

# CLup

Customers Line-Up



## ATD

Acceptance Testing Document

Software Engineering II Project  
Computer Science and Engineering  
Politecnico di Milano  
Version 1.0 - 14 February 2021



**POLITECNICO**  
MILANO 1863

Davide Merli - 10578363  
Dario Passarello - 10566467

---

<b>Deliverable:</b>	Acceptance Testing Document
<b>Title:</b>	Acceptance Testing Document
<b>Authors:</b>	Davide Merli, Dario Passarello
<b>Version:</b>	1.0
<b>Date:</b>	14 February 2021
<b>Download page:</b>	<a href="https://github.com/davidemerli/MerliPassarello/">https://github.com/davidemerli/MerliPassarello/</a>
<b>Reviewed Project:</b>	<a href="https://github.com/PiroX4256/SE2-Piemonti-Pirovano-Sonnino/">https://github.com/PiroX4256/SE2-Piemonti-Pirovano-Sonnino/</a>
<b>Copyright:</b>	Copyright © 2021, Davide Merli, Dario Passarello – All rights reserved

---

# Contents

<b>Table of Contents</b>	<b>3</b>
<b>List of Figures</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Scope	5
1.2 Document Structure	5
1.3 Definitions, Acronyms, Abbreviations	6
1.3.1 Acronyms	6
1.4 Revision History	7
1.5 Reference Documents	7
1.6 Used Tools	7
1.7 Team Contacts	7
<b>2 Installation</b>	<b>8</b>
2.1 Running the system	8
2.2 Building the backend	8
2.3 Deploying the backend locally	8
2.4 Running tests	9
<b>3 Acceptance Test Cases</b>	<b>10</b>
3.1 Test Batch 1: Registration and Authentication of an account	10
3.1.1 Registration and Authentication	10
3.2 Test Batch 2: Creating a ticket	11
3.2.1 Customer Creates a Ticket	11
3.2.2 Attendant Creates a Ticket	11
3.3 Test Batch 3: Deleting a ticket	12
3.3.1 Customer Deletes a Ticket	12
3.3.2 Attendant deletes a Ticket	12
3.4 Test Batch 4: Admitting a ticket inside the store	13
3.4.1 Admitting a valid ticket	13
3.4.2 Rejecting an invalid ticket	13
3.5 Test Batch 5: Managing the store	14
3.5.1 Creating a new store	14
3.5.2 Creating a new time slot	14
3.5.3 View and Fire store attendants	15
3.6 Test Batch 6: Stressing permissions	16
<b>4 Code Inspection</b>	<b>17</b>
4.1 Backend	17
4.2 Frontend	17
<b>5 Conclusions</b>	<b>17</b>
<b>6 Effort Spent</b>	<b>18</b>
6.1 Dario Passarello	18
6.2 Davide Luca Merli	18

## List of Figures

1	Result of the mvn package command. . . . .	8
2	Homepage of the site deployed locally . . . . .	9
3	All tests passing . . . . .	9

# 1 Introduction

## 1.1 Scope

This document shows how the acceptance testing is performed on an implementation of CLup done by another group.

The acceptance testing performed on prototype is driven by the specifications, the design decisions and the implementation provided by the team in written form on the **RASD**, **DD**, and **ITD**.

The implementation on which this acceptance test is performed is the one made by Nicolò Sonnino and Luca Pirovano and can be found at [this Github repository](#).

## 1.2 Document Structure

**Chapter 1** explains the purpose of this document. Acronyms and Definitions used through the whole document are listed and explained. It provides the history of the document revisions and this very description of all the chapters, together with the documents used as a reference.

**Chapter 2** shows how the installation process was performed on our end (also noting if anything caused problems and how we solved them), following the documentation and instructions provided in the **ITD**.

**Chapter 3** shows all the acceptance test cases performed with their outcome. For each found issue a complete bug report is provided.

**Chapter 4** contains the conclusions of this Acceptance Testing.

**Chapter 5** shows the time spent by each member of the team for writing this document.

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Acronyms

- **S2B**: Software To Be
- **RASD**: Requirement Analysis and Specifications Documents
- **REST**: REpresentational State Transfer
- **API**: Application Programming Interface
- **UX**: User Experience
- **UI**: User Interface
- **QR code**: Quick Response code
- **OS**: Operating System
- **GPS**: Global Positioning System
- **ATD**: Acceptance Testing Document

## 1.4 Revision History

- 1.0: 14 February 2021 - First Release

## 1.5 Reference Documents

- R&DD Assignment A.Y. 2020-2021 - Elisabetta di Nitto, Matteo Giovanni Rossi
- Implementation Assignment A.Y. 2020-2021 - Elisabetta di Nitto, Matteo Giovanni Rossi
- Software Engineering II slides and material - Elisabetta di Nitto, Matteo Giovanni Rossi
- CLup RASD - Nicolò Sonnino, Luca Pirovano
- CLup DD - Nicolò Sonnino, Luca Pirovano
- CLup ITD - Nicolò Sonnino, Luca Pirovano
- [Material Design Guidelines](#)
- [Convventional Commits Guidelines](#)
- [Git - versioning system](#)

