CLup project Roberto Buratti, Hrvoje Hrvoj, Ozan Incesulu

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Roberto Buratti, Hrvoje Hrvoj, Ozan Incesulu
Version:	1.0
Date:	10-12-2020
Download page:	https://github.com/Furcanzo/BurattiIncesuluHrvoj
Copyright:	Copyright © 2020, Roberto Buratti, Hrvoje Hrvoj, Ozan Incesulu – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

1.1.1 Description of the Proposed System

The project "CLup - Customer Line up" is a line spot reservation system planned to be used by managers, clerks, and customers of many local vendors and chains. The system aims to come up with a handy solution for the ongoing issue of proper social distancing management, particularly in grocery shopping, by providing assistance to cope with the customer load for managers and helping customers access products in a safe and controlled manner.

In particular, users will be able to see the locations, get a line number, and book in advance for the grocery stores they would like to visit. Once assigned a line number, the customer will track the estimated time of arrival of the line and wait for the notification that informs about his or her line's forthcoming arrival; hence waiting time in the line in the crowd is minimum. Also, "CLup" provides uniquely generated QR codes per the line number, which can be utilized by the store managers as a proper monitoring tool in the entrances and exits of the locations. The product's general purpose is to keep the congestion levels in line with the locations at a minimum via providing useful features for all the users.

1.1.2 Goals

- G_1 Customers can issue a line number for a location.

- G_2 Customers can issue line numbers for their future visits.

- G_3 Customers can detail their visits by category or product.

- G_4 Customers can plan their visit to the store.

- G_5 Customers may prefer to use alternative time slots or partner stores for their visit.

- G_6 Managers can prevent customers from issuing line numbers.

- G_7 Managers can customize the system to allow optimizations for increased granularity, flow control, and time slot forecasting.

- G_8 Clerks and Managers can monitor the customers through their entrance and exits.

- G_9 Customers can obtain printed line number tickets.

1.2 Scope

1.2.1 Targeted Users

"CLup" aims to resolve the problem of Customers queueing up in front of a location, without control of availability of place in the location and future contact tracing by managers.

Customer:

Customers will be able to obtain specific line numbers for various locations using CLup, which they can track the estimated time available with and also view the location on a map application to plan their visit. Customers can further obtain line numbers for future visits, based on the system's info about the availability of free spots on the specific time intervals. They may prefer to visit a different branch of

the same chain. Furthermore, they can provide specific products they intend to purchase or set an estimated time for their visit to allow finer granularity. Some customers may also prefer to obtain a line number upon visiting the store physically. To plan their visit in a time slot where the location will be less crowded, the users can see the location's occupancy based on already taken line numbers and forecasts provided by the system.

Clerk:
The clerk (which can be a shopping assistant or a security detail) can monitor customers' flow and manually intervene in case of missing line numbers via performing manual checkout for a specific customer or by printing line numbers physically for some customers.

Manager:

The location manager (which could be an actual manager or someone responsible for handling customer management) can provide details regarding the availability of products and the location in general, by setting the opening hours, maximum allowed customers in the shop, in-shop location of different products and categories, the maximum amount of reservations that can be made per customer, line number timeout and the location. Also, for chains and relevant stores, the manager can add chain members for the location.

1.2.2   Relevant Phenomena

Phenomenon	World / Machine	Shared
Line Number	Machine	Yes
Line Number Ticket	World	No
Product	World	No
Product Category	World	No
In-store Location	World	Yes
Occupancy Forecast	Machine	No
Store	World	Yes
Time Slot	Machine	Yes
Ticket Printer	World	Yes
Line Number Timer	Machine	No
Customer Scheduling Algorithm	Machine	No

Table 1: Relevant Phenomena of CLup

1.3   Definitions, Acronyms, Abbreviations

1.3.1   Definitions

- *Location*: the physical location of the business that operates the line reservation system

- *Manager*: the user in charge of executive action within the location

- *Customer*: the user to visit the location

- *Clerk*: the user in charge of handling the entrance and exit of customers

- *Hard-coded Super User*: a single user used to create managers of locations administered by the software vendor

- *Visit Time*: the time interval in which a customer performs a visit to the location

- *Line Number*: A number that indicates the ordering of a specific customer in the line

- *Line Number Ticket*: A physical ticket printed that features the line number and the QR code.

- *Time Slot*: Specific intervals of time determined by the opening hours and average visit time per customer.

- *Partner Store*: A different location included in the same beneficiary chain of command (such as another member of the franchise or store chain) or a mutual agreement with the specific location

- *Product*: Any item, items, service or services demanded by the customer, and provided by the location to the customer.

- *In-store Location*: A location of a specific product or a product category inside the location.

- *Working hours*: The time intervals that the store is open during each day.

- *Maps API*: A third-party mapping service implementation used for location tracking

1.3.2 Acronyms

- **RASD**: Requirement Analysis and Specification Document

- **QR Code**: Quick Response Code

- *API*: Application Programming Interface

1.3.3 Abbreviations

- G_n: n^{th} goal

- D_n: n^{th} domain assumption

- R_n: n^{th} functional requirement

1.4 Revision history

1.5 Reference Documents

- Specification Document: R&DD Assignment AY 2020-2021

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications

- ISO/IEC 18004:2015 QR Code bar code symbology specification

1.6 Document Structure

This document is composed of six sections, each with the purpose described below:

- **Introduction**: This section provides an introduction of the problem, the scope of the project with details regarding the goals, target users, and phenomena. The project's goals are formulated per the description of actions and actors in the Specification Document. Within the project's scope, the properties and duties of different users and user groups are described in the Target Users section. Furthermore, under the project's scope, the relevant phenomena of the project are presented through their relevance to the world and the machine.

- **Overall Description**: This section builds upon the introduction section by providing an expansion of the system's scope and relevant functionalities. The fundamental perspective is provided considering the shared phenomena between the world and the system and the system's integrations with other parties and products. The overall actions that can be conducted over the system are described using a detailed class diagram, portraying all required components of the system, their inner state, and provided functionalities for other components to facilitate their purpose. All core features of the system are listed and explained using user scenarios to understand better the real-world condition on which the use case should apply, a state chart for demonstrating the relationship between various states the system can be in during the execution of the provided core feature scenario. Next, the system's requirements to archive the goals mentioned above and allow the provided scenarios' execution are given. Based on their roles in the system, the user characteristics are further evaluated in this section through the means of the needs that they have, how the system archives those needs via its requirements and goals. No system can be designed without a broad idea of what domain it exists and how much the environment limits its ability to perform its features. Ergo, we provide the standard assumptions of the domain, the system's dependencies to external components, and the world's limits on the system.

- **Specific Requirements**: This section is the main section of the document. The external interfaces that the system requires to function correctly are presented with details and additional mockups if needed. The interfaces section is mainly concentrated on different views of the user interfaces as the user-facing part is the main way to integrate with the system. In this section, the system's functional requirements are presented in detail, a common use case diagram is provided, and each use-case of the system is analyzed with details on a use case description table and a sequence diagram. The performance requirements of the system, with a focus on development speed and scalability, are provided. The system's design constraints, with a focus on different standard compliances, hardware limitations of the system to function, and other constraints, like GDPR, are also elaborated in this section. The vertical aspects that need consideration are evaluated under the Software System Attributes subsection of this document, with an emphasis on Reliability, Availability, Security, Maintainability, and Portability.

- **Formal Analysis Using Alloy**: This section features various models built, and hypothesis verified using Alloy. This section demonstrates that some aspects of the requirements document can be formally proven correct and provide additional information about the proposed system to the engineers.

- **Effort Spent**: This section features the effort table, in which all team members provide a rough estimation of the time spent on the creation of the various sections of the document.

- **References**: This section features different reference materials referred to in this document.

2 Overall Description

2.1 Product Perspective

The CLup system is expected to be developed ground-up, without any core components being shared with other services, and is expected to integrate with few external services. The system will be using a mapping API to allow its various requirements regarding navigation. The system will be used to control, verify, and schedule customer visits to different stores.
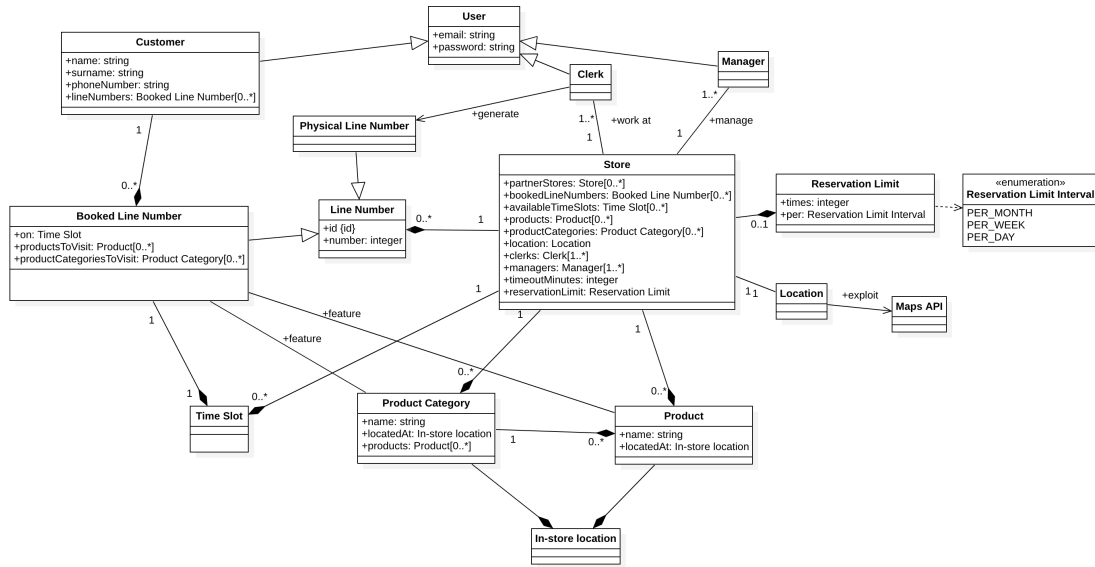

Figure 1: Class Diagram

The **??** provided above illustrates the system's core high-level components and their interactions.

