
A.Y. 2015/2016

Software Engineering 2: “My Taxi Service”

Test Plan Document

Version 1.0

Ivan Antozzi(790962), Riccardo Giambona(788904)

21/01/2016

# 1.Introduction

## 1.1Revision History

No revisions were made to this document.

## 1.2 Purpose and Scope

The purpose of this document is to describe the way that the testing will be performed on our system. This document is meant to be read by all the testing team components.

## 1.3 List of Definitions and Abbreviations

No definitions and abbreviations were used.

### 1.4 List of Referenced Documents

- Design Document
- Integration Plan Example.pdf
- Assignment4 – integration test plan.pdf

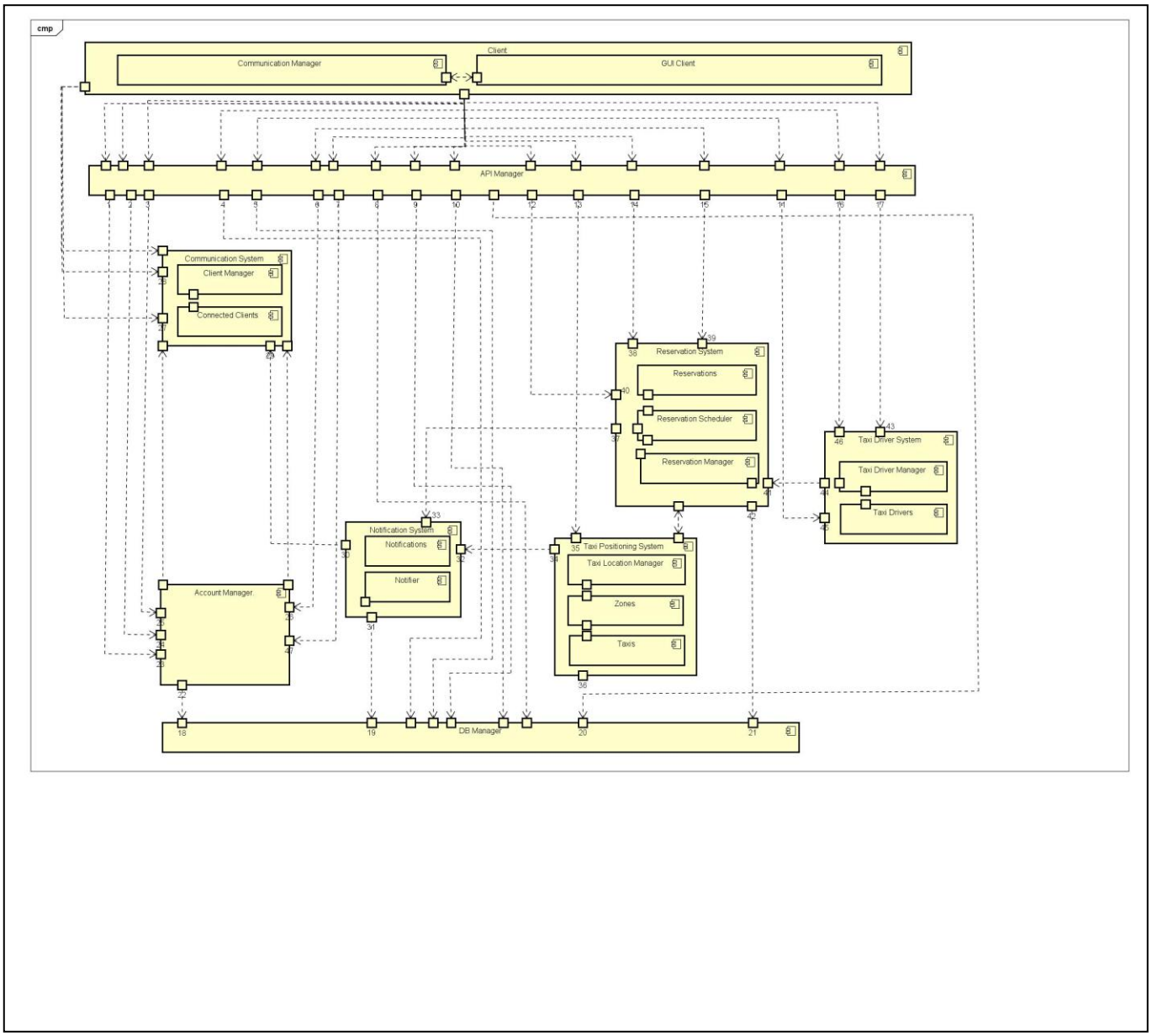
## 2.Integration Strategy

### 2.1 Entry Criteria

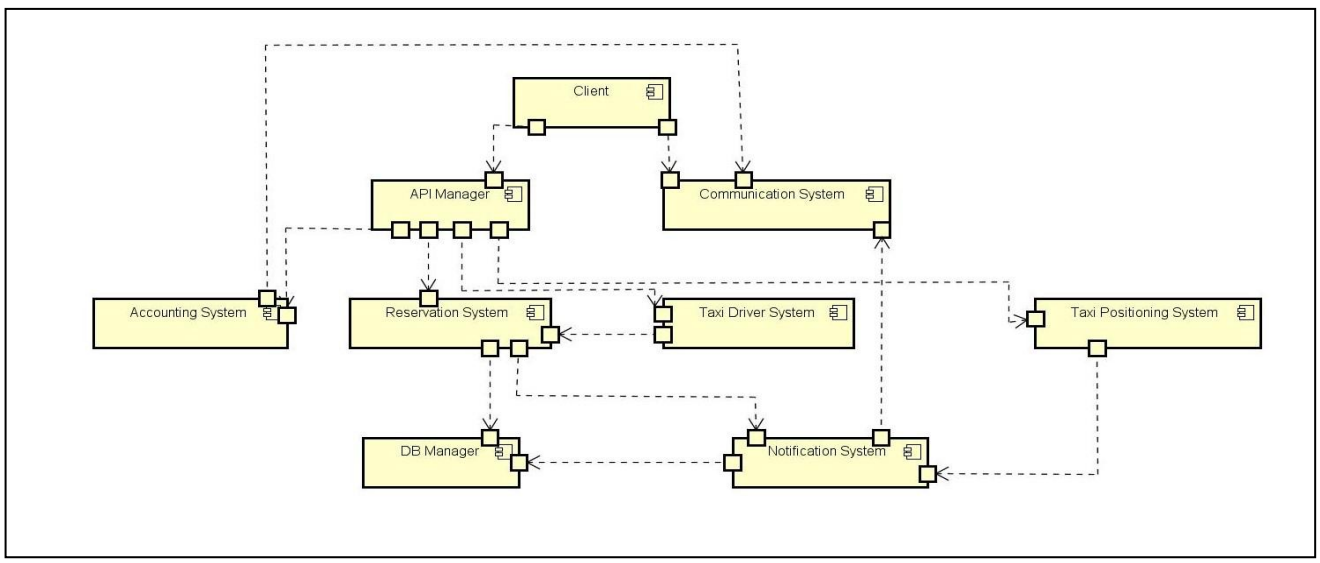
The entry criteria for our integration testing plan is that all the components of each sub system are completely developed, but not unit tested yet.

### 2.2 Elements to be integrated

The components that need to be integrated are the ones that we specified in the design document, in the component view section. Wr reported below the component view in order to clarify to the reader which components are going to be integrated.




## 2.3 Integration Testing Strategy



The graph above is a graph that shows the dependencies between each component and one another. Since we decided to use a bottom up approach, as much as possible, we will start integrating the components that are at the bottom of this graph.

