POLITECNICO
MILANO 1863

# RASD

## Requirements Analysis and Specification Document

**Authors:**

**Antonio Pagliaroli**

**Filippo Pagliani**

**Davide Mangano**

Version:

Date:

Professor: Matteo Rossi

## Contents

# 1 Introduction

## 1.1 Purpose

Our *Requirements analysis and specification document (RASD)* contains the description of the scenarios, the most realistic use cases, and the models describing requirements and specification for the problem under consideration: CLup – Customers Line-up.

This document has the purpose to guide the developer in the realization of the software which can offer concrete help during the Covid-19 emergency.

## 1.2 Scope

CLup – Customers Line-up is an easy-to-use application that, on one side, allows store managers to regulate the influx of people in the building and, on the other side, saves people from having to line up and stand outside of stores for hours on end.

The necessity of an app like CLup rises in order to avoid having crowds inside the store, which typically results in long lines forming outside, which are themselves a source of hazards during the healthcare emergency.

The application would work as a digital counterpart to the common situation where people who are in line for a service retrieve a number that gives their position in the queue.

It offers to the clients three ways to visit the supermarket:

· **Mode 1:** it allows customers to “line up” (i.e., retrieve a number) from anywhere they want, and then wait until their number is called (or is close to being called) to approach the store. In addition, the application could be used to generate QR codes that would be scanned while entering the store, thus allowing store managers to monitor entrances and regulate the influx of people.

· **Mode 2:** in addition to managing lines in real-time, the application could also allow customers to “book” a visit to the supermarket. A customer might indicate also the approximate expected duration of the visit. Alternatively, for long-term customers, this time could be inferred by the system based on an analysis of the previous visits.

- **Mode 3:** fallback options should be available for people who do not have access to the required technology; for example, stores should also have the possibility to hand out physical tickets on the spot, thus acting as proxies for the customers.

The CLup app also includes features that allow the user to suggest alternative slots (of one day, or different days) to visit the store, or to recommend different stores in the same chain if the preferred one is not available, or to notify the available slots in one day/time range.

### 1.2.1 World Phenomena

| | |
|---|---|
| **WP1** | **User needs to go to a store** |
| **WP2** | **User has a smartphone** |
| **WP3** | **Store has long line at the entrance** |
| **WP4** | **User needs to respect rule in order to maintain social distancing due to the pandemic** |
| **WP5** | **Users need to buy different things according to their necessities** |

### 1.2.2 Shared Phenomena

| | |
|---|---|
| **SP1** | **User asks to line up in order to enter a shop** |
| **SP2** | **User is assigned to his place in a virtual line** |
| **SP3** | **CLup must generate a QR code for each user** |
| **SP4** | **Every store has to scan the QR code of the customer to monitor entrances and exits** |
| **SP5** | **CLup notifies the user when it is time to go to the store** |

| | |
|---|---|
| SP6 | User can require a physical ticket for the queue |
| SP7 | User can book a visit to the store for a future moment |
| SP8 | User indicates the categories of what he wants to buy and the duration of the visit |
| SP9 | CLup can suggest to the user alternative slot if the chosen one is occupied |
| SP10 | CLup can suggest a different store if the chosen one is full |
| SP11 | CLup can periodically suggest free slot when they are available based on the habits of the user |

### 1.2.3 Goals

| | |
|---|---|
| G1 | Allows store managers to regulate the influx of people in the building |
| G2 | Allows people to avoid hazards and wasting time lining up outside of the store |
| G3 | Makes it easier to respect social distancing inside the store |
| G4 | Allows every demographic the possibility to use the service easily |

|  |  |
|---|---|
| G5 | Notifies the user of the right time he needs to leave the house and go to the store by keeping track of the distance (between his house and the supermarket) and his travelling habits (by car, on foot, exc.) |
| G6 | Gives the store the possibility to hand out physical ticket if the user has not access to the needed technology |
| G7 | Allows people to book their visit in a future moment |
| G8 | Stores the data concerning the visits of an user |
| G9 | Uses the stored data to find the optimal way to plan the visits |
| G10 | Gives the users the possibility to always find the nearest slot to their necessity, even if in other stores |

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

### 1.3.2 Acronyms

|  |  |
|---|---|
| OTP | One Time Password |
| GPS | Global Positioning System |

### 1.3.3 Abbreviations

| | |
|---|---|
| **WPn** | **World phenomena number n** |
| **SPn** | **Shared phenomena number n** |
| **Gn** | **Goal number n** |

## 1.4 Revision History

This is the first version of the project.

## 1.5 Reference Documents

· Specification Document: “CLup-Customers Line-up Mandatory Project Assignment.pdf”

· Slides of the lectures

# 2. Overall Description

## 2.1 Product Perspective

CLup, or “Customer Line-Up”, it’s a service developed with the purpose of helping both the customer that is using the service and the shop where they are going to. The first one can optimize their visits since they can line up from remote, saving the time that would be wasted in the queue outside; on the other hand the shop can simply offer a better service to his clients and have a steady flow in, avoiding big crowd inside and outside of the shop itself, which is a very serious problem in this specific historic moment. Indeed, even if it could be very useful regardless of this situation, CLup can perform at its best if inserted in a pandemic situation like the current one, since it can help maintain social distance and avoid gatherings.

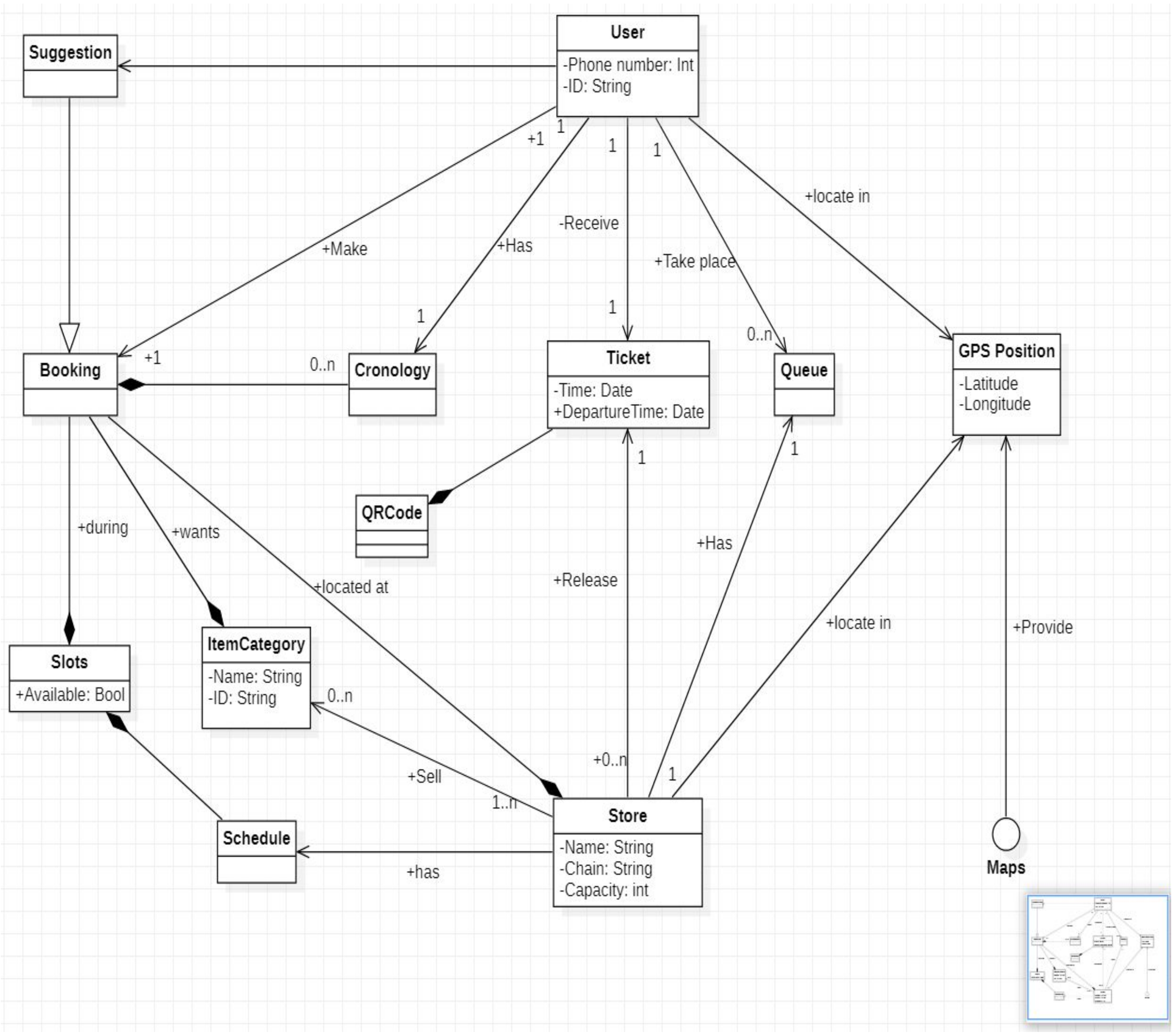
The first time that the user accesses the system, he has to register himself simply providing a valid password and his number, that will be used as his username and will be verified through a OTP that will be sent to it.

The user can interact with the service mainly through three way, but for the first two he has to register himself or log in:

- Once the user has authenticated himself he can select the store in which he would like to go and line up for it, then the system will communicate to him his position in the queue and at which time he should leave the house in order to reach the shop in time.
- Once the user has authenticated himself he can select the store where he would like to go in order to visualize a schedule in which it can be seen all the available slots. Then the user can select the one that better fits with his needs and book his place in line for the time and date selected.
- As a fallback option to make the service accessible to everyone, including the ones that do not have access to a smartphone, the user has the possibility to go physically to the store and ask at the place to line up just leaving his phone number as an identification. At this point the store will give to the customer a ticket that indicates his place in line and the estimated time to wait.

### 2.1.1 UML Description

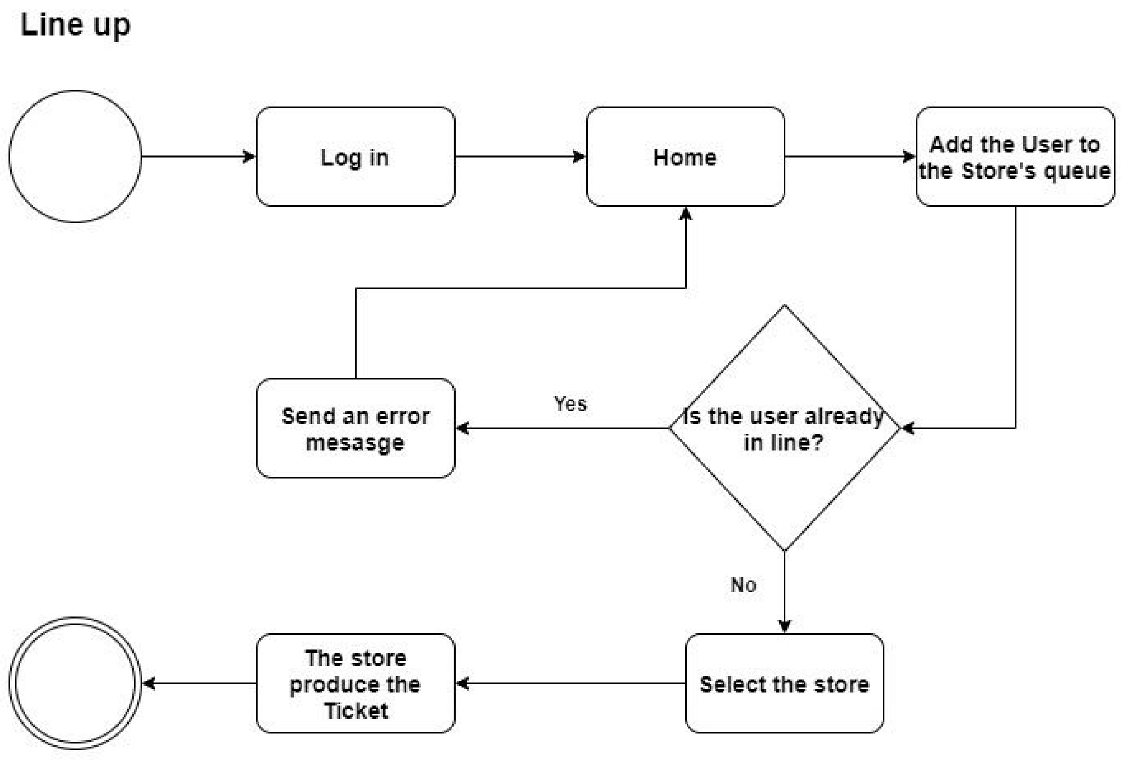
The UML below describes at a high level the model of the system that has to be developed, however it does not pretend to represent every class that would be necessary to define the complete architecture of the system. The user can both download the application on his/her device in order to access the complete service or just use the fallback option if he hasn't access to the required technology.



