POLITECNICO
MILANO 1863

- RASD -

Requirements Analysis and
Specification Document

Computer Science and Engineering

Software Engineering II

A.A. 2020/2021

Daniele Mammone - 10625264
Gianmarco Naro - 10610374
Massimo Parisi - 10583470

Contents

1 Introduction

1.1 Purpose

The main target of this document is to describe the **Customer Line-Up** (*CLup*) software through functional and non-functional requirement. The structure of the document follows the one studied during lectures and aims to describe faithfully the software behaviour in all of its aspects.

CLup is a mobile service usable through app, made both for store managers and customers. It facilitates customers to book a visit to a store and to get a spot on the queue for entering a store and, on the other hand, helps store managers to observe the new strict rules due to *Covid-19*.

1.2 Scope

The main purpose of *CLup* is to facilitate customers to access at a store in **security**, both allowing them to reserve a spot on the queue for entering the store through the app and to book a visit at the store at a determined time of a certain day, selected by the user. Thanks to this, store managers can manage the **affluence** in their store more easily, and moreover can reduce the crowd in front of the store, that is one of the main purpose of the application. The main idea is that the system assigns a number to each customers' reservation, and when a person's number is called, he can enter the supermarket. For this purpose, the app generates a *QR Code* associated to the reservation, useful to check-in and check-out a customer in the store. Thanks to this, the system knows who is inside the store, and saves informations about customers' shopping sessions. By this data, it's possible to estimate time needed by people to complete what they need to do inside the store, and to calculate the *ETA* to enter the store from a specific moment. Also, *CLup* can suggest customers alternatives to their choose, either reserving a spot on the queue, and booking a visit. For customers without an electronic device, it's possible to get into the queue at a totem, installed at each store entry. Since someone may need to use only a part of the store, customers can select only the department of their interest; the aim of this is to decrease waiting times. At least, for reservation made by app, notifications are sent when someone must depart from his location, to get to store in time for his turn. The line up management must be **fair**, to avoid regrettable situations such as can't entering to the store without booking or trying to access the store when it's the turn of somebody else. In the case the queue goes over the store working hours, customers are alerted by a warning message on their tickets.

In the following sections, there are described **World and Shared Phenomenons** through the "World and Machine" paradigm. In Shared Phenomena section, the M stands for phenomenons controlled by Machine and observed by world, W the vice versa.

1.2.1 World Phenomena

WP1	A user enters a store
WP2	A user waits in a lineup
WP3	A user exits the supermarket
WP4	A certain number of people is inside the supermarket
WP5	A certain number of people is at a specific department of the supermarket
WP6	The *Covid-19* pandemic imposes some restrictions on crowds of people

1.2.2 Shared Phenomena

SP1	Customers get a spot on the queue	W
SP2	Customers book a visit to the store	W
SP3	Customers using app knows when they should depart for the store	M
SP4	Customers come to know how much time they have to wait before entering	M
SP5	Customers scan the *QR code* and enters in the supermarket	W
SP6	Customers scan the *QR code* and exits from the supermarket	W
SP7	Customers can indicate the categories of items that they intend to buy	W
SP8	Customers are called to enter the store	M

1.2.3 Goals

G1	Allow customers to select a store and book a visit on a certain date and time from *CLup* app
G2	Allow customers to select a store and take a spot on the queue to enter as soon as possible the store from *CLup* app
G3	Allow customers to book a spot on the queue from a physical ticket dispenser
G4	Allow customers to have suggestions on better store options
G5	Allow customers to decrease waiting times specifying departments they want to visit
G6	Allow customers to manage their reservations
G7	Allow customers to depart from their location in time to avoid waiting too much, and to avoid losing their turn for entering
G8	Allow the store manager to know the real situation of people that are inside the building and in which departments of his store
G9	Allow to grant a fair management of users that can access the building
G10	Allow to manage optimally the influx in the building and avoid gathering inside it
G11	Allow the store manager to interact with customers and their reservations

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

QR Code	Bi dimensional bar code that allows the user to check-in/check-out at the store entries/exits
Reservation	Indicates both booked visits and spots on the queue to enter the store as soon as possible
Customer	The clients of the store, that uses the system to get a reservation to access the store
Store manager	The app user that access to stores' bookings, occupancy and settings, in order to manage the flow of customers
QR Code Reader	Device used to scan customers' *QR Code*
Totem	Electronic device that allows customers to physically get a spot on the queue to enter the store as soon as possible; it allows to specify the same parameters that can be inserted through the app
QR Code Printer	Device used by totems to print *QR Code*
Department	Part of the store that contains the same category of products

1.3.2 Acronyms

RASD	Requirement Analysis and Specification Document
ETA	Estimated Time of Arrival
GPS	Global Positioning System
API	Application Programming Interface
UML	Unified Modeling Language

1.3.3 Abbreviations

WPn	World phenomena number n
SPn	Shared phenomena number n
Gn	Goal number n
Rn	Requirement number n

1.4 Revision History

Version	Date	Changelog
1.0	29/11/2020	First version
1.1	05/12/2020	Update of some domain assumption and functional requirements
1.2	17/12/2020	Update of some use cases and other fixes in order to be consistent with queue management
1.3	18/12/2020	First revision
2.0	19/12/2020	Stylistic revision of the document
2.1	23/12/2020	Deliverable version

1.5 Software and Tools

- LaTeX as software system for docuement preparation
- Alloy as model analyzer
- UMLet for the UML diagrams and other diagrams
- Photoshop for the mockups
- Git & Github as work space. The repository is here.

1.6 Reference Documents

- Specification Document
- Slides of the lectures

1.7 Document Structure

The structure of the document is thought with the intention of allowing simple navigation through it. Also, various abbreviations, highlighted in Abbreviations section, have been used to make the content smoother. Hence, the structure of the document is the following one:

- **Introduction**: introduces in a general way the scope of the application through the analysis of the *World Phenomena*, *Shared Phenomena* and *Goals*. Moreover, the main functions of the software are illustrated and the abbreviations, acronyms and definitions are reported in order to allow an easy reading.
- **Overall Description**: The section starts with a summary description of the *UML* of the software, so as to have a general presentation of the operations of the application. Then, in order to clarify the behavior of the system, there are state charts of the most important and critical functions and the detailed description of all software functions.
- **Specific Requirements**: The main focus of this section is to describe the essential hardware and software interfaces and requirements necessary to *CLup* for providing its services. After this, there is the core of the section containing the use cases that provides detailed information about the interaction with the system.
- **Formal Analysis Using Alloy**: This section describes formally the model using Alloy language, highlighting the main problems of the software, solving them in a formal way.
- **Effort Spent**: The main focus of this section is to track the time spent to complete this project. In particular, is highlighted the subdivision of the working hours of the various sections
- **References**: This section is dedicated to all references used in this project.

2 Overall Description

2.1 Product Perspective

In *Figure 1* is reported an **UML Class Diagram** that represents the domain of the application with main concepts and data involved, including their relationships.

The store managers registers to the application providing all the necessary informations and can decide at a later stage to modify some options (also regarding each department of the store), such as working hours for each week's day, capacities ecc. Customers download the application on their device and register to the service to be able to use it, or get ticket at the store entry. Here we can identify the main aspects related to *CLup*:

- The customer can generate a **reservation**, choosing between a registered chain store (and one of their specific store) or a normal store and, optionally, the departments that they want to access; *CLup* will store the reservation and send customers a (digital) ticket containing the reservation's *ID* and a *QR Code*. Furthermore, the ticket contains the *ETA* to enter the store, and it's always recalculated when it's visualized on the application.
- The customer can **entry** in the store where he has a reservation (when his ticket's number is called to entry) scanning the *QR Code* through the *QR Code Reader*. The system registers his entry.
- The customer **exits** the store reusing the same *QR Code*, updating the number of people inside the store, and the statistics.

The *UML* does not include every class of the actual implementation of the system.

2.1.1 UML Description

The **UML Class Diagram** in *Figure 1* contains many classes and in this section we are going to explain shortly their functions and their scope in the system.

- **Transportation**: Is an abstract class that defines the generic means of transport that could be chosen by the customer, such as:
  - Public transport
  - On foot
  - Bike
  - Car

- **User**: Is an abstract class that defines the generic user that can use the application. An user could be either a customer or a store manager, and access functions based on their privileges in the application.
  - **Customer**: Can generate a reservation and manage his ones (if on app). Moreover, each registered is associated with his preferred mean of transport.
  - **Store manager**: A store manager is associated to a store, and can manage it, modifying its parameters and working hours.
- **Statistic**: Is an abstract class that defines the generic statistic that can be used from the software to infer the customers' shopping time.
  - **Customer statistics**: The system uses the customer statistic in order to provide the average time spent during a visit in the store by a specific user. If the customer, during a booking, decides to not specify the time that will be used during his shopping, the system make an estimation based on his previous visits.
  - **Department statistics**: The system uses the statistic obtained from customers that visit a certain department's store in order to calculate the average time spent by customers in each store's department. Having done this, if a customer does not have his personal statistic and decide to not specify the time that he will use during his shopping, the system can base its estimation on statistics of other customers.
- **Store**: Represents the store with his unique *ID*. Each store is related to its departments, increasing its granularity, so that store manager can control its parameters in a detailed way, and so that customers can choose to visit only a part of it. Each store is associated with its map position, necessary to generate suggestions, and to know how much a customer need to reach the store's position from his location. Moreover, the store has a queue of callable tickets, reservations for other days, the working hours for each day of the week, associated with a flag about store opening in that day, and a list of entered ticket, for contact tracking purposes.
- **Department**: This class represents a store's department and is related to its statistics. The store manager can modify the *maxCapacity* and the *maxAllowedBooking* parameters for each department's store in order to avoid an overcrowding inside the store, and to grant fair balance between allowed booked clients and non booked ones. Moreover, the class takes trace of each customer inside the specific store zone.
- **Chain Store**: Each store can belong to a chain

- **Reservation**: Each reservation is associated to a customer (if got on the app), is managed by the store and it has many attributes that provides informations about both the entry and exit time (expected and effective). Moreover, a reservation has a status:
  - **Pending**: the reservation is in queue and hasn't been already called. So, called, entry and exit times are not present.
  - **Called**: the reservation have been called to enter the store. Called time is present, but not entry and exit one, and the reservation is in store's called tickets.
  - **Discarded**: the reservation have been called, but customer hasn't scanned his *QR Code* in time. Only call time is present, and the reservation isn't in any store reservation's list.
  - **Validated**: the reservation entered, all the time are present but exit time if the customer hasn't already exited, and the reservation is in store's entered reservations. If the customer is still inside, the reservation in department reservations' inside list.
- **Ticket**: Every reservation has a ticket that provides the most important things: *QR Code* and number to be called. Indeed, if a customer want enter or exit the shop, must scan his *QR Code*.

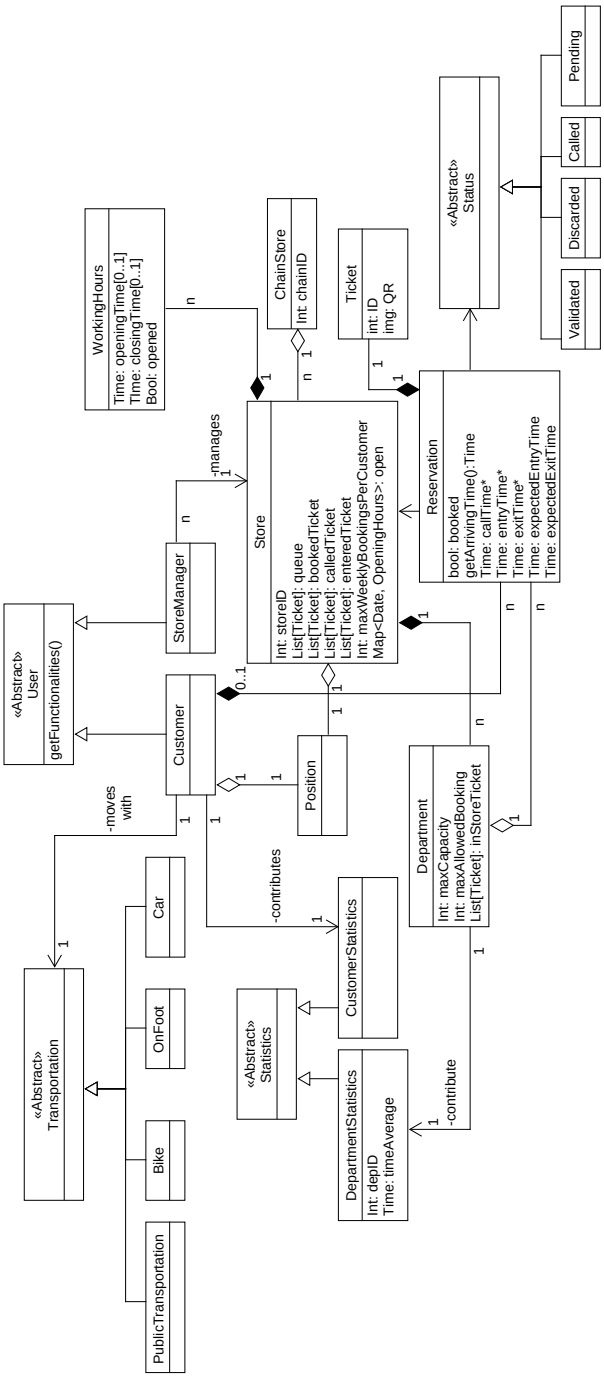

Figure 1: *UML Class Diagram*

2.1.2 State Charts

Now we are going to examine some essential aspects of the application, modelling their behaviours and evolution over time through adequate state diagrams, which are reported below. All the state machines are supposed to start at store opening, and to stop at closing.

