Software Engineering II 2020/2021

February 2021

Acceptance and Test Deliverable

CLup: Customers Line-up Software

Link to GitHub repository

Version: 1

Authors:

R. Cedeno, D. Cumming, A. Pugliese

Tested project:
https://github.com/mfanselmo/AnselmoGomezMunnaluri

Authors of the project:

M. Anselmo, D. Gomez, V. Krishna

Main documents considered:

ITD & RASD

Contents

1. Introduction

The main purpose of this document (Acceptance and Test Deliverable) is to detail the testing performed on another team's CLup software, by analyzing the deployment performed at: https://clup.netlify.app/ as well as the version that can be installed locally and its code. The assessment will be carried out on the following subjects:

1. Backend installation setup
2. Frontend installation setup
3. Acceptance test cases
4. Queueing functionality
5. Understandability of the code
6. Glitches search

The analysis considers the points outlined in the Requirements Analysis and Specification Document (RASD) where the CLup system's features, interfaces, functionalities and conditions of operation are described in detail.

2. Installation Setup

The application is composed of a frontend in React and a backend in Django and SQLite3. The first step is to clone or download the GitHub repository. Then, the steps to install the prototype are divided considering each part.

2.1 Frontend

Before beginning, make sure to have Node.js with version equal or higher than v12.16.1 and lower than v15, as well as NPM v6.13.4 or higher installed on your computer.

1. Inside the folder fronted set the correct environment variables by creating a '.env' file and copying on it what is written in the file '.env.example'.
    a. For some of us, the program was returning error 404 after logging in or signing up into the application because the '.env' was not working properly. With some help from the authors of the project, we realized that we had to manually specify in each one of the code files an environmental variable that indicates the url of the as "http://localhost:8000".
2. Install all the dependencies by running:

    npm install

3. To run the project on localhost:3000

    npm start

4. To run the tests:

    npm run test

2.2 Backend

Before installing the backend, it is required to first install SQLite, adding it to the environment variables. As it is not so trivial, we decided to specify this individually.

2.2.1 SQLite3

1. Download SQLite-tools from the official website and save it in a folder called 'sqlite' in C:/.
2. Make sure to have installed it correctly by running the command sqlite3 inside the folder 'sqlite' and get an output similar to:

    SQLite version 3.29.0 2021-02-10 17:32:03

3. Add SQLite3 to PATH by adding 'C:/sqlite' to it.

2.2.2 Django

1. Inside the backend folder, create a virtual environment:

    py -3 -m venv venv

    cd venv/Scripts

    activate.bat

2. Install all the requirements by running:

    pip install -r requirements.txt

3. Run the migrations to the database:

    python manage.py makemigrations

    python manage.py migrate

4. Run the server:

    python manage.py runserver

3. Acceptance Test Cases

In this section, there is an examination of the functionalities that are implemented in the prototype and an analysis of the execution of a set of test cases extracted from the project's RASD document in order to assess that the requirements are properly fulfilled.

3.1. Implemented functionalities

The analyzed software satisfies a list of functionalities that essentially allow the unregistered users to line-up or scan a ticket to check its state (Figure 1a), allow users to line-up for entering a store on the same day or make the users able to book to enter the store some of the following days (Figure 1b). The system manages the turns of the users and considers a special manager user that is able to scan QR codes in order to check whether a client's entering is accepted or not, check the stores' states and generate tickets (Figure 1c).

