Politecnico di Milano

Master in Computer Science and Engineering

Software Engineering 2 Project: myTaxiService

INTEGRATION TEST PLAN

Authors:
Filippo Leporati
Danilo Fusi

Professor:
Elisabetta Di Nitto

1 TABLE OF CONTENTS

2 INTRODUCTION

This document describes the Integration Test Plan (ITP) for the myTaxiService project. It contains the description of the integration tests for the project.

2.1 REVISION HISTORY

This paragraph records all revisions to the document.

2.1.1 RASD revisions

- Version 1: initial paper.
- Version 2:
  1. Adjustments of the class diagram. The field "taxi_driver_code" in taxi driver class has been canceled. The field "taxi_arrival_time" has been moved from taxi class to request class.
  2. Adjustments of the functional requirements. [FR10] has been modified in order to support a better management of system user and login
- Version 3:
  1. Section 4.1.5 "Planning chart" was added. It contains a Gantt chart, which describes the project's phases.
- Version 4:
  1. Introduction of [FR31]

2.1.2 Document Design revisions

- Version 1: initial paper.
- Version 2:
  1. Gantt chart moved in Rasd document version 3.
  2. Section "Other design decision" modified. Now it describes the management of GPS tracking and taxi position in the system.
  3. In the "Runtime view" a sequence diagram, which describes the creation of a new request, has been added.
- Version 3:
  1. Modification of the interfaces between the components in the application server [see chapter 3.2.3.2]
  2. Introduction of Google Maps API component, in the project architecture.

2.2 PURPOSE AND SCOPE

2.2.1 Purpose

This document describes the plans for testing the integration of the components designed in the previous documents. The purpose of this document is to test the interfaces between the components. Moreover this document needs to explain to the development team what to test, in which sequence, which tools are needed for testing, which stubs/drivers/oracles need to be developed.

2.2.2 Scope

myTaxiService application, which is an information service, aims at optimizing the taxi service of a large city.
In particular, it wants to:
Business Goal 01) Simplify the access of passengers to the service.
Business Goal 02) Guarantee a fair management of taxi queues.
Passengers can request a taxi either through a web application or through a mobile app. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time.
Taxi drivers use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain call.
myTaxiService will provide general functionalities for managing:

- **Profile**
  myTaxiService will manage personal data of the different types of users. Users must be registered to use the service, both for daily users and administrators.
- **Connections**
  myTaxiService will manage the network of connections between users and taxi drivers.
- **Taxis distribution**
  myTaxiService system guarantees a fair management of taxi queues and distribution in the city.
- **Users**
  myTaxiService will manage registering, logging in/out users.

2.3 LIST OF DEFINITIONS AND ABBREVIATIONS

2.3.1 Definitions

Keyword	Definitions
Administrator	The administrator of the system is the person allowed to manage the entire system.
User	Citizen or taxi driver who interact with the application.
Server	Refers to the myTaxiService server.
Taxis' queue	Data structure to manage taxis in the city.
Request	A passenger makes a request when he wants immediately a taxi.
Reservation	A passenger makes a reservation when he does not want taxi immediately, but he wants to select a different time or day.

2.3.2 Abbreviations

Acronym or abbreviation	Definition
RASD	Requirements Analysis and Specification Document
APP	myTaxiService
NFR	Non-functional Requirements
QA	Quality Attributes
FR	Functional Requirement
DBMS	Database Management System
AS	Application Server
JEE	Java Enterprise Edition
QoS	Quality of Service
GUI	Graphical User Interface

2.4 LIST OF REFERENCE DOCUMENTS

1. Analysis document: RASDv3.pdf
2. Document Design: DDv2.pdf
3. Alloy document: 2015_project_Alloy_myTaxiService.als
4. Arquillian: http://arquillian.org
5. JUnit: http://junit.org

3 INTEGRATION STRATEGY

3.1 ENTRY CRITERIA

Integration Test shall begin when the following criteria are met:

1. All the unit tests have been correctly executed and every test has passed successfully.
2. The bug tracking system is in place and available for use by the engineering services and system engineering teams.
3. The System Operations and Administration Team has configured the Integration Test clients and servers for testing. The Test Team has been provided with appropriate access to these systems.
4. The Development Team has placed at least two communicating components to be released for Integration Testing under formal, automated source code and configuration management control.
5. The Development Team or the Release Engineering team has prepared a test release, containing at least two communicating components, both of which have completed Component Testing.
6. The Development Team has prepared release notes that document the functionality in the test release and any known bugs and limitations.

3.2 ELEMENTS TO BE INTEGRATED

This table below list all the components and subcomponent which are going to be tested during the phase of integration testing.

This component derive from the Document Design paper [paragraph 3.2].

Module Number	Module Name	Module Description
1.	Client	The Client module is the software running on the cell phone emulator
2.	Web Server	Web Tier is composed of the web beans, which are part of JavaServer Faces MVC pattern.
3.	Application Server	The business logic tier is composed of all the application logic;
4.	Database	The database is composed of the tables generated using assumptions and needs of the project
5.	User Pages	Web Pages responsible for everything related to users.
6.	UserManagedBean	Beans responsible for everything related to users.
7.	Profile Pages	Pages responsible for everything related to profile.
8.	ProfileManagedBean	Beans responsible for everything related to profile.
9.	Request Pages	Pages responsible for everything related to requests.
10.	RequestManagedBean	Beans responsible for everything related to requests.
11.	Reservation Pages	Pages responsible for everything related to reservations.
12.	ReservationManagedBean	Beans responsible for everything related to reservations.
13.	User Bean	Bean in charge for all functionalities related to users.
14.	UserInterfaceBean	Interface instantiated every time communication between beans is needed.
15.	Request Bean	Bean in charge for all functionalities related to taxi request.

16.	RequestBeanInterface	Interface instantiated every time communication between beans is needed.
17.	Reservation Bean	Bean in charge for all functionalities related to taxi reservations.
18.	ReservationBeanInterface	Interface instantiated every time communication between beans is needed.
19.	Taxi Bean	Bean in charge for all functionalities related to taxi management and distribution.
20.	TaxiBeanInterface	Interface instantiated every time communication between beans is needed.
21.	PositionManager	Bean in charge for all functionalities related to manage user and taxi positions.
22.	Google Maps API	External component dedicated to elaborate the sent position.

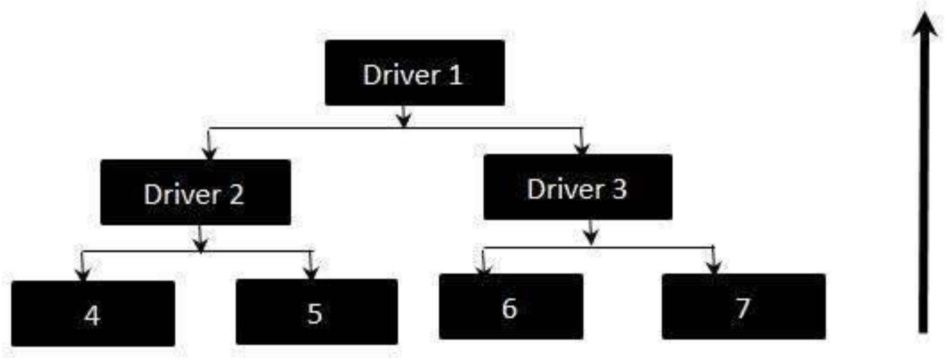
3.3 INTEGRATION TEST STRATEGY

3.3.1 Bottom-up strategy

The items to be tested consist of the integration of the code modules developed, for the myTaxiService project. For testing we choose the bottom-up approach. This means that integration testing starts at the bottom level. The integration tests described in this document are at the component level.

In this approach testing is conducted from sub module to main module, if the main module is not developed a temporary program called DRIVERS is used to simulate the main module.

The objective of the Integration Test phase is to find bugs in the system under test, in the interfaces between components and in the system as a whole, respectively. We intend to do this by running a set of manual and automated test cases against each test release.



This strategy has been choose analysing different pros and cons.

- Advantages:

- Advantageous if major errors occur toward the bottom of the program.
- Test conditions are easier to create.
- Observation of test results is easier.

- Disadvantages:

- Driver Modules must be produced.
- The program as an entity does not exist until the last module is added.

Top-down analysis could be performed but it would bring these undesired disadvantages:

- Stub modules must be produced
- Stub modules are often more complicated than they first appear to be.
- Before the I/O functions are added, representation of test cases in stubs can be difficult.
- Test conditions may be impossible or very difficult to create.
- Observation of test output is more difficult.
- Allows one to think that design and testing can be overlapped.
- Induces one to defer completion of the testing of certain modules.

A driver is a main program that accepts test data and passes this test to the component to be tested and prints relevant results. Drivers must be kept for future integration tests.
Because the two components involved in a specific integration test are already unit tested, the drivers that are made for them can be used in the integration test. This way testing can be performed more efficient.

