POLITECNICO
MILANO 1863

CLup - Customer Line Up

Software Engineering 2 Project 2020/21

Authors

- Francesco Attorre - *10618456*
- Thomas Jean Bernard Bonenfant - *10597564*
- Veronica Cardigliano - *10627267*

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Francesco Attorre, Thomas Jean Bernard Bonenfant, Veronica Cardigliano
Version:	2.0
Date:	07-February-2021
Download page:	https://github.com/FrancescoAttorre/softeng2-attorre-bonenfant-cardigliano
Copyright:	Copyright © 2020, Francesco Attorre, Thomas Jean Bernard Bonenfant, Veronica Cardigliano - All rights reserved

Contents

1 Introduction

A Purpose

The goal of this document (RASD: Requirements Analysis and Specification Document) is to describe a model of the proposed problem capturing the main aspects of it, analyzing its requirements, goals, actors, domain assumptions, scenarios and relative use cases. This document is addressed to the developers of the software to be, to give a simplified view of it. This specific problem has the goal of developing an easy application to allow store managers to regulate the influx of people in the building and to prevent the customers from having to line up outside stores for a long time.

B Scope

B.1 Description of the given problem

In a period of global pandemic, a lot of usually normal actions have become challenges, first of all the common entry into a public place like a supermarket. In this context, CLup digitalizes ticket numbers distribution giving customers the opportunity to queue and to plan store visits from home. This allows people to save time and be safer, and store managers to better control and limit the accesses to the building. This will be obtained scanning the QR codes generated by the app for each incoming customer, that will allow also to take track of who entered the store, when he entered and of other people present at that time. About shared phenomena, customers will have the opportunity to register to the app in order to have customized services, such as average time spent in the shop, and the possibility to book a visit in a specific time slot. When it's time for them to leave for the building, they'll be noticed based on how far they are, to avoid that they leave too early or late creating again the unwanted queues in front of the building. Planning store visits, customers will have the possibility to indicate the expected permanence time and the departments from which they intend to buy, in order to understand where they're gonna spend their time and to be able to better distribute them.

B.2 Current system

So far, people who want to access a building have to take a ticket on site, this means a lot of time spent waiting in queue and risking being infected. CLup was designed to avoid this taking advantage of technological tools. However, as well as booking from home, customers unable to use the app can also take a ticket on the spot, which will be registered in the app by the store manager. In this way the previous technology integrates with the new one avoiding to exclude anyone from the possibility of entering the building.

B.3 Goals

[G 1]. Allow an Activity to register to CLup.

[G 2]. Allow an Activity to insert one or more Buildings.

[G 3]. Allow Store Managers to sign up for a Building.

[G 4]. Allow an AppCustomer to register to CLup.

- [G 4.1]. Allow a RegisteredAppCustomer to login.

[G 5]. Allow an AppCustomer to choose a Building to visit.

- [G 5.1]. Allow a RegisteredAppCustomer to login.
- [G 5.2]. Allow a RegisteredAppCustomer to choose a Building to book for.

[G 5.3]. Allow AppCustomers to view Buildings at a certain distance from their position.

[G 5.4]. Allow AppCustomers to view estimated TravelTime to reach a Building.

[G 6]. Allow AppCustomers to know when they should leave for the Building.

[G 7]. Allow a Customer to acquire a Ticket.

[G 7.1]. Allow an AppCustomer to line up to access a Building.

[G 7.2]. Allow a RegisteredAppCustomer to book a visit to a Building.

[G 7.3]. Allow a PhysicalCustomer to acquire a PhysicalTicket.

[G 8]. Allow Customers to enter the Building when it's their turn.

[G 8.1]. Allow StoreManagers to monitor entrances scanning a QR code for each AppCustomer willing to enter the Building.

[G 8.2]. Allow StoreManagers to monitor entrances of PhysicalCustomers when it's their turn to enter the Building.

[G 9]. Allow a StoreManager to enter LineUpDigitalTickets.

[G 10]. Allow each AppCustomer to see a list of acquired DigitalTickets.

[G 11]. Allow StoreManagers to inform CLup when a Customer leaves a Building.

[G 11.1]. Allow StoreManagers to read an exit QR code.

C Definitions, Acronyms, Abbreviations

C.1 Definitions

- Digital Ticket: digital counterpart of a usual ticket number. It contains the date (including time) of its reservation, the identifier of the Building the customer wishes to visit and the number to add in the priority queue.
- LineUpDigitalTicket: a DigitalTicket that also contains the number added in the priority queue. If it is requested by a StoreManager it will be associated to the number of the PhysicalTicket given to a PhysicalCustomer
- BookingDigitalTicket: a DigitalTicket that contains also the date and time of arrival and estimated leaving and optionally the Departments of the Building that the RegisteredAppCustomer chose.
- Building: physical place managed by an activity registered to the system where customers can enter.
- PhysicalTicket: ticket provided by the StoreManager to a PhysicalCustomer. When collected the number written on it is associated with a new Digital Ticket by the System.
- EstimatedPermanenceTime: for a Booking is the difference between selected Exit Time and selected Arrival Time.
- Waiting Time: estimated time that the Customer must wait in order to enter the Building.
- Arrival Time: when the Customer reaches the Building
- Exit Time: when the Customer exits the Building
- ValidDigitalTicket: DigitalTicket state that grants its owner access to the Building.

- NotValidDigitalTicket: DigitalTicket state with which DigitalTicket's owner cannot enter the Building yet.
- ExpiredDigitalTicket: DigitalTicket that doesn't grant anymore access to the Building.
- BuildingAccessCode: Building's secret code which allows Store Managers to sign up for the associated Building.
- TravelTime: Time needed to reach the Building according to the selected means of transport. Therefore a means of transport must be selected in order to compute it.
- Department: a Building is composed of one or more zones called Departments.
- BuildingCapacity: number of customers generally allowed in the store. In case of more than one Department, this number takes into account the smallest department, to ensure that crowds are avoided.
- BookingCapacity: 40% of BuildingCapacity
- DepartmentSurplus: extra Customers allowed in a specific Department, when it is larger than the base one.
- TimeSlot: time interval multiple of 15 minutes with a starting time. They include all the opening time of the store and each RegisteredAppCustomer can book one of them

C.2 Acronyms

- RASD: Requirements Analysis and Specification Document
- QR code: Quick Response code, a 2D barcode typically used to store information intended to be read using a smartphone.
- GPS: Global Positioning System
- CLup: Customers Line-up
- DD: Design Document

C.3 Abbreviations

- Gn: goal number n
- Rn: requirement number n
- Dn: domain number n

D Revision History

- First version 23-December-2020 : After a first base version, more specific requirements have been inserted to solve various ambiguities that have arisen and a dynamic behavior has been added to the Alloy model.
- Second version 07-February-2021: Errors in enumerating requirements have been fixed.

E Reference Documents

- Specification Document: “R&DD Assignment AY 2020-2021.pdf”
- Slides of the lectures
- Alloy Doc: https://alloytools.org/documentation.html, https://alloydocs-fork.readthedocs.io/

F Document Structure

This RASD in composed by 6 main sections:

- SECTION 1 is the introduction, which explains the purpose of the system, the problem it aims to solve and the context in which it’ll be developed. In this section are shown also the main goals, revision history, references and definitions, acronyms and abbreviations, useful to understand the other sections.
- SECTION 2 is the overall description of the system. It starts with a list of scenarios to communicate the key aspects of how the system’ll work, followed by class diagrams and state charts showing the main features of the project’s structure. In "Product Functions" there’s a list of the functional requirements for each goal that the system’ll have to fulfill and a list of the different types of actors involved. At the end of this section, there are the domain assumptions, assertions on the external world that we assume to hold.
- SECTION 3 contains the external interface requirements with user, hardware, software and communication interfaces. Here, functional requirements are better explained through use case and activity diagrams and a list of use cases with the relative sequence diagrams. This section includes also the mapping table of requirements, and the explanation of non-functional requirements such as performance, reliability, availability, security, maintainability and portability that the software to be should respect, together with the main design constraints.
- SECTION 4 concerns the alloy part, with alloy code and and some graphs showing the models generated from it. Through assertions can be seen what we proved about the model.
- SECTION 5 contains a table with the effort spent by each member of the group.
- SECTION 6 contains the tools used to develop the document.

2 Overall Description

A Product Perspective

A.1 Scenarios

Scenario 1 - Line up for a building

John needs to go to the nearest grocery store. So he opens the CLup application, chooses to go on foot, but he notices that the nearest grocery store has an estimated waiting time of 30 min plus it takes him 5 min to get there. Then he notices that a store a little further has an estimated waiting time of 12 min plus it takes him 10 min to get there, so he requests a digital ticket to line up and leaves home. When he gets to the store he waits in line until it's his turn, opens the application and presents the QR Code associated with his digital ticket to the Store Manager. The Store manager scans the QR Code from his application and receives a "Ticket accepted" notification, so he lets the Customer in. After an hour of shopping, John leaves the grocery store.

Scenario 2 - Book a visit to a building

Bob plans to go to the grocery store the following day, but he can only go at 4 pm as he needs to be back to work at 5 pm. Because of this, he opens the CLup app, registers for the first time and logs in with his credentials (username and password). Bob sets the car as mean of transport and chooses the first Store of the list that the app shows him, which is also the nearest, booking a visit from 4 pm to 5 pm, with an estimated time of one hour and without choosing specific categories of items. The following day he reaches the store in time presenting the QR Code associated with his booking. The store manager scans his QR Code, receives a "Ticket Accepted" notification and lets Bob in. Bob'll show again a QR code to certify the end of the shopping at exactly 4:49 pm before leaving the building.

Scenario 3 - Physically line up to enter a building

Elder Jim wants to go to the grocery store. Unfortunately he doesn't own a smartphone and therefore he does not have the CLup application. At the Store's entrance the Store Manager asks him for a QR Code, but Old Jim, not understanding, complains about the technology. The Store Manager then gives him a PhysicalTicket with the number 57, inserts this number into his CLup Application and tells Jim to wait until his number is called.
15 minutes later the Store Manager receives a notification that informs him to let number 57 enter the building. So Store Manager calls number 57 and Jim approaches and enters the Store.

Scenario 4 - small mistake, wrong building

Jane wants to do shopping before her smart working begins, at 9 o' clock, then opens CLup as soon as she wakes up. She sets a maximum distance of 2 km and "by car" as means of transport, and chooses the first shop in the list. She's the only one in line, so CLup notifies her immediately to leave for the building. Jane, then, leaves home quickly, but she is still a little asleep, so she reaches the wrong Building. Also such Building has just opened, so also there it's up to number 1. The Store Manager scans her QRcode and receives a "Ticket denied" Notification, so he prevents Jane from entering. At this point she realizes the mistake and goes to the right shop, but it is quite far, so when she arrives her ticket has already expired.

A.2 Class Diagrams

Actors - Class Diagram

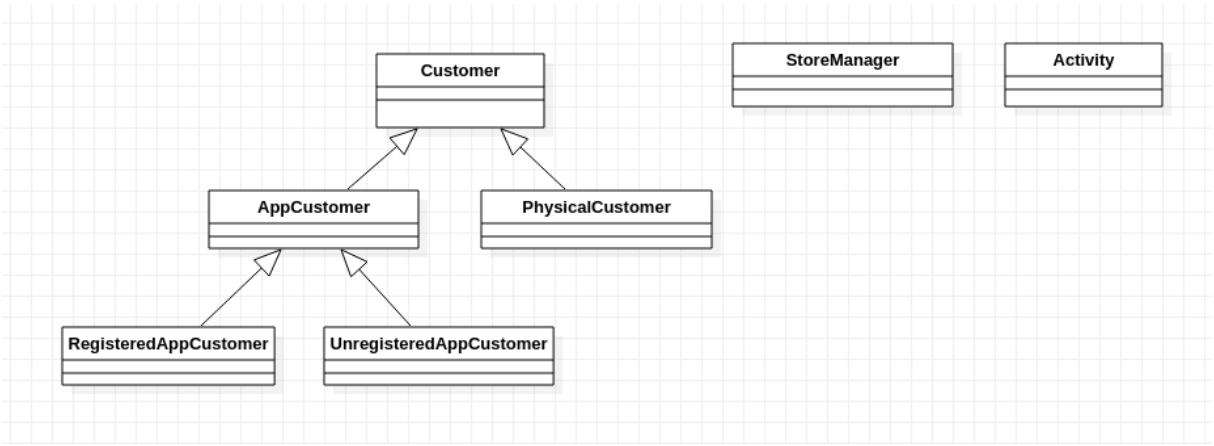

Figure 1: Class Diagram of Actors

State Charts

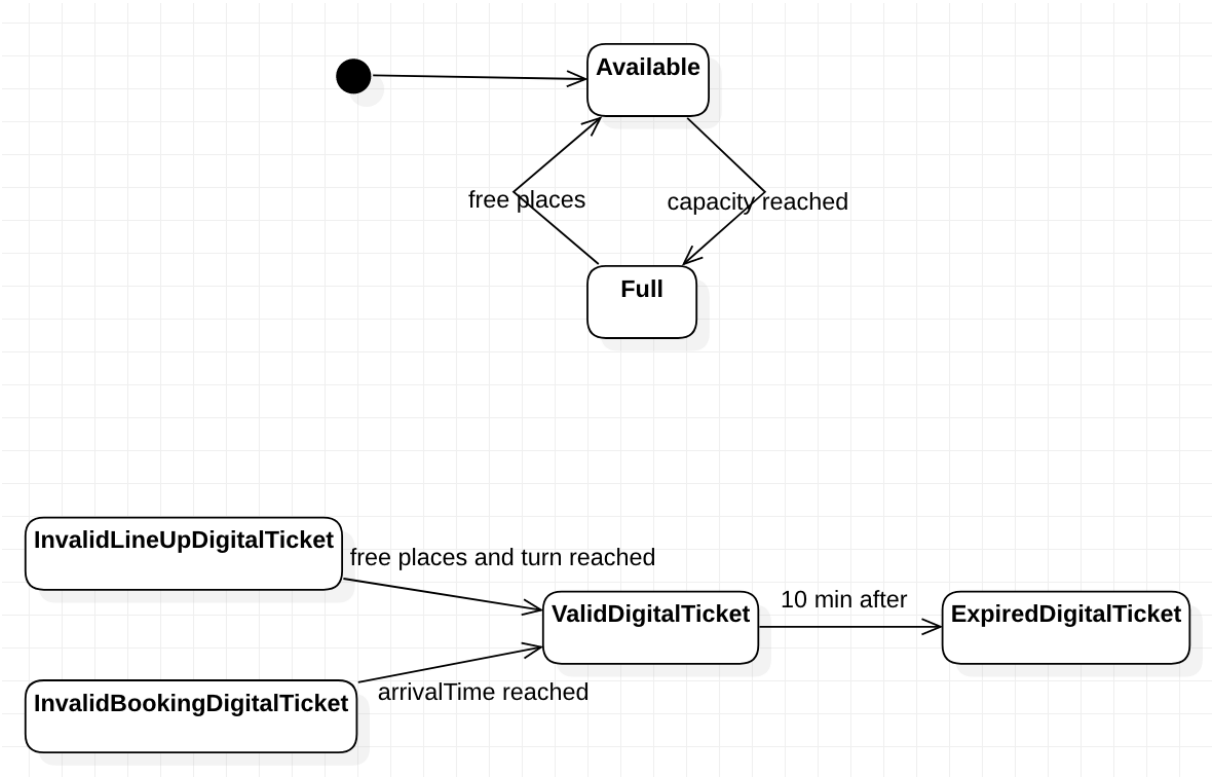

Figure 2: State Diagrams of Buidings (above) and Tickets (below)

