RASD

Software Engineering 2

Antonio Ercolani - 10621728

Vittorio Fabris - 10562731

Riccardo Nannini - 10626268

Academic year 2020/2021

Contents

1 Introduction

1.1 Purpose

The *Requirement Analysis and Specification Document* (RASD) is a standardized document used by software engineers to communicate the most important aspects of a system to be developed to the project stakeholders. The RASD represents the starting point of the so called "waterfall life-cycle" of a software system, therefore it's easy to understand how delicate and strategical is the role it performs. Programmers, testers, designers and everyone involved in the development will refer to this paper and a mistake could potentially cost thousand hours of additional work.

What is said so far is the purpose of this document. This specific RASD is about an application that tries to face the crowding of supermarkets during the coronavirus emergency. In the following section we are going to explain the scope of that software system.

1.2 Scope

The scope of the *CLup* software is to give the users the possibility to queue for the into a grocery store. The stores that have registered to this service generate and distribute tickets to line people correctly, letting them do the shopping only if the QR-code scanned on their ticket is valid.
The software offers different kind of functionalities and features. The main ones are the possibility to line up, book a visit according to the personal commitments and the availability to see suggestions to book a visit in a store that at a certain time during the day would be the best, so that the most overcrowded ones are avoided.

The user can line up in different ways: through a physical ticket taken outside the store, or through the application, that is implemented in a way such that people of every age can use. If there are already too many people queuing outside the shop, the software doesn't allow to distribute new physical tickets.
The software can integrate the information derived from the two sources, virtual and physical bookings, so that overcrowding is avoided in the neighborhood of the store and thanks to the QR ticket validations store managers can monitor entrances and exits. When the user leaves the store, he must show his ticket to the QR-reader again, so that the system can know it and let the queue proceed: another customer will then enter the shop.

On the other hand, users that line up from home concede the system to see where they are, using the GPS position of their device to identify their zone. In this way only nearby grocery stores are shown to the user.
What's more, these categories of users can book a visit at a specific time, if there is a free spot in the schedule organized by the system, otherwise other arrival times or grocery stores are proposed to the user. The customer to be can also chose to let the application show him some suggestions, providing some hints about the choice of the time and the possible alternative stores that have a higher availability of "free safe space" to do shopping.
The system, knowing the position of the user, his distance from the store and the vehicle that he has chosen to use to reach it, will provide him a notification that in a certain moment he should leave to arrive on time at the grocery store. If the user arrives late, the system can reschedule a booking for him, always considering avoiding overcrowding and the time preferences of the customer, inserting him into the line again.
What's more, because the system needs to balance in the best way possible the number of people inside the store, the user that books online his visit is also asked which kind of products he's going to buy. In this way, the system can manage people along the store's aisles, reducing the possible interaction and proximity of customers, distributing their access into the shop in different times if it is necessary. Store managers can increase the amount of people shopping inside the grocery store according to the fact that people are sufficiently distant one with respect to the other and the store is approved to do so by the law, without exceeding the customer's threshold declared when the store has enrolled in the system.
The grocery store also saves the information about the preferences of customers about the products they buy and their habits. In fact, the user lets the system know what he intends to buy and how he reaches the store. The user of the application can see his preferences and statistics too. For example, the store could use these preferences in order to manage a plan for the refill of its shelves and always be ready to satisfy the needs of its customers, avoiding the risk of not having some products to sell, but ordering them from their providers in advance.

1.2.1 World phenomena

WP1	The customer arrives to the store
WP2	The customer shops in the store
WP3	The customer waits outside the store
WP4	The customer waits at home

1.2.2 Shared phenomena

Controlled by the **machine**

SP1	The customer is notified that his turn is coming
SP2	The system generates and sends a ticket to the user
SP3	The ticket machine emits the requested ticket

Controlled by the **world**

SP4	The customer enters the store scanning his QR code
SP5	The customer reserves a place in the queue of a given store using the application
SP6	The customer reserves a place in the queue of a given store using the ticket totem outside the store
SP7	The customer books a visit to the store
SP8	The customer leaves the store scanning his QR code
SP9	The system receives a request to join a store's queue

1.2.3 Goals

G1	At any time the number of people in the store must not be higher than the store limit or the limit is passed but the customers are expected to be in different areas of the store
G2	The physical queue of people outside the store is reduced
G3	Allow users to "line up" for a store from home
G4	Allow users to book a visit to a store
G5	Balance the number of people between different stores or timeframes with suggestions
G6	Allow the store to monitor entries
G7	Collects information in order to build statistics and infer mean data
G8	Allow the store manager to temporarily increase the store capacity

1.3 Definitions, Acronyms, Abbreviations

In this section we explain the meaning of some technical terms used in the document.

QR CODE	A *Quick Response code* is a kind of bar-code, readable by machines to retrieve information
ASAP	An *As Soon As Possible* preference indicates that the customer wants to line up immediately, without booking a visit at a specific time to enter the store
UML	The *Unified Modeling Language* is a modeling language used to describe the design of a software system

1.4 Revision history

1.5 Reference Documents

1.6 Document Structure

This RASD document is structured over 6 main chapters, described below.

Chapter 1 starts with a brief description of the purpose of this document. Then the focus moves on the involved software system, for which is given an overall description of the goals, scope and related phenomena. The chapter ends with secondary sections about the description of technical terms and the history of this RASD document.

Chapter 2 helps understanding the "world" of the analyzed software system. It begins with a UML and then other kind of diagrams that explain how the application interacts with the environment around it. The following sections are about the requirements the system must satisfy, also with a focus on the specific users needs. Finally, in the last section, we list the domain assumption we did.

Chapter 3 goes deep into the topics described in the previous chapter. Begins with the description of the user interface and continue with use cases and sequence diagrams to show how the system behaves in different situations, in respect to the application requirements. Then you will find sections on other system constraints on performance and design. The chapter ends with few words on non-functional requirements.

Chapter 4 tries to describe the main aspects of the software system we have seen so far with an alloy model.

Chapters 5 and 6 contain, respectively, an outline of the effort spent in the project by each team mates and the document references.

2 Overall description

2.1 Product perspective

The CLup system is designed as a complitely new software application. The user side of the software is intended to be used as a mobile application while the store managers are going to interact with the system with a different interface.
In order to estimate the time needed for the user to reach the store, the application needs to interface with the smartphone's localization system and needs the user to indicate the type of transportation that he is going to use.
The system will handle client that are not using the application with tickets provided by a ticket totem located outside the store. This machine will act as an interface to the software application.
Customers using CLup are going to be notified when their turn is coming in order to approach the store while those with a ticket outside the store will monitor a screen extern to the shop in order to know when it's their turn to enter.
In order to access the store, every customer is required to scan their QR code (either from the application or from the physical ticket) when entering and exiting the store: in both cases the system will activate an automatic door.
On the other side of the system, store managers are going to be able to monitor and manage the queue of their store as well as programmed visits.
The below class diagram models the application domain in order to give an overlook on the enviroment where the application is going to work.

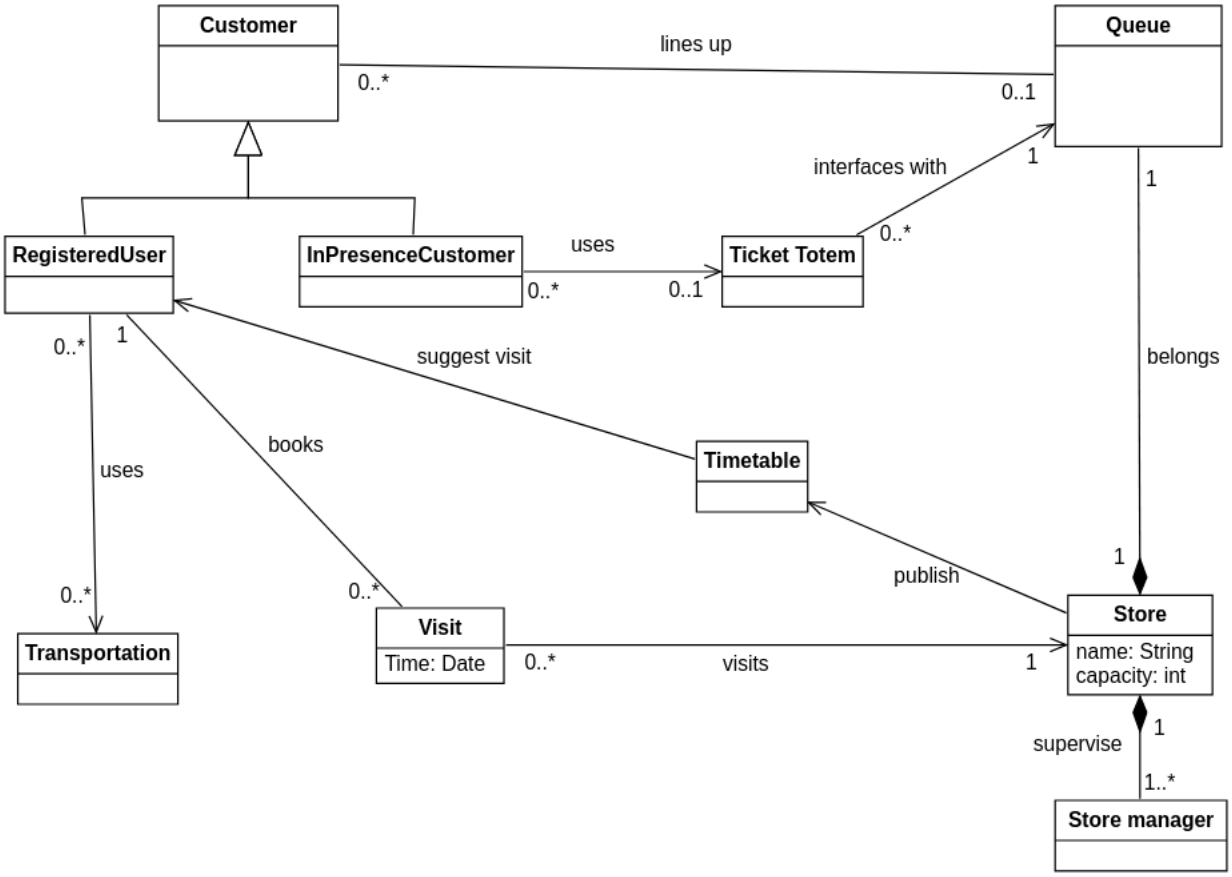

Figure 1: Class diagram

2.1.1 State Diagrams

In this section we want to show the evolution over time of the states of the application and the behavior of the user interacting with the app and the environment. The state diagrams below better explain the progress of the system.

In the first state diagram (Figure 2) it's shown how the user interacts with the environment and the system when he needs to do shopping. He interacts with the system lining up to a store in a virtual way using the mobile or web application, or queuing physically at the store he wants to visit. What's more, to enter the store, he needs to show the QR-code, and then he has to do the same when he exits, so that the queue can proceed.