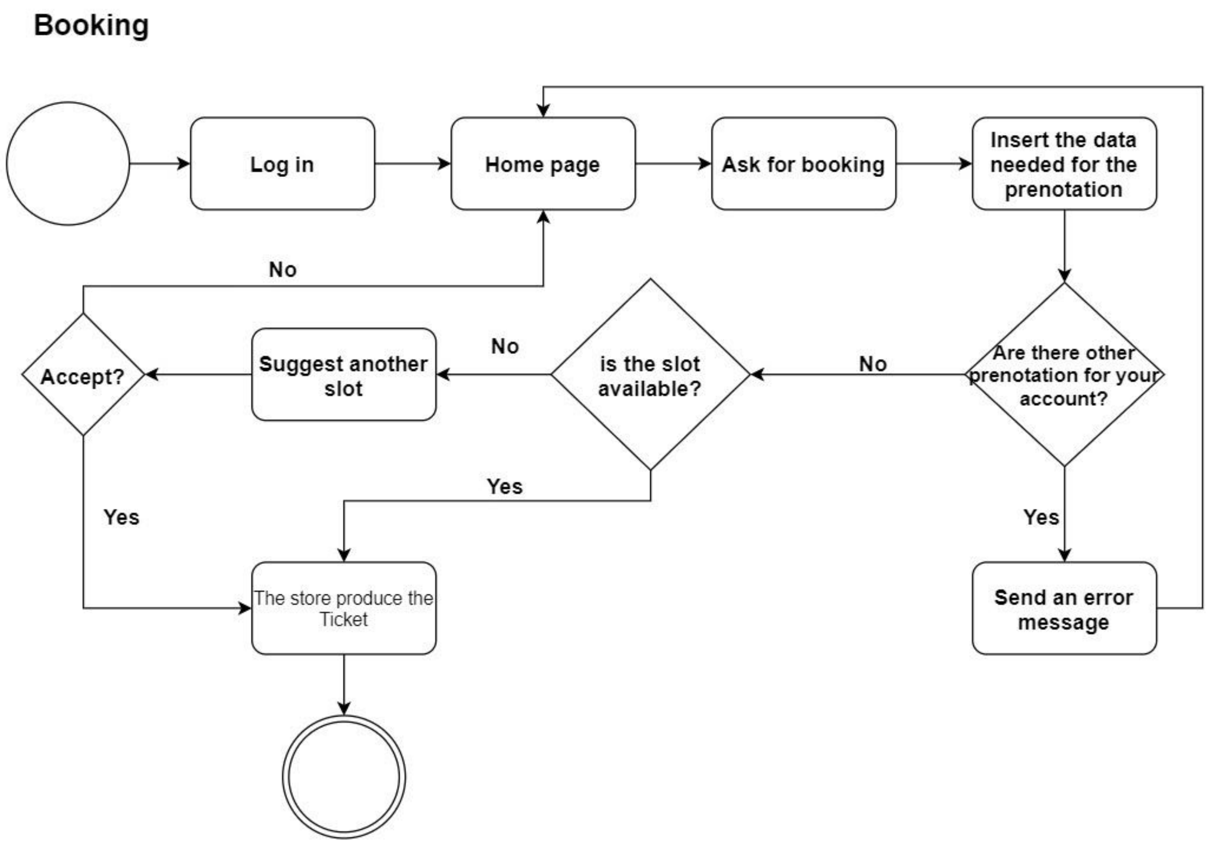
## 2.1.2 State Charts

Now we are going to analyse some critical aspects of the application, modelling their behaviours and showing the evolution over time of their states through the appropriate state diagrams, which are reported below.

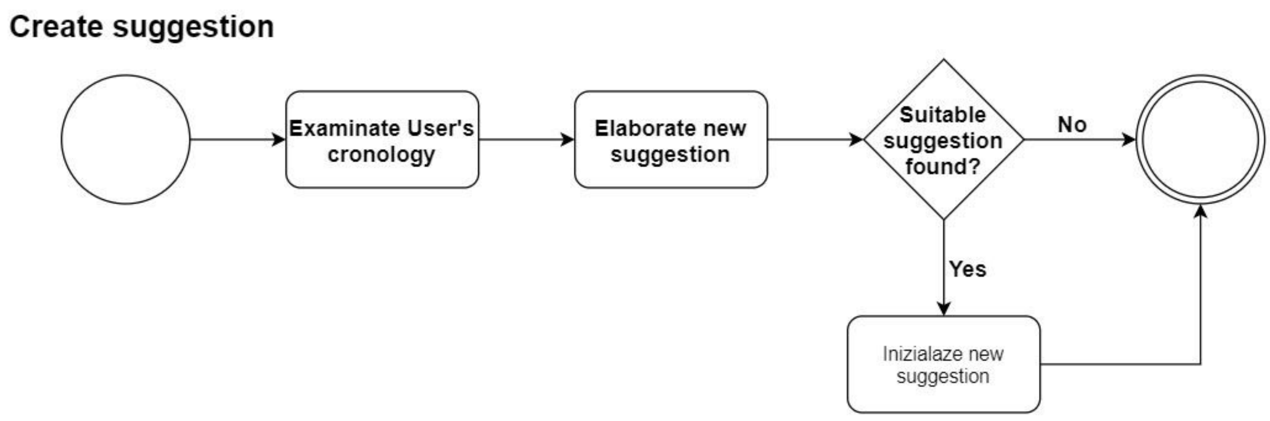
In the first state diagram (figure 1) it is described how a user can get in line showing how the state of the system is changing during the crucial phase of the process:


Line up
Log in
Home
Add the User to the Store's queue
Is the user already in line?
Yes
Send an error mesasge
No
Select the store
The store produce the Ticket

In the second diagram (figure 2) it is described how a user can book a slot in the schedule of a store showing how the state of the system is changing during the crucial phase of the process:



In this last diagram is described how the system can elaborate new suggestions for the user based on his recent visits, showing how the state of the system is changing during the crucial phase of the process:



Create suggestion
Examinate User's cronology
Elaborate new suggestion
Suitable suggestion found?
No
Yes
Inizialaze new suggestion

## 2.2 Product Functions

Here are described some functions of the software. Be aware that some products functions are described also in other parts of this document, particularly in the goal parts and in the functionals requirements parts.

**Line up virtually**

Along with the possibility to book a future visit this function represents the core of the service. Clup offers to the user the possibility to take his place in the queue of a store by remote: Once the user has authenticated himself he can use the application to select a store from a list of available shops and indicate if he’s going alone or with a maximum of four people. Afterwards the system lets him know how much time he has to wait before his turn and even the suggested time of departure (calculated starting from your actual position) so that he can arrive at the store at the perfect time. Finally along with these informations CLup releases a QR code that is required to enter the store and that has to be scanned at the exit.

**Book a visit**

Another very important function of CLup is to give the user the possibility to plan a visit for the future.First of all the user has to identify himself by registering himself or logging in, then he has to select the store he wants to go to from a list of available ones. Subsequently a schedule that shows the free slots available can be viewed in order to select the best option according to the user's needs. Afterwards the system can release the confirmation of his booking indicating the day, the hour, the store selected and providing to the user the QR code that is required to enter the store and that has to be scanned at the exit.

**Line Up physically**

As a fallback option the system offers the chance to those who do not have the possibility to access the technology required to go physically to the store and line up. In this case the store itself has the possibility to add a person in the virtual line through his account and then release a ticket to the client that indicates his place in line, the hour that he is expected to enter the store and the QR code that is required to enter the store and that has to be scanned at the exit.

**Suggest good spot for future visit**

CLup can help the user to find the best moments according to his needs thanks to the data that are stored about every past visit related to his account. According to this statistics the system will try to predict when the user has the necessity of visiting a store and which one he wants to go to in order to suggest the most suitable slot still available before it is booked by someone else.

**Elaborate an optimized schedule for each store**

CLup takes responsibility for the optimization of the schedule on the side of the store. Indeed the timetable is not build just by establishing a certain number of slots per hour based on the size of the store, but by crossing the data provided by the user in his booking about the duration of the visit and the list of items needed, together with the data that are given by the store about his capacity and the organization of the item for sale, in order to guarantee the optimal flow of people to avoid gatherings.

## 2.3 User characteristics

The application has just two type of actors:

- **User:** someone who wants to download the app to have access to the service provided by CLup. Once he has downloaded the app, if it’s the first time he accesses to the service, he has to register himself, otherwise he have just to log in, then he gains access to all the services provided by the application;
- **Store:** it’s the counterpart of the physical store, it has to be registered in order to be indexed in the application. For the registration it has to provide every data needed to the system in order to be recognized and scheduled: name, position,opening hours, if it’s part of a chain or not, the capacity of the store, a validation document and the organization of the items inside the store (more information on the registration can be found later in the document). Once that is logged in, the store has the important task of adding to the queue the users that choose to line up physically (by asking a physical ticket).

## 2.4 Assumptions, dependendencies and constraints

### 2.4.1 Domain assumption

| | |
|---|---|
| **D1** | **The internet connection works properly** |
| **D2** | **Every user must have a phone number if he wants to use the app** |
| **D3** | **Each smartphone must have a GPS sensor** |
| **D4** | **GPS is active when the user runs the app** |

| | |
|---|---|
| D5 | GPS provides the location of the user with a possible error of 100 meters [at most] |
| D6 | The superstore has a max number of clients at the same time |
| D7 | The store managers have a QR code reader |
| D8 | A client cannot enter a supermarket without a number or a reservation or a physical ticket |
| D9 | The app must integrate data received by the store |
| D10 | Users must arrive at the shop in the worst case 10 minutes before or after the indicated time |
| D11 | Every booking contains a date, a starting time and a duration |
| D12 | User can’t book a visit when the store is closed |
| D13 | Each store has outside at least one spot used to print physical tickets |

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

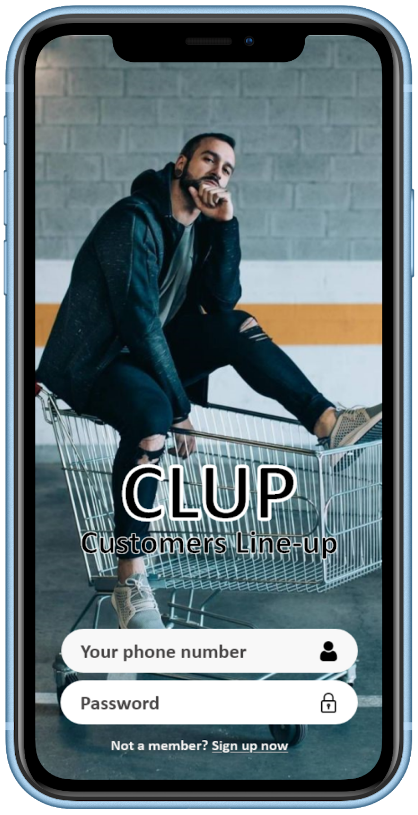
The user has two possibilities to use the service:

The first possibility is the use of a device that allows him to share his position, the date and the time. With this device the user can download the app CLup from the default store of his system (GooglePlay, Apple store, …) and after that he can book a visit or line up in the virtual queue.

The second possibility allows the offline use of the service. Indeed if the users doesn't have access to the required technology, he can anyway use the service thanks to the stores' possibility to hand out physical tickets.

The following mockups give the basic idea of how:

- the icon of the application looks like;
- the mobile app will look like in the login page.



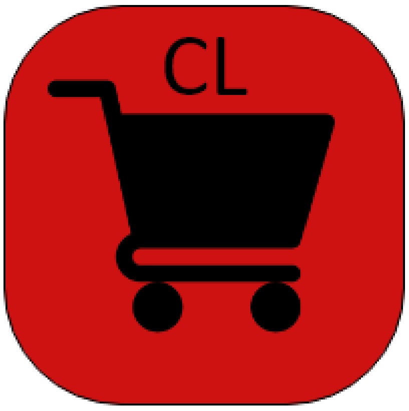


Figure 2 - Icon of the app

Figure 1 - Login Page

### 3.1.2 Hardware Interfaces

The software application contains three main functions and two types of actors, which require different kinds of hardware interfaces.

- For all the functionalities:

  The users must own a smartphone or use a computer with a GPS system in order to book a visit. If they just want to line up in the virtual queue they can also use the slots of the store where they can request a physical ticket. The notification system is available only with the app or the WebApp.

  The stores, instead, can access the system through the App or the WebApp. They first have to register themselves to the system and then they can receive data about the visits and statistics about the shopping.

### 3.1.3 Software Interfaces

The system uses the following external interfaces:

- Regions map:

  We assume that the the system uses a public API to provide to the user:

    - His position in real time for the calculation of travel time by different means of transport (car, on foot, …)
    - A map of all available stores to visit or book

### 3.1.4 Communication Interfaces

The device connects to CLup system via internet connection.

