POLITECNICO
MILANO 1863

Software Engineering 2
A.Y. 2020-2021

CLup – Requirements Analysis and Specification Document

Vincenzo Riccio, Giancarlo Sorrentino, Emanuele Triuzzi

December 22, 2020

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Vincenzo Riccio, Giancarlo Sorrentino, Emanuele Triuzzi
Version:	1.0
Date:	December 22, 2020
Download page:	https://github.com/SirGian99/RiccioSorrentinoTriuzzi
Copyright:	

Contents

Chapter 1

Introduction

1.1 Purpose

CLup aims to provide chains of stores with a reliable solution to the problem of people gathering inside and outside the shops.

To face the problem, the application focuses on its principal causes, which are the management of people inside the store, that often leads to overcrowding, the effectiveness of standard queuing systems and the way people are allowed to visit the stores. Moreover, the system aims to provide a useful tool for store managers in order to help them in administering stores and monitoring their status. In particular, the main goals that CLup aims to achieve, summarized in the table below, are the following:

- Prevent stores from being overcrowded, in order to avoid indoor gatherings while maximizing their occupancy;
- Reduce gatherings of people outside the stores waiting to enter them;
- Provide a more efficient way to access stores, reducing the time customers waste while waiting to do it;
- Help store managers in monitoring the status of the store and regulating the influx of people.

Goals	
G1	Prevent stores from being overcrowded while maximizing their occupancy
G2	Reduce gatherings of people outside the stores waiting to enter them
G3	Reduce the time customers waste while waiting to access the stores
G4	Help store managers in monitoring the status of the store and regulating the influx of people

Table 1.1: Goals

1.2 Scope

During the current situation of emergency, it is fundamental to prevent contacts among people. For this reason, governments impose strict rules concerning social distancing, both for indoor and outdoor contexts.

However, crowding management inside stores like supermarkets and grocery shops could be challenging. Currently, stores limit the maximum number of people allowed, and therefore long queues arise: entering a store for a few minutes might even require hours. Moreover, customers who see a crowded store might avoid lining up to save time and prevent contact with others.

CLup fits into this context allowing customers to remotely line up in a queue of a given store and to be notified when they should head toward it. Furthermore, it allows the customer to book a visit for a store on a specific day and time, which grants him priority over the queued customer.

Users can interact with CLup thanks to two distinct interfaces: one is an easy-to-use application designed for the customers, while the other one is an administrative tool that allows store managers to monitor their stores and modify their parameters.

Moreover, CLup also provides physical proxies outside the stores as a fallback option for customers who want to line up but do not have access to the application.

1.2.1 World Phenomena

The following table illustrates the phenomena that happen in the real world and affect the system, which cannot control or detect them.

World phenomena	
WP1	A person reaches the store
WP2	Some people gather in a specific area of the store
WP3	People wait in front of the store
WP4	The store opens
WP5	The store closes

Table 1.2: World phenomena

1.2.2 Shared Phenomena

The following table illustrates the phenomena that happen in the real world and can be observed or managed by the system. The following notation is used:

- *MC* for machine controlled phenomena (observed by the world)
- *MO* for machine observed phenomena (controlled by the world)

Shared phenomena		
SP1	A customer joins the queue from the application	*MO*
SP2	A customer joins the queue using the physical proxy	*MO*
SP3	An alert is sent to the customer when he should reach the store	*MC*
SP4	A customer cannot reach the store in time anymore	*MO*
SP5	A customer is allowed to enter the store	*MC*
SP5.1	A customer enters the store	*MO*
SP6	A customer leaves the store	*MO*
SP7	A customer request to book a visit to the store	*MO*
SP7.1	A customer specifies the estimated duration of the visit	*MO*
SP7.2	A customer specifies which kind of products wants to buy	*MO*
SP7.3	The system suggests booking alternatives to the customer	*MC*
SP7.4	The system accepts a booking request	*MC*
SP7.5	The system rejects a booking request	*MC*
SP8	The system notifies the customer when specific stores are becoming unavailable	*MC*
SP9	The store manager specifies the maximum occupancy of the store	*MO*
SP10	The store manager specifies the maximum occupancy of the product sections	*MO*
SP11	The manager monitors the status of the store	*MO*

Table 1.3: Shared phenomena

1.2.3 Definitions, acronyms, abbreviations

Definitions, acronyms, abbreviations	
WP*x*	World phenomena number x, according to the table of world phenomena

SPx	Shared phenomena number x, according to the table of shared phenomena
CLup	Also known as the system. It is the software to be developed
Customer application	Also known as application. It is used to access the functions provided by CLup
Administrative tool	The tool provided to store managers in order to administer stores
Proxy	The physical fallback option for customers that want to use CLup but cannot use the application. It is placed outside the store it refers to
Turn Announcement System	An external system which informs customers about who has been allowed by CLup to enter the store
Access Management System	An external system which regulates physical entrances and exits to the store it is associated with by interacting with CLup
App-customer	A customer who uses CLup functions through the application
Proxy-customer	A customer who uses CLup functions through the proxy
Long-term customer	With respect to a certain store, a customer who already used CLup to visit it
Current occupancy	Also known as occupancy. It can be referred to the store or one of its sections. It is the number of people inside it
Maximum occupancy	Refers to the store or one of its sections. It is the maximum number of people allowed to be in that area
Virtual queue	Also known as access queue or simply queue. It represents the set of customers who lined up through the app or the proxy
Line up	With respect to a customer and a store, it is the event of joining the queue
Visit request	A customer's request to visit a store. It can be either a line-up request or a booking request

Line-up request	A request made by the customer to line up for a store
Booking request	A request made by the customer to book a visit for a store
Visit	The realization of a visit request which takes place when a customer enters the store. After the customer exits the store, we talk about *completed visit*, otherwise it is a *visit in progress*
Visit token	A unique token bound to a visit request. It allows the Customer to enter and exit the store

Table 1.4: Definition, acronyms, abbreviations

1.2.4 Revision history

1.0 - First version of the document (December 22, 2020)

1.2.5 Reference documents

- IEEE standard for Software Requirements Specifications, IEEE 29148–2018;
- R&DD Assignment A.Y. 2020–2021;
- Teaching material provided by professors Matteo Rossi and Elisabetta Di Nitto.

1.2.6 Document structure

The reference structure used for the document is the one suggested by professor Matteo Rossi of Politecnico of Milan. It is derived from the IEEE standard, which is used as a reference document (IEEE standard for Software Requirements Specifications, IEEE 29148–2018).

Chapter 1

Chapter 1 is an introduction to the software to be designed and developed and to the problem that it addresses. It presents the goals that should be achieved and an analysis of the context in which the system will be placed.

Chapter 2

Chapter 2 is a more detailed description of the system to be realized, focused especially on a more detailed description of the context, e.g. presenting scenarios and the actors involved, on the product functions and on its requirements. Furthermore, it contains explicit constraints, dependencies and domain assumptions.

Chapter 3

Chapter 3 includes specific requirements, with a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. It includes the description of all the functional and nonfunctional requirements, together with the description of the external interfaces, of the use cases and of possible design constraints.

Chapter 4

Chapter 4 is a formal analysis of the system using the Alloy model, showing also some worlds obtained running the model.

Chapter 5

Last chapter contains a report on the effort spent by all the members of the group while writing the current document.

Chapter 2

Overall description

2.1 Product perspective

2.1.1 UML Class Diagram Description

The provided UML Class Diagram is intended to provide a high-level model of the reality in which CLup is collocated. The following description aims to better highlight the main characteristics of the model.

CLup manages stores of different chains or autonomous ones. In particular each store, which is composed of different product sections, is associated with its managers and with the chain it belongs to, if there is one. A store is also characterized by its name, its address, its maximum occupancy, its working hours and its product sections, which are parameters specified by store managers. Moreover, other significant attributes inferred by the system are the store current occupancy and the average time customers spend inside it. Each product section is identified by its name and is characterized by a current occupancy and by a maximum occupancy.

Customers can visit the store by making a visit request, and are divided into app-customers and proxy customers. More precisely, a customer is identified as a "different" proxy customer every time he makes a line-up request through the proxy. Indeed, since the proxy is meant to be an easy-to-use fallback option, it is not intended to be able to identify customers.

A visit request is made for the desired store and has different attributes, such as the number of people willing to visit the store and the token that will allow them to visit it. CLup provides two kinds of visit requests: line-up requests and booking requests.

Each store maintains a queue of line-up requests, characterized by its length and its estimated disposal time, and holds booking requests made by app-customers.

Booking requests are characterized by attributes such as the date and time requested by the customer and the expected duration of the visit. Moreover, each booking request can also be associated with the store product sections that the app-customer has specified while placing the booking.

The realization of a visit request is represented by a visit, whose starting time represents the time when the customer who made the request entered the store. This means that a line-up request in the queue of the store it refers to is never associated with a visit and, vice versa, a line-up request which is not in the queue is always associated with a visit. When the customer exits the store, the same visit is considered completed.

Further details on the behaviour of the system are specified in the following sections of the document.

