A.Y. 2020-2021

Politecnico di Milano, Software Engineering 2 project

POLITECNICO
MILANO 1863

RASD

Requirement Analysis and Specification Document

version 1.0

Cirino Duilio - 968466

Cocchia Lorenzo - 968139

Table of contents

Table of contents **2**

1. INTRODUCTION **3**
- Purpose 3
- Scope 4
- Definitions, Acronyms, Abbreviations 5
- Revision history 6
- Reference Documents 6
- Document Structure 7

2. OVERALL DESCRIPTION **8**
- Product perspective 8
- Product functions 12
- User characteristics 15
- Assumptions, dependencies and constraints 17

3. SPECIFIC REQUIREMENTS **19**
- External Interface Requirements 19
- Functional requirements 24
- Performance Requirements 52
- Design Constraints 52
- Software System Attributes 52

4. FORMAL ANALYSIS USING ALLOY **54**

5. EFFORT SPENT **61**

REFERENCES **66**

1. INTRODUCTION

A. Purpose

This document is the Requirement Analysis and Specification Document. The purposes of the creation of this document are to describe the system both with functional and non-functional requirements, analysis of the domain and of the goals to achieve through the system, exploitation of the constraints, the limits of the software and illustrate use cases which can occur with the system.
The project has the aim to provide a software to grocery shops to control the entrance of the customers in periods in which social restrictions can be imposed by the regional or State's laws that have the objective to avoid contacts between people to prevent illnesses from spreading. To fulfill this aim there have been identified the following goals that the software-to-be will have to accomplish for the special needs brought by the real word:

G1	make people able not to get in contact with other people	
	1	allow customers to "line-up" remotely effectively
	2	allow customers to know in real-time the current line-up number
	3	allow customers to see the available spots in the registered supermarkets
	4	allow customers to "book a visit" to the store
	5	provide an estimation of the waiting time
	6	provide alarms to customers basing on their position and the position of the store
	7	provide fallback options for people who do not have the access to technologies

	8	allow customers to get into the store when they are allowed to
	9	allow users to register to the system
	10	allow users to access to the system with their credentials
G2	allow managers to regulate the people fluxes inside of the grocery shops	
	1	allow managers to register and manage a new store
	2	allow managers to monitor the entrance
	3	allow managers to handle the line
	4	allow managers to manage employees
	5	allow employees to manage the lines

B. Scope

CLup is a system-to-be whose objective is to regulate queues of customers virtually, in order to prevent long lines of people waiting in front of the shop. The idea of such software generates from a real world need, in this specific case the COVID-19 pandemic. For this reason we found ourselves in a scenario never faced by our current society, a scenario that radically changed our way of living and our habits. One of the many aspects influenced by the situation is the interaction with other human beings, that is because the virus spreads thanks to close contact with other individuals. So every aspect of our daily life has been affected by this condition. The aspect on which CLup focuses is grocery shopping and it has been conditioned because of the strict social distancing rules adopted by every country. Moreover the software will not only be essential during the pandemic, but also in a normal situation it could improve efficiency in

daily life shopping, controlling in a better way the flow of people inside the store, improving customers' satisfaction, avoiding unpleasant queues at the cashiers or every other place inside it. Until now stores had to manage lines of people waiting outside of it, often creating unwanted circumstances where people could come in contact with the virus and help its spreading through the population.

C. Definitions, Acronyms, Abbreviations

line-up at the store	a system that allows the customers to create a line of people which determines the sequential order of the people who have the right to get into the store
book a visit	a system that allows the customers to acquire the right to get into the store in the future, based on a choice made by the customer given a set of options by the software-to-be
line-up number	progressive integer number, reset at every opening time of the grocery, which allows people at whom it is assigned to get into the store when it is called
reservation	when talking about reservations we intend each individual application of a customer to a store, by lining-up or by booking a visit
RASD	Requirements Analysis of Specification Document

QR code	Quick Response barcode system. It is a machine-readable optical label that contains information about the item to which it is attached.
POS	Point Of Sale (for further details, see [3.A.2.1])
Customer	They are the stores' customers. The term includes users both registered and not-registered to the software-to-be system
Gn.m	Goals identification n = macro-goal number m = sub - goal number
Dn	Domain assumptions identification n = domain assumption number
Rn	Requirements identification n = requirement number
Macro-goal	A goal of the project which can contain other sub-goals
Sub-goal	A goal of the project contained in a set of other sub-goals, composing together a macro-goal

D. Revision history

This is the first version of the document.

E. Reference Documents

- IEEE ISO/IEC 29148, 2011. "*Systems and software engineering, Life cycle processes, Requirements engineering*"
- Project specification document, from Software Engineering II course A.Y. 2020/2021

F. Document Structure

This document is composed of 5 chapters.

The first chapter is the introductive one: here we identified the goals of the project within the context in the real world. Moreover, we gave some definition, acronym and abbreviation and provide some information about the revision progression and the references.

The second chapter consists of a high-level description of the software-to-be. Here we listed some scenarios, described the shared phenomena, designed some diagrams such as class diagrams and statecharts in order to provide a formal description of the model and finally we listed all the main functions that the software-to-be will have to accomplish given the requirements.

In the third chapter it is provided a definition of all the possible software requirements, even linked with some domain assumptions and some goals, which can be useful to designers to design the software-to-be. Here there are included requirements related to the external interfaces, the functional and non-functional aim of the system, the non-functional part of the system. Finally, some information about the design constraints and the software system attributes.

The fourth chapter aims to describe the main functionalities model created throughout the document in a formal way, using the Alloy specification language.

In the fifth chapter, there is a description on how much effort the components of the group paid quantified in hours.

2. OVERALL DESCRIPTION

A. Product perspective

In order to provide a perspective of how the product will behave in the real world dynamics, this section provides some examples in terms of scenarios, details on the shared phenomena and a domain model designed through class diagrams and state diagrams.

I. Scenarios

1. Ted is a 16-years-old guy who lives nearby a local grocery store. Ted knows that the only way to access such supermarkets is to use the CLup system, which includes the app in the Play Store. He wants to go shopping the day after so Ted downloads the app, planning to book a visit for the next day. Ted, providing his house's address, is able to see the supermarket is 100m far away from his house as he expected. Additionally, it shows that he can line up to the store, with an estimation time of an hour. Ted decides to line up, and the application gives positive feedback and provides a QR code and a line-up number: 56. When he was preparing to get to the shop, Ted received a push notification and an SMS reminding him to go to the store in order to not lose the allowance to get in. As soon as he arrives at the grocery, he notices a display in which there is displayed the current line-up number, 45. After a while, the display shows 56 so that Ted is able to get into the grocery store scanning his QR code. As soon as he finishes the shopping, he scans the code again and goes back to his house.

2. Marshall is a 65-years-old man who is not keen on new technologies. He is worried because he heard from the news that the only way to go to the supermarket will be through a software-based system, CLup. Marshall refuses to use everything related to the technology so he decides firmly to go to the nearer supermarket without booking anything previously online. As soon as he arrives at the supermarket he notices that he can not get in without a QR code and, thanks to the in-store CLup system, he notices that he can line up to get inside the store. An employee

from the store creates the code with a number: 20. The employee, after this operation, communicates that the estimation time is 25 minutes. From his position it is visible a display which displays numbers, now it is displaying 9. As soon as the 20 is displayed, Marshall goes to the turnstiles and scans the QR code, the turnstile opens and Marshall now can get into the store. His worries about the software-based system are now only a memory, and happily he goes shopping for his family.

3. Lily is a 55-years-old woman who wants to go to a grocery store because she has no more meat in the fridge. She knows that the CLup system allows her to plan a visit to the shop. Through the system she finds out that she can book a visit for today in a supermarket that is not the nearer from her house but that she can reach by foot anyway. Lily is so excited because it is Sunday and the available spot is immediately after lunch time at 3pm and often in those hours the supermarket is really crowded and she believes that the software can avoid that crowd in that part of the day. Lily decides to book the ticket immediately and QR code is created. Lily at 2.50pm is really excited in front of the grocery store and she finds out that her visit has been postponed to 3.20pm due to the crowd inside. Lily is disappointed but she decides to wait in any case. At 3.10pm she thinks that she could try to get inside anyway since she already has the QR code. Lily tries to scan the QR code but again the system answer is that she is not allowed to get into the store because the visit has been moved to 3.20pm. It is now 3.20pm and Lily is allowed to go shopping and after having scanned the QR code with the cashier, she returns back to her home.

4. Robin is a 35-years-old manager from a corporation of grocery stores. From her R&D section it has been proposed to use a new system to manage the queues to the supermarket. She is interested so she decides to look on the Internet to see how it works. The system offers the possibility to register one or more supermarkets so, after an Administrative Board hearing, she decides to register all the grocery stores owned by the corporation. She only has to provide the position of them and to pay the sum communicated by the site. After that the CLup staff have controlled the effective truth of the position of the groceries and installed the hardware and software assets necessary to use the system, the stores are

ready to go. Robin has done a good job and receives an increase on her salary by her boss.

5. Barney is a 40-years-old businessman and he decides to include a visit to the grocery store in his already full schedule, booking it a week earlier than the fixed day at 5pm. The week passes and today Barney has the visit booked to the store. His agenda is full even today, and the meeting that he had at 4pm with an estimated time of 45 minutes is taking too much time: finally it ends at 5.20pm. Barney has booked the visit 20 minutes far away from his office, so he decides to get in contact with the CLup staff to ask if it was possible to postpone the visit to 6.30pm. He finds out that his visit will be delayed, so he has to do again the procedure to book the visit. Barney notices that the first spot available is tonight at 7pm. Barney is on time so he gets into the store and does the shopping that he has to do. At 8pm, at closing time, he gets out of the store and goes back to his home.

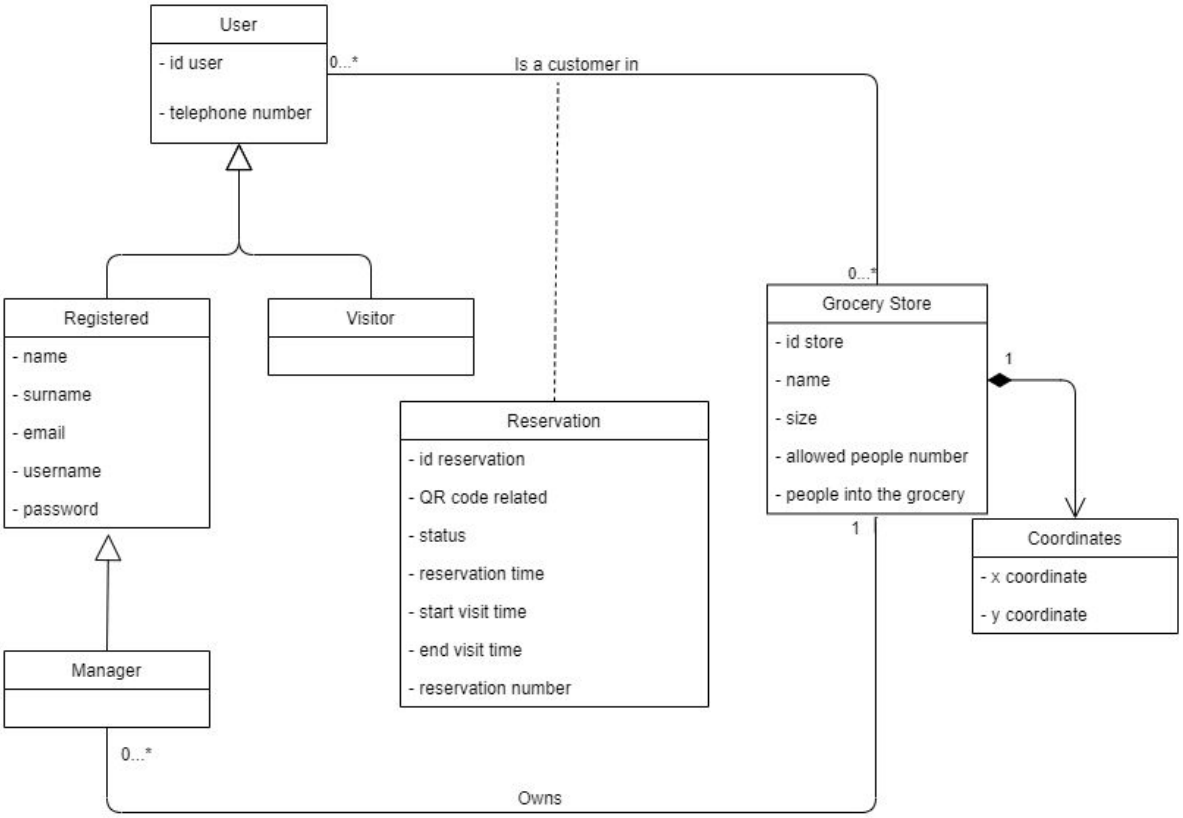
II. Further details on shared phenomena

The perspective of the expected shared phenomena is really important in the case involved by this system mainly because the customers cover a wide spectrum of demographic characteristics and, some of them, could not rely on software-based systems. The product has to be made even for those people which are not really confident with the latest technologies and it has to consider even options for people which will not be able to use the “main” methods through which the product will be delivered such as a mobile application or a web site.

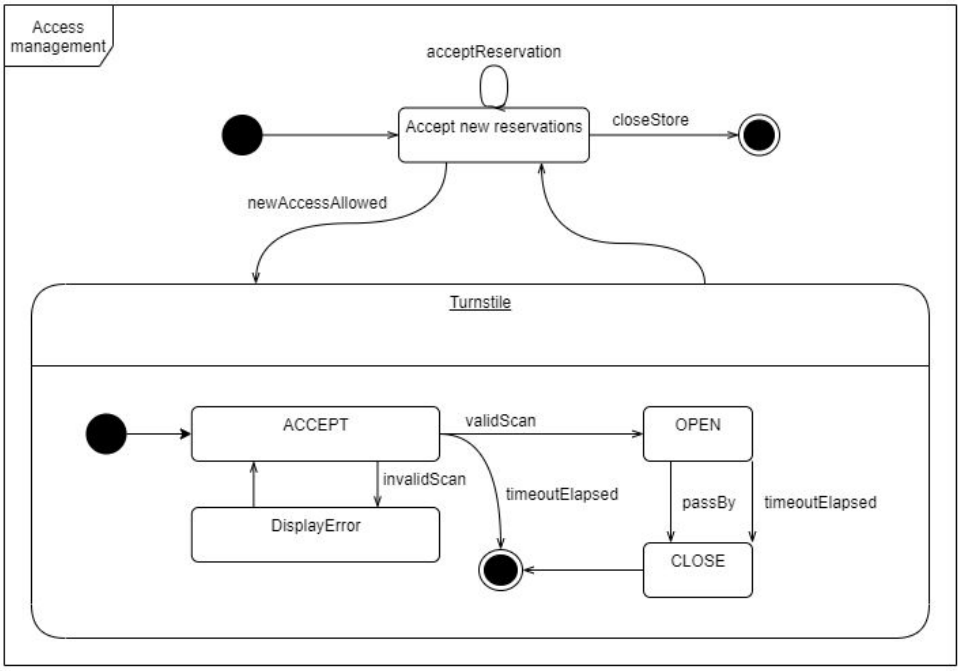
More precisely, the product will have to support even telephone calls to book or manage the visits to the grocery store in order to enhance the availability of the service provided by the product. Moreover, the product will have to provide some well-equipped and well-formed staff into and outside the store so that it can provide assistance to anyone who needs help related to the service. In addition the in-store staff could even be able to generate QR codes for visits so that people can directly go to the supermarket to generate a QR code, or people that do not have the possibility to scan the code by their smart devices or do not own a personal printer can go to the supermarket to print it and gain the permission to get inside the store.