## 3.2 Functional Requirements

### 3.2.1 List of Requirements

| | |
|---|---|
| **R1** | **User can require a ticket or booking a visit only with his telephone number** |
| **R2** | **Shops are certified with an authentication** |
| **R3** | **Shops should register to the application by filling the form with mandatory fields*** |
| **R4** | **Each store receives data only about its clients** |
| **R5** | **Each booking or request to line up must be certificated by a code sent by SMS** |
| **R6** | **Users shall authorise the system to send messages** |
| **R7** | **Users shall authorise the system to use their position** |
| **R8** | **The system must have the access to the data of the store** |
| **R9** | **Users shall authorise the system to acquire their data about purchasing habits** |
| **R10** | **A request of booking is also validated by the system before being confirmed** |
| **R11** | **Shops has no restrict to access the data of their customers** |
| **R12** | **Shops and users have to choose from a list of predetermined items (this is done for users to filter stores, in order to search for the ones providing desired items)** |
| **R13** | **The system has to coordinate physical slot with the app in order to have sequential tickets** |
| **R14** | **Users can choose the type of items from a predefined checklist when booking. They can select “no preference”** |
| **R15** | **Users that want to book a visit should compile a form in all its mandatory fields** |
| **R16** | **The booking system and the line system must be coordinated to ensure social distancing** |
| **R17** | **User can view in a registry the bookings already done** |

| | |
|---|---|
| **R18** | **User has to wait for the booking to be completed before booking again** |
| **R19** | **The system suggests possible alternatives to the clients** |
| **R20** | **The system must suggest other shops in the same municipality** |
| **R21** | **The system must suggest other time slots up to one week later** |
| **R22** | **The notification system is based on the client's habits** |
| **R23** | **The system can build statistics about client’s habits by considering precedent visits** |
| **R24** | **Each ticket and each booking have a QR code** |

### 3.2.2 Mapping

| Goals | Domain Assumption | Requirements |
|---|---|---|
| **G1** | D6, D7, D8 | R2, R3, , , R7, R8, R12, R13, R16, R18, R19, R24 |
| **G2** | D1, D3, D4, D5, D10, D12, D13 | R7, R9, R10, R12, R14, R15, , |
| **G3** | D6, D8, D10 | R9, R11, R14, R16, R19, R20, R21 |
| **G4** | D2, D13 | R1, R13, R24 |
| **G5** | D2, D3, D4, D5, D10 | R4, R7, R9, R11 |
| **G6** | D8, D9, D13 | R13, R24 |
| **G7** | D1, D2, D3, D4. D5, D8, D11, D12 | R1, R2, R5, , R6, R7, R10, R14, R15, R16, , , R17, R18 |
| **G8** | D1, D9, D11 | R2, R4, R7, R8, R9, R11, R17, R23 |
| **G9** | D6, D8, D10, D11 | R5, , R9, R11, R12, R14, R22, R23 |

|  |  |  |
|---|---|---|
| **G10** | D1, D2, D3, D4, D5, D8 | R2, , R6, R7, R8, R9, R11, R19, R20, R21, R22 |

|  |  |
|---|---|
| G1 | Allows store managers to regulate the influx of people in the building |
| D6 | The superstore has a max number of clients at the same time |
| D7 | The store managers have a QR code reader |
| D8 | A client cannot enter a supermarket without a number or a reservation or a physical ticket |
| D13 | Each store has outside at least one spot used to print physical tickets |
| R2 | Shops are certified with an authentication |
| R3 | Shops should register to the application by filling the form with mandatory fields |
| R7 | Users shall authorise the system to use their position |
| R8 | The system must have the access to the data of the store |
| R12 | Shops and users have to choose from a list of predetermined items (this is done for users to filter stores, in order to search for the ones providing desired items) |
| R13 | The system has to coordinate physical slot with app in way to have sequential tickets |
| R16 | The booking system and the line system must be coordinated to ensure social distancing |
| R18 | User has to wait for the booking to be completed before booking again |
| R19 | The system suggests possible alternatives to the clients |
| R24 | Each ticket and each booking have a QR code |
| G2 | Allows people to avoid hazards and wasting time lining up outside of the store |
| D1 | The internet connection works properly |
| D3 | Each smartphone must have a GPS sensor |
| D4 | GPS is active when the user runs the app |
| D5 | GPS provides the location of the user with a possible error of 100 meters [at most] |
| D10 | Users must arrive at the shop in the worst case 10 minutes before or after the indicated time |
| D12 | User can’t book a visit when the store is closed |
| D13 | Each store has outside at least one spot used to print physical tickets |
| R7 | Users shall authorise the system to use their position |
| R9 | Users shall authorise the system to acquire their data about purchasing habits |

| | |
|---|---|
| R10 | A request of a booking it is also validated by the system before being confirmed |
| R12 | Shops and users have to choose from a list of predetermined items to extract information by queries |
| R14 | Users can choose the type of items from a predefined checklist. They can select “no preference” |
| R15 | Users that want book a visit should compile a form in all its mandatory fields |
| R16 | The booking system and the line system must be coordinated to ensure social distancing |
| G3 | Makes it easier to respect social distancing inside the store |
| D6 | The superstore has a max number of clients at the same time |
| D8 | A client cannot enter a supermarket without a number or a reservation or a physical ticket |
| D10 | Users must arrive at the shop in the worst case 10 minutes before or after the indicated time |
| R9 | Users shall authorise the system to acquire their data about purchasing habits |
| R11 | Shops have no restrict to access to the data of their customers |
| R14 | Users can choose the type of items from a predefined checklist. They can select “no preference” |
| R16 | The booking system and the line system must be coordinated to ensure social distancing |
| G4 | Allows every demographic the possibility to use the service easily |
| D2 | Every user must have a phone number if he want to use the app |
| D13 | Each store has outside at least one spot used to print physical tickets |
| R1 | User can require a ticket or booking a visit only with his telephone number |
| R13 | The system has to coordinate physical slot with app ain way to have sequential tickets |
| R24 | Each ticket and each booking have a QR code |
| G5 | Notifies the user of the right time he needs to leave the house and go to the store by keeping track of the distance (between his house and the supermarket) and his travelling habits (by car, on foot, exc.) |
| D2 | Every user must have a phone number if he want to use the app |
| D3 | Each smartphone must have a GPS sensor |
| D4 | GPS is active when the user runs the app |
| D5 | GPS provides the location of the user with a possible error of 100 meters [at most] |
| D10 | Users must arrive at the shop in the worst case 10 minutes before or after the indicated time |

| | |
|---|---|
| R4 | Each shop receives data only about its clients |
| R7 | Users shall authorise the system to use their position |
| R9 | Users shall authorise the system to acquire their data about purchasing habits |
| R11 | Shops have no restrict to access to the data of their customers |
| G6 | Gives the store the possibility to hand out physical tickets if the user has not access to the needed technology |
| D8 | A client cannot enter a supermarket without a number or a reservation or a physical ticket |
| D9 | The app must integrate the data received by the store |
| D13 | Each grocery shop has outside at least one spot used to printing physical tickets |
| R13 | The system has to coordinate physical slot with app ain way to have sequential tickets |
| R24 | Each ticket and each booking have a QR code |
| G7 | Allows people to book their visit in a future moment |
| D1 | The internet connection works properly |
| D2 | Every user must have a phone number if he want to use the app |
| D3 | Each smartphone must have a GPS sensor |
| D4 | GPS is active when the user runs the app |
| D5 | GPS provides the location of the user with a possible error of 100 meters [at most] |
| D8 | A client cannot enter a supermarket without a number or a reservation or a physical ticket |
| D11 | Every booking contains a date, a starting time and a duration |
| D12 | User can’t book a visit when the store is closed |
| R1 | User can require a ticket or booking a visit only with his telephone number |
| R2 | Shop are certified with an authentication |
| R5 | Each booking or request to line up must be certificated by a code sent by message |
| R6 | Users shall authorise the system to send messages |
| R7 | Users shall authorise the system to use their position |
| R10 | A request of a booking it is also validated by the system before being confirmed |
| R15 | Users that want book a visit should compile a form in all its mandatory fields |
| R16 | The booking system and the line system must be coordinated to ensure social distancing |

| | |
|---|---|
| R17 | User can view in a registry the bookings already done |
| R18 | User has to wait for the booking to be completed before booking again |
| G8 | Stores the data concerning the visits of an user |
| D1 | The internet connection works properly |
| D9 | The app must integrate the data received by the store |
| D11 | Every booking contains a date, a starting time and a duration |
| R2 | Shops are certified with an authentication |
| R4 | Each shop receives data only about its clients |
| R7 | Users shall authorise the system to use their position |
| R8 | The system must have the access to the data of the store |
| R9 | Users shall authorise the system to acquire their data about purchasing habits |
| R11 | Shops have no restrict to access to the data of their customers |
| R17 | User can view in a registry the bookings already done |
| R23 | The system can build statistics about client's habits considering precedent visits |
| G9 | Uses the stored data to find the optimal way to plan the visits |
| D6 | The superstore has a max number of clients at the same time |
| D8 | A client cannot enter a supermarket without a number or a reservation or a physical ticket |
| D10 | Users must arrive at the shop in the worst case 10 minutes before or after the indicated time |
| D11 | Every booking contains a date, a starting time and a duration |
| R5 | Each booking or request to line up must be certificated by a code sent by message |
| R9 | Users shall authorise the system to acquire their data about purchasing habits |
| R11 | Shops have no restrict to access to the data of their customers |
| R12 | Shops and users have to choose from a list of predetermined items (this is done for users to filter stores, in order to search for the ones providing desired items) |
| R14 | Users can choose the type of items from a predefined checklist. They can select "no preference" |
| R22 | The notification system is based on the client's habits |
| R23 | The system can build statistics about client's habits considering precedent visits |
| G10 | Gives the users the possibility to always find the nearest slot to their necessity, even if in |

| | |
|---|---|
| | other stores |
| D1 | The internet connection works properly |
| D2 | Every user must have a phone number if he want to use the app |
| D3 | Each smartphone must have a GPS sensor |
| D4 | GPS is active when the user runs the app |
| D5 | GPS provides the location of the user with a possible error of 100 meters [at most] |
| D8 | A client cannot enter a supermarket without a number or a reservation or a physical ticket |
| R2 | Shops are certified with an authentication |
| R6 | Users shall authorise the system to send messages |
| R7 | Users shall authorise the system to use their position |
| R8 | The system must have the access to the data of the store |
| R9 | Users shall authorise the system to acquire their data about purchasing habits |
| R11 | Shops have no restrict to access to the data of their customers |
| R19 | The system suggests possible alternatives to the clients |
| R20 | The system must suggest other shops in the same municipality |
| R21 | The system must suggest other time slots up to one week later |
| R22 | The notification system is based on the client's habits |

### 3.2.3 Use Cases

In the following tables, some examples of use cases are shown.

#### 3.2.3.1 Use Cases Description

1. Enter a queue using the app

| | |
|---|---|
| **Name** | Enter a queue using the app |
| **Actors** | User |
| **Entry Condition** | User and store are logged in (on Web Page/App and they |

| | |
|---|---|
| | have an internet connection) |
| **Event Flow** | 1. User clicks “Enter queue” button<br>2. He chooses the best option between available alternatives in his municipality and clicks it, or he can find every store he wants by clicking the searching bar and then typing its name (and then clicking it). He must insert the number of people entering with him in a text box in this page (Max 5)<br>3. A notification of “Incoming enter queue request” is then sent to the server. Then:<br>a. *If the server accepts the request:* a notification is sent to the user, reporting his waiting time and corresponding time he should leave (based on his travelling habits).<br>b. *If the server refuses the request, or it doesn’t reply in a predetermined time:* a notification of unavailability is sent to the user, then it is asked if he wants to find another store or he wants to book to this store |
| **Exit Conditions** | The user is either in a queue or booked. |
| **Exceptions** | 1. **The store could have reached the limit;** this case should be covered by CLup, but there is the option for the store to send a report specifying it is currently full.<br>2. **The user is already in the queue;** in this case, an error notification is sent to the user. |

### 2. Registration of Store

| | |
|---|---|
| **Name** | Registration of Store |
| **Actors** | Store |
| **Entry Condition** | Store needs to be connected to the internet and, precisely, to the web page or the app of CLup |
| **Event Flow** | 1. Store enters the web page or the app<br>2. Store clicks on “Sign Up” button<br>3. Store inserts the credentials that are asked, together with a certificate that shows validity of data written |

| | |
|---|---|
| | 4. It inserts its own chain via ID number<br>5. Store clicks on “Register” button<br>6. The system validates the data inserted<br>7. An “Accept” alert is sent to the web page or app where the store accessed<br>8. The store clicks on “Seen validation” |
| **Exit Conditions** | The store is now in the system as a valid choice for users. |
| **Exceptions** | 1. **Data inserted are not correct;** in this case:<br>1.1 The application asks store to insert data again; if done > 3 times, the system refuses the store in a definitive way<br>2. **The store is already registered;** in this case a notification is sent to the store<br>3. **The system does not recognise the validation;** in this case:<br>2.1 The system sends an “Error” to store, and the store can:<br>2.1.1 Click on “Acknowledge” and send a claim to the App owners, or<br>2.1.2 Click on “Acknowledge” and exit the app or web page. |