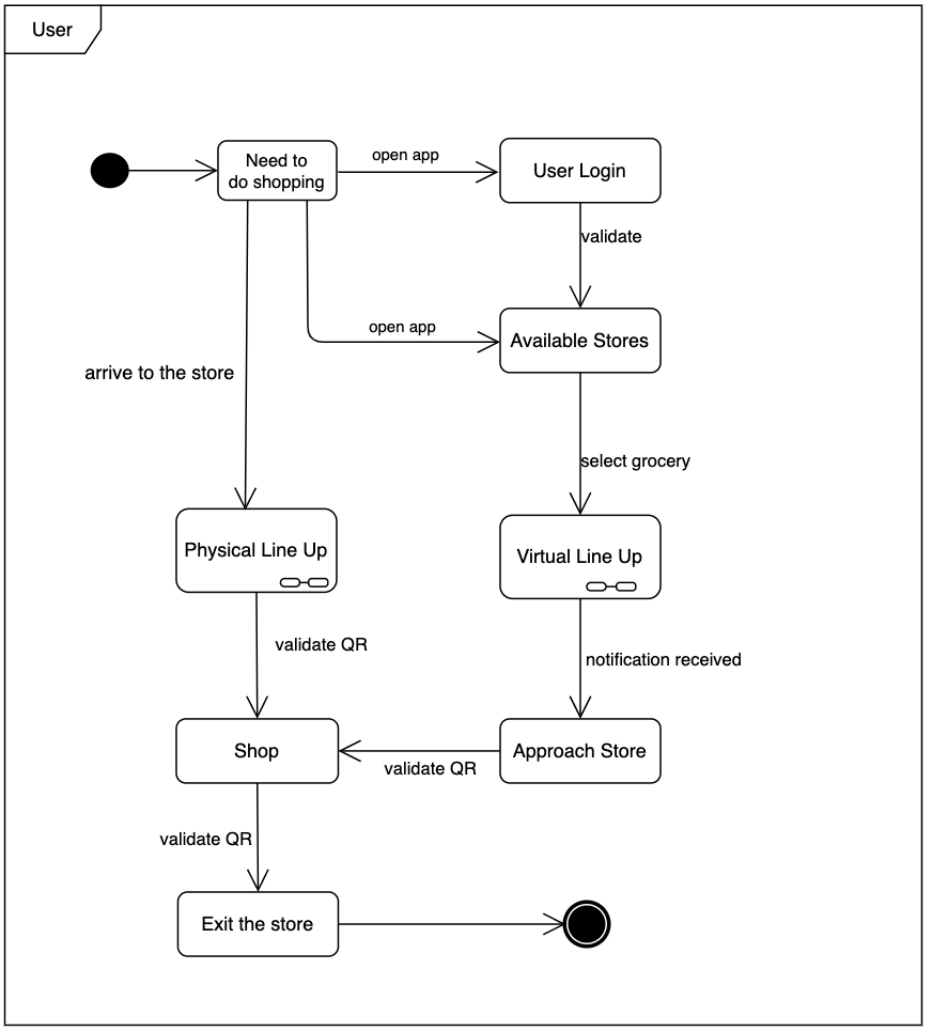

Figure 2: User State Diagram

In this second state diagram (Figure 3) it's better explained the physical line up of the user, that waits and directly gets a ticket from the totem nearby the store. The user then waits for his turn to enter and do the shopping.

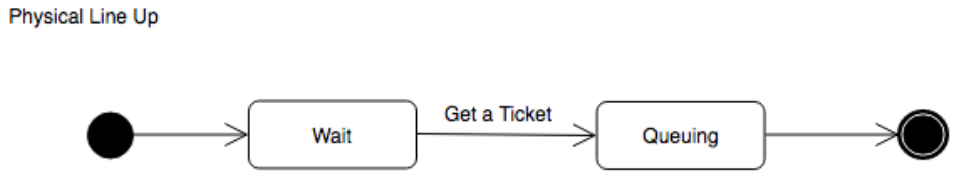

Figure 3: Physical Line Up State Diagram

Here (Figure 4) are shown the steps to go through the virtual line up of the user, that has to face some intermediate steps before gaining the possibility to line up to the store. When the store has been selected, the user needs to indicate the transportation he will use to reach the store. Then, if he doesn't want to line up immediately but wants to book a visit, he will see the available products that he can buy and will have to choose which one of them he wants to buy, so that the system can collect his preferences. Then the user is asked to indicate his permanence into the store, in order to better manage the queue and the next available spots in the timeline. It will also be possible to see some suggestions when booking the time of the visit, otherwise the whole available spots are displayed to the user in order to let him choose the time he prefers.

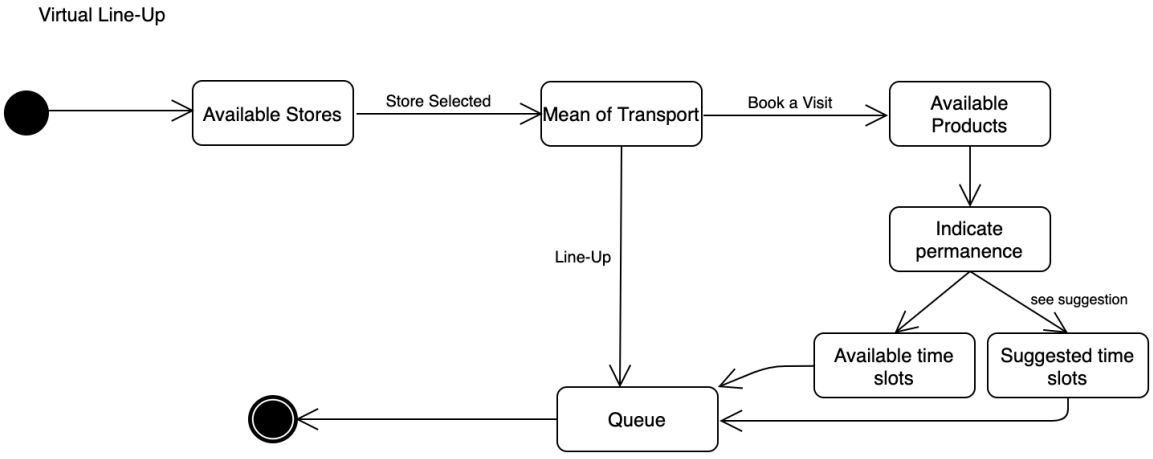

Figure 4: Virtual Line Up State Diagram

2.2 Product functions

In this section are analyzed the main functionalities that the CLup software system must ensure. Goals and requirements are many and they are all connected with each other, therefore, below you will find the description of two "macro-functionalities" that try to group most of them. Said that, in order to go deeply in every system capability, the reader should supplement this section with the next chapter - "Specific requirements".

Queue management

The core function. As the name suggest, the system is able to manage the stores entering queue. This is done with a combination of two sub-functionalities:

- **Line up from home** - allow users to reserve their place in the queue without having to reach the store, they just need to access the application and choose a grocery. To this simple user action corresponds an important data elaboration made by the system, to ensure another important functionality: **alert the costumer when his turn comes**. In fact the system must take into account different factors to alert the user in the right moment:
  - The distance between the user home and the selected store;
  - The means of transport by which the costumer intend to reach the store,
  - How much time take people before the user in the queue to enter the store.

  Once the system has collected all the information, it can compute the right amount of time needed by the costumer to reach the store in the right moment he have to enter in.
- **Line up physically** - the system must give the possibility to line up physically. Therefore, when costumers reserve their place directly at the store the system must collect also this data. Obviously, for them there is no need to compute any kind of time to enter in the store because they wait for their turn right outside.

Booking visits

The second macro-functionality is the possibility of **booking a visit** to a registered store. On the costumers side, this functionality could be similar to the previous one but the main difference lies on the side of the stores. In fact the information that the costumers provide to the system during the booking is essential to the next presented feature.

- **Increase the capacity of the stores** - the system should allow more people in the stores than the standard number. This is made possible by asking to the costumer who book a visit the categories of items he's intended to buy, therefore the system can infer which sectors of the store he's going to occupy. These information are exploited by the system to maximize the capacity of the stores, respecting the social distancing imposed by the emergency rules.

The second feature of the list is also directly linked to the "booking".

- **Suggest alternative booking time-slots** - the system should be able to suggest to the costumer, during the booking procedure, different time-slots or even different stores. These suggestions are computed in order to balance out the number of people among the registered CLup stores.

General store management

The last functionality allows store managers to visualize all the collected information about the store. For example, store managers can check how many customers are queuing outside the store, how many of them are doing shopping in a given moment and the booked visits of the following days.

2.3 User characteristics

The CLup software system involves category of users:

- **Costumers** - are client of the registered stores that want to take advantage of the CLup features. They are interested in the possibility of "line up" from home and in "booking visits", that encourage them to continue go shopping, despite the problems caused by the emergency.
- **Store manager** - these users are represented by the store managers that want to take advantage of the CLup services. They have to follow a registration procedure to be added to the available stores of the system.
  Besides exploit the "remote lining up" function that mainly face social distancing, they are interested in other features such as "increase the capacity of the stores" and "balance out the number of people in the store". In fact exploiting them they could are able to keep profits high, despite the emergency.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain Assumptions

D1	The GPS provides the exact location of the user
D2	The user is honest when he indicates: – How many time he's going to spend inside the store – What kind of products he's intended to buy – The means of transport he'll use to reach the store
D3	It is not possible for a costumer to enter or exit without scanning the QR code

3 Specific requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

In this section we present two mockup samples to give a general idea of the user interface. If the reader is interested in going deeply in the design of the UI, he will find several detailed mockups in the Design Document.

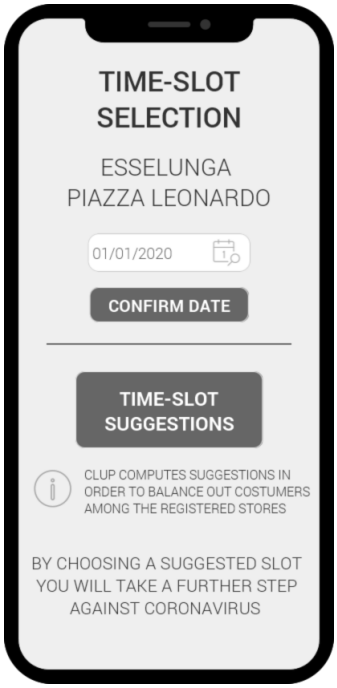

Figure 5: Booking visits screen

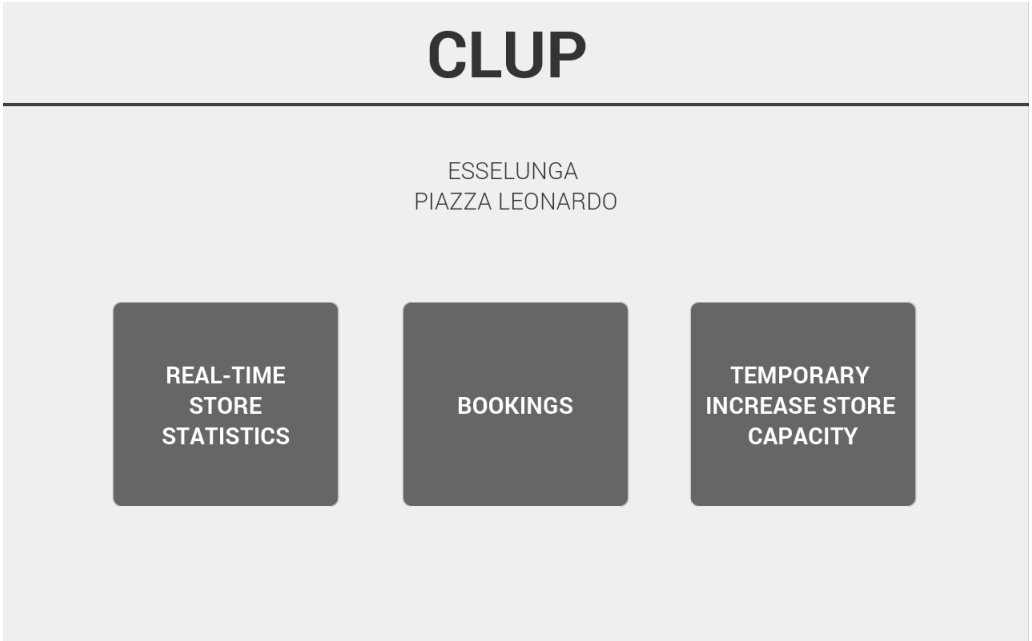

Figure 6: Store manager home screen

3.1.2 Hardware Interfaces

The system requires different external hardware components in order to work.
An **automatic door** needs to be present both in the entrance and the exit of the store. It's necessary in order to effectively regulate and monitor the flow of customers: the doors won't unlock unless the client specific QR Code is scannerized and accepted.
A **QR Code scanner** is required in correspondence of the automatic doors for the reason given above.
At least one **ticket totem** is needed in order to hand out tickets for customers not using the app.
In the end, one or more **monitors** are needed outside the store to let customers know when their turn has come.