2.1.2 UML Class Diagram

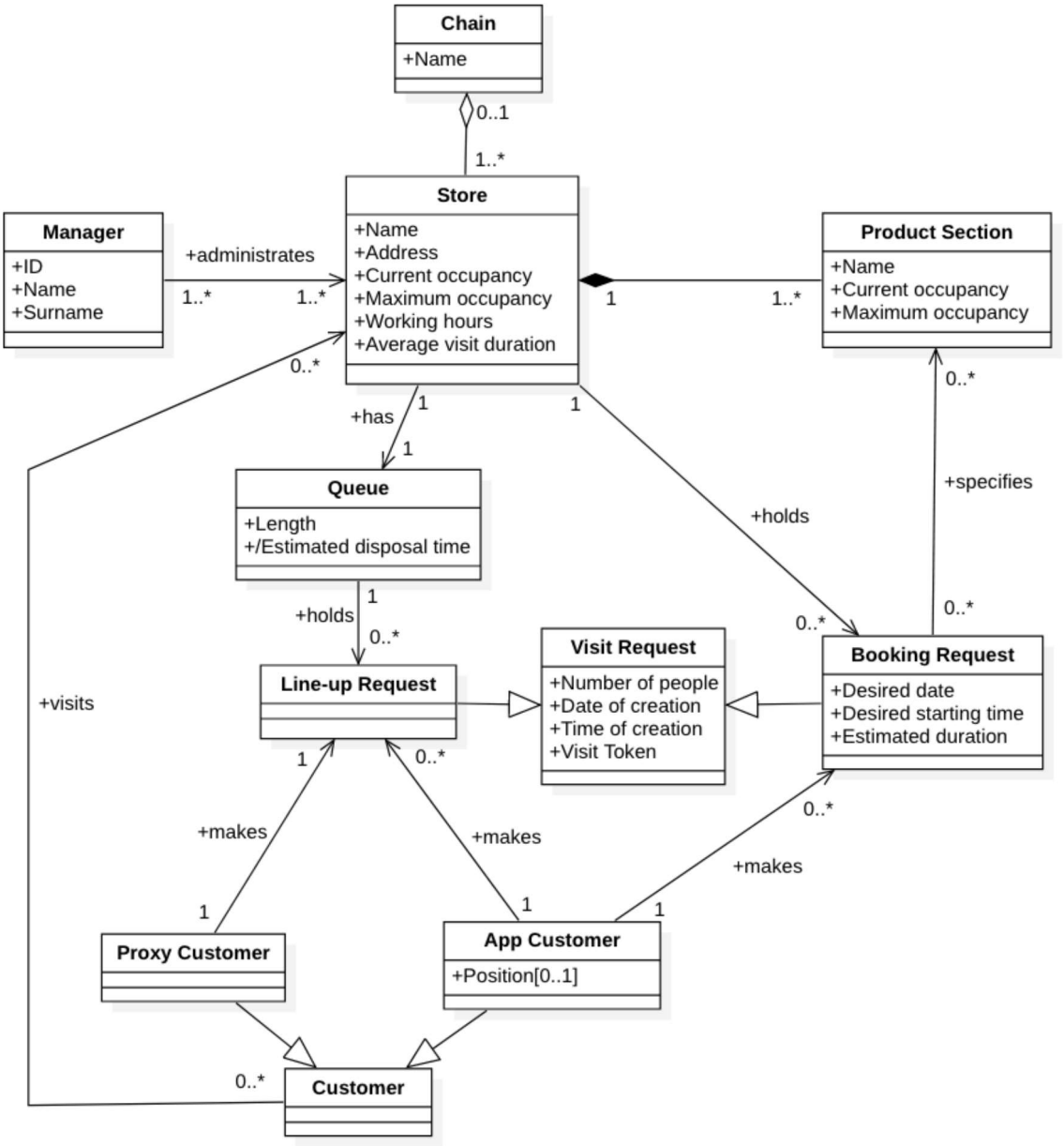

Figure 2.1: UML Class Diagram

2.1.3 State charts

The following state charts are useful/meant to better explain the status of some classes in the UML class diagram.

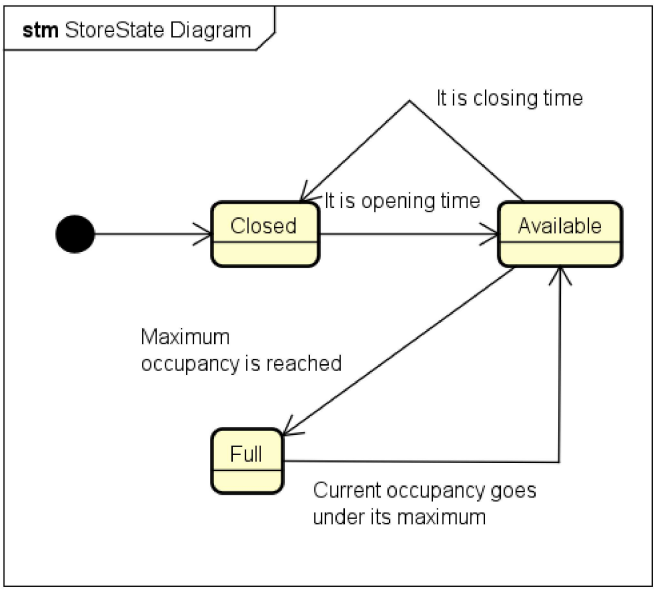

Figure 2.2: State chart 1 – "Store"

The state chart of the store.

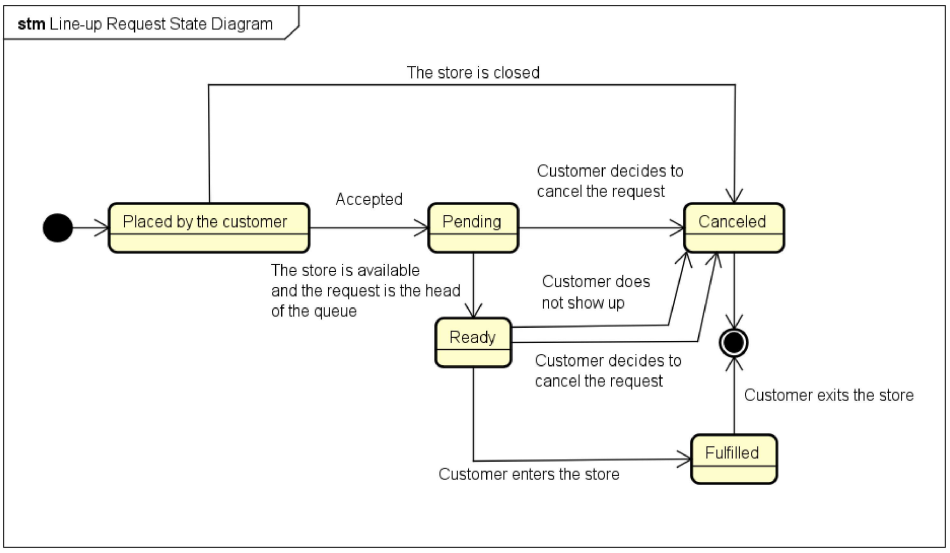

Figure 2.3: State chart 2 – "Line-up request"

The state chart of a line-up request.

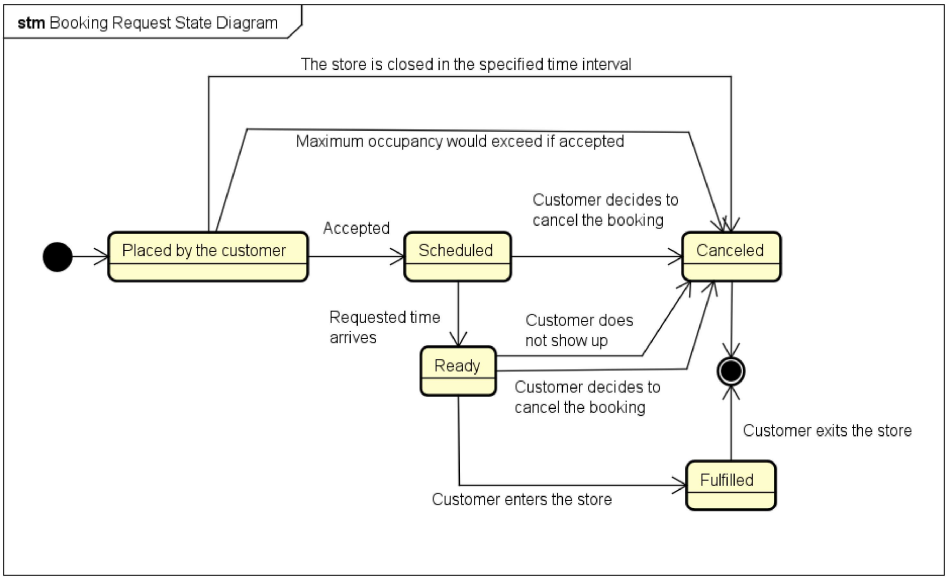

Figure 2.4: State chart 3 – "Booking request"

The state chart of a visit request.