### 3. Book a visit using the app

| | |
|---|---|
| **Name** | Book a visit using the app |
| **Actors** | User |
| **Entry Condition** | User has the App installed on its device and is logged in (so it has an internet connection) |
| **Event Flow** | 1. User clicks on booking option<br>2. User chooses the store that fits his needs and clicks on it in the “Choosing queue to book” page<br>3. It is then shown a box where he can choose month, day, hour and minute he would like to visit the store, together with a box where he has to write how many people will access the store with him. All of these are mandatory fields, so the user has to fill them up; in the end, he can fill a box where he writes the items he needs to buy. After doing so, he clicks on “Book a visit”<br>4. The app then shows the time he should leave his |

| | |
|---|---|
| | house based on the distance and his travelling habits, and the average people that will be present in the store at that time; then the user clicks on “Ok” |
| **Exit Conditions** | User has booked a visit successfully. |
| **Exceptions** | 1. **The store is not available:** suggestions are given to the user about alternative choices.<br>2. **User has already booked;** a notification is sent to the user |

## 4. Send available tickets to print

| | |
|---|---|
| **Name** | Send available tickets to print |
| **Actors** | Store |
| **Entry Condition** | Store is logged in and a user requested to enter by asking for a physical ticket. |
| **Event Flow** | 1. Store clicks on “Options”<br>2. Store clicks on “Alternative entrance”<br>3. Store clicks on “Print Ticket” (there is also alternative of booking)<br>4. CLup successfully enters the user in a queue: the ticket with the QR code is then sent to the store which will print it. |
| **Exit Conditions** | The user is in a queue, with a physical ticket showing the approximate waiting time. |
| **Exceptions** | **CLup refuses the entrance, dued to possible cases of gathering:** the ticket is not sent, instead it is sent a list of stores of the same chain which are available. If there is no close store available of the same chain, different chain’s ones are suggested.<br>Either way, the user is asked if he wants to choose another store or book this one. |

## 5. Registration of user

| | |
|---|---|
| **Name** | Registration of user |
| **Actors** | User |
| **Entry Condition** | The user has a device that can connect to the internet and a working internet connection |
| **Event Flow** | 1. The user clicks on “Register”<br>2. The user clicks on “Register with phone number”<br>3. The user inserts its phone number in a text box, then inserts a password in another textbox, then clicks “Register”. An OTP is then sent to that specified number and the user has to insert it in a text box, then clicks “Ok”. The user needs to click on “Confirm registration” to complete his account. The first time the user enters the app, it is asked if he wants to give permission to treat his data (he can change his decision in “Option”>”Privacy”>”Treatment of data”) |
| **Exit Conditions** | The user is registered in the service. |
| **Exceptions** | ● **The user is already registered;** in this case, a notification is sent of an already existing user; it is then asked if the user has forgotten the mail/password of the account.<br>● **The user did not complete all mandatory fields;** in this case, the user is asked to complete all fields marked in red.<br>● **The user inserted an invalid phone number.**<br>● **The user fails to insert OTP.** |

### 6. Stop the queue

| | |
|---|---|
| **Name** | Stop the queue |
| **Actors** | Store |
| **Entry Condition** | The store is logged in the system (so it is on the Web page/App of CLup and has an internet connection) and it has a sanitary emergency (main example: Covid19 case inside of it) |
| **Event Flow** | 1. The store clicks on “Options”<br>2. The store clicks on “Queue” |

| | |
|---|---|
| | 3. The store clicks on “Stop the queue”. Then it is sent to a page where it has to fill a form, by explaining the motivation behind the emergency closing (e.g. incident happened inside, or Covid19 case inside) and if it is for Covid, it is communicated to authorities. It is also asked the expected time of closure; if it is not known, the queue is assumed stopped until further notification<br>4. The store clicks on “Stop the queue” and then , on confirmation page, on “Confirm”<br>5. A notification is then sent to everyone who was on the queue/had booked a visit during the closure time (it is then asked to them if they want to book a visit to this same store or choose another one). Also an alert is sent to people inside at that time. |
| **Exit Conditions** | The store has stopped its queue and it enters a closure time. |
| **Exceptions** | No exceptions. |

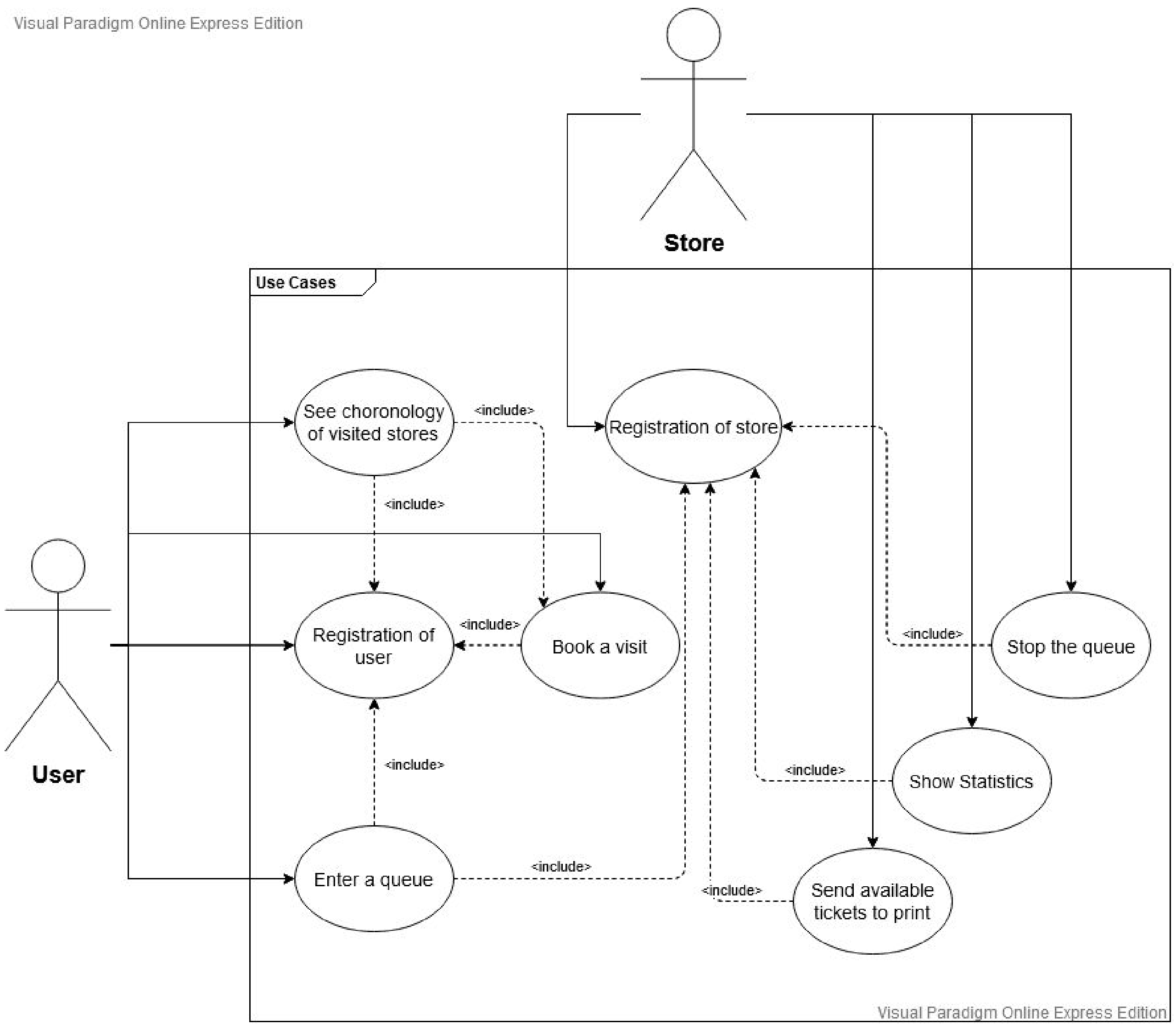
### 7. Show statistics

| | |
|---|---|
| **Name** | Show statistics |
| **Actors** | Store |
| **Entry Condition** | The store is logged in. |
| **Event Flow** | 1. Store clicks on “Options”<br>2. Store clicks on “Statistics”<br>3. A diagram is then shown, showing the trend of the visiting people. It is also shown on how many “Best performing stores for me” it appeared, and also how many people put it as “Preferred” (see next use case)<br>4. Store can click on “Week/Month/Year” to change the time scope |
| **Exit Conditions** | Statistics are shown to store. |
| **Exceptions** | No exceptions. |

## 8. See chronology of visited stores

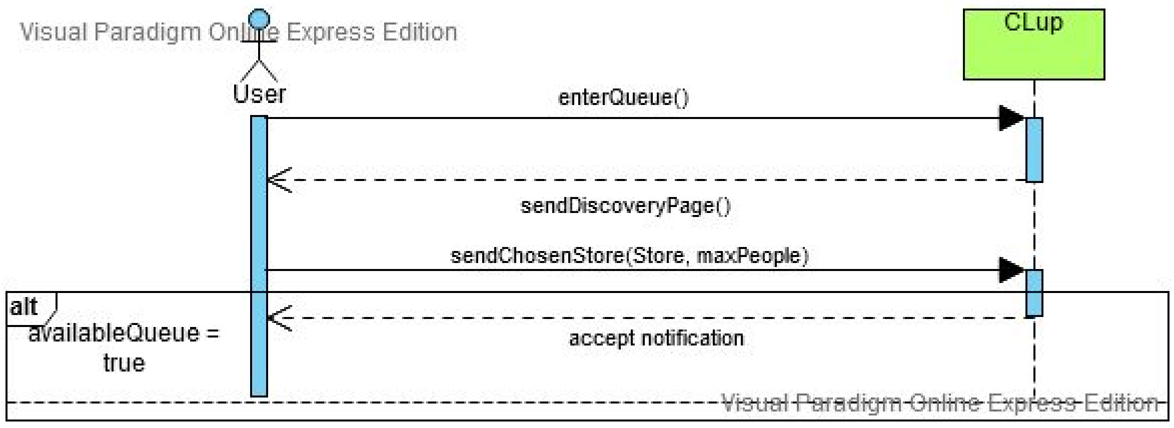
| | |
|---|---|
| **Name** | See chronology of visited stores |
| **Actors** | User |
| **Entry Condition** | The user is logged in the app/Web page and gave consent to the treatment of his data. |
| **Event Flow** | 1. The user clicks on “Options”<br>2. He clicks on “Statistics”. Now an accurate graphical representation of the user chronology of visited stores is shown. He can:<br>a. Click on “Week/Month/Year ago” to modify the period shown.<br>b. Click on “Show more” to see the chart of the most visited stores in the last Week/Month/Year.<br>c. Click on “Best performing stores for me” to see the stores which held minimum waiting time, as well as minimum travelling time (staying time is not kept in mind).<br>d. Click on the star next to the store name to choose it as its favourite (it will be shown ahead of others in the choosing queue page) |
| **Exit Conditions** | The user has its information on its device. |
| **Exceptions** | **The user did not give consent to the treatment of its data.** |

### 3.2.3.2 Use Cases Diagram

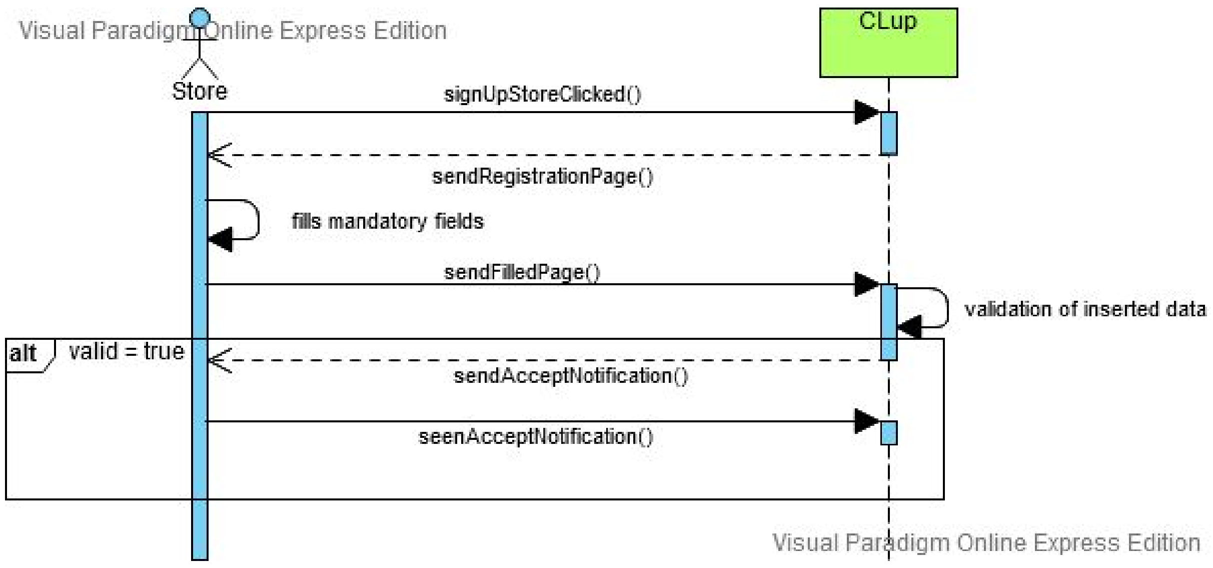

Visual Paradigm Online Express Edition
Store
Use Cases
User
See choronology of visited stores
Registration of store
Registration of user
Book a visit
Stop the queue
Show Statistics
Enter a queue
Send available tickets to print
<include>
Visual Paradigm Online Express Edition