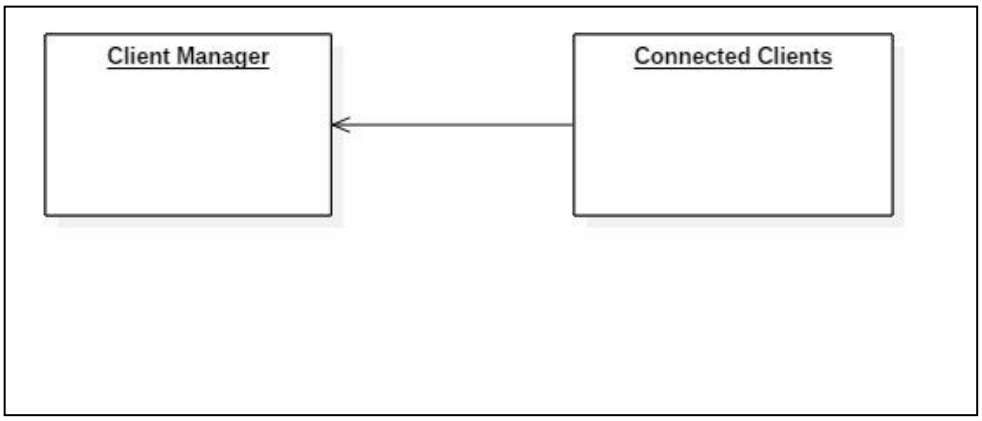
We chose to use the bottom up approach, because in this way, we can test that each single sub system works, before integrating them to one another. We didn't choose the top down approach, because in the design document we have already specified all the needed components and therefore we already have a complete vision of how the system will be. We also evaluated that with the top down approach we would have the need of a complex API manager stub, because it would have had to ansie each single client request. Instead, with the bottom up approach, we still have a API manager stub, but it is simpler to realize because it will only be used to give an interface to the client when testing the client and the communication system, but those methods can be even empty because we are only interested in the testing between the client and the communication system. The API Manager will ony be tested when all the other components have been correctly tested.

## 2.4 Sequence of Components/Function Integration

### 2.4.1 Software Integration Sequence

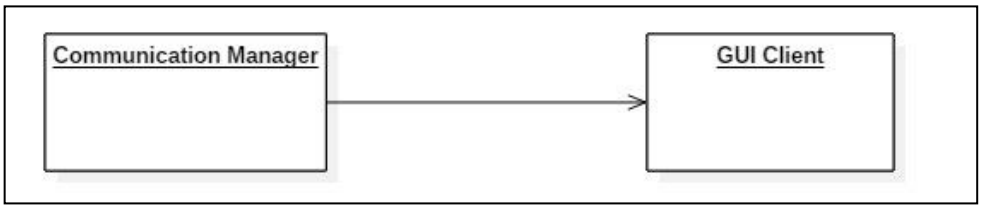
In this schema we assume that the order of the integration of each sub system is specified by the arrows, starting from the DB manager and so on for the others. Since we decide to use a bottom up strategy, it's implicit that we start integrating from the component with less dependences from the other components. According to this strategy, for example " DB Manager → Accounting System" means that the accounting system will call the methods of the DB Manager and not the opposite. Always remember that the "→" means the order of the integration and not that the left component calls the methods of the right component side of the arrow.