2.1.4 Scenarios

Remotely line up for a store

Alice is at home and wants to do some grocery shopping at the store at the end of the street. She opens CLup and checks the store's current queue and the estimated time required to enter. She decides to get in line and starts doing something else. After some time, she receives a notification that suggests her to start heading the store. Thus, she reaches the store. Once there, she opens the app and notices that it is her turn, so she retrieves the visit token in the app in order to get access to the store. When she is done with her shopping and payment, she opens the app in order to exit the store.

Line up for a store using the proxy

Bob is a 70-year-old man who is not so keen on using a smartphone, so he has not got any. He has to buy something at the minimarket near his house. He uses the proxy located outside the store to line up and to receive a visit token, and then waits along the street waiting for his turn to come. When it is his turn, he is informed by the Turn Announcement System of the store and enters it by showing the visit token to the Access Management System. When he is done with his shopping and payment, he uses the visit token in order to exit the store.

Booking a visit for a store

Chuck is at work and he wants to do some shopping before coming back home. Since there are often many people in line to enter the store he wants to go to, he opens the CLup application and books a visit to the store for a specific time interval. Before reaching the store he must pick up his son from school, so he specifies that two people will visit it.

When the time arrives, they have priority over the people in the queue. Since it is their turn, Chuck opens the app in order to get access to the store. After the payment, he opens the app in order to exit the store.

Active suggestions for long-term customers

Isaac wants to book a visit for Saturday at noon at his local shop. Since he is a long term customer of that chain of stores, CLup provides him with an estimated duration of his visit, based on data about his previous visits. However, the system also informs him that, for the selected time interval, the store is often crowded, and so suggests to him different alternatives. Sadly, George can only do shopping in the indicated one, so he places his booking anyway.

Booking alternatives are suggested

Emanuele lives near a store in "via Rubattino" and wants to do some grocery shopping next Saturday morning at 11 a.m. However, that store is already at its maximum occupancy in the selected time interval due to previous bookings by other users. So, when Emanuele tries to place a booking request, the system provides him with two alternatives: the first one is to book a visit on the same day at 6 p.m., when the store, according to bookings and statistics, will be less crowded; the second one is to book a visit for the same time interval in another store of the same chain, which has not reached its maximum occupancy yet. Emanuele chooses an option and fulfils his need.

There is always a way out

Giancarlo is a customer who is currently visiting a store. While in the store, he notices that his phone ran out of battery. After he is done with his shopping, he asks for the help of a store manager, who manually allows him to exit the shop. The occupancy of the shop is then automatically updated.

Forgetting about the visit

Vincenzo is a forgetful man. When his turn comes, he is not outside the store. CLup detects Vincenzo is not showing up and allows another customer to enter.

2.2 Product Functions

CLup is intended to manage different chains of stores and autonomous stores. Each store has at least one manager who can specify, through the administrative tool, different parameters: the name, the address, the working hours, the store maximum occupancy and its product sections (and their relative maximum occupancies). Furthermore, managers can also use the tool to monitor the status of the shop, which includes:

- the number of people inside;
- the visits which are actually in progress;
- the average duration of a visit;

- the actual length and estimated disposal time of the queue;
- all the future booking requests.

CLup regulates the influx of people in a store to prevent it from being overcrowded. To do so, the system provides a way to authorize the entrances to each shop with respect to the number of people inside it.

The system uses unique visit tokens to authorize each customer to enter and exit a store. If the customer uses the application, the visit token will be available in the application itself. Instead, each time a customer uses the proxy of a given store, a visit token is physically provided by the proxy itself. All the accesses are granted by CLup with respect to the store maximum occupancy.

In order to reduce the possibility of queues forming outside the stores it manages, CLup provides the possibility to remotely line up in the queue of the desired store. A customer can join the queue either by using the application offered by CLup or by means of the proxy placed outside the store.

Furthermore, CLup alerts app-customers when they need to reach the store they requested a visit for, taking into account the time they need to get to the shop from their current position with respect to the time at which they will be authorized to enter it. Eventually, the system informs customers when they are allowed to enter the store through the Turn Announcement System and, for app-customers, also through the application.

An app-customer can also use the application to book a visit to a store. Doing so, he has to specify when he wants to reach it, the expected duration of the visit and, if he wants, which kind of products he intends to buy. In the case of a long-term customer, the duration of the visit can also be estimated by CLup taking into account his previous visits to the selected store.

All these data are used by the system to manage who can enter each store in a specific time interval.

When a booking is placed, the system can also decide to reject it if, across the specified time interval, the bookings placed by other customers already maximize the occupancy of the selected store. The system also tries to balance the number of customers across all the working hours of the store. In both cases, it suggests alternative time intervals or alternative stores of the same chain when a customer is booking a visit.

CLup is able to manage the case in which, for any reason, a customer does not show up while he is allowed to enter the store he requested a visit for, while other people are waiting to enter it. Moreover, app-customers can always cancel their visit requests (both line-up and booking requests), if needed.

App-customers can choose to be notified about the availability of specified stores. Indeed, a customer can specify one or more recurrent time intervals in which he is interested in visiting those stores. These intervals can also be inferred by the system based on the customer's attitudes. The system will notify him if, during one of those time intervals, the store of interest is reaching its maximum occupancy. Doing so, the customer can take the opportunity to visit it by booking or lining up.

CLup monitors customer attitudes also to estimate the average visit duration of each store.

2.3 User characteristics

Store manager

The store manager administers his stores by checking their status and modifying their parameters, such as the working hours and maximum occupancy. He can also specify He also specifies the different product sections that compose the store and their relative maximum occupancies. He can also keep track of the real-time store occupancy, and might let people in and out of the store, if needed.

Customer

The customer uses CLup in order to visit a store of a given chain. To do so, he can either line up in the CLup queue by means of the application or the proxy, or also book a visit through the application.

Access Management System

The Access Management System is the system which regulates physical entrances and exits a specific store. It communicates with CLup in order to determine if a customer is allowed to enter it or not, and informs CLup whether the visit begins or ends.

Turn Announcement System

The Turn Announcement System receives information from CLup about customers allowed to enter a specific store in order to inform the ones waiting outside it.

2.4 Assumptions, dependencies and constraints

2.4.1 Table of domain assumptions

Domain assumptions	
A1	Opening and closing time of the store are respected
A2	If the number of people inside the store is not greater than its maximum occupancy, then safety is guaranteed inside the store
A2.1	If the number of people inside each product section is below its maximum occupancy specified by store managers, then safety is guaranteed inside that section
A3	Only customers authorized by CLup can enter and exit the store
A4	On average, the information provided by customers is correct

A4.1	The number of people associated with each line-up request and with each booking request is respected when the visit takes place
A5	Customers who have access to the application prefer to use it rather than the proxy

Table 2.1: Domain assumptions

2.4.2 Table of dependencies

Dependencies
CLup depends on an external system, the Turn Announcement System. Their interaction allows all the customers to be informed about who is allowed to enter the store. This system must be able to reach every type of customer and be able to provide other useful information
CLup depends on an external system, the Access Management System. Their communication lets only customers authorized by CLup visit the store

Table 2.2: Table of dependencies

2.4.3 Constraints

The system must provide a fallback option that allows customers who cannot use the application to line up and visit the store.

Chapter 3

Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

CLup should provide two different user interfaces: the first one is designed for store managers, where the other one is designed for app-customers. First mock-ups of both the interfaces follow.

Figure 3.1: Mockup 1 – Manager view

From his interface, the manager is able to monitor its stores's status. He can also change parameters such as working hours, maximum occupancy and product sections.

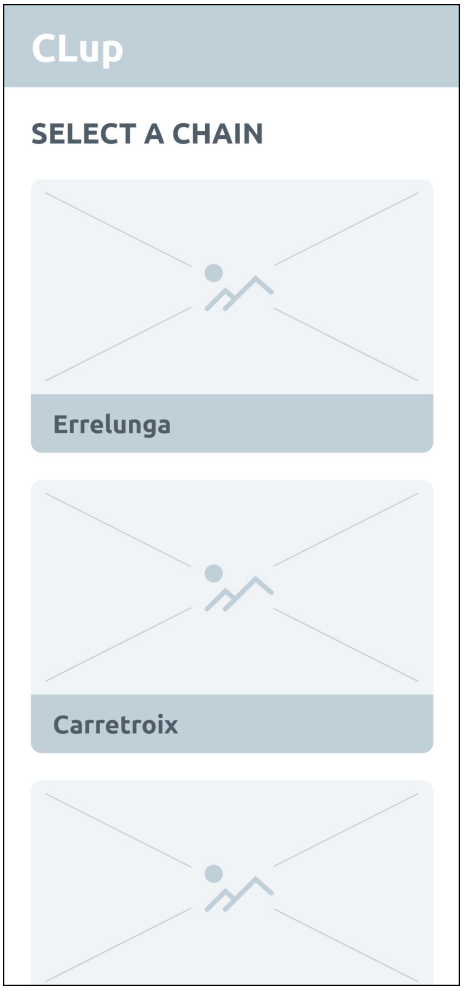

Figure 3.2: Mockup 2 – Select chain

A mock-up representing the GUI that the customer sees when he has to select a chain of stores.

