Politecnico di Milano

Software Engineering 2 Project
a.y. 2020-21

Customers Line-up
Requirements Analysis and Specifications Document

Version 0.0

Banfi Stefano Alessandro
Bresciani Matteo

Referent professor: Di Nitto Elisabetta

December 10, 2020

Contents

Chapter 1

Introduction

1.1 Purpose

1.1.1 General Purpose

The main scope of this document is to define requirements for the application development, in order to make a correct project planification. To do this, we will analyze:

- System;
- Functional and unfunctional requirements;
- Constraints;
- Relationships between stakeholders;
- Possible scenarios and tests;

These will be shown using different languages, starting from the natural to the structured (such as Alloy and UML). Then, we will define the context in which our application will be developed.
During Sars-Cov-2 emergency, several countries imposed the lockdown in order to hinter the virus diffusion. People had to change their habits, in fact they could go out only for necessary needs, such as going to the market or pharmacy. For instance people must pay attention while they're entering in the market due to the long queue, which could increase the possibility of virus diffusion. This fact obliged people to stay for a long time standing up and waiting their turn losing a lots of time.

1.1.2 Goals

- An User can access to the market;

- Put a limit to the number of Users in the market; ??
- User uses his smartphone / mobilephone;
- User leave his accomodation to reach the market;

1.2 Scope

The aim of the project is to develop an application which, thanks to an intuitive interface, will avoid customers waiting outside the market. In order to do that we offer customers three grocery shopping option:

- **Visit**: it's a planned appointment with given date and range time;
- **Reservation**: it consits in reserving a virtual seat in market's queue;
- **Direct Entrance**: it allows aged customers to enter without any bookings;

The main options shown in this documents are the first two in order to avoid customers to line up outside the market. Those are avaiable only for Smart and Mobile Users.
In particular, in order to avoid to wait in line, Users will be allerted by a **notification** (Smart Users) or a **SMS** (Mobile Users). These inform them about his time schedule required to reach the market in time. Only for Smart User the application will provide the position in queue and the time estimation of his turn.
Instead an alphanueric string will be provided to Mobile Users for going in and out from the market. This string for Smart User is converted in two-dimensional bar code using the standard QRCode. This must be submitted at the entry of the market. Instead, the third option is only avaiable if an user is older than 65 years old, but with limitations. For instance they can go grocery shopping only in certain days and time slots.
In particular the Direct Entrance is avaiable only from Monday to Friday in the range time between 9.00 A.M and 13.00 P.M. The reason for this selection is due the fact that in this ranges there are fewer customers than any other periods because are working hours.
This third option is made especially for aged people who don't have even a mobile phone.
Anyway the RASD document is not focused on this last option.

1.2.1 World

It represents the environment in which the system is placed. In particular it's composed by events which are affected by the system, but not directly connected with it. The main *World Phenomena* are:

- An User can access to the market;
- Limit number of Users in the market;

- User can book his appointment for grocery shopping;
- User use his mobilephone / smartphone;
- Respecting social distance;

In our application we are going to focus on this World Phenomena excepting for the last one. //TODOOO

1.2.2 Machine

It represents the portion of system to be developed. The main *Machine Phenomena* are:

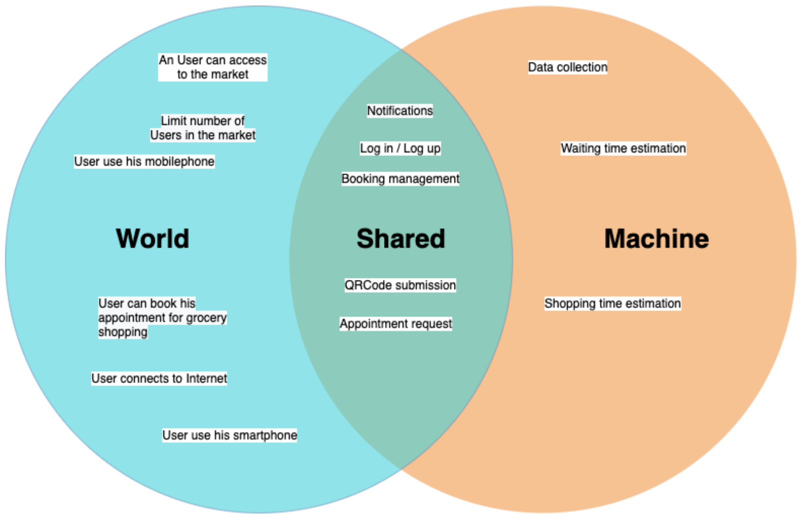
- Internal operations;
- Queue menager;
- Waiting time estimation;
- Data queries;

1.2.3 Shared Phenomena

In this model it needs a common interface to link World and Machine which is composed by the Shared phenomena.
Graphically is represent by an intersection between the World and the Machine. In this way World and Machine phenomena are observed from each other. The main application Shared Phenomena are the following:

- Notifications;
- Sign in / Sign up;
- Booking management;
- QRCode submission;
- Appointment request;

Figure 1.1: World and Machine rapresentation



1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Mobilephone**: Elettronic device without CLup App;
- **Smartphone**: Elettronic devide with Clup App;
- **CLup**: It's the application described in this document. It's an application used to make and manage Booking in order to go grocery shopping; [TODO AGGIUNGERE CLUP OPERATOR]
- **User**: Generic customer who plan to shop in the market. He could be a Smart or Mobile User;
- **Smart User**: User who has got a smartphone which has CLup App. So he's able to manage bookings by himself;
- **Mobile User**: User who hasn't got a smartphone (only mobilephone) and so he's not able to manage Booking by himself. A Mobile User allows a Receptionist to manage his booking by calling a telephone number by interacting with him;
- **Shopping Size**: It's the dimensione of the grocery shopping. It could be *Small*, *Medium*, *Large* depending on the number of items that is going to be purchased;
- **Booking**: It indicates the generic appointment of a User in the market. It could be a Reservation or a Visit;

- **Reservation**: It's a type of Booking. Users simply book a seat at the market's queue. In addition User have to indicate the Shopping Size;
- **Visit**: It's a Booking planned in advance by Users. It is planned by putting the date and the range time in which the User is going grocery shopping. In addition User have to indicate the Shopping Size;
- **Reader**: It reads QRCode at the market's entrance. It allows User to go in;
- **Booking submitted**: It means that the QRCode related to it is already submitted in the Reader;
- **Visit activated**: It's a Visit which already booked but not yet submitted by the User;
- **Reservation activated**: It's a Reservation which already booked but not yet submitted by the User;
- **Waiting Time**: It's the time estimated by the system in which, by prevision, an User waits before he enters in the market;
- **Shopping Time**: It's the time needed by a User to complete his own grocery shopping;
- **Closure Time**: It's the range time in which the market is closed;

1.3.2 Acronyms

- **RASD**: Requirement Analysis and Specification Document;
- **HW**: Hardware;
- **SW**: Hardware;
- **API**: Application Programming Interface;
- **HTTPS**: Hypertext Transfer Protocol Secure;
- **DBMS**: Database Management System;
- **FOL**: First Order Logic;

1.3.3 Abbreviations

- **App**: Application;
- **Gn**: n-th goal;
- **Rn**: n-th requirement;

1.4 Revision history

1.5 Reference Documents

This document is strictly based on:

- The specification of the **RASD and DD assignment** of the Software Engineering II corse, held by professor Matteo Rossi and Elisabetta Di Nitto at the Politecnico di Milano, A.Y 2020/2021;
- **Slides** of Software Engineering 2 course on BEEP;

1.6 Document Structure

Mainly the current document is divided in 4 chapters, which are:

1 **Introduction**: it aims to describe the environment and the demands taken into account for this project. In particular it's focused on the reasons and the goals that are going to be achieved with its development;

2 **Overall Description**: it's a high-level description of the system by focusing on the shared phenomena and the domain model (with its assumption).

3 **Specific Requirements**: it describes in very detail the requirements needed to reach the goals. In addition it contains more details useful for developers (i.e information about HW and SW interfaces);

4 **Formal Analysis**: this section contains a formal description of the main aspect of the World phenomena by using Alloy.

5 **Effort Spent**: it shows the time spent to realize this document, divided for each section;

6 **References**: it contains the references to any documents and to the Softwares used in this document.

Chapter 2

Overall Description

2.1 Product perspective

The project aims to build a system that manage the users' booking without wasting time while they're waiting their own turn outside the market.

An other important aspect is that the application should be for multiple market. Each user at the beginning will choose the market in which are going to go grocery shopping. Indeed, we think that it had better that more markets adopt this system, due to increase safety in social distance.

For a **reservation** the system provides User infomation about queue's dynamic in real time (i.e the number of people ahead). To do this, the application should try, in the best way, to estimate the time that the customers will spend in the market, in order to notify in advance users who are waiting their own turn outside.

In addition Users have to indicate how long the shopping time will be, putting potentially the **size** of the expenditure. This could be *small*, *medium* or *large*. The application analyses the customers' statistics and computes the average shopping time. The calculation considers the time range between the moments in which the User goes in and out. In this way, the other customers in queue will be notified as soon as it's the right time to leave and reach the market in time due to the queue's congestion and his position.

Moreover, it's possible to book a visit choosing the date and the schedule in advance choosing the **visit** option.
The timetable will be splitted into 30 minutes slots, and customers will choose which they want among the free ones.

In addition the market will provide custumer, who don't have the application or even a smartphone, a toll-free number in order to make an appointment.

This would be easy-manageble because a **receptionist** handles the booking and advises the customers with the best shopping option. At the beginning, the user will be sign up for booking: so the receptionist will ask the required data from him and will create a customer profile in the system.

The user can also manage (either delete or postpone), in a later moment, his reservation by calling the same number of the registration. When the turn has been called, the user will be noticed through an SMS. This will contain the identication string that User must show at the entry. Once the user finished, he will scan the code at the exit to open the doors.

In the UML diagram below will be list the main classes in order to understand how the whole system works.

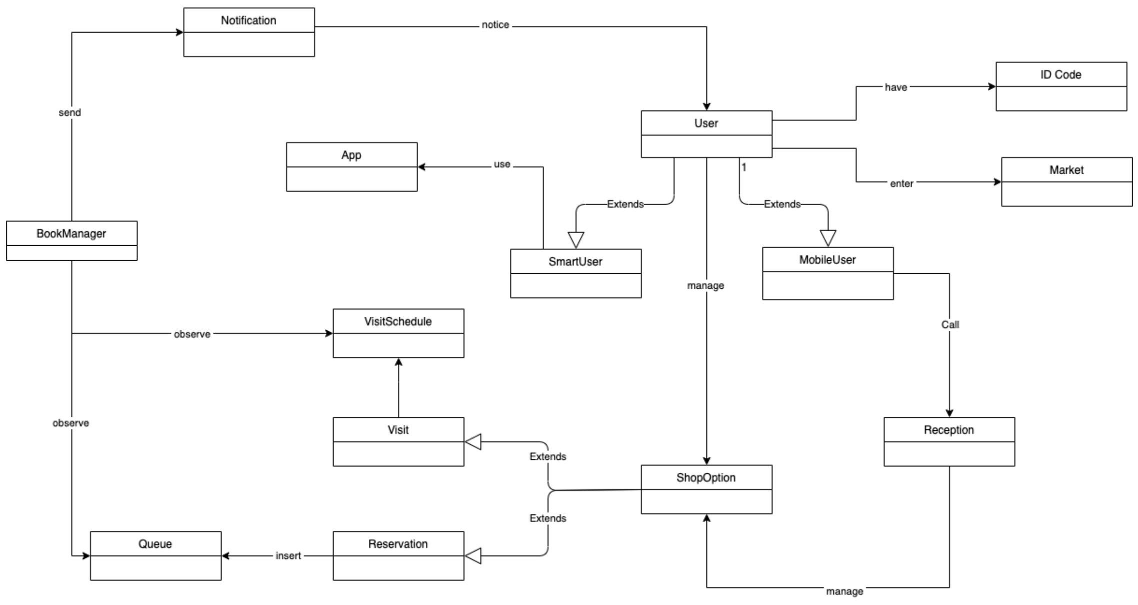


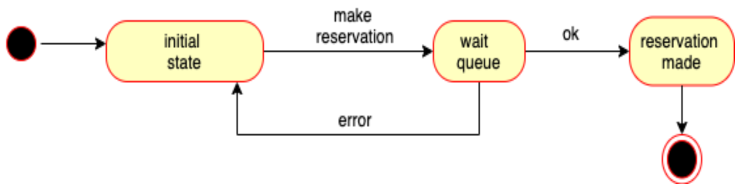
Figure 2.1: Class diagram with UML

As we can see from the class diagram in figure 3.18, the User can book once or a visit or a reservation. In both cases will be provide him a QRcode, which will be submitted to enter in the market. If the the User decided to undo his reservation in the queue could do it by making a cancellation from the application.

In addition the system will notice the User through an SMS or a notice when it's almost his turn (10-15 minutes before).

