AY 2020/2021

Politecnico di Milano

RASD: Requirement Analysis and Specification Document

Alice Piemonti Luca Pirovano Nicolò Sonnino

Professor
Matteo Rossi

Version 1.2
February 14, 2021

Contents

1 Introduction

CLup (Customers Line-up) is an easy-to-use application which intent is to help grocery shopping to face the big challenges that arise during this tough period.

In fact, due to the recent worldwide spread of SARS-CoV-2 (COVID-19), many countries are imposing lockdowns and strict regulations about social distancing such as: the closure of restaurants in the evening, limitations on public transports, curfews, etc.

In particular, supermarkets are imposed to restrict accesses to their stores, in order to avoid having too much people inside them, which stands at the basis of social distancing. Furthermore, the staggered access needed by stores lead to the same problem in the outside, such as long queues and crowds.

Grocery stores need an application which main intent is to manage in an efficient way customers arrival on the outside of the supermarket, and to count the access of people inside them. The advantage is to regulate the influx of people near the stores, according to the applicable strict rules, and to avoid long hours of waiting, saving people from line up.

1.1 Purpose

The aim of the product is to avoid gatherings outside and inside grocery stores, improving the safety of the customers.

This is achieved through monitoring accesses to the buildings, managing time slots for visits and optimizing people flows inside the stores.

The application should provide two types of accesses for customers (as normal users) and stores' attendants (as special users).

Customers will have the possibility to line up in a virtual queue, so that they can wait from a close and safe building until their number is called. In a reasonable time, the application will inform the user when his number is about to be called, so that he can reach the store in the right time.

In addition, users can book a visit to the supermarket for a different time or day. Hence, the customer indicates the slot preferred and, if it is available, the application register the reservation; otherwise a list of alternatives are

displayed, such as the possibility to book in a different slot, or to chose different stores available for that time/day.

The application can also register in advance the duration of the visit of a customer, and a list of categories that the customer intends to buy, so that the system can suggest the best slot and plan in a finer way the visits, in order to guarantee a correct distance through the aisles inside the store. In fact, the application will be able to balance the presence of people in all the areas of the supermarket, as well as the flows of visitors throughout the day.

Customers will be able to activate an additional functionality: the application will send a notification of available slots in a certain day/time range. Hence, the user will be facilitated in the search of an available slot, and this will guarantee that the user won't spend too much time managing the reservations.

Attendants will have the possibility to scan a QR code (generated by the user's application) at the entrance of the store with a specific functionality offered by the application itself; this helps the attendants to verify the correctness of customers' arrivals and, in the meanwhile, monitor the number of entrances.

In addition, attendants will have access to a proper area of the application that permits to behave as an intermidiate (or proxy) and hand out tickets on spot, in order to guarantee the access to those people who doesn't have access to the application (elderly people, people who don't have a smartphone or an Internet access).

Finally, store administrators can register their shop to the platform, becoming accessible to every user that needs to go buying something at it. They can also edit information about the store, such as opening hours, time slot, etc.

The application will be operable freely, widely available and very intuitive, because the range of users (i.e. people who need to go to the grocery store) extends to the entire population.

The user base is expected to be both people with an Internet access and ones without it, from young people to elderly, thanks to the possibility of attendants to act as a proxy.

In order to use this service a registration is needed. A customer can sign up simply through an email address and a password, or via an external identity provider (such as Facebook, Google, etc.). A store administrator, instead, has to insert all the information about the shop to be added. Finally, an attendant has to insert the personal code by which he (she) is identified

in the internal store system (which can be, for example, the badge number).

The application will define the concept of minimum viable product (MVP) as the first step to reach. The identified MVP is the "Retrieve a ticket" (ASAP) functionality.

The application will then be developed modularly, following this distinction:

- **Module 1 (MVP):** retrieve a ticket functionality, together with web interface and application;
- **Module 2:** module 1 + book a visit functionality;
- **Module 3:** module 2 + custom organization deploy.

1.2 Scope

The product shall be called CLup and will let users to plan their shopping session in two different ways:

- **ASAP**: the user will claim the first available ticket and receive an estimated queue time.
- **Reservation**: the user will choose a day/time slot from a list of available ones, in order to book his visit to the structure.

Every customer can choose one of these modalities **remotely** via a mobile app or through a web browser, or **in presence** by asking to a staff member, who will act as an intermediate between the customer and the system.

When customers make a reservation, the system allows them to choose the duration of their visit and insert a list of possible purchases, in order to optimize the estimate of their stay.

In addition to that, the user can change both the slot and the store relying on system's suggestions: the application will display the best recommendation in order to balance the number of people inside a store throughout the day, merging in the same time slots those people who will purchase different product, in order to guarantee a good distancing through the aisles as well. The user can also enable periodically notifications of available slots in a day/time range: this will allow the application to notify the user when a desirable slot (according to day/time preferences) is available.

1.2.1 Phenomena

According to the paper "The World and the Machine" by M.Jackson and P.Zave, we can identify the application domains. The following table describes the world, shared and the machine phenomena, including the reference to which part controls the phenomena.

Phenomenon	Who controls it?	Is shared?
The formation of queues	W	N
Social distancing	W	N
User registration	M	Y
User login	W	Y
Check username and password	M	N
User retrieves a new number	W	Y
Generate a unique number	M	N
Visualize queue number	W	Y
Check time-out	M	N
Notify number is going to be called	M	Y
Generate a QR code	M	N
Visualize QR code	M	Y
Attendant scans QR code	W	Y
Check QR code validity	M	N
Customer retrieves a number on spot	W	Y
User selects day/time of slot	W	Y
Check availability of slot	M	N
User adds visit duration	W	Y
Check visit duration validity	M	N
User indicates list of items	W	Y
Confirm slot has been reserved	M	Y
Cross data	M	N
Make recommendation of alternative slots	M	N
Visualize suggestions of alternative slots	M	Y
Check customers balance	M	N
Accept receipt of notifications	W	Y
Search periodically for available slots	M	N
Notify for available slots	M	Y

Table 1: phenomena table

1.3 Goals

The main objectives of our system are the following:

- **G1: Allow users to retrieve a ticket**
  This is the main feature of the application, through which customers are invited not staying outside the structure. Through an appropriate estimation of each customer's permanence time, the user receives an estimated queue time after which he can go to the store. The number is guaranteed to be unique. The QR Code is needed in order to let store attendants to monitor accesses, scanning the customer's code upon entering.

- **G2: Allow Store Administrators to add a shop to the list of available ones.**
  The shop can be registered by its manager, who can also add opening hours and GPS Position. Furthermore, attendants can sign up and use the system to check-in people.

- **G3: Allow users to "book a visit"**
  Users can book a slot providing the expected duration of the visit. Moreover, users can optionally provide a list which describes the categories of products they intend to buy. Thank to this, visits are optimized and those customers who belong to the same slot, are supposed to buy different types of product, so that they stay in different areas of the store.

- **G4: Let the system to make suggestions to users**
  For long term users, the system provides an estimation of visit duration, relying on their previous shopping sessions data. Furthermore, during the reservation procedure users will receive a list of alternatives. This help to balance out the number of people in the store during the day. The suggestions could be both on the same shop chain and on different chains.

- **G5: Allow users to receive notifications about the expiration of the queue time**
  Customers are informed when their queue slot booked is becoming free, in order to let they go to the store.

1.4 Definitions, Acronyms, Abbreviations

1.4.1 Definitions

- **Customers:** the people whom this service is directed. They can belong to any age and gender. Their main purpose is to request a ticket to schedule their line up at the shop.
- **User:** it is a customer who registers and then uses the application.
- **Store:** it is the provider of the goods that customers want to buy. It is obliged to limit the number of people accessing its building through a line up method, in order to respect the local laws for COVID-19 pandemic.
- **Attendant:** it is a store's employee. Its aim is both to manage entrances and to help customers. For this reasons attendants can scan QR codes from customers' application and also release tickets on spot acting as intermediate for customers.
- **Store Administrator:** it is the profile of the store's page creator. It can manage opening hours, available slots, people per slot, etc.
- **Queue number:** it is the ID assigned to each user. It's composed of alphanumerical elements, and it's needed to access the desired shop. It can be retrieved online or on the spot.
- **Visit:** it refers to the customers entering the shop, and also to their staying time. It is associated to both a certain store and a visit slot.
- **Slot:** it is a day/time range. It can be reserved by a limit number of users in order to guarantee a maximum number of people who are inside a store in every time of the day.
- **QR Code:** it's a graphic representation of a string, which can be easily read through barcode readers. It contains the visual representation of the queue number, in order to let attendants make a customer "check-in".
- **Notification:** it's an alert that a certain event occurred. This alert can be a "Push Notification" on the smartphone, an SMS, an email and so on.

- **Push notification:** it's an automated message sent by an application to a user when the application is not running.
- **Actor:** user, attendant or store administrator.
- **Man in the Middle:** it is a cyber attack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other.
- **Identity Provider:** it is a system entity that creates, maintains, and manages identity information for principals and also provides authentication services to relying applications within a federation or distributed network.

1.4.2 Acronyms

- **ASAP:** As Soon As Possible. It refers to the possibility of getting an appointment on the first available slot.
- **S2B:** Software to Be, it is the one designed in this document and not yet implemented.
- **MVP:** Minimum Viable Product, it is a version of a product with just enough features to be usable by early customers who can then provide feedback for future product development.
- **IdP:** Identity Provider, see definition above.
- **API:** Application Programming Interface, it indicates on demand procedure which supply a specific task.
- **DBMS:** Data Base Management System, it is an interface between the end user and the database, simultaneously managing the data, the database engine, and the database schema in order to facilitate the organization and manipulation of data.
- **UML:** Universal Modelling Language, it is a way of designing application based on its graphic class representation.
- **BPMN:** Business Process Model and Notation, it is a graphical representation for specifying business processes in a business process model.

- **SSL:** Secure Socket Layer, it is a secure protocol developed for sending information securely over the Internet.

1.4.3 Abbreviations

- **ID:** identifier. It's a generally unique sequence of numbers or letters in order to unambiguously identify an entity.
- **Gn:** goal number n
- **Dn:** domain assumption number n
- **Rn:** requirement number n

1.5 Revision History

- December 10, 2020: version 1.0 (first release)
- December 31, 2020: version 1.1
  - MVP concepts;
  - activity diagrams instead of sequence ones;
  - new mockups to improve user experience;
  - typo fixing.
- February 14, 2021: version 1.2
  - fixed a little incoherence between the RASD and the real use case
  - typo fixing;

1.6 Reference Documents

- Specification document: "R&DD Assignment A.Y. 2020-2021"
- Alloy official documentation: https://alloytools.org/documentation.html
- Paper: "Jackson and Zave: the world and the machine"
- UML official specification https://www.omg.org/spec/UML/
- BPMN official specification https://www.omg.org/spec/BPMN/2.0/

1.7 Document Structure

- **Section 1: Introduction**
  This section offers a brief description of the problem and required functionalities.
  It also contains the list of definitions, acronyms and abbreviations that could be found in this document.
  Finally, there are changelog of the document, containing the revisions list and their content, and document structure, which describes the main purposes of the sections of this document.

- **Section 2: Overall Description**
  This section offers a summary description about the overall organization of the system, the Hardware and Software constraints and the interfaces needed to get it work.
  It also contains a description of all the features offered by the application, and of the actors who use it.

- **Section 3: Specific Requirements**
  This section contains several visual mockups in order to explain the interfaces listed in Section 2. It also contains a description of functional requirement through some scenarios, use cases and diagrams.

- **Section 4: Formal Analysis through Alloy**

2 Overall Description

2.1 Perspective

CLup is a complete system that offers the functionalities described in the specific *Product Functions (2.2)* section.

It takes advantage of some useful integrations, such as external IdP, API interfaces for some features, and so on. It also exploits some pre-existent graphic packages in order to make the user experience more comfortable.

CLup provides all the module that are necessary for its execution. However, it must exploit external interfaces to accomplish its requirements.

We can see a brief report of the information stored by the application through the class diagram of figure 1.

2.1.1 User Interfaces

The system should interface with users through devices which must be connected to the Internet.

Everyone that needs to use this service would connect to it through a Web Interface (from an existent domain, like *www.clup.com*) or through a mobile application that can be installed on smartphones (both IOS and Android).

2.1.2 Software Interfaces

CLup will use some important external interfaces in order to accomplish its functionalities.

The first one is a QRCode generator in order to print a check-in code to customers. The ticket unique ID will be sent to the generator interface, which will return a valid QR representation of it.
The code will then be scanned by store attendants for entrances counting.

The second one is an API provided by a map service owner (like *Google Maps* or *OpenStreetMaps*), which will use the position given by the application to return an interactive map with a marker on that exact position.

Another API service used is the one of the DBMS system, which will be adopted in order to query the database in an efficient way.