Overall Class Diagram

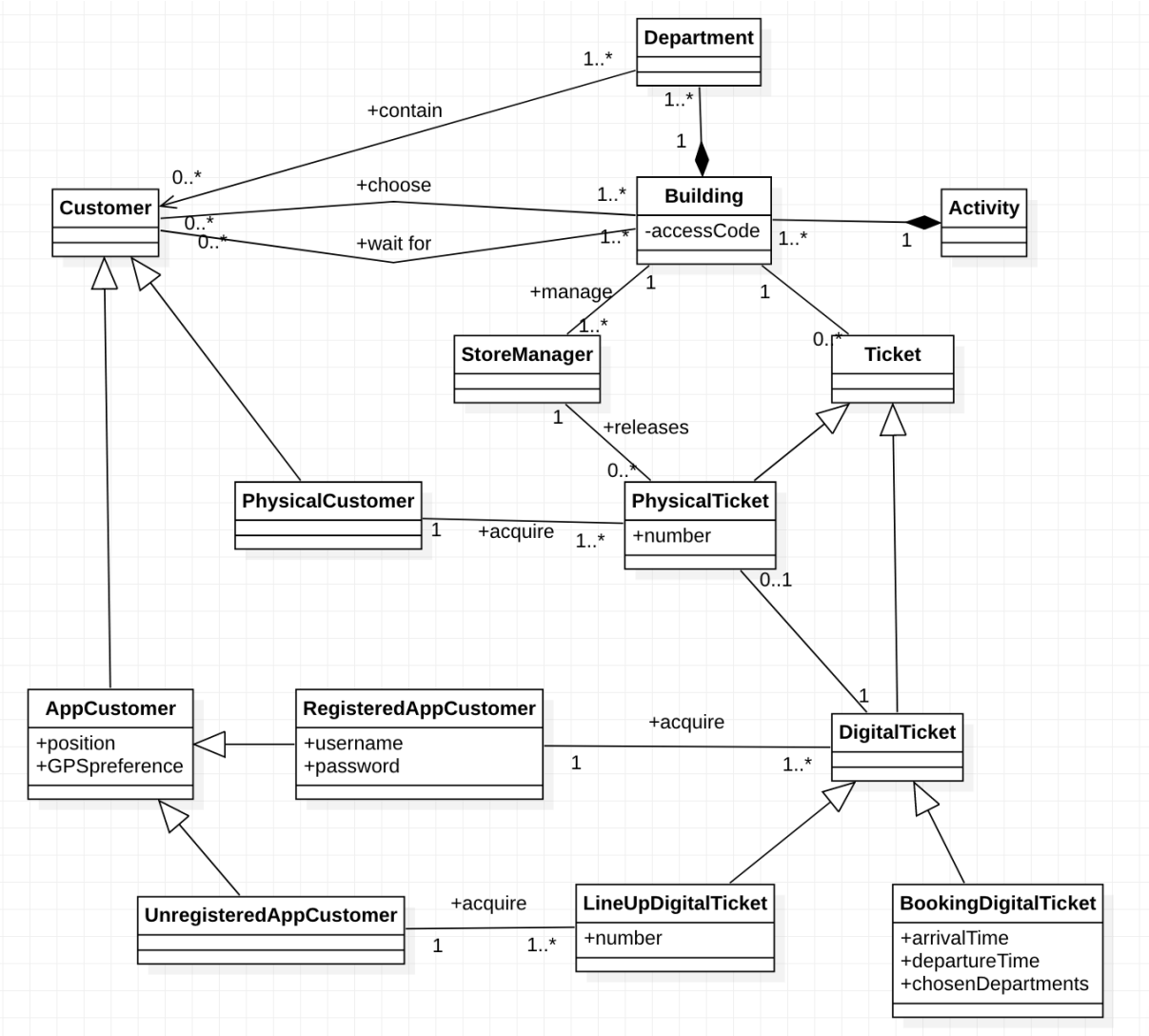

Figure 3: Class Diagram of the system

B Product Functions

In this section we include the most important Requirements with which each Goal, that represents different world phenomena that CLup aims to manage, can be satisfied. The requirements represent shared phenomena, which could be controlled by the world and observed by the machine or vice versa, but in each case they involve both the system, CLup, and the external world, in our case Customers, Activities and Store Managers. We have given prominence to this section in order to clarify any ambiguity of the assignment and make future developments of CLup as simple as possible.

B.1 Requirements

[G 1]. Allow an Activity to register to CLup.
[R1] CLup allows an Activity to start the registration process.
[R2] CLup allows an Activity to register only after providing a username, a password and a P.Iva.
[D1] P.Iva inserted is effectively associated with a real existing subject who carries out an Activity.
[D2] Each Activity name will be unique.

[G 1.1]. Allow an Activity to log in.
[R3] Clup must guarantee each Activity the possibility to login only after providing a username and a password.
[R4] CLup allows to login only if the password he provides is the one associated with his account.

[G 2]. Allow an Activity to insert one or more Buildings.
[R5] CLup allows an already logged in Activity to start a Building insertion.
[R6] CLup allows each Activity to insert one or more Buildings only after providing for each of them a name, an address, a BuildingCapacity and opening and closing time.
[R7] CLup will ensure that Buildings have unique addresses.
[R8] CLup will ask each Activity to add one or more Departments per Building, each of them with a DepartmentSurplus.
[R9] When the Activity adds its Buildings, CLup gives a unique identifier code to each building (the BuildingAccessCode).
[D3] Each Activity is supposed to enter reasonable numbers of BuildingCapacity and DepartmentSurplus, i.e. those defined by law according to the size of the place.
[D4] The BuildingAccessCode is known only by the Activity and is shared only with the Store Managers elected for that Building.
[D5] The position of the Building inserted by the Activity will be correct.
[D6] Each Building is an existent real store.
[D7] Each Building has a BuildingCapacity bigger than zero.

[G 3]. Allow Store Managers to sign up for a Building.
[R10] CLup must give to a StoreManager the possibility to sign up for a Building, entering a BuildingAccessCode.
[R11] CLup allows a StoreManager to sign up for up to one Building.
[R9] When the Activity adds its Buildings, CLup gives a unique identifier code to each building (the BuildingAccessCode).
[D8] A Building will have at least one Store Manager that signs up for it.

[G 4]. Allow an AppCustomer to register to CLup.

[R12] CLup must give to each AppCustomer the possibility to become a RegisteredAppCustomer.
[R13] CLup allows AppCustomer to start a registration process.
[R14] CLup allows AppCustomer to become registered only after providing a username and a password.
[R15] CLup must accept only unique usernames.
[R16] CLup should never allow other AppCustomers to see other RegisteredAppCustomer information.

[G 4.1]. Allow a RegisteredAppCustomer to login.

[R17] CLup must guarantee each RegisteredAppCustomer the possibility to login only after entering a username and password.
[R4] CLup allows to login only if the password he provides is the one associated with his account.

[G 5]. Allow an AppCustomer to choose a Building to visit.

[R18] CLup will provide a list of Buildings, sorting them by the distance from the Customer.
[R19] CLup won't show Buildings further than 10km.
[R20] CLup can access the GPS position of each Customer who gave it the permission.
[R21] CLup has to force the Customer to insert the localization manually if he doesn't allow the system to take it by GPS.
[R22] CLup makes the AppCustomer select the means of transport he will use to reach the Building.
[D9] The GPS giving the position of the customers works properly.

[G 5.1]. Allow an AppCustomer to choose a Building to line up for.

[R23] The AppCustomer should be able to see how many people are currently in the queue for a specific Building.
[R24] CLup does not allow customers to select a Building if its estimated WaitingTime goes beyond closure time.
[R25] CLup must give to an AppCustomer the possibility to choose a Building to line up.

[G 5.2]. Allow a RegisteredAppCustomer to choose a Building to book for.

[R26] CLup must give to an RegisteredAppCustomer the possibility to choose a Building to book for.

[G 5.3]. Allow AppCustomers to view Buildings at a certain distance from their position.

[R27] CLup will allow all AppCustomers to select the maximum distance of the buildings to visualize, filtering them.

[G 5.4]. Allow AppCustomers to view estimated TravelTime to reach a Building.

[R28] CLup has to provide an estimated TravelTime to reach the selected Building according to the chosen means of transport.

[G 6]. Allow AppCustomers to know when they should leave for the Building.

[R22] CLup makes the AppCustomer select the means of transport he will use to reach the Building.
[R29] CLup has to provide an estimated WaitingTime to access the selected Building.
[R30] CLup notifies RegisteredAppCustomers that booked a visit to leave for a Building,

when the time left to reach the TimeSlot's starting time is equal to their TravelTime.
[R31] CLup notifies AppCustomers in queue to leave for a Building, when the difference between estimated WaitingTime and TravelTime becomes less than or equal to zero.
[R32] CLup notifies the AppCustomer to leave for a Building only one time.

[G 7]. Allow a Customer to acquire a Ticket.
[R16] CLup should never allow other AppCustomers to see specific AppCustomer information.

[G 7.1]. Allow an AppCustomer to line up to access a Building.
[R33] Acquiring a new LineUpDigitalTicket, an AppCustomer becomes the last one in the queue.
[R34] CLup must give to an AppCustomer the possibility to acquire a LineUpDigitalTicket.

[G 7.2]. Allow a RegisteredAppCustomer to book a visit to a Building.
[R35] CLup allows RegisteredAppCustomer to acquire a BookingDigitalTicket.
[R36] While booking a visit, CLup allows RegisteredAppCustomer to choose a date.
[R37] While booking a visit, CLup allows a RegisteredAppCustomer to choose the EstimatedPermanenceTime.
[R38] While booking a visit,Clup allows* a RegisteredAppCustomer to insert Departments he wants to visit.
[R39] After acquiring a BookingDigitalTicket, if RegisteredAppCustomer has selected some Departments, he's going to occupy one of the DepartmentSurplus places if still available, otherwise he will occupy one of the BookingCapacity places of the selected Building.
[R40] CLup allows RegisteredAppCustomer to select a TimeSlot.
[R41] CLup won't let RegisteredAppCustomers to choose a TimeSlot if date and EstimatedPermanenceTime haven't been entered.
[R42] CLup won't let RegisteredAppCustomers to acquire a DigitalTicket if a TimeSlot has not been selected.
[R43] CLup has to suggest alternatives if there aren't available TimeSlots.
[R44] The 3 alternative slots suggested by CLup have to be those with fewer bookings in order to balance the influx to the Building.
[R45] After choosing a Building, CLup will provide to the RegisteredAppCustomer a set of TimeSlots for that Building.
[R46] CLup provides to the RegisteredAppCustomer only TimeSlots during which the number of RegisteredAppCustomers who have booked is less than the BookingCapacity for all their durations.
[R47] CLup will allow RegisterAppCustomers to choose TimeSlots multiples of 15 minutes.
[R48] CLup will allow RegisteredAppCustomers to choose from a set of TimeSlots for the chosen date, of the same duration as the EstimatedPermanenceTime.
[R49] CLup provides to the RegisteredAppCustomer only TimeSlots during which no DepartmentSurplus of the chosen Departments has been reached yet.
[R50] CLup will ensure that TimeSlots must be during the Building's Opening hours.
[R51] A RegisteredAppCustomer can book only one visit per Building.
[R52] RegisteredAppCustomer's bookings for different Buildings must not overlap.
[R53] If a RegisteredAppCustomer has already visited a Building with a booking, and shown his QRCode to exit, CLup will suggest as EstimatedPermanenceTime the average of his previous visits' durations.

[D10] RegisteredAppCustomers will visit only the indicated Departments, when entered.

[G 7.3]. Allow a PhysicalCustomer to acquire a PhysicalTicket.
[D11] Each StoreManager is able to provide as many PhysicalTickets as required by PhysicalCustomers.
[D12] StoreManagers will give exactly one PhysicalTicket to each Physical Customer.

[G 8]. Allow Customers to enter the Building when it's their turn.

[G 8.1]. Allow StoreManagers to monitor entrances scanning a QR code for each AppCustomer willing to enter the Building.
[R54] CLup allows AppCustomers to see the QRCode for each acquired DigitalTicket (up to one day after it became an ExpiredDigitalTicket).
[R55] CLup encodes in the QR Code sent to the AppCustomer the information composing the DigitalTicket he has requested (date of reservation, Building identifier and DigitalTicket identifier).
[R56] The StoreManager always receives a Notification from CLup after scanning a QR code.
[R57] CLup must return a NegativeNotification to the StoreManager if the AppCustomer that's trying to enter has a NotValidDigitalTicket.
[R58] CLup must return a NegativeNotification to the StoreManager if the AppCustomer that's trying to enter has an ExpiredDigitalTicket.
[R59] CLup must return a PositiveNotification to the StoreManager if the AppCustomer that's trying to enter has a ValidDigitalTicket.
[R60] When WaitingTime is bigger than or equal to TravelTime, CLup will convert a LineUpDigitalTicket into a ValidDigitalTicket when the estimated WaitingTime becomes zero.
[R61] When WaitingTime is lower than TravelTime at booking, CLup will convert a LineUpDigitalTicket into a ValidDigitalTicket after TravelTime has passed by.
[R62] CLup will convert a BookingDigitalTicket into a ValidDigitalTicket when the current time is equal to the starting time of its associated TimeSlot.
[R63] CLup will convert a DigitalTicket into an ExpiredDigitalTicket 10minutes after it becomes a ValidDigitalTicket.
[D13] Once a QR code has been scanned with a PositiveNotification, the related Customer will actually enter the Building.
[D14] If a StoreManager receives a NegativeNotification when scanning a QR Code he won't allow the AppCustomer owning the QR Code to enter.
[D15] If a Store Manager receives a PositiveNotification when scanning a QR Code he will allow the AppCustomer owning that QR Code to enter.
[D16] Store Managers will always prevent Customers that do not present a Ticket from entering the building.
[D17] The StoreManager allows only one person per Ticket to enter in a Building after presenting a ValidDigitalTicket.

[G 8.2]. Allow StoreManagers to monitor entrances of PhysicalCustomers when it's their turn to enter the Building.
[R16] CLup should never allow other Customers to see a specific RegisteredAppCustomer information.

[R68] CLup must notify a StoreManager when a LineUpDigitalTicket, which he inserted, becomes valid.
[D18] The StoreManager informs PhysicalCustomers in the queue on the spot when it's their turn.

[G 9]. Allow a StoreManager to enter LineUpDigitalTickets.
[R65] A Store Manager should be able to enter LineUpDigitalTickets in CLup only for the Building he is signed in.
[R66] LineUpDigitalTickets entered by a StoreManager will be related to the same Building for which he signed up.
[R67] CLup allows a Store Manager to insert a LineUpDigitalTicket providing the number of a PhysicalTicket.
[D19] While entering a LineUpDigitalTicket, StoreManager will provide to CLup the right number of the PhysicalTicket he is giving to the PhysicalCustomer.
[D11] Each StoreManager is able to provide as many PhysicalTickets as required by PhysicalCustomers.
[D20] StoreManagers will give exactly one PhysicalTicket to each Physical Customer.

[G 10]. Allow each AppCustomer to see a list of acquired DigitalTickets.
[R68] Allow each AppCustomer to see in every moment after taking a LineUpDigitalTicket, the number of Customers with higher priority in the queue.
[R69] CLup allows AppCustomers to see how many Customers are currently in queue for a specific Building.
[R70] CLup allows RegisteredAppCustomers to see in every moment the list of all the acquired BookingDigitalTickets with associated Building name, Arrival Time and Exit Time.
[R54] CLup allows AppCustomers to see the QRCode for each acquired DigitalTicket (up to one day after it became an ExpiredDigitalTicket).
[R71] CLup allows AppCustomers to see in every moment the list of all the acquired LineUpDigitalTickets with Building name and number of Customers with higher priority in the queue.

[G 11]. Allow StoreManagers to inform CLup when a Customer leaves a Building.
[R72] Allow StoreManager to report to CLup that a Customer left a Building when he doesn't show a QR Code.
[D21] Every time that a Customer leaves a Building without showing a QR code, the StoreManager notifies CLup.

[G 11.1]. Allow StoreManagers to read an exit QRCode.
[R57] CLup allows AppCustomers to see the QRCode for each acquired DigitalTicket (up to one day after it became an ExpiredDigitalTicket).
[R73] Allow StoreManager to scan a QR Code from a RegisteredAppCustomer who has booked and wants to leave the Building.
[D22] Each RegisteredAppCustomer who has booked will show his QR Code to exit the Building.

C User Characteristics

C.1 Actors

In this document we refer to Actor with masculine pronouns, but obviously they include both males and females, so he can be interpreted as he/she.

- Store Manager: clerk of the Building, the one who can scan QR codes and manage the acquisition of DigitalTickets on behalf of PhysicalCustomers. He is authorized to use CLup through his BuildingAccessCode.
- Customer: people who're going to enter in the Building, which can be Physical, Registered or Unregistered.
- PhysicalCustomer: customers who want to enter the building without using CLup directly.
- UnregisteredAppCustomer: customers who use CLup without signing up.
- RegisteredAppCustomer: customers who use CLup through an account.
- Activity: companies that own buildings which can be visited using CLup.

D Assumptions, Dependencies and Constraints

D.1 Domain Assumpions

[D 1]. P.iva inserted is effectively associated with a real existing subject who carries out an Activity.

[D 2]. Each Activity name will be unique.

[D 3]. Each Activity is supposed to enter reasonable numbers of BuildingCapacity and DepartmentSurplus, i.e. those defined by law according to the size of the place.

[D 4]. The BuildingAccessCode is known only by the Activity and is shared only with the Store Managers elected for that Building.

[D 5]. The position of the Building inserted by the Activity will be correct.

[D 6]. Each Building is an existent real store.

[D 7]. Each Building has a BuildingCapacity bigger than zero.

[D 8]. A Building will have at least one Store Manager that signs up for it.

[D 9]. The GPS giving the position of the customers works properly.

[D 10]. RegisteredAppCustomers will visit only the indicated Departments, when entered.

[D 11]. Each StoreManager is able to provide as many PhysicalTickets as required by PhysicalCustomers.

[D 12]. StoreManagers will give exactly one PhysicalTicket to each Physical Customer.

[D 13]. Once a QR code has been scanned with a PositiveNotification, the related Customer will actually enter the Building.

[D 14]. If a StoreManager receives a NegativeNotification when scanning a QR Code he won't allow the AppCustomer owning the QR Code to enter.