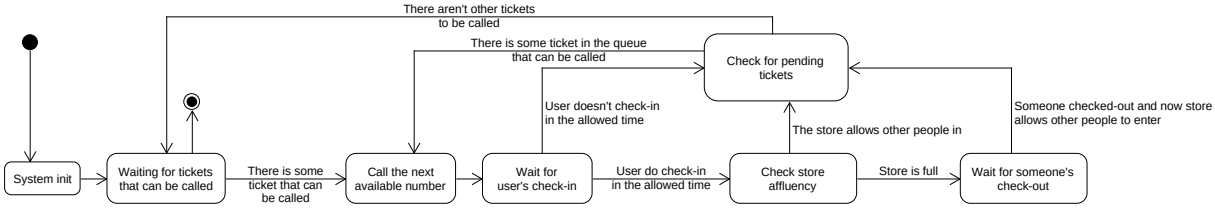

Figure 2: *Number calling system*

When the store manager opens the store, the system initialize itself, clean up the queue if some reservation remained from the previous day, initialize the queue with the booked reservations, and waits for some ticket that can be called. This queue updates each time a daily reservation is generated. After a ticket is available to be called, the system notifies in some way (e.g. through a store employee) that now certain reservation is allowed to enter the store. At this point, the system waits for the scan of the associated *QR Code*. If the customer doesn't check-in in the assigned time, the system discard the ticket and checks if there are other available tickets. If so, it returns in the calling number state; else, it will wait for an eligible ticket to be called. If the user, otherwise, scan his *QR Code* in time, the system checks the affluence of the store. If it's full, the system will wait for someone's check out, in order to check if some ticket can be called. Else, if the store isn't full, the system doesn't have to wait for a check out to check if there is some ticket eligible to be called. When at some time the store manager will close the store, the system begins its shutdown procedure.

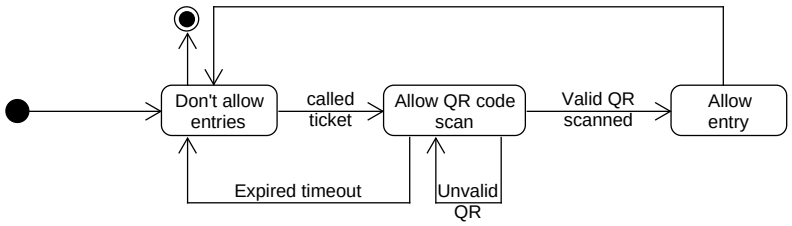

Figure 3: *QR Code scanner state machine for entering the store*

As decribed in *Figure 3*, The system doesn't allow entries until a ticket is called by the system described above. Then, the *QR Code* reader waits for a *QR Code* to be scanned. When a *QR Code* is passed under the optical reader, the system checks if it's a valid one; if so, allows the entry, otherwise notify the wrong *QR Code* and continue waiting for the right one. If the valid *QR Code* isn't scanned in time, the system blocks the entry, discards the current called ticket and waits for the next reservation to be called. Otherwise, if the valid *QR Code* is scanned in time, the system allows and registers the entry.

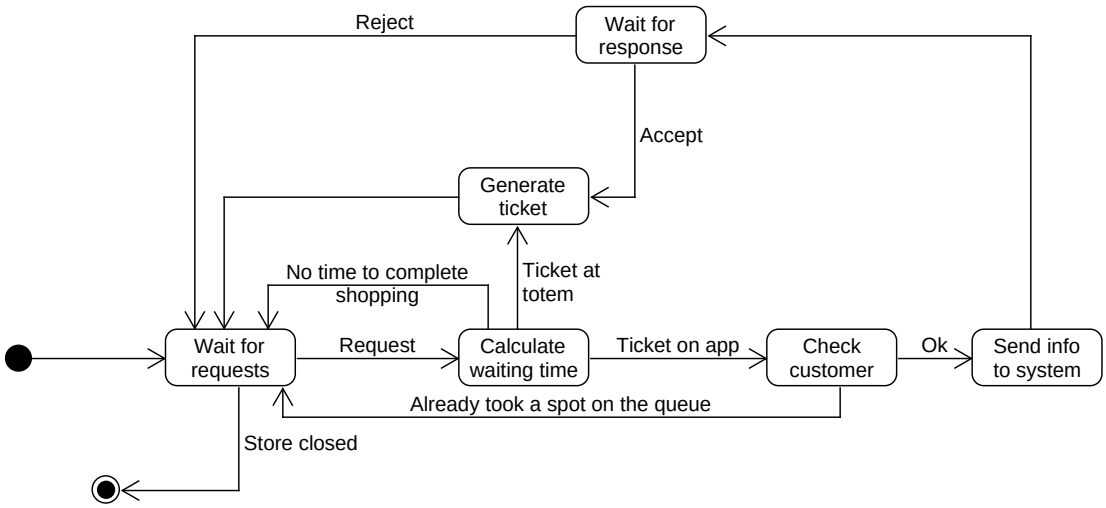

Figure 4: *Ticket generator system state machine for entering as soon as possible in the store*

In *Figure 4* is described the state machine of the generation of ticket to enter as soon as possible in the store. First, the system calculates the waiting time. Then, if there is time to complete shopping and the request arrives from a totem, the reservation is issued; else, if the reservation process happens on the app, and there is time to complete shopping, if the customer isn't already on the queue, the system elaborates data and two things happen, depending on waiting time: if it's low, an automatic response is sent, and the reservation processed; else, when the *ETA* is high, it's waited a response from customer, that can follow some suggestions from the system, or maintain his preference. Another situation may happen: there is no time to complete the shopping, since the customer may require to shop more time than how much is missing to the store closure. In this case, the ticket is not issued.

2.1.3 Scenarios

Scenario 1: User without application
Marta is a university student unfavorable to consumerism and for this reason she does not have a smartphone. She go to her favorite store to do shopping and thanks to our system she can still book a place in the queue to enter in the store. To do this, she just go to the totem placed near the supermarket and fill in the reservation with the requested data, that is the departments she intends to visit and an estimate of the time spent inside the store. In this way, the system will queue her to enter as soon as possible, providing her with an estimated entry time. Knowing this, Marta can still do other activities before being called to enter the store, so as not to create crowds at the store exit. As with the app, Marta has to scan the ticket at the entrance and exit. In this way the system can manage the queue and reservations.

Scenario 2: Fake *QR Code*
Jonathan arrives at the store and notices that he has many people in line before him. Jonathan is not a very patient guy, so having kept another *QR Code*, he tries to skip the line trying to scan it. The system recognizes that the *QR Code* is not valid, therefore it does not allow Jonathan to enter the store, forcing him to respect the queue.

Scenario 3: User books from the application
Adalgiso must go shopping but does not want to wait a long time outside the store and wants to be sure that he can enter the moment he arrives. So, he opens *CLup* on his smartphone, sets up a preferred means of transport and begins a reservation. He selects the store, the departments he wants to visit and the estimated time. Once the booking is complete, the app will send the notification to Adalgiso, inviting him to depart from his position to go to the store in time. When he will arrive, his number will be called shortly.

Scenario 4: Store manager have to smartly manage accesses
Apu is a store manager whose work is made more difficult by the current pandemic, since he has a small shop and no ways to manage crowds. Thanks to our application, now Apu is able to avoid assembles in front his store, he doesn't worry anymore about the number of people inside the store, and thanks to the real time statistics, he is able to regulate better the number of allowed people and booked ones for each department of his small store.

Scenario 5: Arrives near the store closing
Caterina arrives at the store near the closing, but her shopping would last over the closing time. To avoid that she waits in vain, a ticket isn't issued. So, Caterina reduces the departments that she should visit. She may have time to complete the shopping, but she have to hope that the crowd runs fast, or her ticket won't be called. Since there are possibilities to enter, the ticket is issued, but she is alerted that may not be able to enter the store.

2.2 Product Functions

In the previous chapter there were introduced, sketchily, the main features of the software in order to understand, in general, its functioning. Alternatively, in this section will be illustrated and described accurately all the functions that the software allows to do.

2.2.1 Getting a ticket

In order to manage the accesses to stores' buildings, it's used a system based on the "call of numbers". Each user in the queque has a unique *ID*, and when his number is called, the user is authorized to enter. This mechanism works also for the booking feature, since the system generates a number that won't be called until the beginning of the time slot selected by the user, in according to the previously reported criteria. Users can get a ticket both on the application and at a totem installed at the entry of the store (obviously, on the app users are required to select some specific store of their preference from a given list, sorted by the nearest one from the position obtained from the *GPS*). With the generation of the ticket, the user will get both an *ID* and a *QR Code* associate to it, to be scanned at the enter of the store. When getting a ticket on the application, users must be logged in with their credentials (if not, they must register at the services), while at the *totem* they can simply get the ticket without giving any data. At the store's *totem*, customers can only get a ticket to enter as soon as possible. On the app, customers can see at any moment their reservations, along with the expected calling time, and the time needed to reach the store by the preferred mean of transport selected in the settings. *ID* and *QR Code* are always available offline, while the other informations need an internet connection and a *GPS* signal to be updated (without them, inaccurate data may be visualized). On app, customers get a digital ticket with its associated *QR Code*; at *totem*, the ticket and the *QR Code* are printed. On both of them, there is displayed the expected time of call, based on the queue situation at the ticket issuing. This time is indicative, and can decrease if some customers doesn't enter the store. In case it's estimated that a reservation can enter the store after store's closing time, a warning on tickets is printed, along with the time store closes. This reservations are always generated, since, as previously explained, customers may give up line, reducing waiting times. Making the reservation, the user can indicate the department in which he is interested, and using the app can decide both to have a ticket to enter as soon as possible in the store or to plan his visit.

2.2.2 Calling process

The process to admit people without a booking in the store follows a *FIFO* logic: the first one that got a ticket, is the first to access the store. When called, a person is authorized to access the store. To register his entry, he is requested to check-in at the entry in a certain amount of time. Tickets related to the *FIFO* queue can be called as soon as the requested zones are available, and tickets

related to bookings can be called only after the start time of the booked time slot, and will be called around this time, when all the booked zones become available (to avoid starvation on calling bookings, other tickets requesting at least one booked zone won't be called until the booked tickets enters the supermarket). Obviously, in the queue may happen to be some reservations that needs to stay inside the store after the closing time. In such cases, these tickets are not called.

2.2.3 Check-in/Check-out

At the store entries and exits, users have to scan their *QR Code* to respectively Check-in and Check-Out in the store. If someone doesn't check-in in a prefixed amount of time, definable by the store manager, he'll lost his turn in the queue. *QR Code* scan is required for some important reasons: register someone's entry in the store for contact tracing purposes, to know the actual affluence of the store, to mine statistics and to check that the person entering the store is the one called, to avoid stoles of turns.

2.2.4 Plan a visit

The app allows customers to plan a visit in a store of their preference. At the end of the process of getting a reservation, customers can select to plan their visit to the store. After inserting the preferred day (in a 7 day range from the day of the request), and available the time needed by customer to complete the shopping, the app will propose customer some time intervals in which he can enter the store, to satisfy both estimated time of permanency and the maximum simultaneously bookings allowed in a department.

2.2.5 Managing store and single departments

Store managers are able to modify some settings of the store. Regarding the entire store, the editable parameters are:

- The available departments (with the constraint of at least one department)
- For each day of the week, if the store is closed or opened, and if opened the working hours;
- The maximum allowed bookings per week from a single customer;
- The maximum waiting time between a ticket calling and its scan at the entry

Instead, for each department, the store manager is able to define:

- The maximum number of simultaneous customers allowed in it;
- The maximum number of simultaneous booked customers allowed in it.

When a department setting is changed, the already made bookings will be rescheduled at the first available entry time, notifying each user of the change. The priority is to allow customers that booked the visit to enter in the same day, if it's possible. If any time on the same day can be allocated, the bookings will be cancelled and the customers notified. For people in queue, who got the reservation from the *totem* will be alerted from a store assistant of the possible delays, while application users will receive a notification of possible delays. A such operation may reduce in the already booked days the number of admitted non-booked clients. The app also allows store managers to see the real time situation in their stores.

2.2.6 Notify users

If some account have pending reservations, depending on the position of the customer (retrieved by *GPS*), and the preferred option of reaching the selected store (eg. on foot, by car or public transport), the app will notify the customer when he should depart from his position, by the selected mean, to arrive in time to enter the supermarket, without losing its turn and avoiding long waiting times. Notice that not all of the moving options may be available in all the places, since it depends on the used map service. This service is obviously available only with the internet connection and the *GPS* active.

2.2.7 Cancel and modify reservations

If the user can not reach the store in time, he can decide to cancel or modify the booked visit, or to leave his spot on the queue. So, the system deletes the customer from the queue (if inside it) and rearrange the last one. Also store managers are able to perform the same operations on customers' reservations.

2.2.8 Infer waiting time from users

The system can infer the average time spent by a user in the supermarket using previous visits as informations. If the client is taking a spot on the queue or booking a visit on the app, he will be asked to insert a reasonable duration of his visit; however, he can let the application to manage this parameter for him. If there aren't enough informations on the specific client, the app will infer this time from other clients' visits that did a similar shopping. Otherwise, it will find the information from client's previous visits. The same thing happens at the totem, where or the client inserts an estimation, or the system will use the whole stored informations.

2.2.9 Allow a fair management of the accesses

The system is able to manage in a fair way the process of releasing tickets and accepting bookings. In fact, depending on the store's closing time, and other informations in its possession, such as the number of people in queue, the estimated time of permanency and some other statistics, it's able to notify

customers about risk to not enter the store, if any, and to block the generation of new tickets, if there isn't enough time to do the shopping. The store manager is able to decide the maximum number of contemporary bookings allowed in each department, to avoid that it's really difficult to enter the store without a booking. Moreover, a user can make a limited number of bookings per week, and on app can't simultaneously take more than one spot on a queue.

2.2.10 Make suggestions

CLup is able to suggests customers better options than the chosen ones. In fact:

- If the customer is taking a spot in the queue, and the waiting times are really high, depending on customer's *GPS* position and the preferences inserted in the process, some alternative stores with lower waiting times (where it's considered also the time needed to reach store's position) may be proposed.
- If the customer is booking a visit, and the proposed time intervals are not of his interest, it's possible to request, from the same page, suggestions on other near stores with more booking possibilities.

2.3 User Characteristics

CLup gives access to two different sets of functionalities to make easier the life of two categories of users:

- **Customer**: *Covid-19* made challenging going to a store in the right time to avoid crowds and high waiting times. So, for customers, the aim of the application is to make less frustrating going at a supermarket, helping them to plan a visit, to take a spot on queue before exiting home and to be alerted when they should depart from home.

- **Store Manager**: The pandemic made life harder also for store managers. Now, with this system they can manage in an almost automatic and safe way the access to their stores.