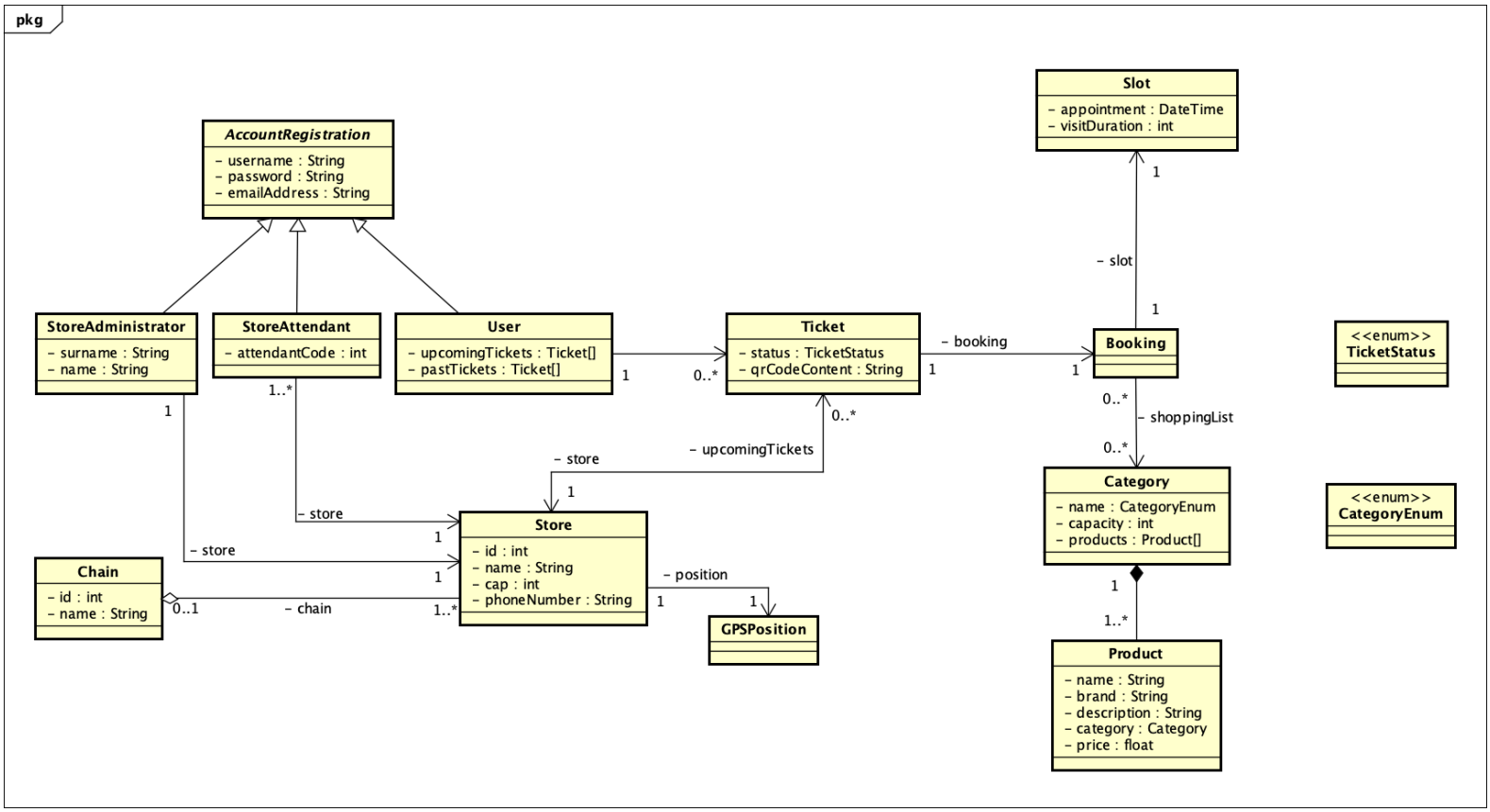


Figure 1: High-level UML diagram with main classes

2.1.3 Hardware Interfaces

The system "as it is" does not provide specific hardware equipment in order to scan QR Codes; this feature will be managed through the mobile application, using smartphones' integrated camera.

2.1.4 Hardware Constraints

Each person interested in using the system described in this document to line up for a shop should have a device connected to the Internet, which could be a smartphone, tablet or personal computer.

The basic requirement is to have an Internet browser installed on the used device. There is then the possibility of downloading and installing the official CLup application from the most important stores, like Apple's *App Store* or Google's *Play Store*.

Shops, instead, should have some devices connected to the service (through web interface or application) in order to monitor queue length, entrances, generate and print tickets on the spot and, if needed, close the booking sessions in advance.

In case of ticket retrieved on the spot, shops must have a printer; it could be a USB one if tickets are generated from a PC, it must be a wireless one otherwise.

2.2 Product Functions

2.2.1 Sign Up

This functionality lets the registration of customers in order to use CLup. The first step is to choose sign up via social accounts or via email and password; if the first one is chosen, the **User** (1.4.1) decides their preferred social platform (Facebook or Google) and they are redirected to the login page of the chosen one and ultimately taken to the home page. On the other hand if the second option is chosen, then the User insert their credentials; if correct an e-mail is sent to the provided one which is required in order to validate it. Finally the User is taken to the login page.

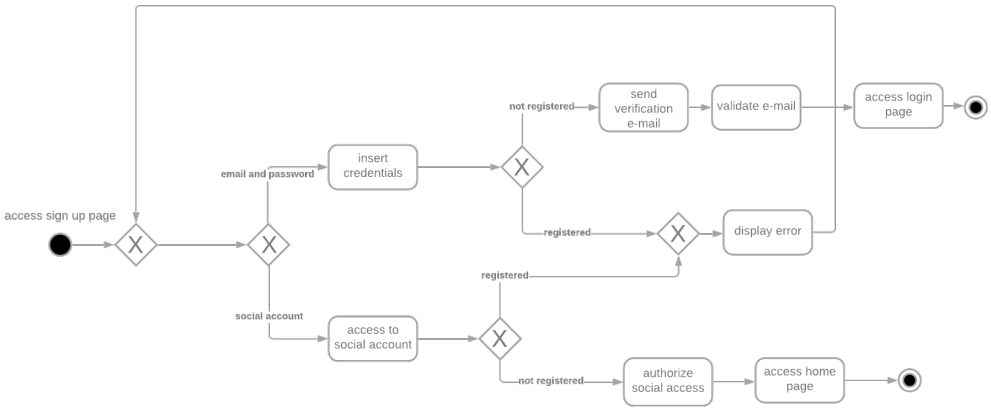


Figure 2: BPMN diagram sign up method

2.2.2 ASAP: As Soon As Possible

This functionality is accessible to all **Users** (1.4.1). The user clicks a button and the system tries to generate the first available ticket for the current day's queue; if the generation fails the system suggests them to book a future visit. Alternatively, the ticket is generated alongside a QR code identifying uniquely users and their queue number. The waiting time is shown to the user until it reaches the last 10 minutes marker alerting them to approach the store.

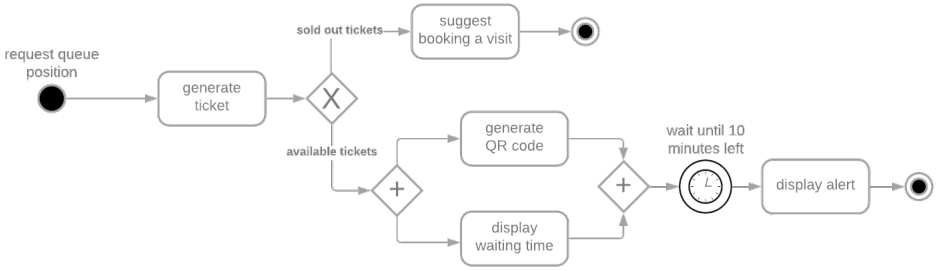


Figure 3: BPMN diagram line-up method

2.2.3 Make a reservation

This functionality is available to every **User** (1.4.1). The user accesses the booking through the menu and the application displays a calendar with all available slots. Alongside this, the system suggests a new booking including various choices: different time slots of the same day, different days, different stores or different chains (if the application is used by different ones). The alternatives are generated through previous data (if exists) or in order to balance the flow to the facilities. After the selection, the reservation is booked and the user is asked to create a shopping list. Finally duration time is generated from previous data, otherwise is inserted by the user.

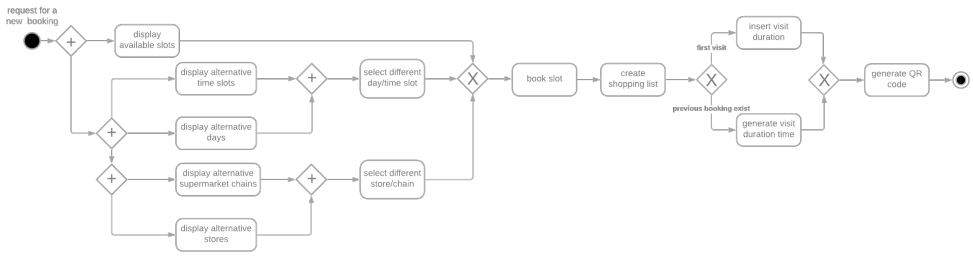


Figure 4: BPMN diagram booking method

2.2.4 Hand out tickets on spot

This functionality is reserved only for staff members. The customer reaches the supermarket/grocery shop and asks the staff to be queued up. The personnel accesses the system which generates and prints a ticket, granting the customer a position in the queue.

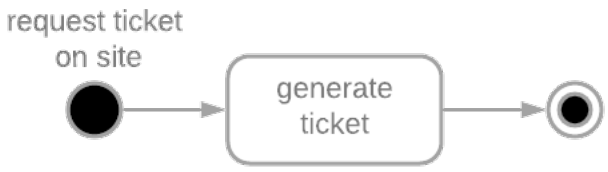


Figure 5: BPMN diagram site method

2.2.5 Periodic notifications

The application occasionally sends notifications to the user with available slots. If the user accepts, they follow the same procedure of the *Make a reservation (2.2.3)*.

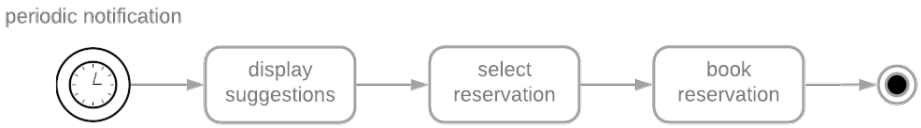


Figure 6: BPMN diagram site method

2.3 Actors

2.3.1 Customer

It's a person with no access to the CLup application. In this case, he (she) goes to the desired shop and requests to be queued up. The store attendant prints the ticket with the position and an expected waiting time.

2.3.2 User

It's a customer, i.e. a person who intend to go to the grocery shop. It is able to use the CLup smartphone application or surf on CLup's website. It uses the service in order to access the stores, according to the regulations. It takes

advantage in using the application since, when it want to access a store, it retrieves the unique number whereby it can stay in a virtual queue and wait from a close and safe building. In addition, thanks to the application, it can either book a reservation for a next moment.

2.3.3 Attendant

It's a worker of a specific store. Its job is to help customers and monitor the entrances of customers. It uses the application for scanning the QR code of users at the entrance of the store. In addition, it can enter in a dedicated section of the application in order to act as an intermediate for those customers who don't have an Internet access, thanks to which it can release tickets on spot. Furthermore, through the application, it can always know the number of customers who are inside the store at any moment.

2.3.4 Store Administrator

It's the administrator of a shop. For the application's purpose only, we identify stores as grocery shops. The store administrator is the person who chooses to adopt the service in order to let attendants managing efficiently the flows of people inside and outside the building, as well as observing the laws on social distancing. It takes advantages through the application since it can knows easily the situation of its stores, such that the upcoming bookings of customers, the influx inside the stores. In addition, the store administrator is the only one who can modify store's information, such that opening hours, slot's capacity, opening hours.

2.4 Assumptions, Dependencies, Constraints

2.4.1 Assumptions

D1: GPS position of the shop is exact and leads to that shop.

D2: The queue number of each ticket is unique.

D3: In case of QR Code entrances monitoring, the attendant must scan the ticket through the mobile application on the smartphone.

D4: Each user who wants to use the online service is needed to have a device connected to Internet (such as PC, Mac, smartphone, etc).

D5: Each attendant knows the store id and correctly registers to it.

D6: The list of products and the duration of the visit inserted are acceptable.

D7: The recommendations are based on the analysis of user's interactions with the application.

D8: The recommendations are such that the number of visits in each store during the day are balanced.

D9: The user does not insert fraudulent information as input (e.g. a wrong visit duration in order to grab the first availabe slot).

D10: Each store has a printer for the handing out tickets on the spot process.

3 Specific Requirements

3.1 Interface Requirements

3.1.1 Customer interfaces

In figure 7 we can see the initial screen of the application. It asks the user to log in or create an account. The user can also sign up with a Google or Facebook account.

