CLup project
Luca De Martini, Alessandro Duico

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Luca De Martini, Alessandro Duico
Version:	1.0
Date:	November-2020
Download page:	https://github.com/luca-de-martini/DeMartiniDuico-sw2
Copyright:	Copyright © 2020, Luca De Martini, Alessandro Duico - All rights reserved

Contents

Table of Contents . . . 3

List of Figures . . . 5

List of Tables . . . 5

1 Introduction . . . 6
- 1.1 Purpose . . . 6
- 1.2 Scope . . . 6
  - 1.2.1 World phenomena . . . 7
  - 1.2.2 Machine phenomena . . . 7
  - 1.2.3 Shared phenomena . . . 7
  - 1.2.4 Goals . . . 8
- 1.3 Definitions, Acronyms, Abbreviations . . . 8
  - 1.3.1 Definitions . . . 8
  - 1.3.2 Acronyms . . . 9
  - 1.3.3 Abbreviations . . . 9
- 1.4 Revision history . . . 9
- 1.5 Reference documents . . . 9
- 1.6 Document structure . . . 9

2 Overall Description . . . 10
- 2.1 Product perspective . . . 10
  - 2.1.1 Class diagrams . . . 10
  - 2.1.2 State machine diagrams . . . 11
- 2.2 Product functions . . . 13
- 2.3 Customer characteristics . . . 14
  - 2.3.1 Actors . . . 14
- 2.4 Assumptions, dependencies and constraints . . . 15

3 Specific Requirements . . . 16
- 3.1 External interface Requirements . . . 16
  - 3.1.1 Customer interfaces . . . 16
  - 3.1.2 Hardware interfaces . . . 20
  - 3.1.3 Software interfaces . . . 20
  - 3.1.4 Communication interfaces . . . 21
- 3.2 Functional requirements . . . 21
  - 3.2.1 Goal-Requirement mapping . . . 21
  - 3.2.2 Scenarios . . . 22
  - 3.2.3 Use cases . . . 23
  - 3.2.4 Sequence diagrams . . . 33
- 3.3 Performance Requirements . . . 41
- 3.4 Design constraints . . . 41
  - 3.4.1 Standards compliance . . . 41
  - 3.4.2 Hardware limitations . . . 41
- 3.5 Software System Attributes . . . 41
  - 3.5.1 Reliability . . . 41
  - 3.5.2 Availability . . . 41
  - 3.5.3 Security . . . 42

3.5.4 Maintainability . . . 42
3.5.5 Portability . . . 42

4 Formal Analysis Using Alloy . . . 43
4.1 Introduction . . . 43
4.2 Alloy code . . . 43
4.2.1 Model description . . . 43
4.2.2 Dynamic properties . . . 44
4.2.3 Assertions . . . 45
4.2.4 Execution results . . . 47

5 Effort Spent . . . 53

References . . . 54

List of Figures

1 World Machine diagram . . . 7
2 High level class diagram . . . 10
3 Statechart of the lifetime of a Booking . . . 11
4 Statechart of the lifetime of a Ticket . . . 12
5 Statechart of the Shop creation and configuration . . . 13
6 Registration interface . . . 16
8 Ticket interface . . . 18
9 Booking interface . . . 19
11 Use Case Diagram: Customer . . . 23
12 Use Case Diagram: Authority . . . 24
13 Sequence Diagram for the registration of a Customer . . . 33
14 Sequence Diagram for a successful login attempt . . . 34
15 Sequence Diagram for a failed login attempt . . . 34
16 Sequence Diagram for the creation of a Booking . . . 35
17 Sequence Diagram for the creation of a Ticket . . . 36
18 Sequence Diagram for the creation of a Shop . . . 37
19 Sequence Diagram for the scan of a valid Token . . . 38
20 Sequence Diagram for the scan of an invalid Token . . . 39
21 Sequence Diagram for the generation of a Substitute Token . . . 40
22 All assertions pass the test on a restricted domain. . . . 47
23 Run for Time 0. This is the initial state. . . . 49
24 Run for Time 1. *Customer1* has used his Ticket to enter *Shop0*. . . . 49
25 Run for Time 2. *Customer3* and *4* enter the same shop with a Ticket and a Booking respectively. . . . 49
26 Run for Time 3. *Customer1* correctly exits *Shop0*, while *Customer0* makes use of a Booking for *Shop1* . . . 50
27 Run for Time 4. *Customer4* exits as well. . . . 50
28 The same instance at Time 0, with WaitingListNode elements shown . . . 51
29 The evolution of the instance at Time 1 . . . 52

List of Tables

1 Introduction

1.1 Purpose

During the COVID-19 pandemic in order to reduce the of transmission of the virus governments have introduced various measures aimed to reduce to a minimum close contacts between people, these measures include social distancing and lockdowns.

During lockdowns people can leave their habitations only for essential needs such as grocery shopping, which makes supermarkets a gathering spot where the virus could spread and consitutes a potential danger for the people visiting.

For this reason the access to essential activities should be limited to lower the density of people inside the activities and allow keeping distances effectively and mitigate the risk.

However this is not of trivial execution, because in order to serve the same amount of people, with reduced maximum capacity, the clients need to access the activity distributed over a longer timespan than regular operation. This means that if too many people want to access the shop in a given moment, some people will be left out and this will result in a line of people waiting to enter the shop.

This is a hazard in and of itself since people outside will stay for a prolonged amount of time in close proximity with each other while waiting for their turn.

The problem of forming lines can be mitigated by handing a number to each client and having people enter in order of arrival. This approach, while better than queueing, still requires all clients to go to the shop and take a ticket. Even though they would not need to stay in a queue, they would need to wait close to the activity nevertheless.

This application aims to provide a digital alternative to the "handing numbers" approach and improve it by allowing people to take their number online, this way people only need to go to the shop when it is their turn to enter and they will not need to hang around the building.

The product will also allow shop managers to effectively monitor the entrances by scanning a QR code associated with the "number" to ensure the safety limits are being followed.

Additionally to the lining up mechanism the application will allow customers to book a visit in the future, which will improve the distribution of customers during the day and the week, since they will be able to choose a less crowded time slot.

CLup will also allow customers to choose the approximate time of their visit and the category of items they wish to buy, which allows making more accurate predictions of the waiting times and, by knowing the areas that they will visit, to better utilize the space in the building, while respecting health and safety measures.

1.2 Scope

Using the "The World and The Machine" model by M. Jackson we can give an overall description of the events involved in the project realization, those which cannot be observed by the system: "The World", those strictly related to the system: "The Machine", and those in common between the two.

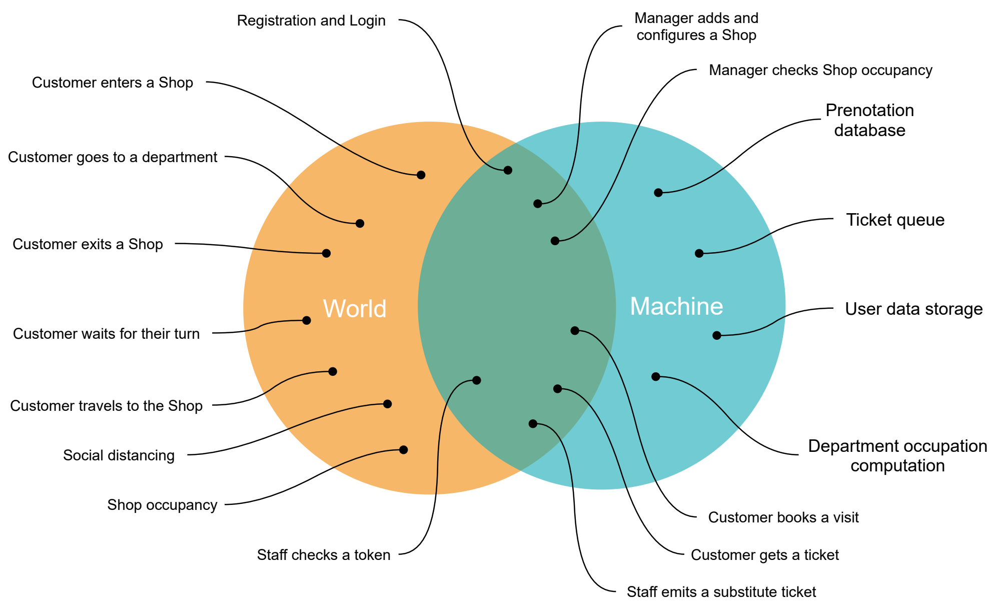


Figure 1: World Machine diagram

1.2.1 World phenomena

- Customer enters a Shop
- Customer goes to a department
- Customer exits a Shop
- Customer waits for their turn
- Customer travels to the shop
- Social distancing
- Shop occuapancy

1.2.2 Machine phenomena

- Customer data storage
- Ticket queue
- Booking data
- Department occupancy computation

1.2.3 Shared phenomena

Controlled by the World

- Registration and login
- Manager adds a Shop
- Manager configures a Shop
- Customer books a visit
- Customer gets a ticket
- Staff checks a token
- Staff emits a substitute ticket
- Manager checks Shop occupancy

Controlled by the Machine

- System sends a reminder to the Customer

1.2.4 Goals

G1 Customers shall be able to acquire a ticket which grants them access to the Shop

G2 Customers shall be able to book a future visit to the Shop

G3 Customers shall be able to enter the Shop at the earliest occasion without waiting in line

G4 Customers shall be reminded about considering the time needed to travel to the Shop

G5 Third Parties shall better exploit the departments of the Shop without breaking social distancing measures

G6 Third parties shall be able to monitor the number of visits to the Shop, to ensure compliance with the laws

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Third party company A company that wants to use our services

Ticket A digital proof of presence in the waiting list

Booking A digital proof of reservation for a visit to the Shop in a specific time slot

Token Either a ticket or a booking, it's associated with a code which can also be represented as a QR code

Staff Company personnel which is responsible of checking entrances

Shop Point of sale of the Third party company

1.3.2 Acronyms

RASD	Requirement Analysis and Specification Document
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
UML	Unified Modeling Language

1.3.3 Abbreviations

[Gn]	n-th goal.
[Dn]	n-th domain assumption.
[Rn]	n-th functional requirement.

1.4 Revision history

v. 1.0 - 22/12/2020 Initial release

1.5 Reference documents

BEEP channel - Mandatory Project Assignment

The world & the machine - M. Jackson, P. Zave

1.6 Document structure

This document is structured in four main chapters, that are as follows:

Section 1 The first chapter serves as an introduction and an overview to the project, describing the main reasons for its development, its goals together with a brief informal description of it.

Section 2 The second chapter serves as a more formal description of the project: it includes class diagrams, state machine diagrams, and it gives details on the shared phenomena and domain models. Class diagrams give a big picture description on how the system should be structured, while state machine diagrams focus on the more relevant entities of the model. Here are also presented all the requirements and domain assumptions the system in project must fulfill and take into considerations, in order to achieve the goals; they are presented each one after the goal it is relevant to.

Section 3 The third chapter presents the specific requirements: use cases and the design constraints the system must satisfy. The use cases are described using natural language, while the constraints are pictured with sequence/activity diagrams. A mockup is also shown as a general idea of how the end product should be, in terms of design and functionalities offered to the end customer.

Section 4 The fourth and last chapter is a formal analysis of the model, made through the use of the open source Alloy language and analyzer, including a graphic representation of it obtained from Alloy Tool.

2 Overall Description

2.1 Product perspective

2.1.1 Class diagrams

The following pages present the UML class diagrams which represent the elements that are relevant to the domain of the application. Only the relevant parts of the system are present for the sake of clarity.

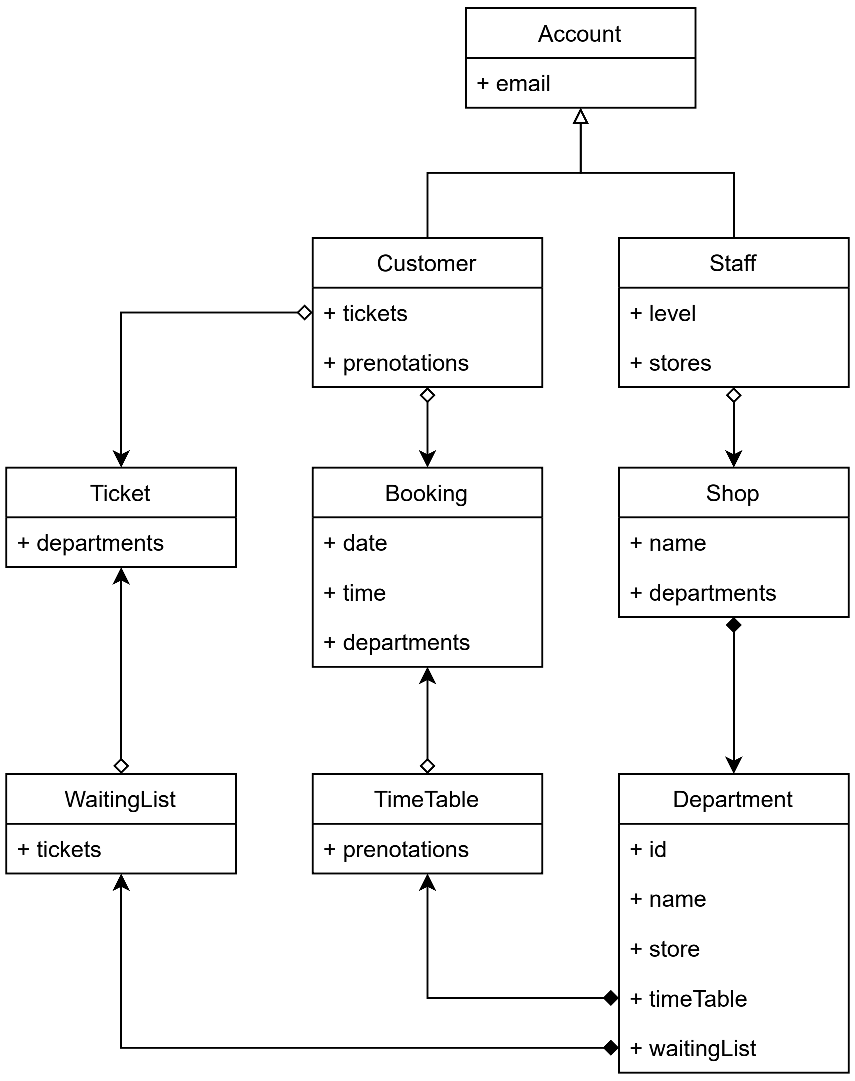


Figure 2: High level class diagram

2.1.2 State machine diagrams

The following diagrams represent a high level description of the evolution of the states of the system during its processes.