[D 15]. If a Store Manager receives a PositiveNotification when scanning a QR Code he will allow the AppCustomer owning that QR Code to enter.

[D 16]. Store Managers will always prevent Customers that do not present a Ticket from entering the building.

[D 17]. The StoreManager allows only one person per Ticket to enter in a Building after presenting a ValidDigitalTicket.

[D 18]. The StoreManager informs PhysicalCustomers in the queue on the spot when it's their turn.

[D 19]. While entering a LineUpDigitalTicket, StoreManager will provide to CLup the right number of the PhysicalTicket he is giving to the PhysicalCustomer.

[D 20]. StoreManagers will give exactly one PhysicalTicket to each Physical Customer.

[D 21]. Every time that a Customer leaves a Building without showing a QR code, the StoreManager notifies CLup.

[D 22]. Each RegisteredAppCustomer who has booked will show his QR Code to exit the Building.

3 Specific Requirements

A External Interface Requirements

A.1 User Interfaces

The device used by the Actors must be a mobile device, so that they'd be able to bring it with them. Detailed illustrations of all the various user interfaces will be provided in the DD (Design document).

A.2 Hardware Interfaces

For Store Managers, the device used has to have a camera in order to scan QR codes and make CLup acquire information about Customers entering/leaving a Building.
The system has to be developed as an application which requires:

- iOS or Android smartphone
- 2G/3G/4G connection
- GPS (optional)

A.3 Software Interfaces

The system uses GoogleMaps's public API to provide the customers the estimated travel time to reach each building, with the chosen means of transport.

A.4 Communication Interfaces

The system has to acquire registration information, such as username and password for the AppCustomers, username and Partita Iva for the Activities and username and BuildingAccessCode for Store Managers. The system must also save the position set by each AppCustomer and, in case of RegisteredAppCustomers, a list of their visit's duration in each building. The latter will be shown to the Customers suggesting them as default duration the average visit time. CLup has also to save buildings information, acquired during buildings registration and to be shown during AppCustomers building choice. Communication between customers and the system occurs via internet connection. To properly use the mobile app, a stable connection is relevant, especially for AppCustomers in order to always receive updates about the number of people with higher priority in the queue. Also Store Managers have to be always connected to the system to notify of customers leaving and to warn physical customers that their turn has come.

B Functional Requirements

B.1 Use case diagram

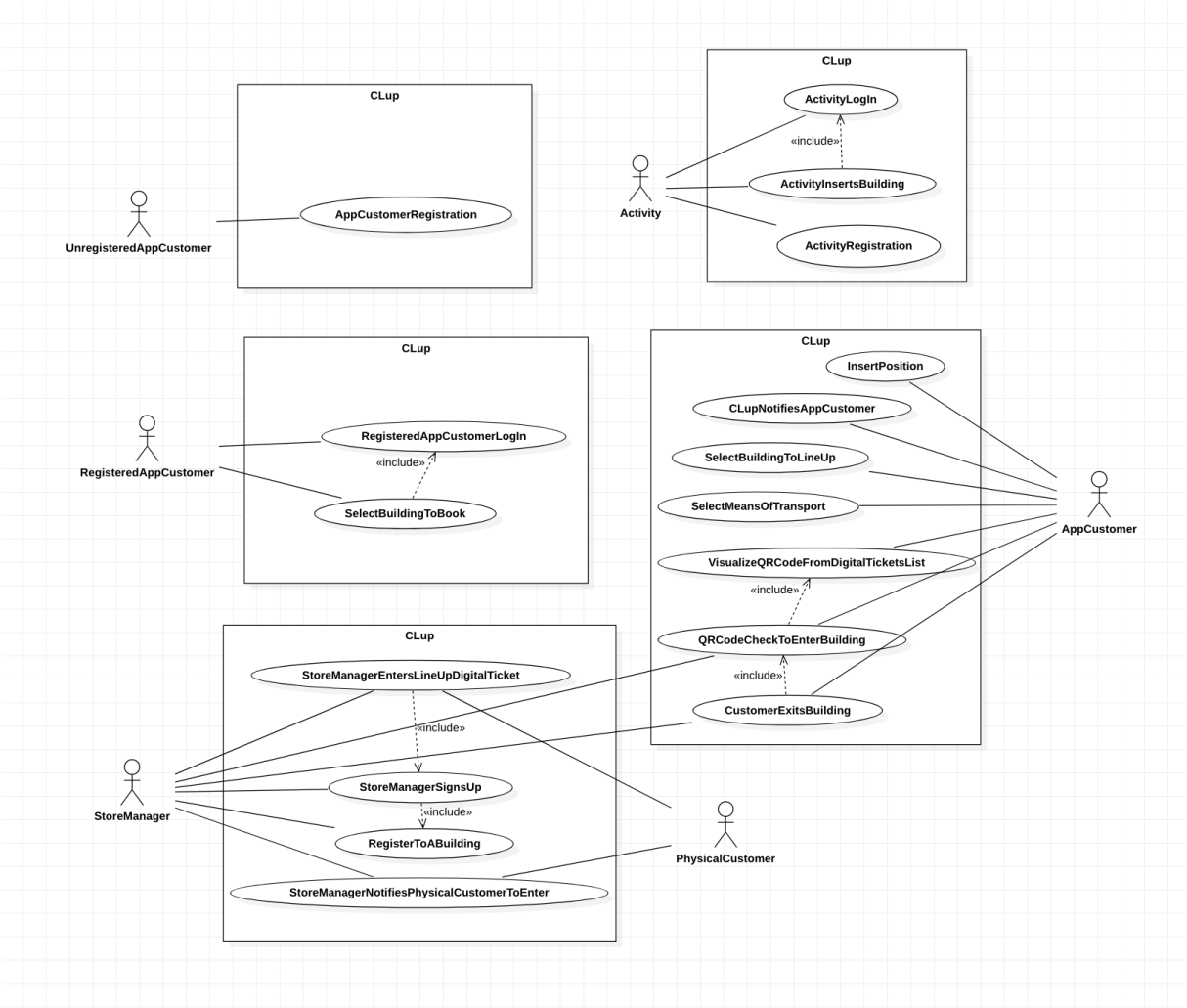

Figure 4: Diagram of use cases

B.2 Activity diagram for Customers

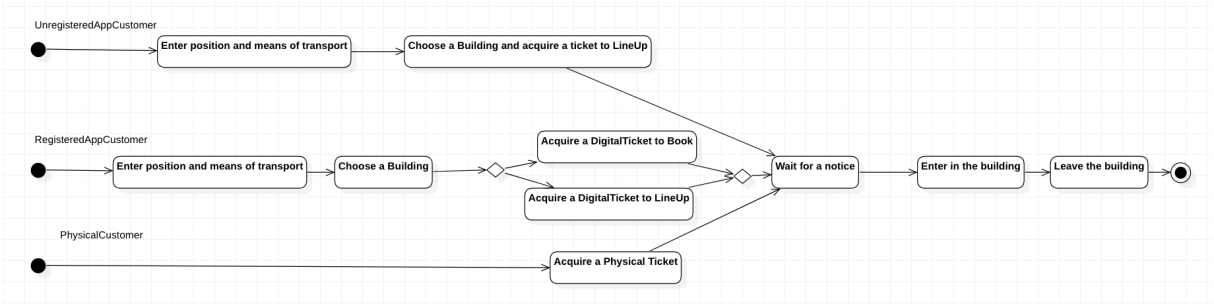

Figure 5: Activity diagram of Customers

B.3 Use Cases and associated Sequence Diagrams

AppCustomer inserts position

Actors

- AppCustomer

Entry Condition

- True

Flow of Event

1. In the homepage, AppCustomer clicks “set position” entering in the localization setting page
2. CLup replies with a pop up providing a choice of preference on GPS localization access by CLup (“Allow CLup to access the GPS position?”)
3. AppCustomer that denied GPS access, has to enter an address manually
4. CLup saves the position acquired from the AppCustomer

Exit Conditions

- CLup has acquired a valid position for AppCustomer

Exceptions

i. InvalidAddress, a manually entered address does not exist or it’s not well formed. Exception handled giving an “invalid address” error and taking back the event flow to event 4.

Special Requirements

- CLup pop up to let the Customer choose whether to allow CLup to access the GPS or not

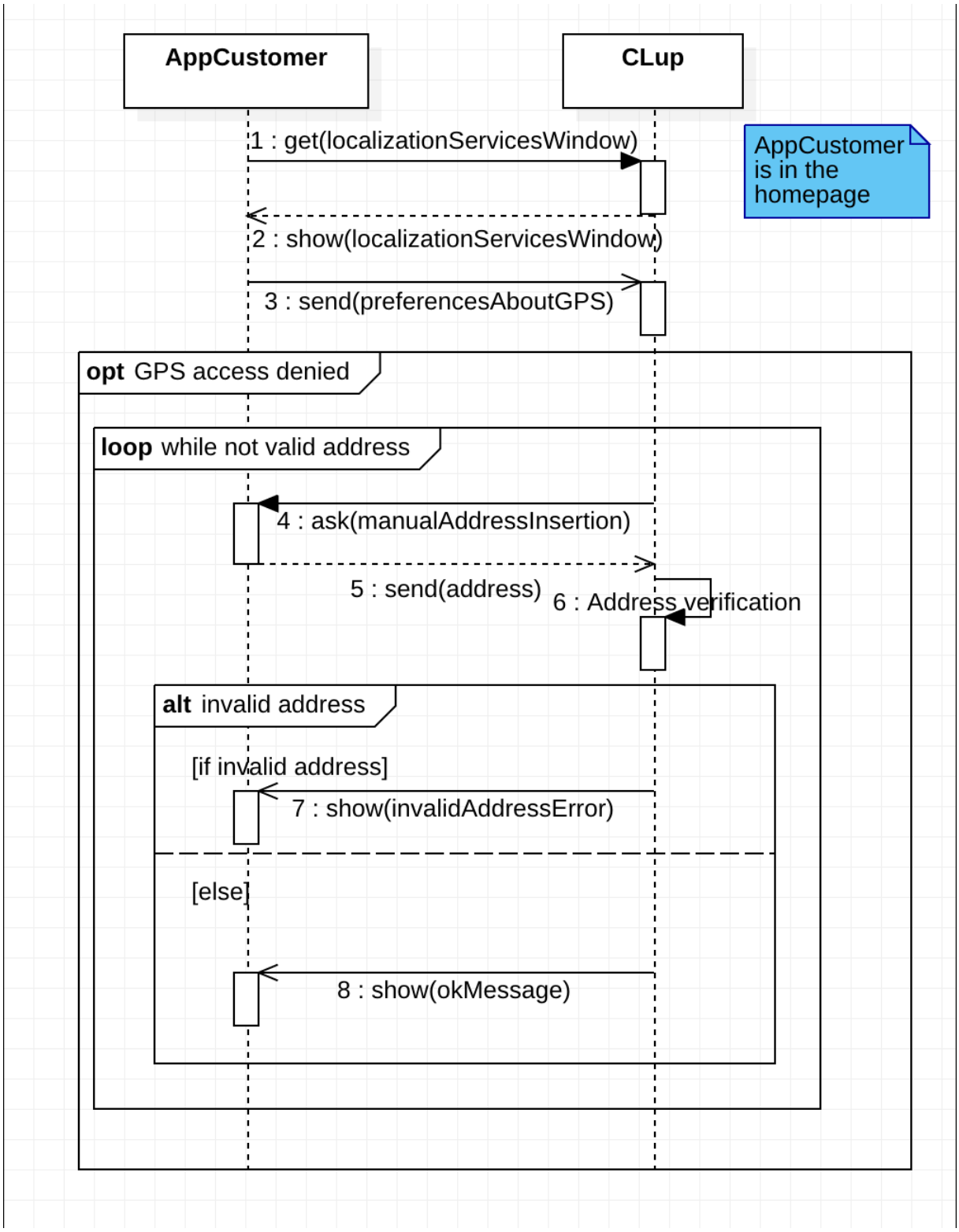

Figure 6: AppCustomer inserts position Sequence Diagram

AppCustomer selects Building to line up

Actors

- AppCustomer

Entry Condition

- AppCustomer has already submitted his position
- AppCustomer has already chosen the means of transport

Flow of Event

1. In the homepage, AppCustomer clicks the “lineUp” button to request a list of reachable Buildings
2. CLup replies with a list of Buildings according to the Customer’s position, sorting them by distance (Each of them has the number of Customers in the queue waiting to enter)
3. AppCustomer selects the Building to queue for from the list
4. CLup shows summary of the selected Building containing: the number of customer inside, the estimated waiting time and the distance
5. AppCustomer click “Confirm” button
6. CLup saves the LineUpDigitalTicket

Exit Conditions

- CLup has acquired a DigitalTicket for the AppCustomer’s selected Building

Exceptions

i. No available Buildings, Exception handled giving a “Not available Buildings” message and taking the AppCustomer to homepage

ii. Estimated WaitingTime greater than closing time for selected Building, exception handled giving a “Building will close” message and taking the Customer back to event 3

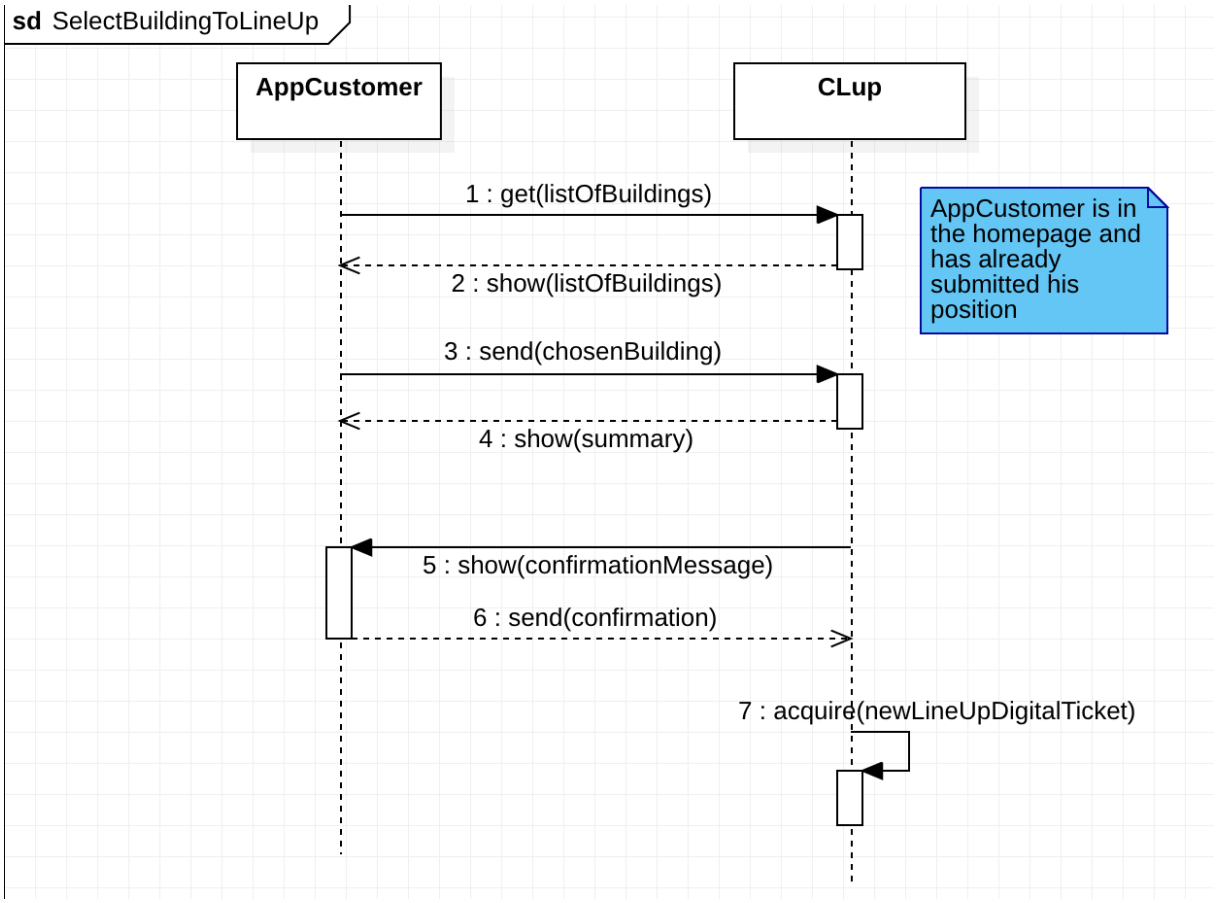

Figure 7: AppCustomer selects Building to line up Sequence Diagram

CLup notifies AppCustomer

Actors

- AppCustomer

Entry Condition

- AppCustomer has already acquired a DigitalTicket
- DigitalTicket's estimated WaitingTime - TravelTime less than or equal to 0
- AppCustomer has not already received a notification for his DigitalTicket

Flow of Event

1. CLup notifies the AppCustomer to leave for the Building associated to his DigitalTicket
2. AppCustomer get to acquired DigitalTicket page by clicking on the notification
3. (Optionally) Customer presses "directions" being redirected to "Google Maps"