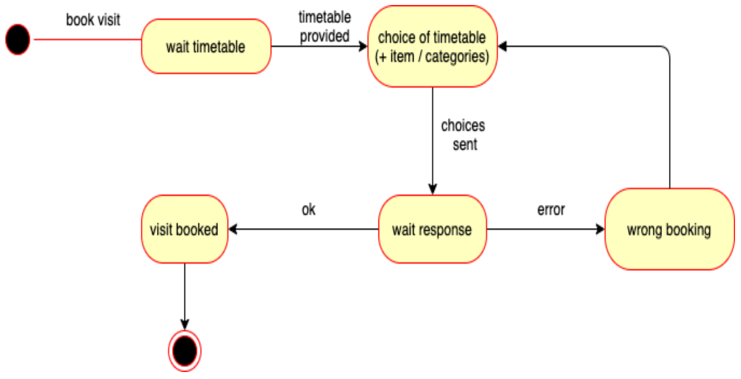
Now we will analyze the interaction between the User and the system, in order to understand possible criticities.

Figure 2.2: State diagram of the reservation in the queue



In the first state diagram (figure 2.2) it can be osserved how a generic User can make a reservation thorugh the system. It's sufficient making this action to be added in queue to enter in the market.
If something goes wrong the User will be redirect to the initial state.
Usually the system reject the request if the market is next to closure (or for logistic problem).

Figure 2.3: State diagram of booking a visit



Instead, the figure 2.3 explains how to book a visit in the market. Once the timetable is provided, the generic user have to select the range time of the visit and the size bag.
If the choice made by the user is wrong (i.e timetable's slot full), the system notify him; the User will try again until his choice is correct.

2.2 Product functions

2.2.1 Queue Manager

One of the most important function is the *Queue Manager* because it must avoid the users waiting too much their turn in the case of a wrong use of notifications. In fact, it have to foresee, through statistics from users' information, the correct time in which the users will enter in the market. This can be done thanks to the notifications sent to the user. It will also have to decide whether to accept or refuse an appointment, depending on the shop closing time and the number of people in queue.

To anticipated the unexpected, the system will use a grace. Let us assume that the maximum number of Users into the market is 100. Well, the system will provide only 90 Users.
The remaining 10 will be used only for emergency or in the time period in which the customer will enter or exit from the market. This "grace" will be a usefull resource for the system because it can manage non-deterministic event, like user's shopping time.
The system will determine the average shopping time according to the information who the user enter during his booking.
In particular accuracy of the computation depends on the quality of User's data. In this way, when the shopping time will be near to the average shopping time, the system will notice the first user in queue to reach the supermarket. Our goal is to monitor the User's influx and to keep it under critical boundary (more or less 95). Therefore this could be possibile by flowing slowly the virtual queue.

2.2.2 Data Collection

The *Data Collection* is essential for the correct behavior of the Queue Manager described previously, because it will have to provide precise dates according to client's information. Therefore, the system will have to ask clients precise questions according to keep useful informations for estimating the shopping time into the supermarket, without violating users' privacy.
In order to achieve this goal, it needs to oblige the user to register himself in the system and to check the item to allow the processing of his personal data, necessary to reserve virtually the seat in the queue.
One of the possible information asked could be the dimension of the expenses. This will be used, with the entry time, to track the number of user inside the market who is finishing. Then, will be possibile to notify in advance users in queue about the closeness of their turn.

2.3 User characteristics

We distinguish the actors into our application based on actions and interactions with the external world:

- *User*: he's a client who has signed in the system and he can book a visit or take his queue number.
- *User in Queue*: he's a User who has taken his own turn in queue and he's waiting for the system notification
- *User visiting*: he's a User who has booked a visit and he's still waiting for entering into the supermarket.
- *User in shop*: he's a client who has taken his own queue ticket or he has booked a visit. After that he is arrived at the supermarket, he has scanned the QR code and he has entered into the shop.

2.4 Assumptions, dependencies and constraints

2.4.1 Assumptions

D1 The system delete the Reservation the Smart User accumulates a delay to reach the market greater than 10 minutes

D2 The system delete the Reservation the Mobile User accumulates a delay to reach the market greater than 15 minutes

D3 The system handles the threshold number of Users allowed in the market

D4 The Smart User have to be connected to Internet through Wi-Fi/Cellular network

D5 The Mobile User have to be connected to Internet through his own mobile operator

D6 A Visit is associated to a Date and a period of time (start/end time);

D7 A Booking is associated to one and only one QRCode

D8 User must have one and only one Visit activated;

D9 User must have one and only one Reservation activated

D10 A Booking belongs to one and only one User

D11 The Waiting Time plus the Shopping Time mustn't exceed the Closure Time

D12 In each Date and Slot Time must contain at most N Visits

Chapter 3

Specific Requirements

3.1 External Interface Requirements

The following mockups provide a model of graphic interface that will show our program.
We will create a user interface that will be user friendly in order to be easy to use also from non-technical users.
The mockups has the aim to illustrate the main aspects of the CLup and the Receptionist's app.

3.1.1 User Interfaces (CLup)

The development of CLup app aims to have a easy-to-use interface for User who has a Smartphone. In particular the application have to be intuitive and reliable in order to give Users the possobility to book an appointment without any obstacles. The following mockups show the main operation allowed by Clup to the Smart User:

Figure 3.1: Login: Users already registered signes in with their credential.

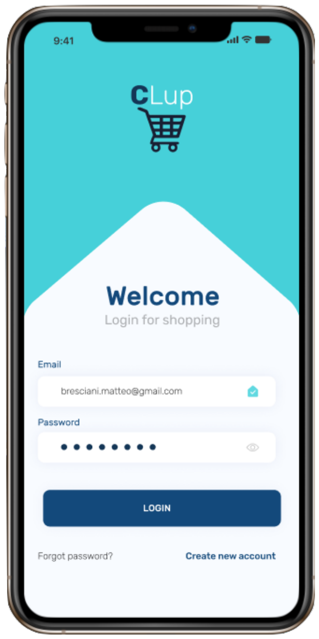
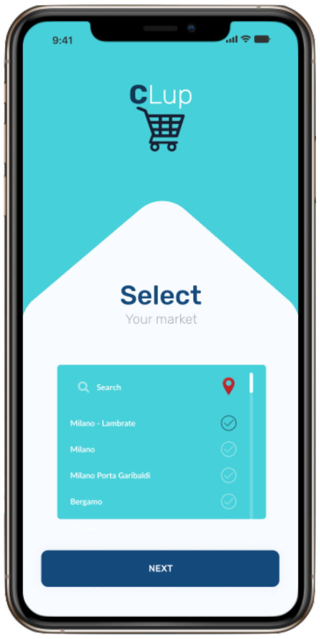
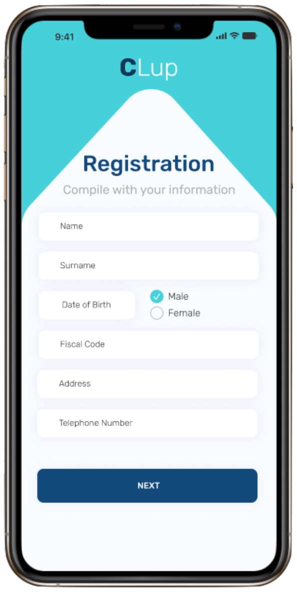
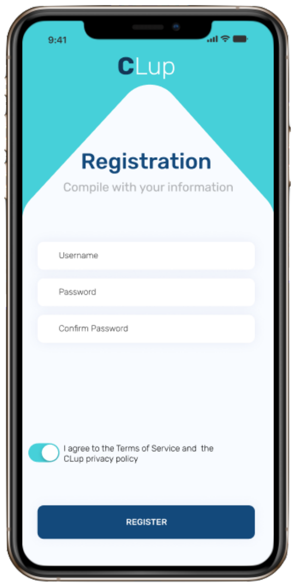


Figure 3.2: Market selection: After the login, the user have to select in which market wants to go shopping, due to his position.





(a) Page 1.


9:41
CLup
Registration
Compile with your information
Username
Password
Confirm Password
I agree to the Terms of Service and the CLup privacy policy
REGISTER

(b) Page 2

Figure 3.3: Registration: Users not registered can register himself putting his own data, e-mail and password. In addition Users have to accepts the Term of Service and the CLup privacy policy to proceed.

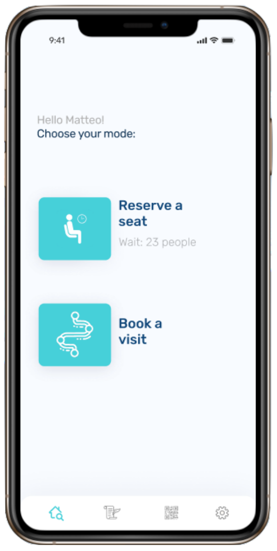
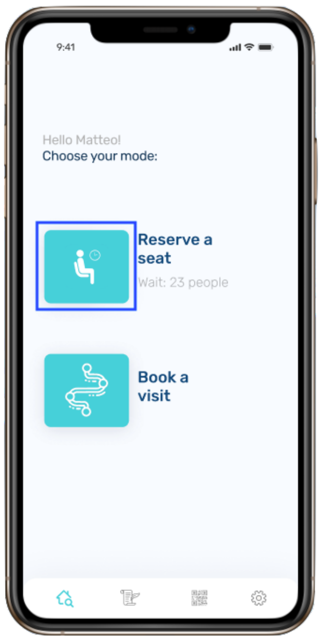
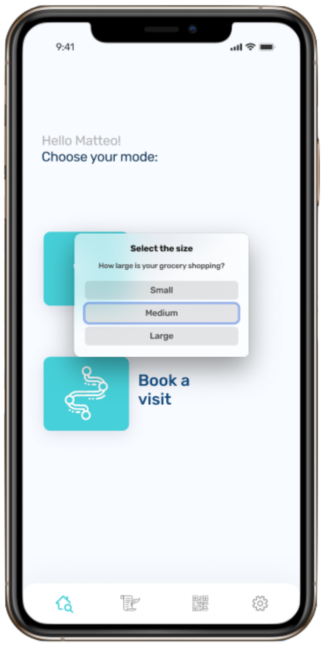


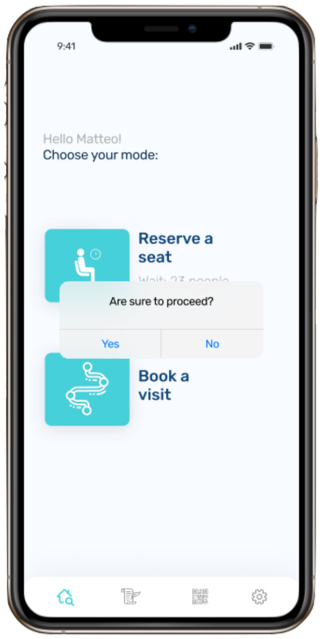
Figure 3.4: Home: Homepage of CLup from which the User can select to book a Visit or a Reservation.



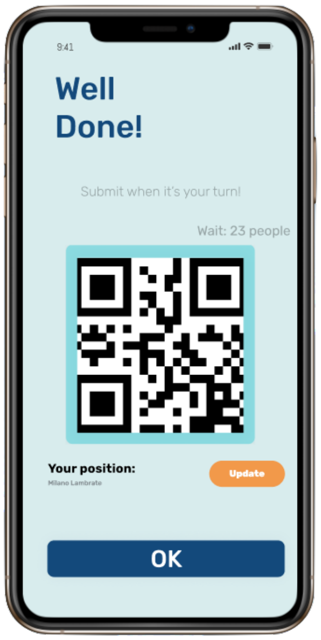
(a) User chooses Reservation



(b) Grocery shopping size selection

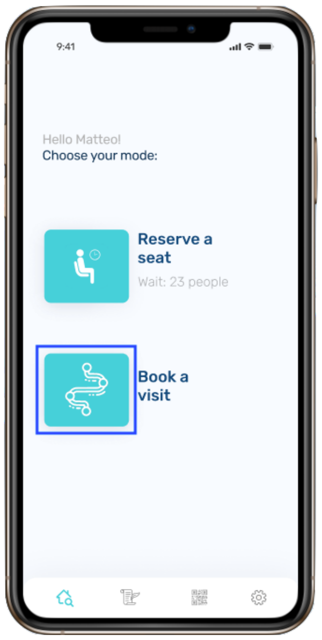


(c) Booking confirm

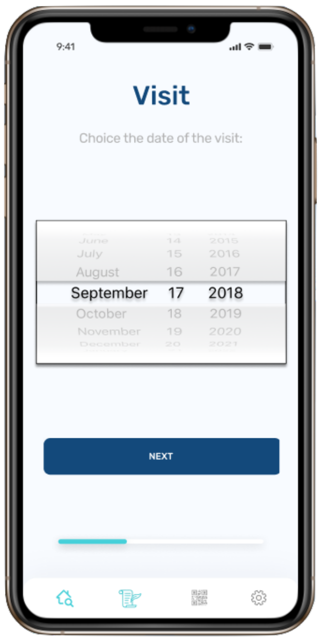


(d) Reservation's QRCode is provided

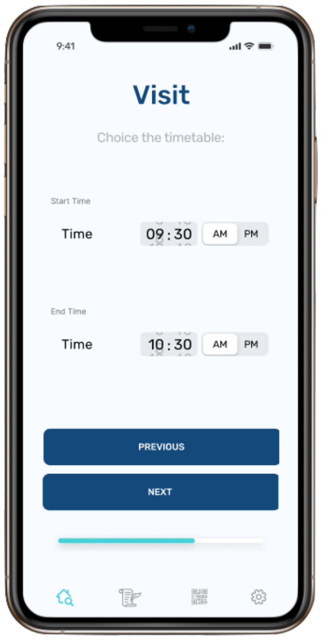
Figure 3.5: Reservation: The Smart User can book a Reservation following the following steps.