The system is designed to be used by different types of users, the **Customer**s whom visit or plan to visit the **Store**, the **Manager**s who are responsible for handling management related tasks like providing store details, handling system stops, and the **Clerk**s who are on active duty for monitoring the entrance and exit of customers. All users on the system are identified with their *e-mail*s and require their *password*s to authenticate. The **Customer**s need to register to the system with their *name*, *surname* and *phoneNumber*, in addition to the information required for authentication. All **Line Number**s feature a unique *id* and an actual *number* that indicate their order in line. Customers can have many **Booked Line Number**s, limited by the *reservationLimit*, from various stores, which provide many **Time Slot**s that the customers can select from under *availableTimeSlots*, which then becomes the time slot the booking is *on*. The line number is to be invalidated after *timeoutMinutes* from its target time interval. The **Store**'s **Location** is set by the **Manager** and observed by customers through a **Maps API**. The **Reservation Limit** imposes a customer can book a line number at most for some *times per* a specific **Reservation Limit Interval** (one of *PER_MONTH*, *PER_WEEK* or *PER_DAY*). Customers' visits can also feature different **Product**s or **Product Category**ies that they might want to visit under their line number's *productsToVisit* and *productCategoriesToVisit*, each having a *name* and *locatedAt* their own **In-store Location**. **Clerk**s can generate **Physical Line Numbers** for customers that do not have the app available for them.

Below some scenarios, statecharts and their brief description are enumerated for the system's core and critical functionalities. It is important to note that the app is targeted at users of all age ranges; thus, the scenarios feature different users of different ages. The statecharts apply to many other scenarios that might occur out of interaction with various types of users.

2.1.1 Book Line Number

Scenario

Ozan wants to visit the new hamburger place that has recently opened just around the corner of his house. However, Ozan does not want to wait in a long line to purchase a burger and some fries. The fast-food joint has incorporated the CLup system in its customer service portfolio, making it possible for Ozan to reserve his ticket beforehand without visiting the location. Ozan downloads the CLup application and books his line number from his house. Using the app, he can plan his route and schedule to the location beforehand and place his order directly upon his arrival.
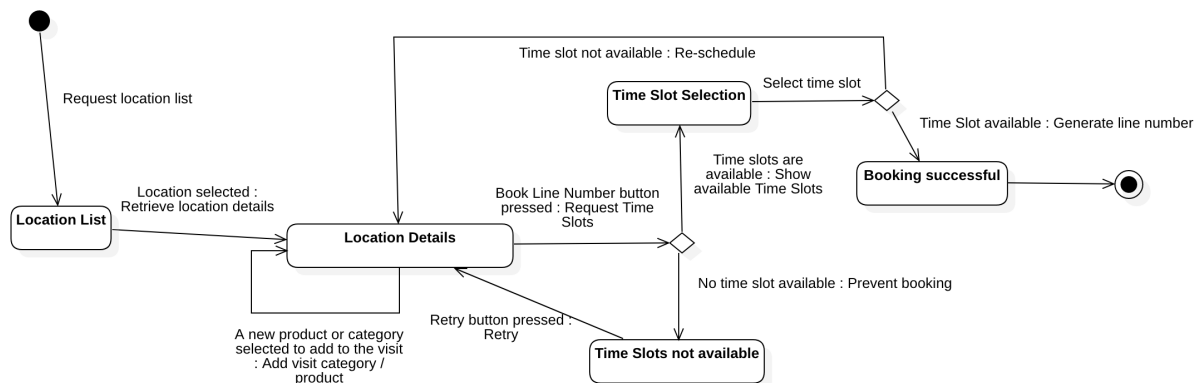
State Chart

Figure 2: State Diagram for feature Book Line Number

The **??** represents the execution flow of one of the core use cases, namely booking a line number for the future. The customer books the line number by first listing what locations are available for them to book. Then, when a customer selects a location, they view the location's details, where they can add products or product categories to their target visit. The customer can then view the time slots that the system will generate specifically filtered for their visit based on the location's provided data and availability. When the customer selects a specific time slot, the system tries to book that specific location. If the system successfully allocates the desired time slot, it displays the success message and the customer's ticket. Else the user is notified of the failure and brought back to the scheduling screen to select another slot.

2.1.2 Get Line Number

Scenario

Roberto lives in Milan during the COVID lockdown and wants to visit a grocery store near his house. He wants to obtain a line number for his visit through his phone not to waste time while waiting in line for other customers to be done with their affairs. He hears that the store he plans to visit has been on CLup, so he downloads the application and opens it. Using the app, Roberto can now see when he should take off to reach the shop without waiting in front of the store.
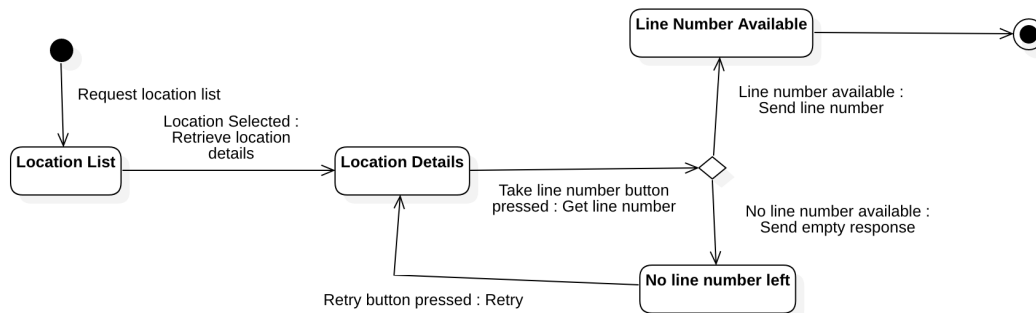
State Chart

Figure 3: State Diagram for feature Get Line Number

The **??** represents another core use case's execution flow, booking a line number for a recent visit. The customer generates a line number by going over the list of available locations, similar to the Book Line Number feature; however, they press the "Take Line Number" button directly this time. Then, the system tries to generate a line number for them, and if it is successful, the user retrieves the resulting line number. If the system can not fulfill the user's request, it communicates the problem to the user and allows a retry.

2.1.3   Print Line Number

Scenario

Hrvoje, a Croatian who recently arrived in Italy, is not aware of the popularity of the CLup app; however, he wants to visit an electronics store to purchase a new phone because his current phone has died out of battery failure. Upon arrival, since he does not have a phone, he can not register to the CLup system; however, the security, that is also in charge of validating line numbers, prints him a ticket that he should keep during his whole visit. Therefore, Hrvoje is allowed in the store, even if he does not have the application, and the shop manager can monitor his entrance and exit.
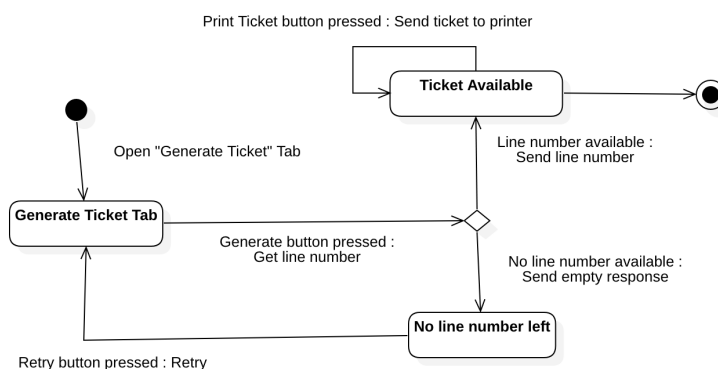
State Chart

Figure 4: State Diagram for feature Print Line Number

The **??** represents the execution flow of a similar use case, printing an actual ticket for on foot visitors.

Upon the arrival of a new customer that does not have a line number, the clerk opens the generate ticket tab and presses the "Generate" button to create a new line number. If all the constraints are satisfied, the system allocates a new ticket number for the customer. The clerk can then print the ticket out using a printer. In case there are no line numbers left, the system notifies the clerk as such, with an option to retry the operation.

2.1.4   System Stop

Scenario

Gianfranco is a team lead in a shoe factory. The shoe factory also has an outlet store, where the local brand sells its shoes directly to customers at a discounted price. One day, a fire erupts due to a malfunction in the automated sewing machine, and it starts to spread all over the building. Gianfranco coordinates the evacuation of the factory and the store. He also uses the CLup application to issue a system stop to prevent further customers from flooding in. All the store customers receive notification regarding this unfortunate event and do not arrive at the location.
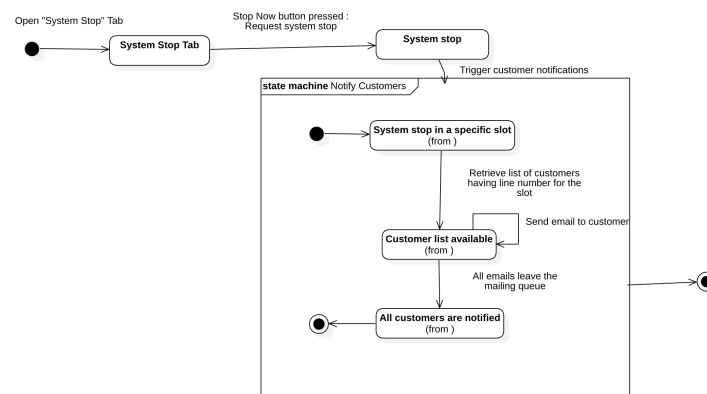
State Chart

Figure 5: State Diagram for feature System Stop

The **??** represents the execution flow for a critical use case, the manager's ability to halt the system so that all line numbers get canceled, all users are notified, and no other line numbers are issued. The manager first opens the System Stop tab on their phone and afterward requests a system stop. This action stops the system from issuing more tickets and triggers the distribution of notifications for the time zones the system is stopped in. In the notification part of the state diagram, the system starts with a request for a stop for a specific time slot and retrieves all the customers that have already scheduled for that slot. The system then starts sending emails to all customers that are in the list provided. When all emails leave the mailing queue, the system has successfully notified all the customers regarding the stop. The state machine of Notifying Customers is re-used to describe the notification mechanism in the **??**.

