POLITECNICO
MILANO 1863

CLup project ATD Roberto Buratti, Hrvoje Hrvoj, Ozan Incesulu

Acceptance Testing Document

Deliverable:	ATD
Title:	Acceptance Testing Document
Authors:	Roberto Buratti, Hrvoje Hrvoj, Ozan Incesulu
Version:	1.0
Date:	14-2-2021
Download page:	https://github.com/Furcanzo/BurattiIncesuluHrvoj
Copyright:	Copyright © 2021, Roberto Buratti, Hrvoje Hrvoj, Ozan Incesulu – All rights reserved

Contents

1 Target Project Description

Project: **CLup**

Created By: **Ferrara Alessandro and Lorenzo Fratus**

Repository: **https://github.com/ferrohd/FerraraFratus**

2 Installation Setup

2.1 Project Setup

To install the prototype we needed to:

1. Install NodeJS, (download from the official website)
2. Install the dependencies, (via *npm i*)
3. Run the application using a custom script provided by the developers (*npm run start*)

The installation guide is easy to follow and complete, but there are some details that can be specified better. For example the specific minimum version of NodeJS that needs to be installed could have been specified, as two of our group members had problems when they run an old version. Another issue was that the developers have provided a public database, and without explicit instructions on how to delete the test entities or how to provision another database to connect to. Therefore, in order to execute our tests faster and have a clean state on each run, we have contacted the developers for the database dump for the production database so that we can provision a similar database.

2.2 Acceptance Test Setup

We have used Cypress to create the acceptance tests for the project, which is a library that allows us to define browser-based tests rapidly. We have used Typescript while creating our tests for typing validation in Javascript. We have created a custom command in Cypress to allow us easily create users with certain properties. For realistically random addresses we have pulled the street name list for all the streets in Milan from OpenStreetMaps API and saved it in Cypress as a fixture. We have created schema instructions to reset the database each time it is connected to, and provided instructions in acceptanceTest.md on how to create a database and run the tests. We have used an extra QR code reader library to read the content of QR codes generated by the server, without having to interfere with the server's code.

3 Acceptance Tests

Acceptance tests are divided in six sections, one for each functionality described in the ITD document:

3.1 createStoreManager

- will ask for the necessary information to register a user (generalization of Store Manager):
  Test related to the requirement R1 of the RASD document,
  Test steps:

  1. //todo description

  Test Passed

- will register a person to the system as a user (generalization of Store Manager):
  Test related to the requirement R1 of the RASD document,
  Test steps:

  1. //todo description

  Test Passed

- will verify that the email provided by a person during the registration process is unique (generalization of Store Manager):
  Test related to the requirement R2 of the RASD document,
  Test steps:

  1. //todo description

  Test Passed

- will verify that the VAT number provided by a store manager during the registration process is unique:
  Test related to the requirement R3 of the RASD document,
  Test steps:

  1. //todo description

  Test Passed

- will create a new store after the registration of store manager:
  Test related to the requirement R4 of the RASD document,
  Test steps:

  1. //todo description

  Test Passed

- A user (generalization of Store Manager) is able to log into the system by entering his personal credentials:
  Test related to the requirement R5 of the RASD document,
  Test steps:

1. //todo description

Test Passed

3.2 createUser

- will ask for the necessary information to register a user (generalization of Clupper):
  Test related to the requirement R1 of the RASD document.
  Test steps:

  1. click on the register button,
  2. check that there are input field for email, password, name and surname.

  Test Passed

- will register a person to the system as a user (generalization of Clupper):
  Test related to the requirement R1 of the RASD document.
  Test steps:

  1. click on the register button,
  2. insert random strings in the fields,
  3. click on register,
  4. log in using the same email and password,
  5. checks if the response from the server has an http status code of 200.

  Test Passed

- will verify that the email provided by a person during the registration process is unique (generalization of Clupper):
  Test related to the requirement R2 of the RASD document.
  Test steps:

  1. click on the register button,
  2. insert random strings in the fields,
  3. click on register,
  4. clear the cookies to invalidate the actual session,
  5. return to the home page,
  6. click on the register button,
  7. insert the same email in the email field,
  8. insert random strings in the other fields,
  9. click on register,
  10. checks that an error message is shown.

