offers, its goals and the relationship between these and the functional and non-functional requirements, attributes and constraints. In addition to that, it will provide possible use cases and system behaviour after the release.

This document is addressed to: procurement that can use it as a contractual base, project manager to control the developing and to have a base for change (or risk) management, to management that can estimate a first baseline (of time and cost) and to developer

*1.2 – Scope*

1.2.1 – Brief problem description

The coronavirus emergency has put a strain on society on many levels. Many countries imposed the lockdowns, which allow people to go out from their home only for essential needs as for working or for grocery shopping. They usually enforced strict rules to the stores and to the people that can go out in specific situation, such as limiting the number of accesses to a building or keeping the distance of at least one meter between people. Often these rules guarantee a store with few people inside, but a long-crowded queue outside.

Grocery store is the main challenge. Everyone need to do grocery shopping, food is an important part of everyone life, so it’s important to minimize the possible risks that a queue outside could take and, also, it’s needed an easy way to manage it, avoiding useless long waiting time, and a simple way to manage the influx of people. In addition to that, because of situation uncertainty, stores don’t want to hire more people to manage this problem.

1.2.2 – Brief system description

“CLup – Customer Line Up” is a system that manages the queue automatically without the people in a physical line. It should, obviously, cover all the functionalities that a good physical queue has, such as not losing the position and knowing when it’s customer’s turn to enter the store. In addition to that, the system will provide the estimated waiting time, a software part will be dedicated to the notifies, so when the customer must go out to reach the store in time, depending on his/her position. Moreover, the customer could decide to book for a specific time or putting in a normal queue.

By a manager view, this app could permit an automated management of the queue, regulating the flux in the building by setting the maximum number of people that could be inside the store at the same time. It will also manage the delays without any human intervention. It could also be integrated with proxies made by the store.

Moreover, in case of booking or line-up impossibility, the system will suggest other stores of the same chain/congregate near the user (if they exist) or other booking time (if they exist and only in case of booking attempt)

1.2.3 – Current systems

The main app in the market is UFirst that guarantees only the possibility to retrieve a number and to know how many people precede the customer. The time in which the customer can “line-up” is statically imposed, so, usually, it’s an hour before the closure. In addition to that there’s not any type of suggestion in case of impossibility to be inserted in the queue. Fortunately, the CLup has more features that can satisfy most of customer and store manager need, as described before.

*1.3 – Goal*

The goals written below are the mandatory parts that this application must have to achieve the purpose it has.

* [G1]: The system must allow customers to line up (on and offline)
* [G2]: The system must allow store managers to regulate the influx of people in the building
* [G3]: The system must provide customers with an estimate of the waiting time
* [G4]: The system must advice the customers that’s theirs turn to enter
* [G5]: The system must allow customer to book a visit in the store
* [G6]: The system must communicate the store manager the possibility to let in the store more people than the limit
* [G7]: The system must alert the customers when they have to leave to arrive in time at the store

[G2] and [G6] are about the management of the flux by the store manager: so how many people could be inside at the same time, or even what is the maximum “size” of every department. In particular [G6] allows the store manager to temporarily increase the maximum number, due to special condition, such as people specifies that they will buy items from different departments, so the maximum number in every department will never be reached and, even with more people, the health measures will be respected. Obviously, the system will communicate and not decide by itself because it isn’t aware about DPCM rules or whatever, so the final decision will be owned by store manager.

[G3] and [G7] are useful to let the customer organize and allow him/her to not lose the turn.  
[G4] could substitute a person outside the building that coordinate who must enter, so every management part of the queue is provided by the system,

*1.4 – Definition, acronyms and abbreviations*

1.4.1 – Definition

* Active customers: customers inside the building at the same time

1.4.2 – Acronyms

* RASD: Requirements Analysis and Specification Document

1.4.3 – Abbreviations

* CLup; Clup – Customer Line up
* [Gn]: n-goal.
* [Dn]: n-domain assumption.
* [Rn]: n-functional requirement.