#### 2.4.1.1 Communication System



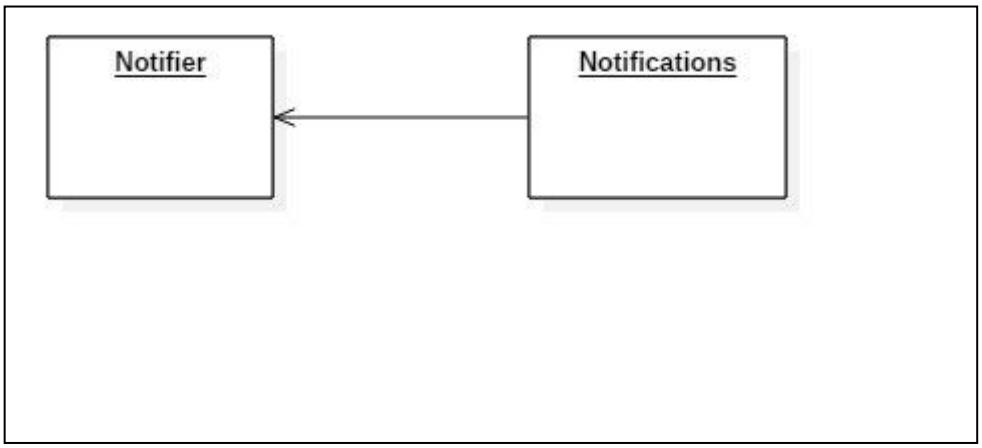
| ID | Integration Test | Paragraphs |
|---|---|---|
| I1 | Connected Clients → Communication Manager | 3.1.1 |

### 2.4.1.2 Client



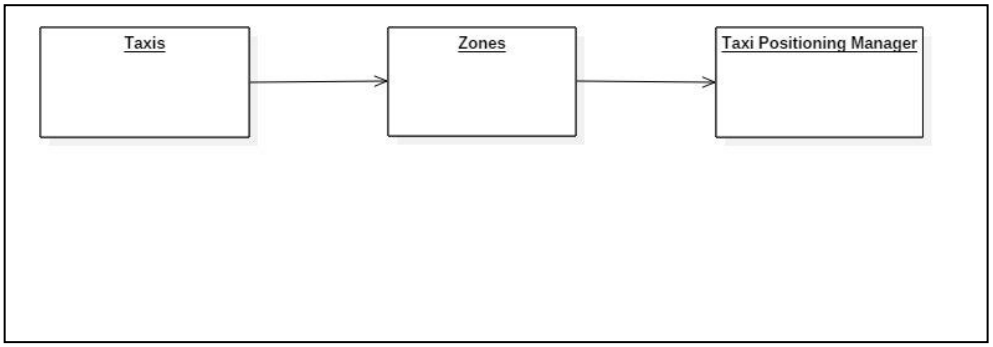

| ID | Integration Test | Paragraphs |
|---|---|---|
| I2 | Communication Manager → GUI Client | 3.1.2 |

### 2.4.1.3 Notification System



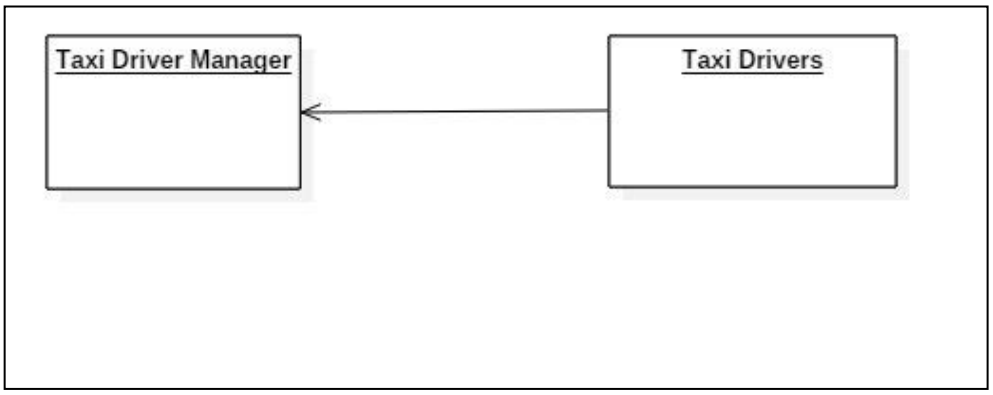

| ID | Integration Test | Paragraphs |
|---|---|---|
| I3 | Notifications → Notifier | 3.1.3 |

