Politecnico di Milano
Academic Year 2015/2016
Software Engineering 2: “myTaxiService”

Test Plan

Massimo Schiavo, Marco Edoardo Cittar

21st January 2016

Summary

- **Section 1:** Introduction
- **Section 2:** Integration strategy
- **Section 3:** Integration tests
- **Section 4:** Stubs and data test required

Introduction

- Purpose and scope

  This document describes how to test the integration of the components of the system. Its purpose is to test if the components are correctly integrated between each other, as described in the Design Document.

  The scope of the document is to check that the application functionalities are correctly implemented, starting from the login process, to the ride requests to be made by the users and the correct handling of all this through the application logic.

Entry criteria

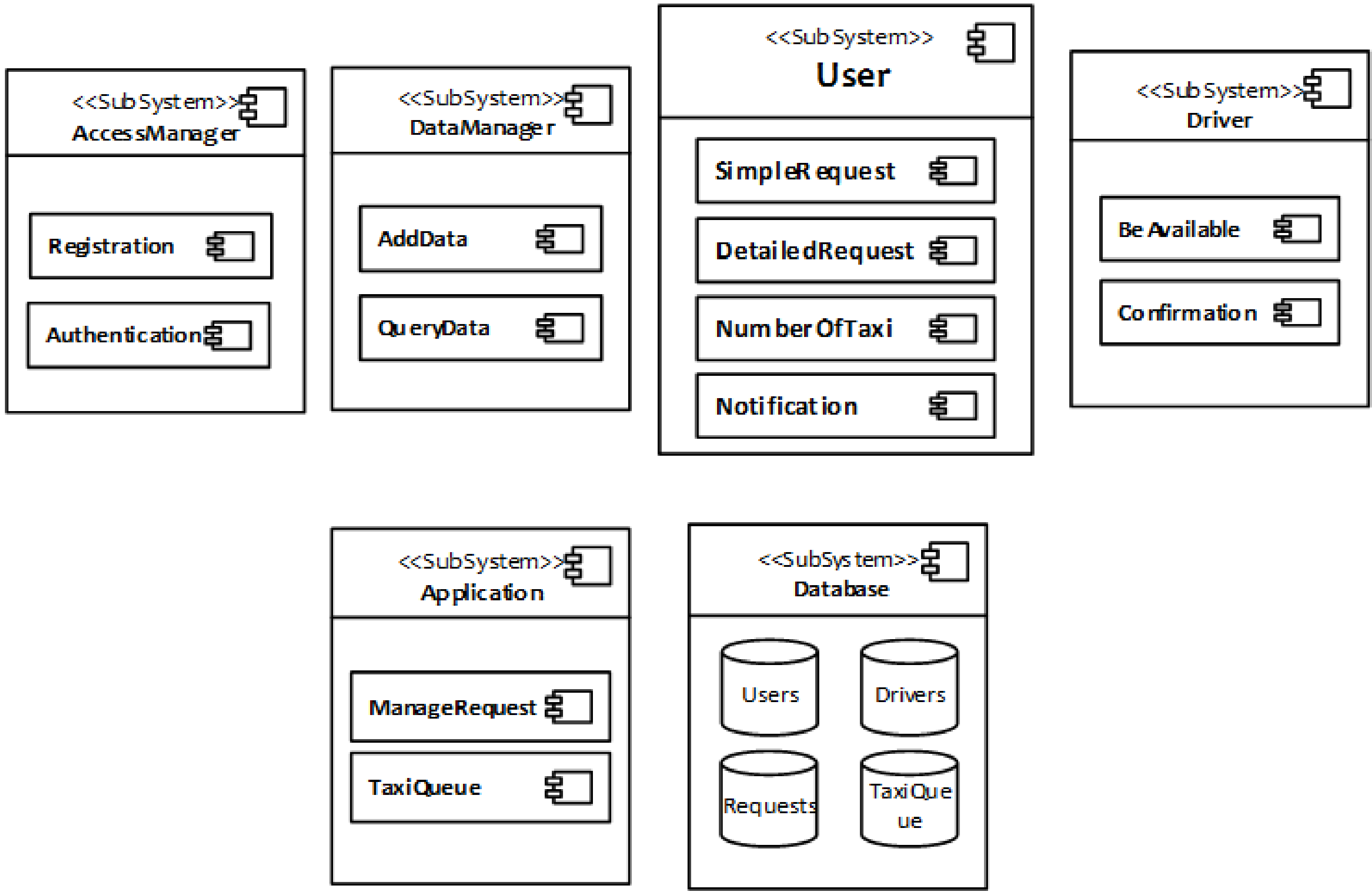
Before starting the integration testing of the subsystems, we recommend to perform a unit test on the Application sub-system in order to check the algorithm correctness, the functional specifications and the structural coverage. This is the major class that contains the logic of myTaxiService. In this class will be implemented, according to the document design already provided, the algorithm of management of the queues of taxis and the algorithm responsible for the forwarding of the requests arrived from the user to the taxi drivers and that's why it's important to perform a unit test on this class before proceeding to the integration test.