*1.5 – References*

* IEEE Recommended Practice For Software Requirements Specifications - IEEE Std 830-1998

*1.6 – Overview*

The RASD is composed by five parts:

1. The first part focus on a briefly introduction to the current situation, to the solutions offered by the product, emphasizing its goals. This section analyses the current system available in the market, highlighting the main differences between them. Moreover, it specifies which conventions the document will adopt, to understand better the following part.
2. The second part offers a perspective of the product, underlining the main functions, attributes, constraints and user characteristic. It will also provide an initial explanation of requirements.
3. The third part focuses on all requirements and on use cases. It will also provide scenarios to justify all the functions.
4. The fourth part provides an explanation with Alloy, reporting also the results.
5. The fifth part is about the effort spent by the team to build this document.

**2 – Overall description**

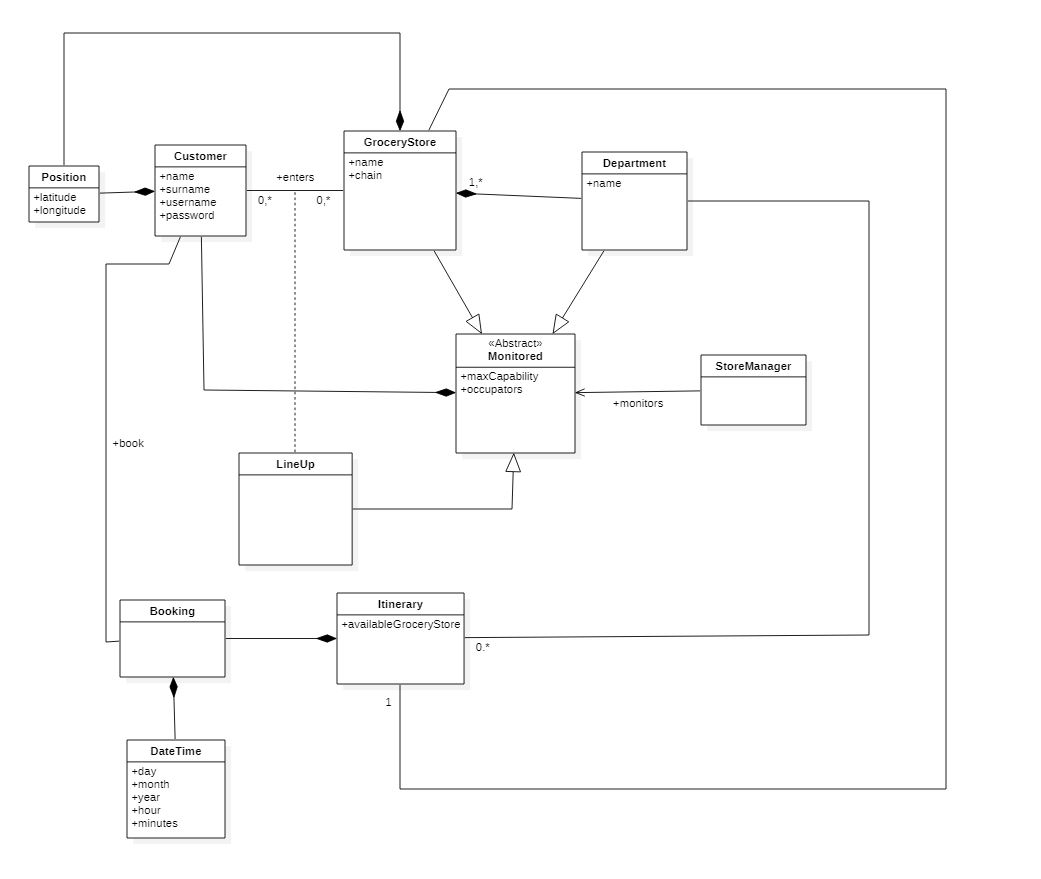
This section of the SRS should describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3 of the SRS, and makes them easier to understand.