Exit Conditions

- AppCustomer has been notified to start reaching the Building for which he has a DigitalTicket

Exceptions

i. NotificationRemoved, AppCustomer doesn't click on the notification. Exception handled by skipping to the end of the Flow of events

Special Requirements

- CLup redirects to "Google Maps" application, giving information on how to reach the selected Building by the chosen means of transport

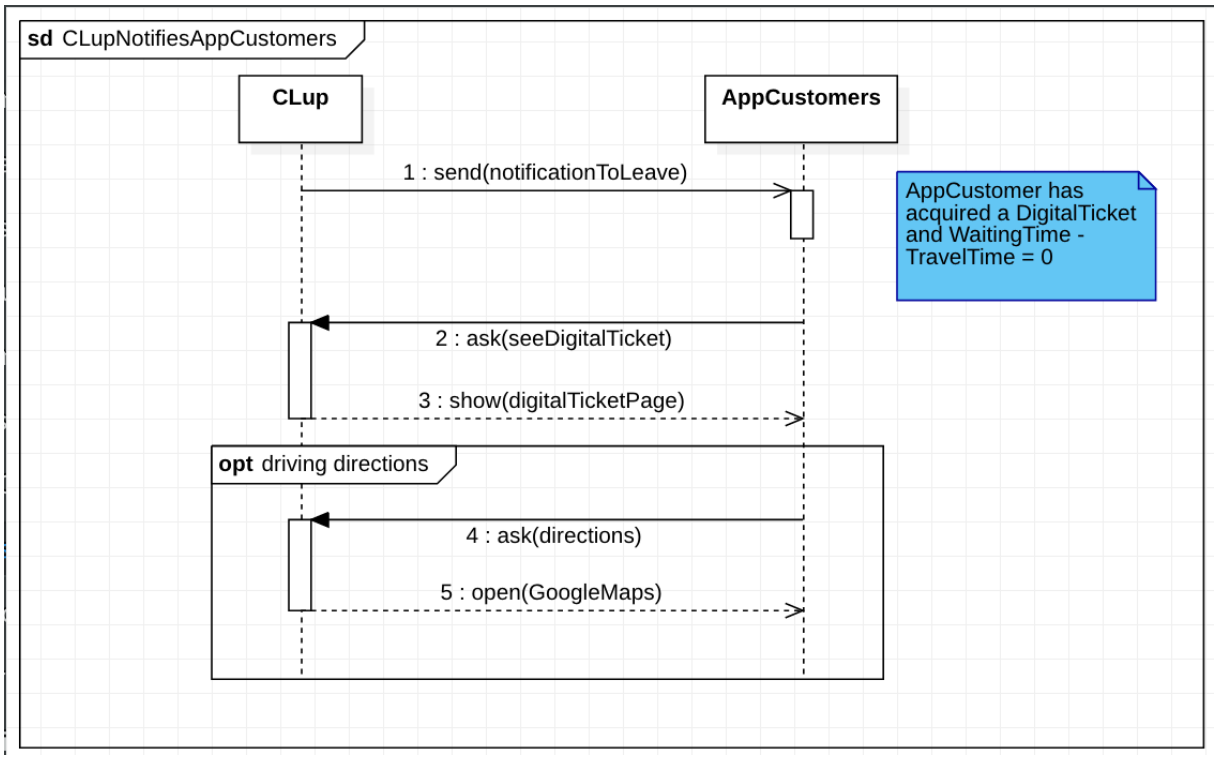

Figure 8: CLup notifies AppCustomer Sequence Diagram

Visualize QRCode from DigitalTicket list

Actors

- AppCustomer

Entry Condition

- True

Flow of Event

1. In the homepage, AppCustomer presses “DigitalTickets list” button
2. CLup replies presenting a list of all the DigitalTickets acquired by the AppCustomer except for the ones that became ExpiredDigitalTicket for at least one day, providing for each of them a brief summary
3. AppCustomer chooses one of his DigitalTickets and clicks on it in order to visualize the QRCode associated
4. CLup replies with presenting the QRCode uniquely associated with the Customer’s DigitalTicket

Exit Conditions

- AppCustomer visualizes a QRCode, associated with an acquired DigitalTicket

Exceptions

i. NoDigitalTicketAvailable, no DigitalTicket has been acquired by AppCustomer. Exception handled by giving a “no digital ticket acquired” message and taking the AppCustomer back to the homepage

Special Requirements

- LineUpDigitalTicket summary contains : Building’s name, number of people with higher priority in the queue
- BookingDigitalTicket summary contains : Building’s name, ArrivalTime and ExitTime

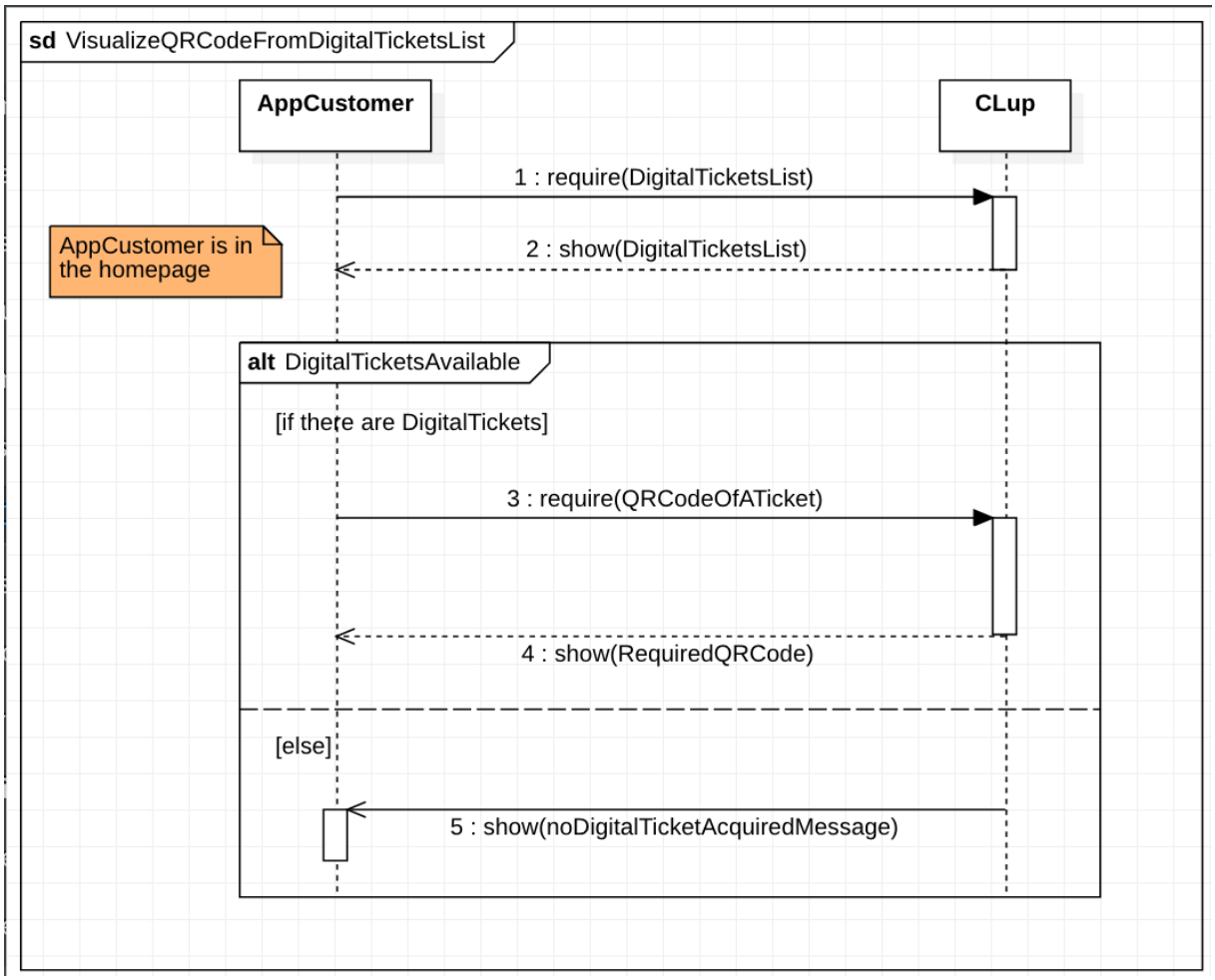

Figure 9: Visualize QRCode from Digital Ticket list Sequence Diagram

QRCode check to enter the Building

Actors

- AppCustomer
- StoreManager

Entry Condition

- AppCustomer visualize QRCode he want to exhibit
- Store Manager has already signed up for a Building

Flow of Event

1. AppCustomer shows QRCode associated with its DigitalTicket to the StoreManager
2. StoreManager clicks on “scan QRCode” , scans QRCode and sends the scanned QRCode to CLup
3. CLup verifies QRCode and sends response to Store Manager
4. StoreManager allows Customer to enter the Building

Exit Conditions

- AppCustomer enters the Building

Exceptions

i. QRCodeResponseNotPositive, response for scanned QRCode received by StoreManger is a NegativeNotification. Exception is handled stopping the use case at 3rd step

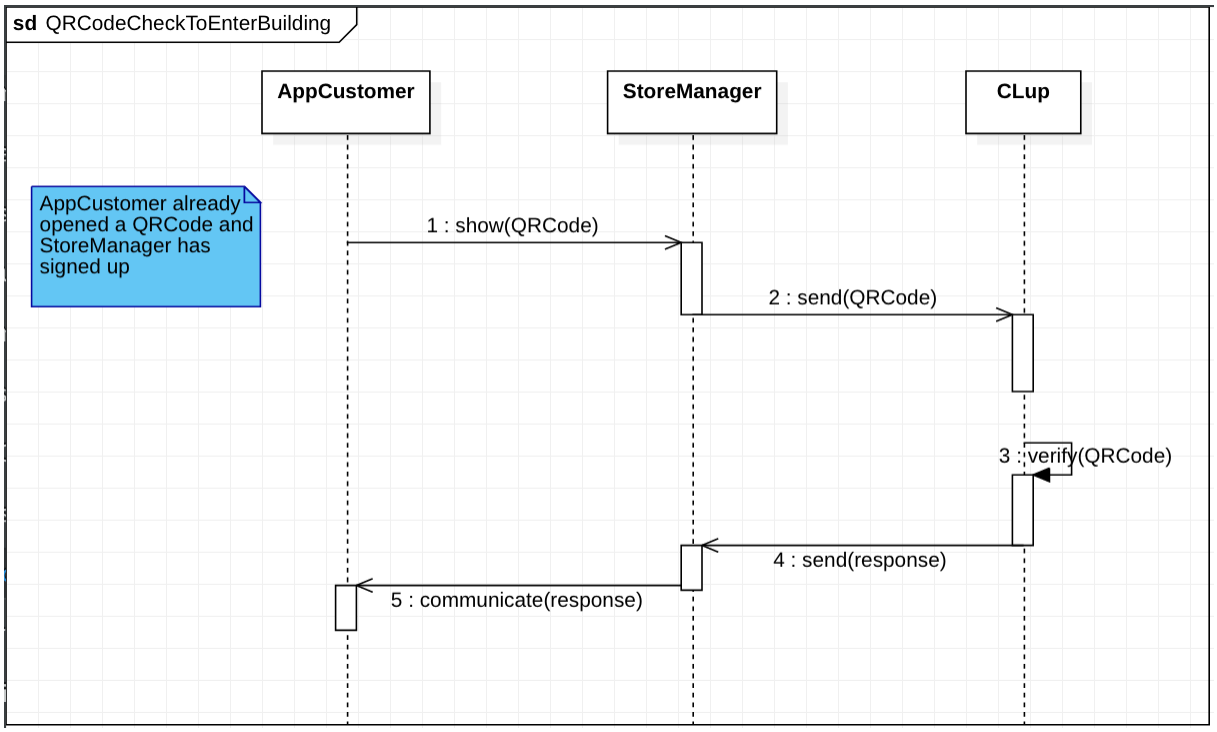

Figure 10: QRCode check to enter the Building Sequence Diagram

Customer exits Building

Actors

- Customer
- StoreManager

Entry Condition

- Customer is inside the Building and approaches to leave it

Flow of Event

1. Customer leaves the Building
2. A StoreManager presses “left” button, to inform CLup that a Customer left the Building
3. CLup queue is updated

Exit Conditions

- Customer lefts the Building and CLup is updated

Exceptions

i. BookVisitCustomer, Customer is a RegisterdAppCustomer which owns a BookingDigitalTicket that allows him to visit the Building. Exception handled by making the StoreManager scan the related QRCode. Event flow skips from 1. to 3. CLup will also update average time of stay of the RegisteredAppCustomer

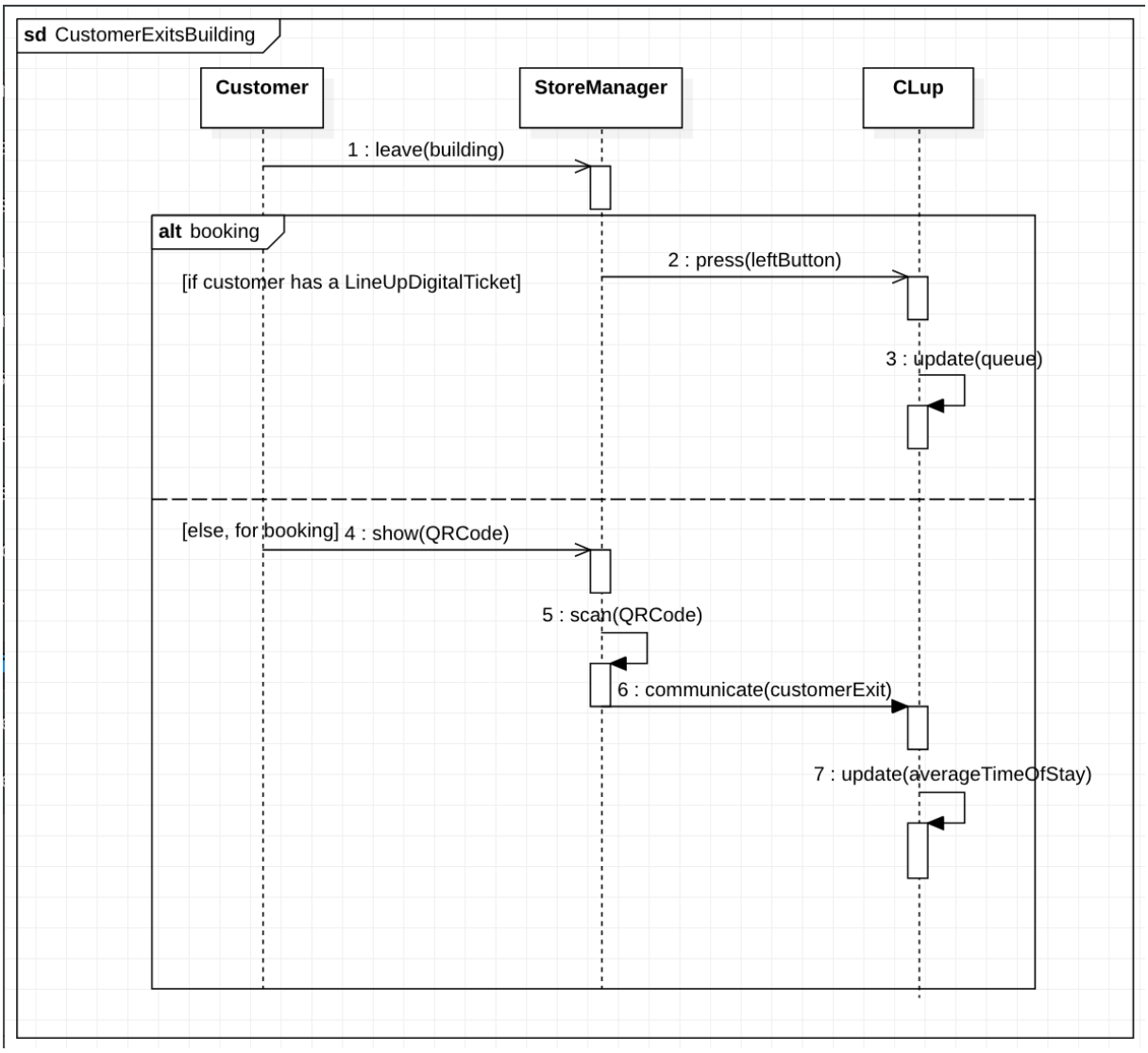

Figure 11: Customer exits Building Sequence Diagram

AppCustomer registers to CLup

Actors

- AppCustomer

Entry Condition

- True

Flow of Event

1. In the homepage, UnregisteredAppCustomer clicks on the “register” button to start the registration process
2. CLup redirect UnregisteredAppCustomer to a registration page that shows a registration form to be filled up with credentials, that are username and password
3. UnregisteredAppCustomer fills username and password and presses the “register” button
4. CLup verify that username inserted is not used by any other AppCustomer
5. CLup saves new account credentials
6. CLup redirect RegisteredAppCustomer to “logged-in-customer homepage”