### 3.2.4 Sequence Diagrams

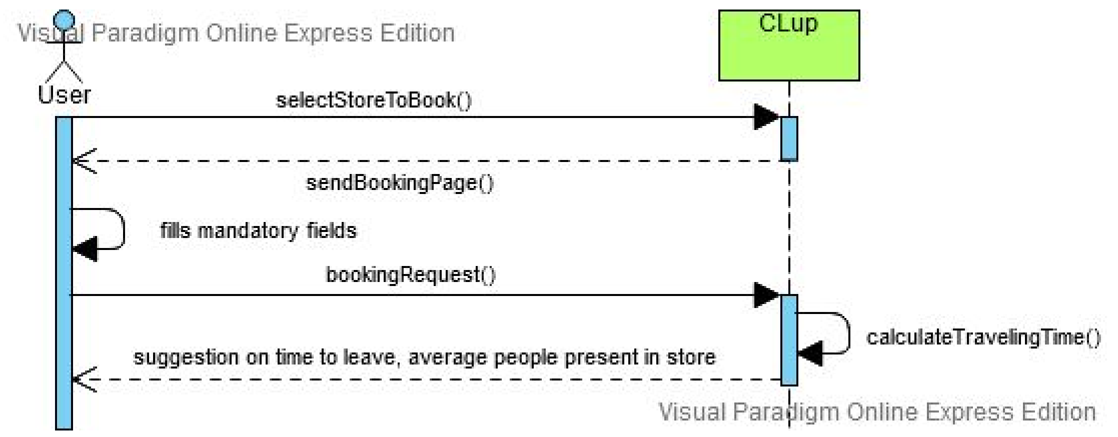
1. **Enter a queue**


Visual Paradigm Online Express Edition
User
CLup
enterQueue()
sendDiscoveryPage()
sendChosenStore(Store, maxPeople)
alt
availableQueue = true
accept notification
Visual Paradigm Online Express Edition

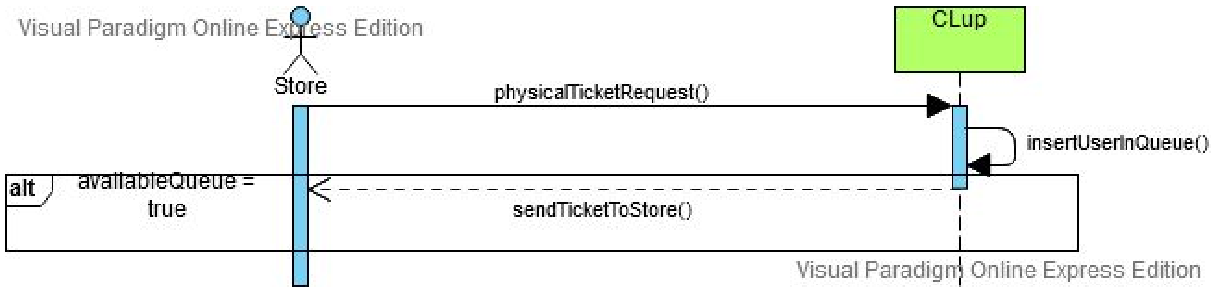
2. **Registration of store**



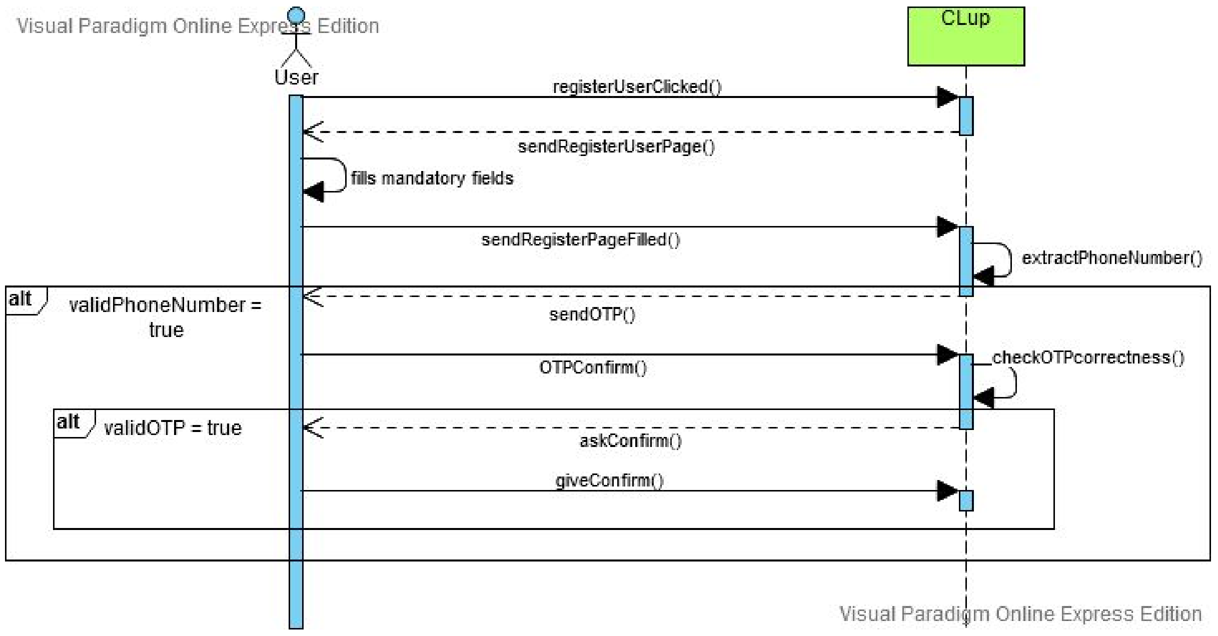
## 3. Book a visit


Visual Paradigm Online Express Edition
User
CLup
selectStoreToBook()
sendBookingPage()
fills mandatory fields
bookingRequest()
calculateTravelingTime()
suggestion on time to leave, average people present in store
Visual Paradigm Online Express Edition

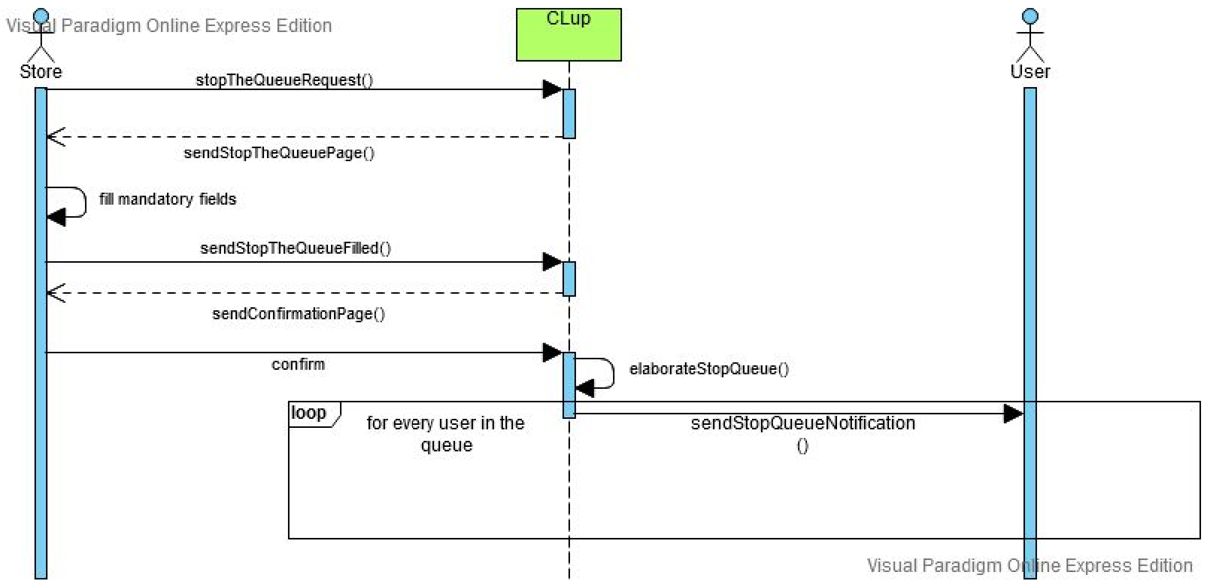
## 4. Send available tickets to print



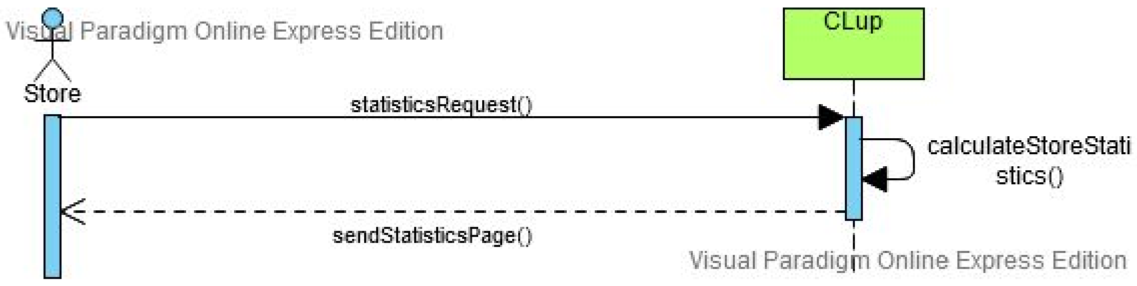
## 5. Registration of user



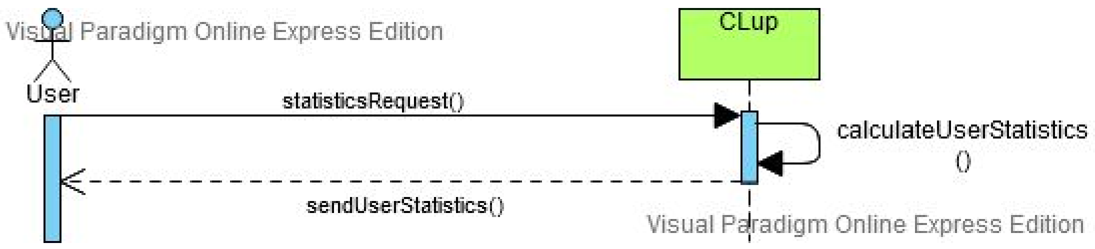
## 6. Stop the queue



## 7. Show Statistics


Visual Paradigm Online Express Edition
Store
CLup
statisticsRequest()
calculateStoreStatistics()
sendStatisticsPage()
Visual Paradigm Online Express Edition

## 8. See chronology of visited stores


Visual Paradigm Online Express Edition
User
CLup
statisticsRequest()
calculateUserStatistics ()
sendUserStatistics()
Visual Paradigm Online Express Edition

### 3.2.5 Scenarios

#### Scenario 1: ATTEMPT TO FILL UP THE STORE

Antonio wants to make a joke at his town's supermarket and to do so he tries to book for 50 people to enter the store in a particular hour, in that way he would be alone in the store at that time. So he opens the app and searches for his supermarket, but he finds out he can only book for a max of 5 people in the same booking. So he tries to book again, in the same hour, but the system notifies him that he can't book again in the same hour. An arguably funny joke has been denied: both Antonio's criminal record and the supermarket attendants are happy!

#### Scenario 2: COVID CASE IN STORE

Koop, a famous italian store chain, uses the CLup service. One day, during a national lockdown dued to Covid, someone starts feeling sick and faints in the store. After calling an ambulance, the store should have made an alert to people inside, while probably closing the store for some time (clients wouldn't know how much, unless they go outside, action that during lockdown they can't do, just to see the closure time). Instead, the store responsible clicks on Options>Queue>Stop the queue, and CLup manages everything itself, by notifying people that were booked with the closure time (and suggesting alternative time slots).

#### Scenario 3: OLD PEOPLE ACCESS

While for youngsters and many adults using an app is an easy-to-do thing, for many of our dear aged ones it could not be the same. For this reason, the only way to access CLup is a phone number, something that nearly everybody has; in this way, the registration can maybe be made by someone younger and close to the aged person, while all he then has to do is click on the store he wants and then read the waiting time. If that person is somehow unable to do such things, like if he has a phone which can't connect to the internet, or he can't read: all he has to do is go to the store; there, CLup will do the job by inserting him in a queue, thanks to the store acting as an intermediate.

