

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** 3](#_Toc440559741)

[1.1. Revision History 3](#_Toc440559742)

[1.2. Purpose and Scope 3](#_Toc440559743)

[*1.2.1. Purpose 3*](#_Toc440559744)

[*1.2.2. Scope 3*](#_Toc440559745)

[1.3. List of Definitions and Abbreviations 3](#_Toc440559746)

[1.4. List of Reference Documents 3](#_Toc440559747)

[**2.** **Integration Strategy** 4](#_Toc440559748)

[2.1. Entry Criteria 4](#_Toc440559749)

[2.2. Elements to be Integrated 4](#_Toc440559750)

[2.3. Integration Testing Strategy 4](#_Toc440559751)

[2.4. Sequence of Component / Function Integration 4](#_Toc440559752)

[*2.4.1. Software Integration Sequence 4*](#_Toc440559753)

[*2.4.2. Subsystem Integration Sequence 4*](#_Toc440559754)

[**3.** **Individual Steps and Test Description** 4](#_Toc440559755)

[**4.** **Tools and Test Equipment Required** 4](#_Toc440559756)

[**5.** **Program Stubs and Test Data Required** 4](#_Toc440559757)

1. **Introduction**
   1. **Revision History**

* **Revision 1**: Document Creation
  1. **Purpose and Scope**

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

* + 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. **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. **List of Definitions and Abbreviations**
  2. **List of Reference Documents**
* Assignment 4 - Integration Test Plan document
* Requirement Analysis and Specification Document of myTaxiService
* Design Document of myTaxiService

1. **Integration Strategy**
   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.

* 1. **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 have to be integrated:

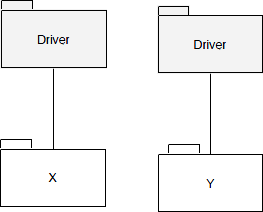
* Client Tier:
  + Mobile application
* Web Server Tier:
  + Mobile API (with the help of JAX-RS)
  + Website Interface (with the help of JSF)
  + Communication Interface with the Application Server Tier
* Application Server Tier:
  + Request Manager
  + Queue Manager
  + Account Manager
  + Location Manager
  + Taxi Manager
  + Database Manager
* 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.
   1. **Integration Testing Strategy**

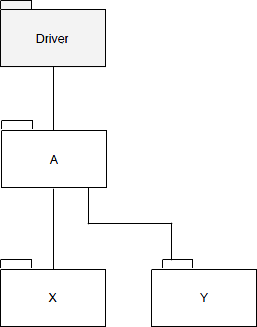
The integration testing strategy will follow the bottom-up approach.

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.

* 1. **Sequence of Component / Function Integration**
     1. **Software Integration Sequence**
     2. **Subsystem Integration Sequence**

1. **Individual Steps and Test Description**
2. **Tools and Test Equipment Required**
3. **Program Stubs and Test Data Required**