POLITECNICO
MILANO 1863

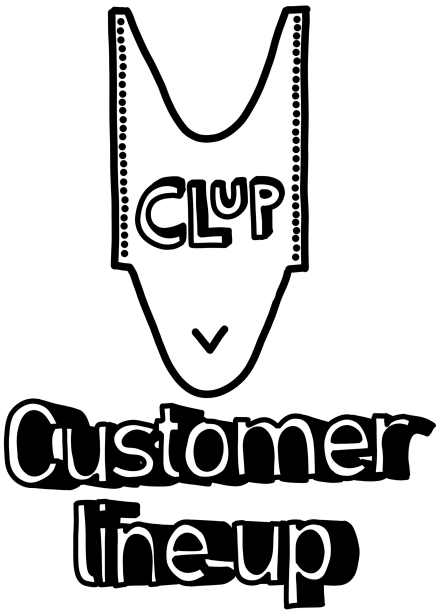
SOFTWARE ENGINEERING 2

Requirement Analysis and Specification Document

December 23, 2020

Authors:
Federico Mainetti Gambera, 10589654
Ludovica Lerma, 10522723

Professors:
Prof. Matteo Rossi
Prof.ssa Elisabetta Di Nitto



Contents

Table of Contents **2**

List of Figures **3**

List of Tables **3**

1 Introduction **4**
1.1 Purpose 4
1.2 Scope: description of the given problem 4
1.3 Definitions, acronyms and abbreviations 4

2 Overall description **6**
2.1 Details about the specification 6
2.1.1 Stakeholders 6
2.1.2 Main funcitons 6
2.1.3 Scenarios 6
2.1.4 Implementation choices and critical aspects 7
2.1.5 Product perspective 9
2.2 Goals 9
2.3 Domain Assumptions 10
2.4 Constraints 11
2.4.1 Regulatory policies 11
2.4.2 Hardware limitations 12
2.4.3 Interfaces to other applications 12
2.5 Use cases 12
2.5.1 Use cases tables 13
2.5.2 Use cases sequence diagrams 22
2.6 User interfaces 26
2.7 Application model 27

3 Requirements **28**
3.1 Functional requirements 28
3.2 Non functional requirements 32
3.2.1 Availability and accessibility 32
3.2.2 Security 32
3.2.3 Scalability 33
3.2.4 Accuracy 33

4 Formal analysis using alloy **34**
4.1 Alloy code 34
4.2 Alloy results 39

5 Effort spent **41**

6 References **41**

List of Figures

1 State diagram: ticket's life cycle . . . 9
2 User searches a shop - sequence diagram . . . 22
3 User lines up - sequence diagram . . . 23
4 User books a visit - sequence diagram . . . 24
5 Customer lines up - sequence diagram . . . 25
6 User interface structure - state diagram . . . 26
7 Application model - UML . . . 27
8 Alloy result 1 . . . 39
9 Alloy result 2 . . . 39
10 Alloy result 3 . . . 39
11 Alloy result 4 . . . 40
12 Alloy result 5 . . . 40
13 Alloy result 6 . . . 40
14 alloy model . . . 40

List of Tables

1 Definitions . . . 5
2 Abbreviations . . . 5
3 Acronyms . . . 5
4 Shop owner's goals . . . 10
5 Client's goals . . . 10
6 Domain assumptions . . . 11
7 All use cases . . . 13
8 Customer registration . . . 13
9 Shop owner registration . . . 14
10 User/Manager log in . . . 14
11 User searches a shop . . . 15
12 User gets more information . . . 15
13 User lines up . . . 16
14 User books a visit . . . 17
15 Customer lines up . . . 17
16 User/customer enters a shop . . . 18
17 User/customer exits the shop . . . 18
18 Manager registers new shop . . . 19
19 Manager updates shop information . . . 19
20 Manager checks shop status . . . 20
21 User cancels previously booked visit . . . 20
22 Customer cancels previously enqueuement . . . 20
23 Manager cancels customer's previously booked visit . . . 21
24 User cancels previous enqueuement . . . 21

1 Introduction

1.1 Purpose

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

Goals of this document are to completely describe the CLup system in terms of functional and non-functional requirements, to analyze the needs of the future users of our system in order to properly model it, to show the constraints and the limitations of our software and finally, to indicate the typical real world cases in which our application will actually be made use of.

This document is addressed to the *ones who take part in the process of development* of the first release of CLup and its aim is to describe the *interactions between the real world and the application.*

1.2 Scope: description of the given problem

In these trying times of global pandemic, such a common matter as going grocery shopping has become a relevant threat to public health. Nethertheless, grocery shopping still remains an essential need which has to be carried out: being so, avoiding crowding up either inside and outside of grocery shops to avoid any source of hazards uprises as a new main issue to focus on.

CLup is a software application that will be implemented in order to face this issue. The application proposes itself to help either the grocery shop owners to adequate to the new governmental rules and grocery shop customers to protect their own health.

In fact, Clup allows customers to *book online their shopping sessions* when and for how long they desire, letting them even specify the categories of items they are willing to buy: in this way the system will be able to grant the rules of social distancing more accurately and let the clients have an even safer experience through their shopping session.

Clup also allows those clients who are not much of a planner to *join virtual queues*, which are meant to substitute the physical ones, for their last minute shopping sessions. The system will permit them to monitor the queue and will even alert them when it's time to leave in order to reach the shop in time for their turn.

Finally, CLup will be of great help to grocery shop owners in *regulating the incoming influx of people* in their shop. As a matter of fact, CLup implements a QR-code based system of monitoring accesses to the shops that will decide to adherit to our service.

Also, CLup will be a convinient way to keep up with the evolving rules and law, giving managers the possibility to immediately customize every aspect of a shop.

1.3 Definitions, acronyms and abbreviations

In order to to avoid any misunderstanding here we present three tables to explain all the *definitions*, *acronyms* and *abbreviations* used throughout the document.

Definitions	
Customer	a person who is going to go grocery shopping and who is not registered in the CLup system
User	a client registered in the CLup system
Shop owner	a grocery shop owner or administrator who hasn't already adhered to CLup
Manager	a grocery shop owner or administrator who adhered to CLup
Totem	the piece of hardware that allows customers to be able to enqueue without having the app downloaded and which is placed outside of the shop it's correlated to

Table 1: Definitions

Abbreviations	
Gn	n^{th}-goal
Dn	n^{th}-domain assumption
Rn	n^{th}-functional requirement

Table 2: Abbreviations

Acronyms	
RASD	Requirement Analysis and Specification Document
API	Application Programming Interface
GPS	Global Positioning System

Table 3: Acronyms

2 Overall description

2.1 Details about the specification

The full specification can be found at the link: https://github.com/LudoLe/LermaMainettiGambera/blob/master/other%20documents/R%26DD%20Assignment%20A.Y.%202020-2021%20(2).pdf.

2.1.1 Stakeholders

The potential stakeholders interested in the deployement of the CLup system are obviously the actors that will interact with it and the nature of the benefits they would derive from CLup have already been discussed. Those actors, as already mentioned, are:

- **Customers**
- **Users**
- **Shop owners**
- **Managers**

Moreover, we can include:

- **Government**: even if indirectly, the government may be surely interested in an application which helps contain a global pandemic and prevent a generalized lock down with clear consequences for the economics of the country.

2.1.2 Main funcitons

As we mentioned in the product scope, the goals of Clup are fundamentally three:

1. help the shop owners to *regulate the influx of incoming people* in their shops;
2. allow customers to *join a virtual queue*;
3. allow customers to *book shopping sessions*.

However, it is important to recall herein that these purposes are primarily imposed by the actual situation of global pandemic, and that the true underlying goal, which defines the objective of our application, is to **safeguard citizens' health.**.

2.1.3 Scenarios

Here we are going to present some useful scenarios in order to describe some of the typical everyday-life situations in which CLup services may be helpful. Our intent is to give the reader a general understanding of the main features of the system.

User lines up Tommy is a student and he's preparing several exams for the upcoming session. He has just now noticed that his fridge is empty and he decides to go grocery shopping. Since he needs to study hard and can't lose any time, he wants to go to the nearest supermarket. He takes out his phone and opens CLup. He immediately looks up for the store he wants to go to and in a few clicks he has enqueued himself! Now it's time for Tommy to wait in the comfort of his house until he receives the notification from the app telling him that it is the right moment to head towards his destination. When he arrives he will find no queue outside the store: all time saved up for his studies!

User books a visit Mr. Zehng sticks to a really strict and rigid work schedule and lately he has found himself unable to go grocery shopping, because of long queues and early closing hours of the nearby supermarkets. Luckily for him, he happens to overhear one of his co-worker talking about the CLup app and about its "book a visit" feature to better help people manage their time. Mr. Zehng decides to try it and, after a few minutes, he has arranged a visit the very next day: he will no longer waste a minute in queues neither he'll be left without dinner.

Manager registers a shop Mr. Hochikawa is the owner of a famous japanese chain of supermarkets. During quarantine, shops are subjected to strict rules in order to safeguard the health of the customers. Indeed, Mr. Hochikawa has been facing a lot of difficulties lately because of how hard it can be to regulate and monitor traffic through all of his shops. Now that he has decided to turn to CLup it has become just a question of a click to monitor the ones he wants whenever he wants to. Mr. Hochikawa's life is much easier and his clients much safer now, thanks to Clup.

Customer lines up Mrs. Sunny would introduce herself as the proud grandmother of three beautiful kids. On Mondays, her grandchildren usually come visit her, so Mrs. Sunny goes to the nearby supermarket to gather what she needs to prepare a tasty meal. She's used to waiting for her turn standing outside in a queue and for someone of her age it's no simple task to stand on her feet for so long. This Monday morning, though, approaching the entrance of the store she is really surprised as she notices that there is no queue, and, getting closer, she notices a new installation blocking her way: a turnstile and a ticket machine next to it. She proceeds to read the instructions illustrated on a sign next to the ticket machine, follows them and easily manages to receive her ticket, which as the estimated time of wait printed on. Now Mrs. Sunny can safely wait her turn while sitting on the bench of the nearby park.

