Chapter 1

Individual Steps and Test Description

In this chapter we are going to fully describe each test, pointing out the pre- and the post-conditions, the purposes and the environment needed.

1.1 DBMS → System Controller - Dispatcher

Test Case Identifier	I1
Components Involved	DBMS → System Controller - Dispatcher
Input Specifications	Create all possible requests by the DBMS.
Purposes	The purposes of this test are: • monitors the proper queuing of requests. • simulates a request generation.
Output Specifications	Check that the request are put into the dispatcher's queue only when they are generated.
Environment Needed	System Dispatcher Stub. DataBaseStub, to simulate the interactions between the DBMS and the database.

1.2 System Controller - Dispatcher → DBMS

Test Case Identifier	I2
Components Involved	System Controller - Dispatcher → DBMS
Input Specifications	We should generate typical data, but in this case it is sufficient to generate desired data for each DBMS interface method[1]
Purposes	The purposes of this test are: • find a data. • saves data into the database. • add/remove data from the database.
Output Specifications	We check the correctness of the searched/modified/added/removed data
Environment Needed	System Dispatcher Stub. DataBaseStub, to simulate the interactions between the DBMS and the database.

1.3 System Controller - Dispatcher → User Creator

Test Case Identifier	I3
Components Involved	System Controller - Dispatcher → User Creator
Input Specifications	None.
Purposes	We want to test the creation of each kind of user.
Output Specifications	Checks if a user is correctly created.
Environment Needed	System Dispatcher Stub. Note that the User Creator returns the created user and the dispatcher has the role to save it into the database.

[1] see the section 2.6 of the Design Document.

1.4 System Controller - Dispatcher → User Checker

Test Case Identifier	I4
Components Involved	System Controller - Dispatcher → User Checker
Input Specifications	We want to create each kind of user.
Purposes	We want to test the correct identification of the users. in the following way. First we pass a correct type of user, then a wrong type and the component should detect both cases.
Output Specifications	Check if correct result is given.
Environment Needed	System Dispatcher Stub. User Creator to create the users.

1.5 Client and User Handler → System Controller - Dispatcher

Test Case Identifier	I5
Components Involved	Client and User Handler → System Controller - Dispatcher
Input Specifications	We want to simulate all kinds of user's request.
Purposes	We test the way each request is enqueued in the dispatcher, thus the method called and the parameters specification.
Output Specifications	Check if a request is correctly enqueued.
Environment Needed	System Dispatcher Stub.

1.6 Client and User Handler → User Checker , Security Manager

Test Case Identifier	I6
Components Involved	Client and User Handler → User Checker , Security Manager
Input Specifications	We simulate different user's requests.
Purposes	The purpose of this group of tests is to check the correct authentication procedure, thus if for each request generated by the user, the Client and User Handler tries to identify it. The tests are several: we start from login (both user and driver) to check the shown homepage/services; then we check various request to verify if an additional check is performed before the interaction with the dispatcher.
Output Specifications	Check if the correct methods into the User Checker are called and check the results.
Environment Needed	System Dispatcher Stub to simulate the requests' queuing. User stub to simulate the interactions with the external world (both the stubs have fixed behaviour for each actions, but they are useful to be identified.).

1.7 Client and User Handler → System Controller - Data Checker , Security Manager

Test Case Identifier	I7
Components Involved	Client and User Handler → System Controller - Data Checker , Security Manager
Input Specifications	Possible inputs by users, both valid and invalid.
Purposes	The purposes are: • check the invocation of data checking procedures on each kind of input. • check the request of encryption of a password only. • check the secure connection request during a login.
Output Specifications	Checks the identification of data errors (invalid emails, names, surnames or the presence of SQL injection). Check if the encryption is called after a password insertion. Check the invocation of secure connections.
Environment Needed	User stub to simulate the interactions with the external world.

1.8 System Controller - Dispatcher → Ride Allocator

Test Case Identifier	I8
Components Involved	System Controller - Dispatcher → Ride Allocator
Input Specifications	We create a ride request into the System Dispatcher.
Purposes	We want to test if the correct methods inside the Ride Allocator are called.
Output Specifications	Checks the correct methods invocation inside the Ride Allocator, ignoring the effects of the calls.
Environment Needed	Here, the System Dispatcher has been implemented.

1.9 Ride Allocator → System Controller - Dispatcher, Client and User Handler

Test Case Identifier	I9
Components Involved	Ride Allocator → System Controller - Dispatcher , Client and User Handler
Input Specifications	We want to simulate the rides' booking.
Purposes	We want to simulate all the ride's booking procedure, both for a future ride and for a zerotime ride. In the latter case we test all the assignment procedure.
Output Specifications	Check if the correct driver is called, check the correct methods invocations and check the system answers in each phase.
Environment Needed	User and Driver stubs to simulate the requests.

1.10 System functionalities test.

Finally, in the **??** we do not focus on the services handling, because the System Dispatcher is the last developed component. In each test where this component is not available, it has been simulated by an appropriate stub defined in the **??**.
Now, we want to check the correctness of each service. These tests are not described here because they are related only to one component functionalities, even if it interacts with some other components. Furthermore, these tests are described in the Unit Test Document, not available for *myTaxyService*.
To have an idea, with the stubs described in the **??** each service will be tested independently, knowing that each other involved component works correctly[2].

[2]This is the reason why these tests are unit ones and not integration ones.