In addition to this, the two type of actors require different hardware component in order to interact with the system.
Users need a smartphone with a localization system (tipically GPS), while store managers can access and monitor the system via computer.

3.1.3 Software Interfaces

The system needs an external software component to handle QR code generation and recognition.

3.1.4 Communications Interfaces

The various actors connect to CLup using an internet connection.
Hardware components such as automatic doors, ticket totem and QR code scanner interface with the system via IP based protocols.

3.2 Functional requirements

3.2.1 List of Requirements

R1	The user has to be logged in the application to be able to virtually line-up
R2	The user and manager applications are clear, intuitive and simple to use
R3.a	When costumers book a visit to the store they have to provide the approximate duration of the visit
R3.b	When costumers book a visit to the store they can provide a list of items they're going to buy
R3.c	When costumers book a visit to the store they have to provide mean of transport to reach the store
R4	The system provides the user a set of alternative available slots to book his visit
R5	The system calculates the availability of a store to let people enter in it in a specific moment
R6	The application can suggest alternative stores to visit in the neighborhood
R7	The system generates tickets with QR codes for the entrance/exit of a store
R8	The system alerts online costumers their turn has arrived with sufficient advance, taking into account the time they need to get the shop from the place where they currently are
R9	The system is able to send notifications to the logged user
R10	The system can provide the user a list of his favorite products when booking online a visit
R11	The system can infer the time of a visit of a user based on his stats and the products he has selected
R12	The system updates the queue of the store when a user enters or exits by making his QR code be read by the QR scanners
R13	The system is able to reschedule the queue in order to optimize it if a customer doesn't show up, allowing the users that want to go shopping as soon as possible to gain some time. Other users that booked a visit at a specific time won't see their booking vary
R14	The system waits for 5 minutes more to see if the late user arrives to the store, then invalidates his ticket and lets the queue go on
R15	The system detects the position of the logged user

R16	The system allows to delete a scheduled visit and booking
R17	The system shows to the logged user all the registered stores in a neighborhood of 5km range
R18	The system integrates online bookings and physical disbursed tickets by totems to manage the queue of a store
R19	The Application shows to the user how many people are remaining to approach/enter the store
R20	The system collects/stores product preferences, visits duration of the users
R21	The number of distributed tickets must not be higher tickets than the store specific threshold
R22	The system lets store managers to increase the store capacity accordingly to the store defined policy
R23	The system lets the manager know the areas of the store where customers are mostly likely going to collect products, also based on their stored preferences
R24	The system makes sure that if the customer wants to book a visit at the same time of another visit he has already booked it notifies the user about the overlapping dates

3.2.2 Mapping

GOALS	DOMAIN ASSUMPTIONS	REQUIREMENTS
G1	D3	R5, R7, R12, R13, R14, R18, R21, R22, R23
G2	D2	R3.a, R3.b, R3.c, R4, R5, R6, R8, R12, R13, R14, R18, R21, R22
G3	D1, D2	R1, R2, R5, R7, R8, R9, R12, R13, R14, R15, R16, R17, R18, R19
G4	D1, D2	R1, R2, R3.a, R3.b, R4, R5, R6, R7, R9, R10, R11, R13, R14, R15, R16, R17, R18, R20, R24
G5	D2	R4, R5, R6, R15, R17
G6	D3	R7, R12, R18, R21, R23
G7	D2	R3.a, R3.b, R3.c, R10, R11, R20, R23
G8	D2, D3	R3.a, R3.b, R5, R7, R11, R12, R22, R23

3.2.3 Scenarios

1. **User lines up from home**:
   Antonio is a young engineering student. As known, engineering requires a lot of effort and dedication in order to keep the pace of the various courses. He does not want to waste time in the queues outside groceries shop that became common in the pandemic times. Luckly he knows the CLup service and decides to line up for his favorite store from home.
   He inserts the type of transportation he is going to use in order to reach the store and joins the virtual queue while he can keep studying safely at home. Once his turn is about to come, he receives a notification from the application right on his smartphone and leaves the house

2. **Customer physically lines up**:
   Maria is an old lady living alone in her apartment in Milan. She does not own a smartphone nor has plan to buy one in the future: she is okay with her old style house phone.
   Maria goes weekly to the grocery shop: she directs herself to the ticket machine, retrieves a ticket and safely waits outside the store that his turn is called on the monitor. Once it is arrived, she simply scans her ticket's QR code and access the store.

3. **User books a visit**:
   Elena is a very busy mother of two. She has very limited time frames where she can shop in a grocery store and not miss other appointments in her schedule.
   Since Elena got to know CLup she loved the service of booking a future visit to a store. She just selects her store and favourite time slot, inserts a few data like estimated shopping time etc. and the magic is done. No more time spent outside the shop and appointment delayed.
   Once the selected day of the visit is arrived, she will receive a notification indicating that she should start to approach the store.

4. **User books a visit with CLup suggestions**:
   Giorgio is a retired old man, with a lot of free time. He knew the CLup application thanks to his grandchildren and now he uses it frequently. When Giorgio books his store visits he usually take into account the CLup suggested slots being aware that these suggestions are made to avoid crowding inside and outside the stores.
   In this way the CLup system gives to Giorgio the possibility to exploit his free time to play a role in the battle against coronavirus.

5. **User arrives late**:
   Riccardo, an enthusiastic user of CLup, received a notification from the system inviting him to reach the store where he previously lined up with the software.
   However, a car accident blocked the road and he finds himself complitely stuck in the traffic. He will eventually reach the store with a big delay and find out that his ticket is expired.
   He then decides to take another ticket from the ticket machine and wait for his turn.

6. **Store manager admints more people in the store**:
   Vittorio, the store manager of a grocery store that adopted CLup, is monitoring the situation of the store through the software interface.
   He receives a notification from the sytem indicating that, given the calculation of the software, there are the right conditions to admit more people and momentarily exceed the store limit.
   He consults his guide lines and decides to go on with the operation: he inserts the required data and confirms his choice.
   Shortly after the selected number of clients enter the store.

3.3 Use Cases

3.3.1 Use case diagram

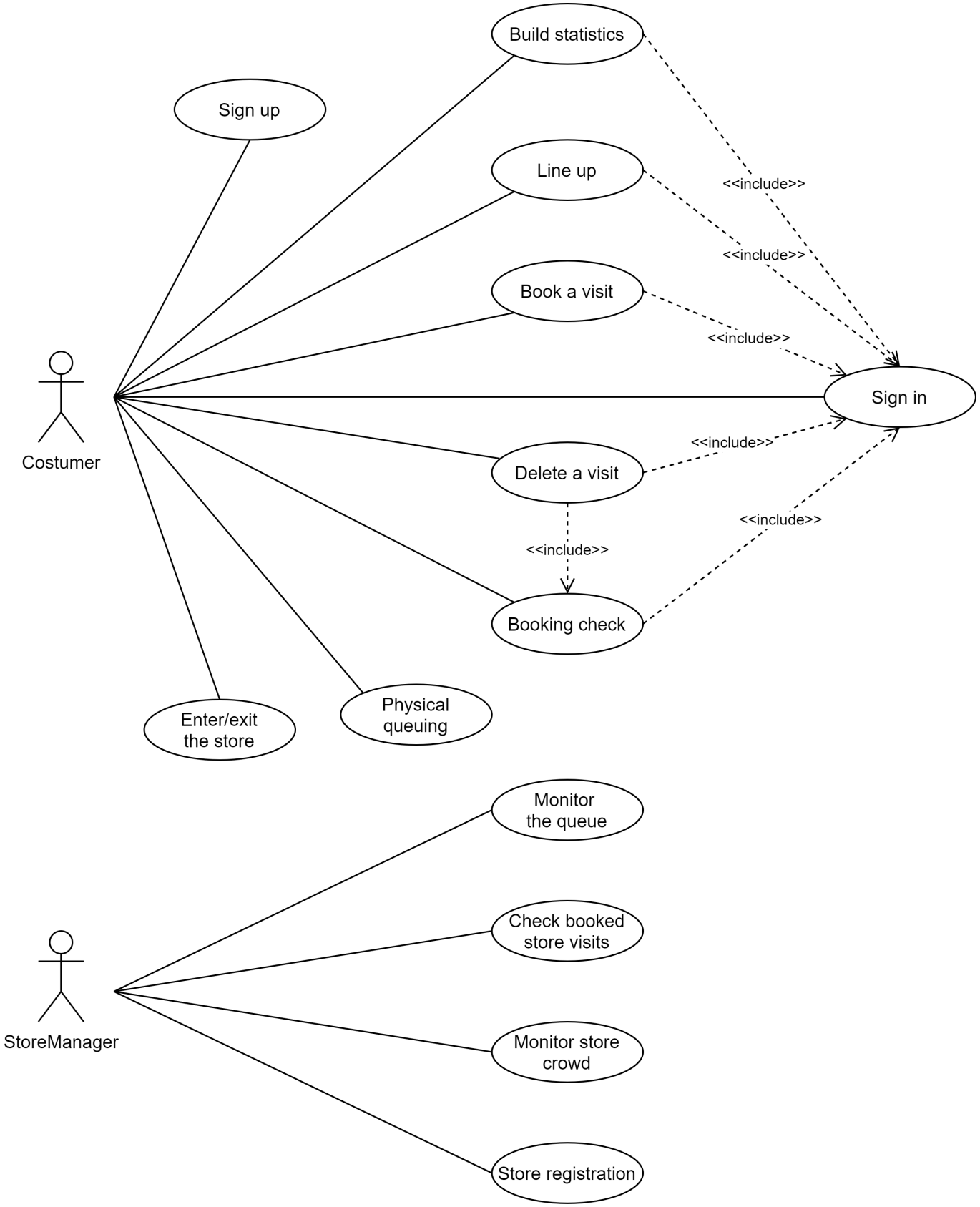

Figure 7

1. **Costumer sign up**

Name	Costumer sign up
Goals	G3 G4
Actor	Costumer
Entry Conditions	1. The costumer has already opened the application 2. The customer has not a CLup account
Events Flow	1. The costumer clicks the "Create new account" button 2. The costumer fills the registration form 3. The costumer checks the filled form fields and confirms them 4. The system checks if the provided data are correct and creates the account, storing the user data
Exit Conditions	The costumer account is successfully created
Exceptions	The system can detect problems in the submitted registration form (fields filled a in wrong way or even left empty). If it happen, the system notifies it to the user and offer the possibly to re-fill it