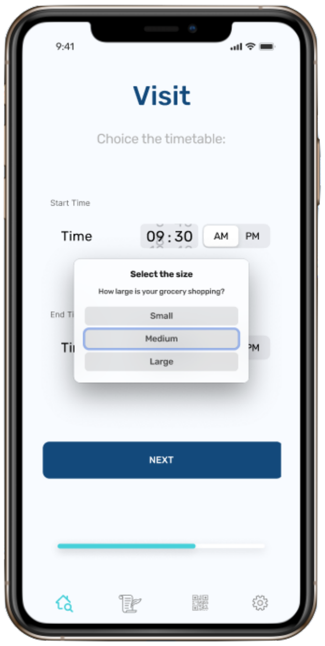
(a) Smart User chooses Visit



(b) Date selection

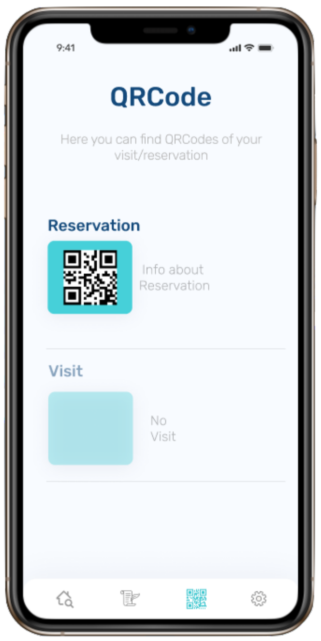


(c) Time period selection

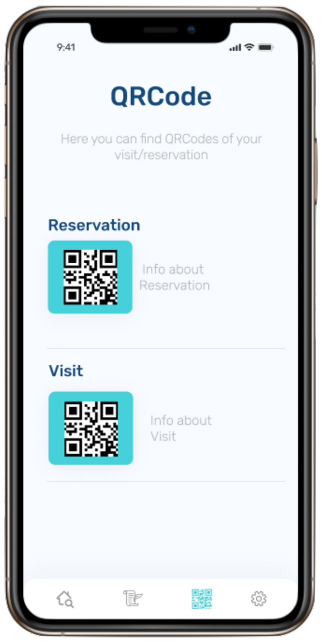


(d) Grocery shopping size selection.

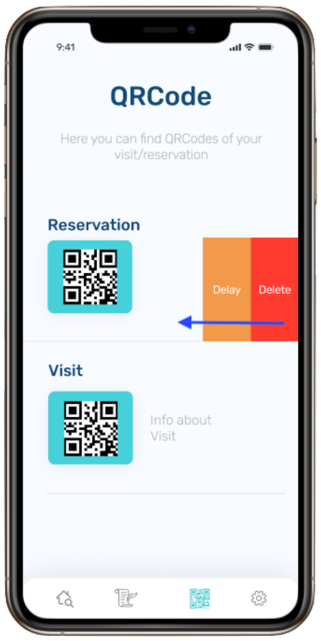
Figure 3.6: Visit: The Smart User can book a Visit following the following steps. At the end Visit's QRCode is provided.



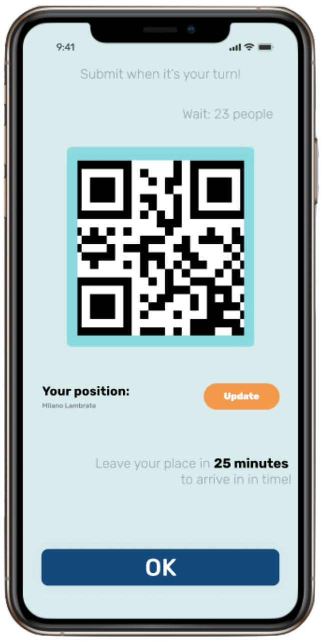
(a) Smart User has booked only a Rservation



(b) Smart User has booked both Visit and Reservation

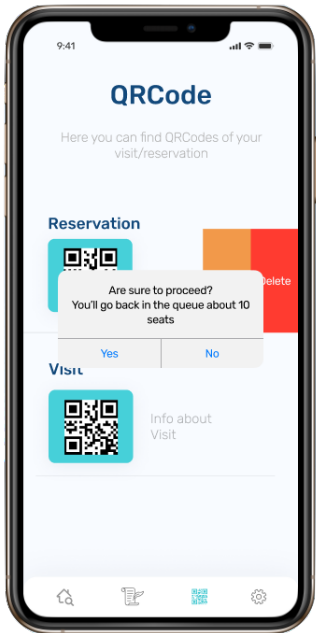


(c) Smart User manage his Reservation


9:41
Submit when it's your turn!
Wait: 23 people
Your position:
Milano Lambrate
Update
Leave your place in 25 minutes to arrive in in time!
OK

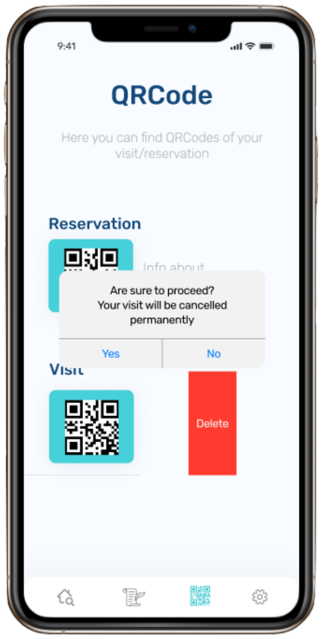
(d) Smart User visualize his own Booking's QRCode

Figure 3.7: QRCode section: in this section the user can manage his booking. A Smart User can cancel, delay (only for Reservation), or visualize his QRCode in order to access to the market.



Are sure to proceed?
You'll go back in the queue about 10 seats
Yes
No

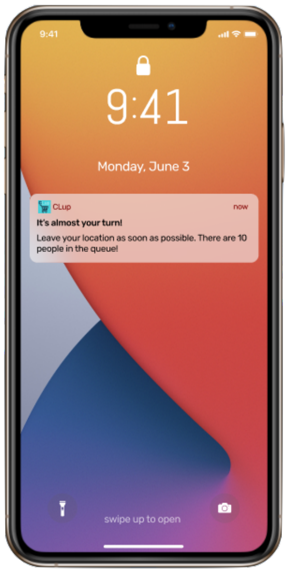
(a) Delay confirmation of a Reservation



Are sure to proceed?
Your visit will be cancelled permanently
Yes
No

(b) Cancellation confirmation of a Visit

Figure 3.8: Cancel and Delay action: A alert will be shown to Smart User to confirm his action.

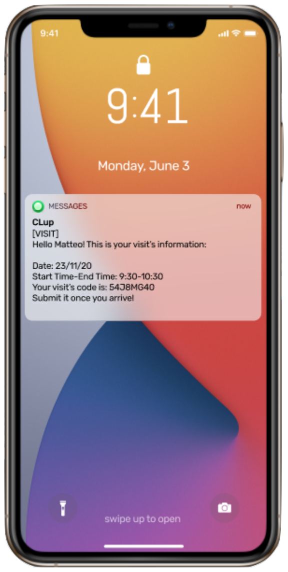


CLup now

It's almost your turn!

Leave your location as soon as possible. There are 10 people in the queue!

(a) A notification is sent to a Smart User to inform him to leave in order to reach the market in time.



MESSAGES now

CLup
[VISIT]
Hello Matteo! This is your visit's information:

Date: 23/11/20
Start Time-End Time: 9:30-10:30
Your visit's code is: 54J8MG40
Submit it once you arrive!

(b) An SMS is sent to a Mobile User with booking information of his Visit. In particular contains the Booking's schedule and the Identification Code to submit.

Figure 3.9: Notifification and SMS: User receive his information about his Booking depending on whether is a Smart or Mobile User.

3.1.2 Receptionist Interfaces (CLup Operator)

CLup Operator, the desktop app, is introduced to give Users who has no Smartphone the possibility to book an appointment at the market. Mobile Users have to call a Receptionist through a customer service number. The Receptionist so aims to interact between the User and the system in order to manage his appointment, acting like a proxy. Even this application must be simple, in order to allow Receptionist to interact with Mobile User in a proper and effective way. The following mockups show the main operation allowed by Receptionist through his application to manage Booking of the Mobile Users.

Figure 3.10: Receptionist signes in with their credential.

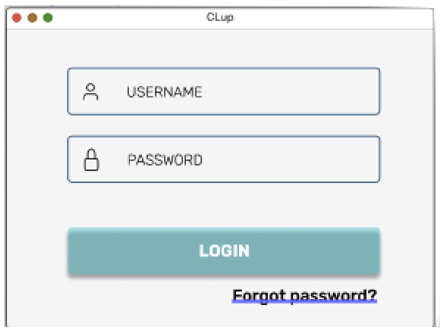


Figure 3.11: Receptionist can register a new Mobile User or select a new one if it's already registered.

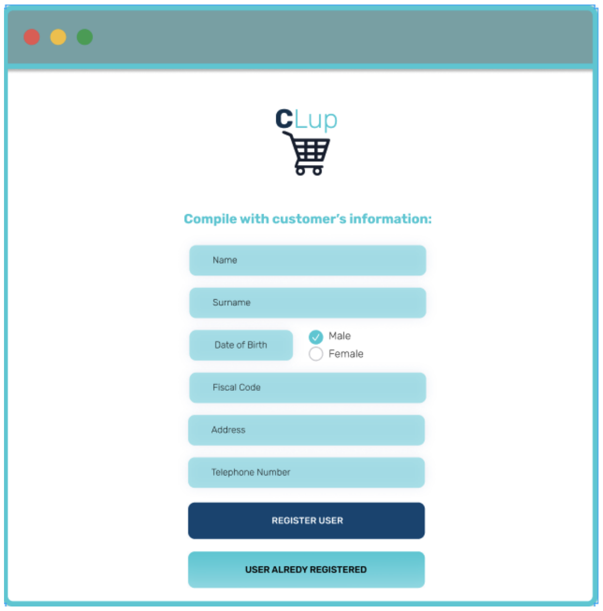
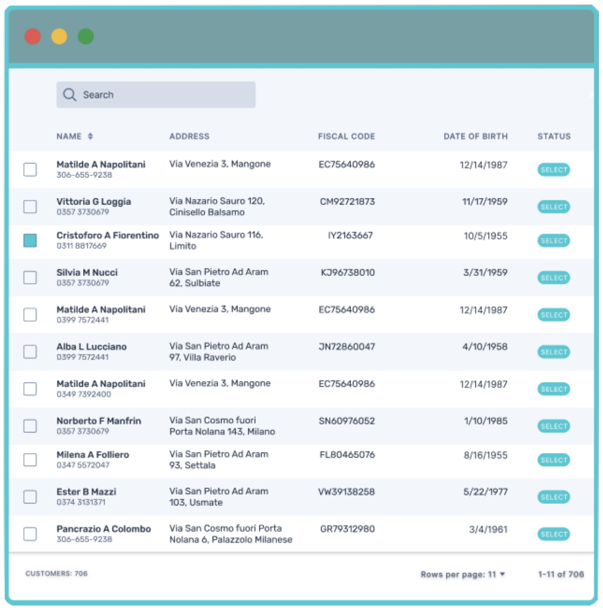


Figure 3.12: Receptionist selects the Mobile User if he's already registered.


Search

	NAME	ADDRESS	FISCAL CODE	DATE OF BIRTH	STATUS
☐	**Matilde A Napolitani** 306-655-9238	Via Venezia 3, Mangone	EC75640986	12/14/1987	SELECT
☐	**Vittoria G Loggia** 0357 3730679	Via Nazario Sauro 120, Cinisello Balsamo	CM92721873	11/17/1959	SELECT
🟦	**Cristoforo A Fiorentino** 0311 8817669	Via Nazario Sauro 116, Limito	IY2163667	10/5/1955	SELECT
☐	**Silvia M Nucci** 0357 3730679	Via San Pietro Ad Aram 62, Sulbiate	KJ96738010	3/31/1959	SELECT
☐	**Matilde A Napolitani** 0399 7572441	Via Venezia 3, Mangone	EC75640986	12/14/1987	SELECT
☐	**Alba L Lucciano** 0399 7572441	Via San Pietro Ad Aram 97, Villa Raverio	JN72860047	4/10/1958	SELECT
☐	**Matilde A Napolitani** 0349 7392400	Via Venezia 3, Mangone	EC75640986	12/14/1987	SELECT
☐	**Norberto F Manfrin** 0357 3730679	Via San Cosmo fuori Porta Nolana 143, Milano	SN60976052	1/10/1985	SELECT
☐	**Milena A Folliero** 0347 5572047	Via San Pietro Ad Aram 93, Settala	FL80465076	8/16/1955	SELECT
☐	**Ester B Mazzi** 0374 3131371	Via San Pietro Ad Aram 103, Usmate	VW39138258	5/22/1977	SELECT
☐	**Pancrazio A Colombo** 306-655-9238	Via San Cosmo fuori Porta Nolana 6, Palazzolo Milanese	GR79312980	3/4/1961	SELECT

CUSTOMERS: 706 Rows per page: 11 ▾ 1-11 of 706

Figure 3.13: Receptionist can manage Mobile User's Booking.