The documentation about what is needed to perform this kind of test is in the fourth section of this presentation.

Section 2: Integration strategy

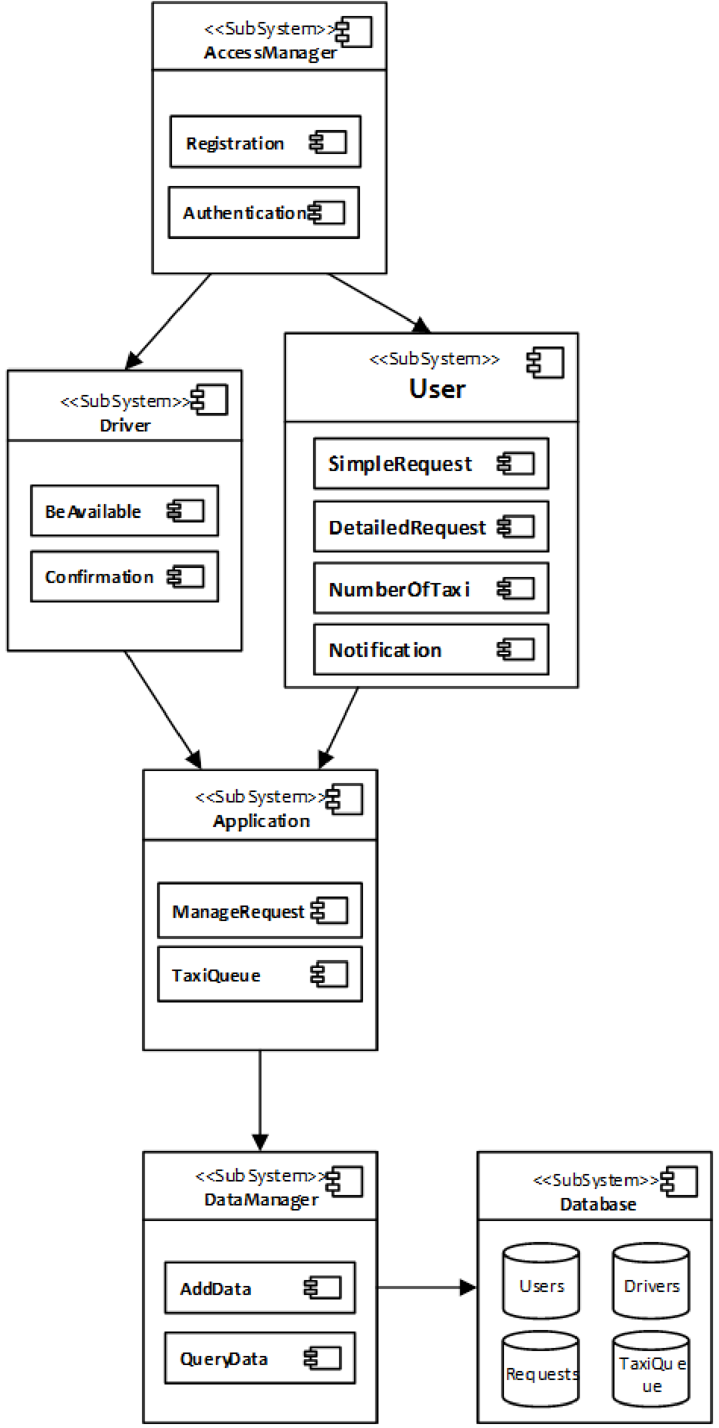
Elements to be integrated

The items to be tested consists of the integration of all the subsystem developed that form myTaxiService.



Integration testing strategy

For testing, we choose the top-down approach. In this approach testing is conducted from the high level interfaces to the core of our application. It's advantageous because the major flaws occur toward the top of the program. It's also a way to simplify the readability of the results of the test case because once the I/O functions are added, representation of test cases is easier. Due to these considerations, in the picture is shown through the arrows the order of integrations between the subsystems.



Integration test I1

Integration Test Identifier	I1T1
Test item(s)	This test procedure verifies the correct integration between AccessManager and the User and Driver components.
Purpose	The test must check if the AccessManager can correctly handle users' input.
Preconditions	A Database and a DataManager stub must be created to simulate the login process.
Procedure Steps	Some input tries will be performed, including the login of a User and a Driver. If the components are correctly integrated, the AccessManager should be able to differentiate between the two types of login and show the correct personal page.

Integration test I2

Integration Test Identifier	I2T1
Test item(s)	This test procedure verifies the correct integration between the User and the Application components.
Purpose	The test must check if the User component can correctly dispatch users' requests.
Preconditions	I1T1 must have been already correctly executed.
Procedure Steps	The two main functionalities of this component are tested, that are SimpleRequest and DetailedRequest. If the components are correctly integrated the Application should correctly receive the requests from the User and will then proceed to manage them.

Integration test I2

Integration Test Identifier	I2T2
Test item(s)	This test procedure verifies the correct integration between the Driver and the Application components.