  Test Passed

- A user (generalization of Clupper) is able to log into the system by entering his personal credentials:
  Test related to the requirement R5 of the RASD document.
  Test steps:

  1. click on the register button,
  2. insert random strings in the fields,
  3. click on register,
  4. clear the cookies to invalidate the actual session,
  5. return to the home page,
  6. click on the sign in button,
  7. insert the same email and password,
  8. click on the login button,
  9. checks that the system land on the page /explore that is the main page of a logged in user.

  Test Passed

3.3 joinDigitalQueue

Before this tests a customer is registered and logged in. //todo update

- Select a supermarket:
  Test used to check if, after selecting a supermarket from the list, the clupper will be able to see its details: (name, number of customers already in line).
  Test steps:

  1. save the store name and number of customers of the first store in the list,
  2. click on the first store in the list,
  3. check if the name of the store and the number of custumer are present in the new page.

  Test Passed

- Join a queue:
  Test related to the requirements R6, R8 and R16 of the RASD document.
  Test steps:

  1. a new manager (with related store) is created,
  2. a new customer is created and logged in,
  3. click on the new store,
  4. click on the button to join the queue,
  5. click on the button to return to the main page(/explore),
  6. log out,
  7. log in with the same user,
  8. click on the button to see the tickets,
  9. check if the ticket is equal to the one showed in the registration step.

Test Passed

- Leave the queue:
  Test related to the requirement R9 of the RASD document.
  Test steps:

  1. a new manager (with related store) is created,
  2. a new customer is created and logged in,
  3. click on the new store,
  4. click on the button to join the queue,
  5. click on the button to return to the main page(/explore),
  6. log out,
  7. log in with the same user,
  8. click on the button to see the tickets,
  9. click on the button to delete the ticket,
  10. reclick on the button to see the tickets,
  11. checks that an error message that says that there are no ticket is shown.

  Test Passed

- TODO: Test name
  Test related to the requirement R7 of the RASD document.
  Test steps:

  1.

  Test Passed

3.4 joinPhysicalQueue

- Check if the tickets store address matches:
  Test related to the requirements R6 and R8 of the RASD document.
  Test steps:

  1. login as a manager,
  2. //todo I don't understand

  Test Passed

- Check if the ticket cannot be closed before it is printed:
  Test used to check if the system prevent the manager from closing the page with the ticket before printing it.
  Test steps:

  1. login as a manager,
  2. click on the button for issuing tickets,
  3. click on the close button,

4. checks that an error message is shown.

Test Passed

- Check if the ticket can be printed:
  Test used to check if the system allow the manager to print a new ticket.
  Test steps:

  1. login as a manager,
  2. click on the button for issuing tickets,
  3. check if exists a button for printing,
  4. click on the print button,
  5. check that the property calledOnce is set to true.

  Test Passed

- Check if the ticket can be deleted:
  Test related to the requirement R9 of the RASD document.
  Test steps:

  1. login as a manager,
  2. click on the button for issuing tickets,
  3. check if exists a button for printing,
  4. click on the print button,

  Test Passed

- Check if the newly created ticket increases number in line-up:
  Test used to check if creating a new ticket actually increment the size of the queue.
  Test steps:

  1. a new manager (with related store) is created,
  2. login as the new manager,
  3. checks that the number of customer in the queue is 0,
  4. click on the button for issuing tickets,
  5. click on the print button,
  6. click on the close button,
  7. checks that the number of customer in the queue is incremented to 1.

  Test Passed

- Check if manager can issue new ticket if the store is full:
  Test used to check if the system prevents the manager to issuing more tickets than the store limit.
  Test steps:

1. //todo I don't understand

Test Passed

3.5 storeOverview

Before this tests a manager is registered and logged in.

- The system is able to retrieve the number of customers currently in a store queue, The system is able to retrieve the number of customers currently inside a store:
  Test related to the requiremnts R31 and R32 of the RASD document.
  Test steps:

  1. login as a customer,
  2. get a ticket from the store of the manager registered at the beginning,
  3. //todo I don't understand

  Test Passed

- A store manager is able to change the maximum store capacity at any time:
  Test related to the requirement R17 of the RASD document.
  Test steps:

  1. //todo I don't understand

  Test Passed

- Store capacity can not be reduced below the current amount of customers inside:
  Test used to check if the system prevents the manager to change the maximum number of user in the store below the actual number of the customers in the store.
  Test steps:

  1. login as a customer,
  2. get a ticket from the store of the manager regstered at the beginning,
  3. login as a second customer,
  4. get a ticket from the store of the manager regstered at the beginning,
  5. login as a manager,
  6. scan the tickets,
  7. update the capacity of the store to 1,
  8. check if a error message is shown.