In case he decides to create an account, the user must insert his email and a password. Attendants and store managers who want to sign up must go on the dedicated section (clicking on the link in the buttom). In this case, further information must be added.

For example, in the case of a Store Administrators, the system would ask them the details of their store (name, position, etc).
Instead, in the case of Store Attendants, it would ask them the store in which they are actual working, and their personal attendant code. The system will check the veracity of the data in order to avoid wrong registrations (for example, a customer sign up as an attendant).

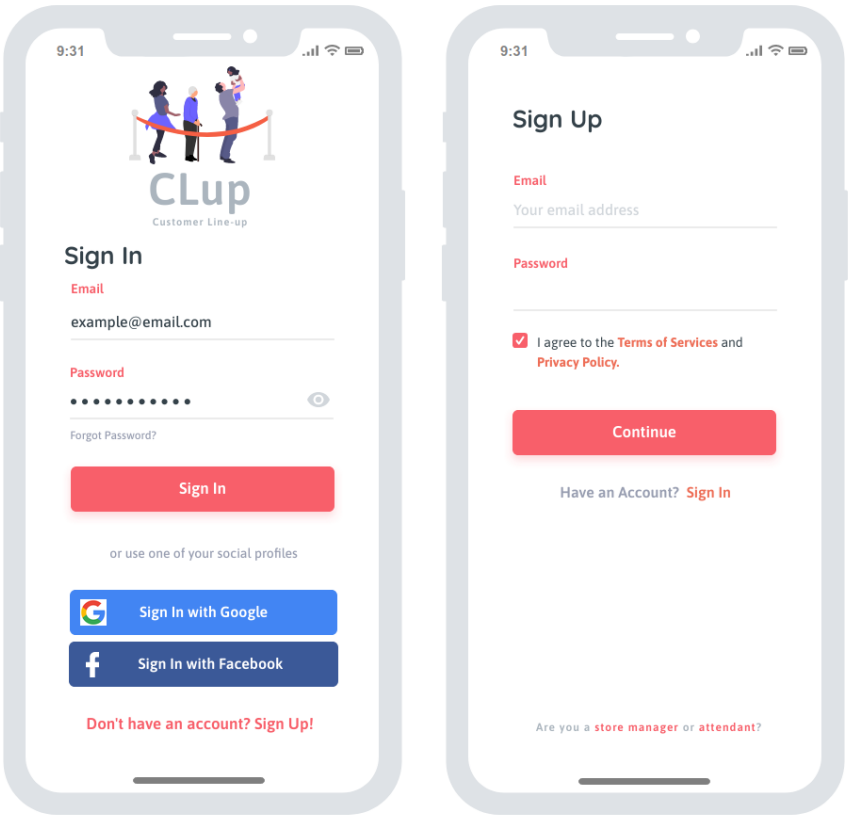


Figure 7: Sign In and Sign Up procedures.

The home page is described in figure 8. As you can see it is simple and intuitive: you can see immediately the list of his active bookings and you can access to the main functionalities with just one tap.
Once here, there is the possibility of checking the details of upcoming bookings or creating new ones.

When you are on a details page, all the information about the selected booking is listed. Through a button, it is possible to cancel that booking (if it is an upcoming one).

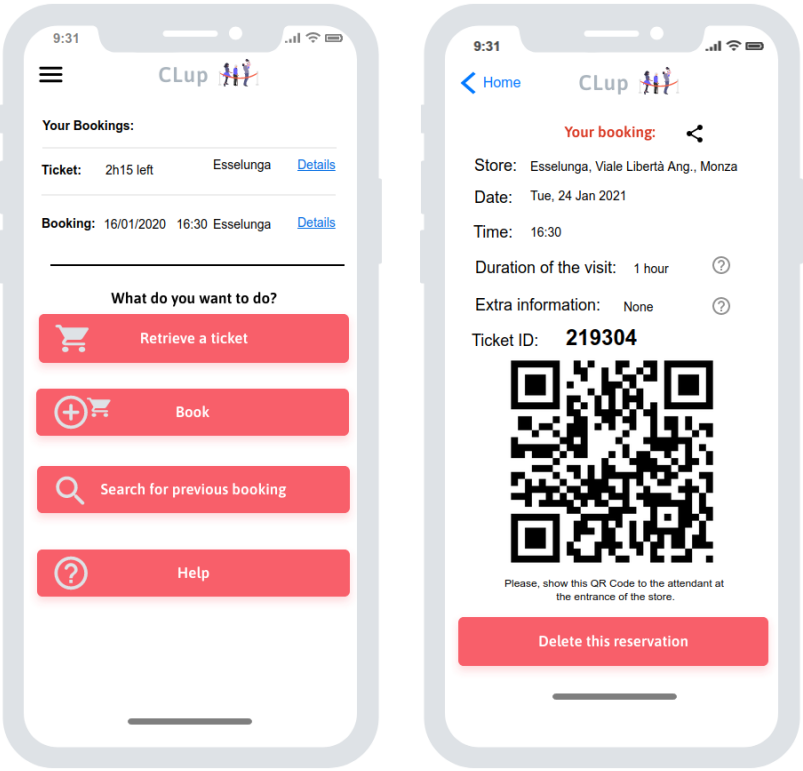


Figure 8: Home page of the application & Booking details page.

Finally, from the home page you can request a new queue number with just a bounch of clicks. As always, the procedure of retrieving a ticket is simple and intuitive, as represented in figure 9. Otherwise, if you use "Book" function (see figure 10) you can book an available slot in a different day and time. Moreover, you can add additional information in order to extend the number of available slots (for example, if you know you want to make a fast visit).

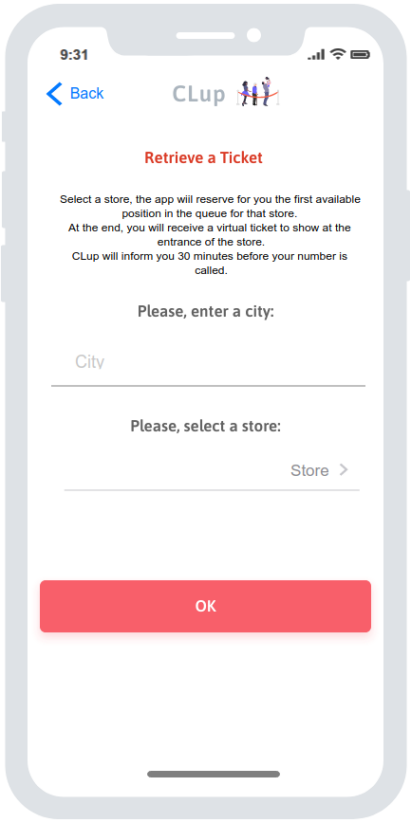


Figure 9: Line-up functionality.

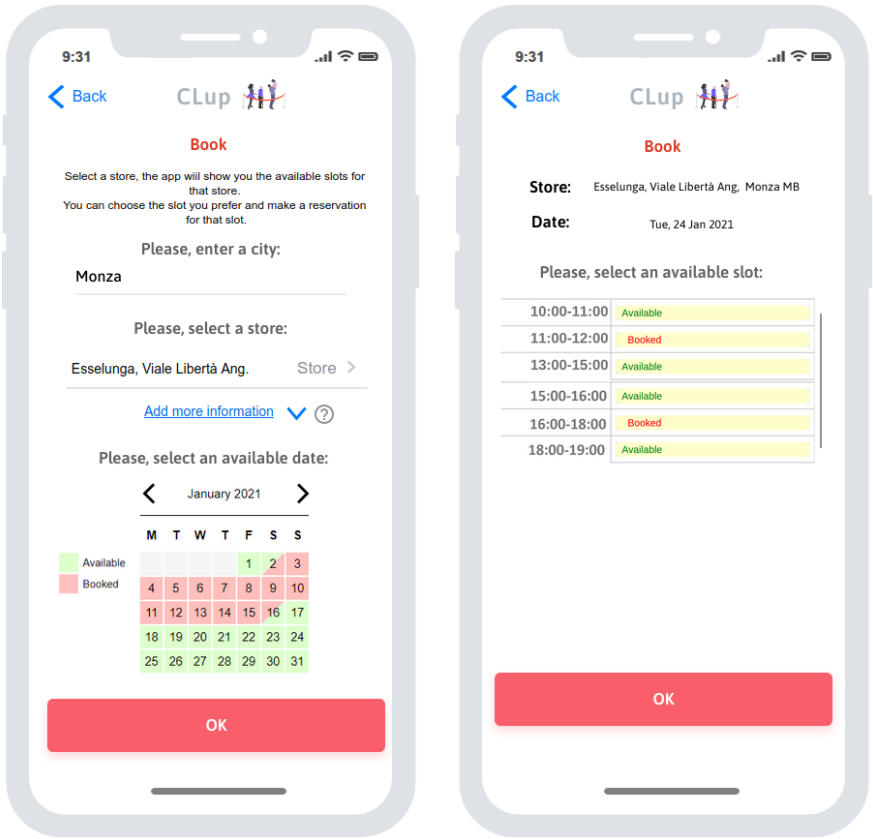


Figure 10: "Book a visit" procedure with unavailable date/time.

3.1.2 Store Administrator interfaces

In figure 11 there is a brief overview of the web interface of Store Administrator. As it can be seen, there are four macro areas, each one with its own features. The store administrator can edit information about its grocery shop, its time slot, can manage the attendants and can also see the list of upcoming booking for the shop.

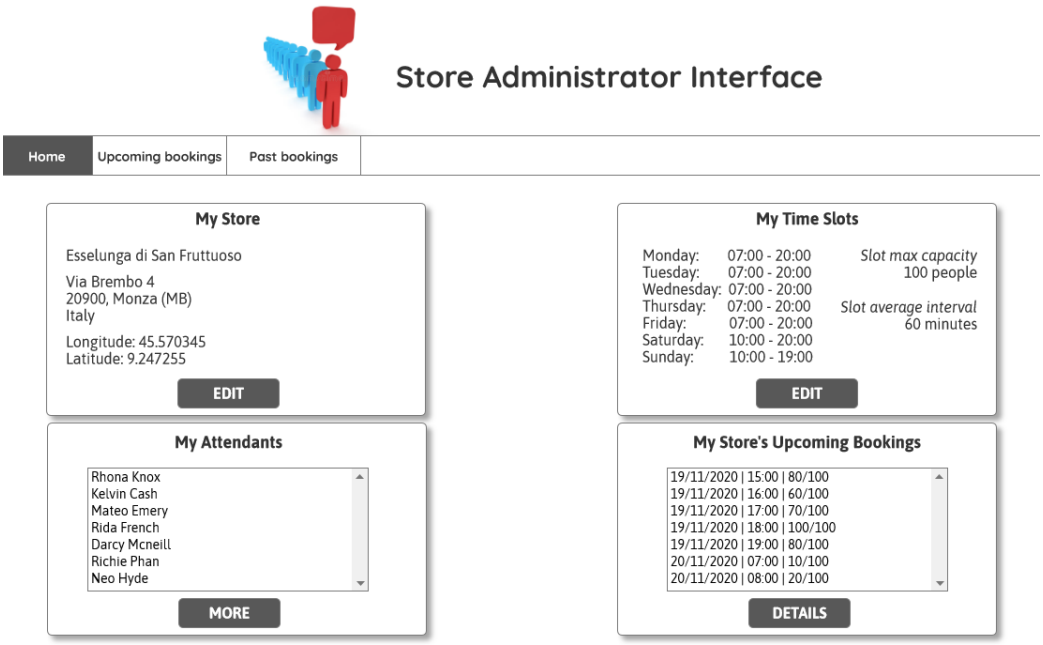


Figure 11: Store administrator home page

3.1.3 Hardware Interfaces

As said before, the application "as it is" does not provide any hardware interface. However, there is the possibility of integration with shops' system and hardware in order to monitor entrances through the given QR code.

3.1.4 Software Interfaces

The software, through appropriate APIs, could manage the reading of the QR codes in order to check their validity and their consistence with actual date and time.

3.2 Functional Requirements

3.2.1 User Scenarios

Scenario 1

Alice is an housewife. She lives with her husband and four children in a small village, not too far from Milan. Due to the Covid-19 spread, the government

of the region decided to impose a maximum of 30 people inside grocery stores simultaneously. Some days before, Alice went to the nearest supermarket and she discovered there was a queue with more than 50 people in-line; it would take more than two hours for entering, so Alice decided to give up. Just yesterday, the same supermarket launched CLup, an application which helps to manage queues remotely. Alice downloaded it on her smartphone as soon as she discovered it: thanks to CLup she can now retrieve a virtual number and wait for her turn from home, without losing time in a real queue outside the supermarket.