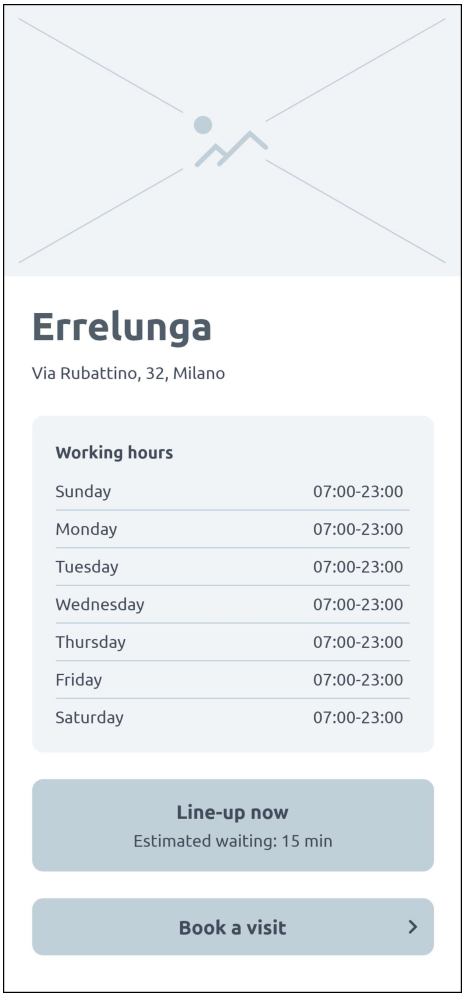

Figure 3.3: Mockup 3 – Store view

A mock-up representing the GUI that the customer sees when selecting a store.

3.1.2 Hardware interfaces

Customer's device

The customer is supposed to use a smartphone to interact with CLup. In order to interact with the system, the smartphone should have an internet connection.

Manager's device

The manager is supposed to use a personal computer to interact with CLup. In order to interact with the system, it should have an internet connection.

Proxy

Since the proxy should provide customers with an easy way to line up, it must have at least:

- An internet connection
- A way to allows customers to make a line-up request for a single person
- A way to physically provide them with the visit token associated with the line-up request

Turn Announcement System

The Turn Announcement System should offer an interface that allows CLup to inform it about the needed information to be provided to the customers.

Access Management System

CLup provides an API to be used by the Access Management System in order to authorize the visit of a customer and inform CLup about the beginning and ending of the visit.

3.1.3 Software interfaces

Map API

In order to properly send notifications, the system must estimate the time needed for the customer to reach the store. Thus, a map service should be used to get that estimation based on the customer current position.

3.2 Functional requirements

3.2.1 Table of functional requirements

The following requirements refer to a given store of a given chain, both managed by CLup.

Functional requirements	
R1	The system shall allow managers to specify the maximum occupancy of the store
R2	The system shall allow managers to monitor entrances
R3	The system shall allow managers to monitor exits
R4	The system shall authorize accesses to the store
R4.1	The system shall authorize customers to enter if and only if the store would not exceed the maximum number of people allowed inside it
R5	The system shall provide a way to line up in the virtual queue of the store
R6	The system shall provide a way to exit the queue before entering the store

R7	The system shall alert the app-customer when it is time to reach the store
R8	The system shall provide the possibility to book a time interval for visiting the store
R9	The system must not allow customers to book a visit in a time interval if, over its duration, bookings by other users already maximize store occupancy
R10	When booking a visit, the system shall allow customers to specify what kind of products they intend to buy
R11	The system shall provide the possibility to cancel a booked visit before entering the store
R12	While making a booking request, the system shall suggest alternative time intervals if the demand of the chosen one is too high
R13	While making a booking request, the system shall suggest alternative stores of the same chain if the demand for the chosen time interval in the selected store is too high
R14	The system shall allow managers to regulate entrances
R15	The system shall allow managers to regulate exits
R16	The system shall notify a customer when, during a specific time interval, a specified store is reaching its maximum occupancy
R17	The system shall keep track of the average duration of a generic visit to the store
R18	The system shall manage the case in which customers do not show up when it is their turn to enter the store

Table 3.1: Functional requirements

3.2.2 Mapping goals-assumptions-requirements

Goals	Assumptions	Requirements
G1	A2, A2.1, A3, A4, A4.1	*R1, R4, R4.1, R5, R9, R10, R12, R13, R14, R15, R18*
G2	A1, A4, A5	*R5, R7, R8, R9, R12, R13, R17, R18*
G3	A1, A4	*R5, R6, R7, R8, R9, R11, R12, R13, R16, R17, R18*
G4	A3, A4.1	*R1, R2, R3, R14, R15, R17*

Table 3.2: Mapping of goals-assumptions-requirements

3.2.3 Use cases

Use case 1

Name	Checking the turn
Actor	Customer
Entry condition	The Customer has a visit token
Event flow	1. The Customer checks if can enter the store 2. The system provides him with the required information
Exit condition	The Customer is aware if he can or cannot enter the store
Exception	None

Table 3.3: Use case 1 – "Checking the turn"

Use case 2

Name	Checking the turn using the application
Actor	Customer
Entry condition	The Customer has a visit token obtained through the application

Event flow	1. The Customer opens the application 2. The Customer checks if he can enter the store for which he made the visit request 3. The application provides him with the required information
Exit condition	The Customer is aware of whether he can or cannot enter the store
Exception	None

Table 3.4: Use case 2 – "Checking the turn using the application"

Use case 3

Name	Checking the turn in presence
Actor	Customer, Turn Announcement System
Entry condition	The Customer has a visit token
Event flow	1. The Customer reaches the store for which he made the visit request 2. The Customer checks the Turn Announcement System 3. The Turn Announcement System provides him with the required information obtained by CLup
Exit condition	The Customer is aware of whether he can or cannot enter the store
Exception	None

Table 3.5: Use case 3 – "Checking the turn via the Turn Announcement System"

Use case 4

Name	Visiting the store
Actor	Customer, Access Management System
Entry condition	The Customer is allowed to enter the store

Event flow	1. The Customer moves toward the entrance of the store 2. The Customer is granted to enter by the external Access Management System using his visit token 3. The Customer enters the store together with the other people specified in the visit request 4. The external Access Management System informs CLup that the Customer is inside the store 5. The Customer does his shopping and is going to exit the store 6. The Customer is granted to exit by the external Access Management System using his visit token 7. The Customer exits the store together with the other people specified in the visit request 8. The external Access Management System informs CLup that the Customer has exited the store
Exit condition	CLup is aware of the actual number of people inside the store
Exception	Different exceptions can arise: – The Customer is not yet allowed to enter the store – The Customer has lost or does not have access to his visit token. If the Customer is not already in the store, he loses the possibility to enter. If the Customer is already in the store, he can exit it only with the help of a store manager – The Customer does not show up

Table 3.6: Use case 4 – "Visiting the store"

Use case 5

Name	Place a visit request
Actor	Customer
Entry condition	The Customer wants to visit a store
Event flow	1. The Customer chooses to line up or to book a visit 2. The Customer uses the dedicated function provided by CLup 3. The Customer receives with a visit token

Exit condition	The Customer has what he needs to access the store
Exception	Exceptions can arise while lining up or booking a visit

Table 3.7: Use case 5 – "Place a visit request"

Use case 6

Name	Place a line-up request (lining up)
Actor	Customer
Entry condition	The Customer wants to line up for a store and the store is open
Event flow	`1.` The Customer interacts with one of the interfaces provided to him by CLup. He can use both the application or the proxy. If the Customer uses the application, he also selects the desired chain and store `2.` The Customer uses the function to line up offered by CLup. In case of app-customers, they also indicate the number of people who want to visit the store `3.` An access request is generated with an associated visit token, which is provided to the Customer `4.` The Customer receives the visit token
Exit condition	The Customer has what he needs to access the store
Exception	The visit request cannot be generated. This can happen due to different causes: – a communication error occurs – the store is closed An error is shown to the Customer and the lining up operation is not completed

Table 3.8: Use case 6 – "Place a line-up request (lining up)"

Use case 7

Name	Booking a visit (place a booking request)
Actor	Customer
Entry condition	The Customer wants to book a visit to a store
Event flow	1. The Customer opens the application and selects the desired chain and store 2. The Customer selects the function for booking a visit 3. The Customer receives information about the store from CLup, such as its product sections 4. The Customer specifies when he wants to enter the store 5. The Customer specifies the estimated duration of the visit. If he is a long-term customer, an approximate duration is suggested by CLup 6. The Customer can specify which kind of products he intends to buy 7. The Customer specifies the number of additional people who wants to visit the store with him 8. The Customer confirms that he wants to book a visit 9. A booking is generated and a visit token is associated to it 10. The Customer receives the visit token
Exit condition	The Customer has what he needs to access the store

Exception	Different exceptions can arise: – The selected store is closed during the time interval specified by the Customer – The store has already reached its maximum occupancy during the chosen time interval. Then the application suggests alternative time intervals or alternative available stores of the same chain. However, if the Customer does not specify a different time interval or a different store he cannot place the booking – The store is likely to reach its maximum occupancy during the chosen time interval. Then the application suggests an alternative time interval or other stores of the same chain in order to balance the number of people inside the store selected by the Customer. However, the Customer can book the visit anyway – A communication problem occurs after step 6 An error is shown to the Customer and the booking operation is not completed