#### Scenario 4: NEED TO REORGANIZE STORE DEPARTMENTS

Carrefive, a famous french store chain, noticed that during the lockdown inside its stores some departments were cause of gathering, while some others were not. It tried to investigate, but without support it was really difficult to understand where people go while inside, and the inspection was made in a few stores. But thanks to CLup support, everything is available in its Show Statistics page, where the store can see the most visited departments, and try to better reorganize its placement.

### Scenario 5: INCOMING CHRISTMAS SHOPPING

What happens when December comes? Stores are flooded with people. This can end up in gatherings, and gatherings usually make the government really upset: this can be the cause for further restrictions. Imagine a national lockdown that allowed 50 people inside a store at the same time; the next day comes an order of only letting 30 people in. What should a store do? Thanks to CLup, just modify a number. After setting the maxQueue, a notification is sent to people who were in the queue (in a place which was > 30) and the store has solved everything. And if it was changed again from 30 to 50 people? After changing maxQueue, unexpected gatherings in particular departments could happen, but thanks to Show Statistics, the store can keep an eye on that to -in case- change its arrangement.

### Scenario 6: AN UNEXPECTED SUGGESTION

Adolfo has always gone to Pence, a famous german store chain, because it was the best tradeoff between distance and convenience. But during the Covid pandemic, Pence is usually full of people, and also its prices have risen a lot, so it’s not really convenient to go there anymore. So what should he do? Maybe search on Google for stores close to him, but he would not know the waiting time, and it could happen that he would lose all his afternoon by standing in front of the entrance. Thanks to CLup, not only he can search for stores next to him, but he can see the waiting time, and, even in the case he does not search for them, periodic suggestions come to him in the form of notification, telling him the best choice of stores, in particular the more compatible with his day and travelling habits.

### Scenario 7: TRY TO BETTER ORGANIZE HER LIFE

Sigismonda always searches for more precise and easy ways to optimize her life. She finds out that usually, much of her day in the lockdown period is gone by waiting in line outside the store. She could try to see Google services, which offer closure times and levels of gathering, but they are usually mistaken because of the assumption that everybody has the GPS on at any time. Now, thanks to CLup, she can not only access the store the day and time she wants, maybe even in the most visited hours, but she can also do it without waiting a minute, thanks to CLup serving as an intermediate. Now Sigismonda has found some extra time in her day!

## 3.3 Performance Requirements

The system should be able to send a booking or a request to line up to the stores not after 15 seconds since it has been received.

Moreover, it must be able to guarantee the simultaneous connection of 1.000.000 individuals.

## 3.4 Design Constraints - Non-Functional Requirements

### 3.4.1 Standards compliance

The code should follow the requirements contained in this document. Furthermore, its comments should be clear and focused.

### 3.4.2 Hardware limitations

The software application requires a device able to capture the position to book a visit. In alternative the users that don't need to book can use a physical spot to request the ticket.

The stores can only use a computer, even if without localitation system, because their position is fixed at the moment of registration.

Both devices can be able to send data to the system via internet connection.

### 3.4.3 Any other constraint

A user has a limitation of at most 1 request to line up or book in one specific day.

## 3.5 Software System Attributes - Non-Functional Requirements

### 3.5.1 Reliability and availability

The system is available for 23 hours a day. That means that it has 1 hours a day to update its database with the exact data of each booking and to build statistics day by day. In this range of time the system should be up for 98% of time, which means that the occurrence of fault and service recovery should be contained around 7 days for years.

In order to guarantee this time of downtime the system must have an appropriate infrastructure with a backup system, which stores at least the data of booking and of stores, located in different offices that replicates the core services for covering general failure of the main system without losing all the historical data. Appropriate personnel must guarantee adequate recovery time of the hardware that stops working properly. In case the system is not working properly users must be aware about the failure within 15 minutes with a specific message addressed to the devices. An equivalent message should be sent when the system goes back to work properly.

### 3.5.2 Security

The data stored in the system has to be encrypted and also the password of the store has to be hashed before stored. Moreover, in case of password recovery this should never be sent in clear. The system should be protected against intrusion from agents that are not authorized to access it.

### 3.5.3 Maintainability

The system must be written in Java and must guarantee a high level of maintainability. Code must be written with a good standard, with a high level of abstractions without hard-code as well. Code must be written with large use of comments that cover all aspects of code itself. Code must provide a testing routine that covers at least 80% of the entire code, excluding software interface.

### 3.5.4 Portability

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

### 3.5.5 Scalability

Because the new system is very helpful to manage the sanitary emergency and, in the future, to manage the number of clients, many citizens and stores could exploit this service, the system must be scalable without the necessity of reformulating the core part of the software for a rising number of individuals until 8.000.000. In particular the system must guarantee the requested level of reaction time (15 seconds) for 8.000.000 individuals as users.

## 3.6 Additional Specifications

### 3.6.1 Mandatory Fields

At the moment of the registration, a store will have to compile the following mandatory fields:

- City, Province, CAP, State
- Name
- Certification document, which proves the fact of being a real store
- Email
- Chain
- Opening time
- Capacity
- List of the sections

The user, instead:

- Telephone number
- Password
- Name, Surname

### 3.6.2 Types of Item

During the compiling of a booking, the user can choose from a list of item/section predefined by the specific store that can choose between:

- bakery department
- delicatessen department
- games department
- frozen food department
- housewares department
- beverage department
- parapharmacy
- fruit and vegetable department
- fish department
- meat department
- other

### 3.6.3 Types of Query

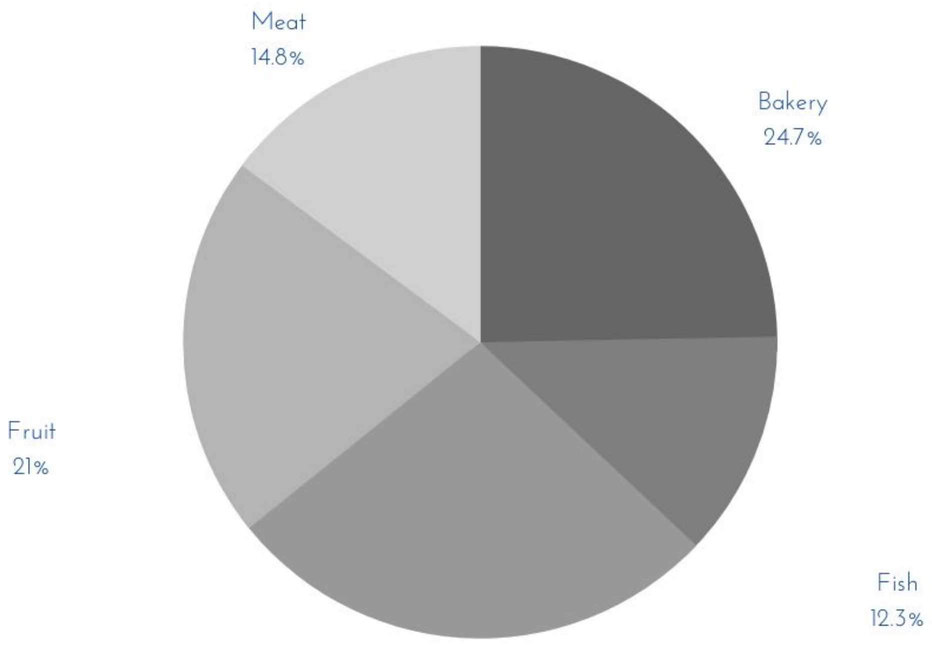
Stores which want to extract information about a specific visitator that can insert his name or his phone number on the dedicated field and, if this user has visited this store in the past, then will receive the list of all the booking and the request of line up of this one. Each store can also build statistics for each user about the duration of the visit and the purchased items and general statistics about all the visitors and on the flux of people in a specific time.

The user can also choose the following type of queries to extract general information:

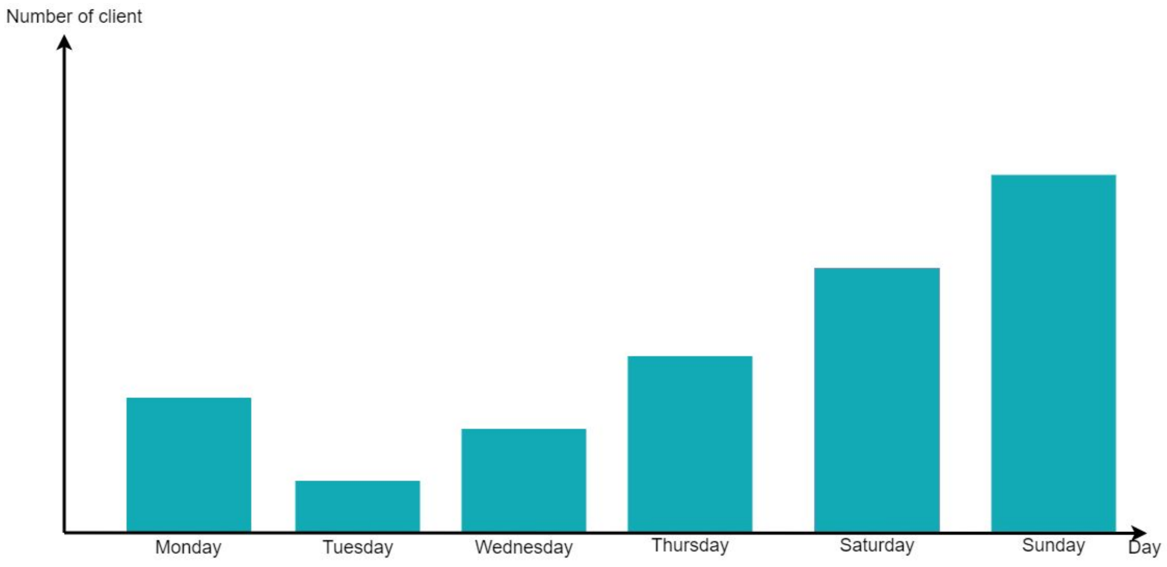
- A map which contains all the stores in the region. In this map the stores can be available to be selected or not if they have stopped the queue.
- A list of stores that contain only the stores that have specific items selected by the user.

The above functionalities provide the user to choose the granularity of the territory between the entire region or one single municipality.

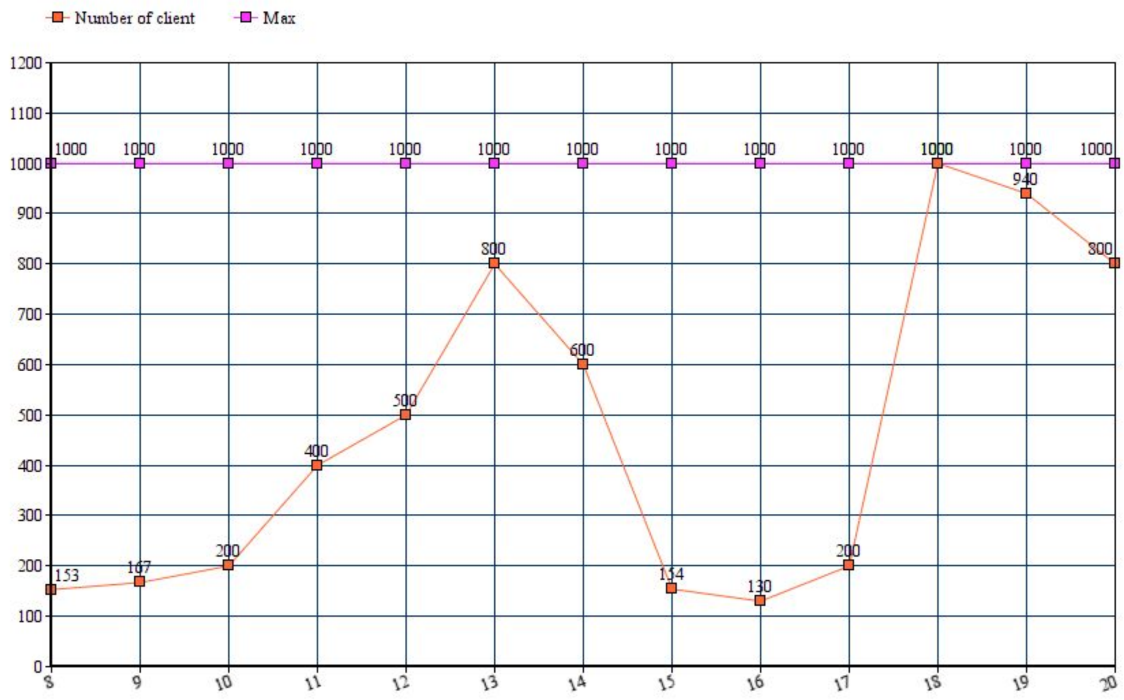
- A pie chart which show the most visited department



- A histogram which shows the weekly trend of number of visit



- A line chart chart which shows the trend of visitors during the day



The above functionalities provide the store to choose the level of granularity. If a store belongs to a specific chain it can see the statistics of other stores of the same chain, or of one specific store. It’s forbidden to see the statistics of other stores that don’t belong to the same chain.