Manager updates shop's info Miss Lebon owns a little grocery shop downtown and lately she had a hard time keeping up with the continuous changes of the governmental rules and to keep informed her employees and her clients about all the necessary and sudden changes about the shop schedule or about how many people are allowed in the shop at a time. If in the past times this has been a very effortful and costly task for her it has now become much easier to do thanks to the Clup updating functions, which manage to keep everybody informed with almost no effort!

These scenarios presented highlight the main functionalities of CLup, however the application provides severals more.

2.1.4 Implementation choices and critical aspects

What follows is a short overview of the application workflow: from the real world to the machine computation. We decided to give space to this description because we considered of relevant importance for the reader to embrace a more general idea of CLup before getting in the further details of the application scopes. We take advantage of this overview also to specify some mechanisms through which our application will work.

- **Reach clients**

  To reach the entire population of possible clients we'll provide access to our system via website and mobile application. However, a hardware component will be installed nearby the adhering shops, a *totem*, so our system will be accessible by everyone.

- **Collect data from the people that interact with the system**

  It is clear that the service we intend to deliver can retrieve most of the fundamental data through the pieces of software that implements it -some of them supported by the totem-. However, the matter of keeping track of the people getting inside and outside each shop still remain unsolved. Hence, to support our system to retrieve this information some additional pieces of hardware will be necessary: a *QR-code scanner* and a *turnstile*. Indeed, when a user needs to enter or exit a shop, they'll simply need to scan the QR-code they are provided with.

  In any case, the main source of data we are provided with are the ones fed into the system by the users that book visists and enqueue themselves.

- **Tickets and tickets management system**

  Even if *booking a visit* and *enqueuing* seems to be two totally different functions to the end user, by the system perspective they are quite the same thing: a user who books a visit can be managed as if they were enqueuing for a specific hour and day, and a user who decides to enqueue themself, as if they were booking a visit for the first available slot of time.

  Although, analyzing the behavior of the queue the only real difference between visit-tickets and queue-tickets comes to the surface. In fact, a *queue* can be seen as a list that keeps track of users' turns to enter a shop: it's not a static structure, as it would be for a list of booked visits, but it evolves through time or because of particular events -such as users leaving queues or canceling booked visits or staying in shops longer than what they specified and others-. Finally we can affirm that visit-tickets cannot shift in time, they are scheduled, but queue-tickets can.

  So once a user enqueues themselves or books a visit the system generates the same kind of ticket but will manage it differently, according to what said above. Because of this, one state diagram is sufficient to represent the states in which both of the ticket can happen to be. The state diagram is shown in the picture [1]. A ticket can be in one of the following states:

  - **valid ticket**: when a ticket is created it is always *valid*, and that means that is waiting to be used to enter a shop.
  - **in use ticket**: a ticket is *in use* from the time it is used to enter the shop, to the time it is used to exit the shop.
  - **used ticket**: a *used* ticket is a ticket that has been successfully used to exit a shop.
  - **expired ticket**: *expired* ticket is a ticket that can't be used to enter a shop because the turn they were valid for has passed. This state could have been avoided by integrating it in the *used* state, but in order to have a better understanding of the life cycle of tickets we included it in here.

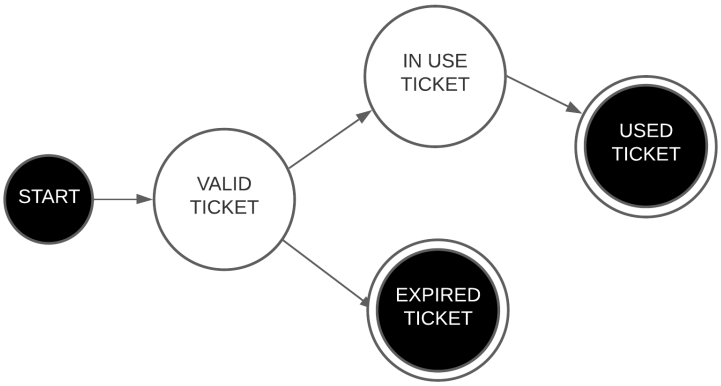


Figure 1: State diagram: ticket's life cycle

- **Making estimations**

  All the above mentioned data will be useful to make correct estimations about the status of the shop, and, thus, for the user-experience sake. The more we have reliable and early information about users' movements, the more our estimations will be precise and the general user-experience pleasant. Unfortunatly, since the data we are working with are provided by people, they are likely to be unrealiable.

  The best example of an estimation problem concerns queue management: without having control over people's actions, we cannot ensure precision regarding queue times estimations, because there may be sudden shifts of turns. The most important thing we can do is to build a good *algorithm* to handle the queues, imagining that in most scenarios it will have no problem and it will be efficient and accurate, but in some edge cases it may not be able to organize the enqueuements and booked visits in a perfectly no-time-waste way, therefore not optimizing the influx of people in the shops. To minimize the risk and to make the best estimations we intend to provide to the user the capability of helping the system with all the possible information, such as, for example, how long their permancence in the shop will last, or if they are intentioned to *cancel* reservations or enqueuements. We'll provide further details about the agorithm we intend to use in the Design Document.

2.1.5 Product perspective

CLup services will not be integrated with any other system, it will be implemented from scratch and in a three-tier architecture following a distributed Model-View-Controller design pattern.

The *Server-side* will host the entire logic of the application: the *controller*. The *Client-side* will take care of the presentation. Last, but not least, a dedicated server will host the database in which the system will store all the data.

2.2 Goals

The following tables show in details all the main goals that our application meets.

We managed to divide them in two main categories: goals related to the *shop owners/managers* and goals related to the *clients/users*. Also, we have assigned a unique identifier to each one of them in order to be able to refer to them later in the document.

Shop owners

Identifier	Goal
G1	Allow manager to sign on the system
G2	Allow a manager to sign in the system
G3	Allow a manager to register their store/stores on the system
G3.1	Allow a manager to register basic info about shops
G3.2	Allow a manager to divide their stores in areas
G3.3	Allow a manager to register the items in the areas
G4	Allow a manager to update their shops informations and settings
G5	Allow a manager to check the general status and the statistics of their shops
G6	Allow a manager to cancel a previously booked shopping session for a customer

Table 4: Shop owner's goals

Clients

Identifier	Goal
G7	Allow a user to sign on the system
G8	Allow a user to sign in the system
G9	Allow a user to search
G10	Allow a user to join a virtual queue
G11	Allow a customer to join a virtual queue from nearby the shop[1]
G12	Allow a user to book a shopping session at a grocery store
G12.1	Allow a user to select the duration of their shopping session
G12.1	Allow a user to select the time for their shopping session from a list of available options
G12.2	Allow a user to select categories of items they are willing to buy
G13	Allow a user to retrieve information about their previously booked visits
G14	Allow a user to retrieve information about current enqueuements
G15	Allow a user to retrieve information about shops
G16	Allow a users and customers to enter and exit stores with QR-codes
G17	Allow a user to exit a previously joined queue
G18	Allow a customer to cancel a previously booked visit
G19	Allow a user to cancel a previously booked visit[1]

Table 5: Client's goals

2.3 Domain Assumptions

The following table lists all the assumptions over which we cannot have control, but we assume as verified, and, as with the goals in the section above, we provide them with a unique identifier to keep track of them in the document.

[1] with the facility of a hardware support, it will be better specificed later on this document.

[1] We do not need a high level precision of users' positions in order to have a funcitoning system, but, obviously, the more precise the better the user expirience.

identifier	Domain Assumption
D1	Data provided from GPS is valid and accurate with regard to the real position of the users with a less than 50 meters radius [1]
D2	Google Maps' paths calculator makes correct time estimations
D3	A person cannot get in the store without having scanned a valid QR-code
D4	A person cannot get out of the store without having scanned a valid QR-code
D5	Each store registered on the system is provided with the necessary functioning hardware
D6	Data provided from managers is legit
D6.1	Managers provide the maximum number of people allowed in the store by the law
D6.2	Managers will provide the correct address of their store
D6.3	Managers will provide the correct schedule of their store
D6.4	Managers will provide the correct categories of items they sell
D6.5	Managers will correctly map items with the respective category
D7	Every category of a shop is matched with an area non overlapping with any other category area
D8	A customer will take the shortest path to the store
D9	Customers will stay in the store approximately the time they have claimed they would
D10	Customers will shop the items they have claimed they would
D11	Phone numbers are unique
D12	Emails are unique
D13	Notification sent to the users, may them be managers or clients, will be received and comprehended by them
D14	People have no malicious behavior against the app
D15	Managers' emails are correctly validated by PEC system (or any secure email system provided by the country of pertinence)
D16	Users are aware of the process required to scan a QR-code
D17	One QR-code scanned means that one and only one person enters the store

Table 6: Domain assumptions

2.4 Constraints

2.4.1 Regulatory policies

As CLup is an application that deals with real people data, it must follow strict rules and laws regarding the privacy of its users.

Here we show a list of mandatory aspects to consider in order to correctly release an application like CLup:

- protect application and users' interests with Terms ans Conditions;
- ask users permission to retrieve and use GPS data;
- regulate cookie management;
- telephone numbers and emails will not be used for commercial purpose;

- all data about user's booked visits or enqueued clients accessible for statistics and monitoring purposes is stored in an anonymous way.

2.4.2 Hardware limitations

In order to reach all kind of users and customers, our application wants to be the most accessible as possible. To achive this goal we intend to release various version of the application for different devices:

- **Mobile app**: Requires a smartphone with internet connection, either Android or Ios. It is not necessary to have GPS to run and use the application, but only to have access to some features.
- **Web app**: Requires a modern browser with javascript enabled and with an internet connection. The capability of retrieving user location is not necessary, but in order to have access to some features is mandatory.
- **Shops hardware**: Our intent is to have shop owners install hardware facilities that comprehend a totem, a QR-code scanner and an electronic turnstile. The totem will have internet connection and run a special version of the web application. The QR-code scanner and electronic turnstile will work togheter to allow access to the stores.

2.4.3 Interfaces to other applications

Our application will rely on external services to handle some features:

- Our sytem will interact wit Google Maps API.
- Interface with SMS gateway provider via standard SMS rest APIs.
- Interface with database.

2.5 Use cases

In this section we want to present all the major activities concerning the interaction between the application and the actors.