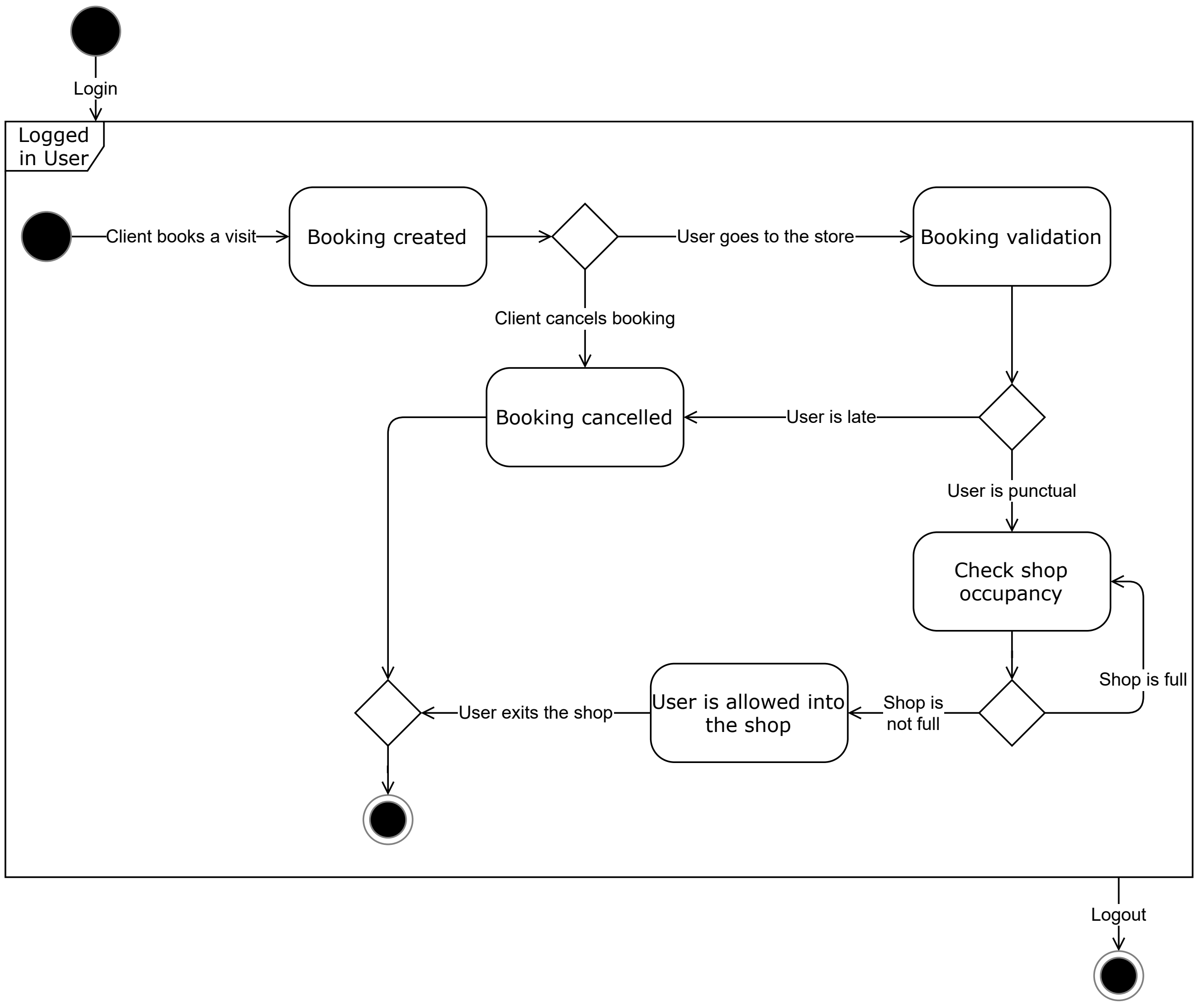


Figure 3: Statechart of the lifetime of a Booking

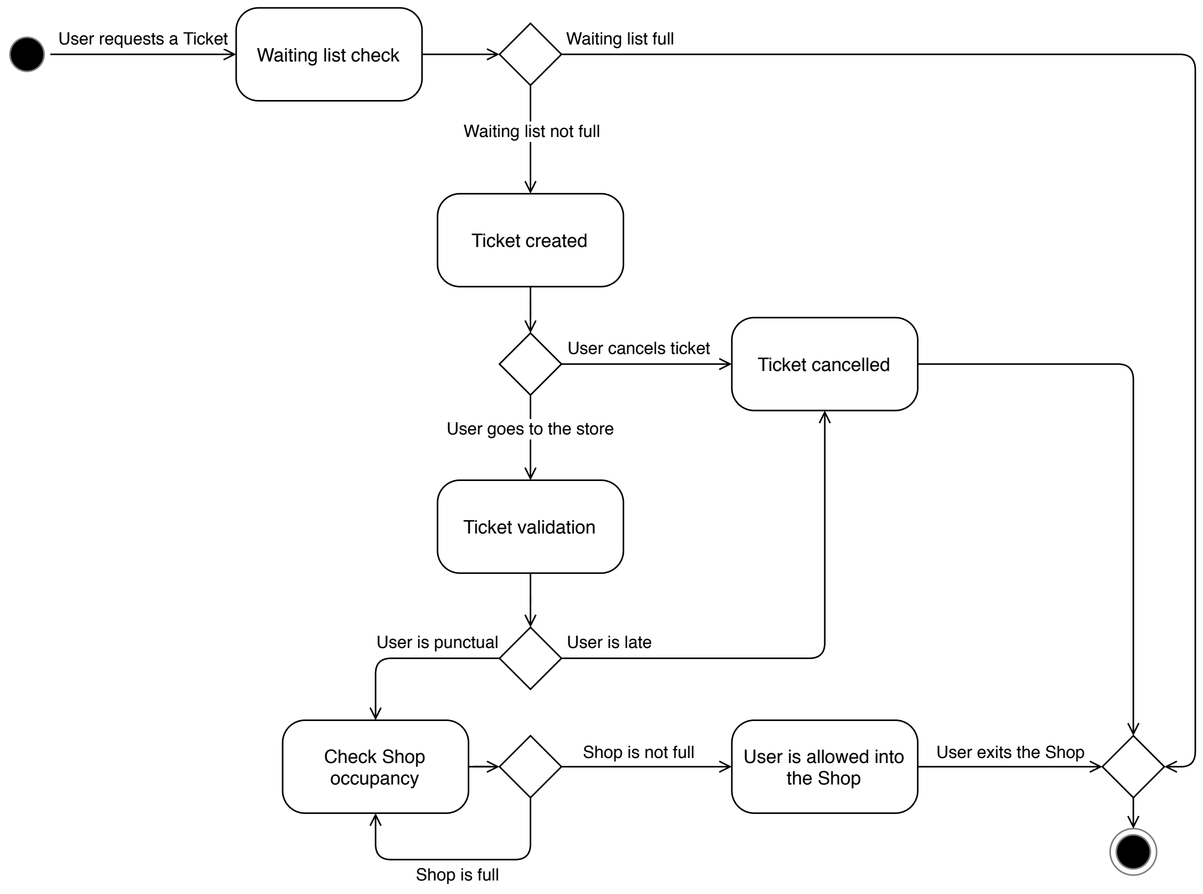


Figure 4: Statechart of the lifetime of a Ticket

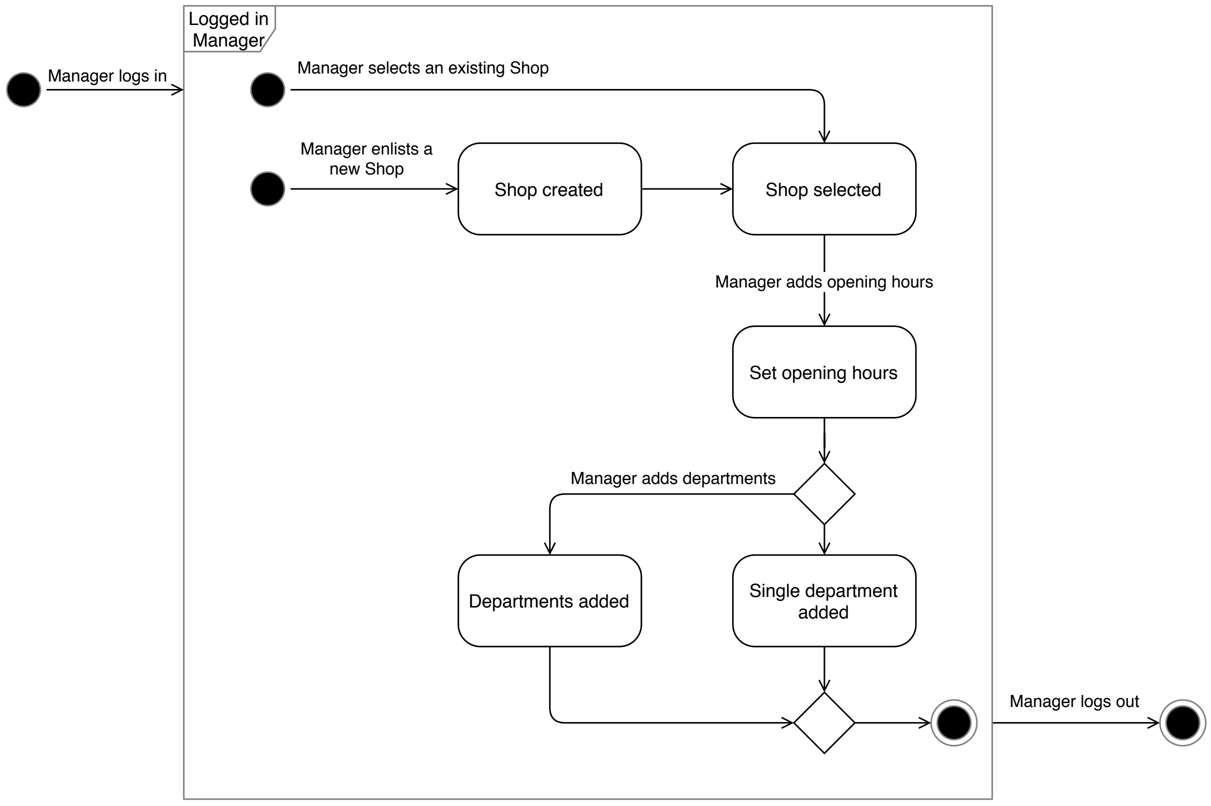


Figure 5: Statechart of the Shop creation and configuration

2.2 Product functions

CLup offers the Customers two ways of accessing a Third Party Shop: they can request a Ticket to for an immediate visit, or make a Booking for a future visit. The proofs of reservation (Ticket or Booking) must be validated before entering the Shop. An alternative must be guaranteed on-location for customers who cannot access the service. The configuration of Shop is handled by the Third Party, which is able to set the opening times and the departments. When *CLup* is operative, the Third Party can monitor the occupancy of the Shop, detailed into the various departments.
Here is an overview of *CLup* functions, subdivided into three sets, related to the Customers, the Third Party clerks and the Third Party managers:

Basic Customer functions These functions are available to all Customers:

- **Registration and Login**
  Customers should be able to create a personal account for the App. By logging in with their account, Customers can access the other functions. For ease of use the app should allow users to stay logged in, this way the service can be accessed without inserting the login credentials every time.
- **Obtaining a Ticket**
  CLup has a waiting list of the customers who are currently waiting to enter the Shop. Registered Customers can request to join the list via the App, if the queue is not full for the day they will be given a ticket. The ticket is associated with a *unique* Token that can be shown to the staff. Customers are granted access to the shop by showing the token to the staff when it's their turn. Notifications can be sent to the Customers' devices to remind them when it is their turn to enter the Store.

- **Booking a visit**
  CLup keeps a table of booked visits to the Shop. Registered Customers can book a visit for a specified time and date via the App if the time slot is not full. A successful booking is associated with a *unique* Token that can be shown to the staff. Customers are granted access to the shop by showing the token to the staff in the time slot they bookid. Notifications can be sent to the Customers' devices to remind them about upcoming bookings.
- **Specifying item categories**
  When booking a visit or getting a ticket, Customers can specify categories of products they intend to buy. This data can be used to better optimize the occupancy of the departments of the Shop.

Third Parties base functions These functions should be accessible to all the employees of the Third Party, since they are required for the proper working of the system, at an operational level:

- **Login**
  Third Parties clerks and managers shall be provided with valid credentials to login into the system.
- **Substitute ticket distribution**
  Third Parties shall be able to generate a ticket for Customers who do not have the required technologies, which allows them to join the waiting list. Booking is only allowed via the App, since it requires more detailed information;
- **Ticket and Booking validation**
  CLup allows Third Parties to check Customers' tickets against those stored in the system. This is fundamental to deny forged tickets.

Third Parties administration functions These functions must only be accessible to the managers of the Third Party, since they control administrative aspects of the service, which handle sensible or critical data:

- **Shop creation and configuration**
  Third Parties shall be able to enlist their store in the *CLup* system, specifiying opening hours and departments with the respective capacities. This is only possible with a manager account, while the clerks should only be allowed to;
- **Shop occupancy checking**
  CLup allows Third Parties to monitor entrances, to ensure all of the imposed restrictions are being followed. These data are also a source of useful insights in a business perspective.

2.3 Customer characteristics

CLup is meant to be accessible for customers of all demographics.

2.3.1 Actors

- **Customer**: people who need to go shopping

- **Shop manager**: someone who owns a retail commercial activity which needs to operate at limited capacity
- **Staff**: people working at a Shop

2.4 Assumptions, dependencies and constraints

D1 Each customer creates only one account

D2 The information provided by the manager is correct

D3 The maximum capacity provided by the manager is compliant to active health and safety measures

D4 Most of the customers that take a ticket or book a visit will actually visit the shop

D5 The staff will validate tickets and bookings on entry and exit

D6 Transgressors will be handled by the staff

D7 Clients will be instructed to use the applicative if possible

D8 Shops are added by the legitimate owner

D9 Only existing shops are added to the system

3 Specific Requirements

3.1 External interface Requirements

The *CLup* frontend is a web application that can be accessed from web browsers, both from mobile and desktop devices. The following section will give a comprehensive description in terms of hardware, software and communication interfaces.

3.1.1 Customer interfaces

Login and Registration
When first opening the application, Customers are presented with a registration page. They are asked to provide only the essential information: an *e-mail address* and a *password*. The interface also shows a small button to switch to the login page, in case the customer has already signed in on another device or the session has expired. The page is identical to the one for the Registration, except for the Login button. In this case, the small button gives the Customers the ability to switch back to the registration page, which is needed if multiple people share the same device.