Scenario 2

Rodolfo is a middle-aged man who lives in Rome. He has CLup installed on his iPhone since the beginning of the lock-down in Italy. Whenever Rodolfo decides it's time to go shopping for some food, he launches the application, he logs in, and he retrieves his number. This time is 355. Rodolfo knows that the nearest Esselunga is the most popular, so it always takes at least 40 minutes until his number is called. For this reason, he starts to watch the latest episode of his favorite tv series. After 30 minutes, the iPhone vibrates: the application CLup is notifying that in 10 minutes the number 355 will be called. Hence, Rodolfo reaches safely the store (there are just a bunch of people who are entering), then he shows to the attendant the QR code on the iPhone and he calmly goes to buy the necessary.

Scenario 3

A full-time student at Politecnico di Milano decided to stay in his apartment in Milan during the lockdown, although he started to follow all the lectures remotely. He studies a lot and he can't go to the grocery store whenever he wants, due to the huge amounts of lessons. But, a couple of days before, he uses to go on CLup website and book a slot: he logs in and he goes to the booking section. He selects the date and the time he preferred and one of the nearest available stores. Usually, the system doesn't suggest him other combinations (in order to balance people throughout the day) since he usually books for late evening slots, which are very little frequented. He doesn't even have to insert the expected duration time for the visit since

he always stays 30 minutes, so the system autocomplete the field with that value. Then, he selects the list of item's categories he intends to buy: pasta, snacks, beverages, fruits and cleaning. At the end, he confirms everything, and the page displays the QR-code. He clicks on the sharing button and sends it to his smartphone, via Telegram. Once he receives the message, he knows everything worked fine.

Use case diagram

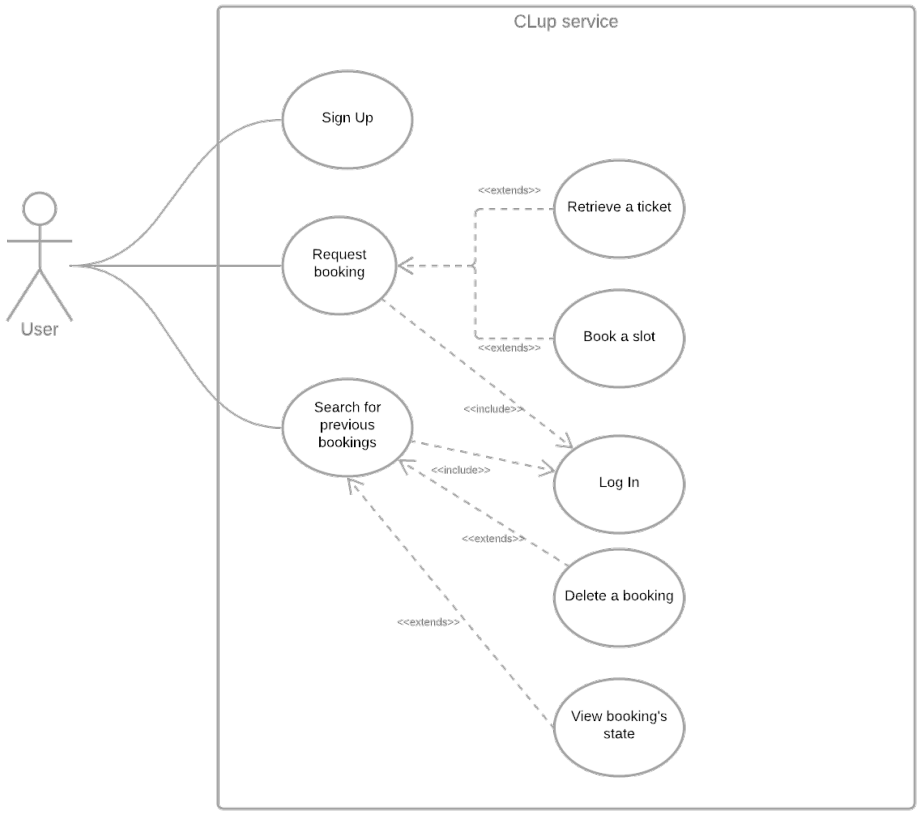


Figure 12: User - Use Case Diagram

Use case tables

ID	1
Name	Sign Up User with email
Actor	User
Entry conditions	• User has downloaded and opened the application on his smartphone
Input	email to use for the registration
Events flow	• The application displays the Sign In screen • User clicks on "Sign Up" • The application displays a list of fields that User must compile: email and password • User inserts the mandatory data and accepts the "Terms of Services" • User clicks on confirm button • The application displays the acceptance of registration and invite User to go on his inbox in order to confirm the registration • User opens his inbox, checks the e-mails and clicks on confirmation link
Exit conditions	User registration has been successful: user data are stored in the database of the system. User can now Login with his credentials.
Output	• The email of the user is stored in the database of the application • The user receives the email of confirmation

Exception 1	User insert an e-mail which is already stored in the database. So, after the user inserts his data and clicks on confirm, the application displays an error page which tells that user is already registered to the service and invites him to login with that e-mail.
Exception 2	User inserts an invalid e-mail. So, after user clicks on confirm button, the application displays the same page and an error message, which suggests to the user to check the e-mail inserted or to change it.

Table 2: Sign Up User

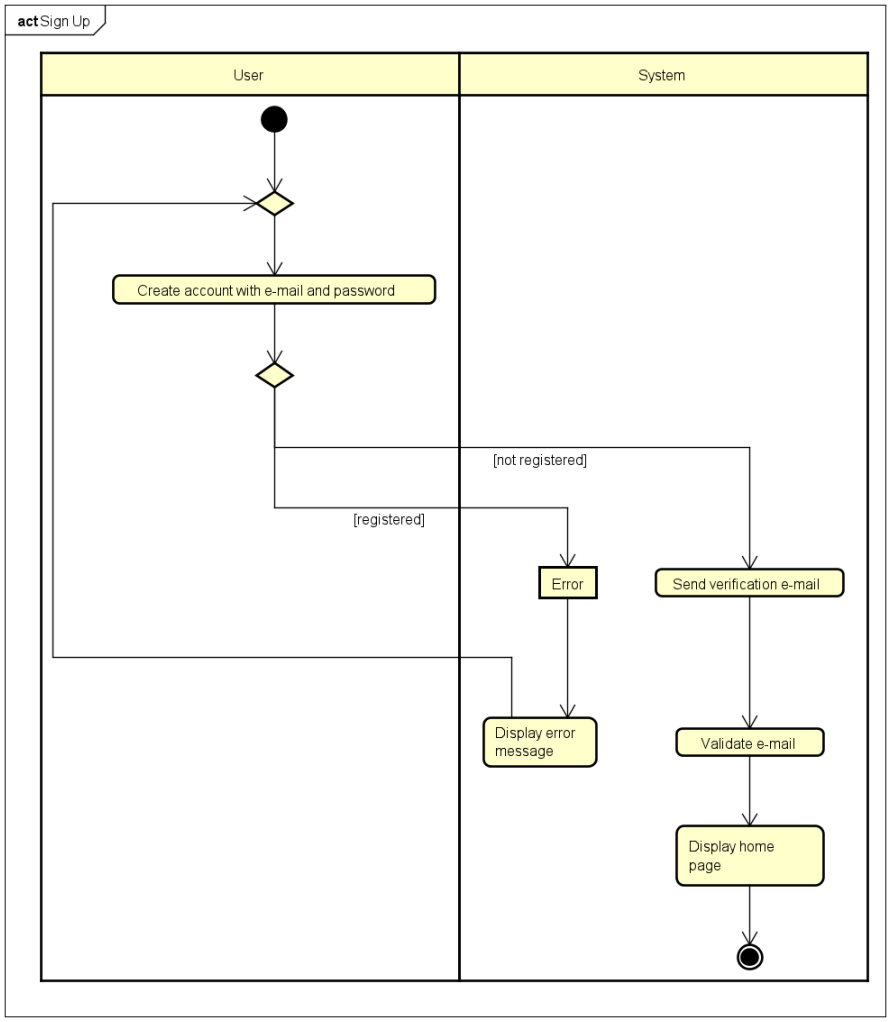


Figure 13: *Sign Up activity diagram*

ID	2
Name	Login User on the application
Actor	User

Entry conditions	• User has downloaded and opened the application on his smartphone • User has registered to the service
Input	User email and password associated to a valid registration
Events flow	• The system displays the Login page • User inserts in apposite fields the credentials for logging in and presses the Login button • The system checks the correctness of the credential inserted • The system displays the home page of the application
Exit condition	User is logged in
Exception 1	User inserts wrong combination of credentials and presses Login button. In this case, the system detects the error and the application displays the Login page with an error.

Table 3: Login User

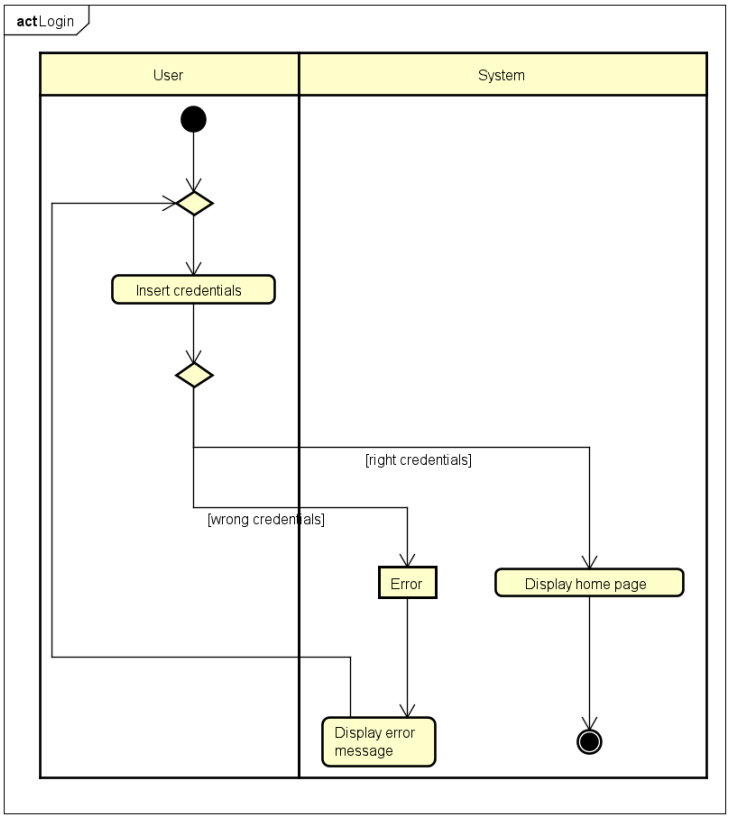


Figure 14: *Login activity diagram*

ID	3
	Retrieve a ticket
Actor	User
Entry conditions	User has logged in

Events flow	• User presses "Retrieve a ticket" button • The application displays a page with a brief description of the functionality and two boxes to fill out (city and store name) • User inserts the city name, selects the preferred store and clicks on continue button • The application displays a confirm popup, with the essential details of the upcoming booking • User presses on confirm button
Exit condition	The application displays the unique ID number of the ticket, the QR-code associated to it and some useful data.
Output	• The system has the information about the ticket generated and its owner • The user sees the ticket on the top of his home page or in the previous bookings section
Exception 1	User presses on confirm button but it has got a ticket in his current bookings, which is still valid. In this case, the application displays an error page which suggests to go on the tickets' section.
Exception 2	User doesn't like the date/time of the first slot available. In this case, it has to press on Cancel button in order to go back to the home page or, in alternative, it can close the application.