2. **Store registration**

Name	Store registration
Goals	G5 G6 G7
Actor	Store manager
Entry Conditions	1. The store manager has already opened the application 2. The store is not registered yet
Events Flow	1. The store manager clicks the "Store registration" button starting the registration procedure 2. The manager fills the registration form with the store information and submits it to the system 3. The system checks if the mandatory fields are correctly filled 4. The system continues the store registration procedure checking: a. The validity of the information provided by the manager b. If the store is able to satisfy all the CLup hardware requirements 5. The system completes the store registration storing the collected data
Exit Conditions	The store is successfully registrated
Exceptions	1. The system can detect problems in the filled registration form (fields filled a in wrong way or even left empy) and offer the possibly to re-fill it 2. The system could discover that the information provided by the system manager are not valid. In this case the registration procedure must be restarted 3. The store could not be able to satisfy the hardware requirements, therefore the system doesn't allow that store to register in the CLup system

3. **Costumer sign in**

Name	Costumer sign in
Goals	G3 G4
Actor	Costumer
Entry Conditions	1. The costumer has already opened the application 2. The customers has a CLup own account
Events Flow	1. The costumer fills the sign in form with his access credentials 2. The costumer checks the filled fields and clicks the "sign in" button 3. The system checks the filled form fields 4. The system redirects the user to the "available stores" screen
Exit Conditions	The costumer is logged-in successfully
Exceptions	The system can detect problems in the filled sing-in form (fields filled a in wrong way, fields left empty or the provided credentials don't belong to any user). If it happen, the system notifies it to the user and offer the possibly to re-fill it

4. **Costumer line-up**

Name	Costumer line-up
Goals	G2 G3
Actor	Costumer
Entry Conditions	The user has already logged-in
Events Flow	1. The costumer selects a store from the displayed ones 2. The system shows to the costumer the available actions (line-up or book a visit) of the store 3. The costumer clicks on the "Line-up" button starting the line-up procedure 4. The costumer inserts the means by which he will go to the store 5. The system retrieves the costumer GPS position 6. The system confirm the operation to the user and inserts him in the queue 7. The system compute the time needed to the user to reach the selected store
Exit Conditions	The costumer is correctly inserted in the store queue and he will be alerted when he has to leave home to reach the store
Exit Conditions	The system could not be able to retrieve the GPS position of the costumer, therefore the alert mechanism will be disabled

5. **Booking check**

Name	Booking check
Goals	G4
Actor	Costumer
Entry Conditions	The user has already logged-in
Events Flow	1. The costumer clicks the "show bookings" button 2. The system redirect the user to a new screen where are displayed his bookings
Exit Conditions	The costumer bookings are showed correctly
Exceptions	None

6. **Book a Visit**

Name	Book a Visit
Goals	G2 G4 G7
Actor	User
Entry Conditions	The system is showing the available actions of the selected store and the user selects the "Book a Visit" option
Events Flow	1. The user indicates what kind of products he's going to buy 2. The user says how much time he will spend shopping 3. The system stores the user's preferences 4. The system displays the user a timetable with the available slots to enter the store, taking into account the average time the user wants to spend inside the store and the products he wants to buy 5. The user selects one of the available time-slots 6. The system rearranges the queue inserting the user in it, reducing the available spots in the timetable 7. The system generates the virtual ticket with the QR code for the user
Exit Conditions	The system makes the ticket visible to the user
Exceptions	None

7. **Book a Visit with suggestions**

Name	Book a Visit with suggestions
Goals	G2 G4 G5 G7
Actor	User
Entry Conditions	The system is showing the available actions of the selected store and the user selects the “Book a Visit” option
Events Flow	1. The user indicates what kind of products he’s going to buy 2. The user says how much time he will spend shopping 3. The system stores the user’s preferences 4. The system computes the time-slots suggestions 5. The user clicks the ”time-slot suggestions” button 6. The system displays the user a timetable with the suggested time-slots, even from different stores 7. The user selects one of the suggested time-slots 8. The system rearranges the queue inserting the user in it, reducing the available spots in the timetable 9. The system generates the virtual ticket with the QR code for the user
Exit Conditions	The system makes the ticket visible to the user
Exceptions	None

8. **Delete Visit**

Name	Delete Visit
Goals	G4
Actor	User
Entry Conditions	The user has booked a visit
Events Flow	1. The user clicks "check your bookings" button 2. The system displays the user the list of booked visits 3. The user selects the visit he wants to delete 4. The system deletes the scheduled visit and all the related data, updating the queue of the store
Exit Conditions	The user has successfully deleted his scheduled visit and the system has correctly updated the line to let the queue of the store proceed
Exceptions	None

9. **Physical Queuing**

Name	Physical Queuing
Goals	G2, G6
Actor	User
Entry Conditions	The user is arrived at the store
Events Flow	1. The user makes a ticket request at the totem outside the store 2. The system checks if the physical queue length is lower than the threshold 3. The system updates the queue, inserting another customer in the first available timetable slot 4. The user takes his ticket with the QR code
Exit Conditions	The user has the ticket to enter the store and can wait without standing near other people
Exceptions	If the threshold is reached the ticket is not emitted and an error message is displayed. He will need to try to join the queue later or try to line-up through the app

10. **Customer enters/exits the store**

Name	Customer enters/exits the store
Goals	G1 G6
Actor	User
Entry Conditions	The user is queuing from home and then gets a notification to approach the store
Events Flow	1. The user arrives at the store 2. The user validates his ticket at the entrance through the QR reader because his turn has come 3. The system updates the queue 4. The user is doing shopping 5. The user validates his ticket at the exit of the store through the QR reader 6. The system updates the queue
Exit Conditions	The system has correctly updated the queue of the store and is ready to allow a new user join the store
Exceptions	If the customer arrives at the store and his ticket is not valid anymore to enter the store (the validity of his QR code has expired) he will have to reschedule from the beginning his visit to the store

11. **Build Statistics / Preferences**

Name	Build Statistics / Preferences
Goals	G5 G7
Actor	User
Entry Conditions	The user has booked a visit at least 10 times and the system has stored his preferences
Events Flow	1. The user clicks on the "statistics" button 2. The system retrieves the user statistics 3. The results are shown to the user
Exit Conditions	The user can see what are his favourite products and the average time he usually spends inside the stores
Exceptions	None

12. **Allow more people in the store**

Name	Allow more people in the store
Goals	G1 G2 G5 G6
Actor	Store manager
Entry Conditions	The system notifies that, given the preferences of the users, it is possible to allow more people in the store
Events Flow	1. In the main page the Store manager clicks on the "Temporary Increase Store Capacity" button entering a dedicated page 2. The Store manager selects the amount of additional customer he wants to allow in the store 3. The Store managers clicks the "Confirm" button 4. The system acknowledges the event and displays a confirmation message
Exit Conditions	The system allows the additional selected number of customers in the store
Exceptions	The system detected that there are no longer the conditions to allow more people in the store safely and does not let the Store manager confirm, showing an explanatory message and bringing the interface back to the main page

13. **Monitor the queue**

Name	Monitor the queue
Goals	G1 G6
Actor	Store manager
Entry Conditions	The Store manager is logged in
Events Flow	1. The Store manager clicks on the "Real-time store statistics" button in the main page 2. The system retrieves the information and renders a dedicated page 3. The Store manager interface is redirected to the dedicated page
Exit Conditions	The Store Manager is able to see the actual queue status on his device
Exceptions	none

14. **Check booked store visits**

Name	Check booked store visits
Goals	G4
Actor	Store manager
Entry Conditions	The Store manager is logged in
Events Flow	1. The Store manager clicks on the "Bookings" button in the main page entering a dedicated page 2. The Store manager chooses the time span he wants to visualize 3. The system retrieves the information relative to the desired time span and updates the page with the information
Exit Conditions	The Store Manager sees the information about the booked visits in the desidered time frame
Exceptions	none

15. **Monitor store crowd**

Name	Monitor store crowd
Goals	G1 G6
Actor	Store manager
Entry Conditions	The Store manager is logged in
Events Flow	1. The Store manager clicks on the “Internal Status” button in the main page 2. The system retrieves the information and renders a dedicated page 3. The Store manager interface is redirected to the dedicated page
Exit Conditions	The Store Manager can see the current state of customers inside the store
Exceptions	none