Table 3.9: Use case 7 – "Booking a visit (place a booking request)"

Use case 8

Name	Reaching the store
Actor	Customer
Entry condition	The Customer who used the application to request the visit should now head to the store if he wants to arrive in time
Event flow	1. CLup retrieves the Customer's position 2. Considering the distance between his position and the store, a notification is sent to the Customer when he should head to the store in order to arrive in time. This is done with respect to the estimated time in which the Customer should be able to enter the store 3. The Customer reads the notification
Exit condition	The Customer is aware that he should head to the store

Exception	The Customer location cannot be sent to CLup. Possible causes: – the position is not available – a communication error occurs For this reason, no notification is sent

Table 3.10: Use case 8 – "Reaching the store"

Use case 9

Name	The store is filling up
Actor	Customer
Entry condition	The Customer expresses the will to be notified about the availability of a store in specific time intervals
Event flow	1. The information about the store and the time interval the Customer wants to be notified about is either inferred by CLup or specified by the Customer himself 2. That time interval is likely to be saturated (i.e. no other bookings in that time interval are going to be allowed or the current estimated disposal time of the queue is going to reach the end of the time interval) 3. CLup notifies the Customer about it
Exit condition	The Customer is aware that the store is reaching its maximum occupancy during the time intervals he is interested in
Exception	None

Table 3.11: Use case 9 – "The store is filling up"

Use case 10

Name	Administering the store
Actor	Manager
Entry condition	The Manager wants to edit some parameters of one of the stores he manages

Event flow	1. The manager uses the tool to check the current status of the store 2. If needed, the manager adjusts store parameters
Exit condition	The Manager is aware of the status of the store and CLup is up to date with Manager choices
Exception	The parameters specified by the Manager are not valid

Table 3.12: Use case 10 – "Administering the store"

Use case 11

Name	Let a customer exit the store
Actor	Customer, Manager, Access Management System
Entry condition	The Customer has asked the Manager to exit the store because he has lost his visit token
Event flow	1. The Manager uses the administrative tool to allow the Customer to exit the store 2. The Customer exits the store by means of the Access Management System 3. CLup updates the store occupancy
Exit condition	CLup is aware of the actual number of people in the store. The Customer has left the store
Exception	None

Table 3.13: Use case 11 – "Let a customer exit the store"

Use case 12

Name	Cancel visit
Actor	Customer
Entry condition	The Customer has a visit token and is not already in the store

Event flow	1. The Customer asks CLup to delete a visit request he made before 2. CLup processes and deletes the request
Exit condition	The Customer's visit token is not valid anymore
Exception	None

Table 3.14: Use case 12 – "Cancel visit"

3.2.4 Use case diagram

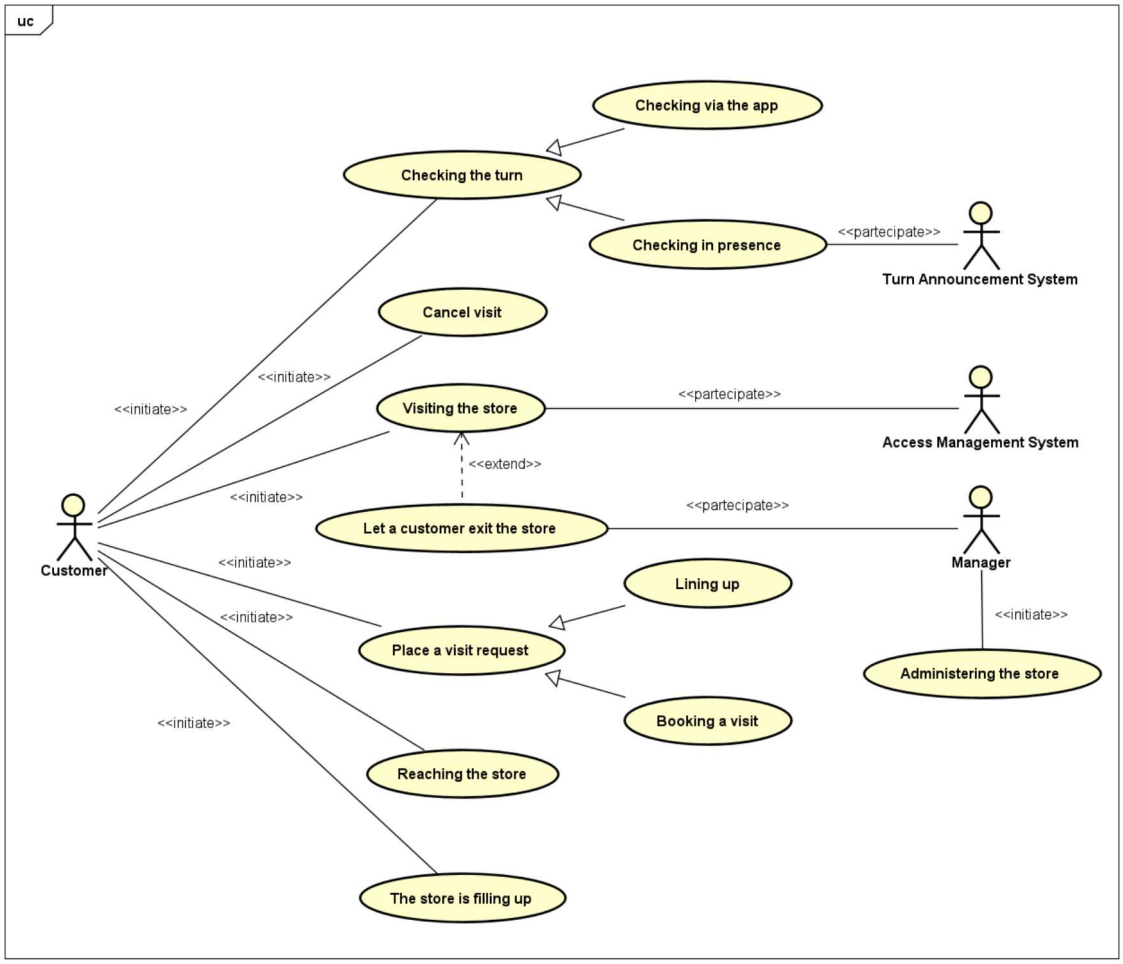

Figure 3.4: Use case diagram

3.2.5 Sequence Diagrams

The following diagrams are some of the most significant Sequence Diagrams, and each of them refers to the homonym Use Case.

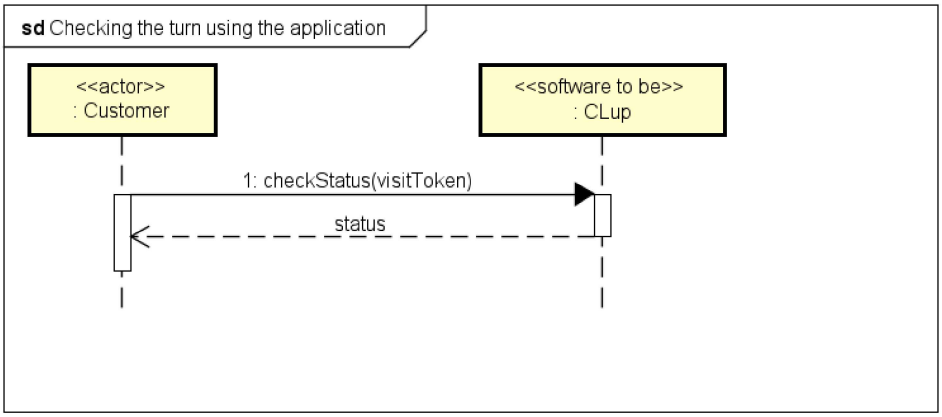

Figure 3.5: Sequence diagram 1 – "Checking the turn using the application"

This sequence diagram shows the interaction between an app-customer checking the turn and CLup.

