POLITECNICO
MILANO 1863

Acceptance Test Deliverable

Customers Line-up

Alessandro Ferrara - Lorenzo Fratus

Professor: Elisabetta di Nitto

February 13, 2021

Version: 1.0

Contents

1. Analyzed project

A. Authors

The authors of the analyzed prototype are:

- Jesus Rodrigo Cedeño Jimenez
- Diego Andres Cumming Cortes
- Angelly de Jesus Pugliese Viloria

B. Repository

The repository of this project can be found at this link.

C. Reference documents

The following documents were considered during the acceptance test:

- Assignment document A.Y. 2020/2021 ("Requirement Engineering and Design Project: goal, schedule, and rules")
- Assignment document A.Y. 2020/2021 ("I&T assignment goal, schedule, and rules")
- Requirement Analysis and Specification Document - CLup: Customers Line-up Software, version 2 (referenced as "RASD")
- Design Document - CLup: Customers Line-up Software, version 2 (referenced as "DD")
- Implementation and Test Deliverable - CLup: Customers Line-up Software, version 1 (referenced as "ITD")

2. Installation setup

A. Installation guide

The installation guide is split into two *README* files (one for each main folder).

The guide for the installation of the back-end (*server* folder) is not so clear, you need to install many dependencies and manually set environment variables.
From a non-poweruser perspective this is quite a lot to handle.

To use a local instance of the database, it is assumed that the user has already installed and to is able to use *PostgreSQL*.
Unfortunately, no structure of the database is presented in the guide nor in the ITD so it is not possible to create a functioning database without looking into the DD or the code.

B. Position spoofing

As stated in the ITD (*2.2. Map*), the map presents the user with the list of stores only in a radius of 2.2km.
Furthermore, the system does not provide for the insertion of new points of sale.

In order to use the application it is therefore necessary to "spoof" the position through an external plugin.

This information, essential for the functioning of the system, should have been included in the installation guide.

3. Acceptance test cases

Before manually testing the main functions of the application, automatic tests included in the software has been performed to verify the overall correctness of the system. No errors has been found at this stage.

A. Login and registration

A.1. Registration

The registration process is straight forward, requires a username, password and acceptance of the *Terms of Service and Privacy Policy*. The latter checkbox is purely aesthetic. After clicking on the "Continue" button, the user is redirected to the login page.

A.2. Login

From the login page is possible to insert a valid combination of username and password in order to authenticate. After clicking on the "Login" button, the user is redirected to the main page of the application that displays the map.

A.3. Input validation

Some tests have been carried out using invalid inputs and these are the results:

- Registration with correct credentials: ***201 Created*, redirect to "login"**
- Registration without username, password or the acceptance of the *Term of Service and Privacy Policy*: ***201 Created*, redirect to "login"**
- Registration with an already existing username: ***500 Internal Server Error***
- Login with correct credentials: ***200 OK*, redirect to "supermarkets_list"**
- Login with wrong credentials: ***401 Unauthorized***
- Login without username or password: ***401 Unauthorized***

There is neither client-side nor (apparently) server-side validation of input data, so requests with missing or incorrect data can be made.

The only operations not allowed are the registration with an already existing username and the login with incorrect (or missing) credentials, probably because those queries are rejected from the DBMS (as can be guessed from the status code of the responses).

Registration without username was possible only once as the empty string is considered a valid username and therefore a second registration would imply more users with the same username.

Unfortunately, there is no feedback when an operation is rejected by the server. This is not critical but it allows the user to better understand what is happening.

B. Main page

The main page features a map with a list of stores and, on the right, four different control buttons (one at the top and three at the bottom).

B.1. Stores

Stores are color coded based on waiting time. Clicking on an icon displays a popup with two buttons: “Line up” and “Book”.

A strange delay in updating the store waiting time can be seen when performing operations such as queuing or canceling a ticket.