*2.1 – Product perspective*

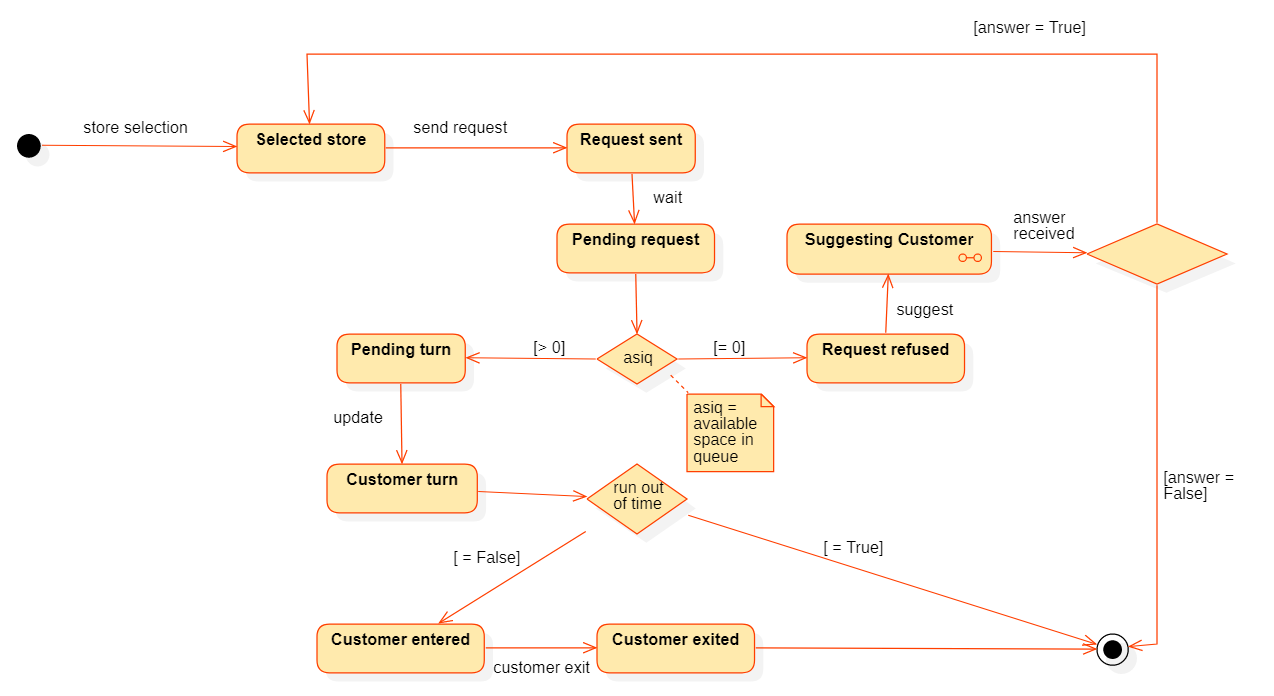
This product will be developed from scratch and, as said in the next section, it will use an external tool (Google Maps) to collect user position on which the alert and calculation system is based. It will ensure precision to our calculation and a better user experience because, usually, the user is accustomed to this tool. \*Moreover, to guarantee data security, an external third part will develop the authentication system. \*

The product, except for those, is almost totally self-contained.