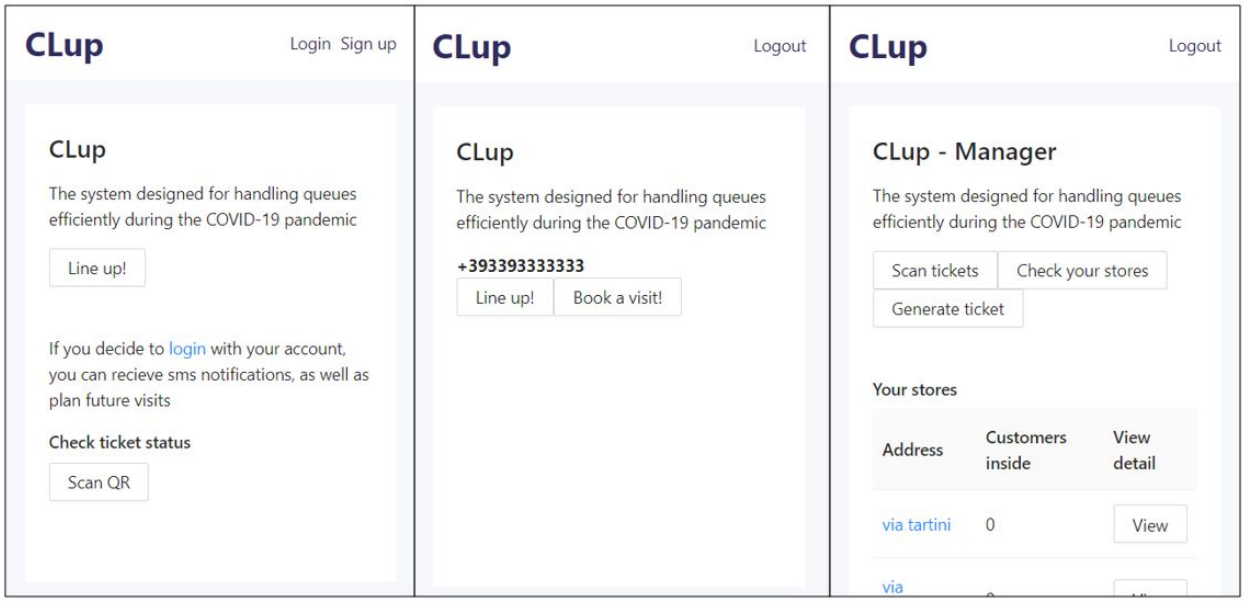

Figure 1: User interface for (a) unregistered user, (b) registered user and (c) manager user.

In particular, the following set of functionalities are included in the implemented software:

1. Users can sign up in the application
2. Users can log in and out from the session
3. Registered and Unregistered users can line-up to a store to enter the same day.
4. The manager user can produce tickets.

5. The users can check their ticket status and the remaining time to have their turn.
6. An Unregistered user can scan a QR code in order to check the ticket status.
7. The manager user can scan a QR code in order to accept or reject the entering
8. A Registered user can book the enter the store one of the following days.
9. A manager can check how many people are currently in his/her store.

3.2. Test cases

3.2.1. Customer Login

Once registered, we complete the authentication form (phone number and password) then the system redirects to the registered user page (Figure 1b). **The test case is accepted**.

3.2.2. Manager Login

The authentication form is filled with the manager credentials provided by the authors of the project and the system redirects to the manager page (Figure 1c). **The test case is accepted**.

3.2.3. Customer Register

Once in the main page, we click in the “Sign Up” button, then we complete the different fields (phone number, email address, username and password) and the system redirects to the register user page (Figure 1b). **The test case is accepted**.

3.2.4. Book a visit

Once in the registered user page (Figure 1b), we click in the Book button, then we select a store, a date and a time slot. After this, the system redirects to the ticket page, which shows the QR code, the time to enter, the store’s name, the address, the status and the options to print it or cancel it. According to the RASD, a selection of categories and estimated visit duration have to be specified by the user, but these functionalities are not included in the prototype so **the test case is accepted**.

3.2.5. See Booked Visits

When the users have one or more line-up or book requests, according to the RASD, the application is expected to have a button to access all the scheduled tickets. In order to see the scheduled line-ups or bookings after doing a request, we have to actually go back and refresh the application with the browser. Since there is no button to access the list, **the test case is rejected.**

3.2.6. Scan QR Code at Entrance

If we use the scanner of the manager user as a store's scanner device, the system accepts and registers the entrance as long the user's turn has passed (even if the user is late, whose decision to let enter or not will be handled by the stores' employees) and the store of the ticket is contained within the ones associated with the manager that is scanning. When the scanning is registered, it updates the monitor in which the manager has access to check how many people are shopping. **The test case is accepted**.

3.2.7. Register Shop Exit

Once a Customer's entrance has been registered, if the store manager scans the QR code again, the exit is registered so the monitor that shows the number of people shopping is updated. The used QR code is no longer valid to enter the store again. **The test case is accepted**.

3.2.8. Request Ticket

If we use the manager account in a store device, a store employee could press the option to generate a ticket in order to print it and provide it physically to a customer, which is able to use it just as one generated without a manager account. **The test case is accepted**.

3.2.9. View Shop Statistics

From the manager's account main page (Figure 1c), the manager can click on "Check your stores", then select a store and the system redirects to a page in which the manager can see the store name, the address, the next open slot, the amount of people inside the store, its maximum capacity, but a button to go back is missing. According to the RASD, a back button was required so it has to be included in order to accept this test case. **The test case is rejected**.