3.3.2 Test cycles

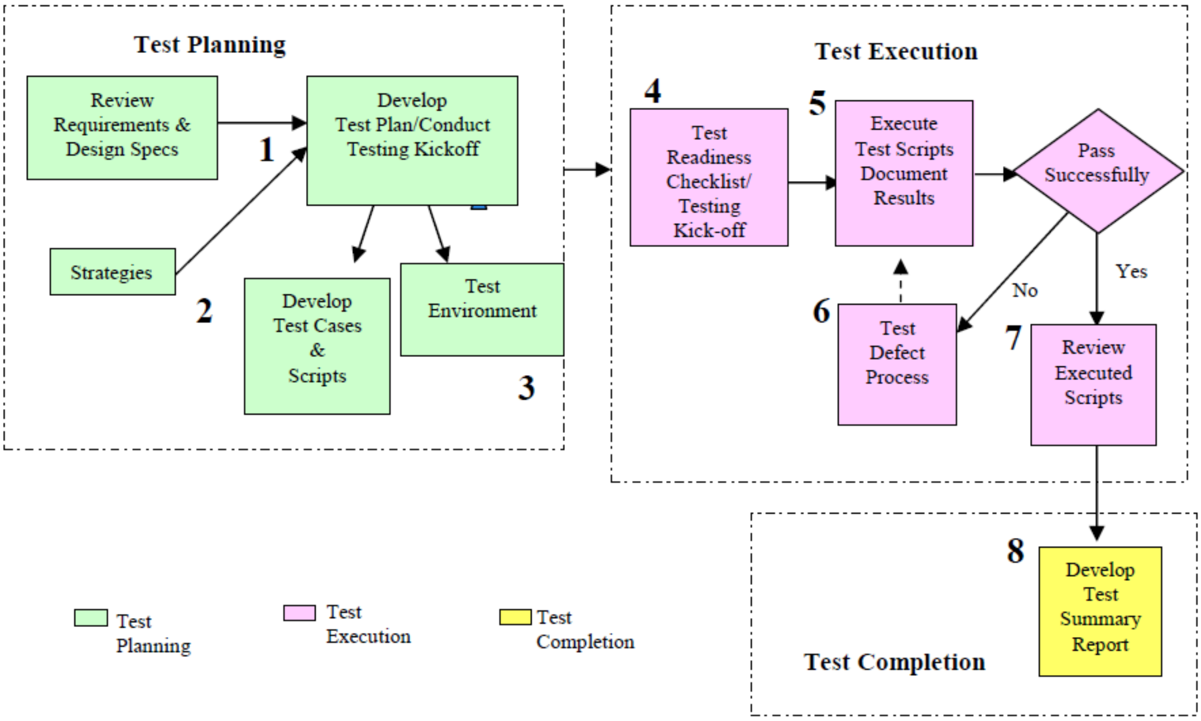
At the beginning of each test cycle, the team shall assign a “basket” of test cases to each tester. Each tester’s basket shall be different from one test cycle to the next, to ensure that any invalid assumptions on one tester’s part do not result in a test escape. Once assigned their basket of tests, these testers will follow the test steps outlined to execute each test case, and repeat the process until they have finished their list. If they empty their basket before to the end of a cycle, they shall assist other testers by coordinating a reassignment of the uncompleted test to themselves. If all tests are completed, the test team shall report all the bugs that have been found.

At the highest level, the test strategy can be summarized as follows:

- Develop automated and manual tests to cover all the quality risks identified, focusing on behavioural factors observable at some user interface.
- Add test data or conditions within existing cases or new test cases to cover critical customer usage profiles and customer data.
- Use exploratory testing in areas that were not addressed previously and that appear, due to test results or intuition, to be at high risk of bugs. Update or define new test cases to cover found bugs.
- Run tests across a “customer-like” mix of server, network, and client configurations (the test environment).

3.3.3 Testing Process

The diagram below outlines the Test Process approach that will be followed.


Test Planning
Review Requirements & Design Specs
Develop Test Plan/Conduct Testing Kickoff
1
Strategies
2
Develop Test Cases & Scripts
Test Environment
3
Test Execution
4
Test Readiness Checklist/ Testing Kick-off
5
Execute Test Scripts Document Results
Pass Successfully
6
Test Defect Process
No
Yes
7
Review Executed Scripts
8
Develop Test Summary Report
Test Completion
Test Planning
Test Execution
Test Completion

Process	Step	Action
Test Planning Phase	1.	Develop Test Plan • Review application requirements and/or Design Specifications, if applicable. • Develop a list of scope deliverables in the Test Plan. • Perform risk analysis, identifying critical functions. • Review and document approaches and processes for testing • Identify prerequisites (training for testers, readiness of test data and testing environment) before testing can begin.
	2.	Develop Test Scripts • Trace requirements to Test Case. • Write detailed test scripts and results expected from each script. • Submit test scripts for review and approval before the formal execution
	3.	Test Environment • Ensure that test environment is set up or verified. • Ensure that all test accounts have been established and are active. • Ensure installation qualification for application software program is complete.
Test Execution	4.	• Testing Kick-off

Phase	5.	Execute Test Scripts and Capture actual Test Results • Each executed test script must be signed and dated by, so that the test can be properly documented and that the test results support the overall pass or fail status of the test case.
	6.	Test Defect Process • If script fails, log test defects to document discrepancies where test results are not as expected. • Depending on the outcome of the test defect, re-execute the script after the software or script has been modified.
	7.	Review Executed Test Scripts • Ensure test results and/or documentation aligns with test steps
Test Completion Phase	8.	Develop Test Summary Report • Document test execution results in Test Summary Report. • Submit the Test Summary Report for approval.

3.3.4 Risks and Mitigation Strategies

Potential risks to the testing efforts and the impact of the potential risks are categorized as medium, high or low. The following table shows the impact and mitigation strategies for each potential risk:

Risk	**Probability of Risk (Low, Medium, High)**	**Impact of Risk (Low, Medium, High)**	**Mitigation Strategy**
The executor of the test script may also be the creator due to resource constraints.	M	M	Use other resources identified in the document for the execution of the test scripts.
Test Scripts are not approved before test execution.	M	H	The team will review all scripts as they are completed before the execution.
Development is not frozen before the integration testing.	M	H	The development team will make the test team aware of any code changes to the testing environment and ensure no further code changes/ development is in progress during test execution.
The test environment is not the same configuration/set up as the production environment.	M	H	The system testing effort will use the same configuration/set up as the production environment.

3.4 SEQUENCE OF COMPONENT & FUNCTION INTEGRATION

Figures 1.1 below shows the components of the myTaxiService application. These figure is derived from the Document Design, chapter 3.2. The link and the number represent the order of integration testing.

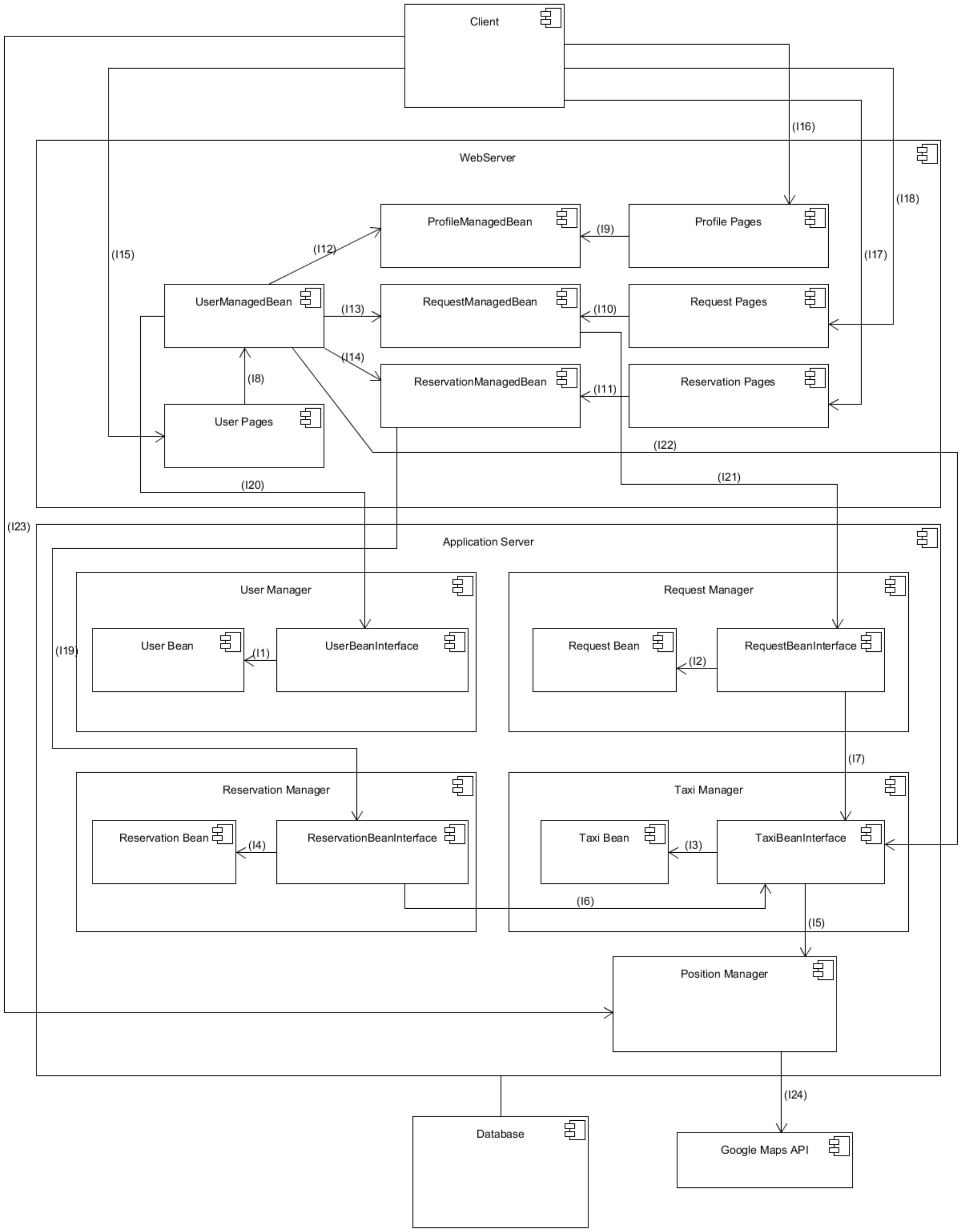
According to the strategy proposed in the previous section, the integration tests proceed from the bottom to the top. So first, lower subcomponent are tested and integrated and then these sets of subcomponent are integrated with the higher ones.

The integration test follows these principles. First the subcomponents are integrated, i.e. the components of the Web Server and the Application Server; then the higher component are integrated to each other, i.e. the clients, the Server and the Database. The number written on the link gives an order for the integration test. Of course some test may be executed in parallel to speed up the process, without compromising the efficiency of the tests.

Each component at lower hierarchy is tested individually and then the components that rely upon these components are tested.
The lower components, i.e. the leaves of the integration tree, are the following:

- WebServer subcomponents
  1. ProfileManagedBean
  2. RequestManagedBean
  3. ReservationManagedBean
- Application Server subcomponents
  1. UserBean
  2. RequestBean
  3. ReservationBean
  4. TaxiBean
  5. PositionManager

All the other components are linked with these lower components in the test plan. The arrow in the picture below represents the dependencies between two components. The arrowhead indicates that the pointed component depends, in the integration testing, on the other component.


