POLITECNICO DI MILANO

Computer Science and Engineering

Project of Software Engineering 2: "*myTaxiService*"

Integration Test Plan Document

Author: Andrea Maioli (mat. 852429)
Reference Professor: Mirandola Raffaela

Summary

1. Introduction

1.1. Revision History

- **Revision 1**: Document Creation
- **Revision 2**: SMS Gateway integration and Administrator Entity Bean

1.2. Purpose and Scope

This document represents the Integration Test Plan Document for the myTaxiService application.

1.2.1. Purpose

The purpose of this document is to describe how the integration test of the software will take place, specifying which tools will be used and the approach to follow.

1.2.2. Scope

myTaxiService is a mobile and web application that support the reservation of a taxi and its dispatchment for a large city.
The goal of this application is to simplify the access to the taxi system and manage it in a more efficient way with respect to a fair management of the taxi queues.

1.3. List of Definitions and Abbreviations

- **API**: stands for *Application Programming Interface* and consist in a set of routines, protocols and tools for building software application. They can facilitate the integration of new features into existing applications, even external from the considered one.
- **Credentials**: combination of email and password.
- **Cross Site Scripting**: vulnerability that permits the injection of client-side code into a page.
- **DBMS**: stands for *Data Base Management System* and consist in a system software for creating and managing databases. It provides users and programmers a systematic way to create, retrieve, update and manage data.
- **DD**: Design Document
- **Drop-off point**: location in which the passenger will be drop off by the taxi. It is specified by the passenger when entering the taxi vehicle.
- **GPS**: stands for *Global Positioning System* and is a system that uses satellites to determine the position of a vehicle.
- **HTTP**: stands for *HyperText Transfer Protocol* and is the underlying protocol user by the World Wide Web. It defines how messages are formatted and transmitted, and what actions web servers and browsers should take in response to various commands.
- **Pick-up point**: location in which the passenger will be picked up from the taxi. It is automatically specified by the application or by the passenger (if the request comes from a phone call).
- **RASD**: Requirements Analysis and Specifications Document

1.4. List of Reference Documents

- Assignment 4 - Integration Test Plan document
- Requirement Analysis and Specification Document of myTaxiService
- Design Document of myTaxiService

2. Integration Strategy

2.1. Entry Criteria

All the classes and functions must be well documented using JavaDoc and tested using JUnit-tests, and all the bugs found must be fixed.
Is also important that code inspection is performed on all the code, in order to guarantee that all the conventions are respected and to find possible issues with the code itself.

Before starting the integration testing phase, RASD and DD documents must be updated and delivered.

2.2. Elements to be Integrated

Accordingly to the Design Document is possible to identify four different subsystems, one for each tier, and for each of them is possible to identify the components that contains:

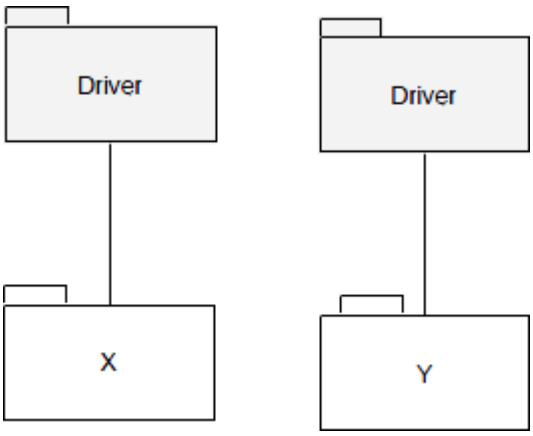
- Client Tier:
  - Mobile application
  - Web Browser
- Web Server Tier:
  - Web Server Controller
  - Mobile API (with the help of JAX-RS)
  - Website Interface (with the help of JSF)
- Application Server Tier:
  - Request Manager
  - Queue Manager
  - Account Manager
  - Location Manager
  - Taxi Manager
  - Entity Beans (Queue, Area, Driver, Passenger, Operator, Request and User)
- Database Server Tier:
  - DBMS

The integration process can be divided into two different phases:

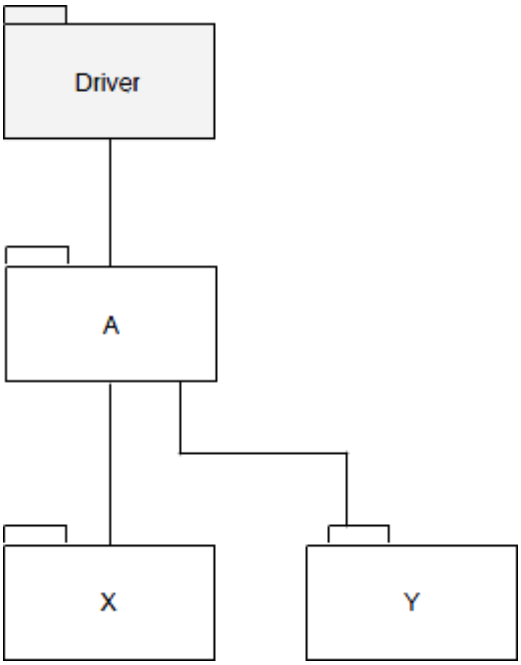
1) Integration between components that compose the same subsystem.
2) Integration of different subsystems.

2.3. Integration Testing Strategy

The integration testing strategy will follow the bottom-up approach.
The first step will start with the lower-level component of the “uses” hierarchy and integrates it with a driver, that is a routine that simulates the behaviour of the upper level modules that are not yet integrated.



At every next step, a new module will be integrated and will replace the driver that simulates it.
Also, a new driver must be constructed, until the top of the “uses” hierarchy is reached.