3.3.2 Sequence Diagrams

1. Costumer sign in

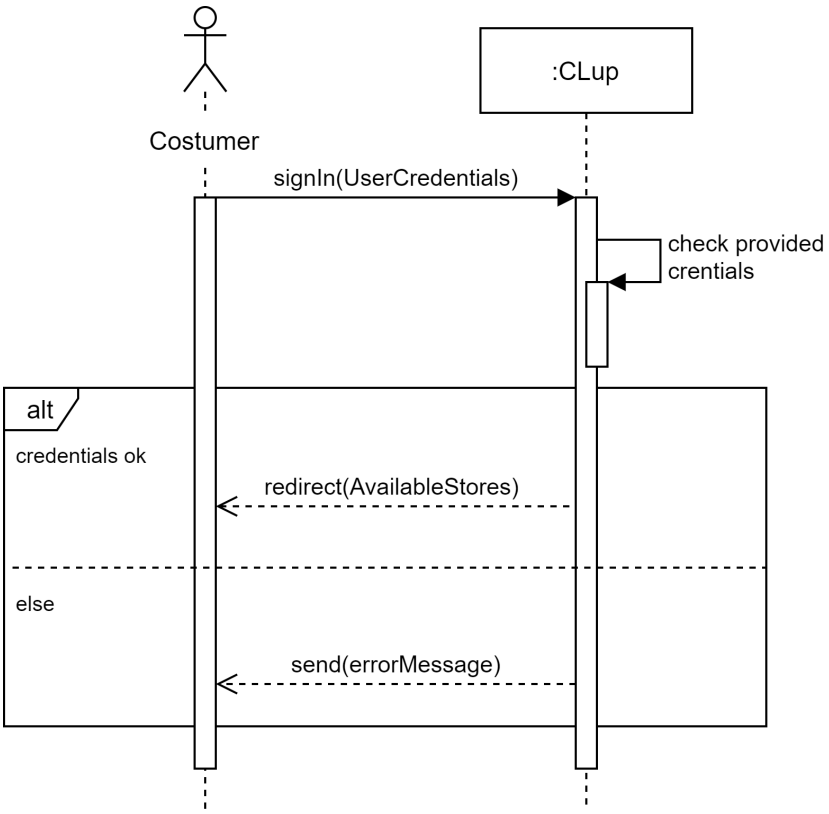

Figure 8

2. Costumer sign up

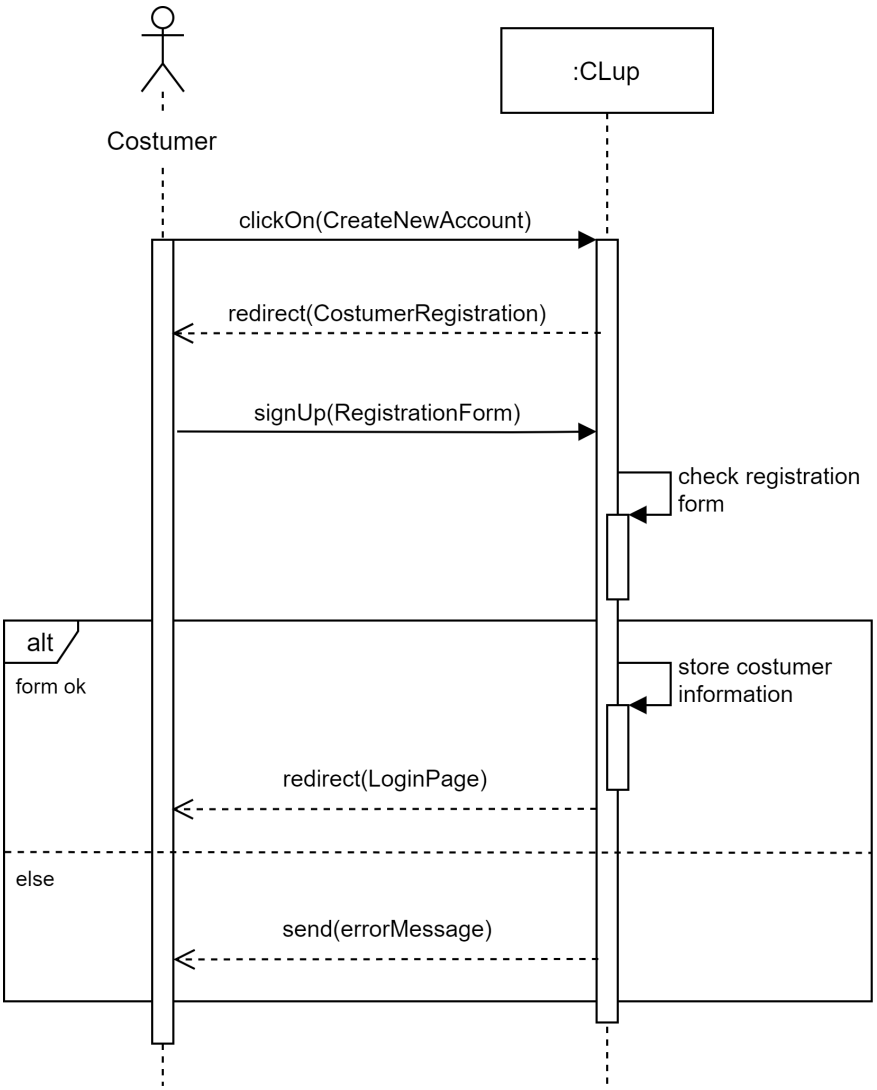

Figure 9

3. Store registration

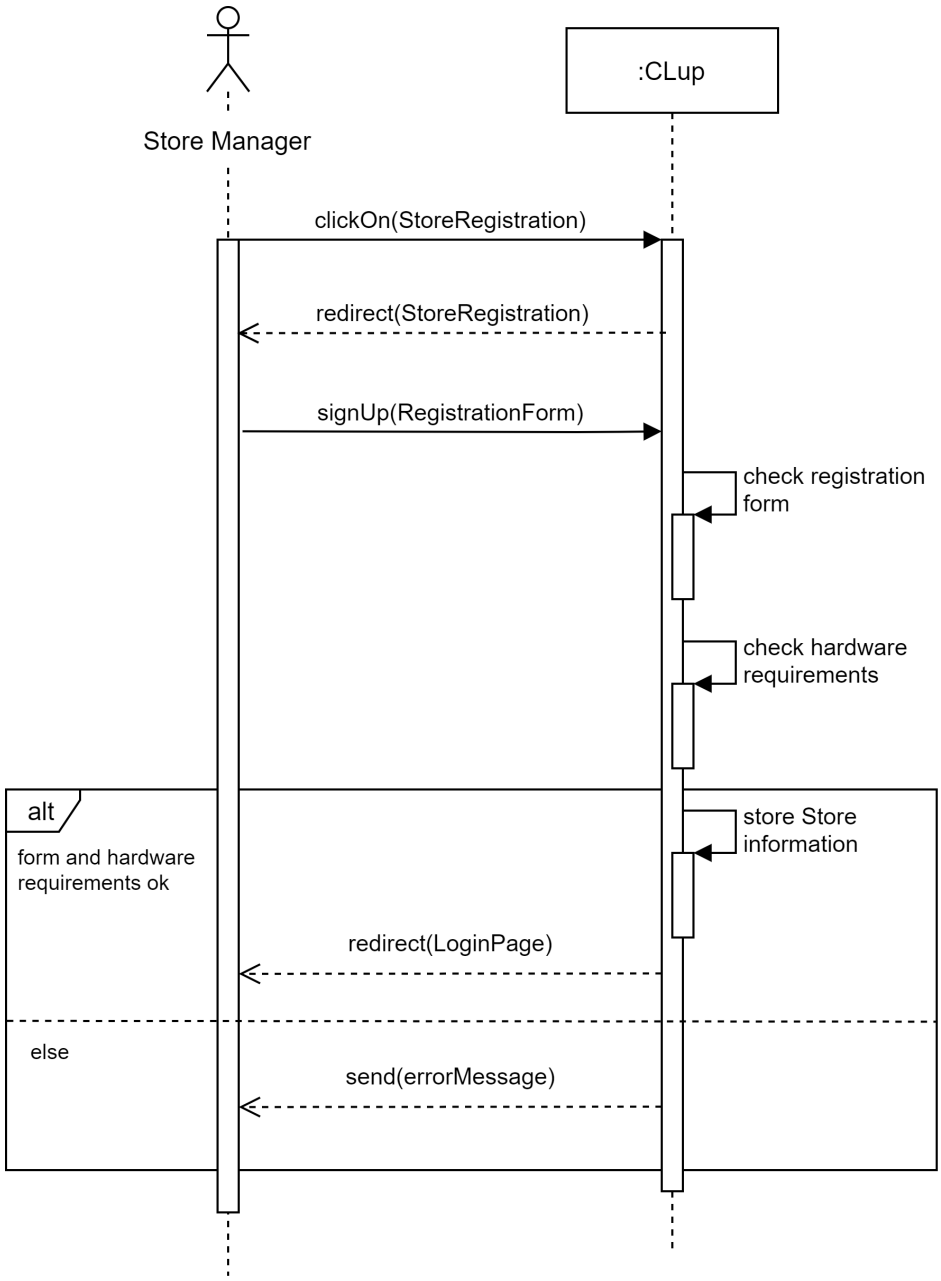

Figure 10

4. Line up

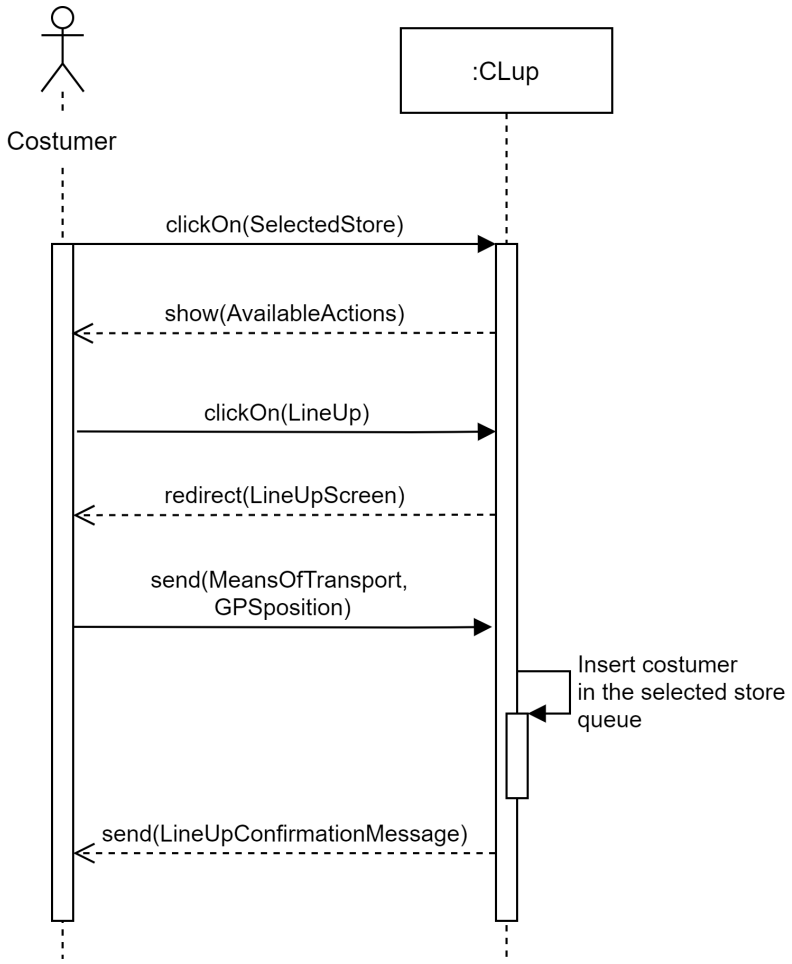

Figure 11

5. Book a Visit

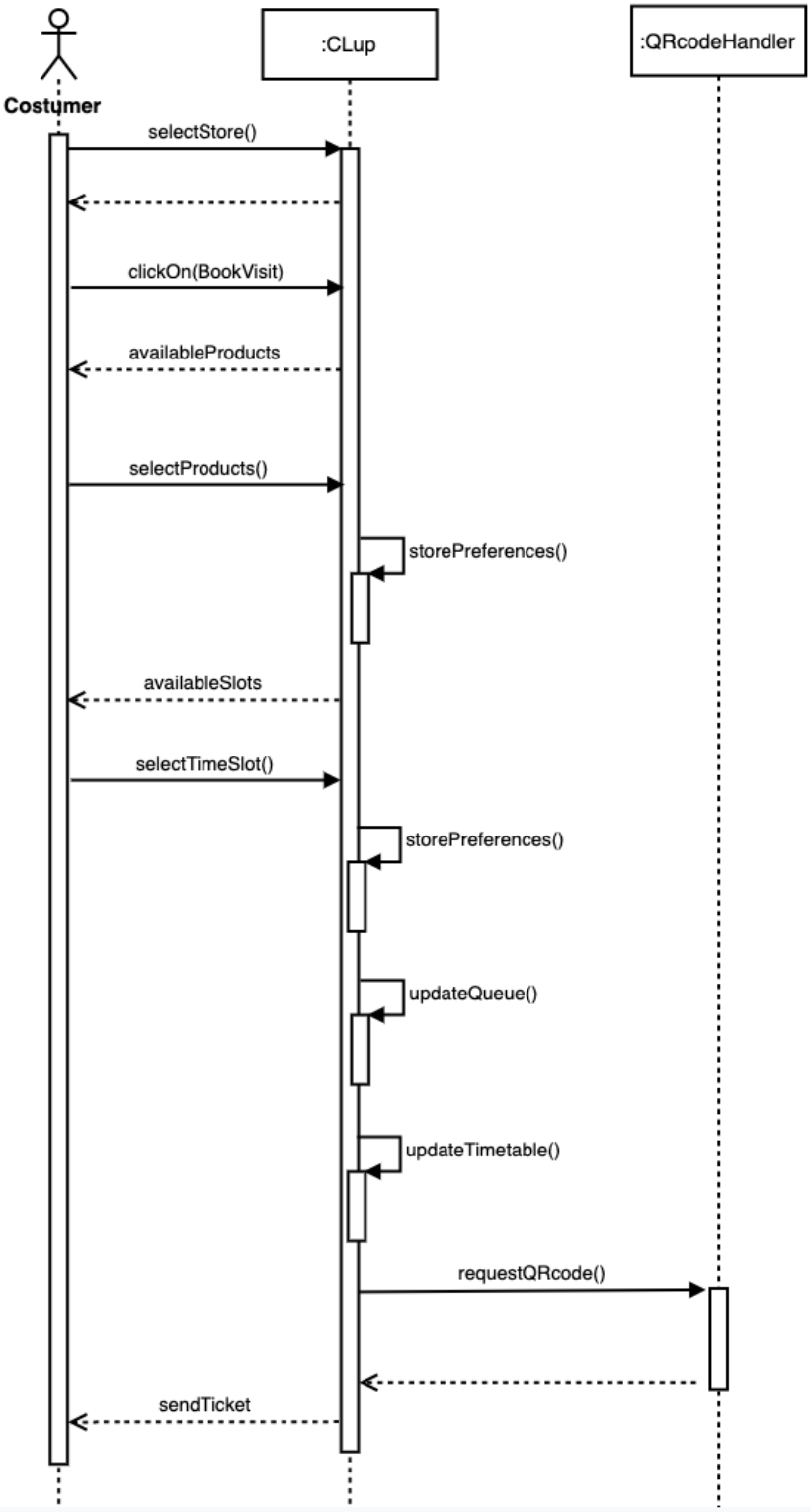

Figure 12