III. Domain model

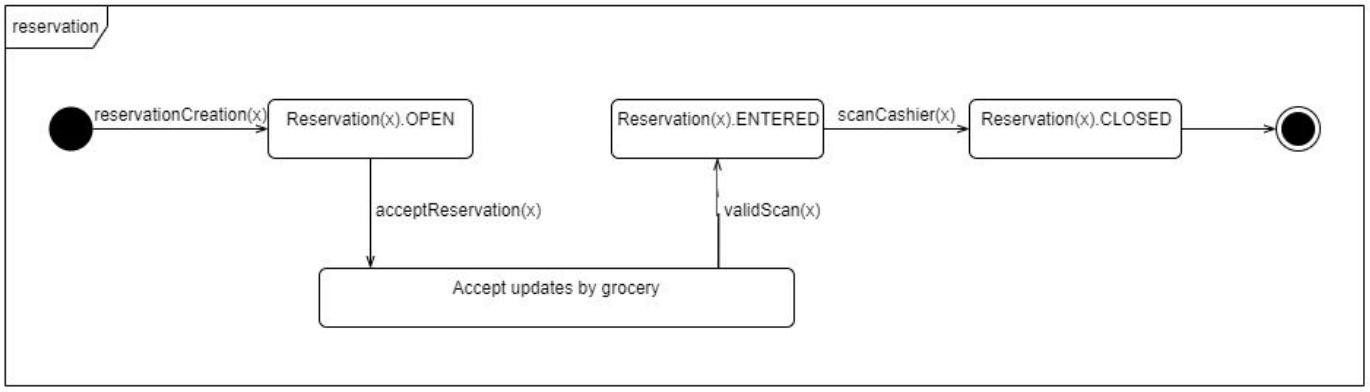
Here we can see the model of the domain designed as a class diagram.



In the state diagram below we can see the various states through which the access is granted to the customers.



Here we can see in detail the lifecycle dynamics of a certain reservation “x” as a state diagram.



B. Product functions

This section will describe the functions the service will offer, paying close attention to every detail concerning what they will actually do and its interaction with the entire environment.

B.1. Search stores

Every user of the service, be it registered or not, will be able to see every store, in their proximity or in a given location (specified by the user), that leans on CLup for customer entry management, pointing out if it’s possible to currently line up to a specific shop. The service would also suggest a list of frequently chosen stores (if the user has a previous history of reservations), to let the user easily take a look at the situation of its favourite shops.

B.2. Line-up *

After selecting an available store thanks to [B.1.], the user will be able to line up at the selected one and see the estimated time [B.4.] of its possible reservation, whether it is registered or not. If a user chooses to line up, the service will ensure its addition to the store’s queue. The user will also be asked to clarify the type of visit he intends to carry out (short, medium, long), and by so doing, get a more precise time estimation [B.4.]. After that the user will immediately receive a number that identifies his reservation

(in the form of a QR code or a serial number), the code will be required at the entrance of the store and the exit, verified by the interaction with another actor such as a scanner or an operator (preferably the first one during the pandemic). In case the user will present at the store with just the serial number it shall ask an operator to convert it to a QR code that will be handed to the user by a specific machine. The user will use the QR there generated to have access to the store.

B.3. Book a visit *

In addition to [B.2.] a user will also be able to decide if he wants to book a visit to the store, meaning that he will be able to choose the exact time and date of its visit to the selected store if possible.
The visit will be modifiable in case of need, both the user and the store workers can modify it. In both cases the user will be notified.

B.4. Time estimation

Another important feature the service will offer is the time estimation for the visit to the store. It's present in every usage situation of the software and it serves the purpose of suggesting the user when his turn to enter the store arrives.
In addition to this, users will receive a suggestion about the time of departure, based on their current position, to arrive in time for their reservation. Customers will also be notified of the timing status of their reservation whenever a change will happen.
This functionality will be based on customers' way of behaving, (average time of visit, from an historic of previous visits) or based on the duration he intends to spend by grocery shopping (short, medium, long). Whenever not specified, time estimation will be based on a medium stay inside the store.

B.5. Registration

Registration is present for both stores and customers. For the first registration to the service is mandatory, whereas customers can also benefit from the product functionality as a visitor, without the need to register. For the stores, during registration (mandatory for stores which

apply to the service) managers will be asked to give information about them (also for ensuring their actual existence) and preferences over the management of line-up, book a visit and time estimation.

B.6. Store management

The software-to-be will also allow store managers and employees the possibility to manage every aspect of the service which involves the store.
All of this will be changeable on request or directly by the store.
Stores will also be able to generate, on request, an entry code to every person that for any reason couldn't be able to get direct access to the service.

Optional functionality on reservations (*)

In case of emergency, reservations will be available for each user (registered or visitor) only once per day for both lining up and booking a visit together. The only exception will be if the reservation will not be used at the time the number will be called, enabling the user to make another reservation for the same date. The emergency state will be enabled or disabled only by the software-to-be makers whose intervention must be immediate.

C. User characteristics

This section will focus his attention on the various users of the service, clarifying their characteristics and needs, in order to better understand the reason behind certain choices and requirements that will be later specified (Section 3 of this document).

1. Customers

Customers are the more numerous target for this service and most of the product functions are thought to make them satisfied and inclined to use it.
They can be divided into three big main categories :

1.1. Technologically literate:

These are the people who are able to make complete use of such softwares. For this audience are thought more complex structured functionalities such as specifying a list of categories to buy and the possibility to add favourites. Registration is easy and they will not have any problem doing it, making their experience more personal and overall better. Also line-up and book a visit will not be problematic at all.

1.2. Technologically semi-literate:

These people are decent with such softwares, have used it before, so they know where to put their hands, but too much complexity wouldn't be ideal for them, because it could cause some trouble for their experience and deter them from its usage.
To satisfy their needs the software must provide an easy-to-use interface, making the use of the product functionality immediate and unambiguous (few pages and everything clearly visible).

1.3. Technologically illiterate:

These people are not good at all using any type of software. Especially for them, an easy-to-use interface is needed. The option to get in line physically, by retrieving the number is also made for them.

2. Stores/ Grocery shops/ Shops (synonyms used stated for clarity)

Stores are the other main target of CLup, they are needed for the service to be used by customers, because without shops they wouldn't be able to line up for anything. The software-to-be will offer them the possibility to manage the influx of people in their buildings.
Two type of targets are involved for stores:

2.1. Managers:

They will be the ones to register the stores to CLup and will also be able to manage every single aspect that regards the store such as: time periods for customers stays, business hours for the store and the adding, editing and cancellation of reservations.

2.2. Employees:

They will be the ones interacting with the customers so their usage of the software will mainly comprehend the fallback options, handing tickets to customers and taking reservations on the spot with managers' privileges. They need to know the basics to use the software and will have to be educated if needed.

D.Assumptions, dependencies and constraints

I. Dependencies and constraints

A. Regulatory policies

1. The software must guarantee that no more than a set number of people can get access to the building.
2. The system must not make any use of the private information of the users, such as email, password, telephone number and similars, their use must be limited to the system itself.

B. Hardware limitations
    1. Mobile device
        a) iOS or Android smartphone or tablet with internet access
        b) Non iOS and android phones must be able to make a call
    2. Computer
        a) Computer with internet access
C. Interfaces to other applications
    1. Interface with SMS gateway providers, to send notifications to clients.
    2. Interface with a maps applicative
D. Parallel operation
    1. The server supports parallel operations from different clients and different stores.

II. Domain assumptions

In the following table we inserted all the assumptions that affect our document, the first column will be the identifier for the said assumptions, the second will contain a precise description of the assumptions:

D.1	In case of any modification the store manager proceeds to update it on the service with success
D.2	Informations such as position, employees, business hours, book spots availability, maximum capiency and stay durations given by the store manager are always correct and registered successfully
D.3	Request to line-up is successful when the store is open
D.4	Request to book a visit is successful if done up to one hour prior the time of the visit and if a visit spot is available for that time
D.5	Whenever a line-up reservation is shifted in time, the customer is notified successfully

D.6	The customer is successfully notified in time when it must leave to approach the store in time
D.7	An employee will always be present for customers to interact with
D.8	Stores have a waiting area where people can stay to wait their turn
D.9	Distancing inside and outside the building's store is guaranteed
D.10	Customers are able to know when their number is called in the case they're in the waiting area
D.11	Users of the service have a internet connection or phone connection
D.12	Every reservation has a stay duration
D.13	Every reservation has an identification number
D.14	Time estimation is always correct
D.15	The username must be unique
D.16	If a store is shown in the list of available stores it means it actually exists
D.17	Data is successfully stored in the system that is able to always get access to it
D.18	Store's equipment always works
D.19	Customers are never in groups of more than 1 person

3. SPECIFIC REQUIREMENTS

In this section we include the specific requirements of the software-to-be in terms of UI, hardware and software interface, communication interface and specifically we will explicate the functional requirements. Additionally we will provide a sight on the non-functional requirements and on the constraints related to the design.

A. External Interface Requirements

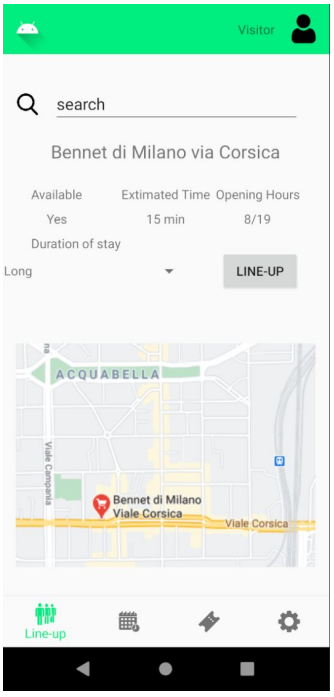
This section is made to provide some information about the inputs and outputs from the software-to-be.

E.1User Interfaces

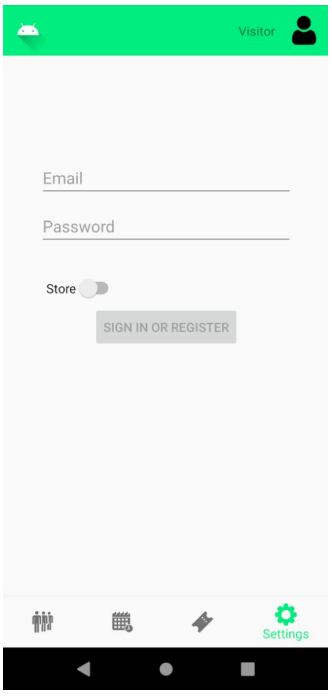
The following mockups are intended to show how just a few pages of the mobile app and web app will look, just to give an idea for the making of the user interfaces. The design document will contain better described and more precise models for the user interfaces.

E.1.1 Mobile mockup interfaces

a.Line-up view

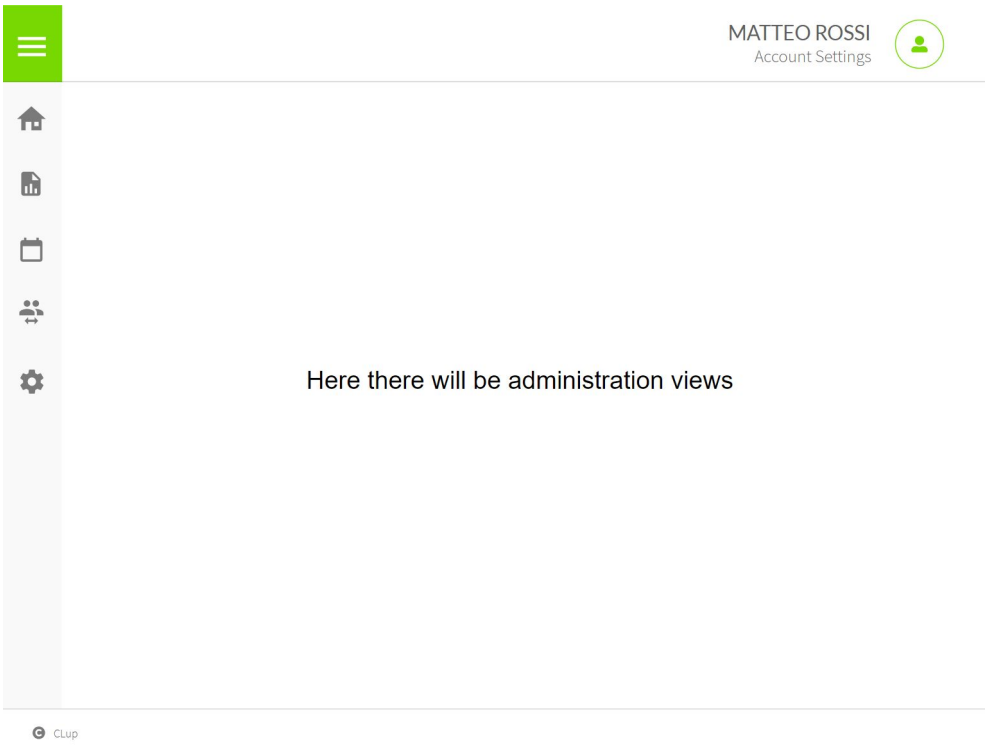


b. Login as visitor

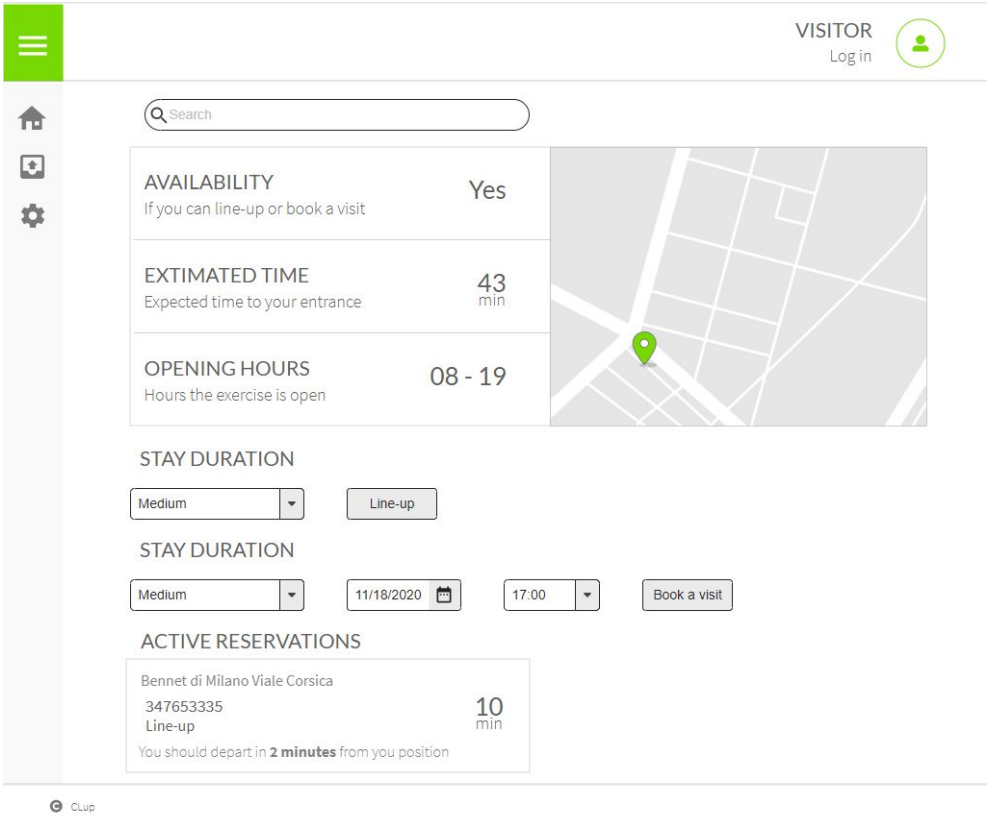


E.1.2 Web app mockup interfaces

a. Store manager view



b. Visitor main page view



E.2 Hardware Interfaces

In order to provide the entire service of the software-to-be it is needed to retrieve some hardware components. Here we provide some description of the hardware with some visual examples: it is not needed that the service will provide exactly the hardware interfaces exposed in the examples but the minimum requirements described have to be satisfied.