Exit Conditions

- RegisteredCustomer has an account that can be accessed by username and password provided during registration in registration form
- RegisteredCustomer is now logged in

Exceptions

i. UsernameNotAvailable, username is used by another RegisteredAppCustomer. Exception handled by giving a “username already taken” message and take back to point 2

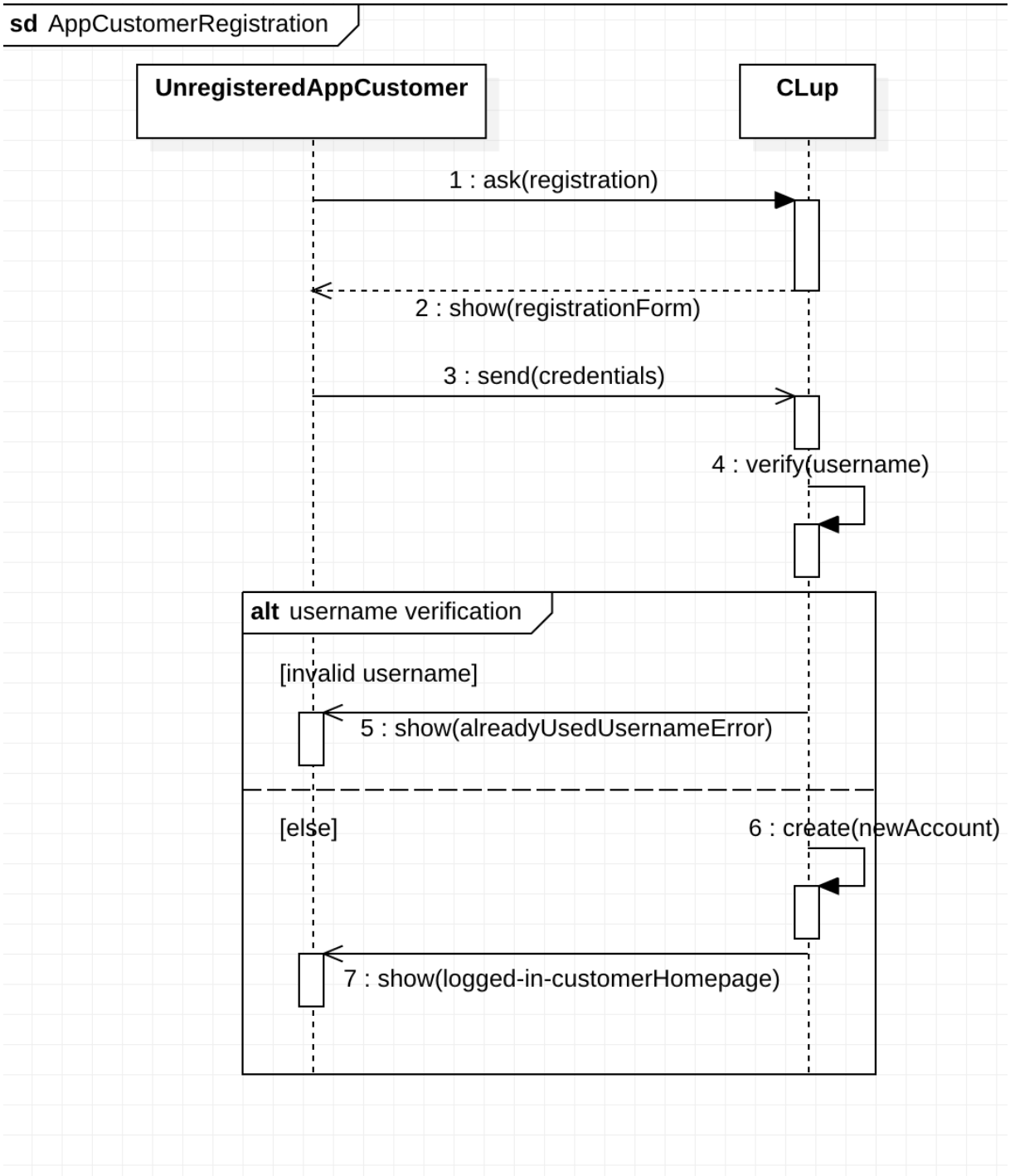

Figure 12: AppCustomer registers to CLup Sequence Diagram

RegisteredAppCustomer logs in

Actors

- AppCustomer

Entry Condition

- RegisteredAppCustomer has not already logged in
- RegisterdAppCustomer has already registered to CLup

Flow of Event

1. In the homepage, RegisteredAppCustomer clicks “login” button to start the login process
2. RegisteredAppCustomer fills up with his account’s username and password, then presses “login” button
3. CLup verify that an account is associated with the inserted username and password
4. CLup redirect user to “logged-in-customer homepage”

Exit Conditions

- RegisterdAppCustomer is now logged in
- RegisteredAppCustomer is now in the logged-in homepage

Exceptions

i. InvalidAccountCredentials, username and password do not match an AppCustomer account. Exception handled by giving an “Invalid username and/or password” message, and taking back the RegisteredAppCustomer to point 2

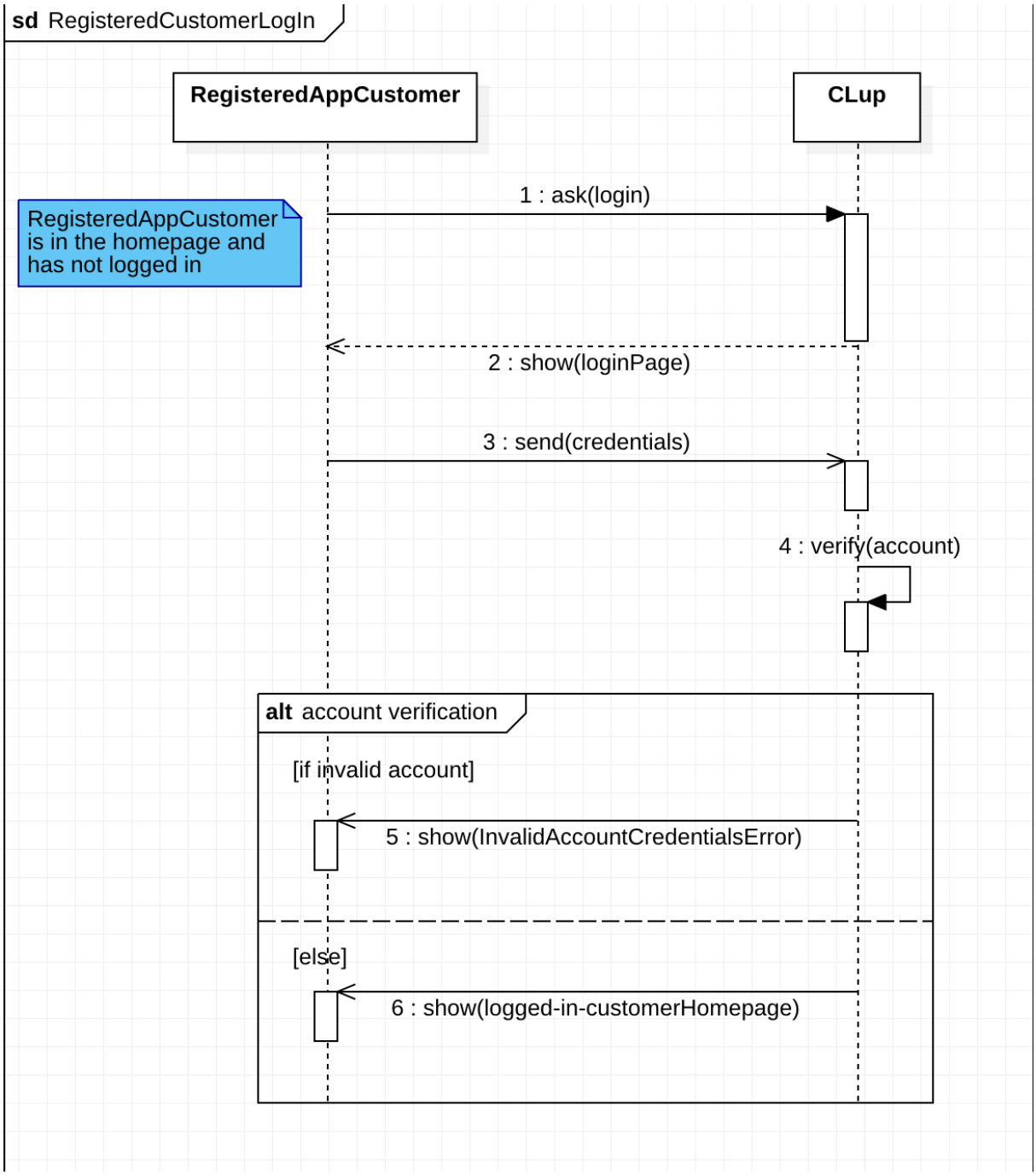

Figure 13: RegisteredAppCustomer logs in Sequence Diagram

RegisteredAppCustomer selects Building to book

Actors

- RegisteredAppCustomer

Entry Condition

- RegisteredAppCustomer has already given a position and means of transport

Flow of Event

1. In the logged-in homepage, RegisteredAppCustomer clicks “book” button to request a list of reachable Buildings
2. CLup replies with a list of Buildings according to RegisteredAppCustomer’s position, sorting them by distance
3. RegisteredAppCustomer selects a Building from the list for which he wants to book a visit
4. RegisterdAppCustomer provides a date for the visit
5. CLup computes the EstimatedPermanenceTime as the average of the previous booked visits duration
6. RegisteredAppCustomer optionally modifies the EstimatedPermanenceTime for the visit
7. RegisteredAppCustomer optionally chooses the set of Departments he wants to visit from the selected Building’s Departments
8. RegisteredAppCustomer clicks “Confirm” button in order to let CLup save the selected preferences
9. CLup provides available TimeSlots (computed according to the preferences of RegisteredAppCustomer)
10. RegisteredAppCustomer selects a TimeSlot and clicks “Acquire Ticket”
11. CLup saves a DigitalTicket made-up from the RegisteredAppCustomer preferences

Exit Conditions

- RegisteredAppCustomer has acquired a BookDigitalTicket for the selected Building, at the preferred TimeSlot
- (if inserted) BookDigitalTicket contains information about Departments user want to visit

Exceptions

i. NoAvailableBuilding, there are no Building to book a visit with selected position and means of transport. Exception handled giving a “No available building to book for your position and means of transport” and taking back RegisteredAppCustomer to logged-in homepage, stopping the use case

ii. NoAvailableTimeSlot, selected Building has no available TimeSlots. Exception handled by giving two types of suggestions to RegisteredAppCustomer. CLup has to suggest at most 3 alternative Buildings of the same Activity sorted by TravelTime and at most 3 alternative TimeSlots for the day (of opening) following the chosen date, sorted by their start time if a Building suggestion is selected, the Building preferences is updated and flow taken back to event 4. If a TimeSlot suggestion is selected, event flow skips to event 11

iii. AlreadyBookedBuilding, RegisteredAppCustomer has already a BookDigitalTicket for the selected Building. Exception handled by giving "One book per Building" and take back flow to event number 3

iv. BookOverelap, RegisteredAppCustomer has a BookDigitalTicket with a TimeSlot that overlaps the current TimeSlot selected. Exception handled by "overlapping time slots" and take back flow to event number 4

v. FirstBooking, cannot compute average time of stay because of no previous data available (no previous visit or no previous exit time for a visit, if one exist average time of stay can be computed). Exception handled by skipping event 5 and force EstimatedPermanence-Time insertion in event 6

vi. NoDepartmentAvailabe, if there are no instantiated Departements for that Building. Exception handled by taking flow to event 8.(skipping events 7)

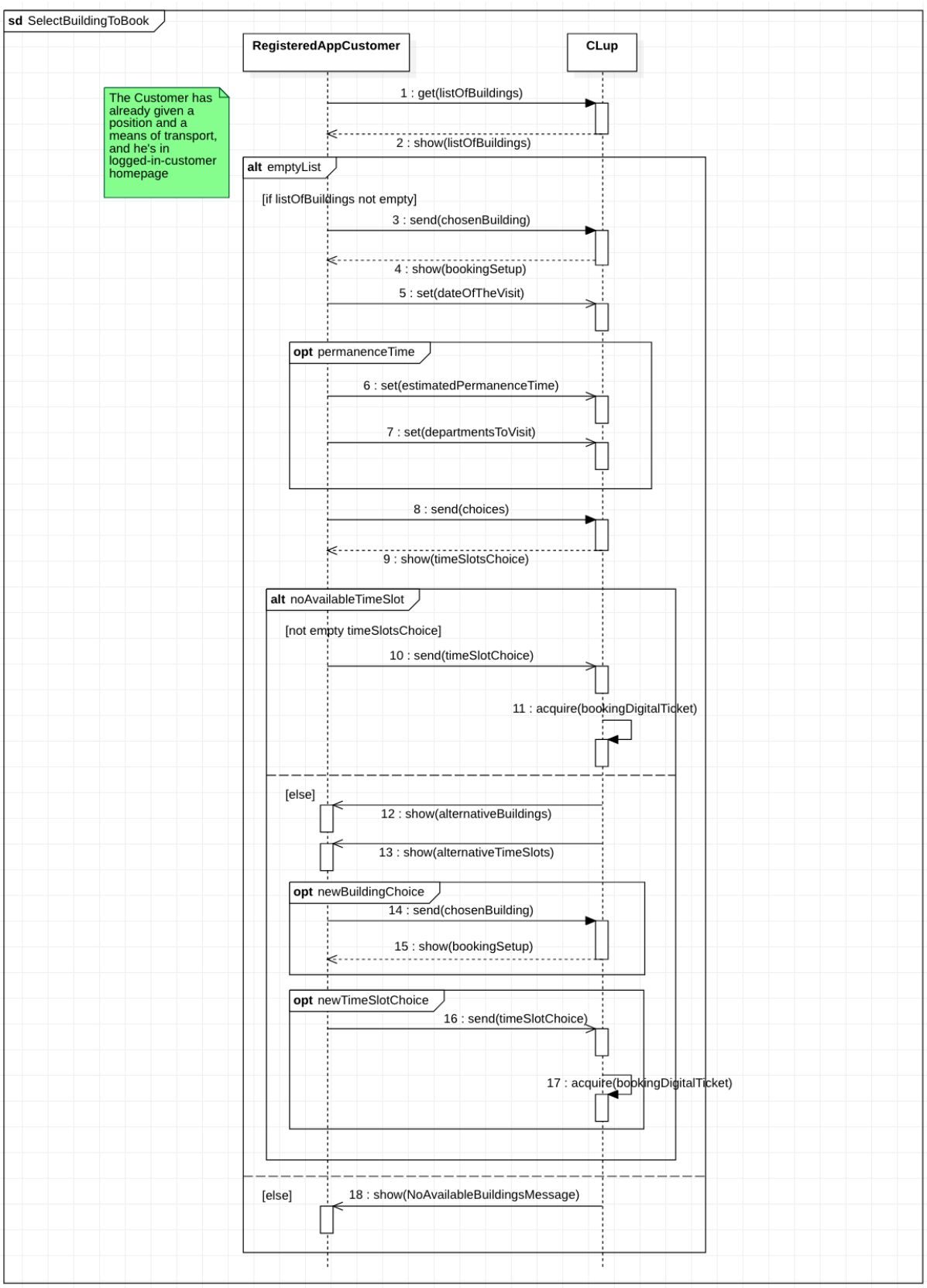

Figure 14: RegisteredAppCustomer selects Building to book Sequence Diagram

AppCustomer selects means of transport

Actors

- AppCustomer

Entry Condition

- True

Flow of Event

1. In the homepage, AppCustomer clicks button “select means of transport”
2. (Optionally) AppCustomer sets a “max distance” parameter, if AppCustomer wants to shrink the search area
3. AppCustomer choices the means of transport (Driving, Transit, Walking, Cycling) and clicks “confirm” button
4. CLup saves AppCustomer’s preferences

Exit Conditions

- CLup has acquired AppCustomer’s preferred means of transport for next ticket and optionally a maximum search distance