6. Book a Visit with suggestions

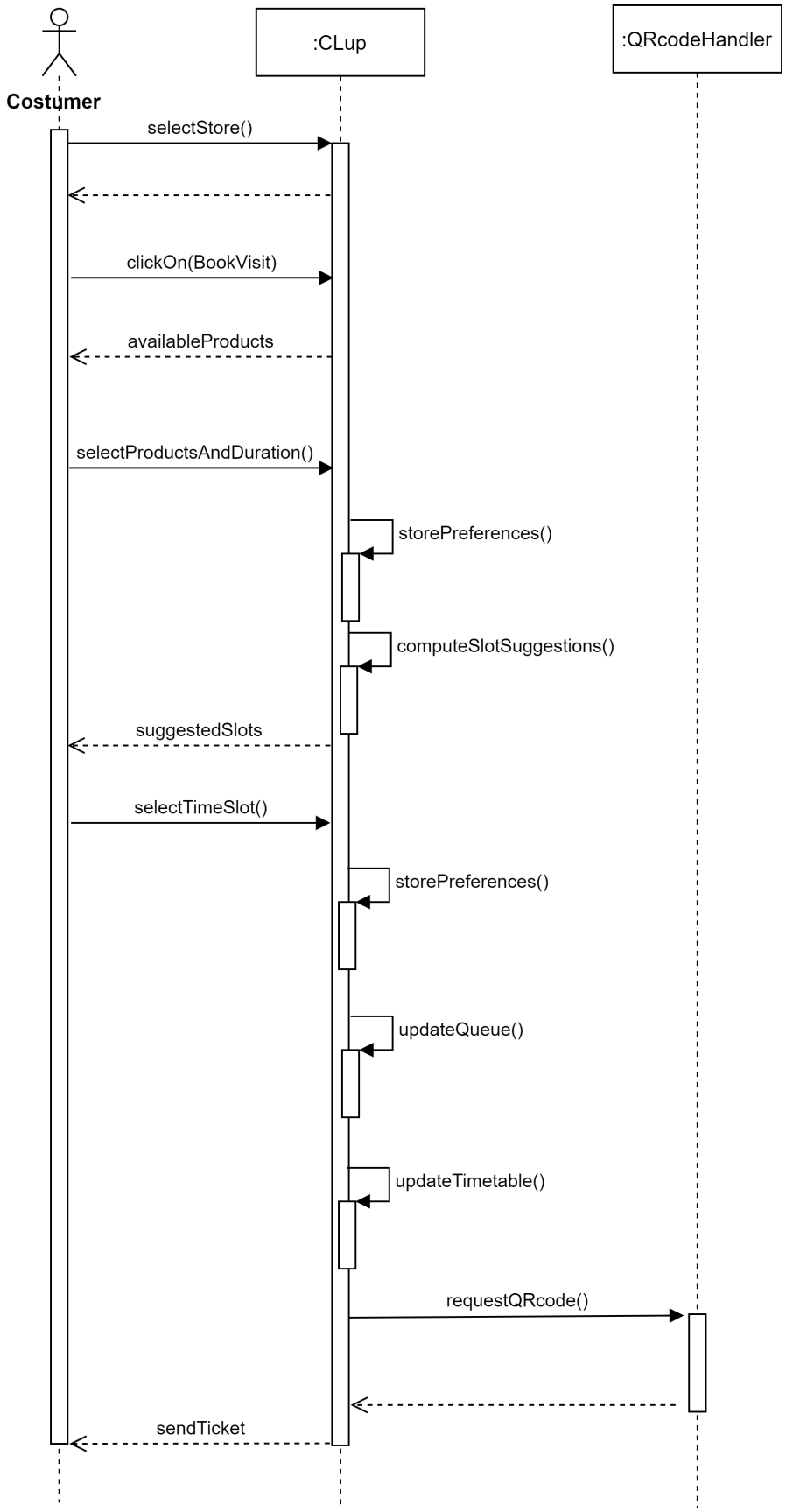

Figure 13

7. Delete Visit

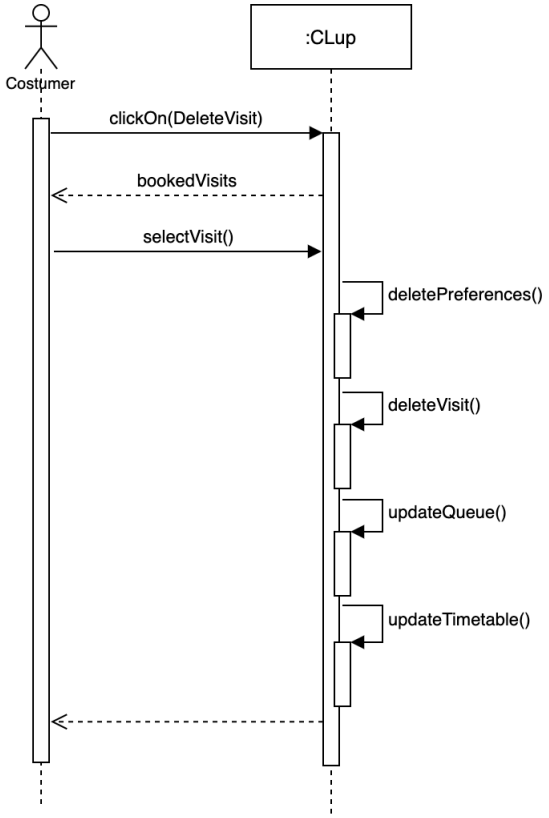

Figure 14

8. Physical Lining Up

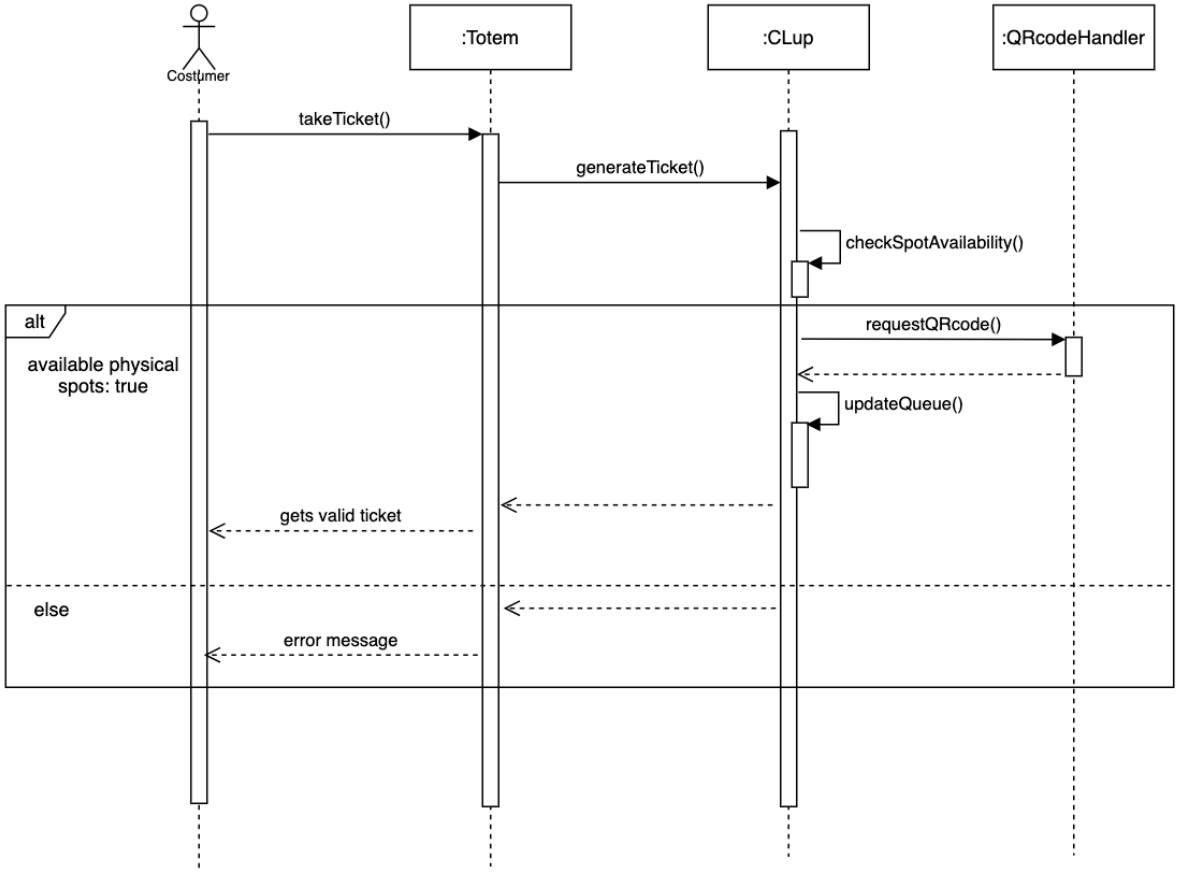
Costumer
:Totem
:CLup
:QRcodeHandler
takeTicket()
generateTicket()
checkSpotAvailability()
alt
available physical spots: true
requestQRcode()
updateQueue()
gets valid ticket
else
error message

