POLITECNICO
MILANO 1863

A.Y. 2020/2021

Computer Science and Engineering

Software Engineering 2 Project

CLup - Customers Line-up

Requirement Analysis and Specification Document

- -
- -
- -

Deliverable:	RASD
Title:	CLup - Requirement Analysis and Verification Document
Authors:	
Version:	1.0
Date:	2020-12-22
Download page:	https://github.com/rb-sl/SoftEng2Project2020
Copyright:	© 2020 – All rights reserved

Contents

List of Figures

1 Introduction

1.1 Purpose

This document represents the RASD (Requirement Analysis and Specification Document).

The aim of this document is to completely describe the system in terms of functional and non-functional requirements, analyse the many different ways that the users may behave in order to model the system, show the constraints and the limit of the software and indicate the typical use cases that will occur after the release.

This document is the baseline for project planning and estimation of size, cost, schedule. It can be also an useful starting point for the legal binding contract.

1.2 Scope

1.2.1 Problem description

CLup is a software-to-be that will help stores and customers preventing dangerous crowds during shopping due to the current Covid-19 Pandemic. This system should offer the possibility to line up from home for stores while still allowing to physically line up for stores. Moreover, it should offer the possibility to book a visit for a chosen store and get notifications about stores occupancy during the week. The system should also allow store managers to add their stores to the system and monitor entrances.

The application has to be easy-to-use, scalable and reliable.

1.2.2 Goals

[G0] Allow prospective e-Customers to register to the application

[G1] Allow Prospective Store Managers to register to the application by adding their store

[G2] Allow e-Customers to line up from home

- **[G2.1]** Allow e-Customers to dequeue

[G3] Allow Store Managers to monitor entrances

[G4] Allow Physical Customers to line up

- **[G4.1]** Allow Physical Customers to dequeue

[G5] Allow e-Customers to book a visit

- **[G5.1]** Allow e-Customers to delete a visit

[G6] Suggest alternative slots to balance the number of customers

[G7] Allow e-Customers to manage slot notifications

1.2.3 Phenomena

This section is about the phenomena that can occur during the life of the application. They are divide into World Phenomena and Shared Phenomena[4]. World Phenomena happen in the real world and are not visible to the application, while Shared Phenomena occur in the real world but are visible to the application. Shared Phenomena can be controlled either by the Machine (application) or by the World:

- (W) means that the shared phenomena is controlled by the world
- (M) means that the shared phenomena is controlled by the machine

World Phenomena

- Going grocery shopping
- An e-Customer follows a path to the store
- A customer chooses what to buy
- A customer follows a path inside the store
- Customers stand too close near the store
- A wing of the store opens or closes
- The store makes special offers

Shared Phenomena

- An e-Customer is localized (M)
- An e-Customer gets a ticket or books a visit (W)
- An e-Customer gets a slot notification (M)
- A Customer gets added to a queue (M)
- A Customer visits the store for a period of time (W)
- An e-Customer chooses a supermarket (W)
- A store is at full capacity (W)
- A store manager changes the opening times (W)

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Available store**: a store that can host other customers, based on closing time
- **Book a visit**: action of requesting entrance at a specific date and time (that will be guaranteed)
- **Delay window**: number of minutes a customer is allowed to enter after their turn is called (if using a ticket) or their entrance time is reached (if after a visit)
- **Get a ticket**: action of requesting the entrance as soon as possible
- **Historical data**: information about precedent store flows stored by date/time
- **Web mapping service**: a service external to the application (such as Google Maps)
- **Long term user**: for a store, an user who visited it for at least a number of times defined by the Store Managers
- Parameters:
  - **M**: the dimension (in minutes) of the delay window specified by the Store Manager
  - **P%**: parameter specified by the Store Manager that indicates the minimum percentage of places that is reserved to tickets
- **Reasonably available store**: a store for which waiting time is less than a user defined parameter
- **Spatial locality**: distance from the user inferior to a user defined parameter
- **Store information**: refers to store id, address, customer capacity, opening times, allowed delay of a customer, items/categories
- **User**: an user of the application. As per the user diagram, the term encompasses both e-Customers and Store Managers

1.3.2 Acronyms and Abbreviations

- **eC**: e-Customer
- **SM**: Store Manager
- **RASD**: Requirement Analysis and Specification Document
- **[Gn]**: n-th goal
- **[Rn]**: n-th requirement
- **[Dn]**: n-th domain assumption
- Internet Protocols:
  - **HTTPS**: Hypertext Transfer Protocol Secure
  - **TLS**: Transport Layer Security
  - **REST**: REpresentational State Transfer

1.4 Revision history

Version	Date	Comment
1.0	2020-12-22	First release

1.5 Reference documents

R&DD Assignment AY 2020-2021, *Software Engineering 2's BeeP page*

1.6 Document Structure

This document is composed of the following parts:

- **Section 1** is an introduction to the document and offers an overview of the project, describing what its goals are and giving pointers to the following sections
- **Section 2** gives an overview of the project, showing its usages and its domain. Actors are also stated in this section, alongside state and class diagrams
- **Section 3** identifies the system's requirements, both functional and non-functional, then presents the application's use cases and its sequence diagrams
- **Section 4** contains an Alloy model of the system with proofs of consistency
- **Section 5** reports the hours spent by this group's components and the tool used during the development of this document

2 Overall Description

2.1 Product perspective

The application's users will be able to interact through either mobile apps or web-based applications (predominantly with an interface optimized for mobile devices).

When used by customers, the system will need to use a web mapping service such as Google Maps to provide a position (if needed) and calculate travel times.

On the other hand, when used in stores, it will need to be connected to QR code readers (either mobile devices or dedicated scanners) ticket printers, and screens.

2.1.1 Scenarios

The following scenarios are provided to describe possible situations during the usage of CLup.

Scenario 1 Nathan, during the pandemic, needs a way to safely go grocery shopping, avoiding long queues out of supermarkets. His friend John suggests him to download the CLup app to line up from home; Nathan is intrigued and immediately downloads it, signs up and enqueues for his favorite shop.

Scenario 2 After being enqueued for about thirty minutes, Nathan receives a notification telling him his turn is close and he needs to leave home. After reaching the supermarket he waits for some minutes until his number gets displayed by the shop; he then passes the QR code shown by the application at a scanner and enters the shop. Once he has done, he scans again the code at the exit and goes home.

Scenario 3 Ms. Margareth is not accustomed to technology, so she does not use a smartphone; as she needs to go grocery shopping, she goes directly to the store and gets a physical ticket, so she can wait outside with few people, as most use CLup. Once she sees her number is reached, she scans the ticket and enters.

Scenario 4 Nathan knows Julie, the owner of a bakery close to his house, so he tells her about CLup. She is curious and downloads the app, registering her shop. She then logs in as Store Manager into the tablet she uses for work, so that her employees can use it to scan codes and monitor the entrance of customers.

Scenario 5 Anthony works for a supermarket that adopted CLup. As a store manager, he has the duty of keeping updated the store's inventory and make customers respect their turns. Luckily, the store decided to install some QR code readers connected to CLup at entrances, so he doesn't have to scan every customer's code himself.

Scenario 6 As Joe had some free time, he decided to get a physical ticket for the supermarket, hoping few people were already lined up. After 10 minutes of being enqueued he receives an important call and needs to leave; he decides to scan his ticket to dequeue and let the people behind him enter some time before.

Scenario 7 Andrew is at home, waiting for his turn after he requested a ticket. Suddenly he receives a notification from CLup, letting him know that someone in front of him dequeued and giving him the possibility to enter earlier than expected. He gladly accepts and gets ready to go.

Scenario 8 Anastasia knows she needs to go shopping for groceries the next day but has a very tight schedule. Luckily, she can use CLup to book a visit for the exact time she chooses, so she requests a reservation. The next day she can enter without having to line up, thus saving much of her time.

Scenario 9 Confident about his free time, Nick booked three visits for the next week. The day before the second one he remembered he was actually busy, so he logged in the application and canceled the reservation.

2.2 Product functions

The major objective of the application is to decrease queues outside stores; this is achieved by allowing customers to line up from home and book visits for specific dates and times. These functions are integrated with alerts telling the user when to leave home, along with possibility to subscribe to notifications about store occupancy throughout the week to better plan store visits.

For customers not using the application, options to take physical tickets and wait outside store are offered.

2.3 User characteristics

Due to the nature of the application, users can span a wide variety of people with different needs of accessibility and functions.

2.3.1 Actors

Actors represent the groups of people that interact with the application. The relations among the active actors are shown in Figure 1.

- **Prospective e-Customer**: A person who would like to use the application as customer
- **Prospective Store Manager**: A person who would like to register their store
- **Physical Customer**: A customer not registered to the application
- **e-Customer**: A customer using the application
- **Store Manager**: A user managing a store

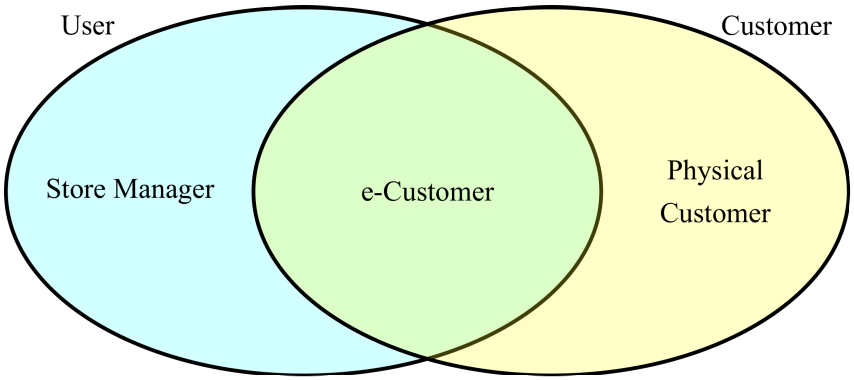


Figure 1: Active actor relations

2.4 Assumptions, dependencies and constraints

2.4.1 Text assumptions

Due to the ambiguity created by the natural language, we made the following assumptions about the specification.

- For each store exists only one store manager; this way, if more people need to use the functions they will need an already logged in terminal
- Some places in stores will need to be reserved to tickets, both in their physical and virtual form, to avoid creating situations where reservations monopolize the entrances
- QR codes will need to be scanned both at entrance and exit, in order to better know the store's occupation and how much time customers spend inside

2.4.2 Domain assumptions

[D0] Prospective users will complete the registration process
[D1] Only people enqueued access the stores (both app and physical)
[D2] Information provided by e-customers corresponds to their will
[D3] Information provided by users upon registration is correct
[D4] Customers enter only if allowed by the application
[D5] Customers respect the choices made during a ticket creation or while booking a visit
[D6] Calculated travel time is correct
[D7] Information about stores inserted by SMs is correct
[D8] Customers entering a store exit after some time
[D9] If a client exits at a given time, he must have entered in the same opening period
[D10] Ticket printers and QR readers work properly
[D11] The internet connection is always working
[D12] Users accept to receive notifications from the application
[D13] Stores show the current customer number
[D14] The external web mapping system is always available and there is signal

2.5 Domain Diagrams

The high-level class diagram of the application can be seen in Figure 4, while Figure 2 contains the state diagram for tickets; the beginning of the $CanEnter$ state corresponds to the start of the delay window. Figure 3 shows the acceptance states of a store during its opening hours.