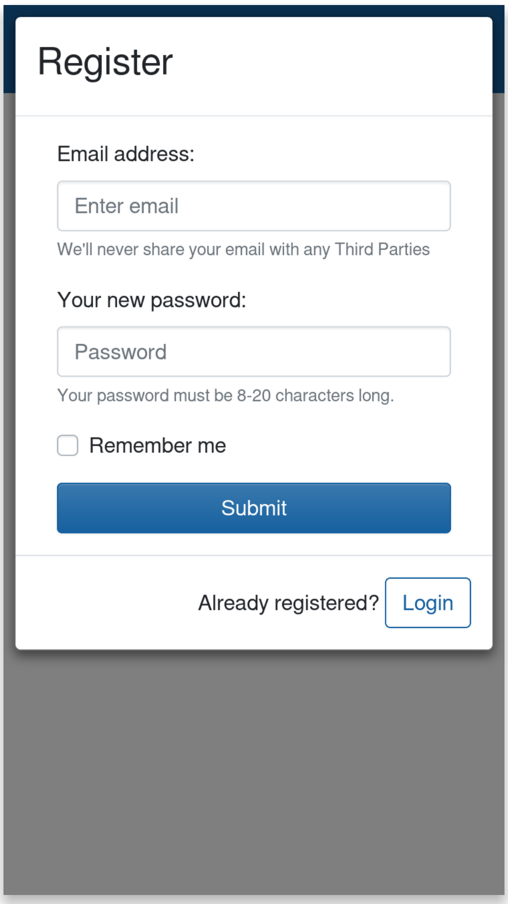
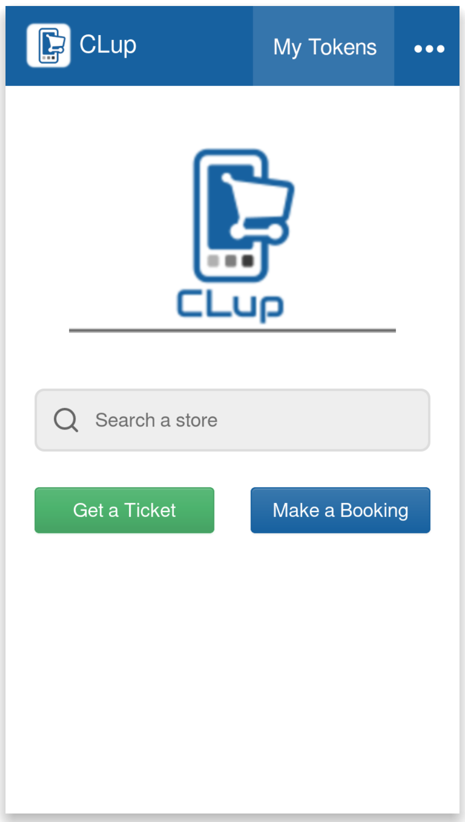


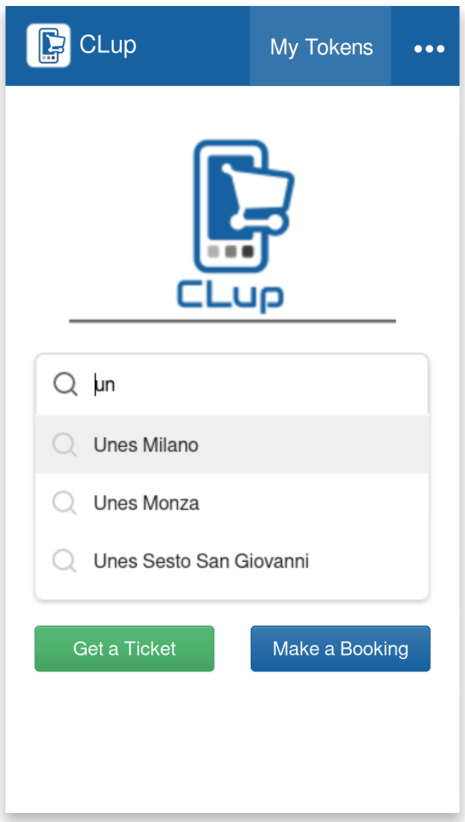
Figure 6: Registration interface

Shop selection page

After login/registration, the app displays a Shop search bar that dynamically updates its results while the Customers are typing. After selecting a Shop from the results, the two buttons "Get a Ticket" and "Make a Booking" are enabled, which are linked to the Ticket form and Booking form, respectively.



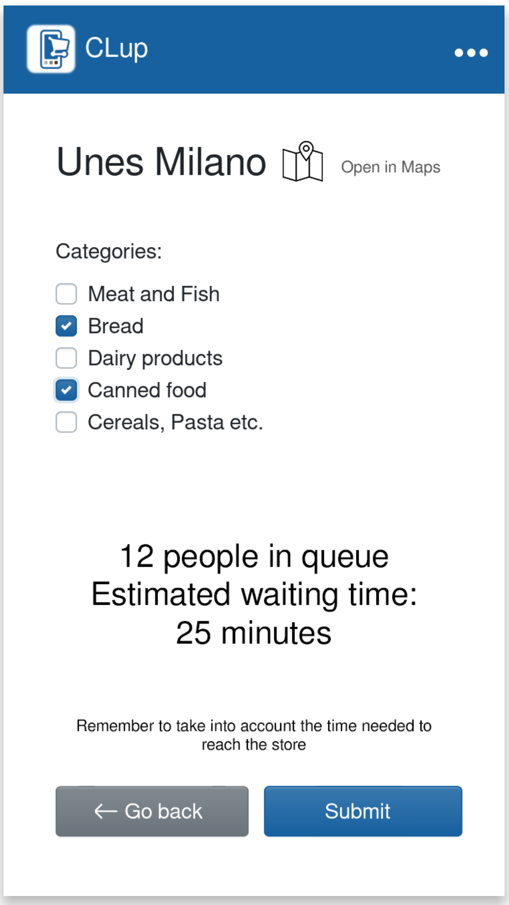
(a) Shop selection interface



(b) Dynamic search in the shop selection interface

Ticket form

This page prompts Customers for an optional choice of categories they want to buy. Customers can confirm the Ticket request by pressing the *"Submit"* button. In both the Ticket form and the Booking form, Customers are always given option to reset the app back to the Shop selection page.



Categories:

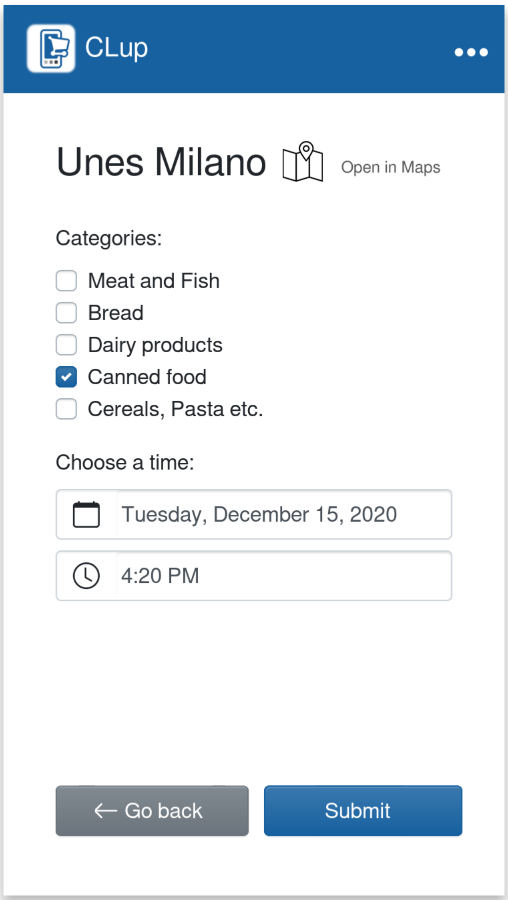
- [ ] Meat and Fish
- [✓] Bread
- [ ] Dairy products
- [✓] Canned food
- [ ] Cereals, Pasta etc.

12 people in queue
Estimated waiting time:
25 minutes

Remember to take into account the time needed to reach the store

Figure 8: Ticket interface

Booking form
This page is identical to the previous, except for the added time selection. A table with days as columns and hours as rows shows the time slots with lower occupancy. When Customers select the preferred day, the visualization updates with specific data for the day, giving an in-depth visualization of restricted to the opening hours. Given this, Customers can easily choose a time for their visit. As in the Ticket form, Customers can confirm the Booking request by pressing the "Submit" button.


CLup

Unes Milano

Open in Maps

Categories:

- [ ] Meat and Fish
- [ ] Bread
- [ ] Dairy products
- [✓] Canned food
- [ ] Cereals, Pasta etc.

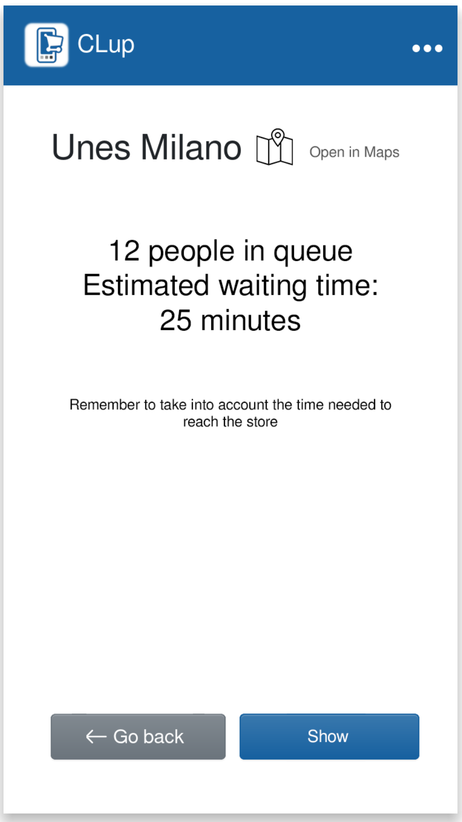
Choose a time:

Tuesday, December 15, 2020

4:20 PM

Figure 9: Booking interface

Token display
After generating a Ticket or a Booking, and until its expiration, the app will display the token as a fullscreen QR code right after opening, or after Login (if the login timeout has passed). This makes it very quick to have the Token scanned by the clerks outside of a shop.


CLup

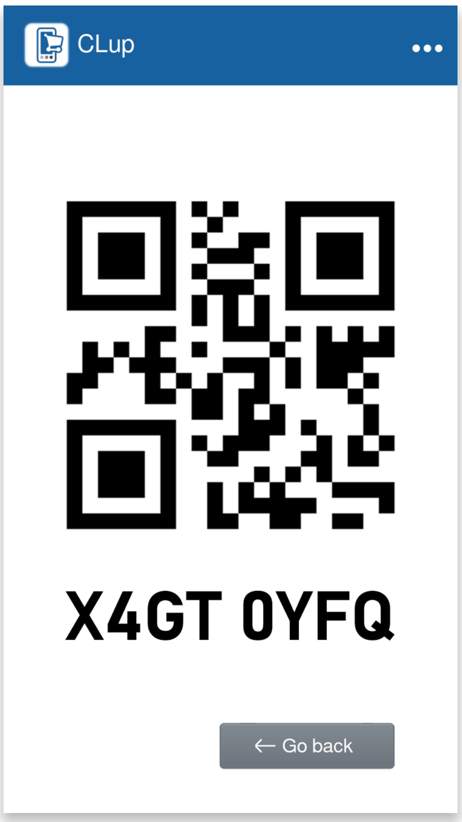
Unes Milano

Open in Maps

12 people in queue
Estimated waiting time:
25 minutes

Remember to take into account the time needed to reach the store

(a) My Tokens interface


CLup

X4GT 0YFQ

(b) Show token interface

3.1.2 Hardware interfaces

The *CLup* client shall be available for devices capable of rendering a web page with javascript and internet access, this includes smartphones, tablets and personal computers.

Although not strictly required, the devices used by the staff should have a camera to scan QR codes

3.1.3 Software interfaces

- **Web browser**: the applicative requires a web browser capable of rendering HTML5 web pages with Javascript
- **Maps service**: although not required for the core functionality, having access to a maps service on the device will improve customer experience

3.1.4 Communication interfaces

- *Customer*: requires a internet connection in order to get a token. Access to the store can be done offline if necessary.
- *Staff*: requires a internet connection in order to communicate with the server and validate tokens or emit substitute tickets.

The system shall use the HTTPS protocol to provide secure communication from the client to the server.

3.2 Functional requirements

3.2.1 Goal-Requirement mapping

G1 Customers shall be able to acquire a ticket which grants them access to the Shop

- **R1** The system shall keep track of the list of Customers waiting to visit each Shop
- **R2** The system shall allow customers to request the right to visit a shop as soon as possible
- **R3** The system shall give Customers a Token associated with their position in the waiting line
- **R4** The Staff shall be able to scan Customer generated Tokens using a camera
- **R5** The Staff shall be able to scan Customer generated Tokens using a textual input
- **R6** Given a Token, the Staff applicative shall be able to verify its validity
- **R7** Given a Token, the Staff applicative shall be able to verify the respective position in the waiting line
- **R8** Given a Token, the Staff applicative shall be able to mark it as used and update the list of Customers currently inside the Shop

G2 Customers shall be able to book a future visit to the Shop

- **R9** The system shall keep track of the Customers that want to visit a Shop in the future
- **R10** The system shall allow customers to choose a time in the future in which they wish ti visit a Shop
- **R11** The system shall give Customers a Token associated with their booking
- **R4** The Staff shall be able to scan Customer generated Tokens using a camera
- **R5** The Staff shall be able to scan Customer generated Tokens using a textual input
- **R6** Given a Token, the Staff applicative shall be able to verify its validity
- **R7** Given a Token, the Staff applicative shall be able to verify the respective position in the waiting line
- **R8** Given a Token, the Staff applicative shall be able to mark it as used and update the list of Customers currently inside the Shop

G3 Customers shall be able to enter the Shop at the earliest occasion without waiting in line

- **R1** The system shall keep track of the list of Customers waiting to visit each Shop
- **R9** The system shall keep track of the Customers that want to visit a Shop in the future
- **R12** The system shall ask Customers to specify the approximate duration of their visit

- **R13** The system shall automatically infer an estimate visit duration for returning customers
- **R14** The system shall give Customers an estimate of the waiting time remaining before it's their turn
- **R15** The system shall notify Customers when their turn is about to come

G4 Customers shall be reminded about considering the time needed to travel to the Shop

- **R14** The system shall give Customers an estimate of the waiting time remaining before it's their turn
- **R16** The system shall be able to connect to a Maps service to show information about travel time

G5 Third Parties shall better exploit the departments of the Shop without breaking social distancing measures

- **R17** Customers shall be able to specify the categories of items they intend to buy
- **R18** The system shall keep track of the number of customers visiting the Shop on a Department basis

G6 Third parties shall be able to monitor the number of visits to the Shop, to ensure compliance with the laws

- **R17** Customers shall be able to specify the categories of items they intend to buy
- **R18** The system shall keep track of the number of customers visiting the Shop on a Department basis

3.2.2 Scenarios

Scenario 1
Maria is gathering ingredients to bake a cake, but she realizes she doesn't have any milk. Maria lives near to a grocery shop so she opens the web app, she writes the shop name in the search bar and she is presented with the most relevant matches. She clicks on the shop and then she chooses *"Get a Ticket"*. Now she checks the dairy section and confirms, receiving her ticket. CLup shows that Maria's turn will be in about 10 minutes. Five minutes later she checks back on the app and the approximate time is now 5 minutes, so she leaves and she goes to the shop. At the shop Giovanni, a memeber of the staff, is at the entrance waiting for customers. Maria is welcomed by Giovanni, she opens the application and she shows him the ticket, Giovanni scans the ticket with his phone, Maria is just in time, so she can enter the shop and do her groceries. At the exit she is invited by Mario, another member of the staff to scan her ticket, she shows the ticket one last time, then she leaves the shop and returns home ready to bake a cake.

