POLITECNICO
MILANO 1863

CLup

Requirements Analysis and Specification Document

Authors:
Francesco Fulco Gonzales(10614882) , Alberto Latino(10600138)

Version: 1.0

Professor: Elisabetta Di Nitto

Contents

1. Introduction

1.1. Purpose

This document is the Requirements Analysis and Specification Document (RASD) for the CLup system. It has the purpose of communicating an understanding of the requirements, explaining both the application domain and the system to be developed. It also provides a reference for project planning and cost estimation, as well as giving a baseline for software evaluation with regard to testing, verification and validation and guiding the long term development of the software throughout its lifetime.

In response to the coronavirus health emergency, several governments have decided to restrict access to stores to avoid having crowds in closed spaces. This has led to long queues waiting outside stores. The CLup software aims to solve this problem and make customers avoid spending a huge amount of time in a physical queue, by allowing them to book a virtual queue ticket directly from home. CLup includes two main functionalities:

- Basic service: allows users to virtually line up at the store by booking a digital ticket (with QR code) and notifies them shortly before their number is called;

- Book a visit: users can book a ticket (with QR code) for a visit at the store, indicating the time slot when they want to shop, the expected duration of the visit and product categories they intend to buy.

Here below are listed the goals of the application:

G.1 Allow store managers to regulate the influx of people in the building
G.2 Allow people to virtually line up instead of standing outside of stores
G.3 Allow people to plan a visit to the supermarket
G.4 Allow to keep safety distance among people inside the stores

1.2. Scope

1.2.1. World Phenomena

- User carries a smartphone
- User goes to the supermarket
- Users amass outside of the supermarket
- Users amass inside of the supermarket

1.2.2. Shared Phenomena

Controlled by the world and observed by the machine.

- Individuals register to the system
- Individuals log into the system
- Store managers register themselves and their store to the system
- A user gets a ticket to virtually line up
- A user books a visit at the supermarket
- User requests a ticket from the proxy machine located in site (fallback option)
- Users insert the mean of transport they intend to use to reach the store
- Users scans the QR code provided by the system at the entrance scanner
- Users scans the QR code provided by the system at the exit scanner
- Store managers monitor the influx of people from the web app

Controlled by the machine and observed by the World.

- The system generates a ticket with its QR code
- The system gives an estimation of waiting time in the queue
- The system notifies the user when it is almost his turn
- The proxy device prints a ticket with QR code and waiting time
- The system validates customers' tickets both at the entrance and the exit
- The system opens and closes doors of the stores
- The system informs users of availability of timeslots and product categories

1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

- **Ticket:** Virtual item associated to a QR code that allows users to virtually line-up and uniquely identify their visit to the supermarket
- **Expired Ticket:** Ticket that do not allow people to enter the store anymore
- **Time Slot:** Specific span of time that can be chosen by the user to go shopping.
- **Statistics:** Facts about visit duration and time spent at the store inferred through previous data analysis
- **QR Code:** a machine-readable code consisting of an array of black and white squares, used to store information of the ticket that can be read by a device equipped with a camera
- **Access Control System:** System that once validated a QR code, allows customers to enter or exit the store

1.3.2. Acronyms

- **UI:** User Interface through users require available functionalities.

- **UX:** User Experience, way in which the user interacts with the service.

1.3.3. Abbreviations

- **G.i**: i-th goal
- **R.i:** i-th requirement
- **D.i:** i-th domain assumption
- **SM:** Store Manager
- **QTA:** Queue Ticket Assistant
- **ACS:** Access Control System

1.4. Revision history

23/12 First version

1.5. Reference Documents

Specifications document :"R&DD Assignment 2020-21.pdf"

1.6. Document Structure

Chapter 1: Introduces and gives the purpose of the CLup software including its goals. Here we include definitions, acronyms abbreviations, and revision history as well.

Chapter 2: here, an overall description of the software is given. The product perspective is illustrated and the software's functionalities are explained more in detail. It also contains a class diagram and statecharts used to better describe the software.

Chapter 3: Interface requirements such as user, software and hardware interfaces. Here we also include functional and non functional requirements. Functional requirements are associated with use cases and sequence diagrams. Non functional requirements such as performance and system attributes are also included.

Chapter 4: here we include the alloy code, then metamodels generated from it.

Chapter 5: Effort spent

Chapter 6: reference documents

2. Overall Description

2.1. Product perspective

The CLup software needs to be installed by users and supermarkets. Stores register to the system providing all necessary information for the system such as departments' capacities for queue customers and visiting customers. The user installs the software on his device in order to get tickets or book time slots for shopping while store managers can register themselves and their store through a webapp. Here are the main aspects related to CLup:

- The user selects one of the registered supermarkets and gets a ticket for its virtual queue. Through the user's GPS and the means of transport selected, the system computes the time needed to get to the store in order to send a notification of the user's turn in advance. A QR code associated with the ticket is generated for a validity check both for entering and exiting the store.

- The user can also book a time slot when he intends to go shopping. The user should also specify the products' categories he intends to buy in order to avoid crowding in specific departments. A QR code associated with the ticket is generated for a validity check at the store.

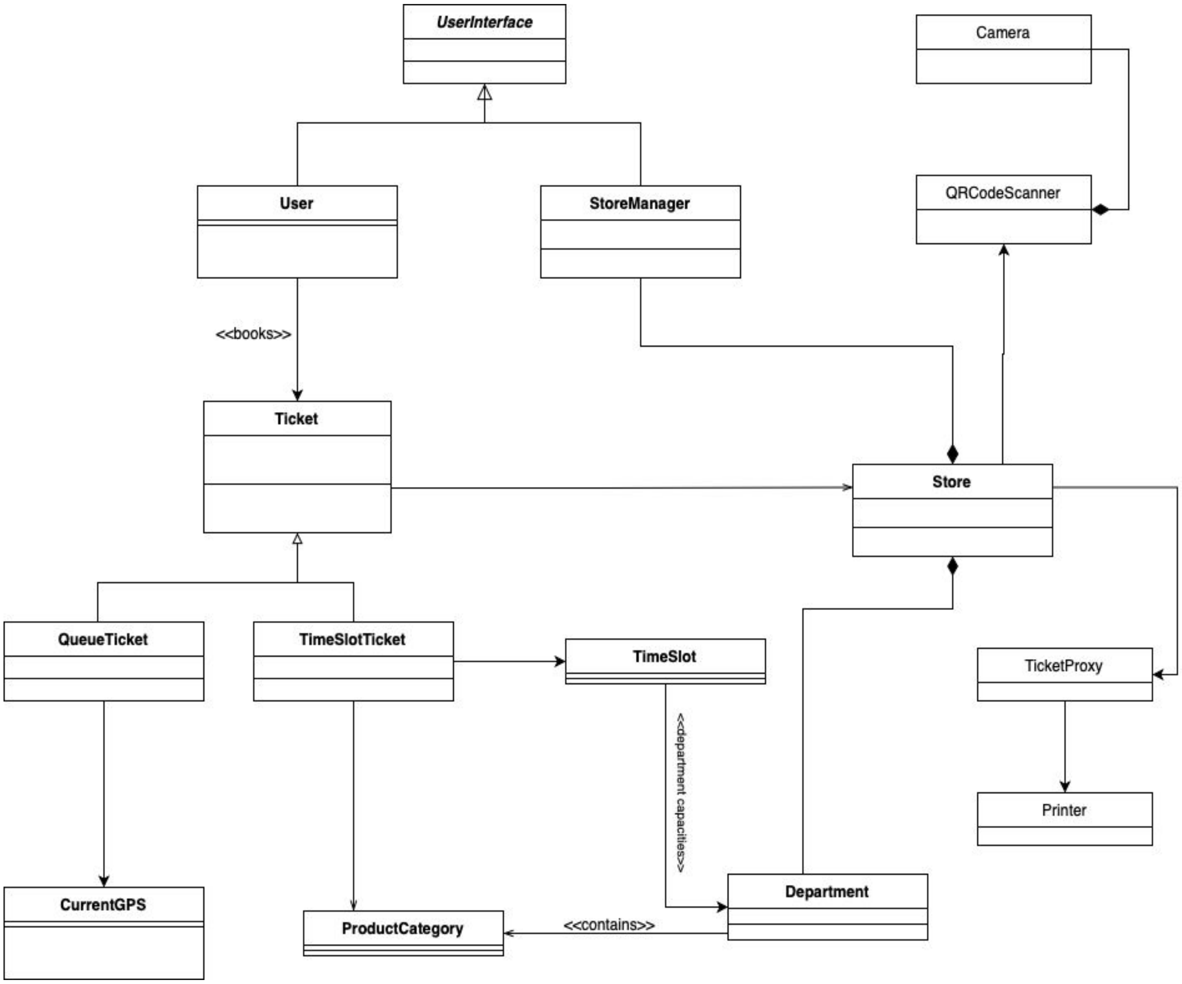
- People who aren't registered to the app can line-up anyway by using a proxy device connected to the system. This device equipped with a printer will print a ticket with QR code and estimated waiting time. A Queue Ticket Assistant is an employee of the store that helps people to do tickets through the fallback option.

- At the entrance of the stores a QR code reader scans the code provided by the user with a device equipped with a camera. After the validity check, the system allows the customer to enter the store and stores the entry time.

- Upon exiting the store, a QR code reader scans the same QR code provided at the entrance by customers in order to let them exit and store the exit time. Through entry and exit time the system will compute statistics about the time spent at the store by each user.