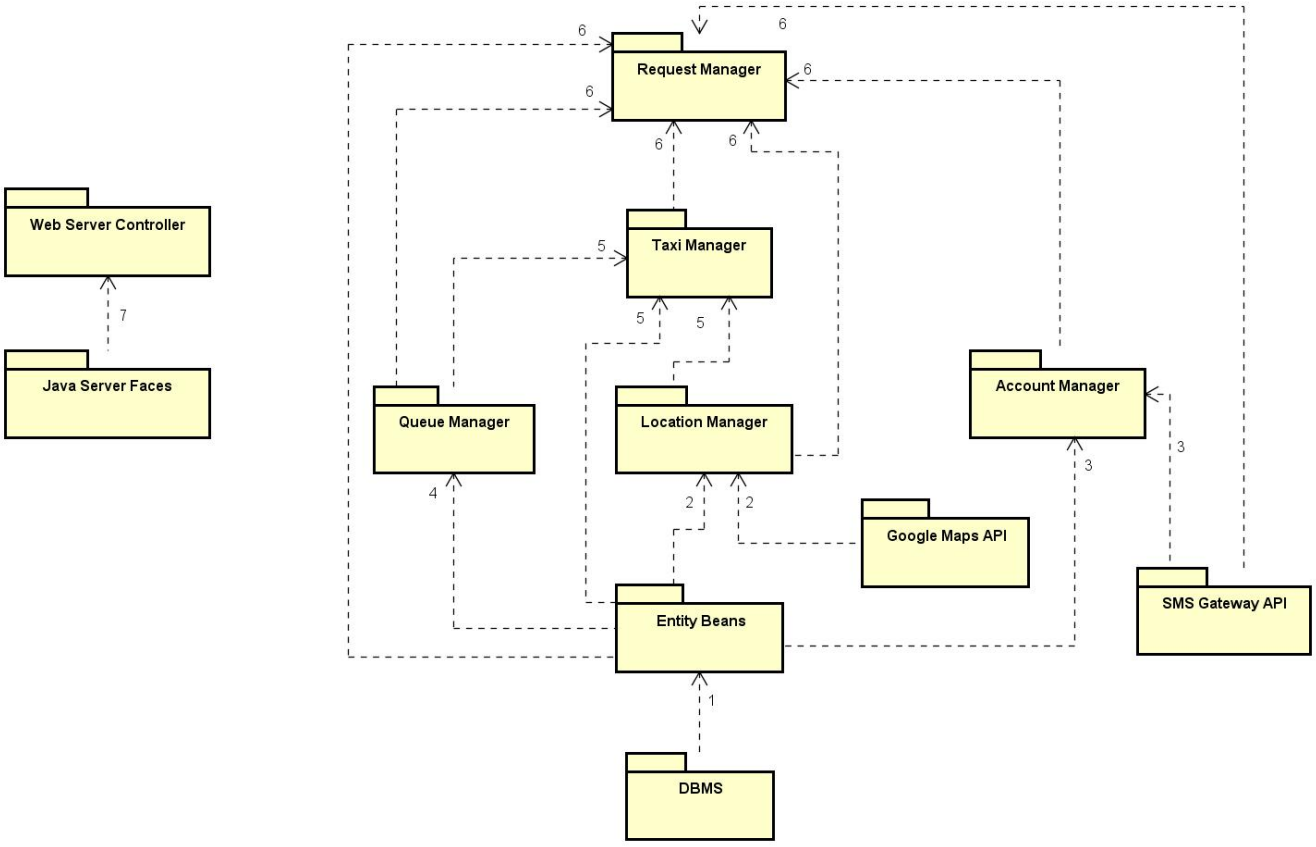


This decision is taken due to the nature of the system described into the Design Document. Each subsystem can be composed by different components that communicates by each other, and those components are already tested using JUnit. In addition, the integration of each subsystem will not be hard due to the communication interface used by each of them (RMI, HTTP, RESTful API).
This decision also limits the number of stubs needed for integration because the bottom-up approach doesn’t need any.

2.4. Sequence of Component / Function Integration

According to the approach described in the previous chapter, the integration will start from the components with the minimum number of dependencies. This prevents the implementation of stubs because when the integration of a component takes place, the components in which it relies on have been already integrated.

2.4.1. Software Integration Sequence



Database Server Subsystem

The Database Server Subsystem doesn't need any integration at software level.

Application Server Subsystem

The integration of the software components for the Application Server Subsystem starts from the Entity Beans (Queue, Area, Driver, Passenger, Administrator, Operator, Request and User) components, which have no dependencies and all the other components relies on them. Is important that before starting the integration testing of the components of this subsystem, all the Entity Beans must be tested with a Testing Database.

Web Server Subsystem

This subsystem contains only the Web Server Controller. This component directly provides the RESTful API, so the only integration to be done is between it and the Java Server Faces component that will be integrated on this subsystem.

Client Subsystem

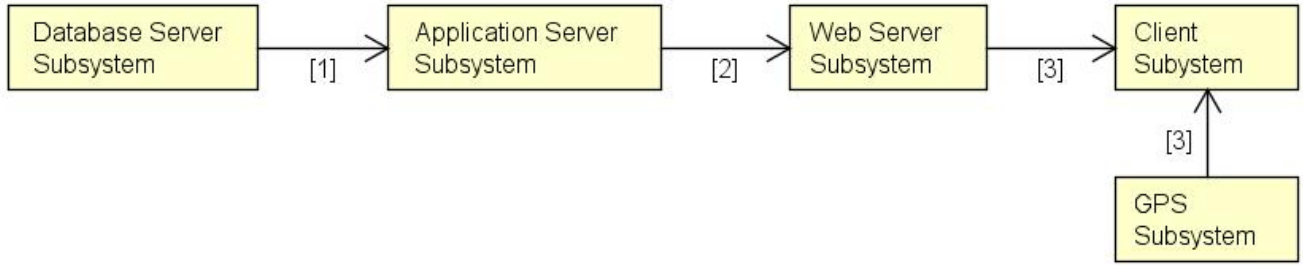
The Client Subsystem doesn't need any integration at software level.

Integration sequence for the software components:

1) Integration of the Testing DBMS with all the Entity Bean (Area, Queue, User, Driver, Passenger, Operator, Request).
2) Integration of Entity Beans (Area, Queue) with Location Manager.
3) Integration of Entity Beans (User) and SMS Gateway API with Account Manager.
4) Integration of Entity Beans (Area, Queue, Driver) and Queue Manager.
5) Integration of Entity Beans (Driver, Queue, Area), Location Manager and Queue Manager with Taxi Manager.
6) Integration of Entity Beans (Area, Queue, Passenger, Operator, Driver, Request), SMS Gateway API, Location Manager, Account Manager, Queue Manager and Taxi Manager with Request Manager.
7) Integration of the Java Server Faces with the Web Server Controller.