2.4 Assumptions, Dependencies, Constraints

2.4.1 Domain Assumptions

DA1	Date and time on the devices on which *CLup* runs are always correct
DA2	Internet connection works always without errors
DA3	Customer's position retrieved by *GPS* is accurate
DA4	In each store, different objects belonging to the same category are in the same department
DA5	Totems always work properly and are not damaged
DA6	The customer's smartphone screen is not damaged and the *QR Code* is readable
DA7	The *Maps API* always calculate the optimal route
DA8	Every store has a unique name and address combination
DA9	*QR Code* readers are always working
DA10	The store capacity inserted by store manager is always correct
DA11	Customer tries to respect their estimate, without remaining over time
DA12	Each customer scans his *QR Code* at the enter and enters the supermarket only through the allowed entries.
DA13	Each paper ticket is not ruined and readable.
DA14	The working days and hours of the store inserted in the system are corrected
DA15	All users are ready to depart for the store when they are notified

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Customers can use the service using a personal mobile device (eg. a smartphone, a smartwatch or a tablet), or through a totem externally installed at each store to get a spot on the queue. Store managers can also use a device to manage the store. Here there are some mockups of the application. A complete description of all the *UI* will be in the Design Document, attached to this specification document.

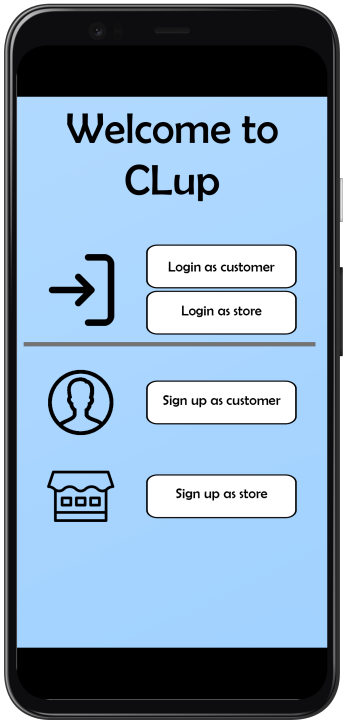

Figure 5: *Home page*

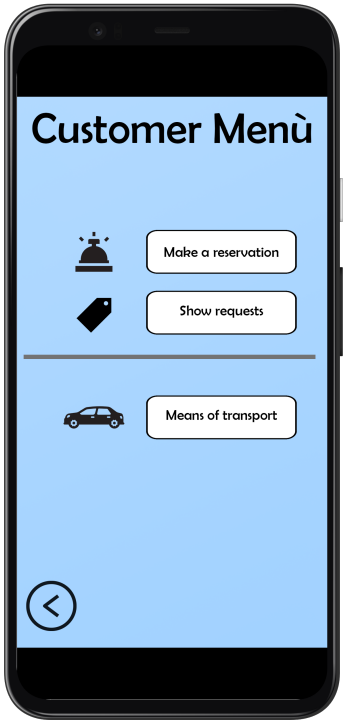

Figure 6: *Customer menu*

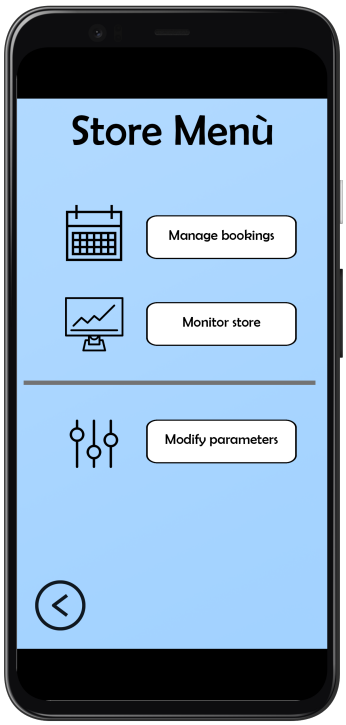

Figure 7: *Store manager menu*

Figure 8: *Modify store parameters*

3.1.2 Hardware Interfaces

The managers' side of the application doesn't require any special hardware requirement except for a network module installed in their device, since all of the information required are already in the system. Clients' devices, anyway, require a working *GPS* module, a working network module and a not broken display to scan the generated *QR Code* at the store entry. Instead, at the store are needed a device with a ticket printer and a working network module to generate and print tickets, some (at least one for entry and one for exit) optical scanners to scan *QR Codes* and another device with a working network module to manage ticket calling.

3.1.3 Software Interfaces

The system can send notification to the customer with regard to notify him to depart in time for reach the store and to avoid losing his turn in lineup.

To do this, the system uses a public *API* to get customer exact position by his *GPS* module and, so, calculate the optimal route from customer's position to the store. The exact position is also used to sort by distance the list of bookable store. Since the alert is sent by a notification, the system use the *API* of a notification service to provide notifications to customers. Moreover, notifications are used when the store manager modifies customers' reservations. Instead, the system uses the *API* of a mail service when a store manager contacts some client.

3.1.4 Communication Interfaces

It is necessary to develop a communication interface in order to allow the different systems to communicate one to each other, achieving the integration of the store's side and of the user's side.

3.2 Functional Requirements

3.2.1 List of Requirements

R1	The system must allow the customers to register
R2	The system must allow store managers to register their store
R3	The system must allow customers to log in
R4	The system must allow store managers to log in
R5	The system allows the customers to view their visits
R6	The system allows the customers to cancel their visits
R7	The system allows the customers to modify their visits
R8	The system allows the customers to select their favourite means of transportation
R9	The system allows customers to select the departments in which they are interested in doing shopping
R10	The system must let in customers only if it's their turn
R11	The system must consider the estimate shopping time inserted by customers
R12	The system must show the customers of the time periods in which they can enter the store, accordingly to the estimate of customers' shopping time
R13	The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
R14	The system can send notification to the clients
R15	The system is able to ask for the position of the customers
R16	The system permits to store manager to modify some critical parameters of the store related to customers' affluence management
R17	The system allows the manager to establish the maximum simultaneously allowed booked clients in a specific department
R18	The store manager can view the reservation of each client

R19	The store manager can modify the reservation of each client
R20	The store manager can cancel the reservation of each client
R21	The store manager can handle the working days and hours of the store, for each day of the week
R22	The system knows the situation in real time of each store
R23	The system takes trace of each customer entry and exit from the store
R24	The system contains a list of bookable stores
R25	The system is able to print a paper ticket
R26	The system can reasonably estimate the time needed from a customer to complete his shopping
R27	The system saves clients' tickets
R28	The system is able to smartly call clients with a ticket to enter the building depending on reservations and people inside the building
R29	The system is able to scan and analyse a *QR Code*
R30	The system is able to send emails
R31	The system knows how the maximum number of bookings allowed weekly per customer
R32	The system can automatically rearrange reservations, if necessary

3.2.2 Mapping with goals

- **G1: Allow customers to select a store and book a visit on a certain date and time from *CLup* app**
  - **R1**: The system must allow the customers to register
  - **R3**: The system must allow customers to log in
  - **R9**: The system allows customers to select the departments in which they are interested in doing shopping
  - **R11**: The system must consider the estimate shopping time inserted by customers
  - **R12**: The system must show the customers of the time periods in which they can enter the store, according to store's booking policies and the estimate of customers' shopping time
  - **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
  - **R15**: The system is able to ask for the position of the customers
  - **R22**: The system knows the situation in real time of each store
  - **R24**: The system contains a list of bookable stores
  - **R26**: The system can reasonably estimate the time needed from a customer to complete his shopping
  - **R27**: The system saves clients' tickets

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA3**: Customer's position retrieved by *GPS* is accurate
  - **DA8**: Every store has a unique name and address combination
  - **DA14**: The working days and hours of the store inserted in the system are corrected

- **G2: Allow customers to select a store and take a spot on the queue to enter as soon as possible the store from *CLup* app**
  - **R1**: The system must allow the customers to register
  - **R3**: The system must allow customers to log in
  - **R9**: The system allows customers to select the departments in which they are interested in doing shopping
  - **R11**: The system must consider the estimate shopping time inserted by customers
  - **R15**: The system is able to ask for the position of the customers

  - **R22**: The system knows the situation in real time of each store
  - **R24**: The system contains a list of bookable stores
  - **R26**: The system can reasonably estimate the time needed from a customer to complete his shopping
  - **R27**: The system saves clients' tickets

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA3**: Customer's position retrieved by *GPS* is accurate
  - **DA8**: Every store has a unique name and address combination
  - **DA14**: The working days and hours of the store inserted in the system are corrected

- **G3: Allow customers to take a spot on the queue from a physical ticket dispenser**

  - **R9**: The system allows customers to select the departments in which they are interested in doing shopping
  - **R11**: The system must consider the estimate shopping time inserted by customers
  - **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
  - **R25**: The system is able to print a paper ticket
  - **R26**: The system can reasonably estimate the time needed from a customer to complete his shopping
  - **R27**: The system saves clients' tickets

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA5**: Totems always work properly and are not damaged
  - **DA14**: The working days and hours of the store inserted in the system are corrected

- **G4: Allow customers to have suggestions on better store options**
  - **R1**: The system must allow the customers to register
  - **R3**: The system must allow customers to log in
  - **R8**: The system allows the customers to select their favourite means of transportation
  - **R11**: The system must consider the estimate shopping time inserted by customers
  - **R12:** The system must show the customers of the time periods in which they can enter the store, accordingly to the estimate of customers' shopping time
  - **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
  - **R15**: The system is able to ask for the position of the customers
  - **R22**: The system knows the situation in real time of each store
  - **R24**: The system contains a list of bookable stores
  - **R26**: The system can reasonably estimate the time needed from a customer to complete his shopping

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA3**: Customer's position retrieved by *GPS* is accurate
  - **DA7**: The *Maps API* always calculate the optimal route
  - **DA8**: Every store has a unique name and address combination
  - **DA14**: The working days and hours of the store inserted in the system are corrected
- **G5: Allow customers to decrease waiting times specifying departments they want to visit**
  - **R1**: The system must allow the customers to register
  - **R3**: The system must allow customers to log in
  - **R9**: The system allows customers to select the departments in which they are interested in doing shopping
  - **R11**: The system must consider the estimate shopping time inserted by customers
  - **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
  - **R23**: The system takes trace of each customer entry and exit from the store

  - **R26**: The system can reasonably estimate the time needed from a customer to complete his shopping

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA14**: The working days and hours of the store inserted in the system are corrected

- **G6: Allow customers users to manage their reservations**
  - **R1**: The system must allow the customers to register
  - **R3**: The system must allow customers to log in
  - **R5**: The system allows the customers to view their visits
  - **R6**: The system allows the customers to cancel their visits
  - **R7**: The system allows the customers to modify their visits
  - **R27**: The system saves clients' tickets

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors

- **G7: Allow customers to depart from their location in time to avoid waiting too much, and to avoid losing their turn for entering**
  - **R1**: The system must allow the customers to register
  - **R3**: The system must allow customers to log in
  - **R8**: The system allows the customers to select their favourite means of transportation
  - **R9**: The system allows customers to select the departments in which they are interested in doing shopping
  - **R11**: The system must consider the estimate shopping time inserted by customers
  - **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
  - **R14**: The system can send notification to the clients
  - **R15**: The system is able to ask for the position of the customers
  - **R23**: The system takes trace of each customer entry and exit from the store

  - **R26**: The system can reasonably estimate the time needed from a customer to complete his shopping
  - **R27**: The system saves clients' tickets

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA3**: Customer's position retrieved by *GPS* is accurate
  - **DA7**: The *Maps API* always calculate the optimal route
  - **DA8**: Every store has a unique name and address combination
  - **DA15**: All users are ready to depart for the store when they are notified

- **G8: Allow the store manager to know the real situation of people that are inside the building and in which department of his store**
  - **R2**: The system must allow store managers to register their store
  - **R4**: The system must allow store managers to log in
  - **R23**: The system takes trace of each customer entry and exit from the store
  - **R29**: The system is able to scan and analyse a *QR Code*

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA9**: *QR Code* readers are always working
  - **DA12**: Each customer scans his *QR Code* at the enter and enters the supermarket only through the allowed entries.

- **G9: Allow to grant a fair management of users that can access the building.**
  - **R2**: The system must allow store managers to register their store
  - **R4**: The system must allow store managers to log in
  - **R10**: The system must let in customers only if it's their turn
  - **R17**: The system allows the manager to establish the maximum simultaneously allowed booked clients in a specific department
  - **R21**: The store manager can handle the working days and hours of the store, for each day of the week

  - **R31**: The system knows how the maximum number of bookings allowed weekly per customer

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA11**: Customer tries to respect their estimate, without remaining over time

- **G10: Allow to manage optimally the influx in the building and avoid gathering inside it**

  - **R2**: The system must allow store managers to register their store
  - **R4**: The system must allow store managers to log in
  - **R10**: The system must let in customers only if it's their turn
  - **R16**: The system permits to store manager to modify some critical parameters of the store related to customers' affluence management
  - **R23**: The system takes trace of each customer entry and exit from the store
  - **R26**: The system saves clients' tickets
  - **R28**: The system is able to smartly call clients with a ticket to enter the building depending on reservations and people inside the building
  - **R29**: The system is able to scan and analyse a *QR Code*
  - **R32**: The system can automatically rearrange reservations, if necessary

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors
  - **DA9**: *QR Code* readers are always working
  - **DA6**: The customer's smartphone screen is not damaged and the *QR Code* is readable
  - **DA12**: Each customer scans his *QR Code* at the enter and enters the supermarket only through the allowed entries
  - **DA13**: Each paper ticket is not ruined and readable
  - **DA14**: The working days and hours of the store inserted in the system are corrected

- **G11: Allow the store manager to interact with customers and reservations**
  - **R2**: The system must allow store managers to register their store
  - **R4**: The system must allow store managers to log in
  - **R14**: The system can send notification to the clients
  - **R18**: The store manager can cancel the reservation of each client
  - **R19**: The store manager can modify the reservation of each client
  - **R20**: The store manager can view the reservation of each client
  - **R26**: The system saves clients' tickets
  - **R30**: The system is able to send emails

  - **DA1**: Date and time on the devices on which *CLup* runs are always correct
  - **DA2**: Internet connection works always without errors

3.2.3 Use Cases

3.2.3.1 UC1: Registration of a customer

Name	Registration of a customer
Actors	Customer
Entry Condition	Customer has the internet connection available and has accessed on the application on its device
Event Flow	1. Customer visualizes the initial page of the app 2. Customer clicks on "Sign up as customer" button 3. Customer inserts a username, a password, an e-mail, his name, his surname and his phone number as mandatory fields 4. The system checks the data of the customer 5. The system saves the information of the customer
Exit Conditions	Customer is successfully registered to the application
Exception	1. The username is already present in the system 2. The customer did not fill up all the mandatory fields with valid data If one or more of the above situations occur, the application will throw an error message and will return to the registration form page

