



POLITECNICO

MILANO 1863

ITD

Implementation and Test deliverable

Authors: Gabriele DIGREGORIO
Enrico MASSARO
Vanessa TAMMA

Version: 0.7
Date: 03 February 2021
Professor: Elisabetta Di Nitto

CONTENTS

1	INTRODUCTION	3
1.1	Purpose.....	3
1.2	Scope.....	3
1.3	Definitions, Acronyms, Abbreviations	4
1.3.1	Definitions	4
1.3.2	Acronyms	4
1.3.3	Abbreviations.....	4
1.4	Revision history.....	4
1.5	Reference Documents	5
1.6	Document Structure.....	5
2	Implemented Requirement	6
2.1	Overview	6
2.2	Traceability.....	6
3	DEVELOPMENT DETAILS	8
3.1	Adopted Development Frameworks	8
3.2	Development Solutions.....	8
4	SOURCE CODE STRUCTURE.....	10
5	TEST	11
5.1	Integration Test.....	11
5.1.1	User Test.....	11
5.2	Shop	15
5.1.2	Shop Test	15
5.2	Functional System Test.....	17
5.2	Performance System Test	18
6	INSTALLATION INSTRUCTIONS	18
7	EFFORT SPENT	19
8	REFERENCES.....	21

1 INTRODUCTION

1.1 Purpose

The purpose of the project CLup (Customer Line-up) is to develop a digital system of lining up that saves people from having to stand outside of stores for hours, avoids crowds inside the store, and, more in general, allows to regulate the influx of people in the stores.

The idea is to create a digital version of the traditional mechanism of lining up that is easy to use by everyone. In this way, the system would help to deal with the strict rules imposed by the government due to the global pandemic.

The system should give customers the possibility to line up from their home and approach to store only when their number is close to being called. This mechanism should avoid the situation in which the customers wait for their shift in the proximity of the store that is not an acceptable scenario in a lockdown situation.

1.2 Scope

The software should represent a digital alternative to the situation in which people retrieve a physical number that gives their position in the queue when they want to enter a store.

CLup should provide three main features:

- **Lining up:** allows customers to line up from their homes and avoids crowds outside the stores. It should include a notification system that alerts people when their number is close to being called. These alerts should consider the time customers need to get to the shop from the place they currently are and should be based on precise estimation of the waiting time. Moreover, CLup must provide effective fallback options for people who do not have access to the required technology.
- **Booking:** allows customers to book a visit to the supermarket. Since the time that it takes to visit a supermarket is not uniform, the system should give to user the possibility to specify an estimation of the duration of the visit. Alternatively, it might infer this information analysing the previous visits, if any.
- **Suggestion:** suggests different time slots for visiting the store (also on different days) to deal better with the restriction in the number of people inside the store. Alternatively, the system should propose other available supermarkets to the customers and alerts them in case a new time slot becomes available (e.g. after the deleting of a booking by another customer).

The customer that wants to use the service must be registered. Thanks to this, the system would be able to track the lining up, the booking, and the duration of the previous visits and use this information to manage better the influx of people and estimates with acceptable accuracy the waiting time.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Time slot	Period or day that can be chosen for a booking by the customers.
Reservation	A word that might indicate either a booking or a lining up in a specific store.
Active reservation	Lining up or booking that is not yet expired. It means that the reservation has been taken but customers still have to wait for their shift.
Store manager	Manager, cashier, or generic employee of a store.
Available shop	A shop that is registered into the system.
MIT License	

1.3.2 Acronyms

Clup	Customer Line-up
MVC	Model View Controller
RASD	Requirements Analysis and Specification Document
SSL	Secure Sockets Layer
TLS	Transport Layer Security

1.3.3 Abbreviations

R.n	Requirement n-th
------------	------------------

1.4 Revision history

DATE	DESCRIPTION
23/01/2021	Creation of the document.
24/01/2021	Structure of the second chapter and introduction.
28/01/2021	Adopted development framework.
29/01/2021	First test cases.
30/01/2021	Installation guide.