2.1.5   Schedule System Stop

Scenario

Giuseppe is the owner of a famous local pizza place. He is using the CLup system in his store to manage the customer lines. He is also preparing a new radio advertisement for his store, with an advertisement agency. They have a meeting next week during work hours, and he does not have anyone to give the authority to manage the store. Therefore, he has to shut the store down for half the day. He

schedules a system stop from the CLup system to prevent ticket numbers from being issued on that day for his store.
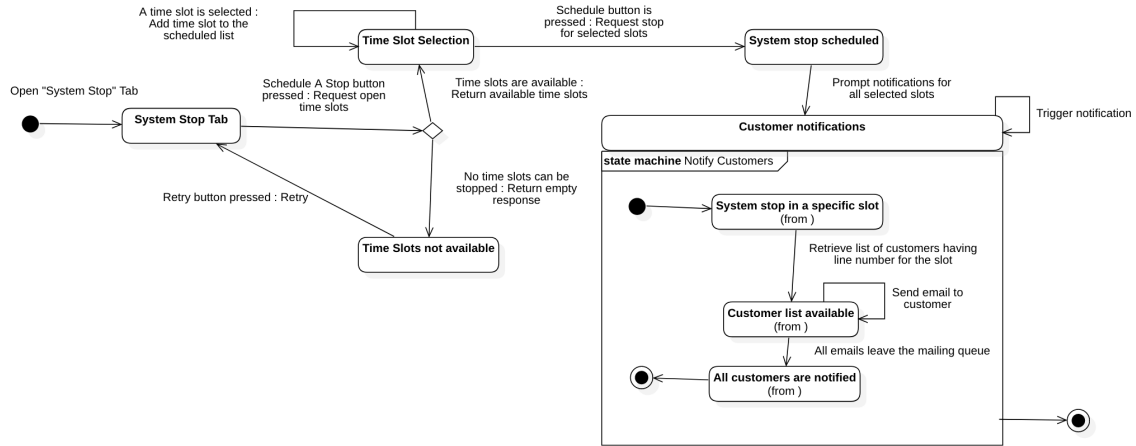
State Chart

Figure 6: State Diagram for feature Schedule System Stop

The **??** represents the final core use case of the system that is considered significant, scheduling the system to stop in the future. To start this flow, the manager first opens the System Stop tab and presses the schedule a stop button. The system then pools all the available slots that the manager can issue a stop on and returns the list to the time slot selection screen. If no such slot is available, a screen indicating such a case is visible, with an opportunity to retry the action. The manager selects specific time slots from the time slot selection list that the system shall stop for and press the Schedule button to schedule a system stop. A scheduled system stop also sends a notification to all customers that have booked a time slot for that shop before the system stop. The mechanism of customer notification is detailed in the description of the **??**, so it will not be repeated here.

2.2 Product functions

2.3 User characteristics

2.4 Assumptions,dependencies and constraints

2.4.1 Domain Assumptions

- D_1 %80 of the customers and all clerks and managers have basic ICT skills, has an e-mail address that they are willing to use to authenticate to the system, and has a smartphone or equivalent device that can connect to the Internet, have a browser that supports UTF-8, display QR codes and has a mapping application.

- D_2 Locations will be visited by no more than 1000 people in any time slot.

- D_3 %98 of the customers will arrive at the given location either without a ticket or with a ticket that has not timed out.

- D_4 E-mail addresses are not shared by multiple users of the system.

- D_5 Clerks' mobile devices are equipped with at least one camera that the system can use.

- D_6 All users have a basic understanding of how the line numbering system works and respects the system's ordering.

- D_7 Managers have an estimate for the number of reservations that their location can at most have.

- D_8 Managers' device has location services with a location acquisition error for no more than 20 meters.

- D_9 Clerks are continually monitoring the locations entrances and exits.

- D_{10} Locations have printing equipment in 5 meters range of all the entrances that can print QR codes and line numbers.

- D_{11} At least one manager is available in the location during the working hours.

- D_{12} The customer's entry and exit to the store are determined by whether the clerks have checked them in and out.

- D_{13} The customer has their line number or line number ticket available with them through their visit, including their exit from the store

2.4.2 Dependencies

2.4.3 Constraints

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Below are some sample screenshots from the project's user interfaces with Samsung Galaxy S9 viewport:
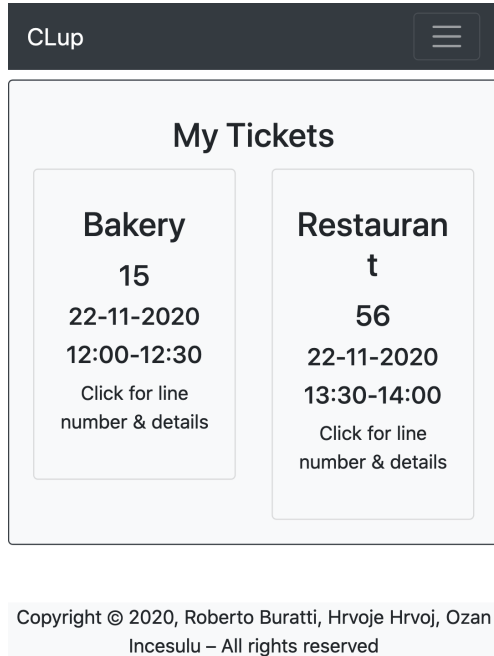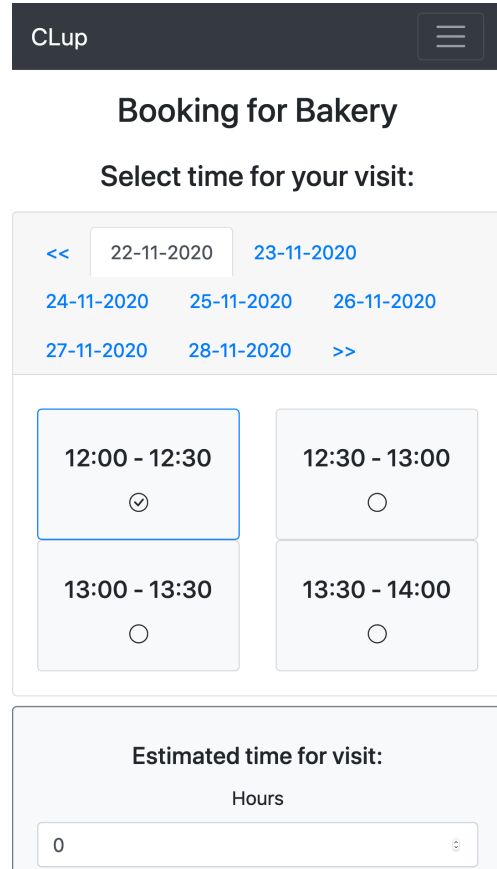

Figure 7: Screenshot: List tickets

Figure 8: Screenshot: Booking a new ticket

Figure 9: Screenshot: Edit location screen of manager
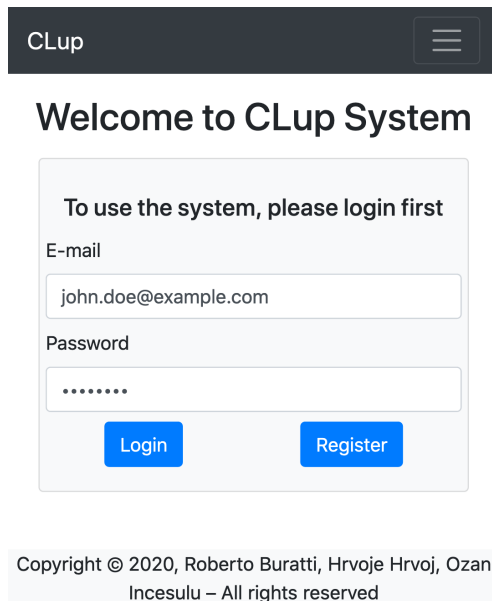
Figure 10: Screenshot: Ticket generate screen of clerk

Figure 11: Screenshot: Login screen

Figure 12: Screenshot: Register screen

3.1.2   Hardware Interfaces

3.1.3   Software Interfaces

3.1.4   Communication Interfaces

3.2   Functional Requirements

3.2.1   Customer
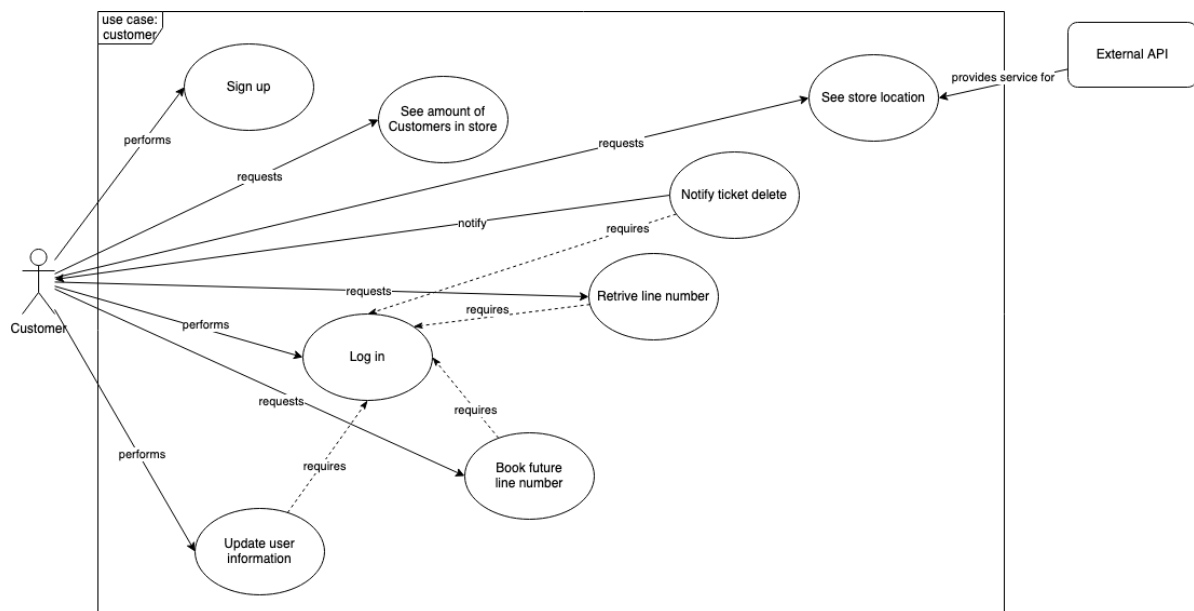
Figure 13: Use Case Diagram for Customer

Use cases

Name	**Book future line number**
Actors	Customer
Entry conditions	The customer is logged in the application and wants to book a visit to the store.
Event flows	• The customer clicks on the "Book a Visit" button in the application. • The application asks the time slot and the estimated time of the visit presenting as default value the average of the previous times of visit of the same user. • The customer sets the time slot and the estimated time of their visit. • The application asks what category of products the customer wants to buy. • The customer set the products categories. • The application requests the line number from the server. • The application generates the QR code on the response from the server.
Exit conditions	The customer has booked a visit for the store.
Exceptions	• The server cannot retrieve the line number since the time slot is full.

