Viktor Skachkov

Test Planning

Fontys S3-CB-01

14.04.2022

Versioning table

Version	Changes
1.1	I created the document and started writing tests.
1.2	I added the Mockito and WebMvc tests I have written.
1.3	I added the Cypress tests I have written for the frontend.

Table of contents

Contents

Prelude 5
Test strategy 6
User Acceptance Tests 7
    Frontend Tests 7
    Backend Tests 8
        Controller Tests 8
        Mockito Tests 8
Conclusion 9

Prelude

An owner wants to open a new pizzaria in town and has hired me to create an application to manage everything.

Below you will read the test cases I want you to perform. With these tests and your feedback I will be able to improve our application and make it 100 percent dummy proof and a viable app for all users.

Test strategy

There will be tests that test the functionalities in the backend and tests that test the functionalities in the frontend. Tests in the backend will be divided in two types – Mockito tests which will test the use cases in the business layer and WebMvc tests which will test all the methods in the controllers. There will be dozens of Mockito tests each of which will test a different use case and several controller test classes each of which will test a different controller and contain multiple tests. Both tests can imitate the database, so that the real one doesn't get filled with junk upon inspecting the application.

The frontend will be tested with Cypress tests. They go through the whole application and test all the functionalities in the process. There will be three big tests which will test the functionalities of the frontend and go through the three biggest processes of the react application – adding meals as an employee, ordering food and making reservations. Each of the tests will include an authentication and authorization either as a client or as an employee, so that the features can be used. I will try to make Docker containers for the frontend and the backend as well as a separate MYSQL database, so that the application can be tested without filling the real database with junk data.

User Acceptance Tests

Frontend Tests

Acceptance Requirement	Priority	Result
Checking if the user can log in	High	Success
Checking if the client can add a meal to the cart	High	Success
Checking if the client can increase or decrease the number of a meal	Low	Success
Checking if the client can add adding to the meal	Low	Success
Checking if the client can delete a meal from the cart	Medium	Success
Checking if the client can add cutleries to their order	Low	Success
Checking if the employee can complete an order	High	Success
Checking if the client can make a reservation	Medium	Success
Checking if the employee can reject a reservation	Low	Success
Checking if the employee can approve a reservation by assigning a table to it	High	Success
Checking if the meals can be filtered by category	Low	Success
Checking if the employee can add a new meal	High	Success
Checking if the employee can update a meal	Medium	Success
Checking if the employee can delete a meal	Medium	Success

Backend Tests

Controller Tests

- Cart Controller Tests: 7/7 success rate
- User Controller Tests: 3/4 success rate
- Order Controller Tests: 13/16 success rate
- Reservation Controller Tests: 0/9 success rate

Mockito Tests

- 21/21 success rate

Success Rate

The success rate of the tests is 77.1%.

Conclusion

The backend tests have a 77% success rate while the frontend tests have a 100% success rate. All of the tests for the reservation controller fail for some reason but the methods themselves work properly. Overall, most of the tests pass and the reason why some don't pass is because they depend on a lot of different use cases which can make the testing process difficult.