E.2.1 POS system

The in-store staff has to have the ability to create by themselves a QR code in order to exploit [2.B.2] and [2.B.3] functions. Due to some user characteristics (especially [2.C.1.3]) it is needed that the QR code can be scanned. In this case we consider it appropriate that the QR code is printed from a portable component owned by the staff: the system has to provide even to the staff an easy-to-use interface so that they can easily generate QR codes, eventually delete some others and generally speaking manage them. In this case the hardware component can be both a plug-in for some other device such as a smartphone or a tablet or can even be a stand-alone component. Here we provide an example of it. This component should even manage payments, in the case of eventual extensions of the software-to-be.

Minimum requirements to fulfill:

- print QR codes.
- provide hardware to manage the line by software.
- provide hardware to manage the booking system by software.

E.2.2 Smart turnstile

This hardware interface is needed to force customers to respect the entrance to the supermarket and to managers to control the entryway and the exit, so that it can be exploited the [2.B.2] function in an efficient way. This can be made with a traditional turnstile. With the aim to support the QR code reading, the component has to provide some machine with an optical reader that can read the QR codes. For this purpose, we think about a turnstile styled with the same design of the ones that you can find in Milan's Subway made by ATM.

Minimum requirements to fulfill:

- read QR codes.
- provide software and hardware to be able to check automatically if a certain QR code is allowed to get inside the store.

E.2.3 Smartphone

Users could be able to use the software-to-be system even through a mobile device. This hardware should support the software that covers all the functionalities of both customers that manager sides.
Minimum requirements to fulfill:

- Internet connection availability.
- GPS and push notifications permissions.
- Not-inverted colors and not-poor contrastated effects on the display so that QR code can be readable by the turnstile.

E.2.4 Traditional phone

The software-to-be system should even be accessible via phone especially for [2.C.1.3] customers who could not have the access or the skills to use a software system. Especially the services [2.B.2], [2.B.3] and [2.B.4] should be

accessible via a green number with a wide time availability over the day, in every day of the week and in every week of the year.
Minimum requirements to fulfill:

- No less than 8 hours per day availability.
- Support to staff in order to provide a store-dependent protocol made to provide to the customer [2.B.2] and [2.B.3] services.

E.3 Software Interfaces

Software interfaces have to be the most easy-to-use as possible because the targeted user can be wide and can comprehend a huge number of demographic, geographic and socio-cultural characteristics of the customers. In addition they have to permit as many extensions as possible so that some grocery companies could apply their own company software in order to customize their product.

E.3.1 Maps interface

In order to provide [2.B.1] functionality it is necessary to present a maps interface. Maps should highlight which are the closest stores to the current position of the customer. In addition, within this interface, it can be made the time calculation in order to exploit [2.B.4] product function and make the interaction with the user more intuitive and comprehensible.

E.4 Communication Interfaces

The software-to-be will have various components, and some communication interfaces will be necessary in order to communicate by remote.

E.4.1 Internet connection

In order to provide a remote service it is necessary to have a connection with the latest technologies, it has to be provided an infrastructure so that customers can interact with the application with cellular networks such as 5G/4G or LTE.

E.4.2 Telephone connection

To provide even more options to the customer to use all the functionalities of the software-to-be, it has to be provided even a telephone connection,

with an availability that has to be the most wide as possible. This option is thought especially for customers types [1.2] and [1.3].

E.4.3 Staff communication

With the aim to clarify every misunderstanding that the software could create, it has to be provided even some communication service with the in-store staff. Anyone should have the opportunity to go to the store and talk with someone from the staff. This is thought especially for [1.3] customers.

B. Functional requirements

In this section we will provide the functional requirements for each system operation or goal described in the previous sections ([1.A]), given some domain constraint for each case. The section begins with a map which links every goal to a set of requirements and a set of domain assumptions. After that, we listed some use cases definitions, provided some illustration of them and designed the relative sequence diagrams: all with the support of UML modeling.

B.1 Goal, requirements and domain assumptions mapping

The following table summarizes all the requirements and domain assumptions related to each goal, and the domain assumptions that are done.

	make people able not to get in contact with other people	
G.1.1	allow customers to line-up	
D.3	R1	The customer must be able to add himself to the current line
D.5	R2	The customer must provide at least a telephone number to line up
D.6	R3	The customer must be able to search for specific groceries
D.11	R4	The customer must be able to do the research basing on his position
D.13	R5	The customer must be able to make the research basing on a position provided by himself
D.15	R6	When a line-up request concludes successfully, an access code attached to a number has to be provided to the customer
D.17	R7	When it comes the turn of the lined-up user, he must be able to get into the grocery store
G.1.2	allow customers to know in real-time the current line-up number	
D.11	R8	The customer must be able to know the current reservation number everytime, even during the opening hours of the grocery store
D.13	R9	Current reservation number must be provided in real-time and on-demand
D.16	R10	The current reservation number must be refreshed as soon as it is possible
D.17		
G.1.3	allow customers to see the available spots in the registered supermarkets	
D.2	R3	The customer must be able to search for specific groceries
D.11	R4	The customer must be able to do the research basing on his position
D.16	R5	The customer must be able to make the research basing on a position provided by himself
D.17	R11	The customer must be able to know all the available spots in every registered grocery store
	R12	Registered customers must be able to access to shortcuts in order to see their previous groceries choices
G.1.4	allow customers to book a visit to the store	
D.2	R13	The customer must be allowed to book for an available spot in a certain grocery shop
D.4	R14	The customer must provide at least the telephone number to book a visit
D.11	R15	When a book-a-visit request concludes successfully, an access code attached to a reservation time has to be provided to the customer
D.13	R16	Registered customers must be able to access to shortcuts in order to book same reservations made in the past
D.16		
D.17		
G.1.5	provide an estimation of the waiting time	
D.2	R17	The customer must be able to know the estimated time of his visit: both for the line-up and the book-a-visit functions
D.5	R18	The estimated time must include even the estimation time necessary to move from the desired location to the store
D.6	R19	In case of deletions, the time has to refresh its value
D.12	R20	For registered users the estimation time must be based even on their personal behaviour
D.14		
G.1.6	provide alarms to customers basing on their position and the position of the store	
D.5	R21	In the case in which the customer has the technology to support alarms, he must be notified considering the estimation time

D.6	R22	For the line-up reservations, the alarm has to be provided basing on the current reservation number
D.11	R23	For the book-a-visit reservations, the alarm must be provided basing on the reservation time chosen by the customer
G.1.7		provide fallback options for people who do not have the access to technologies
D.7	R24	Fallback options must include all the product functions
D.8		
D.9		
D.10		
D.18		
G.1.8		allow customers to get into the store when they are allowed to
D.8	R25	The hardware must support visually the entrance of the customer with a display
D.9	R26	The hardware must support physically the entrance of the customer
D.13		
D.18		
D.19		
G.1.9		allow users to register to the system
D.15	R26	Registration must include name, username, email and password
D.16	R27	Registration must be possible from all the communication technologies supported
G.1.10		allow users to access to the system with their credentials
D.11	R28	Access must be possible providing email and password
D.15		
D.17		
G2		allow managers to regulate the people fluxes inside of the grocery shops
G.2.1		allow managers to register and manage a new store
D.1	R29	Managers must be able to add a new grocery in the list
D.2	R30	Managers must be able to update informations about the store
D.11	R31	Managers must be registered to the system
D.15		
D.17		
G.2.2		allow managers to monitor the entrance
D.11	R32	Managers must be registered to the system
D.17	R33	Managers must be able to see data mined by the application
D.18		
G.2.3		allow managers to handle the line
D.1	R34	Managers must be able to add to the line a reservation
D.11	R35	Managers must be able to edit a reservation in the line
D.18	R36	Managers must be able to delete a reservation in the line
G.2.4		allow managers to manage employees
D.1	R37	Managers must be able to add employees
D.11	R38	Managers must be able to delete employees
G.2.5		allow employees to manage the lines
D.11	R39	Employees must be able to log with the credentials provided by the manager
D.18	R40	Employees must be able to add users to the line

The previous table is oriented to expose the requirements related to goals rather than the domain assumptions. To better understand the domain assumptions related to each goal, it is provided a matrix: in the x-axis there are listed all the goals, in the y-axis all the domain assumptions. If a

domain assumption is made for a certain goal, then the related cell is signed with a “X”.

D \ G	G1.1	G1.2	G1.3	G1.4	G1.5	G1.6	G1.7	G1.8	G1.9	G1.10	G2.1	G2.2	G2.3	G2.4	G2.5
D1											X		X	X	
D2			X	X	X						X				
D3	X														
D4				X											
D5	X				X	X									
D6	X				X	X									
D7							X								
D8							X	X							
D9							X	X							
D10							X								
D11	X	X	X	X		X				X		X	X	X	X
D12					X										
D13	X	X		X				X							
D14					X										
D15	X								X	X	X				
D16		X	X	X					X						
D17	X	X	X	X						X	X	X			
D18							X	X				X	X		X
D19								X							

B.2 Use cases definition

In this section we will define some use cases in which it is possible to see the relations between needs and goals of the stakeholders and all the domain assumptions and requirements related to that. Note that we do not distinguish between registered and not-registered users because the system has to be accessible to everyone in all of its main functionalities.

B.2.1. A customer lines up

Actors	Customer
Goals	[G.1.1][G.1.5]
Input conditions	The user already accessed the home page
Events flow	1. The customer searches a grocery providing a position 2. The customer chooses a grocery in which

	line-up 3. An estimation of the waiting time is provided to the customer 4. The customer provides his telephone number 5. The customer confirms to line-up 6. A line-up number is provided to the customer
Output conditions	The customer is actually in the grocery's line, when the current line-up number of the store is the same of the customers he has the right to get in. A confirmation SMS is sent to the customer. Time estimation is constantly provided.
Exceptions	1. The same telephone number results in another line-up reservation in the same line 2. The store is closed at the moment of the confirmation All the exceptions are handled redirecting to an error page and going back the event flow to number 1.

B.2.2. A customer learns about the current grocery's line-up number

Actors	Customer
Goals	[G.1.2]
Input conditions	There is no input condition
Events flow	1. The customer chooses a grocery store 2. The customer sends the command to get the current line-up number 3. A message is sent to the customer containing the current line-up number, the line-up number assigned to the customer and the waiting estimated time
Output conditions	The current line-up number has not changed

Exceptions	1. The customer is not lined up to the store in which he asked the current line-up number The exception is handled with an additional information embedded to the message sent in point 3 of the event flow.

B.2.3. A customer watches the available spots in a given supermarket

Actors	Customer
Goals	[G.1.3]
Input conditions	There is no input condition
Events flow	1. The customer selects a grocery of which he wants to know the available spots 2. The customer chooses a preferred date 3. The customer chooses a preferred time 4. A set of available spots is provided to the customer
Output conditions	The customer is not appearing in the grocery's reservations
Exceptions	1. There are no spots available for the date preferred 2. There are no spots available for the time preferred 3. There is no more the spot chosen from the ones provided The exceptions 1 and 3 are handled by sending an error message describing the error that just happened and redirecting the customer to step 1 of the event flow. Exception 2 is handled sending an error message and redirecting the customer to step 2 of the event flow.

B.2.4. A customer books a visit to the grocery

Actors	Customer
Goals	[G.1.4][G.1.5][G.1.6]
Input conditions	The customer has already seen the available spots for a certain grocery store, a certain date and a certain time
Events flow	1. The customer selects the available spot from the ones provided by the software-to-be 2. The customer provides his telephone number 3. A confirmation message is sent to the customer 4. At the right time and the right date, a notification is sent to the customer considering the estimated time of waiting
Output conditions	The customer is actually occupying the spot reserved even for the software-to-be
Exceptions	1. The available spot is no more available for concurrency reasons 2. The telephone number is linked to another reservation yet The exceptions are handled by providing an error message to the customer and going back to the point 2 of the event flow for the exception 2. For the exception 1 user after have received the message is redirected to the selection of the reservation spots (user case B.2.2)

B.2.5. A customer uses a fallback option

Actors	Customer, Employee
Goals	[G.1.5][G.1.7][G.2.5]

Input conditions	The employee has already logged into the system with his credentials
Events flow	1. The customer gets in contact with an employee and asks to line-up to a grocery 2. The employee presents the availability of the grocery for that day 3. An estimation of the waiting time is provided to the customer 4. The customer chooses to line-up 5. The customer provides the telephone number 6. The customer provides a position 7. The employee inserts the line-up reservation in the system 8. A line-up number is provided to the customer
Output conditions	The customer is lined up
Exceptions	1. A line-up with the telephone number provided by the customer has been already done in that day 2. The available spot for the day is no more available These exceptions have to be handled by providing a protocol to the employee which has to consist of providing an error communication and asking the customer to go back to the number 1 in the event flow.

B.2.6. A customer gets into the grocery store

Actors	Customer
Goals	[G.1.1][G.1.2][G.1.8]
Input conditions	The customer has gained the access through the system, the customer is in possession of the QR code
Events flow	1. The customer is notified for the coming of his turn

	2. The customer uses the QR code to access the grocery store 3. The customer goes into the grocery and does the shopping 4. The customer scans again the QR code before of his exit
Output conditions	The entrance and exit logs of the customer are registered into the system. Turnstile behaves correctly.
Exceptions	1. The customer does not scan the QR code within 5 minutes from the allowed time 2. The QR code is scanned before of the right line-up number 3. The QR is scanned before of the right hour decided in the booking The 1 exception is handled by notifying the customer that he has lost the allowance to get into the store. The QR code is unabilitated, the customer can no longer go on with the event flow with this QR code. The 2 and 3 exceptions are handled by displaying the error in the turnstile display and forcing the user to wait his turn: the event flow has to remain at 2.

B.2.7. A user registers

Actors	User
Goals	[G.1.9]
Input conditions	There is no input condition
Events flow	1. The user expresses his will to register 2. The user provides his telephone number 3. The user provides a valid username 4. The user provides a password and repeats the password to be sure to insert the right password 5. A confirmation message is provided to the

	user
Output conditions	The user results enlist in the database of the application
Exceptions	1. The telephone number is already used by another account 2. The user is already used by another account 3. The repeating password is not the same of the first one 4. Some server error is thrown The exceptions are all handled by providing an error message. Exceptions from 1 to 3 have to result in asking again the related information (phone number, username or repeating password). Exception 4 causes the repeating of the whole process after the receiving of the error message.

B.2.8. A user logs in

Actors	User
Goals	[G.1.10]
Input conditions	The user is in a digital platform of the software-to-be
Events flow	1. The user provides his username 2. The user provides his password 3. The user submits his data 4. A confirmation message is sent 5. The user is redirected to the homepage of the application
Output conditions	The user have no more to insert his credentials to exploit the software-to-be functions
Exceptions	1. The credentials are incorrect The exception is handled by providing the reason

	for the error and asking if the user is actually registered to the platform.

B.2.9. A manager registers a store as a manager

Actors	Manager
Goals	[G.2.1]
Input conditions	The manager is registered as manager
Events flow	2. The manager accesses the homepage of the software-to-be 3. The manager expresses his will to add a new grocery 4. The manager fills a form including the address and the name of the grocery 5. A validation check is made by the system 6. A confirmation message is sent
Output conditions	The grocery is now part of the system. Through the shop, there can be exploited all the main functionalities of the software-to-be.
Exceptions	1. The manager has already registered some grocery for that address 2. The validation does not go well (there is no existent grocery in that address, there are problems in the verifications) The exceptions are handled by showing an error message explaining what happened and redirecting the manager so that the event flow returns to the point 2

B.2.10. A manager retrieves the information about the grocery's data

Actors	Manager

Goals	[G.2.2]
Input conditions	The manager has already registered some store as a manger
Events flow	1. The manager access to the platform with his credentials 2. The stores related to his account are provided 3. The manager selects a grocery 4. Some general data is provided to the manager 5. Some more detailed data are retrieved on-demand
Output conditions	The data is correctly provided to the manager
Exceptions	1. Credentials are not correct or the credentials inserted are related to an account which has not the privileges to get such informations The exception is handled by providing an error message and by forcing the user to not go further with the events flow unless right credentials are inserted.