Table 2: Use Case: Book future line number
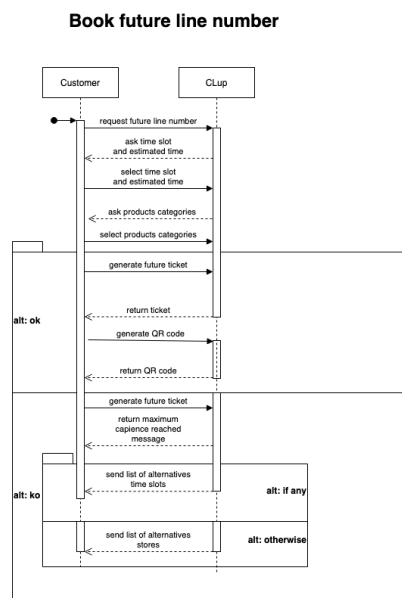
Figure 14: Sequence Diagram for Use Case: Book future line number

Name	**See amount of customers in the store**
Actors	Customer
Entry conditions	The customer is logged in the application and wants to know how many customers are in the store in order to decide whether to book a visit or retrieve a line number
Event flows	• The customer clicks on the "Live Store Info" button. • The application sends a count request to the server. • The server returns the live data for the amount of customers in the store for each time slot.
Exit conditions	The customer knows the amount of customers in the store and can plan their visit.
Exceptions	

Table 3: Use Case: See amount of customers in the store

See amount of customers in store
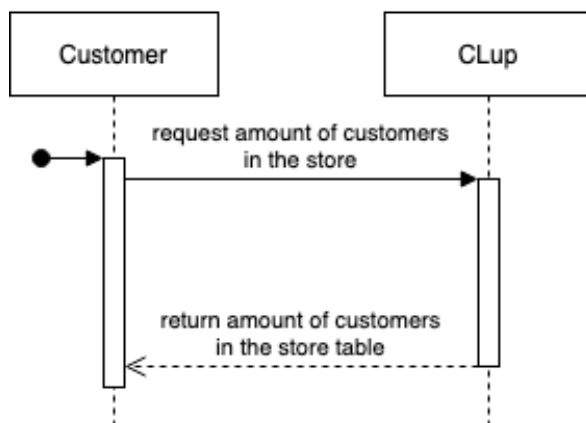
Figure 15: Sequence Diagram for Use Case: See amount of customers in the store

Name	**See store location**
Actors	Customer, Maps API
Entry conditions	The customer needs to know where the store is located and the travel time
Event flows	• The customer clicks on the "Store Location" button. • The application contacts the Maps API with the store location. • The Maps API returns a map from the customer position to the store with a time estimation.
Exit conditions	The customer knows how to go to the store and the travel time needed.
Exceptions	• The customer's smartphone doesn't provide or allow access to location services.

Table 4: Use Case: See store location
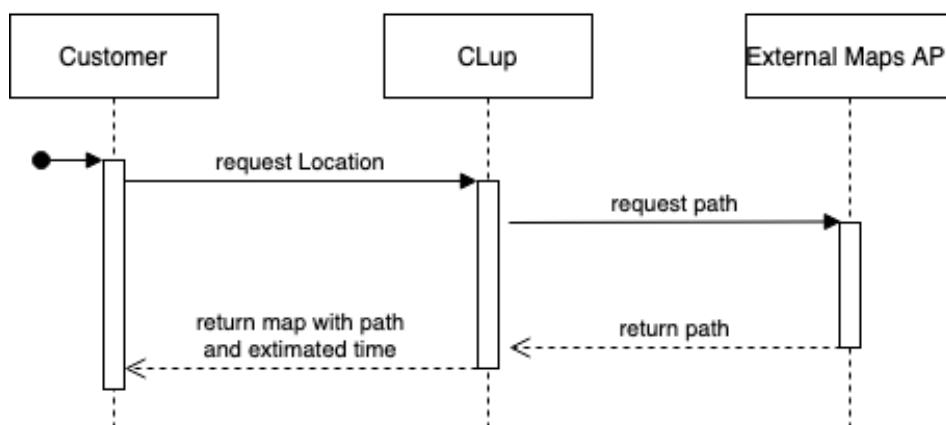
Figure 16: Sequence Diagram for Use Case: See store location

Name	Sign up
Actors	Customer
Entry conditions	The customer opened the application and they are not registered yet, and they want to register.
Event flows	• The customer clicks on the "Sign Up" button. • The customer inserts their credentials. • The application sends the information to the server. • The server stores the data related to the user. • The server returns an acknowledgement to the app.
Exit conditions	The customer is now registered and can use all the functionalities of the app.
Exceptions	• The e-mail provided in the registration form is already used by another user. • The credentials provided in the registration form are ill-formatted.

Table 5: Use Case: Sign up
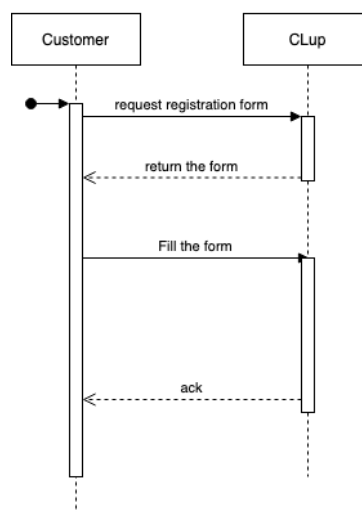
Figure 17: Sequence Diagram for Use Case: Sign up

Name	Login
Actors	User
Entry conditions	The user has already sign up and wants to login to use CLup functionalities
Event flows	• The customer clicks on the "Login" button • The user provides their login credentials. • The system authenticates the user
Exit conditions	The user is logged in and can now use all the CLup functionalities
Exceptions	• The credentials are wrong

Table 6: Use Case: Login
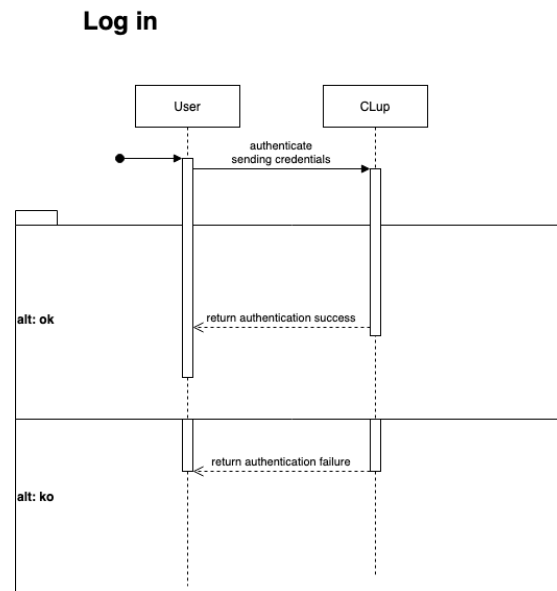
Figure 18: Sequence Diagram for Use Case: Login

Name	Notify ticket delete
Actors	Customer
Entry conditions	A ticket of the Customer was deleted from the server.
Event flows	• The system sends an email to the customer to notify the deletion of their ticket
Exit conditions	The customer is informed about the deletion of their ticket.
Exceptions	•

Table 7: Use Case: Notify ticket delete

Notify ticket delete
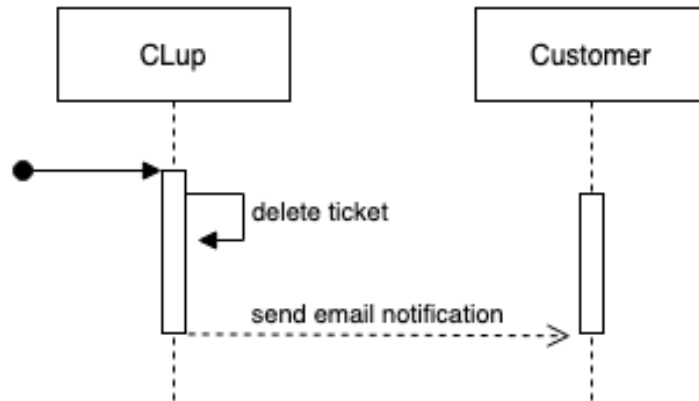
Figure 19: Sequence Diagram for Use Case: Notify ticket delete

Name	Update user information
Actors	User
Entry conditions	The user wants to change an information of themselves.
Event flows	• The user clicks on the "Update User Information" button. • The application shows a form precompiled with user's previous information. • The user changes the information on the respective fields and presses the "Submit" button. • The system saves the changes.
Exit conditions	The user successfully updated their information.
Exceptions	• The new information provided is ill-formatted.

Table 8: Use Case: Update user information

Update user information
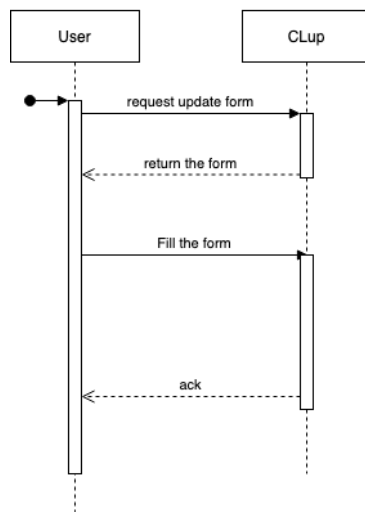
Figure 20: Sequence Diagram for Use Case: Update user information

Name	**Retrieve line number**
Actors	Customer
Entry conditions	The customer is logged in the application and wants to retrieve a line number.
Event flows	• The customer clicks on the "retrieve line number" button in the application • The application returns an ETA and ask confirm to the customer. • The customer confirms. • The customer ask the application to generate the QR code. • The application returns the QR code.
Exit conditions	The customer have retrieved a line number.
Exceptions	• The server cannot retrieve the line number since the time slot is full.