We will present a table for each use case and, for the most important ones, a list of sequence diagrams that will cover all the interaction between the actors and the system in a graphical way. For better clarity in the exposure and navigation throught the document we present an organized index (table [7]) of all the use cases with the relatives links to tables and, if present, to sequence diagrams.

	Description table	Sequence Diagram
1. Customer registration	tab.8	
2. shop owner registration	tab.9	
3. User logs in	tab.10	
4. Manager logs in	tab.10	
5. User searches shop	tab.11	fig.2
6. User gets more information	tab.12	
7. User lines up	tab.13	fig.3
8. User books a visit	tab.14	fig.4
9. Customer lines up	tab.15	fig.5
10. User/customer enters shop	tab.16	
11. User/customer exits shop	tab.17	
12. Manager registers new shop	tab.18	
13. Manager updates shop's information and settings	tab.19	
14. Manager checks shop status	tab.20	
15. User cancels previously booked visit	tab.21	
16. Customer cancels previously booked visit	tab.22	
17. Manager cancels customer's previously booked visit	tab.23	
18. User cancels previous enqueuement	tab.24	

Table 7: All use cases

2.5.1 Use cases tables

Name	Customer registration
Actors	customer
Entry condition	the customer is already on the front page of the web or mobile application
Flow of events	1. the customer selects the sign up option 2. the customer fills in all of the mandatory fields and then confirms 3. the system sends an SMS with a code to the customer's previously specified phone number and waits for an a answer 4. the customer validates his phone number by inserting the code 5. the systems sends an email with a code to the customer's previously specified address and waits for an a answer 6. the customer validates his email address by inserting the code 7. the system create a new account and redirects the new user to his personal home page
Exit condition	the customer successfully ends the registration process and becomes a new user
Exceptions	- timer expiration - wrong codes - email or phone number already registered or not valid - the customer insterts not valid informations in one or more mandatory fields

Table 8: Customer registration

Name	Shop owner registration
Actors	shop owner
Entry condition	the shop owner is already on the front page of the web or mobile application
Flow of events	1. the shop owner selects the sign up option 2. the shop owner specifies he wants to sign up as a manager 3. the customer fills in all of the mandatory fields[2] and then confirms 4. the system sends an SMS with a code to the shop owner's previously specified phone number and waits for an a answer 5. the shop owner validates his phone number by inserting the code 6. the systems sends an email with a code to the shop owner's previously specified address and waits for an a answer 7. the shop owner validates his email address by inserting the code 8. the systems validates the email address via PEC system 9. the system create a new account and redirects the new manager to his personal home page
Exit condition	the shop owner successfully ends the registration process and become a new manager
Exceptions	- timer expiration - wrong codes - email or phone number already registered or not valid - PEC email not verified - the customer insterts not valid informations in one or more mandatory fields

Table 9: Shop owner registration

Name	User/Manager log in
Actors	user or manager
Entry condition	the actor is already on the front page of the web or mobile application
Flow of events	1. the actor selects the sing in option 2. the actor compiles the email and password fields and confirms by clicking on the log in button 3. the system redirects the actors to the respective home pages
Exit condition	the actors are successfully redirected to their respective personal home page
Exceptions	- email or password aren't correct

Table 10: User/Manager log in

Name	User searches a shop
Actors	user
Entry condition	user logged in and on his web or mobile personal home page
Flow of events	1. user selects the search a shop option 2. the system gets the user's position using GPS and after retrieving the shops adhering to CLups near him, shows them in a map and, also, in a list 3. the user either types the name of the shop they want to find in the search bar or navigates the map or scrolls the list 4. the user selects a shop 5. the application retrieves all the necessary information about the selected shop's status and information, displays them to the user with the list of possible action the user can take: book a visit or enqueue or get more info
Exit condition	the user successfully finds the shop he was looking for
Exceptions	- the shop searched doesn't exists or isn't adhering to CLup - GPS is off
Notes	an user searches for a shop in order to get general informations or check its status or to enqueue himself or to book a visit

Table 11: User searches a shop

Name	User gets more information
Actors	user
Entry condition	the user is successfully at the exit condition of "user searches a shop" use case (table 11)
Flow of events	1. the user selects the get more info option 2. the system retrieves further details about the shop and displays them to the user 3. the user visualizes the informations
Exit condition	the user can visualize the information needed

Table 12: User gets more information

Name	User lines up
Actors	user
Entry condition	the user is successfully at the exit condition of "user searches a shop" use case (table 11)
Flow of events	1. the user selects the line up option 2. the user specifies the estimated time he thinks he will spend at the shop 3. optionally the user either activates GPS and specifies by what mean they are going to reach the shop or specifies how much time he needs to reach the shop 4. if the store has divided his store in categories, the user can optionally specify what categories he is interested in 5. the system adds the user to the queue and displays a constantly updated page with all the relevant information about the enqueuement
Exit condition	the user is correctly enqueued and needs to wait for his turn
Exceptions	- the enqueuement is not possible - GPS is not active
Notes	the page with all the enqueuement information can be also reached by the user at any moment from his personal home page when he is currently enqueued

Table 13: User lines up

Name	User books a visit
Actors	user
Entry condition	the user is successfully at the exit condition of "user searches a shop" use case (table 11)
Flow of events	1. the user selects the book a visit option 2. the user specifies the estimated time he thinks he will spend at the shop 3. if the store has divided his store in categories, the user can optionally specify what categories he is interested in 4. the system retrieves the possible slots and displays the options to the user 5. the user chooses the slots he prefers and confirms 6. the system registers the visit, sends confirmation to the user and redirects him to a page with all the relevant infromation about the booked visit
Exit condition	the user has correctly booked a visit
Exceptions	- the reservation is not possible - the user can't find any available slot
Notes	-the order of the various information requested to the user is important in order to let the system decide what possible slots are available -The page with all relevant information about the visit is easily accessible: from the user's personal home page there's an option to check all the reservation, this option leads to a page with all the reservations and by selecting one the user can reach the said page

Table 14: User books a visit

Name	Customer lines up
Actors	customer
Entry condition	the customer is at the entrance of the shop and have read and understood the functioning of the hardware
Flow of events	1. the customer visualize information about the status of the shop 2. the customer select the enqueue option on the totem 3. the customer specifies the estimated time he thinks he will spend at the shop 4. the system books a visit for the customer at the first available slot 5. the totem retrieves the visit information and prints a ticket with them 6. the customer takes the ticket
Exit condition	the cusotmer succesfully retrieves his ticket and has a booked visit registered
Exceptions	- enqueuement is not possible - hardware is not functioning properly

Table 15: Customer lines up

Name	User/customer enters the shop
Actors	user, customer
Entry condition	have previously enqueued or have previously booked a visit
Flow of events	1. the actor arrives at the shop 2. the actor correctly places the QR-code on the QR-code scanner facility at the entrance of the shop 3. the system analyze the QR-code 4. the system unlocks the turnstile 5. the actor enters the shop
Exit condition	the user/customer get inside the shop
Exceptions	- the QR code is not valid - the hardware is not functioning properly

Table 16: User/customer enters a shop

Name	User/customer exits the shop
Actors	user, customer
Entry condition	have previously entered the store with a QR code
Flow of events	1. the actor arrives at the exit of the shop 2. the actor correctly places the QR-code on the QR-code scanner facility at the exit of the shop 3. the system analyze the QR-code 4. the system unlocks the turnstile 5. the actor exit the shop
Exit condition	the user/customer get out of the shop
Exceptions	- the QR code is not valid - the hardware is not functioning properly

Table 17: User/customer exits the shop

Name	Manager registers new shop
Actors	manager logged in and on his web or mobile personal home page
Entry condition	
Flow of events	1. the manager selects the option to register a new shop, a page pops up with a form to be filled 2. the manager fills in the mandatory fields (position of the shop, schedule of the shop, opening days, name of the shop, maximum number of people allowed in the shop, optionally the items sold in the shop divided in categories or, if preferred, just the categories) 3. the manager submit the data inserted 4. the system verifies the data inserted 5. the system correctly registers the shop 6. the system sends confirmation to the manager
Exit condition	the shop has been correctly registered
Exceptions	- any of the data inserted in the mandatory fields is invalid

Table 18: Manager registers new shop

Name	Manager updates shop's information and settings
Actors	manager
Entry condition	manager logged in and on his web or mobile personal home page
Flow of events	1. the manager selects the update shop info option 2. the system redirects the manager to a dedicated page showing all the editable informations and settings 3. the manager selects the desired information or setting to updadte 4. the manager modifies the selected information or settings 5. the manager confirms the update 6. the system updates the shop informations 7. the system sends confirmation to the manager
Exit condition	the shop informations or settings are correctly modified
Exceptions	- some new information or setting inserted from the manager is invalid

Table 19: Manager updates shop information

Name	Manager checks shop status
Actors	manager
Entry condition	manager logged in and on his web or mobile personal home page
Flow of events	1. the manager selects the check shop status option 2. the system retrieves all the informations about the shop and displays them to the manager (number of people actually inside the shop, number of people enqueued, estimated time of enqueuement, upcoming booked visits, etc) 3. the manager correctly visualizes such informations
Exit condition	the manager correctly visualizes the shop status

Table 20: Manager checks shop status

Name	User cancels previously booked visit
Actors	user
Entry condition	user logged in, on his web or mobile personal home page
Flow of events[3]	1. the user selects the check previously booked visits option 2. the system retrieves the user's previously booked visits 3. the user selects one of his previously booked visits 4. the user selects the cancel prenotation option 5. the system cancels the booked visit and notifies the user of success
Exit condition	the user successfully canceles the visit
Exceptions	- user has no previously booked visits

Table 21: User cancels previously booked visit

Name	Customer cancels previous enqueuement TODO: come detto per i requirements: decidiamo che termini usare, enqueuement o booked visit...
Actors	customer
Entry condition	customer infront of the a shop totem that is displaying his home page
Flow of events	1. the customer selects the cancel a booked visit option 2. the user insterts the code of his ticket on the totem 3. the system cancels the booked visit and notifies the customer of success
Exit condition	the customer successfully cancels his enqueument
Exceptions	- the customer has no booked visit - the customer uses the ticket from an already elapsed visit