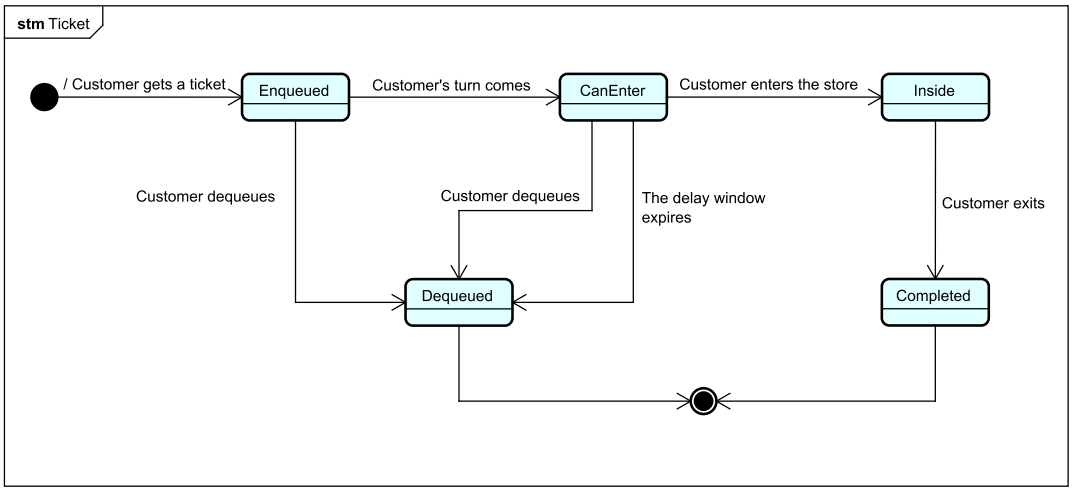


Figure 2: Ticket state diagram

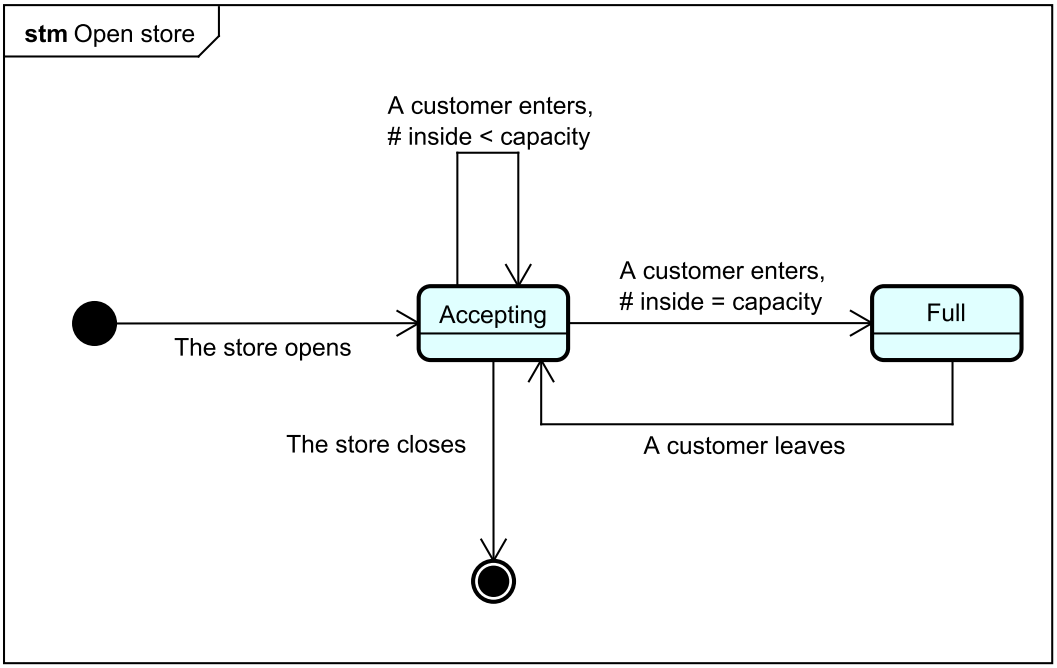


Figure 3: Open store diagram

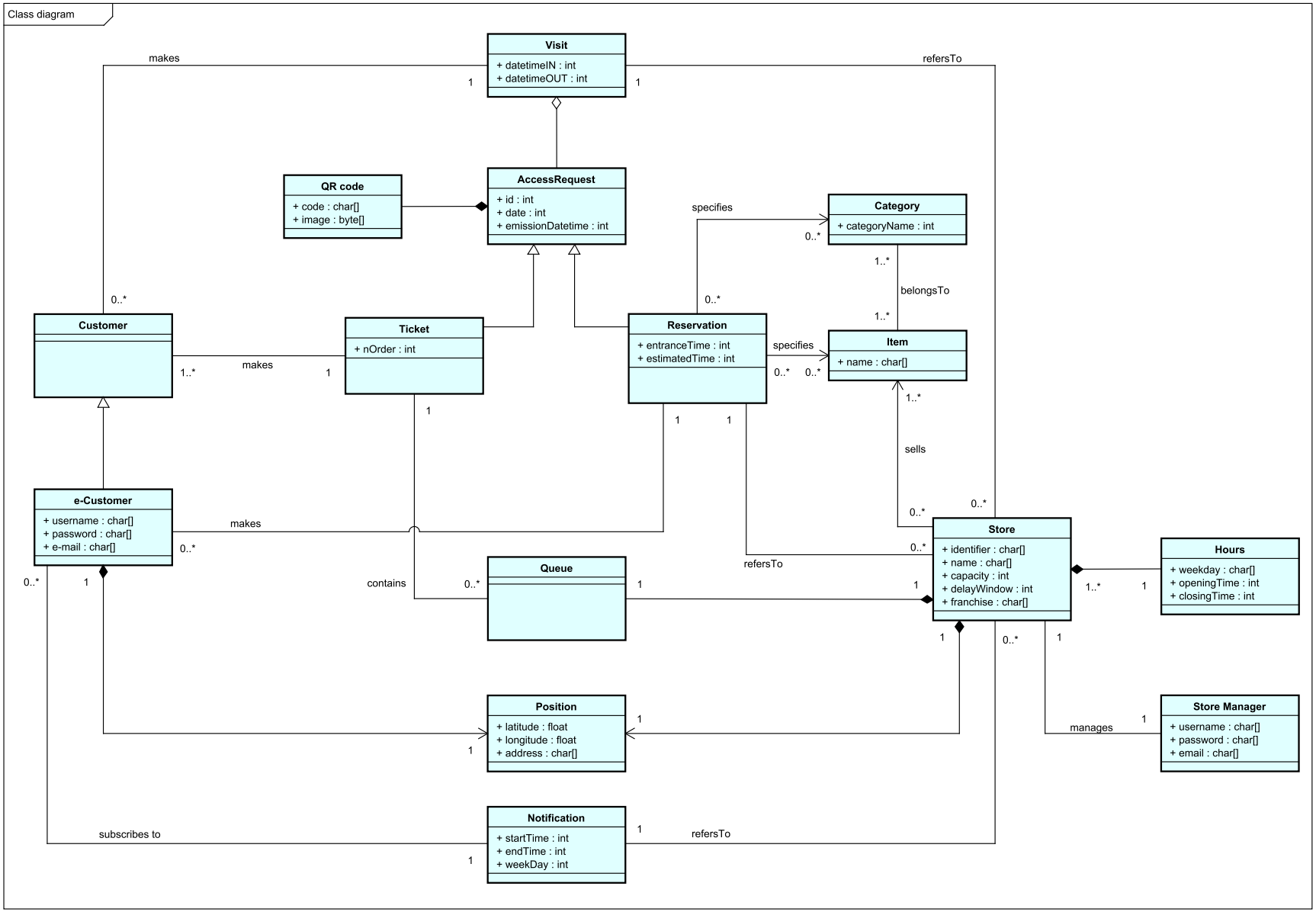


Figure 4: Class Diagram

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

In this section some mock-ups are illustrated for the mobile application; web-based screens are similarly optimized for mobile devices, so they are not reported. An important point of these screens is the overall cleanness aimed at making the interface as intuitive as possible for the potential audience.

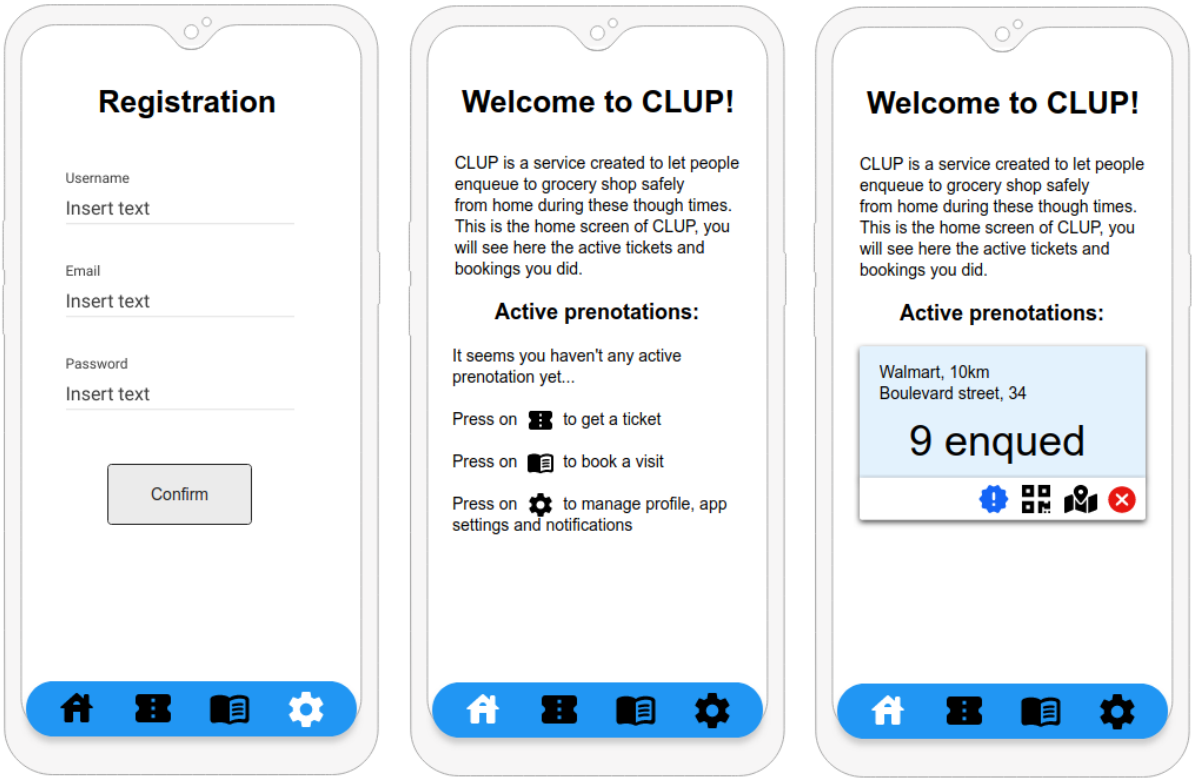
In Figure 5 some welcome screens are shown, at registration time (5a) or for an already logged in e-Customer (in Figure 5c is shown an active ticket).

Figure 6 shows the creation of a ticket, through the insertion of parameters (6a) and the store selection, with queue information (6b).

Specific functions for visit booking are illustrated in Figure 7, with parameters (7a), optional items (7b) and visit information (7c). The reservation completes after a screen analogous to Figure 6b.

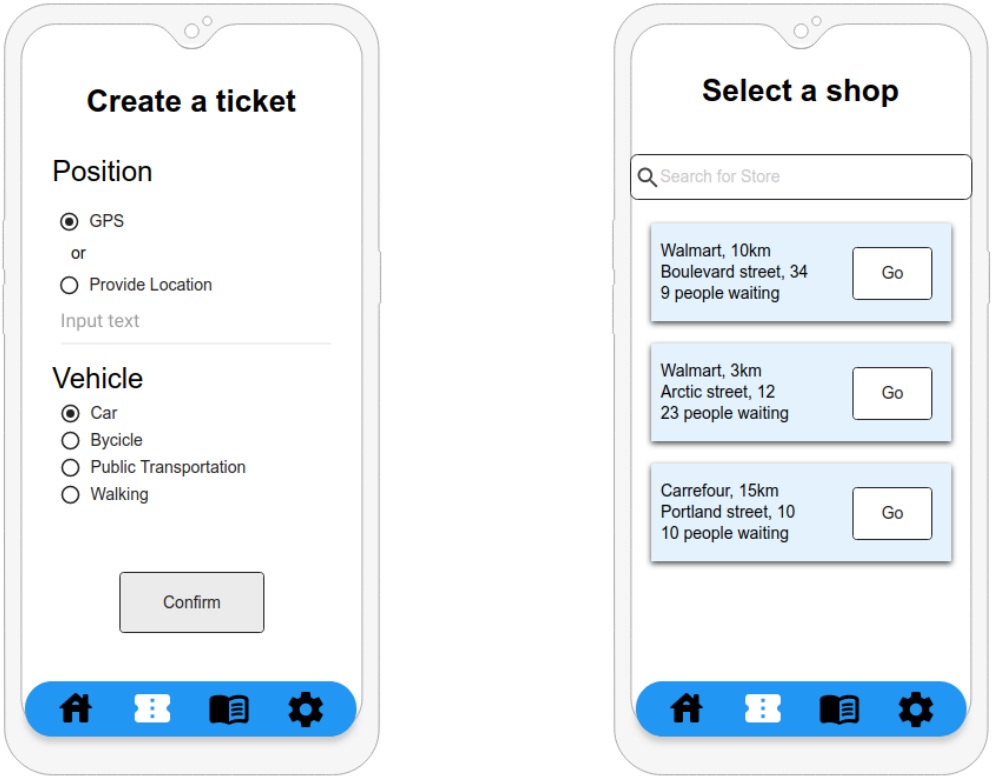
Figure 8 contains screens related to the entrance, namely the QR code (8a) and the suggested route to the store (8b); the alternative suggestions function is performed by these two (that offer the suggestion) and from screens 5c (that alerts of a possible alternative through the blue symbol).

