POLITECNICO
MILANO 1863

REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT

CLup – Customers Line-up

Yasmin Awad, Lorenzo Carpaneto, Giovanni Dispoto

Contents

1 Introduction

The aim of the following documentation is to provide an overview of the project *CLup - Customers Line-up*: an application whose goal is to allow customers to shop safely. During the Covid-19 pandemic a new problem arose in people's lives. In a world where social distance is no longer a choice but a necessity, it must be found a way to ensure the safety of people in their daily lives, even in places open to the public, such as grocery stores. Having to comply with health and hygiene regulations, only a small amount of people can enter a specific grocery store at a given time, with the result of the formation of very long queues outside the buildings. **CLup** allows to manage customers in order to minimize their time spent enqueued and avoiding overcrowding inside grocery stores. With CLup the customer can line up from home, getting a ticket for the grocery stores or booking Visits in specific time-slots for their needs. This documentation will illustrate a description of the system in terms of its functional and non-functional requirements. We are going to discuss goals, constraints, limits and principal use cases of the application. The document is addressed to the developers, that will implement the requirements, and to the stakeholders, that will supervise the development process.

1.1 Purpose

Customer Line-up (CL-up) is an application that aims to provide users with the tools to obtain access to Stores in a safe way, i.e. avoid as much as possible queue formation outside the Stores and limits the number of people inside them at a given time. All of this is made in order to comply as much as possible with every social distancing rules omitted by the competent authorities. To pursue this goal, the application provides services both to the Owners and the Managers of the Stores, and to the Customers. The application allows Users to register either as a Managers or Customers. Tools are provided to allow Customers to book Visits or Tickets to the Stores remotely, that is for example from their own home, allowing a greater and more efficient way for maintaining social distancing.

There are 2 ways offered by CLup to enter the Stores. The first one involves the reservation of a Ticket that puts the Customer in a virtual queue to enter the Store as soon as possible. The application has also the role of alerting the Customer in order to let them arrive in time for their turn, checking and keeping track of the state of the virtual queue. The second one allows the Customer to book a Visit, that is an entrance to the Store on a specific day, for a specific shopping time-range. To book a Visit the Customer must also specify the category of groceries they are intended to buy, in order to give them a map of the Store and an optimal route for their Visit. Furthermore, the application will not only suggest to the Customer the duration of their shopping based on their data (collected through previous Visits and Tickets), but also recommend the closest Stores available for booking Visits in a day-time range, always based on the Customer's habits.

Regarding Managers, the application gives them the possibility to create a specialized profile in which they can request 'the creation of a Store', i.e. a virtual space that will be identified as their own Store, and through which Customers will be able to book Tickets or Visits to enter the shop itself. The Managers have in fact the task of inserting

and helping the application in the creation and definition of the Store and its rules. All this information may vary over time and must be kept up to date in order to guarantee a better service from the application.

The application also provides a way of getting a Paper Ticket directly from the Store through apposite Totems, without being subscribed into the application. Each Paper Ticket is associated with an entrance time and a QR-code.

People inside the Stores should have booked a Visit or got a Ticket. To check this, the QR-code of the Customer (or the QR-code attached to a Paper Ticket) needs to be checked at the entrance of the Store. In practice, each Customer is associated with a QR code that will appear in their profile. Moreover, to allow the virtual queue to slide adequately, each QR code must be scanned also at the exit.

1.1.1 Goals

G.1 Allow Guests to authenticate as Manager or Customer

G.2 Line up Consumers in an effective way, to avoid overcrowding, long waiting time and regulate the influx of people inside the Stores.

G.3 Allow Customers to virtually line up for Stores requesting a Ticket from different locations and devices.

- G.3.1 Alert Customers when to depart to arrive in time for their shopping.
- G.3.2 Guarantee a paper ticketing service as a fallback option for people who do not have access to the required technology.

G.4 Allow Customers to virtually line up for Stores booking Visits to the Store in different days from different locations and devices.

- G.4.1 Guide Customers inside the Store with a map, to shorten the time of their Visit and minimize contact with other people.
- G.4.2 Notify the Customer in day-time range, if possible Visits which might interest them are available.

1.2 Scope

Following the definition originally proposed by M. Jackson and P. Zave in 1995, we will distinguish world phenomena that are event occurring in real world from machine that is the software to be. World and Machine comunicate with Shared Phenomena, they could be events controlled by the world and observed by the machine or events controller by machine and observed by the world.

- **World**

S.W.1 **People going into Stores**: people going to stores to buy grocery and other products.

S.W.2 **Queue formation**: the creation of long lines outside the stores.

S.W.3 **People moving inside Store**: people moving into store in order to find grocery

S.W.4 **Social Distancing**: circumstance in which the society imposes certain hygienic, sanitary and social distancing rules.

- **Shared**

Controlled by the world, observed by the machine

S.S.1 A Customer can register and sign up to the Service.

S.S.2 A Consumer can take a Ticket for a Store.

S.S.3 Scan the QR code of the Consumer before entering into the Store.

S.S.4 A Customer can book a Visit to the Store specifying categories of grocery.

S.S.5 A Manager can upload a Store map.

Controlled by the machine, observed by the world

S.S.6 A logged Customer can get a ticket to virtual line up.

S.S.7 Send notification to a Customer to remind time for the Ticket reservation.

S.S.8 Show map to the Customer with a built path for the Visit.

S.S.9 The System built a path suggested for a Customer Visit.

S.S.10 Compute information and statistics about a Customer in order to suggest him duration for the Reservation and Stores of interest.

S.S.11

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

D.1 *Login Manager*: during the login phase and registration phase, check the validity and correctness of the given credentials.

D.2 *Ticket Manager*: is a component that takes care of Ticket requests for a Store.

D.3 *Visit Manager*: is a component that takes care of Visit requests for a Store.

D.4 *Map Manager*: is a component that takes care of the generation of an optimal path for a specific Visit and a specific Store.

D.5 *Data Analytics Manager*: is a component that takes care of analysis of the habits of Customers and make suggestions. It is a Recommender System.

D.6 *Queue Manager*: is a component that takes care of the managing of the Queue.

D.7 *Mailing System*: the service which handles the email dispatching.

D.8 *Store*: synonym of Grocery Store. A Store has one associated queue and Consumers can reserve Tickets or book Visits to shop there.

D.9 *Secure Password*: Password that follows the Password Policy

D.10 *Password Policy*: according to NIST, the password must:

- the use of both upper-case and lower-case letters (case sensitivity)
- inclusion of one or more numerical digits
- inclusion of special characters, such as , #, $
- prohibition of words found in a password blocklist
- prohibition of words found in the user's personal information item prohibition of use of company name or an abbreviation
- prohibition of passwords that match the format of calendar dates, license plate numbers, telephone numbers, or other common numbers

D.11 *Type of Users*: the type of users can be either Manager or Costumer.

D.12 *Department*: macro area of the store. It can contain multiple categories of grocery.

D.13 *Owner*: a Manager who owns a specific Store.

D.14 *Capacity of the Store*: maximum number of people admitted inside the Store simultaneously.

D.15 *Virtual Ticket*: a Ticket reserved by a Costumer through the CLup application.

D.16 *Paper Ticket*: a Ticket reserved by a Consumer through the Totem

D.17 *Ticket*: the set of Virtual Tickets and Paper Tickets. Identifies both indistinctly.

D.18 *Visit*: a booked Visit reserved by a Customer through the CLup application.

D.19 *Ticket/Visit expiration*: A ticket or a Visit expires if the Customer does not delete them neither uses them, and the time to use them is past (the current time is greater then the entrance time on the ticket or visit).

D.20 *Temporal Quantum*: a time space of 5 minutes.

D.21 *Reservation*: any time of booking or requesting an entrance to the Store by a Consumer. It can be a Paper Ticket, a Visit or a Virtual Ticket.

D.22 *Duration of Shopping*: Duration of the visit booked.

D.23 *Queuing Mechanism*: the mechanism which handles the behaviour related to the queue.

D.24 *Totem*: a machine used to generate Paper Tickets, each one is related to a specific Store.

D.25 *Scanner*:a machine used to validate Tickets, each one is related to a specific Store.

D.26 *Queue Scaling Process*: process thanks to which Reservations are moved within the queue of the Store according to certain rules, so that the Capacity of the Store is respected and so that a Consumer who needs an additional amount of time to finish their shopping gets that time.

D.27 *Validated*: QR code associated to a Reservation that was correctly scanned at the entrance and at the exit.

D.28 *Slow Hash Function*: Hash function that is more difficult to calculate (for instance Bcrypt) w.r.t fast hash function that are easy to compute for instance SHA-1 (Secure Hash Algorithm - 1) or MD5 (Message Digest 5)

D.29 *Time To Reserve*: 10 seconds (time for the Customer to confirm a reservation)

D.30 *Special Credentials*: credentials which are used to verify that a manager can create a new Store (certified e-mail).

1.3.2 Acronyms

A.1 *API*: Application Programming Interface

A.2 *NIST*: National Institute of Standards and Technology

A.3 *IEEE*: Institute of Electrical and Electronics Engineers

A.4 *DBMS*: Database Management System

A.5 *GPS*: Global Positioning System

A.6 *HTTPS*: Hypertext Transfer Protocol Secure

A.7 *UML*: Uniform Modeling language

A.8 *UX* : User Experience

1.3.3 Abbreviations

AB.1 $[G_i]$: i-th goal.

AB.2 $[R_i]$: i-th requirement.

AB.3 $[D_i]$: i-th definition.

AB.4 $[DA_i]$: i-th domain assumption.

1.4 Reference Documents

- Specification document: R&DD Assignment AY 2020-2021
- IEEE 830-1993: IEEE Recommended Practice for Software Requirements Specifications
- Alloy documentation
- UML documentation

1.5 Revision History

Date	Version	Comments
15 November 2020	1.0	first release
21 December 2020	1.1	Use Cases regarding Visits updated and minor changes (especially to the requirement section)

1.6 Document structure

According to IEEE standard, the RASD is structured into 5 sections

- **Introduction** contains informal presentation of the project, introducing the main goals of the S2B. It also contains all the world phenomena and shared phenomena which the System have to interact. This section contains also all the acronyms, definitions and abbreviations used in the document.
- **Overall Description** contains detailed description of the product perspective and product functions. This section also contains all the assumption, dependencies and contraints in order to get application functional.
- **Specific Requirements** contains some mockup UI of the application, software and hardware requirements and communication interface used for communication.
- **Formal Analysis using Alloy** contains a formal modelling of the project, in order to proof consistency of the core functionality .
- **Effort Spent** show the effort spent in developing the RASD.

2 Overall Description

2.1 Product perspective

In this section we are analyzing all the shared phenomena listed before.