Table 9: Use Case: Retrieve line number

Retrive line number
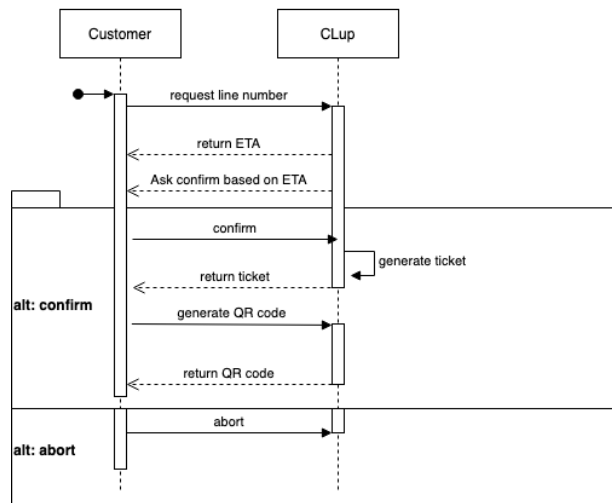
Figure 21: Sequence Diagram for Use Case: Retrieve line number

3.2.2 Clerk
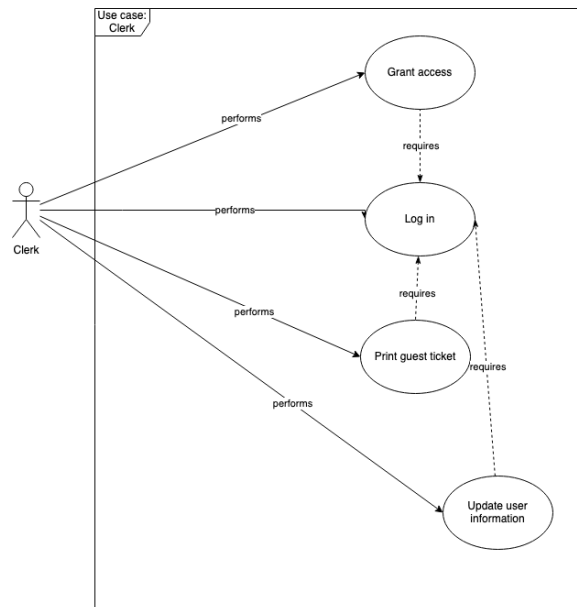
Figure 22: Use Case Diagram for Clerk

Use cases

Name	Grant access
Actors	Clerk, Customer
Entry conditions	The customer has already obtained the QR code and they plan on entering the store.
Event flows	• The Clerk scans the QR code from the customer's smartphone or printed ticket using the application • The application analyzes the QR code and contacts to the Server • The server decides if the customer can enter based on the information received in the QR code • The server responds to the clerk application with the result • The clerk let the customer enter the store
Exit conditions	The customer enters the store
Exceptions	• The server communicates to the clerk that the customer cannot enter because their line number is not available.

Table 10: Use Case: Grant access
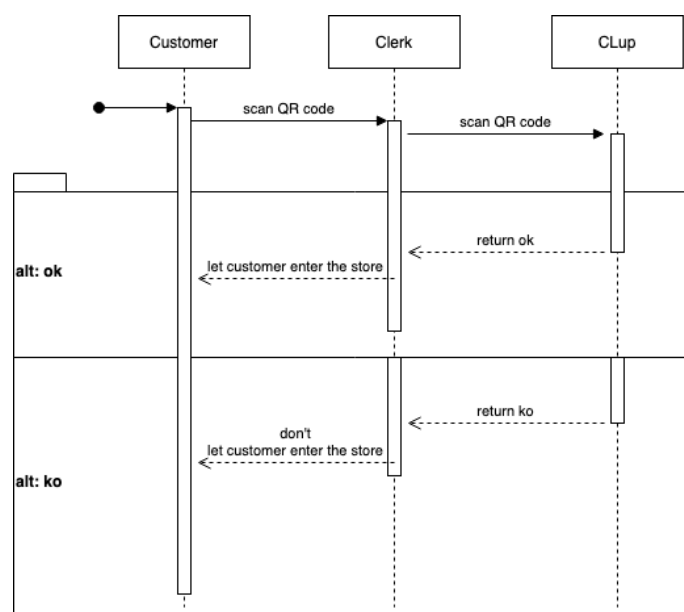
Figure 23: Sequence Diagram for Use Case: Grant Access

Name	**Print guest ticket**
Actors	Clerk, Customer
Entry conditions	The customer has arrived to the location, doesn't have a ticket on their smartphone, and needs a physical ticket to enter.
Event flows	• The customer asks the clerk for a ticket. • The clerk generate a ticket using the app. • The application sends a request to the server to generate a ticket. • The server generates a line number and a ticket. • The server sends the ticket back to the app. • The clerk prints the ticket. • The clerk gives the ticket to the customer.
Exit conditions	The customer has a ticket.
Exceptions	• The server cannot generate a line number due to capacity constraints.

Table 11: Use Case: Print guest ticket
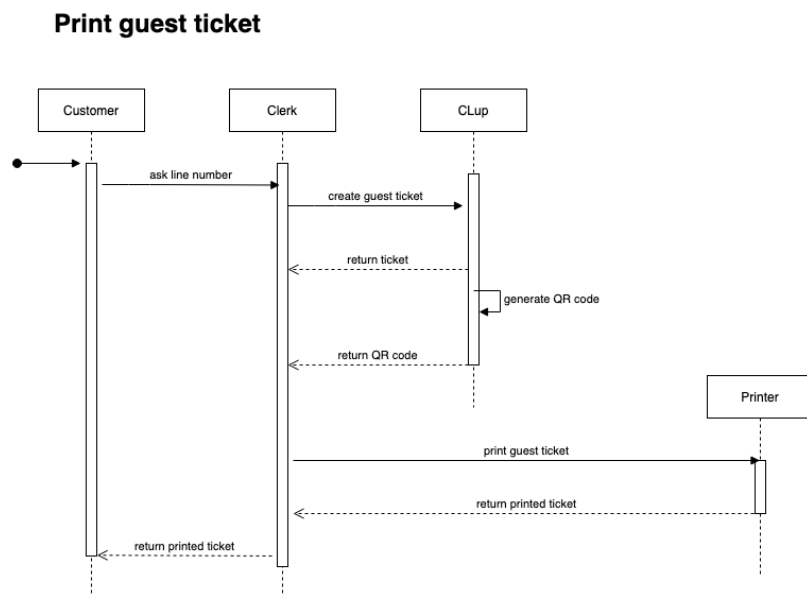
Figure 24: Sequence Diagram for Use Case: Print guest ticket

3.2.3 Manager
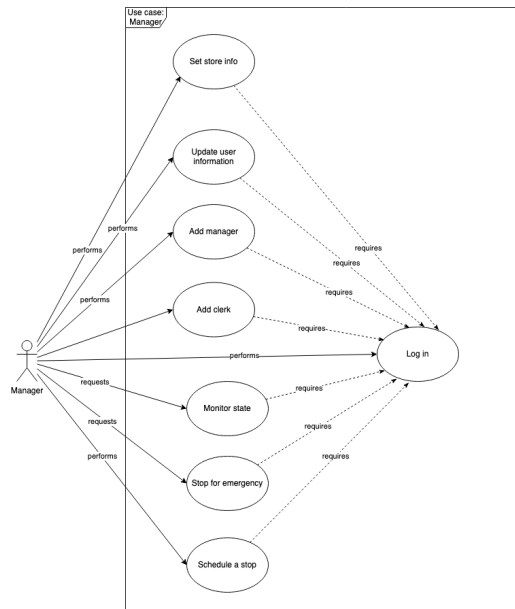
Figure 25: Use Case Diagram for Manager

Use cases

Name	Initialize
Actors	Manager
Entry conditions	The manager of the store needs to set the basic information of the store in order to start the service
Event flows	• The manager clicks on the initialize button • The application shows the location form to the manager • The manager fills the form and sends ıt to the server via the app • The server registers the information and acknowledges
Exit conditions	The system is initialized and can offer all its functions
Exceptions	• Some mandatory parts of the form are not filled.

Table 12: Use Case: Initialize

Set store info
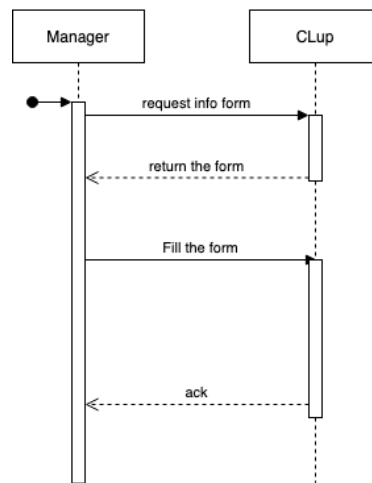
Figure 26: Sequence Diagram for Use Case: Initialize

Name	Monitor state
Actors	Manager
Entry conditions	The manager wants to monitor the number of customers in the store in real time.
Event flows	• The manager clicks on the "Monitoring" button. • The application sends the number request to the server. • The server returns the number of customers in the store. • The application repeats the process periodically as long as the monitoring page is open.
Exit conditions	The manager is informed on the number of the customers in the store in real time.
Exceptions	•

Table 13: Use Case: Monitor state

Monitor state
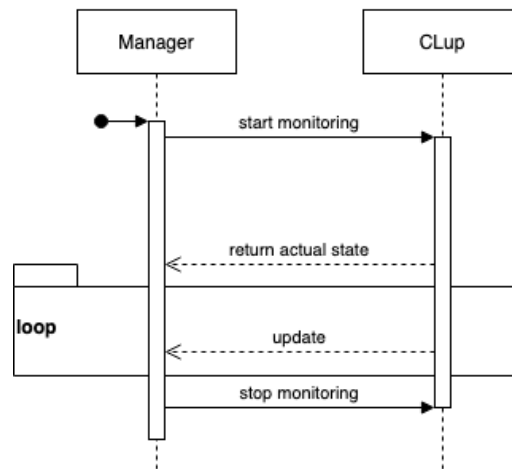
Figure 27: Sequence Diagram for Use Case: Monitor state

Name	**Schedule a stop**
Actors	Manager
Entry conditions	The manager wants to schedule a period of time in which the store will be closed so the users can not book a visit for that time.
Event flows	• The manager clicks on the "Schedule a Stop" button. • The application shows a form for the time of the stop. • The manager fills the form and submits it to the server through the app. • The server stores the information in the database and returns an acknowledgement.
Exit conditions	The system has a scheduled stop stored in its database and will use it to prevent customers from booking a visit in that time period.
Exceptions	• There is already a planned stop in that period.