Client
WebServer
(I16)
(I18)
(I17)
(I15)
ProfileManagedBean
(I9)
Profile Pages
(I12)
UserManagedBean
(I13)
RequestManagedBean
(I10)
Request Pages
(I14)
ReservationManagedBean
(I11)
Reservation Pages
(I8)
User Pages
(I22)
(I21)
(I20)
(I23)
Application Server
User Manager
User Bean
(I1)
UserBeanInterface
Request Manager
Request Bean
(I2)
RequestBeanInterface
(I19)
(I7)
Reservation Manager
Reservation Bean
(I4)
ReservationBeanInterface
Taxi Manager
Taxi Bean
(I3)
TaxiBeanInterface
(I6)
(I5)
Position Manager
(I24)
Database
Google Maps API

ID	Integration Test	Paragraph
I1.	UserBeanInterface -> UserBean	4.1
I2.	RequestBeanInterface -> RequestBean	4.2
I3.	TaxiBeanInterface -> TaxiBean	4.3
I4.	ReservationBeanInterface -> ReservationBean	4.4
I5.	TaxiBeanInterface -> PositionManager	4.5
I6.	ReservationBeanInterface -> TaxiBeanInterface	4.6
I7.	RequestBeanInterface -> TaxiBeanInterface	4.7
I8.	UserPages -> UserManagedBean	4.8
I9.	Profile Pages -> ProfileManagedBean	4.9
I10.	Request Pages -> RequestManagedBean	4.10
I11.	Reservation Pages -> ReservationManagedBean	4.11
I12.	UserManagedBean -> ProfileManagedBean	4.12
I13.	UserManagedBean -> RequestManagedBean	4.13
I14.	UserManagedBean -> ReservationManagedBean	4.14
I15.	Client -> UserPages	4.15
I16.	Client -> ProfilePages	4.16
I17.	Client -> ReservationPages	4.17
I18.	Client -> RequestPages	4.18
I19.	ReservationManagedBean -> ReservationBeanInterface	4.19
I20.	UserManagedBean -> UserBeanInterface	4.20
I21.	RequestManagedBean -> RequestBeanInterface	4.21
I22.	UserManagedBean -> TaxiBeanInterface	4.22
I23.	Client -> Position Manager	4.23
I24.	PositionManager -> Google Maps API	4.24

4 INDIVIDUAL STEPS AND TEST DESCRIPTION

This section details the black box tests needed to ensure that each component of myTaxiService is interacting with other components as expected.

4.1 INTEGRATION TEST I1

Test Case Identifier	I1-T1
Test Case Name	Check method called
Test Case Description	The purpose of this test is to check whether the correct method from the interface is called or not.
Item(s) to be tested	
1	UserBeanInterface
2	UserBean
Specifications	
Input	**Expected Output/Result**
-	The correct method is called.
Procedural Steps	
1	UserBeanInterface calls the method of UserBean
2	Check if the correct method is called
3	Repeat this test for all methods of UserBean

4.2 Integration Test I2

Test Case Identifier	I2-T1	
Test Case Name	Check method called	
Test Case Description	The purpose of this test is to check whether the correct method from the interface is called or not.	
Item(s) to be tested		
1	RequestBeanInterface	
2	RequestBean	
Specifications		
Input		**Expected Output/Result**
-		The correct method is called.
Procedural Steps		
1	RequestBeanInterface calls the method of RequestBean	
2	Check if the correct method is called	
3	Repeat this test for all methods of RequestBean	

4.3 Integration Test I3

Test Case Identifier	I3-T1	
Test Case Name	Check method called	
Test Case Description	The purpose of this test is to check whether the correct method from the interface is called or not.	
Item(s) to be tested		
1	TaxiBeanInterface	
2	TaxiBean	
Specifications		
Input	**Expected Output/Result**	
-	The correct method is called.	
Procedural Steps		
1	TaxiBeanInterface calls the method of TaxiBean	
2	Check if the correct method is called	
3	Repeat this test for all methods of TaxiBean	

4.4 INTEGRATION TEST I4

Test Case Identifier	I4-T1
Test Case Name	Check method called
Test Case Description	The purpose of this test is to check whether the correct method from the interface is called or not.
Item(s) to be tested	
1	ReservationBeanInterface
2	ReservationBean
Specifications	
Input	**Expected Output/Result**
-	The correct method is called.
Procedural Steps	
1	ReservationBeanInterface calls the method of ReservationBean
2	Check if the correct method is called
3	Repeat this test for all methods of ReservationBean

4.5 Integration Test I5

Test Case Identifier	I5-T1
Test Case Name	Get position of a taxi
Test Case Description	The purpose of this test is to get the current position of a taxi
Item(s) to be tested	
1	TaxiBeanInterface
2	PositionManager
Specifications	
Input	**Expected Output/Result**
-	The position of the taxi is returned after step (2).
Procedural Steps	
1	TaxiBeanInterface requests the position of a taxi to the PositionManager
2	PositionManager elaborates the request and returns the position

4.6 Integration Test I6

Test Case Identifier	I6-T1
Test Case Name	Assign a taxi to the reservation
Test Case Description	The purpose of this test is to assign a taxi to the reservation
Item(s) to be tested	
1	ReservationBeanInterface
2	TaxiBeanInterface
Specifications	
Input	**Expected Output/Result**
ReservationBeanInterface receives the reservation	A taxi is assigned to the reservation
Procedural Steps	
1	ReservationBeanInterface send the reservation to the TaxiBeanInterface
2	TaxiBeanInterface elaborates the request, assigned a taxi to the reservation and notifies ReservationBeanInterface

Test Case Identifier	I6-T2
Test Case Name	Deletion of a reservation
Test Case Description	The purpose of this test is to release the taxi assigned to the reservation
Item(s) to be tested	
1	ReservationBeanInterface
2	TaxiBeanInterface
Specifications	
Input	**Expected Output/Result**
ReservationBeanInterface receives the reservation	The taxi assigned is released from the reservation
Procedural Steps	
1	ReservationBeanInterface send the reservation to the TaxiBeanInterface
2	TaxiBeanInterface releases the taxi from the reservation and notify ReservationBeanInterface

4.7 Integration Test I7

Test Case Identifier	I7-T1
Test Case Name	Assign a taxi to the request
Test Case Description	The purpose of this test is to assign a taxi to the request
Item(s) to be tested	
1	RequestBeanInterface
2	TaxiBeanInterface
Specifications	
Input	**Expected Output/Result**
RequestBeanInterface receives the request	A taxi is assigned to the request
Procedural Steps	
1	RequestBeanInterface send the request to the TaxiBeanInterface
2	TaxiBeanInterface elaborates the request, assigned a taxi to the request and notify RequesBeanInterface

Test Case Identifier	I7-T2
Test Case Name	Deletion of a request
Test Case Description	The purpose of this test is to release the taxi assigned to the request
Item(s) to be tested	
1	RequestBeanInterface
2	TaxiBeanInterface
Specifications	
Input	**Expected Output/Result**
RequestBeanInterface receives the request	The taxi assigned is released from the request
Procedural Steps	
1	RequestBeanInterface send the request to the TaxiBeanInterface
2	TaxiBeanInterface releases the taxi from the request and notify RequestBeanInterface

4.8 Integration Test I8

Test Case Identifier	I8-T1
Test Case Name	Login success
Test Case Description	The purpose of this test is to ensure that UserManagedBean can handle the case of successful login.
Item(s) to be tested	
1	UserPages
2	UserManagedBean
Specifications	
Input	**Expected Output/Result**
1.UserPages sends the login form. 2. UserManagedBean receive the login form.	1. UserManagedBean notifies the transition to Login Successful page after step (3).
Procedural Steps	
1	User pages send the login parameters to UserManagedBean
2	Parameters are elaborated by the application server
3	UserManagedBean change its status and notify

Test Case Identifier	I8-T2
Test Case Name	Incorrect Login Failure
Test Case Description	The purpose of this test is to ensure that UserManagedBean can handle the case of unsuccessful login.
Item(s) to be tested	
1	UserPages
2	UserManagedBean
Specifications	
Input	**Expected Output/Result**
1.UserPages send the login form. 2. UserManagedBean receive the login form.	1. UserManagedBean notifies the transition to Login Incorrect page after step (3).
Procedural Steps	
1	User pages send the login parameters to UserManagedBean
2	Parameters are elaborated by the application server
3	UserManagedBean change its status and notify

Test Case Identifier	I8-T3
Test Case Name	Add an account
Test Case Description	The purpose of this test is to ensure that UserManagedBean can handle the case when an account is added in the system.
Item(s) to be tested	
1	UserPages
2	UserManagedBean
Specifications	
Input	**Expected Output/Result**
1. UserManagedBean receive the registration form. 2. UserManagedBean receive the login form, after registration	1. UserManagedBean notifies the transition to Registration Successful page after step (3). 2. UserManagedBean notifies the transition to Login Successful page after step (6).
Procedural Steps	
1	User pages send the registration parameters to UserManagedBean
2	Parameters are elaborated by the application server
3	UserManagedBean change its status and notify
4	User pages send the login parameters to UserManagedBean
5	Parameters are elaborated by the application server
6	UserManagedBean change its status and notify

Test Case Identifier	I8-T4
Test Case Name	Add a taxi driver
Test Case Description	The purpose of this test is to ensure that UserManagedBean can handle the case when a new taxi driver is added in the system by the administrator.
Item(s) to be tested	
1	UserPages
2	UserManagedBean
Specifications	
Input	**Expected Output/Result**
1. UserManagedBean receive the registration form. 2. UserManagedBean receive the login form, after registration.	1. UserManagedBean notifies the transition to Registration Successful page after step (3). 2. UserManagedBean notifies the transition to Login Successful page after step (6).
Procedural Steps	
1	User pages send the registration parameters to UserManagedBean
2	Parameters are elaborated by the application server
3	UserManagedBean change its status and notify
4	User pages send the login parameters to UserManagedBean
5	Parameters are elaborated by the application server
6	UserManagedBean change its status and notify