1. **Customer register to Application**: If a User wants to register to the service he have to provide email and password. When the process is completed, the System generate an unique QR code associated to him, used when he goes to Stores.
2. **Manager register to Service**: If a Manager wants to register has to provide his/her email and password.
3. **Customer get a Virtual Ticket**: When the Customer wants to go to a Store, he needs to login into the application, search for a Store near him and enqueuing. To do that he needs to specify the duration of the shopping and the mean of trasport he intend to use to get to the Store from the place he is. In this instant the System provide a time.
4. **Consumer get a Paper Ticket**: When the Consumer wants to go to a Store, at the entrace he/she can get a ticket. On the ticket there is a time in which the Consumer will enter inside.
5. **Scan QR Code**: Before entering and exit employees scan Customers or Consumer QR code in order to start validating the ticket and collect information. The validating process will be closed once the Customer or Consumer scan again the QR code at the exit. If ticket is associated to a Customer then the information are used for profiling and also for updating queue. In instead is only a Consumer, then the information will be used only for updating queue.
6. **Customer books a Visit**: When a Customer books a Visit he has to specify where he wants to go, the day of the Visit, the duration of the Visit and choosing a time-slot available and insert the categories of items that he have to buy.
7. **Alert Customer**: The System alert the Customer that he/she have to go to the store to let him be in time for it.
8. **Showing Map during the Shopping**: In order to minimize the time spent inside the Store and for avoid too much interaction with other Consumers, the application suggest a path to the Customer with a Visit using information of categories of item he wants to buy.
9. **Computing statistics of the Customer**: In order to improve the UX, the application compute statistics of Stores visited, time spent and in general Customers habits. These information are used for suggest Stores and the shopping time. This information is used in order to notify Customers of interested Stores available.

2.1.1 Class Diagram

The class diagram in Figure 1 shows a possible structure of the CustomersLine-up architecture. It should be noted that although the Consumer is not represented, he will be the one to request the generation of Paper Tickets from the Totem in the real world.

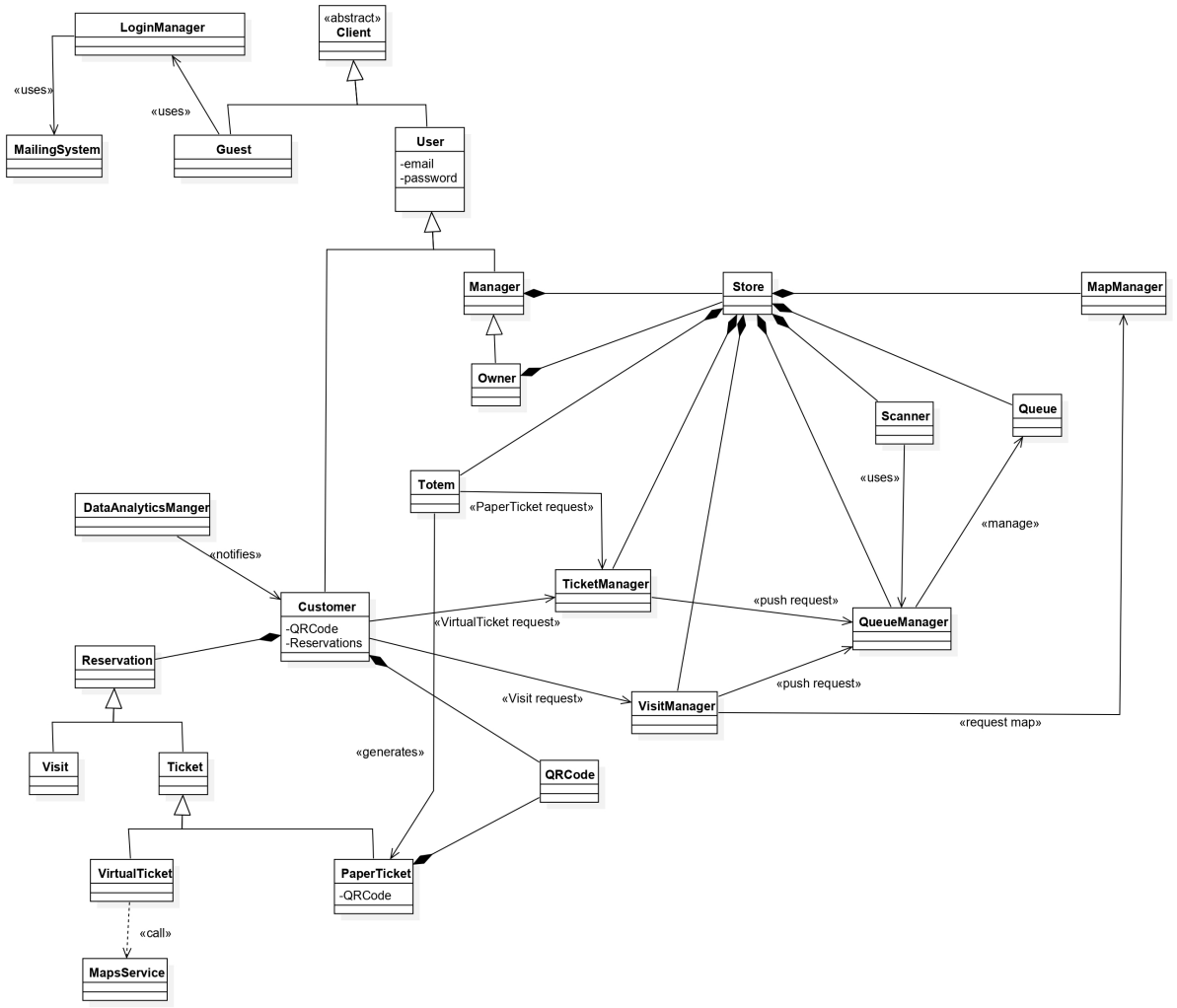

Figure 1: CustomersLine-up class diagram.

2.2 Product functions

Detailed description of the product functions

1. **Get a Ticket**
   A Customer correctly logged in, can see all the near Stores or the ones inferred by the Data Analytics Manager using his/her habits. After selecting one, he/she can get a ticket. In order to get a ticket he/she have to specify the duration of the shopping and the means of transport that he/she wants to use to get to store. For long term Customers the time of the shopping is inferred and suggested using historical information. After providing this information, the System checks that there are no overlapped reservation. Moreover the System checks that there is enough space in the queue of the Store to satisfy his request. Before confirm, the Customer can see how much time he/she have to wait before entering in. During this phase, the System check that the user can satisfy time constraint, that is if the time distance from the place he/she is, is small enough for letting the Costumer arrive in time to the Store with respect to the mean of transport. Moreover, the System checks that the Customer doesn't have other Virtual Tickets. If not, reject the request. After enqueuing, the System notify the Customer when he/she has to star from home. During the permanence inside the Store he/she can see the

map of the store and the position of various departments.

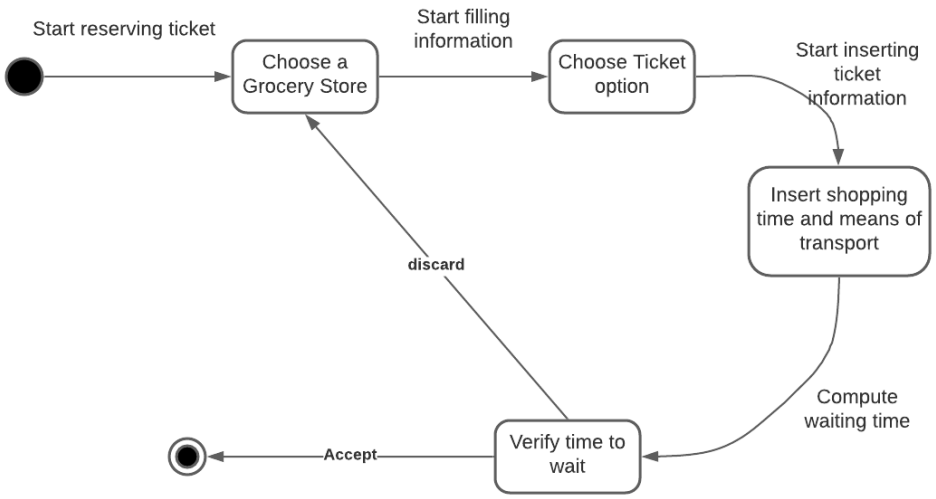

Figure 2: Getting ticket phase

2. **Book a Visit**
   A Customer correctly logged in, can see all the near Stores. After selecting one, he/she can book a Visit. After selecting a Store he/she have to specify the day of interest for the Visit, the duration of the Visit, select a time slot available and specify the categories of grocery that he/she want to buy in order to allow the System to calculate the best path to follow inside the Store in order to maximize social distancing. The duration of the Visit store is inferred and suggested using historical information. During the visit, the Customer can open the app and see the path suggested from the System.

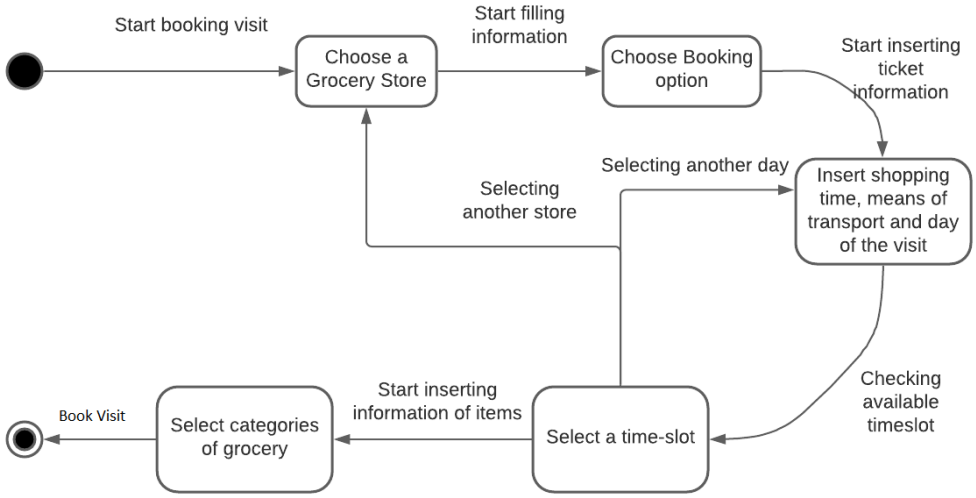

Figure 3: Booking a visit

3. **Add a Store**
   A Manager correctly logged in can add a new Store. In order to create a Store a Manager should specify the name of the Store, the address, the maximum number of people that can enter inside the store according to health directives (i.e. the Capacity), maximum number of people for each department of the store and

opening hours. During this phase the Manager could specify other Managers of the store. After creation, he/she have to specify also credentials used by the System to verify that the store exists. At the end of this phase, a Manager upload a map of the stores, reporting information of the store such as walls, departments and category of grocery for each department. Once a Manager created a Store, he/she become the Owner of the Store.

4. **Manage a Store**
   A Manager correctly logged in can edit information of all his Stores. He/she can edit the maximum number of people according to new health directives, modify the maximum number of people inside each department, map of the Stores, opening and closing hours of the Store, the percentage of Capacity dedicated only to Tickets, see codes or generate them to activate the Scanner and the paper ticket machines. The Owner of a Store can also modify the Managers, transfer ownership or delete a store from the System.

5. **Perform a visit with Virtual Ticket**
   When is the turn of a Customer enqueque, before entering, he/she have to show his/her qr code to scanner machine in order to start validating the visit. Once the Customer finish his shopping, during the payment, he/she scan one more time the qr code, in order to Validate the visit. During the Visit the Customer can see the map in order to optimize his/her time spent in searching items.

6. **Perform a visit with Paper Ticket**
   When the Consumer arrives to the Stores, get a Paper Ticket from the ticket machine. On the Paper Ticket is printed the entering time and his QR code. Before entering the Customer has to scan the QR Code with the scanner machine. Once the Consumer finish his shopping, during the payment, he/she scan one more time the QR Code, in order to register the exit.