2.4.2. Subsystem Integration Sequence

The integration of the different subsystems will follow the bottom-up approach and the subsystem with the less dependencies is the Database Server Subsystem, so the integration process will start with it.



Integration sequence for the subsystems:

1) Integration of the Database Server Subsystem with the Application Server Subsystem.
2) Integration of the Application Server Subsystem with the Web Server Subsystem.
3) Integration of the Web Server Subsystem and the GPS Subsystem with the Client Subsystem.

3. Individual Steps and Test Description

3.1. Software

Test Case Identifier	I1T1
Test Item(s)	DBMS → Entity Bean "Area"
Input Specification	Typical query on the table "Area".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "Area" execute the expected query on the DBMS.

Test Case Identifier	I1T2
Test Item(s)	DBMS → Entity Bean "Queue"
Input Specification	Typical query on the table "Queue".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "Queue" execute the expected query on the DBMS.

Test Case Identifier	I13
Test Item(s)	DBMS → Entity Bean "User"
Input Specification	Typical query on the table "User".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "User" execute the expected query on the DBMS.

Test Case Identifier	I1T4
Test Item(s)	DBMS → Entity Bean "Passenger"
Input Specification	Typical query on the table "Passenger".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "Passenger" execute the expected query on the DBMS.

Test Case Identifier	I1T5
Test Item(s)	DBMS → Entity Bean "Driver"
Input Specification	Typical query on the table "Driver".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "Driver" execute the expected query on the DBMS.

Test Case Identifier	I1T6
Test Item(s)	DBMS → Entity Bean "Operator"
Input Specification	Typical query on the table "Operator".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "Operator" execute the expected query on the DBMS.

Test Case Identifier	I1T7
Test Item(s)	DBMS → Entity Bean "Administrator"
Input Specification	Typical query on the table "Administrator".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "Administrator" execute the expected query on the DBMS.

Test Case Identifier	I1T8
Test Item(s)	DBMS → Entity Bean "Request"
Input Specification	Typical query on the table "Request".
Output Specification	All the requested operations are made on the table and all the expected data is returned from the query.
Environment Needs	Testing Database, Glassfish Server, Driver for the Entity Bean
Purpose	This test check if the called methods of the Entity Bean "Driver" execute the expected query on the DBMS.

Test Case Identifier	I2T1
Test Item(s)	Google Maps API → Location Manager
Input Specification	Request from Location Manager to the Google Maps API.
Output Specification	The request returns the expected information.
Environment Needs	Google Maps API, Glassfish Server, Driver for the Location Manager
Purpose	This test checks if the correct information about a given location (provided in form of latitude and longitude) is retrieved from the Google Maps API.

Test Case Identifier	I2T2
Test Item(s)	Entity Bean "Area" → Location Manager
Input Specification	Methods call from Location Manager to the Entity Bean "Area".
Output Specification	Check if the Location Manager calls the correct methods of the Entity Bean "Area".
Environment Needs	Testing Database, Glassfish Server, Driver for the Location Manager
Purpose	This test checks if the correct information about a given Area are retrieved from the Entity Bean "Area".

Test Case Identifier	I2T3
Test Item(s)	Entity Bean "Queue" → Location Manager
Input Specification	Methods call from Location Manager to the Entity Bean "Queue".
Output Specification	Check if the Location Manager calls the correct methods of the Entity Bean "Queue".
Environment Needs	Testing Database, Glassfish Server, Driver for the Location Manager
Purpose	This test checks if the correct information about a given Queue are retrieved from the Entity Bean "Queue".

Test Case Identifier	I2T4
Test Item(s)	Entity Bean "Administrator" → Location Manager
Input Specification	Methods call from Location Manager to the Entity Bean "Administrator".
Output Specification	Check if the Location Manager calls the correct methods of the Entity Bean "Administrator".
Environment Needs	Testing Database, Glassfish Server, Driver for the Location Manager
Purpose	This test checks if the correct information about a given Administrator are retrieved from the Entity Bean "Administrator". This Entity Bean is used in order to verify if an user is authorized to call a certain method (e.g.: editArea).