## 1.6 Used Tools

- Github - for code hosting and version control
- L<sup>A</sup>T<sub>E</sub>X - to write this entire document
- Visual Studio Code + Latex Workshop - as a L<sup>A</sup>T<sub>E</sub>X environment

## 1.7 Team Contacts

- Dario Passarello: [dario.passarello@mail.polimi.it](mailto:dario.passarello@mail.polimi.it)
- Davide Merli: [davideluca.merli@mail.polimi.it](mailto:davideluca.merli@mail.polimi.it)

## 2 Installation

### 2.1 Running the system

The system was fully deployed and ready to test: we were able to install the mobile application by downloading it directly from the Google Play Store; the webapp was hosted at <https://clup.cf> (though it was not listed in the ITD document)

The backend REST API was also deployed and was available at <https://clup.cf/api/>

We also installed successfully the mobile application using the provided build files.

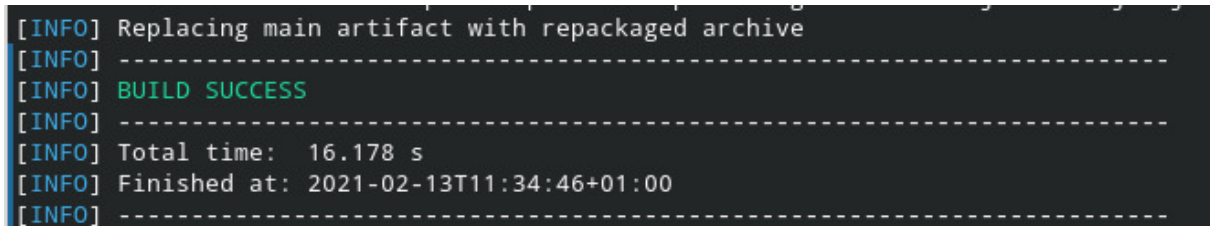
### 2.2 Building the backend

Following the section 5 of the ITD, we installed the requirements Java JDK 15 (open-jdk) and Maven.

After pulling the repository from Github we ran the command

```
$ mvn package -Dmaven.test.skip=true
```

in the folder IT/Implementation/Backend, as specified in the provided instructions. The build was successful.



```
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 16.178 s
[INFO] Finished at: 2021-02-13T11:34:46+01:00
[INFO] -----
```

Figure 1: Result of the mvn package command.

### 2.3 Deploying the backend locally

As stated in the chapter 6 of the ITD a mysql data source should be available to the application, in order to make it work.

We hosted locally a mysql instance in a docker container and configured the application.yaml accordingly.

In the same folder of the application jar file we ran

```
$ java -jar clupapplication-1.1.jar --spring.config.location=application.yaml
```

Running this will start both the API service and the Web Application, on localhost.



By connecting with a web browser to the address localhost:8084 (the port is configurable in application.yaml), the CLup home page is displayed correctly. We tried to register an account, and that worked properly, testifying that the database was initialized successfully.

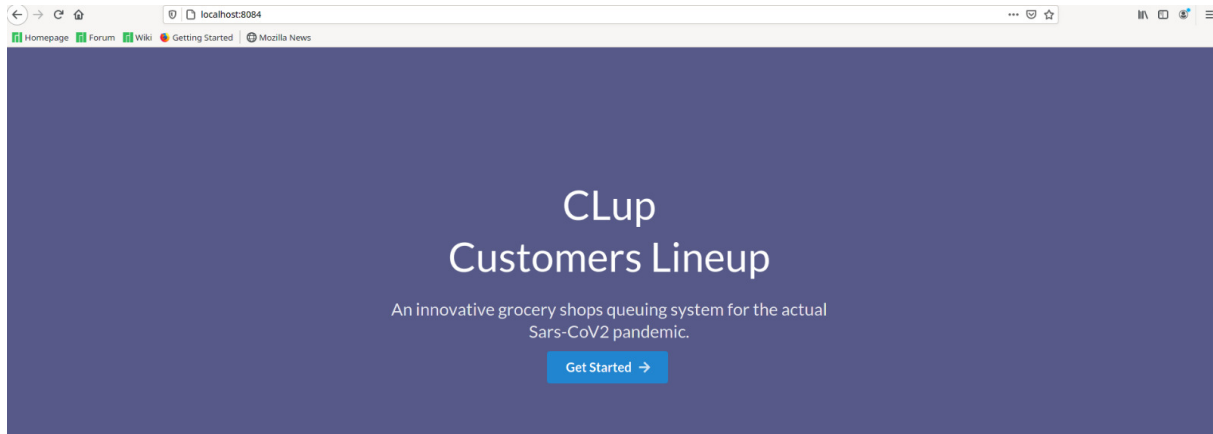


Figure 2: Homepage of the site deployed locally

## 2.4 Running tests

Using instructions provided in chapter 5 we ran the tests. The application.yaml was not found in the folder specified in the instructions. Copying the application.yaml file used in the deployment in the src/main/resources directory solves the issue (if the database used for testing is the same used for the deployment).