4.9 INTEGRATION TEST I9

Test Case Identifier	I9-T1
Test Case Name	Remove an account
Test Case Description	The purpose of this test is to ensure that ProfileManagedBean can handle the case when an account is removed from the system.
Item(s) to be tested	
1	ProfilePages
2	ProfileManagedBean
Specifications	
Input	**Expected Output/Result**
1. ProfileManagedBean receive a cancellation request from ProfilePages	1. ProfileManagedBean notifies the transition to Registration Successful page after step (3). 2. ProfileManagedBean notifies the transition to Login Incorrect page after step (6).
Procedural Steps	
1	Profile pages send the cancellation request to ProfileManagedBean
2	Parameters are elaborated by the application server
3	ProfileManagedBean change its status and notify
4	User pages send the login parameters to ProfileManagedBean
5	Parameters are elaborated by the application server
6	ProfileManagedBean change its status and notify

Test Case Identifier	I9-T2
Test Case Name	Change password
Test Case Description	The purpose of this test is to ensure that ProfileManagedBean can handle the case when an account wants to change its password.
Item(s) to be tested	
1	ProfilePages
2	ProfileManagedBean
Specifications	
Input	**Expected Output/Result**
1. ProfileManagedBean receive a password modification request from ProfilePages	1. ProfileManagedBean notifies the transition to Modification Successful page after step (3). 2. ProfileManagedBean notifies the transition to Login Incorrect page after step (6). 3. ProfileManagedBean notifies the transition to Login Successful page after step (8).
Procedural Steps	
1	Profile pages send the password modification request to ProfileManagedBean
2	Parameters are elaborated by the application server
3	ProfileManagedBean change its status and notify
4	UserPages send the login parameters to ProfileManagedBean
5	Parameters are elaborated by the application server
6	ProfileManagedBean change its status and notify
7	Parameters are elaborated by the application server
8	ProfileManagedBean change its status and notify

4.10 INTEGRATION TEST I10

Test Case Identifier	I10-T1	
Test Case Name	Create a new request	
Test Case Description	The purpose of this test is to ensure that RequestManagedBean can handle the case when a new request Is created	
Item(s) to be tested		
1	RequestPages	
2	RequestManagedBean	
Specifications		
Input	**Expected Output/Result**	
RequestManagedBean receive the correct form of the new request	RequestManagedBean notifies the transition to Successful Request Creation page after step (3)	
Procedural Steps		
1	RequestPages send the correct form of the new request	
2	Parameters are elaborated by the application server	
3	RequestManagedBean change its status and notify	

Test Case Identifier	I10-T2
Test Case Name	Cancel a request
Test Case Description	The purpose of this test is to ensure that RequestManagedBean can handle the case when a request is cancelled
Item(s) to be tested	
1	RequestPages
2	RequestManagedBean
Specifications	
Input	**Expected Output/Result**
RequestManagedBean receive the information about the request to cancel	RequestManagedBean notifies the transition to Successful Request Cancellation page after step (3)
Procedural Steps	
1	RequestPages send the information about the request to cancel
2	Parameters are elaborated by the application server
3	RequestManagedBean change its status and notify

4.11 INTEGRATION TEST I11

Test Case Identifier		I11-T1	
Test Case Name		Create a new reservation	
Test Case Description		The purpose of this test is to ensure that ReservationManagedBean can handle the case when a reservation is created	
Item(s) to be tested			
1	ReservationPages		
2	ReservationManagedBean		
Specifications			
Input		**Expected Output/Result**	
ReservationManagedBean receive the correct form of the new reservation.		ReservationManagedBean notifies the transition to Successful Reservation Creation page after step (3)	
Procedural Steps			
1	ReservationPages send the correct form of the new request		
2	Parameters are elaborated by the application server		
3	ReservationManagedBean changes its status and notify		

Test Case Identifier	I11-T2
Test Case Name	Cancel a reservation
Test Case Description	The purpose of this test is to ensure that ReservationManagedBean can handle the case when a reservation is cancelled
Item(s) to be tested	
1	ReservationPages
2	ReservationManagedBean
Specifications	
Input	**Expected Output/Result**
ReservationManagedBean receive the information about reservation to cancel	ReservationManagedBean notifies the transition to Successful Reservation Cancellation page after step (3)
Procedural Steps	
1	ReservationPages send the correct information about the reservation to cancel
2	Parameters are elaborated by the application server
3	ReservationManagedBean changes its status and notify

Test Case Identifier	I11-T3
Test Case Name	Modify a reservation
Test Case Description	The purpose of this test is to ensure that ReservationManagedBean can handle the case when a reservation is modified
Item(s) to be tested	
1	ReservationPages
2	ReservationManagedBean
Specifications	
Input	**Expected Output/Result**
ReservationManagedBean receive the information about reservation to modify	ReservationManagedBean notifies the transition to Successful Reservation Modification page after step (3)
Procedural Steps	
1	ReservationPages send the correct information about the reservation to modify
2	Parameters are elaborated by the application server
3	ReservationManagedBean changes its status and notify

4.12 INTEGRATION TEST I12

Test Case Identifier	I12-T1	
Test Case Name	Login success	
Test Case Description	The purpose of this test is to ensure that ProfileManagedBean can handle correctly the login	
Item(s) to be tested		
1	UserManagedBean	
2	ProfileManagedBean	
Specifications		
Input		**Expected Output/Result**
1. UserManagedBean has accepted an user login 2. Test I15-T1 is executed correctly		1. ProfileManagedBean notifies the transition to Profile Information page after step (2)
Procedural Steps		
1	UserManagedBean notifies user login information to ProfileManagedBean	
2	ProfileManagedBean changes its status and notify	

4.13 INTEGRATION TEST I13

Test Case Identifier	I13-T1	
Test Case Name	Display Request List	
Test Case Description	The purpose of this test is to ensure that RequestManagedBean can handle correctly a request to visualize the list of requests.	
Item(s) to be tested		
1	UserManagedBean	
2	RequestManagedBean	
Specifications		
Input	**Expected Output/Result**	
UserManagedBean received a correct request to visualize the list of requests from UserPages	RequestManagedBean notifies the transition to Display Request List page after step (2).	
Procedural Steps		
1	UserManagedBean notifies the request to visualize the list of requests	
2	RequestManagedBean changes its status and notify	

4.14 INTEGRATION TEST I14

Test Case Identifier	I14-T1
Test Case Name	Display Reservation List
Test Case Description	The purpose of this test is to ensure that ReservationManagedBean can handle correctly a request to visualize the list of reservations.
Item(s) to be tested	
1	UserManagedBean
2	ReservationManagedBean
Specifications	
Input	**Expected Output/Result**
UserManagedBean received a correct request to visualize the list of reservations from UserPages	ReservationManagedBean notifies the transition to Display Reservation List page after step (2).
Procedural Steps	
1	UserManagedBean notifies the request to visualize the list of reservations
2	ReservationManagedBean chage its status and notify

4.15 INTEGRATION TEST I15

Test Case Identifier	I15-T1	
Test Case Name	Login Success	
Test Case Description	The purpose of this test is to ensure that a user can login to the System.	
Item(s) to be tested		
1	Client	
2	UserPages	
Specifications		
Input/External Dependencies	**Expected Output/Result**	
The user has an account.	The user is directed to the ProfilePages Screen	
Procedural Steps		
1	Type the username and password into the login form screen.	
2	Click the “Sign In” button	

Test Case Identifier	I15-T2	
Test Case Name	Incorrect Login Failure	
Test Case Description	The purpose of this test is to ensure that incorrect username and password pairs do not allow the user to access the main features of the System.	
Item(s) to be tested		
1	Client	
2	UserPages	
Specifications		
Input/External Dependencies	**Expected Output/Result**	
The user knows an incorrect account username and password pair.	The user is directed to the LoginErrorRetry Screen (inside the UserPages component)	
Procedural Steps		
1	Type the incorrect username and password into the login form screen.	
2	Click the “Sign In” button	

Test Case Identifier	I15-T3	
Test Case Name	Add an account	
Test Case Description	The purpose of this test is to ensure that a user accounts can be added to the system.	
Item(s) to be tested		
1	Client	
2	UserPages	
Specifications		
Input/External Dependencies	**Expected Output/Result**	
The user has correct parameters to insert into the registration form.	1. No error message appears after step 2. The same result from the Login success test (I15-T1).	
Procedural Steps		
1	Type the correct information in the registration form (email, username, name, surname, password, telephone…)	
2	Click the “Register” button	
3	Perform the Login Success Test(I15-T1) with the username and password in step (1)	

Test Case Identifier	I15-T4
Test Case Name	Register new taxi driver
Test Case Description	The purpose of this test is to ensure that the administrator can add a new taxi driver in the system.
Item(s) to be tested	
1	Client
2	UserPages
Specifications	
Input/External Dependencies	**Expected Output/Result**
1. The administrator is logged in. 2. Taxi driver information provided	1. No error message appears after step (3) 2. The same result from the Login success test (I15-T1).
Procedural Steps	
1	Type the correct information in the taxi driver registration form(email, username, name, surname, password, telephone, driving license…)
2	Click the “Register taxi” button
3	Perform the Login Success Test(I15-T1) with the username and password in step (1)

Test Case Identifier	I15-T5
Test Case Name	Block an account
Test Case Description	The purpose of this test is to ensure that the administrator can block an account in the system.
Item(s) to be tested	
1	Client
2	UserPages
Specifications	
Input/External Dependencies	**Expected Output/Result**
1. The administrator is logged in. 2. Information of the account to be blocked	1. Error message appears after step (3) 2. The same result from the Unsuccessful Login test (I15-T2).
Procedural Steps	
1	Administrator insert account information to block it
2	Click the “Block Account” button
3	Perform the Unsuccessful Login Test(I15-T1) with the username and password in step (1)

4.16 INTEGRATION TEST I16

Test Case Identifier	I16-T1	
Test Case Name	Remove an account	
Test Case Description	The purpose of this test is to ensure that a user account can be removed from the system.	
Item(s) to be tested		
1	Client	
2	Profile Pages	
Specifications		
Input/External Dependencies	**Expected Output/Result**	
The user knows the username and the password of an existing account.	1. No error message appears after step 2. The same result from the Incorrect Login test (I15-T2).	
Procedural Steps		
1	Login to the user account with username and password.	
2	Click on "Delete account"	
3	Perform the Incorrect Login Failure test (I15-T2) with the username and password in step (1)	