Test Case Identifier	I3T1
Test Item(s)	Entity Bean “User” → Account Manager
Input Specification	Methods call from the Account Manager to the Entity Bean “User”.
Output Specification	Check if the Account Manager calls the correct methods of the Entity Bean “User”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Account Manager
Purpose	This test checks that the User information can be correctly created, modified or selected from the Database with the help of the Entity Bean “User”.

Test Case Identifier	I3T2
Test Item(s)	Entity Bean “Administrator” → Account Manager
Input Specification	Methods call from the Account Manager to the Entity Bean “Administrator”.
Output Specification	Check if the Account Manager calls the correct methods of the Entity Bean “Administrator”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Account Manager
Purpose	This test checks if the correct information about a given Administrator are retrieved from the Entity Bean “Administrator”. This Entity Bean is used in order to verify if an user is authorized to call a certain method (e.g.: editUser).

Test Case Identifier	I3T3
Test Item(s)	SMS Gateway API → Account Manager
Input Specification	Methods call from the Account Manager to the SMS Gateway API.
Output Specification	An SMS is sent to the given user.
Environment Needs	Testing Database, Glassfish Server, Driver for the Account Manager
Purpose	This test checks that a SMS containing the login code can be sent to an user that is an administrator.

Test Case Identifier	I4T1
Test Item(s)	Entity Bean “Area” → Queue Manager
Input Specification	Methods call from the Queue Manager to the Entity Bean “Area”.
Output Specification	Check if the Queue Manager calls the correct methods of the Entity Bean “Area”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Queue Manager
Purpose	This test checks that the Area information can be correctly selected from the Database with the help of the Entity Bean “Area”.

Test Case Identifier	I4T2
Test Item(s)	Entity Bean “Queue” → Queue Manager
Input Specification	Methods call from the Queue Manager to the Entity Bean “Queue”.
Output Specification	Check if the Queue Manager calls the correct methods of the Entity Bean “Queue”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Queue Manager
Purpose	This test checks that the Queue information can be correctly selected and modified from the Database with the help of the Entity Bean “Queue”.

Test Case Identifier	I4T3
Test Item(s)	Entity Bean “Driver” → Queue Manager
Input Specification	Methods call from the Queue Manager to the Entity Bean “Driver”.
Output Specification	Check if the Queue Manager calls the correct methods of the Entity Bean “Driver”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Queue Manager
Purpose	This test checks that the Driver information can be correctly selected from the Database with the help of the Entity Bean “Driver”.

Test Case Identifier	I5T1
Test Item(s)	Entity Bean “Area” → Taxi Manager
Input Specification	Methods call from the Taxi Manager to the Entity Bean “Area”.
Output Specification	Check if the Taxi Manager calls the correct methods of the Entity Bean “Area”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Taxi Manager
Purpose	This test checks that the Area information can be correctly selected from the Database with the help of the Entity Bean “Area”.

Test Case Identifier	I5T2
Test Item(s)	Entity Bean “Queue” → Taxi Manager
Input Specification	Methods call from the Taxi Manager to the Entity Bean “Queue”.
Output Specification	Check if the Taxi Manager calls the correct methods of the Entity Bean “Queue”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Taxi Manager
Purpose	This test checks that the Queue information can be correctly selected from the Database with the help of the Entity Bean “Queue”.

Test Case Identifier	I5T3
Test Item(s)	Entity Bean “Driver” → Taxi Manager
Input Specification	Methods call from the Taxi Manager to the Entity Bean “Driver”.
Output Specification	Check if the Taxi Manager calls the correct methods of the Entity Bean “Driver”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Taxi Manager
Purpose	This test checks that the Driver information can be correctly selected and modified from the Database with the help of the Entity Bean “Driver”.

Test Case Identifier	I5T4
Test Item(s)	Location Manager → Taxi Manager
Input Specification	Methods call from the Taxi Manager to Location Manager.
Output Specification	Check if the Taxi Manager calls the correct methods of Location Manager.
Environment Needs	Testing Database, Glassfish Server, Driver for the Taxi Manager
Purpose	This test checks that the Taxi Manager is able to properly manage the location information associated to a Taxi, update the Taxi associated area and compute the estimated time to get to a request pick-up point.