Scenario 2
Marco has a busy schedule for the week and he needs to go and get a suit for an important meeting. He looks at his schedule and finds a spot on thursday, so he opens the web app, he writes the shop name in the search bar and she is presented with the most relevant matches. He clicks on the shop and then he chooses *"Make a Booking"*. Then he checks the clothing section and clicks on thursday on the calendar luckily there is a free time slot. So he inputs the time and confirms, the app responds with the token for the booking. On thursday Marco goes to the shop in time for his booking, he is welcomed by Francesca, a staff member, and shows her the QR code associated with his booking, she scans it using her phone and Marco is allowed in. At the exit he is invited to scan his booking one last time before leaving.

Scenario 3
Attilio was never a fan of technology and he does not have a smartphone, unfortuanately his grandchild

Matteo is not around to get a ticket for him, so he picks up his bicycle and goes to the grocery shop. At the entrance to the shop Francesco, a staff member, informs Attilio about the new procedure to enter the shop, since Attilio does not have a smartphone Fancesco uses his device to create a substitute ticket and prints it. Francesco hands Attilio the ticket, then he tells him that there are already 5 people waiting, so he will have to wait for about 10 minutes. Ten minutes later Attilio goes back to the entrance where Francesco scans the ticket letting Attilio go in. At the exit he is invited to scan the ticket one last time before heading home.

3.2.3 Use cases

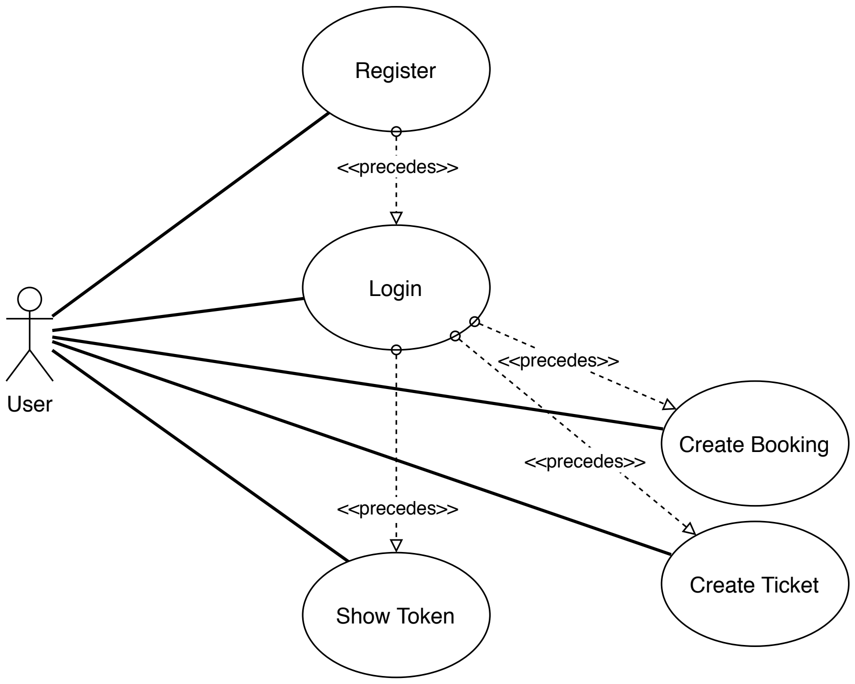


Figure 11: Use Case Diagram: Customer

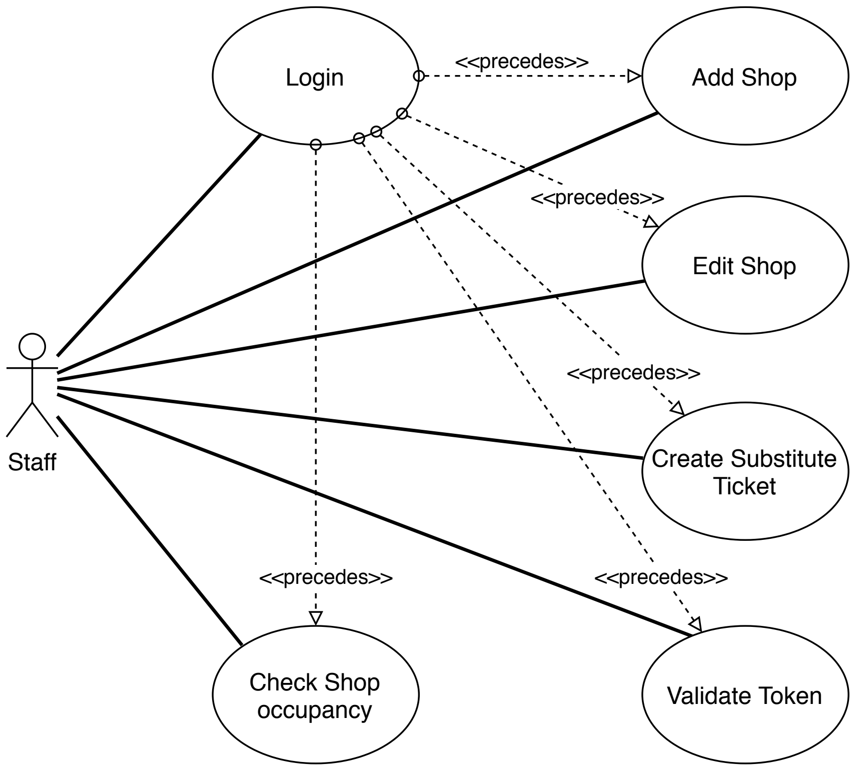


Figure 12: Use Case Diagram: Authority

Name	Customer creates an account
Actor	Customer
Entry condition	• The Customer has not created an account yet
Event flow	1. The Customer opens the web app 2. The app asks for login or registration 3. The Customer clicks on *"Register"* 4. The Customer inserts their email and password 5. The Customer clicks on *"Submit"* 6. The app prompts the Customer to check their email for a confirmation link 7. The Customer clicks on the confirmation link
Exit condition	• Customer account is added to the database • The Customer can login
Exceptions	• The Customer already exists • The Customer does not click confirmation link

Name	Customer logs in
Actor	Registered Customer
Entry condition	• The Customer has an account
Event flow	1. The Customer opens the web app 2. The app asks for login or registration 3. The Customer inserts their email and password 4. The Customer clicks on *"Submit"* 5. The Customer receives confirmation from the server 6. The Customer is brought to the home page
Exit condition	• The Customer is logged in • The Customer can use other features
Exceptions	• Wrong credentials

Name	Customer creates a booking
Actor	Customer
Entry condition	• The Customer has logged in
Event flow	1. The Customer types all or part of the shop name in the search bar 2. Results are dynamically updated 3. The Customer selects one of the results 4. The Customer clicks on *"Make a booking"* 5. The Customer is brought to the *Booking form* 6. The Customer chooses a day from the calendar 7. The Customer inputs the desired time of visit 8. The Customer can choose the departments it will visit 9. The Customer clicks submit 10. The Customer receives confirmation from the server and is shown the new booking
Exit condition	• Booking is added to the database • The Customer can now find the new token in his tokens
Exceptions	• Shop does not exist • Shop is already completely booked • The Customer tries to book an already full time slot

Name	Customer creates a ticket
Actor	Customer
Entry condition	• The Customer has logged in
Event flow	1. The Customer types all or part of the shop name in the search bar 2. Results are dynamically updated 3. The Customer selects one of the results 4. The Customer clicks on *"Get a Ticket"* 5. The Customer is brought to the *Get ticket form* 6. The Customer clicks submit 7. The Customer receives confirmation from the server and is shown the new ticket
Exit condition	• Ticket is added to the database • The Customer can now find the new token in his tokens
Exceptions	• Shop does not exist • Shop is closed for the day • Shop queue is full

Name	Customer shows a token
Actor	Customer
Entry condition	• The Customer has logged in • The Customer has created at least one token
Event flow	1. The Customer clicks on *"My Tokens"* 2. The Customer selects one of the tokens in the list 3. A QR code and associated text code are shown, either of which can be scanned with the staff applicative

Name	Manager adds a shop
Actor	Staff
Entry condition	• The Manager is logged in with an administrative account
Event flow	1. The Manager opens the "Manage Shops" section 2. The Manager clicks on add a Shop 3. The Manager inserts the Shop name and location 4. The Manager adds departments to the Shop and specifies the max capacity of each 5. The Manager specifies the opening hours 6. The Manager clicks on submit 7. The Manager receives confirmation from the server
Exit condition	• The new Shop is added to the database • Customers can find the Shop in hteir search results
Exceptions	• The Shop already exists

Name	Staff scans a valid token
Actor	Staff
Entry condition	• The Staff member is Logged in • The Staff member selected one of the shops they have access to • A Customer shows a token that is valid at the current time
Event flow	1. The Staff member opens the scan token section 2. The Staff member uses the camera of the device to scan the token from an Customer or inputs the code as text 3. The token is sent to the server for validation 4. The Staff member receives positive confirmation from the server 5. The Customer is allowed in
Exit condition	• The token is marked as used • The store occupancy statistics are updated
Exceptions	• The token is for another shop

Name	Staff scans an invalid token
Actor	Staff
Entry condition	• The Staff member is Logged in • The Staff member selected one of the shops they have access to • A Customer shows a token which is either expired or not yet valid
Event flow	1. The Staff member opens the scan token section 2. The Staff member uses the camera of the device to scan the token from an Customer or inputs the code as text 3. The token is sent to the server for validation 4. The Staff member receives a negative response from the server 5. The Customer informed that either their ticket is expired or that it's too early 6. The Customer is not allowed in
Exit condition	• The token is marked as used • The store occupancy statistics are updated
Exceptions	• The token is for another shop

Name	Staff creates a substitute ticket
Actor	Staff
Entry condition	• The Staff member is Logged in • The Staff member selected one of the shops they have access to • A Customer without the applicative asks for a ticket
Event flow	1. The Staff member opens the create substitute ticket section 2. The Staff member asks the Customer and inputs the departments they wish to visit 3. The Staff member clicks on submit 4. The Staff member prints the token and hands it to the Customer 5. The Staff member reads the esimated wait time to the Customer
Exit condition	• The substitute ticket is added to the database • The Customer can use the token to access the Shop when it's their turn in the queue
Exceptions	• The queue is full

Name	Manager checks shop occupancy
Actor	Staff
Entry condition	• The Manager is Logged in with an administrative account
Event flow	1. The Manager selects a Shop from those available 2. The system shows the available occupancy statistcs
Exceptions	• The Manager does not have any listed shop