Test Case Identifier	I16-T2
Test Case Name	Change password
Test Case Description	The purpose of this test is to ensure that the password of an account can be modified.
Item(s) to be tested	
1	Client
2	Profile Pages
Specifications	
Input/External Dependencies	**Expected Output/Result**
The user knows the username and the password of an existing account.	1. No error message appears after step 2. The same result from the Incorrect Login test (I15-T2) after step (6). 3. The same result from the Login Success test (I15-T1) after step (7)
-Procedural Steps	
1	Login to the user account with username and password.
2	Click on "Change password"
3	Enter the existing password, a valid new password and a confirmation of the new password
4	Click the "Submit Change" button
5	Click the "Logout" button
6	Perform the Incorrect Login Failure test (I15-T2) with the username and old password
7	Perform the Login Success test (I15-T1) with the username and new password

4.17 INTEGRATION TEST I17

Test Case Identifier	I17-T1
Test Case Name	Create a new reservation
Test Case Description	The purpose of this test is to ensure that a new reservation can be created.
Item(s) to be tested	
1	Client
2	Reservation Pages
Specifications	
Input/ External Dependencies	**Expected Output/Result**
1. The user knows correct parameters to create a reservation. 2. The user is registered in the system	The device displays a Correct Reservation Creation Page.
Procedural Steps	
1	Login with username and password
2	Click “New Reservation” button
3	Insert in the form the correct parameter (day, time, num.passenger, pickup location)
4	Click ”Submit reservation” button

Test Case Identifier	I17-T2
Test Case Name	Cancel a reservation
Test Case Description	The purpose of this test is to ensure that a reservation can be cancelled.
Item(s) to be tested	
1	Client
2	Reservation Pages
Specifications	
Input/ External Dependencies	**Expected Output/Result**
1. The user knows the information of reservation. 2. The user is registered in the system 3. The reservation can be cancelled	The device displays a Correct Reservation Cancellation Page.
Procedural Steps	
1	Login with username and password
2	Select the reservation to cancel
3	Click “Cancel Reservation” button
4	Click ”Confirm” button

Test Case Identifier	I17-T3	
Test Case Name	Modify a reservation	
Test Case Description	The purpose of this test is to ensure that a reservation can be modified.	
Item(s) to be tested		
1	Client	
2	Reservation Pages	
Specifications		
Input/ External Dependencies	**Expected Output/Result**	
1. The user knows the parameter for the new reservation. 2. The user is registered in the system 3. The reservation can be modified	The device displays a Correct Reservation Modification Page.	
Procedural Steps		
1	Login with username and password	
2	Select the reservation to modify	
3	Click “Modify Reservation” button	
4	Insert the new parameters	
5	Click “Confirm Modification” button	

4.18 Integration Test I18

Test Case Identifier	I18-T1	
Test Case Name	Create a new request	
Test Case Description	The purpose of this test is to ensure that a new request can be created.	
Item(s) to be tested		
1	Client	
2	Request Pages	
Specifications		
Input/ External Dependencies	**Expected Output/Result**	
1. The user knows correct parameters to create a request. 2. The user is registered in the system	The device displays a Correct Request Creation Page.	
Procedural Steps		
1	Login with username and password	
2	Click “New Request” button	
3	Insert in the form the correct parameter (day, time, num.passenger, pickup location)	
4	Click ”Submit request” button	

Test Case Identifier	I18-T2
Test Case Name	Cancel a request
Test Case Description	The purpose of this test is to ensure that a request can be cancelled.

Item(s) to be tested	
1	Client
2	Request Pages

Specifications	
Input/ External Dependencies	**Expected Output/Result**
1. The user knows the information of the request. 2. The user is registered in the system 3. The request can be cancelled	The device displays a Correct Request Cancellation Page.

Procedural Steps	
1	Login with username and password
2	Select the request to cancel
3	Click “Cancel Request” button
4	Click ”Confirm” button

Test Case Identifier	I18-T3	
Test Case Name	Accept request	
Test Case Description	The purpose of this test is to ensure that a taxi driver can accept a request using his/her mobile device	
Item(s) to be tested		
1	Client	
2	Request Pages	
Specifications		
Input/ External Dependencies	**Expected Output/Result**	
1. The taxi driver receive a request 2. The user who makes the request is registered in the system	The device displays a Correct Request Acceptation Page.	
Procedural Steps		
1	Login with username and password	
2	Wait for the incoming request	
3	Click “Accept Request” button	

4.19 Integration Test I19

Test Case Identifier	I19-T1	
Test Case Name	New reservation	
Test Case Description	The purpose of this test is to register the new reservation	
Item(s) to be tested		
1	ReservationManagedBean	
2	ReservationBeanInterface	
Specifications		
Input	**Expected Output/Result**	
The reservation is received from the ReservationPages	The reservation is registered in the system	
Procedural Steps		
1	ReservationManagedBean sends the reservation to ReservationBeanInterface	
2	ReservationBeanInterface elaborates the reservation and notify ReservationManagdBean	

Test Case Identifier	I19-T2
Test Case Name	Deletion of a reservation
Test Case Description	The purpose of this test is to delete the reservation
Item(s) to be tested	
1	ReservationManagedBean
2	ReservationBeanInterface
Specifications	
Input	**Expected Output/Result**
The reservation is received from the ReservationPages	The reservation is deleted from the system
Procedural Steps	
1	ReservationManagedBean sends the reservation to delete to ReservationBeanInterface
2	ReservationBeanInterface deletes the reservation and notify ReservationManagdBean

4.20 Integration Test I20

Test Case Identifier	I20-T1	
Test Case Name	Login success	
Test Case Description	The purpose of this test is to ensure that UserBeanInterface can handle the case of successful login	
Item(s) to be tested		
1	UserManagedBean	
2	UserBeanInterface	
Specifications		
Input	**Expected Output/Result**	
UserManagedBean receives the parameters of the login	UserBeanInterface notifies the login success	
Procedural Steps		
1	UserManagedBean sends the parameters to UserBeanInterface	
2	UserBeanInterface elaborates the parameters and notify UserManagedBean	

Test Case Identifier	I20-T2	
Test Case Name	Incorrect Login Failure	
Test Case Description	The purpose of this test is to ensure that UserBeanInterface can handle the case of unsuccessful login	
Item(s) to be tested		
1	UserManagedBean	
2	UserBeanInterface	
Specifications		
Input	**Expected Output/Result**	
UserManagedBean receives the parameters of the login	UserBeanInterface notifies the login failure	
Procedural Steps		
1	UserManagedBean sends the parameters to UserBeanInterface	
2	UserBeanInterface elaborates the parameters and notify the login failure to UserManagedBean	

Test Case Identifier	I20-T3
Test Case Name	Add an account
Test Case Description	The purpose of this test is to ensure that UserBeanInterface can handle the case when an account is added in the system
Item(s) to be tested	
1	UserManagedBean
2	UserBeanInterface
Specifications	
Input	**Expected Output/Result**
1. UserManagedBean receives the registration form 2. UserManagedBean receives the login form, after registration	1. UserBeanInterface notifies the registration success 2. UserBeanInterface notifies the login success
Procedural Steps	
1	UserManagedBean sends the registration to UserBeanInterface
2	UserBeanInterface elaborates the registration and notifies UserManagedBean
3	UserManagedBean sends the login parameters to UserBeanInterface
4	UserBeanInterface elaborates the parameters and notifies UserManagedBean

4.21 Integration Test I21

Test Case Identifier	I21-T1
Test Case Name	New Request
Test Case Description	The purpose of this test is to register the new request
Item(s) to be tested	
1	RequestManagedBean
2	RequestBeanInterface
Specifications	
Input	**Expected Output/Result**
The request is received from the RequestPages	The request is registered in the system
Procedural Steps	
1	RequestManagedBean sends the request to the RequestBeanInterface
2	RequestBeanInterface elaborates the request and notify RequestManagedBean

Test Case Identifier	I21-T2
Test Case Name	Deletion of a request
Test Case Description	The purpose of this test is to delete the request
Item(s) to be tested	
1	RequestManagedBean
2	RequestBeanInterface
Specifications	
Input	**Expected Output/Result**
The request is received from the RequestPages	The request is deleted from the system
Procedural Steps	
1	RequestManagedBean sends the request to delete to the RequestBeanInterface
2	RequestBeanInterface deletes the request and notify RequestManagedBean

Test Case Identifier		I21-T3	
Test Case Name		Request accepted	
Test Case Description		The purpose of this test is to handle correctly a request accepted by a taxi	
Item(s) to be tested			
1	RequestManagedBean		
2	RequestBeanInterface		
Specifications			
Input		**Expected Output/Result**	
The event which a request is accepted is received from the RequestPages		The request is activated and the taxi is assigned to the request	
Procedural Steps			
1	RequestManagedBean sends the taxi code which accept the request to the RequestBeanInterface		
2	RequestBeanInterface assigns the taxi to the request and notify RequestManagedBean		

4.22 Integration test I22

Test Case Identifier	I22-T1
Test Case Name	Create new taxi zone
Test Case Description	The purpose of this test is to create a new taxi zone
Item(s) to be tested	
1	UserManagedBean
2	TaxiBeanInterface
Specifications	
Input	**Expected Output/Result**
1. The admin is logged in 2. Positions to add in the zone	The new taxi zone is created
Procedural Steps	
1	UserManagedBean sends the request to create a new taxi zone to the TaxiBeanInterface
2	Information are processed by the system
3	TaxiBeanInterface notifies the creation of the new area

Test Case Identifier	I22-T2
Test Case Name	Delete a taxi zone
Test Case Description	The purpose of this test is to delete a taxi zone
Item(s) to be tested	
1	UserManagedBean
2	TaxiBeanInterface
Specifications	
Input	**Expected Output/Result**
1. The admin is logged in 2. Information about taxi zone to delete	The new taxi zone is deleted
Procedural Steps	
1	UserManagedBean sends the request to delete a taxi zone to the TaxiBeanInterface
2	Information are processed by the system
3	TaxiBeanInterface notifies the cancellation of the taxi area

Test Case Identifier	I22-T3	
Test Case Name	Change queue configuration	
Test Case Description	The purpose of this test is to change the configuration of a taxi queue in a taxi area	
Item(s) to be tested		
1	UserManagedBean	
2	TaxiBeanInterface	
Specifications		
Input		**Expected Output/Result**
1. The admin is logged in 2. Taxi area to modify 3. New queue configuration of the area		The taxi queue is correctly modified
Procedural Steps		
1	UserManagedBean sends the request to modify a queue in a specific are to the TaxiBeanInterface	
2	Information are processed by the system	
3	TaxiBeanInterface notifies the correct modification of area’s queue	

4.23 Integration Test I23

Test Case Identifier	I23-T1
Test Case Name	Send client position
Test Case Description	The purpose of this test is to verify whether the system receives correctly the position of a client.
Item(s) to be tested	
1	Client
2	Position Manager
Specifications	
Input/Initial dependencies	**Expected Output/Result**
1. Client already logged in 2. Correct position of the client already known 3. Device able to locate itself	1. The position is received and managed correctly 2. Position received in Position Manager is equal to the one obtained in Input (2). 3. No positions differences after step (2)
Procedural Steps	
1	Client notifies its position retrieved by the internal GPS
2	Position manager received the position

4.24 Integration Test I24

Test Case Identifier	I24-T1	
Test Case Name	Retrieve a map with position	
Test Case Description	The purpose of this test is to obtain a map in which is indicated the position provided	
Item(s) to be tested		
1	Position Manager	
2	Google Maps API	
Specifications		
Input/Initial dependencies	**Expected Output/Result**	
1. Correct position of the client provided	1. The position is received and managed correctly 2. PositionManager receive a HTTP link to the map with the position.	
Procedural Steps		
1	Position Manages query the external component providing the position	
2	External API elaborates the request	
3	Position Manager receives a HTTP link to the map.	