7. **Perform a booked visit**
   Before entering, the Customer scan his/her qr code. When he/she is inside, he can open the app and see the calculated path to follow in order to buy everything that he/she needs. Once the Customer finish his shopping, during the payment, he/she scan one more time the qr code, in order to register the exit.

2.3 User Characteristics

CL-up can be used from both Customers and Managers. We take for granted that the Users have access to Internet and that they allow the System to access their location in real time. We also take for granted that Users are able to install and use the mobile application. The characters are:

- Guest: anyone who downloads and opens the app but still has to sign up or log in. He/She cannot use any of the tools provided by CL-up.
- User: a registered Guest.

- Consumer: anyone who need to enter a Store, that is anyone who has a Ticket or a Visit.
- Customer: someone who has logged-in with his/her credentials which is recognized by the System by his/her email, and is identified by his/her QR code. He/She can access their profile, request a Ticket or a Visit for a Store, and visualize them in the purchased section.
- Manager: someone who has logged-in with his/her credentials which is recognized by the System by his/her email. He/She can access their profile, create and manage Stores.
- Owner: a Manager that has created a Store, or who inherited a Store from another Manager. A Manager can be an Owner of one or more Stores. Every Store has only one Owner.

2.4 Assumptions, Dependencies and Constraints

2.4.1 Assumptions

DA.1 We assume that Paper Tickets will be few (if a Store has a lot of Reservations only a small amount of them are Paper Tickets).

DA.2 Each Store provides at least a Scanner.

DA.3 Each Store provides at least a Totem.

DA.4 Consumers who enter a Store use a ticket, a paper ticket or a Visit and the vast majority of them scan their QR both at the entrance and at the exit.

DA.5 Consumers that exceed time specified during reservation are few.

2.4.2 Dependencies

1. A DBMS is needed in order to store Users' credentials and Customers data.
2. An external Mailing System is used to send mails.
3. The system relies on a Recommender System to analyze data and infer suggestions for each Customer.
4. Customers Line-up is not responsible for the non-compliance with the hygienic, sanitary and social distancing rules.

2.4.3 Constraints

1. Smartphones are equipped with an OS compatible with the application.
2. Properly working Internet Connection.
3. User gives permission to the app to use GPS.
4. The permission to acquire and store personal data must be given by the User.

3 Specific Requirements

3.1 External Interface Requirements

CL-up is a mobile based application, that can be also consulted using a web browser. Additional details regarding the hardware, software and communication interface requirements are described and shown below. There will also be some prototypes of User Interface for users who use mobile.

3.1.1 User Interfaces

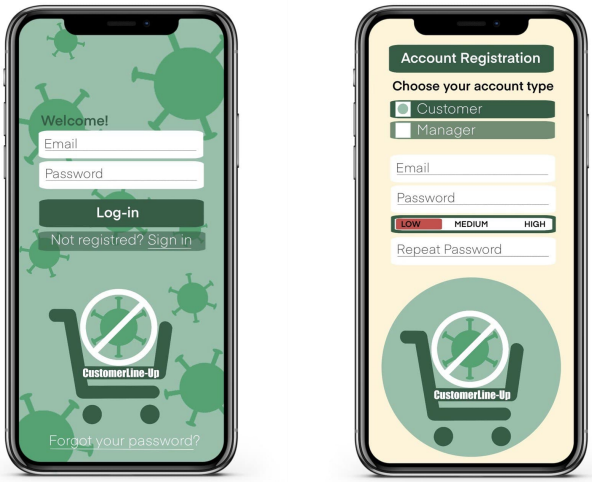

Figure 4: Guest login interface and signup interface.

Figure 5: Customers interfaces for list of Stores (for doing a Reservation), list of Reservations and profile.

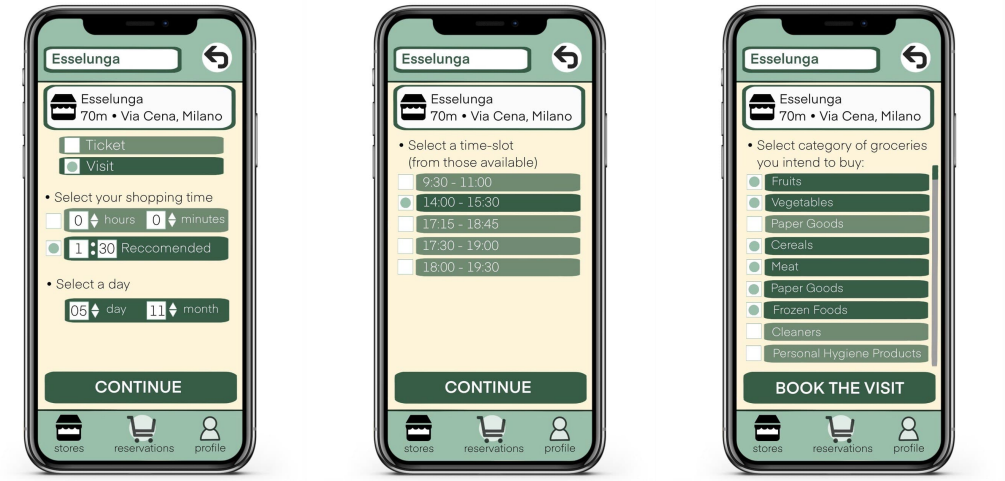

Figure 6: Customer interfaces for booking a Visit, once they have chosen the Store.

Figure 7: Customer interfaces for booking a Virtual Ticket, once they have chosen the Store.

Figure 8: Manager interfaces to view the membership and owned Stores.

3.1.2 Hardware Interfaces

The application is available for any type of mobile, having Internet access, a relatively recent browser, a GPS function, and enough storage space to accommodate the application itself.

3.1.3 Software Interfaces

- Operating System: iOS, Android.
- Web browser.
- Web Server application.
- DBMS.
- Visit System: use third part services for GPS function.
- Mailing System: APIs to send emails to the User.

3.1.4 Communication Interfaces

The System runs over HTTPS protocol for the communication with application and notification system. The System also communicate with DBMS.

3.2 Functional Requirements

LOGIN & REGISTER SERVICE

R.1 The User must be able to register or to log in if already registered.

R.2 Check if the User credentials are valid:

- Check that the email is in the right format.
- Check that the email that the user is providing has not been used for previous registrations.
- Check that the password is a Secure Password.

R.3 If the credentials are valid, the System sends a confirmation email, to let the User activate their account.

- If the User does not confirm their credentials within 24 hours, the account is removed and the credentials deleted.

R.4 Allow to log in using personal credentials:

- Check if credentials are valid.
- In case the email and password inserted are correct, allow the User to access all the functionalities available for the Type of User they are.
- If the email and/or password inserted are wrong, the System must deny the access.

- If the username and/or password inserted are wrong, the System must notify the User.

R.5 Allow to change credentials, if the new credentials are in the right format.

R.6 Allow to change password if it has been forgotten, through the personal email.

R.7 Provide to each Customer a unique QR code which will be used as identifier within the System.

R.8 Users must have the possibility to delete their accounts.

TICKETING SERVICE

R.9 The Customer must be allowed to request a Ticket, specifying:

- The Store where he/she wants to shop.
- The duration of his/her shopping.
- The means of transport to get to the Store.

R.10 For Customers who are requesting a Ticket, the System should:

- Verify that their time distance from the Store is such that they can arrive in time for their turn.
- Insert the Customer Ticket request in the queue in order to let the Customer arrive in time for his/her shopping.
- Provide to the Customer the estimated waiting time before their turn.
- Notify Customers when they should depart from home in order to arrive in time for their turn.
- The Customer, after receiving the request for confirming the Ticket reservation, has Time to Reserve to do it, otherwise the Ticket is cancelled.

R.11 Each Customer can request only one Ticket. When they delete a request or they finish shopping in a Store and check outwith the Ticket or when the ticket expires, the Customer will be able to request another Ticket.

R.12 Every person on a Ticket can view the time in which they should enter the Store and the indicated duration of their shopping.

BOOKING SERVICE

R.13 The Customer must be allowed to request a Visit, specifying:

- The Store where he/she wants to take a Visit.
- The day of the Visit.
- The duration of his/her Visit.
- The time slot for the Visit.
- The Categories of Grocery he/her is going to buy.

R.14 For long time Customers, while booking a Visit or a Ticket, the System should infer, analyzing previous Reservations of that Customer, the expected duration of the Visit or Ticket. The Customer is free to use the suggestion or specify themselves the duration.

R.15 The System should provide to the Customer a map of the Store.

- For Customers who have booked a Visit, the System should suggest to the Customer an optimal shopping path in order to buy everything they need, to minimize possible contacts between people inside the Store and to shorten the time of the visit.
- Each time the Customer views the map, the System checks if the current map is coherent with the Store information and updates it if not.

R.16 Allow Customers to consult their pending requests.

R.17 The Customer must be allowed to delete a Visit or a Ticket from his/her pending requests.

R.18 Use third parties services to enable the localization of Customers and Stores.

R.19 Notify Customers of available slots in a day-time range inferring information from their previous Visits.

R.20 Each Customer can book up to three Visits in a single day.

R.21 If a Reservation is expired, or it has served its purpose, or it has been deleted, it will be removed and will not be visible to the Customer anymore.

STORE MANAGEMENT SERVICE

R.22 Allow Managers to create Stores, becoming the Owner of the Store.

- Allow to specify name of the Store and his address.
- Allow Managers to specify the Capacity of the Store.
- Allow Managers to specify maximum number of people for each Department of the Store.
- Allow Managers to specify Store opening hours.
- Allow managers to specify how many days in advance a Visit can be booked
- Allow Managers to specify the percentage of Capacity of the Store reserved exclusively for Ticket's reservations.

R.23 Allow Owner to delete their Stores.

R.24 Verify the Store creation, acquiring specific credentials (for instance a Certified Email) that can be used to verify the validity of the Store and of the Manager.

R.25 Allow the Owner and Managers to edit the Store:

- Allow Managers to edit the Capacity of the Store.
- Allow Managers to edit maximum number of people for each Department of the Store.

- Allow Managers to edit Store opening hours.
- Allow Owner to add other Managers to their Store(s).
- Allow Owner to remove Managers from their Store(s).
- Allow Owner to hand over the ownership of the Store to one of its Managers.
- Allow managers to edit how many days in advance a Visit can be booked.

R.26 All the information about the Store can be updated by their Managers at anytime.

- If the updates change opening hours or the Capacity of the Store or how many days in advance a Visit can be booked, the Manager will have two options: make the updates effective within 24 hours, losing booked Visits which are not valid anymore, or make the updates effective within the current amount of how many days in advance a Visit can be booked without taking a chance on losing booked Visits.
- If some Visits are cancelled, the System notify the Customers involved.

R.27 Allow Managers to upload a map of their Stores:

- Provide a GUI to specify the structure of the Store to the System (walls, Departments, category of grocery for each Department)
- Allow Managers to specify Categories of groceries for each Department.

R.28 Provide a specific **Scanner App** to scan the QR codes of the Consumers and to interact with the Queuing Mechanism.

- Once a Scanner scan a QR code it can visualize one of this messages: if the Consumer of that QR code can enter the Store, or if he is still in the queue (it's not his turn yet), or if his reservation has been deleted (because of Queue Scaling Process or because he never reserved anything).
- The Managers of a Store can create a unique identifier linked to the Store to input it to Scanner APP to enable its functions.