3.2.4 Sequence diagrams

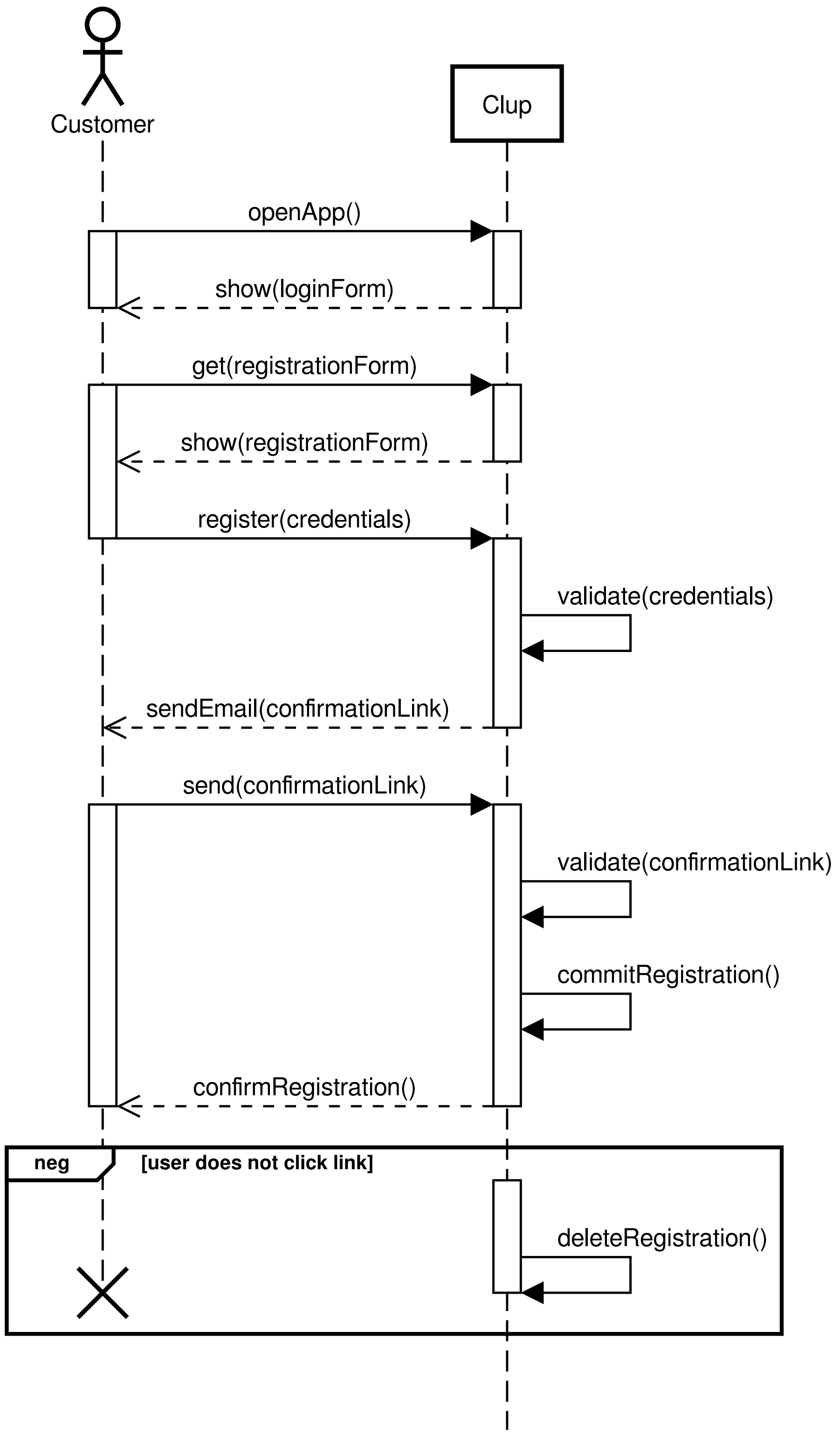


Figure 13: Sequence Diagram for the registration of a Customer

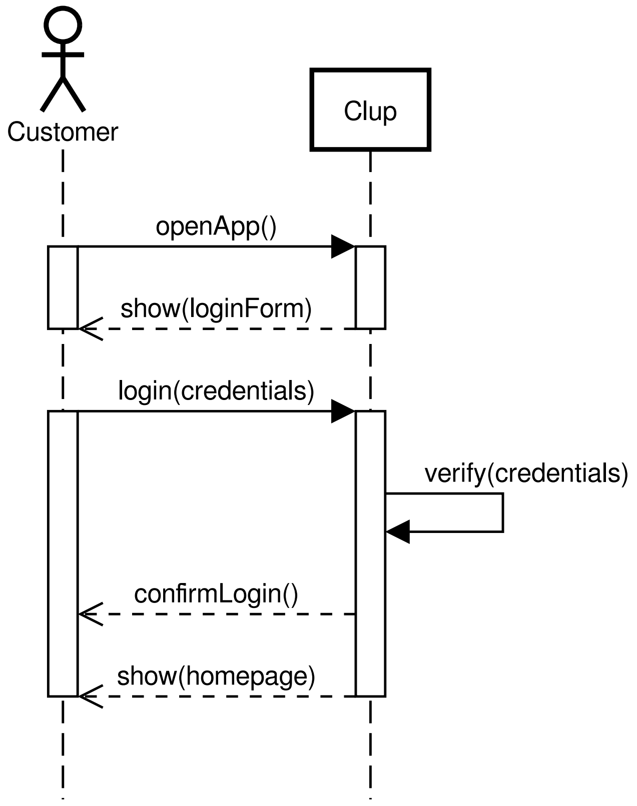


Figure 14: Sequence Diagram for a successful login attempt

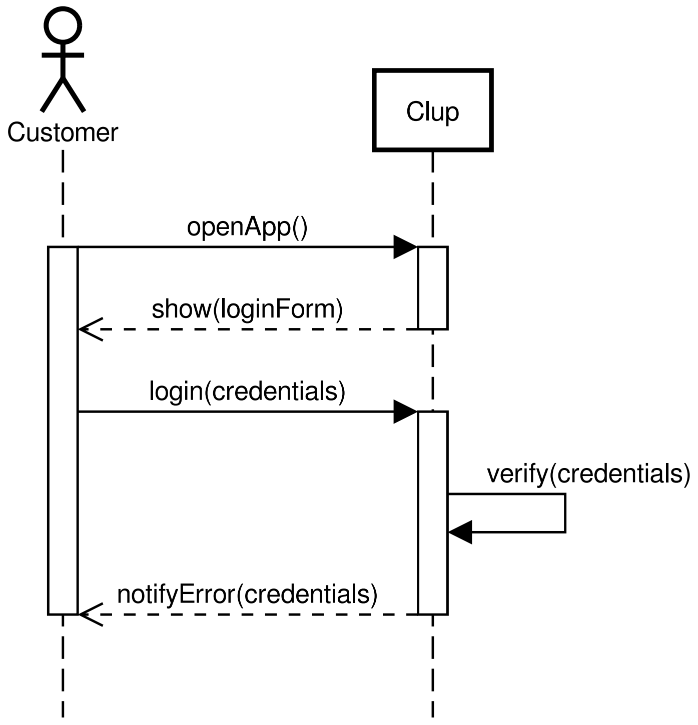


Figure 15: Sequence Diagram for a failed login attempt

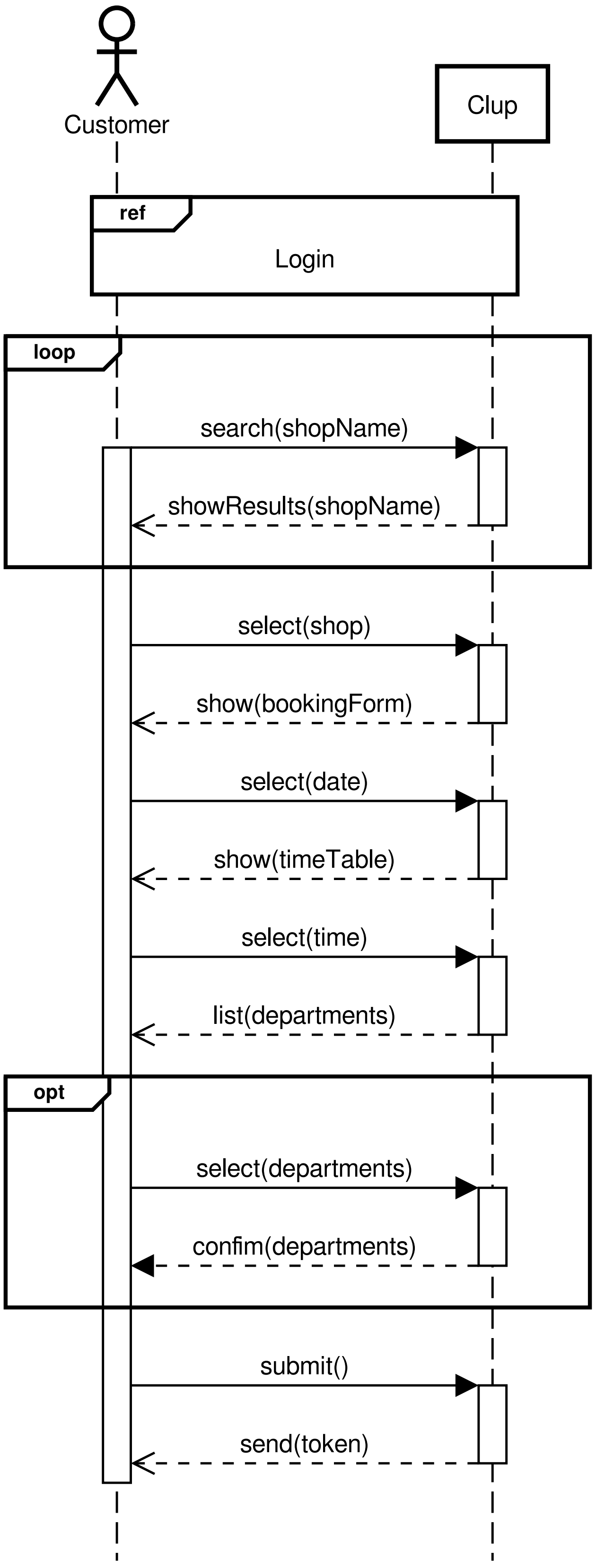


Figure 16: Sequence Diagram for the creation of a Booking

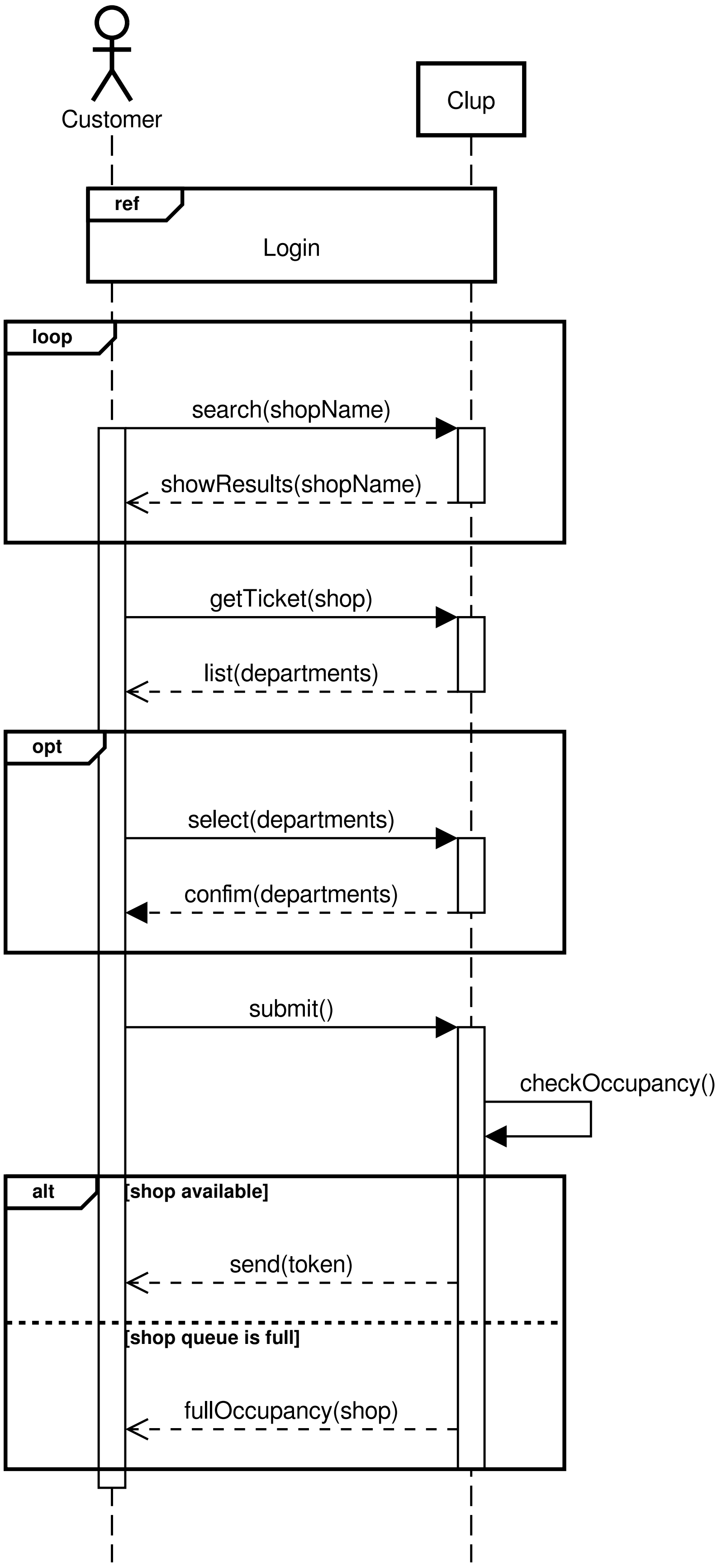


Figure 17: Sequence Diagram for the creation of a Ticket

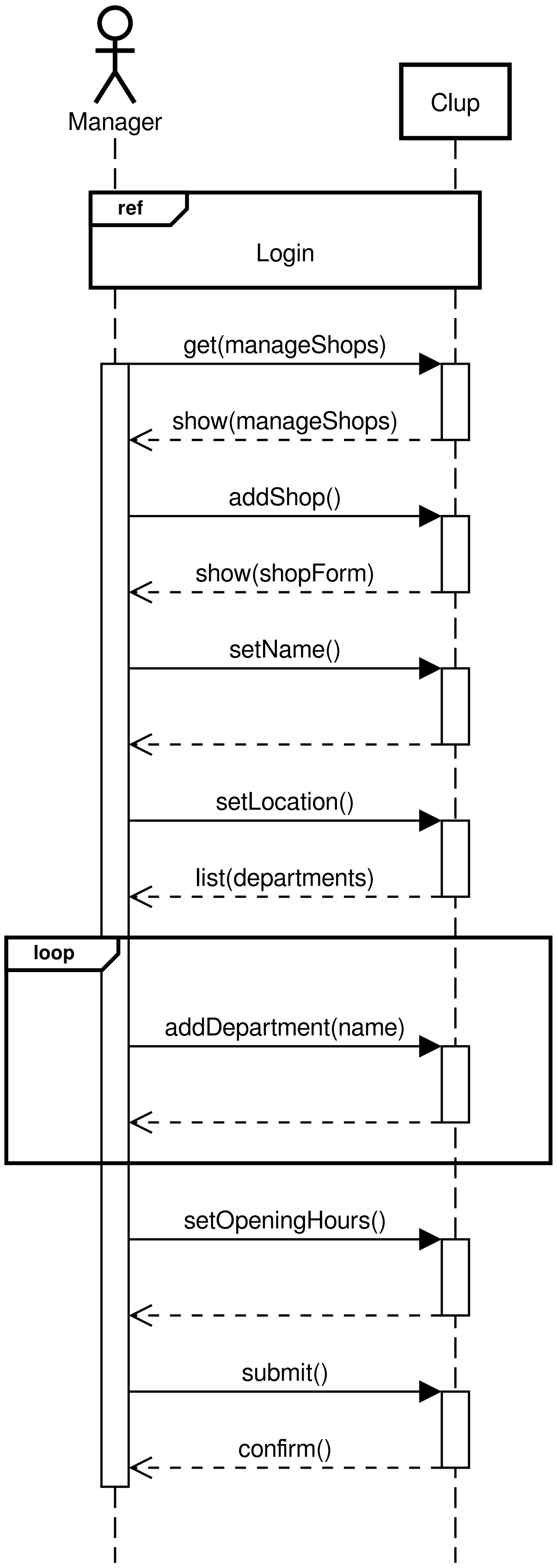


Figure 18: Sequence Diagram for the creation of a Shop

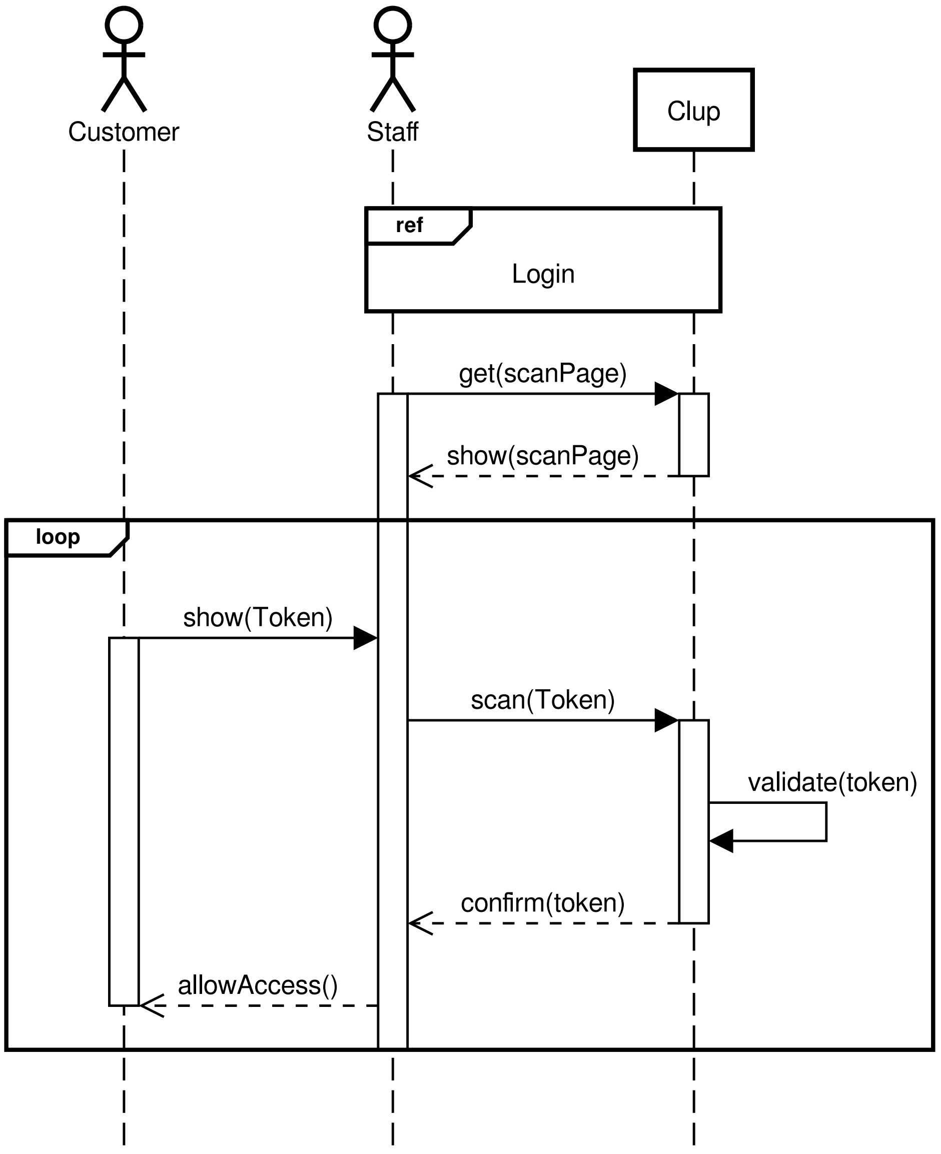


Figure 19: Sequence Diagram for the scan of a valid Token

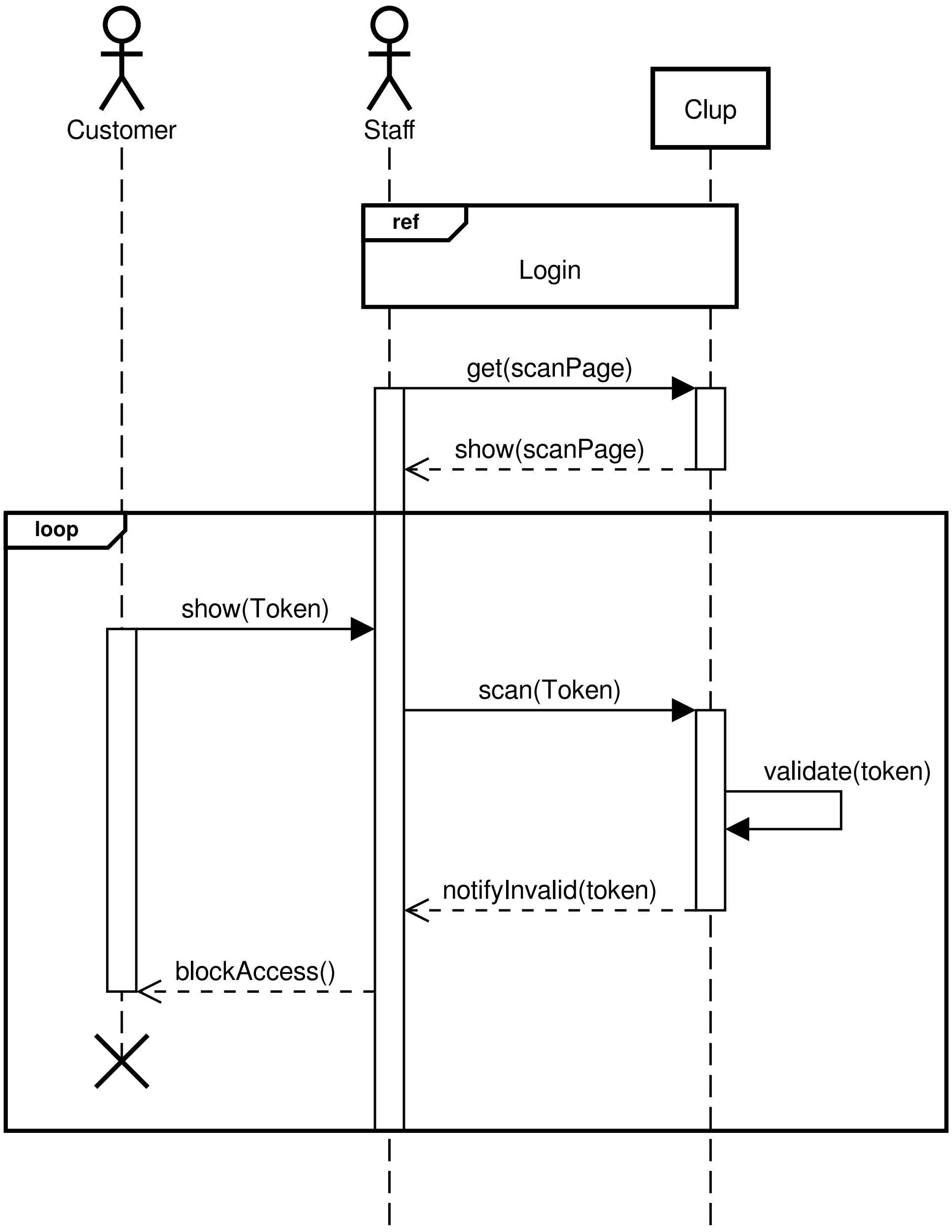


Figure 20: Sequence Diagram for the scan of an invalid Token

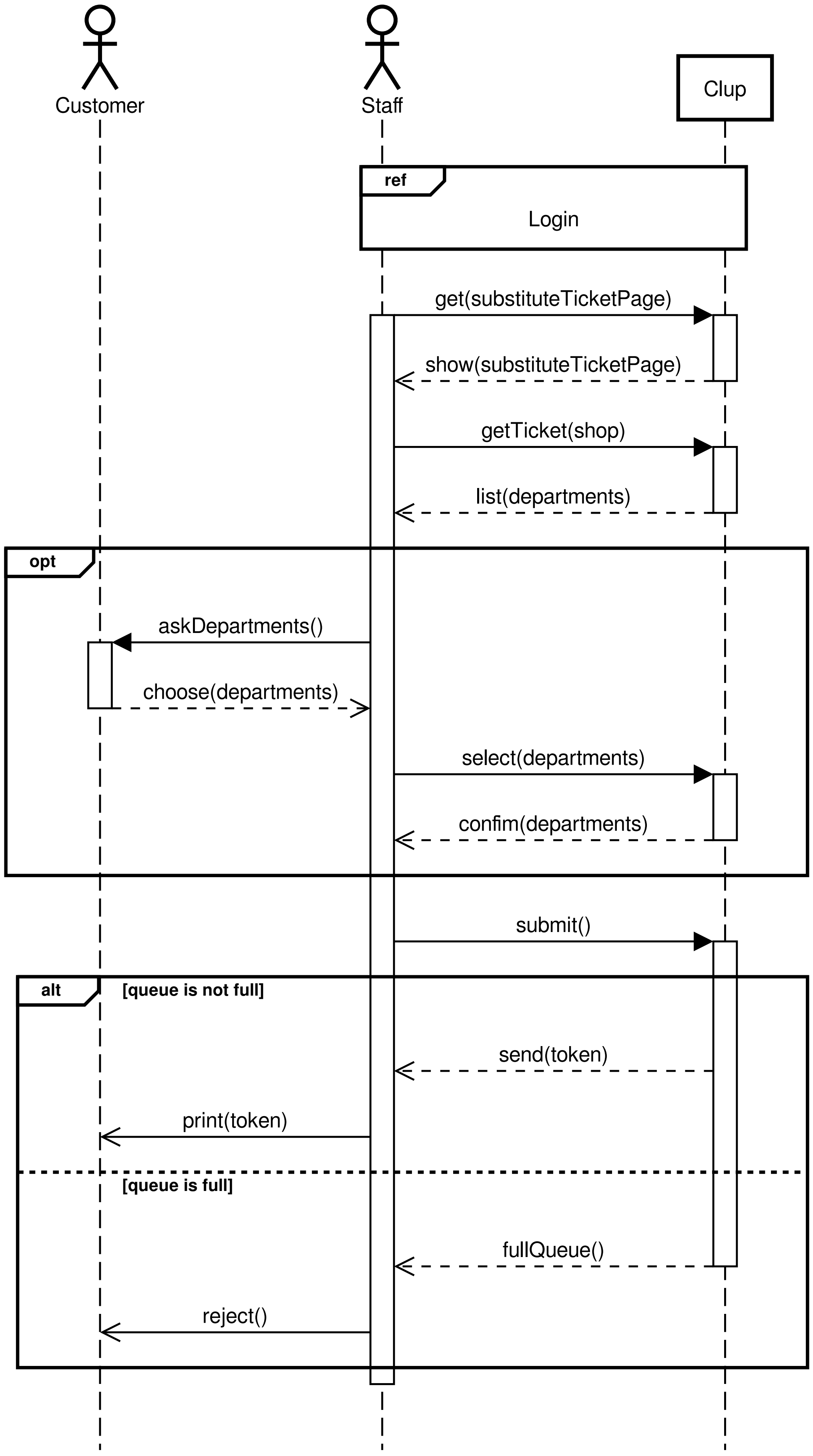


Figure 21: Sequence Diagram for the generation of a Substitute Token

3.3 Performance Requirements

- Validating a token must be done as soon as possible and in no more than 5 seconds from when the token was acquired by the staff. This is required to achieve sufficient throughput and prevent queues from forming at the entrance.
- When obtaining a token the customer shall receive a response from the server within 15 seconds, since in this case time is not critical.
- Tokens shall be added to the system as soon as possible within 30 seconds.

3.4 Design constraints

3.4.1 Standards compliance

The web server shall comply to the HTTPS[1] protocol and target the HTTP/2[2] standard for communication and the HTML5[3] standard for web pages.

3.4.2 Hardware limitations

Running the app requires a smartphone that supports one of the modern web browser engines; older cellphones whose official support has been discontinued shall not be supported. To allow the validation of tickets by the Third Party Staff, the device should also be equipped with a camera of any resolution above 8 Megapixel.