Purpose	The test must check if the Driver component can correctly receive users' requests and reply to them.
Preconditions	I2T1 must have been already correctly executed. The stubs described in the 4.3 section must have been created.
Procedure Steps	At first it must check that the Driver, by setting himself as available, is correctly inserted in a queue. Then a simulated request from the stub is sent to the Driver which then should act properly, either if he answers the request or not.

Integration test I3

Integration Test Identifier	I3T1
Test item(s)	This test procedure verifies the correct integration between the Application and the DataManager.
Purpose	The test must check if the Application component can correctly make query and modification requests of the Database through the DataManager.
Preconditions	A Database stub is needed.
Procedure Steps	The Application should try dispatching the query and add calls to the DataManager. Some examples of these calls can be the query for the first available driver in a queue and the modification of the order of a taxi queue after the Driver's answer to a request.

Integration test I4

Integration Test Identifier	I4T1
Test item(s)	This test procedure verifies the correct integration between the DataManager and the Database components.
Purpose	The test must check if the DataManager can correctly query and modify the Database.
Preconditions	I3T1 must have been already correctly executed.
Procedure Steps	The DataManager should be able to receive query and modification requests from the logic (Application) of the system, correctly execute them and eventually return the desired data.

Stubs and data test required

- At the first step of our integration, a stub that simulates the database response correlated to a logging or registering action is required.
- A stub that simulate operations on a database is also needed.
- At the second step of the integration test different stubs are required:
  - A stub for drivers which by clicking on be_available gains the right to enter in a queue list.
  - A stub that simulate a pending request.
  - A stub that simulate the timing out of the response time with the relative consequences.
- By integrating the remaining components there are no stubs to add.