Table 4: Retrieve a ticket

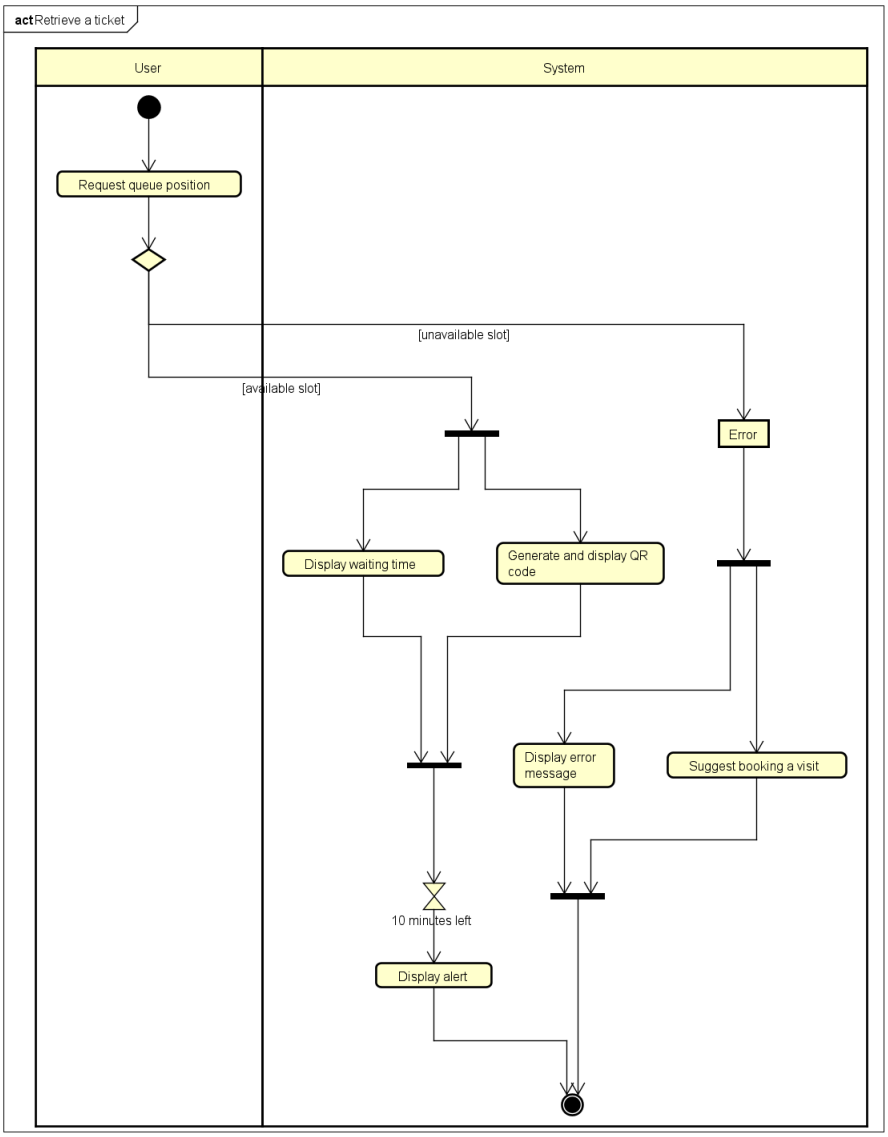


Figure 15: *Retrieve a ticket activity diagram*

ID	4
	Book a visit
Actor	User

Entry conditions	• User has logged in
Events flow	• User presses "Book" button • The application displays a screen with some boxes to fill out (city name, store name, date) • User inserts the city name and selects the store name • The user can add more information: change the expected duration time or add a list of categories he intends to buy • The application update the page with the day slots already available for a certain range (e.g. for the current month or for the next month) for the store selected, based also on the quantity of information added • User selects the day he prefers (from the available ones) • The application displays a screen with the available slots in the day selected • User selects the time slot he prefers (from the available ones) • The application displays a popup with the summary of all the previous selections and all useful information for the upcoming booking • User checks his selections and presses the confirm button • The system adds the booking to the database and generates the unique ID number and the QR code
Exit condition	The application displays the summary page of the booking, with the preferences selected by the user, the ID number and the QR code generated

Output	• The system has the information about the ticket generated and its owner • The user sees the booking on the top of his home page or in the previous bookings section
Exception 1	User has another reservation in the same day. In this case, the application displays an error message and redirect User to the home page.

Table 5: Book a visit

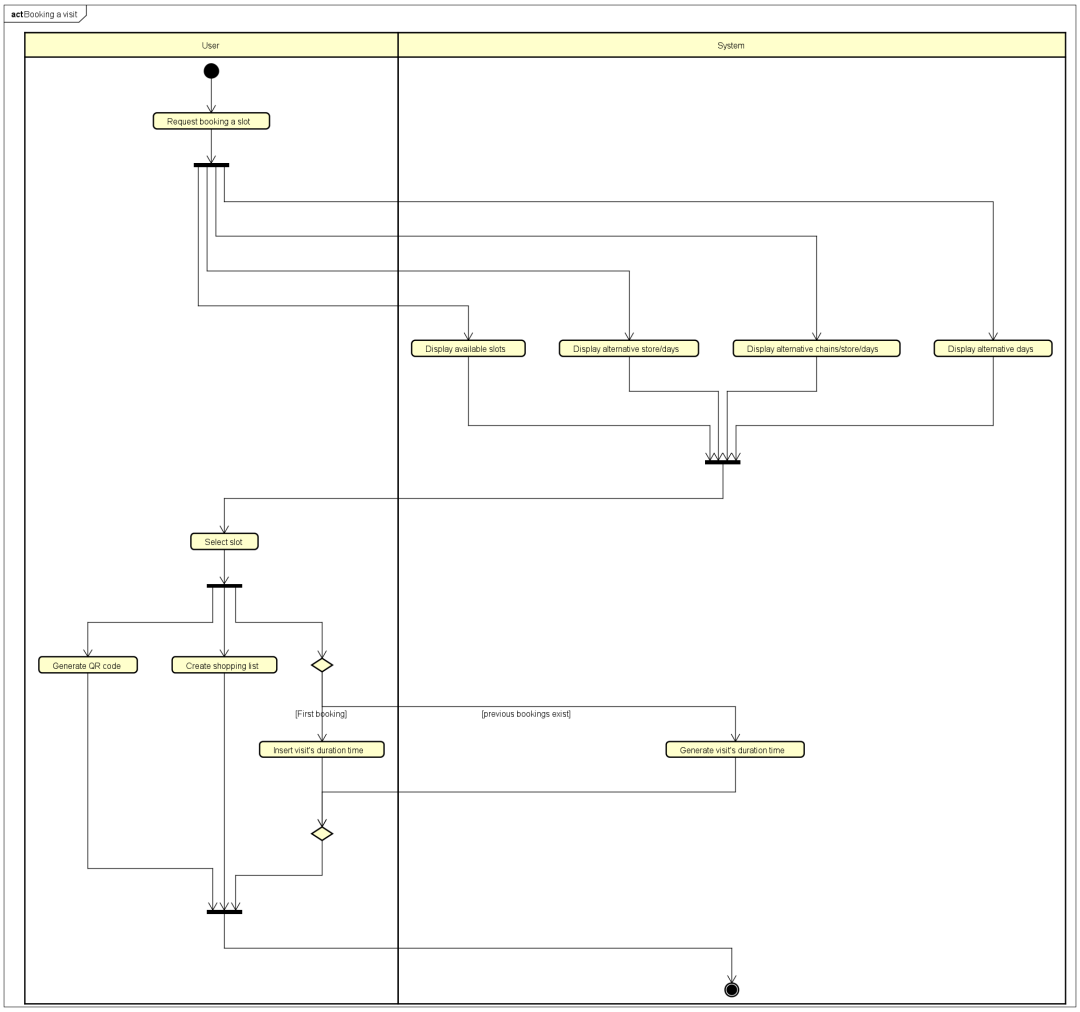


Figure 16: *Booking a visit diagram*

ID	5
	Remove a booking
Actor	User
Entry conditions	• User has logged in • User has at least one upcoming booking
Events flow	• User clicks on a reservation already done • The application answers with the details of the booking selected, such as: booking ID, day and time of the booking, store selected. • User presses the button "Delete this reservation" • The application displays a popup with the notice that the booking is going to be removed permanently • User confirms the removal • The system proceeds with the deletion of the booking from the database
Exit condition	The application notifies that the booking has been removed correctly and it returns to the home page.
Output	• The system has removed the information about the booking removed • The user doesn't see the booking neither on the top of his home page or in the previous bookings section
Exception 1	User wrongly presses the delete button. In this case, the application displays the popup with the notice that the booking is going to be removed and the user presses on "Cancel" button.

Table 6: Remove a booking

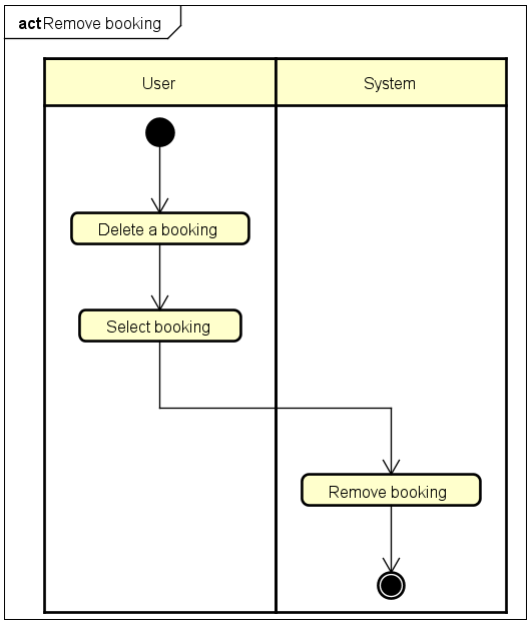


Figure 17: *Remove booking activity diagram*

3.2.2 Attendant Scenarios

Scenario 4

Hassan works at the new Coop located in Marsala street in Monza. Once the store decided to adopt CLup for the management of flows in and out the supermarket, the manager of Hassan assigned him the task of monitoring the main entrance. Therefore, Hassan downloaded the application and registered as a staff member. Hassan's job is to scan the QR code of the customers with his smartphone, and to make sure that everything works fine. The application always displays a counter, so that Hassan can know exactly how many people there are inside the store. However, it does never happen that the counter reaches the maximum value allowed for laws. Hassan is happy for the introduction of CLup because it has got everything needed so that his job became very easy.

Scenario 5

Salvatore is a young boy who lives in a little village near Marsala, in Sicily. The chain of supermarkets where he works as an assistant has introduced a new application for managing queues, in order to avoid gatherings outside its buildings. However, Salvatore knows that there are a lot of elderly people in his village, so how can they use the application without even having a smartphone? Surprisingly, Salvatore came to know that the service has got a function which allows him to act as an intermediate for the aged people who come to his store. His job now is to log in on the application as a staff member and take the credentials of customers who need help. After that, he asks to customers when they want to come again, so that he can reserve them a slot. The application generates the QR code, which he prints and gives to the customer. Salvatore's days are full of work now, because there are a lot of old ladies who need an help, but he knows he must done his job for community's sake.

Use case diagram

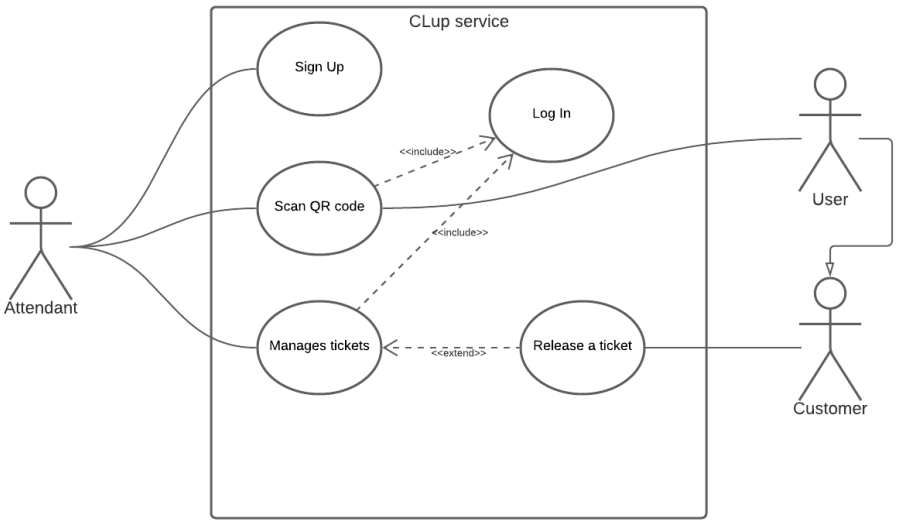


Figure 18: Store Attendant - Use Case Diagram

Use case tables

ID	6
	Scan QR code
Actors	• Store Attendant • User
Entry conditions	• Store Attendant has logged in • User has logged in • User has made a booking
Input	QR Code of the booking
Events flow	• Store Attendant presses on the button "Scan" • The application displays a page with the QR code scanner • User goes on booking details and he shows the QR Code of his booking to the attendant • The attendant frames the QR code • The system reads the QR code • The system checks if the QR Code scanned is associated to an acceptable reservation, according to the actual date, time and location • The application notifies with a popup whether if the QR code is valid or not
Exit condition	The application displays the home page with the total number of people inside the store at that moment

Output	• The system marks the booking as "in use" in his data • In the application booking's details the booking is marked as "in use"
Exception 1	The system cannot read the QR code. In this case, the application displays an error status which invites the attendant to frame the QR code again.
Exception 2	The system cannot read the QR code for 3 times consecutively. In this case, the application displays an error status which invites the attendant to make sure that the QR code is original.
Exception 3	The system reads a QR code with an unacceptable day/-time/store reservation. In this case, the application displays the details of the reservation in order to show where is the problem (e.g. it's too early or too much late or it isn't the correct date/store).

