Chapter 1

Program Stubs and Test Data Required

This chapter is dedicated to a detailed description of each required program stub to the integration tests.

1.1 Program Stubs

The Integration Test of *myTaxyService* requires four program stubs to perform some interactions with the external environment and/or components, like the database.

1.1.1 User Simulator

The User Simulator is a stub able to act as a normal user. It offers the following methods:

- *register*, to simulate a registration with fixed data;
- *login*, to simulate a login with the same credentials defined in the registration phase;
- *profileManagement*, to simulate the management of the user's personal profile. The change is only one for each execution and it is fixed as for he

user's data.

- *zerotimeReservation*, that accepts as parameter a GPS coordinate and an address. It simulate a booking of a zerotime ride. The starting position of the ride is given as in the case where the user has activated the GPS and, to be precise, that value is the one passed as parameter.
- *futureReservation*, to perform a future ride booking. For each calling of the method the departure is exactly three hours after the current time, whereas the two addresses are the same passed as parameter to the method[1].

Note that the two remaining functionalities (check the reservations and read the alerts) are not provided by this stub.

1.1.2 Driver Simulator

The Driver Simulator is a stub able to act as a driver. It offers two main methods:

- *accept*, to simulate the System request for a ride's assignment. In each method invocation it is performed the action of accepting a ride. An other version of this method, called *randomRide* can either accept or deny a ride, if someone wants to test this specific case.
- *startWaiting*, that accepts a GPS position and with this value it simulates the *Start Waiting Time* functionality.

Note that the working hours handling is not provided.

1.1.3 DataBase Stub

This stub is able to work as a real database for the other components. The methods offered by this stub allow to save and receive data as in a real interaction: in the former case the fake database perform no actions, whereas in the latter case fixed data are returned by each method. We do not point out on each

[1]Both for this method and for the previous one, the addresses are not fixed as other data. The reason for this choice is easy: we can simulate the booking with valid values, depending on the simulated city.

method definition: to have an idea see the DBMS interface into the Design Document and imagine an application of them to the rides' data.

1.1.4 Dispatcher Stub

TO BE WRITTEN