### 2.4.1.4 Taxi Positioning System



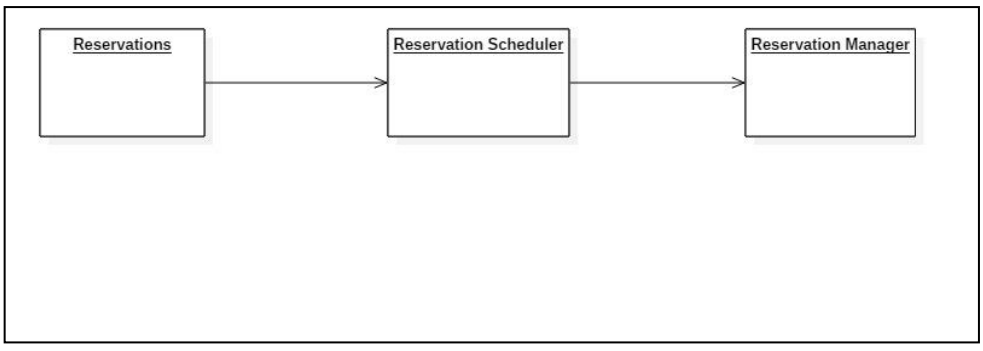

| ID | Integration Test | Paragraphs |
|---|---|---|
| I4 | Taxis → Zones | 3.1.4 |
| I5 | Zones → Taxi Positioning Manager | 3.1.5 |

### 2.4.1.5 Taxi Driver System



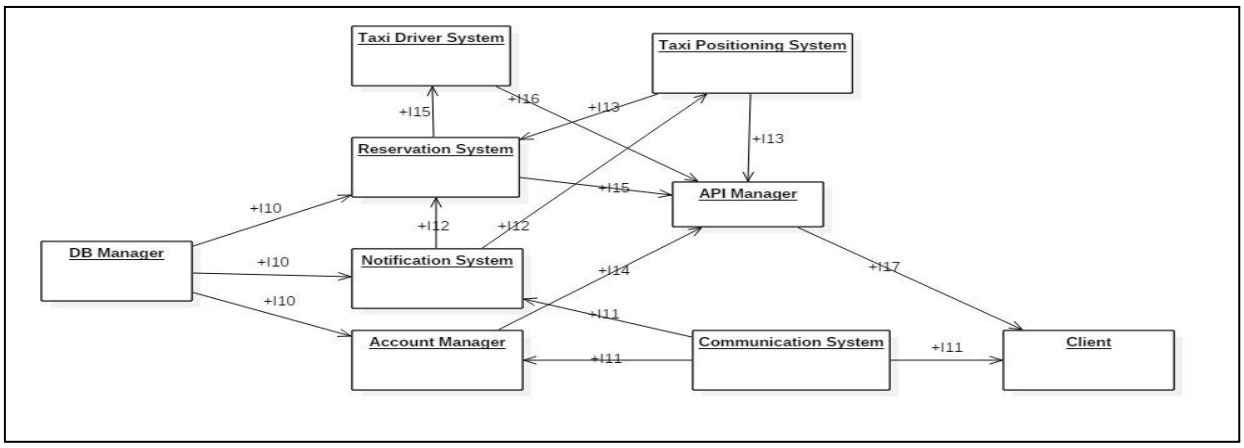

| ID | Integration Test | Paragraphs |
|---|---|---|
| I6 | Taxi Drivers → Taxi Driver Manager | 3.1.6 |

### 2.4.1.6 Reservation System




| ID | Integration Test | Paragraphs |
|---|---|---|
| I7 | Reservations → Reservation Scheduler | 3.1.7 |
| I8 | Reservation Scheduler → Reservation Manager | 3.1.8 |