3.2.10. Manage Shop Slots And Sections

According to the RASD, the feature associated with this use case was aimed to make manager users able to edit slot numbers and sections in the stores, but the section handling functionalities are not included in the prototype so **this test case is ignored**.

4. Additional Comments

4.1. Queueing

The queueing is a tool that has the function of handling the booking and line-up requests for users that want to enter a specific store. The importance of this tool lies in the possibility of making a queue of requests that gives every user a turn to enter the store, this can be connected to the amount of users inside the store changing dynamically according to the users that exit by registering their code when concluding operations.

According to the RASD, the application was supposed to manage the queues, however, we found that it is not used to manage the line-ups and bookings, but instead the turns are defined merely by the hour that the users request to shop (now if line-up or booking time on another day), so the order in which the users are allowed to enter have no relation with the order in which they have requested their ticket.

We understood by analyzing the code, that there is a limit of bookings for every time slot, but this is not connected to the amount of users inside the store, so even though there is a limit, the users can keep accumulating inside each location, easily surpassing the maximum capacity of every place.

4.2. Missing Comments in the Code

As we expect that the code should be understood by a team of developers, the programming solutions should be explained in comment lines within the code lines, otherwise, the pieces of code are very hard to comprehend, therefore, it would be very difficult to modify, improve or update the developing. As a software engineering project is expected to have many changes along its lifetime, the developed product should be as clear and understandable as possible in order to allow the development team to act fast when needed and ease the verification process. The code files are missing explanatory comment lines, so the purpose of the different pieces of code might be very hard to figure out.

4.3. Glitch: Lineup using incorrect phone number format

The flow of events that causes the glitch are the following:

1. The user registers with a phone number with format 3xx xxxxxxx
2. Successful login with the registered phone number
3. The user goes to the ‘lineup’ page
4. The user selects the desired store
5. The user confirms the lining up

6. The system cannot complete the lining up returning an error with the following ‘Make sure your phone number matches the format 3xxxxxxxxx’, which is different to the one used at sign up (visualization of this error in Figure 2).

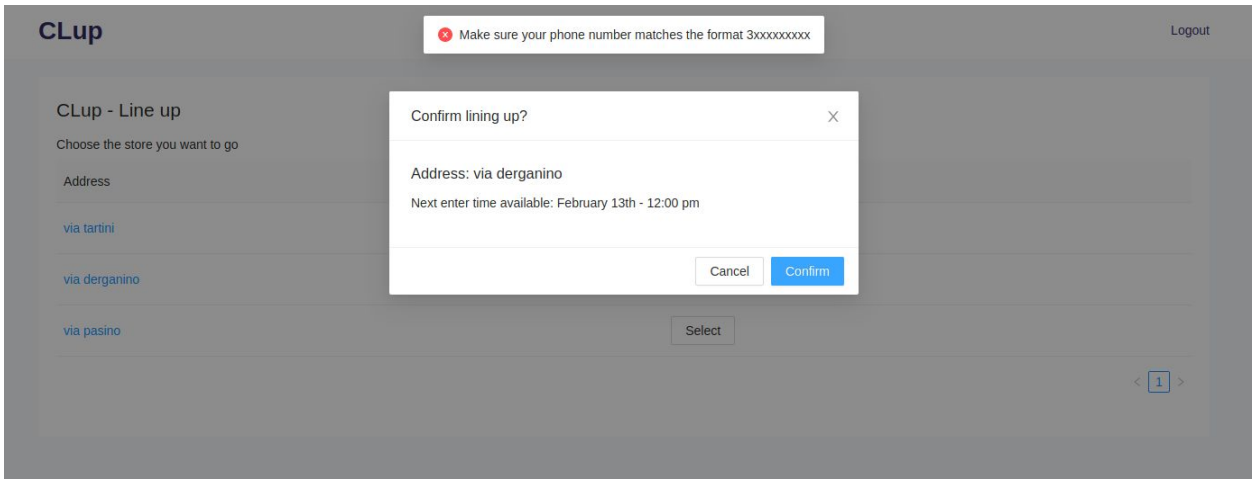

Figure 2: Error message because of unexpected input format

5. Effort Spent

Jesús Rodrigo Cedeño Jimenez	4 hours
Diego Andrés Cumming Cortés	4 hours
Angelly de Jesús Pugliese Viloria	4 hours