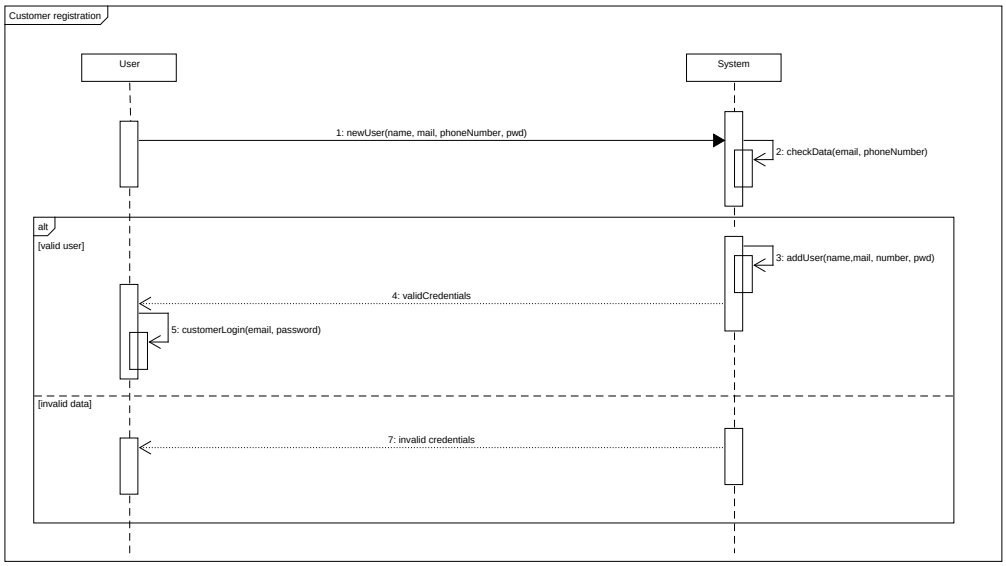

Figure 9: *Sequence Diagram of Customer Registration*

Required functional requirements:

- **R1**: The system must allow the customers to register

3.2.3.2 UC2: Registration of a store

Name	Registration of a store
Actors	Store manager
Entry Condition	Store manager has the internet connection available and has accessed the application on its device
Event Flow	1. Store manager visualizes the initial page of the app 2. Store manager clicks on "Sign up as store" button 3. Store manager compile all the mandatory fields concerning the store 4. Store manager loads a certification document which proves that it is a real store 5. The system validates the certification 6. The system confirms the registration of the store 7. The system saves the information of the store
Exit Conditions	The store is successfully registered to the application
Exception	1. The store is already present in the system 2. The store manager did not fill up all the mandatory fields with valid data 3. The certification is invalid If one or more of the above situations occur, the application will throw an error message and will return to the registration form page

Required functional requirements:

- **R2**: The system must allow store managers to register their store

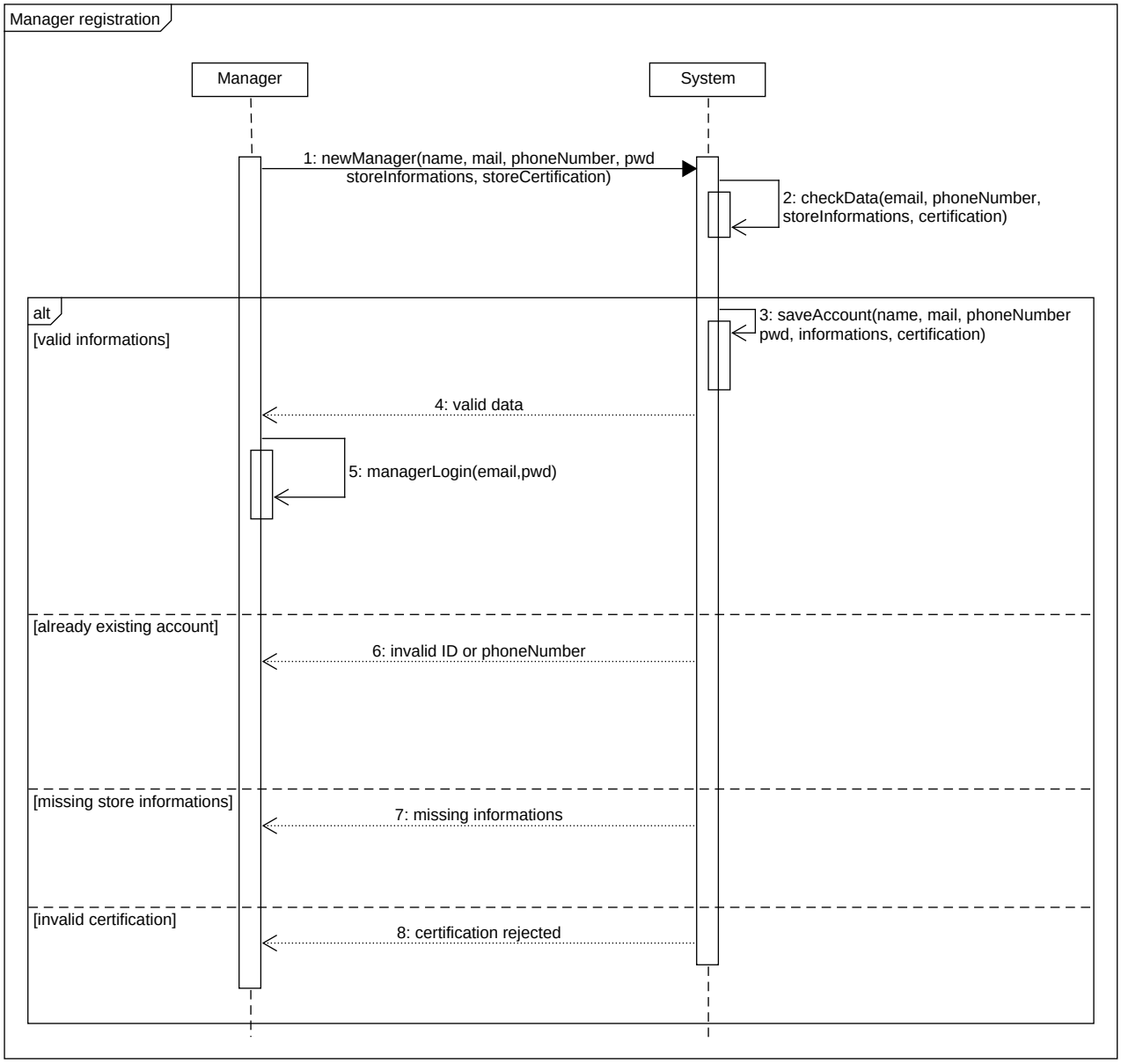

Figure 10: *Sequence Diagram of Store Manager Registration*

3.2.3.3 UC3: Login of a customer

Name	Login of a customer
Actors	Customer
Entry Condition	Customer is already registered to the application service
Event Flow	1. Customer accesses the application through its device 2. Customer clicks on “Login as customer” button 3. The system opens the “Login as customer” page 4. Customer compiles the fields “Username” and “Password” 5. Customer clicks on “Login” button 6. The system opens the “Customer menu” page
Exit Conditions	Customer has successfully logged in
Exception	1. Customer enters invalid email 2. Customer enters invalid password If one or more of the above situations occur, the application will throw an error message and will return to the “Login as customer” page

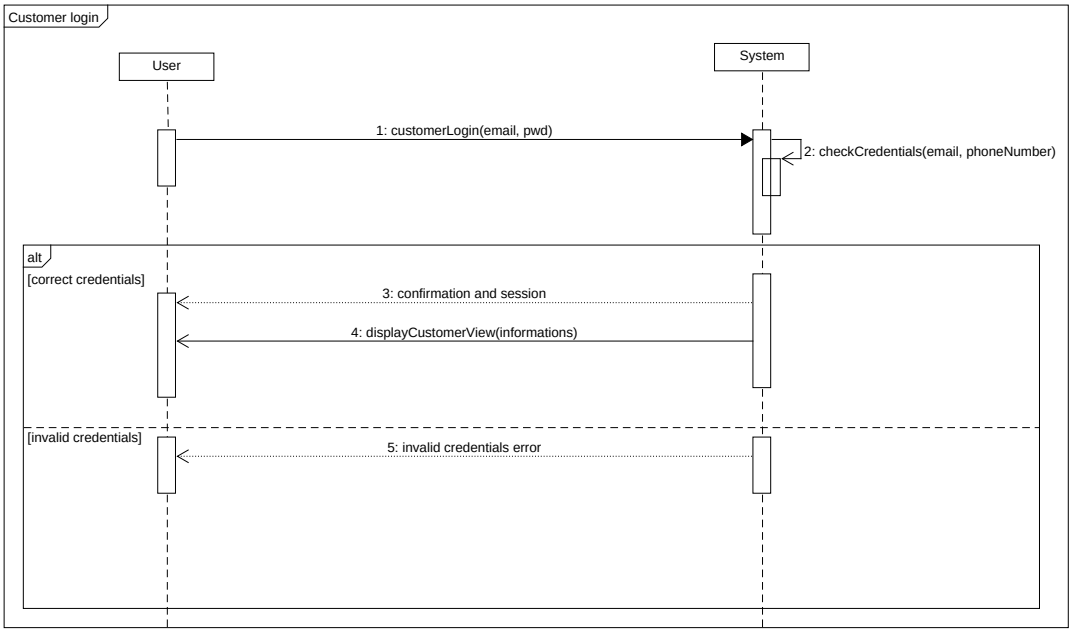

Figure 11: *Sequence Diagram of Customer Login*

Required functional requirements:

- **R3**: The system must allow customers to log in

3.2.3.4 UC4: Login of a store manager

Name	Login of a store manager
Actors	Store manager
Entry Condition	Store manager's store is already registered to the application service
Event Flow	1. Store manager accesses the application through its device 2. Store manager clicks on "Login as store" button 3. The system opens the "Login as store" page 4. Store manager compiles the fields "ID" and "Password" 5. Store manager clicks on "Login" button 6. The system opens the "Store menu" page
Exit Conditions	Store manager has successfully logged in
Exception	1. Store manager enters invalid ID 2. Store manager enters invalid Password If one or more of the above situations occur, the application will throw an error message and will return to the "Login as store" page

Required functional requirements:

- **R4:** The system must allow store managers to log in

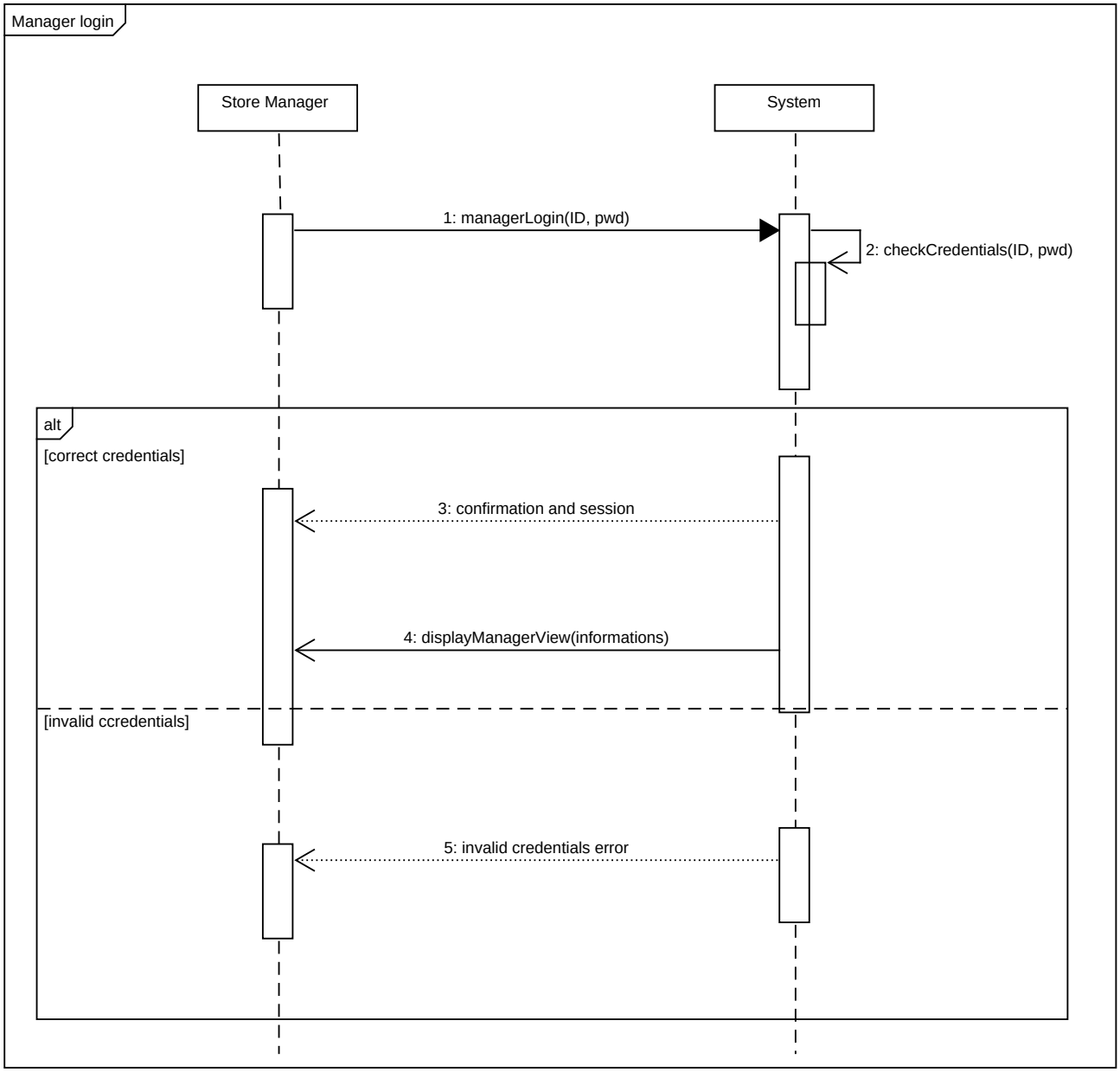

Figure 12: *Sequence Diagram of Store Manager Login*

3.2.3.5 UC5: Customer makes a reservation

Name	Customer makes a reservation
Actors	Customer
Entry Condition	Customer is already logged in the application service
Event Flow	1. Customer clicks on "Make a reservation" 2. The customer can see the list of stores in his city and can filter this list by choosing a specific chain from a drop down menu, selects a store and clicks on Next 3. Customer can see the list of all possible objects' category and can select some of them (optional), and then clicks on Next 4. Customer can estimate a duration for his shopping, or let the system to do it, and then clicks on Next 5. Customer can see two button, "As soon as possible" and "Choose a time slot" (a) if the clicks on "As soon as possible" button, the system will check that the customer isn't already on the queue, and will calculate the waiting time to enter. If a ticket with low waiting time is issuable, the system will generate it, saves it on both itself and user's app, showing it on the latter. Otherwise, the system suggests him other options, included the previously selected store. If any good, customer accepts one of the suggested store and gets the reservation, otherwise cancel the procedure. (b) if the customer clicks on "Book", the user can choose a time to enter the store among the time intervals available on the "Time slots" page to generate and save the reservation. If the customer is not satisfied with the proposed options, can click on the "Suggestion" button, and the system will retrieve him other stores, where he can book with the selected preferences. Selecting the store, the "Time slots" page updates with the new store's options. When the customer is satisfied with the selection, clicks on the Book button and the system will issue the reservation.
Exit Conditions	Customer has successfully made a reservation
Exception	1. Customer click on a timeslot no longer available or tries to book more than the set limit of reservations per week The application will throw an error message and will reload the "Time slots" page (updating it) 2. In the selected store where get a spot on the queue there isn't availability in the day, and there isn't any available near store If the above situation occurs, the application will throw an error and leave the client the possibility to book his visit.