R.29 Provide a specific **Paper Tickets App** to generate paper tickets in the Store and to interact with the Queuing Mechanism.

- The Paper Ticket app will allow every person to get a Paper Ticket directly from the Store using a Totem.
- In the Paper Tickets there will be specified the same information provided with the online Ticket and it will work the same way. The associated QR code will be automatically generated by the Paper Ticket app, and it will be different from the QR codes of each Customer (i.e. User registered in the main app).
- The Managers of a Store can create a unique identifier linked to the store to input it to the Paper Ticket app to enable its functions.

R.30 Each Store has one and only one queue.

- The queue for each day is as long as the number of minutes the Store is open divided by 5, ie a Temporal Quantum.

- Every Temporal Quantum of the queue contains as many elements as the Capacity of the Store.
- Every time a Consumer makes a Reservation, some spaces on the queue are searched and reserved, in order to let him have enough time for the duration of his shopping.
- Consumers who have booked a Visit will be inserted in the same queue of the ones who have got a (paper or virtual) Ticket, but they will have higher priority.
- Consumers with a Paper Ticket have a higher priority than the ones with Virtual Tickets.
- Only the Customers will have the entering time updated in real time.
- If the Consumer will not show up in time for their Reservation, their Reservation will be considered invalid and it will be cancelled.
- A Consumer which does not go out of the shop within the duration declared will activate the Queue Scaling Process and the duration of his/her visit will be extended.
- The scaling of Consumers in the queue has to be done in such a way that Consumers can be scaled only forward and not backwards.
- During the Queue Scaling Process, any scaling that causes a Reservation to exceed the closing hours will lead to the cancellation of the Reservation.
- During the Queue Scaling Process, any scaling that causes a Virtual Ticket or Visit to overlap with another Virtual Ticket or Visit of the same Costumer (also for different Stores), there will be a cancellation of one of the new Reservation.
- Every time a Reservation of a Customer is cancelled, the Customer must be notified.
- The last 30 minutes in which a Store is open cannot be booked by any type of Reservation, so that they can be exploited within the Queue Scaling Process, in order to eventually have to eliminate a smaller number of Reservations.
- The queue should be made in such a way that after the closing hours of a Store, it delete every old Reservations, that is every Reservation made for that day. In this way just future booked Visits are preserved.
- The queue has to be made in such a way that, also if there are more opening and closing hours of a Store in a single day,closing hours can't be booked and can't be scaling in between.

R.31 Suggestions are based on the habits of the Customers, inferred by the Data Analytics Manager. In a first period will be suggested Stores which are available and geographically close to the Customer (within 1 km).

Goal to requirements and domain assumptions Matrix

GOALS	REQUIREMENTS	DOMAIN ASSUMPTIONS
G.1	R.1, R.2, R.3, R.4, R.5, R.6, R.8, R.21, R.24	—
G.2	R.7, R.22, R.23, R.18, R.25, R.26, R.27, R.28, R.30, R.29	DA.1, DA.2, DA.3, DA.4, DA.5
G.3	R.9, R.10, R.11, R.12, R.17, R.18	—
G.4	R.13, R.14, R.15, R.16, R.17, R.18, R.19, R.20, R.31	—

3.2.1 Use Cases

UC.1 Registration

UC.2 Login

UC.3 Update Credentials

UC.4 Lost Password

UC.5 Get Ticket

UC.6 Book Visit

UC.7 Access Store

- Using Ticket
- Using Visit

UC.8 Delete Ticket/Visit

UC.9 Create Store

UC.10 Activate Scanner

UC.11 Activate Paper-ticket machine

UC.12 Update Store info

Traceability Matrix

USE CASES	GOALS	REQUIREMENTS	DOMAIN ASSUMPTIONS
UC.1	G.1	R.1, R.2, R.3, R.7	—
UC.2	G.1	R.1, R.2, R.4	—
UC.3	G.1	R.5	—
UC.4	G.1	R.6	—
UC.5	G.2, G.3	R.9, R.10, R.11	—
UC.6	G.2, G.4	R.13, R.20, R.14, R.20	—
UC.7	G.2, G.3, G.4	R.15, R.18, R.31	DA.2, DA.3, DA.4, DA.4, DA.5
UC.8	—	R.17, R.21	—
UC.9	—	R.22, R.24	—
UC.10	G.2, G.3	R.28	DA.2
UC.11	G.2, G.3	R.29	DA.3
UC.12	G.2	R.26, R.25	—

UC.1: Registration

Description	A Guest wants to create an account for the main application.
Actors	***User***, ***Login Manager*** and ***Mailing System***
Preconditions	The User downloads and opens the application.
Flow of Events	1. The User clicks on the Sign Up button. 2. The User fills the editable fields with the requested information: email, password and if they want to register as Customer or Manager. 3. The Login Manager checks if all the credentials are valid. 4. If the credentials are valid, the Login Manager sends a confirmation email to the User to activate the account. 5. The User receives the mail with a specific URL to activate the account. By clicking on the given URL, they activate the account. 6. The User is now a Customer, they get a personal unique QR-code.
Post-Conditions	Activation of a new account. The User can access the services of the product.

Exceptions	1. If the User does not enter valid credentials (invalid email or a password which is not a Secure Password [D.9]), the registration fails and the user is prompted to restart the registration process. 2. If the User does not confirm their credentials within 24 hours, the account is removed and the credentials deleted.

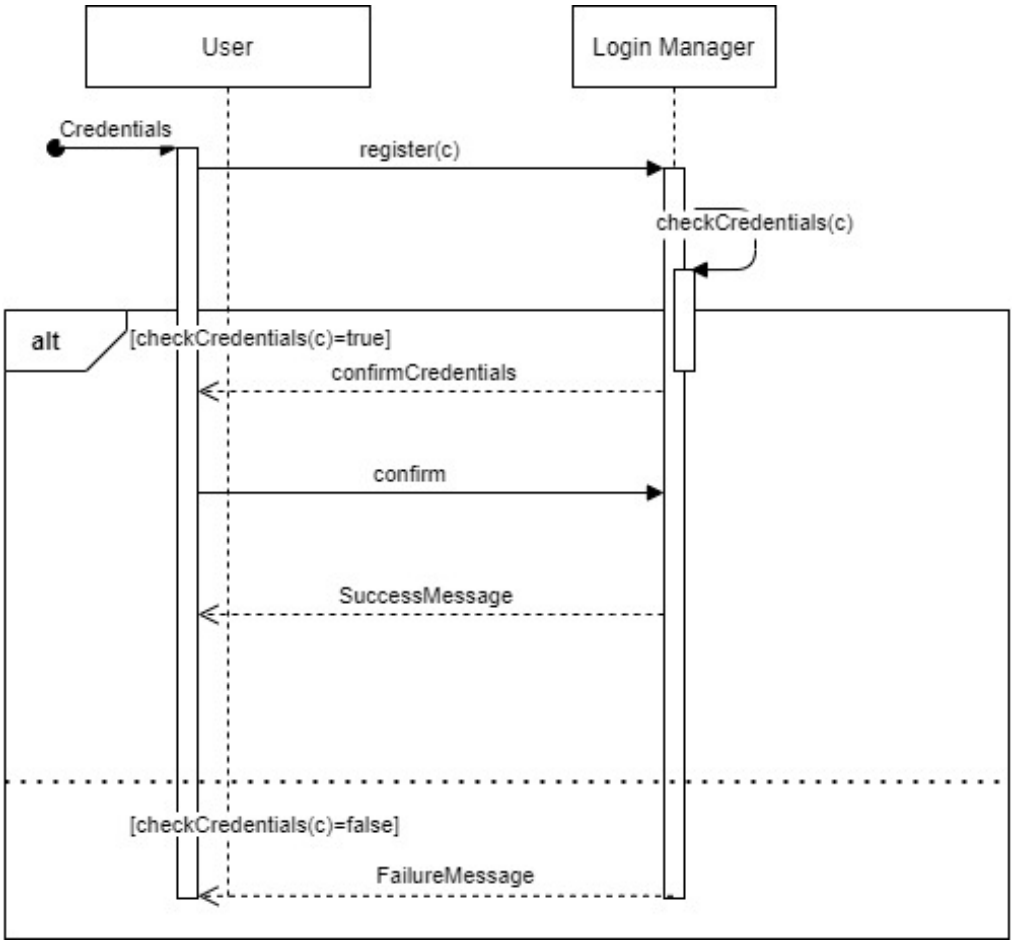

Figure 9: Registration Sequence diagram, if the User confirms the credentials when they have to. Also according to temporal constrain for credential acceptance

Figure 10: Registration

UC.2: Login

Description	A Customer wants to log-in into the main application.
Actors	***Customer***, ***Login Manager***
Preconditions	The Customer opens the application on their device and is registered.
Flow of Events	1. The Customer fills the editable fields with the requested information: email and password 2. The Customer clicks on the Sign In button. 3. The Login Manager checks if the credentials are valid. 4. If the credentials are valid, the Customer successfully logs-in the application.
Post-Conditions	The Customer logs-in the application with their personal account.
Exceptions	1. If the Customer does not enter valid credentials, a failure message pops up on the screen prompting them to re-enter the credentials.

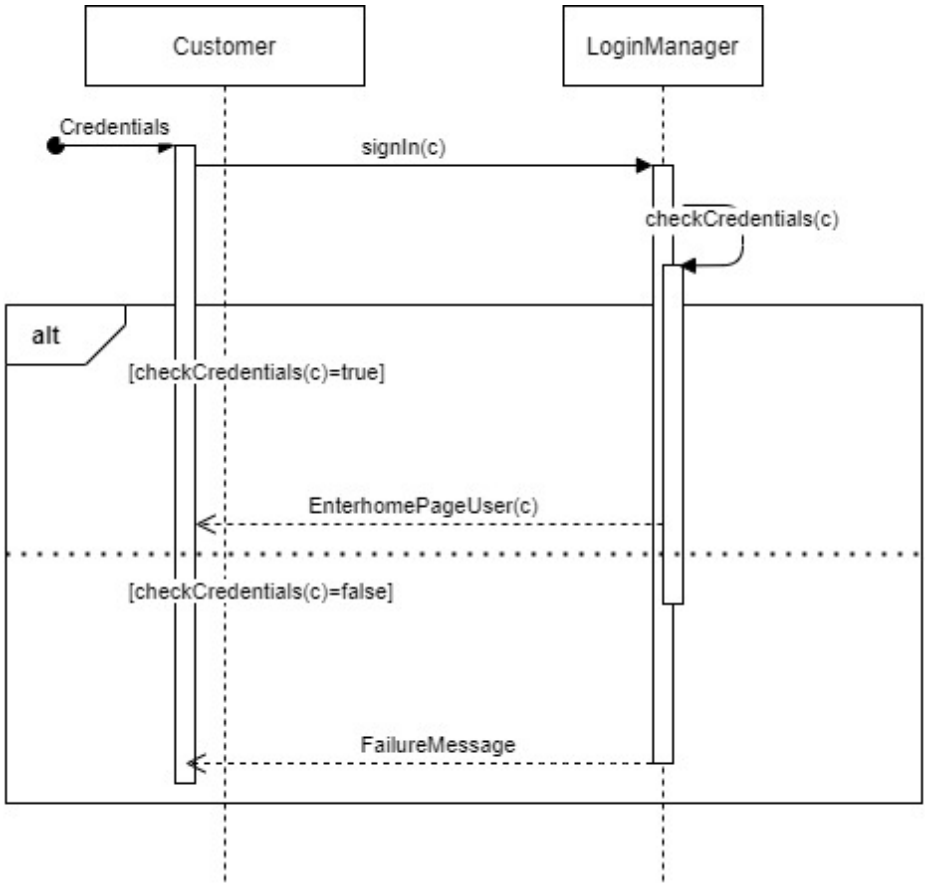

Figure 11: Login Sequence diagram

UC.3: Update Credentials

Description	A Customer wants to update their credentials.
Actors	***Customer*** and ***Login Manager***
Preconditions	The Customer opens the application on their device and is logged in.
Flow of Events	1. The Customer clicks on the update credential button, from the home page. 2. The Customer fills the editable fields with the requested information: email and password, and confirms the operation through a button. 3. The Login Manager checks if the new credentials are valid. 4. If the new credentials are valid, the Customer has successfully updated their credentials
Post-Conditions	The Customer successfully updates their credentials.
Exceptions	1. If the Customer does not enter valid credentials, a failure message pops up on the screen, and the credentials are not updated.