Test Case Identifier	I5T5
Test Item(s)	Queue Manager → Taxi Manager
Input Specification	Methods call from the Taxi Manager to Queue Manager.
Output Specification	Check if the Taxi Manager calls the correct methods of Queue Manager.
Environment Needs	Testing Database, Glassfish Server, Driver for the Taxi Manager
Purpose	This test checks that the Taxi Manager is able to add or remove the considered taxi from a Queue, in front of an update of the taxi status.

Test Case Identifier	I6T1
Test Item(s)	Entity Bean "Area" → Request Manager
Input Specification	Methods call from the Request Manager to the Entity Bean "Area".
Output Specification	Check if the Request Manager calls the correct methods of the Entity Bean "Area".
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Area information can be correctly selected from the Database with the help of the Entity Bean "Area".

Test Case Identifier	I6T2
Test Item(s)	Entity Bean "Queue" → Request Manager
Input Specification	Methods call from the Request Manager to the Entity Bean "Queue".
Output Specification	Check if the Request Manager calls the correct methods of the Entity Bean "Queue".
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Queue information can be correctly selected from the Database with the help of the Entity Bean "Queue".

Test Case Identifier	I6T3
Test Item(s)	Entity Bean "Passenger" → Request Manager
Input Specification	Methods call from the Request Manager to the Entity Bean "Passenger".
Output Specification	Check if the Request Manager calls the correct methods of the Entity Bean "Passenger".
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Passenger information can be correctly selected from the Database with the help of the Entity Bean "Passenger".

Test Case Identifier	I6T4
Test Item(s)	Entity Bean "Operator" → Request Manager
Input Specification	Methods call from the Request Manager to the Entity Bean "Operator".
Output Specification	Check if the Request Manager calls the correct methods of the Entity Bean "Operator".
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Operator information can be correctly selected from the Database with the help of the Entity Bean "Operator".

Test Case Identifier	I6T5
Test Item(s)	Entity Bean “Driver” → Request Manager
Input Specification	Methods call from the Request Manager to the Entity Bean “Driver”.
Output Specification	Check if the Request Manager calls the correct methods of the Entity Bean “Driver”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Driver information can be correctly selected from the Database with the help of the Entity Bean “Driver”.

Test Case Identifier	I6T6
Test Item(s)	Entity Bean “Request” → Request Manager
Input Specification	Methods call from the Request Manager to the Entity Bean “Request”.
Output Specification	Check if the Request Manager calls the correct methods of the Entity Bean “Request”.
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Request information can be correctly created, modified and selected from the Database with the help of the Entity Bean “Request”.

Test Case Identifier	I6T7
Test Item(s)	SMS Gateway API → Request Manager
Input Specification	Methods call from the Request Manager to the SMS Gateway API.
Output Specification	An SMS is sent to the given user.
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that a SMS can be sent to an user that is reported as not found from a driver.

Test Case Identifier	I6T8
Test Item(s)	Location Manager → Request Manager
Input Specification	Methods call from the Request Manager to Location Manager.
Output Specification	Check if the Request Manager calls the correct methods of Location Manager.
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Request Manager is able to properly manage the coordinates associated to a request (pick-up and drop-off points), get its Area and correctly compute its ETA.

Test Case Identifier	I6T9
Test Item(s)	Account Manager → Request Manager
Input Specification	Methods call from the Request Manager to Account Manager.
Output Specification	Check if the Request Manager calls the correct methods of Account Manager.
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Request Manager is able to properly manage the user’s information associated to a request and contact him/her.

Test Case Identifier	I6T10
Test Item(s)	Queue Manager → Request Manager
Input Specification	Methods call from the Request Manager to Queue Manager.
Output Specification	Check if the Request Manager calls the correct methods of Queue Manager.
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Request Manager is able to interact with the Queue of the associated request's area. (For instance: get the first driver in a queue, move a driver to the bottom of the queue, remove a driver from a queue)

Test Case Identifier	I6T11
Test Item(s)	Taxi Manager → Request Manager
Input Specification	Methods call from the Request Manager to Taxi Manager.
Output Specification	Check if the Request Manager calls the correct methods of Taxi Manager.
Environment Needs	Testing Database, Glassfish Server, Driver for the Request Manager
Purpose	This test checks that the Request Manager is able to change the status of the taxi associated to the request.