02/02/2021	Minor changes and improvements.
03/02/2021	Time slots solution explanation.
	Performance tests and new integration tests.
	Implemented requirement checking.
	Source Code Structure.

1.5 Reference Documents

- *Requirement Engineering and Design Project: goal, schedule, and rules*
- *I&T assignment goal, schedule, and rules*
- Slides of the course *Software Engineering 2*

1.6 Document Structure

The scope of this document is to describe the developing and testing phases of the CLup project. Here are present some integrations on what already described in the previous documents, namely RASD and DD. The document is composed of the seven following chapters:

- **Chapter 1:** provides an introduction to the purposes and the whole scenario of the software, as already specified in the previous documents. First, it includes the general description of the system, then there is a sufficiently detailed specification of the main features that the system should provide. Lastly, it includes the list of abbreviations, acronyms, and definitions used in the document, the revision history, and the reference documents.
- **Chapter 2:** includes the traceability of the requirements with the details about which ones are covered by functionalities already implemented at the current status of developing the software system.
- **Chapter 3:** specifies the framework, programming languages, and other details about the development phase. It is an integration on the descriptions already given in the DD;
- **Chapter 4:** includes an overview of the structure of the source code of the application and its organization in folders.
- **Chapter 5:** includes the main test cases and the approach to the test phase. It expands the section already presents in the DD;
- **Chapter 6:** describes the phases to install the application and the needed prerequisite;
- **Chapter 7:** shows the amount of time that each member has spent to produce the document.
- **Chapter 8:** specifies the reference documents and online resources used during the production of this document.

2 Implemented Requirement

2.1 Overview

The focus during the implementation phase was on the main functionalities of the software.

The system, also during an early release stage, should guarantee customers to line up and book at the store. These two macro-functionalities are essential for the whole system and they are the bases for any more advanced features.

Other implemented functions are strictly linked to the previous ones and are chosen according to a priority criterion. The priority is given to the basic functionalities, while the accessory features are left for future developments.

Hence, a quick overview of the implemented features is provided below. At the current state of development, the system should:

- give to customers the possibility to line up and book at the store. For this purpose, the system should allow the customer to see the list of the stores registered in the system and, for each of them, check the available time slots for the booking. Obviously, the customer should be also able to see the list of the active reservations, delete them if needed and see the full history of the past reservations. The software should check the limits on the number of concurrent active reservations for each customer and shop;
- generate a suitable QR code that can be used to enter and exit the store. The QR code is generated starting from the reservation ID which is an unique code associated to each reservation (either booking and lining up);
- show to customers an estimation of the waiting time based on the other reservations already present in the system. The customer should be also able to insert the estimated visit time to improve the general waiting time estimation. More advanced functions like the automatic inferring of the expected duration of a visit based on the previous visits of the customer are left for future developments and not implemented.

2.2 Traceability

R.#	Description	Implemented
R.1	The system generates a single QR code to enter and exit the store for each booking or lining up.	
R.2	The system allows to get a reservation for the supermarket.	
R.3	The system provides customers a precise estimation of the waiting time.	
R.4	The system allows customers only one booking at a time for each shop.	

R.5	The system alerts the customers before their shift according to the geolocation information.	
R.6	The system allows people (who do not have access to the required technology) to digitally line up directly when they are at the store.	
R.7	The system suggests alternative time slots for visiting the store when the desired one is not available.	
R.8	The system suggests alternative stores when the desired one is not available.	
R.9	The system allows customers to insert the approximate expected duration of the visit.	
R.10	The system infers customers' expected duration of the visit based on an analysis of the previous visits.	
R.11	The system allows store managers to get a lining up only for their own store.	
R.12	The system provides periodic notifications of available time slots in a day/time range.	
R.13	The system shows the list of shops.	
R.14	The system shows the available time slots for each grocery.	
R.15	The system provides a QR code printing service.	
R.16	The system requires a sign up/login.	
R.17	The system shows active reservations.	
R.18	The system shows the history of reservation of each customer.	
R.19	The system allows users to delete a reservation.	
R.20	The system allows customers only one lining up at a time for each shop.	
R.21	The system uses information about the customer that exit the store to infer better the waiting time.	
R.22	The system allows the store manager to scan the QR codes.	