Table 22: Customer cancels previously enqueuement

Name	Manager cancels customer's previously booked visit
Actors	manager
Entry condition	the manager is logged in and on his web or mobile personal home page
Flow of events	1. manager selects the cancel customer's previously booked visit option 2. the system ask the manager to insert the code of the ticket to retrieve the desiderd booked visit 3. the manager inserts the code 4. the system cancel the booked visit and notifies the user of the success of the process
Exit condition	the booked visit is successfully canceled
Exceptions	-

Table 23: Manager cancels customer's previously booked visit

Name	User cancels previous enqueuement
Entry condition	user logged in, on his web or mobile personal home page
Flow of events[4]	1. the user selects the virtual ticket in his home page 2. the system retrieves and displays all the information about the ticket and the queue 3. the user selects the dequeue button 4. the system correctly dequeue the user and notifies the user of success 5. the system notifies all the user that changed position in the queue
Exit condition	the user successfully cancels the visit
Exceptions	- the user is not enqueued

Table 24: User cancels previous enqueuement

2.5.2 Use cases sequence diagrams

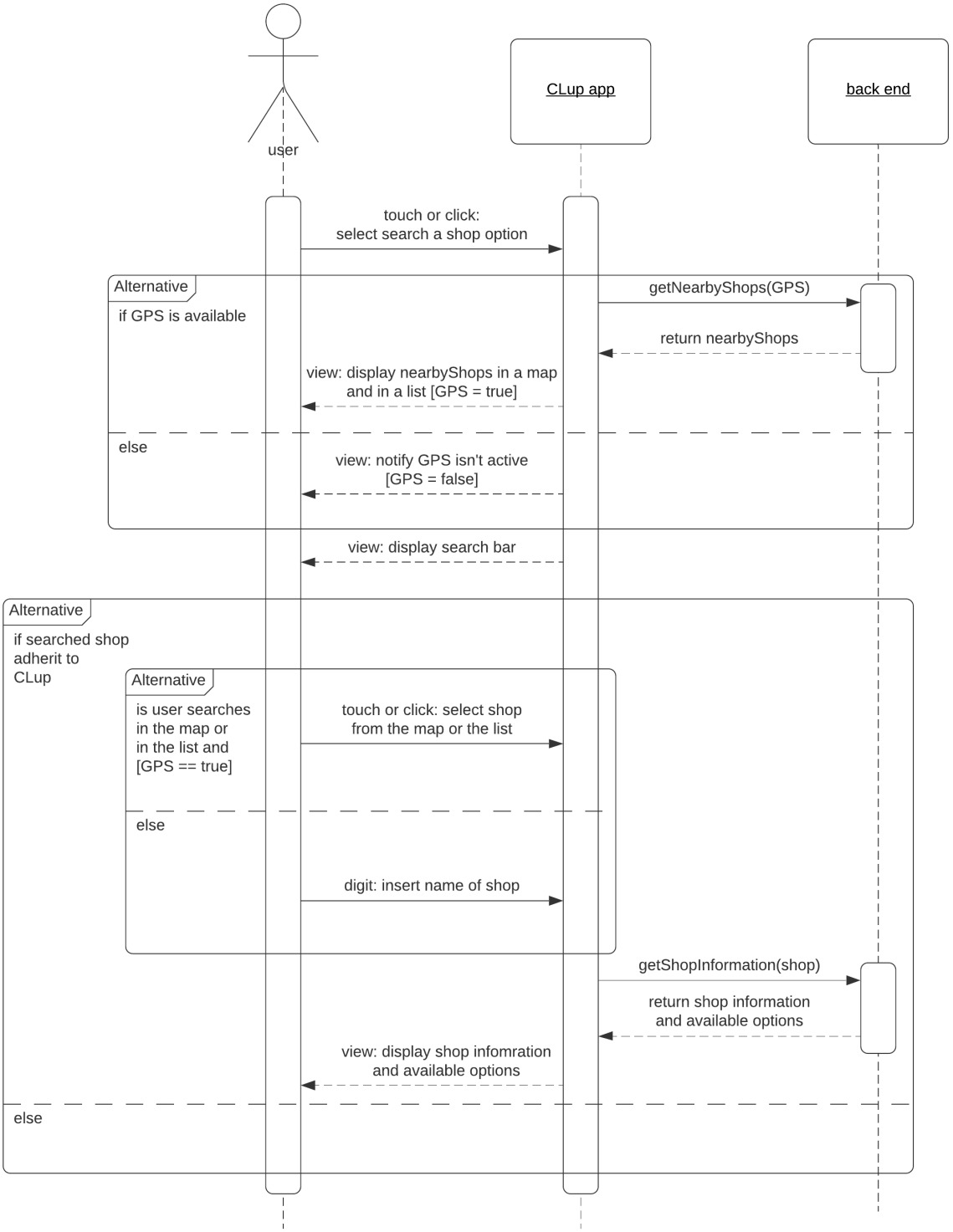


Figure 2: User searches a shop - sequence diagram

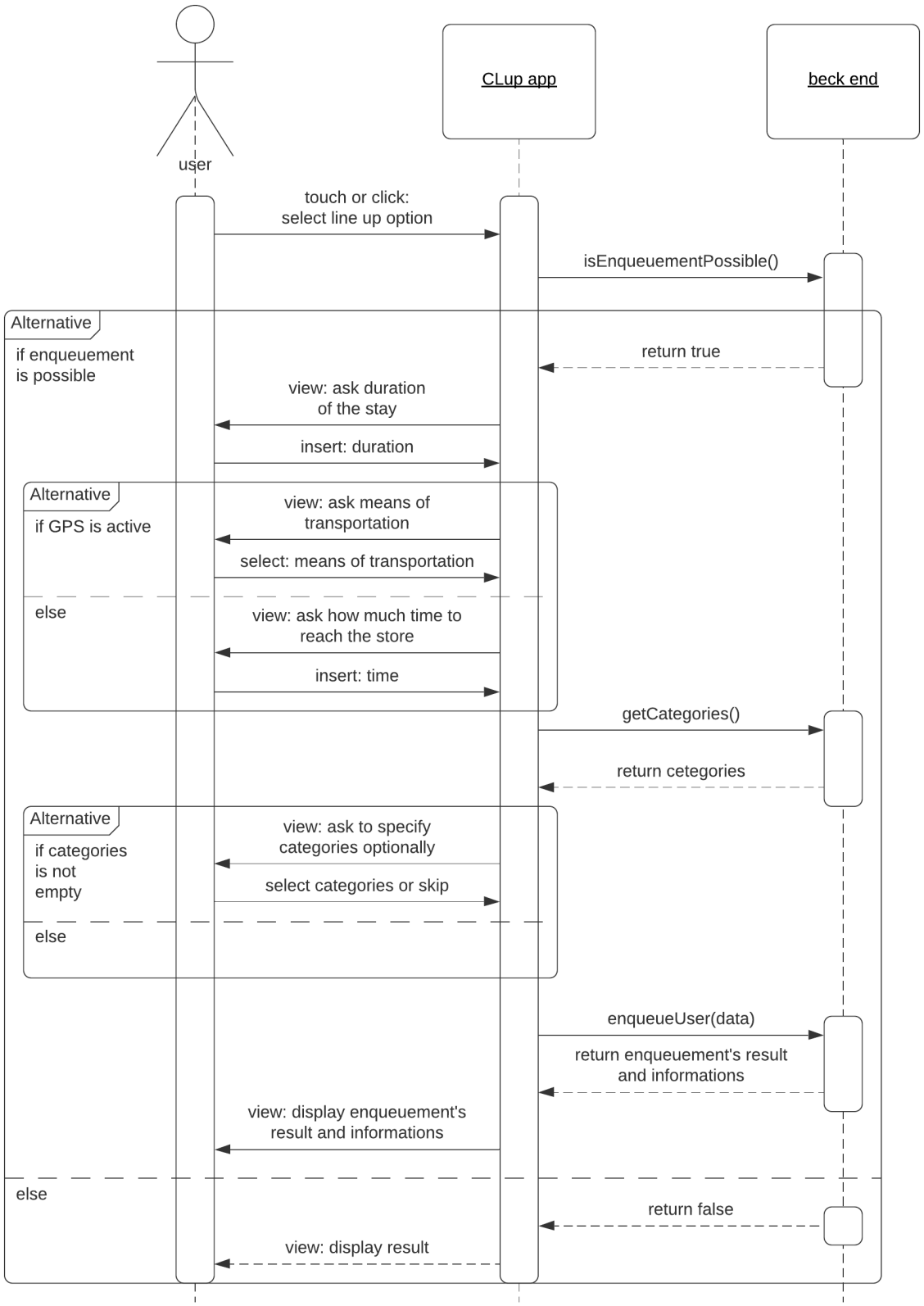


Figure 3: User lines up - sequence diagram

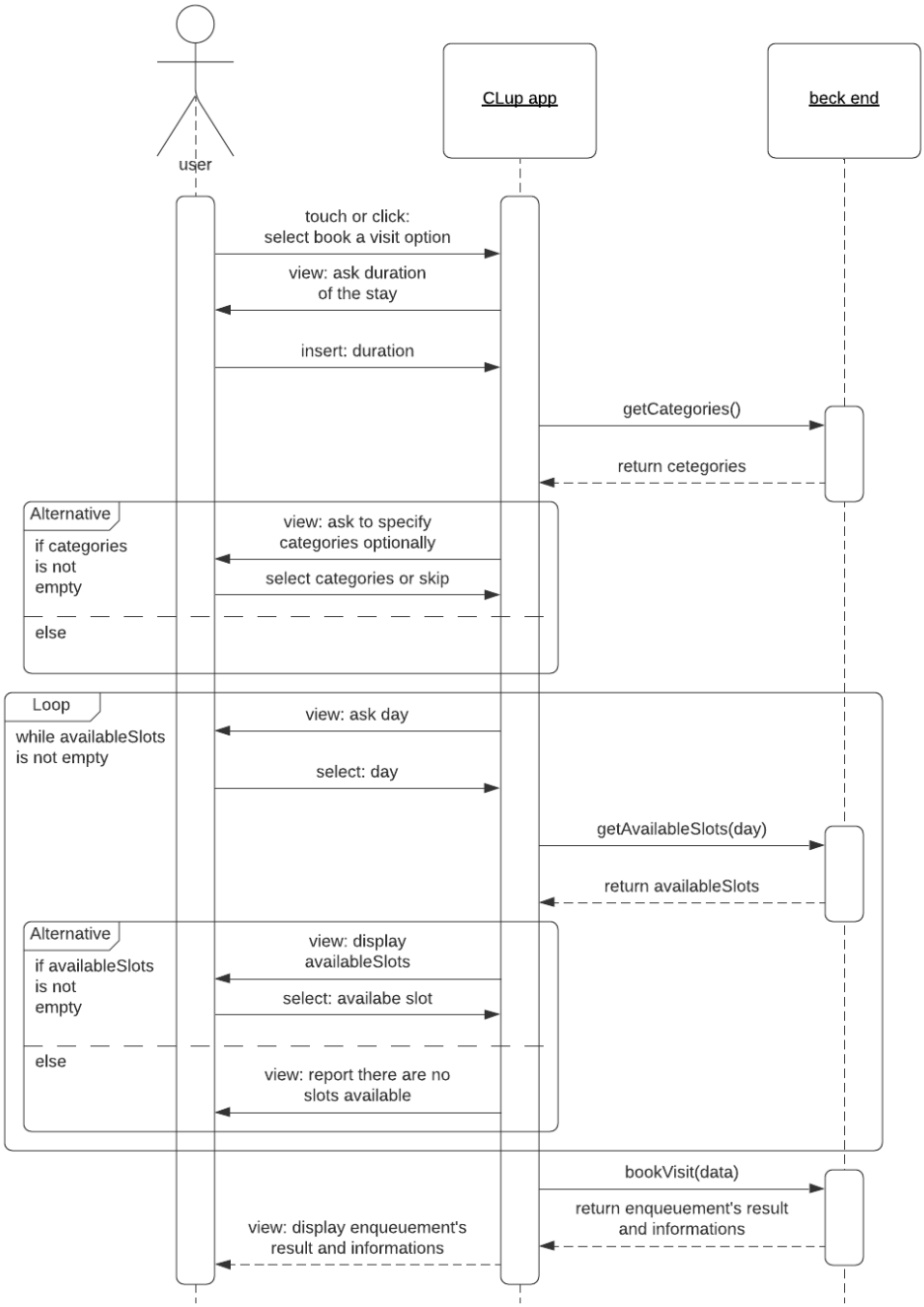


Figure 4: User books a visit - sequence diagram

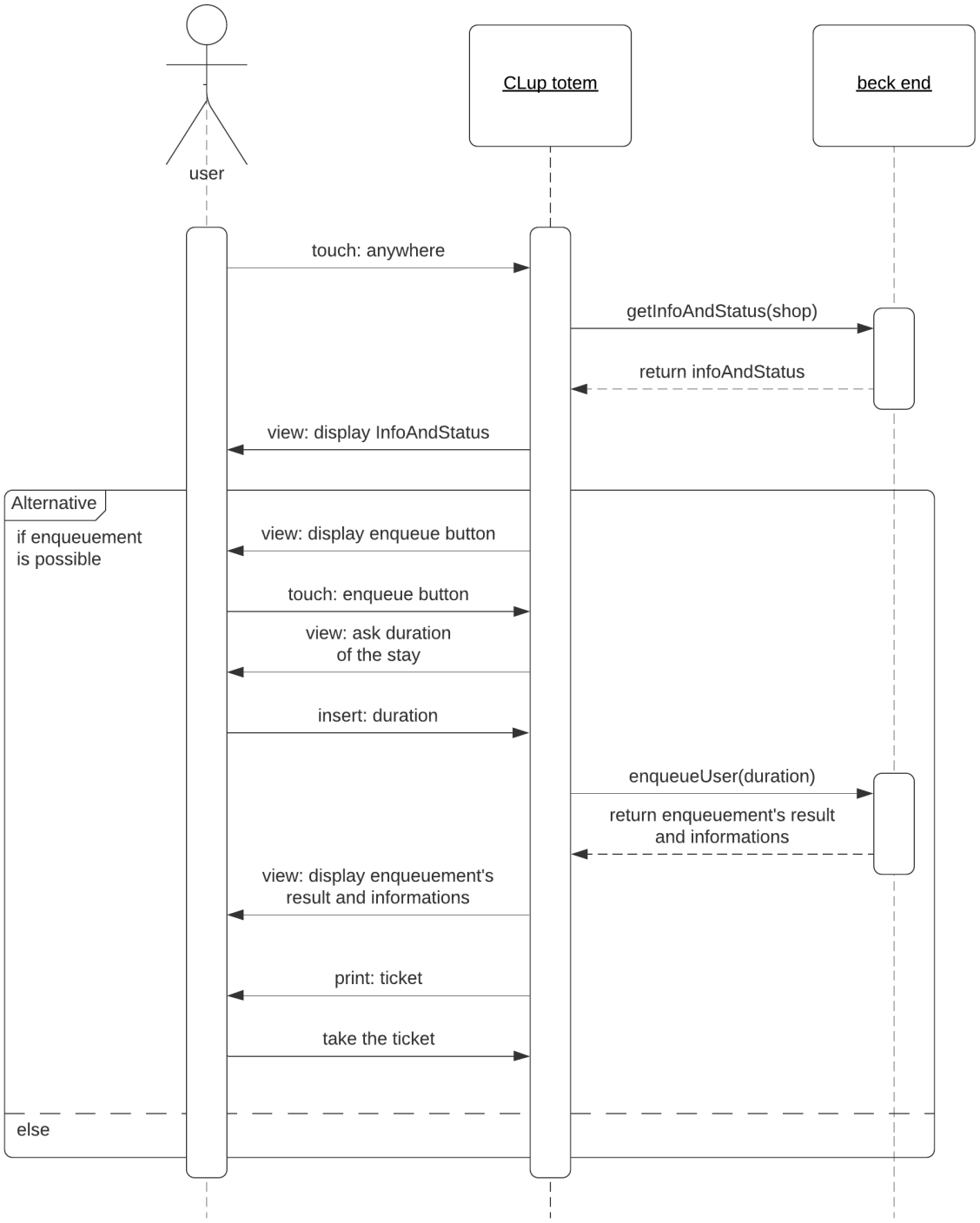


Figure 5: Customer lines up - sequence diagram

2.6 User interfaces

In this section we want to better clarify the *interactions* between a *regular user and CLup system*, through either the web application or the mobile app.

The following diagram wants to connect all the user-related use cases listed previously, to give the reader an high-level understanding of how they are linked. By doing so the diagram shows also how the *user interface* is structured, leaving space for mockups in the Design Document.

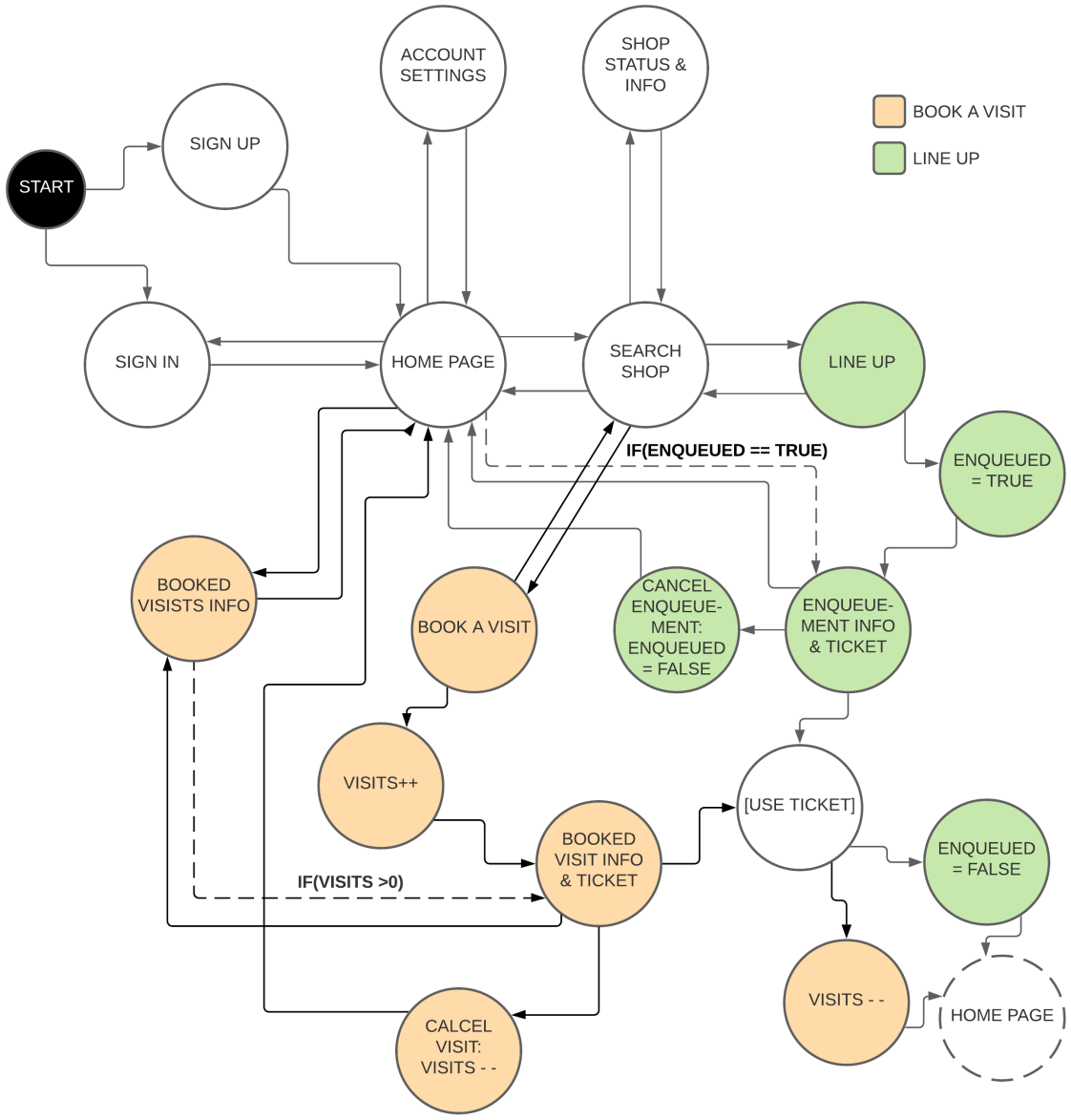


Figure 6: User interface structure - state diagram

2.7 Application model

Here we present an early idea of an high-level abstraction of our application's structure. To be noticed is that it is mainly modeled with focus on the data and their interaction.