### 2.4.2 Subsystem integration sequence




| ID | Integration Test | Paragraphs |
|---|---|---|
| I9 | DB Manager → Account Manager<br>DB Manager → Reservation System<br>DB Manager → Notification System | 3.2.1 |
| I10 | Communication System → Notification System<br>Communication System → Account Manager<br>Communication System → Client | 3.2.2 |
| I11 | Notification System → Reservation System<br>Notification System → Taxi Positioning System | 3.2.3 |
| I12 | Taxi Positioning System → Reservation System<br>Taxi Positioning System → API Manager | 3.2.4 |
| I13 | Account Manager → API Manager | 3.2.5 |
| I14 | Reservation System → Taxi Driver System<br>Reservation System → API Manager | 3.2.6 |
| I15 | Taxi Driver System → API Manager | 3.2.7 |
| I16 | API Manager → Client | 3.2.8 |

# 3. Individual Steps and Test Descriptions

We assume in the environmental needs that the previous tests were done successfully.

## 3.1 Subsystem Component Integration

We suppose that for each output specification the error are handled correcctly.

### 3.1.1 Integration Test Case I1

| | |
|---|---|
| **Test Case Identifier** | I1T1 |
| **Test Item(s)** | Connected Clients→ Communication Manager |
| **Input Specifications** | Simulate a Client connection |
| **Output Specifications** | Check if the Client has been added to the connected Clients' list. |
| **Environmental Needs** | Client |

**Input Specification Details**

1. Open a Socket Connection to our server
2. Close a previously open socket connection

**Output Specification Details**

1. Check if the Client has been added to the connected Clients' list.
2. Check if the Client has been removed to the connected Clients' list.

### 3.1.2 Integration Test Case I2

| | |
|---|---|
| **Test Case Identifier** | I2T1 |
| **Test Item(s)** | Communication Manager →GUI Client |
| **Input Specifications** | Test the connection between the two components and the return answers. |
| **Output Specifications** | Check if the link between the two components effectively works. I can see it from the correct/wrong return answers. |
| **Environmental Needs** | Communication Manager Driver |

In this kind of test we test that for each user interface ,that we defined in the RASD, that needs the comunication, calls correctly the interfaces exposed by the communication manager. Of course, since the communication manager needs the other system to be completely developed and tested, this test will be more accurate when integration testing is over and all the other components work correctly between each other.

### 3.1.3 Integration Test Case I3

| | |
|---|---|
| **Test Case Identifier** | I3T1 |
| **Test Item(s)** | Notifications→Notifier |
| **Input Specifications** | Create typical notifier input |
| **Output Specifications** | Check if the notifications has been correctly added to the notifications list and once sent removed to the same list |
| **Environmental Needs** | Notifier Driver |

**Input Specification Details**

1. Create a notification of incoming taxi.
2. Create a notification of a new client request to the taxi driver.
3. Create a notification of assigned zone changed.

**Output Specification Details**

I have to check that after the notifier took care of each notification, these notifications are added to the notifications' list until they are correctly sent to the clients.

### 3.1.4 Integration Test Case I4

| | |
|---|---|
| **Test Case Identifier** | I4T1 |
| **Test Item(s)** | Taxis→Zones |
| **Input Specifications** | Execute typical zones methods |
| **Output Specifications** | Check if Taxis were added ore removed to the zones lists. |
| **Environmental Needs** | Zones Driver |

**Input Specification Details**

1. Add a new taxi to a zone.
2. Remove a taxi to a zone.
3. Set the desired number of taxi for each zone.

**Output Specification Details**

1. A taxi was correctly added to the taxi's component of the specified zone.
2. A taxi was correctly removed to the taxi's component of the specified zone.
3. The desired number of taxis was correctly set in the specified zone.