Store Manager's screens are showed in Figure 9.



(a) Registration page (b) Welcome page (c) Welcome page (active elements)

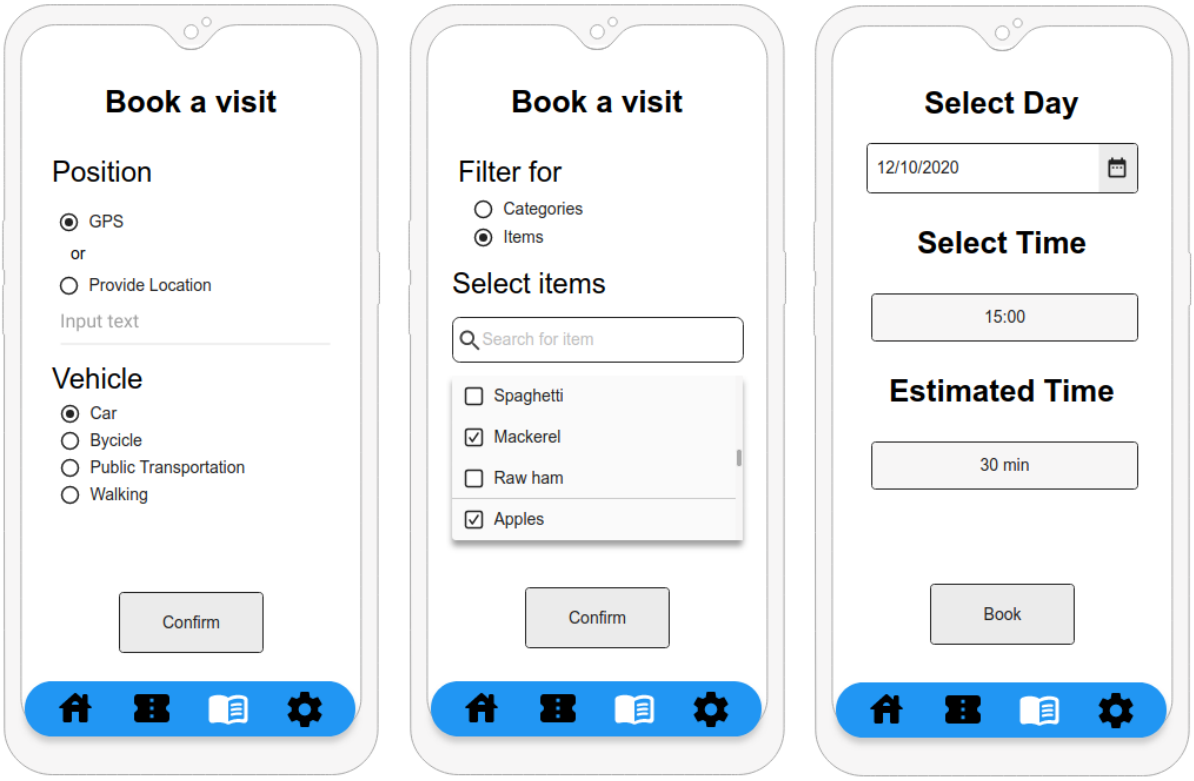
Figure 5: Welcome screens



(a) Ticket creation

(b) Store selection

Figure 6: Ticket creation



(a) Parameters choice

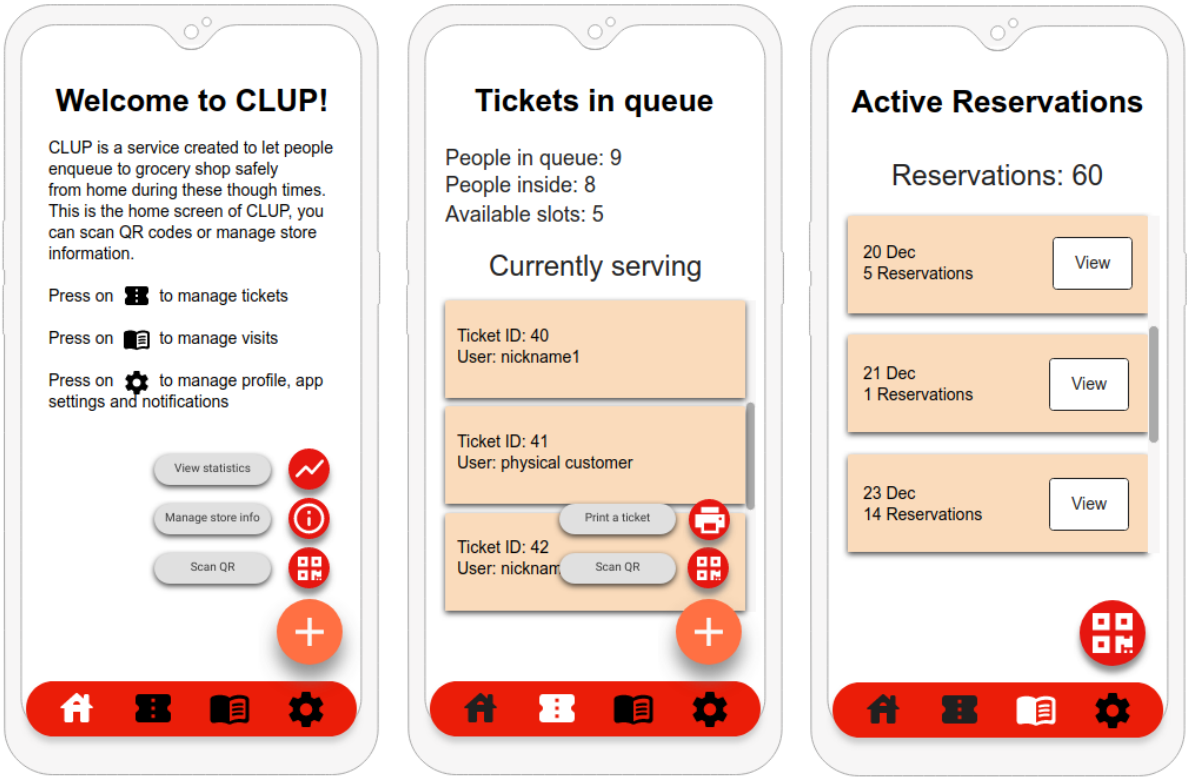
(b) Item selection

(c) Entrance date and time selection

Figure 7: Booking screens

(a) Qr code screen (b) Map showing the recommended route

Figure 8: Access information screens



(a) Welcome screen (b) Tickets (c) Reservations

Figure 9: Store manager's screens

3.1.2 Hardware interfaces

On the e-Customer side, hardware interfaces will be the user's own devices, especially mobile.

For what concerns Store Managers, the system will be interfaced to a device with a camera or to a dedicated QR reader to scan codes, a printer for physical tickets and a screen to display the current number.

3.1.3 Software interfaces

CLup will offer two separate user interfaces based on the user's functions (e-Customer or Store Manager), either through mobile or web-based applications.

The system itself will not offer public interfaces for other applications, but will need to be connected to the web mapping service through its API.

3.1.4 Communication interfaces

As security is a major concern, all client-server communication will use HTTPS, based on the security of TLS 1.3. APIs will be accessed through the REST protocol, encrypted using TLS as well.

3.2 Functional requirements

[R0] Must allow users to provide credentials
[R1] Username must be unique
[R2] Store id must be unique
[R3] Must allow store managers to add or modify their store's information
[R4] Users must be able to login
[R5] Must be able to provide the list of available stores in the user's proximity
[R6] Must know the e-Customer's position
  [R6.1] Must localize the e-Customer or let them provide manually a position
[R7] Must assign a unique and sequential reservation id
[R8] Must generate a QR code to be scanned at entrance and exit
[R9] Must let e-Customers choose their mean of transport
[R10] Must allow the entrance when it is the customer's turn
[R11] Must allow a delay of at most M minutes on a customer's turn
[R12] Must show e-Customers the number of enqueued customers in front of them
[R13] Must show users historical data on given weekdays
[R14] At least P% places must be reserved for tickets
[R15] Must notify e-Customers within a suitable time from entrance
  [R15.1] Notification time must be based on current store occupation
  [R15.2] Notification time must be based on estimated travel time
[R16] Must let e-Customers advance if a preceding customer dequeued
[R17] Must keep track of people inside the stores
  [R17.1] Must allow scanning QR codes on entrance and exit
[R18] Must not allow the entrance of more customers than prescribed
[R20] Physical tickets must be placed on the same queue as virtual ones
[R21] Physical tickets must be printed
[R22] Must allow Physical Customers to scan their code in order to dequeue
[R23] e-Customers who book a visit must be allowed to enter at their chosen time
[R24] Must let e-Customers insert the expected duration of the visit

[R24.1] Must be able to infer and suggest the duration for long term customers

[R25] Must let e-Customers insert a list of items/categories they intend to buy

[R26] Alternatives must be based on information provided by the e-Customer and store occupation (both real time and historical)

[R27] e-Customers must be able to subscribe to the notification service

[R28] Must let e-Customers unsubscribe from the notification service

[R29] The system must notify the e-Customers based on their choices

3.3 Use cases

Figures 10 and 11 show the use case diagrams.

3.3.1 A prospective e-Customer registers

Actors	Prospective e-Customer
Goals	[G0]
Input conditions	The Prospective e-Customer is on the home page and wants to start the registration process
Events flow	1) The Prospective e-Customer clicks on the “Customer Sign Up” button on the home page 2) The Prospective e-Customer inserts their new credentials 3) The Prospective e-Customer clicks on the “Registration” button 4) The system saves the data 5) The system shows a message stating that registration ended successfully
Output conditions	Registration process ended successfully. From now on the new user can use the system as an e-customer
Exceptions	- Credentials inserted by the Prospective e-Customer are not suitable (e.g. invalid email) - Username provided by the Prospective e-Customer is already taken Exceptions are handled by showing an error message to the Prospective e-Customer to inviting them to restart the registration process. Event flow restarts from point 2

3.3.2 A Prospective Store Manager registers a store

Actors	Prospective Store Manager
Goals	[G1]
Input conditions	The Prospective Store Manager wishes to register their store
Events flow	1) The Prospective Store Manager clicks on the “Register a store” button on the home page 2) The Prospective Store Manager inserts their new credentials 3) The Prospective Store Manager inserts their store information 3) The Prospective Store Manager clicks on the “Registration” button 4) The system saves the data 5) The system sends an email about the registration

Output conditions	The registration process ended up successfully. From now on the Store Manager can access their dedicated functions
Exceptions	- Credentials or information inserted by the Prospective Store Manager are not suitable (e.g. invalid store address) - Username provided by the Prospective Store Manager is already taken - The store is already registered Handled by showing an error message to the Prospective e-Customer to inviting them to restart the registration process. Event flow restarts from point 2

3.3.3 An e-Customer lines up

Actors	e-Customer
Goals	[G2]
Input conditions	e-Customer is already logged in and wants to enqueue
Events flow	1) The e-Customer clicks on the "New ticket" button to access the section 2) The e-Customer inserts current location and mean of trasport 3) The app searches for available stores and shows them to the e-Customer 4) The e-Customer chooses a store and presses "generate" 5) The system enqueues the e-Customer for their desired store and generates a QR code for entrance
Output conditions	The process is concluded correctly and the QR code is saved on the device, under tickets
Exceptions	- No stores are available for the defined parameters Handled by showing an error pop-up asking to check the parameters or try again later - Selection becomes unavailable during the selection Handled with an error message

3.3.4 An e-Customer deletes a ticket

Actors	e-Customer
Goals	[G2.1]
Input conditions	The e-Customer is logged, has an active ticket and wishes to dequeue
Events flow	1) The e-Customer accesses the ticket page 2) The e-Customer presses the "Cancel ticket" button and confirms
Output conditions	The system updates the queue and shows a confirmation message
Exceptions	