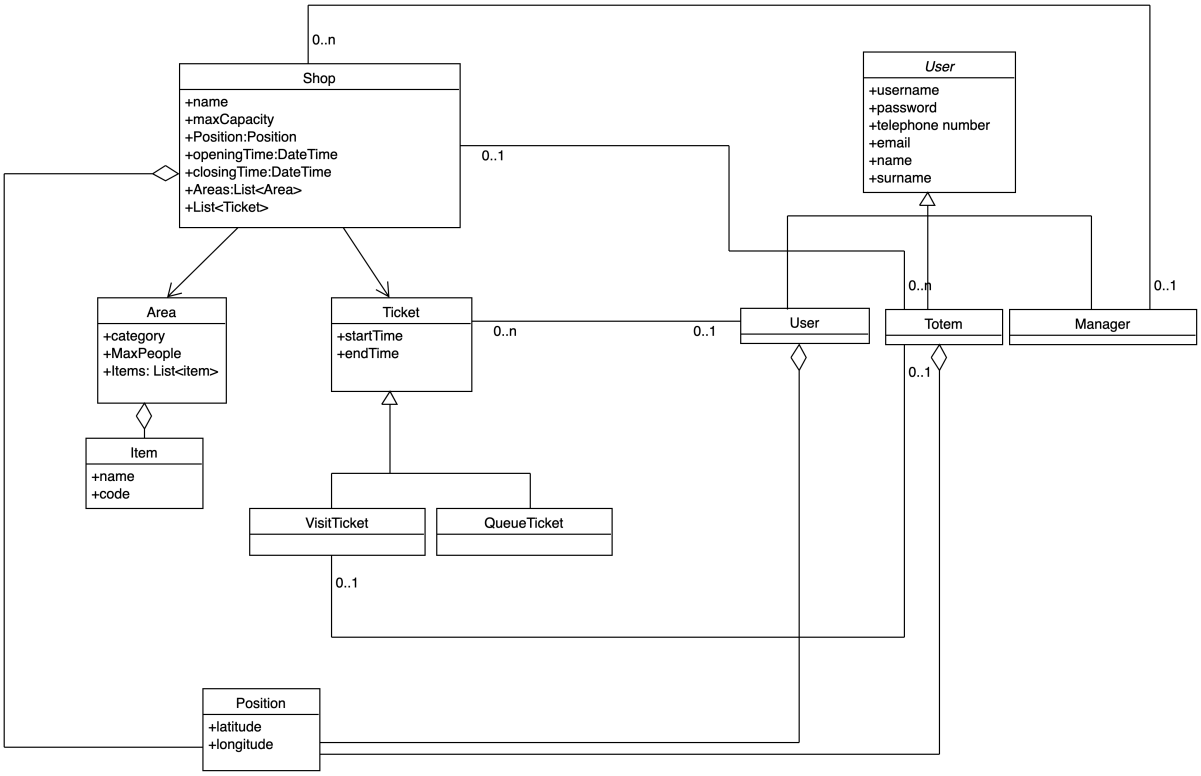


Figure 7: Application model - UML

3 Requirements

3.1 Functional requirements

G1 Allow manager to sign on the system

- **R1** A shop owner must be able to begin the sign on process.
- **R2** The system must require the shop owner to provide all the credentials needed.
- **R3** The system must check that the credentials don't belong to another account already registered to the system.
- **R4** The system must verify the email provided by the shop owner sending a unique code to the address and requesting it from the registration interface.
- **R5** The system must verify the phone number provided by the shop owner by sending a unique code via SMS and requesting it from the registration interface.
- **R6** The system must verify the credentials through PEC API.
- **R7** When the sign on process is completed and the informations provided have been verified by the system, a new account must have been produced.

G2 Allow a manager to sign in the system

- **R8** The manager must be able to begin the sign in process.
- **R9** The system must require the manager to insert email address and password to authenticate.
- **R10** The system must be check if the credentials inserted match an existing account.
- **R11** The system must present to the manager a solution to reset forgotten credentials.
- **R12** The system may allow only managers who provide the corrects pairs of emails and password to sign in.

G3 Allow a manager to register their store/stores on the system

- **R13** The manager must be able to begin the process of registering a store.
- **R14** The system must require the manager to provide the address of the shop.
- **R15** The system must require the manager to provide the schedule of the shop.
- **R16** The system must require the manager to provide the opening days of the shop.
- **R17** The system must require the manager to provide the name of the shop.
- **R18** The system must require the manager to provide the maximum number of people allowed in the shop.
- **R19** The manager can optionally provide to the system a subdivision of the shop in areas with a maximum number of people allowed in each area.
- **R20** The manager can optionally provide to the system the items that are sold in each area.
- **R21** The system must communicate the result of the process to the manager.

G4 Allow a manager to update their shops informations and settings

- **R22** The manager must be able to begin the process of updating the store informations.
- **R23** The system must require the manager to select the information to update.

R24 The system must require the manager to update the information selected.

R25 The system must update the information.

R26 The system must communicate the result of the process to the manager.

G5 Allow a manager to check the general status and the statistic of their shops

R27 The manager must be able to begin the process of checking the status of an owned shop.

R28 The system must be able to retrieve the informations about the number of people currently in the shop.

R29 The system must be able to retrieve the informations about the number of people enqueued.

R30 The system must be able to retrieve the informations about the estimated total duration of the queue.

R31 The system must be able to retrieve the informations about all of the shopping session booked with the attached information of when they will happen and how long they will last.

R32 The system must be able to retrieve the informations about the medium time of permanence in the shop.

R33 The system must be able to retrieve the informations about the medium time of wait in the queue.

R34 The system must be able to retrieve the informations about the medium number of people enqueued at a given hour.

R35 The system must be able to retrieve the informations about the medium number of people inside the shop at a given hour.

R36 The system must present to the manager the information retrieved.

G6 Allow a manager to cancel a previously booked shopping session for a customer

R37 The manager must be able to begin the process of canceling a previously booked shopping session for an user.

R38 The system must be able to retrieve the shopping session booked in his shops.

R39 The system must require the manager to identify the shopping session to cancel.

R40 At the end of the process the system must inform the manager of the result of the process.

G7 Allow a user to sign on the system

R41 The user must be able to begin the sign on process.

R42 The system must require the user to provide all the credentials needed.

R43 The system must check that the credentials don't belong to another account already registered to the system.

R44 The system must verify the email provided by the user sending a unique code to the address and requesting it from the registration interface.

R45 The system must verify the phone number provided by the user by sending a unique code via SMS and requesting it from the registration interface.

R46 When the sign on process is completed and the informations provided have been verified by the system, a new account must have been produced.

G8 Allow a user to sign in the system

R47 The user must be able to begin the sign in process.

R48 The system must require the user to insert email address and password to authenticate.

R49 The system must be check if the credentials inserted match an existing account.

R50 The system must present to the user a solution to reset forgotten credentials.

R51 The system may allow only users who provide the corrects pairs of emails and password to log in.

G9 Allow a user to search a shop

R52 The user must be able to begin the process of searching a shop.

R53 If the user provides his GPS position to the system, the system must be able to ask the user a radius and to retrieve all the shop within that area.

R54 The system must be able to ask the user a keyword to search a shop.

R55 The system must be able to retrieve all the shops matching the inserted keyword.

R56 The system must be able to present to the user the shops found in a map or in a list.

R57 The user must be able to select a shop from the ones provided by the system.

R58 The system must be able to retrieve general informations of the selected shop and to display them to the user.

G10 Allow a user to join the virtual queue

R59 After searching a shop, the user must be able to begin the process of joining the virtual queue of a shop.

R60 The system must be able to understand if an enqueuement is possible or if his demand must be rejected, and the system must display this information to the user.

R61 The system must be able to estimate the queue duration and display this information to the user.

R62 The system must require the user to provide the approximate duration of the shopping session and must check if is valid.

R63 If the user provides his GPS position to the system, the system must be able to ask to the user by what means it is going to go to the shop and to retrieve the estimated time using Google Maps API. Also the system must be able to send to the user a notification 5 minutes before the estimated time to get to the shop.

R64 The system must be able to notificate the user with the awaiting time.

G11 Allow a customer to join a virtual queue from the spot

R65 The system must be able to display through the totem the general informations of the shop.

R66 After approaching the totem, the customer must be able to begin the process of joining the virtual queue of the shop.

R67 The system must be able to estimate the queue duration and display this information to the customer.

R68 The system must be able to insert the permanence time and the system must check if it is possible.

R69 The system must be able to book an anonymous visit for the user at the exact time of end of the estimated duration of the queue.

R70 The system must be able to create a QR-code that refers to the booked visit.

R71 The system must be able to print a ticket with all the relevant information: shop, time, date and QR-code.

G12 Allow a user to book a shopping session at a grocery store

- **R72** After searching a shop, the user must be able to begin the process of booking an online session process.
- **R73** If the user provides his GPS position to the system, the system must be able to ask to the user by what means he is going to go to the shop and to retrieve the estimated time using Google Maps API. Also the system must be able to send to the user a notification 5 minutes before the estimated time to get to the shop.
- **R74** The system must require the user to provide the date and the hour in which the shopping session will begin.
- **R75** The system must require the user to provide the approximate duration of the shopping session.
- **R76** The system must be able to optionally ask the user what categories they are interested in.
- **R77** The system must be able to optionally ask the user what items they are interested in.
- **R78** if a queue exists, the system must ensure that the first available booking session for the user to select will start at least after the awaiting time of the queue.