Table 7: Scan QR code

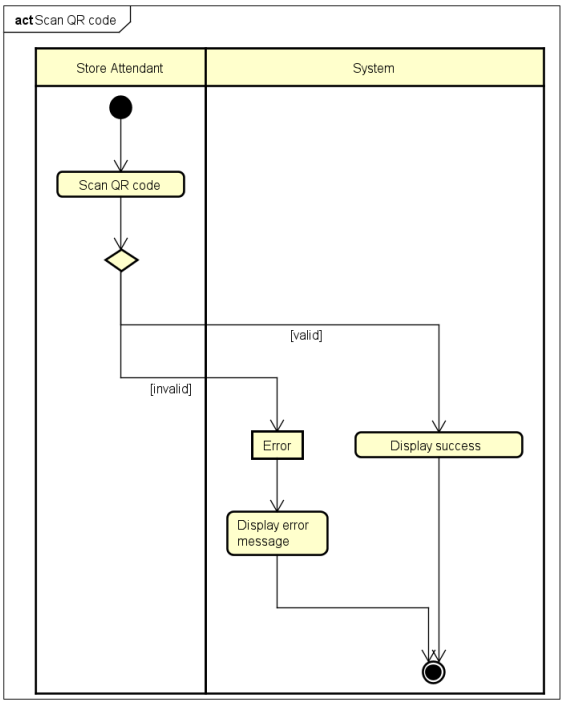


Figure 19: *Scan QR code activity diagram*

ID	7
	Release a ticket
Actor	• Store Attendant • Customer
Entry conditions	• Store Attendant has logged in • The application is connected to a working printer

Input	• Log in data of the store attendant are in the database
Events flow	• Store Attendant presses on the button "Generate a ticket" • The application displays a page with a list of fields to be filled • Customer tells to the attendant the city and the store for which he wants to make a booking • Attendant fills out the fields with customer information • The application updates the page with the available days in the current month, for the store selected • Store attendant select a day from the available ones • The application displays the time slots for the day selected • Store attendant selects an available slot • The application displays the confirmation page with a summary of the data inserted • The attendant checks the data are correct (with the help of customer) and confirms • The system inserts the new ticket in the database and generates the unique ID number and the QR code associated • The application displays a page with the QR code and all useful information of the ticket generated • Store Attendant clicks on "print" icon • The application sends the page to the printer
Exit condition	The application notifies that the pdf is going to be printed and it displays the home page of Store Attendant

Output	• The generated ticket is added to attendant's previous bookings • The system stored all data for that booking • The details page of the booking is printed
Exception 1	There are not slots compatible with customer's desires. In this case, Store Attendant can cancel the procedure and the application returns to the home page.
Exception 2	The printer returns an error. In this case, the application deletes from the database the ticket generated and displays an error message.

Table 8: Release a ticket

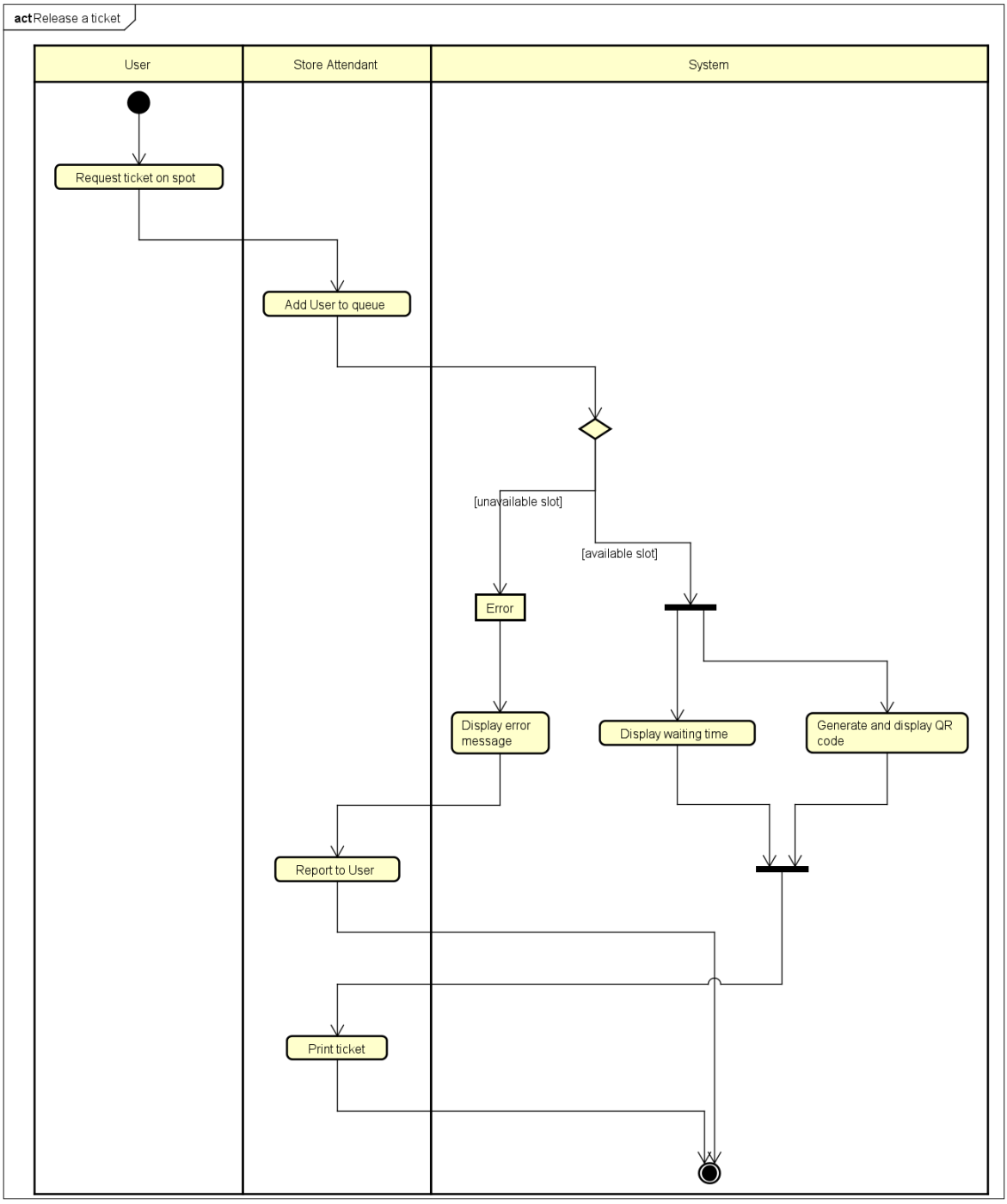


Figure 20: *Release a ticket activity diagram*

ID	8
	Delete a reservation
Actor	Store Attendant
Entry conditions	• Store Attendant has logged in • There is at least one past booking in the previous bookings section
Input	Log in and booking data are stored in the system
Events flow	• Store Attendant presses on the button "delete a reservation" • The application displays a page with the list of all the tickets generated by all the Store Assistant of that store and a search bar • Store Attendant makes a search based on day/time of the booking he wants to delete • The system checks the record in the list and displays the results • Store Attendant selects the record he wants to delete • The system shows the details of the booking selected • Store attendant clicks on "Delete this reservation" • The system deletes the reservation from the database and displays a message of deletion completed
Exit condition	The application returns to the home page
Output	• The booking has been removed from the system • The booking is not visible on store attendant's application anymore

Exception 1	The system doesn't find any record associated to the name and surname inserted. In this case, the application displays an error message.

Table 9: Delete a reservation

3.2.3 Store Administrator Scenarios

Scenario 6

The Esselunga store located in San Fruttuoso (Monza) imposed a maximum limit of 100 people inside the store at the same moment, in order to observe the regulations of government. Due to this, a long queue always forms, from the entrance of the supermarket to the church of Triante's neighborhood. For this reason, Marco, which is the store administrator in action on the territory of Monza and Brianza, decided to adopt CLup in his stores. Thanks to CLup the customers can now take reservations and waiting their turn from home, avoiding the formation of queues and crowds nearby the stores.

Scenario 7

Giannamaria is the manager of the biggest grocery shop in Rome. The store can contain more than a thousand people at the same time, in normal conditions, but due to the coronavirus spread and the consequent laws about social distancing inside buildings, the store must limit the number of entrances at a maximum of 300 people. Giannamaria always thought it would have been difficult to keep track of the people entering and exiting in such a big store, so she immediately decided to adopt CLup. Thanks to the application, she can make use of only a couple of attendants in order to managing the entrances, and she can know exactly how many people there are in the store, at any time, through the Store Administrator section. Moreover, she can easily manage time slots and change timetables with few procedures, through the application. According to Giannamaria, the managing of the store has never been so easy: in fact, she is thinking about to continue to use CLup even after the virus situation has been solved.

Scenario 8

The manager of Eurospin's chain of discount in Italy asked to the store administrators of each area to draw up a document about the more frequented hours of their customers. Months ago, the stores converted to the use of CLup application. Therefore, the store administrators of each area can make use of it in order to write the document and make statistics. In fact, the application has a section, only for the store administrators, where they can see the list of all the upcoming and all the past bookings made in a specific store by the customers, with all the necessary details. So, the store administrators export the list on an Excel file and, starting from that, they can quickly make all the statistics they need to.

Use case diagram

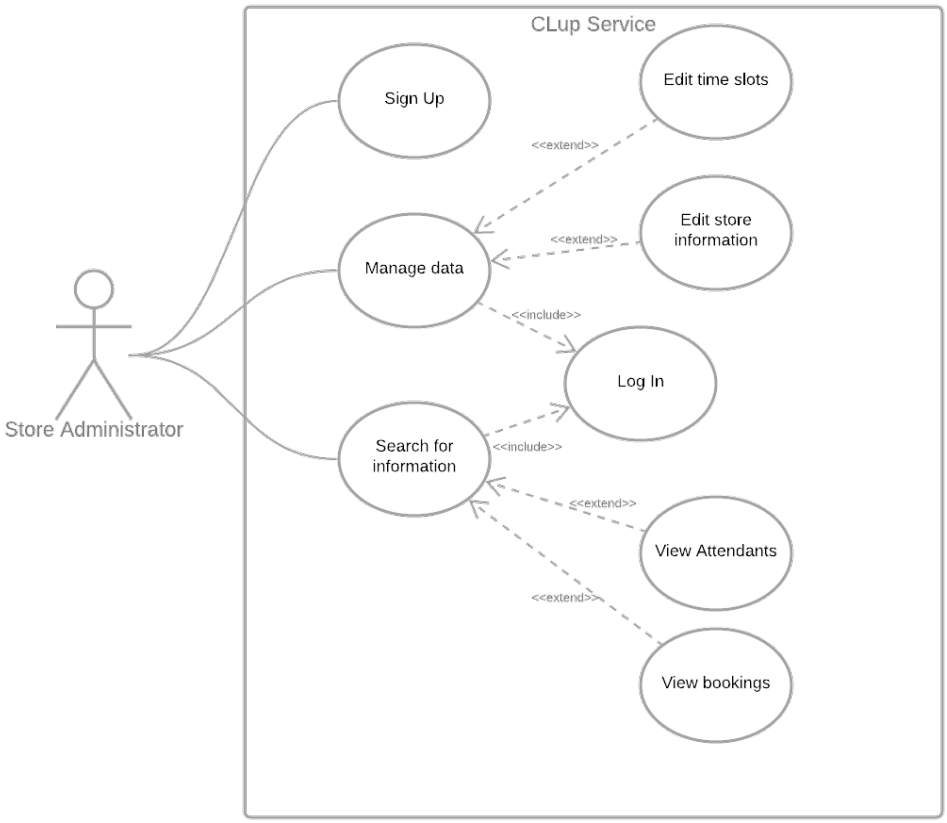


Figure 21: Store Administrator - Use Case Diagram

Use case tables

ID	9
	Modify time slots
Actor	Store Administrator
Entry conditions	• Store Administrator has logged in
Events flow	• Store Administrator goes to "My Time Slots" section and clicks on the edit button • The system extract the data from the database and the application displays the required information, such as: name of the store, location, capacity of the store, time of the slots, opening hours. • Store Administrator makes his modifications, for example extending the opening hours of the store, and it clicks on confirm button • The application checks whether if the changes are acceptable and then it displays a confirm page with the summary of the changes • Store manager confirms again
Exit condition	The application displays a page with the updated information of the store
Exception 1	Store Administrator inserts unacceptable value of time. In this case, the system detects the inconsistency of the data and displays an error message.

Table 10: Modify time slots

ID	10
	View Upcoming Bookings Details
Actor	Store Administrator

Entry conditions	• Store Administrator has logged in
Events flow	• Store Administrator goes to upcoming bookings section and clicks on the "Details" button • The system takes from the database the list of all the reservations made for that store and the application displays the list • Store Administrator clicks on one of the bookings
Exit condition	The application displays a page with the details of the selected booking, such as: ID of the ticket, date and time, list of product's categories.
Exception 1	CLup application cannot reach the database. In this case, an error message is displayed.

Table 11: View Upcoming Bookings Details

3.2.4 Requirements

This subsection summarizes the goals describe in 1.3, showing the requirements and the domain assumptions for each of them.

-G1: Allow users to retrieve a ticket

- **R1:** The system allows registered users to select the store where they want to shop.
- **R2:** The system allows users to retrieve a queue number.
- **R3:** The system generates a QR Code associated to the ticket.
- **D2:** The queue number of each ticket is unique.
- **D4:** Each user who wants to use the online service is needed to have a device connected to Internet (such as PC, Mac, smartphone, etc).