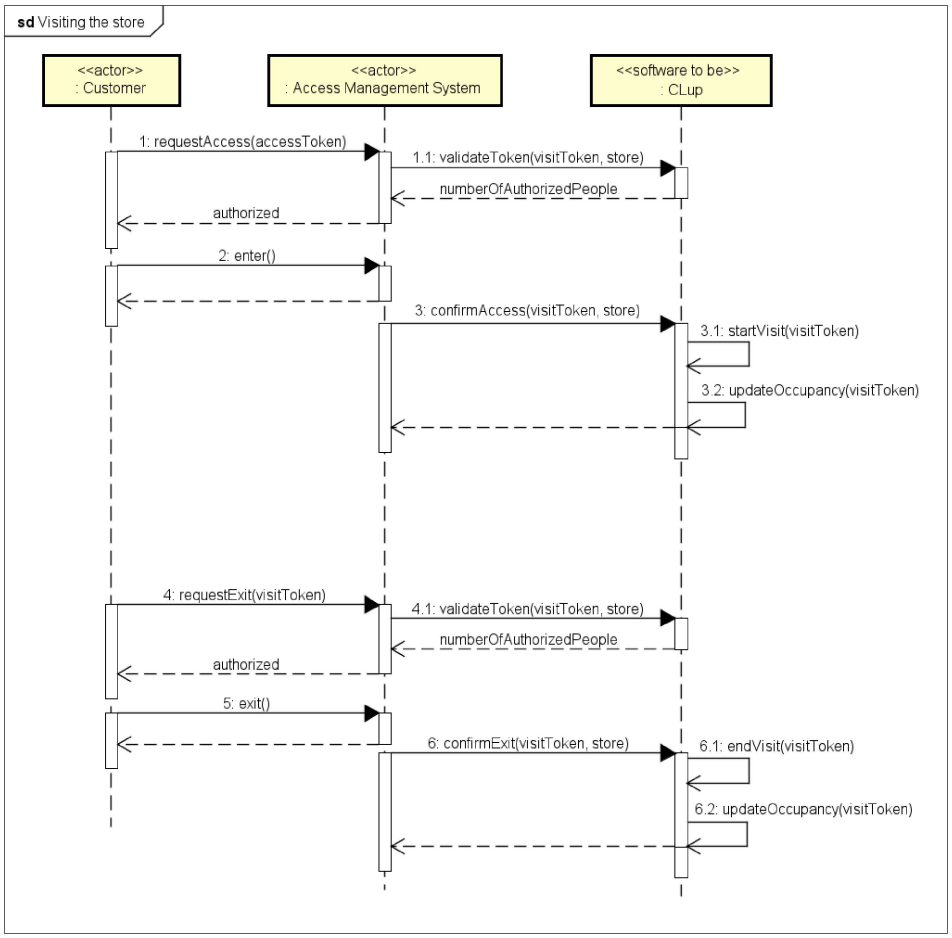

Figure 3.6: Sequence diagram 2 – "Visiting the store"

This sequence diagram illustrates the dynamic behaviour of the system when a customer visits the store.

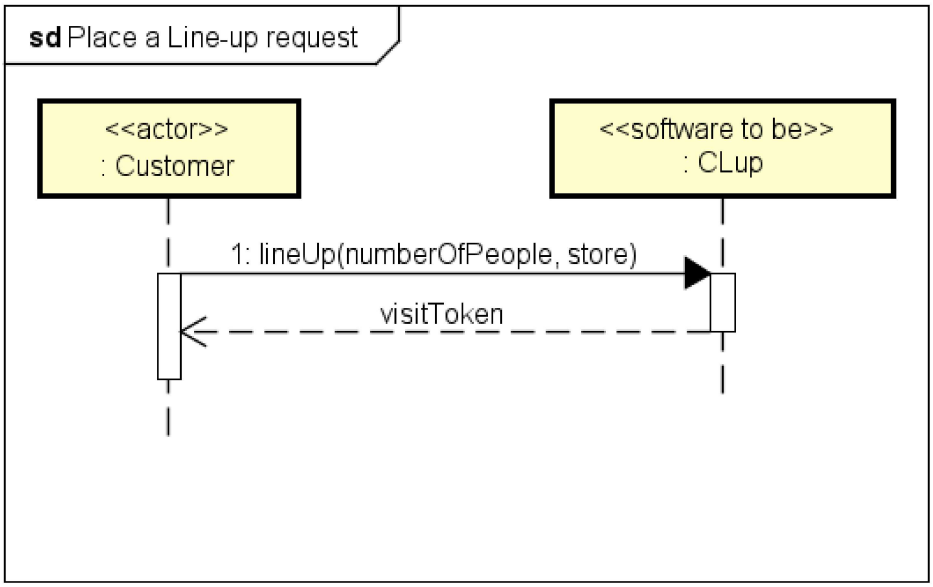

Figure 3.7: Sequence diagram 3 – "Place a line-up request"

Sequence diagram illustrating the interaction between a customer who lines up and CLup.

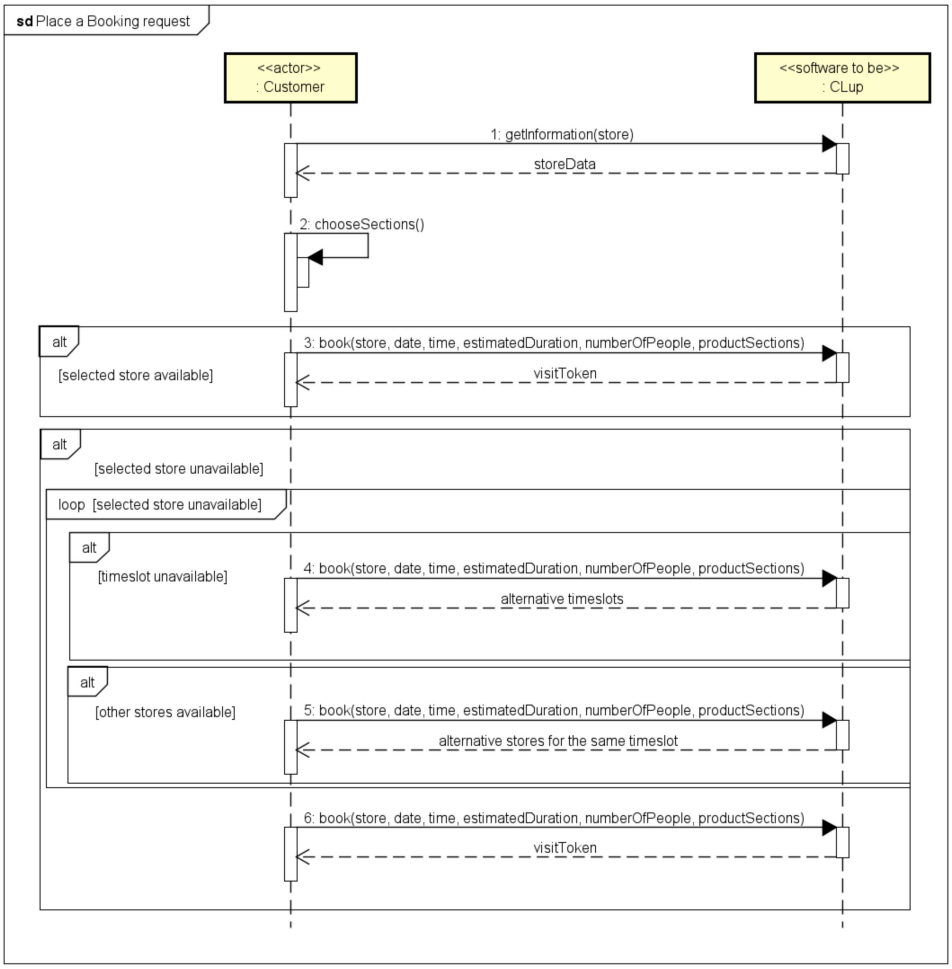

Figure 3.8: Sequence diagram 4 – "Place a booking request"

This sequence diagram shows the interaction between an app-customer placing a booking request and CLup.

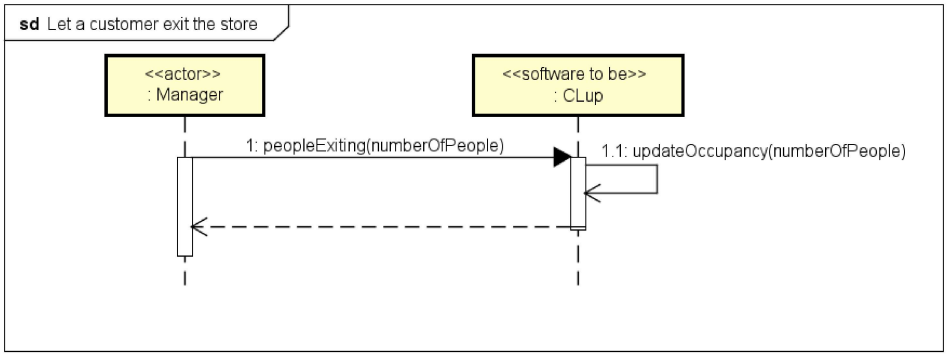

Figure 3.9: Sequence diagram 5 – "Let a customer exit the store"

This sequence diagram illustrating the dynamic behaviour of the system when a store manager allows a customer exit the store.

3.2.6 Activity diagrams

The following activity diagrams represent the interaction between both app-customers and proxy-customer with the system.