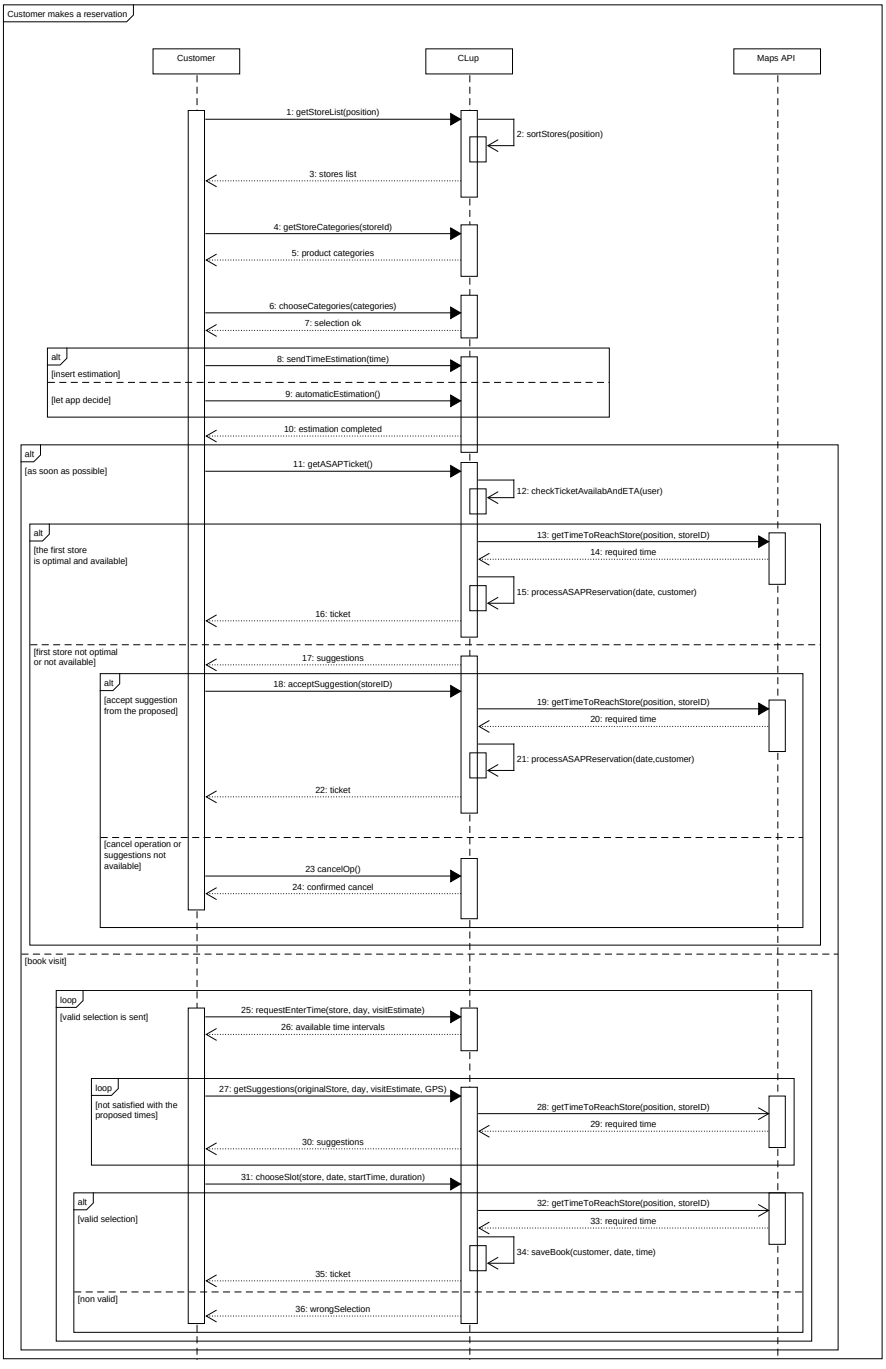

Figure 13: *Sequence Diagram of Getting Ticket from App Procedure*

Required functional requirements:

- **R9**: The system allows the customers to select some or all the departments in which the customers are interested in doing shopping
- **R12**: The system must show the customers of the time periods in which they can enter the store, accordingly to the estimate of customers' shopping time
- **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
- **R15**: The system is able to ask for the position of the customers
- **R22**: The system knows the situation in real time of each store
- **R23**: The system takes trace of each customer entry and exit from the store
- **R24**: The system contains a list of bookable stores
- **R26**: The system can reasonably estimate the time needed from a specific user to complete his shopping
- **R27**: The system saves clients' tickets

3.2.3.6 UC6: Customer visualizes reservations

Name	Customer visualizes reservations
Actors	Customer
Entry Condition	Customer is already logged in the application service
Event Flow	1. Customer clicks on "Show requests" button 2. The app show the list of bookings made and tickets requested 3. The customer select the desired option 4. The system opens the detail page of the selected option, that includes the "*QR Code*" and the number that should be called, among with the real time entry date and time and time needed to reach the store.
Exit Conditions	Customer can visualize the reservation
Exception	1. The client has not pending requests If the above situation happen, the app shows an error message and bring the client to the home page. 2. The internet connection is not available If the above situation happen, the app will load the local cached informations.

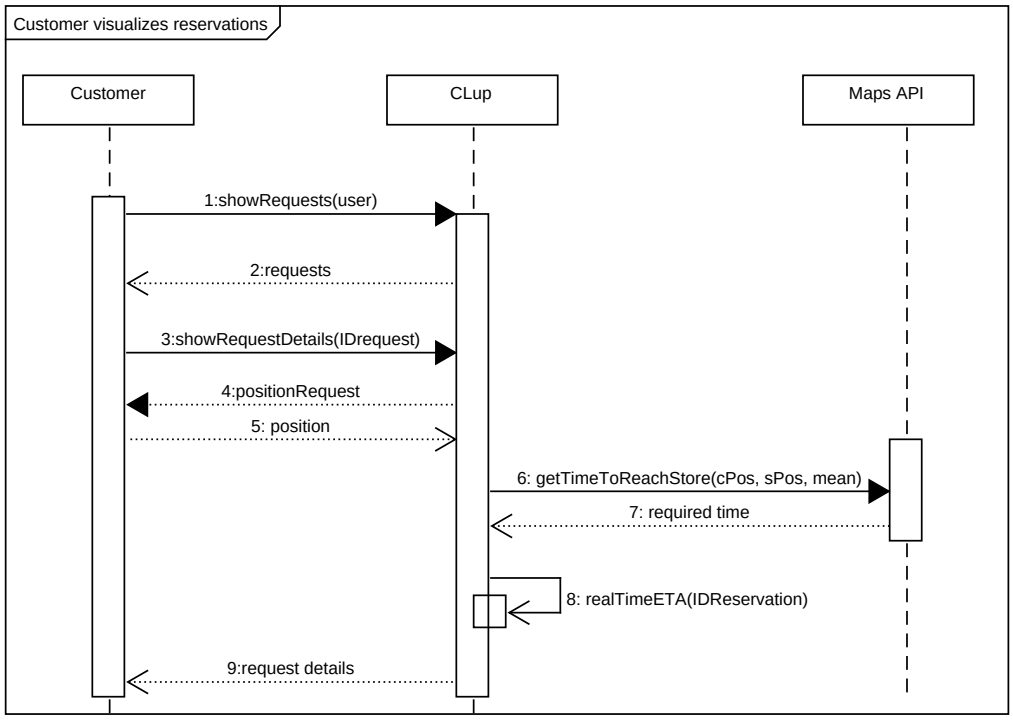

Figure 14: *Sequence Diagram of Visualizing Customers' Reservations in app*

Required functional requirements:

- **R5**: The system allows the customers to view their visits
- **R8**: The system allows the customers to select their favourite means of transportation
- **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
- **R15**: The system is able to ask for the position of the customers
- **R27**: The system saves clients' tickets

3.2.3.7 UC7: Manager modifies store parameters

Name	Store manager modifies store parameters
Actors	Store manager
Entry Condition	Store manager is already logged in the application service
Event Flow	1. Store manager clicks on "Modify parameters" button 2. Store manager can see both the parameters relatives to the whole store (eg. ID, Pwd, Working Hours for the selected day of the week), and each department with its parameters (max capacity and simultaneous allowed bookings) (a) Store manager can choose a day of the week using the "Change Day" button, if the currently selected is of no interest to him and have to modify some options on store's working hours. (b) Store manager modify one, more or none of the parameters of the store or of some departements (clicking on the pencil shaped button of the desidered department) (remind that only working hours are dependent on the selected day) (c) Store manager choose to add/delete some department respectively clicking on "Add a department" and on the bin shaped button of the desidered department (d) Store manager clicks on Save Changes 3. The system saves and processe the changes (notifying clients if necessary) and bring him back to "Store menu" page
Exit Conditions	Store manager has successfully updated store parameters
Exception	1. Store managers enters an invalid value for one or more parameters 2. The store manager tries to add an already existing department If the above situation occur, the application will throw an error message and will return to "Modify parameters" page

Required functional requirements:

- **R14**: The system can send notification to the clients
- **R16**: The system permits to store manager to modify some critical parameters of the store related to customers' affluence management
- **R17**: The system allows the manager to establish the maximum simultaneously allowed booked clients in a specific department
- **R21**: The store manager can handle the working days and hours of the store, for each day of the week

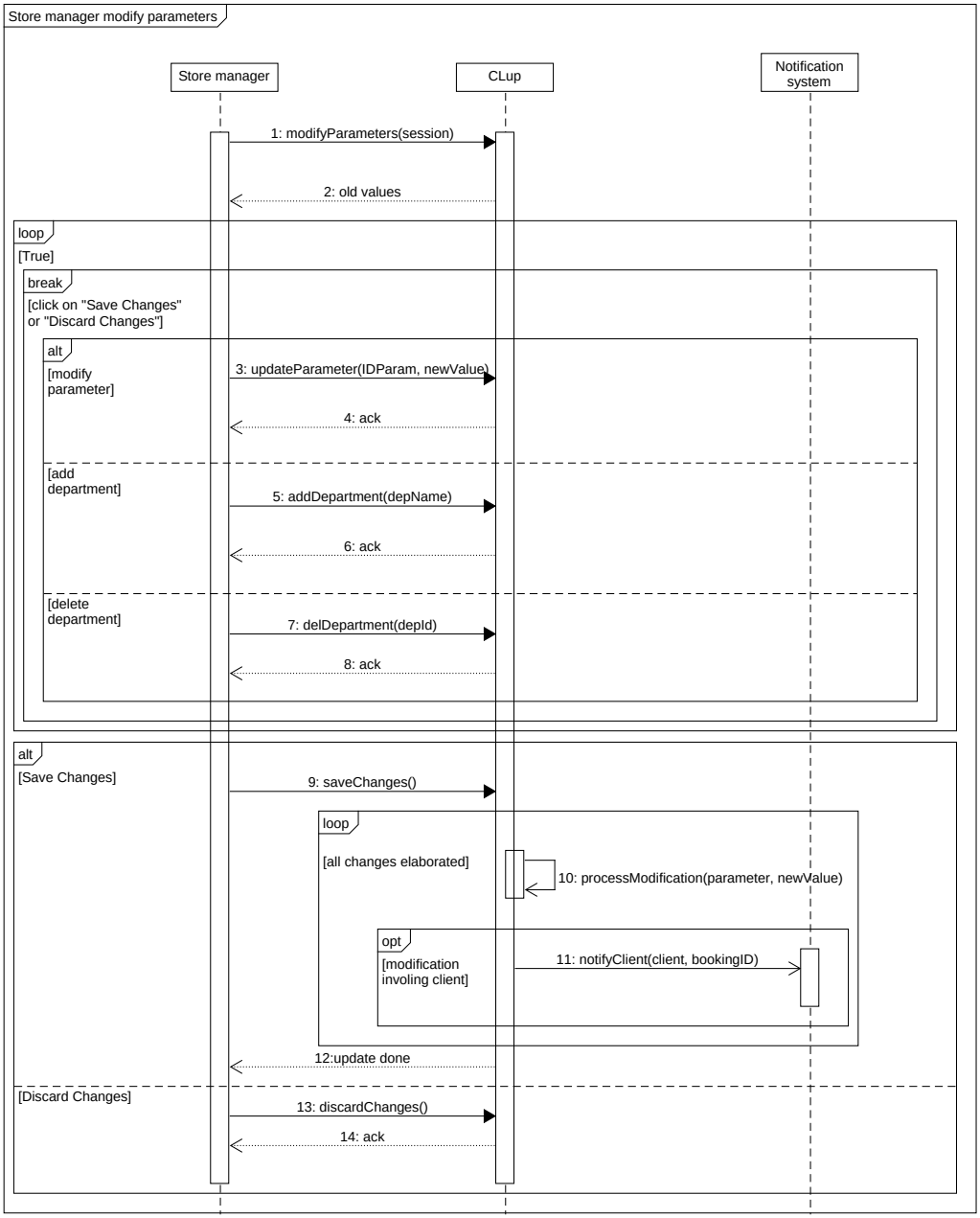

Figure 15: *Sequence Diagram of Visualizing Customers' Reservations in app*

3.2.3.8 UC8: The store manager monitors the store situation

Name	Store manager monitors store situation
Actors	Store manager
Entry Condition	Store manager has opened the app and is already logged in
Event Flow	1. Manager clicks on "Monitor store" button 2. The app shows a page with statistics on the store, including number of people inside the store, percentage of store occupation and the number of daily access to the building (a) If the manager wants, he can see the same statistics per store zone by clicking the "monitor zones" button. The system will show all the zones sorted by criticity
Exit Conditions	Store manager can see statistics on the store
Exception	None