  Test Failed:
  There is no indication in the document on what should happen if the amount of shoppers in store are larger than the new capacity. We expect some sort of an error, but it just continues with an illegal text like 2/1.

3.6 ticketScan

- R19 - A store manager is able to scan a customer's ticket at the entrance of the store before letting him in
  R22 - The system is able to remove a customer's ticket from the store queue when it is scanned at the entrance
  R21 - The system is able to perform a validity check on a ticket scanned by a store manager and to inform him about the result
  Test is used to check whether scanning a customer's ticket allows access and updates the customer report accordingly.
  Test steps:

  1. Create a new store by registering as a manager
  2. Create and login as a new customer
  3. Generate a ticket for the customer
  4. Login as the store manager to validate the ticket
  5. Check that a success message is displayed and the numbers reported are updated to reflect the entrance.

  Test Passed

- R23 - A customer's ticket is scanned at the exit
  Test is used to check whether a customer's exit from the store is reflected on the store numbers.
  Test steps:

  1. Create a new store by registering as a manager
  2. Create and login as a new customer
  3. Generate a ticket for the customer
  4. Login as the store manager to validate the ticket twice, which corresponds to checking in and out
  5. Check that a success message is displayed and the numbers reported are updated to reflect the exit of the customer.

  Test Passed

- R19 - A store manager is able to scan a customer's ticket only if the number of customers currently inside does not exceed its maximum capacity
  R21 - The system is able to perform a validity check on a ticket scanned by a store manager and to inform him about the result
  Test is used to check whether the manager can not scan another customer in when the maximum store capacity is reached. Test steps:

  1. Create a new store with a capacity 1 by registering as a manager
  2. Create and login as a new customer
  3. Generate a ticket for the customer
  4. Create and login as another customer

5. Generate a ticket for that customer customer
6. Login as the store manager to validate both of the tickets
7. Check that an error message is displayed and the numbers reported do not exceed the set capacity.

Test Passed

- R23 - The system is able to invalidate a customer's ticket
  R21 - The system is able to perform a validity check on a ticket scanned by a store manager and to inform him about the result
  Test is used to detect that the customers ticket is invalid after exiting from the store
  Test steps:

  1. Create a new store by registering as a manager
  2. Create and login as a new customer
  3. Generate a ticket for the customer
  4. Login as the store manager to validate the ticket twice, which corresponds to checking in and out and trying an invalid ticket.
  5. Check that an error message is displayed and the numbers reported are updated to reflect the exit of the customer, but not a change for the third scan.

  Test Passed

- no one should be able to enter before his turn (We can't find a specific requirement for this)
  R21 - The system is able to perform a validity check on a ticket scanned by a store manager and to inform him about the result
  Test is used to verify that the ordering is assured, even though the requirements don't mention this, it is specified in the function definition and various parts of the RASD.
  Test steps:

  1. Create a new store with a capacity 1 by registering as a manager
  2. Create and login as a new customer
  3. Generate a ticket for the customer
  4. Create and login as another customer
  5. Generate a ticket for that customer customer
  6. Login as the store manager to validate the second ticket
  7. Check that an error message is displayed indicating that the customer is not at the head of the queue.

  Test Passed

4 Conclusion

In conclusion the analyzed system is robust and well done, even if some details need to be fixed. Some problems that we revealed are:

- The project's ITD document states that all addresses are in region Lombardia for simplicity. Therefore, all addresses that we create were in Milan. However, even though we have explicitly stated the province code (MI) in the address, the location database tried to match the address to Bergamo (BG).
- Following the previous issue, the Geolocation API failed to execute our addresses multiple times, we had to implement a retry logic for this.
- Also, regarding the same issue we have detected that a certain failure causes the app to crash completely. We had to alter the code to prevent the crash so that our tests are replicable (Line 24 in /src/controller/services/StoreManagerServices.js)
- The queue management is too simple since it just set the ETA to 5 minutes for each person before in the queue even if the maximum store capacity is not reached. This causes an unnecessary delay in the store exiting and doesn't consider the size of the store.
- There is no error when the manager reduces the shop's capacity below the amount of customers already present in the store. Even though this is not a regular case to happen, this is not enlisted as impossible in Domain Assumptions, and it is not mentioned as a requirement.