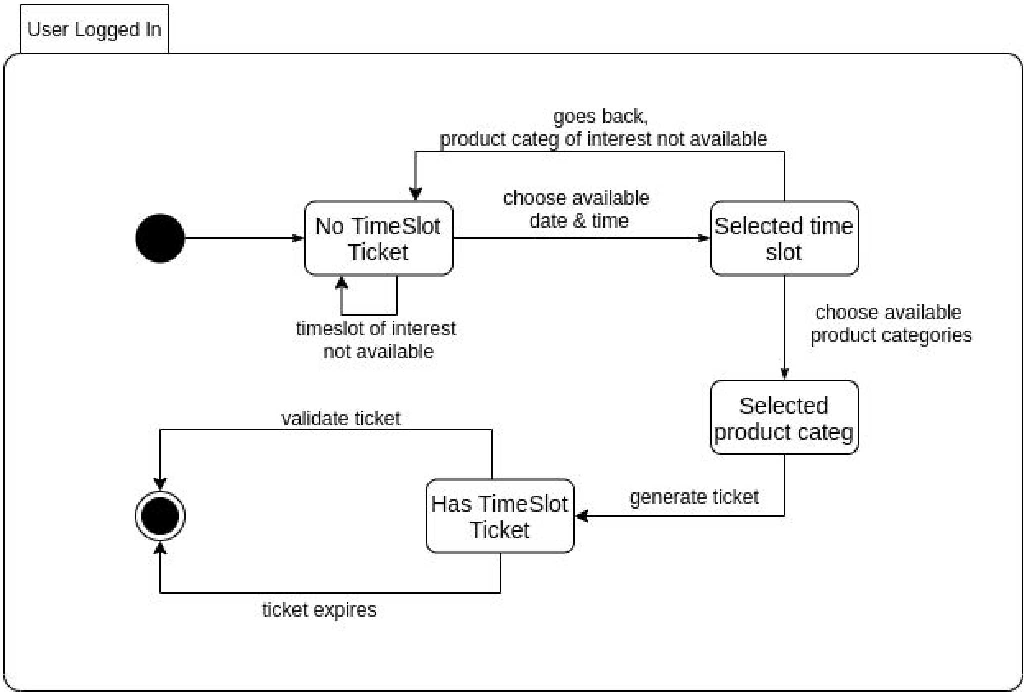
- The store manager monitors the influx of people at the store and visualizes statistics through a web app.

2.1.1. Class Diagram

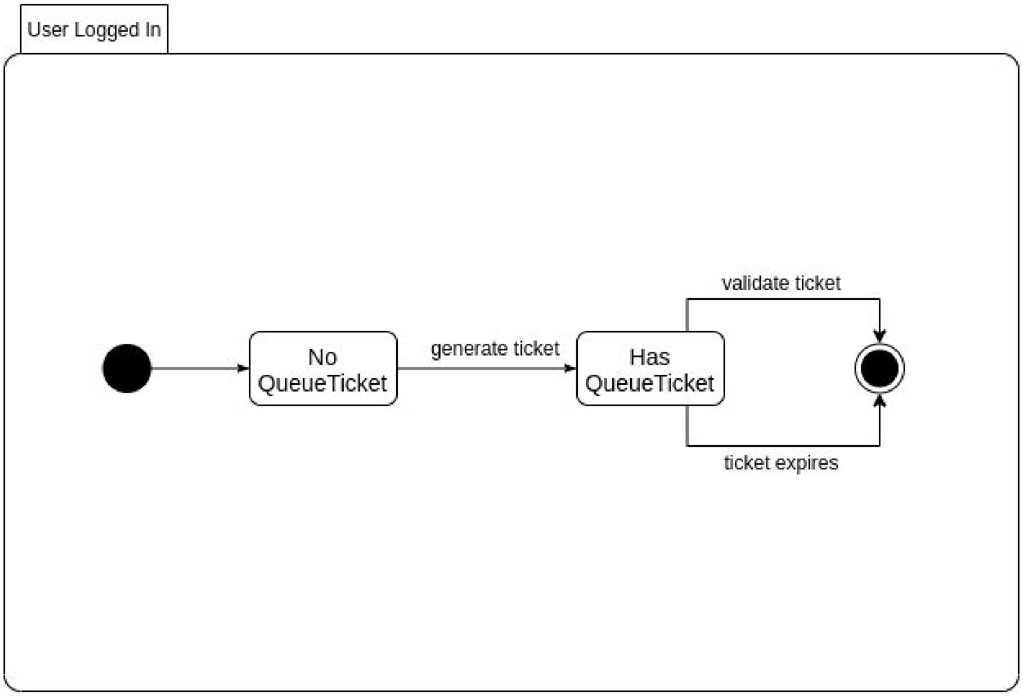


2.1.2. State Charts

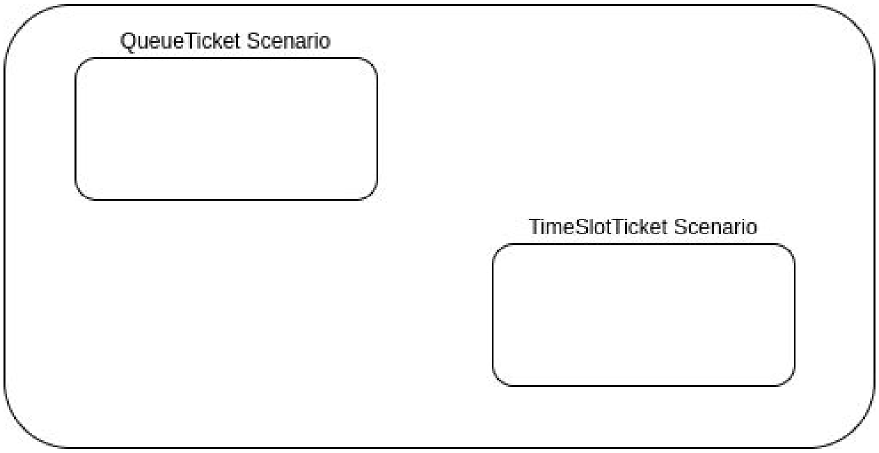
User Books TimeSlot



User Lines Up



Interaction of the two states above



The charts above represent the most significant business flows of the app, here are some other minor flows:

- Registration of user
- Registration of a supermarket, setting product categories, departments and capacities for each department
- Login for user and supermarket
- Ticket generation as proxy for customers (fallback option)
- Update of information about supermarket (capacity etc)
- Update user profile

2.1.3. Scenarios

Scenario 1: Marco is a businessman, he is too busy to go to the grocery store. The covid-19 health emergency has led to very long waiting queues in front of the stores. This has heavily affected Marco's free time and his business schedule. Thanks to CLup, Marco avoids spending almost two hours in the queue and as a consequence has time for one more meeting before going to the supermarket.

Scenario 2: Francesca is a woman mother of two children. She can go to the grocery only on Sunday morning at 11 and she can’t leave her children alone for up to one hour. If she spent up to two hours in the store’s queue, she wouldn’t be able to be away from home for the desired time. Therefore, thanks to the “Book a visit feature” of CLup, she can plan her shopping at 11 on Sunday and be sure to enter the store at that time, without waiting in the virtual queue for her turn to be called.

Scenario 3: Fabio manages one of the biggest supermarkets of Milan. Due to government restrictions for facing up to the covid-19, he can’t allow people to freely enter the store. Instead, he has to regulate the influx of people being sure that the number of customers who enter the store is below a certain threshold established by the government. Thanks to CLup, Fabio has an automated system that helps him to allow access to the fixed number of customers without making them ammas in the queue outside the store. It is known that a reduced capacity leads to lower revenues for the store, nevertheless the segmentation of departments of the “Book a visit” feature slightly increases the allowed number of customers in the store, bringing benefits to the store’s business.

Scenario 4: The introduction of such a technology to regulate the influx of people at stores would affect the shopping of lots of eldery people who are not comfortable with technology. But thanks to the CLup the fallback option, also this kind of person is able to retrieve a physical ticket in front of the store with the help of a queue ticket assistant.

2.2. Product functions

The CLup software aims to reduce crowding in and outside stores. Providing such a service that allows users to get tickets directly from home has a huge impact in tackling the overcrowding problem. Here we include more precisely the main functionalities of the software:

Get a queue ticket
The main functionality of the software is to allow registered people to virtually line up in the queue of a specified store directly from home. Once a user has requested the ticket, the system generates it with a unique number and QR code.The QR code, that is assigned to the ticket, will be scanned and validated upon entering the store. During the ticket request phase, the amount of time required by the user to reach the store is computed by the system, after having received the user’s GPS position. This is done in order to make the system notify the user in advance of his turn. For people who are not able to use a smartphone or are not registered to the app, an in place proxy device prints tickets for lining up.

Book a visit
Besides getting a ticket for the queue, users can also indicate a precise day and time when they intend to go shopping. This way, people do not have to wait until their number is called but can directly show up at the indicated time. Users might also indicate the approximate expected time of the visit. Alternatively, for long term customers, this time could be inferred by the system through analysis of previous shopping data. This feature also allows customers to indicate the categories of products they are going to buy in order to make the system manage capacities of different departments. Also the visit ticket is validated at the entrance by a QR code.

Monitoring time statistics

In order to better manage influx of people and avoid crowding, entrance and exit time of customers is stored for future data analysis. This data is used to infer the duration of a visit if the user does not explicitly specify it. Furthermore, these data could also be used to predict when a user is going to get out of the store. This is useful to notify people in advance when it is almost their turn for the “lining up” feature.

2.3. User characteristics

There are four main actors:

- **Guest:** someone that downloaded the app but is not yet registered to access the functionalities.
- **User:** someone that is correctly logged into the system and can access the software’s functionalities.
- **Store Manager (SM):** employee of a supermarket who has to interface the store with the system. More precisely, he has to specify the capacity of departments and of the whole store.
- **Queue Ticket Assistant (QTA):** an employee of the store that helps people to retrieve their paper ticket at the proxy device.
- **Physical person:** a person that has not downloaded the app who is served by the QTA to get a ticket.