B.2.11. A manager handles the line

Actors	Manager (or Employee)
Goals	[G.2.3]
Input conditions	The manager has already logged in and selected a grocery
Events flow	1. The manager chooses to manage the line or the reservations booked 2. Add/edit/delete options are provided to the manager 3. The manager chooses his preference 4. A confirmation message is provided
Output conditions	The add/edit/delete operation is effective in the

	system
Exceptions	1. The operation can not be done The exception is handled by providing an error message and redirecting the manager back to the point 2 in the events flow.

B.2.12. A manager adds an employee to the list of the store

Actors	Manager, Employee
Goals	[G.2.4]
Input conditions	The manager has already logged in and selected a grocery
Events flow	1. The manager chooses to edit the Employee list of the grocery 2. The manager chooses to add an Employee 3. The Manager provides name, surname and email of the Employee 4. The Employee is notified through an email with the login credentials to the platform
Output conditions	The add operation is effective in the system. The Employee is able to add users to the line from now on
Exceptions	1. There is already an employee with the provided email The exception is handled by showing an error message and redirecting the Manager to the point 1 of the event flow

B.2.13. A manager deletes an employee from the list of the store

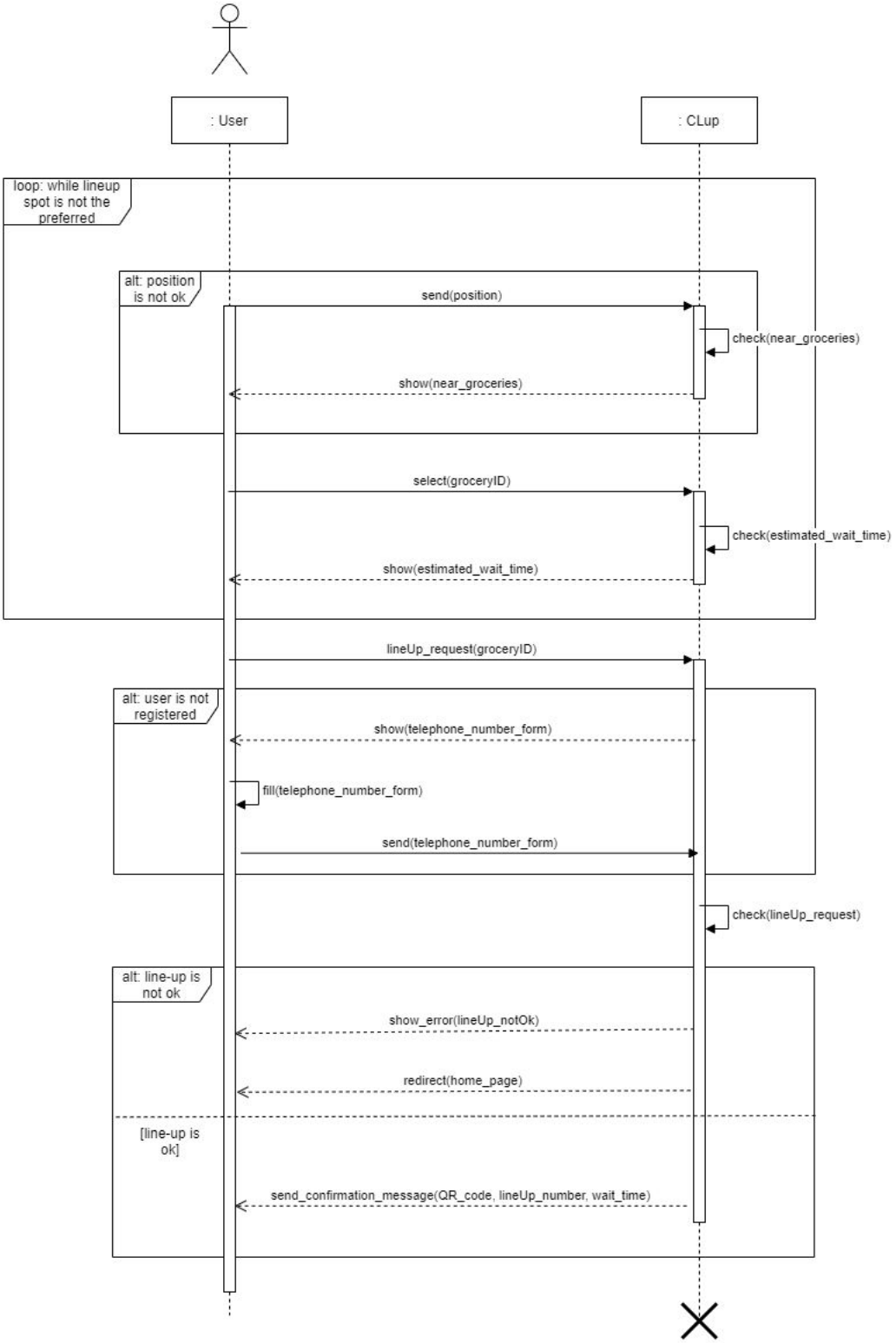
Actors	Manager, Employee

Goals	[G.2.4]
Input conditions	The manager has already logged in and selected a grocery
Events flow	1. The manager chooses to edit the Employee list of the grocery 2. The manager chooses to delete an Employee 3. A confirmation message is shown to the manager 4. The Employee is notified through an email
Output conditions	The delete operation is effective in the system. The Employee is no more able to add users to the line from now on, and his login trails are ineffective
Exceptions	1. The operation can not be made The exception is handled by showing the error reason and redirecting the user so that the event flow returns to point 1

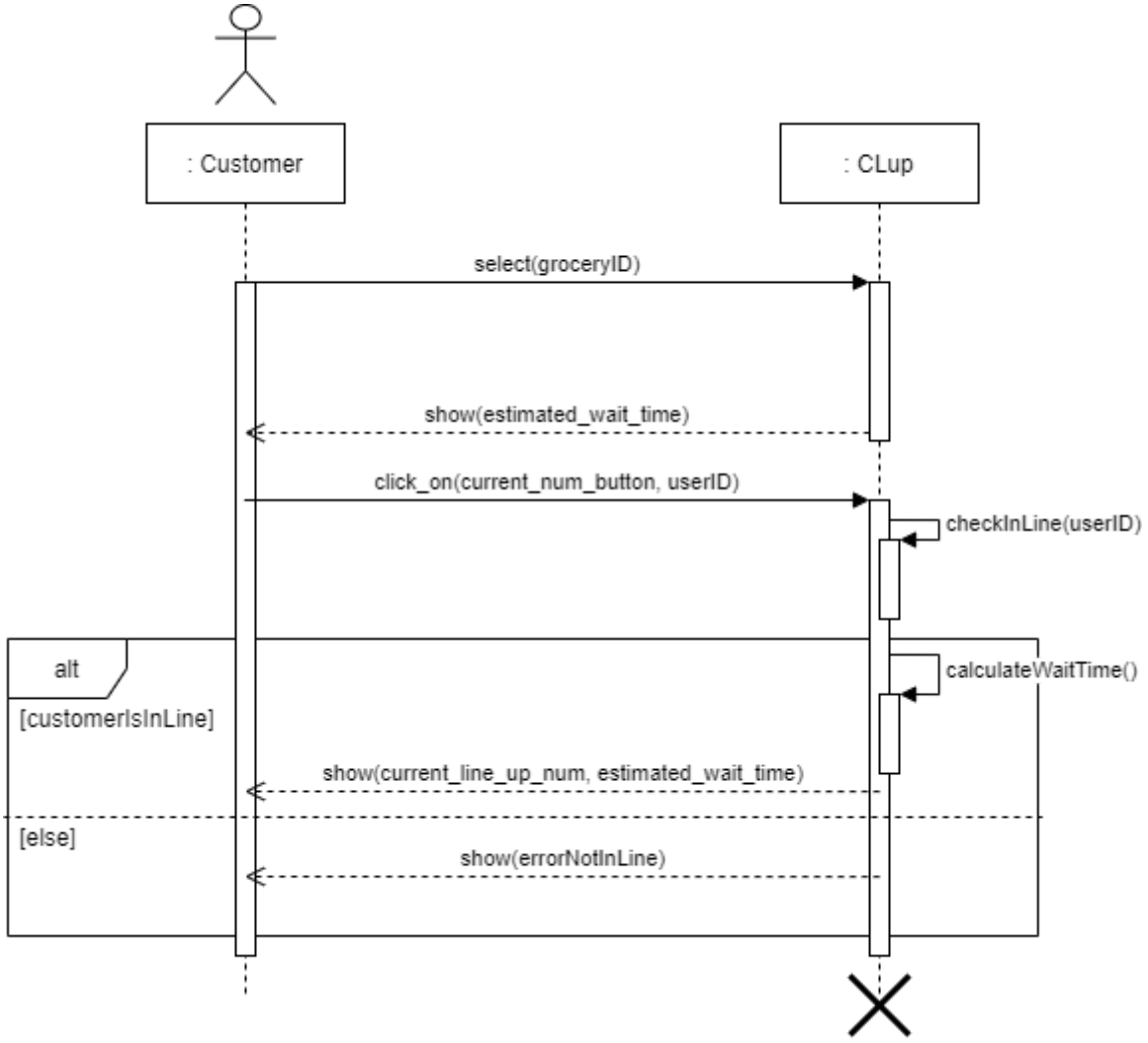
B.3 Use cases illustration

Here we provide a use case diagram model in UML sequence diagrams.

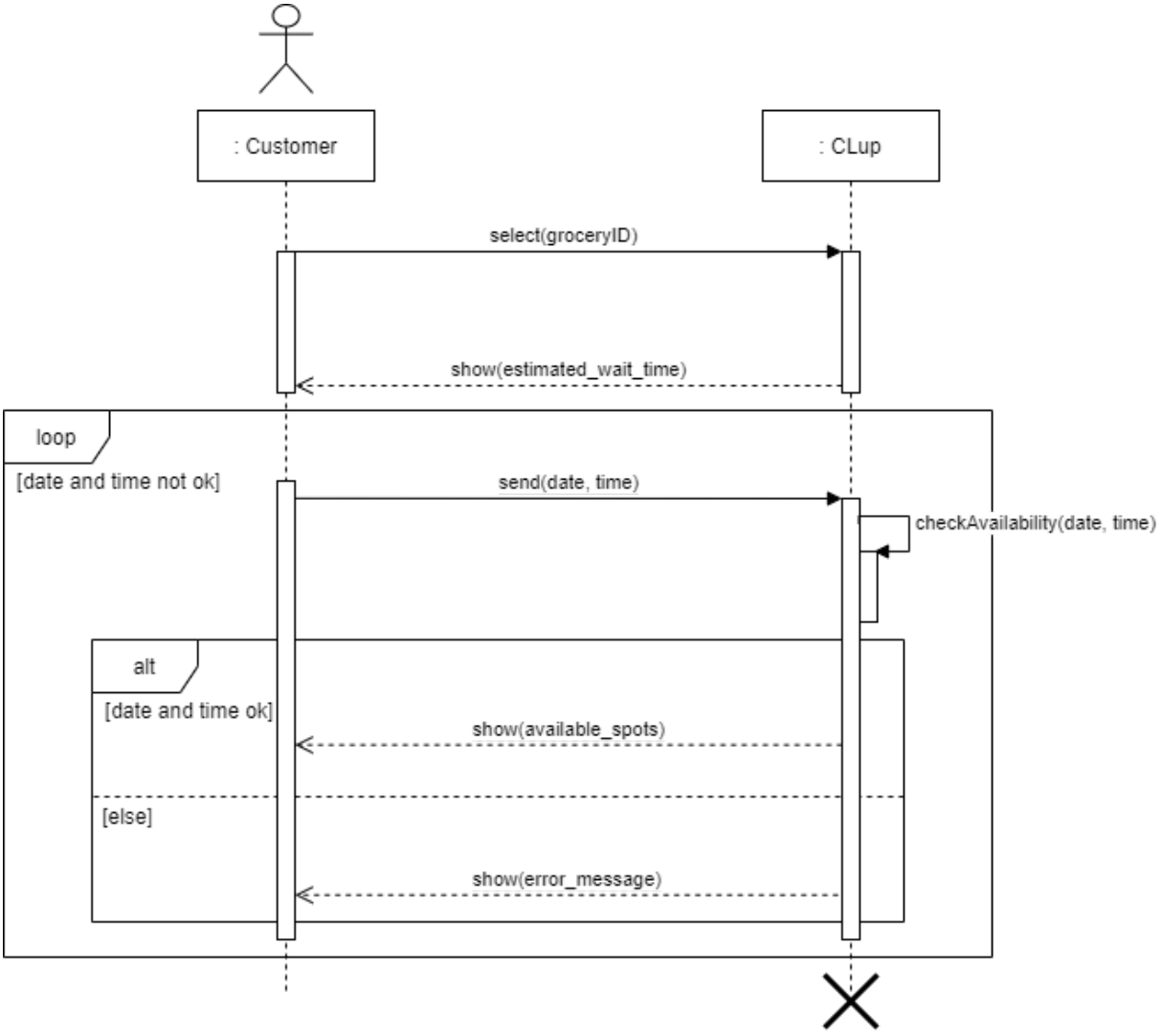
B.3.1. A customer lines up



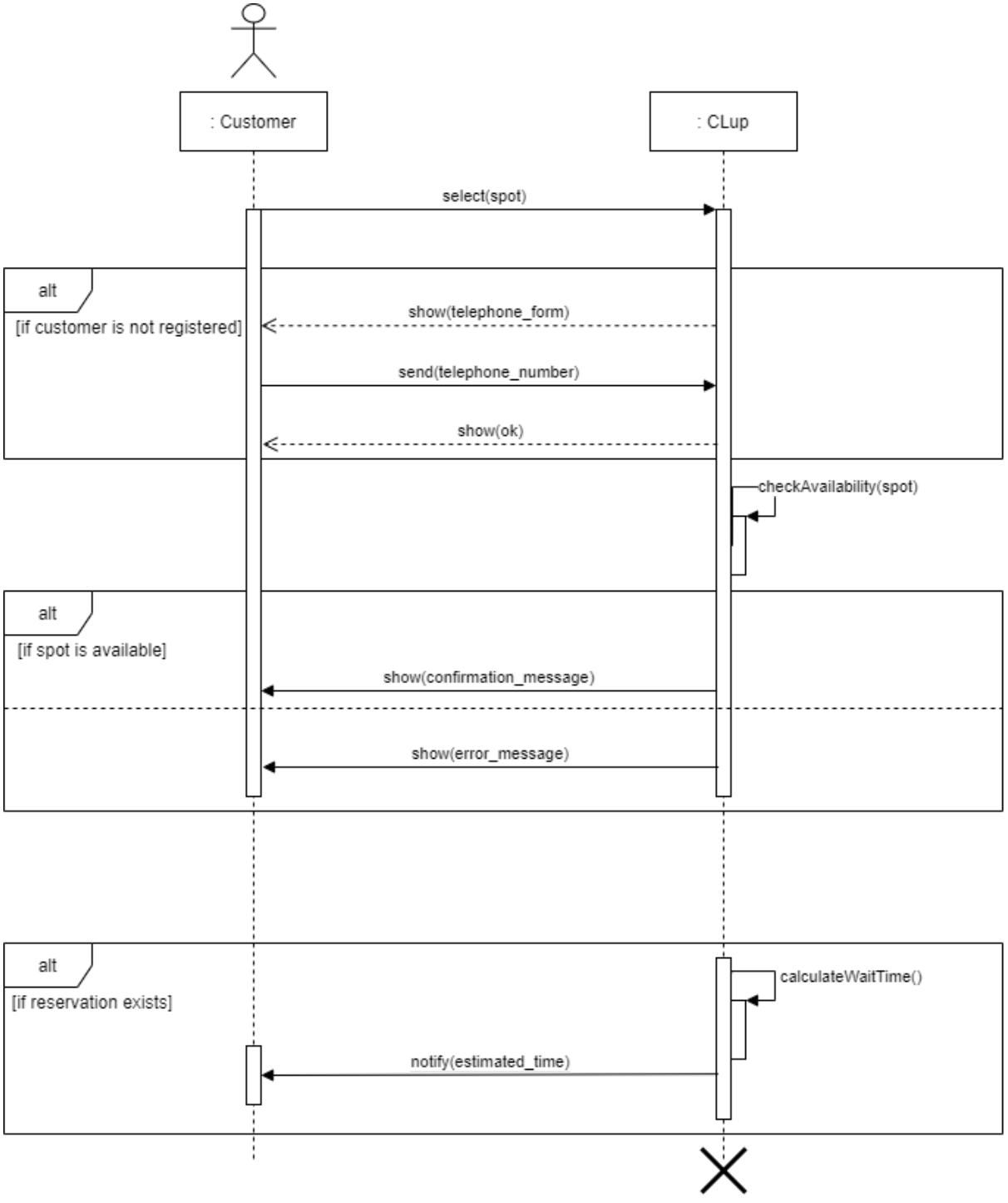
B.3.2. A customer learns about the current grocery's line-up number