UC.4: Lost Password

Description	A Customer wants to log-in into the main application but they have forgotten the password.
Actors	***Customer***, ***Login Manager***, ***Mailing System***
Preconditions	The Customer opens the application on their device and is registered.
Flow of Events	1. The Customer clicks on the recover password button, in the log-in page. 2. The Customer enter their mail in the editable box, and they click the recover password button. 3. The Login Manager checks if the mail is associated to a registered Customer. 4. If the mail is associated to a registered Customer, the Login Manager sends a mail to the user with appropriate instructions. 5. The Customer follows the instructions on the email and is able to access their account.
Post-Conditions	The Customer can log-in the application.
Exceptions	1. If the Customer does not enter a mail associated to a registered user, the procedure fails and a failure message pops up on the screen, prompting the User to insert a mail associated to one account.

UC.5: Get Ticket

Description	A Customer requests a Virtual Ticket for lining up for entering a specific Store.
Actors	***Customer***, ***Ticket Manager***, ***Queue Manager*** Check These
Preconditions	The Customer opens the application on their device and is logged in.

Flow of Events	1. The Customer clicks on the reservation tab in the home page. 2. The Customer searches for the Store they want to go to, typing its name in the apposite text box or looking at the suggestions in the apposite list. 3. The Customer clicks on the Store they want to go to, from the list which has spawned in the mean time. 4. The Customer checks the Ticket check-box. 5. The Customer specifies the duration of their shopping using the apposite GUI elements. The Customer can either input a duration or select one which is inferred by the DataAnalytics Manager. 6. The Customer specifies with which means of transport they will go to the Store. 7. Using third party API, and the current location of the Customer, the System compute how much time the Customer needs to go to the Store 8. The Customer clicks on the get Ticket Button and the Ticket is sent to the Ticket Manager of the Store. 9. The Ticket Manager checks if the information on the ticket is sound. 10. The Ticket Manager sends the information on the Ticket to the Queue Manager of the Store, that looks for the available positions in the queue. 11. The Queue Manager places the Customer in the queue accordingly and sends the information to the Ticket Manager. 12. The System alerts the Customer on how much time they would have to wait for their turn and asks for confirmation. 13. If the Customer clicks on the ”reserve” button respecting the Time To Reserve constraint [D.29], and has no other valid ticket, the ticket request is successful and the ticket is reserved.
Post-Conditions	The Customer gets the ticket.

Exceptions	1. If the Customer exits the application before hitting the reserve button, the request is silently discarded (The Ticket Manager sends a message to the Queue Manager which deletes the Ticket from the queue). 2. If there are no position available in the Queue, the system logs to the Customer a failure message and the request is discarded. 3. If the Customer is too far away from the store to arrive and finish their shopping in time for the closure of the Store, the system logs to the User a failure message the request is discarded. 4. If a Customer already has a Virtual Ticket in his reservations, the System does not allow the Customer to get another one, logs a failure message and the request is discarded. 5. If the Customer takes too long (more than Time To Reserve [D.29]) to click the ”reserve” button, the Ticket is cancelled.

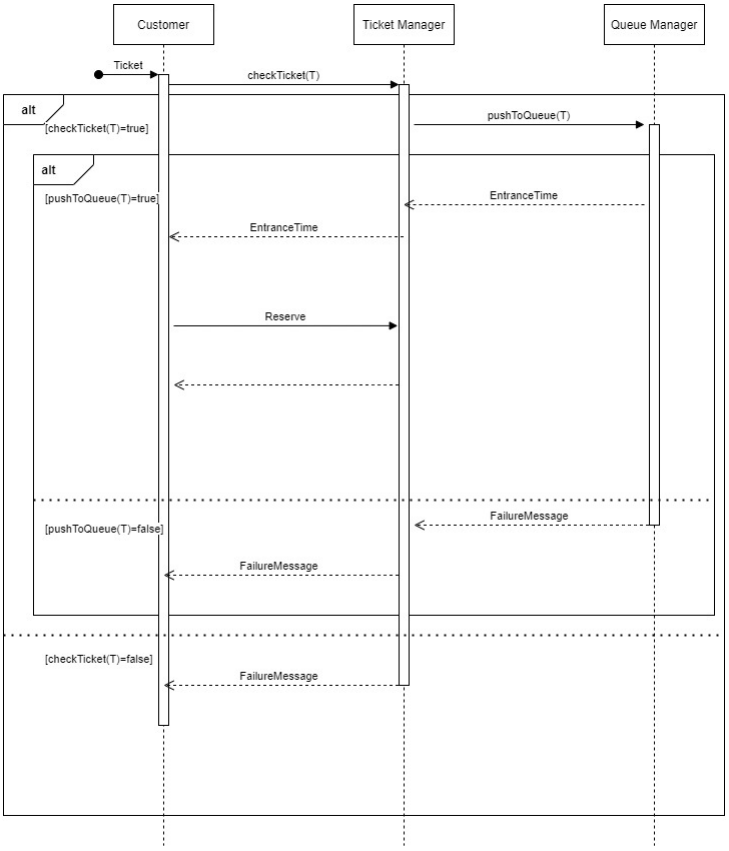

Figure 12: Get Ticket Sequence diagram, after having compiled the ticket form, and with the Customer clicking the reserve button respecting its time constraint.

UC.6: Book Visit

Description	A Customer books a Visit for a specific Store.
Actors	***Customer***, ***Visit Manager***, ***Queue Manager***
Preconditions	The Customer opens the application on their device and is logged in.

Flow of Events	1. The Customer searches for the Store they want to go to, typing its name in the apposite text box or looks at the suggestions in the list. 2. The Customer clicks on the Store they want to go to, from the list which has spawned in the mean time. 3. The Customer checks the Visit check-box. 4. The Customer specifies the duration of their Visit using the apposite GUI elements. The Customer can either input a duration or select one which is inferred by the Data Analytics Manager. 5. The Customer specifies which day they are planning to do the Visit. 6. Clicking the apposite button, the Customer sends the Visit to the Visit Manager of the Store. 7. The Visit Manager checks if the information is valid and sends to the Queue Manager of the specified Store the Visit information. 8. The Queue Manager sends the available time slots in the specified day to the Customer. 9. The System displays time slots available, in the day requested by the Customer to visit the specific Store. The Customer can choose one of those. 10. The Customer can select the categories of items they would like to buy. 11. The Customer can book the Visit, clicking on the apposite button, if in the specified day they do not have already booked 3 Visits. 12. The Customer sends the Visit information to the Visit Manager, which confirms the booking to the Queue Manager and this inserts the Customer into the Queue of the Store, in the specified day, and at the specified time. 13. The Queue Manager communicates to the Map Manager that a new Visit has been added and it sends them the Visit information.
Post-Conditions	The Customer books the Visit.

Exceptions	1. If the Customer has booked three Visits in specified day, the system logs a failure message and discards the Visit. 2. If there are not any available time-slots (of any Store) the System logs a failure message. 3. If, when the Customer sends the confirmation of the booking, the time-slot selected is not available anymore, the System logs an error message.

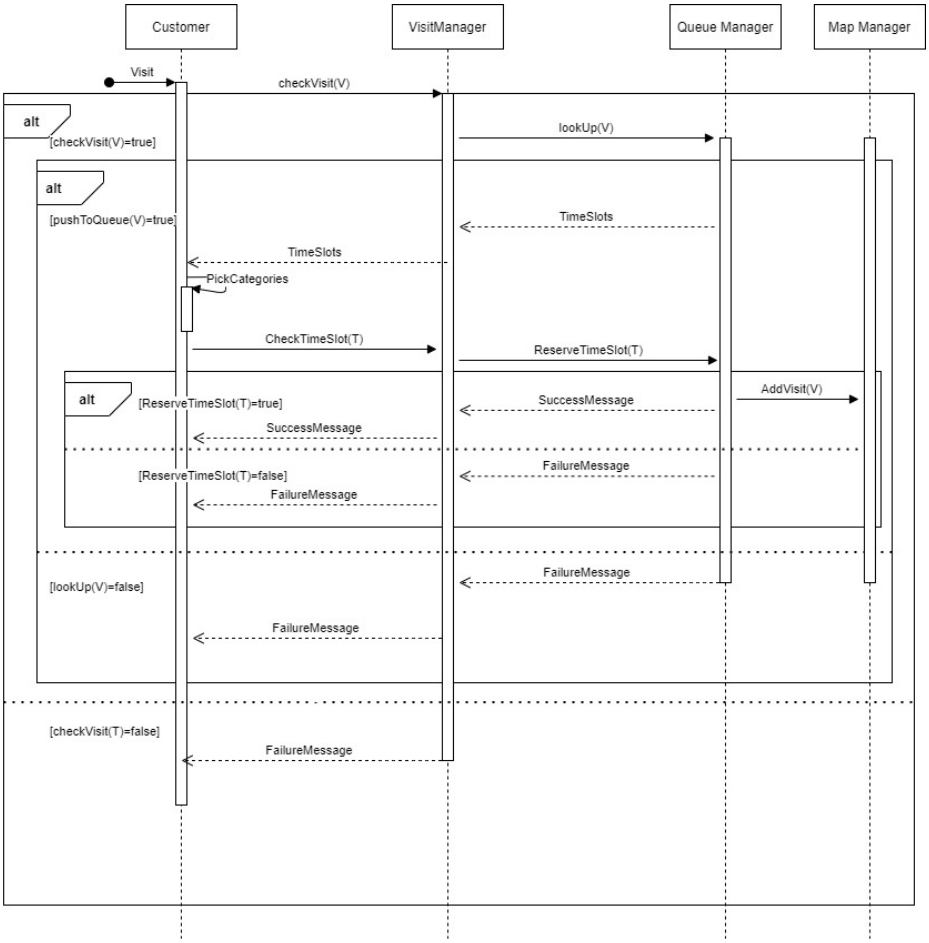

Figure 13: Book Visit Sequence diagram, after having compiled the Visit form.