G13 Allow a user to retrieve informations about his previously booked visits

- **R79** The system must allow the user to begin the process of checking the booked sessions.
- **R80** The system must be able to retrieve the booked session of a user.
- **R81** The system must present to the user the booked visits categorized in active and inactive.
- **R82** The user must be able to select one of the booked visists presented.
- **R83** The system must be able to retrieve all the informations about a booked visit and display it to the user.

G14 Allow a user to retrieve informations about current enqueuements

- **R84** The system must allow the user to begin the process of retrieving informations about current enqueuements, only if the user is currently enqueued.
- **R85** The system must retrieve all the informations about the user's queue and display it to the user.

G15 Allow a user to retrieve informations about shops

- **R86** After searching a shop, the user must be able to begin the process of retrieving more informations about a shop.
- **R87** The system must be able to retrieve all the informations about a shop and display them to the user.

G16 Allow a users and customers to enter and exit stores with QR-codes

- **R88** The system must be able to correctly scan a QR-code.
- **R89** The system must be able to check the validity of the QR-code.
- **R90** The system must be able to interact with the turnstile enabling or disabling it.

G17 Allow a user to exit a previously joined queue

- **R91** The user must be able to begin the process of exiting a previously joined queue, only if the user is currently enqueued.

R92 The system must be able to correctly remove the user from the queue.

R93 The system must be able to correctly confirm the result of the operation to the user.

G18 Allow a customer to cancel a previously booked visit

R94 The customer must be able to begin the process of canceling a previously booked visit.

R95 The system must allow the user to enter a code in the totem to cancel his booked visit.

R96 The system must be able to correctly remove the booked visit.

R97 The system must be able to correctly confirm the result of the operation to the user.

G19 Allow a user to cancel a previously booked visit

R98 The user must be able to begin the process of canceling a previously booked visit, only if the user currently has an active visit booked.

R99 The system must be able to correctly cancel a booked visit.

R100 The system must be able to correctly confirm the success of the operation to the user.

3.2 Non functional requirements

3.2.1 Availability and accessibility

Since the service we are providing is intended to completely replace any current method used, the system must be always available, avoiding as much as possible down-times. Very small deviations from this requirement will be obviously acceptable.

In addition to the availability, we want to ensure accessibility to the largest variety of people possible: every person should be able to interact with our service. To ensure this property, we deliver different products: the ones who can access an internet connection and a smarthphone can easily entry the system through the mobile app or the web app, and the ones who don't have a smarthphone can also use the web app from a computer, for those who doesn't have neither a smartphone nor a computer there is the possibility to use one of the totems placed at the entrance of stores.

Also all the hardware and the software will be built with regards to all of those people with disabilities, ensuring an easy navigation through the interfaces.

3.2.2 Security

The most critical data our application will handle is the users credential. Handling this aspect is one of our main secuirty concern, making sure any third party or malicious actors will never come across.

Also the application will work with metadata about users, such as position, enqueuements, booked visits, items they are willing to buy, and so on. Our purpose is to store only the strictly necessary information in order to have the functions of the application work; anything else will be converted in anonymous data, in order to create general statistics, or discarded.

We must provide security to shops owners too, ensuring that their virtual shops will not be attacked. One of the major thread we need to face is the fact that a large amount of people can make shops

unavilable by creating huge queues and booking all the possible visits on purpose. In the first release of the system, this issue will not be covered, but we'll develope the application so that counter measures can be implemented easily in the future releases.

3.2.3 Scalability

The complexity of our system won't increase linearly with the growth of its reach since every shop will be mainly managed separately. Thanks to this, scalability will just be a matter of increasing computational resources.

Also, since the first release of the system is not going to be a final release, we'll produce the software so that extra features can be embodied easily. An example of future feature may be something that is missing and we have not considered, or discarded on purpose to better focus on main functions, such as the nice feature "enqueue with a friend", which will provide different people with their own smartphones with differents ticket to enter the shop at the same time. A gamified system is also planned for developement in order to encourage a correct behaviour while using the app, such as getting to the shop in time or using the cancelling featutures as less as possible.

3.2.4 Accuracy

CLup system will handle different types of informations, and each one requires different degrees of accuracy:

- *Informations about shops*: the more precise the better it is. Since we are not the ones providing these informations we cannot ensure the precision, but this task is left to the shops managers.
- *Informations about position*: all the informations about user's position are not needed to make our application work, but only to have some extra features. Obviously the most accurate the better, but since it is not essential, there isn't a limit required.

4 Formal analysis using alloy

4.1 Alloy code

For the evaluation of the model and the elicitation of the requirements we used the specification language Alloy which enabled us to express the structural and behavioral constraints of the software system.

```
lenght in the document:
open util / time
open util / time
open util / boolean
open util / ordering[Ticket] as ord

sig Position {
    latitude:one Int,
    longitude:one Int
}

abstract sig User {
    username:one Int,
    userTickets:set Ticket
} {
    username > 0
}

sig AppUser extends User {
    position:Position one -> Time
}

sig Totem extends User {}

sig Shop {
    name:one Int,
    maxCapacityShop:one Int,
    shopArea:set Area,
    shopTickets:set Ticket,
    shopTotems:set Totem,
    shopStatus:ShopStatus lone -> Time
} {
    maxCapacityShop > 0
    name > 0
}

abstract sig ShopStatus {}
one sig Close extends ShopStatus {}
one sig Open extends ShopStatus {}

sig Area {
    name:one Int,
    maxCapacityArea:one Int,
    areaItems:set Item
} {
    maxCapacityArea > 0
    name > 0
}

sig Item {
```



```
    name:one Int,
} {
    name > 0
}

abstract sig TicketStatus {}
one sig InUse extends TicketStatus {}
one sig Expired extends TicketStatus {}
one sig Valid extends TicketStatus {}

abstract sig Ticket {
    id:one Int,
    ticketStatus:TicketStatus lone -> Time,
    ticketArea:set Area
} {
    id > 0
}

sig VisitTicket extends Ticket {}
sig QueueTicket extends Ticket {}

-- -- STATIC MODEL-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

-- -- UNIQUENESS OF KEYS-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

fact ticketIdIsUnique {
    no disjoint t1, t2:Ticket | t1.id = t2.id
}

fact usernameIsUnique {
    no disjoint u1, u2:User | u1.username = u2.username
}

fact shopNameIsUnique {
    no disjoint s1, s2:Shop | s1.name = s2.name
    and no user:User, shop:Shop | user.username = shop.name
    and no ticket:Ticket, shop:Shop | ticket.id = shop.name and ticket.id =
        ↪ user.username
}

fact usernameIsUnique {
    no disjoint u1, u2:User | u1.username = u2.username
}

-- -- UNIQUENESS OF KEYS-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

-- -- begin TICKET constraints-- -- -- -- -- -- -- -- -- -- -- -- -- -- --

--Valid: the QR-code associated is scannable
--InUse: The QR-code associated has been scanned once
--Expired: QR-code associated no longer scannable

fact ticketStatusChart {
    --A ticket is always created as Valid
    all t:Ticket | one t1:Time | t.ticketStatus.t1 = Valid
    all t:Ticket, t2:Time |
        --Once a ticket is expired, it cannot change status
                (t.ticketStatus.t2 = Expired implies all t3:Time | gte[t3, t2
                    ↪ ] implies t.ticketStatus.t3 = Expired)
                and
                --Once a ticket is "InUse" it cannot go back to "Valid"
        (t.ticketStatus.t2 = InUse implies all t4:Time | gte[t4, t2] implies
```



```
            ↪ t.ticketStatus.t4 ≠ Valid)
}

--all of the tickets has timestamp associated to an openState of a shop
fact ticketsToOpenShop {
    all ticket:Ticket | all t1:Time | (ticket.ticketStatus.t1 = Valid or
        ↪ ticket.ticketStatus.t1 = Expired or ticket.ticketStatus.t1 = InUse)
    implies
        (
            some shop:Shop | shop.shopStatus.t1 = Open and ticket in shop.
                ↪ shopTickets
        )
}

--ticket associated to Totems are visit ticket
fact VisitTicketToTotems {
    all totem:Totem | totem.userTickets = VisitTicket
}

--i ticket generati dal totem sono necessariamente associati allo shop a cui
    ↪ e' associato il totem
fact totemTicketTotemShop {
    all ticket:Ticket, shop:Shop, totem:Totem | (ticket in totem.userTickets
        ↪ and totem in shop.shopTotems) implies ticket in shop.shopTickets
}

--no ticket that isnt associated to a user and to a shop
fact ticketToUserantoShop {
    all ticket:Ticket | one user:User, shop:Shop | ticket in user.userTickets
        ↪  and ticket in shop.shopTickets
}


-- -- -- -- -- -- -- -- -- -- --end TICKET constraints-- -- -- -- -- -- --

-- -- -- -- -- -- -- -- -- -- --begin USER constraints-- -- -- -- -- -- --

--no interested in users no associated to tickets
fact noUserToTicket {}

--each totem must be associated to a shop
fact totemToShop {
    all totem:Totem | one shop:Shop | totem in shop.shopTotems
}

--two useres cannot have the same ticket
fact twoUsersDifferetTickets {
    no disjoint u1, u2:User, t:Ticket | t in u1.userTickets and t in u2.
        ↪ userTickets
}

--a user cannot have two "InUse" tickets active at the same time
fact noTwoTicketsAtTheSameTimeInUsePerUser {
    no timestamp:Time | one user:User | all t1, t2:Ticket | t1 in user.
        ↪ userTickets and t2 in user.userTickets and t1.ticketStatus.
        ↪ timestamp = InUse and t2.ticketStatus.timestamp = InUse
}

--personalmente introdurrei qualcosa tipo che un utente puo' avere al massimo
    ↪  tot ticket, ma vedi te
fact noMoreThan3TicketsAtTheTime {
    no user:User | #(user.userTickets) > 4
```



```
}
--a user can only be associated to a queue ticket per time
fact onlyOneQueueTicketPerUser {
    all user:User | no disjoint t1, t2:QueueTicket | t1 in user.userTickets
        ↪ and t2 in user.userTickets
}

-- -- -- -- -- -- -- -- -- -- -- end USER constraints -- -- -- -- -- -- --

-- -- -- -- -- -- -- -- -- -- --begin SHOP constraints-- -- -- -- -- -- --

--the number of "InUse" tickets associated to a shop must be fewer than the
    ↪ MAX CAPACITY of the shop at all times
fact inUseTicketsInAShopLessThanMaxCapacity {
    all s:Shop | let x = s.shopTickets | all timeStamp:Time | x.ticketStatus.
        ↪ timeStamp = InUse implies gte[s.maxCapacityShop, #x]
}

--a totem belongs to one and just one shop
fact oneTotemOneShop {
    all totem:Totem | no disjoint shop1, shop2:Shop | totem in shop1.
        ↪ shopTotems and totem in shop2.shopTotems
}

-- -- -- -- -- -- -- -- -- -- -- end SHOP constraints -- -- -- -- -- -- --

-- -- -- -- -- -- -- -- -- -- --begin AREA & ITEMS constraints -- -- -- --

--each area must be associated to a shop
fact areaToShop {
    all area:Area | one shop:Shop | area in shop.shopArea
}

--each item must be associated to an area
fact itemToArea {
    all item:Item | one area:Area | item in area.areaItems
}

--an item belongs to one and only one Area
fact oneItemOneShop {
    all item:Item | no disjoint area1, area2:Area | item in area1.areaItems
        ↪ and item in area2.areaItems
}

--each item must be associated to an area
fact disjointCAPACITY {
    no disjoint area1, area2:Area | area1.maxCapacityArea = area2.
        ↪ maxCapacityArea
}

--the numbers of area associated to a ticket must be less or equal than all
    ↪ of the areas
fact areaToShop {
    all ticket:Ticket | let area = ticket.ticketArea | #area > 0
}
--an area belongs to one and only one shop
fact oneAreaOneShop {
    all area:Area | no disjoint s1, s2:Shop | area in s1.shopArea and area in
        ↪  s2.shopArea
}

--each area has a max capiency.the sum of the max capiency of all of the
```



```
    ↪ areas of a shop must be the same as the max capiency of the shop
fact sumCapiencyAreasEqualToCapiencyShop {
    all shop:Shop | shop.maxCapacityShop = sum(shop.shopArea.maxCapacityArea)
}

--the number of "InUse" tickets associated to a area must be fewer than the
    ↪ MAX CAPACITY of the area at all times
fact inUseTicketsInAShopLessThanMaxCapacity {
    all ticket:Ticket | let areas = ticket.ticketArea | all area:Area, time:
        ↪ Time | area in areas and ticket.ticketStatus.time = InUse implies #
        ↪ ticket ≤ area.maxCapacityArea
}

--every ticket is associated to an area and to a shop:the area to which that
    ↪ ticket is associated must belong to the shop the ticket is associated
fact ticketToAreaToShop {
    all ticket:Ticket, area:Area, shop:Shop | (area in ticket.ticketArea and
        ↪ ticket in shop.shopTickets) implies(area in shop.shopArea)
}

-- -- -- -- -- -- -- -- -- -- -- -- --end AREA & ITEMS constraints-- -- --

-- -- -- -- -- -- -- -- -- -- -- -- --DYNAMIC MODEL-- -- -- -- -- -- -- --


pred isShopFull[s:Shop, t:Time, x:s.shopTickets] {
    s.shopStatus.t = Open and x.ticketStatus.t = InUse and #x = s.
        ↪ maxCapacityShop
}

pred isShopOpen[s:Shop, t:Time] {
    s.shopStatus.t = Open
}
pred isATicketInUse[t:Ticket, time:Time] {
    t.ticketStatus.time = InUse
}

pred hasAUserAQueueTicket[user:User, t:QueueTicket] {
    t in user.userTickets
}

pred userEnqueue[shop:Shop, user:User, time:Time, ticket, ticket2:QueueTicket
    ↪ ] {
// preconditions
    not isShopFull[shop, time, shop.shopTickets]
    not hasAUserAQueueTicket[user, ticket2]
// postconditions
    ticket in shop.shopTickets
    ticket in user.userTickets
    ticket.ticketStatus.time = Valid
}

pred userBookAVisit[shop:Shop, user:User, time:Time, ticket:VisitTicket] {
// preconditions
    not isShopFull[shop, time, shop.shopTickets]
// postconditions
    ticket in shop.shopTickets
    ticket in user.userTickets
    ticket.ticketStatus.time = Valid
}

pred userCancelAEnqueuement[shop:Shop, user:User, time:Time, ticket:
```



```
    ↪ QueueTicket] {
// preconditions
    isShopOpen[shop, time]
// postconditions
    ticket in shop.shopTickets
    ticket in user.userTickets
    ticket.ticketStatus.time = Expired
}

pred userCancelAVisit[shop:Shop, user:User, time:Time, ticket:VisitTicket] {
// postconditions
    ticket in shop.shopTickets
    ticket in user.userTickets
    ticket.ticketStatus.time = Expired
}

pred show {
    #Shop = 1
    #Totem = 1
    #User ≥ 1
    #Shop.shopTickets ≥ 1
    #User.userTickets ≥ 1
    #QueueTicket ≥ 1
    #VisitTicket > 1
    #InUse = 1
    #Expired = 1
    #Valid = 1
    #Area ≥ 1
    #Item ≥ 1
}
```