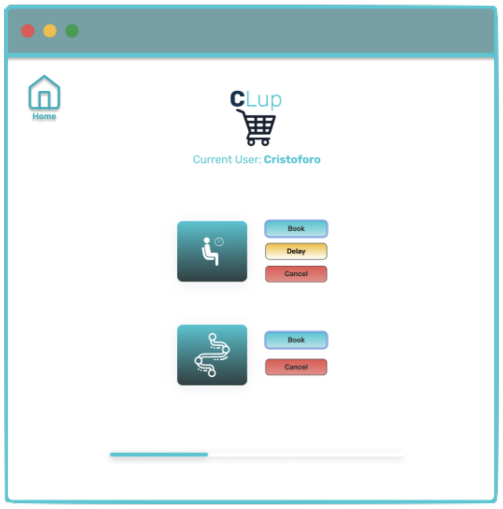


Figure 3.14: Receptionist selects the grocery shopping size told by Mobile User.

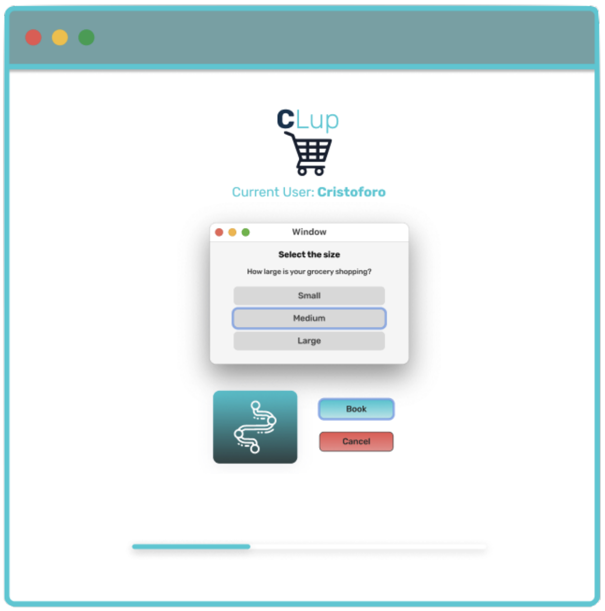


Figure 3.16: Booking made by Recpetionist is finished.

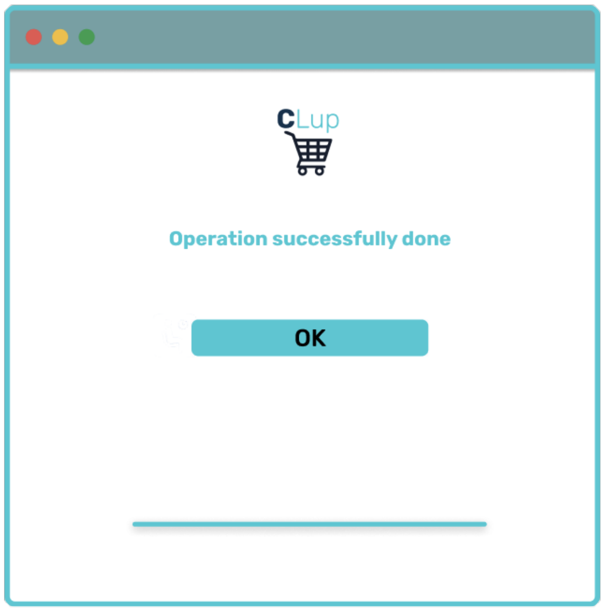
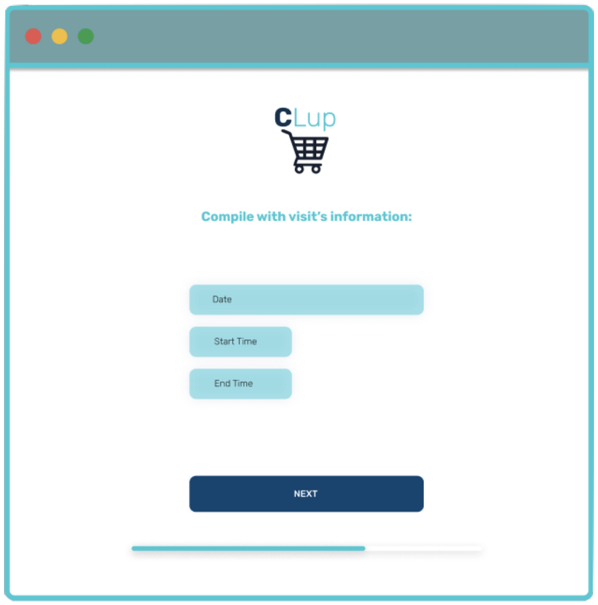


Figure 3.15: Receptionist inserts visit information told by Mobile User.



3.1.3 Hardware Interfaces

The supermarket will have two scanners in order to validate the reservation:

- One at the entry, that will read QR code and will confirm client reservations in case they will be valid.
  The scanner will reject reservation if QR code will result invalid or will have passed to much time from its call.
- One at the exit, that when clients will finish shopping allow them to exit to supermarket opening the doors.
  Another features of exit QR code is to monitor the numbers of client into the shop.

The scanners will be used to obtain useful information about client shopping time

3.1.4 Software Interfaces

CLup notice User when leave in order to reach the market in time also thanks to Google Maps APIs. Infact CLup use them in order to determine how much time a User have to put due to his position. The choice of using Google Maps APIs derives from the easy usability, frequency update and very large documentation. The system, in addition to CLup and CLup Operator application doesn't use any external interfaces.

3.1.5 Communication Interfaces

Every User uses a HTTPS protocol to make request to CLup's servers. In this way informations sent are safe due to encryption provided by the TLS protocol. The same is used by Receptionist to communicate with the server.

3.2 Functional Requirements

G1 User enters once arrived at the market

- **R1** Smart User received a notification due to reach the market;
- **R2** Mobile User received a SMS due to reach the market;
- **R3** The system provide the time estimation to reach the market;
- **R4** The User chooses the start and end time of the Visit;
- **R5** The User can postpone his turn by 10 turns;
- **R8** The system provides the number Users in queue;
- **D1** The system delete the Reservation the Smart User accumulates a delay to reach the market greater than 10 minutes
- **D3** The system handles the threshold number of Users allowed in the market
- **D4** The Smart User have to be connected to Internet through Wi-Fi/Cellular network
- **D5** The Mobile User have to be connected to Internet through his own mobile operator

G2 Put a limit to the number of Users in the market

- **R4** The User chooses the start and end time of the Visit;
- **R6** The system get the entrance time by a User when submits his own QRCode;
- **R7** The system get the exit time by a User when submits his own QRCode;
- **R9** The system provides the estimation time of leaving to reach the market in time;

G3 Smart User can make a Reservation

- **R8** The system provides the number Users in queue;
- **R10** The Smart User must be registered;
- **R11** The Smart User must be already logged in;
- **R12** The Mobile User must be registered;

- **R15** The User must select a date in which the market is opened;
- **R18** The system provides Smart User the Booking's QRCode in CLup application;
- **D4** The Smart User have to be connected to Internet through Wi-Fi/Cellular network
- **D7** A Booking is associated to one and only one QRCode
- **D9** User must have one and only one Reservation activated
- **D10** A Booking belongs to one and only one User

G4 Smart User can make a Visit

- **R10** The Smart User must be registered;
- **R11** The Smart User must be already logged in;
- **R12** The Mobile User must be registered;
- **R14** User have to choose the size of his grocery shopping between Shopping size
- **R15** The User must select a date in which the market is opened;
- **R16** The User must select a start and end time available;
- **R18** The system provides Smart User the Booking's QRCode in CLup application;
- **D4** The Smart User have to be connected to Internet through Wi-Fi/Cellular network
- **D6** A Visit is associated to a Date and a period of time (start/end time);
- **D7** A Booking is associated to one and only one QRCode
- **D8** User must have one and only one Visit activated;
- **D12** In each Date and Slot Time must contain at most N Visits

G5 Mobile User can make a Reservation

- **R8** The system provides the number Users in queue;
- **R10** The Smart User must be registered;
- **R11** The Smart User must be already logged in;
- **R12** The Mobile User must be registered;
- **R13** The Mobile User provide personal data to the Receptionist;
- **R14** User have to choose the size of his grocery shopping between Shopping size
- **R17** The Mobile User calls the Receptionist;

- **R19** The system provides Mobile User the Booking's Identification Code through SMS;
- **R20** Smart User have to be recognized at the entrance;
- **D5** The Mobile User have to be connected to Internet through his own mobile operator

G6 Mobile User can make a Visit

- **R10** The Smart User must be registered;
- **R11** The Smart User must be already logged in;
- **R12** The Mobile User must be registered;
- **R13** The Mobile User provide personal data to the Receptionist;
- **R14** User have to choose the size of his grocery shopping between Shopping size
- **R15** The User must select a date in which the market is opened;
- **R16** The User must select a start and end time available;
- **R17** The Mobile User calls the Receptionist;
- **R19** The system provides Mobile User the Booking's Identification Code through SMS;
- **R20** Smart User have to be recognized at the entrance;

G7 Smart User can cancel a Booking

- **R10** The Smart User must be registered;
- **R11** The Smart User must be already logged in;
- **R12** The Mobile User must be registered;
- **R21** User must have an activated Visit not yet submitted;
- **R22** User must have an activated Reservation not yet submitted
- **D4** The Smart User have to be connected to Internet through Wi-Fi/Cellular network

G8 Mobile User can cancel a Booking

- **R10** The Smart User must be registered;
- **R11** The Smart User must be already logged in;
- **R12** The Mobile User must be registered;
- **R13** The Mobile User provide personal data to the Receptionist;
- **R17** The Mobile User calls the Receptionist;

R21 User must have an activated Visit not yet submitted;

R22 User must have an activated Reservation not yet submitted

D5 The Mobile User have to be connected to Internet through his own mobile operator

Requirements

R1 Smart User received a notification due to reach the market

R2 Mobile User received a SMS due to reach the market;

R3 The system provide the time estimation to reach the market;

R4 The User chooses the start and end time of the Visit;

R5 The User can postpone his turn by 10 turns;

R6 The system get the entrance time by a User when submits his own QRCode;

R7 The system get the exit time by a User when submits his own QRCode;

R8 The system provides the number Users in queue;

R9 The system provides the estimation time of leaving to reach the market in time;

R10 The Smart User must be registered;

R11 The Smart User must be already logged in;

R12 The Mobile User must be registered;

R13 The Mobile User provide personal data to the Receptionist;

R14 User have to choose the size of his grocery shopping between Shopping size

R15 The User must select a date in which the market is opened;

R16 The User must select a start and end time available;

R17 The Mobile User calls the Receptionist;

R18 The system provides Smart User the Booking's QRCode in CLup application;

R19 The system provides Mobile User the Booking's Identification Code through SMS;

R20 Smart User have to be recognized at the entrance;

R21 User must have an activated Visit not yet submitted;

R22 User must have an activated Reservation not yet submitted

3.3 Use Case

3.3.1 User

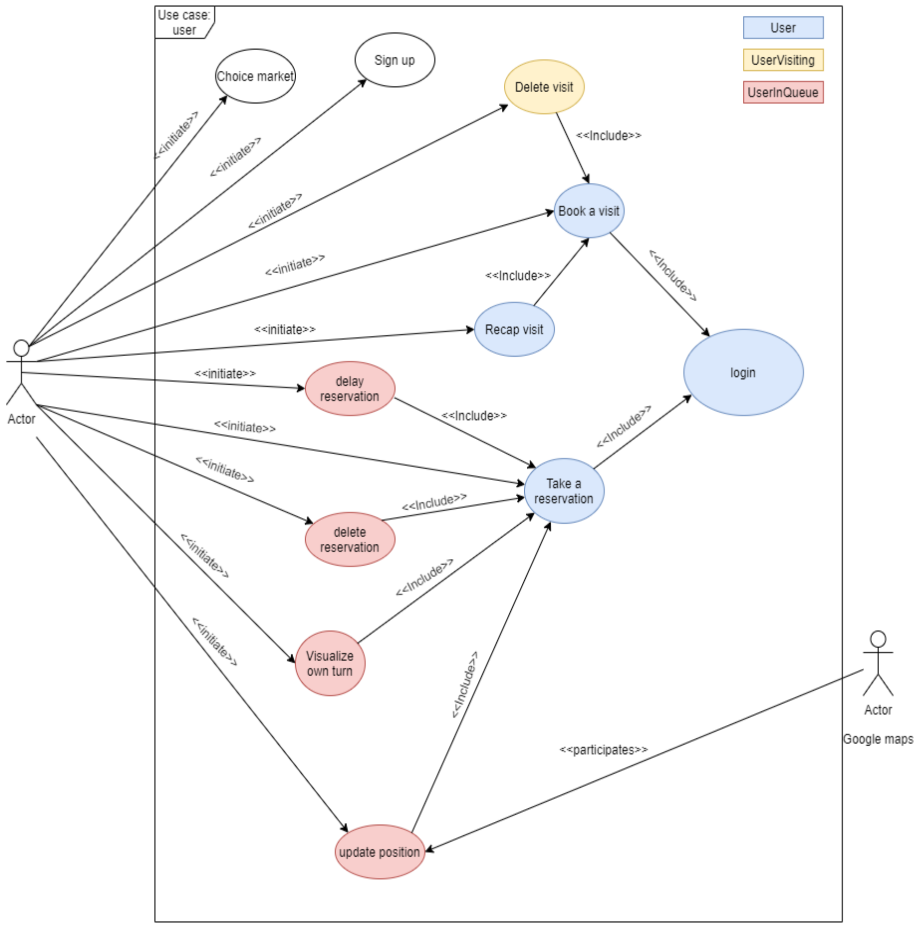
Scenario 1 Giovanna, a career woman who is always in trouble to find free time, needs to go grocery shopping for her family. Indeed, once she finished working, she goes to the market and, due to the lockdown, have to wait in line for hours to have access to it. The result is that, coming back home later, she can't put some time in her children. However, in the past days, she discovered CLup App which allows her to book a visit in the market in advance, by only putting the range time avaiable and the size of the expenditure. In this way, Giovanna will save a lot of time and will stay longer with her children, instead of waiting in line outside the market.