Required functional requirements:

- **R9**: The system allows the customers to select some or all the departments in which the customers are interested in doing shopping
- **R23**: The system takes trace of each customer entry and exit from the store

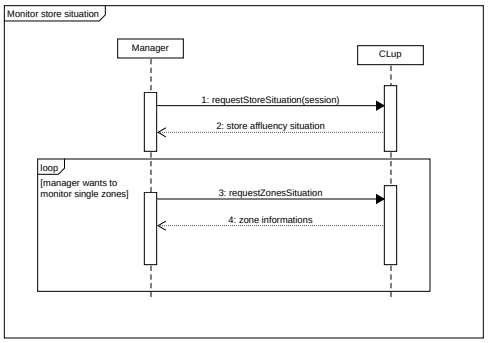

Figure 16: *Sequence Diagram of Visualizing Store and Departments Situation*

3.2.3.9 UC9: The store manager manages customers bookings

Name	The store manager manages customers bookings
Actors	Store manager
Entry Condition	Store manager has opened the app and is already logged in
Event Flow	1. Manager clicks on "Manage bookings" button 2. The app shows a list of reservations, with some of their details in preview (such as chosen date and time slot) 3. The manager can choose one of them, and the app will show some possibilities to the manager (a) The manager can press on the button "Contact client" to contact the client for some reasons. i. The app will show an interface where the manager can insert the email text ii. The manger clicks on the "Send" button to send the email iii. The app returns to the previous page (b) The manager can click on the button "Cancel booking" to cancel a reservation i. The app will show a dialog box where the manager can put-in an optional message, explaining the reasons of the cancel ii. The manager clicks on the "Delete button" iii. The app will show a confirmation box to ask if the manager is sure to proceed iv. The manager clicks on "Yes" to confirm the deletion, and notifies the customer v. The app closes the dialog box (c) The manager can choose to reschedule a booking i. The app will show a dialog box where the manager can choose a new time slot and insert a message explaining the reasons ii. The manager clicks on "Modify" to modify the booking iii. The app closes the dialog box
Exit Conditions	Store manager can manage the store and is able to edit reservations
Exception	1. There isn't any reservation If the above situation occur, the app will show a dialogue box with an error message.

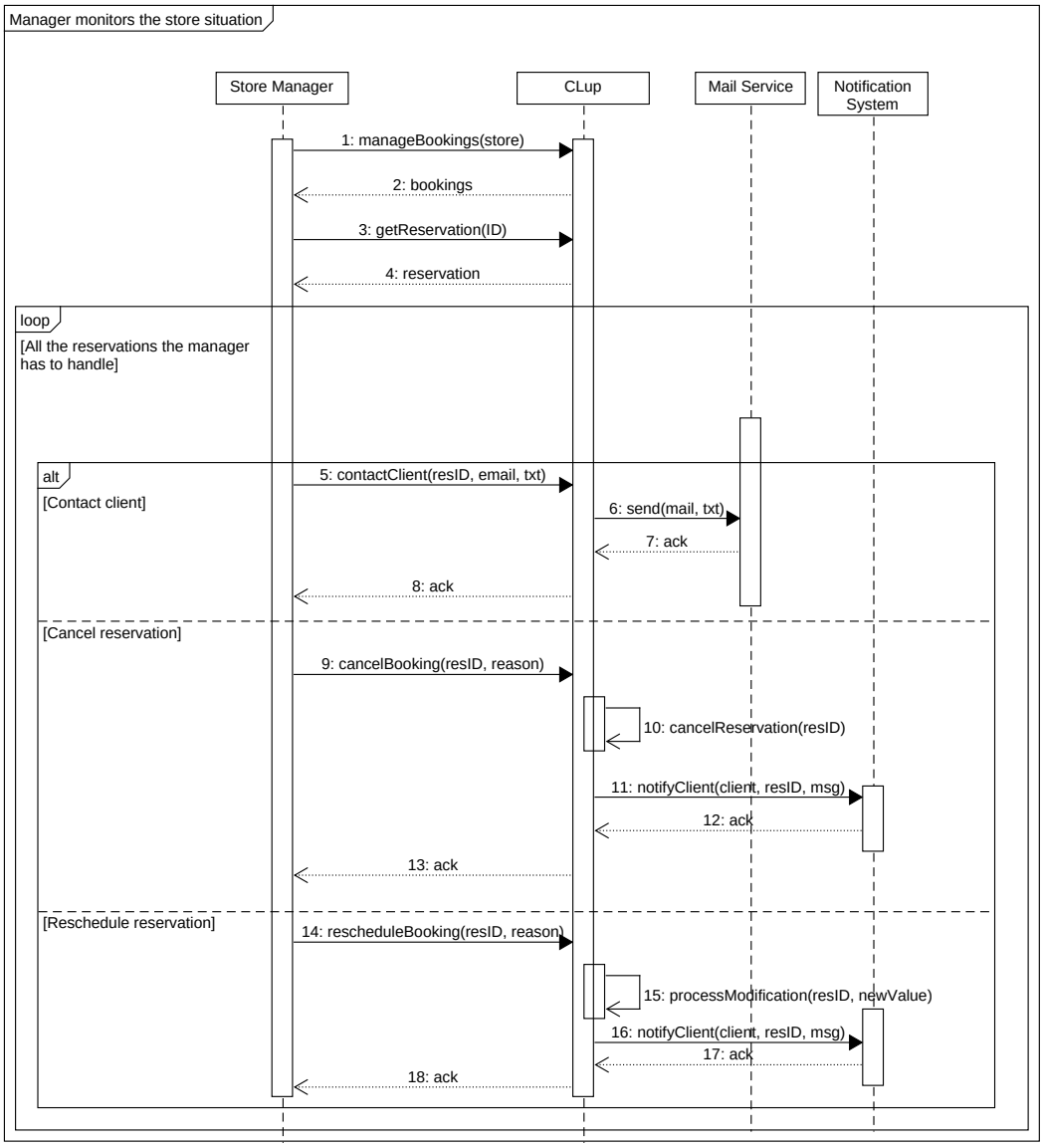

Figure 17: *Sequence Diagram of Visualizing Store and Departments Situation*

Required functional requirements:

- **R14**: The system can send notification to the clients
- **R18**: The store manager can view the reservation of each client

- **R19**: The store manager can modify the reservation of each client
- **R20**: The store manager can cancel the reservation of each client
- **R27**: The system saves clients' tickets
- **R30**: The system is able to send emails
- **R32**: The system can automatically rearrange reservations, if necessary

3.2.3.10 UC10: Customers reservations management

Name	Customers reservations management
Actors	Customer
Entry Condition	The customer has the application opened, is logged in and has at least one pending request
Event Flow	1. The user press on the "Show requests" button 2. The app shows a page with the requests made by the client 3. The user selects the desidered requests 4. The app will show the requests details 5. The user press the "edit" button (a) If the selected requests is a ticket, the customer can delete it pressing the "Delete button" i. The system will show a confirmation dialogue ii. The client press the yes button iii. The ticket is deleted and the app will return to the previous screen if there are other requests, or to the main menu otherwise (b) If the selected request is a booking, the client can both click the delete button, and follow the above procedure or can select the "modify button" i. If the modify button is selected, the app ask the customer all the parameters asked when making a reservation. This parameters can be modified, and at the end the modification confirmed, or discarded.
Exit Conditions	The client can modify his reservation
Exception	1. There isn't any reservation If the above situation occur, the app will show a dialogue box with an error message.

Required functional requirements:

- **R5**: The system allows the customers to view their visits
- **R6**: The system allows the customers to cancel their visits
- **R7**: The system allows the customers to modify their visits
- **R8**: The system allows the customers to select their favourite means of transportation
- **R9**: The system allows the customers to select some or all the departments in which the customers are interested in doing shopping
- **R11**: The system must consider the estimate shopping time inserted by customers
- **R12**: The system must show the customers of the time periods in which they can enter the store, accordingly to the estimate of customers' shopping time
- **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
- **R15**: The system is able to ask for the position of the customers
- **R22**: The system knows the situation in real time of each store
- **R26**: The system can reasonably estimate the time needed from a specific user to complete his shopping
- **R27**: The system saves clients' tickets
- **R32**: The system can automatically rearrange reservations, if necessary

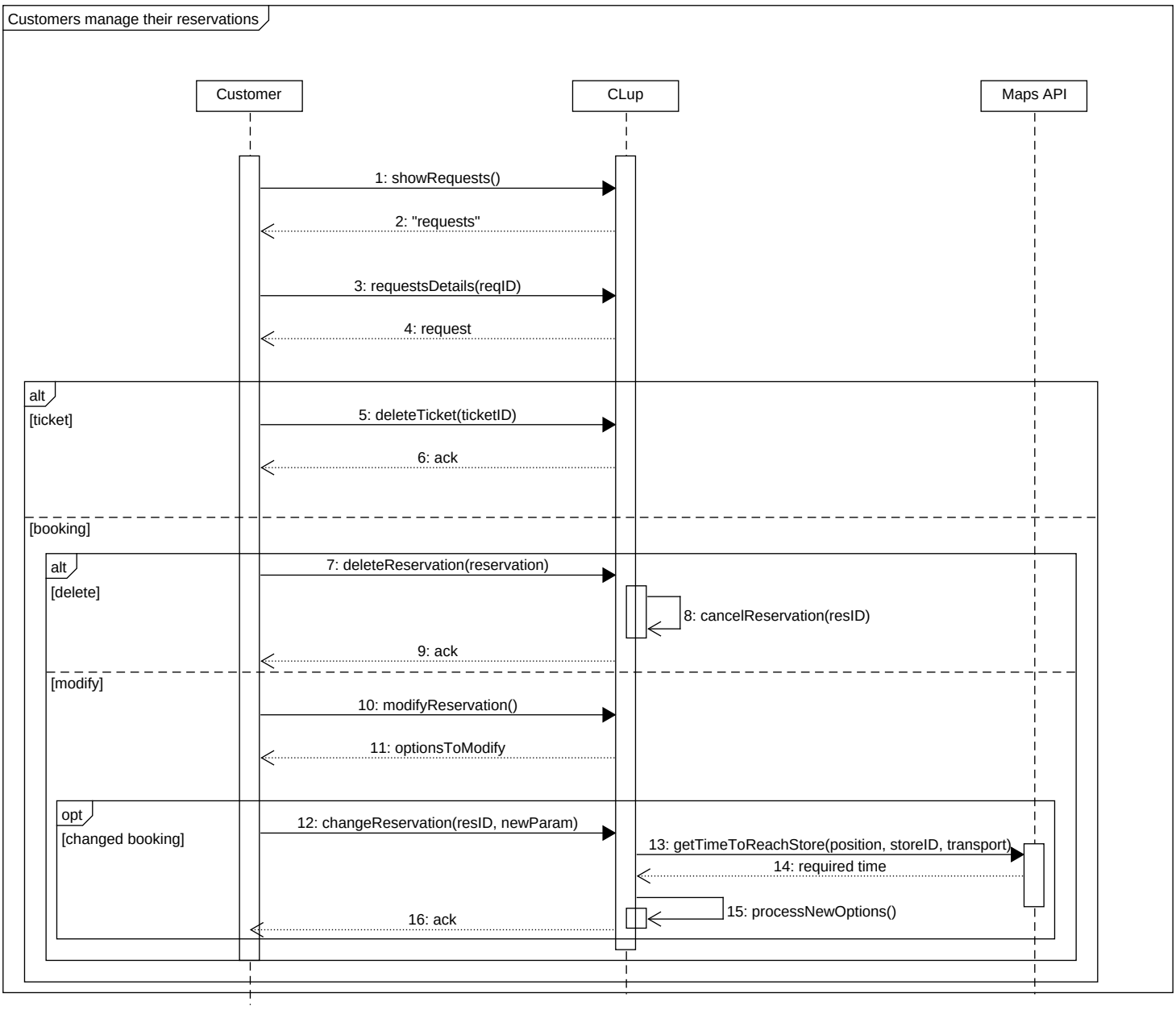

Figure 18: *Sequence Diagram of managing reservations customer side*

3.2.3.11 UC11: Customer get a ticket with the totem

Name	Customer get a ticket with the totem
Actors	Customer
Entry Condition	The customer is at the store and is using the totem
Event Flow	1. Customer clicks the button "Get Ticket" 2. Customers can see the list of all possible objects' categories and can select some of them (optional) 3. The customer enters the estimation of shopping time, or let the system to infer it 4. Client confirms the options selected and the system generates and prints the associated number, the *QR Code* and the *ETA*, with a warning if there is some risk on being able to enter only after the store is closed.
Exit Conditions	The client gets a ticket
Exception	1. Client has not completed the operation in the prefixed time 2. Client need more time to complete shopping than the remaining from the store's closing If the above situation occur, the application will throw an error message and will return to "Home" page

Required functional requirements:

- **R9**: The system allows the customers to select some or all the departments in which the customers are interested in doing shopping
- **R11**: The system must consider the estimate shopping time insert by customers
- **R13**: The system have to make a reasonable estimate of when a user with a spot on the queue is able to enter the store
- **R25**: The system is able to print a paper ticket
- **R26**: The system can reasonably estimate the time needed from a specific user to complete his shopping
- **R27**: The system saves clients' tickets