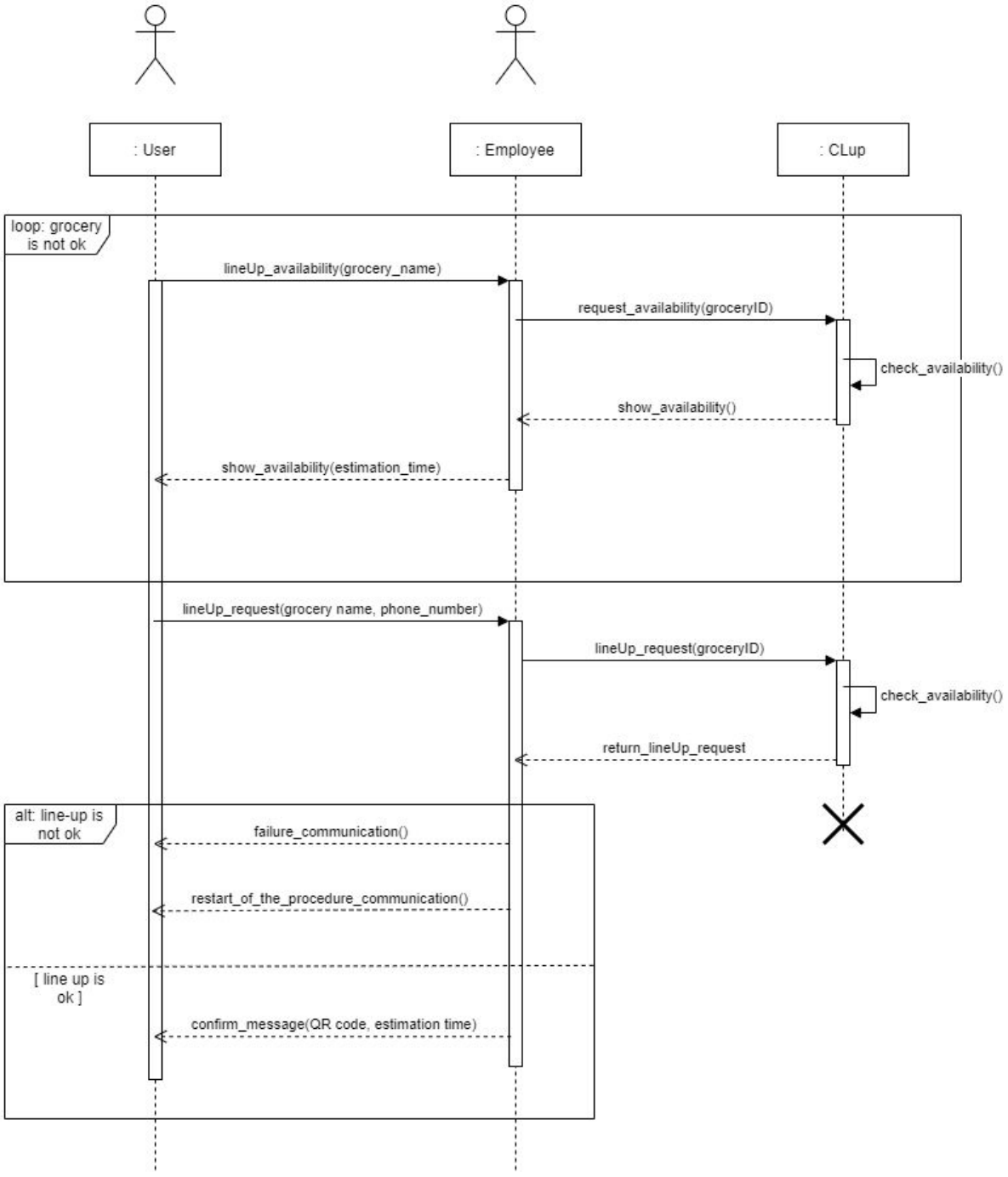
B.3.3. A customer watches the available spots in a given supermarket



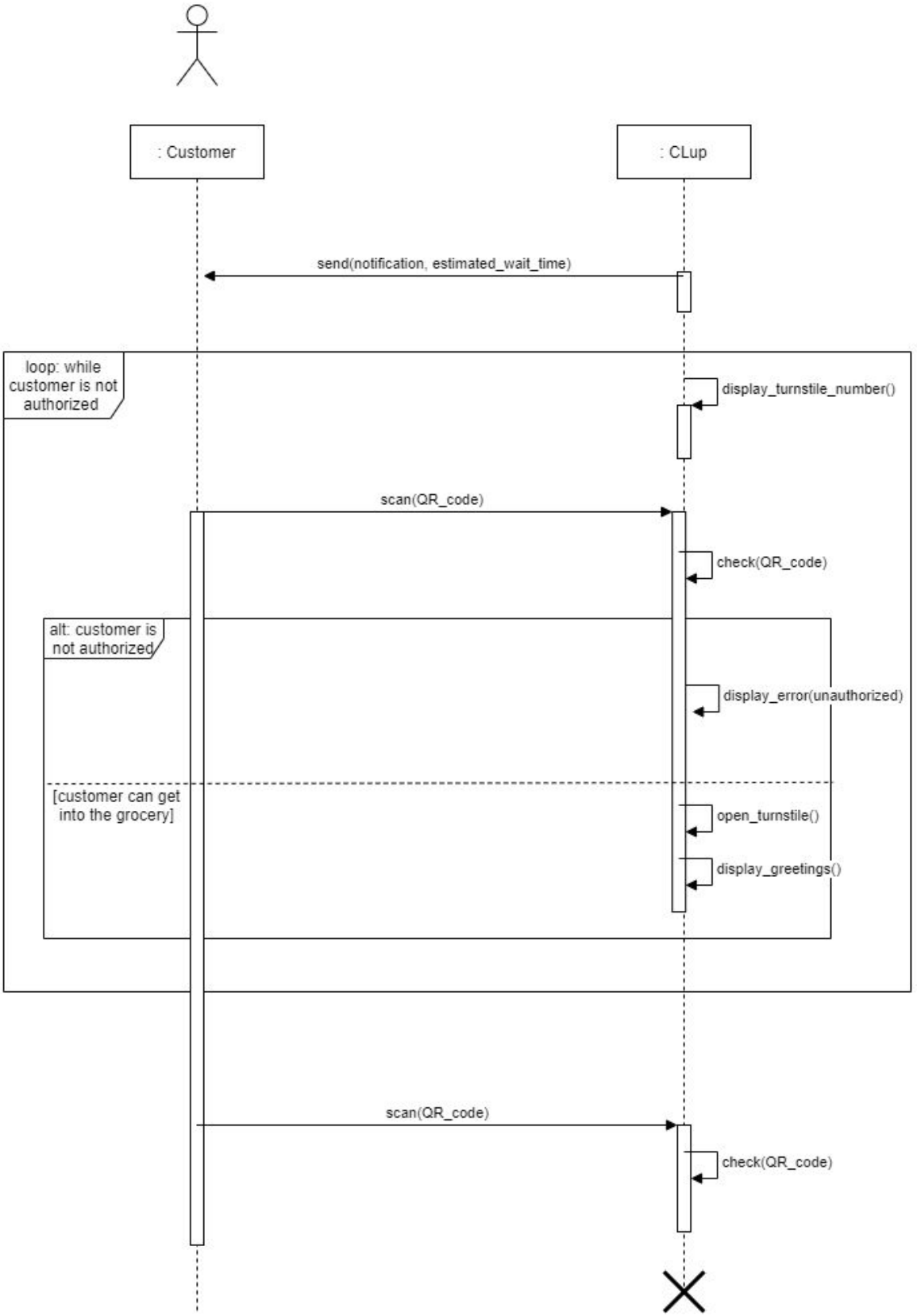
B.3.4. A customer books a visit to the grocery



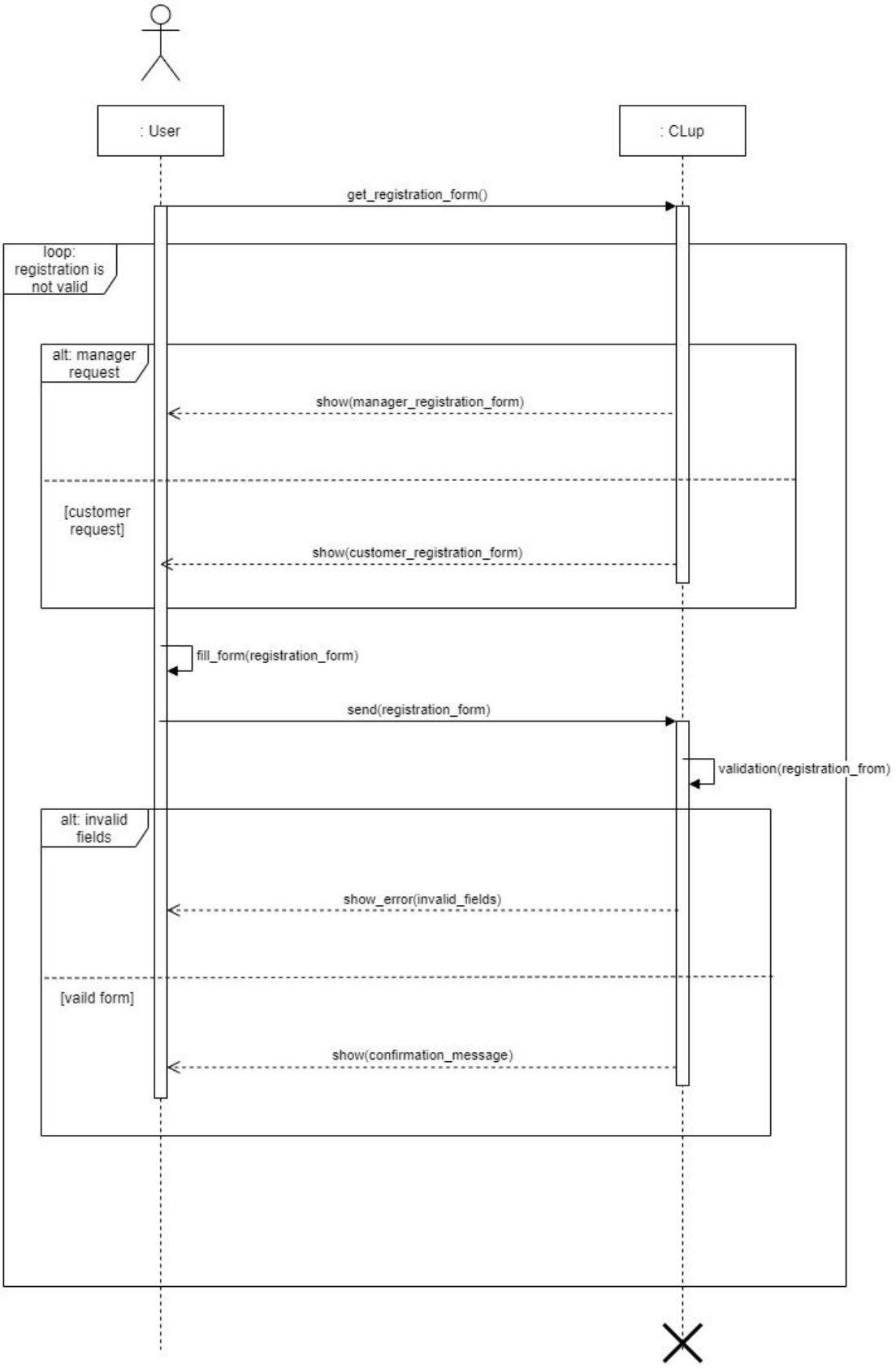
B.3.5. A customer uses a fallback option



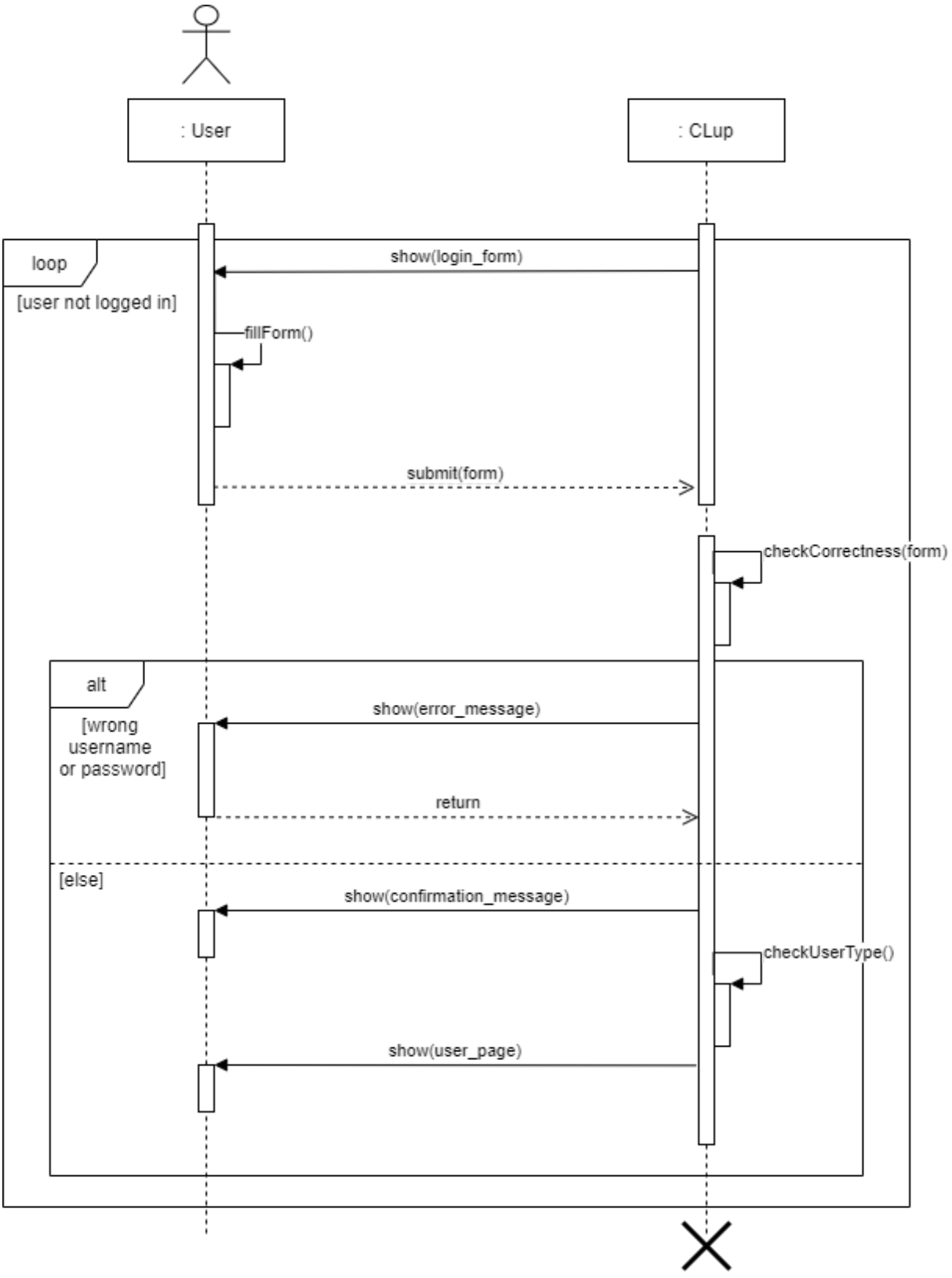
B.3.6. A customer gets into the grocery store