Nevertheless, due to Covid-19 emergency, the market will be always filled.

Scenario 2 Jonhathan, on the advice of his grandchild, bought a new smartphone. In addition started using it and installing usefull applications like CLup. In particular, with it, he'll be able to make a reservation in market's queue. In this way, CLup will notify him when, accordingly to his position, leave to get the market in time for his turn. Finally, once he arrived, can go in there by simply scanning the QRCode sent before. At the end Jonhathan will go shopping without waiting on feet his turn during a cold winter's day.

Name	Login
Actor	User
Entry condition	• The user has register • The user opened the application
Events flow	• The user open the application • Enters username and password • Click "Login button"
Exit condition	User log in
Exceptions	• User enters wrong username • User enters wrong password

Figure 3.17: Use case USER



Name	Sign up
Actor	User
Entry condition	• User enters for the first time on the app • The user hasn't registered
Events flow	• The user selects "Create new account" option • The user enters required fields • The user accepts CLup privacy policy • The user clicks "Register" button
Exit condition	The users is registered
Exceptions	• The users has already been registered • The user did not accept CLup privacy policy • The user enters username that has already been used • The user enters prohibited characters

Name	Book a visit
Actor	User
Entry condition	• The user has logged in
Events flow	• The user click on Home Page • Click on “Book a visit” button • Select the visit date • Select the visit time • Insert shopping size • Click on “Next” button
Exit condition	The user book a visit
Exceptions	

Name	Take reservation
Actor	User
Entry condition	• The user has logged in
Events flow	• The user click on “Home Page” menù • Click on “Reserve a seat” button • Confirm the reservation
Exit condition	• The user has been queued • The QR code has been associated with user
Exceptions	The shop is closed

Name	Delete Visit
Actor	UserVisiting
Entry condition	• The user has logged in • The user took a visit
Events flow	• The user clicks on “QR code” menù • The visits and reservation are listed • The user clicks on “Delete” button near the visit • The user confirms the cancellation
Exit condition	• The system delete user visit • The system make available date and time of user visit
Exceptions	

Name	Delete reservation
Actor	UserInQueue
Entry condition	• The user has logged in • The user took a reservation
Events flow	• The user clicks on “QR code” menù • The visits and reservation are listed • The user clicks on “Delete” button near the reservation • The user confirms the cancellation
Exit condition	The system delete user reservation
Exceptions	

Name	Delay reservation
Actor	UserInQueue
Entry condition	• The user has logged in • The user took a reservation
Events flow	• The user clicks on “QR code” menù • The visits and reservation are listed • The user clicks on “Delay” button near the reservation • The user confirms the delay
Exit condition	• The user turn will be shift ten places after. • The delay button will be disabled.
Exceptions	• The user queue is too small • The user delay function has already been used

Name	Recap visit
Actor	User
Entry condition	The user has logged in
Events flow	The user click on “history” menù
Exit condition	The visits and reservation are listed
Exceptions	

Name	Visualize own turn
Actor	UserInQueue
Entry condition	• The user has logged in • The user took a reservation
Events flow	The user clicks on “history” menù
Exit condition	The system show the user turn near the reservation QR code
Exceptions	

Name	Update position
Actor	UserInQueue
Entry condition	• The user has logged in • The user took a reservation
Events flow	• The user clicks on “QR code” menù • The user clicks on “Update position”
Exit condition	The system calculate the number of customers in queue after which will send the message
Exceptions	• GPS position too far from supermarket • Invalid GPS position • GPS is inactive

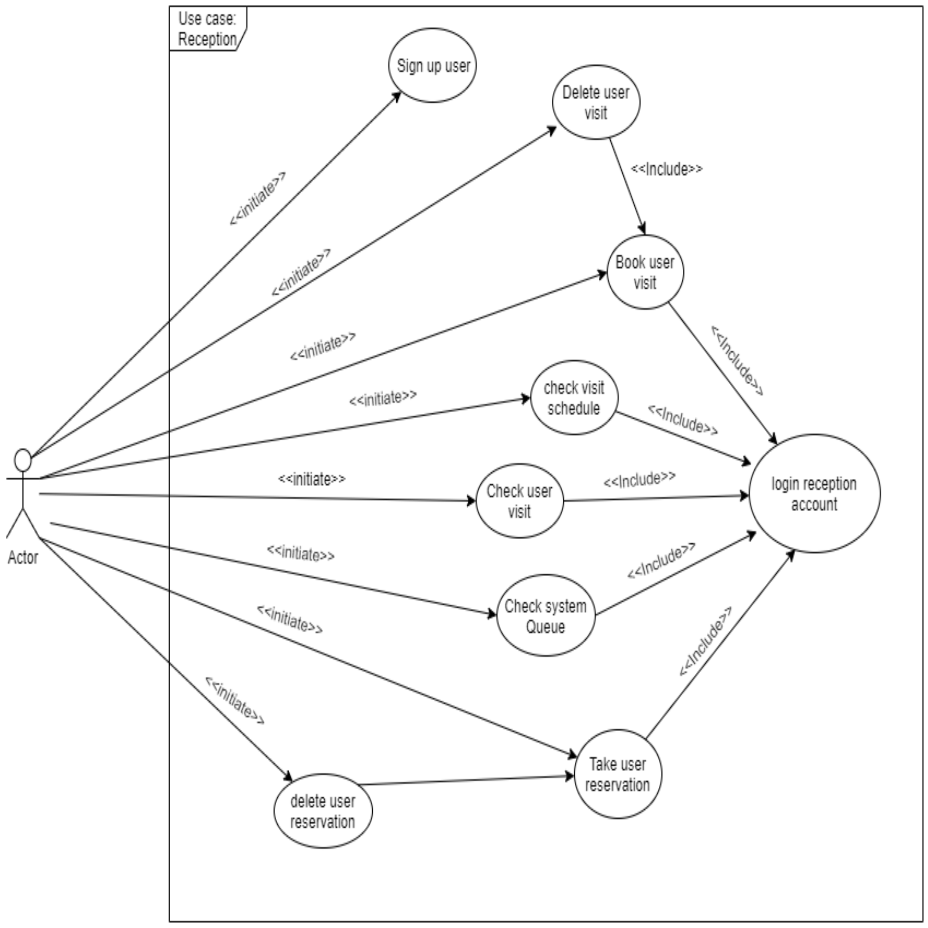
Name	Choice market
Actor	User
Entry condition	User login for the first time on the app
Events flow	• Select the supermarket • Click “Next” to confirm
Exit condition	The system calculate the number of customers in queue after which will send the message
Exceptions	The user is linked to supermarket

3.3.2 Reception

Scenario 3

Gustavo, an elderly person, he discovered recently a new time saver and usefull service at the market.
It consists to book a visit at the market by simply calling the number found in an advertisment.
Due to the fact that Gustavo is sick of waiting too much in the queue decide to call this market number to book the visit.
On the other side Marta, a gentle receptionist who works for the market, answers to Gustavo’s call; she takes care of the registration of his own data, the credentials and the all visit information (i.e data and range time).
Gustavo will be notified about the appointment with an SMS on his mobilephone in time.
In addition the SMS will provide the schedule for the visit and the code which will be submitted at the entrance.

Figure 3.18: Use case RECEPTION



Use cases

Name	Sign up user
Actor	Reception
Entry condition	• The receptionist has logged in • The user has not registered
Events flow	• The receptionist asks required information to do registration • The receptionist fills the fields • The receptionist confirms the registration
Exit condition	The user has registered
Exceptions	• The phone cuts out • The user has already registered

Name	Delete user visit
Actor	Reception
Entry condition	• The receptionist has logged in • The user wants delete a visit
Events flow	• The receptionist asks user information • The receptionist asks visit information • The receptionist checks the visit on the menù • Delete user visit clicking “delete” button
Exit condition	The user visit is deleted
Exceptions	The user has not booked any visits

Name	Book user visit
Actor	Reception
Entry condition	• The receptionist has logged in • The user must be registered
Events flow	• The receptionist asks the credentials • The receptionist enters the credentials in the system to check registration • Click “select” button on user row • Click “book a visit” button • The receptionist check free slot and agree with user date and time • Enter date,start time, end time • The receptionist confirms the visit
Exit condition	The user book a visit
Exceptions	The phone cuts out

Name	Check user visit
Actor	Reception
Entry condition	• The receptionist has logged in • The user must be registered
Events flow	• The receptionist check user registration • The receptionist check user visits
Exit condition	The system list all user visits
Exceptions	• The user does not exist • The user has not booked any visits

Name	Check visit schedule
Actor	Reception
Entry condition	The receptionist has logged in
Events flow	• The receptionist click on calendar • The system lists all free dates • The receptionist click on date
Exit condition	The system lists all date free slot
Exceptions	

Name	Check system queue
Actor	Reception
Entry condition	The receptionist has logged in
Events flow	The receptionist clicks on “Check queue”
Exit condition	The system lists all user in queue information
Exceptions	The queue is empty

Name	Take user reservation
Actor	Reception
Entry condition	• The receptionist has logged in • The user must be registered
Events flow	• The receptionist asks the credentials • The receptionist enters the credentials in the system to check registration • Click “select” button on user row • Click “reserve a sit” button • The receptionist confirms the reservation
Exit condition	The user has been queued
Exceptions	The queue is full

Name	Delete user reservation
Actor	Reception
Entry condition	• The receptionist has logged in • The user must be registered
Events flow	• The receptionist asks user information • The receptionist checks the reservation on the menù • Delete user reservation clicking “delete” button
Exit condition	The user reservation is deleted
Exceptions	The user has not booked any reservation

Name	Login reception account
Actor	Reception
Entry condition	The receptionist open desktop app
Events flow	• The receptionist enter the credentials to log in • Click on "Sign in" button
Exit condition	The receptionist has logged in
Exceptions	• Wrong password • Wrong username

3.3.3 Sequence Diagrams

In this section are reported the main operation (Reservation and Visit) made by Users to make an appointment in the market. Note that are attached request of User with a Smartphone. Requests for Mobile Users are the same, with a difference: the presence of the actor Receptionist, who acts as a proxy in order to manage the requests.

Figure 3.19: Sequnce diagram of a Visit request from Smart User.

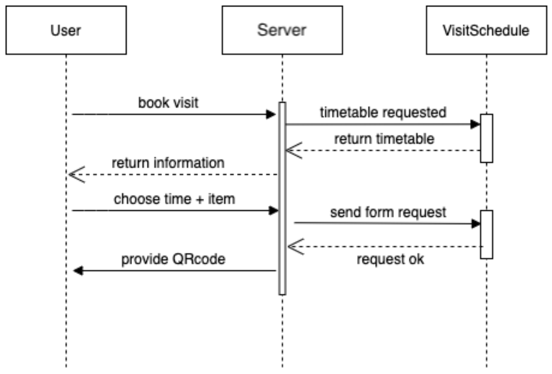
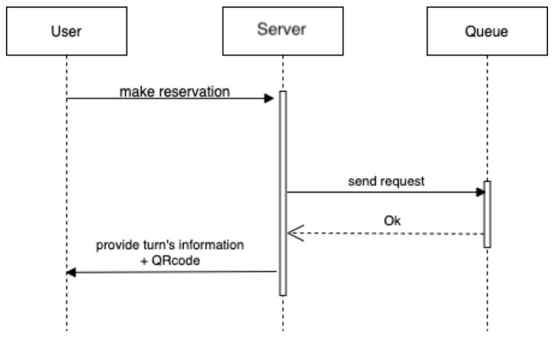


Figure 3.20: Sequnce diagram of a Reservation request from Smart User.



3.4 Performance Requirements

The system has been designed to give possibility to Users to make and to manage their booking every time they need. So it's essential that the CLup's server have to handle a big number of connections.
The most reliable requests must be about the booking. User could book his appointment when he needs without any error or denial. Anyway there's one exception: the booking of a reservation is not avaialable when a market is closed. It's important that the server has enough resources to accept a large number of request which we estimate a maximum of 1000.