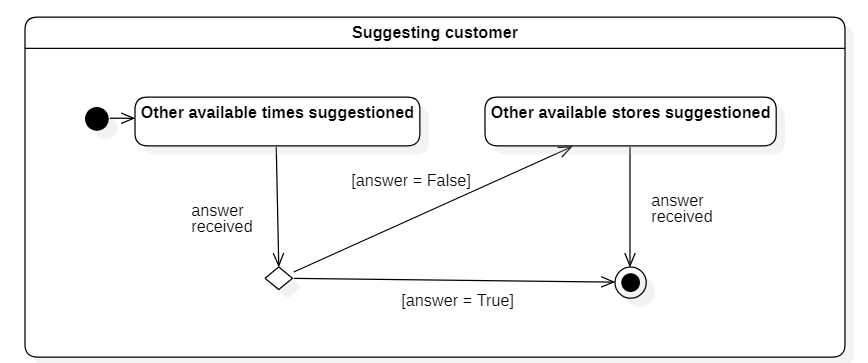
To analyse better the structure, a UML class diagram is presented below, which is a high level description of the component of the system.



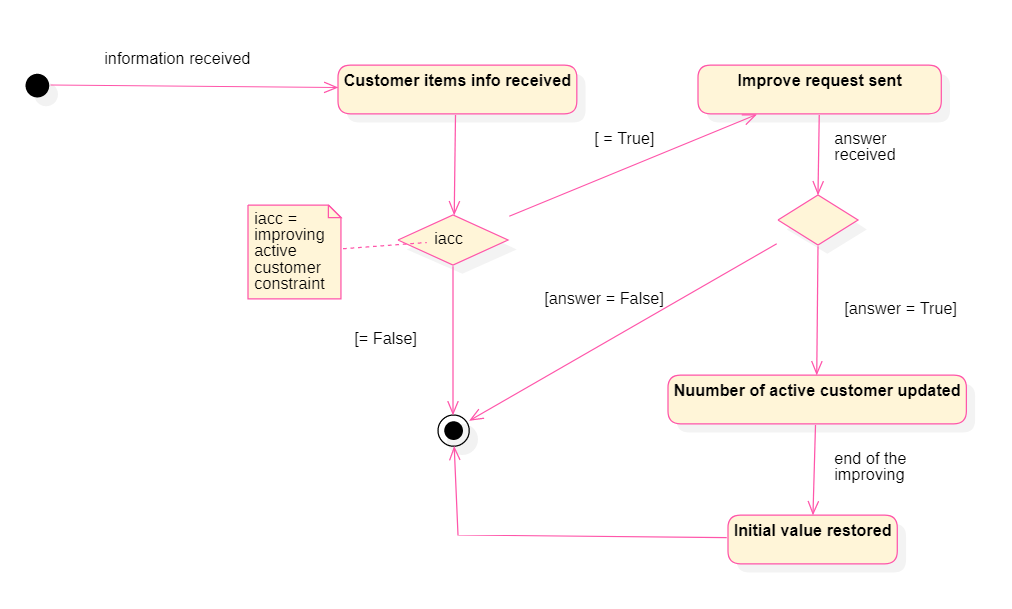
According to the possible situations and to the different point of view (e.g. for the store manager or the customer), the system will change its state many times and to explain the development of the changing the state-charts will be introduced below.



In this first diagram a customer view is offered, it covers the both cases (“retrieve number online” and “book a visit). Initially a store is selected, a request of line-up formulated and sent. If it is confirmed, the customer is officially in waiting for his/her turn until it’s time to enter. If he/she entered before the running out of time, the application will remain in this state until he/she will have finished the visit, otherwise it will directly reach the final state.   
If the request is rejected, the system will suggest the customer with other possibilities that are analysed below the next diagram. If the customer will accept the suggestion, the process will restart, otherwise it will finish.



The “suggesting customer” state varies depending what case is. In the “retrieve a number online”, this part contains only the “other available stores suggested”. In the “book a visit”, this state contains what the diagram above shows, so, in the beginning, it recommends other available times and, if the user rejects it, then it behaves like the first case.



The last critical aspect is the case of improving the customer inside the building due to the specified items they will buy. The system will send a request to the store manager, who can accept it or reject it. In case of positive answer, the number is improved, then restored after the specific situation.

In addition, it will be presented a brief description of the main characteristics of the interfaces hereunder.