5 TOOLS AND TEST EQUIPMENT REQUIRED

This section deals with all tools and test equipment needed to accomplish the integration tests. It also explain why and how this tool are going to be used.

5.1 ARQUILLIAN

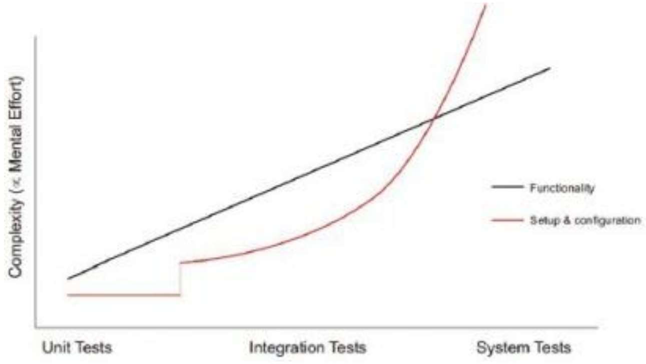
Arquillian is an innovative and highly extensible testing platform for the JVM that enables developers to easily create automated integration, functional and acceptance tests for Java middleware. It is open source testing framework integrated as a JUnit TestRunner.
Arquillian handles all the plumbing of container management, deployment and framework initialization. Arquillian brings the test to the runtime so the developer don't have to manage the runtime from the test (or the build). Arquillian eliminates this problem by covering all aspects of test execution, which involves:

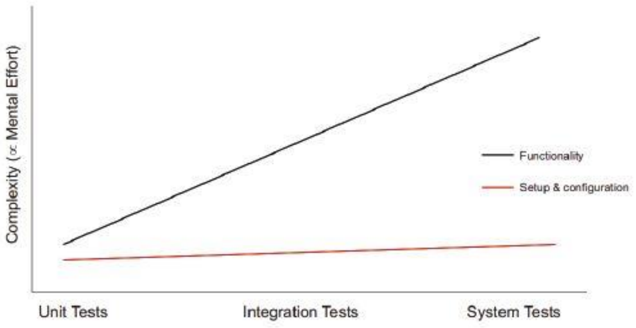
1. Managing the lifecycle of the container (or containers)
2. Bundling the test case, dependent classes and resources into a ShrinkWrap archive (or archives)
3. Deploying the archive (or archives) to the container (or containers)
4. Enriching the test case by providing dependency injection and other declarative services
5. Executing the tests inside (or against) the container
6. Capturing the results and returning them to the test runner for reporting

The developer has a full control over which classes are deployed and injected. Arquillian executes JUnit tests and has full access to the contents of the `src/test/java` folder. Arquillian consists of a runner and container part. The `arquillian-junit` runner executes tests and realizes dependency injection inside a test case.
After the configuration of dependencies, Arquillian can be used as the test runner. The unit tests are executed by Arquillian transparently. Maven, Ant, or even the IDE will just use Arquillian instead of the original test runner to execute the tests. Although this indirection is transparent, it allows Arquillian to inject the deployed Contexts and Dependency Injection (CDI) or Enterprise JavaBeans (EJB) components or other Java EE resources directly to the tests. Arquillian is only a thin layer above the application server implementation. It boots GlassFish, JBoss, or Tomcat behind the scenes and executes the tests on a "native" embedded application server. It uses, for example, Embedded GlassFish behind the scenes.

This tool is going to be used in the majority number of test cases planning for this project. This framework will provides an easily way to test the subcomponents. In fact it does not require to create mock classes or objects, which may takes some times, but simply inject EJB or CDI directly in the test. Moreover it is based on the common and frequently used Junit framework. Arquillian will used to test EJB, CDI and JPA.

The two graphs below make an effort comparison whether Arquillian is used (right picture) or not (left picture). The efficiency in terms of energy and productivity is evident.





5.1.1 Usage scenarios

With the strategy defined above, where the test case is executed in the container, anything the developer can do in an application, he/she can now do in the test class.

A common scenario is testing an EJB session bean. The tester can simply do a JNDI lookup to get the EJB reference and the test becomes a client of the EJB. But having to use JNDI to get a reference to the EJB is inconvenient. Arquillian allows you to use the @EJB annotation to inject the reference to an EJB session bean into the test class.

EJB session beans are one type of Java EE resource the tester may want to access. He/she can access any resource available in a Java EE container. Any of these resources can be injected directly into your test class using the Java EE 5 @Resource annotation.

The test class can access any bean in the ShrinkWrap-defined archive, provided the archive contains a beans.xml file to make it a bean archive. The tester can inject bean instances directly into the class using the @Inject annotation allowing to create a bean instance when needed in the test. Of course, the developer can do anything else he/she can do with CDI within the test as well.

Another important scenario in integration testing is performing data access. If the ShrinkWrap-defined archive contains a persistence.xml descriptor, the persistence unit will be started when the archive is deployed and you can perform persistence operations. The developer can obtain a reference to an EntityManager by injecting it into the class with @PersistenceContext. Alternatively, he/she can execute the persistence operation indirectly through an EJB session bean or a managed bean.

5.2 MANUAL TESTING

Manual testing can be used for integration testing, and in fact should be used to some degree, because automated tests cannot spot all forms of unexpected error conditions. Especially bugs having to do with layout and things related to “look and feel” experience, which are rather common in web apps.
Manual testing is based on the domain and application knowledge of the tester.

This type of testing is used frequently when it is necessary to test the layout of web pages. They are also used the graphical interface of mobile devices. Automated tests cannot be carried out as in this test are evaluated usability and appearance. These manual testing will be used as an evaluation tool for graphical interfaces, so that they could be accessible to everyone and user-friendly.

5.2.1 Usage scenarios

The common scenario, in which this kind of manual testing approach is used, is user interface testing. User interface testing is a testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - tool bar, menu bar, dialog boxes and windows etc.

This list underline the characteristics of GUI testing that the tester should take into account.

1. GUI is a hierarchical, graphical front end to the application, contains graphical objects with a set of properties.
2. During execution, the values of the properties of each objects of a GUI define the GUI state.
3. It has capabilities to exercise GUI events like key press/mouse click.
4. Able to provide inputs to the GUI Objects.
5. To check the GUI representations to see if they are consistent with the expected ones.
6. It strongly depends on the used technology.

The following checklist will ensure detailed GUI Testing.

- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI.
- Check Error Messages are displayed correctly.
- Check for Clear demarcation of different sections on screen.
- Check Font used in application is readable.
- Check the alignment of the text is proper.
- Check the Color of the font and warning messages is aesthetically pleasing.
- Check that the images have good clarity.
- Check that the images are properly aligned.
- Check the positioning of GUI elements for different screen resolution.

Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.

5.3 TEST EQUIPMENT (HARDWARE/SOFTWARE REQUIREMENTS)

This section identifies the resources that are needed for testing. This should include the physical characteristics of the facilities, including the hardware, software, special test tools, and other resources needed (office space, etc.) to support the tests.

5.3.1 Facilities required

The team will need a lab are for the test equipment. The lab area should be approximately 100 m^2 in size. The lab area needs multiple power outlets on each side of the room. A table in the center of the room would also allow for easy-access test team technical discussions.

5.3.2 Hardware required

To enable the team to testing in an optimal environment, the lab needs to contain 2 copies of the system under test: the hardware components in the system [as seen in Document Design, chapter 3], in a single set are a Database Server, a WebServer, a client PC with a web browser that supports Java and the mobile devices. The two systems allow the team to test several components in parallel.

Also, if a single system exhibits a strange bug, it can be left in that state for developer debugging and analysis, while testing continues on the other system.

5.3.3 Software required

Software required in this system is minimal. The database server needs the appropriate database (MySQL) installed, setup, and configured properly.

The WebServer and Application server need GlassFish (ver. 4.1.1) so that they can act like real WebServer and Application server machine.

The client machine needs Java 1.7 installed and properly configured with Internet Explorer 6.0 (or newer) and other browsers (see below).
The mobile phone client needs Android 4.1.1 or iOS 8 installed and properly configured with myTaxiService application installed.

5.3.4 Special requirements

Additional tools and software may need to be purchased or otherwise acquired or reused. Such software is used to execute special tests that are best run and results recorded and analyzed by automated means.

5.4 TEST CONFIGURATIONS AND ENVIRONMENTS

There are 2 clients, running various browsers, and operating system combinations. There are 2 different mobile devices clients, running various operating system. There are 2 sets of database, Web, and app servers (clusters).

