CLup

Covid Line up

1. INTRODUCTION
2. Purpose

This document has the purpose to clearly define the functionalities that the system-to-be will provide, the goals it strives to achieve, indicate general use cases and describe its limitations as to guide the engineers’ job and the stakeholders’ decision making.

The system tries to put an end to overcrowding inside common spaces and physical queues as much as possible, as to reduce the possibility of getting infected by Covid-19 while doing a daily activity such as grocery shopping. It will incentivize its users to line up virtually to go to said shops and permit to the shop managers to check how many people are inside at any time.

These goals are formally defined as the following:

* + - Store Manager related:

|  |  |
| --- | --- |
| G1 | The store manager should track how many people are inside the store at any point in time. |
| G2 | The store manager should regulate the influx of people that can enter inside the store. |

* + - Store stakeholders and users

|  |  |
| --- | --- |
| G3 | Everybody should be able to maintain social distancing inside the stores. |
| G4 | Everybody should be able to maintain social distancing in front of the stores. |
| G5 | Stores should allow as many customers inside as possible |

* + - Customers

|  |  |
| --- | --- |
| G6 | Anyone who wants to book a visit to any store should decide their desired time to go. |
| G7 | Anyone who wants to go in any store should not need to stand in queue in front of it. |

As subgoals of these last two goals we would have:

Sub-G8: “The ability to know which stores are available to go to, at any moment.”

Sub-G9: “Users that booked a visit through the app should be able to shop without standing in queue in front of the store.”

1. Scope

In the following tables are listed the most relevant world and shared phenomena. In this application, it is critical to consider, among world phenomena, the behaviour of the user, since there are some factors (e.g. W1,W2,W6,W7) that if not considered and handled, can create problems that may lead to unwanted situation, as the creation of a line in front of the store.

1. World Phenomena

|  |  |
| --- | --- |
| **W1** | User reaches the shop |
| **W2** | User lines up in front of the shop |
| **W3** | User enters the shop |
| **W4** | User exits the shop |
| **W5** | User gets into a corridor to get a product |
| **W6** | User takes physical ticket with him |
| **W7** | User loses physical ticket |

1. Shared Phenomena

|  |  |  |
| --- | --- | --- |
|  |  | Controlled by |
| S1 | Virtual User gets ticket through the app | World |
| S2 | Virtual User books a visit through the app | World |
| S3 | Physical User requests ticket from the dispenser | World |
| S4 | Physical User retrieves ticket from the dispenser | Machine |
| S5 | User scans QR code at entrance | World |
| S6 | User scans QR code at exit | World |
| S7 | Turnstile opens | Machine |
| S8 | Turnstile closes | Machine |
| S9 | Virtual User is notified about his coming turn | Machine |
| S10 | Virtual User registers providing requested info | World |
| S11 | Virtual User queries available markets | World |
| S12 | Shop manager queries statistics on shop entrances | World |

1. Definitions, Acronyms, Abbreviations

Physical user: the person who goes directly to the market without using the application.

Virtual user: any person who uses the app to line up virtually and asks for a ticket.

User: Either a physical user or a virtual user.

Ticket: QR code that permits you to enter inside the market at a certain time written on the ticket.

Store, Market, Supermarket, Shop: Any building that provides goods and services in return for money and are connected to the CLup application.

Social distancing: the personal space to any single person as an area of radius 1m.

Front of stores: the area that is property of the store and in which people queue up to wait for their turn.

Authorized Account: Account associated to a Shop Manager, formally authorized through adequate procedures.

Inactive User: a virtual user that does not click on any button in the “Get a ticket” or “Book a visit” page.

1. Revision history

Group meetings:

1st meeting: Defined a very high level of what we want our application to be and what it will provide, by brainstorming scenarios and possible stakeholders’ needs and wants. Duration: 1.5h, 14/10/2020

2nd meeting: Defined scenarios and some key World and Shared Phenomena. We categorized the shared phenomena into World/Machine controlled.

Duration 1.5h, 17/10/2020

3rd meeting: Revised the R&DD document of the last year’s group.

Duration 1.5h, 24/10/2020

4th meeting: Defined Use Cases

Duration 1.5h, 14/11/2020

5th meeting: Defined Functional Requirements, Domain Assumptions and Goals  
Duration 2h, 21/11/2020

6th meeting: Reviewed Goals and defined all the External Interfaces  
Duration 1.5h, 25/11/2020

7th meeting: Reviewed Use Cases and defined new Requirements  
Duration 1.5h, 01/12/2020

8th meeting: Added sections of Alloy code into the document  
Duration 1h 06/12/2020

1. Reference documents
2. Graphic demo of mobile application: <https://customerlineup.bubbleapps.io>
3. ISO/IEEE standards for the engineering of requirements for systems and software products: [https://standards.ieee.org/standard/29148-2018.html](https://standards.ieee.org/standard/29148-2018.html%20)
4. Document Structure

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

1. OVERALL DESCRIPTION
2. Product perspective:
3. Scenarios
4. Hajsen wishes to buy groceries but remembers that the nearest market is small, and he would probably have to wait for an hour. Instead of going downstairs and waiting in line:

* He opens the app on his device and clicks on the button to “Get a ticket”
* He chooses the market he wants to go from a map
* The system shows the first available hour to enter the market
* Hajsen decides to go at that time and clicks on “Confirm”
* The system sends a notification to remind about his appointment, and Hajsen gets ready to go
* He arrives in the market in the assigned time and opens his app again
* He clicks on the “Show ticket” and scans it in the apposite machine
* After he finishes buying the groceries and paying for it, he opens the app and shows the ticket to the cashier
* Now he can exit the market

1. Giulio has just remembered that he promised to her fiancée a special dinner the following day. Since it is too late and he has no time to do the shopping, he decides to book a visit to the nearest supermarket to his home for the following day:

* He opens the app on his device and clicks on the button to “Book a visit”
* He selects the time he would want to go
* He selects the available supermarkets for the chosen time from a map
* The system asks Giulio an estimate of how much time his visit will last and a list of items (or categories of items) he intends to buy. Since Giulio has not a clear idea of what he will purchase and how much the visit will take, he clicks on “Confirm” leaving the two previous fields empty
* The system sends a notification to remind him the visit, so Giulio gets ready to go
* He arrives in the market in the assigned time and opens his app again
* He clicks on the “Show ticket” and scans it in the apposite machine
* After he finishes to do the shopping and paying for it, he opens the app and shows the ticket to the cashier
* He proceeds to exit the market

1. Shalini is the manager of one of the grocery shops of the chain “Ellelunga” and she wants to check on peak times how many people are entering inside the shop

* She opens the application on her device
* She presses on the button log in as manager
* She logs in with her credentials
* On the home page she sees the button “Statistics and Diagnostics” and presses it
* In front of her there are number of effective and expected entrances for the current week

1. Alberto B. hates technology, so he gambles his luck and tries to enter inside the market, unfortunately there are no available places to enter so he takes a ticket from the dispenser

* He gets to the ticket dispenser in front of the shop and presses the button to get a ticket
* Reading the ticket, he sees the time spot in which he can enter the market
* Since the time spot assigned is 2 hours later, he decides to do other things he had to do instead of queuing in front of the shop
* When the time is getting close to the appointment time, he gets back to the shop
* He retrieves the ticket from his pocket and scans it in the apposite machine, that lets him enter the market
* After he finishes buying the groceries and paying for it, he shows the ticket to the cashier
* Now he can exit the market

1. World and Shared phenomena details

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

1. Class Diagram
2. Statecharts
3. Ticket’s status

Tickets are crucial to guarantee the virtual (and physical) lining up mechanism. An *“Available”* ticket becomes *“Booked”* when it is booked either virtually or physically. The user must scan the *“Booked”* ticket within 5 minutes his indicated time slot (“Activated” state) in order to enter the store, otherwise the ticket will expire and become useless. The user, who owns the ticket, after scanning it to enter the store, will use the same ticket to exit. If the user’s ticket becomes unavailable (e.g., the physical ticket is lost, the user’s device has low battery) while he is doing the shopping, the user is allowed to exit the store anyway. In this case, the *“Activated”* ticket will expire after an hour from the indicated time slot and will be considered scanned anyway.

1. Product functions:

The most important aspects of the system to be are the regulation and management of the entrances in stores as seen in the above scenarios.

The system-to-be should inform its users of what stores are available to go to and allow them to take a ticket and line up without needing to be in presence in front of the store.

The system should also be a guidance of help to the store managers by providing information about the number of people who have entered.

1. User Characteristics

Virtual User: A person who has a smartphone or any smart device that can connect to the internet and the application as to virtually line up or book a visit. He will have to show the image of the ticket of his appointment in the right place in the entrance of the store.

Physical User: A person who goes directly to the shop and takes a ticket with the date and the time written on the card, from the dispenser. He will have to show the card of the ticket of his appointment in the right place in the entrance of the store.

Cashier: An employee of the shop who will provide with the correct scanning of the ticket before the exit of a Physical/Virtual User. If the market has a self check-out department, the turnstile will serve as the aforementioned employee.

Manager: An employee of the shop who is interested in checking the number of entrances or exits and in regulating them as needed. He will have the possibility to use our application for the statistics of entrances and exits.

1. Assumptions, dependencies and constraints

|  |  |
| --- | --- |
| A1 | Users will take the shortest path to shops |
| A2 | User which specified a shopping list will spend 90% of their time within that visit in the departments related to the declared list |
| A3 | Data about store departments and items collocation in departments is correct |
| A4 | Users in stores will abide to local norms on social distancing |
| A5 | Only one person will enter the store per ticket |
| A6 | All people who are in front of the store have an appointment to enter the store within the next 15 minutes |

1. SPECIFIC REQUIREMENTS
2. External Interface Requirements:  
   1. User Interfaces:
3. Graphical user interface, application

   Description automatically generatedMap

   Description automatically generatedMobile App, this must be easy to use, as it will have to be used from people of all ages, this means that the interface must be very minimal and direct. The app will allow virtual users to get a ticket, book visits, as well as monitor their tickets and visits, either if the virtual user has an account or not. If user does not login, the app will just consider him as a guest.   
   Here we can see some draft mockups[[1]](#footnote-1) with colours and graphics not optimized:

Shop selection

Home page of the app

Graphical user interface, text, application, chat or text message

Description automatically generatedQr code

Description automatically generatedDiagram

Description automatically generatedThe app will allow user to define first either the shop or the date and time, in the case of “Book a visit”, it is not shown in this demo because it is similar and irrelevant.

book a visit as registered user

book a visit as Guest

Ticket proposal

Ticket Confirmation

In the case of “book a visit”, after the shop selection, the app will allow registered users to select their shopping list.