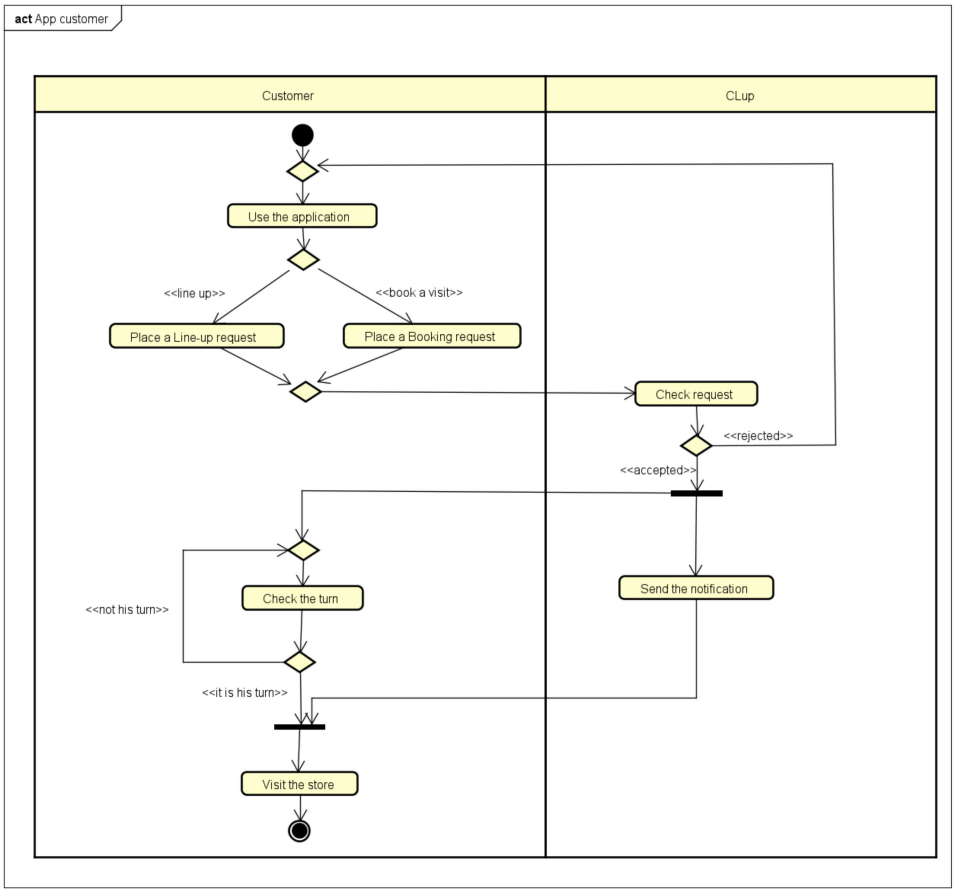

Figure 3.10: Activity diagram 1 – "App customer"

This activity diagram shows the flow of events related to a customer who uses the application and wants to visit the store.

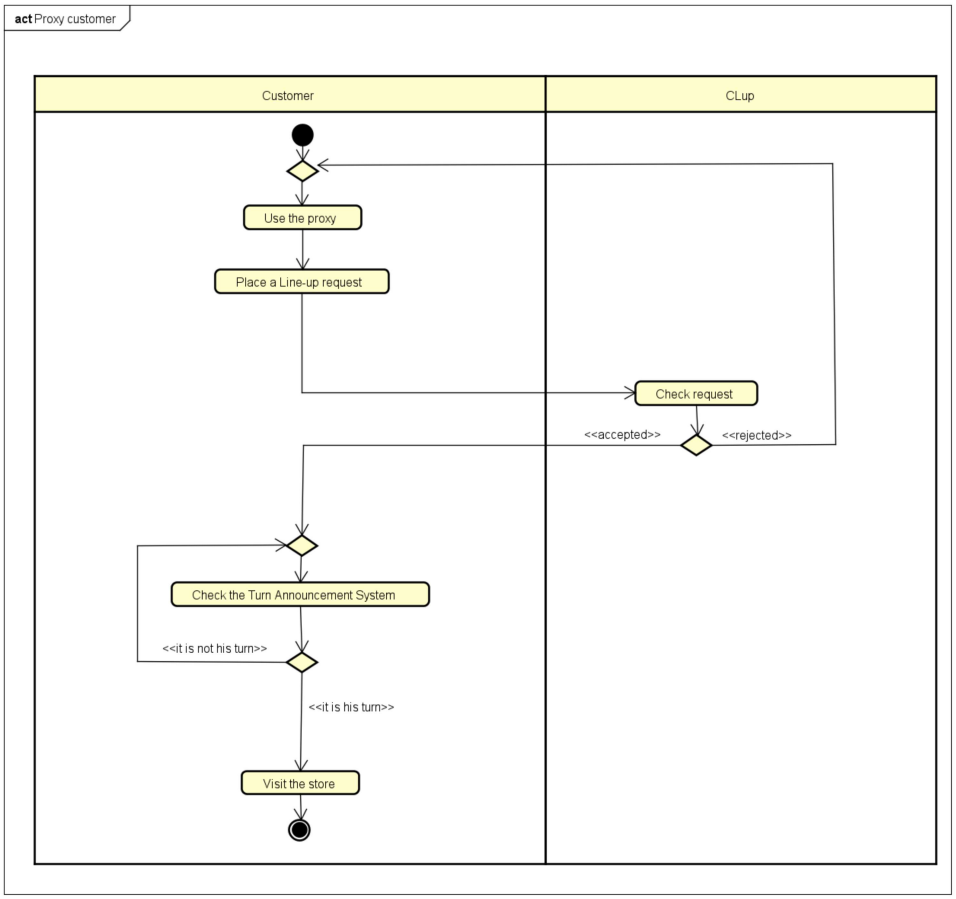

Figure 3.11: Activity diagram 2 – "Proxy customer"

This activity diagram shows the flow of events related to a customer who uses the proxy and wants to visit the store.

3.2.7 Traceability

Goal	Requirements	Use cases
G1	R1, R4, R4.1, R5, R9, R10, R12, R13, R14, R15, R18	UC1, UC2, UC3, UC4, UC5, UC6, UC7, UC11
G2	R5, R7, R8, R9, R12, R13, R17, R18	UC1, UC2, UC3, UC5, UC6, UC7, UC8
G3	R5, R6, R7, R8, R9, R11, R12, R13, R16, R17, R18	UC1, UC2, UC5, UC6, UC7, UC8, UC9, UC12
G4	R1, R2, R3, R14, R15, R17	UC4, UC10, UC11

Table 3.15: Traceability matrix

3.3 Performance requirements

The system, in normal operating conditions, should be able to handle all line-up and booking requests of each customer. It should be able to manage, without a loss in performance, a large number of simultaneous visit requests which, in a non optimal case, are likely to be in the order of the sum of all maximum occupancies of each store managed by CLup.

The system responses and notifications must be sent within 3 seconds from the triggering event.

3.4 Design constraints

3.4.1 Standard compliance

The whole software should be compliant with the General Data Protection Regulation (GDPR).

3.5 Software system attributes

3.5.1 Reliability

The context in which CLup operates is characterized by a strong focus on safety and distancing between people. Thus, the system must guarantee a high degree of reliability in order to prevent dangerous situations in which safety constraints are violated both inside and outside the store.

3.5.2 Availability

In order to help the store manager to regulate the influx of people to the store and prevent gatherings, CLup should be available the 99% of the time.

3.5.3 Security

The system should focus on protecting against attacks which affect its availability and reliability. It is also fundamental to protect both customers and managers from man-in-the-middle attacks. They should exchange messages and information only with CLup, while keeping private information safe (e.g. customer's current position and future movements).

CLup shall also identify all of the store managers before allowing them to use the store administrative tool.

3.5.4 Maintainability

Over time, the system could need corrections, improvements and additions. For this reason, the system must be well-documented and adaptable to changes.

3.5.5 Portability

The CLup customer application should be compatible with most of the smartphones currently on the market. This way, everyone can benefit from the system, save time and

avoid crowding by queuing remotely and by booking visits. The same should apply for the administrative tool provided to managers, which should be compatible with most of the currently available computers on the market.

Moreover, CLup should be platform independent in order to increase degrees of freedom when choosing the hardware to use and when defining its possible future upgrades.

3.5.6 Accessibility

The application should be easy to use, in order to allow all the different possible customers to visit the store.

3.6 Other requirements

- CLup shall give priority to booking requests over line-up requests;
- Customers can remotely line up in a store's queue only if they are not in the queue of any store at that moment;
- Customers can book a visit for a specific time interval only if they have not booked any other visit which overlaps with that time interval;
- Customers can book a visit for a store and a specific time interval only if it starts before the current queue disposal time of that store;
- If a customer inside the store is no longer able to retrieve his visit token, the system should provide a way to let him out of the store while being aware of the exiting event.

Chapter 4

Alloy

4.1 Alloy description

The Alloy model presented below aims to show how it is possible to build a coherent and consistent solution to the problem explained in this document, taking into account all the already specified constraints. To do so, the Alloy model neglects some aspects of the UML that are not strictly useful to do this kind of analysis, while focusing on the principal behaviours of the system.

As it can be seen by executing the "show" predicate, which includes some constraints of convenience on the number of instances of some signatures, all the specified facts allow to build a feasible instance of the problem that is coherent with all the requirements and constraints.

For the sake of simplicity, time intervals and time in general are represented using integers, and the behaviour of product sections with respect to their maximum occupancy is not deeply investigated. Furthermore, the signatures "Date" and "Token" are not deeply characterized for the same simplicity reasons.