3.3.5 A store manager modifies the store’s information

Actors	Store Manager

Goals	[G3]
Input conditions	The Store Manager is logged in and wants to update some information
Events flow	1) The Store Manager clicks on the "Modify store" button 2) The Store Manager modifies the information 3) The Store Manager clicks on "Update store"
Output conditions	The system updates the information and presents a message
Exceptions	- The Store Manager leaves an empty field - The Store Manager insert a not well formed field Both handled with an error message

3.3.6 An e-Customer goes to the store

Actors	Store Manager, e-Customer
Goals	[G2] [G3] [G5]
Input conditions	The e-Customer is enqueued and received a notification
Events flow	1) The e-Customer reaches the store 2) The e-Customer waits for his turn outside until his number is reached 3) The e-Customer shows their QR code to get scanned 4) The e-Customer is allowed to enter and shops 5) The system removes the e-Customer from the queue and counts them as inside 6) The e-Customer shows the QR code and is allowed to exit
Output conditions	The system removes the e-Customer from inside the store
Exceptions	- The e-Customer does not enter in their delay window The e-Customer won't be allowed to enter and the queue is shifted - The e-Customer tries to scan the wrong QR code The e-Customer is notified of the wrong code and is not allowed to enter

3.3.7 A Physical Customer lines up

Actors	Physical Customer
Goals	[G4]
Input conditions	A customer not registered to the system wants to enter the shop
Events flow	1) The Physical Customer requests the ticket to enter the queue 2) The ticket is printed and the Physical Customer receives it
Output conditions	The process is concluded correctly and the QR code is printed
Exceptions	- The Physical Customer cannot enqueue because the shop is closing or there are too many customers in the queue The system does not allow the customer to enqueue

3.3.8 A Physical Customer dequeues

Actors	Physical Customer
Goals	[G4.1]
Input conditions	An already enqueued Physical Customer wants to dequeue
Events flow	1) The Physical Customer scans the QR-code at entrance of the shop by stating their intentions 2) The Physical Customer exits the queue
Output conditions	The Physical Customer is successfully dequeued and the system proceeds to clean the ordering of the queue
Exceptions	- The Physical Customer goes away without scanning the QR-code The system manages this exception by keeping the customer in queue until its delay window is expired and then removes it

3.3.9 A Physical Customer shops

Actors	Physical Customer, Store manager
Goals	[G3]
Input conditions	The Physical Customer is at the store
Events flow	1) The Physical Customer waits for their turn outside 3) The Physical Customer shows their QR to get scanned 4) The system removes the Physical Customer from the queue and counts them as inside 5) Physical Customer is allowed to enter and shops 6) Physical Customer shows the QR code and is allowed to exit
Output conditions	The Physical Customer exits and is removed from the store occupation
Exceptions	- The Physical Customer does not enter in their delay window The Physical Customer won' t be allowed to enter - The Physical Customer tries to scan the wrong QR code handled by resuming the flow from point 3)

3.3.10 An e-Customer books a visit

Actors	e-Customer
Goals	[G5]
Input conditions	The e-Customer is logged in and wants to book a visit
Events flow	1) The e-Customer enters in the booking section 2) The e-Customer inserts their mean of transport and their position 3) The e-Customer inserts the store, the date and the hour he wants to book 4) The system registers the reservation
Output conditions	The booking process ended up successfully. From now on the visit is recorded in the database
Exceptions	- No slots are available for the given parameters Handled by asking the eC to insert different parameters

3.3.11 An e-Customer deletes a visit

Actors	e-Customer
Goals	[G5.1]
Input conditions	The e-Customer is logged in and wants to delete a visit
Events flow	1) The e-Customer enters in the booking section 2) The e-Customer clicks on the “active visits” button 3) The system shows the list of active visits 4) The e-Customer selects the visit they want to delete and clicks on the delete button 5) The system deletes the visit from the database
Output conditions	Deletion process ended successfully. From now on the deleted visit is no longer stored in the database
Exceptions	- There are no active visits The system handles this exception by showing an information message and redirects the user to the home page

3.3.12 An e-Customer changes the ticket after an alternative slot is proposed

Actors	e-Customer
Goals	[G2]
Input conditions	The e-Customer is logged in, already got a ticket and the chosen store’s flow is high
Events flow	1) The system computes a series of alternatives based on the e-Customer’s choice 2) The system presents to the e-Customer the alternatives 3) The e-Customer chooses an alternative
Output conditions	The system dequeues the e-Customer from their current queue and adds them to the new one
Exceptions	- No alternatives are available Handled by not showing any alternative

3.3.13 An e-Customer subscribes to periodic notifications

Actors	e-Customer
Goals	[G7]
Input conditions	The e-Customer is logged in
Events flow	1) The e-Customer accesses the set notifications section 2) The e-Customer sets the notification parameters (i.e. time, store) and confirms
Output conditions	The e-Customer is now registered to notifications based on their parameters
Exceptions	- No slots are available for the given parameters Handled by asking the e-Customer to insert different parameters

3.3.14 An e-Customer unsubscribes from slot notifications

Actors	e-Customer
Goals	[G7]
Input conditions	The e-Customer is subscribed to the notification service
Events flow	1) The e-Customer enters in the notification section 2) The e-Customer clicks on the "Active notifications" button 3) The system shows the active notifications 4) The e-Customer selects the service that they want to unsubscribe from 5) The e-Customer clicks on the "Unsubscribe" button
Output conditions	The process ended up successfully and the e-Customer is no longer subscribed to the notification
Exceptions	- The e-Customer is not subscribed to any notification service Handled by showing an error and redirecting the eC to the notification page

3.3.15 An e-Customer gets localized

Actors	e-Customer
Goals	[G2] [G5] [G6] [G7]
Input conditions	The e-Customer is logged in and is compiling a form with a "current position" field
Events flow	1) The e-Customer reaches the position field 2) The system obtains the e-Customer’s position from the "current position" field, if it is empty the web mapping service is used
Output conditions	The system knows the e-Customer’s position
Exceptions	- The position inserted can not be found by the web mapping service Handled by showing an error message and asking to re-insert the position

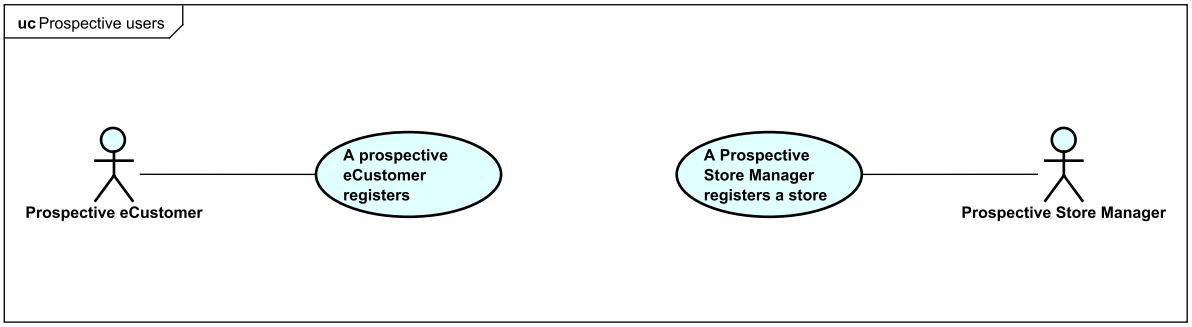


Figure 10: Prospective users use cases

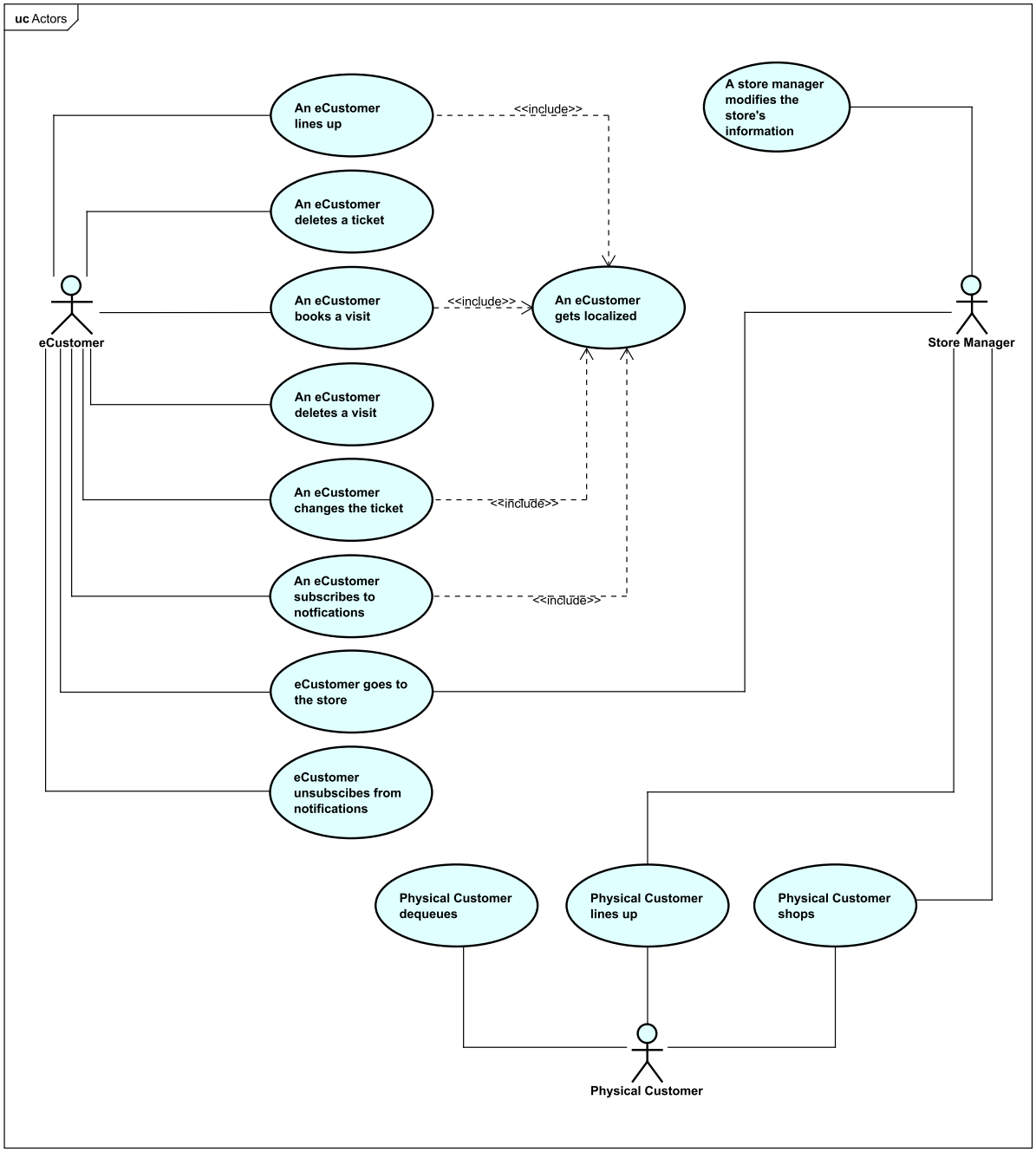


Figure 11: Actors use cases

3.4 Sequence diagrams

Some sequence diagrams:

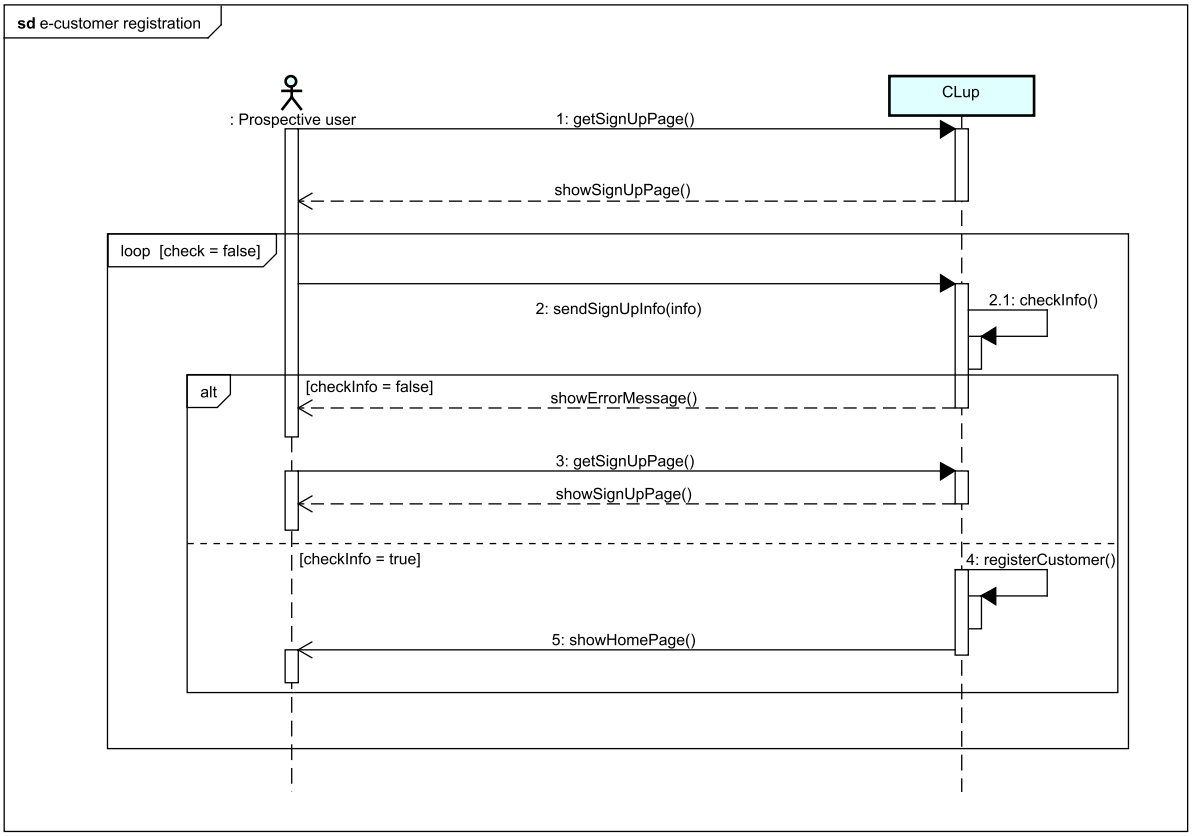


Figure 12: e-Customer registration

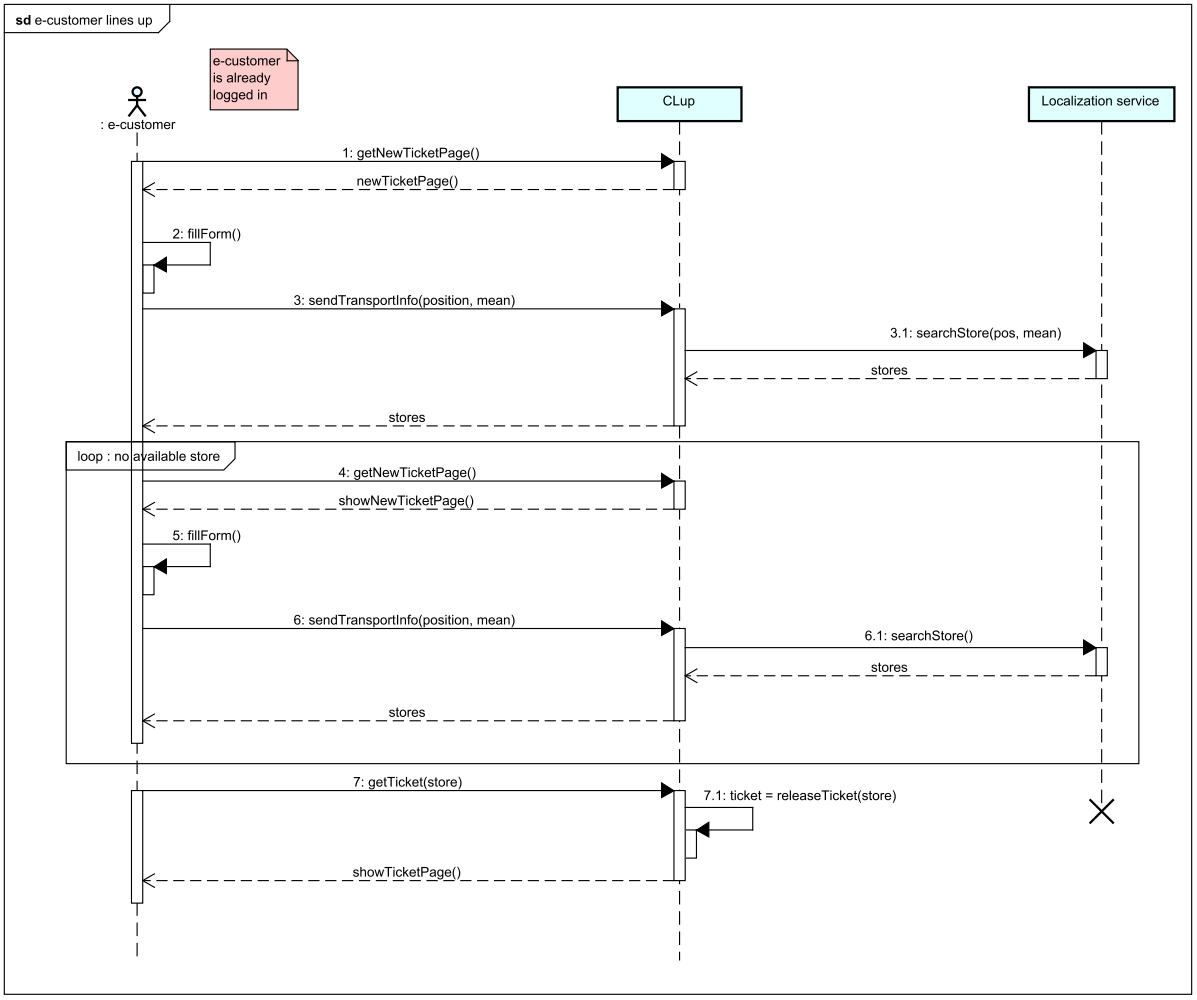


Figure 13: e-Customer lines up

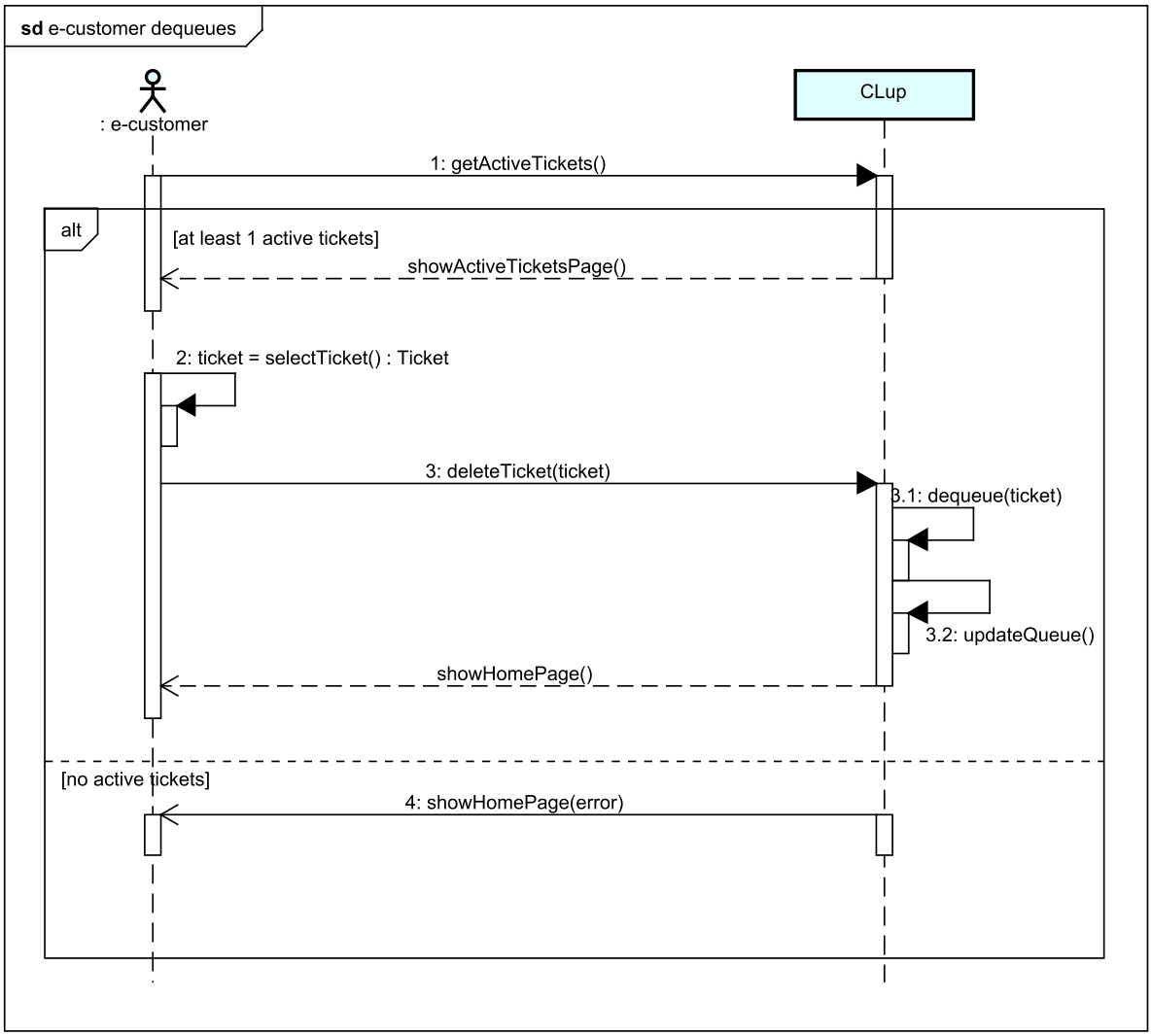


Figure 14: e-Customer dequeues

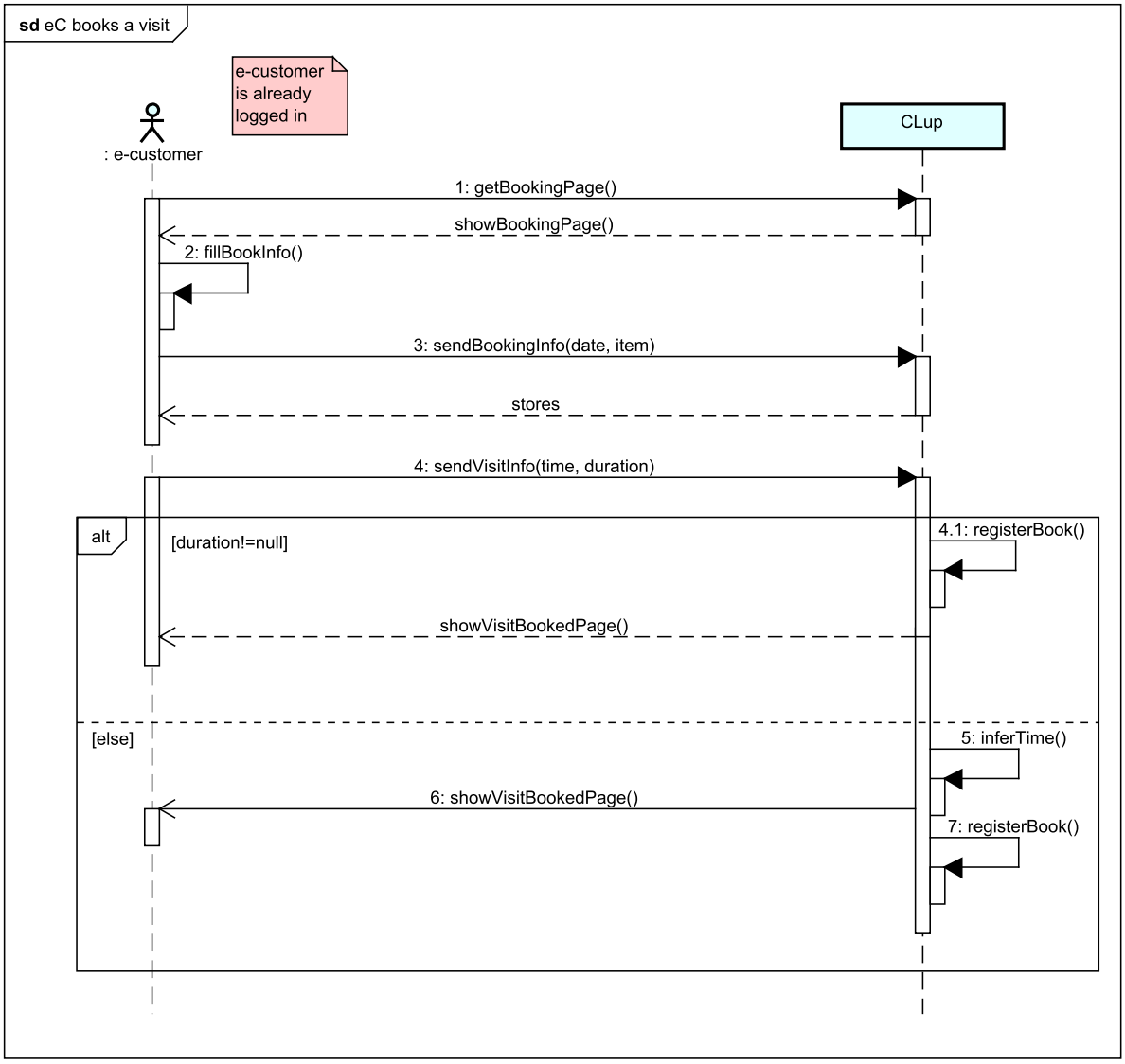


Figure 15: e-Customer books a visit

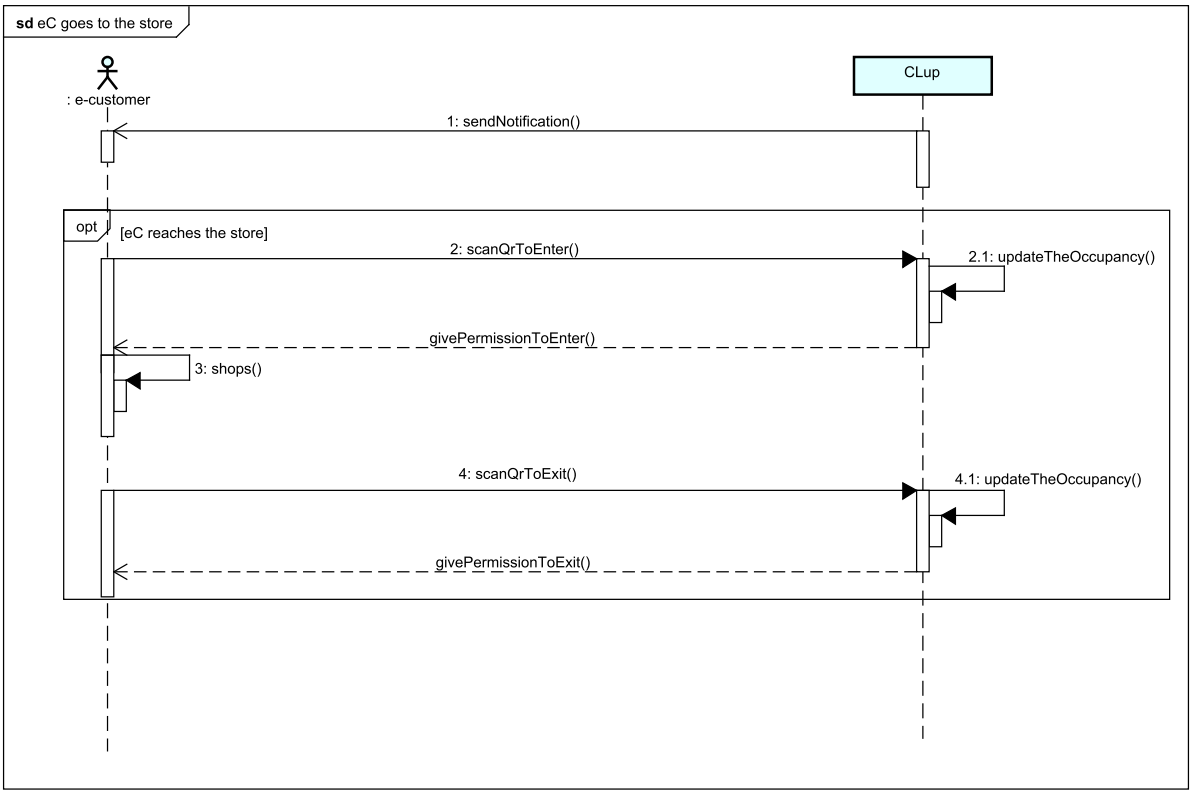


Figure 16: e-Customer goes to the store

3.5 Goal Mapping

In this section, goals are mapped to requirements and domain assumptions.

3.5.1 [G0] Allow prospective e-Customers to register to the application

[R0] Must allow users to provide credentials
[R1] Username must be unique

[D0] Prospective users will complete the registration process
[D3] Information provided by users upon registration is correct
[D11] The internet connection is always working

3.5.2 [G1] Allow Prospective Store Managers to register to the application by adding their store

[R0] Must allow users to provide credentials
[R1] Username must be unique
[R2] Store id must be unique
[R3] Must allow store managers to add or modify their store's information

[D0] Prospective users will complete the registration process
[D3] Information provided by users upon registration is correct
[D7] Information about stores inserted by SMs is correct
[D11] The internet connection is always working

3.5.3 [G2] Allow e-customers to line up from home

[R4] Users must be able to login
[R5] Must be able to provide the list of available stores in the user's proximity
[R6] Must know the e-Customer's position
  [R6.1] Must localize the e-Customer or let them provide manually a position
[R7] Must assign a unique and sequential reservation id
[R8] Must generate a QR code to be scanned at entrance and exit
[R9] Must let e-Customers choose their mean of transport
[R10] Must allow the entrance when it is the customer's turn
[R11] Must allow a delay of at most M minutes on a customer's turn
[R12] Must show e-Customers the number of enqueued customers in front of them
[R13] Must show users historical data on given weekdays
[R14] At least P
[R15] Must notify e-Customers within a suitable time from entrance
  [R15.1] Notification time must be based on current store occupation
  [R15.2] Notification time must be based on estimated travel time
[R16] Must let e-Customers advance if a preceding customer dequeued

[D1] Only people enqueued access the stores (both app and physical)
[D2] Information provided by e-customers corresponds to their will
[D4] Customers enter only if allowed by the application
[D5] Customers respect the choices made during the input phase
[D6] Calculated travel time is correct
[D7] Information about stores inserted by SMs is correct
[D8] Customers entering a store exit after some time
[D9] If a client exits at a given time, he must have entered in the same opening period
[D10] Ticket printers and QR readers work properly
[D11] The internet connection is always working
[D12] Users accept to receive notifications from the application
[D13] Stores show the current customer number
[D14] The external web mapping system is always available and there is signal

3.5.4 [G2.1] Allow e-Customers to dequeue

[R4] Users must be able to login

[D11] The internet connection is always working

3.5.5 [G3] Allow Store Managers to monitor entrances

[R3] Must allow store managers to add or modify their store's information
[R4] Users must be able to login
[R17] Must keep track of people inside the stores
  [R17.1] Must allow scanning QR codes on entrance and exit
[R18] Must not allow the entrance of more customers than prescribed

[D1] Only people enqueued access the stores (both app and physical)

[D4] Customers enter only if allowed by the application
[D5] Customers respect the choices made during the input phase
[D7] Information about stores inserted by SMs is correct
[D8] Customers entering a store exit after some time
[D9] If a client exits at a given time, he must have entered in the same opening period
[D10] Ticket printers and QR readers work properly
[D11] The internet connection is always working
[D13] Stores show the current customer number

3.5.6 [G4] Allow customers to physically line up

[R7] Must assign a unique and sequential reservation id
[R8] Must generate a QR code to be scanned at entrance and exit
[R10] Must allow the entrance when it is the customer's turn
[R11] Must allow a delay of at most M minutes on a customer's turn
[R20] Physical tickets must be placed on the same queue as virtual ones
[R21] Physical tickets must be printed

[D1] Only people enqueued access the stores (both app and physical)
[D4] Customers enter only if allowed by the application
[D8] Customers entering a store exit after some time
[D9] If a client exits at a given time, he must have entered in the same opening period
[D10] Ticket printers and QR readers work properly
[D11] The internet connection is always working
[D13] Stores show the current customer number

3.5.7 [G4.1] Allow Physical Customers to dequeue

[R11] Must allow a delay of at most M minutes on a customer's turn
[R22] Must allow Physical Customers to scan their code in order to dequeue