1. Physical ticket dispensers, that will be installed in front of each shop, acting as proxies for the system. Physical ticket dispensers will allow physical users to get a ticket for the shop to which the dispenser belongs to.
2. Admin interface, that will be a more statistics-oriented panel, accessible through a desktop app (requiring an Authorized Account), that will allow the shop manager to login from an authorized device and monitor entrances and statistics as the average duration of a visit.   
   1. Hardware Interfaces:
3. Virtual Users must have a device that can download and run the app. To use all the functionalities, as the notifications about the traveling time to a shop when the time is close to user’s turn, the device must have GPS turned on.
4. Dispensers must have at least a screen to display the first available time slot for a ticket and to ask for confirmation. There must also be two clearly distinguishable buttons for accepting or declining tickets, as to allow for an easy interaction with the customers, there could be then other types of interface to provide more accessibility.
5. The users of the system-to-be will have to interact with turnstiles gates in entrance and possibly in exit, in special cases documented above. In addition, the turnstiles gates will have a QR code scanner.
6. The cash register will have a QR code scanner for all the users who are about to exit from the store.
   1. Software Interfaces:
7. The system will need to access to some external APIs to access to maps, needed for user localization, user-to-store distance and time estimation and to find stores near a given address.
8. The system will require an interface with a memory storage unit to manage store and ticket data and to allow lookups.
   1. Communication Interfaces:
9. Virtual user devices connect to the system via Internet.
10. Ticket dispensers can connect to system via ethernet cable or by a wireless connection.
11. Shop manager’s device connects to the system via Internet.
12. Functional Requirements:



|  |  |
| --- | --- |
| R1 | **The system shall** allow users to get a ticket with a date and time that shows when to go to a certain store, virtually |
| R2 | **The system shall** allow users to get a ticket with a date and time that shows when to go to a certain store, physically |
| R3 | **The system shall** allow users to book a visit virtually with their desired store, up to the next 7 upcoming days |
| R4 | **The system shall** allow users to look up on a map available registered stores where to go to |
| R5 | **The system shall** ask users how much he or she thinks the trip will last |
| R6 | **The system shall** allow users to be identified by their phone unique ID |
| R7 | **The system shall** allow users to be identified by a username of their choosing |
| R8 | **The system shall** notify the user who has been inactive for 30 seconds while on the confirmation page of booking a ticket, on other available stores he could go to |
| R9 | **The system shall** allow its users to insert information about which categories or items they want to buy |
| R10 | **The system shall** infer how long it will take for customers to buy all products |
| R11 | **The system shall** store data about registered users' habits of expenses |
| R12 | **The system shall** inform users periodically of the (at most) 10 closest stores' available time slots for the day |
| R12.1 | **The system shall** allow its users to select which store(s) to get informed about, by default none |
| R12.2 | **The system shall** allow its users to select which time slots he is interested to get informed about, by default none |
| R12.3 | **The system shall** allow its users to select how often to get notified, by default never |
| R13 | **The system shall** allow the user to scan its QR code in entrance through the turnstiles |
| R14 | **The system shall** allow the user to scan its QR code in exit through the turnstiles or cash register |
| R15.1 | **The system shall** unlock turnstiles after a unique QR code scan in entrance |
| R15.2 | **The system shall** unlock turnstiles after a unique QR code scan in exit |
| R15.3 | **The system shall** lock the turnstiles after a push has occurred |
| R16 | **The system shall** mapthe QR code of a scan to the right user |
| R17 | **The system shall** notify the user when it’s time for his turn to enter the store |
| R18 | **The system shall** manage how many people can enter inside the store for each timeslot |
| R19 | **The system shall** calculate the maximum number of people allowed inside each store |

* 1. Users

There is a subtle difference between the last two functional requirements, in which the last provides the social distance inside stores, so customers might have their personal space while the second to last provides the social distance outside stores, in the sense that people enter divided in sections as to not have crowds outside of the store.

* 1. Store Managers:

|  |  |
| --- | --- |
| R20 | **The system shall** count the number of entrances and exits each day, for each market |
| R21 | **The system shall** store the number of entrances and exits for each market |
| R22 | **The system shall** allow store managers to regulate the number of entrances allowed in the store hourly. |
| R23 | **The system shall** calculate the maximum number of people allowed inside each store as to provide social distancing |
| R24 | **The system shall** allow store managers to see how many customers have entered the store for the past 7 days |
| R25 | **The system shall** allow store managers to register their stores |
| R26 | **The system shall** allow store managers to input store locations |
| R27 | **The system shall** allow store managers to input what sections of goods the store has |
| R28 | **The system shall** allow store managers to input the dimensions of the store |

1. Use Cases
2. Register Account

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | No entry condition |
| Events flow | 1. The user opens the CLup app on his smartphone and clicks on the “Create account” button. 2. The user fills all the mandatory fields. 3. The user clicks on the “Confirm” button. 4. The user receives a notification confirming the registration. |
| Exit condition | Virtual User’s data are saved into the database and the registration terminates successfully |
| Exceptions | 1. The user chooses an already registered username or email 2. The user does not fill one or more mandatory fields 3. The use inserts not valid information in one or more fields   For all the exceptions the system notifies the user that an error occurred. The Events flow starts again from point 2. |

