|  |  |  |
| --- | --- | --- |
| Project Team |  |  |
| Rozalina Miladinova 3285588  Aleksandar Georgiev 3229742  Nidhi Sharma 3233006  Nguyen Bao Quoc 3477347  Fares Alsalama 3396029 |  |  |



**Test document**

|  |
| --- |
|  |

Contents

[Introduction 3](#_Toc27334951)

[Goal 3](#_Toc27334952)

[Way of testing 3](#_Toc27334953)

[Information about the tests 1](#_Toc27334954)

## Introduction

This document will provide information about the plan which will be followed during the testing of the Airport baggage simulation application. In the sections below can be found:

* Information about the way of testing
* Information about the tests
* Results of the tests

## Goal

The main goal for testing is to check and verify the reliability and integrity of the application, and to minimize the risk of failures when the application is in production. Therefor testing must be done thoroughly. This test document is a guidance and reference to perform the testing and will also show the results of the tests.

## Way of testing

The method that is used for testing the application is the unit test. Unit testing is a software testing method by which individual units of source code, with associated control data, usage procedures, and operating procedures, are tested to determine whether they are implemented accordingly with the requirements or not. We choose to see each method as an unit, instead of a complete class as an unit. Scenarios are used with the unit to test for trustworthy behavior. Multiple scenarios can be coupled with an unit, and multiple units can be coupled with a scenario. Because of the limited time we had for the testing, we focused the testing on the most important classes.

## Information about the tests

A summary of the tests.

TimerTracker tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr | Target {class/method(s)} | Testing name {method} | Scenario | Expected outcome | Actual Outcome |
| 1 | TimerTracker / TimerTracker() | TimerService\_ShouldRunNewTimer\_OnRunNewTimer() | New timer is created | Time is greater than 200 | Time is greater than 200 |
| 2 | TimerTracker / GetTicksSinceSimulationStart() GetTimeSinceSimulationStart() | TimerService\_ShouldReturnSameTimeSinceStart\_FromBothMethods() | TimerTracker is asked to give the time since the simulation started | Both methods should return the same time | The same time is returned |
| 3 | TimerTracker / ConvertMillisecondsToTimeSpan(int milliseconds) | TimerService\_ConvertMillisecondsToTimeSpan\_ShouldConvertCorrectly() | Time in milliseconds is converted to timespan | Converted time is one second | One second |

NodeCreationService tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr | Target {class/method(s)} | Testing name {method} | Scenario | Expected outcome | Actual Outcome |
| 4 | NodeCreationService / CreateCheckinDesk() | ICheckInDesk\_Create() | Creating checkin | Created checkin | checkin |
| 5 | NodeCreationService / CreatePrimarySecurity() | CreatePrimarySecurity\_Create() | Creating primary security | Created primary security | Primary security |
| 6 | NodeCreationService / CreateConveyorOneToOne() | CreateConveyorOneToOne\_Create() | Creating coveyor | Created conveyor | Conveyor |
| 7 | NodeCreationService / CreateMda() | CreateMda\_Create() | Creating main distribution area | Created MDA | MDA |
| 8 | NodeCreationService / CreateDropoff() | CreateCheckinDispatcher\_Create() | Creating dropoff | Created dropoff | Dropoff |

TransportingNode tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr | Target {class/method(s)} | Testing name {method} | Scenario | Expected outcome | Actual Outcome |
| 9 | ConveyorOneToOne/ConveyorOneToOne() | ConveyorAssignLengthTest() | Assign length to Conveyor | ConveyorOneToOne Length is set to 3 | ConveyorOneToOne with length 3 |
| 10 | ConveyorOneToOne/ConveyorOneToOne() | ConveyorAssignNodeIdTest() | Assign a name to Conveyor | ConveyorOneToOne NodeId is set to conv | Name of ConveyorOneToOne Nodeid is conv |
| 12 | TransportingNode/SetSingleNextNode() | ConveyorSetDestinationTest() | Next destination node for conveyor is set to a specific node | ConveyorOneToOne destination is set to PrimarySecurity | Destination of ConveyorOneToOne is PrimarySecurity |
| 13 | ConveyorOneToOne/PassBaggage() | ConveyorSetStatueBusyTest() | When ConveyorOneToOne.Length is less than 2 NodeStatue should be set to Busy | NodeStatus is equal to busy | NodeStatus is busy |
| 15 | ConveyorOneToOne/PassBaggage() | ConveyorPassBaggageTest() | Pass Baggage from specified conveyor to next destination | Baggage is passed and no exception is thrown | Baggage is passed without exception |
| 16 | ConveyorOneToOne/TransportingNode /Stop() | ConveyorStopTest() | Transporting baggage from conveyor is stopped | The system throw argument exception within PassBaggage method | Argument exception is thrown |
| 17 | ConveyorOneToOne/PassBaggage() | ConveyoryIsBusyTest() | When passing the baggage while Convery is busy argument exception is set | The system throw an argument exception within passBaggage method | Argument exception is thrown |