3.5 Software System Attributes

3.5.1 Reliability

CLup should be available 24/7 in order to allow Customers to generate Tokens at any time of the day. Downtime due to maintenance shall be during the night and no longer than 2 hours, furthermore, customers and third parties shall be notified at least 3 days prior the scheduled downtime.

3.5.2 Availability

CLup does not have a critical nature, however any downtime could cause serious problems to the management of the entrances to Shops. Hence, 99% availability during daytime is required for *CLup* to be effective. At night time, the availability requirement can be relaxed to 95%.

[1] RFC2818: HTTP Over TLS
[2] RFC7540: Hypertext Transfer Protocol Version 2 (HTTP/2)
[3] HTML 5.2 W3C Recommendation, 14 December 2017

3.5.3 Security

All communication should be over HTTPS protocol to provide privacy, data integrity and authentication. Customer passwords shall be stored in a salted hash format. The system should keep thorough logs of the activity of logged in customers and of login attempts. The registration and login procedure should not disclose informations about existing customers and repeated login attempts shall be limited. Further details about the cryptographic functions and protocols employed will be discussed in the Design Document.

3.5.4 Maintainability

The system components shall be realized with high modularity and orthogonality between modules. This allows modifications to the code to be localized and leave the other components unaffected, minimizing the time required to fix problems with the system. The backend system shall be able to run a test instance of the service that can be used by developers to test out features before deploying them to the main instance.

3.5.5 Portability

Clup should be easily deployable on a dedicated machine, on a virtual private server or on a cloud hosting service. The server should be able to run both natively and in a containerized environment for maximum portability.