Table 14: Use Case: Schedule a stop

Schedule a stop
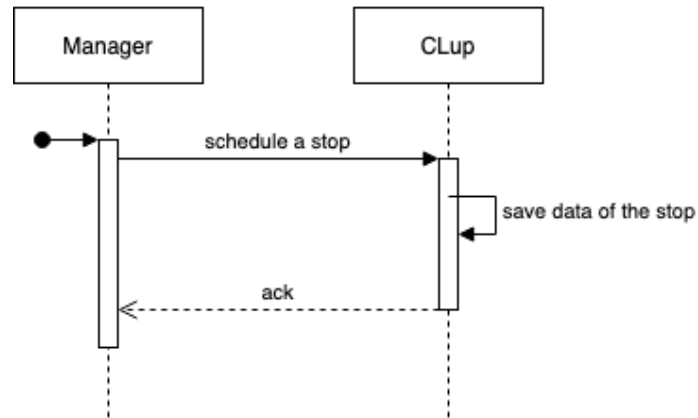
Figure 28: Sequence Diagram for Use Case: Schedule a stop

Name	Stop for emergency
Actors	Manager
Entry conditions	An emergency occurred and the manager wants to immediately stop the system from distributing line numbers.
Event flows	• The manager click on the "Emergency Stop" button • The application asks for confirmation from the manager. • The manager confirms the stop of the system. • The application sends the system stop request to the server. • The server stops the service and return an acknowledgement.
Exit conditions	The system has interrupted the service.
Exceptions	• The manager aborts the operation.

Table 15: Use Case: Stop for emergency
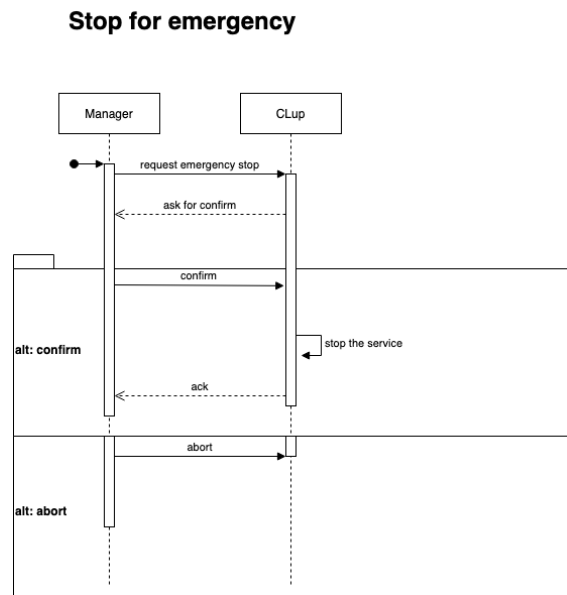
Stop for emergency

Figure 29: Sequence Diagram for Use Case: Stop for emergency

Name	**Add Clerk**
Actors	Manager
Entry conditions	The manager wants to add a new clerk to the system
Event flows	• The manager click on the "add clerk" button • The system asks the data of the new clerk (e.g. the credentials) • The manager inserts the data and press "submit"
Exit conditions	The new clerk is added to the system
Exceptions	• The data of the clerk are incomplete or incorrect

Table 16: Use Case: Add Clerk
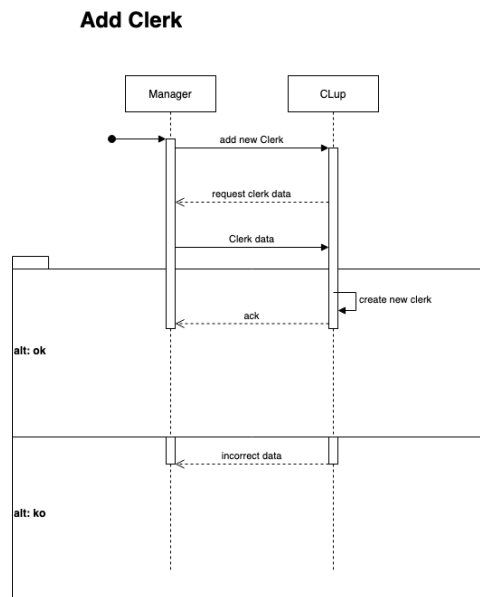
Add Clerk

Figure 30: Sequence Diagram for Use Case: Add Clerk

Name	**Add Manager**
Actors	Manager
Entry conditions	The manager wants to add a new manager to the system
Event flows	• The manager click on the "Add Manager" button • The system asks the data of the new manager (e.g. the credentials) • The manager inserts the data and press "Submit"
Exit conditions	The new Manager is added to the system
Exceptions	• The data of the Manager are incomplete or incorrect

Table 17: Use Case: Add Manager
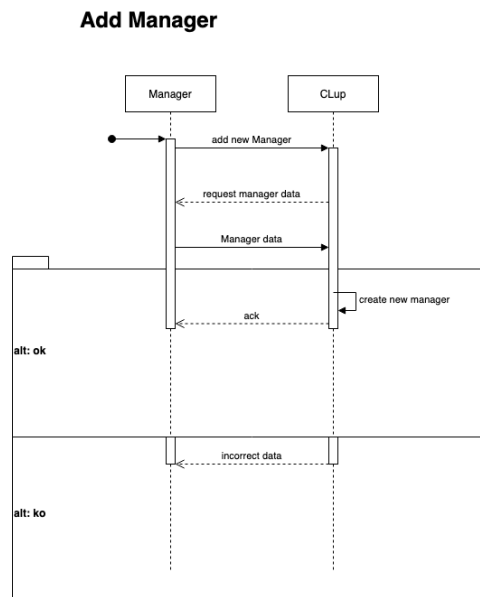
Add Manager

Figure 31: Sequence Diagram for Use Case: Add Manager

3.2.4 Requirements

- R_1 The system must allow users to authenticate using their e-mail address and password.

- R_2 The system must allow customers to register using their e-mail address, their name, surname, phone number, and a new password.

- R_3 Managers must be able to add additional managers and clerks as users.

- R_4 Managers must be able to set and update location-specific information, that is, the maximum number of customers in the location at any given time, opening and closing hours of the store per each day, line number timeout, the limit of reservation per customer on a predetermined time interval that is one of the month, week or day, and location of the place

- R_5 Managers can add any other location as a partner store.

- R_6 Managers can stop the system from issuing any more tickets for a given day

- R_7 Managers can schedule the system stop for a future time.

- R_8 Managers can set the in-shop locations for different categories and product items.

- R_9 In case of a system stop, no further line numbers can be issued for the given time slots.

- R_{10} In case of a system stop, all line numbers in the stop time slots has to be canceled.

- R_{11} The system must cancel those line numbers that the customer did not arrive at the location for more than the set timeout interval.

- R_{12} In case of ticket cancellation, the customer must be notified with an e-mail notification.

- R_{13} Clerks must register the entrance and exit of customers via scanning the QR code for their line number.

- R_{14} Clerks must be able to generate line number tickets in a compatible printer format.

- R_{15} Customers must be able to obtain a line number, except when the system is stopped, or the store is full.

- R_{16} Customers must be able to obtain line numbers for different time slots in the future.

- R_{17} Customers can not obtain line numbers that exceed the quantity per time interval limits.

- R_{18} Customers can not obtain line numbers for time intervals that the system is stopped by a manager.

- R_{19} Customers must be able to see the estimated time available for their line number.

- R_{20} Customers must be able to set or update their phone number, password, name, and surname.

- R_{21} Customers can select a specific product or product categories they plan to visit in the location while obtaining a line number.

- R_{22} Customers can set an estimated time for their visit while obtaining a line number.

- R_{23} Customers must be able to view the shop location

- R_{24} Customers can view the occupation forecasts for the location at different time slots.

- R_{25} Customers can see the alternative suggestions for time slots while obtaining a line number for the future.

- R_{26} Customers can view the occupancy for the partner stores if the preferred time slot is not available while obtaining a line number.

- R_{27} The system must be able to provide a forecast for each location's occupancy for any given time based on past visits.

3.2.5 Summary Table

Goal	Requirements	Domain Assumptions
G_1	$R_1, R_2, R_{11}, R_{15}, R_{17}, R_{18}$	D_1, D_{13}
G_2	$R_1, R_2, R_{11}, R_{16}, R_{17}, R_{18}, R_{26}$	D_1
G_3	$R_1, R_2, R_8, R_{21}, R_{22}$	D_1
G_4	$R_1, R_2, R_{12}, R_{19}, R_{23}, R_{24}, R_{27}$	D_1, D_3, D_6, D_8
G_5	$R_1, R_2, R_5, R_{24}, R_{25}, R_{26}$	D_1
G_6	$R_1, R_3, R_6, R_7, R_9, R_{10}, R_{12}, R_{18}$	D_1, D_{11}
G_7	$R_1, R_3, R_4, R_{11}, R_{17}, R_{27}$	D_1, D_7
G_8	R_1, R_3, R_{13}, R_{20}	$D_1, D_3, D_4, D_5, D_6, D_9, D_{11}, D_{12}, D_{13}$
G_9	R_3, R_{14}	D_2, D_6, D_{10}, D_{13}

Table 18: Summary Table

3.2.6 Tracability Matrix

Requirement	Use case
R_1	Login
R_2	Sign Up
R_3	Add Clerk
	Add Manager
R_4	Initialize
R_5	Initialize
R_6	Stop for Emergency
R_7	Schedule a Stop
R_8	Initialize
R_9	Schedule a Stop
	Stop for Emergency
R_{10}	Schedule a Stop
	Stop for Emergency
R_{11}	Retrieve Line Number
R_{12}	Notify Ticket Delete
R_{13}	Grant Access
R_{14}	Print Guest Ticket
R_{15}	Retrieve Line Number
R_{16}	Book Future Line Number
R_{17}	Book Future Line Number
	Initialize
R_{18}	Book Future Line Number
	Schedule a Stop
	Stop for Emergency
R_{19}	Retrieve Line Number
R_{20}	Update User Information
R_{21}	Book Future Line Number
R_{22}	Book Future Line Number
R_{23}	See Store Location
R_{24}	See Amount of Customers in the Store
R_{25}	Book Future Line Number
R_{26}	Book Future Line Number
R_{27}	See Number of Customers in the Store

Table 19: Tracability Matrix

3.3   Performance Requirements

3.4   Design Constraints

3.4.1   Standards compliance

3.4.2   Hardware limitations

3.4.3   Any other constraint

3.5   Software System Attributes

3.5.1   Reliability

3.5.2   Availability

3.5.3   Security

3.5.4   Maintainability

3.5.5   Portability

4   Formal Analysis Using Alloy

4.1   Alloy introduction

In this section is presented the Alloy representation of some critical aspects the system must present. It is worth highlighting that below only stable states are considered, in which the system may be found. As a consequence, no edge cases are treated.
In particular, the following features are modelled:
- There can't never be more customers in a section then the maximum capacity of that section, in any given moment.
- A booking is possible only by a registered customer.
- Every visit to the store is granted by an access title.