[D10] Ticket printers and QR readers work properly
[D11] The internet connection is always working
[D13] Stores show the current customer number

3.5.8 [G5] Allow e-Customers to book a visit

[R4] Users must be able to login
[R5] Must be able to provide the list of available stores in the user's proximity
[R6] Must know the e-Customer's position
  [R6.1] Must localize the e-Customer or let them provide manually a position
[R7] Must assign a unique and sequential reservation id
[R8] Must generate a QR code to be scanned at entrance and exit
[R9] Must let e-Customers choose their mean of transport
[R10] Must allow the entrance when it is the customer's turn
[R11] Must allow a delay of at most M minutes on a customer's turn
[R13] Must show users historical data on given weekdays
[R14] At least P% places must be reserved for tickets

[R15] Must notify e-Customers within a suitable time from entrance

[R15.2] Notification time must be based on estimated travel time

[R23] e-Customers who book a visit must be allowed to enter at their chosen time

[R24] Must let e-Customers insert the expected duration of the visit

[R24.1] Must be able to infer and suggest the duration for long term customers

[R25] Must let e-Customers insert a list of items/categories they intend to buy

[D1] Only people enqueued access the stores (both app and physical)

[D2] Information provided by e-customers corresponds to their will

[D4] Customers enter only if allowed by the application

[D5] Customers respect the choices made during the input phase

[D6] Calculated travel time is correct

[D7] Information about stores inserted by SMs is correct

[D8] Customers entering a store exit after some time

[D10] Ticket printers and QR readers work properly

[D11] The internet connection is always working

[D12] Users accept to receive notifications from the application

[D14] The external web mapping system is always available and there is signal

3.5.9 [G5.1] Allow e-Customers to delete a visit

[R4] Users must be able to login

[D11] The internet connection is always working

3.5.10 [G6] Can suggest alternative slots to balance the number of customers

[R4] Users must be able to login

[R6] Must know the e-Customer's position

[R6.1] Must localize the e-Customer or let them provide manually a position

[R17] Must keep track of people inside the stores

[R26] Alternatives must be based on information provided by the e-Customer and store occupation (both real time and historical)

[D4] Customers enter only if allowed by the application

[D5] Customers respect the choices made during the input phase

[D6] Calculated travel time is correct

[D7] Information about stores inserted by SMs is correct

[D8] Customers entering a store exit after some time

[D9] If a client exits at a given time, he must have entered in the same opening period

[D11] The internet connection is always working

[D14] The external web mapping system is always available and there is signal

3.5.11 [G7] Allow e-Customers to manage slot notifications

[R4] Users must be able to login
[R6] Must know the e-Customer's position
[R6.1] Must localize the e-Customer or let them provide manually a position
[R17] Must keep track of people inside the stores
[R27] e-Customers must be able to subscribe to the notification service
[R28] Must let e-Customers unsubscribe from the notification service
[R29] The system must notify the e-Customers based on their choices

[D2] Information provided by e-customers corresponds to their will
[D7] Information about stores inserted by SMs is correct
[D8] Customers entering a store exit after some time
[D9] If a client exits at a given time, he must have entered in the same opening period
[D11] The internet connection is always working
[D12] Users accept to receive notifications from the application
[D14] The external web mapping system is always available and there is signal

3.6 Non-functional requirements

An important non-functional requirement is the ease of use, given the wide variety of people potentially using the system. Requirements related to the system are as follows.

3.6.1 Performance

CLup, both for the app and the web application, should provide fast responses to user actions, possibly under 3 seconds[2]. This result should account for about 150000 daily requests, based on competitor's data[5].

3.6.2 Availability and reliability

As a service used for primary needs connected to stores potentially open all day and every period of the year, CLup should be very robust and reliable, offering an availability of at least 0.999 (that corresponds to less than a day of downtime every year).

3.6.3 Security

All user information will need to be securely stored by the application. The communication between clients and server should be protected alongside QR codes, in order to ensure fairness in queue management too.

3.6.4 Maintainability

As in all software applications, maintainability should be a core property of CLup, enabling changes both related to modifications in functionalities and bug fixing.