B.1.1. Line up

Clicking on the “Line up” button opens the QR code page showing the newly created ticket and the expected waiting time or a 5-minute countdown before the ticket expires (if the waiting time is 0).

This feature works as per the documentation. As expected, it is not possible to line up if there is already a ticket for any shop.

B.1.2. Book

Clicking on the “Book” button opens a page to insert the wanted date and hour. The system allows to get a ticket from one hour to one week in advance, it would have been a good idea to point it out in the interface.

Upon confirmation, the system shows the page containing the generated QR code where a countdown called waiting time is shown. In this case it would probably have been more appropriate to display the date and time of validity of the ticket.

As before, the function does what is expected and it is not possible to book if a ticket for any store already exists.

B.2. Filter

The first button in the lower right corner is the filter. Clicking on it displays a popup.

From this popup is possible to select one or more store *brands* and a maximum waiting time. By clicking on "Apply" the filter is set and, by clicking on "Clear" the filter is removed, in both cases the map is updated.

This is a simple and useful function and it works as one would expect. The only drawback is that is important to standardize the store *brands* in order to avoid duplication.

B.3. Buy *ASAP*

The second button in the lower right corner opens a page with a list of stores sorted by distance and waiting time. This page allows to line up directly from the list. There are no particular comments on this feature.

B.4. QR code

The last button in the lower right corner opens the page that the user is redirected to after requesting a ticket.

This page contains a QR code (representing the username) and the expected waiting time (or the countdown to the ticket expiration). The page also shows two buttons to "Go back to map" and to "Cancel request". Both buttons take the user back to the main page but the second one deletes the current ticket.

Opening the browser *Network* interface it is possible to notice that every second the page sends two requests to the server: *qrcode* and *remainingTime*.
Since this is not necessarily a real-time system, it is a waste of resources to send requests this frequently. One update every minute would probably suffice.

B.5. Logout

In the upper right corner of the screen there is a button to log out. This feature works as expected.

C. Security issues

The testing activity highlighted the fact that there are no security checks on the data that the application sends to the server.

Specifically, changing the username in the body of HTTP requests made it possible to queue for another user (even if it did not exist) and also to cancel another user's ticket.

This test was not done for the booking functionality but it is safe to assume that it would behave the same way.

One solution is to verify that the author of the request matches the user who is affected. This can be done by leveraging the already implemented JSON Web Token (JWT).

A software that does not implement an authentication and authorization mechanism is vulnerable to malicious requests.

4. Additional comments

A. Software compatibility

The application does not seem to work on the latest version of the Safari web browser (14) unlike what is specified in the RASD (*3.1.3. Software Interfaces*). The map appears completely gray, therefore unusable. Not a big deal, it was just worth pointing out.

B. Software architecture

B.1. Front-end

In the DD, it is stated on several occasions (e.g. *2.1. Overview*) that the client-side application runs on the user's device. This appears to conflict with other parts of the same document where the front end is regarded as a web server (e.g. *2.4. Deployment view*).

B.2. Server-side rendering and APIs

The system has been divided into two servers:

- The first one (*clup* folder) is responsible for providing the user interfaces by means of server-side rendering (using Vue.js).
- The second (*server* folder) is responsible for the application logic and the communications with the database.

These two server are connected via APIs.

Implementing a server-side rendering mechanism is a waste in the presence of a set of APIs as the page rendering operation can be performed directly from the client side (common practice).

C. Limits of the acceptance testing

The system has been developed assuming that each store has a physical device capable of scanning tickets before letting customers in and out.

This machine should also be able to issue tickets as a fallback for customers who are not registered in the application.

Unfortunately, this assumption limits the range of tests that can be performed on the application as currently the user experience ends with the issuance of the ticket.

5. Effort spent

Pair programming

Topic	Hours
General structure	0.5h
Acceptance testing	3.5h

Ferrara Alessandro

Topic	Hours
Code analysis	2h

Fratus Lorenzo

Topic	Hours
Documentation analysis	2h