2.4. Assumptions, dependencies and constraints

Here we include the domain assumptions:

D.1 Each user has a device to use the app
D.2 Internet connection works properly
D.3 Each smartphone has a GPS sensor
D.4 Each smartphone is uniquely associated to a person
D.5 An account has to be uniquely associated to a person
D.6 Each store admin should provide correct data about the store capacity
D.7 Each registered store has a QR code reader
D.8 Each registered store has a device as a proxy for fallback option
D.9 Customers have to scan the QR code provided by the system
D.10 Users spend an amount of time at the store reasonably similar to the average computed by statistics
D.11 Users who specify the expected duration of the visit should observe it
D.12 Users should buy the same product categories they specified on the app
D.13 Users should show up on time
D.14 A ticket must correspond only to one entering person and he corresponds to the account’s owner
D.15 Users who use the app should not use the fallback option

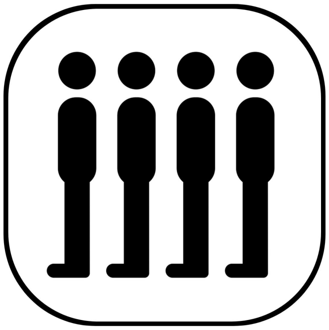
3. Specific Requirements

3.1. External Interface Requirements

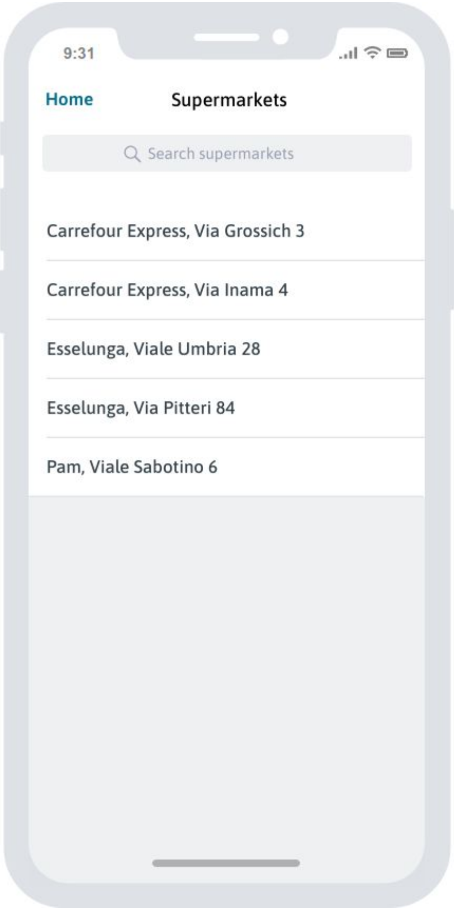
3.1.1. User Interfaces

The following mockups are meant to communicate the look and feel of the app.

Icon of the app



Supermarket Search and User Interfaces to select functionality



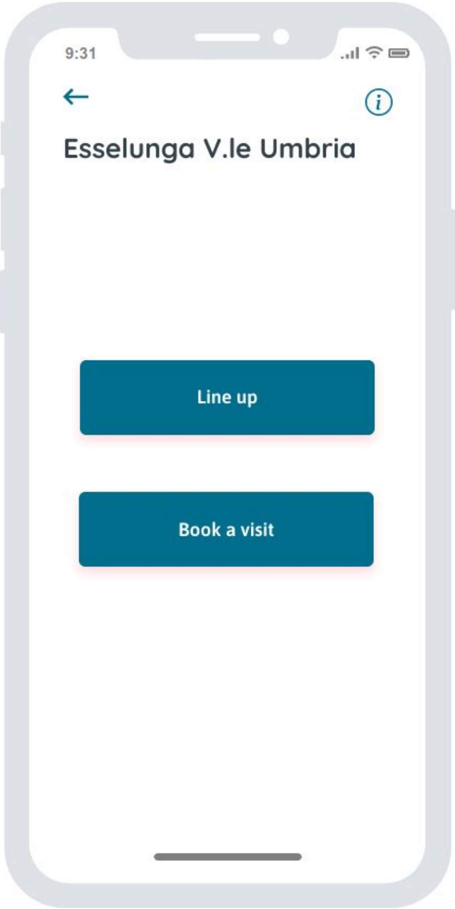
Carrefour Express, Via Grossich 3

Carrefour Express, Via Inama 4

Esselunga, Viale Umbria 28

Esselunga, Via Pitteri 84

Pam, Viale Sabotino 6



3.1.2. Hardware Interfaces

The system has three main functions and three main types of actor, which require different hardware interfaces.

- **Get a queue ticket and book a visit**
  The users must own a smartphone to get a queue ticket or to book a visit. In particular the smartphone has to be equipped with a **GPS sensor** for getting a queue ticket. Infact, the system calculates the commute time based on the user's GPS position.For providing the fallback option stores must have an **on site device** connected to the system through an internet connection that issues tickets for the queue.The supermarket must have an automatic **QR code scanner** connected to the system through an internet connection and placed at the entrance that lets people in the supermarket. Another QR code scanner is required at the exit of the store to allow customers exit.

- **Monitor statistics**
  Store Managers can manage store's information and monitor statistics through a web app connected through internet connection. Therefore they must own a **personal computer or a device** able to navigate the web.

3.1.3. Software Interfaces

The system uses the following external software interfaces:

- **Maps Service**
  The system uses a map service that allows it to compute the time to commute from the user location, taken from the smartphone GPS, the store location and the means of transport indicated.

- **QR code scanner**
  The system must communicate with the QR code scanner's software to get tickets data and validate them.

3.1.4. Communication Interfaces

The system uses the HTTPS protocol to transmit data over the internet.

3.2. Functional Requirements

3.2.1. Requirements

Definition of use case diagrams, use cases and associated sequence/activity diagrams, and mapping on requirements

List of requirements:

R1. Users must be able to register to the app, inserting personal information
R2. The system allows Store Managers to register their supermarket inserting the required information.
R3. Upon registering the supermarket, Store Managers must register themselves, providing a document that certifies their role.
R4. The system allows users to log in to access the functionalities of the app.
R5. The system allows store managers to log in to monitor influx of people
R6. The system allows users to select the supermarket they want to line up or book a visit for
R7. Users can get a ticket with a QR code to virtually line up
R8. Users who already have a valid ticket may not get another one or book a visit
R9. When getting a ticket to line up, the system must compute the time it takes for the user to get to the supermarket based on their position and chosen mean of transportation
R10. The system allows users to book a visit if capacity constraints are satisfied
R11. The system allows users to choose a day and a time for their visit among availables
R12. Upon booking a visit, users must insert product categories they intend to buy
R13. Upon booking a visit, users must insert how long they intend to stay at the supermarket. Long term customers may not insert such information and the system will infer it from previous visits.
R14. The system should send a reminder to the user, at a time based on the estimated commute time.
R15. The system must be able to let customers in the store by scanning the ticket's QR code.
R16. The system allows users to exit the store by scanning the same QR code provided at the entrance.
R17. The system allows guests to get a ticket to line up from the proxy device.
R18. The system allows one person to enter for each ticket

3.2.2. Mapping

Goals	Domain Assumptions	Requirements
G.1	D1, D2, D4, D5, D6, D7, D8?, D9, D11, D12, D14	R1, R2, R4, R5, R7, R8, R10, R12, R13, R15, R16, R17, R18

G.2	D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D14	R1, R2, R4, R6, R7, R9, R14, R15, R16, R17, R18
G.3	D1, D2, D4, D5, D6, D7, D9, D11,D12, D14	R1, R2, R4, R6, R10, R11, R12, R13, R15, R16,
G.4	D1, D2, D5, D6,D7, D9, D10, D11, D12, D14	R1, R2, R4, R5, R8,R10, R12, R13, R15, R16, R18

3.2.3. Use Cases

Here we include the most relevant use cases:

- **Store Registration**

Name	Store registration
Actors	Store manager
Entry Condition	Store manager has access to the webapp through a working internet connection
Event Flow	1. The store manager accesses the webapp 2. He clicks on “Register” 3. Inserts his data and fills mandatory fields 4. Inserts store data 5. Inserts a document that proves that he is the manager of that supermarket 6. System validates document 7. System accepts registration 8. System inserts the store and manager data in its database
Exit Conditions	The store manager and its store are successfully registered into the system
Exception	The manager or the store are already

	registered Certification is not valid

- **User Registration**

Name	User registration
Actors	Guest
Entry Condition	A person has downloaded the app and has a working internet connection
Event Flow	1. Clicks on “Register” 2. Inserts his data and fills mandatory fields 3. Account is registered by the system
Exit Conditions	The guest now is registered into the system and becomes a User
Exception	User is already registered Email is already associated to another account

- **Store Manager Login**

Name	Store manager login
Actors	Store manager
Entry Condition	Store manager is registered to the system Store manager has a working internet connection
Event Flow	1. Store manager connects to the webapp 2. Clicks on Login 3. Enters username and password 4. System checks credentials validity

	5. System redirect to home page
Exit Conditions	The store manager is logged in and can access system functionalities
Exception	Store manager enters invalid username or password

- **User Login**

Name	User login
Actors	User
Entry Condition	The user is registered to the app The user has a working internet connection
Event Flow	1. User opens the app 2. Clicks on Login 3. Insert username and password 4. System checks credentials validity 5. The app shows the home
Exit Conditions	User is logged in and can accesses the services of CLup
Exception	Username or password are incorrect