-G2: Allow Store Administrators to add a shop to the list of available ones

- **R4:** The system allows Store Administrators to add GPS Position and opening hours of the store.
- **R5:** The system allows Store Attendants to register with their attendant code.
- **D1:** GPS Position of the shop is exact and leads to that shop.
- **D4:** Each user who wants to use the online service is needed to have a device connected to Internet (such as PC, Mac, smartphone, etc).
- **D6:** Each attendant knows the personal code and registers to the correct shop.

-G3: Allow users to "book a visit"

- **R1:** The system allows registered users to select the store where they want to shop.
- **R3:** The system generates a QR Code associated to the ticket.
- **R6:** The system allows users to select a day/time slot from the available ones.
- **R7:** The system allows users to add a list of products (categories) to purchase and the duration time of the visit.
- **D4:** Each user who wants to use the online service is needed to have a device connected to Internet (such as PC, Mac, smartphone, etc).
- **D7:** The list of products and the duration time inserted are acceptable.

-G4: Let the system to make suggestions to users

- **R8:** The system computes a prediction of expected time duration of a customer's visit.
- **R9:** The system recommends alternative day/time slots or store/chains to a user.
- **D8:** The recommendations are based on the analysis of user's interactions with the application.
- **D9:** The recommendations are such that the number of visits in each store during the day are balanced.

-G5: Allow users to receive notifications about the expiration of queue time

- **R10:** The system takes note about the actual queue number.
- **R11:** The system notifies the user when its queue number is going to be called.

3.2.5 Traceability Matrix

Requirement	Use cases
R1	Request booking
R2	Request booking
R3	Request booking
R4	Manage data
R5	Attendant's sign up
R6	Book a slot
R7	Book a slot
R8	Book a slot
R9	Book a slot
R10	Request booking
R11	View booking's state

Table 12: Traceability matrix schema.

3.3 Performance Requirements

Of course, the system should have a good response time, which could be included between 0.1 and 2 seconds. Otherwise, customers and store workers may think that the service is interrupted or does not work.

The average workload of the system is expected to be very high, because of its principal functionality. However, it depends on the number of stores that would adopt it. With a rough calculation, the system must guarantee at least 1 millions of operations per day.

Through a good distribution of the application it is possible to accomplish this specific requirement.

The module 3 of the application provides the installation and deploymeny of CLup on the organization's server, in order to customize the user experience and avoid service downtime in case of CLup's servers problems.

3.4 Design constraints

3.4.1 Hardware Constraints

On stores side, each attendant is required to have a device from which it is possible to check-in customers by scanning their QR Code. For this functionality stores could have a Barcode Reader connected through usb to a personal computer or, otherwise, they could use a mobile application connected to their system.

Each customer, instead, must have a mobile device on which grab a ticket or make a reservation.

In addition to that, each store must have a printer in order to accomplish to the "Grab on spot" feature. If the store contains a totem with a computer it suffices to be a USB printer. Otherwise, if the tickets are generated from the attendant at the entrance, the store must have a Wi-Fi printer that connects to the attendant mobile device.

3.4.2 Privacy Constraint

The system, since the main application area is the european one, must be compliant to EU's GDPR law. Furthermore, the data must pass through a secure connection, in order to prevent *Man in the Middle* attacks.

In case of authentication with an identity provider different from the application one, user's data will be treated from an external company, not associated with CLup. In that case, the privacy policy of the Identity Provider is applied.

When a customer, store attendant or store manager registers to the application, the privacy policy must be read and accepter. Otherwise, the user will not be able to use the service.

3.5 Software System Attributes

3.5.1 Easy usability

The system must be very easy to use, because the user base target is various.

In fact, anyone needs to go to grocery shop, from the teenager to the elderly man or woman.

The application should then be accessible to anyone in this age range. Of course, in case of people who can not use this system, there is the possibility

of retrieving tickets on the spot.

3.5.2 Reliability

The system must prevent downtime, in order to let people going to the grocery shop in the opening hours.
The highest number of simultaneous accesses is expected in the most frequented shopping hours (early morning or late afternoon).

3.5.3 Availability

The system must be available as much as possible, with a minimum value of 96% of time.

3.5.4 Security

Communication between parties are encrypted and goes on a secure channel (through SSL protocol).

Furthermore, the database guarantees that performed operations are always authorized (*e.g.. a customer cannot modify information of a store*).

3.5.5 Cross Platform

The system must be supported on the principal operating systems, either mobile and computers. It is in any case usable on every web browser, from every device.

The downloadable application, instead, must be supported by IOS and Android.

3.5.6 Maintainability

The system must be designed in such a way that permits future addition of functionalities with minimum effort. Design techniques must in fact guarantee an high reusability.

3.5.7 Modularity

The system is designed to be modular, as described in the purpose. The most important module is the "Retrieve a ticket" one, which permits to each shop to implement a low cost queue system.

In case of more complex needs, there are modules 2 and 3, which provides the "Book a visit" functionality and the "Custom deploymeent" service respectively.

4 Formal Analysis

4.1 Alloy Code

```
abstract sig AccountRegistration {
    username: one Username,
    password: one Password,
    emailAddress: one EmailAddress
}

sig Username{}
sig Password{}
{ //each password is associated to a registration (some registration may
    ↪ have the same password)
        all p: Password | (some r: AccountRegistration | r.password = p)
}
sig EmailAddress{}

sig User extends AccountRegistration{
        upcomingTickets: set Ticket,
        pastTickets: set Ticket,
}

sig Ticket{
        store: one Store,
        status: one TicketStatus,
        booking: one Booking,
        bookingId: one Int,
        qrCode: one QRCode
}
{
        bookingId > 0
}

sig QRCode{}
{  // There can not be two tickets with the same QR Code
        no disj t1, t2: Ticket | t1.qrCode = t2.qrCode
}

//Abstract definition of Ticket status, with its extensions
abstract sig TicketStatus{}
one sig RESERVED extends TicketStatus{}
one sig SCHEDULED extends TicketStatus{}
one sig USED extends TicketStatus{}
one sig CANCELLED extends TicketStatus{}

sig Booking{
        slot: one Slot,
        shoppingList: set Category
}

sig Slot{
        appointment: one DateTime,
        visitDuration: one Int,
}
```

```
{
        visitDuration ≥ 0
}

sig DateTime{}
{
        no disj s1, s2: Slot | s1.appointment = s2.appointment
}

sig Product{
        name: one ProductName,
        brand: one Brand,
        description: lone Description,
        price: one Float,
        category: one Category,
        quantity: one Int,
}
{
        price.beforePoint > 0 and
        quantity > 0
}

sig ProductName{}
sig Description{}
sig Brand{}

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

abstract sig Category{
        capacity: one Int,
        products: set Product
}
{
        capacity > 0
}

sig BEVERAGES extends Category{}
sig BAKERY extends Category{}
sig JARREDGOODS extends Category{}
sig DAIRY extends Category{}
sig BAKINGGOODS extends Category{}
sig FROZENFOOD extends Category{}
sig MEAT extends Category{}
sig CLEANERS extends Category{}
sig PAPERGOODS extends Category{}
sig PERSONALCARE extends Category{}
sig OTHER extends Category{}

sig StoreAttendant extends AccountRegistration {
        workingStore: one Store,
        attendantCode: one Int,
```

```
}
{
        attendantCode > 0
}

sig StoreAdministrator extends AccountRegistration {
        store: one Store,
        name: one Name,
        surname: one Surname
}

sig Name{}
sig Surname{}

sig Store{
        name: one Name,
        chain: lone Chain,
        position: one GPSPosition,
        cap: one Int,
        upcomingBookings: set Ticket,
        phoneNumber: one PhoneNumber
}

sig Chain{}

sig StoreList{
        stores: set Store
}
{
        no s: Store | s not in stores
}

sig PhoneNumber {}
{ //There can not exists more than one store with the same phone number
        no disj s1, s2: Store | s2.phoneNumber = s1.phoneNumber
}

sig GPSPosition {}
{  //There can not exists more than one store with the same GPS position
        no disj s1, s2: Store | s2.position = s1.position
}

-----------------------------------------------------------------------

// Facts

fact{ //a unique username is associated to each registration
        no disj r1, r2: AccountRegistration | r1.username = r2.username
}

fact{ //a unique email address is associated to each registration
        no disj r1, r2: AccountRegistration | r1.emailAddress = r2.
            ↪ emailAddress
}

fact { //There can not exist two products with the same name and the same
    ↪ brand}
```

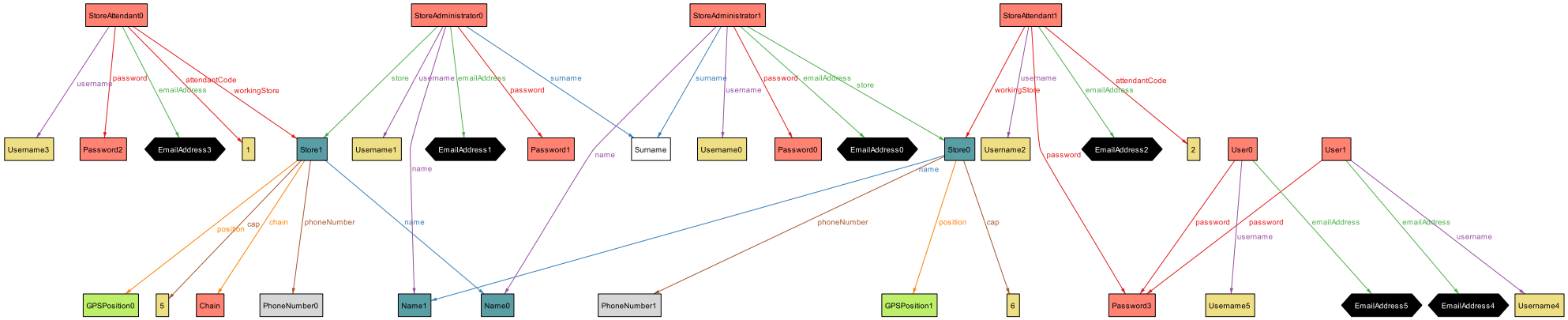
```
        no disj p1, p2: Product | p2.name = p1.name and p2.brand = p1.brand
}

fact { //for each user, the set of bookings must be different
        all disj u1, u2: User | (u1.pastTickets + u1.upcomingTickets)
                & (u2.pastTickets + u2.upcomingTickets) = none
}

fact { //a user can not have two bookings at the same time
        all u: User | (no disj t1, t2: Ticket |
                t1 in u.upcomingTickets and t2 in u.upcomingTickets
                        and t1.booking.slot.appointment = t2.booking.slot.
                            ↪ appointment)
}

fact { //a ticket refers only to one user
        all t: Ticket | (no disj u1, u2: User | (t in u1.upcomingTickets and
            ↪  t in u2.upcomingTickets) or (t in u1.pastTickets and t in u2
            ↪ .pastTickets))
}

fact { //a ticket always has a user
        all t: Ticket | (one u: User | t in u.upcomingTickets or t in u.
            ↪ pastTickets)
}

fact { //each user has disjoint upcoming and past bookings
        all u: User | u.upcomingTickets & u.pastTickets = none
}

fact { //each booking id is unique
        no disj t1, t2: Ticket | t1.bookingId = t2.bookingId
}

fact { //A booking refers only to one ticket
        no disj t1, t2: Ticket | t1.booking = t2.booking
}

fact { //A booking cannot exist without an associated ticket
        all b: Booking | one t: Ticket | t.booking = b
}

fact { //There can not exist one slot without any booking
        all s: Slot | some b: Booking | b.slot = s
}

fact { //two products with the same name could not be in different
    ↪ categories
        no disj p1, p2: Product | (p1.name = p2.name and not (p1.category =
            ↪ p2.category))
}

fact { //Each attendant code is unique
        no disj a1, a2: StoreAttendant | a1.attendantCode = a2.attendantCode
}

fact { //Each store has at most one adminstrator
```