Test Case Identifier	I7T1
Test Item(s)	Web Server Controller → Java Server Faces
Input Specification	Web Server Controller sends the information to be displayed to JSF.
Output Specification	Check if JSF displays the given output in the correct way.
Environment Needs	Testing Database, Glassfish Server, Stub of the Application Server Subsystem
Purpose	This test checks if JSF can properly communicate with the Web Server Controller without errors.

3.2. Subsystems

Test Case Identifier	SI1T1
Test Item(s)	Database Server Subsystem → Application Server Subsystem
Input Specification	Application Server call the methods of the Entity Beans.
Output Specification	The Database Server execute the expected queries on the DBMS.
Environment Needs	Testing Database, Glassfish Server, Java Persistence API, Entity Beans must be implemented, Driver for Entity Beans
Purpose	This test checks if the Application Server and Database Server can properly communicate without errors, and verify if the expected results of the queries are returned.

Test Case Identifier	SI2T1
Test Item(s)	Application Server Subsystem → Web Server Subsystem
Input Specification	The Web Server Subsystem calls the Remote EJB on the Application Server Subsystem.
Output Specification	The action seen on the system is the expected one.
Environment Needs	Glassfish Server, Application Server's software components must be completed, driver for the Web Server Subsystem (can be a Browser)
Purpose	This test checks if the Web Server and the Application Server can properly communicate without errors, and verify if the expected actions are performed on the system.

Test Case Identifier	SI3T1
Test Item(s)	GPS Subsystem → Client Subsystem
Input Specification	The Mobile Client makes calls to the API exposed by the GPS Component of the mobile device.
Output Specification	The GPS Component return the location of the device.
Environment Needs	Driver for the Mobile Application
Purpose	This test checks if the Mobile Application can properly communicate with the GPS component of the device without encountering any error.

Test Case Identifier	SI3T2
Test Item(s)	Web Server Subsystem → Client Subsystem
Input Specification	The client makes requests to the web server.
Output Specification	The returned pages are the expected one and the requested action is performed.
Environment Needs	Glassfish Server, Web Server must be completed, Browser
Purpose	This test checks if the Client can properly access the Web Application without encountering any error, and verify that the HTTP requests are properly managed by the Web Server.

Test Case Identifier	SI3T3
Test Item(s)	Web Server Subsystem → Client Subsystem
Input Specification	The Mobile Application makes requests to the RESTful API exposed by the Web Server.
Output Specification	The response to the request is the expected one.
Environment Needs	Glassfish Server, Web Server must be completed, iOS or Android Emulator, Mobile Application developed
Purpose	This test checks if the Client can properly access the Mobile Application without encountering any error, and verify that the API requests are properly managed by the Web Server.

4. Tools and Test Equipment Required

In order to automate the integration testing phase, these software tools are required:

- **Mockito**: is a testing framework that allows to abstract dependencies, generating mock objects, drivers and stubs.
- **Arquillian**: is a testing framework that allows to execute test cases inside a java container.
- **JMeter**: is a software designed to load test functional behavior and measure performance. In the integration testing phase can be used to automate the integration test between the Web Server and the Client.

5. Program Stubs and Test Data Required

In order to start the integration testing phase without having the entire system developed, these drivers and stubs must be used:

- **Testing Database**: all the testing environment must include a DBMS configured in the same way of the production DBMS, but with a less number of instances. This will prevent the waste of resources and time in the integration testing phase, but will grant to work with the same data.
- **Stubs of the Application Server Subsystem**: this stub will provide a small set of data used for the Web Server Subsystem testing, without having the Application Server ready.
- **Tiny API Client**: this driver will be used for the testing of the RESTful API provided by the Web Server Subsystem, without having a client application ready.
- **Drivers for each component**: this is needed for simulating a call for the component methods and verify the result on the integrated components.

6. Appendix

6.1. Hours of Work

The redaction of the entire document took about 25 hours of work.

6.1. Reference Documentation

- Requirements Analysis and Specification Document
- Design Document
- “Assignment 4: Integration Test Plan”