- **Get queue ticket**

Name	Get queue ticket
Actors	User

Entry Condition	A user is logged in and has a working internet connection The store is currently open
Event Flow	1. The user selects a store where he wants to shop 2. Clicks on get a ticket 3. The system collects the user position from the GPS 4. The system calculates the time required for the user to reach the store 5. The system generates the ticket number 6. The system generates the QR code associated to the ticket 7. The system stores the ticket data 8. The system computes the queue waiting time 9. The system sends QR code and waiting time to the user 10. The system notifies the user when it’s almost his turn
Exit Conditions	The user has a valid queue ticket for entering the indicated store
Exception	The user already has a valid ticket

- **Book a visit**

Name	Book a visit
Actors	User
Entry Condition	The user is logged in He has a working internet connection

Event Flow	1. The user selects a store where he wants to shop 2. Clicks on book a visit 3. Chooses one of the available time slots for a specific day 4. Chooses one of the available products ‘categories based on departments available capacity 5. Confirms the booking 6. The system generates the QR code associated to the ticket 7. The system stores the booking data
Exit Conditions	The user has a valid ticket for shopping at the indicated time
Exception	The user already has a valid ticket

- **Fallback option**

Name	Fallback option
Actors	Guest
Entry Condition	The person is not registered to the app Proxy device is connected to the system through a working internet connection
Event Flow	1. A customer requests a ticket to the proxy at the store 2. The system generates a ticket number 3. The system generates a QR code associated to the ticket

	4. The system stores the ticket data 5. The proxy device prints a QR code with queue waiting time 6. The customer pick up the ticket
Exit Conditions	The customer has a valid queue ticket for entering the store when it’s his turn
Exception	The device stop working for hardware problems

- **Ticket entry validation**

Name	Ticket entry validation
Actors	User Guest
Entry Condition	The user has a ticket for entering the store The guest has a ticket retrieved at the proxy QR code scanners have a working internet connection
Event Flow	1. A customer is entering the store 2. Customer scans the QR code at the machine 3. The machine reads the QR code 4. System checks ticket validity 5. The ticket is valid 6. System opens the door 7. System updates that the ticket has been validated at the entrance 8. System saves the customer’s

	entry time 9. System allows the customer to exit the store once the same QR code will be scanned
Exit Conditions	The customer has been allowed to enter the store Ticket is not valid anymore for entering the store Ticket is valid for exiting the store
Exception	Ticket shown is not valid(because already validated or customer has shown up too late)

- **Ticket exit validation**

Name	Ticket exit validation
Actors	User Guest
Entry Condition	User/Guest has already scanned the code at the entrance and he is inside the store QR code exit scanner are connected to the system through an internet connection
Event Flow	1. Customer is exiting the store 2. Scans the code at the exit doors 3. The system checks that the same ticket was validated at the entrance and is valid for the exit 4. System opens the exit doors 5. System invalidates the ticket for future exits

	6. System stores the exit time 7. System computes the time spent by the customer at the store and updates statistics for registered users
Exit Conditions	Customer exited the store Ticket is not valid anymore for exiting the store
Exception	Ticket scanned has been already scanned for a previous exit Ticket scanned is different from the one previously scanned at the entrance

- **Store Manager visualizes statistics**

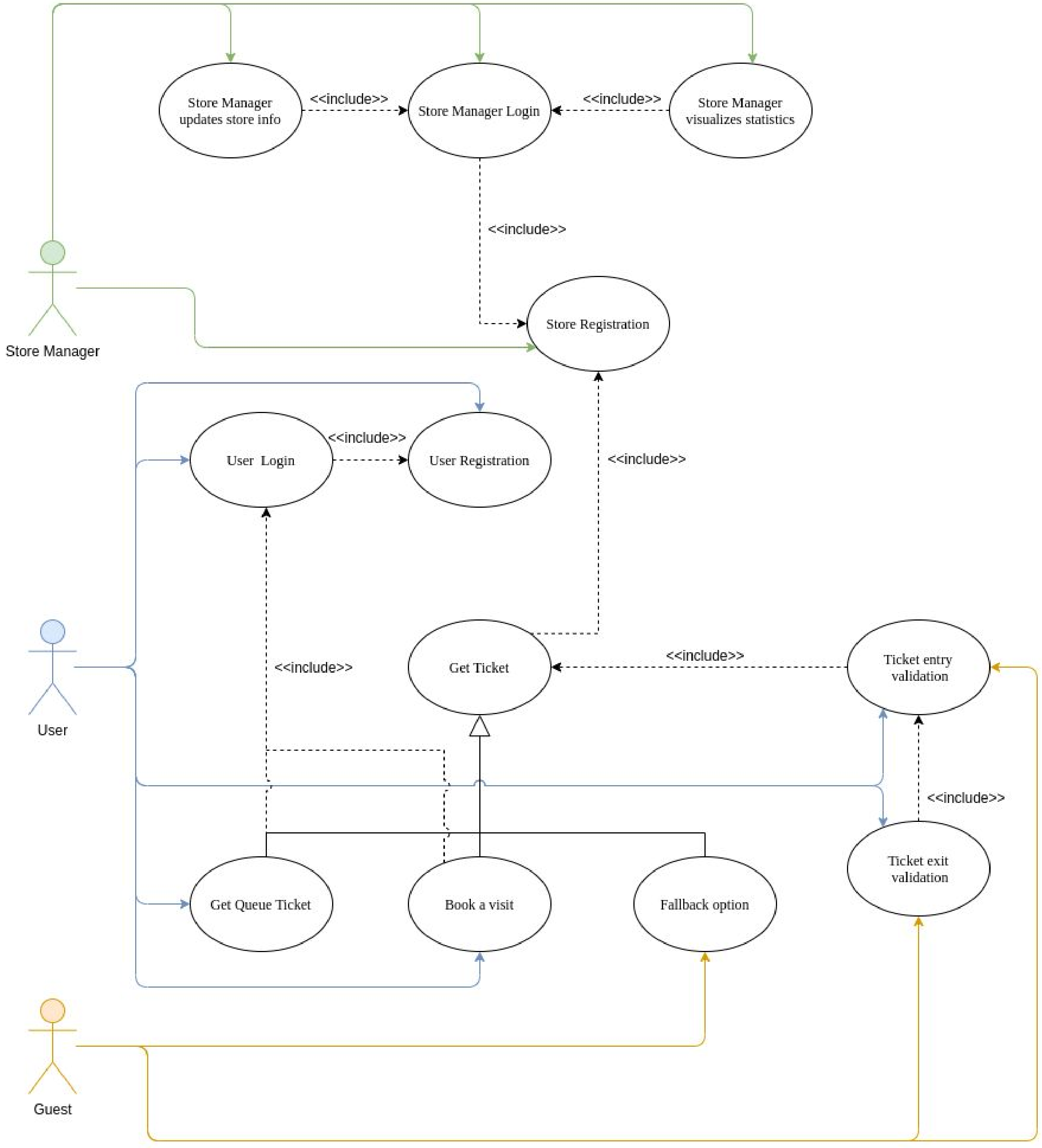
Name	Store manager visualizes statistics
Actors	Store manager
Entry Condition	Store manager has logged in through the webapp Store manager has a working internet connection
Event Flow	1. The store manager clicks on “show statistics” 2. The system retrieves number of people who are currently shopping at the supermarket 3. The system retrieves statistics on shopping at the supermarket 4. The system displays data
Exit conditions	Store manager can visualize data and statistics

Exception	No data are available

- **Store Manager updates store info**

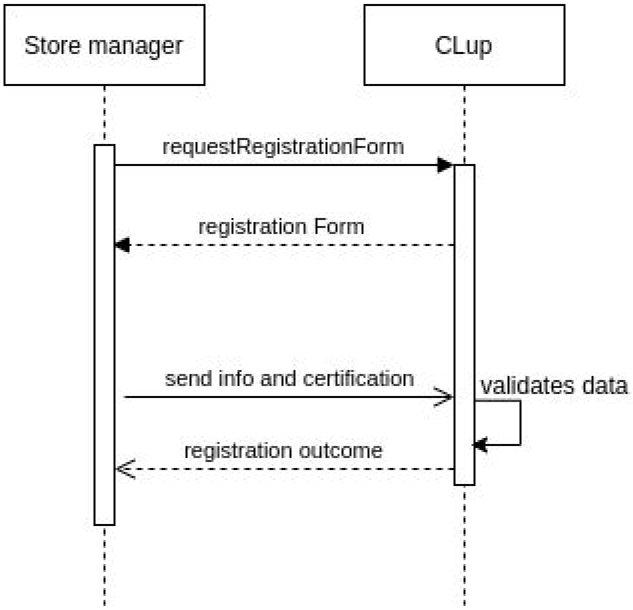
Name	Store manager updates store info
Actors	Store manager
Entry Condition	Store manager has logged in through the webapp Store manager has a working internet connection
Event Flow	1. The store manager clicks on update store info 2. Store manager insert updated data about store departments, product categories and capacities 3. Clicks confirm 4. System checks if changes are consistent, e.g. a new department inserted was actually already in the DB 5. System accepts changes
Exit conditions	Store capacity and other data are updated
Exception	Inconsistencies in updating data