4 Formal Analysis Using Alloy

4.1 Introduction

This chapter presents a formal analysis of the application. Since the application domain is time dependent at its core, we decided to model the evolution of the system over time, with focus on the entrances to the Shops. The model focuses on Customers, Tickets and Bookings: the Staff can be seen as an intermediary and moderator between the Customers and the system, therefore, by assuming that the Staff will follow the rules, we can simplify the model for clarity and ease of comprehension. The scenario being analyzed starts from a state in which Customers already have acquired the tokens and enter the Shops. The queue uses strict constraints, by allowing only the first in queue to enter the Shop. In practice, the constraint can be relaxed to increase throughput if needed. In particular, the model checks the following properties:

- No department in a Shop exceeds its occupancy limits
- Customers cannot enter a Shop without a valid Token
- Customers can use a Booking to enter a Shop only at the time specified
- Customers cannot cut the waiting line for a Shop
- The same Token cannot be used for multiple visits

4.2 Alloy code

4.2.1 Model description

The first section describes the signatures of the objects which are relevant for the formal analysis and their representation invariants.

```
open util/ordering[Time]

sig Time {}

sig Shop{
        departments: disj some Department,
        queue: WaitingListNode lone -> Time,
}
fact everyNodeInQueueHasRightShop{
        all s: Shop, t: Time | s.queue.t.*next.ticket.shop in s
}
fact departmentsOfShop {
        all d: Department | some s: Shop | d in s.departments
}

sig Department{
        maxVisitors: Int,
}
fact positiveMaxVisitors{
        all d: Department | d.maxVisitors ≥ 0
}
fun departmentOccupancy[d: Department, t: Time]: Int {
        #d.~(visiting.t.departments)
}

// Subset of departments visited
```

```
sig Visit {
        departments: some Department
}
fact uniqueVisits {
        all v, v': Visit | v ≠ v' implies v.departments ≠ v'.departments
}

sig WaitingListNode {
        ticket: disj Ticket,
        next: disj lone WaitingListNode,
}

fact waitingListNodeNoCycles{
        all t : WaitingListNode |
                t not in t.^next
}

pred popWaitingList[h, h': WaitingListNode, t: Ticket] {
        h' = h.next
        t = h.ticket
}

sig Customer{
        tokens: disj set Token,
        visiting: Visit lone -> Time,
}

abstract sig Token{
        associatedVisit: one Visit,
        shop: Shop
}
fact tokenVisitShopConsistency {
        all tok: Token | tok.associatedVisit.departments in tok.shop.departments
}
fact tokenIsOwned {
        all tok: Token | some c: Customer | tok in c.tokens
}

sig Booking extends Token{
        timeSlot: one Time,
}
sig Ticket extends Token{}
```

4.2.2 Dynamic properties

This section describes the rules that guarantee the correct evolution of the system over time. In particular, the `Trace` fact rules the transition from an instant to the next for Customers and for the Waiting List.

```
pred hasValidBooking[c: Customer, v: Visit, t, t': Time] {
        some b: Booking | {
                b in c.tokens
                b.timeSlot = t'
                b.associatedVisit = v
        }
}

pred hasValidTicket[c: Customer, v: Visit, t, t': Time] {
        some tick: Ticket |{
                tick in c.tokens
                tick.associatedVisit = v
                popWaitingList[tick.shop.queue.t, tick.shop.queue.t',tick]
                tick.shop.queue.t' not in tick.shop.queue.(prevs[t'])
        }
}
```

```
pred enter[c: Customer, v: Visit, t, t': Time] {
        hasValidBooking[c,v,t,t'] or hasValidTicket[c,v,t,t']

        all d: v.departments | departmentOccupancy[d, t'] =< d.maxVisitors
        c.visiting.t = none
        c.visiting.t' = v
}

pred exit[c: Customer, v: Visit, t, t': Time] {
        c.visiting.t = v
        c.visiting.t' = none
}

pred stay[c: Customer, v: Visit, t, t': Time] {
        c.visiting.t = v
        c.visiting.t' = v
}

// Time consistency
fact Trace {
        all c: Customer | c.visiting.first = none
        all t: Time - last | {
                all c: Customer | {
                        some v: Visit |
                                enter[c,v,t,t.next] or
                                exit[c,v,t,t.next] or
                                stay[c,v,t,t.next]
                        or stay[c, none, t, t.next]
                }
                all s: Shop | {
                        s.queue.t = s.queue.(t.next) or {
                                some c: Customer, v: Visit | {
                                        v.departments in s.departments
                                        hasValidTicket[c,v,t,t.next]
                                }
                        }
                }
        }
}
```

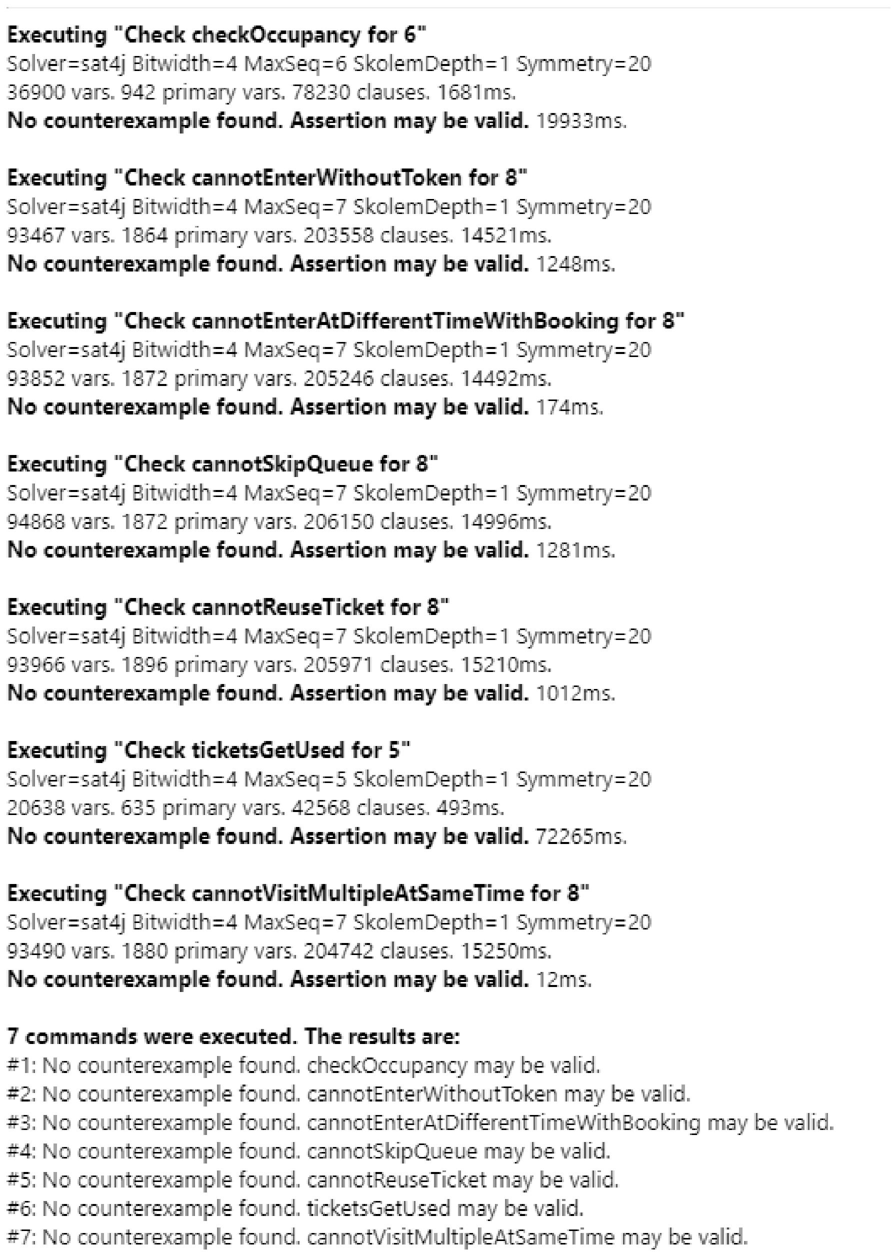
4.2.3 Assertions

This section lists some of the property that will be guaranteed *if* the invariants described in the previous sections hold. The The result of the evaluation of these assertions will be presented in the next section.

```
assert checkOccupancy{
        all t: Time, d: Department | {
                departmentOccupancy[d, t] =< d.maxVisitors
        }
}
assert cannotEnterWithoutToken {
        no c: Customer | {
                some t: Time | {
                        c.visiting.t ≠ none
                        c.visiting.t not in c.tokens.associatedVisit
                }
        }
}
assert cannotEnterAtDifferentTimeWithBooking {
        no c: Customer, v: Visit, t: Time | {
                // Customer has no tickets
                c.tokens & Ticket = none
                // Customer visits some departments
                c.visiting.t = none
                c.visiting.(t.next) = v

                no b: Booking | {
                        b in c.tokens
                        b.associatedVisit = v
                        b.timeSlot = t.next
                }
        }
}
assert cannotSkipQueue {
        no c: Customer, v: Visit, t: Time | {
                // Customer has no booking
                c.tokens & Booking = none

                // Customer visits some departments
                c.visiting.t = none
                c.visiting.(t.next) = v

                no tick: Ticket | {
                        tick in c.tokens
                        tick.associatedVisit = v
                        tick in tick.shop.queue.t.ticket // Ticket was first in queue at t
                }
        }
}
assert cannotReuseTicket {
        no c: Customer, v: Visit, tick: Ticket | {

                // Tick is the only token valid for this visit
                tick.associatedVisit = v
                v not in (c.tokens - tick).associatedVisit

                some t1,t2,t3: Time | {
                        lt[t1,t2]
                        lt[t2,t3]
                        c.visiting.t1 = v
                        c.visiting.t2 ≠ v
                        c.visiting.t3 = v
                }
        }
}
assert cannotVisitMultipleAtSameTime {
        no c: Customer, v1,v2: Visit, t: Time {
                v1 ≠ v2
                c.visiting.t = v1
                c.visiting.t = v2
        }
}
assert ticketsGetUsed {
        no c: Customer, v: Visit, tick: Ticket, t: Time - last | {
                // Customer has no booking
                c.tokens & Booking = none

                tick.associatedVisit = v
                c.visiting.t ≠ v
                c.visiting.(t.next) = v

                tick.shop.queue.t.ticket = tick
                tick in tick.shop.queue.(nexts[t]).ticket
        }
}
```

4.2.4 Execution results

This section shows the experimental result of the formal analysis.



```
Executing "Check checkOccupancy for 6"
Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20
36900 vars. 942 primary vars. 78230 clauses. 1681ms.
No counterexample found. Assertion may be valid. 19933ms.

Executing "Check cannotEnterWithoutToken for 8"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
93467 vars. 1864 primary vars. 203558 clauses. 14521ms.
No counterexample found. Assertion may be valid. 1248ms.

Executing "Check cannotEnterAtDifferentTimeWithBooking for 8"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
93852 vars. 1872 primary vars. 205246 clauses. 14492ms.
No counterexample found. Assertion may be valid. 174ms.

Executing "Check cannotSkipQueue for 8"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
94868 vars. 1872 primary vars. 206150 clauses. 14996ms.
No counterexample found. Assertion may be valid. 1281ms.

Executing "Check cannotReuseTicket for 8"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
93966 vars. 1896 primary vars. 205971 clauses. 15210ms.
No counterexample found. Assertion may be valid. 1012ms.

Executing "Check ticketsGetUsed for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
20638 vars. 635 primary vars. 42568 clauses. 493ms.
No counterexample found. Assertion may be valid. 72265ms.

Executing "Check cannotVisitMultipleAtSameTime for 8"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
93490 vars. 1880 primary vars. 204742 clauses. 15250ms.
No counterexample found. Assertion may be valid. 12ms.

7 commands were executed. The results are:
#1: No counterexample found. checkOccupancy may be valid.
#2: No counterexample found. cannotEnterWithoutToken may be valid.
#3: No counterexample found. cannotEnterAtDifferentTimeWithBooking may be valid.
#4: No counterexample found. cannotSkipQueue may be valid.
#5: No counterexample found. cannotReuseTicket may be valid.
#6: No counterexample found. ticketsGetUsed may be valid.
#7: No counterexample found. cannotVisitMultipleAtSameTime may be valid.
```

Figure 22: All assertions pass the test on a restricted domain.

```
pred enterAndExit {
        some t, t', t'': Time, c: Customer, v: Visit | {
                c.visiting.t = none
                c.visiting.t' = v
                c.visiting.t'' = none
                lt[t, t']
                lt[t', t'']
        }
}
pred enterExitTicketBooking {
        some c1,c2: Customer {
                c1 ≠ c2
                c1.tokens & Ticket = none
                c2.tokens & Booking = none
                some t, t', t'': Time, v: Visit {
                        c1.visiting.t = none
                        c1.visiting.t' = v
                        c1.visiting.t'' = none
                        lt[t, t']
                        lt[t', t'']
                }
                some t, t', t'': Time, v: Visit {
                        c2.visiting.t = none
                        c2.visiting.t' = v
                        c1.visiting.t'' = none
                        lt[t, t']
                        lt[t', t'']
                }
        }
}
pred visitDifferentShops {
        some c: Customer, s1,s2: Shop, t1, t2: Time {
                t1 ≠ t2
                s1 ≠ s2
                c.visiting.t1 ≠ none
                c.visiting.t2 ≠ none
                c.visiting.t1.departments in s1.departments
                c.visiting.t2.departments in s2.departments
        }
}

check checkOccupancy for 6
check cannotEnterWithoutToken for 8
check cannotEnterAtDifferentTimeWithBooking for 8
check cannotSkipQueue for 8
check cannotReuseTicket for 8
check ticketsGetUsed for 5
check cannotVisitMultipleAtSameTime for 8

run {} for 6 but exactly 2 Shop, exactly 3 Department, exactly 3 Ticket, exactly 3 Booking,
    ↪ exactly 4 Customer, exactly 4 Visit
run visitDifferentShops for 5 but exactly 5 Customer
run enterAndExit for 8 but exactly 5 Customer, exactly 5 Token, exactly 2 Booking, exactly 5
    ↪ Department, exactly 2 Shop, exactly 4 Visit
run enterExitTicketBooking for 8 but exactly 5 Customer, exactly 6 Token, exactly 3 Booking,
    ↪ exactly 3 Department, exactly 2 Shop, exactly 4 Visit
```

All presented predicates have valid instances in the domain.

The following figures show the evolution of a scenario of the enterAndExit predicate for 5 consecutive Time instants. When a Customer visits a Shop the graph shows a *visiting* relation from the customer to one of the Visit elements, which point to the departments of the shop concerning the visit.

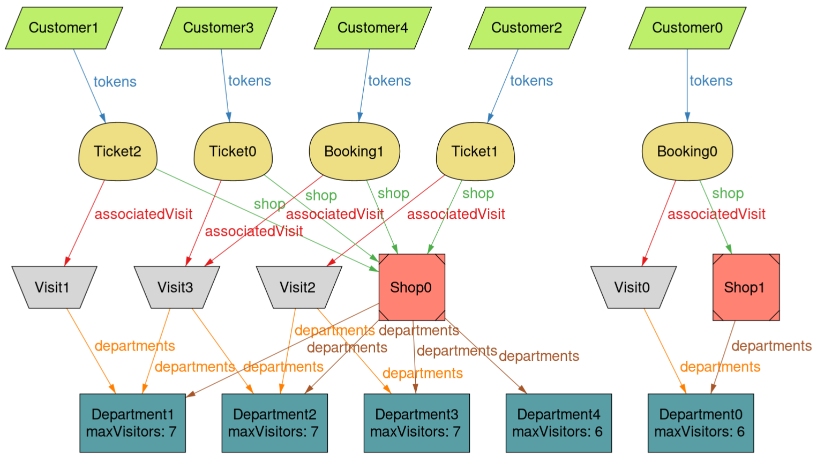


Figure 23: Run for Time 0. This is the initial state.