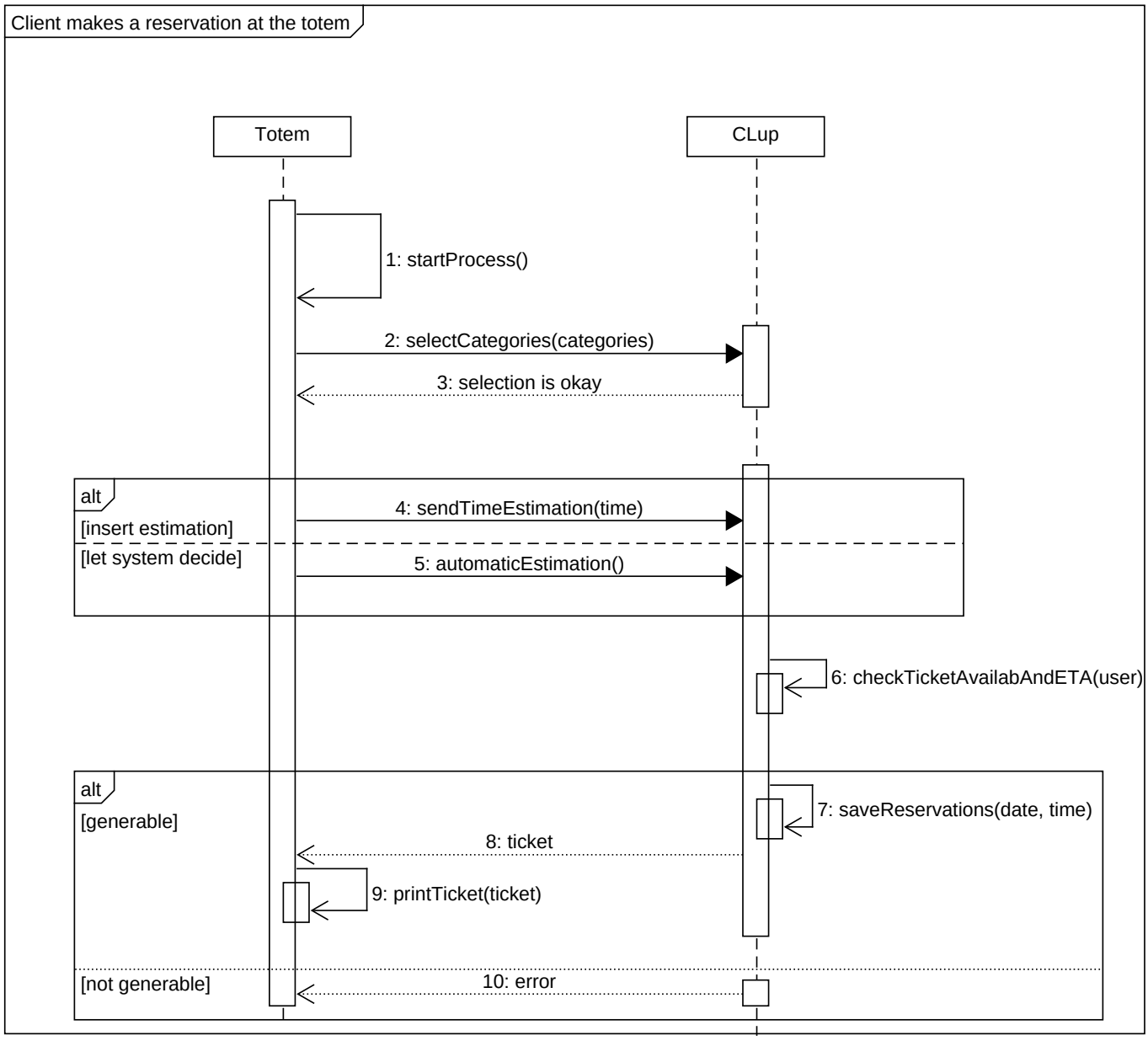

Figure 19: *Sequence Diagram of getting a ticket at a physical dispenser*

3.2.3.12 UC12: Customer selects the prefered means of transport

Name	Customer selects the preferred mean of transport
Actors	Customer
Entry Condition	Customer is already logged in the application service
Event Flow	1. Customer clicks the "Means of transport" button 2. Customer can see a list of means of transport 3. Customer select his preferred means of transport 4. The system gets his *GPS* positions and checks if the selected option is available. 5. If the selected option is available, the app saves the option. Otherwise, shows an error message and invite the customer to repeat again the operation.
Exit Conditions	The client selects the preferred mean of transport
Exception	None

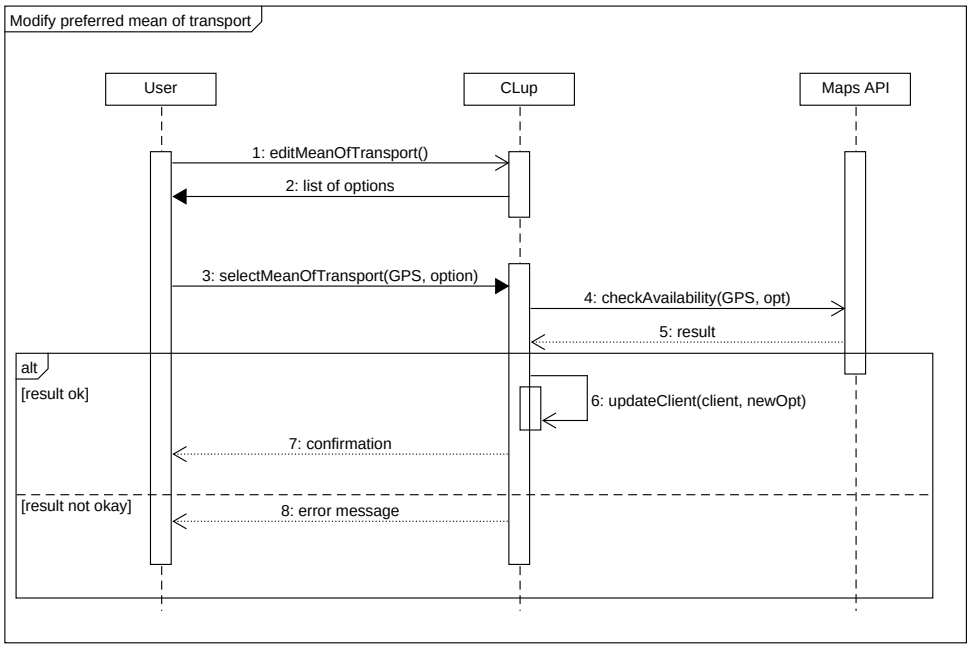

Figure 20: *Sequence Diagram of changing preferred mean of transport*

Required functional requirements:

- **R8**: The system allows the customers to select their favourite means of transportation
- **R15**: The system is able to ask for the position of the customers

3.3 Use Case Diagram

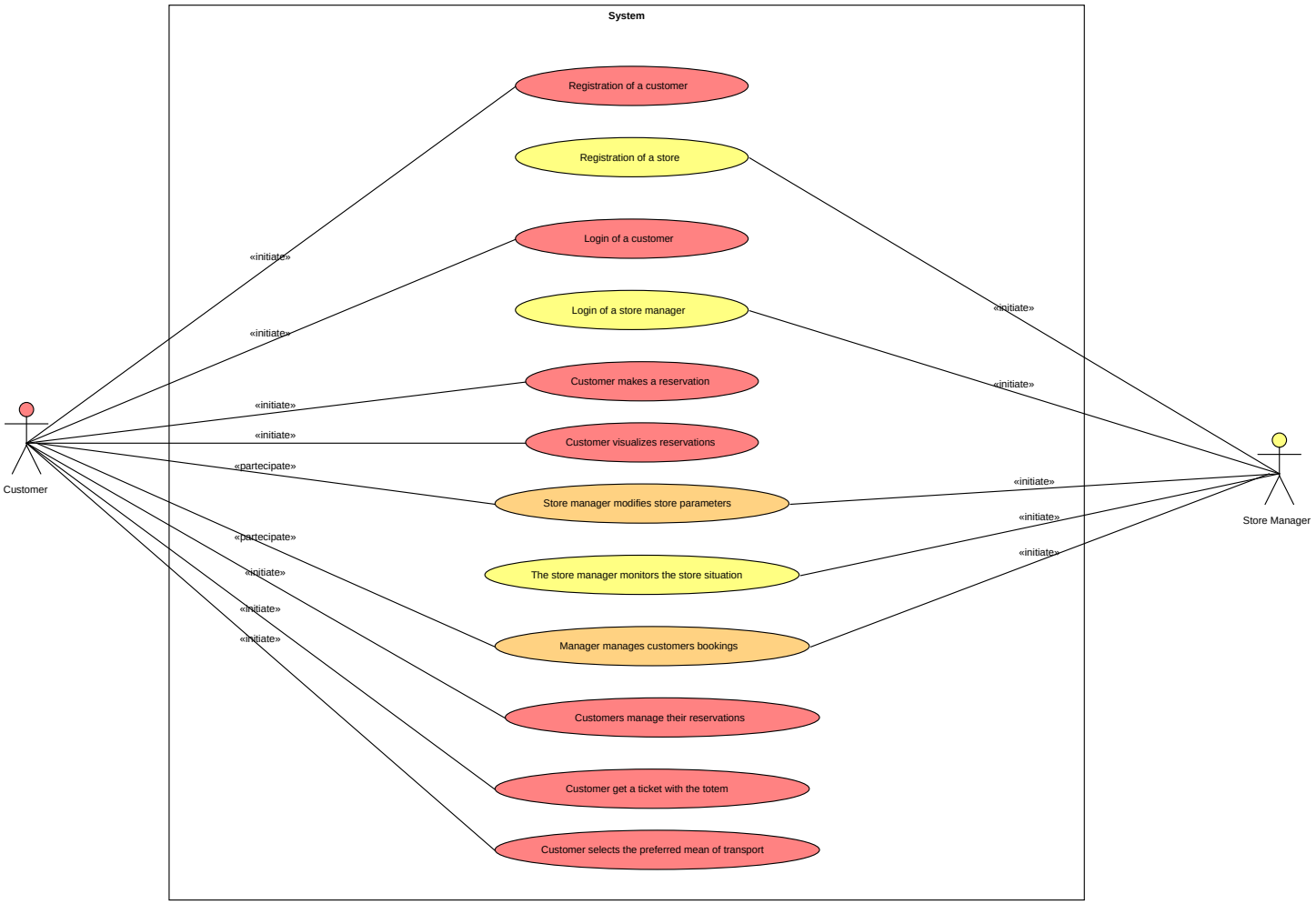

Figure 21: *Use Case Diagram*

3.4 Performance Requirements

CLup app is aimed to reduce gatherings due to the *Covid-19* pandemic. So, some components needs a certain responsiveness. Here there are described some parts of the system that are critical to the scope of entire system.

- **Requests saving system**: When a client got a reservation, the system needs to process and save it in a very low time, less than 5 seconds, since the requests are fundamental in making estimations about waiting times.
- ***QR Code* processing**: Each customer must scan his *QR Code* when he enters the store. This means that, to avoid delays and crowds of people, the component processing *QR Codes* must be really fast, and process them in no more than 5 seconds after the code is retrieved from the optical scanner.
- **Ticket calling system**: As for *QR Codes* processing, after a customer exits the store, the system must be able to process the next ticket to be called, if any, in a strict time, to avoid that people waits outside the store for long times. This process must not last more than 10 seconds.
- **User notifier**: The system must responsively notify users that they must depart for the store in time, with a security margin to avoid lost of turns.
- **Application data updates**: When the customer need to use the application for any purpose, the app must be responsive, and each object required over the network must be available in no more than 5 seconds from its request; if the retrieve from the net fails, the app will show the local stored data to avoid embarrassing situations (eg. the client must retrieve his *QR Code* to access the store and internet isn't available)
- **Totem ticket printing**: The totem must process each request and print the ticket in no more than 5 seconds.
- **Optical scanners**: Each optical scanner must take no more than 5 seconds to read and interpret a *QR Code* and its content.

The above reported performance constraints must be always verified; that implies the system must be highly scalable to respect these timings. There isn't any limit on the number of registered stores and users, so the system must respect these constraints with at least 200 simultaneously connected users per each store inside the system.

3.5 Design Costraints

3.5.1 Standards Compliance

System is compliant to some standards, such as:

- Generated *QR Codes* are compliant to *ISO/IEC 18004:2015 standard*
- Network messages are exchanged through the internet protocol *TCP/IP*

The same doesn't applies to requests of travel times and for mail sending, since they are achieved by *APIs* of some external services.

3.5.2 Hardware Limitations

To work properly, *CLup* needs some type of hardware, in base of the considered component.

- **Customer Device:** Customer needs to have a device with a working network module, a *GPS* module and a high resolution display for the *QR Code* to be scanned.
- **Totem:** the totem must have a working network module, a ticket printer and a module to make possible interfacing between customer and totem.
- ***QR Code* readers:** *QR Code* readers must have a working network module, and a optical scanner to scan *QR Codes*.

3.6 Software System Attributes

3.6.1 Reliability

The system must have a very low probability of failure (less than 0,0001%) to avoid inconveniences in crowd managing.

3.6.2 Availability

Due to the critic aspects the application handles, is required a 99,99% of system reliability. It means that the application will be down only for 52.60 minutes per year, an estimate that is acceptable. Having a greater downtime per year, may lead to inconveniences in handling the queue outside a store, bringing to a possible assembly of people.

3.6.3 Security

Each communication between client and server is made over a secure transport protocol (eg. TLS). For each request the system will authenticate the user so that everyone access only to data he is authorized. An encryption and decryption system must be implemented so that QRs' Content is encrypted to avoid someone can forge a malicious ticket. Moreover, each user password is stored using its hash value.

3.6.4 Maintainability

Since rules can change in every moment, the software must be developed in an extendible way, so that it's easy to add new functions required by new restrictions. Moreover, the software can be used even if after the pandemic. So, it might be required to implement new feature to make the system more appealing.

3.6.5 Portability

The application must be portable on the majority mobile OSes. For this reasons, the app will use a notification service accessible through the OS APIs. Furthermore, the software is thought so that is possible to implement different external services in different versions.

3.7 Additional Specifications

The system guarantees unicity of each customer's email and of each store manager's ID.