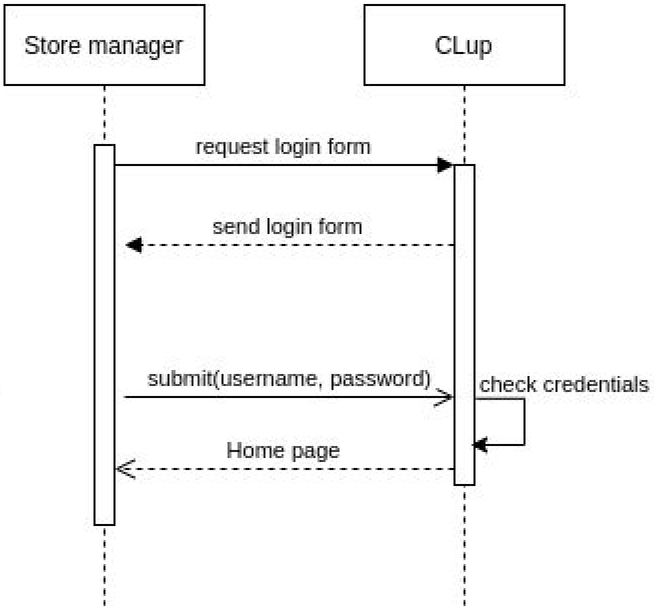
Use case diagrams



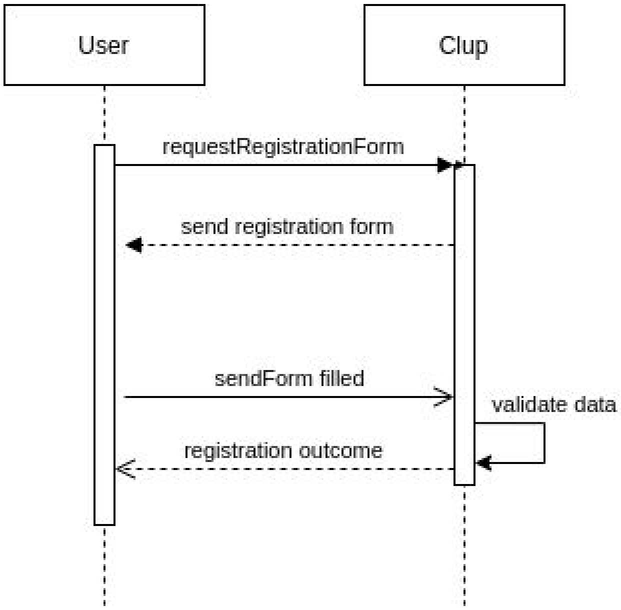
Sequence diagrams

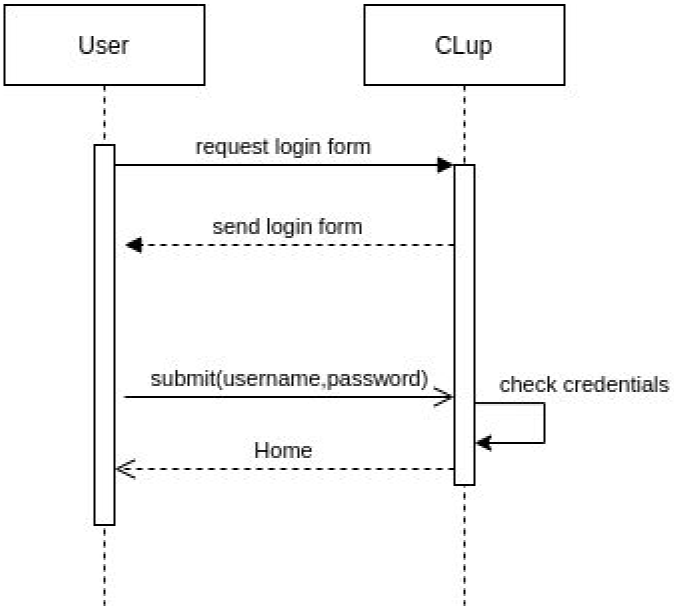
Store Registration & Store Manager Login