1. Login into Account

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | The virtual user is already into the CLup app homepage |
| Events flow | 1. If the user wants to login as a guest, he directly selects either “Get a ticket” or “Book a visit” button, otherwise he inserts username and password into the “Username” and “Password” fields, respectively. 2. The user clicks on the “Login” button. 3. The system redirects the user to the CLup app homepage |
| Exit condition | The virtual user is successfully redirected to the CLup app homepage |
| Exceptions | 1. The virtual user clicks on the “Login” button but either the username or the password is wrong. The system notifies the user about the error. The Events flow starts again from point 1. |

1. Get Physical Ticket

|  |  |
| --- | --- |
| Actors | Physical User |
| Entry condition | The user is in front of the ticket dispenser of the store where he would want to do the shopping |
| Events flow | 1. The user clicks on the “Get ticket” button 2. The user visualizes through the ticket dispenser’s screen the first available time slot. 3. The user clicks on the “Confirm” button 4. The ticket dispenser prints to the user the physical ticket containing: 5. The provided time slot. 6. The store’s name and address. 7. The QR code to enter (and exit) the store. |
| Exit condition | The user successfully receives the physical ticket. |
| Exceptions | 1. The user refuses the provided time slot clicking on the “Cancel” button.   The ticket is not dispensed. The user either goes away or starts again the procedure to get a physical ticket (Events flow restarts from event 1). |

1. Get Virtual Ticket

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | The virtual user is already logged into CLup app or he wants to access as a guest |
| Events flow | 1. The user selects the “Get a ticket” button. 2. The system redirects the user to a page where he can select the store where he would want to go (from a map). 3. The user selects a store from the map. 4. The system provides the first available time slot for the selected store. 5. The user clicks on the “Confirm” button. 6. The system notifies the user that the procedure has been successfully managed. 7. The system sends to the user the virtual ticket containing: 8. The user’s selected time slot. 9. The store’s name and address. 10. The QR code to enter (and exit) the store. |
| Exit condition | The user successfully receives the virtual ticket. |
| Exceptions | 1. The user refuses the provided time slot clicking on the “Cancel” button.   The system redirects the user to the homepage (Events flow starts again from event 1). |

1. Suggestion Alternative Stores (Get Virtual Ticket)

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | The virtual user is already logged into CLup app or he wants to access as a guest |
| Events flow | 1. The first four events are the same of the “Get Virtual Ticket” use case. 2. The user remains inactive for 30 seconds. 3. The system notifies to the user the possibility to check among a list of further suggested stores. 4. The user clicks on the received notification. 5. The system provides to the user a list with further available stores with an available time slot preceding the already provided one. |
| Exit condition | The user successfully receives the list of further suggested stores. |
| Exceptions | 1. The user clicks on the “Confirm” button without being inactive 30 seconds.   The Events flow proceeds from event 6 of “Get Virtual Ticket” use case.   1. The user does not click on the notification.   The user can only confirm or reject the provided time slot for the selected store (the Events flow proceeds from event 5 of “Get Virtual Ticket” use case). |

1. Book Visit

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | The virtual user is already logged into CLup app or she wants to access as a guest |
| Events flow | 1. The user selects the “Book a visit” button. 2. The system redirects the user to a page where she can select the time slot or the store where she would want to go (from a map) 3. The user selects the time slot, the system provides, through the map, the first 10 available stores closest to the user’s current position. 4. The user selects a store from the map, the system provides the list of available time slots for the next 7 days. 5. The user selects either a store from the map (case a) or a time slot (case b), then she clicks on the “Confirm” button. 6. The system redirects the user to a page where she can indicate the approximate expected duration of the visit, the exact list of items and the categories of items she intends to purchase. 7. The user optionally fills the previous fields, then she clicks on the “Confirm” button. 8. The system notifies the user that the procedure has been successfully managed. 9. The system sends to the user the virtual ticket containing: 10. The user’s selected time slot. 11. The store’s name and address. 12. The QR code to enter (and exit) the store. |
| Exit condition | The user successfully receives the virtual ticket. |
| Exceptions | 1. The user clicks on the “Cancel” button.   The system redirects the user to the homepage (Events flow starts again from event 1).   1. The user’s GPS is unavailable (case 3.a).   The system asks the user to insert an address and provides the 10 available stores closest to the user’s address (Events flow continues from event 4). |

1. Suggestion of Alternative Time Slots (Book visit)

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | The virtual user is already logged into CLup app or she wants to access as a guest |
| Events flow | 1. The first three events are the same of the “Book visit” use case (the user selects a store: case 3.b). 2. The user remains inactive for 30 seconds. 3. The system notifies to the user the possibility to check available time slots of further suggested stores. 4. The user clicks on the received notification. 5. The system provides to the user a list of available time slots of further suggested stores close to selected one. |
| Exit condition | The user successfully receives the list of available time slots. |
| Exceptions | 1. The user selects a provided time slot without being inactive 30 seconds.   The Events flow proceeds from event 5 of “Book visit” use case.   1. The user does not click on the notification.   The user can select only the initial provided time slots (the Events flow proceeds from event 4 of “Book visit” use case). |