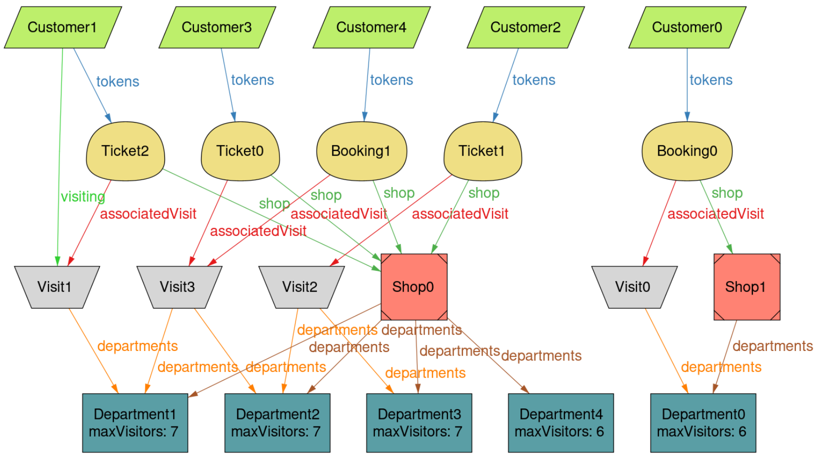


Figure 24: Run for Time 1. *Customer1* has used his Ticket to enter *Shop0*.

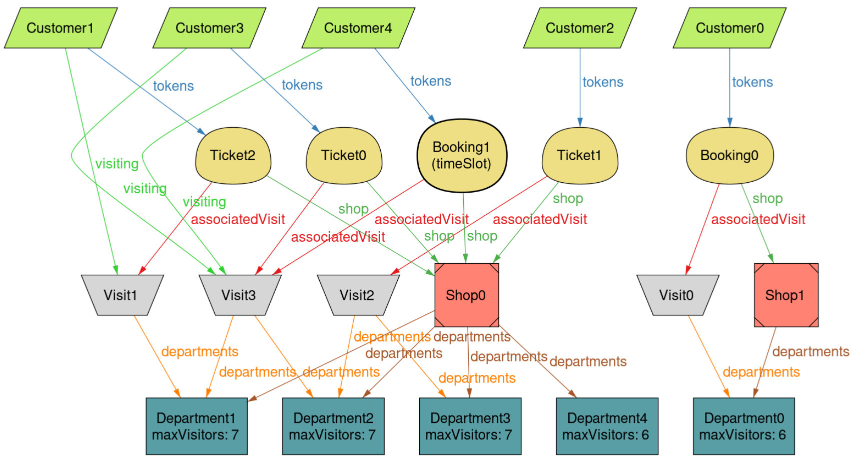


Figure 25: Run for Time 2. *Customer3* and *4* enter the same shop with a Ticket and a Booking respectively.

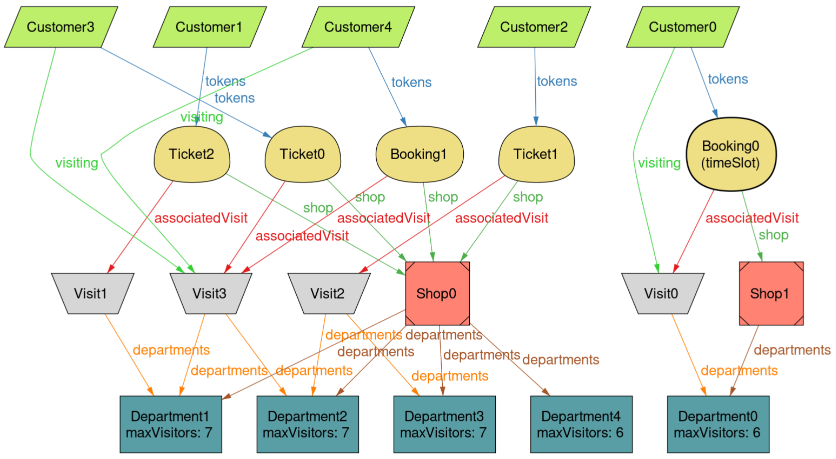


Figure 26: Run for Time 3. *Customer1* correctly exits *Shop0*, while *Customer0* makes use of a Booking for *Shop1*

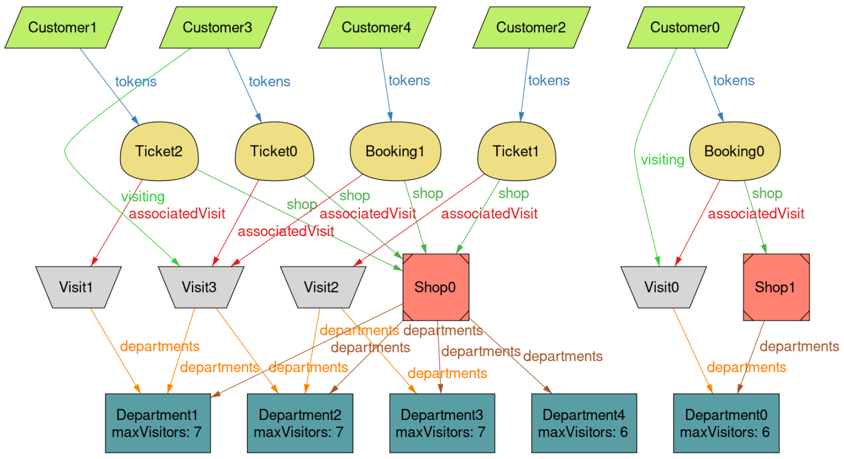


Figure 27: Run for Time 4. *Customer4* exits as well.

Figure 28 and 29 show a detailed view of the same instance, displaying the working principle of the waiting List. Each *WaitingListNode* represents the position of a Customer in the *Waiting list*, associated with their Ticket. The first in queue for each Shop is marked by the *queue* relation. In the first figure, set at Time 0, we see that *Customer1* is the first in queue, since their Ticket is in the first WaitingListNode for *Shop0* (marked by the *queue* relation), thus he is allowed to enter at the next time frame. Indeed, in the second figure, set at Time 1, a new *visiting* relation between *Customer1* and *Visit1* is created, to represent the customer visiting *Department1* of *Shop0*. As expected, the *queue* relation of the Shop has now moved forward to the next WaitingListNode, advancing the queue.

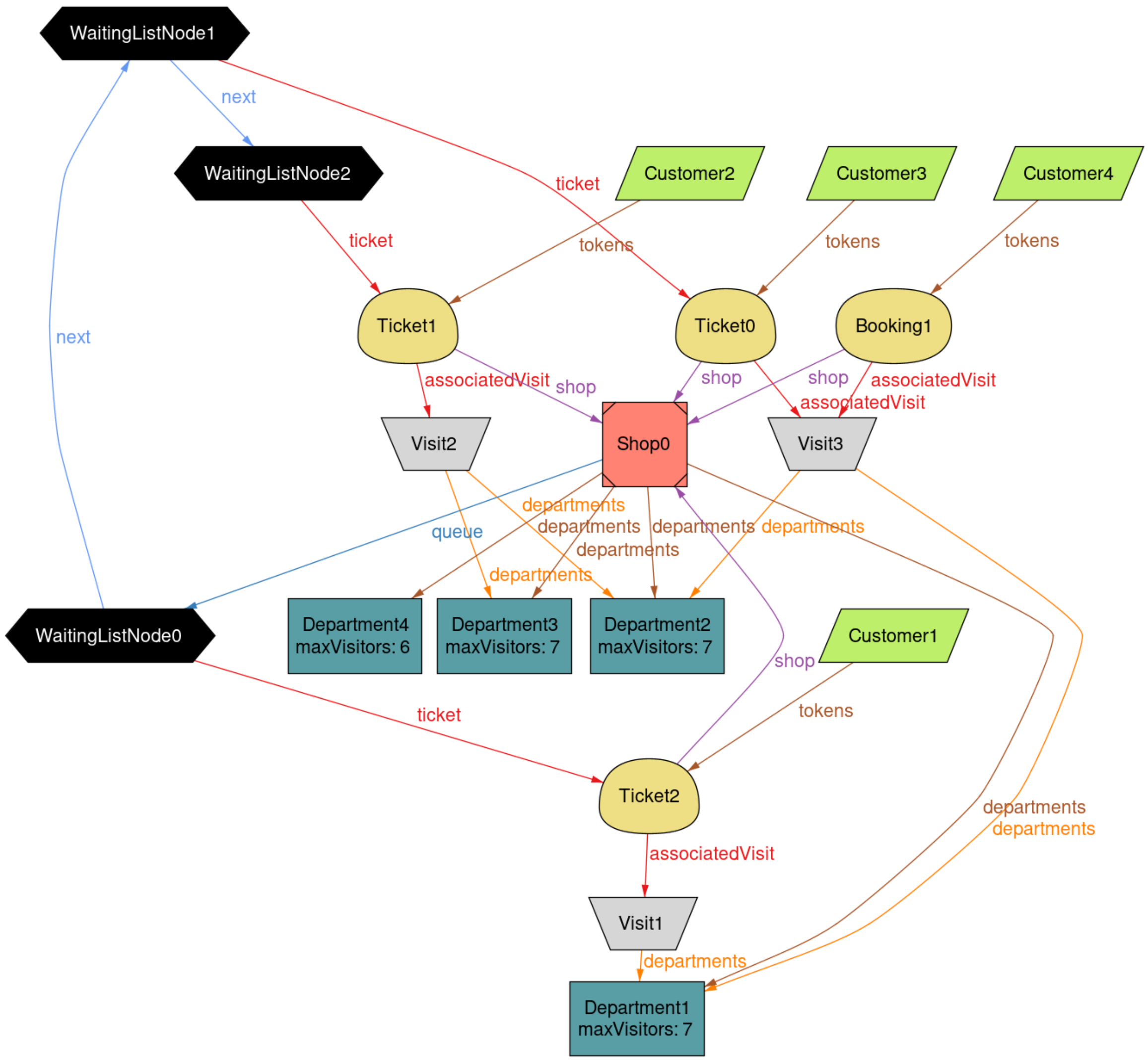


Figure 28: The same instance at Time 0, with WaitingListNode elements shown

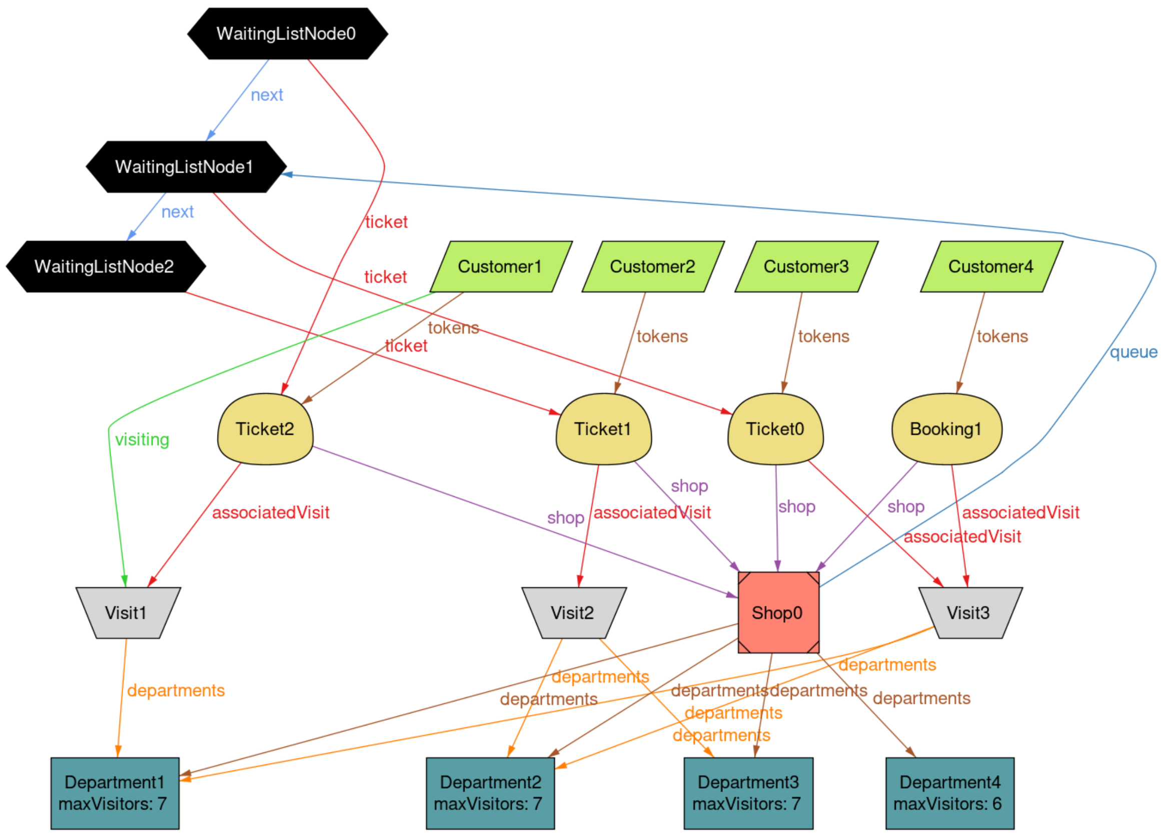


Figure 29: The evolution of the instance at Time 1

5 Effort Spent

Luca De Martini

Date	Hours	Description
2020-11-04	2h	Introduction & Purpose
2020-11-24	7h	Introduction & Scope
2020-11-25	2h	Class diagrams
2020-12-01	4h	State diagrams & Assumptions
2020-12-03	3h	Requirements
2020-12-06	2h	Requirements
2020-12-09	3h	Scenarios and Use Cases
2020-12-14	1h	Use Cases
2020-12-15	2h	Use Cases and review
2020-12-16	1h	Review
2020-12-17	1h	Goal Requirement mapping
2020-12-18	1h	Review
2020-12-19	3h	Mockups and Review
2020-12-20	3h	Alloy
2020-12-21	4h	Alloy
2020-12-22	5h	Alloy
2020-12-23	1h	Review
	45h	

Alessandro Duico

Date	Hours	Description
2020-11-04	2h	Introduction & Purpose
2020-11-24	7h	Introduction & Product functions
2020-11-25	2h	Product functions
2020-12-01	4h	State diagrams
2020-12-03	3h	Customer Interfaces
2020-12-06	1h	Requirements
2020-12-09	3h	Use Case Diagrams
2020-12-15	4h	Sequence Diagrams and Mockups
2020-12-16	2h	Sequence Diagrams
2020-12-18	6h	Mockups
2020-12-19	3h	Review and Alloy
2020-12-20	3h	Alloy
2020-12-21	4h	Alloy
2020-12-22	5h	Alloy
	49h	

References