# 4. Formal Analysis Using Alloy

## 4.1 Alloy Code

In this paragraph there will be presented the Alloy code relative to the model we designed.

```
abstract sig Position{}
abstract sig Bool{}
abstract sig Queue{}
abstract sig PrenotationType{
        store : one Store,
        slot : set Slot,
        entranceTime : one Time
}

abstract sig User{
        ticket : lone Ticket
}

one sig True extends Bool{}
one sig False extends Bool{}

sig LinedUp extends PrenotationType{
}

sig physicalUser extends User{}

sig appUser extends User{
        userPosition : one GPSPosition,
        phoneNumber : one  Int,
        password : lone String ,
        chronology: set Booking,
        currPrenotation: lone PrenotationType,
        name : lone String,
        surname: lone String,
        suggestion : set Suggestion
}

sig QRCode{}

sig Ticket {
        user : one User,
        store : one Store,
        qrcode : one QRCode,
        entranceTime : one Time,
        departureTime : one Time,           //time to leave the house
        companions : set User,
        ticketSlot : set Slot
}
{
        #ticketSlot>0
        #companions>=0
}

sig Suggestion {
        store : one Store,
        slots : set Slot,
        accepted : one Bool
}

sig Booking extends PrenotationType{
        user : one User,
        companions  : set physicalUser,
        itemNeeded : set ItemCategory,
        accepted : one Bool
}{
        #companions>=0
        #slot>0
```

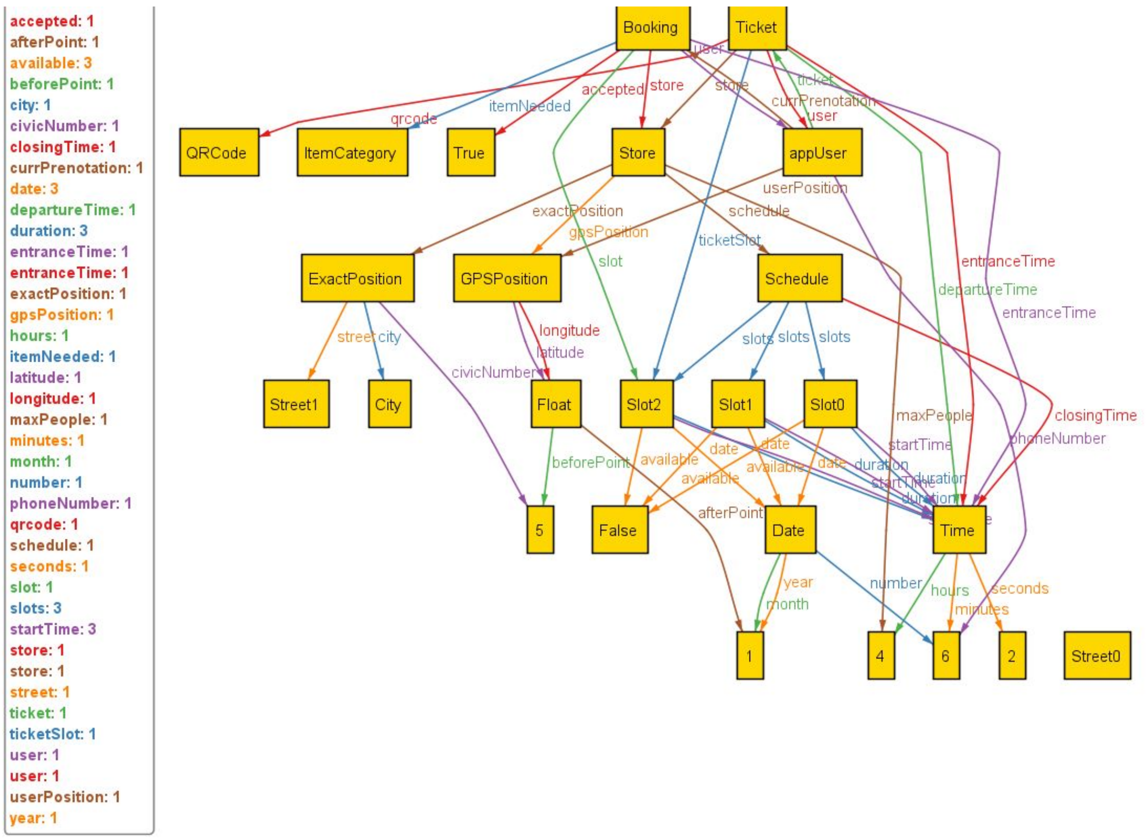
```
}

sig ItemCategory{
        name: lone String
}

sig Schedule {
        slots : set Slot,
        openingTime : set Time,        //opening single day of the week
        closingTime : set Time         //closure time of single day of the week
}

sig Slot {
        startTime : one Time,
        date: one Date,
        duration : one Time,
        available : one Bool,
        numPeople : set User           //booked people for that slot

}

sig Store{
        name : lone String,
        chain : lone String,
        gpsPosition : one GPSPosition,
        exactPosition : one ExactPosition,
        schedule : one Schedule,
        bookings : set Booking,
        itemSold : set ItemCategory,
        maxPeople : one Int
}

sig Street{}

sig City{}

sig Float{
beforePoint: one Int,
afterPoint: one Int
} {
afterPoint>0
}

sig GPSPosition extends Position{
        latitude: one Float,
        longitude: one Float
        }

sig ExactPosition extends Position{
        city: one City,
        street: one Street,
        civicNumber: one Int
}

sig Date{
        number: one Int,
        month: one Int,
        year: one Int
        } {
        number>0
        month>0
        year>0
}

sig Time{
        hours: one Int,
        minutes: one Int,
        seconds: one Int
        } {
        hours>=0
        minutes>=0
        seconds>=0
}

//FACTS
```

```
//Ticket’s user has that ticket
fact {
        all u:User, t:Ticket | u.ticket = t implies t.user = u
}
//All tickets have an unique QRCode
fact ticketAreUnique{
        all disj t1,t2:Ticket | t1.qrcode!=t2.qrcode
}

//If a PrenotationType exist, it is connected to an user
fact allPrenotationHasUser {
        all p:PrenotationType | one user:User | p in user.currPrenotation
}

//Every ticket has only one user
fact allTicketHasUser{
        all t:Ticket | one user:User | t in user.ticket
}

//Every slot is present in a schedule
fact allSlotInSchedule{
        all slot : Slot | one schedule:Schedule | slot in schedule.slots
}

//
fact {
        all s:Slot, t:Ticket, st:Store | (t.ticketSlot = s and st.schedule.slots = s and st= t.store)
        implies
        ((sum t:Ticket | #t.companions)=#s.numPeople)
}

//Every booking in the chronology must have same user
fact sameUserInTheChrono {
        all b : Booking, u: User | b in u.chronology implies u = b.user
}

//Store keeps track of every booking made to him
fact allAcceptedBookingInStore {
        all b: Booking | one s: Store | b in s.bookings
        implies
        b.store = s and b.accepted = True
}

//If an user is either booked or in line the user has a ticket
fact ifYouHavePrenotationYouHaveTicket {
        all t: Ticket | all b: Booking | all l: LinedUp | one u : User |
        (b in u.currPrenotation or l in u.currPrenotation) and u = t.user
        implies
        t in u.ticket
}

//If an user has a ticket he is either booked or in line
fact ifYouHaveTicketYouHavePrenotation {
        all t: Ticket | all b: Booking | all l: LinedUp | one u : User |
        t = u.ticket
        implies
        (b in u.currPrenotation or l in u.currPrenotation) and u = t.user
}

//An accepted Booking must contain correct data consistent with the Ticket
fact bookingCreatesTicket { // 9 fact
        all b: Booking, t: Ticket | (b.accepted = True and( t in b.user.ticket))
        implies
        (b.store = t.store and
        b.slot = t.ticketSlot)
}

//Every user has a distinct unique phone number
fact phoneNumberIsUnique {
        all disj u1,u2 : appUser | u1.phoneNumber!=u2.phoneNumber
}

//If a time slot is unavailable, which means its not in opening time range,

//it contains no people in line
fact slotUnavailable {
        all s : Slot | (s.available = False) implies (#s.numPeople=0)
}

//Every user can accept only one suggestion
fact onlyOneSuggestionAccepted {
        all user :appUser, s1,s2:Suggestion |
        (s1 in user.suggestion and s2 in user.suggestion and s1.accepted=True and s1!=s2)
        implies
        s2.accepted=False
}

//An user can only have one ticket at a time
fact oneTicketAtTime {
        all disj t1,t2:Ticket | t1.user!=t2.user
}

//The opening time must be antecedent to closure time
fact openingTimeBeforeClosure{
        all s : Schedule | s.openingTime.hours < s.closingTime.hours
}

//The maximum number of companions is 4 (5 counting the user)
fact maxPrenotation{
        all b:Booking | #b.companions  <6
}

//If user is in line, numVis = 1; if user is booked, he can have numVis >= 1 (and <= 5)
fact RegulateNumCompanions {
        all u: User | u.currPrenotation in Booking
        implies
        (u.ticket.companions=u.currPrenotation.companions)
        else
        u.ticket.companions = none
}

//If booking is accepted, User.CurrBooking == Booking
fact AcceptedBooking{
        all b: Booking |
        b.accepted = True
        implies
        b in b.user.currPrenotation
}

//If booking is refused, it should not appear in store bookings nor in User.currBoking
fact RefusedBooking{
        all b: Booking |
        b.accepted = False
        implies
        (b not in b.user.currPrenotation and b not in b.store.bookings)
}

//Two store should not have same position (unique identifier)
fact NoSamePosition{
        no disj s,z: Store | s.exactPosition = z.exactPosition
}

//Time slots not in opening time should be unavailable
fact availableTimeSlots{
        all s: Store |
        (s.schedule.slots.startTime.hours < s.schedule.openingTime.hours
        implies
        s.schedule.slots.available = False)
        and
        (s.schedule.slots.startTime.hours >= s.schedule.closingTime.hours
        implies
        s.schedule.slots.available = False)
}

//For every slots, number of people should be less than maxPeople
fact noMorePeopleThanLimit{
        all s: Store | #s.schedule.slots.numPeople =< s.maxPeople
}

//PREDICATES

//A general scenario with an user, a store and corresponding ticket and booking
```

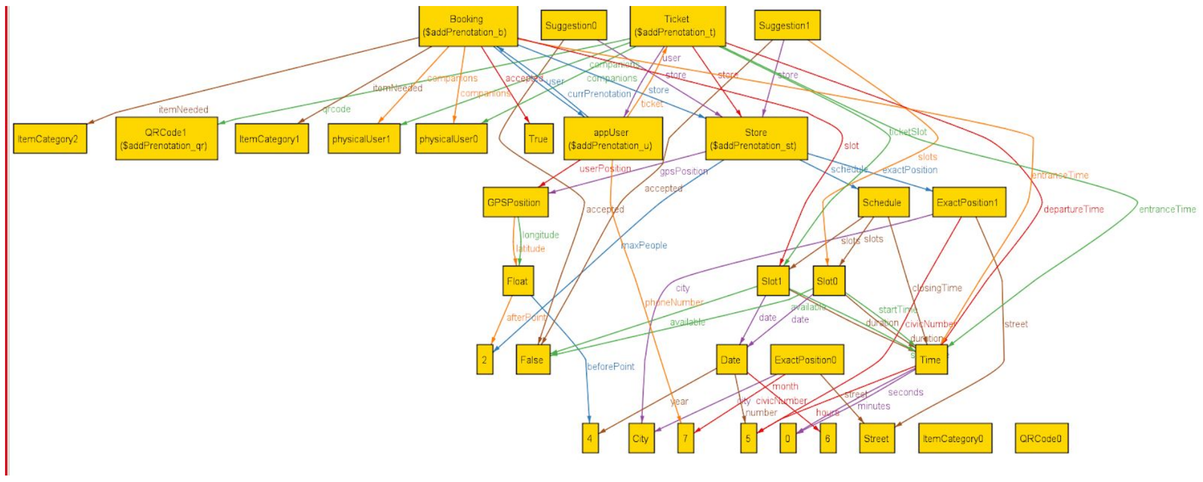
```
pred show {
        #User = 1
        #appUser = 1
        #Store = 1
        #Slot =3
        #Ticket = 1
        #Booking =1
}

//Adding an user to the queue (NOT booking)
pred addLineUp(u:User,st:Store,line: LinedUp,t:Ticket,qr:QRCode){
        u.currPrenotation = line
        u.ticket = t
        t.user = u
        t.store = st
        t.ticketSlot = line.slot
        t.qrcode=qr
        t.entranceTime=line.entranceTime
}

//Add a booking with number of companions > 0
pred addPrenotationWithCompanions(u:User,st:Store,b:Booking,t:Ticket,qr:QRCode){
        u.currPrenotation= b
        b.accepted= True
        u.ticket = t
        t.user = u
        t.store = st
        t.ticketSlot = b.slot
        t.qrcode=qr
        t.entranceTime = b.entranceTime
        #b.companions=2
}

//User accepting a suggestion coming from CLup
pred acceptSuggestion (u:User,t:Ticket,qr:QRCode,s1:Suggestion,s3:Suggestion){
        s3!=s1
        all s2:Suggestion | (s2 in u.suggestion and s2!=s1) implies s2.accepted = False
        s1.accepted=True
        s1 in u.suggestion
        s3 in u.suggestion
        u.currPrenotation.store=s1.store
        u.currPrenotation.slot=s1.slots
        u.currPrenotation.accepted= True
        u.ticket = t
        t.user = u
        t.store = s1.store
        t.ticketSlot = s1.slots
        t.qrcode=qr
        t.entranceTime = u.currPrenotation.entranceTime
}

run acceptSuggestion for 4
```