4.2   Alloy code

```
//Signatures-----------------------------------

abstract sig Store {
    sections : some Section
}

one sig PrimaryStore extends Store{
    partnerStores : set PartnerStore
}

sig PartnerStore extends Store {}

sig Section {
    maxCapacity: one Int
}{
    maxCapacity > 0
}

sig Email {}

sig Password {}

abstract sig User {
    email : one Email,
    password : one Password
}

sig Customer extends User {}

some sig Manager extends User {
    store : one Store
}

some sig Clerk extends User {
    manager : one Manager,
```

```
        store : one Store
    }{
        manager.store = store
    }

    abstract sig AccessTitle{
        customer : lone Customer,
        sections: some Section,
        estimatedEnterTime : one Time,
        estimatedExitTime: one Time,
    }{
        estimatedEnterTime.timestamp < estimatedExitTime.timestamp
    }

    sig Ticket extends AccessTitle{
        lineNumber : one ActualLineNumber
    }

    sig Booking extends AccessTitle{
        lineNumber : one BookedLineNumber
    }{
        estimatedEnterTime.timestamp > lineNumber.timeSlot.start.timestamp
        estimatedEnterTime.timestamp < lineNumber.timeSlot.end.timestamp
        #customer = 1
    }

    sig Visit{
        accessTitle : one AccessTitle,
        enterTime : one Time,
        exitTime : lone Time
    }{
        enterTime.timestamp < exitTime.timestamp
    }

    //UTC standard format: number of seconds from 1970/01/01 00:00:00
    sig Time {
        timestamp : one Int
    }{
        timestamp>=0
    }

    sig TimeSlot {
        start : one Time,
        end : one Time
    }{
        start.timestamp < end.timestamp
    }

    abstract sig LineNumber {
        timeSlot : one TimeSlot,
```

```
        number : one Int
    }{
        number >= 0
    }

    sig ActualLineNumber extends LineNumber{}

    sig BookedLineNumber extends LineNumber{}

    //Facts-----------------------------------------

    fact credentialInUsers {
        all mail : Email | mail in User.email
        all pw : Password | pw in User.password
        no mail : Email, disj user1, user2 : User | mail in user1.email && mail
            ↪  in user2.email
        no pw : Password, disj user1, user2 : User | pw in user1.password && pw
            ↪  in user2.password
    }

    fact everySectionInAStore {
        all section : Section | section in Store.sections
        no section : Section, disj store1, store2 : Store | section in store1.
            ↪ sections && section in store2.sections
    }

    fact everyLineNumberInAnAccessTitle {
        all ln : LineNumber | ln in (Ticket.lineNumber + Booking.lineNumber)
        no ln: LineNumber, disj t1, t2 : Ticket | ln in t1.lineNumber && ln in
            ↪ t2.lineNumber
        no ln: LineNumber, disj t1, t2 : Booking | ln in t1.lineNumber && ln in
            ↪  t2.lineNumber
        all disj ln1, ln2 :LineNumber | ln1.number = ln2.number => ln1.timeSlot
            ↪  != ln2.timeSlot
    }

    fact AVisitForEachAccessTitle {
        no at: AccessTitle, disj visit1, visit2 : Visit | at in visit1.
            ↪ accessTitle && at in visit2.accessTitle
    }

    fact everyPartnerSoreInAPrimaryStore {
        all partnerStore : PartnerStore | partnerStore in PrimaryStore.
            ↪ partnerStores
    }

    fact bookOnlyAStore {
        all accessTitle : AccessTitle | one store : Store | accessTitle.
            ↪ sections in store.sections
    }
```

```
fact consistentQueue {
    all disj at1, at2 : AccessTitle |
        ((at1<:Ticket).lineNumber.number < (at2<:Ticket).lineNumber.number
            ↪ && (at1<:Ticket).lineNumber.timeSlot = (at2<:Ticket).
            ↪ lineNumber.timeSlot) =>
            at1.estimatedEnterTime.timestamp =< at2.estimatedEnterTime.
                ↪ timestamp

    all disj at1, at2 : AccessTitle |
        ((at1<:Booking).lineNumber.number < (at2<:Booking).lineNumber.number
            ↪  && (at1<:Booking).lineNumber.timeSlot = (at2<:Booking).
            ↪ lineNumber.timeSlot) =>
            at1.estimatedEnterTime.timestamp =< at2.estimatedEnterTime.
                ↪ timestamp
}

fact noOverBooking {
    all time : Time, section : Section |
        #{at : AccessTitle | at in
            {visit : Visit |
                ((visit.enterTime.timestamp =< time.timestamp && visit.
                    ↪ exitTime.timestamp>= time.timestamp) ||
                (visit.enterTime.timestamp =< time.timestamp && visit.
                    ↪ accessTitle.estimatedExitTime.timestamp>= time.
                    ↪ timestamp) ||
                (visit.accessTitle.estimatedEnterTime.timestamp =< time.
                    ↪ timestamp && visit.accessTitle.estimatedExitTime.
                    ↪ timestamp>= time.timestamp) )&&
                section in visit.accessTitle.sections
            }.accessTitle
        } =< section.maxCapacity
}

fact equallyLongTimeSlots{
all ts1, ts2 : TimeSlot | ts1.end-ts1.start = ts2.end-ts2.start
}

//Assertions------------------------------------

assert noMoreVisitsThanAccessTitle {
    #Visit =< #AccessTitle
}
check noMoreVisitsThanAccessTitle

assert atLeastACustomerForHavingABooking {
    #Customer = 0 => #Booking=0
}
check atLeastACustomerForHavingABooking
```

```
assert neverMoreCustomerThanMaxCapacity {
    all time : Time, section : Section |
        #{c : Customer | c in
            {visit : Visit |
                ((visit.enterTime.timestamp =< time.timestamp && visit.
                    ↪ exitTime.timestamp>= time.timestamp) ||
                (visit.enterTime.timestamp =< time.timestamp && visit.
                    ↪ accessTitle.estimatedExitTime.timestamp>= time.
                    ↪ timestamp) ||
                (visit.accessTitle.estimatedEnterTime.timestamp =< time.
                    ↪ timestamp && visit.accessTitle.estimatedExitTime.
                    ↪ timestamp>= time.timestamp) )&&
                section in visit.accessTitle.sections
            }.accessTitle.customer
        } =< section.maxCapacity
}
check neverMoreCustomerThanMaxCapacity

//Worlds-------------------------------------------

//Normal Condition
pred world1{
    #PartnerStore=1
    #Section=3
    #User=5
    #Customer=3
    #AccessTitle=5
    #Visit=3
    #TimeSlot=5
}

//No Customers
pred world2{
    #Customer=0
}

//Saturated Store
pred world3{
    #PartnerStore=0
    #Section=1
    Section.maxCapacity=3
    #Visit=3
    no cus : Customer | cus not in AccessTitle.customer
    #TimeSlot=1
    #Time = 2
}

run world1 for 10
run world2 for 10
run world3 for 10
```

4.3 World generation

In the following paragraph are displayed examples of worlds generated from the predicates featured in the Alloy implementation.
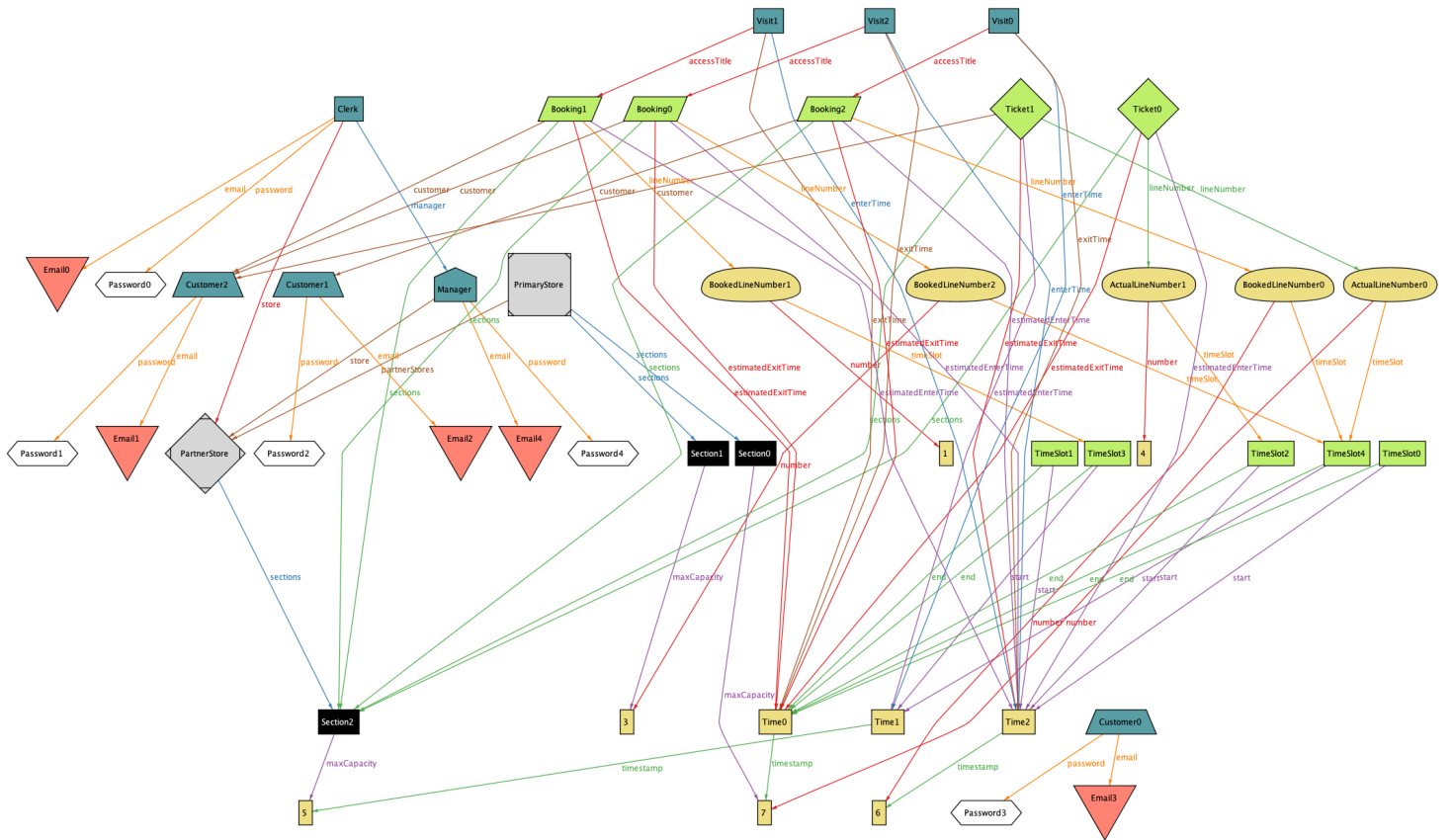