System	Name	IP Address	Other Software
Server Cluster 1			
Database1	DB1	192.168.6.10	MySQL
WebServer1	Web1	192.168.6.20	Glassfish
Application Server1	App1	192.168.6.30	Glassfish
Server Cluster 2			
Database2	DB2	192.168.6.11	MySQL
WebServer2	Web2	192.168.6.21	Glassfish
Application Server2	App2	192.168.6.31	Glassfish

System	Name	IP Address	OS	Other Software
		Browser Stations		
Browser Client1	BC1	DNS	MacOS	Google Chrome
Browser Client2	BC2	DNS	Windows	Microsoft Explorer
		Mobile Clients Stations		
Mobile Client 1	MC1	DNS	Android	-
Mobile Client 2	MC2	DNS	iOS	-

6 PROGRAM STUBS AND TEST DATA REQUIRED

This section, based on the test strategy and test design described above, identify any program stubs/driver or special test data required for each integration step.

General assumption on data
The integration test data will be maintained and managed by the Development Team. An integration database will be used for testing. This database is already populated with data that closely resembles production data so there will be no need to initialize this data or load additional data. Database scripts will be used for tables that need to be repopulated due to database structure changes. Test data used and test results will be documented in the integration test results documents and tracked via an error log by the Development Team member designated to perform integration testing.

6.1 INTEGRATION TEST I1

Test Case Identifier	**I1-T1**	
	Name	**Description**
Stub Required	-	-
Driver Required	UserBeanInterface	It performs the call of the method and receives the value returned (if any)
Special data required	-	-

6.2 INTEGRATION TEST I2

Test Case Identifier	**I2-T1**	
	Name	**Description**
Stub Required	-	-
Driver Required	RequestBeanInterface	It performs the call of the method and receives the value returned (if any)
Special data required	-	-

6.3 INTEGRATION TEST I3

Test Case Identifier	I3-T1	
	Name	**Description**
Stub Required	-	-
Driver Required	TaxiBeanInterface	It performs the call of the method and receives the value returned (if any)
Special data required	-	-

6.4 INTEGRATION TEST I4

Test Case Identifier	I4-T1	
	Name	**Description**
Stub Required	-	-
Driver Required	ReservationBeanInterface	It performs the call of the method and receives the value returned (if any)
Special data required	-	-

6.5 INTEGRATION TEST I5

Test Case Identifier	I5-T1	
	Name	**Description**
Stub Required	-	-
Driver Required	TaxiBeanInterface	It performs the request and receives the taxi position
Special data required	Taxi	The taxi for which the position is returned.

6.6 INTEGRATION TEST I6

Test Case Identifier	I6-T1	
	Name	**Description**
Stub Required	PositionManager	It localizes the taxis near the origin of the reservation
Driver Required	ReservationBeanInterface	It performs the request for assigning a taxi to the reservation
Special data required	The reservation	The reservation data

Test Case Identifier	**I6-T2**	
	Name	**Description**
Stub Required	-	-
Driver Required	ReservationBeanInterface	It performs the request for deleting the reservation
Special data required	The reservation	The reservation data to delete

6.7 Integration Test I7

Test Case Identifier	**I7-T1**	
	Name	**Description**
Stub Required	PositionManager	It localizes the taxis near the origin of the request
Driver Required	RequestBeanInterface	It performs the task assigning a taxi to the request
Special data required	The request	The request data

Test Case Identifier	**I7-T2**	
	Name	**Description**
Stub Required	-	-
Driver Required	RequestBeanInterface	It performs the task deleting the request
Special data required	The request	The request data to delete

6.8 Integration Test I8

Test Case Identifier	**I8-T1**	
	Name	**Description**
Stub Required	-	-
Driver Required	UserPages	It performs the login request and receives a Boolean result.
Special data required	Login data	Correct login parameters.

Test Case Identifier	I8-T2	
	Name	**Description**
Stub Required	-	-
Driver Required	UserPages	It performs the login request and receives a Boolean result.
Special data required	Login data	Incorrect login parameters.

Test Case Identifier	I8-T3	
	Name	**Description**
Stub Required	-	-
Driver Required	UserPages	It performs the registration request and receives a Boolean result.
Special data required	Registration data	Correct registration parameters.

Test Case Identifier	I8-T4	
	Name	**Description**
Stub Required	-	-
Driver Required	UserPages	It performs the taxi driver registration request and receives a Boolean result.
Special data required	Taxi driver registration data	Correct registration parameters.

6.9 INTEGRATION TEST I9

Test Case Identifier	I9-T1	
	Name	**Description**
Stub Required	1. UserPages 2. UserManagedBean	They performs the login to check the cancellation.
Driver Required	ProfilePages	It performs the cancellation of the account and receives a Boolean result.
Special data required	-	-

Test Case Identifier	I9-T2	
	Name	**Description**
Stub Required	1. UserPages 2. UserManagedBean	They performs the login to check the password modification.
Driver Required	ProfilePages	It performs the password modification and receives a Boolean result.
Special data required	Password data	The new and the old password

6.10 INTEGRATION TEST I10

Test Case Identifier	I10-T1	
	Name	**Description**
Stub Required	RequestBean	Elaborate the parameters of the request to check if they are correct.
Driver Required	RequestPages	It performs the request creation and receive a Boolean result.
Special data required	Request parameters	The correct parameters to create a request (user, location, time)

Test Case Identifier	I10-T2	
	Name	**Description**
Stub Required	RequestBean	Elaborate the parameters of the request to check if they are correct.
Driver Required	RequestPages	It performs the request cancellation and receive a Boolean result.
Special data required	Request parameters	The request to cancel.

6.11 INTEGRATION TEST I11

Test Case Identifier	I11-T1	
	Name	**Description**
Stub Required	ReservationBean	Elaborate the parameters of the reservation to check if they are correct.
Driver Required	ReservationPages	It performs the reservation creation and receive a Boolean result.
Special data required	Reservation parameters	The correct parameters to create a reservation (user, location, time, date)

Test Case Identifier	I11-T2	
	Name	**Description**
Stub Required	ReservationBean	Elaborate the parameters of the reservation to check if they are correct.
Driver Required	ReservationPages	It performs the reservation cancellation and receive a Boolean result.
Special data required	Reservation parameters	The correct parameters to create a reservation (user, location, time, date)

Test Case Identifier	**I11-T3**	
	Name	**Description**
Stub Required	ReservationBean	Elaborate the parameters of the reservation to check if they are correct.
Driver Required	ReservationPages	It performs the reservation modification and receive a Boolean result.
Special data required	Reservation parameters	The correct parameters to modify a reservation (user, location, time, date)

6.12 INTEGRATION TEST I12

Test Case Identifier	**I12-T1**	
	Name	**Description**
Stub Required	UserPages	Trigger the event of displaying personal profile pages.
Driver Required	UserManagedBean	Change its status after the event of displaying personal profile page and receives a boolean result.
Special data required	-	-

6.13 INTEGRATION TEST I13

Test Case Identifier	**I13-T1**	
	Name	**Description**
Stub Required	UserPages	Trigger the event of displaying personal requests list page.
Driver Required	UserManagedBean	Change its status after the event of displaying personal request list and receives a boolean result.
Special data required	-	-

6.14 INTEGRATION TEST I14

Test Case Identifier	**I14-T1**	
	Name	**Description**
Stub Required	UserPages	Trigger the event of displaying personal reservations list page.
Driver Required	UserManagedBean	Change its status after the event of displaying personal reservations list and receives a boolean result.

Special data required	-	-

6.15 Integration Test I15

Test Case Identifier	**I15-T1**	
	Name	**Description**
Stub Required	1. UserManagedBean 2. UserBean	Check if the login form and parameters are correct.
Driver Required	Client	It performs the filling of login form and its submission
Special data required	Login data	Correct parameters to satisfy the login.

Test Case Identifier	**I15-T2**	
	Name	**Description**
Stub Required	1. UserManagedBean 2. UserBean	Check if the login form and parameters are incorrect.
Driver Required	Client	It performs the filling of login form and its submission
Special data required	Login data	Incorrect parameters to satisfy the login.

Test Case Identifier	**I15-T3**	
	Name	**Description**
Stub Required	1. UserManagedBean 2. UserBean	1.Check if the registration form and parameters are correct. 2.Perform the login to verify the correctness of registration
Driver Required	Client	It performs the filling of registration form and its submission
Special data required	Registration data	Correct parameters to satisfy the registration.

Test Case Identifier	**I15-T4**	
	Name	**Description**
Stub Required	1. UserManagedBean 2. UserBean	1. Check if the taxi driver registration form and parameters are correct. 2.Perform the login to verify the correctness of registration
Driver Required	Client	It performs the filling of registration form and its submission

Special data required	Registration taxi driver data	Correct parameters to satisfy the registration.

Test Case Identifier	**I15-T5**	
	Name	**Description**
Stub Required	1. UserManagedBean 2. UserBean	1. Check if the registration form and parameters are correct. 2.Perform the login to verify the correctness of registration
Driver Required	Client	It performs the filling of registration form and its submission
Special data required	Account information	Correct parameters to block an account.

6.16 Integration Test I16

Test Case Identifier	**I16-T1**	
	Name	**Description**
Stub Required	1. UserManagedBean 2. UserBean	1.Perform the login to verify the correctness of cancellation
Driver Required	Client	It performs the cancellation of the account
Special data required	-	-

Test Case Identifier	**I16-T2**	
	Name	**Description**
Stub Required	1. UserManagedBean 2. UserBean	1.Perform the login to verify the correctness of password modification
Driver Required	Client	It performs the modification of the account's password
Special data required	Password	The old and the new password for the account.

6.17 Integration Test I17

Test Case Identifier	**I17-T1**	
	Name	**Description**
Stub Required	ReservationBean	Check if the reservation form is correct.
Driver Required	Client	It performs the filling of reservation form and its submission
Special data required	Reservation data	Correct parameters to satisfy the reservation (user,date,time,location).

Test Case Identifier	I17-T2	
	Name	**Description**
Stub Required	ReservationBean	Check if the reservation can be deleted correctly.
Driver Required	Client	It performs the cancellation of the reservation.
Special data required	Reservation data	The reservation to cancel.

Test Case Identifier	I17-T3	
	Name	**Description**
Stub Required	ReservationBean	Check if the reservation can be modified correctly.
Driver Required	Client	It performs the modification of the reservation.
Special data required	Reservation data	The reservation to modify.

6.18 Integration Test I18

Test Case Identifier	I18-T1	
	Name	**Description**
Stub Required	RequestBean	Check if the request form is correct.
Driver Required	Client	It performs the filling of request form and its submission
Special data required	Request data	Correct parameters to satisfy the request (user,date,time,location).