In particular, the model shows that:

- a visit can take place only after its request has been placed;
- a customer can line-up only for a store at a time;
- a customer cannot visit more than one store at the same time;
- a customer cannot book two overlapping time intervals among all the stores;
- it is not possible to book a visit if that booking would make the store exceed its maximum occupancy;
- it is not possible to book a visit which does not fits in the store working hours;
- it is not possible to book a visit that starts before the current queue disposal time;
- it is not possible to line up when the store is closed;
- the current store occupancy is the number of people in the store at that moment;
- tokens, manager ids and product sections are unique.

4.2 Alloy model

```
sig Date {}
sig TimeInterval {
    start : Int,
    end   : Int
} {
    start >= 0 and
    start < end
}
sig Token {} {
    Token = Visit_Request.visit_token
}

sig ID {} {
    ID = Manager.id
}

sig Manager {
    id      : ID,
    stores : some Store
}

sig Chain {
    stores : some Store
}

sig Store {
    parent_chain      : Chain,
    current_occupancy : Int,
    maximum_occupancy : Int,
    working_hours     : some TimeInterval,
    sections          : some Product_Section,
    visit_request         : set Visit_Request,
    queue             : Queue,
    managers          : some Manager
} {
    current_occupancy >= 0 and
    current_occupancy <= maximum_occupancy
}

sig Queue {
    store                   : Store,
    length                  : Int,
    estimated_disposal_time : Int,
    requests                : set LineUp_Request
} {
    estimated_disposal_time >= 0 and
    length = #requests
}

sig Product_Section {
    store             : Store,
    current_occupancy : Int,
    maximum_occupancy : Int
} {
    current_occupancy >= 0 and
    maximum_occupancy >  0 and
    current_occupancy <= maximum_occupancy
}

abstract sig Visit_Request {
    store            : Store,
    number_of_people : Int,
    date_of_creation : Date,
    time_of_creation : Int,
    visit_token      : Token,
```

```
    customer          : Customer,
    visit                          : lone Visit
} {
    number_of_people > 0 and
    time_of_creation >= 0
}

sig LineUp_Request extends Visit_Request {
}

sig Booking_Request extends Visit_Request {
    desired_date     : Date,
    desired_interval : TimeInterval,
    product_sections : set Product_Section
}

sig Visit {
    request              : Visit_Request,
    starting_time        : Int,
    ending_time      : lone Int
}{
    starting_time >= ending_time
}

abstract sig Customer {
    lineUp_requests : set LineUp_Request
}

sig App_Customer extends Customer {
    booking_requests : set Booking_Request
}

sig Proxy_Customer extends Customer {
} {
    #lineUp_requests = 1
}


---------MAPPINGS--------
--A visit request is associated with a visit iff that visit is associated with that
--request
fact mapping_VR_Visit {
    all vr : Visit_Request, v : Visit | vr.visit = v iff v.request = vr
}

--A visit request refers to a certain store and that store has the considered visit
--request in the associated set of visit requests
fact mapping_VR_Store {
    all vr : Visit_Request, s : Store | vr.store = s iff (vr in s.visit_requests)
}

--A line-up request is in a store queue iff it is referred to it and the
--visit has not begun S
fact mapping_LUR_Queue {
    all lur: LineUp_Request, q: Queue | lur in q.requests iff (lur.store = q.store
        and lur.visit = none)
}

--A line-up request refers to a certain customer and that customer has the
--considered request in the associated set of lineup requests
fact mapping_C_LUR {
    all c: Customer, lur: LineUp_Request | lur in c.lineUp_requests iff
        lur.customer = c
}

--A booking request refers to a certain app-customer and that app-customer has
--the considered request in the associated set of booking requests
fact mapping_AC_BR {
```

```
    all ac: App_Customer, br: Booking_Request | br in ac.booking_requests iff
        br.customer = ac
}

--A store refers to a certain chain and that chain has the considered store in the
--associated set of stores
fact mapping_Store_Chain {
    all chain: Chain, store: Store | store in chain.stores iff
        store.parent_chain = chain
}

--A queue refers to a certain store and that store has the considered queue as
--queue
fact mapping_Store_Queue {
    all q: Queue, s: Store | q.store = s iff s.queue = q
}

--A manager refers to certain stores and those store have the considered
--manager in the associated set of managers
fact mapping_Store_Manager {
    all m: Manager, store: Store | m in store.managers iff store in m.stores
}

--A product section refers to a certain store and that store has the considered
--section in the associated set of product sections
fact mapping_Store_Sections {
    all ps: Product_Section, s: Store | ps in s.sections iff s = ps.store
}

--A booking request refers to a certain store and to certain product section.
--Those sections must be sections of the store the request refers to
fact mapping_StoreSections_BookingSections {
    all ps: Product_Section, b: Booking_Request | ps in b.product_sections iff
        ps.store=b.store
}


--------FUNCTIONS--------
fun inProgressVisitRequests[s: Store]: set Visit_Request {
    {vr: Visit_Request | vr.store = s and vr.visit != none}
}

fun storeOverlappingBooking[s: Store, t: TimeInterval, d: Date]: set
    Booking_Request {{
        b: Booking_Request |
            b in s.visit_requests and b.desired_date = d and
            ((b.desired_interval.start <= t.start and
                b.desired_interval.end >= t.start) or
            (b.desired_interval.start >= t.start and
                b.desired_interval.start <= t.end))
}}


----------FACTS----------
--A line-up request can be associated to its visit only if the visit starts the
--same day of the line-up request and after the lining up
fact visit_after_LineUp {
    all v: Visit, lur: LineUp_Request |
        v.request = lur iff v.starting_time >= lur.time_of_creation
}

--A booking request can be associated to its visit only if the visit starts in the
--desidered date and after the desidered starting time
fact visit_after_Booking {
    all v: Visit, b: Booking_Request |
        v.request = b iff v.starting_time >= b.desired_interval.start
}
```

```
--A customer can line up for only a store at a time
fact one_LineUp_AtATime {
    all c: Customer | #{lur: LineUp_Request | lur in c.lineUp_requests and
        lur.visit=none}<=1
}

--A customer cannot visit more than one store at the same time
fact no_MoreThanOne_ActiveVisit {
    (all c: Proxy_Customer | #{vr: Visit_Request| vr in c.lineUp_requests and
        vr.visit != none and vr.visit.ending_time = none}<=1)
    and
    (all c: App_Customer | #{vr: Visit_Request| vr.visit != none and
        vr.visit.ending_time = none and (vr in c.lineUp_requests or
            vr in c.booking_requests)}<=1)
}

--A customer cannot book two overlapping time intervals among all the stores
fact no_OverlappingBookings_ForCustomers {
    all b1: Booking_Request, b2: Booking_Request |
    (
        b1!=b2 and b1.customer = b2.customer and b1.desired_date = b2.desired_date
            implies (
                (b1.desired_interval.end < b2.desired_interval.start)
                or
                (b1.desired_interval.start > b2.desired_interval.end)
            )
    )
}

--It is not possible to book a visit if the booking would make the store exceed
--its maximum occupancy
fact no_TooMany_OverlappingBookings {
    all s: Store, t: TimeInterval, d: Date |
        (sum b: storeOverlappingBooking[s, t, d] | b.number_of_people) <=
            s.maximum_occupancy
}

--It is not possible to book a visit which does not fits in the store working
--hours
fact no_Bookings_WhenClose {
    all br: Booking_Request | #{t: TimeInterval | t in br.store.working_hours
        and (br.desired_interval.start >= t.start and br.desired_interval.end <=
            t.end)}>=1
}

--It is not possible to book a visit that starts before the current queue disposal
--time
fact no_Booking_Before_Queue {
    all b: Booking_Request | b.date_of_creation = b.desired_date implies
        b.desired_interval.start > (b.time_of_creation +
            b.store.queue.estimated_disposal_time)
}

--It is not possible to line up when the store is closed
fact no_LineUps_WhenClose {
    all lu: LineUp_Request | #{t: TimeInterval | t in lu.store.working_hours
        and lu.time_of_creation >= t.start and lu.time_of_creation <= t.end}>=1
}

--The current store occupancy is the number of people in the store at that moment
fact storeOccupancy {
    all s: Store | s.current_occupancy = (sum x: inProgressVisitRequests[s] |
        x.number_of_people)
}


-----UNICITY CONSTRAINTS------
--Each visit token is unique
```

```
fact uniqueToken {
    all t: Token, vr1: Visit_Request, vr2: Visit_Request |
        t in Visit_Request.visit_token and
        (vr1.visit_token = t and vr2.visit_token = t implies vr1 = vr2)
}

--Each manager has a unique id
fact uniqueID {
    all id1 : ID, m1 : Manager, m2 : Manager |
        id1 in Manager.id and
        (m1.id = id1 and m2.id = id1 implies m1 = m2)
}

--A product section cannot be shared between stores. Each store has its own
--product sections
fact uniqueProductSection {
    all ps : Product_Section, s1 : Store, s2 : Store |
        ps in Store.sections and
        ps in s1.sections and ps in s2.sections implies s1 = s2
}

pred show {
    #Chain = 2
    #Store = 3
    #Visit >= 1
    #Proxy_Customer >= 3
    #App_Customer >= 3
    #Queue.requests >= 3
}
run show for 8
```