4 Formal Analysis Using Alloy

In this section a formal analysis of the previously described system is done, through the MIT's Alloy Tool.
This analysis doesn't aim to cover all the system aspect. For example, no dynamic analysis have been done. Instead, here is conducted a static analysis of the most critic part the system: the absence of overcrowding in stores, and the fairness in reservations management. In doing this, for sake of simplicity, some aspects are simplified: e.g. in validated reservations we consider the trivial case where each exit time is already defined at the entry, the queue is not managed, there's only a list of ticket to be called, and opening/closing times are not treated, only if a store is closed on some day.
In particular, the principal functionalities that have been tested are:

- In each store department, there aren't more people than its capacity, and the maximum booked customers parameter is respected
- There isn't any customer with more than one spot on a store queue
- Reservations status is coherent with the list of reservations each store have
- No reservations on closure day, no same reservations across different store
- Coherence between timestamps of reservations notification, calling, entering and exit
- Customers are always notified to depart in time for the store through notification

```
1  //SIGNATURES
2  abstract sig ReservationType{}
3  one sig Booked extends ReservationType{}
4  one sig ASAP extends ReservationType{}
5
6  abstract sig ReservationStatus{}
7  one sig Pending extends ReservationStatus{}
8  one sig Called extends ReservationStatus{}
9  one sig Discarded extends ReservationStatus{}
10 one sig Validated extends ReservationStatus{}
11
12 sig Date{}
13
14 sig Notification {
15   timestamp: one Int,
16   reservation: one Reservation
17 }
18
19 sig VisitStatistic {
20   visitedDepartments: set Department,
```

```
21    timeInside: Int
22  }
23
24  sig UID{}
25  sig password{}
26
27  abstract sig Registration{
28    uid: one UID,
29    pwd: one password
30  }
31
32  sig Customer extends Registration {
33    customerRes: set Reservation,
34    notifications: set Notification
35  }
36
37  sig StoreManager extends Registration {
38    store: one Store
39  }
40
41  sig Store {
42    storeDeps: some Department,
43    reservations: set Reservation,
44    calledReservations: set Reservation,
45    enteredReservations: set Reservation,
46    maximumBookingsPerClient: Int,
47    closedDates: set Date,
48    currentDate : Date
49  } {
50    reservations & calledReservations & enteredReservations =
        none
51    currentDate not in closedDates
52    all r: enteredReservations | r.status = Validated
53    all r: storeDeps.inside | r in enteredReservations
54    all r: storeDeps.inside | r.date = currentDate
55  }
56
57  sig Reservation {
58    status: one ReservationStatus,
59    date: one Date,
60    type: one ReservationType,
61    departments: some Department,
62    store: one Store,
63    callTimestamp: lone Int,
64    enterTimestamp: lone Int,
65    exitTimestamp: lone Int,
66  } {
67    all disj d1, d2: departments | d1 != d2
68    all d: departments | d in store.storeDeps
69    all s: Store | s != store implies no d: s.storeDeps | d in
```

```
    departments

  status = Validated iff callTimestamp != none and
    enterTimestamp != none and exitTimestamp != none and
  callTimestamp < enterTimestamp and enterTimestamp <
    exitTimestamp

  status in (Discarded + Called) iff callTimestamp != none
    and enterTimestamp = none and exitTimestamp = none

  status = Pending iff callTimestamp = none and
    enterTimestamp = none and exitTimestamp = none

  date & store.closedDates = none
  status = Validated implies this in store.
    enteredReservations
}

sig Department{
  maxBook: one Int,
  capacity: one Int,
  inside: some Reservation
} {
  #inside <= capacity
  #inside.type -> Booked <= maxBook
  maxBook <= capacity
  all disj r1, r2: inside | r1 != r2
  all r: inside | r.status = Validated
}

//FACTS
fact noRegistrationWithSameUID {
  all disj r1, r2: Registration | r1.uid != r2.uid
}


fact noASAPOtherTheDay {
  all s: Store | no r: s.reservations | r.type = ASAP and r.
    date != s.currentDate
}

fact OneToOneCustomerReservationsAndNotifications {
  (all disj c1, c2: Customer |
    c1.customerRes & c2.customerRes = none and
    c1.notifications & c2.notifications = none)
  and (all r: Reservation | one c: Customer | r in c.
    customerRes)
}

fact calledReservationNotInStore {
```

```
112   all s: Store |
113     all r: s.calledReservations |
114       no d: s.storeDeps | r in d.inside
115 }
116
117 fact oneASAPPerCustomer {
118   all s: Store | no disj r, r': s.reservations | one c:
      Customer | r in c.customerRes and r' in c.customerRes and
       r.date = r'.date and r.type = ASAP and r'.type = ASAP
119 }
120
121 fact noSameDepartmentAcrossDifferentStores {
122   all disj s, s' : Store | s.storeDeps & s'.storeDeps = none
123 }
124
125 fact onlyValidatedInsideTheStore {
126   all d: Department | all r: d.inside | r.status = Validated
127 }
128
129 fact noStoreWithoutManager {
130   all s: Store | one m: StoreManager | m.store = s
131 }
132
133 fact noCustomerWithMoreThanAllowedBookings {
134   all c: Customer |
135     all r: c.customerRes |
136       all s: Store |
137         #(r.store -> s) < s.maximumBookingsPerClient and r.
      type = Booked
138 }
139
140 fact allNotificationSentBeforeCallTime {
141   all n: Notification | one r: Reservation | n.timestamp < r
      .callTimestamp and
142   all r: Reservation | one n: Notification | n.reservation =
       r
143 }
144
145 fact noStoresWithSameReservation {
146   all disj s1, s2: Store |
147     (s1.reservations + s1.calledReservations + s1.
      enteredReservations) &
148     (s2.reservations + s2.calledReservations + s2.
      enteredReservations) = none
149 }
150
151
152 fact noStoreWithSomeOtherStoreReservation {
153   no s: Store |
154     all r: Reservation |
```

```
155       r.store != s and r in (s.reservations + s.
      calledReservations + s.enteredReservations)
156 }
157
158 fact noDepartmentWithoutStore {
159   all d: Department | one s: Store | d in s.storeDeps
160 }
161
162 fact reservationStatus {
163   all r: Reservation |
164     (r.status = Pending implies
165       r in r.store.reservations and
166       not r in r.store.calledReservations and
167       not r in r.store.enteredReservations) and
168     (r.status = Called implies
169       not r in r.store.reservations and
170       r in r.store.calledReservations and
171       not r in r.store.enteredReservations) and
172     (r.status = Validated implies
173       one v: VisitStatistic | v.timeInside = r.
      enterTimestamp - r.exitTimestamp and
174       v.visitedDepartments = r.departments and
175       not r in r.store.reservations and
176       not r in r.store.calledReservations and
177       r in r.store.enteredReservations
178       ) and
179     (
180       r.status = Discarded implies
181       not r in r.store.reservations and
182       not r in r.store.calledReservations and
183       not r in r.store.enteredReservations
184     )
185 }
186
187
188 fact onlyReservationsOfTheDay {
189   all s: Store | all r: s.calledReservations | r.date = s.
      currentDate
190 }
191
192 fact noCustomerOverlappingReservations {
193   all c: Customer | no disj r1, r2: c.customerRes |
194     r1.date = r2.date and (r1.enterTimestamp > r2.
      enterTimestamp and r1.exitTimestamp < r2.exitTimestamp)
195 }
196
197 //FUNCTIONS
198 fun getCustomerStatistics[s: Store, c: Customer]: set
      VisitStatistic {
199   {stat : VisitStatistic |
```

```
200     all r: s.enteredReservations |
201       r in c.customerRes and stat.visitedDepartments = r.
        departments and
202       stat.timeInside = r.exitTimestamp - r.enterTimestamp
203   }
204 }
205
206 //at this point of the specification document, no algorithm
        to intepretate departments'
207 //statistics has been described or implemented, so that here
         the function will return
208 //only the client visit time in a visit including the
        department,
209 //just to model the functionality of getting departments
        statistics
210
211 fun getDepartmentStatistic[d: Department, s: Store]: set Int
         {
212   {
213     i: Int |
214       all r: s.enteredReservations |
215         d in r.departments and i = r.exitTimestamp - r.
        enterTimestamp
216   }
217 }
218
219 pred checkModel() {
220 #Customer >= 3
221 #StoreManager >= 3
222 #Store >= 3
223 #Store.storeDeps >= 2
224 #Reservation >= 6
225 #Department >= 2
226 #Reservation.departments >= 2
227 #Store.closedDates >= 1
228 #Date >= 3
229 }
230
231 run checkModel for 12
```

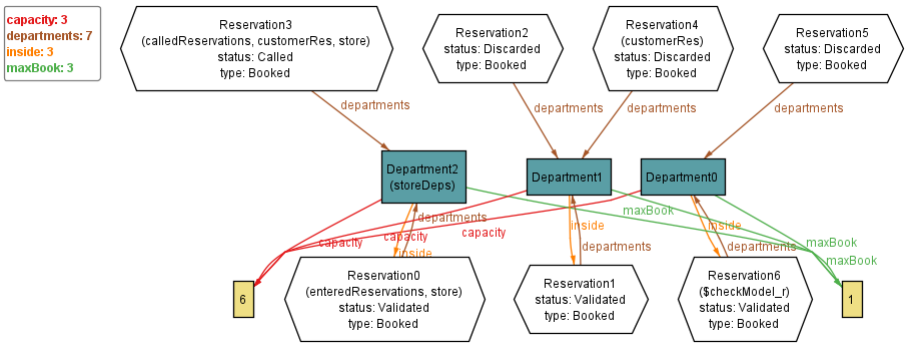

Figure 22: *Department Capacity respected. In this model, it's shown that the parameters of capacity and maximum allowed booked customers are respected*

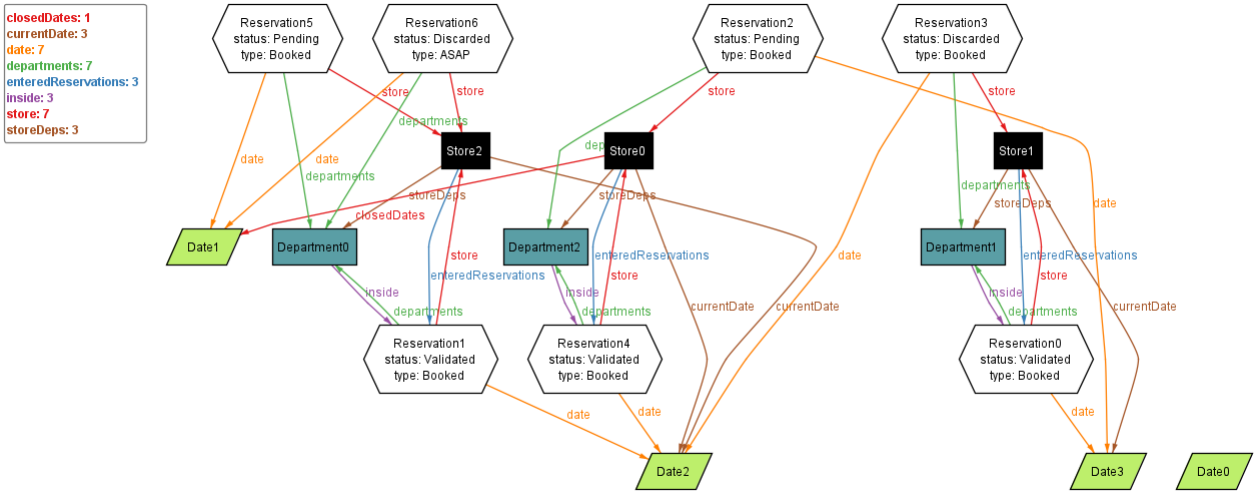

Figure 23: *Coherent reservations status. Here the intent is to show that the status of all the reservations is coherent*

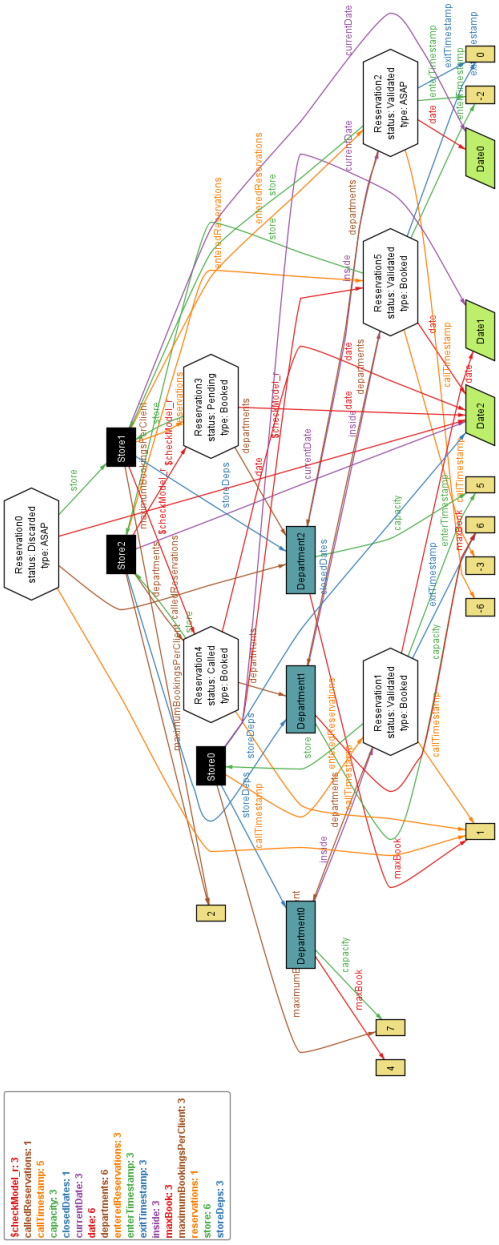

Figure 24: *Customer reservations. Here is shown how constraints on customer's ability to make a reservation are respected*

5 References

- Alloy Documentation: `https://alloytools.org/documentation.html`
- LaTeXDocumentation: `https://www.latex-project.org/help/documentation/`

6 Effort Spent

Task	Daniele Mammone's Hours
Introduction	2
Product Perspective	2
Product Functions	6
Domain Assumptions	3
State Charts	4
Use Cases	8
Class Diagram	1
External Interface Requirements	1
Mockups	0
Functional Requirements	6
Non-Functional Requirements	1
Formal Analysis Using Alloy	6
Scenarios	1
Revision	5
Document Drafting	3
Total Hours	49

Task	Gianmarco Naro's Hours
Introduction	2
Product Perspective	3
Product Functions	3
Domain Assumptions	3
State Charts	1
Use Cases	4
Class Diagram	3
External Interface Requirements	1
Mockups	0
Functional Requirements	7
Non-Functional Requirements	1
Formal Analysis Using Alloy	2
Scenarios	2
Revision	5
Document Drafting	6
Total Hours	43

Task	Massimo Parisi's Hours
Introduction	2
Product Perspective	2
Product Functions	2
Domain Assumptions	3
State Charts	2
Use Cases	7
Class Diagram	3
External Interface Requirements	4
Mockups	5
Functional Requirements	5
Non-Functional Requirements	1
Formal Analysis Using Alloy	2
Scenarios	0
Revision	2
Document Drafting	2
Total Hours	42