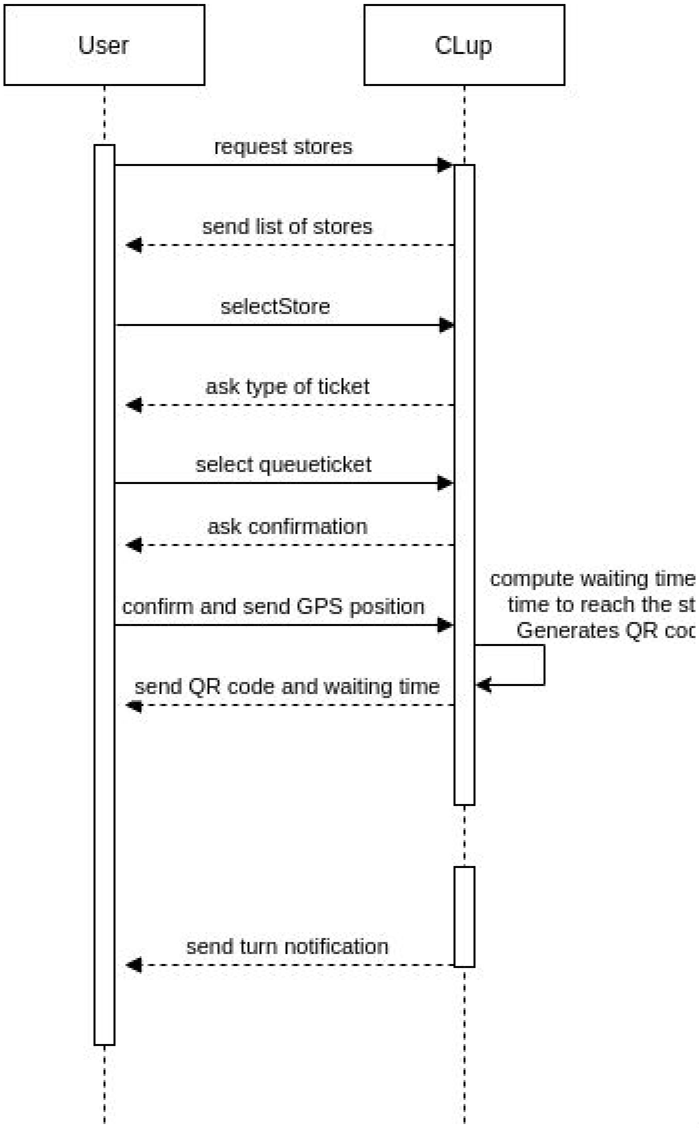


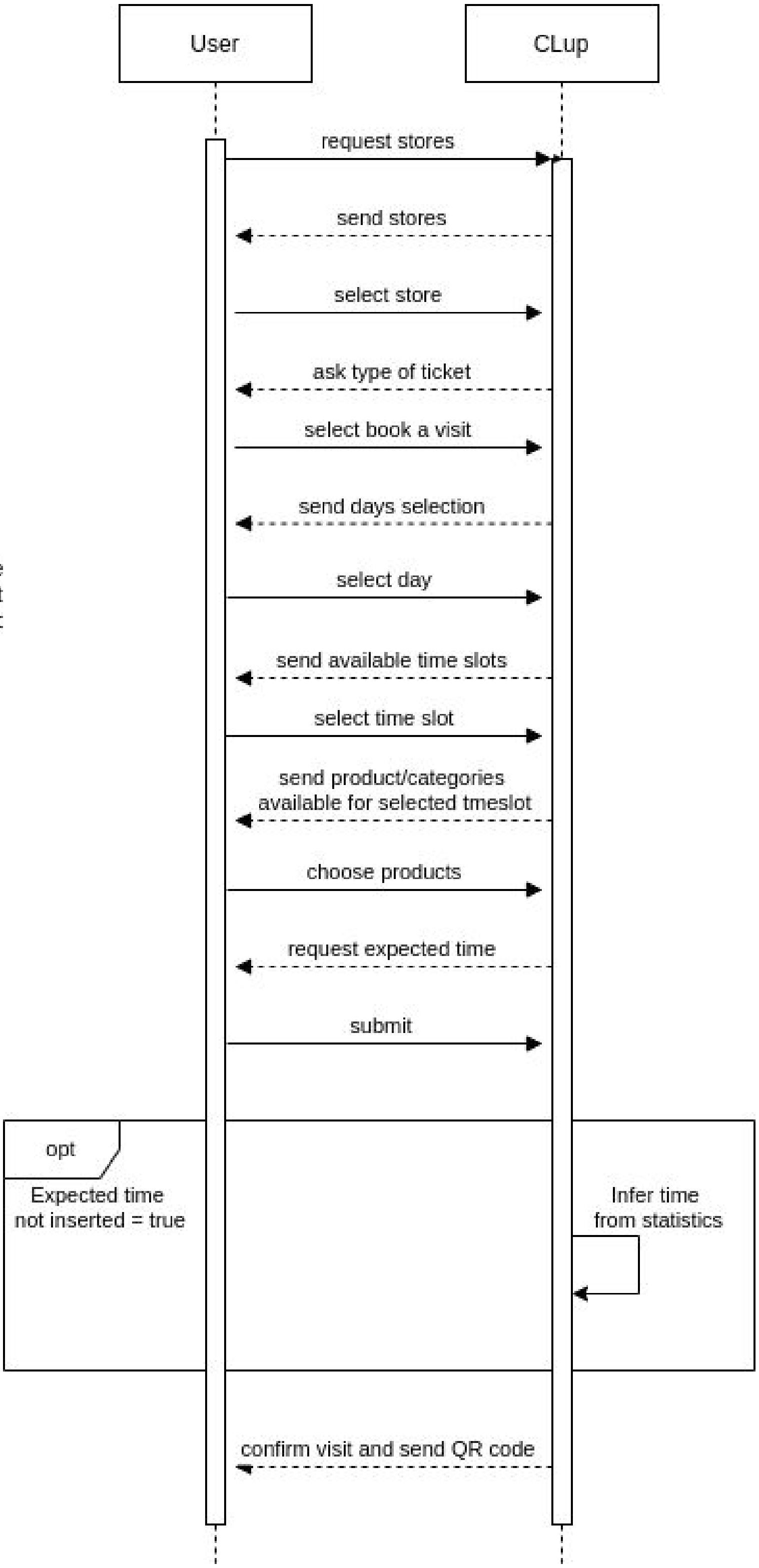
User Registration & User Login



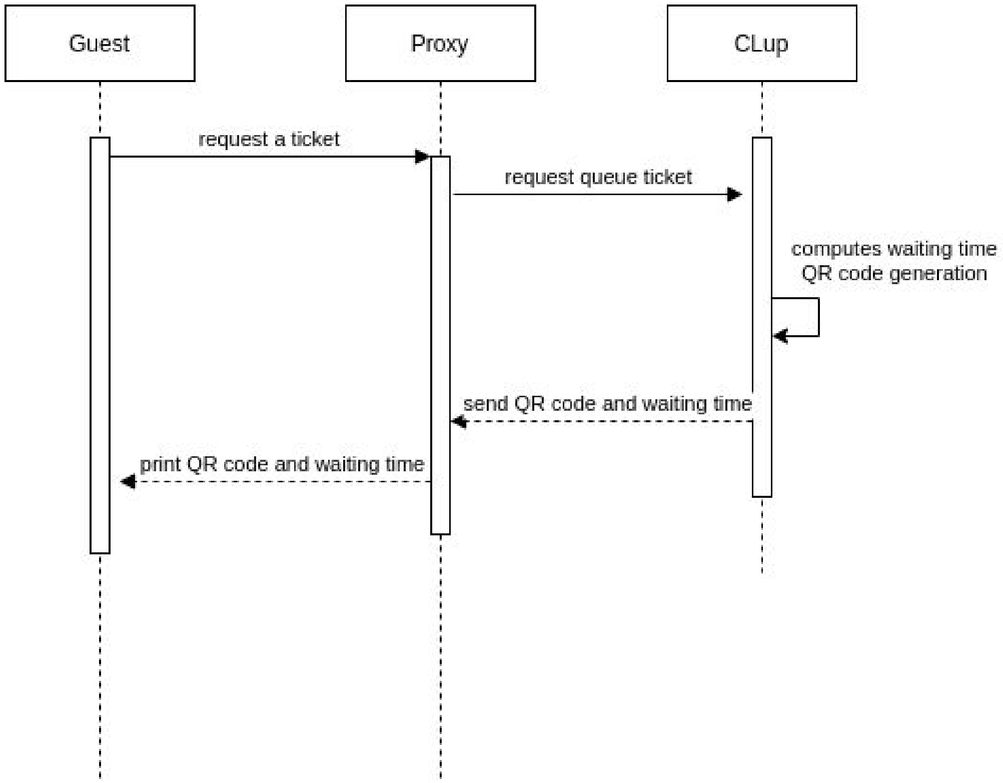


Get a Queue Ticket & Book a visit

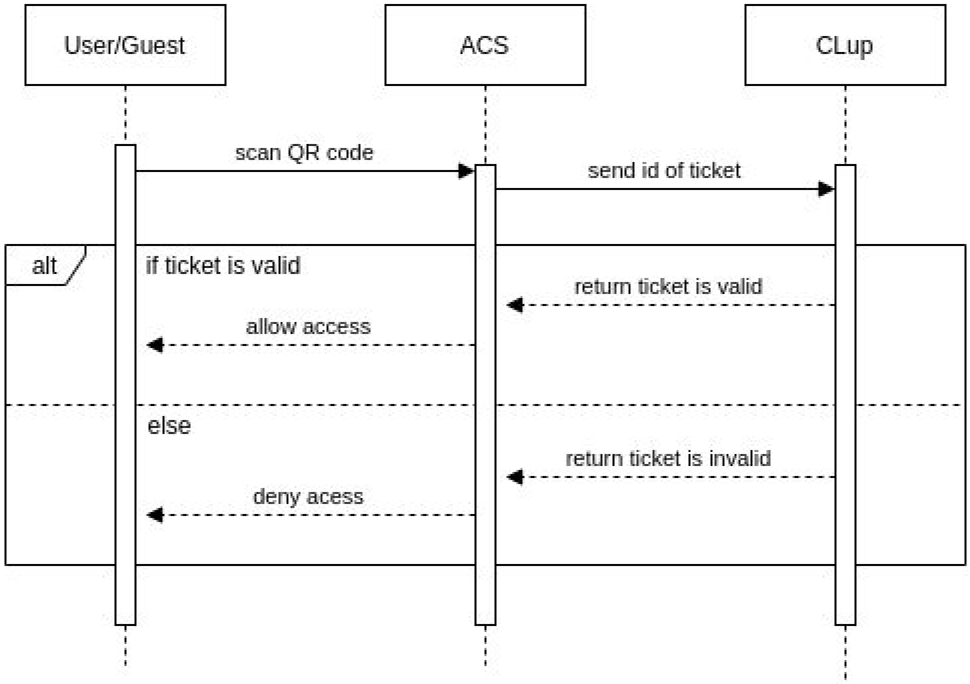




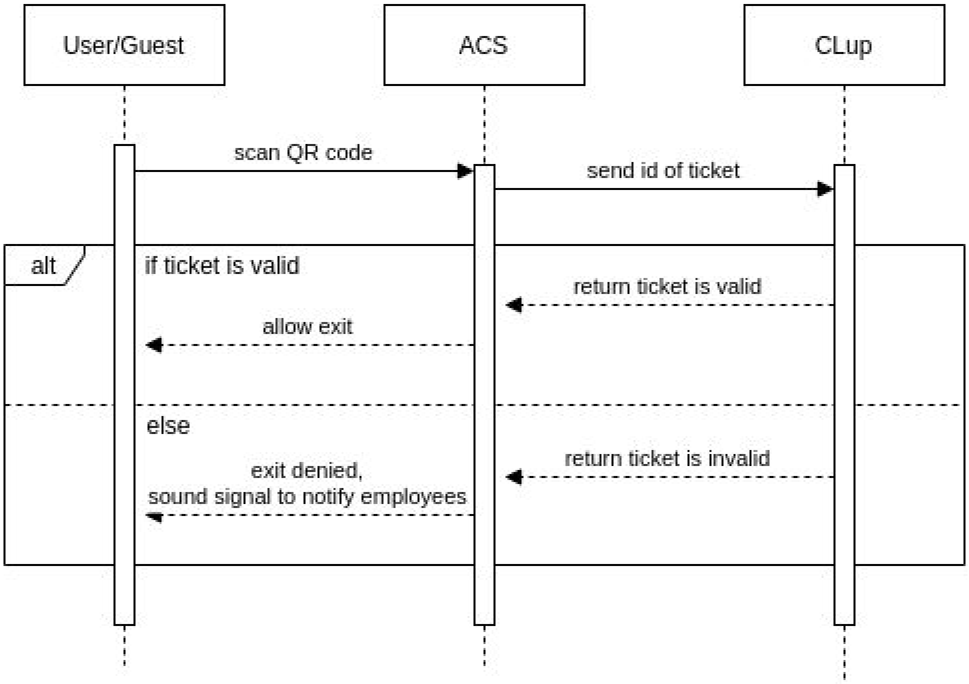
Fallback Option



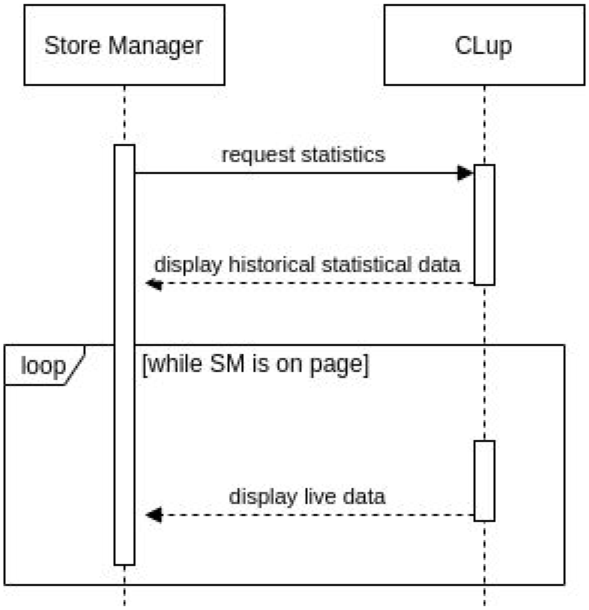
Ticket entry validation



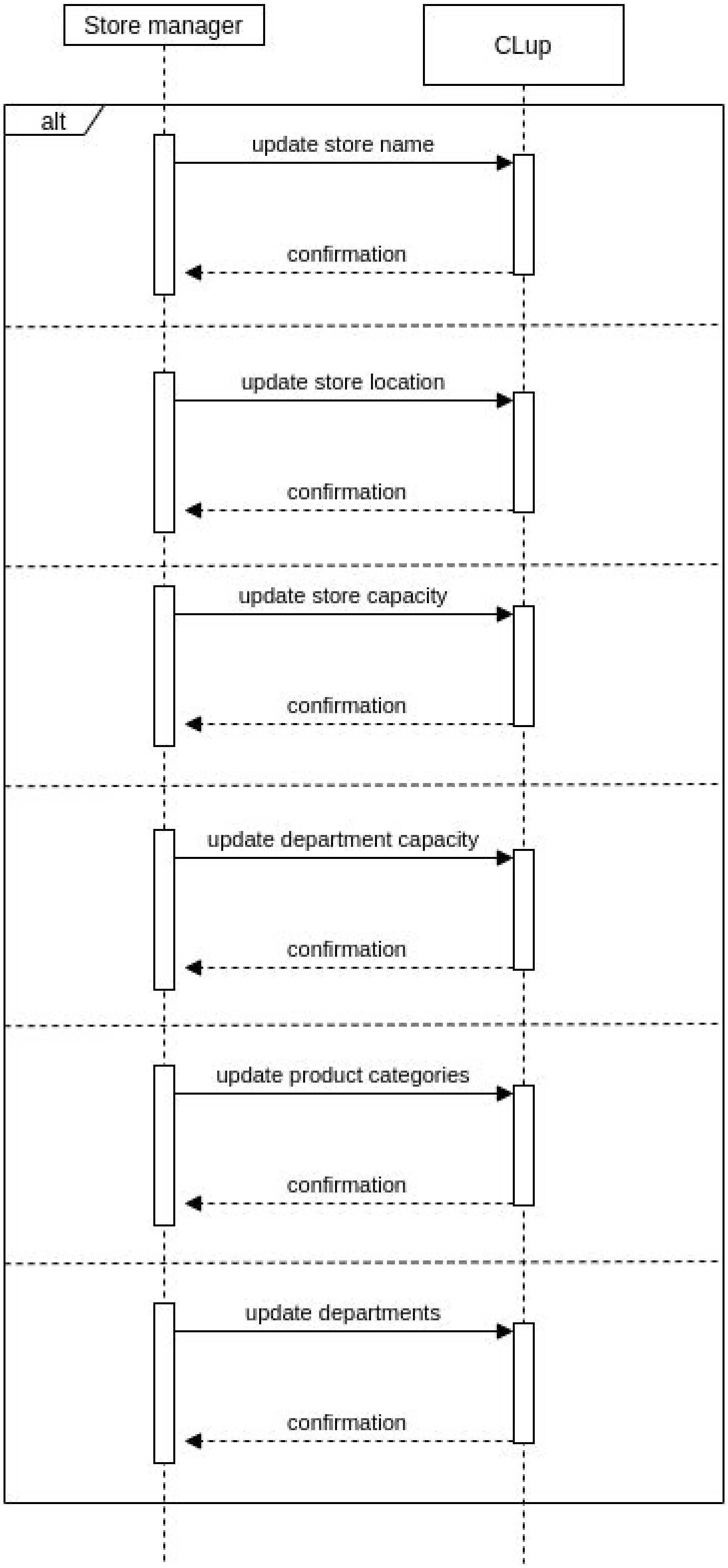
Ticket exit validation



Store Manager visualizes statistics



Store Manager updates store info



3.3. Performance Requirements

The system should have a short response time, it means that whenever a user requests a ticket both for the queue and the visit, the system should send it to the user's device within 10 seconds. Furthermore, the system should synchronize in a reasonable time the booked tickets with the QR code scanners at the entrance in order to reduce time for letting customers in. The number of connected people should be at least 200.000, if we consider that CLup software is going to be used in a city like Milan.

3.4. Design Constraints

3.4.1. Standards compliance

The code should follow the requirements contained in this document. In addition, it should be commented in a clear way in order to make it comprehensible to others.

3.4.2. Hardware limitations

The CLup software's users need a mobile with GPS sensor.

Store managers can access the software's functionalities through a webapp running on a web browser.

The QR code scanners must be able to scan the code provided by customers through working cameras.

The proxy device (fallback option) must be equipped with a printer to issue tickets.

All the previously mentioned devices must be connected to the system with a working internet connection to make them send/receive data.

3.5. Software System Attributes

3.5.1. Reliability

The reliability of this software is critical, as it is meant to be used by the general public and has a safety critical purpose. In fact, since this software has a great impact on the influx of people management, it should always output correct results for all of its functionalities. Moreover, in order to make every functionality work, it should also be able to detect and prevent hardware faults, since this system heavily depends on several physical devices.

3.5.2. Availability

The CLup software should be up as long as possible. Since the whole entries of customers at stores depend on the system, its failure would jeopardize the stores' business. Therefore it should be available upwards of 99% of the time. Maintenance should be done during the night to minimize the impact of any eventual failure. The system should have a proper infrastructure that allows it to recover backed-up data in case of failure. Furthermore, a distributed architecture that replicates services in different locations, should be in place in order to minimize the chances of downtime.

3.5.3. Security

Since this software has a huge impact on public health and contains data of potentially millions of people, its security is an absolute priority. In particular, store managers' passwords and users data should be encrypted in order to avoid store capacity's data to be changed by malicious people, with a consequence on stores' crowding. Also tickets' data can be accessed by authorized people and ticket's owners.

3.5.4. Maintainability