1. Suggestion of Alternative Stores (Book visit)

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | The virtual user is already logged into CLup app or she wants to access as a guest |
| Events flow | 1. The first three events are the same of the “Book visit” use case (the user selects a time slot: case 3.a). 2. The user remains inactive for 30 seconds. 3. The system notifies to the user the possibility to lookup further available stores for the selected time slot. 4. The user clicks on the received notification. 5. The system provides to the user a list with further available stores. |
| Exit condition | The user successfully receives the list of further available stores. |
| Exceptions | 1. The user selects a provided store without being inactive 30 seconds.   The Events flow proceeds from event 5 of “Book visit” use case.   1. The user does not click on the notification.   The user can select only the initial provided stores (the Events flow proceeds from event 4 of “Book visit” use case). |

1. Enter/Exit Store

|  |  |
| --- | --- |
| Actors | User, Cashier |
| Entry condition | The user is in front of the store indicated in his ticket |
| Events flow | 1. The user scans his QR code through the QR code reader of the turnstile in entrance. 2. The user enters the store. 3. The user does the shopping. 4. The user pays at the cash register. 5. The cashier scans the QR code of the user’s ticket though the QR code reader of the cashier desk 6. The user exits the store |
| Exit condition | The user successfully enters/exits the store. |
| Exceptions | 1. The user arrives to the store 5 minutes or more after the time slot indicated in his ticket.   The user either gets a physical ticket (see “Get physical ticket” use case) or goes away.   1. The user does not buy anything.   The user exits the store scanning his QR code through the QR code reader of the turnstile in exit.   1. After the user enters the store, his physical ticket is lost or his virtual ticket becomes unavailable.   The user exits the store without scanning the QR code. Moreover, the ticket is considered scanned two hours after the time slot indicated in the user’s ticket |

1. Periodic Notification of Time Slots

|  |  |
| --- | --- |
| Actors | Virtual User |
| Entry condition | The virtual user is already logged into CLup app or she wants to access as a guest |
| Events flow | 1. The user accesses the notification panel. 2. The user selects one or more stores she wants to be notified about. 3. The user selects a day or a time range. 4. The user selects how often she wants to be notified. 5. The user clicks on the “Confirm” button. 6. The system notifies the user that the procedure has been successfully completed. |
| Exit condition | The user’s notification preferences are correctly updated. |
| Exceptions |  |

1. Monitor Entrances

|  |  |
| --- | --- |
| Actors | Store Manager |
| Entry condition | The store manager access CLup app through an authorized account. |
| Events flow | 1. The store manager selects a store between those he manages. 2. The system redirects the store manager to a page where he can: 3. See the statistics about entrances of the selected store. 4. Regulate the influx of people entering the store by setting a parameter. |
| Exit condition | The store manager can see the statistics and regulate the influx of people entering the selected store |
| Exceptions |  |

1. Sequence Diagrams
2. *![Immagine che contiene tavolo

   Descrizione generata automaticamente]()*Register Account
3. **Login into Account
4. **Get Virtual Ticket
5. **Book Visit
6. Performance Requirements:

On the basis of studies done by ISTAT on the Italian number of population and by the US government on the habits of customers of grocery shops (references in point 6 of document), we extract some main points needed on the calculation of the performance requirements. These main points are:

* + 1. There are around 750 thousand families in Milan only.
    2. There are around 25 million families in Italy.
    3. On average Saturday sees more families go grocery shopping (from 10% to 40% more)
       1. In the US from 29-30 million go on a weekday, while on the weekend 33-41 million families go grocery shopping
    4. The average family goes grocery shopping 1.6 times a week as of 2019

Based on this information we can deduce that on the worst case the system to be will have to manage on peak hours 360 thousand users contemporarily in one hour in Italy.

Since it is important to have fast responses to users’ requests, a response time to any request, comprising those of the external API’s should be under 0.5 seconds.

1. Design Constraints:  
   1. Software Compliance:

The software will comply with local laws regarding COVID-19 and its safety related regulations.

Furthermore, the software will comply with local laws about data treatment and usage such as the EU General Data Protection Regulation [GDPR 2016/679](https://eur-lex.europa.eu/legal-content/EN/AUTO/?uri=CELEX:02016R0679-20160504&qid=1532348683434).

* 1. Hardware Limitations:

There are inherent hardware limitations in which the system will find itself.

It is important for the software to be designed in a way that keeps count of its portability. This way it will need to be run on computationally weak (old) devices too.

1. Software System Attributes:  
   1. Reliability:

The system will have to handle up to 10% more requests with respect to the worst-case scenario defined in the Performance Requirements section.

* 1. Availability:

Because of the system’s importance on the provision of social distancing for one of the most crucial aspects of our lives, it will have to be up for 99.9% of the time. (Downtime: less than 9 hours)

* 1. Security:

Security is important for the sole reason of the registered users’ passwords that will have to be stored. Since these passwords might be shared with other applications or programs with more sensitive data, the passwords should be encrypted with up-to-date technologies. All data should be stored in compliance with GDPR’s regulation.

* 1. Maintainability:

The system-to-be has a strong need of maintainability and extendibility because of its nature. The core system will have to adapt to many different countries’ laws so the most important aspect will be its reusability of logical components such as the booking aspect or the queue managing.