After doing these steps running

```
$ mvn test
```

produces the following output

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 20, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.263 s
[INFO] Finished at: 2021-02-13T11:42:34+01:00
[INFO] -----
```

Figure 3: All tests passing

This means that all the test pass, as shown in the ITD.

## 3 Acceptance Test Cases

### 3.1 Test Batch 1: Registration and Authentication of an account

The following tests check if the various Registration and Authentication features work properly, without having noticeable security flaws.

This test is repeated with these conditions:

- Register and login as a customer in the application
- Register and login as a customer in the webapp
- Register and login as an attendant in the application
- Register and login as an attendant in the webapp
- Register and login as a store manager in the webapp

#### 3.1.1 Registration and Authentication

- **Test Goal:** Check if the registration and authentication processes behave properly even with erroneous input
- **Prerequisites:** None
- **Steps:**
  1. Open the Application
  2. Tap 'Create New account'
  3. Tap 'Register with Email'
  4. Click 'Signup' without filling the forms
  5. Add invalid email address/Invalid password
  6. Add a previously used email address
  7. Fill the form correctly and click 'Signup'
  8. Logout pressing the icon and confirm
  9. Log-in with the same e-mail but different password
  10. Log-in with the same e-mail and password
- **Outcome:** In all cases everything works as intended and there are no noticeable problems with the authentication features

## 3.2 Test Batch 2: Creating a ticket

### 3.2.1 Customer Creates a Ticket

- **Test Goal:** Check if the creation of a ticket works properly
- **Prerequisites:**
  - Customer account credentials
  - A store with a free timeslot
- **Steps:**
  1. Login with the customer account
  2. Click ‘Select Store’
  3. Browse the map
  4. Select a store from the dropdown menu and confirm
  5. Click ‘Line-Up’ and confirm
  6. Click the button to see the QR Code
  7. Try to book another ticket
  8. Check if the application notifies when the time to enter the store is approaching
- **Outcome:** In all cases everything works as intended

### 3.2.2 Attendant Creates a Ticket

- **Test Goal:** Check if the creation of a ticket works properly
- **Prerequisites:**
  - Store Attendant account credentials
  - A free timeslot in the store he has been assigned to work in
- **Steps:**
  1. Login with the attendant account
  2. Click Generate Ticket
  3. View The QR code of the generated ticket
  4. Check if the ticket appears in the webapp and in the ‘Delete Booking’ section
- **Outcome:** In all cases everything works as intended

### 3.3 Test Batch 3: Deleting a ticket

#### 3.3.1 Customer Deletes a Ticket

- **Test Goal:** Check if the cancellation of a ticket works properly
- **Prerequisites:**
  - Customer account credentials
  - A valid Active Ticket
- **Steps:**
  1. Tap the ‘Info’ icon on the ticket
  2. Tap ‘Delete’
- **Outcome:** A minor issue was found
  - **Issue:** When deleting the ticket in the mobile application it does not always disappear from the ticket list. The ticket is correctly voided in the system but simply it does not disappear from the ticket list viewed from the user.
  - **Fix:** The user can fix this problem swiping down in order to refresh the application page and the ticket will correctly disappear.
  - **Severity:** Very Low. It is a minor UX issue that does not create logical problems.

#### 3.3.2 Attendant deletes a Ticket

- **Test Goal:** Check if the cancellation of a ticket done by an attendant works properly
- **Prerequisites:**
  - Customer account credentials
  - A valid ticket for entering a store
  - Attendant account credentials, working in the same store of the valid ticket
- **Steps:**
  1. Login as the attendant
  2. Tap the ‘Delete Booking’ button
  3. Select the ticket to cancel in the ‘Info’ button and confirm action in the dialog
  4. See if the ticket properly disappeared
- **Outcome:** In all cases everything works as intended

## 3.4 Test Batch 4: Admitting a ticket inside the store

### 3.4.1 Admitting a valid ticket

- **Test Goal:** Check if a customer with a valid ticket is admitted in the store by the system
- **Prerequisites:**
  - Customer account credentials
  - QR code of a ticket for the current timeslot
  - Attendant account credentials, working in the same store of the ticket
- **Steps:**
  1. Login with the attendant account
  2. Tap ‘Scan QR code’ button
  3. Scan the QR Code
- **Outcome:** The ticket is accepted as expected

### 3.4.2 Rejecting an invalid ticket

- **Test Goal:** Check if a not yet valid ticket is rejected by the system
- **Prerequisites:**
  - Customer account credentials
  - QR code of a ticket for a timeslot in the future
  - Attendant account credentials, working in the same store of the ticket