The code must be written in an object-oriented language, in order to ensure a high level of abstraction. The decision falls on Java. Also commented code is required to make the software more readable, as well as thorough documentation. Unit and integration tests must come along with every addition to the code, which has to be covered by at least 80%.

3.5.5. Portability

The software must run on different devices and operating systems such as Windows, MacOS. As far as mobile devices are concerned it must run on Android and iOS.

3.5.6. Scalability

High level of abstraction has to guarantee future improvements on the software. It also has to guarantee a bigger number of connected users without making remarkable changes.

4. Formal Analysis Using Alloy

This section should include a brief presentation of the main objectives driving the formal modeling activity, as well as a description of the model itself, what can be proved with it, and why what is proved is important given the problem at hand. To show the soundness and correctness of the model, this section can show some worlds obtained by running it, and/or the results of the checks performed on meaningful assertions.

4.1. Alloy code

- Users can retrieve one ticket at a time
- Users may only choose available products in book a visit feature
- A ticket is associated only to a person
- Ticket status affect user's entry/exit authorization

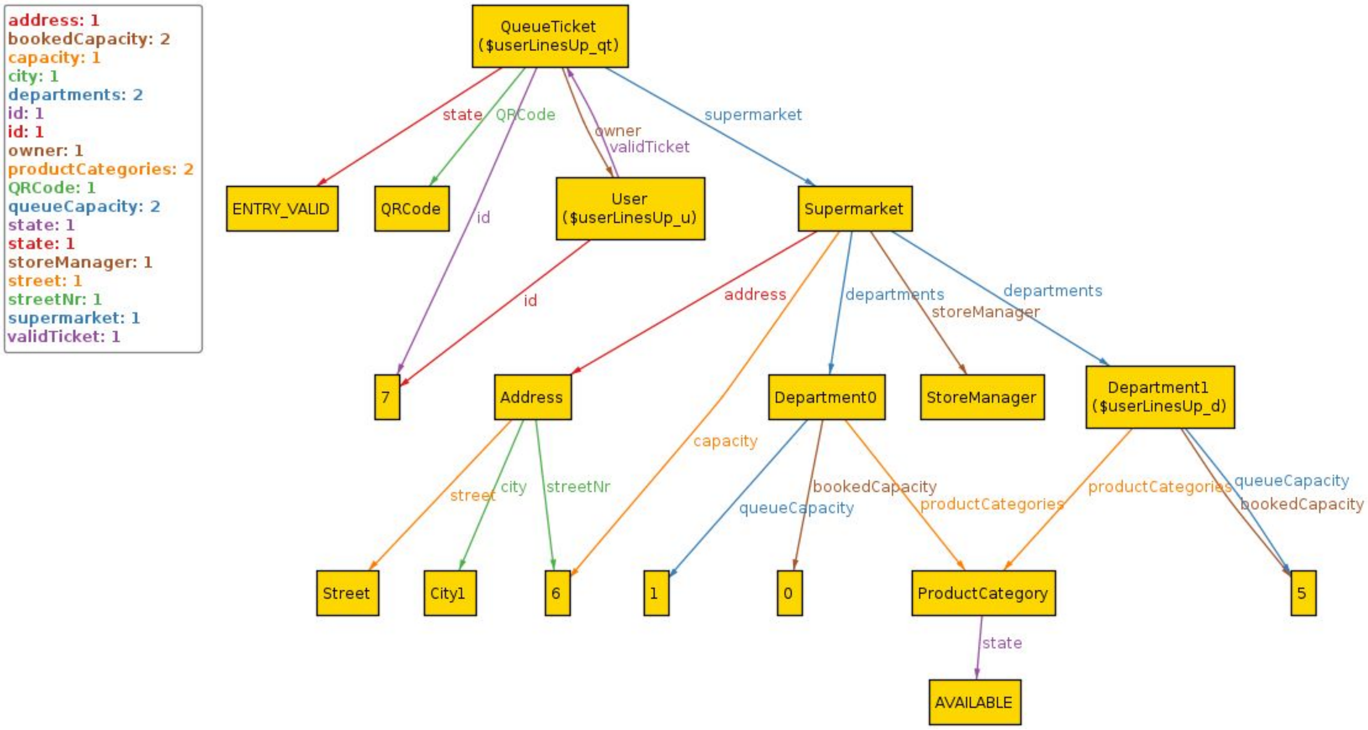
```
sig City {}
sig Street {}
sig Address {
city: one City,
street: one Street,
streetNr: one Int
} {
streetNr > 0
}

sig DateTime {
date: one Date,
time: one Time
}

sig Date {
year: one Int,
month: one Int,
day: one Int
} {
year > 0
month >= 1
day >= 1
}

sig Time {
hour: one Int,
minute: one Int,
second: one Int
} {
hour >= 0
minute >= 0
second >= 0
}

sig User {
id: one Int,
expiredTickets: set Ticket,
validTicket: h Ticket
} {
id > 0
disj [validTicket, expiredTickets]
}

sig StoreManager {}

sig Supermarket {
address: one Address,
storeManager: one StoreManager,
departments: some Department,
capacity: one Int
}

sig Department {
bookedCapacity: one Int,
queueCapacity: one Int,
productCategories: some ProductCategory,
} {
bookedCapacity >= 0
queueCapacity >= 0
}

sig QRCode {}

abstract sig TicketState {}
one sig ENTRY_VALID extends TicketState {}
one sig EXIT_VALID extends TicketState {}
one sig EXPIRED extends TicketState {}

abstract sig Ticket {
id: one Int,
supermarket: one Supermarket,
QRCode: one QRCode,
state: one TicketState,
entryTime: lone DateTime,
exitTime: lone DateTime
} {
id > 0
}

sig FallbackTicket extends Ticket {}

sig QueueTicket extends Ticket {
owner: one User
}
```

```
sig BookedTicket extends Ticket {
owner: one User,
slot: one TimeSlot,
chosenProdCategories: some ProductCategory
}

sig TimeSlot {
date: one Date,
startTime: one Time,
endTime: one Time
}

abstract sig AvailabilityState {}
sig AVAILABLE extends AvailabilityState {}
sig UNAVAILABLE extends AvailabilityState {}

sig ProductCategory {
state: one AvailabilityState
}



//FACTS
fact uniqueUserId {
no disj u1, u2: User | u1.id = u2.id
}

fact supermarketsHaveDifferentManagers {
no disj s1, s2:Supermarket | s1.storeManager = s2.storeManager
}

fact supermarketsHaveDifferentLocations {
no disj s1, s2:Supermarket | s1.address = s2.address
}

fact uniqueTicketId {
no disj t1, t2: Ticket | t1.id = t2.id
}

fact uniqueQRCode {
no disj t1, t2: Ticket | t1.QRCode = t2.QRCode
}

fact productCategoriesInDepartmens {
```