3.6.5 Portability

The client system should be developed to be compatible with most of the mobile Operating Systems (for the mobile application) and browsers (for the web app). The latter option should prioritize mobile environments (i.e. smartphones, tablets) without precluding access to desktop-based systems, even if they are not the primary target of the application.

3.7 Design constraints

3.7.1 Hardware and software limitations

The hardware and software requirements are stated here, divided by usage:

- For the mobile application:
  - Operating system: Android 6.0+ or iOS 9+
  - The device must be internet enabled
- For the web application, a device with either:
  - Mozilla Firefox 19.0+
  - Google Chrome 25.0+
  - Microsoft Edge
  - Safari 5.1+
  - Opera 12.1+

A mobile device used by a Store Manager will also need a camera.

3.7.2 Privacy policies

The system will need to access some device data in order to function properly; the following table summarizes the permission requests:

Permission	e-Customer		Store Manager	
	Mobile	Web	Mobile	Web
Storage	Mandatory			
Camera			Mandatory	Mandatory
Localization	Optional	Optional		

Moreover, in order to comply with privacy regulations[3], CLup:

- Will ask for the user's consent for their data processing
- Will not use any of the user data for purposes different from the offered service
- Will not request personal data that is not necessary to offer the service
- Will not keep any personal data once it is not needed anymore
- Will ensure privacy and security by design

4 Formal Analysis Using Alloy

This section describes the model defined through the Alloy language[1]. It is used to define the subset of the domain used to handle tickets, that represents the most critical portion of the system.

As it can be noted in Section 4.5, the chosen bitwidth is 7, thus allowing the generation of integers between -64 and $+63$ to meet all constraints.

4.1 Signatures

```
// Ticket states
enum State {
    ENQUEUED,
    CANENTER,
    DEQUEUED,
    INSIDE,
    COMPLETED,
    WRONG
}

// All clients of stores
sig Customer {
    tickets: set Ticket,
    visits: set Visit,
    insideStore: lone Store
}

// Store signature
sig Store {
    capacity: one Int,
    delayWindow: one Int,
    queue: one Queue,
    inside: set Customer,
    hours: set OpeningHours
} {
    capacity > 0
    delayWindow > 0
    #inside =< capacity
}

// Signature containing sequences of tickets
sig Queue {
    store: one Store,
    tickets: seq Ticket
}

// Ticket signature
sig Ticket {
    nOrder: one Int,
    customer: one Customer,
    queue: one Queue,
    state: one State
} {
    nOrder >= 0
}

```

```
47      // Signature representing a visit that occurred
48      sig Visit {
49          customer: one Customer,
50          accessreq: one Ticket,
51          store: one Store,
52          weekday: one Int,
53          datetimeIN: one DateTime,
54          datetimeOUT: one DateTime
55      } {
56          weekday >= 1 and weekday =< 7
57          accessreq.queue.store = store
58          accessreq.customer = customer
59          isBefore[datetimeIN.time, datetimeOUT.time]
60          datetimeIN.time != datetimeOUT.time
61      }
62
63      // Stores' opening hours
64      // In order to simplify the model,
65      // no store is open during day changes
66      sig OpeningHours {
67          weekday: one Int,
68          openingTime: one Time,
69          closingTime: one Time
70      } {
71          weekday > 0 and weekday =< 7
72          isBefore[openingTime, closingTime]
73          and (openingTime.hours != closingTime.hours
74              or openingTime.minutes != closingTime.minutes)
75      }
76
77      // Signature used to model time
78      sig Time {
79          hours: one Int,
80          minutes: one Int
81      } {
82          hours >= 0
83          hours < 24
84          minutes >= 0
85          minutes < 60
86      }
87
88      // Signature to model a date
89      // (to simplify the year is assumed as 2020)
90      sig Date {
91          month: one Int,
92          day: one Int
93      } {
94          month >= 1 and month =< 12
95          day >= 1
96          (month = 4 or month = 6 or month = 9 or month = 11)
97              implies day =< 30
98          else (month = 2)
99              implies day <= 29
100         else day =< 31
101     }
102
103     // Union of date and time
```

```
104     sig DateTime {
105         date: one Date,
106         time: one Time
107     }
```

4.2 Facts

```
1     // Fact to define ticket unicity in queues
2     fact uniqueTicketInQueue {
3         // No different tickets having the same number
4         no disj t, t': Ticket | all q: Queue |
5             t in q.tickets.elems
6             and t' in q.tickets.elems
7             and t.nOrder = t'.nOrder
8
9         // No ticket is in a queue more than once
10        all q: Queue | !q.tickets.hasDups
11    }
12
13    // Facts modeling necessary relations between signatures
14    fact implications {
15        // Customer <=> Ticket
16        all c: Customer | all t: Ticket |
17            t in c.tickets iff t.customer = c
18
19        // Customer <=> store
20        all c: Customer | all s: Store |
21            s = c.insideStore iff c in s.inside
22
23        // Queue <=> Store
24        all q: Queue | all s: Store |
25            q.store = s iff s.queue = q
26
27        // Queue => Ticket
28        all t: Ticket | all q: Queue |
29            t in q.tickets.elems implies t.queue = q
30
31        // Visit <=> Customer
32        all v: Visit | all c: Customer |
33            v.customer = c iff v in c.visits
34    }
35
36    // A ticket can correspond to at most one visit
37    fact oneTicketPerVisit {
38        no disj v,v': Visit |
39            v.accessreq = v'.accessreq
40    }
41
42    // A customer cannot request a ticket for a store
43    // they are already enqueued for
44    fact noSameCustomerInQueue {
45        all q: Queue | no disj t, t': Ticket |
46            t in q.tickets.elems
47            and t' in q.tickets.elems
48            and t.customer = t'.customer
49    }
```

```
50
51      // A customer inside a store must have entered with a ticket
52      fact customerInsideRequiresTicket {
53          all c: Customer | all s: Store | c.insideStore = s implies
54              one t: Ticket | t.customer = c and t.state = INSIDE
55      }
56
57      // Ticket Ordering
58      fact TicketOrder {
59          all t: Ticket |
60              t.nOrder = plus[t.queue.tickets.idxOf[t], 1]
61      }
62
63      // Ticket states
64      fact ticketStates {
65          all t: Ticket |
66          (!ticketInside[t]
67          and !ticketInVisit[t]
68          and !ticketInItsQueue[t])
69              implies t.state = DEQUEUED
70          else ticketInside[t]
71              implies t.state = INSIDE
72          else ticketInVisit[t]
73              implies t.state = COMPLETED
74          else ticketInItsQueue[t] and !canEnter[t]
75              implies t.state = ENQUEUED
76          else ticketInItsQueue[t] and canEnter[t]
77              implies t.state = CANENTER
78          else t.state = WRONG
79      }
80
81      // A customer cannot have a visit for a ticket
82      // if they are still inside
83      fact noVisitWhileInside {
84          no v: Visit |
85              v.customer in v.store.inside
86      }
87
88      // A customer cannot be contemporarily inside a store
89      // and enqueued for it
90      fact noQueuedWhileInside {
91          no t: Ticket |
92              t in t.queue.tickets.elems
93              and t.customer in t.queue.store.inside
94      }
95
96      // A customer cannot have a visit and be in a queue
97      // with the same ticket
98      fact noEnqueudAndVisit {
99          no v: Visit |
100             v.accessreq in v.store.queue.tickets.elems
101     }
102
103     // A visit can only occur during one opening period of a store
104     fact visitDuringOpening {
105         all v: Visit |
106         some h: OpeningHours |
```

```
            h in v.store.hours
            and h.weekday = v.weekday
            and isBefore[h.openingTime, v.datetimeIN.time]
            and isBefore[v.datetimeOUT.time, h.closingTime]
    }
```

Particular care has been used in defining the fact `ticketStates`; their meaning derives from four possible attributes of a ticket: it can be in the queue it's referring (Q), it can be in a visit (V), and it can correspond to a customer inside a store (I). Based on this, the following truth table has been realized to define the ticket states:

Q	V	I	State
0	0	0	DEQUEUED
0	0	1	INSIDE
0	1	0	COMPLETED
0	1	1	X
1	0	0	CANENTER/ENQUEUED
1	0	1	X
1	1	0	X
1	1	1	X

Where the `CANENTER` and `ENQUEUED` states are distinguished through the `canEnter` predicate. Every state indicated by an `X` corresponds to an inconsistent state that must not happen.

4.3 Predicates and functions

```
// Predicate to check if a time is before another one
pred isBefore[t, t': Time] {
    t.hours < t'.hours or
    ((t.hours = t'.hours) and t.minutes =< t'.minutes)
}

// Checks if a ticket's queue points to it
pred ticketInItsQueue[t: Ticket] {
    t in t.queue.tickets.elems
}

// Checks if a visit containing the ticket exists
pred ticketInVisit[t: Ticket] {
    some v: Visit | v.accessreq = t
}

// Checks if the ticket's customer is inside the ticket's store
pred ticketInside[t: Ticket] {
    t.customer in t.queue.store.inside
}

// Checks if a ticket is one of the first in its queue
pred canEnter[t: Ticket] {
    t in t.queue.tickets.subseq[0, minus[
        minus[t.queue.store.capacity, #t.queue.store.inside]
    , 1]].elems
}

// Predicate used to show a world
```

```
    // with exactly one ticket for each state
    pred showOneTicketPerState {
        // Imposing some opening time and visit time
        // to better visualize the world
        all h: OpeningHours |
            minus[h.closingTime.hours, h.openingTime.hours] > 3
        all v: Visit |
            minus[v.datetimeOUT.time.hours, v.datetimeIN.time.hours] < 3

        // Limiting 1 ticket for each customer
        // for visualization purposes
        all c: Customer |
            #c.tickets = 1

        one t: Ticket | t.state = ENQUEUED
        one t: Ticket | t.state = DEQUEUED
        one t: Ticket | t.state = COMPLETED
        one t: Ticket | t.state = INSIDE
        one t: Ticket | t.state = CANENTER
    }
```

4.4 Assertions

```
    // Assertion to check that no entrance is allowed
    // at maximum capacity
    assert noEnterIfFull {
        all s: Store | all t: Ticket |
        (#s.inside = s.capacity and t.queue = s.queue)
            implies !canEnter[t]
    }

    // This assert checks that all tickets that are generated
    // correspond to a correct state
    assert noWrongState {
        no t:Ticket | t.state = WRONG
    }
```

4.5 Alloy analysis

As a proof of consistency, the following commands were run, thus obtaining the results in Figure 17.

```
run canEnter for 5 but 7 Int
check noEnterIfFull for 5 but 7 Int
check noWrongState for 5 but 7 Int
run showOneTicketPerState for 5 but 7 Int, exactly 1 Store
```

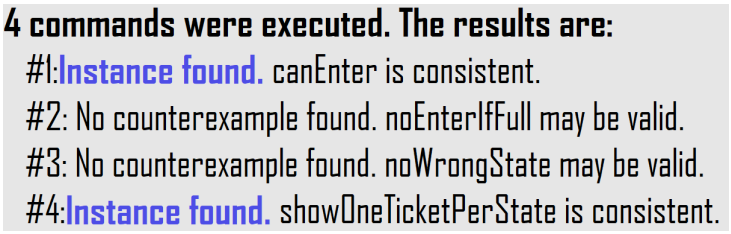


Figure 17: Alloy analysis results

The generated world can be seen in Figure 18, while Figure 19 offers an example where exists a ticket for each consistent state (Non-relevant signatures are hidden in Figure 20).