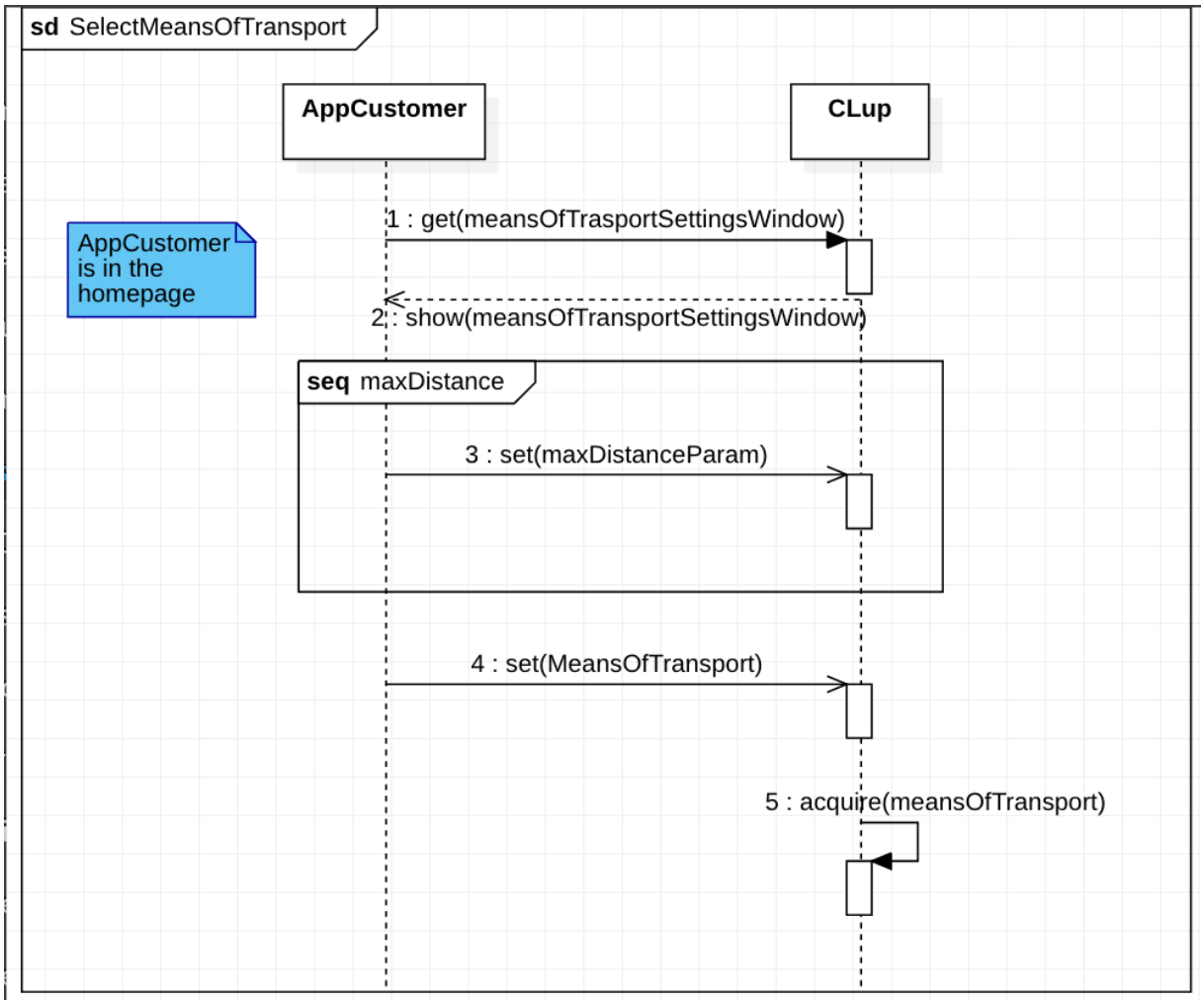

Figure 15: AppCustomer selects means of transport Sequence Diagram

Activity registers to CLup

Actors

- Activity

Entry Condition

- True

Flow of Event

1. In the homepage, Activity presses the “Activity registration” button in order to begin the registration process
2. Activity enters its username, password and P.Iva
3. CLup saves an account for Activity

Exit Conditions

- Activity has registered to CLup with provided username, password and P.Iva

Exceptions

i. AlreadyRegisteredActivity, Activity has already registered to CLup. Exception handled giving an “Activity already registered” message and returning to step 2

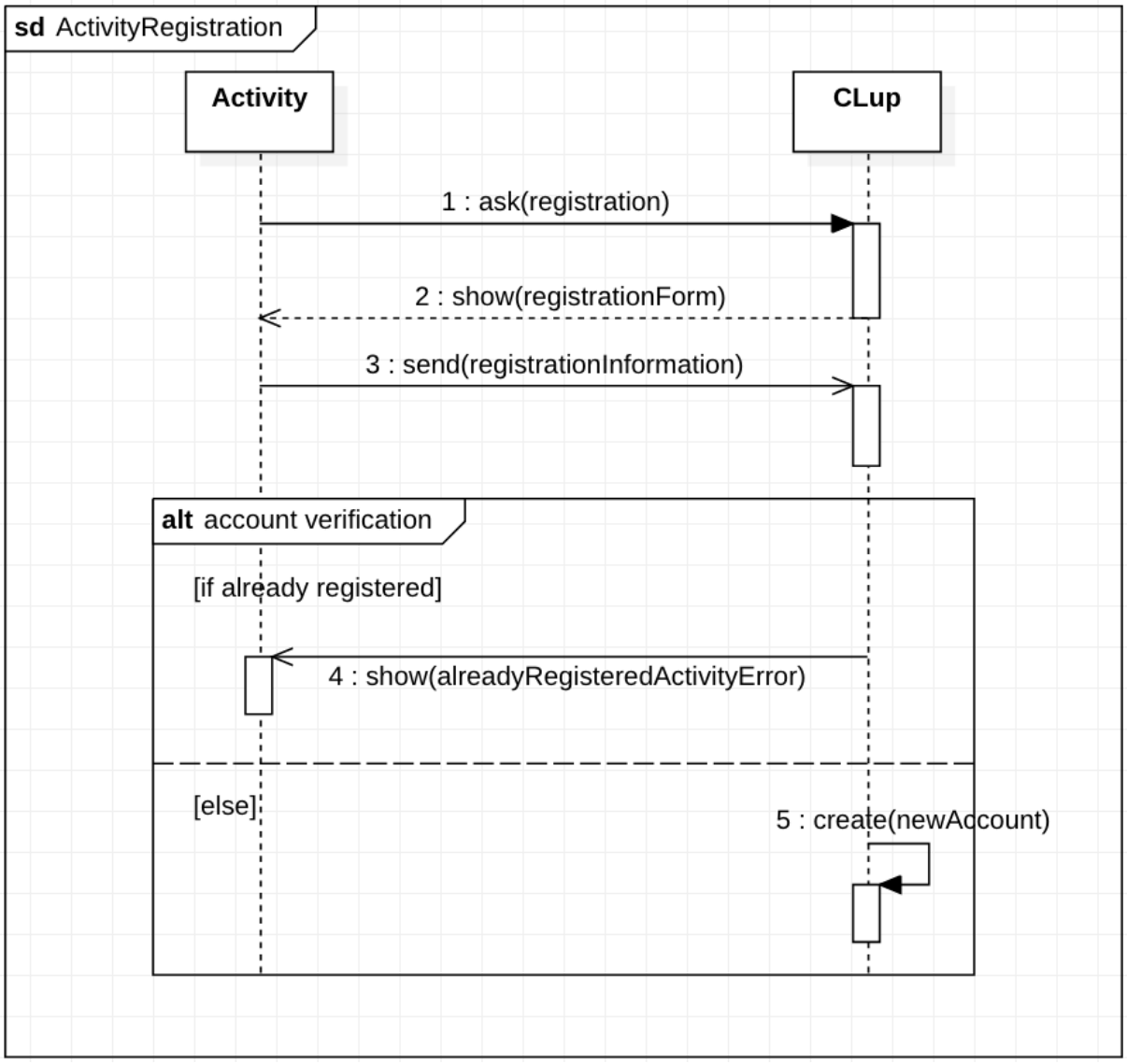

Figure 16: Activity registers to CLup Sequence Diagram

Activity logs in

Actors

- Activity

Entry Condition

- Activity has already registered to CLup
- Activity has not already logged in

Flow of Event

1. Activity clicks “Activity login” button in order to log in to CLup as an Activity
2. Activity provide username and password associated with its account
3. CLup redirect Activity to activity homepage

Exit Conditions

- Activity has registered to CLup with provided name

Exceptions

i. InvalidAccountCredentials, username and password do not match an Activity account. Exception handled by giving an “Invalid username and/or password” message, and taking back the Event Flow to point 2

Activity inserts Building

Actors

- Activity

Entry Condition

- Activity has already logged in

Flow of Event

1. In Building Management page, Activity clicks “add new Building”
2. Activity provides a name, an address, a BuildingCapacity and opening and closing time
3. (optionally) Activity provides at most 20 Departments name with a SurplusCapacity for each of them
4. Activity clicks “confirm”
5. CLup saves the new Building
6. CLup produces, saves and shows a BuildingAccessCode for the new Building

Exit Conditions

- Activity has added a new Building

Exceptions

i. NonUniqueAddress, address inserted for a Building is not unique among all Buildings in CLup. Exception handled by giving “building already added”, and stopping the use case, taking back Activity to Building Manager page

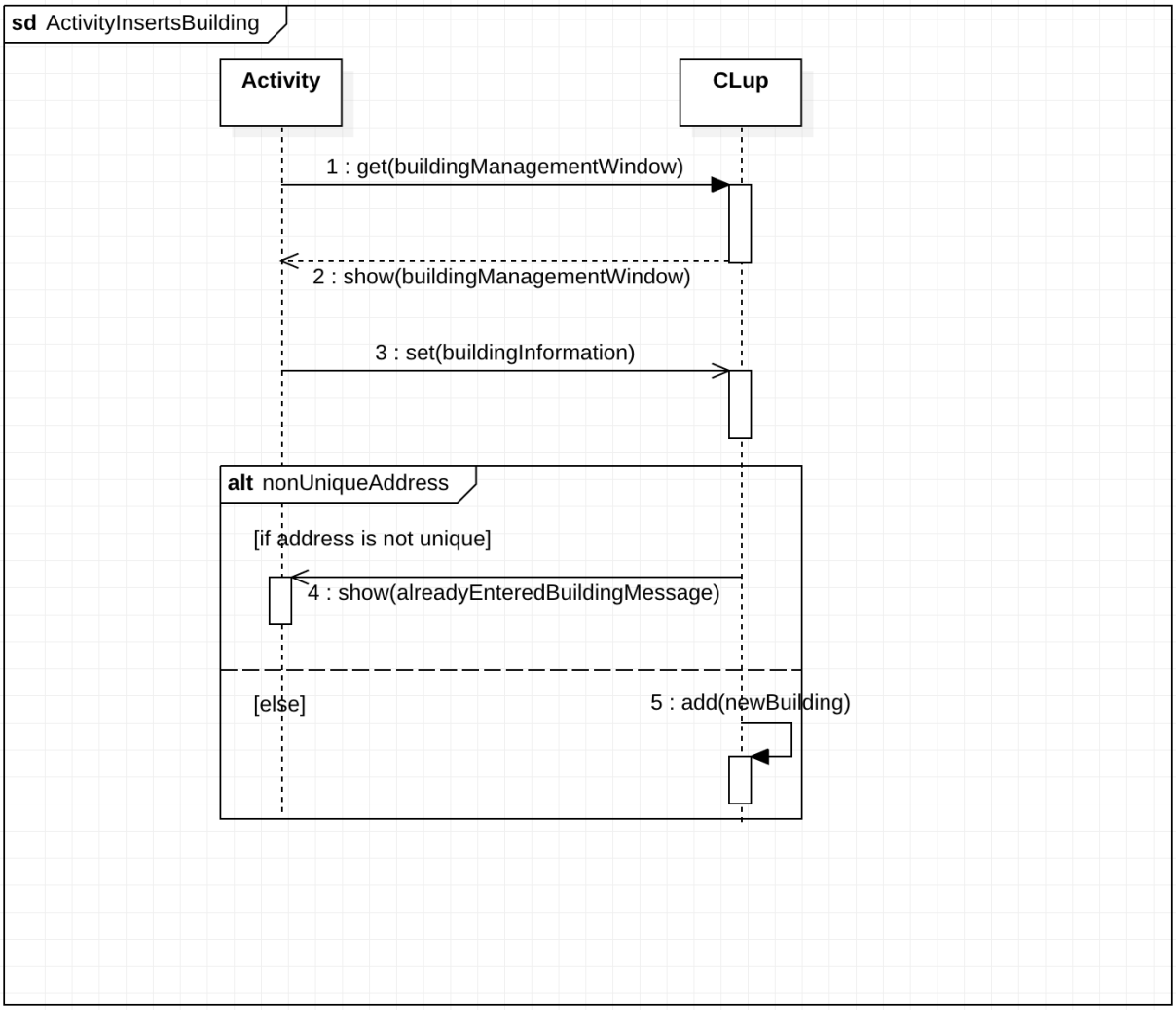

Figure 17: Activity inserts Building Sequence Diagram

StoreManager signs up

Actors

- StoreManager

Entry Condition

- StoreManager who wants to sign up for a Building, has received the BuildingAccessCode associated with that Building

Flow of Event

1. In the homepage, StoreManager clicks on “sign up as store manager” button
2. StoreManager provides the BuildingAccessCode
3. CLup redirects StoreManager to StoreManager page
4. CLup updates

Exit Conditions

- StoreManager has signed up for the Building associated with his BuildingAccessCode

Exceptions

i. WrongBuildingAccessCode, code inserted is not a valid one. Exception handled by giving “wrong access code” and taking back flow to event 2

StoreManager enters LineUpDigitalTicket

Actors

- StoreManager
- PhysicalCustomer

Entry Condition

- PhysicalCustomer is near the Building
- StoreManager has already signed up for the Building

Flow of Event

1. PhysicalCustomer requests a PhysicalTicket from the StoreManager
2. StoreManager clicks on “enter DigitalTicket”
3. CLup sends a confirmation message
4. CLup updates
5. StoreManager gives PhysicalTicket to the PhysicalCustomer

Exit Conditions

- PhysicalCustomer acquired a PhysicalTicket and StoreManager has entered a LineUpDigitalTicket associated with it

Exceptions

i. ClosingBuilding, estimated WaitingTime greater than closing time of the Building. Exception handled giving a “Building close to closing” message to the StoreManager and stopping the use case at step 2

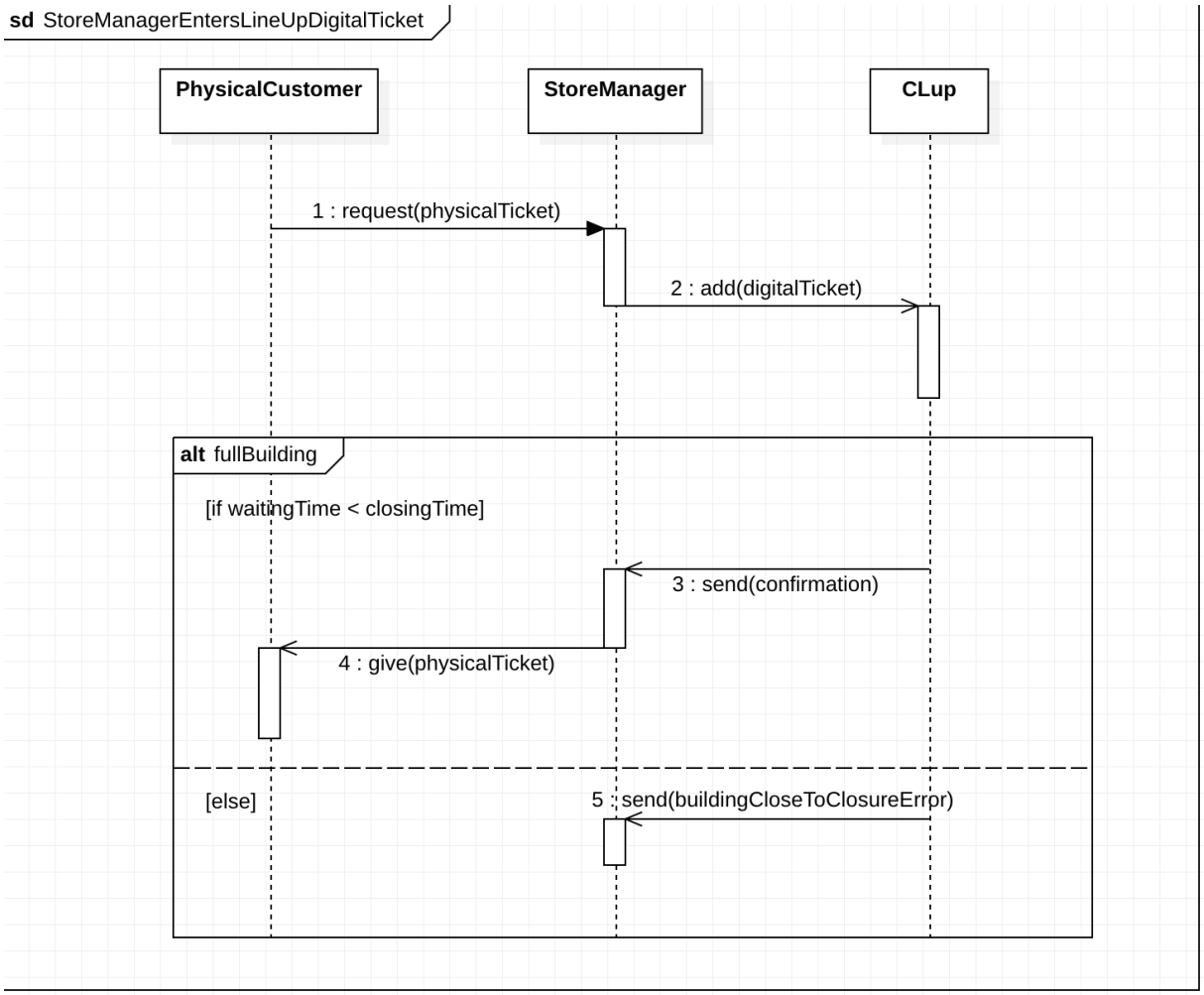

Figure 18: StoreManager enters LineUpDigitalTicket Sequence Diagram

StoreManager notifies PhysicalCustomer to enter

Actors

- StoreManager
- PhysicalCustomer

Entry Condition

- LineUpDigitalTicket entered by the StoreManager for a PhysicalCustomer becomes Valid