* 1. Portability:

Since the goal is to make it easy for all people to use our system-to-be, it must be operatable in different operating system environments.

1. FORMAL ANALYSIS USING ALLOY

The following alloy formal analysis has the goal to prove the correct functionality and the formal proof of some of the defined goals for the proposed system. We focus on:

* (G3)The goal of user’s safety inside store, modelled as the ability of the system to grant that the maximum number of people inside the store in never exceeded nor tickets are given when the maximum is reached, and given the assumption that people will behave with respect to the local norms (A5).
* (G6, G7, G8)The goals of the users about being able to get tickets and visits in available time slots, so that they do not need to line up in front of the store. These goals are modelled through predicates, showing the correct working of the system, that can give tickets to users both physically through a dispenser and virtually through their devices. Moreover, the advanced functionality of booking a visit is modelled too, with the user choosing both the date and the time slot. The system is shown to work correctly giving tickets only for available time slots.
* (G1, G2) The goals about the shop manager being able to track and regulate the influx of people in the store. This is modelled through a ticket system, that allows to monitor and regulate the maximum number of people admitted inside the store at each timeslot and the current general situation about tickets.
* The soundness of the proposed system, showing that users cannot shop without having their entrance monitored, thus they cannot shop without a ticket or a visit booked. This is modelled with a predicate. The model also considers possible particular cases as the fact that users could lose their ticket after the entrance, with the system recognizing this and correctly considering concluded such a ticket after the end of its time slot.

1. Signatures

sig Market{

dispensers: some TicketDispenser,

cashDesks: some CashDesk,

entrances: some QRCodeReader, //Representing the entrance of the market, with turnstiles and relative QRCode reader

freeExit: some QRCodeReader, //Representing the exit for users that did decide to not buy or that did not find what they wanted

manager: one StoreManager,

departments: set Department,

ticketManager : one TicketSystem

}{

freeExit & entrances = none

freeExit & cashDesks.reader = none

entrances & cashDesks.reader = none

}

sig TicketSystem{

maxUsersPerSlot: one Int,

insideMarket: set Ticket,

line: set Ticket,

pastTickets: set Ticket,

currentTimeSlot: one TimeSlot,

currentDate: one Date,

availableSlots: Date -> TimeSlot

}{ //no ticket is in more different states at the same time

#(insideMarket & line)=0

#(insideMarket & pastTickets)=0

#(line & pastTickets)=0

//all tickets insideMarket are tickets booked for the current Date and Time Slot

all t: Ticket | t in insideMarket implies (t.ticketDate = currentDate and t.ticketTimeSlot = currentTimeSlot)

//Available Dates and Time Slots are the ones for which the number of tickets is less than the max number admitted to grant safety and that

// are not Dates or Slots different from the current for which there are past tickets (No available Slots in past dates)

all d:Date, ts:TimeSlot | (d->ts) in availableSlots iff (#((insideMarket + line) & ticketDate.d & ticketTimeSlot.ts) < maxUsersPerSlot and

no t: Ticket | t in pastTickets and (t.ticketDate != currentDate and d=t.ticketDate or t.ticketTimeSlot != currentTimeSlot and d=t.ticketDate and ts=t.ticketTimeSlot ))

maxUsersPerSlot > 0

}

sig TicketDispenser{

distributedTickets: set PhysicalTicket

}

sig QRCode{}

sig QRCodeReader{

scanned: set QRCode

}

sig CashDesk{

reader: one QRCodeReader

}

sig StoreManager {}

abstract sig SafetyStatus{}

one sig Safe extends SafetyStatus{}

one sig UnSafe extends SafetyStatus{}

abstract sig User {

userStatus: one SafetyStatus

}

sig VirtualUser extends User{

userVTickets: set VirtualTicket,

userVisits: set Visit,

userDevice: some SmartDevice

}

sig PhysicalUser extends User{

userPTickets: set PhysicalTicket

}

sig Date{}

sig TimeSlot{}

abstract sig Ticket{

ticketDate: one Date,

ticketTimeSlot: one TimeSlot,

ticketCode: one QRCode

}

sig PhysicalTicket extends Ticket{}

sig VirtualTicket extends Ticket{}

sig Visit extends Ticket{

shoppingListCategories : set Category,

shoppingListItems : set Item

}

sig Item{}

sig Category{

products: some Item

}

sig Department{

categories: some Category

}

sig SmartDevice{

localizationDevice: lone GPS

}

sig GPS{}

1. Utilities

//gets all the qr code readers that scanned the given ticket

fun scannedBy[t: Ticket] : set QRCodeReader {

scanned.(t.ticketCode)

}

//gets all tickets managed by the given ticket system

fun getAllTickets[ts: TicketSystem] : set Ticket {

ts.insideMarket + ts.line + ts.pastTickets

}

//gets all the visits and tickets of the given virtual user

fun getVUserTicketsAndVisits[u: VirtualUser] : set Ticket {

u.(userVTickets + userVisits)