Figure 15

9. User Enters / Exits the Store

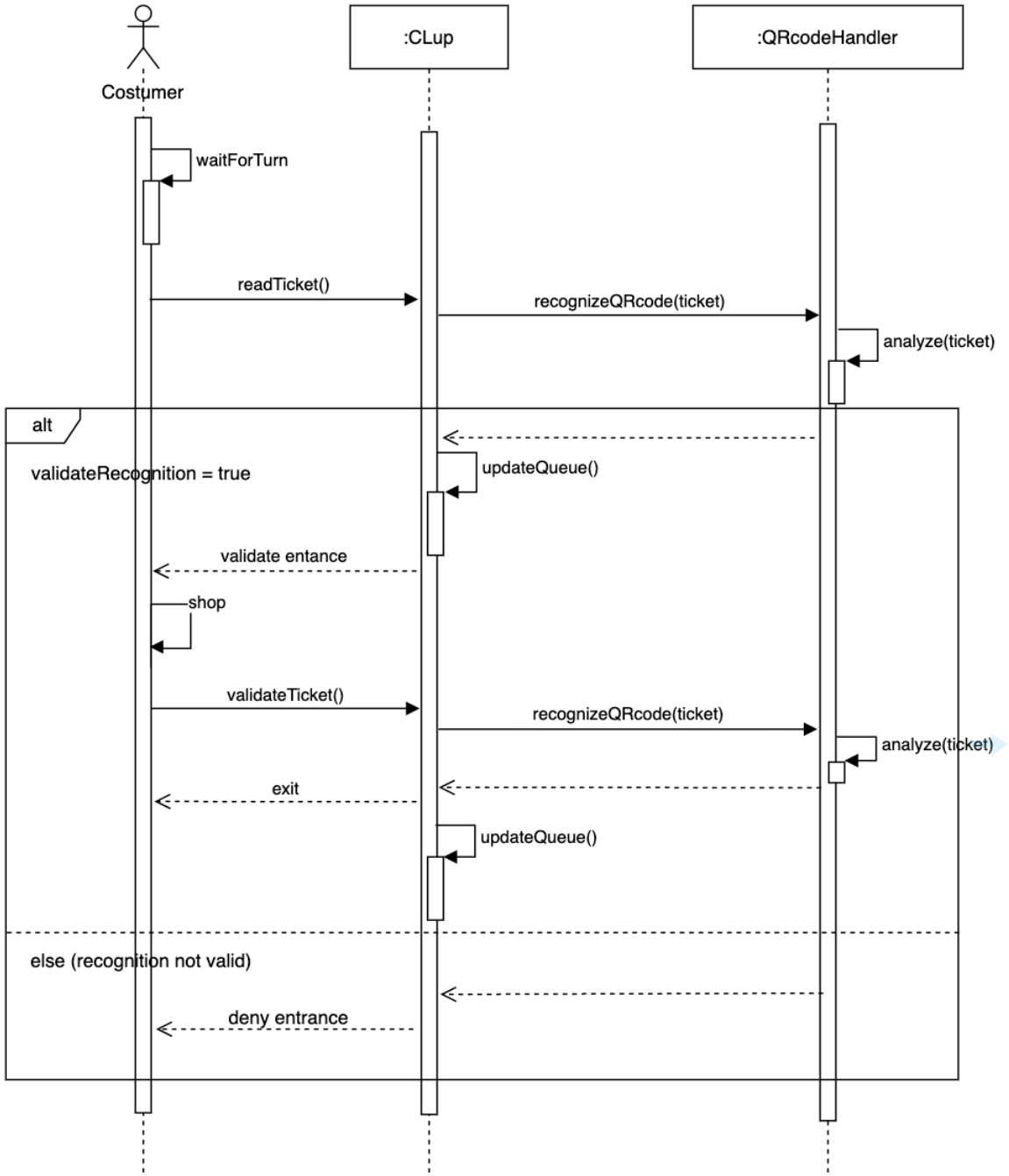

Figure 16

10. Build Statistics / Preferences

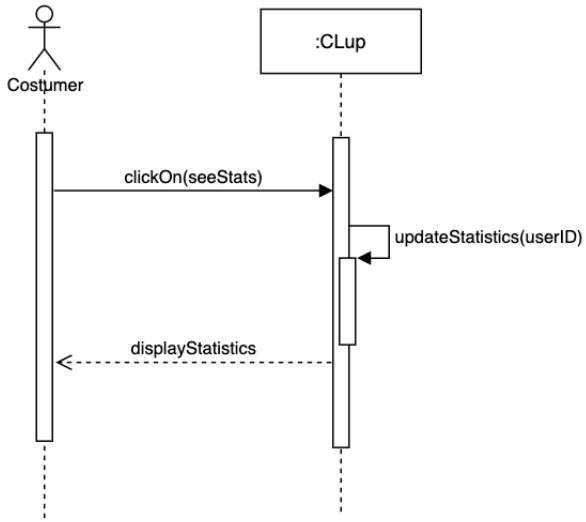

Figure 17

11. Allow more people in the store

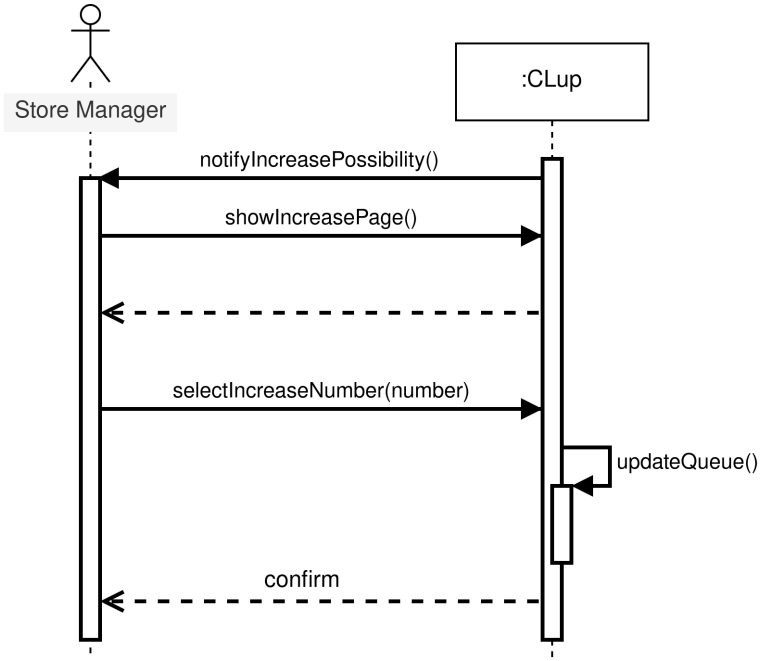

Figure 18

3.4 Performance Requirements

The system will face a large amount of data. Considering a single store with 50 costumers every hour no problem, but the CLup software must support thousands of registered store and all of them have hundreds of clients every day. Handling this quantity of information brings a meaningful complexity that could cause slowdowns and other problems, therefore, performance should not be underestimated. The system must ensure responses in 1-10 milliseconds (not considering network latency) to every user request, despite the thousands of simultaneous interactions. Then, reliability must be another key feature of the system, therefore there must be less failures as possible.

3.5 Design Constraints

3.5.1 Hardware limitations

Every costumer, in order to take full advantage of every system functionality, must be able to face the hardware requirement presented in the *Hardware interfaces* subsection. In fact, the lack of any of them will disable some CLup feature for the user.
For example, if the costumer device isn't equipped with a GPS, the system won't be able to retrieve his position and compute the store reaching time. Therefore, the alert mechanism will be disabled for this user.

3.5.2 Other constraints

Regarding the functionality *Increase the capacity of the store*, if the store wants to exploit this feature, it must define a policy specifying the conditions to increase the store capacity (i.e. the maximum number of people allowed in the different sections of the store).
In addition the UI should be simple and accessible in order to be used by a range of users that includes all demographics.

3.6 Software System Attributes - Non-Functional Requirements

3.6.1 Reliability and Availability

The system should be up at least for the 99,9% of the time, so the average time between it goes down and then it is recovered (MTTR) should not be higher than 9 hours per year. This also implies that the system should always guarantee that the needed services and functions are satisfied.
This high availability of the system needs to be provided because it is related to an emergency situation, and the users should always be able to decide to go shopping when they need (also because in this way they are likely to be distributed during the day). If this condition is not satisfied, there's the high risk that when the service will be recovered, all the people would go at the same time to the stores, and the overcrowding problem won't be avoided anymore.
Because we want to ensure this high grade of availability, a system of redundant servers can be considered. However, we can also take into account to have a prepared team that can recover the system as fast as possible when it's needed, while the users have been notified both when the system is down and when its services are restored and work properly.

3.6.2 Security

Passwords of the users and of the managers in the stores need to be protected, so before storing them they need to be correctly hashed. A similar treatment has to be made also for the preferences pointed out by the users and stored by the system when they are lining up. What's more, for what concerns the managers, a periodical update of their password can guarantee a higher protection of the data collected and used by the store. Of course, the system should be protected against intrusion from agent that are not authorized to access it.

3.6.3 Maintainability

As previously said, the system must be flexible and easy to maintain. For this reason a proper use of design pattern has to be made, as well as a good level of abstraction, so that if the code needs to be extended with new features, everything could be made without any kind of deep code changes. Code must also be well

commented and tests on main features of the system should always be working.

3.6.4 Portability

The software must run on different platforms, being compatible with the different devices that can be used: for example, managers at the store and users at home could use computers, so the services of the system need to be made usable on Windows OS, Linux OS and MacOS; because of the mobile application, it must be supported a version for Android, iOS and HarmonyOS.

3.6.5 Scalability

Everyone will use Clup because of emergency reasons, so the scalability of the system is one of the most important issues to deal with. An high reaction time needs to be reached, so as a consequence it must be set a proper number of processors, devices and memory.