Test Case Identifier	I18-T2	
	Name	**Description**
Stub Required	RequestBean	Check if the request can be deleted correctly.
Driver Required	Client	It performs the cancellation of the request.
Special data required	Request data	The request to cancel.

Test Case Identifier	I18-T3	
	Name	**Description**
Stub Required	RequestBean	Check if the request can be accepted.

Driver Required	Client	It performs the acceptation of the request.
Special data required	Request data	The request to accept.

6.19 Integration Test I19

Test Case Identifier	**I19-T1**	
	Name	**Description**
Stub Required	1. TaxiBeanInterface 2. PositionManager	1. It assigns the taxi to the reservation 2. It localizes the taxi near the origin of the reservation
Driver Required	1. ReservationManagedBean 2. ReservationPages	1. It performs the reservation request to ReservationBeanInterface 2. It collects the reservation data from the user
Special data required	The reservation	The reservation data

Test Case Identifier	**I19-T2**	
	Name	**Description**
Stub Required	TaxiBeanInterface	It releases the taxi assigned previously to the reservation
Driver Required	1. ReservationManagedBean 2. ReservationPages	1. It performs the request of deletion to ReservationBeanInterface 2. It collects the reservation data from the user
Special data required	The reservation	The reservation data

6.20 Integration Test I20

Test Case Identifier	**I20-T1**	
	Name	**Description**
Stub Required	-	-
Driver Required	1. UserManagedBean 2. UserPages	1. It performs the request of login to UserBeanInterface and receives the Boolean value (true) 2. It collects the login data from the user
Special data required	The login	The login data

Test Case Identifier	I20-T2	
	Name	**Description**
Stub Required	-	-
Driver Required	1. UserManagedBean 2. UserPages	1. It performs the request of login to UserBeanInterface and receives the Boolean value (false) 2. It collects the login data from the user
Special data required	The login	The login data

Test Case Identifier	I20-T3	
	Name	**Description**
Stub Required	-	-
Driver Required	1. UserManagedBean 2. UserPages	1. It performs the request of registration and login to UserBeanInterface 2. It collects the registration data and the login data from the user
Special data required	The registration The login	The registration data The login data

6.21 Integration Test I21

Test Case Identifier	I21-T1	
	Name	**Description**
Stub Required	1. TaxiBeanInterface 2. PositionManager	1. It assigns the taxi to the request 2. It localizes the taxi near the origin of the request
Driver Required	1. RequestManagedBean 2. RequestPages	1. It performs the request request to RequestBeanInterface 2. It collects the request data from the user
Special data required	The request	The request data

Test Case Identifier	I21-T2	
	Name	**Description**
Stub Required	TaxiBeanInterface	It releases the taxi assigned previously to the request
Driver Required	1. RequestManagedBean 2. RequestPages	1. It performs the request of deletion to RequestBeanInterface

		2. It collects the request data from the user
Special data required	The request	The request data

Test Case Identifier	**I21-T3**	
	Name	**Description**
Stub Required	-	-
Driver Required	1. RequestManaged 2. RequestPages	1. It performs the request of deletion to RequestBeanInterface 2. It collects the request data from the user
Special data required	The request	The request to accept

6.22 Integration Test I22

Test Case Identifier	**I22-T1**	
	Name	**Description**
Stub Required	TaxiBean	Required to elaborate the data and information
Driver Required	UserManagedBean	It performs the request
Special data required	-	-

Test Case Identifier	**I22-T2**	
	Name	**Description**
Stub Required	TaxiBean	Required to elaborate the request
Driver Required	UserManagedBean	It performs the request
Special data required	-	-

Test Case Identifier	**I22-T3**	
	Name	**Description**
Stub Required	-	-
Driver Required	UserManagedBean	It notifies that a request has been accepted
Special data required	1. The request accepted 2. The taxi which accepts the request.	1. The information about the request accepted 2. The information about the taxi which accepted the request.

6.23 Integration Test I23

Test Case Identifier	I23-T1	
	Name	**Description**
Stub Required	-	
Driver Required	Client	It performs the request
Special data required	Client position	The position of the client obtained with a different localization system. This position is compared with the position obtained using the device.

6.24 Integration Test I24

Test Case Identifier	I24-T1	
	Name	**Description**
Stub Required	-	
Driver Required	Position Manager	It queries the external component
Special data required	Client position	The position of the client in coordinates.

7 Bug Tracking & Bug Process

During testing, the team members normally encounter behaviour that goes against a specified or implied design requirement in the product. When this happens, we will document and reproduce the bugs for the developers.

Expectation of a bug:

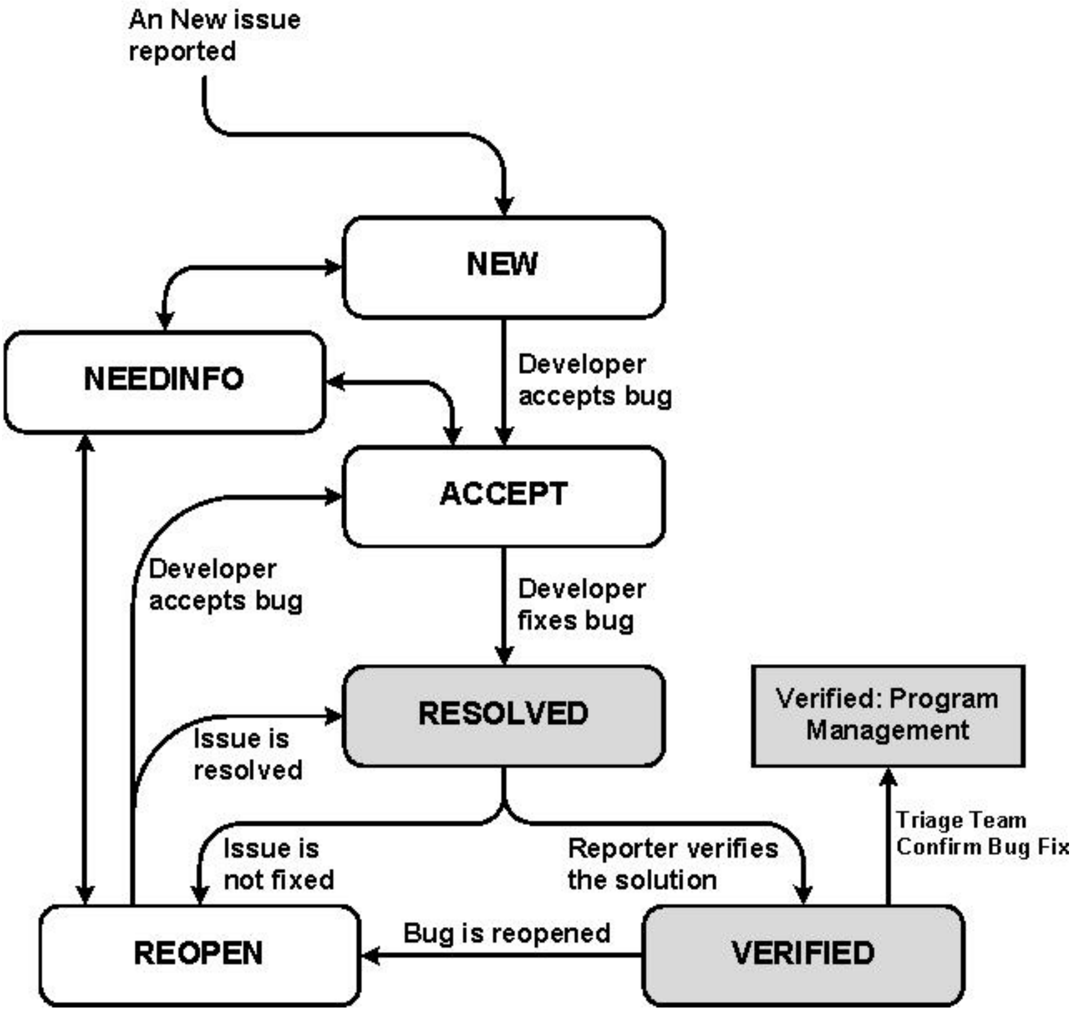
- Keep track of what version of the application the bug is found
- Determine if bug has already been written up
- Indicate the steps to reproduce the bug – write enough details for others looking at the bug to be able to duplicate it; exclude unnecessary steps.
- Actual results – be specific on your findings.
- Expected results – how the product should behave based on the specified or implied requirements.
- Implications – How does the defect affect the quality of the product?

The following chart defines the impact levels to be used when entering bugs.

Impact	Definitions
1 – Fatal	**Test Stopper:** If you can't access a function and need the bug to be fixed immediately.
2 – Serious	**Beta Stopper:** This is a bug that users would experience such as: data corruption, calculation errors, incorrect data and system crash on common user scenarios, significant risk and major UI defects.
3 – Minor	**Live Release:** A bug that must be fixed before the product is officially completed, crashes, content, and UI and graphic changes required for release.

7.1 BUG LIFE CYCLE

All the issues found while testing will be logged into Bugzilla bug tracker.



7.2 Bug Report Form

Problem Report #: __________

Program ______________________________ Release ____________________ Version __________

Report Type (1-6)_____

1 - Coding error
2 - Design issue
3 - Suggestion
4 - Documentation
5 - Hardware
6 - Query

Severity (1-3)_____

1 - Fatal
2 - Serious
3 - Minor

Attachments (Y/N) __________

If yes, describe:

____

____

Problem Summary ____

Can you reproduce the problem? (Y/N) __________

Problem & how can it be reproduced?

____

____

____

Suggested fix (optional input)

____

____

____

Reported By: ____ Date: ____________________

Items below are for use only by the development team

Functional Area:______________________ Assigned To:______________________

Comments:

____

____

Status_____

1 - open 2 - closed

Prioirity (1-5)_____

Resolution (1-9)_____

1 - Pending
2 - Fixed
3 - Irreproducible
4 - Deferred
5 - As designed
6 - Can't be fixed
7 - Withdrawn by reporter
8 - Need more info
9 - Disagree with suggestion

Resolution Version __________

Resolved By: ____ Date: ____________________

Tested By: ____ Date: ____________________

Treat as Deferred (Y/N)_____