}

//gets the ticket that manages the given ticket

fun getTicketManager[t: Ticket] : one TicketSystem{

(line + insideMarket + pastTickets).t

}

////checks if the two markets are the result of a new ticket insertion

pred checkEqualStateMarketPlusNewTicket[t: Ticket, m,m1 : Market]{

m1.ticketManager.(pastTickets + insideMarket) = m.ticketManager.(pastTickets + insideMarket) and

m1.ticketManager.line= m.ticketManager.line + t and

m1.dispensers=m.dispensers and m1.cashDesks=m.cashDesks and m1.entrances=m.entrances and

m1.freeExit=m.freeExit and m1.manager=m.manager and m1.departments=m.departments

}

//checks if the two users are the result of a new virtual ticket insertion

pred checkEqualStateVUserPlusNewTicket[t: VirtualTicket, u,u1: VirtualUser]{

u1.userVTickets=u.userVTickets + t and u1.userVisits = u.userVisits and u1.userDevice = u.userDevice

}

//checks if the two users are the result of a new visit insertion

pred checkEqualStateVUserPlusNewVisit[v: Visit, u,u1: VirtualUser]{

u1.userVTickets=u.userVTickets and u1.userVisits = u.userVisits + v and u1.userDevice = u.userDevice

}

//checks if the two users are the result of a new physical ticket insertion

pred checkEqualStatePUserPlusNewTicket[t: PhysicalTicket, u,u1: PhysicalUser]{

u1.userPTickets=u.userPTickets + t

}

1. Facts

//No market gives more tickets per time slot than the max allowed by the shop manager

fact{

all m: Market, d: Date, t: TimeSlot |

let marketTickets= getAllTickets[m.ticketManager] |

#(marketTickets & ticketDate.d & ticketTimeSlot.t) <= m.ticketManager.maxUsersPerSlot

}

//definition of user safety

fact{

all vu: VirtualUser | vu.userStatus = Safe iff

( no t: Ticket, ts: TicketSystem | t != none and t in (vu.userVTickets + vu.userVisits) and

t in ts.insideMarket and #(ts.insideMarket) > ts.maxUsersPerSlot )

all pu: PhysicalUser | pu.userStatus = Safe iff

( no t: Ticket, ts: TicketSystem | t != none and t in pu.userPTickets and

t in ts.insideMarket and #(ts.insideMarket) > ts.maxUsersPerSlot )

}

//each ticket can be scanned at most 2 times: 2 times is a past ticket that has entered and exited, 1 time is only entered, 0 times is in line

fact {

all t:Ticket |

#scannedBy[t] <= 2 and

//ticket is past if it has been scanned 2 times, one at entrance and one at exit

( (#scannedBy[t] = 2 ) implies ( let ts = getTicketManager[t] | t in ts.pastTickets and #(scannedBy[t] & (ticketManager.ts).entrances) = 1 and #(scannedBy[t] & ((ticketManager.ts).freeExit + (ticketManager.ts).cashDesks.reader)) = 1)) and

//ticket is inside if it has been scanned 1 time at entrance

(#scannedBy[t] = 1 iff let ts = getTicketManager[t] | t in ts.insideMarket and #(scannedBy[t] & (ticketManager.ts).entrances) = 1 ) and

//ticket is in line if it has been scanned 0 times and it is not referring to a past Date, it the latter is true then the ticket is past (ticket invalidated because of user not presenting)

( #scannedBy[t] = 0 implies ( (no t1: Ticket | t1 != t and getTicketManager[t1] = getTicketManager[t] and t1 in getTicketManager[t1].pastTickets and #scannedBy[t1] = 2 and

(t1.ticketDate != getTicketManager[t1].currentDate and t.ticketDate = t1.ticketDate or

t1.ticketTimeSlot != getTicketManager[t1].currentTimeSlot and t.ticketDate = t1.ticketDate and t.ticketTimeSlot=t1.ticketTimeSlot )) implies

t in getTicketManager[t].line else t in getTicketManager[t].pastTickets))

}

//each physical ticket is generated by one dispenser belonging to the ticket's market

fact{

all pt: PhysicalTicket | one d: TicketDispenser | pt in d.distributedTickets and dispensers.d = ticketManager.((line + insideMarket + pastTickets).pt)

}

//no same Ticket in two different markets' TicketSystem, in any of the states (inside, past, in line)

fact {

no disj ts1,ts2: TicketSystem |

#( getAllTickets[ts1] & getAllTickets[ts2]) != 0

}

//no same dispenser/cash desks/departments/ticketManager for disjoint markets

fact {

no disj m1,m2: Market |

#(m1.dispensers & m2. dispensers + m1.cashDesks & m2.cashDesks +

m1.departments & m2.departments + m1.ticketManager & m2.ticketManager) != 0

}

//each ticket has a different QRCode

fact {

no disj t1,t2: Ticket |

#(t1.ticketCode & t2.ticketCode)!= 0

}

//each category is associated to a Department in each Market

fact{

all disj d1,d2: Department | departments.d1 = departments.d2 implies no c: Category | c in d1.categories and c in d2.categories

}

//different markets have no same QRcodeReader

fact{

all disj m1,m2: Market | (m1.freeExit + m1.entrances + m1.cashDesks.reader) & (m2.freeExit + m2.entrances + m2.cashDesks.reader) = none

}

//each Device has a different GPS device

fact {

all disj d1,d2: SmartDevice |

(d1.localizationDevice != none and d2.localizationDevice != none) implies d1.localizationDevice != d2.localizationDevice