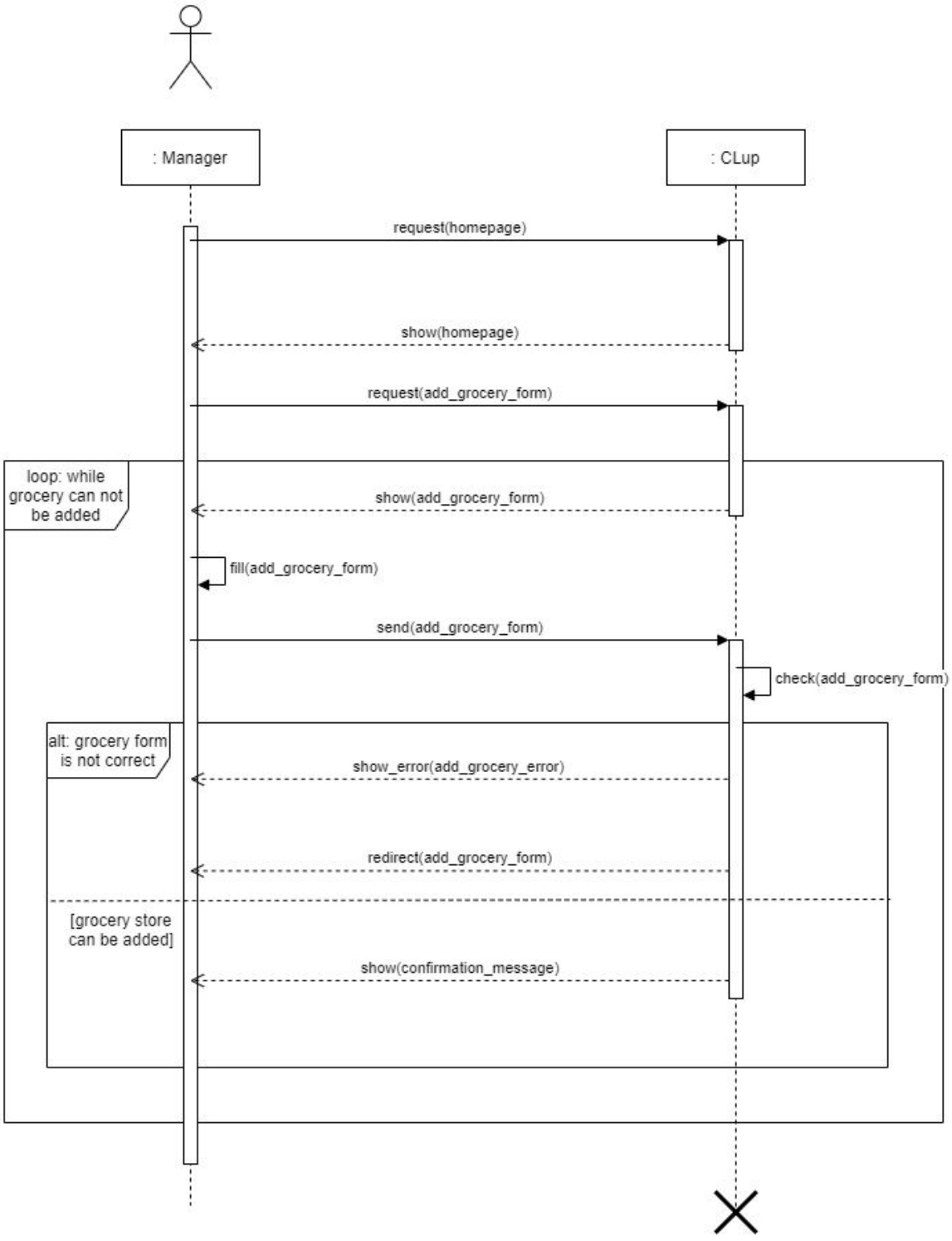
B.3.7. A user registers



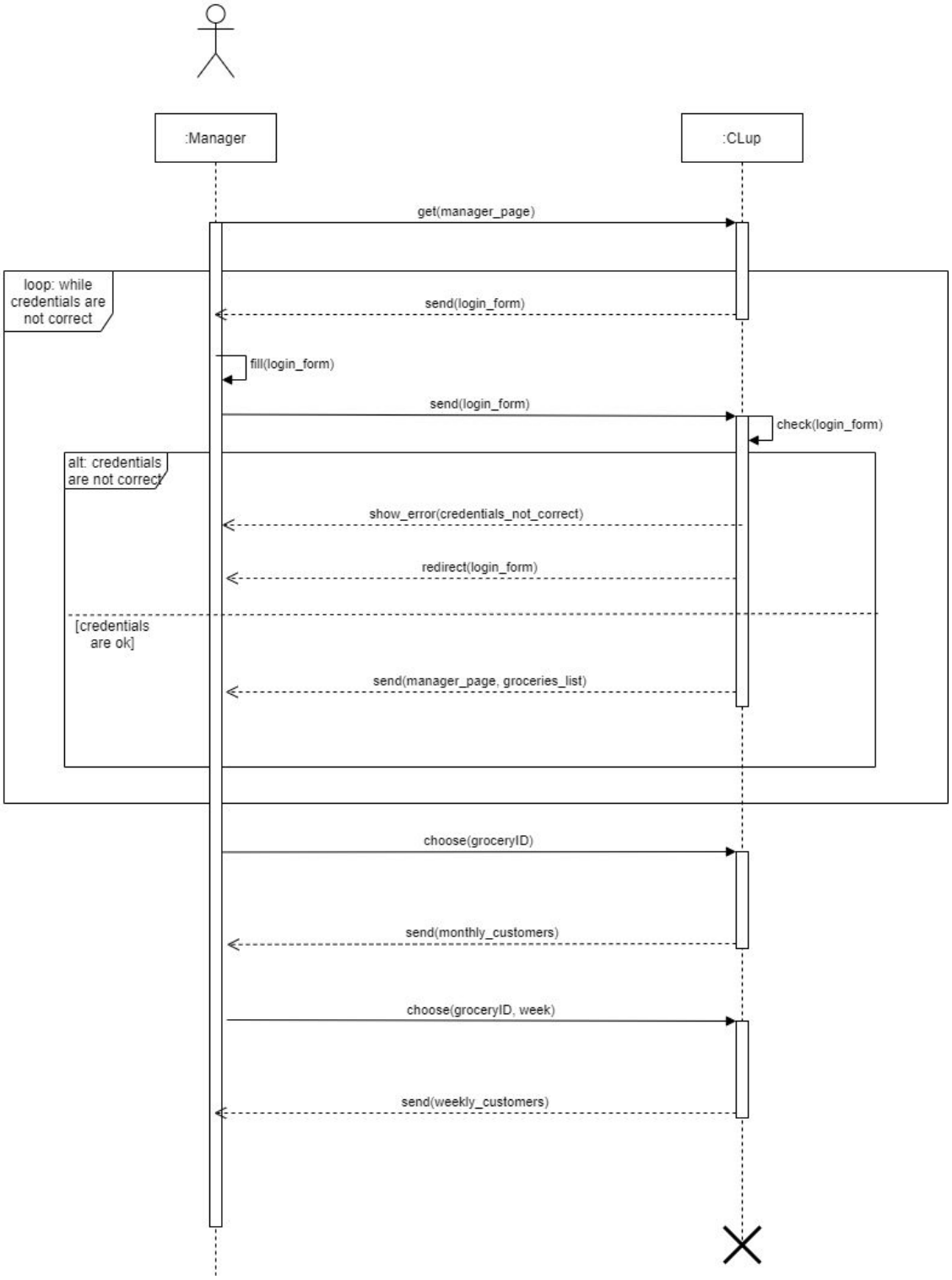
B.3.8. A user logs in



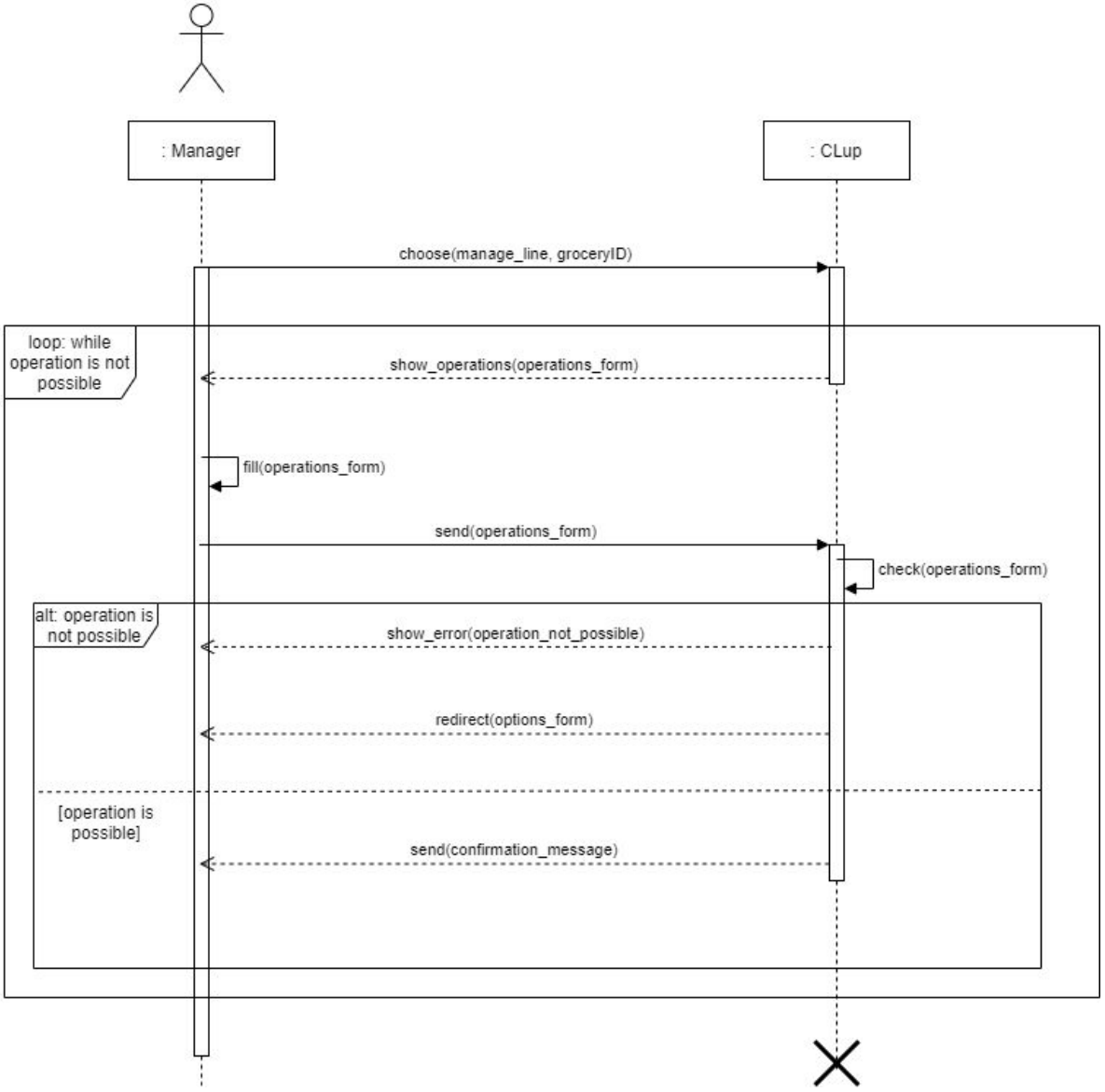
B.3.9. A manager registers a store as a manager



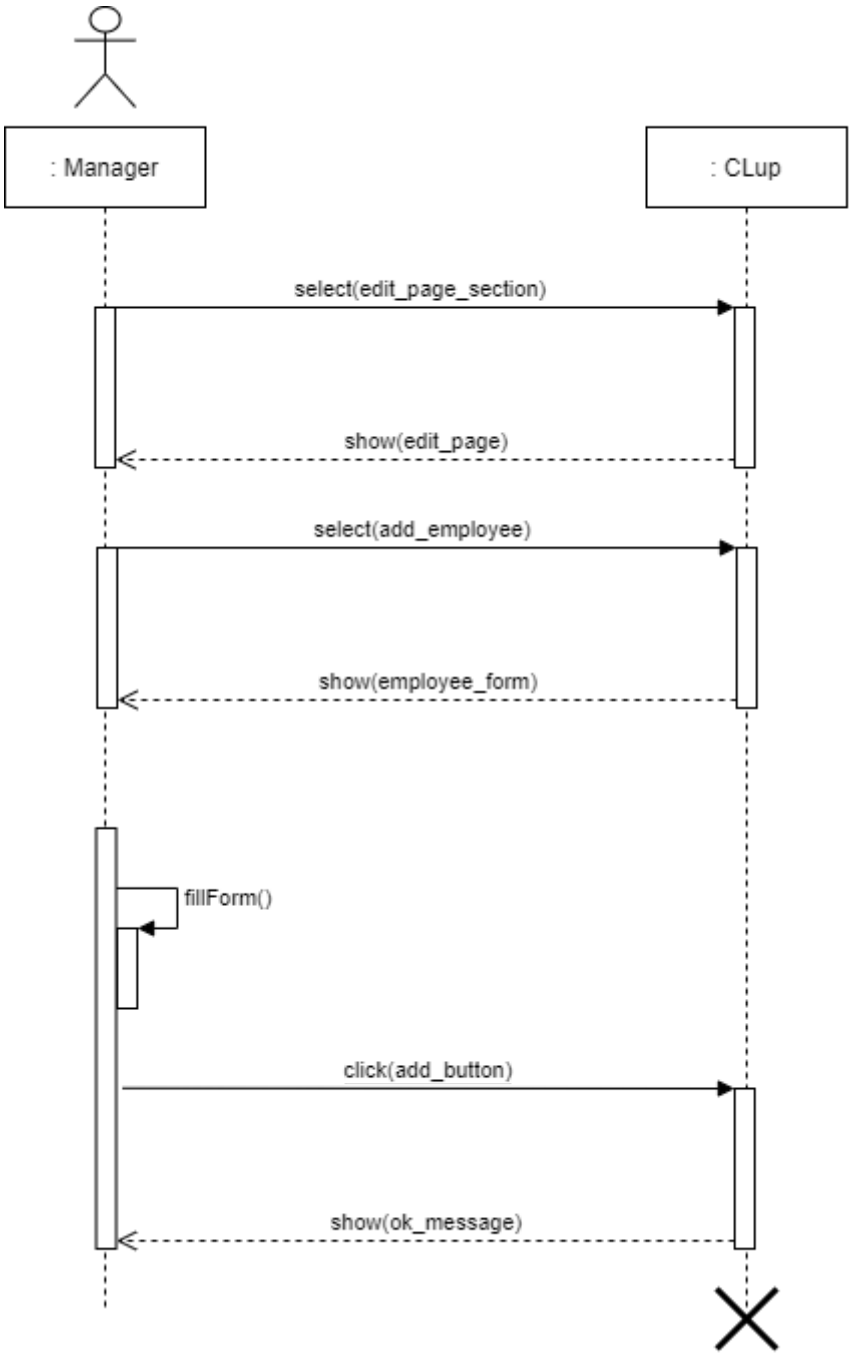
B.3.10. A manager retrieves the informations about the grocery's data