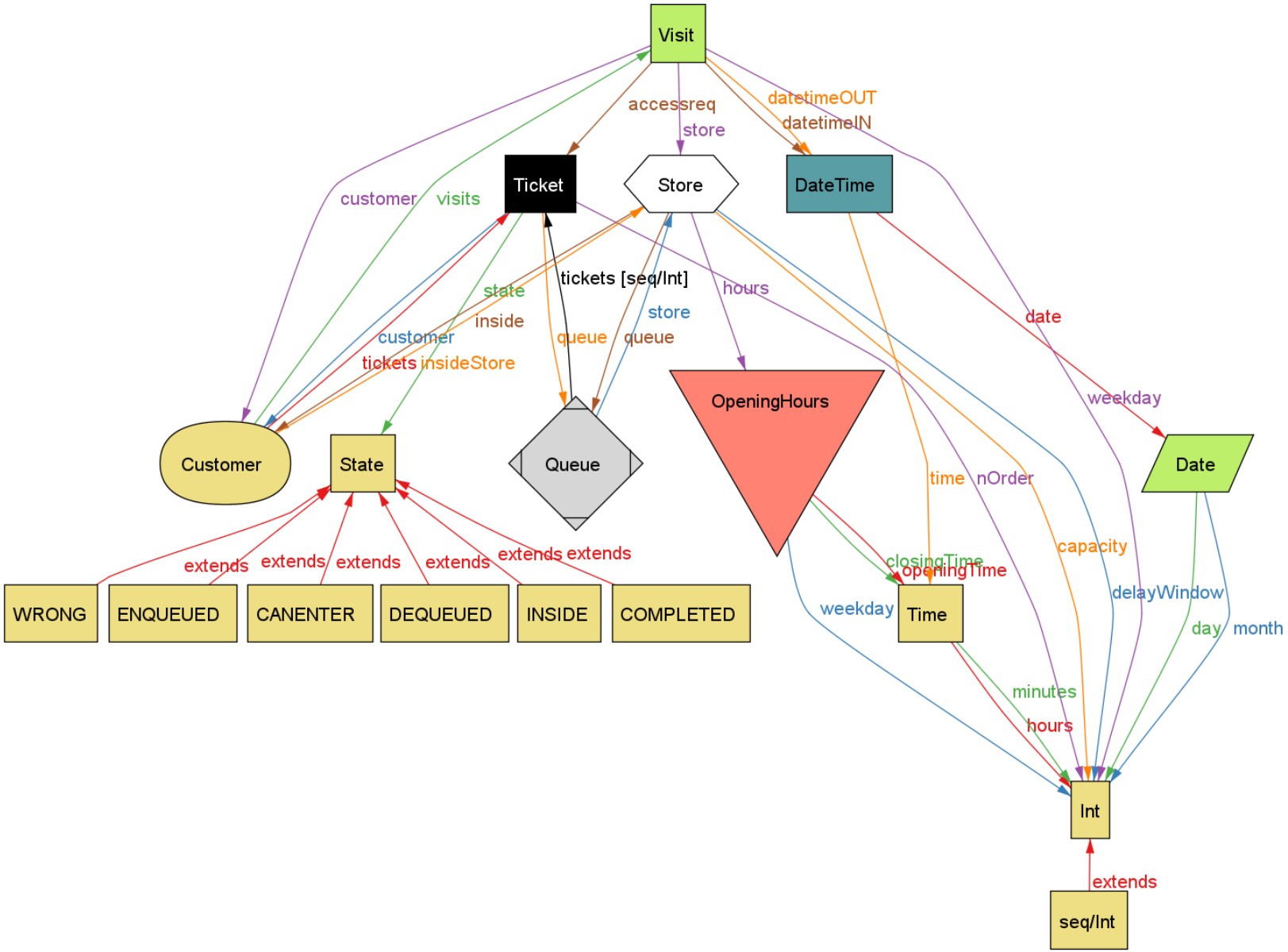


Figure 18: Alloy metamodel

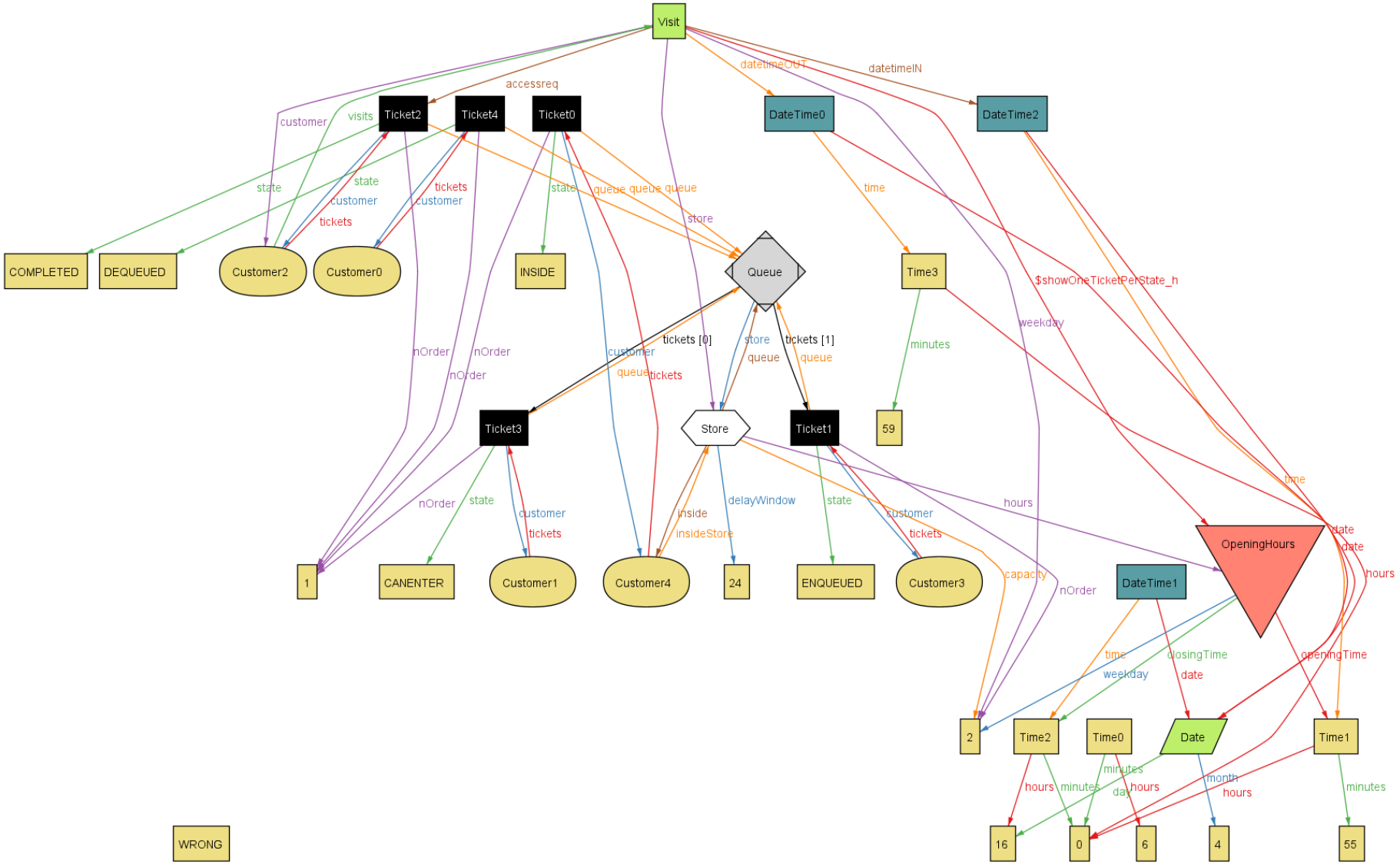


Figure 19: Result of running `showOneTicketPerState`

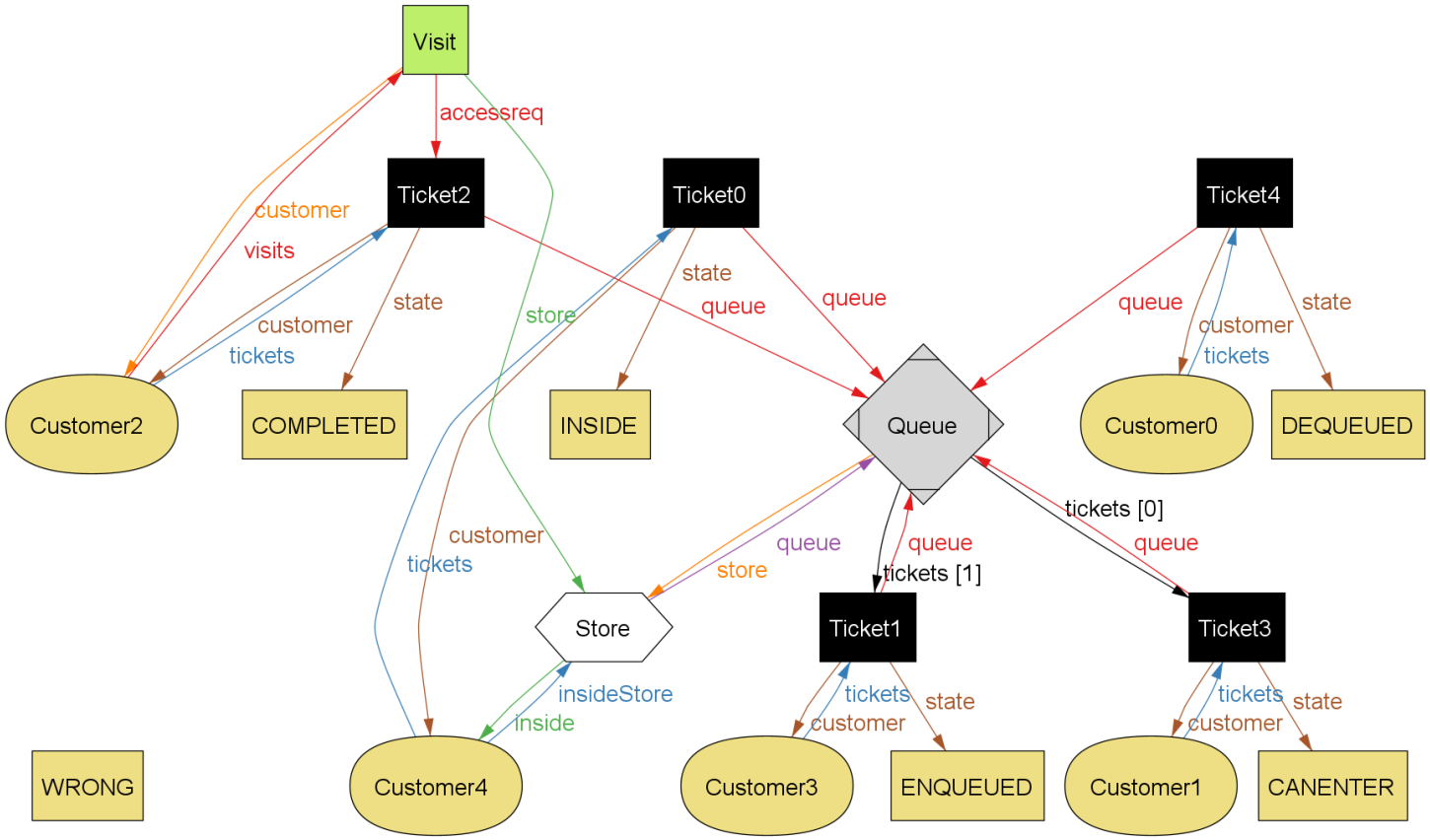


Figure 20: Result of running `showOneTicketPerState` (Only relevant signatures)

5 Effort and tools

5.1 Effort spent breakdown

				TOTAL
Previous RASD analysis	6	6	6	6
World & machine	2	2	2	2
Goal analysis	5.5	5.5	5.5	5.5
Requirement analysis	8.5	8.5	8.5	8.5
Domain analysis	3.5	3.5	3.5	3.5
Document writing	9	3	3	12
UI mockup	0.5	0.5	7	7
Class diagram	6.5	6.5	6.5	6.5
Use cases	10	8.5	9	13.5
Scenarios	1	1.5	1	2.5
Sequence diagrams	-	2.5	-	2.5
Alloy	16	16	16	22
TOTAL	68.5	64	68	91.5

5.2 Software tools

This is the list of tools used during the development of this document:

- **TeXstudio**: Text editor
- **MikTeX**: LaTeX compiler
- **Astah UML**: UML modeling (class diagram, sequnce diagrams, state diagrams)
- **Moqups**: User interface mock-ups
- **Alloy modeling tools**: Alloy analyzer

References

[1] Alloy documentation. URL: http://alloytools.org/documentation.html.

[2] Daniel An. Find out how you stack up to new industry benchmarks for mobile page speed. Technical report, Google, 2017. URL: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/.

[3] European Commission. General Data Protection Regulation (GDPR), 2018. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679.

[4] Michael A. Jackson. The world and the machine, 1995. URL: http://mcs.open.ac.uk/mj665/icse17kn.pdf.

[5] Biagio Simonetta. Ufirst, storia l'app anti-code nata dalla segreteria studenti di Roma Tre. *Il Sole 24 Ore*, 2020. URL (IN ITALIAN): https://www.ilsole24ore.com/art/ufirst-storia-l-app-anti-code-nata-segreteria-studenti-roma-tre-ADEGEGV.