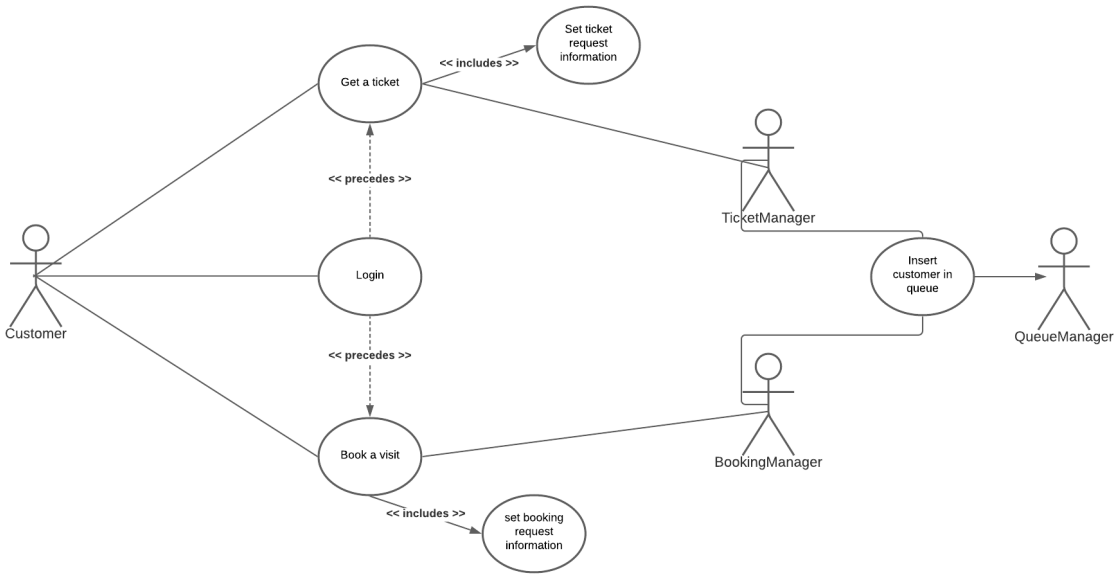

Figure 14: Get Ticket and Book Visit

UC.7: Access Store - Using Ticket

Description	A Customer shops into a store, accessing it with a ticket.
Actors	***Customer***, ***Queue Manager***,***Scanner***
Preconditions	The Customer has a reserved ticket.
Flow of Events	1. The System tells the Customer when to depart from their place to go to the Store on time with the specified means of transport. 2. Once arrived at the Store the Customer checks the entrance time on the Ticket (it could be updated). The Customer enters the Store when specified by the ticket. 3. The Customer scans their personal QR-code (which can be found in the profile tab in the app) with the Scanner provided by the Store. 4. The Customer enters the Store and tries to comply with the duration specified on the ticket. 5. The system collects the duration of the shopping, storing it in the Database for analytics purposes. 6. If the Customer takes more time to shop than the duration specified, the Queue Manager applies a scaling process to avoid overcrowding. 7. After paying the Customer is prompted by the Store's operator to scan again the QR-code.
Post-Conditions	The Customer has done their shopping in the desired Store.

Exceptions	1. If the Customer cannot make it in time, the ticket reserved is invalidated by the Queue Manager.

UC.7: Access Store - Using Visit

Description	A Customer shops into a store, accessing it with a Visit.
Actors	***Customer***, ***Queue Manager***, ***Scanner***
Preconditions	The Customer has a valid Visit booked.
Flow of Events	1. Once arrived at the Store the Customer checks the entrance time on the Visit they booked (it could be updated). The customer enters the Store when specified by the Visit. 2. The Customer scans their personal QR-code (which can be found in the profile tab in the app) with the Scanner provided by the Store. 3. The Customer enters the Store and tries to comply with the duration specified on the ticket. 4. If the Customer is late, the Queue Manager automatically updates the queue to avoid overcrowding. 5. The Customer is guided by the app through the store thanks to a map which show them the path to arrive to the categories specified during the booking of the Visit and tries to minimize both the path length and the interaction with other people (to reduce overcrowding). The map is provided by the Map Manager. 6. The system collects the duration of the Visit, storing it in Database for analytics purposes. 7. After paying the Customer is prompted by the Store's operator to scan again the QR-code.
Post-Conditions	The Customer has done their Visit in the desired Store.
Exceptions	1. If the Customer cannot make it in time, the ticket reserved is invalidated by the Queue Manager.

UC.8: Delete Booked

Description	A Customer deletes a booked Visit or a Ticket.
Actors	***Customer***, ***Visit Manager***, ***Queue Manager***
Preconditions	The Customer has booked a Visit or reserved a Ticket.

Flow of Events	1. The Customer goes onto the reservation tab. 2. The Customer presses onto the reservation they want to delete. 3. The Customer presses on the delete button. 4. The Queue Manager receives a notification with the information of the reservation. 5. The Queue Manager deletes the reservation. 6. If the reservation was a Visit, the Queue Manager notifies its deletion to the Map Manager which deletes the information about the Visit as well. 7. The Queue Manager updates the queue status accordingly.
Post-Conditions	The Customer has deleted their Visit/Ticket reservation.
Exceptions	1. During the Visit/shopping time with the Ticket, the Costumer cannot delete the corresponding Ticket/Visit.

UC.9: Create Store

Description	A Manager creates a new Store.
Actors	***Manager***, ***Mailing System***
Preconditions	The Manager is logged-in into the main application.
Flow of Events	1. The Manager clicks on the create Store button in the Store tab. 2. The Manager inputs the requested information in the editable boxes. 3. The Manager inputs Special Credentials D.30, to certify the authenticity of the Store. 4. The System checks the credentials and sends the Manager a confirmation mail. 5. The Customer follows the instructions on the email to activate the Store. 6. The Store is created, and the Manager becomes the Owner of the Store.
Post-Conditions	The Store is created, and the Manager becomes the Owner of the Store.

Exceptions	1. If the information given is invalid, the System denies the Store creation and logs to the screen an error message.

UC.10: Activate Scanner

Description	A Manager activates a scanner associated with one of their Stores.
Actors	***Manager***, ***Queuing Manager***
Preconditions	The Manager opens the application on their device and is associated to a Store.
Flow of Events	1. The Manager clicks on the interested Store 2. The Manager clicks on the generate scanner button 3. The System generates a unique code, the scanner code associated with the Queue Manager of the Store. 4. The Manager can send this code to their clerks/machines. 5. The clerks with the code can access a specific application and input it in the specific editable text box. 6. The clerks can now use this app to scan the QR-codes of the Customers. 7. Each scan can influence the queue status.
Post-Conditions	The Store is able to scan the tickets of the Customers.
Exceptions	1. If the code input in the app is wrong, the System outputs a failure message on the screen.

UC.11: Activate Paper Ticket

Description	A Manager activates a paper-ticket service associated with one of their Stores.
Actors	***Manager***, ***Queue Manager***
Preconditions	The Manager opens the application on their device and is associated to a Store.

Flow of Events	1. The Manager clicks on the interested Store 2. The Manager clicks on the generate paper ticket button 3. The System generates a unique code, the paper-ticket code associated with the queue of the Store. 4. The Manager can use this code in the apposite machine, to set up a paper ticket service 5. The Consumers can now get tickets from this machine. 6. Each tickets influences the queue status.
Post-Conditions	The Store is able to produce paper tickets for the Consumers.
Exceptions	1. If the code input in the app is wrong, the System outputs a failure message on the screen.

UC.12: Update Store Info

Description	A Manager wants to update Store information.
Actors	***Manager***, ***Queue Manager***
Preconditions	The Manager opens the application on their device and is associated to a Store.
Flow of Events	1. The Manager clicks on the interested Store 2. The Manager clicks on the modify information button. 3. The Manager fills the editable fields with the updated information and confirms the operation through a button. (Simple Manager and Owner, have different editable fields). 4. If the Manager has edited information which are linked to the Queue Manager (max number of people, opening hours, amount of days in advance in which a Visit can be booked), the System logs a message: these changes can be uploaded within 24 hours, taking a chance on canceling some Visits previously booked, or can be uploaded in the current amount of days in advance in which a Visit can be booked, without losing any Visit. 5. The Queue Manager handles the changing of the information linked to the queue in the due time. 6. The System sends notifications to Customers if their Visits have been cancelled.

Post-Conditions	The Manager successfully updates the Store information.
Exceptions	1. If the Manager inputs some invalid information, the operation fails and is cancelled and the System logs a failure message on the screen.

3.3 Performance Requirements

The main focus of the application is relative to queue mechanism. When an user wants to equeue, once filled all the information for ticket requests, the System should respond in 5 seconds. The Customer must interact with application accepting or discarding the ticket in Time to Reserve (D.29). If the time expires, the ticket is automatically deleted. Regarding the booking of a Visit, the System should respond in 30 seconds. This constrain is relaxed in order to prioritize the managing of tickets that needs high responsivity and because there is no need of Customer interaction.

3.4 Design Constraint

3.4.1 Standards Compliance

The System doesn't need to be compliant to any standard since it doesn't interact with standardized systems or any sensitive information.

3.4.2 Hardware Limitations

In order to run the application, at least is required

1. 3G connection at 1 Mbps
2. 50 MB of memory space
3. 2 GB of RAM
4. GPS sensor

3.5 Software System Attributes

3.5.1 Reliability

The System should guarantees a 99% of reliability: its components must have a failure rate that guarantees this goal.

3.5.2 Availability

All the minor patches are released during the night, in order to minimize down time. The System will be available 24/7 and must guarantee 99.9% of availability, with at most 1.44 minutes of downtime per day.

3.5.3 Security

User credential are stored inside the DBMS using a randomly generated salt and possibly a slow hash function. All the communication between client and server are done over HTTPS protocol.

3.5.4 Maintainability

All the code should be written using good software engineering practice such as exhaustive documentation of the code, unit test and use of design patterns.

3.5.5 Portability

The application should be developed using cross-platform technologies that allows to develop Native application for better UX. Also the server side of the application should be developed using cross-platform technologies.

4 Formal Analysis Using Alloy

This section is dedicated to the description of the model through the use of the Alloy modeling language, which allows us to describe the world in which our application lives.

4.1 World Generated: main characteristics

In the Figure 15 it is possible to observe the main aspects of the System and its relations. In order to show also the details of the world, the various components will be examined separately in the next sections.

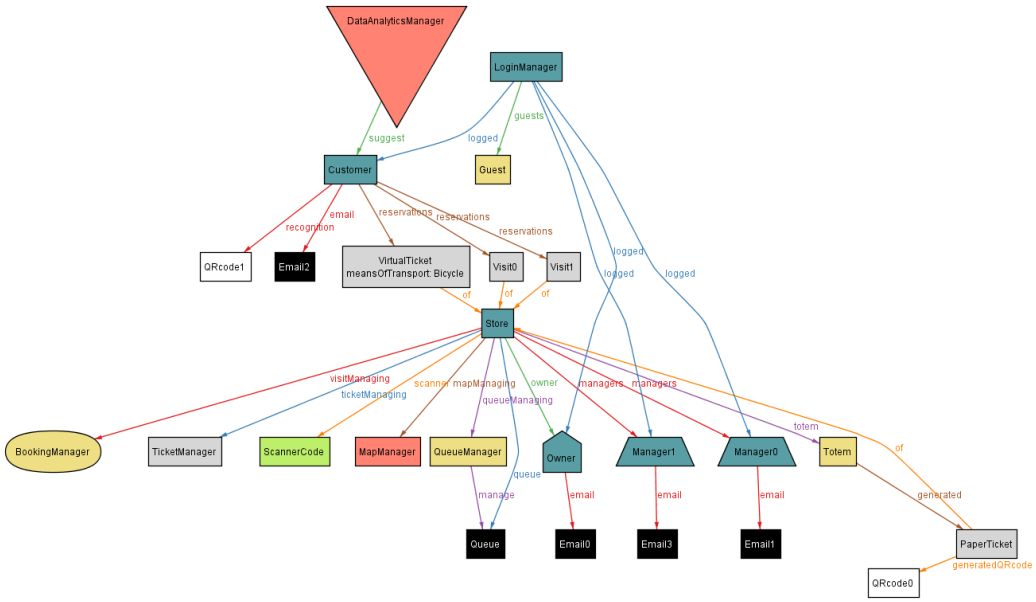

Figure 15: Alloy-generated world that shows the most important interactions.