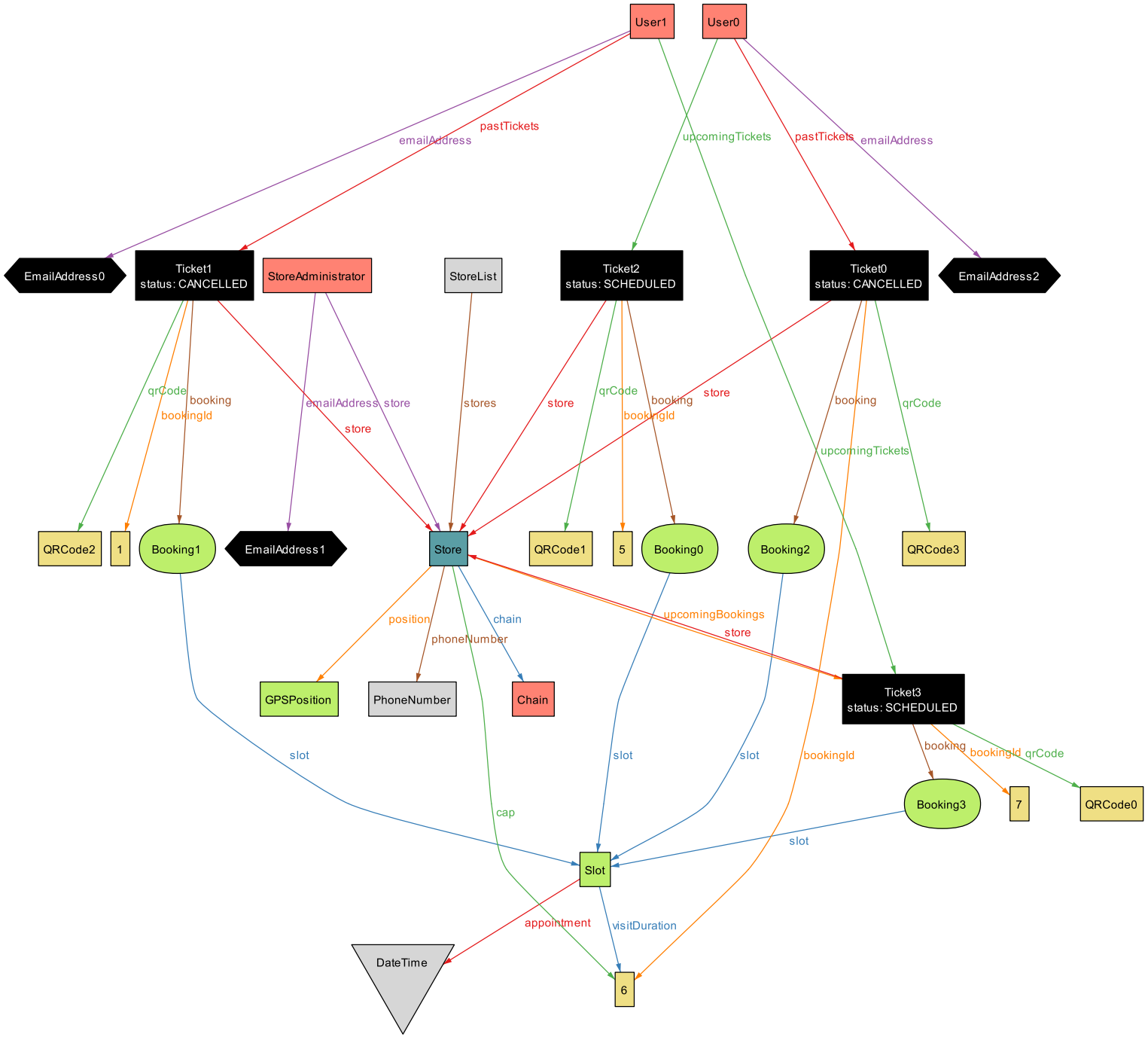
```
        no disj sa1, sa2: StoreAdministrator | sa1.store = sa2.store
}

fact { //A cancelled or used ticket cannot be in upcoming tickets
        all u: User | (all t: Ticket | t.status ≠ SCHEDULED implies u.
            ↪ upcomingTickets & t = none)
}

fact { //An active ticket cannot be in past tickets
        all u: User | (all t: Ticket | t.status = SCHEDULED implies u.
            ↪ pastTickets & t = none)
}

fact { //If a ticket exists, it must be scheduled, reserved, cancelled or
    ↪ used
        all t: Ticket | t.status = SCHEDULED or t.status = RESERVED or t.
            ↪ status = USED or t.status = CANCELLED
}

fact { //The store name must be unique
        no disj s1, s2: Store | s1.name = s2.name
}

fact { //A chain must me unique
        no disj c1, c2: Chain | c1 = c2
}

fact { //An active ticket cannot be in past ticket list
        all t: Ticket | not(t.status = RESERVED and one u: User | t in u.
            ↪ pastTickets) and not(t.status = SCHEDULED and one u:User | t
            ↪ in u.pastTickets)
}

fact { //If a ticket is scheduled, it must have a duration greater then 0
        all t:Ticket | not(t.status = SCHEDULED and t.booking.slot.
            ↪ visitDuration = 0)
}

fact { //A store can not exist without an administrator
        all s: Store | one a: StoreAdministrator | a.store = s
}

fact { //A ticket always has a QR Code
        all t: Ticket | one q: QRCode | t.qrCode = q
}

fact { //A DateTime is always associated to a slot
        all s: Slot | one d: DateTime | s.appointment = d
}

fact { //There can not exist a chain without any store
        all c: Chain | some s: Store | s.chain = c
}

fact { //A chain can contain more than one store
        all c: Chain | some s: Store | s.chain = c
}
```

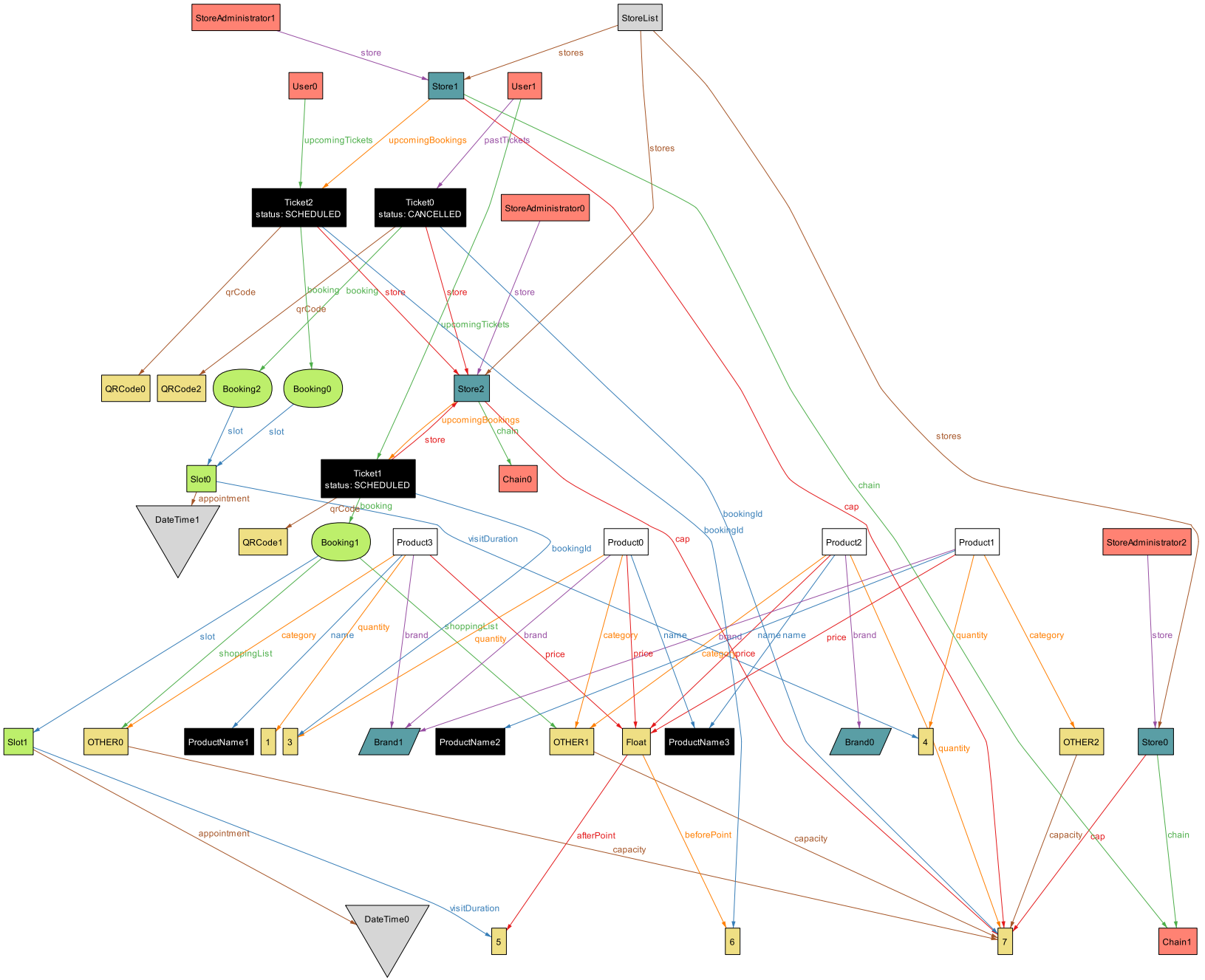
```
fact { //There cannot be a category without products
        all c: Category | some p: Product | p.category = c
}
-------------------------------------------------------------
//assertions

// G1: Allow users to retrieve a ticket
assert retrieveTicket{
        all t: Ticket | t.status = RESERVED implies (one u: User | t in u.
            ↪ upcomingTickets
                and t.booking.slot.visitDuration = 0)
}
check retrieveTicket for 4

//G2: Allow Store Administrators to add a shop to the list of avail-able
    ↪ ones
assert newStore {
        no s: Store | one sl: StoreList | s not in sl.stores
}
check newStore for 4

//G3: Allow users to "book a visit"
assert bookAVisit{
        all t:Ticket | t.status = SCHEDULED implies (one u: User | t in u.
            ↪ upcomingTickets
                and t.booking.slot.visitDuration > 0)
}
check bookAVisit for 4

-------------------------------------------------------------
//predicates

// simulation  that  shows  the  result  some registrations
pred world1 {
        #User = 2
        #Password > 1
        #StoreAdministrator = 2
        #StoreAttendant = 2
        #Store = 2
        #Brand = 0
        #ProductName = 0
        #Description = 0
        #DateTime = 0
}
run world1 for 6

// simulation that shows a set of upcoming bookings and a set of past
    ↪ bookings
pred world2 {
        #User = 2
        #Store = 1
        #Ticket = 4
        some u:User | #(u.pastTickets) > 0 and #(u.upcomingTickets) > 0
}
run world2 for 4
```

```
pred world3 {
        #User = 2
        #Password > 1
        #Store = 3
        #Chain = 2
        #Ticket > 2
        #Slot > 1
        #Category > 2
        #Product > 3
}
run world3 for 10
```

4.2 Worlds


StoreAttendant0
StoreAdministrator0
StoreAdministrator1
StoreAttendant1
username
password
emailAddress
attendantCode
workingStore
store
surname
Username3
Password2
EmailAddress3
1
Store1
Username1
EmailAddress1
Password1
Surname
Username0
Password0
EmailAddress0
Store0
Username2
EmailAddress2
2
User0
User1
name
position
cap
chain
phoneNumber
GPSPosition0
5
Chain
PhoneNumber0
Name1
Name0
PhoneNumber1
GPSPosition1
6
Password3
Username5
EmailAddress5
EmailAddress4
Username4


User1
User0
emailAddress
pastTickets
upcomingTickets
pastTickets
emailAddress
EmailAddress0
Ticket1
status: CANCELLED
StoreAdministrator
StoreList
Ticket2
status: SCHEDULED
Ticket0
status: CANCELLED
EmailAddress2
qrCode
booking
bookingId
emailAddress
store
store
stores
store
qrCode
booking
bookingId
store
booking
qrCode
upcomingTickets
QRCode2
1
Booking1
EmailAddress1
Store
QRCode1
5
Booking0
Booking2
QRCode3
position
phoneNumber
chain
upcomingBookings
store
GPSPosition
PhoneNumber
Chain
Ticket3
status: SCHEDULED
slot
slot
slot
bookingId
booking
bookingId
qrCode
Booking3
7
QRCode0
cap
slot
Slot
appointment
visitDuration
DateTime
6


StoreAdministrator1
StoreList
store
stores
User0
Store1
User1
upcomingTickets
upcomingBookings
pastTickets
stores
Ticket2
status: SCHEDULED
Ticket0
status: CANCELLED
StoreAdministrator0
qrCode
booking
booking
store
store
store
qrCode
upcomingTickets
QRCode0
QRCode2
Booking2
Booking0
Store2
upcomingBookings
chain
slot
slot
store
stores
Slot0
Ticket1
status: SCHEDULED
Chain0
chain
appointment
qrCode
booking
cap
DateTime1
QRCode1
Booking1
Product3
visitDuration
bookingId
Product0
cap
bookingId
Product2
Product1
StoreAdministrator2
slot
shoppingList
category
name
quantity
brand
shoppingList
quantity
brand
price
category
price
brand
name
name
price
brand
quantity
category
store
Slot1
OTHER0
ProductName1
1
3
Brand1
ProductName2
OTHER1
Float
ProductName3
Brand0
4
quantity
OTHER2
Store0
appointment
afterPoint
beforePoint
capacity
capacity
capacity
cap
chain
DateTime0
visitDuration
5
6
7
Chain1

5 Effort spent

Student	Time for S.1	S.2	Time for S.3	Time for S.4
Alice Piemonti	3h	5h	13h	1h
Luca Pirovano	3h	8h	10h	8h
Nicolò Sonnino	3h	6h	11h	8h