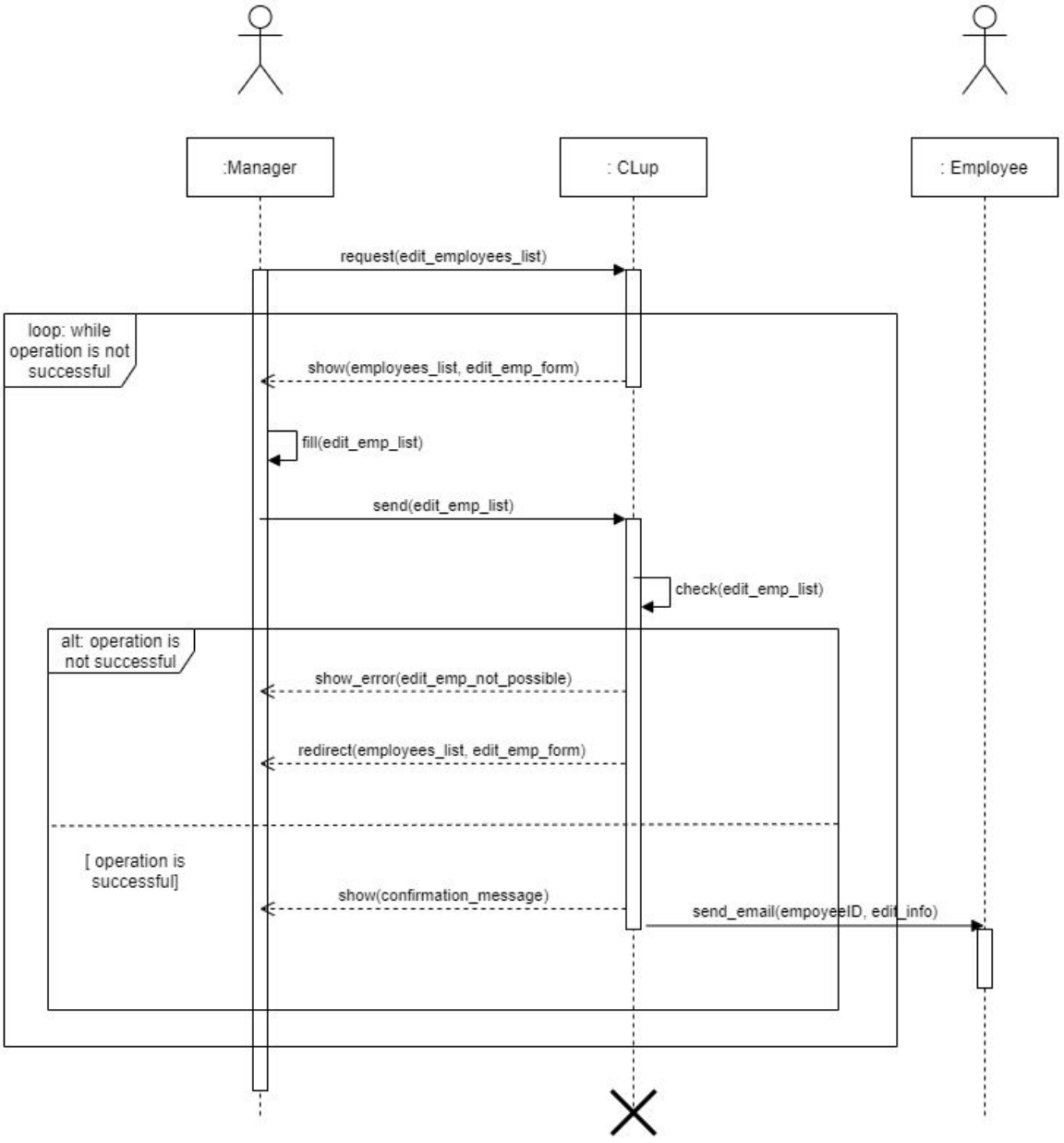
B.3.11. A manager handles the line



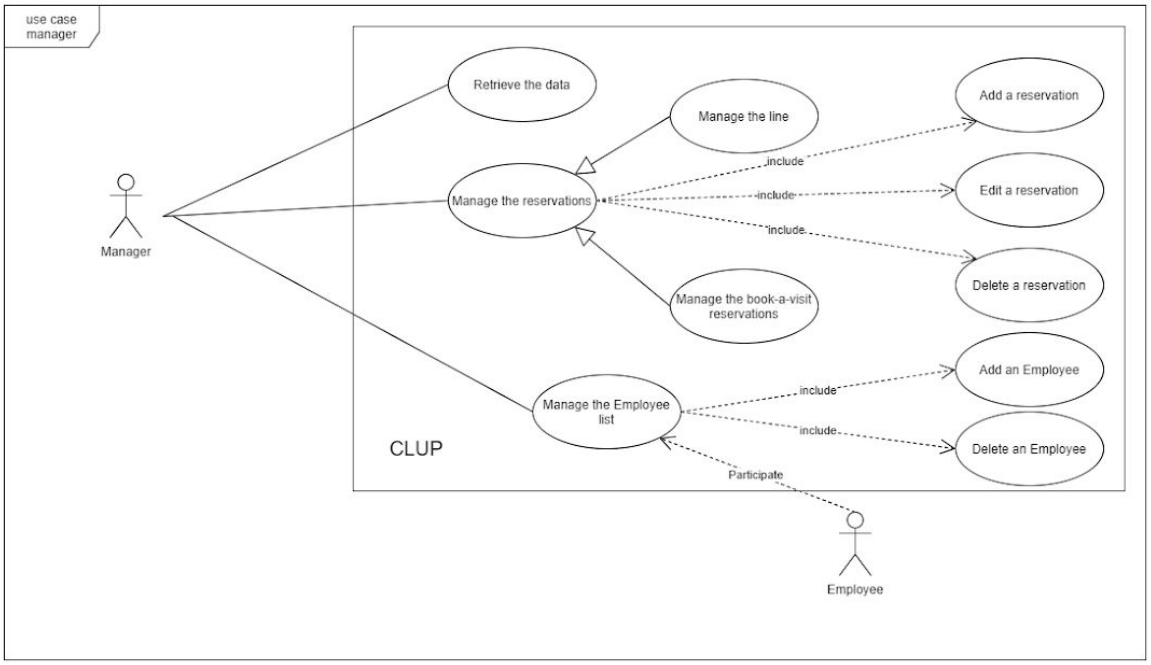
B.3.12. A manager adds an employee to the list of the store

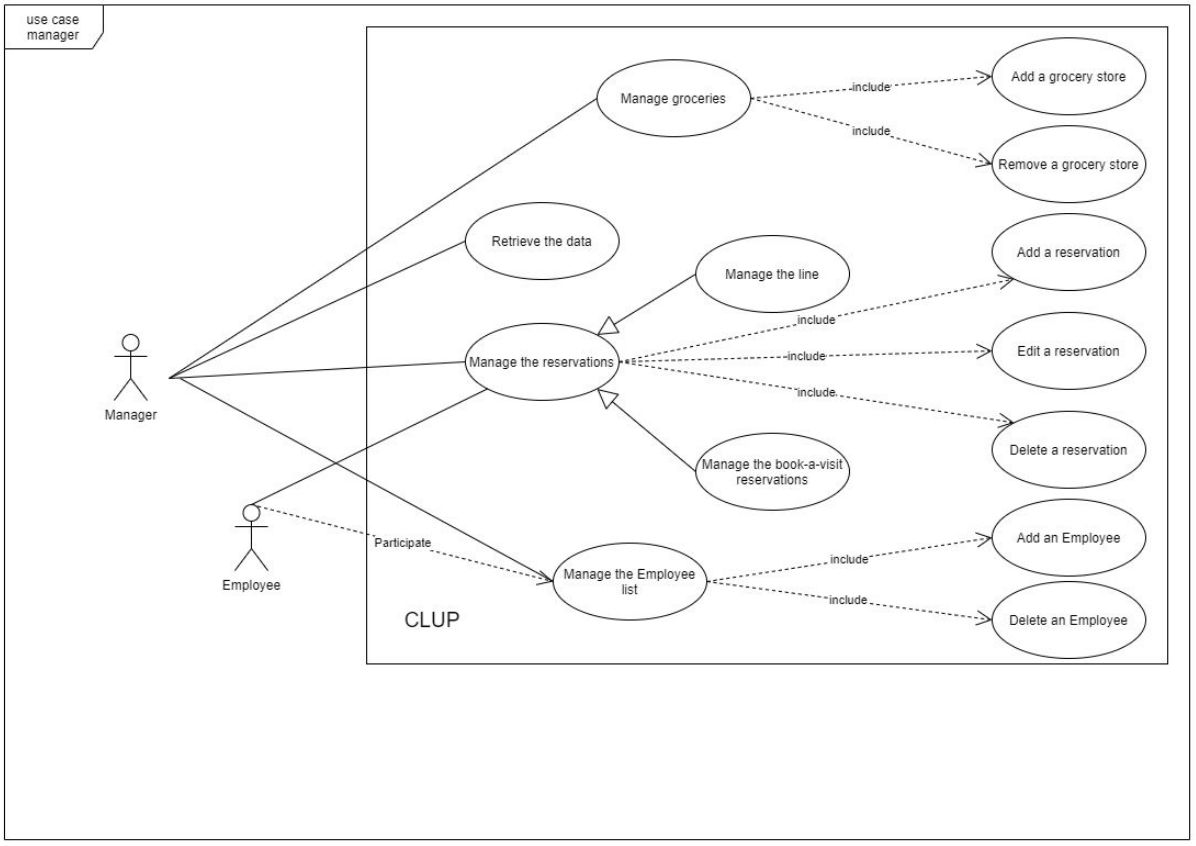


B.3.13. A manager deletes an employee from the list of the store


:Manager
: CLup
: Employee
request(edit_employees_list)
loop: while operation is not successful
show(employees_list, edit_emp_form)
fill(edit_emp_list)
send(edit_emp_list)
check(edit_emp_list)
alt: operation is not successful
show_error(edit_emp_not_possible)
redirect(employees_list, edit_emp_form)
[operation is successful]
show(confirmation_message)
send_email(empoyeeID, edit_info)

B.4 Use cases' sequence diagrams





B. Performance Requirements

In this section we will include performance requirements of the software-to-be. These can be summarized in the only wide constraint that all the services provided by the software-to-be have to be provided in the most rapid time possible, so that the user can have a real-time perception of the product.. As a threshold we can set 5 seconds: all the product functions ([2.B]) have to be provided in no more than 5 seconds.

C. Design Constraints

E.1 Standard compliance

The code should follow the requirements contained in this document. Furthermore, it should use, whenever possible, design patterns in the development to allow better maintainability and readability.
Data and information must be stored and must be persistent during the whole life of the service.

E.2 Hardware limitations

The system can be developed with any hardware in order to satisfy the requirements and goals described in this document.

E.3 Any other constraints

Since the software development team will be composed by students, there will be taken into account the fact that the project could not be developed with a full-time participation and that the technical skills, as well as the experience, could be limited.

D. Software System Attributes

E.1 Reliability

The system must guarantee a high level of reliability, and every maintenance need or routine must be done during night hours because that's a time when the system will not be of use to anyone.

E.2 Availability

The system must be available at any time, for people to make use of it without lacking an essential part of everyone's life such as grocery shopping.

E.3 Security

Users' information will be stored safely in the database and never used for any purpose not related to the service itself.

E.4 Maintainability

The system must be written in java and guarantee a high level of maintainability. Code must be written with a high level of abstraction and without hard-code as well. Code must be written with large use of comments that cover all aspects of the code itself to improve readability.

E.5 Portability

The software must run in different platforms like Windows operating system, Linux operating system and Mac operating system. Also, must support Android and iOS operating systems for mobile devices.

4. FORMAL ANALYSIS USING ALLOY

In this chapter it is presented a formal representation of the main functionalities of the application. This representation is provided with some assumptions done to better analyse the model in its main functionalities. Indeed, it is modelyzed the reservation modality, including the “line-up” and the “book-a-visit” features. In this domain, there are made some assumption:

- This description does not include the formal representation of time management: it is assumed that the estimation time provided and described in the previous chapters is present and coherent with the time estimation yet described.
- It is assumed that when the Customer scans his personal QR code after that the system has provided the allowance to him, he will get into the store and then scan again the QR code in order to get out from the grocery.
- The formal analysis does not include that there are a fixed number of spots to occupy with “book-a-visit” reservations, not linked with the “line-up” ones. Those spots are directly proportional to the maximum number of the people who can gain access into the store.
- The formal analysis does not include a “hardware” representation, it is assumed that the access turnstiles support successfully the entrance of the customers with the allowance to get into the store, denying successfully the entrance to those who have not such allowance.

4.1. Code

```
// --------------------signatures--------------------------
sig Coordinates {
    latitude: one Int,
    longitude: one Int
 }

sig QRCode { }

sig User {}
sig Username {}
sig RegisteredUser extends User {
    username: one Username
}
sig Manager extends RegisteredUser{ }

abstract sig Status { }

sig OPEN extends Status { }
sig ENTERED extends Status { }
sig CLOSED extends Status { }

abstract sig Reservation {
    contract: User -> Grocery,
    status: one Status,
    accessCode : one QRCode
 } {
   #contract = 1
}

sig LineUpRes extends Reservation { }

sig BookAVisRes extends Reservation { }

sig Queue {
    stack: seq LineUpRes
}

sig Grocery {
    location: one Coordinates,
    allowedEntrance: one Int,
    intoTheGrocery: set User,
    owner: one Manager,
    queue: one Queue,
} {
       #intoTheGrocery <= allowedEntrance
    allowedEntrance > 0
}

// --------------------functions--------------------------

// function to extract user of a reservation
fun getUserFromRes[r : Reservation] : one User {
        (r.contract).Grocery
}

// function to extract grocery from a reservation
fun getGroceryFromRes[r : Reservation] : one Grocery {
        User.(r.contract)
}

// -------------graphical utils facts-------------------

// to better undrstand the domain graphically

// no status not assigned
fact statusExistence{
    all s : Status | some r : Reservation | r.status = s
}

//every coordinate has a grocery
fact everyCoordHasAGrocery {
    all c : Coordinates | one g : Grocery | c = g.location
```

```
73  }
74
75  // --------------------facts----------------------------
76
77  // definitions of statuses behaviour and logic
78
79  // ENTERED status
80  fact enterStatusDef {
81          all r: Reservation | (r.status = ENTERED) iff
82              (getUserFromRes[r] in getGroceryFromRes[r].intoTheGrocery  and
83                  (no r1 : Reservation | r1.accessCode != r.accessCode and
84                      getUserFromRes[r] = getUserFromRes[r1] and
85                          r1.status = ENTERED)
86              )
87  }
88
89  // OPEN status
90  fact openStatusDef {
91      all r: Reservation | (r.status =  OPEN) implies
92          (getUserFromRes[r] not in getGroceryFromRes[r].intoTheGrocery)
93      all lr : LineUpRes |  lr.status = OPEN iff
94          one q : Queue | one g : Grocery |
95              (getGroceryFromRes[lr] = g and lr in q.stack.elems and g.queue = q)
96  }
97
98  // CLOSED status
99  fact closeStatusDef {
100         all r: Reservation | (  r.status = CLOSED ) implies
101             (getUserFromRes[r] not in getGroceryFromRes[r].intoTheGrocery)
102 }
103
104 // all reservations in a queue are OPEN
105 fact openStatusResInAQueue{
106     all q : Queue | all r : Reservation |
107         r in q.stack.elems implies r.status = OPEN
108 }
109
110 // all users into a grocery have a correct reservation
111 fact userInsideHaveCorrectRes {
112     all u : User | all g : Grocery | u in g.intoTheGrocery iff
113         one r : Reservation | (u = getUserFromRes[r] and
114             getGroceryFromRes[r] = g and r.status = ENTERED )
115 }
116
117 // no Manager can be into any grocery
118 fact noManagerCanBeInsideAGrocery{
119     all g : Grocery| all m : Manager | m not in g.intoTheGrocery
120 }
121
122 // ensure that a registered user has a different username from the others
123 fact uniqueUsername {
124     all u1: RegisteredUser, u2: RegisteredUser | (u1 = u2) iff u1.username =
        u2.username
125 }
126
127 // ensure that a reservation has a different access code from the others
128 fact QRcodeIsUnique {
129     no disjoint r1, r2: Reservation | r1.accessCode = r2.accessCode
130 }
131
132 // ensure that a coordinate has longitude and latitude different from the others
133 fact coordinatesDefinition {
134     all c1: Coordinates, c2: Coordinates | (c1 = c2) iff
135         (c1.latitude = c2.latitude and c1.longitude = c2.longitude)
136 }
137
138 // ensure that a grocery has a location different from the others
139 fact uniqueLocation {
140     all g1: Grocery, g2: Grocery | (g1 = g2) iff g1.location = g2.location
141 }
142
143 // ensure that a grocery has a queue different from the others
```