Flow of Event

1. CLup notifies StoreManager, providing the number associated with LineUpDigitalTicket that became Valid
2. StoreManager notifies PhysicalCustomers waiting near the Building providing them the number received by CLup
3. PhysicalCustomer shows PhysicalTicket to the StoreManager
4. PhysicalCustomer enters the Building

Exit Conditions

- PhysicalCustomer has entered the Building for which he was waiting

Exceptions

i. WrongPhysicalTicketNumber, a PhysicalCustomer shows a PhysicalTicket with a number different from the one provided by the StoreManager. Exception handled by taking back the flow to event 2

ii. NoCustomerAvailable, there isn't a PhysicalCustomer with the PhysicalTicket's number provided by StoreManager. Exception handled by stopping the use case, only after repeating once the event 2

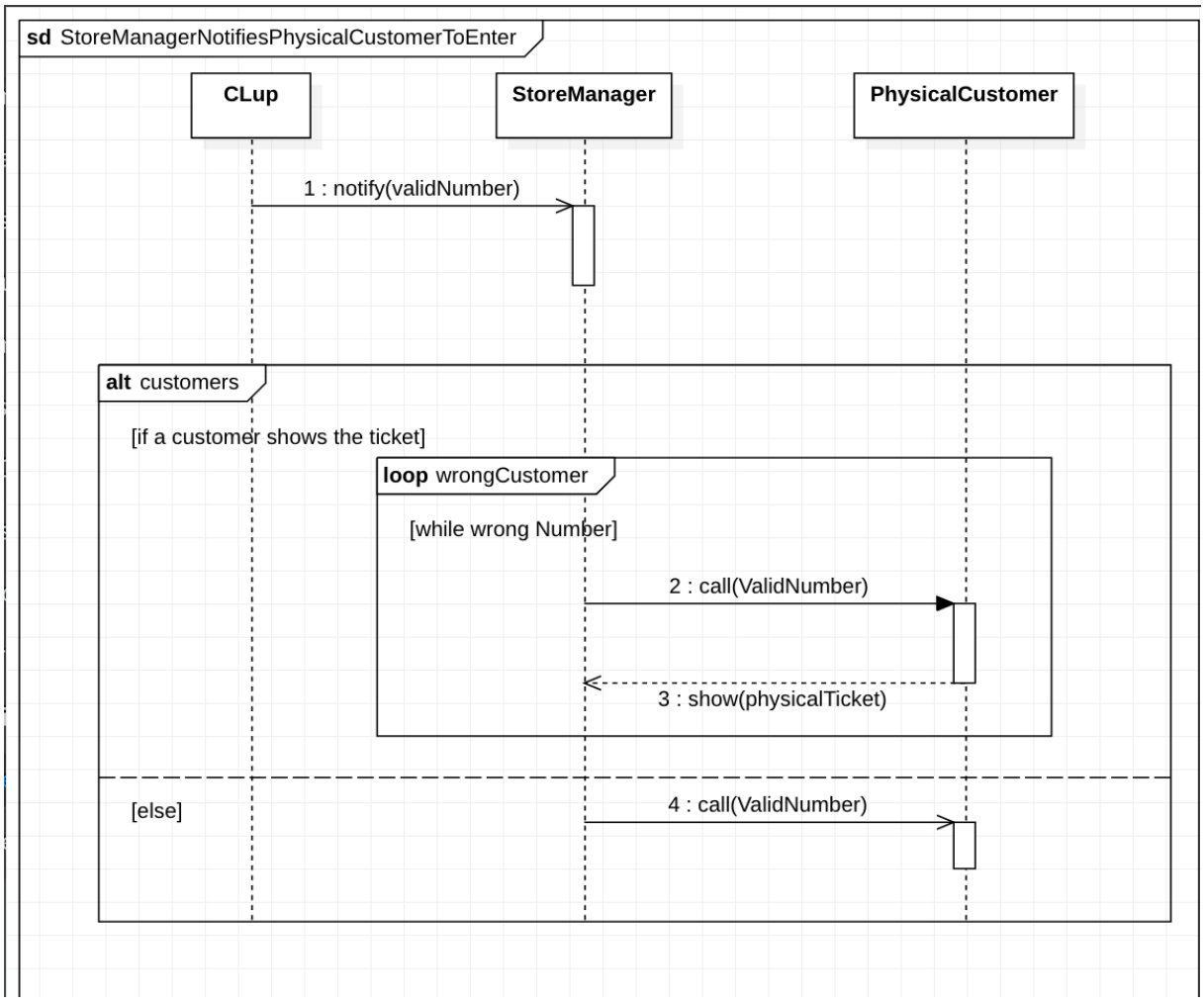

Figure 19: StoreManager notifies PhysicalCustomer Sequence Diagram

B.4 Mapping table of requirements

GOAL	*DOMAIN ASSUMPTION*	*REQUIREMENTS*
G1	D1, D2	R1, R2
G1.1		R3, R4
G2	D3, D4, D5, D6, D7	R5, R6, R7, R8, R9
G3	D8	R8, R10, R11
G4		R12, R13, R14, R15, R16
G4.1		R12, R13, R14, R15, R16
G5	D9	R18, R19, R20, R21, R22
G5.1		R23, R24, R25
G5.2		R26
G5.3		R27
G5.4		R28
G6		R22, R29, R30, R31, R32
G7		R16
G7.1	D10	R33, R34
G7.2	D11, D12	R35, R36, R37, R38, R39, R40, R41, R42, R43, R44, R45, R46, R47, R48, R49, R50, R51, R52, R53
G8	D13, D14, D15, D16, D17	R54, R55, R56, R57, R58, R59, R60, R61, R62, R63
G8.1	D18	R16, R64
G9	D19, D20, D21	R65, R66, R67
G10		R68, R69, R70, R71
G11	D21, D22	R54, R72, R73

C Performance Requirements

CLup must guarantee the simultaneous connection of a large number of people, but it can vary according to the performance of the App in the market. To make the app effective, according to its aim, it must be assumed that most people who own a smartphone will use it, so the minimum number of connections will depend on the territory on which the app is used. We also assume that the number of people queuing is slightly higher (60%) than the number of people booking. Moreover, during the night, the load to be managed will be almost 80% lower, since fewer people are expected to visit stores. The system should process requests in 5 seconds or less.

D Design Constraints

D.1 Standards Compliance

No particular standard is required to develop this system.

D.2 Hardware Limitations

Store Manager devices must necessarily have a camera in order to scan Customers' QR codes. There are no other hardware limitations since the interfaces are quite simple.

D.3 Any other constraint

All the Actors must have an internet connection for their mobile device used to access CLup.

E Software System Attributes

E.1 Reliability

The system must be available 24/7, except for any short unexpected interruption. The average time between the occurrence of a fault and the recovery must be of max 15 minutes. That would not be so serious since CLup doesn't manage emergency situations, however this will result in a delay in entering the building.

E.2 Availability

The system should be able to respond quickly (in average at most 15 minutes) to faults and to have a fast recovery since building management would be suspended in case of service interruption. Therefore, there must be personnel ready to solve any unexpected issue. All the Actors using the App have to be notified when a system failure occurs.

E.3 Security

The credentials of AppCustomers, Activities and StoreManagers will be stored by CLup, but they must be kept safe, preventing anyone external to the system from accessing them. Also any other information, such as AppCustomers' position or Activities' P.Iva, will be visible only to the system.

E.4 Maintainability

The implementation of the system must guarantee that small adjustments can be made in an easy way. The system has to have a high maintainability especially in order to face possible variations in the number of requests. To do so, the code must be properly commented and organized, avoiding hard-coding.

E.5 Portability

Client interfaces of the system will be available as a mobile App for the main operative systems, Android and iOS.

4 Formal Analysis Using Alloy

A Metamodel

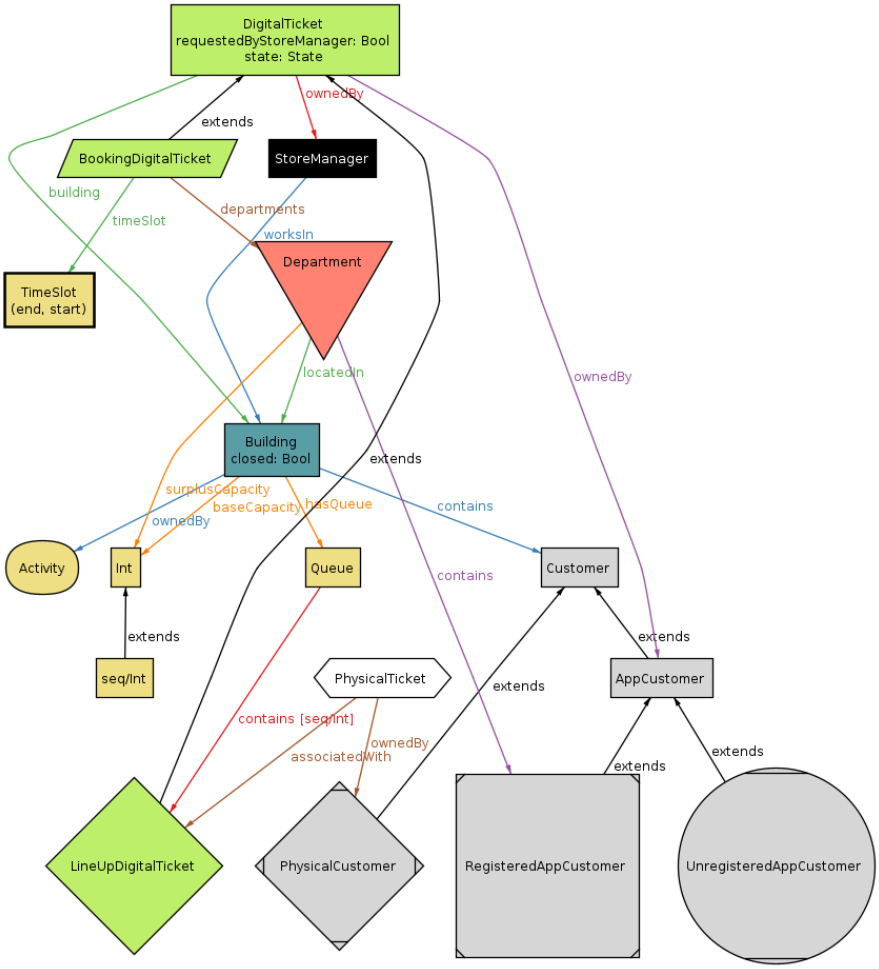

Figure 20: Alloy Metamodel

B Dynamic Examples

Dynamic examples projected on time.

```
run {(some q: Queue, t: Time | #q.contains[t] = 2) and (some t: Time, b: Building, ac:
    ↪ AppCustomer | ac.enters[b,t])} for 3 but 0 PhysicalCustomer, exactly 3 AppCustomer
    ↪ , exactly 1 Building
```

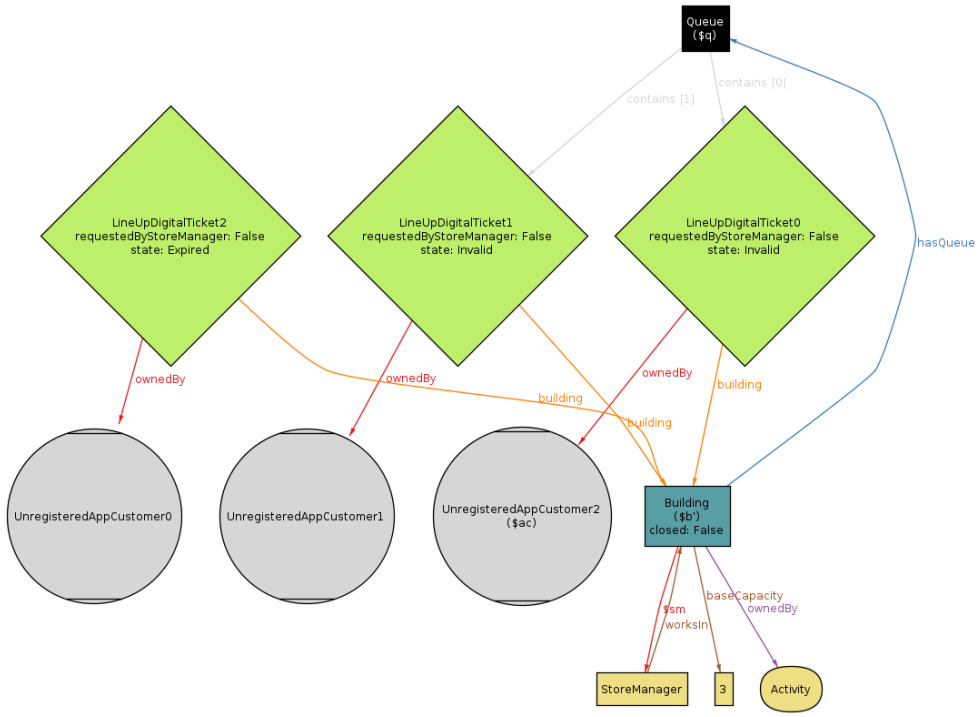

Figure 21: Time 0 - Initial condition with three line-up digital tickets and corresponding unregistered app customers, a building and its queue