4.2 Alloy results

```
run userBookAVisit for 5
run userCancelAEnqueuement for 5
run userCancelAVisit for 5
run userEnqueue for 5
```

Figure 8: Alloy result 1

```
Executing "Run show"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  9993 vars. 603 primary vars. 25581 clauses. 171ms.
  Instance found. Predicate is consistent. 126ms.
```

Figure 9: Alloy result 2

```
Executing "Run userBookAVisit for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  21825 vars. 1260 primary vars. 48589 clauses. 424ms.
  Instance found. Predicate is consistent. 281ms.
```

Figure 10: Alloy result 3


```
Executing "Run userCancelAEnqueuement for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  21562 vars. 1260 primary vars. 47932 clauses. 191ms.
  Instance found. Predicate is consistent. 254ms.
```

Figure 11: Alloy result 4

```
Executing "Run userCancelAVisit for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  21488 vars. 1260 primary vars. 47802 clauses. 227ms.
  Instance found. Predicate is consistent. 184ms.
```

Figure 12: Alloy result 5

```
Executing "Run userEnqueue for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  21868 vars. 1265 primary vars. 48711 clauses. 226ms.
  Instance found. Predicate is consistent. 140ms.
```

Figure 13: Alloy result 6

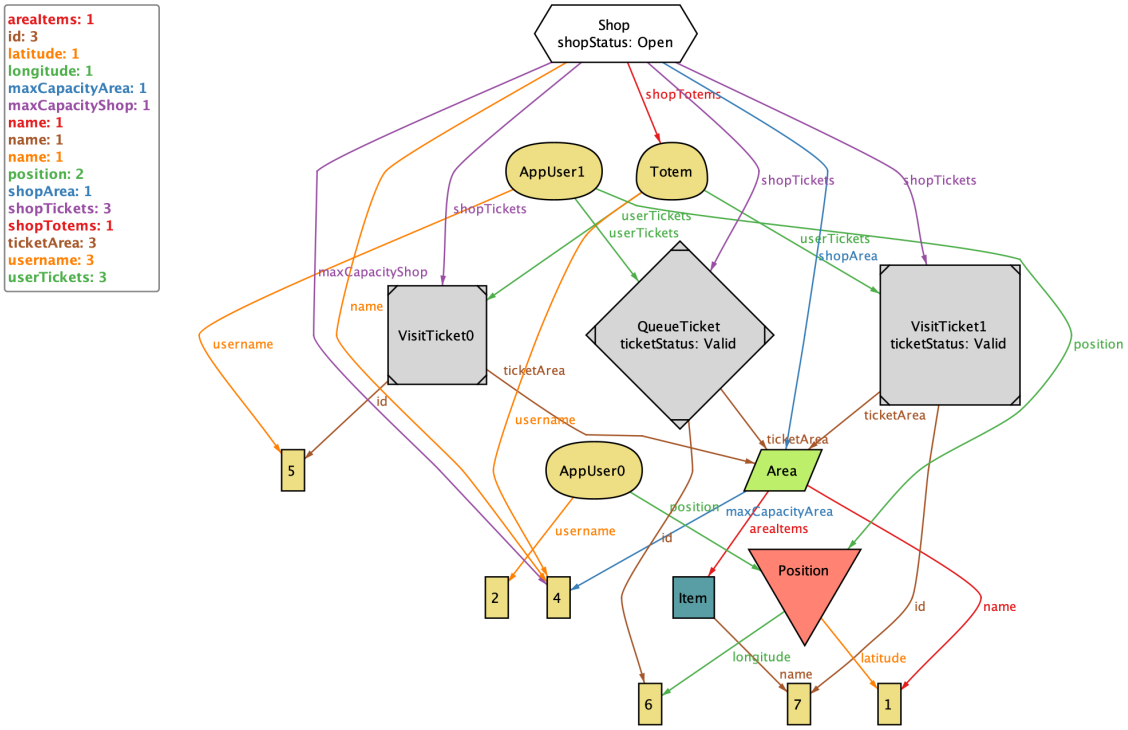


Figure 14: alloy model

5 Effort spent

Amount of time spent in the process of writing this document:

- *Ludovica Lerma*: $\sim$ 48 h.
- *Federico Mainetti Gambera*: $\sim$ 48 h.

6 References

Specification Document The assignment "R&DD Assignment A.Y. 2020-2021.pdf" can be found at this link.

ISO/IEC/IEEE 29148 Document about systems and software engineering, life cycle processes and requirements engineering.

RASD sample from A.Y. 2016-2017.pdf Old RASD document for the Power Enjoy project.

RASD sample from A.Y. 2015-2016.pdf Old RASD document for the myTaxiService project.

Alloy official Alloy documentation from `https://alloy.readthedocs.io/en/latest/index.html`

LaTeX several latex tutorials from `https://www.overleaf.com/learn`

slides from the course of Software engineering 2 from Politecnico of Milan

Photoshop for simple image manipulations

Visual Studio Code for writing LaTeX

diagrams and models `https://lucid.app/`, `https://draw.io/` and the UMLet extension for Visual Studio Code