*2.1.1 – User interfaces*

The user interfaces must contain short message with a neutral font. It must be intuitive due to the vast audience nature. The notifications must be simple and, also, with short message, such as “It’s time to go” for the incoming time of departure, or “It’s your turn”. In addition to that, the used colours can’t be green and red to explain different concepts, such as “red” for “not your turn” and “green” for “it’s your turn” because of the possible presence of daltonism. Hereunder, some examples are showed.

*2.1.2 – Hardware interfaces*

The system has no hardware interfaces.

*2.1.3 – Software interfaces*

The system doesn’t provide any API to external applications.

*2.2 – Product functions*

In the direction of defining unambiguous boundaries for the system in development, a specific list of the functions that the product will/won’t offer is provided and precisely explained. The first four is related to the user need, the second part regards the store manager uses.

*2.2.1 – User Registration/Authentication*

It guarantees that a user can register himself/herself to the application to gain all the service the system provides.

*2.2.2 – Retrieve a number / Book a visit*

The two functions are quite similar: “retrieve a number online” could be seen as a “book a visit” with the first time available. The system permits the possibility to put the customer in the online queue. A unique number will be assigned to every customer and their waiting time will be continuously updated. A notification will be also sent in two main cases:

1. The waiting time is updated to 0 and it is customer turn
2. The waiting time is equal (or greater with a range of tolerance) to the estimating spent time in reaching the store.

*2.2.3 – Cancel the reservation*

In case of possible mistakes, the users are able to cancel the reservations done and they can make another one. The possibility to delete that is forbidden in case the waiting time is lower than a certain tolerance value.

*2.2.4 – Queue management*

The system manages all aspects around the queue: it maintains the customers position in the “line”, it supports an integration “by hand” from the manager, it keeps tracking of the user inside the building to avoid the overgoing of the constraint. In addition to that, it controls and manages the user delays (with a tolerance parameter) or reservations cancels and organizes the queue in a way that can support them, avoiding, for instance, the possibility to give 5 minutes to a user that needs 10 minutes to reach the store.

The end of the queue is managed dynamically, indeed from the closure store time the system calculates when to stop accepting reservations.

*2.2.5 – Registration of the store*

It’s possible to register every store by own manager through authentication, also in case of store chain. The app supports the insertion of the timetable, the type of items, the mean average of the customers visit time, the maximum number of people inside the store, the maximum number of people per department.

*2.2.6 – Values management*

The application dynamically supports the changing of the main value previously described by the store manager

*2.2.7 – Collecting user data*

The system collects for every user the time of his/her visit to calculate a better approximation of the same, in a way that the queue management is more effective.

*2.2.8 – Store Manager Authentication*

Every store manager is able to login in the application and accesses to the other management functions.

*2.2.9 – Creation of special accounts*

Not all people could access the required technology, therefore the system permits every store to create a special account that can make how many reservations they want. These are seen as the same made by online users, so they don’t have privilege to skip the line or to access immediately, except for an absent customer and the next one could reach the store in the necessary time.

*2.2.10 – Display store’s info*

The store manager is able to see dynamically the collected data about the store, so what is the average of the visit, how many customers are inside the building, how long the queue is.

CHOICES AND REASONS:

1. WHY DID YOU PUT INTERFACES IN SECTION 2 AND NOT IN THE 3rd?   
   🡪 We choose it because the customer and the user are a part of the RASD audience: they’re interested in high-level functions and not in specific requirements. Surely, interfaces are a key point for customer (because they will use them to manage the system) and for customer (bad interfaces implied a bad app/system according to them [and to us]). So why we have to put in the third part that are more technical and less interesting for them. It’s a customer/user-oriented choice.
2. ASSUMPTION: booking 🡪 select store 🡪 select time slot 🡪 if not available: suggest other slots in same store 🡪 if rejected: suggest other stores available in previous time selected.