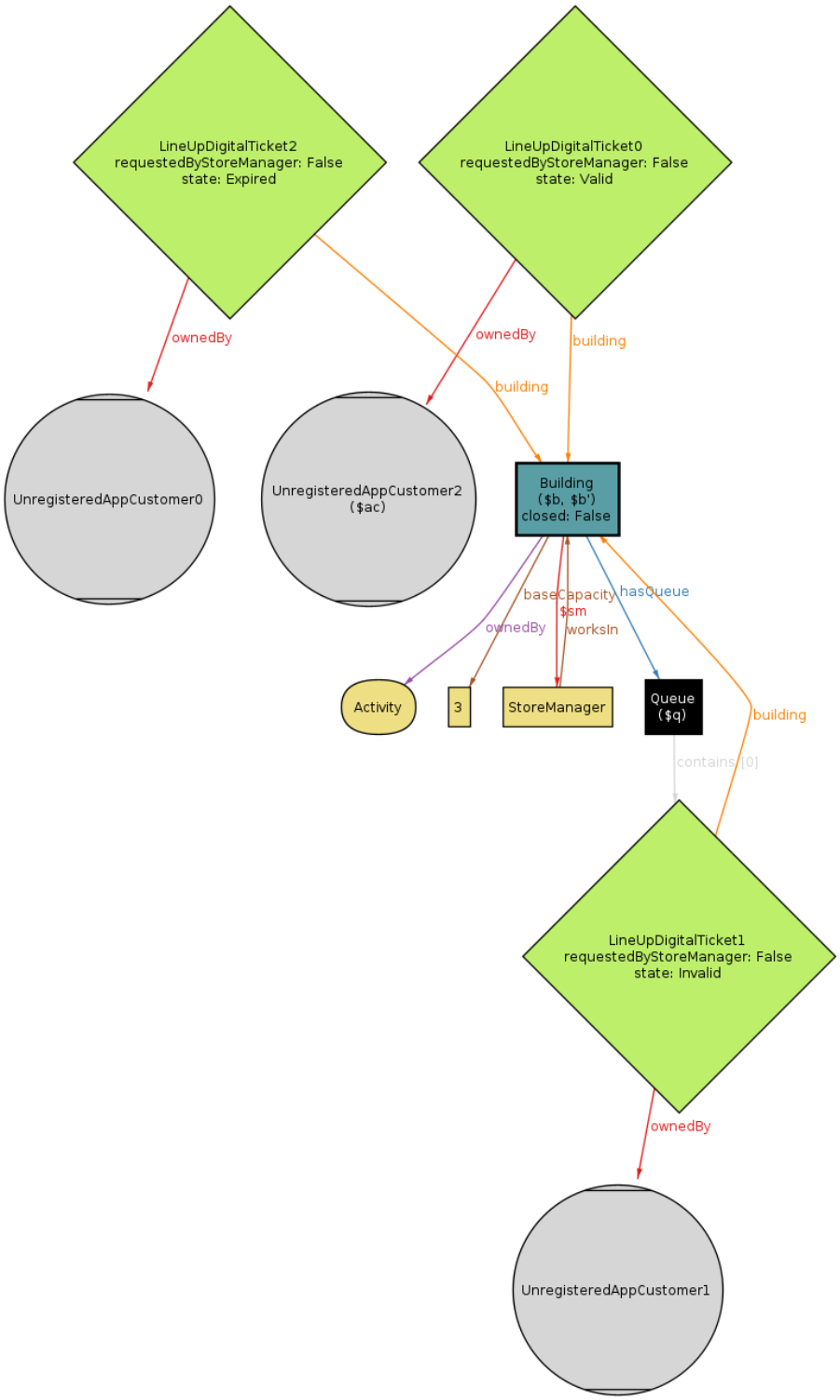

Figure 22: Time 1 - UnregisteredAppCustomer1 can now enter the building

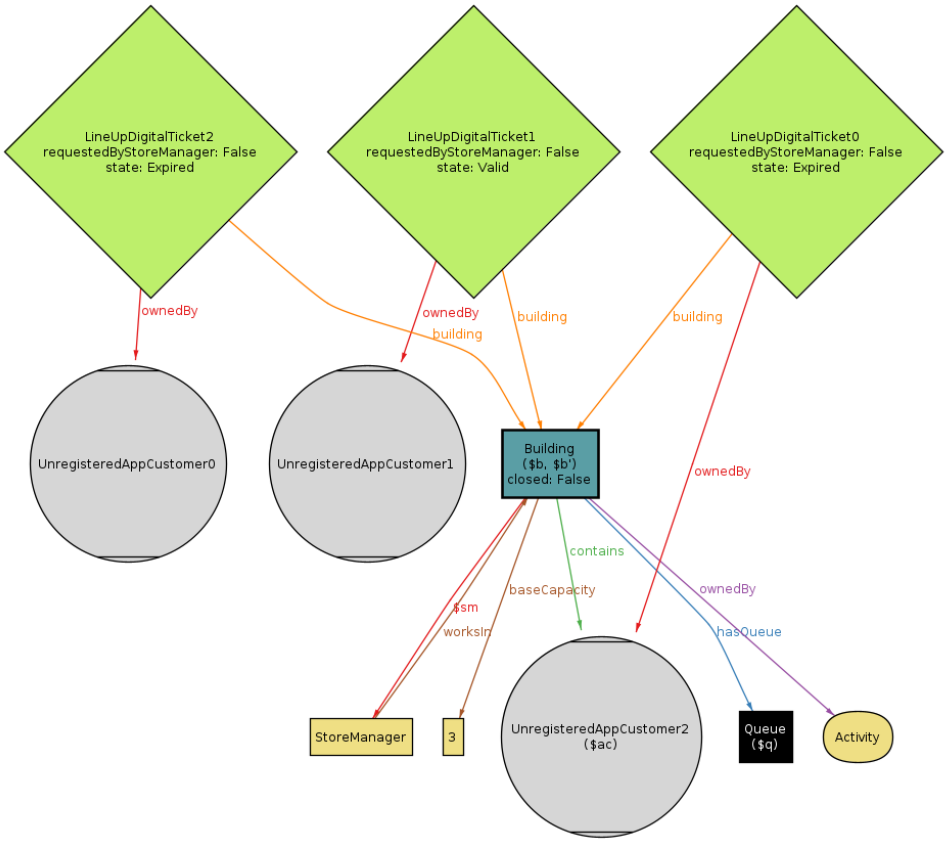

Figure 23: Time 2 - UnregisteredAppCustomer2 enters the building

C Alloy Code

```
//Utilities Signatures
open util/time
open util/sequniv

abstract sig Bool {}
one sig True extends Bool {}
one sig False extends Bool {}



//Customers signatures

abstract sig Customer {}

abstract sig AppCustomer extends Customer {}
sig PhysicalCustomer extends Customer {}

sig RegisteredAppCustomer extends AppCustomer {

}
sig UnregisteredAppCustomer extends AppCustomer {}

//---------------------------------------------------------------

//Store Manager signature

sig StoreManager {
        worksIn: one Building
}

//Activity signature

sig Activity {}


//Building signature

sig Building {
        ownedBy: Activity,
        closed: dynamic[Bool],
        baseCapacity: Int,
        contains: dynamicSet[Customer],
        hasQueue: one Queue

} {
        baseCapacity > 0 and all t: Time | #(noDepCustomers[t]) ≤ baseCapacity
}


sig Department {
        locatedIn: one Building,
        surplusCapacity: Int,
        contains: dynamicSet[RegisteredAppCustomer]
} {
        surplusCapacity ≥ 0 and all rac: RegisteredAppCustomer, t : Time |
        (rac in contains.t iff rac in locatedIn.contains.t)
        and #(contains.t) ≤ surplusCapacity
}


//-------------------TICKET SIGNATURES-----------------------------


abstract sig State {}
one sig Invalid, Valid, Expired extends State {}

abstract sig DigitalTicket {
        state: dynamic[State],
        ownedBy: one (AppCustomer + StoreManager),
```

```
        requestedByStoreManager: one Bool,
        building: one Building
} {
        (requestedByStoreManager in True iff ownedBy in storeManagers[building]) and
        (requestedByStoreManager in True implies one pt : PhysicalTicket | this = pt.
            ↪ associatedWith)

}

sig LineUpDigitalTicket extends DigitalTicket {

} {
        all t : Time | state.t = Invalid implies (one q: Queue | this in t.(q.contains).
            ↪ elems)
}

sig TimeSlot {
        start: Time,
        end: Time
} {
        start in prevs[end]
}

sig BookingDigitalTicket extends DigitalTicket {
        timeSlot: one TimeSlot,
        departments: set Department

}
{
        requestedByStoreManager in False and ownedBy in RegisteredAppCustomer and
        (#departments > 0 implies departments.locatedIn = building)
}


sig PhysicalTicket {
        ownedBy: one PhysicalCustomer,
        associatedWith: one LineUpDigitalTicket
} {
        associatedWith.requestedByStoreManager in True
}

sig Queue {
        contains: Time -> seq/Int -> lone LineUpDigitalTicket,

} {
        (all t: Time | not t.contains.hasDups and all ld: LineUpDigitalTicket |
        one b : Building | this = b.hasQueue and
        (ld in t.contains.elems implies ld.state.t = Invalid and ld.building = b))
        and
        (all t: Time | no idx: seq/Int | #(t.contains[idx]) = 0
        and (some idx': seq/Int | gt[idx', idx] and #(t.contains[idx']) > 0))
}


//----------------------FUNCTIONS-----------------------


fun storeManagers[b: Building]: StoreManager {
        worksIn.b
}

fun Building.customers[t: Time]: Customer {
        this.contains.t
}


fun AppCustomer.tickets[]: DigitalTicket {
        ownedBy.this
}

fun PhysicalCustomer.tickets[]: PhysicalTicket {
        ownedBy.this
```

```
}

fun AppCustomer.tickets[b: Building]: DigitalTicket {
        this.tickets[] & building.b
}

fun PhysicalCustomer.tickets[b: Building]: DigitalTicket {
        (this.tickets[]).associatedWith & ownedBy.(storeManagers[b])
}

fun Building.noDepCustomers[t: Time]: Customer {
        this.contains.t - (locatedIn.this).contains.t
}

fun TimeSlot.timeInterval[]: Time {
        {t: Time | gte[t, this.start] and lte[t, this.end]}
}

//Union of Valid Tickets and Tickets associated to Customers that entered. (not including
    ↪  department specific tickets)
fun numberOfValidOrInside[b: Building, t : Time]: Int {
        #({lt: LineUpDigitalTicket | lt.state.t = Valid and lt.building = b} +
        {lt : LineUpDigitalTicket | one ac : AppCustomer | lt in ac.tickets[b] and ac in
            ↪ b.noDepCustomers[t] }
                + {lt: LineUpDigitalTicket | one pc : PhysicalCustomer | lt in pc.tickets
                    ↪ [b] and pc in b.noDepCustomers[t]
                + {bdt: BookingDigitalTicket | bdt.state.t = Valid and not bdt.
                    ↪ isDepartmentSpecific}} )
}


//------------------ FACTS ----------------------------------------

fact everyBuildingHasOneSM {
        all b : Building | some sm : StoreManager | b = sm.worksIn
}

fact CustomerHasAtMostOneTicketForEachBuilding {
        all ac : AppCustomer | #ac.tickets[] = #(ac.tickets[].building)
}

fact NoCustomerInsideWhenClosed {
        all t : Time | all b : Building | b.closed.t in True implies #(b.contains.t) = 0
}


fact PhysicalTicketAssociatedToOnlyOneDT {
        all disj pt1, pt2 : PhysicalTicket | pt1.associatedWith ≠ pt2.associatedWith
}

//A Customer cannot be inside a Building without entering it.
fact BuildingContainsCustomersWhoEntered {
        (all t : Time, b: Building, ac: AppCustomer | (ac in b.contains.t iff (one t1 :
            ↪ Time | (lte[t1, t] and ac.enters[b, t1] and
        no t2: Time | gt[t2, t1] and lte[t2,t] and ac.exits[b, t2]))))
        and
        (all t : Time, b: Building, pc: PhysicalCustomer | (pc in b.contains.t iff (one
            ↪ t1 : Time | (lte[t1, t] and pc.enters[b, t1] and
        no t2: Time | gt[t2, t] and lte[t2,t] and pc.exits[b, t2]))))
}


fact CustomersExitsOnlyThroughExit {
        all t, t1: Time, ac: AppCustomer, b:Building | ((ac in b.customers[t] and gt[t1,t
            ↪ ] and
        not ac in b.customers[t1]) implies some t2: Time | lte[t2,t1] and ac.exits[b,t2])
}

fact CustomersExitsOnlyThroughExit {
        all t, t1: Time, pc: PhysicalCustomer, b:Building | ((pc in b.customers[t] and gt
            ↪ [t1,t] and
        not pc in b.customers[t1]) implies some t2: Time | lte[t2,t1] and pc.exits[b,t2])
```

```
}

//Describe when BookingDigitalTickets become Valid
fact BookingDigitalTicketValidity {
        all t: Time, bt : BookingDigitalTicket | bt.state.t = Valid iff t in (prev[bt.
            ↪ timeSlot.start] + (bt.timeSlot).timeInterval[])
}

//Describe how Tickets become Valid exiting the Queue
fact LineUpDigitalTicketValidity {
        all t: Time - time/first, b: Building | #(b.hasQueue.contains[prev[t]]) > 0 and
            ↪ sub[b.baseCapacity, numberOfValidOrInside[b,prev[t]]] > 0 implies
                b.hasQueue.contains[t] = b.hasQueue.contains[prev[t]].delete[0] else b.
                    ↪ hasQueue.contains[t] = b.hasQueue.contains[prev[t]]

}


fact CustomerIsInOneBuildingAtATime {
        no t : Time | some disj b1, b2 : Building | #(b1.contains.t & b2.contains.t) > 0
}

fact DigitalTicketStateMachine {
        all t: Time - time/first, dt : DigitalTicket |
        (dt.state.t = Expired implies (dt.state.(prev[t]) in Valid + dt.state.t)) and
                (dt.state.t = Valid implies (dt.state.(prev[t]) in Invalid + dt.state.t))
                    ↪  and
                (dt.state.t = Invalid implies dt.state.(prev[t]) = dt.state.t)
}


fact No2PhysicalCustomerWith2Tickets {
        no disj pt1, pt2 : PhysicalTicket, pc: PhysicalCustomer  |
        pt1.ownedBy = pc and pt2.ownedBy = pc
}


fact ValidTicketsDontExceedCapacity {
        all b: Building, t: Time | numberOfValidOrInside[b,t] < b.baseCapacity
}


//Describes the evolution of the dynamic model
fact Trace {
        (all t: Time - time/first | some b: Building |
                 (one ac : AppCustomer | ac.enters[b,t] or ac.exits[b,t]) or
                 (one pc : PhysicalCustomer | pc.enters[b,t] or pc.exits[b,t]) or
                 (b.hasQueue.contains[t] = b.hasQueue.contains[prev[t]].delete[0])
                 and #b.hasQueue.contains[prev[t]] > 0)

}


//------------------PREDICATES----------------------------------------


pred AppCustomer.enters[b: Building, t: Time] {
        --precondition
        not this.inBuilding[prev[t]]     --AppCustomer was not in a building before
            ↪ entering
        this.hasValidDigitalTicket[b, prev[t]]  --Before entering his ticket was valid

        --postcondition
        this in b.customers[t]                          --AppCustomer is in the Building

        let ticket = this.tickets[b] |
                ticket in BookingDigitalTicket
                implies (this.exits[b, next[ticket.timeSlot.end]] and
                        (all t' : (ticket.timeSlot).timeInterval[] | ticket.
                            ↪ isDepartmentSpecific
                        implies
```

```
                                (all d : ticket.departments |
                                this in d.contains.t' and
                                no d': locatedIn.b - ticket.departments | this in d'.
                                    ↪ contains.t')
                        else
                                this in b.noDepCustomers[t']))
                                    ↪
                else this in b.noDepCustomers[t]
}

pred AppCustomer.exits[b: Building, t: Time] {
        --precondition
        this.inBuilding[prev[t]]
        this in b.customers[prev[t]]

        --postcondition
        not this in b.customers[t]
        not this.inBuilding[t]
        this.tickets[b].state.t = Expired
}

pred PhysicalCustomer.exits[b: Building, t: Time] {
        --precondition
        this.inBuilding[prev[t]]
        this in b.customers[prev[t]]

        --postcondition
        not this in b.customers[t]
        not this.inBuilding[t]
        this.tickets[b].state.t = Expired
}



pred BookingDigitalTicket.isDepartmentSpecific {
        #(this.departments) > 0                         --Departments are specified
}

pred PhysicalCustomer.enters[b: Building, t: Time] {
        --precondition
        not this.inBuilding[prev[t]]
        this.hasValidPhysicalTicket[b, prev[t]]


        --postcondition
        this in b.customers[t] and this in b.noDepCustomers[t]
}


pred Customer.inBuilding[t: Time] {
        this in Building.customers[t]
}

pred AppCustomer.hasValidDigitalTicket[b: Building, t:Time] {
        #(this.tickets[b]) > 0
        this.tickets[b].state.t = Valid
}

pred PhysicalCustomer.hasValidPhysicalTicket[b: Building, t: Time] {
        one dt : this.tickets[b] | dt.state.t = Valid
}

assert RAC_doesNotVisitOtherDepartments {
        no t: Time, rac : RegisteredAppCustomer, d : Department |
        rac in (d.locatedIn).contains.t
        and rac not in d.contains.t and rac in d.contains.(next[t])
}

assert NotEnters2Times {
        no t1, t2:Time, ac: AppCustomer, b: Building | gte [t2,t1] and ac in b.contains.
            ↪ t1 and not ac in b.contains.t2
        and (ac.tickets[b]).state.t2 = Valid
```

```
}

assert NoDigitalTicketForSameBuilginAndPerson {
        (no b : Building, ac : AppCustomer | #(ac.tickets[b]) > 1) and
        (no b : Building, pc : PhysicalCustomer | #(pc.tickets[b]) > 1)

}

assert NoInvalidTicketOutsideQueue {
        no t: Time, ld: LineUpDigitalTicket, b : Building | ld.state.t = Invalid and ld.
            ↪ building = b and ld not in t.((b.hasQueue).contains).elems
}

assert NoCustomersInsideWhitoutEntering {
        (no t: Time, b: Building, dt: DigitalTicket, pc: PhysicalCustomer | dt in pc.
            ↪ tickets[b] and
                some t1: prevs[t] | pc not in b.contains.t1 and dt.state.t1 = Expired and
                    ↪  pc in b.contains.t)


        and

        (no t: Time, b: Building, dt: DigitalTicket, ac: AppCustomer | dt in ac.tickets[b
            ↪ ] and
                some t1: prevs[t] | ac not in b.contains.t1 and dt.state.t1 = Expired and
                    ↪  ac in b.contains.t)

}

run {} for 4 but exactly 1 PhysicalCustomer, exactly 2 RegisteredAppCustomer, exactly 1
    ↪ Building
```

5 Effort Spent

Task	*Name*	*Time spent*
Goals and Requirements	Francesco Attorre	10 h*
Goals and Requirements	Thomas Jean Bernard Bonenfant	10 h*
Goals and Requirements	Veronica Cardigliano	10 h*
Use cases	Francesco Attorre	13 h
Use cases	Veronica Cardigliano	2 h
Diagrams	Veronica Cardigliano	12 h
Alloy	Francesco Attorre	2 h
Alloy	Thomas Jean Bernard Bonenfant	18 h
Alloy	Veronica Cardigliano	3 h
Scenarios	Veronica Cardigliano	0.5 h
Scenarios	Thoma Jean Bernard Bonenfant	0.5 h
LaTeX (doc. composition)	Veronica Cardigliano	2 h

* General discussion about the project and specifications, organization of the project and division of tasks, collective discussion and review of requirements (functional and non functional), requirements mapping

6 Used Tools

Tools used to create this RASD document:

- StarUML: for all the UML diagrams
- LaTeX: to create the pdf
- Alloy Analyzer 5.1.0: to model the system in Alloy
- GitHub: for the repo of the project
- GoogleDoc: for a shared editing of the document