3 DEVELOPMENT DETAILS

3.1 Adopted Development Frameworks

As anticipated in the DD, the software system has been implemented using the **Laravel framework**.

It is a web application framework that is free and open-source (MIT License). Moreover, Laravel follows the **Model View Controller (MVC) pattern**.

Using the Laravel framework leads to several advantages, including:

- tools for dependency injection, queues, and real-time events;
- high scalability, especially horizontal scalability;
- huge support from the community.

Furthermore, the modularity provided by Laravel makes it easier and more efficient to expand the functionalities of the system.

Laravel encourages reusing of code through a great number of third-party libraries that allow saving time in the development of common features encouraging the reuse of well-tested solutions.

Lastly, **PHP**, which is a general-purpose scripting language, has been used because Laravel is a PHP framework.

The mobile application has been developed in Java. **Java Programming Language** is a general-purpose, concurrent, strongly typed, class-based object-oriented language. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages.

Kotlin can also be used for the development of mobile apps on Android, but Java has been chosen because it is equipped with more documentation and has a larger community.

AlterVista is a site that offers free hosting. They provide various tools such as PHP 5 and 7, MySQL database without queries limits, free HTTPS certificate, FTP with encrypted secure connection, mod_rewrite and .htaccess management, Cron jobs tool. AlterVista was chosen as it is one of the most reliable and complete free hosting companies. However, it was not used to run cronjobs as it is incompatible with the needs of the application.

Hence **cron-job.org** was used. It is a site that allows making cronjobs for free and each job can be executed up to 60 times an hour. It was selected as it has no limitations and satisfies the application needs.

3.2 Development Solutions

To create a coherent integration between booking and lining up mechanisms, the available time has been split into time slots. Each time slot has a fixed duration, namely 15 minutes.

This solution permits to manage the estimation of the duration of the visit efficiently and effectively. Indeed, the customers can select an expected duration of 15, 30, 45, or 60 minutes which correspond at 1, 2, 3, or 4 time slots, respectively.

The capacity of the shop (i.e. the maximum number of people allowed at the same time) is associated with each time slot. For each new reservations that involve a specific time slot, its capacity is decremented by one. If the capacity goes to zero, the system must not accept new reservations in that time slot.

Moreover, adopting this solution the computation of the waiting time is fast and efficient. Indeed, given a reservation, the associated time slot is always known. Hence, the waiting time with a maximum error of 15 minutes¹ can be computed as the difference between the starting time of the time slot and the current time.

Furthermore, it reduces the overall computational complexity for the lining up and booking operations thanks to a smaller number of checks needed w.r.t. other solutions. This simplification reduces also the probability of errors during the writing of the code.

¹ under the hypothesis that the customer provides a sufficiently precise estimation of the duration of the visit

4 SOURCE CODE STRUCTURE

5 TEST

5.1 Integration Test

As planned in the DD, the selected strategy for the integration was of thread type. Moreover, to integrate as well as possible the test phase with the implementation phase, the incremental integration has been executed during the development, as soon as one component was released.

Postman has been used for integration testing, firstly because it is a collaborative platform, secondly because it is a comprehensive tool to automate the testing of our app components.

Tests are automated by creating test suites that can run again and again. Postman can be used to automate many types of tests including functional tests, integration tests, end-to-end tests, regression tests, mock tests, etc. It offers a sleek user interface with which to make HTML requests, without the hassle of writing a bunch of code just to test an API's functionality.

It is both used for integration and performance tests.