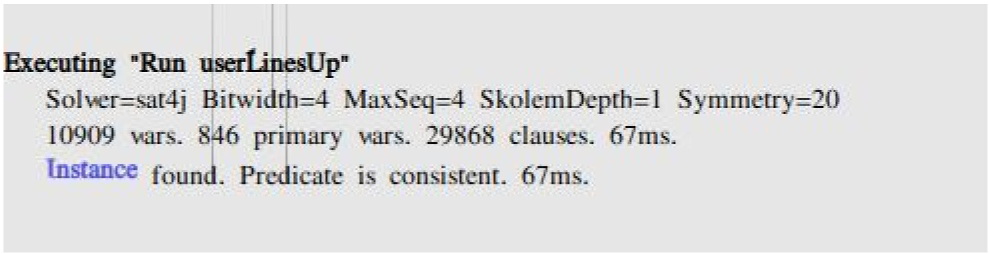
```
all p: ProductCategory, d: Department, s: Supermarket |
p in d.productCategories and d in s.departments
}

fact ticketQueueOwnerOwnsTicket{
all t: QueueTicket, u : User | (t.owner = u) iff (t in u.expiredTickets or t = u.validTicket)
}

fact bookTicketOwnerOwnsTicket {
all t: BookedTicket, u: User | (t.owner = u) iff (t in u.expiredTickets or t = u.validTicket)
}

fact  userOwnsAValidStateTicket {
all u: User | u.validTicket != none
implies (u.validTicket.state = ENTRY_VALID or u.validTicket.state = EXIT_VALID)
}

fact ticketStateConstraints {
all t: Ticket | t.entryTime = none iff t.state = ENTRY_VALID
all t: Ticket | (t.entryTime != none and t.exitTime = none) iff t.state = EXIT_VALID
all t: Ticket | t.entryTime = none implies t.exitTime= none
all t: Ticket | t.exitTime != none implies t.entryTime != none
all t: Ticket | (t.entryTime != none and t.exitTime != none) iff t.state = EXPIRED
}

fact pastTicketsAreExpired {
all u: User, t: Ticket | (t.state = EXPIRED iff t in u.expiredTickets)
}

fact oneTicketOneOwner {
no disj u1, u2: User | u1.validTicket = u2.validTicket
all disj u1, u2: User | u1.expiredTickets & u2.expiredTickets = none
}

fun sumDepartmentsCapacity[dpts: set Department] : one Int {
sum x : dpts | x.bookedCapacity + x.queueCapacity
}

fact totalCapacity {
all s: Supermarket | sumDepartmentsCapacity[s.departments] = s.capacity
}

//PREDICATES
```

```
pred userBookedVisit(u: User, bt: BookedTicket, pc: ProductCategory, d: Department, s:
Supermarket) {
d in s.departments
pc in d.productCategories
u.validTicket = bt
bt.owner = u
d.bookedCapacity = 5
pc.state = AVAILABLE
#ProductCategory = 2
}

run userBookedVisit

pred userLinesUp(u: User, qt: QueueTicket, d: Department) {
d.queueCapacity = 5
u.validTicket = qt
qt.owner = u
}

run userLinesUp
```

Metamodel of predicate userLinesUp

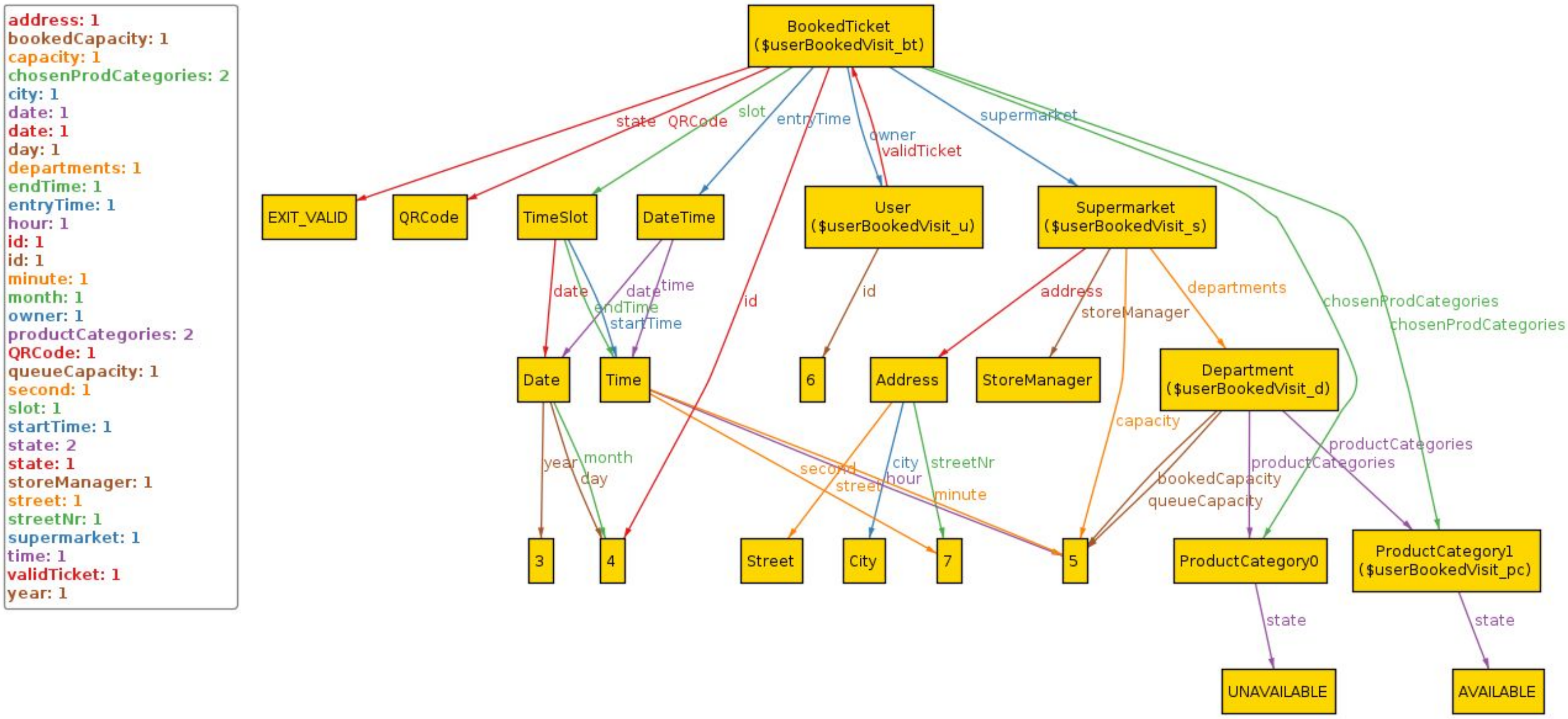


Result of execution

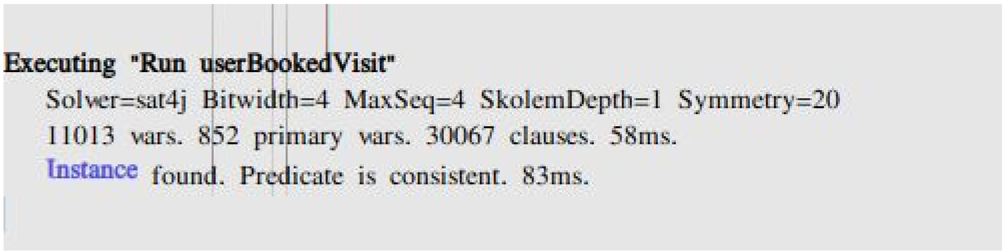


```
Executing "Run userLinesUp"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   10909 vars. 846 primary vars. 29868 clauses. 67ms.
   Instance found. Predicate is consistent. 67ms.
```

Metamodel of predicate userBookedVisit



Result of execution



```
Executing "Run userBookedVisit"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   11013 vars. 852 primary vars. 30067 clauses. 58ms.
   Instance found. Predicate is consistent. 83ms.
```

5. Effort Spent

In this section you will include information about the number of hours each group member has worked for this document.

Description	Tot
Read requirements, initial discussion	3
Index and document format	2
Goals, definitions, phenomena	6
World/machine/shared phenomena	7
UML, State diagram	5
Completed UML, state diagram	3
Product Functions	2
Functions and Actors	3
Hardware and Functional requirements	5
Use Cases definitions	3
Use Cases definitions and diagrams	6
Sequence diagrams	6
Scenarios	4
Alloy	20
Final Revision	2

6. References

- 830-1984 - IEEE Guide for Software Requirements Specifications
- Software Engineering 2 Course Slides
- Alloy Official Documentation - https://alloytools.org/