- **Steps:**
  1. Login with the attendant account
  2. Tap ‘Scan QR code’ button
  3. Scan the QR Code
- **Outcome:** The ticket is rejected as expected

## 3.5 Test Batch 5: Managing the store

### 3.5.1 Creating a new store

- **Test Goal:** Create a new store with the store manager account
- **Prerequisites:** Store Manager account credentials
- **Steps:**
  - Login as Store Manager
  - Click ‘Create new Store’
  - Fill the form with incomplete information
  - Submit the form
  - Add all information correctly
  - Submit the form
  - Check in the Customer Application if the store is displayed
- **Outcome:** The store is created and displayed in all maps as expected

### 3.5.2 Creating a new time slot

- **Test Goal:** Create a new store with the store manager account
- **Prerequisites:** Store manager account credentials with a store bound to this account
- **Steps:**
  - Login as Store Manager
  - Click ‘Edit Store’
  - Fill the ‘Add Slot’ form
  - Click ‘Add’
  - Check if the slot is correctly created
- **Outcome:** A minor issue was found
  - **Issue:** A slot without a time field is created if AM or PM is not specified in the ‘Starting Hour’ textbox. The page didn’t raise any error. If a slot without a time field is present in the day, the customer will not find any free slots for that day.
  - **Fix:** The manager can delete easily the bugged timeslot and create a new one. This will fix the issue.
  - **Severity:** Low. The issue is easily detectable and fixable by the manager.

### 3.5.3 View and Fire store attendants

- **Test Goal:** Create a new store with the store manager account
- **Prerequisites:**
  - Store Manager account credentials
  - At least an attendant account in the store bound to the manager
- **Steps:**
  - Login as Store Manager
  - Click the ‘Attendants’ button
  - Click ‘Fire’ in an attendant row
  - Confirm the action
  - Check if the attendant is correctly fired
- **Outcome:** An issue was found
  - **Issue:** The Attendant could login again with his account credentials. They can scan valid tickets and admit them into the store, however they can’t see and delete active bookings.
  - **Fix:** The user can’t fix this issue manually.
  - **Severity:** Medium. A fired attendant has the rights of admitting people in the store. This is not however a big issue because a valid qr code is still needed, and it is not trivial forging a ticket.

### 3.6 Test Batch 6: Stressing permissions

- **Test Goal:** Check if application permissions exceptions are handled correctly
- **Prerequisites:**
  - Store Attendant account credentials
- **Steps:**
  - Login as Store Attendant
  - Click the ‘Scan QR code’ button
  - Click ‘No’ when asked to allow camera permissions
  - Login as Store Attendant
  - Click again the ‘Scan QR code’ button
  - Click ‘Deny and don’t ask again’ when asked to allow camera permissions
- **Outcome:** The app behaves correctly and does not halt/show uncorrect behavior if camera permissions are not allowed by the user. It simply responds with an ‘Invalid’ dialog, and returns to the previous screen.



## 4 Code Inspection

### 4.1 Backend

The code is well structured and organized, and each software module has its own folder. Javadoc is provided as documentation and covers the whole application.

The backend tests are well written and they cover most of the use cases.

### 4.2 Frontend

The frontend is also well organized and correctly divided in sub-packages. Fewer documentation is provided with respect to the backend code. Taking in consideration that framework used is Flutter, the majority of the code is self-explanatory: it is only composed of trees of Widgets, which mainly compose the graphics of the application. The application logic is documented, and Dart's Doc comments standard (represented by '///') is respected.

Attempts to make the UI code look more slim (and reuse custom Widgets to avoid repeating code) were made.

## 5 Conclusions

The prototype behaves correctly on almost every tried test, and the found issues are not serious and would require only little work to be fixed.

The presented application is graphically very similar to the mockups shown in the documentation, so it was easy and not confusing comparing the two. This is valid for both the mobile application and the webapp.

The whole system is near to be production ready, because it is very stable and offers a broad range of features to all the actors involved (starting from the customers up to the Store Managers).

## 6 Effort Spent

### 6.1 Dario Passarello

- Acceptance testing with the CLup application: 4 hours
- Deploying the backend locally: 2 hours
- Writing ATD: 6 hours

Total: 12 hours

### 6.2 Davide Luca Merli

- Acceptance testing with the CLup application: 3 hours
- Deploying the backend locally: 1 hours
- Browsing through the source code: 1.5 hours
- Writing ATD: 5 hours

Total: 10.5 hours