}

//each ticket system is associated to one and only one market

fact{

all t: TicketSystem |

#(ticketManager.t) = 1

}

//eachQRCodeReader is associated to a desk, an entrance or an exit

fact {

all qr: QRCodeReader | #(freeExit.qr + entrances.qr + reader.qr) = 1

}

//each QRCode is associated to one ticket

fact{

all c: QRCode | #(ticketCode.c) = 1

}

//each ticket dispenser is associated to one market

fact{

all td: TicketDispenser | #(dispensers.td) = 1

}

//each cashDesk is associated to one market

fact{

all cd: CashDesk | #(cashDesks.cd) = 1

}

//each department is associated to one market

fact{

all d: Department | #(departments.d) = 1

}

//each item is associated to one category

fact{

all i: Item | #(products.i) = 1

}

//each GPS is associated to a SmartDevice

fact {

all g: GPS | #(localizationDevice.g) =1

}

//each SmartDevice is associated to a Virtual User

fact {

all sd: SmartDevice | #(userDevice.sd) =1

}

//each category is associated at least to one department

fact{

all c: Category | #categories.c >= 1

}

//each ticket of any type belongs to a correct type of user

fact{

all vt: VirtualTicket | #(userVTickets.vt) = 1

all pt: PhysicalTicket | #(userPTickets.pt) = 1

all vs: Visit | #(userVisits.vs) = 1

}

1. Dynamic Model

//G3. User inside stores are safe

assert allUsersInsideStoresAreSafe{

all u: User | u.userStatus= Safe

}

//G7. If there are available time slots in the current date, user can get a ticket and be put in line

pred vUserGetsTicket[m,m1: Market, u,u1: VirtualUser]{

some ts: TimeSlot | ts in m.ticketManager.currentDate.(m.ticketManager.availableSlots) and

some t: Ticket | t.ticketDate = m.ticketManager.currentDate and t.ticketTimeSlot = ts and

t in m1.ticketManager.line and

checkEqualStateMarketPlusNewTicket[t, m,m1] and

checkEqualStateVUserPlusNewTicket[t, u,u1]

}

//G8. If there are available time slots in the current date, physical user can get a ticket from the dispenser

pred pUserGetsTicket[m,m1: Market, u,u1: PhysicalUser]{

some ts: TimeSlot | ts in m.ticketManager.currentDate.(m.ticketManager.availableSlots) and

some t: Ticket | t.ticketDate = m.ticketManager.currentDate and t.ticketTimeSlot = ts and

t in m1.ticketManager.line and m1.dispensers.distributedTickets =m.dispensers.distributedTickets + t and

checkEqualStateMarketPlusNewTicket[t, m,m1] and

checkEqualStatePUserPlusNewTicket[t, u,u1]

}

//G6. If the selected date and time slot are available, virtual user can book a visit in that date and time slot

pred vUserBooksVisit[m,m1: Market, u,u1: VirtualUser,ts: TimeSlot, d: Date]{

(d->ts) in m.ticketManager.availableSlots and

some v: Visit | v.ticketDate = d and v.ticketTimeSlot = ts and

checkEqualStateVUserPlusNewVisit[v,u,u1] and checkEqualStateMarketPlusNewTicket[v,m,m1]

}

//User cannot shop whitout having registered their entrance scanning QRCode

assert noExitWithoutEntering{

no t: Ticket | (one qrReader: QRCodeReader | t.ticketCode in qrReader.scanned and qrReader in (Market.freeExit + Market.cashDesks.reader)) and

no qrReader: QRCodeReader | t.ticketCode in qrReader.scanned and qrReader in Market.entrances

}

1. Results

run vUserGetsTicket for 5

run vUserBooksVisit for 5

run pUserGetsTicket for 5

check noExitWithoutEntering for 5

check allUsersInsideStoresAreSafe for 5

StoreManager,exactly 1 TicketDispenser,exactly 1 CashDesk,exactly 2 VirtualUser,exactly 1 PhysicalUser, exactly 4 Ticket

*Text

Description automatically generated*

1. Generated World

Here we report a simple and a more complex world generated by the alloy model.

*Diagram

Description automatically generated*The simple one is easy to understand, and highlights the properties mentioned in paragraph 4.1, it is clear the association between tickets and users, either physical or virtual, and the correct modelling of the situation of the market (only one market reported for ease of reading).

**The model can also generate more complex and multi-market worlds, of more difficult reading, but still correct. Here is reported one example:

1. EFFORT SPENT
2. REFERENCES

* How often families go grocery shopping per week : <https://www.statista.com/statistics/251728/weekly-number-of-us-grocery-shopping-trips-per-household/>
* How many families are there in Milan: <https://www.tuttitalia.it/lombardia/18-milano/statistiche/popolazione-andamento-demografico/>
* Which day is the busiest for grocery shopping: <https://www.forbes.com/sites/joanverdon/2020/04/08/best-time-for-grocery-shopping-tips-from-in-store-traffic-data/#:~:text=The%20busiest%20days%3F,box%20stores%2C%E2%80%9D%20he%20said.>
* How many families go grocery shopping on weekends in respect to weekdays: <https://www.insider.com/best-time-to-go-grocery-shopping-2018-3>
* How many families are there in Italy: <https://www.istat.it/it/files/2018/12/C03.pdf>

1. Mockups are taken from an interactive graphic demo with graphic-driving purposes only, linked in the “reference documents” paragraph. [↑](#footnote-ref-1)