5.1.1 User Test

1. Success Login

This test simulates a login request performed with valid credentials that should be accepted by the system.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

2. Error Login

This test simulates a login request performed with invalid credentials that should not be accepted by the system.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

3. Sign Up Success

This test simulates a Sign Up request performed with invalid data.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

4. Sign Up Error

This test simulates a Sign Up request performed with valid data.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

5. Get All Reservations Success

This test simulates an error-free request of the history of reservations by the customer.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

6. Get All Reservations Error

This test simulates the request of the history of reservations by the customer when some errors occur.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

7. Get Active Reservations Success

This test simulates an error-free request of the list of active reservations by the user.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

8. Get Active Reservations Error

This test simulates the request of the list of active reservations by the user when some errors occur.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

5.2 Shop

5.1.2Shop Test

1. Get All Shops Success

This test simulates the request of the list of available shops when no errors occur.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

2. Get All Shops Error

This test simulates the request of the list of available shops when some errors occur.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

3. Get Available Time Slots Success

This test simulates the request of the list of available time slots of the selected shop when no errors occur.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

4. Get Available Time Slots Error

This test simulates the request of the list of available time slots of the selected shop when some errors occur.

Test	Description	Expected Result
Status Code	Checks if the response status code of the HTTP request is equal to 200 (request correctly received, understood, and accepted)	Status Code is 200
Response Format	Checks if the response of the request is in JSON format	Response in JSON format
Schema Validation	Checks if the format of the JSON and its content are coherent with what expected	The content and the format are coherent

5.2 Functional System Test

This phase has been conducted on the complete system to test the behaviors and the outputs returned by the system in different scenarios. For each case, in the following tables are specified the type of the test, a description of the actions that have been executed during the test, and the expected result (i.e. the desired outputs).

Although in general it should be done by a different team, due to organizational reasons, also the system test has been performed by the developing team.

Due to the nature of these tests, they are performed manually.

1. Login

This test checks the correctness of the login procedure, which is the first action that a user should do in the application.

2. Home Activity

This test checks the access to the different sections of the application and some other basic operations performed on the Home Page of the application.

3. Lineup

This test simulates the series of actions that a user should perform to get a lining up to the desired shop.

4. Booking

This test simulates the series of actions that a user should perform to get a booking to the desired shop.

Test	Description	Expected Result
Open Booking Create View	Click on the <i>Book</i> button of a shop on the Book Tab.	The view for creating a new booking is displayed without errors.
Time Slots	Select a date from the calendar shown on the page, then click on the button depicting the expected visit duration and click on the <i>Book</i> button to create a new booking.	If there are no available time slots, shown an error message to inform the user. Otherwise, the list of available time slots is displayed without errors.
Create booking	Select the preferred time slot to enable the <i>Book</i> button, then click on it to create a new booking.	A dialog containing a message from the server is correctly shown.

5.2 Performance System Test

6 INSTALLATION INSTRUCTIONS

The installation requires an Android Device with at least Android 8.1. Before of proceed to install the application, check that is enabled the installation from unknown sources. After that, you need to install the clup.apk .

The application needs an internet connection to work properly.

It is possible to create a new account or use the following credentials:

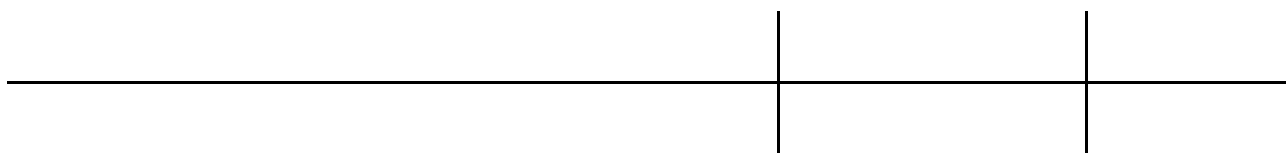
Email: prova@prova.com

Password: prova

7 EFFORT SPENT

This section shows the amount of time that each member has spent to produce the document. Please notice that each section, diagram, and specification is the result of coordinated work. The column *Member* specifies only the main contributor (or contributors, if more than one) for each topic but should not be interpreted as a lack of participation by other team members for that topic.

[illegible]



8 REFERENCES

- The diagrams have been made with: <https://www.visual-paradigm.com/> and <https://lucid.co/it>
- The mockups have been made with: Adobe XD
- Sequence Diagram Reference: <https://www.uml-diagrams.org/sequence-diagrams-reference.html>
- *UML - Imparare a descrivere sistemi orientati agli oggetti graficamente e in modo standard*, APOGEO, Enrico Amedeo
- Wikipedia - https://en.wikipedia.org/wiki/Main_Page