3.5 Design Constraints

3.5.1 Standards compliance

Especially, the system will be released in the main digital distribution platform (such as App Store or Google Play). So it must follow their guidelines in order to have a proper and a lawful distribution.
In addition, due to the fact that it retrives and analyses many sensitive data, application must respect tha main privacy guidance. In particular in Europe must follow the General Data Privacy Regulation (GDPR), due to a safe and aware processing of data.

3.5.2 Users in the market

An important constraint that must applied is about the number of presences in the market of the Users. In fact, due to the Covid-19 pandemic, it's important that Users respect the social distance rule that is at least 1 meter.
So it's necessary that the market's provide the maximum number of Users allowed in the market that is, in a general way, it will be indicated with N.

3.5.3 Hardware limitations

- 2G/3G/4G/5G connection: they're essential due to server connection to compile booking request;
- GPS: it's used to allow user to estimate the time spending to reach the market in time. But it's not mandatory for booking;

3.5.4 Any other constraint

There are no other constraint.

3.6 Software System Attributes

3.6.1 Reliability

The application have to provide user the possibility to book both reservation and visit successfully.

3.6.2 Availability

Due to the fact that nowadays grocery shopping is avaiable almost all day, the availability of the system is very high. So the required availability is close to 99%.
However, the reservation function has a lower avaibility because of the inability to book a seat in queue if the market is closed in some hours of the day.

3.6.3 Security

Security is one of the most critical aspect of CLup. So User's sensitive data are stored safely in DBMS accessibile only by strict level of privilege.
In addition, communication towards the application server (like login or booking requests) are implemented using HTTPS protocol, which, with TLS protocol, ensure the encryption of every packets. Another important risk that must by avoid is the Denial of Service, which occurs when the server is unavailable due to incoming traffic flooding from malevoulous Users.
So it's important that the Server mitigates this, by using for instance SYN cookies or any possible arrangements in order to allocate less resources if it's not necessary.

3.6.4 Maintainability

The application implementation must be oriented towards an high scalability in order to gurantee an efficient and cheaper maintenance. This could be done by using design patterns.

3.6.5 Portability

The system must be smoothly portable almost for the main smartphone on the market (i.e Android or iOS). So CLup must be distributed for the main mobile store (i.e App Store and Google Play), coding it with the main program languages such as Android or Swing. Instead, CLup Operator must be compatible with the main Operating system such as Windows and Mac OS X.

Chapter 4

Formal Analysis Using Alloy

.

4.1 Alloy Model

In this section we show the model of the system, by formalizing it with Alloy. This step is strictly important, due to its consistency. In particular it describes the basic structure and the behavior of our system based on FOL. In our alloy model we will show a typical day in the market.
We will describe the main objects involved as signature, with relative constraints, trying to reach a compromise between readability and complexity.

We will exclude from the model some classes, and some tests, because they are useless to describe the model. For example, we will exclude:

- the *Elderly people* that are not involved with Reservation/Visit and therefore they will enter directly in the market in certain time slots;
- the *Receptionist* that acts as intermediary between Mobile User and the system.
- *Market maximum capacity test*: even if this is one of most important goal to reach, we could not describe it in a sufficient way.

In the following model, we discuss the possible choices that the users can do, such as booking a Reservation/Visit, and the constraints imposed by their choices. In particular we describe:

- *Reservation and its insertion* on the Queue with the number assigned to it;
- *Visit booking* between the time slots entered by the users and the shopping time based on its size choosen.

An other important aspect it's the formalization of the QRCode. Users enters in the market with a QRCode if and only if it's valid but not submitted yet. Otherwise, with different boolean values the QRCode signature could have different meaning. All possibile type of QRCode could be:

- *Valid and not submitted*: QRCode owner has booked an appointment (either a Reservation or a Visit) and he hasn't entered yet;
- *Submitted and valid*: the QRCode owner is in the market ;
- *Submitted and not valid*: the QRCode is already used by the owner;

The model is organized temporaly in time slots of 30 minutes each. This way make easier the Visit schedule. The time estimation of each appointment is determined by the bag size. In particular it could be:

- *Small*: it occupies 1 slot in the schedule, that is 30 minutes;
- *Medium*: it occupies 2 slots in the schedule, that is 60 minutes;
- *Large*: it occupies 3 slots in the schedule, that is 90 minutes;

In the following model, we describe the situation in which one reservation is booked and consequently inserted in the queue. (figure 4.6)

In particular we will list 2 queue:

- Q is the queue before the insertion.
- Q2 is the queue after the insertion.

Instead, in the figure **??** describes the situation in which one reservation is deleted from the queue. This means that the reservation will be removed by its user. We prefer not to update the reservation number, because it implies to duplicate all reservation in the queue behind the one that was cancelled. This could involve less readability.

```
open util / boolean

------SIGNATURE------
//abstract signature for a generic user. It will be extended
abstract sig User{
   age: Int,
   hasVisit: set Visit,
   hasReservation: set Reservation
}

sig SmartUser extends User
{
hasPosition: lone Position,
hasApp: Bool
}{
hasApp = True
}

sig MobileUser extends User
{
number:  phoneNum,
hasApp: Bool
}{
hasApp = False
}

one sig Date
{
day : String,
time : Time
}
{
day in ("Monday"+"Tuesday"+"Wednesday"+"Thursday"+"Friday"+"Saturday"+"Sunday
    ↪ ")
}

sig phoneNum{}
sig Position{}


sig Visit{
     hasSize: Size,
     startTime:  Time,
     endTime: Time,
        ID_code:  QRCode
}{
startTime.minutes in (0+30)
endTime.minutes in (0+30)
startTime.seconds = 0
endTime.seconds = 0
}

sig Reservation{
        hasSize: Size,
        ID_code:  QRCode,
        reservationTime : Time,
        numberInQueue: lone Int
}
//Size bag for a grocery shopping
sig Size{
        dimension : String
}{dimension in ("Small"+"Medium"+"Large")}


sig QRCode{
        isSubmitted: Bool,
        isValid: Bool
}
```

```
//List of the Users who are wiaitng to enter in the market
sig Queue{
listOfReser : set Reservation
}

//List of Users that has scheduled a Visit
one sig VisitSchedule{
listOfVisits : set Visit
}

one sig Market
{
userInShop : set User,
state: String
}
{
state in ("Opened"+"Closed")
}

sig Time
{
hours: Int ,
minutes : Int,
seconds: Int
}
{
hours > 0 and hours ≤24
minutes ≥0 and minutes < 60
seconds ≥0 and seconds < 60
}



----PREDICATE----
pred reservePrec [ t1 : Time , t2 : Time]
{
t1.hours < t2.hours or ( t1.hours = t2.hours and t1.minutes < t2.minutes) or
    ↪  ( t1.hours = t2.hours and t1.minutes = t2.minutes and t1.seconds ≤
    ↪ t2.seconds)
}

pred showReservation{
Market.state = "Opened"
#SmartUser = 2
#MobileUser = 1
#Visit =0
#Reservation = 3
#Queue = 1
#Queue.listOfReser = 2
}

pred showVisit{
#Market.userInShop > 0
Market.state = "Opened"
#SmartUser = 1
#MobileUser = 1
#Visit = 2
#VisitSchedule.listOfVisits = 1
#Reservation=0
#Queue = 0
}

pred showMarketOpened{
Market.state = "Opened"
#SmartUser = 2
#MobileUser = 2
#Visit >0
#Reservation> 2
#Queue = 1
#Queue.listOfReser >1
}



pred showMarketClosed{
Market.state = "Closed"

}


pred addInQueue[q, q2 :Queue, r: Reservation]
{
q2.listOfReser = q.listOfReser + r
#q2.listOfReser = add[#q.listOfReser,1]
}

pred showAdd[q, q2:Queue, r: Reservation]
{
#Queue = 2
 addInQueue[q,q2, r]
#Reservation > 1
#q.listOfReser > 1
}

pred delInQueue[q, q2 :Queue, r: Reservation, u: SmartUser, u2 : SmartUser]
{
r in u.hasReservation and r in q.listOfReser
u2.hasReservation = u.hasReservation - r
u2.hasVisit = u.hasVisit
u2.age = u.age
u2.hasPosition = u.hasPosition
q2.listOfReser = q.listOfReser - r
#q2.listOfReser = sub[#q.listOfReser,1]
}

pred showDel[q, q2:Queue, r: Reservation, u: SmartUser, u2 : SmartUser]
{
#Queue = 2
 delInQueue[q,q2, r,u,u2]
#Reservation > 1
#q.listOfReser > 1
}


----FACT----

// The Reservation can be booked only when the shop is opened
fact onlyInOpenTime
{
all r: Reservation | r.reservationTime.hours > 8 and r.reservationTime.hours
     ↪ < 20
all v: Visit | v.startTime.hours > 8 and v.endTime.hours < 20 and reservePrec
     ↪ [v.startTime,v.endTime]
}


// A Visit is accepted by the reader if and only if
fact visitAccepted
{
all v: Visit |   (v.ID_code.isSubmitted = True and  v.ID_code.isValid = True)
     ↪   <=> (reservePrec[v.startTime,Date.time] and  reservePrec[Date.time,v
     ↪ .endTime])
}
```

```
// Condition for a Visit finished
fact visitFinish
{
all v: Visit |   (v.ID_code.isSubmitted = True and  v.ID_code.isValid = False
    ↪ )  <=> reservePrec[v.endTime,Date.time]
}

// Condition for a Visit not started yet
fact visitEarly
{
all v: Visit |   (v.ID_code.isSubmitted = False and  v.ID_code.isValid = True
    ↪ )  <=> reservePrec[Date.time,v.startTime]
}


// The reservation time cannot exceed data time
fact reservationNotInFuture
{
all r : Reservation | reservePrec[r.reservationTime,Date.time]
}

//  Every User must have at most 1 Reservation active
fact sameMomentReserv
{
no r1 : Reservation, r2 : Reservation | r1.reservationTime = r2.
    ↪ reservationTime and r1 not = r2
}

//Function that returns the number of user in queue before r1 (r1 included)
fun numberQueue [r1: Reservation] : set Reservation {
{  r: Reservation | r.ID_code.isSubmitted = False and r.ID_code.isValid =
    ↪ True and  reservePrec[r.reservationTime, r1.reservationTime] }

}

//Assign the queue's cardinality to itself
fact cardinalityQueue {
all r: Queue.listOfReser |  r.numberInQueue = #numberQueue[r]
}


//Assign none if a Reservation is not in queue
fact reservationQueue
{
all r: ( Reservation - Queue.listOfReser ) | r.numberInQueue = none
}


//For each Visit assignes time slot based on the bag size
fact visitSlot
{
all v: Visit | v.hasSize.dimension = "Small" <=>  ((  minus[v.endTime.hours,v.
    ↪ startTime.hours]= 1 and  (v.startTime.minutes - v.endTime.minutes) =
    ↪ 30 ) or (( v.endTime.hours = v.startTime.hours) and minus[v.endTime.
    ↪ minutes, v.startTime.minutes] = 30 ))
all v1: Visit | v1.hasSize.dimension = "Medium" <=>  (( minus[v1.endTime.hours
    ↪ ,v1.startTime.hours] = 1 ) and  v1.startTime.minutes = v1.endTime.
    ↪ minutes )
all v2: Visit | v2.hasSize.dimension = "Large" implies  (( minus[v2.endTime.
    ↪ hours, v2.startTime.hours] = 2 and  minus[v2.startTime.minutes, v2.
    ↪ endTime.minutes] = 30 ) or (minus[ v2.endTime.hours ,v2.startTime.
    ↪ hours] = 1 and minus[v2.endTime.minutes ,v2.startTime.minutes] = 30 )
    ↪ )
}
```

```
// When the market is closed the number of people in the market is equal to 0
fact marketClose
{
Market.state = "Closed" implies ( #User = 0 )
}

// For each User only one QRCode is valid and submitted at each time
fact sserOneActive{
all u : User | lone h : (u.hasVisit.ID_code + u.hasReservation.ID_code ) | h.
     ↪ isSubmitted = True and h.isValid = True
}

// Condition for a User to be in the Market
fact userInShop{
all u :  Market.userInShop  | one q : QRCode | q in ( u.hasVisit.ID_code + u.
     ↪ hasReservation.ID_code) and q.isSubmitted = True and q.isValid = True
}

fact userInShop2
{
no u: (User - Market.userInShop) | one q : QRCode |   q in ( u.hasVisit.
     ↪ ID_code + u.hasReservation.ID_code) and q.isSubmitted = True and q.
     ↪ isValid = True

}

// It doesn't exist a QrCode not valid and not submitted
fact notExistQR_codeMalevolus{
no q : QRCode | q.isSubmitted = False and q.isValid = False
}

//Every position is associated to a SmartUser
fact smartUserCanOwnPosition
{
all p : Position | one su : SmartUser |  p in su.hasPosition
}


fact openMarket
{
Market.state = "Opened" <=> (Date.day in ("Monday"+"Tuesday"+"Wednesday"+"
     ↪ Thursday"+"Friday") and Date.time.hours > 8 and Date.time.hours < 20
     ↪ )
}

fact closeMarket
{
not Market.state ="Opened" implies Market.state ="Closed"
}

fact uniqNumber
{
all pn : phoneNum | one mb : MobileUser |  pn in mb.number
}


fact noUnderAge
{
all u : User | u.age > 18 and u.age < 100
}

// Each Visit belongs to one User
fact visitBelongUser
{
all v : Visit | one u1 : User | v in u1.hasVisit
}

// Each User must have at most one Visit active
```

```
fact only1VisitActive
{
all u : User |  lone v1 : Visit |  v1 in u.hasVisit and v1.ID_code.isValid =
     ↪ True
}

// Condition for a Visit to be scheduled
fact visitInSchedule
{
all v : Visit | (v.ID_code.isValid = True and v.ID_code.isSubmitted= False) ≤
     ↪ > ( v in VisitSchedule.listOfVisits)
}

// Each Reservation belongs to one User
fact reservBelongUser
{
all r : Reservation | one u : User | r in u.hasReservation
}

// Each User must have at most one Reservation active
fact only1ReservationActive
{
all u : User | lone r : Reservation | r in u.hasReservation and r.ID_code.
     ↪ isValid = True
}

// Condition for a Rservation to be insterted in the Queue
fact reservationQueue
{
all r : Reservation |  ( r.ID_code.isSubmitted = False and r.ID_code.isValid
     ↪ = True ) ≤> ( r in Queue.listOfReser )
}

// For each booking the QRCode is unique
fact oneQRforVisit {
all q : QRCode | (q in Visit.ID_code or q in Reservation.ID_code)
no v1 : Visit, v2 : Visit | v1.ID_code = v2.ID_code and v1 not = v2
no r1 : Reservation, r2 : Reservation | r1.ID_code = r2.ID_code and r1 not =
     ↪ r2
no r : Reservation, v: Visit | v.ID_code = r.ID_code
}
---------------------------------------------------------

--<LET>

//Cardinality of QRCode submitted and valid
let many [QRCode] = {
 q: QRCode | q.isValid = True and q.isSubmitted=True
}

--</LET>
---------------------------------------------------------


--<ASSERT>
//Check that user in shop are equal to person that have a QRCode valid and
     ↪ submitted
assert totPerson{
#Market.userInShop = #many[QRCode]
}

//Check that if the market is closed implies that there are no users with
     ↪ either Reservation or Visit submitted and valid
assert noBooking{
Market.state = "Closed" implies #{Visit + Reservation} = 0
}
```

```
--</ASSERT>--
/*
        check totPerson
        check noBooking
*/

run showMarketOpened  for 8 but  8 Int
/*
        run showMarketClosed for  8 Int
        run showReservation for 8 but  8 Int
        run showVisit for 6 Int
        run showAdd for 6 Int
        run showDel for 6 Int
*/
```