4.1.1 Login Manager and Data Analytics Manager

The Login Manager is in charge of providing the login functionality. each Guest must enter their login credentials to interface with the app features. A logged in User can be a Manager or a Customer. Each User has its own personal Email. Each Customer is identified by his personal unique QRCode, and has his own Reservations. Moreover, each Customer will receive suggestions from the Data Analytics Manager, which is in charge of analyze data about Customers. All this details are shown in the Figure 16.

4.1.2 Customers and their Reservations

As we can see in the Figure 17 Customers are identified by a unique QRCode. Each Customer can have some Reservations planned, which can be a Visit or a *VirtualTicket*. Every Visit is characterized by the Day of the Visit, the *TimeSlot* for that Visit, a List of Groceries that the Customer wants to buy and a Duration. Each *VirtualTicket* is chacterized by a time of entrance and a duration. Moreover each Reservation refers to one Store. A single Customer cannot reserve more than 3 Visits in the same Day for the same Store.

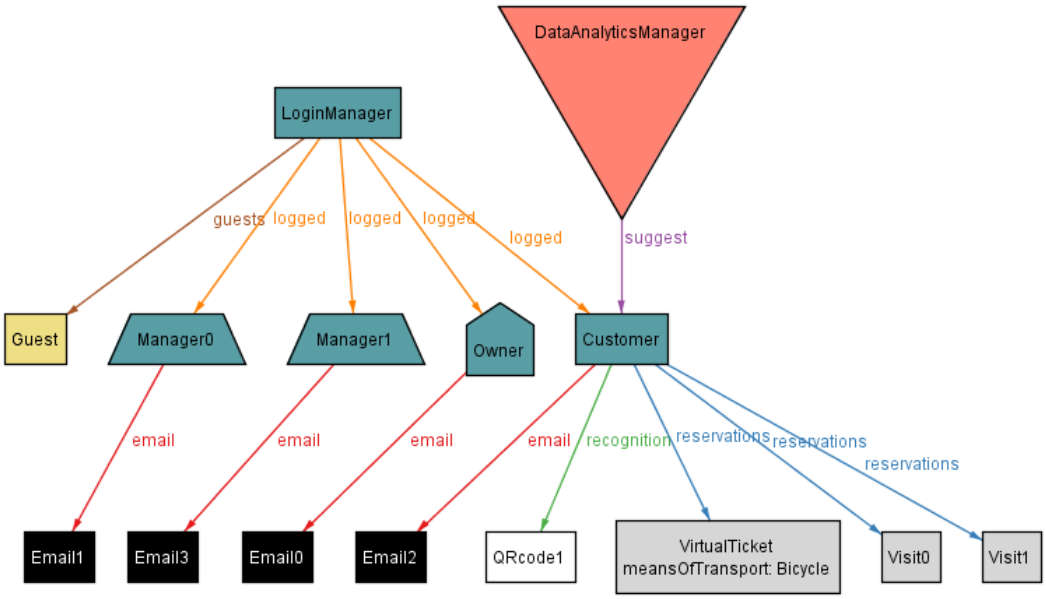

Figure 16: Alloy-generated world that shows the Login Manager and the Data Analytics Manager functionalities.

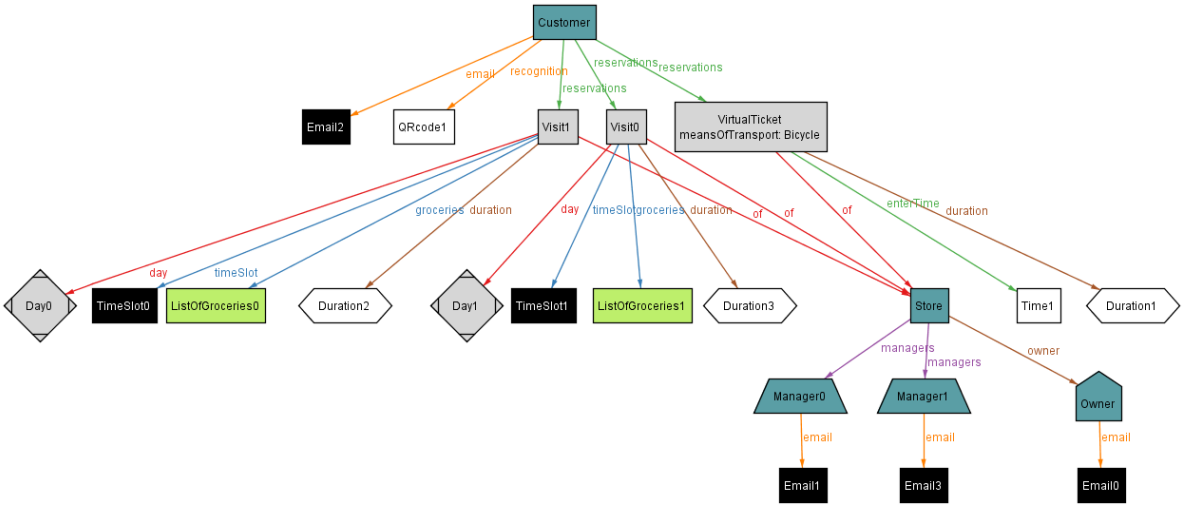

Figure 17: Alloy-generated world that shows Customers and Reservations characteristics.

4.1.3 Stores and their components

Each Store has one and only one Owner, but it can have other Managers who collaborate in the management of the Store. Moreover each Store has its own *BookingManager*, *TicketManager* and *MapManager*. This components are made in order to manage all the requests of Tickets and Visits for that Store. It also has a *ScannerCode* and a *Totem*. The Totem has the task of generating *PaperTickets* for that particular Store, which will be characterized by a unique QRcode. The Reservations made for a Store are stored in the personal unique Queue of the Store, which is managed by the Queue Manager. A Queue has a certain number of Temporal Quantums, and each Temporal Quantum has a number of spaces equal to the capacity of the Store.

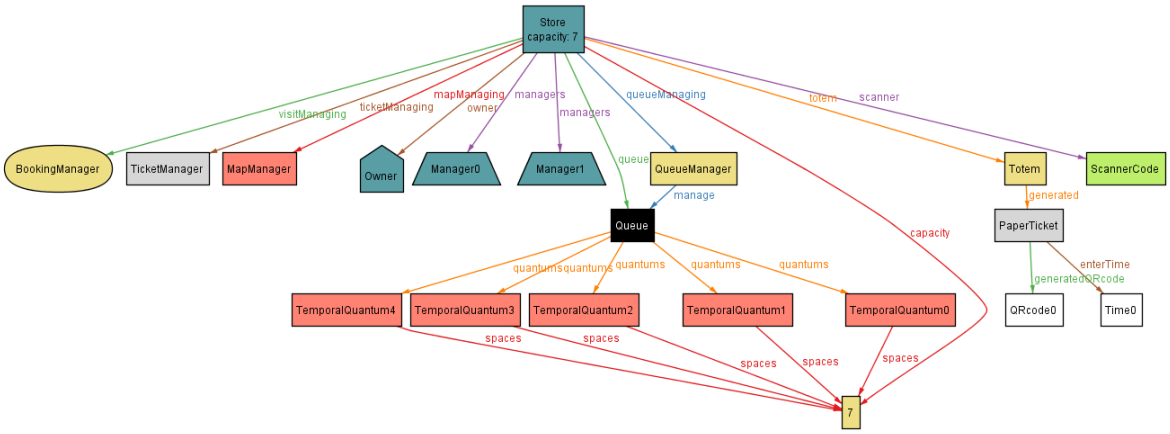

Figure 18: Alloy-generated world that shows Stores and their components.

4.1.4 The Whole World Generated

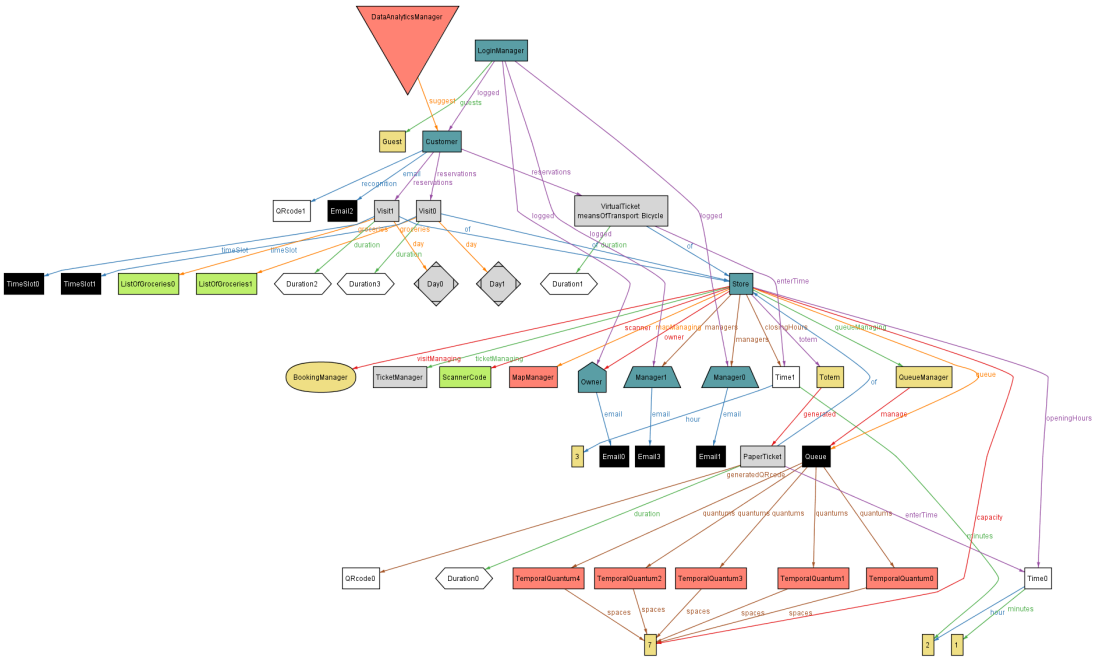

Figure 19: Alloy-generated entire world.