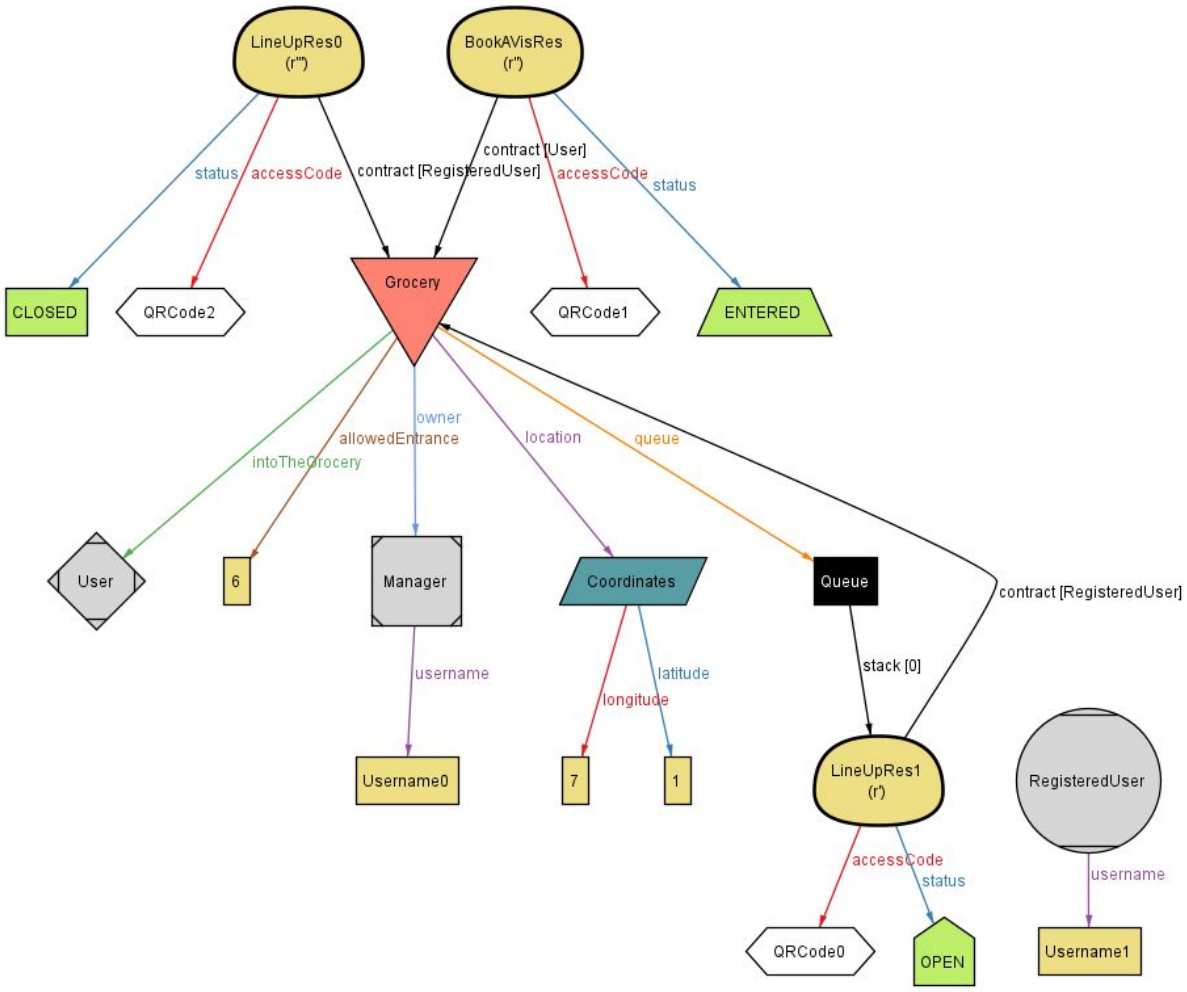
```
fact uniqueQueue {
    all q: Queue, g1: Grocery | q in g1.queue iff
        no g2: Grocery |  g1 != g2 and q in g2.queue
}

// ensure no duplicates in the queues
fact enqueuedElementsMustBeUnique {
    all q: Queue | not q.stack.hasDups
}

// all reservations are enqueued if open
fact allResAreInQueueOrClosed {
    all r : Reservation | one q : Queue | r in q.stack.elems iff
        not ((r.status = CLOSED or r.status = ENTERED ) and
            q = getGroceryFromRes[r].queue)
}

// every queue is owned by a grocery
fact allQueuesAreInAGrocery {
    all q : Queue | q in Grocery.queue
}

// managers can't make a reservation
fact managerCantMakeAReservation{
    all r : Reservation | getUserFromRes[r] not in Manager
}

// every manager is registered
fact managerIsRegistered {
    all g : Grocery | g.owner in RegisteredUser
}

// no duplicate users in every Queue
fact noDupsInAQueue {
  all q : Queue | all r1 : Reservation, r2 : Reservation |
    not (r1 != r2 and r1 in q.stack.elems and r2 in q.stack.elems and
        getUserFromRes[r1] = getUserFromRes[r2])
}

// --------------------predicates-------------------------

pred reservationBeginAndFinish {
    some r : Reservation | r.status = OPEN
    some r : Reservation | r.status = ENTERED
    some r : Reservation | r.status = CLOSED
}

pred usersCanBookAVisitToAGrocery {
    #BookAVisRes > 0
}

pred usersCanLineUpToAGrocery {
    #LineUpRes > 2
    some g : Grocery | #g.queue.stack.elems > 1
}

pred peopleCanEnterInTheGrocery {
    some g : Grocery | #g.intoTheGrocery > 1
}

pred show {
    #Grocery = 2
    #Reservation > 2
}
```

4.2. Results

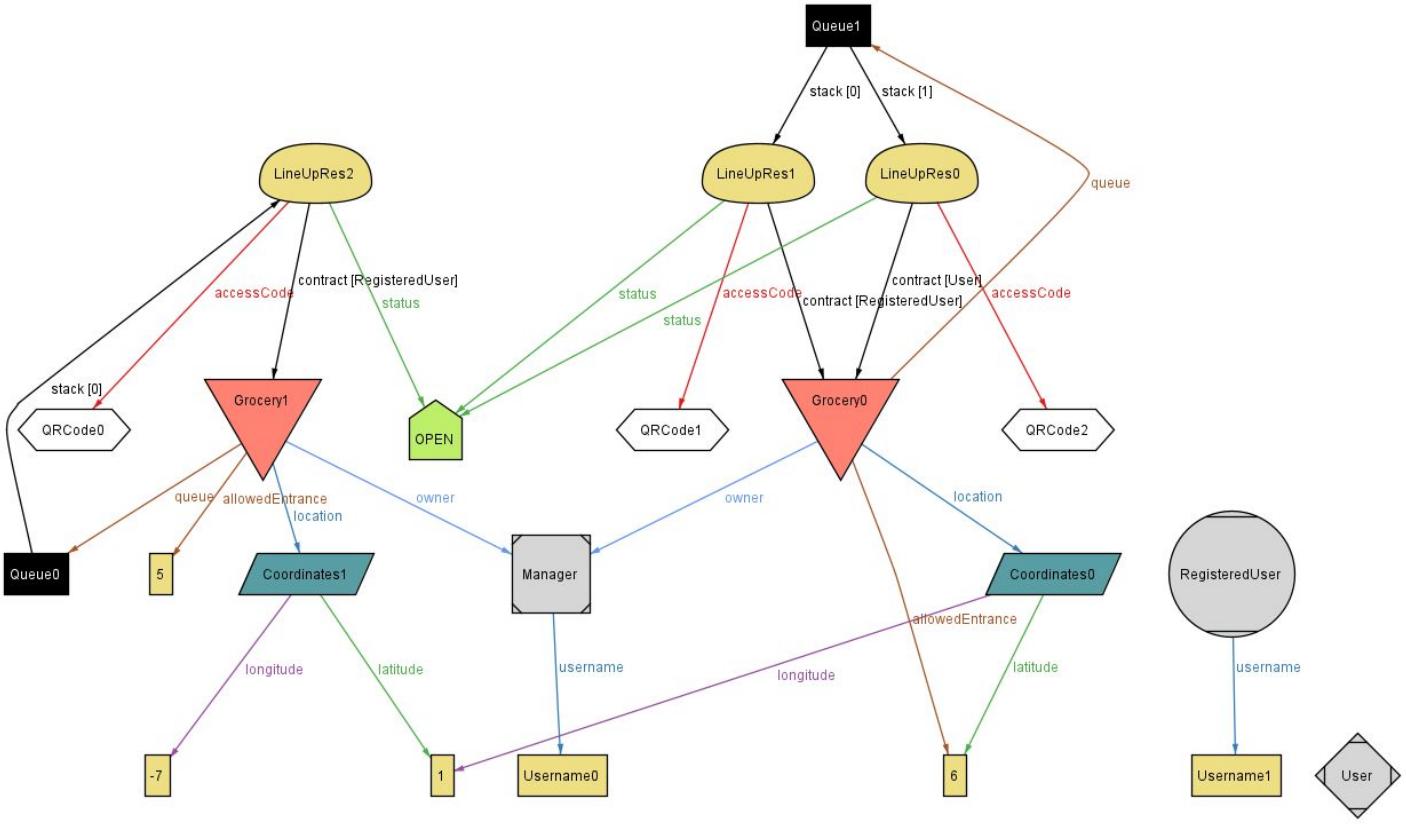
```
Executing "Run reservationBeginAndFinish"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4415 vars. 327 primary vars. 10205 clauses. 12ms.
  Instance found. Predicate is consistent. 7ms.

Executing "Run usersCanBookAVisitToAGrocery"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4292 vars. 318 primary vars. 9960 clauses. 13ms.
  Instance found. Predicate is consistent. 8ms.

Executing "Run usersCanLineUpToAGrocery"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4391 vars. 321 primary vars. 10240 clauses. 21ms.
  Instance found. Predicate is consistent. 10ms.

Executing "Run peopleCanEnterInTheGrocery"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4329 vars. 321 primary vars. 10041 clauses. 16ms.
  Instance found. Predicate is consistent. 25ms.

Executing "Run show"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4297 vars. 318 primary vars. 9984 clauses. 15ms.
  Instance found. Predicate is consistent. 11ms.

5 commands were executed. The results are:
  #1: Instance found. reservationBeginAndFinish is consistent.
  #2: Instance found. usersCanBookAVisitToAGrocery is consistent.
  #3: Instance found. usersCanLineUpToAGrocery is consistent.
  #4: Instance found. peopleCanEnterInTheGrocery is consistent.
  #5: Instance found. show is consistent.
```

4.3. Generated world

a. Run reservationBeginAndFinish



b. Run show


Queue1
stack [0]
stack [1]
LineUpRes2
LineUpRes1
LineUpRes0
queue
accessCode
contract [RegisteredUser]
status
status
accessCode
contract [User]
contract [RegisteredUser]
accessCode
stack [0]
Grocery1
Grocery0
QRCode0
OPEN
QRCode1
QRCode2
queue
allowedEntrance
location
owner
owner
location
Queue0
5
Coordinates1
Manager
Coordinates0
RegisteredUser
allowedEntrance
longitude
latitude
username
longitude
latitude
username
-7
1
Username0
6
Username1
User

5. EFFORT SPENT

Duilio Cirino:

Day	Hours spent	Brief description
25-10	1	Wrote the scope of the project and annotated minor things
28-10	2	General group discussion on the job already done and on scheduling and future tasks
29-10	1,30	Began to structure the product functions paragraph, also thinking about certain characteristics of each function
30-10	1,15	Almost finished writing the product functions paragraph
4-11	2	General group discussion on the job already done and on scheduling and future tasks
5-11	1	Continued product functions, adding one more subsection and finalizing the others. Made an initial structure of the user characteristics paragraph and started writing it
7-11	2	Finished product

		functions and almost finished user characteristics
10-11	2	Finished user characteristics and began constraints and domain assumptions
11-11	2,30	General group discussion on the job already done and on scheduling and future tasks
13-11	1	Finished constraints and setup domain assumptions
16-11	1	Finished domain assumptions and started working on the mockups of the interface
17-11	2	Finished user interfaces
20-11	0,45	Added definitions and made statements about clarity and goals. Domain assumptions adjusted.
21-11	1	Adjusted mockups in user interfaces and added design constraints and software system attributes
23-11	1,30	Finished domain assumption mapping and began use cases diagrams
25-11	1,30	General group

		discussion on the job already done and on scheduling and future tasks
27-11	2,30	Finished sequence diagrams
2-12	1,30	General group discussion on the job already done and on scheduling and future tasks (Alloy)
11-12	1,15	General group discussion on the job already done and on scheduling and future tasks
5-12/16-12	10	Alloy modeling

Lorenzo Cocchia:

Day	Hours spent	Brief description
25-10	1,30	Specific analysis of the document in order to find the goals, goals description
26-10	1,40	Started thinking about some scenarios: scenarios a, b and c.
28-10	2	General group discussion on the job already done and on scheduling and future tasks
1-11	3	Product perspective finished, including class and state diagrams and

		scenarios editing
4-11	2	General group discussion on the job already done and on scheduling and future tasks
8-11	2	External Interfaces Requirements exploited (not UI)
10-11	0,30	Exploited communication interfaces
11-11	2,30	General group discussion on the job already done and on scheduling and future tasks
11-11	3	Punctualizations on external interfaces requirements (not UI), functional requirements
16-11	1,30	Begin of use cases definition
18-11	2,30	Use cases definition finished
22-11	2,30	Update of system functional requirements, use cases UML (1, 5, 6)
23-11	1	Continued sequence diagrams (8, 9, 11)
23-11	1,30	Fixed some problems related to the use cases (added a new 6,9 use case)

25-11	1,30	General group discussion on the job already done and on scheduling and future tasks
30-11	1,30	Update of some sequence diagram (1-6-9) and first skeleton of the Alloy model in a different document
2-12	1,30	General group discussion on the job already done and on scheduling and future tasks (Alloy)
7-12	2	Alloy modeling
11-12	1,15	General group discussion on the job already done and on scheduling and future tasks
12-12	1,45	General review
16-12	2	Final revision
18-12	2	Final revision - goal 1.8 added and GRD matrix created
18-12	2	Final revision - new state diagram, goals restyled, GRD matrix added

6. REFERENCES

- Google spreadsheet in which we discussed and edited the mapping between domain assumptions, goals and functional requirements.
- Google doc in which we discussed and edited the Alloy model