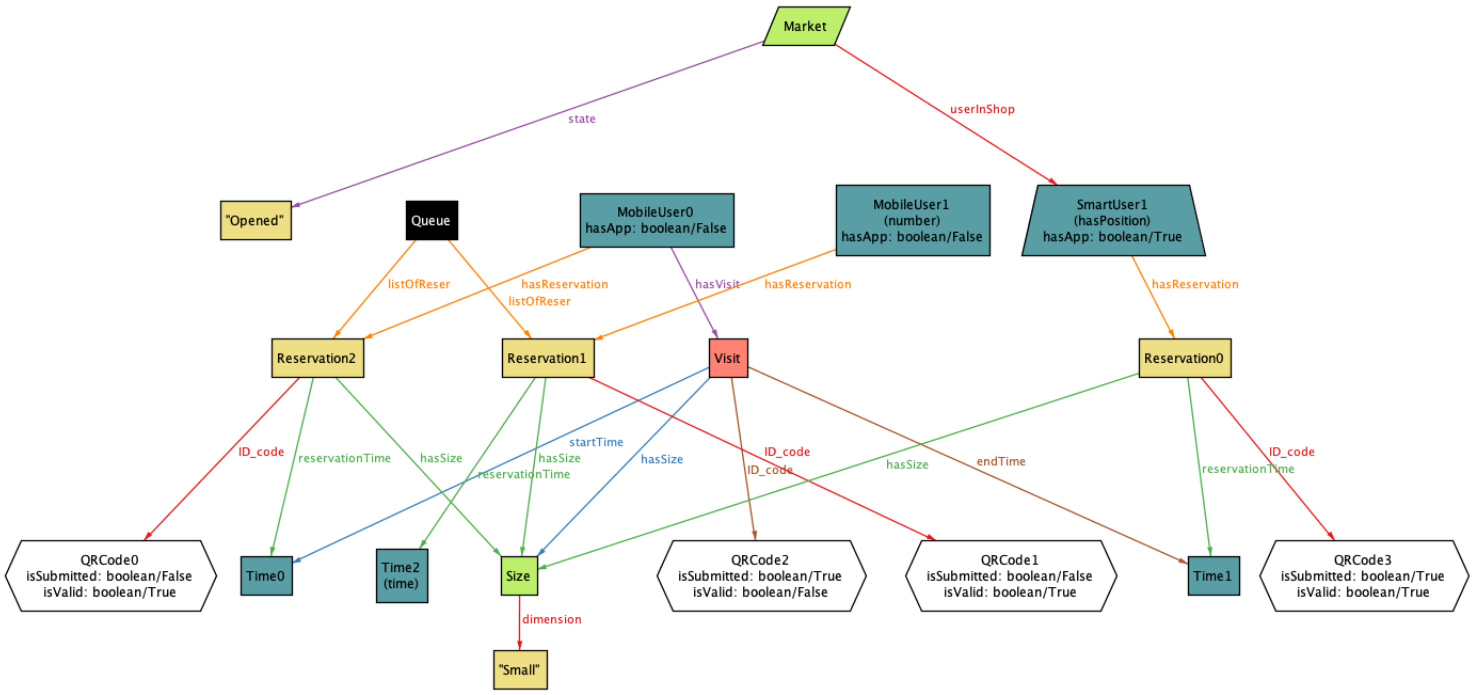


Figure 4.1: Rapresentation of the market opened in a generic day.

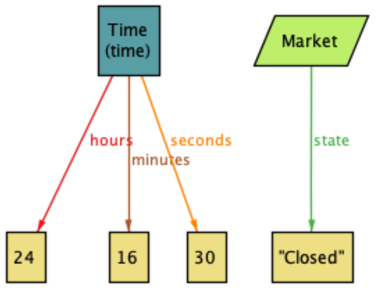


Figure 4.2: The market is closed. In this case, there aren't any users in the market.

Figure 4.3: Rapresentation of the market opened in a generic day. This is the result of the showReservation predicate that focuses on Reservaitons.

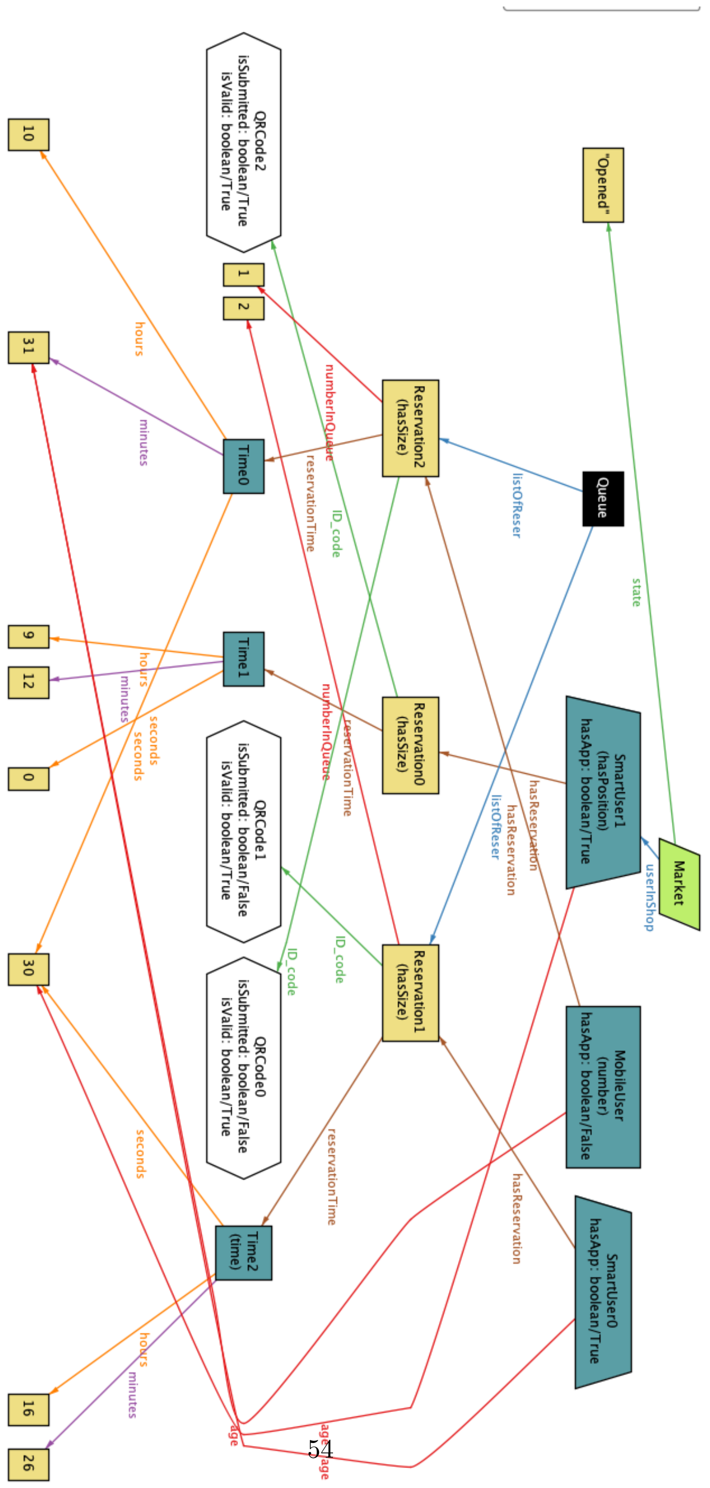
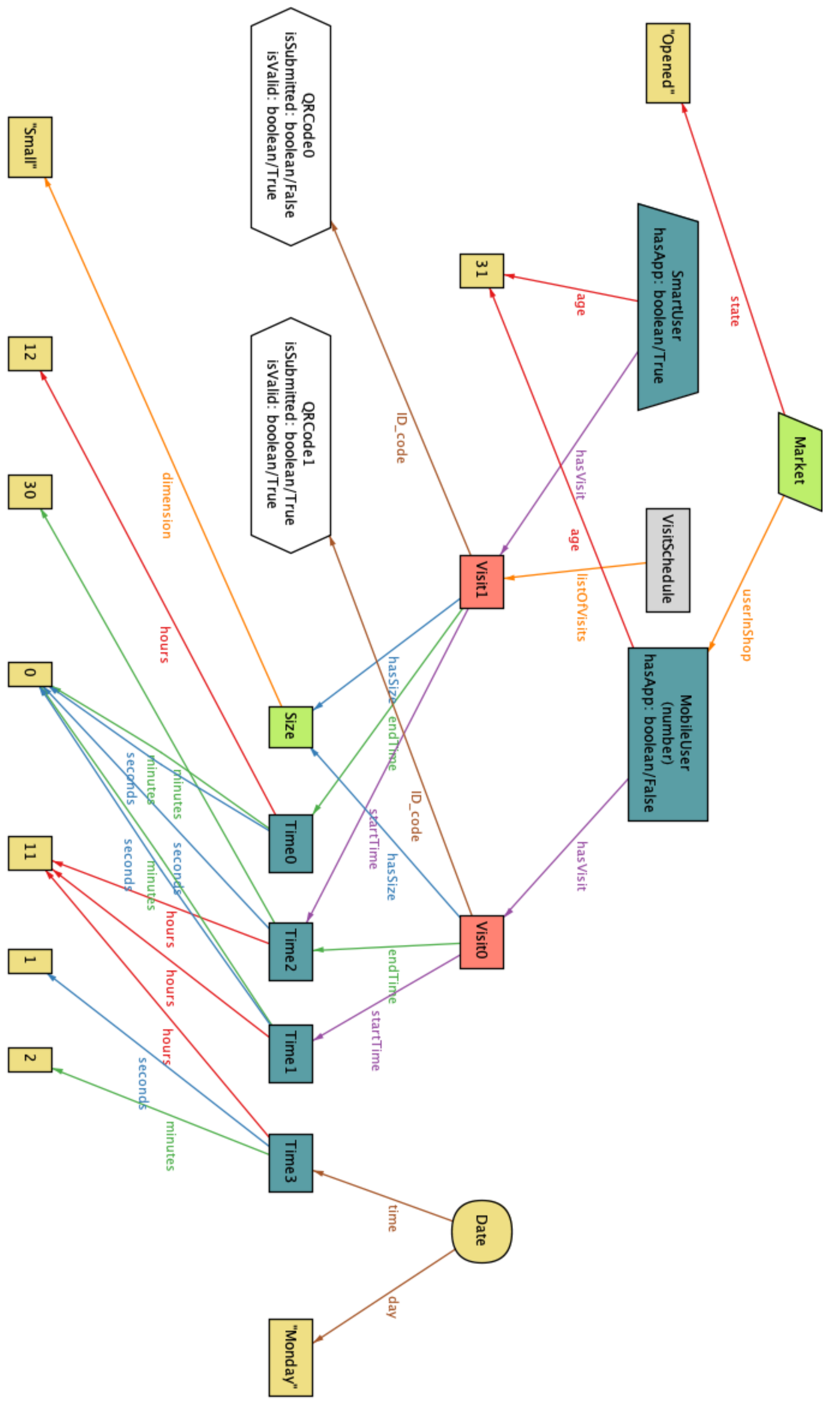
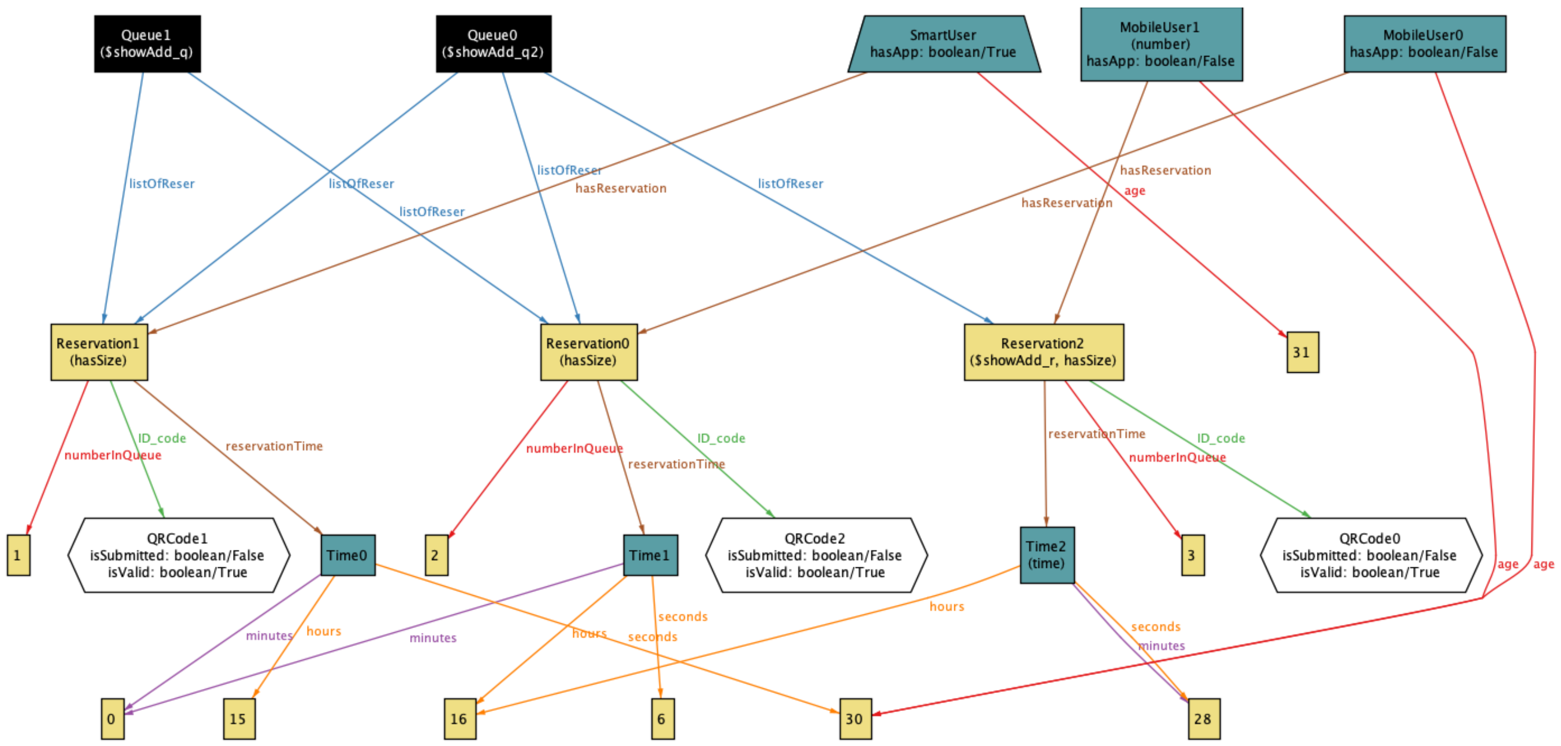