Figure 32: World 1

World 1 describes the model in normal conditions, its only purpose is to show an instance of the model without checking any particular property.
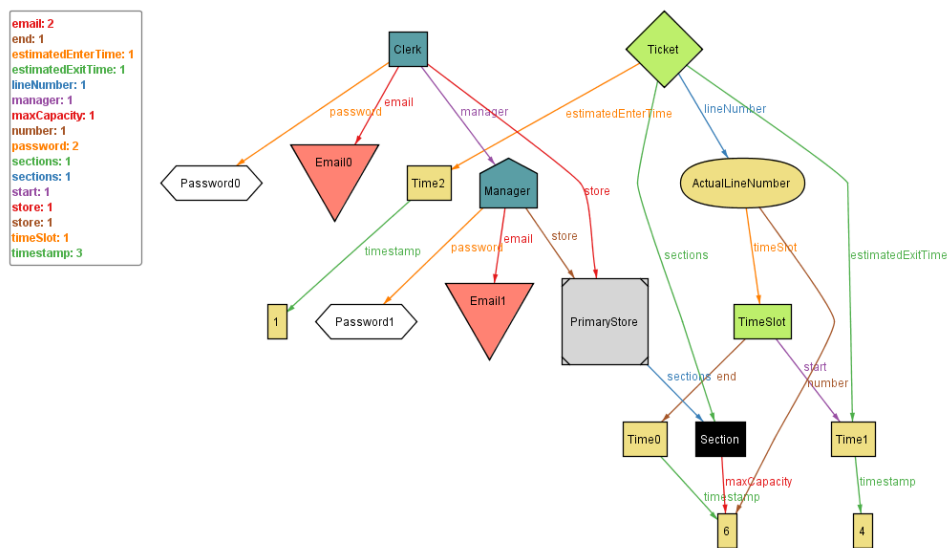
Figure 33: World 2

World 2 describes the model when there are no registered customers, so that there can not be bookings and all the tickets are made as guest going to the clerk in the store.
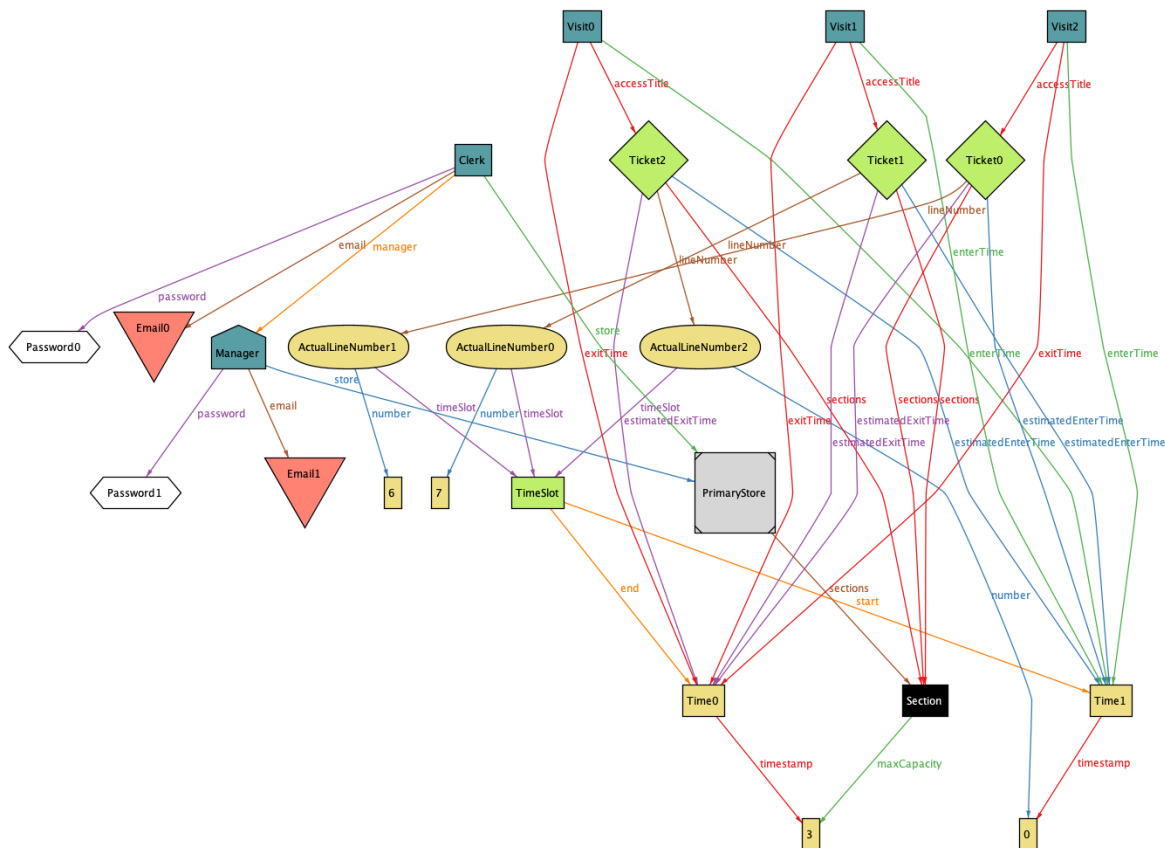

Figure 34: World 3

World 3 describes the model when there is only one section and it has the same number of customers as the maximum capacity.

4.4   Alloy Analyzer results

6 commands were executed.

The results are:

#1: No counterexample found. noMoreVisitsThanAccessTitle may be valid.

#2: No counterexample found. atLeastACustomerForHavingABooking may be valid.

#3: No counterexample found. neverMoreCustomerThanMaxCapacity may be valid.

#4: Instance found.world1 is consistent.

#5: Instance found.world2 is consistent.

#6: Instance found.world3 is consistent.

5   Effort Spent

Date:	Person:	Part:	Time (in hours):	Description:
10/10/2020	Roberto Buratti	Alloy implementation	1	First attempt to implementing alloy model
11/10/2020	Roberto Buratti	Alloy implementation	1	First attempt to implementing alloy model
17/10/2020	Roberto Buratti	Alloy implementation	0.75	First attempt to implementing alloy model
18/10/2020	Ozan Incesulu	General Structure	0.75	Imported and built the general document structure, switched LF -> CRLF for Windows, replaced template parts with names, year and project title
18/10/2020	Ozan Incesulu	Introduction	1.5	Start writing introduction by adding comments for draft goals, adding further subsections, writing basic definition of the system, some abbreviations, definitions, acronyms and references
18/10/2020	Roberto Buratti	Use-case	2	Sequence diagrams
24/10/2020	Ozan Incesulu	Document Structure	1.5	Write the document structure of the introduction, provide some comments and assumptions regarding how other parts of the document shall be structured.
24/10/2020	Roberto Buratti	Use-case	2	Sequence diagrams
25/10/2020	Roberto Buratti	Use-case	2	Sequence diagrams
26/10/2020	Ozan Incesulu	Scope	1.25	Write the Scope section, by defining different users of the system and define different phenomena with the categories they belong. Also add additional definitions.
26/10/2020	Aydin Javadov	Scope	1	Extend the goal test to provide details about system function
29/10/2020	Ozan Incesulu	Introduction	0.25	Add the clerk role to the system instead of using door automation
30/10/2020	Roberto Buratti	Use-case	2	Sequence diagrams
31/10/2020	Ozan Incesulu	Domain Assumptions	1	Write domain assumptions for the system, considering different users and phenomena
31/10/2020	Ozan Incesulu	Requirements	1.5	Write system requirements for the system based on agreed goals of the system.
01/11/2020	Roberto Buratti	Use-case	1	Use case tables
06/11/2020	Ozan Incesulu	Goals	1	Write system goals, merge changes from previous team member, some extra housekeeping tasks.
06/11/2020	Roberto Buratti	Use-case	1	Use case tables
08/11/2020	Roberto Buratti	Use-case	0.75	Use case tables
09/11/2020	Roberto Buratti	Use-case	1.25	Use case diagrams
14/11/2020	Ozan Incesulu	Use-case	1.5	Export diagrams to images, try to align everything, fix grammar and text
14/11/2020	Roberto Buratti	Alloy implementation	1.25	Fixing the previous alloy model
15/11/2020	Roberto Buratti	Alloy implementation	1.5	Fixing the previous alloy model
15/11/2020	Ozan Ince-	Product Per-	3	Create state diagrams, write scenarios and explana-