Acceptance testing for CLup - AvciAzzoniFasana

Authors:

- Avci Oguzhan
- Azzoni Francesco
- Fasana Corrado

Installation Setup

First, MySQL was installed, creating a user for clup and connecting to MySQL Workbench.
Then, using the dump in the repository, we imported the database schema.

Then, following the instructions, we imported the project into an IDE and edited `src/main/resources/application.properties` to match our MySQL configuration setting the database username, password and URL. Then we ran the main class from the IDE. The server started logging startup information so we moved to building the mobile application.

Using Android Studio we imported the mobile application project. We changed the `com.example.qapp.utility.ParametersServer.PATH` constant to the ip of the server. Then, connecting an android device via Android Debugging Bridge we built and ran the application on a physical device.

Testing

Demo executable

We then ran `com.server.clup.demo.Demo` to generate shops that we can use with the mobile application.

> Note: We had some problems running the Demo, since it's a JUnit test, but requires stdin input to terminate and cleanup correctly (otherwise the database will be left in a state that will prevent restarting the server), which is not allowed in IntelliJ, so we switched to Eclipse.

Account creation

We tapped on Sign up and created an account. After closing the application we logged back in and confirmed that the application correctly identified the account. We were not allowed to use the application without logging in. [R17][R26]

Now using the application, we can see that some cities have appeared, and by clicking on one of them we can click on a chain and store, ending up on the page for the store. On the store page we can check the schedule and we have access to the lining up and the booking functions.

Lining up

We first tested the lining up function. After tapping on the line up button we were asked the time to arrive to the store, then we received our ticket. The ticket correctly showed a QR code and was present in "My tickets" even after logging out and logging back in. [R11][R14]

A push notification was displayed when it was time to enter the store. [R13]

If the ticket was not used for some time, after interacting with the App it disappeared and was not available anymore. [R8]

We tried lining up with a duration that exceeded the shop time schedule and were correctly forbidden to do so. [R16]

Also, if we tried lining up twice for a store we were stopped. [R24]

Booking

We tested the booking function. From the store page we checked out the schedule, then tapped on "book a visit", there we specified a date, time and duration, then confirmed. We were then able to choose the categories we wished to visit and confirm. When it was the time for the booking we received a notification. [R12][R19][R21][R23]

The booking request was allowed only if the time slot was at least two hours in the future. [R18]

Ticket validation

Then, we tested the Ticket Validator demo, which is a second Java project. The validator uses two constants for configuration, one with the store id and the other for entrance and exit. To test entrance and exit we had to find the id of the store in the DB and set the `CommunicationModule.STORE_ID` constant to its value, then build two executables for each store we wanted to test, one for entrance and one for exit.
After that, we tried to run the Manager application but, to get it running, we had to manually specify the Java SDK in the `pom.xml`.

As for keeping track of the entries and exits [R5], we noticed that the ticket validator let us use the same ticket to enter multiple times after the ticket was activated and used. The reason behind this is that tickets are only disabled after the exit from the Store is logged.
The requirement [R3] "Allowing managers to visualize the information about the real-time store crowding" has not been implemented with a UI. The only way to test that the entrances and exits were correctly tracked is to directly query the DB.

We generated a ticket from the app and tried to validate it within the Ticket Validator, at first with the correct `STORE_ID` and then with a different one. The validator that was set up for the right store successfully accepted tickets and the other one rejected them, as expected.
We tried to cut the connection to the Validator service but no error was shown. When offline, every ticket was rejected.

We tested requirement [R6] by using a Ticket before receiving the notification and [R7] by generating 2 Tickets in a situation where the maximum allowed was only one. The application didn't show any error, however the Ticket Validator did not let us in the store, as required.

As for the claim "The system must give precedence to customers that are more in danger" [R10], it is satisfied in the sense that physical customers have priority over the virtual customers, i.e. those who have got a Ticket or a Booking.

Automated tests

We ran the automated tests one at a time, since the `Demo` test would block and the `QueueManagerTest` would fail preventing us of the tests together.
All tests ran individually passed.

Implemented features:

- **R5** The system must keep track of the people’s act of entering and exiting the store.
- **R6** The system must not let a customer in if he does not have an enabled ticket for the store.

- **R7** The system must not let a customer in the store if the number of people in the store is equal to the maximum allowed.
- **R8** After TTA plus 30 minutes since the set-off notification has been sent the ticket expires.
- **R10** The system must give precedence to customers that are more in danger.
- **R11** The system can hand out virtual tickets to the customer for lining up at a specific store.
- **R12** The system must provide the possibility for customers to book a visit to a store.
- **R13** The system sends a set-off notification taking into consideration the TTA specified at the moment of the line-up and the current situation in the store.
- **R14** The system sends an entrance notification and enables the ticket when the customer turn arrives.
- **R16** The system must not release a ticket if it estimates that the customer will not be able to enter the store before the closing time given the current situation of the queue.
- **R17** The User has access to virtual line up and booking functionalities if and only if he provides valid credentials.
- **R18** The system must be able to accept a booking request if the specified day/time is at least two hours later than the current time.
- **R19** In a booking request, the system must allow the user to select the categories of the products that he would like to buy.
- **R21** During a booking request, a non-long-term customer must provide the system with an approximate shopping time.
- **R23** The system can let more customers in the store (thus increasing the maximum number of people allowedin) if it detects that booking customers are going to buy different things.
- **R24** If the customer already holds a valid ticket for line up at a specific store the system must not release another line-up ticket for the same store.
- **R26** An external user must register to the system to gain access to Q functionalities.

Suggested improvements

- Provide maven configuration and instructions to build and test the project without an IDE.
- Extract the demo to a separate (not JUnit) entrypoint since it prevents running all tests together (and consequently automation)
- Parametrize server configuration using environment variables or config files.
- Add server side logging to keep track of unexpected behaviour, making debugging easier.

There are no particular problems with the quality of the source code other than the problems we reported earlier and the absence of working automated build scripts (pom.xml etc.)