UX Document

Individual Track Assignment

14/06/2022 Eindhoven
Version: 1.0

Esther Wolfs: 3329984

Tutor:
Tim Kurvers
Márcio Paixão Dantas

Version history

Version	Date	Author(s)	Changes	State
0.1	13/06/2022	Esther Wolfs	Start writing UATs, add testing strategy	Started

1. Introduction

In this document the testing strategy will be explained. It will also contain the User Acceptance Tests. The results of these tests will be covered in a separate document.

2. User Acceptance Tests

This is a list of the User Acceptance Tests.

1. Users should be able to view featured rooms on the homepage without being logged in
2. Users should be able to view all rooms on the “all rooms” page
3. Users should be able to login
4. Users should be redirected to their profile page after login
5. Users should be able to view their personal information after login
6. Users should be able to update their personal information
7. Users should be able to see the reservation form after login
8. Users should be able to make a reservation
9. Users should not be able to select a room that has 0 amount in hotel
10. Users should be able to create an account

11. Employees should be able to view all reservations
12. Employees should be able to check reservations in/out
13. Employees should be able to add new rooms
14. Employees should be able to update rooms

15. Admins should be able to create new employees
16. Admins should be able to view all employees

3. Test Strategy

3.1 Unit Tests

To make sure all methods work as expected, unit tests will be written for all methods in the different layers. The unit tests will only be made for the backend. All five of the service classes in the business layer are tested. Mockito is used for mock data, so the actual database will not be flooded with test data. The mock instances of the dependencies will be injected in the mock service class that is being tested. Both happy flows as well as expected exceptions that should be thrown are tested.

This is also done for the controller layer. When authentication or authorisation is needed, mockito will create a mock user. For most of the controller layer only the happy flows are tested.

Only code that is written by me is being tested, so not the code that was provided by Márcio Paixão Dantas is not being tested, for example the access token classes and the security configuration classes. The reason for this is because I didn’t write this code, otherwise I would be testing someone else's code.

There are no unit tests for the repository layer. This application uses Java Persistence API for the repository. There are no custom SQL queries being used, so it would not be relevant to make unit tests for the repository layer.

Automated testing is set up in the CI/CD pipeline, which means the unit tests automatically run whenever code is pushed to the main branch.

3.2 Integration Tests

There are no integration tests for this application, as I am not yet sure how to make them.