## 4.2 Metamodel

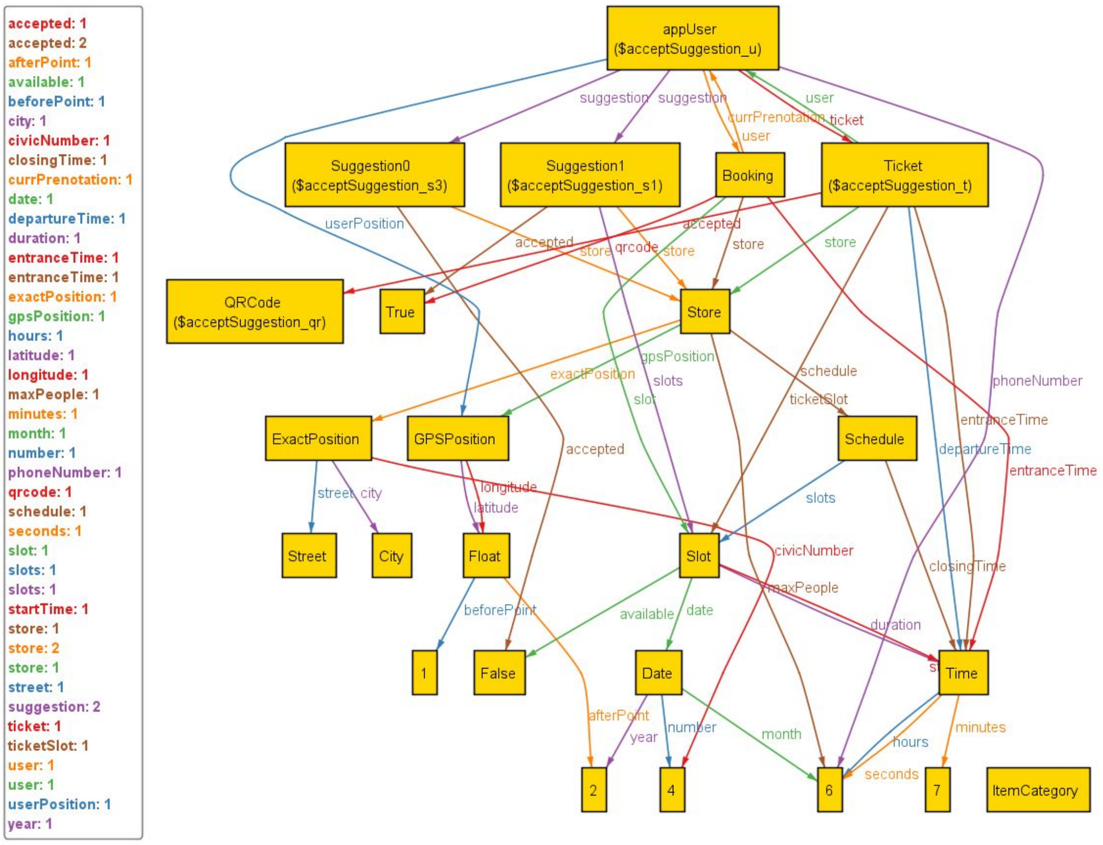
### 1. Show



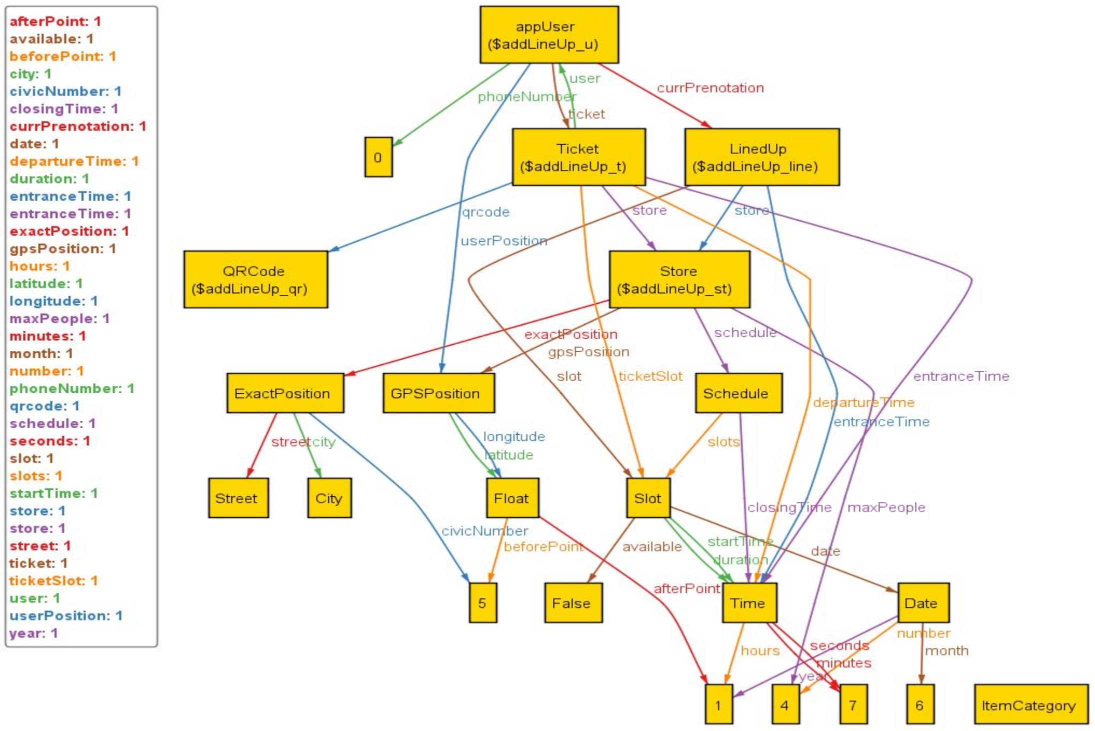
### 2. Add prenotation



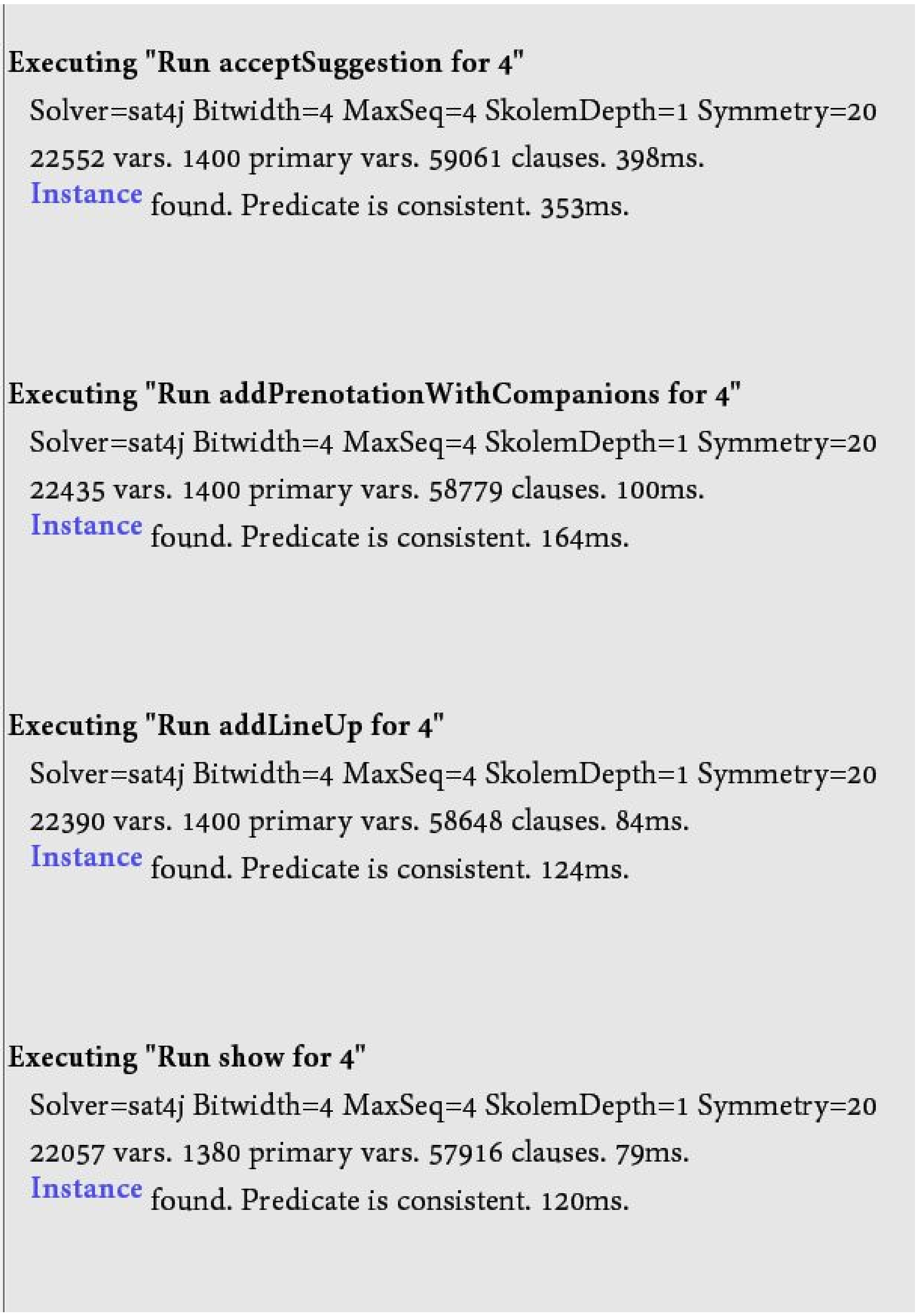
## 3. Accept suggestion



## 4. Add LineUp



## 4.4 Results of predicates



```
Executing "Run acceptSuggestion for 4"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   22552 vars. 1400 primary vars. 59061 clauses. 398ms.
   Instance found. Predicate is consistent. 353ms.

Executing "Run addPrenotationWithCompanions for 4"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   22435 vars. 1400 primary vars. 58779 clauses. 100ms.
   Instance found. Predicate is consistent. 164ms.

Executing "Run addLineUp for 4"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   22390 vars. 1400 primary vars. 58648 clauses. 84ms.
   Instance found. Predicate is consistent. 124ms.

Executing "Run show for 4"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   22057 vars. 1380 primary vars. 57916 clauses. 79ms.
   Instance found. Predicate is consistent. 120ms.
```

## 5. Effort Spent

**Mangano Davide**

| Topic | Hours |
|---|---|
| General Discussion | 2h |
| Use Cases | 4h |
| Build Scenarios | 3h |
| Build Use case Diagram | 2h |
| Build Sequence Diagram | 4h |
| Revision on Use Cases | 1h |
| Alloy Code | 8h |
| Mapping Revision | 2h |
| General Revision | 2h |

**Pagliaroli Antonio**

| Topic | Hours |
|---|---|
| Discussion about introduction | 1h |
| Project setup and part1 | 5h |
| Discussion about part 2 and 3 | 1h |
| Domain assumption and requirement | 4h |
| Mapping | 6.30h |
| Mock up | 3h |
| Section 3.1 3.3 3.4 3.5 3.6 | 4h |
| Alloy | 10h |
| General Revision | 1h |

**Pagliani Filippo**

| Topic | Hours |
|---|---|
| Discussion about introduction | 1h |
| Part1 | 3h |
| Product perspective | 2h |
| Product function | 2h |
| User characteristics | 1.30h |
| State charts and UML | 6h |
| Alloy | 13h |
| General Revision | 1h |

# 6. References

- The state chart was made with https://app.diagrams.net/ [DrawIo]
- The UML Diagram was made with StarUML
- The mockups were made with Strumento di UI/UX design e collaborazione [Adobe XD]
- The Use Case Diagram and Sequence Diagrams were made with Visual Paradigm Online - Suite of Powerful Tools
- The Alloy code development has been supported by Appendix B: Alloy Language Reference
- Torretta