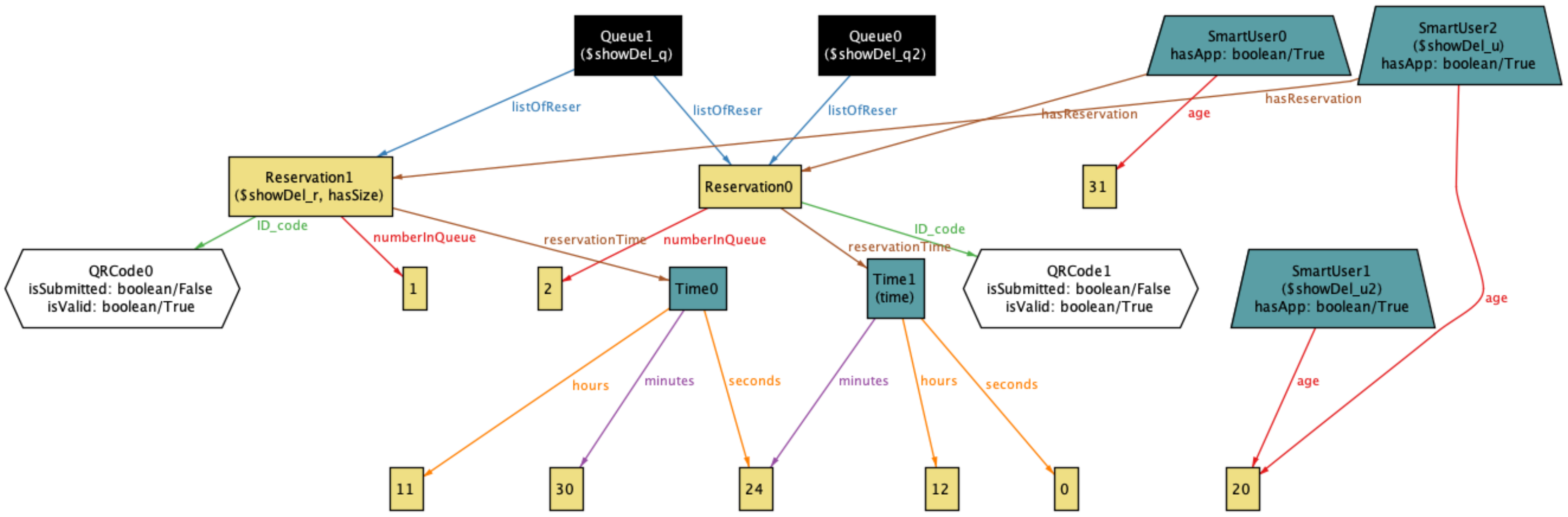
Figure 4.4: Rapresentation of the market opened in a generic day. This is the result of the showVisit predicate that focuses on Visits.





Queue1 ($showAdd_q)
Queue0 ($showAdd_q2)
SmartUser hasApp: boolean/True
MobileUser1 (number) hasApp: boolean/False
MobileUser0 hasApp: boolean/False
listOfReser
hasReservation
age
Reservation1 (hasSize)
Reservation0 (hasSize)
Reservation2 ($showAdd_r, hasSize)
31
numberInQueue
ID_code
reservationTime
QRCode1 isSubmitted: boolean/False isValid: boolean/True
QRCode2 isSubmitted: boolean/False isValid: boolean/True
QRCode0 isSubmitted: boolean/False isValid: boolean/True
Time0
Time1
Time2 (time)
hours
minutes
seconds
1
2
3
0
15
16
6
30
28

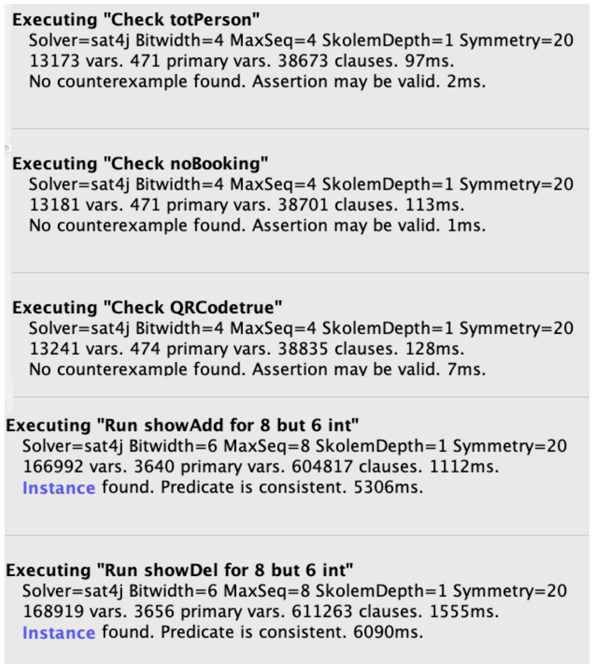
Figure 4.5: Queue insert reservation



Queue1 ($showDel_q)
Queue0 ($showDel_q2)
SmartUser0 hasApp: boolean/True
SmartUser2 ($showDel_u) hasApp: boolean/True
listOfReser
hasReservation
age
Reservation1 ($showDel_r, hasSize)
Reservation0
31
ID_code
numberInQueue
reservationTime
QRCode0 isSubmitted: boolean/False isValid: boolean/True
QRCode1 isSubmitted: boolean/False isValid: boolean/True
SmartUser1 ($showDel_u2) hasApp: boolean/True
Time0
Time1 (time)
hours
minutes
seconds
1
2
11
30
24
12
0
20

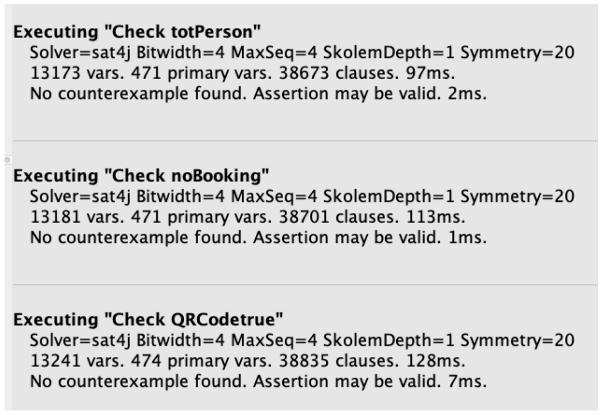
Figure 4.6: Queue remove reservation

4.2 Alloy Results



```
Executing "Check totPerson"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13173 vars. 471 primary vars. 38673 clauses. 97ms.
   No counterexample found. Assertion may be valid. 2ms.

Executing "Check noBooking"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13181 vars. 471 primary vars. 38701 clauses. 113ms.
   No counterexample found. Assertion may be valid. 1ms.

Executing "Check QRCodetrue"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13241 vars. 474 primary vars. 38835 clauses. 128ms.
   No counterexample found. Assertion may be valid. 7ms.

Executing "Run showAdd for 8 but 6 int"
   Solver=sat4j Bitwidth=6 MaxSeq=8 SkolemDepth=1 Symmetry=20
   166992 vars. 3640 primary vars. 604817 clauses. 1112ms.
   Instance found. Predicate is consistent. 5306ms.

Executing "Run showDel for 8 but 6 int"
   Solver=sat4j Bitwidth=6 MaxSeq=8 SkolemDepth=1 Symmetry=20
   168919 vars. 3656 primary vars. 611263 clauses. 1555ms.
   Instance found. Predicate is consistent. 6090ms.
```

Figure 4.7: Results of the predicates



```
Executing "Check totPerson"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13173 vars. 471 primary vars. 38673 clauses. 97ms.
   No counterexample found. Assertion may be valid. 2ms.

Executing "Check noBooking"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13181 vars. 471 primary vars. 38701 clauses. 113ms.
   No counterexample found. Assertion may be valid. 1ms.

Executing "Check QRCodetrue"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13241 vars. 474 primary vars. 38835 clauses. 128ms.
   No counterexample found. Assertion may be valid. 7ms.
```

Figure 4.8: Results of the assers

Chapter 5

Effort Spent

In the following tables it's summerized the effort time spent from us. In particular it has to be mentioned that we work together

Chapter/Task	**Hours**
Git setup	
Introduction	
Overall Description	
Specific Requirements	
Formal Analysis with Alloy	
Implementation Integration Test	2
	Total
	35,5

Table 5.1: Matteo Bresciani's effort

Chapter/Task	**Hours**
Git setup	
Introduction	
Overall Description	
Specific Requirements	
Formal Analysis with Alloy	
Implementation Integration Test	2
	Total
	35,5

Table 5.2: Stafano Banfi's effort

Chapter 6

References

6.1 Software used

- **LaTeX**: used to write and to build the document [https://www.draw.io/];
- **Draw**: used to create class, sequence and case diagram [https://www.draw.io/];
- **GitHub**: used to store and manage project repository [https://github.com/];
- **GitHub Desktop**: is the official GitHub application which allows us to contribute to the project repository in an easy way [https://desktop.github.com];
- **Figma**: used to design mockups provided in this document [https://www.draw.io/];
- **Alloy Tool**: used to describe the system model in Alloy language [https://www.draw.io/];

6.2 Bibliography

- Slides of *Software Engineering 2* course [https://beep.metid.polimi.it/];
- *R&DD Assignment A.Y. 2020-2021* [https://beep.metid.polimi.it/];
- *GDPR* [https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679];