In Figure 19 there is the representation of the whole world generated by Alloy, by running the following predicate:

```
pred show{
        #Guest = 1
        #Queue = 1
        #TemporalQuantum = 5
        #Totem = 1
        #ScannerCode = 1
        #Manager = 3
        #Visit = 2
        #VirtualTicket = 1
        #PaperTicket = 1
        #Customer = 1
}

run show for 20
```

4.2 Model Check

```
-- All VirtualTickets and Visits belong to a Customer, all PaperTickets are generated by
    ↪ a Totem --
assert noAloneReservations{
        all vt: VirtualTicket | one c: Customer {
                vt in c.reservations
        }
        all v: Visit | one c: Customer{
                v in c.reservations
        }
        all pt: PaperTicket | one t: Totem{
                pt in t.generated
        }
}

-- There are no more than 3 Visits per day for only one Customer --
assert maxThreeVisitPerDayForACustomer{
        all d: Day | all c: Customer | #(c.reservations & day.d) ≤ 3
}

-- Each Email is of one and only one User and is unique for each User --
assert uniqueEmail{
        all disj u1,u2: User | u1.email ≠ u2.email
        all u: User | #u.email = 1
}

-- Unique QR code --
assert uniqueQRcode{
        //There are no QRCodes in common between Customers and PaperTickets
        all disj c: Customer | all disj pt: PaperTicket { c.recognition ≠ pt.
            ↪ generatedQRcode }
        //Each PaperTicket has a unique QRcode
        all disj c1,c2: Customer | c1.recognition ≠ c2.recognition
        //Each Customer has a unique QRcode
        all disj p1,p2: PaperTicket | p1.generatedQRcode ≠ p2.generatedQRcode
}

-- Capacity and number of spaces in each TemporalQuantum of a Store --
assert capacityAndSpaces{
        all s: Store | s.capacity = s.queue.quantums.spaces
}

-- Unique Components for each Store --
assert componentsStore{
        all s: Store | #(s.ticketManaging) = 1 ∧ #(s.visitManaging) = 1 ∧ #(s.
            ↪ queueManaging) = 1 ∧ #(s.mapManaging) = 1
}

-- Each Queue is of one and only one Store and is unique for each Store --
assert uniqueQueue{
        all disj s1,s2: Store | s1.queue ≠ s2.queue
        all s: Store | #s.queue = 1
}

-- Each Store has only one Owner --
assert oneOwner{
        all s: Store | #s.owner = 1
}
```

The results of the running of all the assertions above can be found in the Figure 20. As you can see, no counterexample has been found.

Figure 20: Results for running all the assertions made. No counterexample has been found.

4.3 Whole Alloy Model

```
-- SIGNATURES AND FACTS --
abstract sig MeansOfTransport { }
one sig Car extends MeansOfTransport { }
one sig Walk extends MeansOfTransport { }
one sig PublicMeans extends MeansOfTransport { }
one sig Bicycle extends MeansOfTransport { }

one sig LoginManager{
        logged: some User,
        guests: some Guest
}

one sig DataAnalyticsManager{
        suggest: set Customer
 }{
        (#DataAnalyticsManager > 0) <=> (#Customer > 0)
}

sig TicketManager{ }{
        one s:Store | this in s.ticketManaging
}

sig BookingManager{ }{
        one s:Store | this in s.visitManaging
}

sig QueueManager{
        manage: one Queue
 }{
        one s:Store | this in s.queueManaging
}

sig MapManager{ }{
        one s:Store | this in s.mapManaging
}

sig  Duration{ }{
        //Each Duration belongs to a Reservation
        one r:Reservation | this in r.duration
}

sig ListOfGroceries{ }{
        //Each ListOfGroceries belongs to a Visit
        one v:Visit | this in v.groceries
        //All Visits have different ListOfGroceries
        all disj v1,v2:Visit | v1.groceries != v2.groceries
}

sig Day{ }{
        //Each Day belongs to a Reservation
        one r:Reservation | this in r.day
}

sig TimeSlot{ }{
        //Each TimeSlot belongs to at least one Visit
        one v:Visit | this in v.timeSlot
}

sig Time{
        hour: one Int,
        minutes: one Int
}

sig Char{ }{
        //Each Char belongs to an email
        one e:Email | this in e.email
}

sig Guest { }{
        //Each Guest has to be managed by a LoginManager
        one lm:LoginManager | this in lm.guests
}
```

```
//Email as a sequence of char
sig Email{
     email: seq Char
}{
        //Each Email belongs to a User
        one u:User | this in u.email
}

sig QRcode{ }{
        //Each QRcode belongs to a Customer or to a PaperTicket
        one c:Customer | one pt:PaperTicket{
                this in c.recognition ∨ this in pt.generatedQRcode
        }
        //There are no QRcode in common
        all disj c:Customer | all disj pt:PaperTicket{
                #(c.recognition & pt.generatedQRcode)=0
        }
        all disj p1,p2:PaperTicket | p1.generatedQRcode ≠ p2.generatedQRcode
        all disj c1,c2:Customer | c1.recognition ≠ c2.recognition
}

abstract sig Reservation {
        duration: one Duration,
        of: one Store
}

abstract sig Ticket extends Reservation{
        enterTime: one Time
}

sig PaperTicket extends Ticket {
        generatedQRcode: one QRcode
}{
        // Each Ticket is associate to one and only one Consumer
        one t:Totem | this in t.generated
}

sig VirtualTicket extends Ticket {
        meansOfTransport: one MeansOfTransport
}{
        // Each VirtualTicket is associate to one and only one Customer
        one c:Customer | this in c.reservations
}

sig Visit extends Reservation {
        day: one Day,
        timeSlot: one TimeSlot,
        groceries: one ListOfGroceries
}{
        // Each Visit is associated to one and only one Customer
        one c:Customer | this in c.reservations
}

sig ScannerCode {}{
        //Each ScannerCode is of a Store
        one s:Store | this in s.scanner
}

sig Queue {
        quantums: some TemporalQuantum
} {
        // Foreach queue, each quantum has the same spaces
        all q1, q2: quantums | q1.spaces =q2.spaces
        // Each queue has to be associated to one and only one Store
        one s:Store | this = s.queue
        //queueManager of the Store manage manage only the queue of the that Store
        all s:Store | s.queueManaging.manage = s.queue
}

abstract sig User {
        email: one Email
}{
        //Each User has to be logged by a LoginManager
        one lm:LoginManager | this in lm.logged
}
```

```
sig Customer extends User{
        reservations: some Reservation,
        recognition: one QRcode
}{
        // Each customer has at most 3 visits each day
        all d:Day | #(reservations & day.d) ≤ 3
        //each customer receives suggestions from the DataAnalyticsManager
        all d:DataAnalyticsManager | this in d.suggest
}

//Check that no one has an email used by another Customer
fact emailUnique{
        all u, u1: User |  (u.email = u1.email) iff (u = u1)
}



sig Manager extends User {}{
        one s:Store | this in s.managers ∨ this in s.owner
}

sig Owner extends Manager {}{
        // Each owner associated to a one and only one Store
        one s:Store | this =s.owner
}

sig Store {
        owner: one Owner,
        managers: set Manager,
        scanner: one ScannerCode,
        totem: some Totem,
        capacity: one Int,
        queue: one Queue,
        openingHours: one Time,
        closingHours: one Time,
        ticketManaging: one TicketManager,
        visitManaging: one BookingManager,
        queueManaging: one QueueManager,
        mapManaging: one MapManager
}{
        // Queue available places equal to the capacity of the associated store (for each
            ↪  store)
        queue.quantums.spaces = capacity
        // for each store capacity has to be greater than 0
        capacity > 0
        // For each Store, the owner is not also a manager.
        owner not in managers
        // Each store must close after opening
        openingHours.hour < closingHours.hour
}

sig TemporalQuantum{
        spaces: one Int
}{
        // Each Temporal quantum is associated to one and only queue
        one q:Queue.quantums | this = q
}

sig Totem {
        generated: set PaperTicket
}{
        //Each Totem is of a Store
        one s:Store | this in s.totem
}


-- ASSERTIONS --

-- All VirtualTickets and Visits belong to a Customer, all PaperTickets are generated by
    ↪ a Totem --
assert noAloneReservations{
        all vt:VirtualTicket | one c:Customer {
                vt in c.reservations
        }
```

```
        all v: Visit | one c: Customer{
                v in c.reservations
        }
        all pt: PaperTicket | one t: Totem{
                pt in t.generated
        }
}

-- There are no more than 3 Visits per day for only one Customer --
assert maxThreeVisitPerDayForACustomer{
        all d: Day | all c: Customer | #(c.reservations & day.d) ≤ 3
}

-- Each Email is of one and only one User and is unique for each User --
assert uniqueEmail{
        all disj u1,u2: User | u1.email ≠ u2.email
        all u: User | #u.email = 1
}

-- Unique QR code --
assert uniqueQRcode{
        //There are no QRCodes in common between Customers and PaperTickets
        all disj c: Customer | all disj pt: PaperTicket { c.recognition ≠ pt.
            ↪ generatedQRcode }
        //Each PaperTicket has a unique QRcode
        all disj c1,c2: Customer | c1.recognition ≠ c2.recognition
        //Each Customer has a unique QRcode
        all disj p1,p2: PaperTicket | p1.generatedQRcode ≠ p2.generatedQRcode
}

-- Capacity and number of spaces in each TemporalQuantum of a Store --
assert capacityAndSpaces{
        all s: Store | s.capacity = s.queue.quantums.spaces
}

-- Unique Components for each Store --
assert componentsStore{
        all s: Store | #(s.ticketManaging) = 1 ∧ #(s.visitManaging) = 1 ∧ #(s.
            ↪ queueManaging) = 1 ∧ #(s.mapManaging) = 1
}

-- Each Queue is of one and only one Store and is unique for each Store --
assert uniqueQueue{
        all disj s1,s2: Store | s1.queue ≠ s2.queue
        all s: Store | #s.queue = 1
}

-- Each Store has only one Owner --
assert oneOwner{
        all s: Store | #s.owner = 1
}


-- PREDICATE --

pred show{
        #Guest = 1
        #Queue = 1
        #TemporalQuantum = 5
        #Totem = 1
        #ScannerCode = 1
        #Manager = 3
        #Visit = 2
        #VirtualTicket = 1
        #PaperTicket = 1
        #Customer = 1
}


-- EXECUTION --

run show for 20
```

5 Effort Spent

Lorenzo:

	Description	Hours
1	First layout project creation.	0.2
2	Revision recap first call.	0.5
3	Revision and insertion of the scopes .	2.5
4	Multiple calls	8
5	requirements and update requirements labels	3
6	Use cases	7.8
7	Refactor different parts of the document	1
8	Small revisions	0.5
9	Alloy	4
10	Sequence diagrams	1.5
11	Goal to req and dom Matrix	0.75
12	Update UC	2
13	Small updates	1
14	Final Call first revision	3
15	Update UC	0.5
16	**TOTAL**	36.25

Yasmin:

	Description	Hours
1	Write down recap first call	1.5
2	First draft some GUI	1
3	Introduction and Purpose	1.5
4	Goals draft	1.5
5	Requirements draft	2.5
6	Revision	1.5
7	GUI	4.5
8	Multiple calls	8
9	Queue Requirements	1.2
10	User Characteristics	0.5
11	Interfaces chapters	0.6
12	Class Diagram	1
13	Alloy modeling and chapter	13
14	Final Call first revision	3.5
15	Images and chapters polishing	0.6
16	**TOTAL**	42.4

Giovanni:

	Description	Hours
1	Multiple calls.	8
2	Final Call first revision	3.5
3	First draft of goals and requirements	3
4	Writing Introduction	1
5	Setting up of whole document structure and writing small subsection (Reference documents, document structure, definitions etc..)	3.5
6	Writing Production functions and perspective	4
7	Product function and perspective diagrams	0.3
8	Use cases diagrams	0.3
9	Performance requirements and design constraints	1.5
10	Alloy	1.5
11	Other work revision and small refactor	3
12	**TOTAL**	29.6