4 Formal analysis using Alloy

4.1 Alloy code

This section is dedicated to the creation and analysis of a formal model made using *alloy*.
The main goals of this model are giving a formal description to the specifications of the software to be as well as the application domain where the software will work.
The model has been developed starting from the entities of the UML model of the application domain in Section 2.1.
Moreover, this formal description allows to prove different properties of the model and assess its logical soundness.
For the sake of simplicity, the model does not consider the possibility for the store manager to temporarily increase the limit of customer.
In particular, this properties are assessed:

- Users can queue for at most one store at a time
- The number of customers inside a store cannot exceed the declared capacity of the shop itself
- Customers cannot be in queue as long as the store capacity is not reached
- The ticket released either physically by the ticket machine or virtually by the system are valid and consistent

The alloy code below describes different entities, the main are:

- Customer, an abstract signature that is extended by RegisteredUser (i.e. an user of our system) and InPresenceCustomer (i.e. a customer that physically reached the store, not using our system)
- Store, which describes the physical shop with its properties
- Queue, the entity that the system is going to handle
- Visit, an entity describing a future booked visit handled by the system

```
abstract sig Customer{
ticket: one Ticket
}


--CLup user
```

```
sig RegisteredUser extends Customer{
meansOfTranportation: lone Transportation,
visits: set Visit,
position: lone Position
}

--Normal customers, non CLup user
sig InPresenceCustomer extends Customer{
}

sig Date{}

sig QrCode{}

abstract sig Ticket{
number: one Int,
code: QrCode,
store: Store
}

--ticket created by the system
sig VirtualTicket extends Ticket{}

--ticket released by the ticket machine
sig PhysicalTicket extends Ticket{}

--booked visit
sig Visit{
date: Date,
itemsToBuy: set Item,
store: Store
}

--store item
sig Item {
quantity: one Int
} {
quantity > 0
}

sig Transportation {}

--opening days
sig Timetable {
opening: some Date
}

--store queue
sig Queue{
inLine: set Customer,
length: one Int,
interfaces: some TicketMachine
} {
length≥0
}

sig Store{
name: set String,
```

```
queue: Queue,
capacity: one Int,
insideCustomers: set Customer,
timetable: Timetable,
location: one Position
} {
capacity > 0
}

sig StoreManager {
supervises: Store
}

sig Position{}

sig TicketMachine{
emittedTickets: set PhysicalTicket
}



//FACTS

--a customer cannot interact (i.e. queueing or being inside) two different
    ↪  stores at the same time
fact UbiquityConstraint1 {
all customer: Customer | all disj s1,s2: Store |
(customer in s1.insideCustomers or customer in s1.queue.inLine) implies
not (customer in s2.insideCustomers or customer in s2.queue.inLine)
}

--a user cannot have two booked visits in the same date
fact UbiquityConstraint2{
all user: RegisteredUser | all v1, v2: Visit |
(v1 ≠ v2 and v1 in user.visits) implies not(v2 in user.visits and v1.date
    ↪ = v2.date)
}


--a customer cannot be both inside the store and in the queue
fact UbiquityConstraint3{
all store: Store | all customer: Customer |
customer in store.insideCustomers implies
not (customer in store.queue.inLine)
}


--ticket consistency
fact TicketLogic {
all customer: Customer | all t: Ticket |
(t in customer.ticket) implies (customer in t.store.insideCustomers or
customer in t.store.queue.inLine)
}


--physical tickets are emitted only by ticketmachines and the store for
    ↪ which the ticket is
--valid is the store whose ticket machine belongs
```

```
fact PhysicalTicketConsistency{
all ticket: PhysicalTicket | one ticketMachine: TicketMachine, s :Store |
ticket in ticketMachine.emittedTickets and s in ticket.store and
ticketMachine in s.queue.interfaces
}


fact PhysicalTicketConsistency2{
all customer: InPresenceCustomer | one ticketMachine: TicketMachine |
customer.ticket in ticketMachine.emittedTickets
}


--QrCodes are unique
fact UniqueQrCode {
no disj t1,t2: Ticket | t1.code = t2.code
}


--visit belongs to only one user
fact OneOwnerForVisit{
all visit: Visit | one user: RegisteredUser |
visit in user.visits
}


--every ticket belongs to a customer
fact OneOwnerForTicket{
all t: Ticket | one customer: Customer |
t in customer.ticket
}


fact {
all item: Item | some visit: Visit |
item in visit.itemsToBuy
}


fact {
all transportation: Transportation | some user: RegisteredUser |
transportation in user.meansOfTranportation
}


fact {
all t: Timetable | one store: Store |
t in store.timetable
}


fact {
all d: Date | some visit: Visit |
d in visit.date
}


--no visits can be booked when the store is not open
```

```
fact OpeningHours{
all visit: Visit, s: Store |
(s in visit.store) implies (visit.date in s.timetable.opening)
}


--no one is in queue if there is residual capacity in the store
fact QueueLogic {
all store: Store |
(#store.insideCustomers < store.capacity) implies
(#store.queue.length = 0)
}


--a store cannot exceed his capacity limit
fact CapacityConstraint {
no store: Store | #store.insideCustomers > store.capacity
}


fact QueueLength{
all queue: Queue | #queue.inLine = queue.length
}


--Ticket Machines belong to one queue, and consequently store, at a time
fact TicketMachineConsistency{
no ticketMachine: TicketMachine | some q1, q2: Queue |
q1 ≠ q2 and ticketMachine in q1.interfaces and
ticketMachine in q2.interfaces
}


fact TicketMachineConsistency2{
all ticketMachine: TicketMachine | one queue: Queue |
ticketMachine in queue.interfaces
}


--every queue belongs to one store only
fact {
all q: Queue | one store: Store |
q in store.queue
}




//ASSERTIONS

assert QueueConsistency{
no store: Store |
#store.queue.length > 0 and #store.insideCustomers < store.capacity
}
check QueueConsistency


assert LegalLimit {
```

```
no store: Store |
#store.insideCustomers > store.capacity
}
check LegalLimit


assert OneStoreAtTime {
no customer: Customer | all s1,s2: Store |
(s1 ≠ s2 and (customer in s1.insideCustomers or customer in s1.queue.
   ↪ inLine)) and
(customer in s2.insideCustomers or customer in s2.queue.inLine)
}
check OneStoreAtTime


assert TicketValidity {
no ticket: Ticket | all s: Store |
s not in ticket.store and ticket in s.queue.interfaces.emittedTickets
}
check TicketValidity




pred OneStore {
#RegisteredUser > 1
#InPresenceCustomer > 1
#Transportation < 2
#Item < 2
#Store = 1
#TicketMachine < 3
}

run OneStore for 5


pred TwoStores {
#RegisteredUser > 1
#InPresenceCustomer > 1
#Transportation < 2
#Store = 2
#TicketMachine < 3
#Visit < 2
}

run TwoStores for 5
```

4.2 Meta model

Follows below a world generated by the OneStore run:

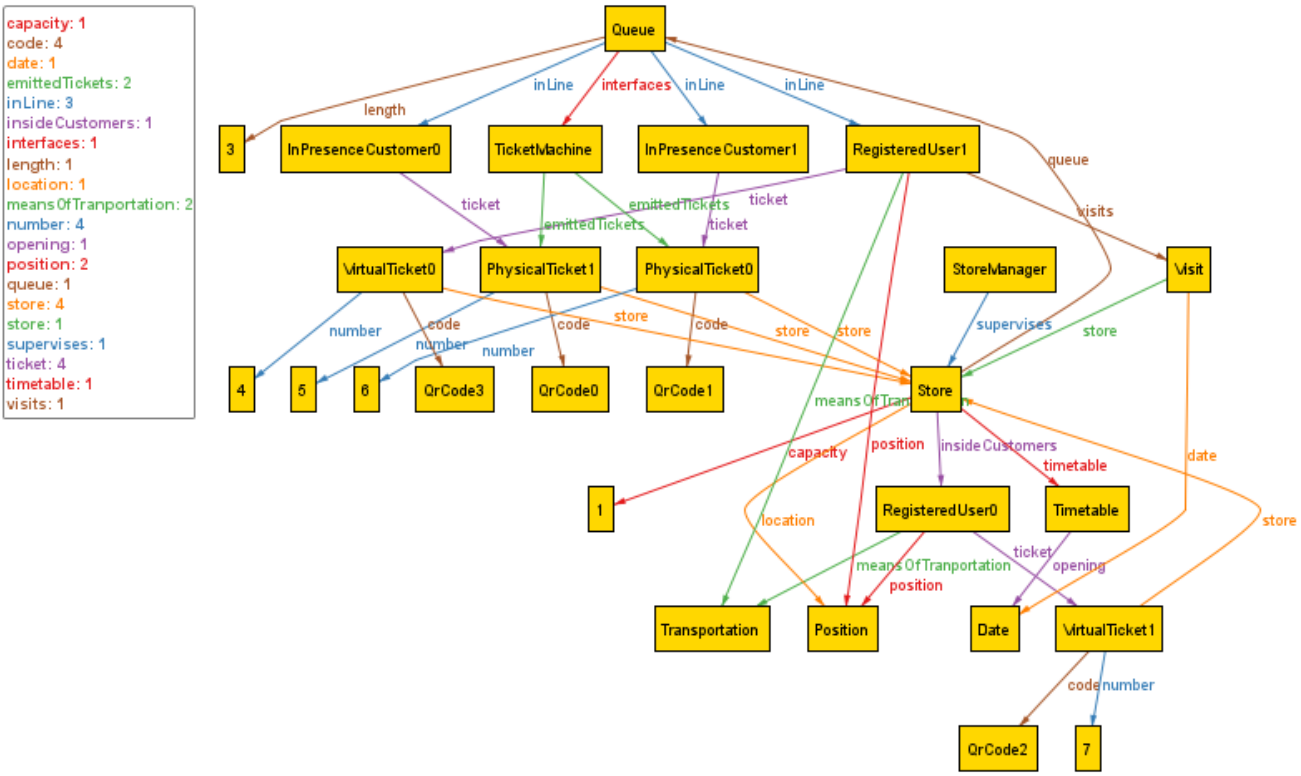

Figure 19: World generated by OneStore predicate

//forse potrei mettere un altro modello fatto con un altra run?

The generated world underlines the entities and constraints expressed in the alloy code.
As expected, the store has one and only one queue containg both CLup users and normal customers (with their respective tickets) while the store is at his full capacity.

4.3 Result of Assertions and Predicates

```
Executing "Check OneStoreAtTime"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  5924 vars. 429 primary vars. 13913 clauses. 0ms.
  No counterexample found. Assertion may be valid. 16ms.
```

```
Executing "Check LegalLimit"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  6123 vars. 429 primary vars. 14613 clauses. 16ms.
  No counterexample found. Assertion may be valid. 0ms.
```

```
Executing "Check QueueConsistency"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  6294 vars. 429 primary vars. 15235 clauses. 16ms.
  No counterexample found. Assertion may be valid. 16ms.
```

```
Executing "Check TicketValidity"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  5957 vars. 429 primary vars. 13907 clauses. 54ms.
  No counterexample found. Assertion may be valid. 15ms.
```

```
Executing "Run OneStore for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  14248 vars. 920 primary vars. 32041 clauses. 32ms.
  Instance found. Predicate is consistent. 31ms.
```

```
Executing "Run TwoStores for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  14248 vars. 920 primary vars. 32041 clauses. 32ms.
  Instance found. Predicate is consistent. 31ms.
```

5 Effort spent

Antonio Ercolani:

Discussion on the first section	**2h**
Purpose and Document Structure	**1,5h**
Product Functions and User Characteristics	**3h**
Discussion on Requirements and Use Cases	**3h**
Use cases	**3h**
Performance requirements and Design Constraints	**1h**
UI mockups	**0,5h**
Use case diagram	**0,5h**
Sequence diagrams	**1h**

Vittorio Fabris:

Discussion on the first section	**2h**
Scope	**4h**
State Diagrams	**3,5h**
Discussion on Requirements and Use Cases	**3h**
Use cases	**3,5h**
Sequence diagrams	**4,5h**
Requirements and Domain Assumptions	**1,5h**
Non Functional Requirements	**3h**
Final Discussion	**3,5h**

Riccardo Nannini:

Discussion on the first section	**2h**
World and Shared Phenomena & Goals	**3h**
Product perspective & class diagram	**3h**
Discussion on Requirements and Use Cases	**3h**
Use Cases & 3.1 section	**3h**
Scenarios and sequence diagram	**2,5h**
Alloy section	**15h**

6 References