|  |
| --- |
| Viktor Skachkov |
| Test Planning |
| Fontys S3-CB-01 |

|  |
| --- |
| 14.04.2022 |

Table of contents

Contents

[Test strategy 3](#_Toc100857363)

[Cart Tests 4](#_Toc100857364)

[Order Tests 5](#_Toc100857365)

[Reservation Tests 6](#_Toc100857366)

[User Tests 7](#_Toc100857367)

[Conclusion 8](#_Toc100857368)

# 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

I will include tests with assertions that show if two different values are equal or not to test if the methods return the correct values. There will be one single test for each method. The tests will have the exact same names as the methods they are testing to avoid any confusions.

The tests won’t test the real classes and methods but instead they will mock Administration, Data Helper and Controller classes which will contain the exact same methods and use but will be connected to another database to prevent the real database from being filled with junk data.

The tests will be divided in 4 different classes, one for each part of the data that should be tested (CartTests, OrderTests, ReservationTests and UserTests).

The CartTests class will test the methods from the MockCartController, the OrderTests class will test the methods from the MockOrderController, the ReservationTests class will test the methods from the MockReservationController and the UserTests class will test the methods from the MockUserController.

For now, the tests will mostly check if the methods that send data actually send the correct data because this will be necessary for displaying elements on the frontend.

## 

## Cart Tests

The pizzeria will contain a cart where the user can save meals to order later.

GetCart() – Tests if the whole cart can be called. Result: succeeded

GetOrderedMeal() – Tests if a specific ordered meal can be called via index. Result: succeeded

GetMeals() – Tests if all the meals can be called. Result: failed

## Order Tests

GetAddings() – Tests if all the addings can be called. Result: succeeded

GetMeals() – Tests if all the meals can be called. Result: succeeded

GetDeserts() – Tests if all the deserts can be called. Result: succeeded

GetSalads() – Tests if all the salads can be called. Result: succeeded

GetPizza() – Tests if all the pizzas can be called. Result: succeeded

GetPasta() – Tests if all the pasta meals can be called. Result: succeeded

GetMeal() – Tests if a specific meal can be called via index. Result: succeeded

GetCategories() – Tests if all the categories can be called. Result: succeeded

## Reservation Tests

GetReservationRequest() – Tests if a specific reservation request can be called via index. Result: succeeded

GetTables() – Tests if all the tables can be called. Result: succeeded

GetTable() – Tests if a specific table can be called via index. Result: succeeded

GetUnappprovedReservationRequests() – Tests if all the unapproved reservation requests can be called. Result: succeeded

GetApprovedReservationRequests() – Tests if all the approved reservation requests can be called. Result: succeeded

## User Tests

GetUser() – Tests if a specific user can be called via username and password. Result: succeeded

GetWorkers() – Tests if all the workers can be called. Result: succeeded

GetClients() – Tests if all the clients can be called. Result: succeeded

## Conclusion

All of the tests except for one were successful. The problem must be found and fixed but so far all attempts were unsuccessful.