### 3.1.5 Integration Test Case I5

| | |
|---|---|
| **Test Case Identifier** | I5T1 |
| **Test Item(s)** | Zones→Taxi Positioning Manager |
| **Input Specifications** | Execute typical taxi positioning manager methods |
| **Output Specifications** | Check if the correct methods are called in the zones classes. |
| **Environmental Needs** | Taxi Positioning Driver |

**Input Specification Details**

1. Specify a new GPS position for a taxi.

**Output Specification Details**

1. The taxi is added in the correct zone for the specified GPS location.

### 3.1.6 Integration Test Case I6

| | |
|---|---|
| **Test Case Identifier** | I6T1 |
| **Test Item(s)** | Taxi Drivers →Taxi Driver Manager |
| **Input Specifications** | Execute typical taxi driver manager methods |
| **Output Specifications** | Check if the taxi drivers list is correctly handled |
| **Environmental Needs** | Taxi Driver Manager Driver |

**Input Specification Details**

1. Create a new ride, that has been accepted by the taxi driver.
2. Specify a new state for the taxi driver.
3. Specify a new taxi for the taxi driver.

**Output Specification Details**

1. Check that the taxi driver manager component calls the accept existing ride interface of the reservation system correctly.

2. A new state was correctly set for the specified taxi driver.
3. The new taxi was correctly set for the specified taxi driver.

### 3.1.8 Integration Test Case I7

| | |
|---|---|
| **Test Case Identifier** | I7T1 |
| **Test Item(s)** | Reservations →Reservation Scheduler |
| **Input Specifications** | Create a typical taxi reservation |
| **Output Specifications** | Check if the reservations listi s correctly handled |
| **Environmental Needs** | Reservation Scheduler driver |

**Input Specification Details**

1. Create an istantaneous reservation.
2. Create a future reservation.

**Output Specification Details**

1. No reservations were added to the reservations' list.
2. The future reservation was correctly stored in the reservations component and the reservations list is ordered in a ascendent way based on the reservation date.

### 3.1.9 Integration Test Case I8

| | |
|---|---|
| **Test Case Identifier** | I8T1 |
| **Test Item(s)** | Reservation Scheduler→Reservation Manager |
| **Input Specifications** | Create a typical taxi reservation |
| **Output Specifications** | Check if a reservation is scheduled properly by the reservation scheduler |
| **Environmental Needs** | Reservation Manager driver, I7 |

**Input Specification Details**

1. Specify a new reservation.
2. Specify an existing reservation that has to be deleted.
3. Specify a reservation that has to be updated.
4. Specify a reservation that must be accepted by the taxi driver.

**Output Specification Details**

1. An istantaneous reservation was handled correctly by the scheduler and a separate thread was started to handle the entire cycle of the reservation(notify the taxi driver, accept the ride ecc…)

2. Check that the reservation has been correctly deleted from the scheduler component and consequently from the reservation component.
3. Check that the reservation has been correctly updated in the scheduler component and consequently in the reservation component. In this step it is also necessary to verify if the update stored correctly the updated data in the db.
4. Verify that the reservation manager informs correctly the reservation scheduler about the reservation acceptance.

## 3.2 Subsystems Integration

In this section we want to integrate the subsystems of our application. There are some subsystems(e.g. taxi positioning system) in which there are components not relevant for the subsystems integration and components that were largely tested before. In the case explained as example before the taxis list and the zones list aren't useful for the sub system integration and the main component (taxi location manager) has already been tested. The main scope of this sub system integration is to verify that the interfaces of each component work properly in a coherent way by comparing that to the internal components.

### 3.2.1 Integration Test Case I9

| | |
|---|---|
| **Test Case Identifier** | I9T1 |
| **Test Item(s)** | DB Manager → Account Manager |
| **Input Specifications** | Test all the account manager functionalities that involve DB. |
| **Output Specifications** | Check if the information was correctly read and stored in the DB |
| **Environmental Needs** | Account Manager Driver, Communication System Stub |

**Input Specification Details**

All the following inputs must be specified in SQL language and the corresponding strings must be passed to the execute query method in the DB manager.

1. **Create user:**Specify new user data in order to create a new account.
2. **Login:** Specify username and password.
3. **Modify profile Data:** Specify new user data in order to modify an existing account.
4. **Update Password:** Specify new password.

**Output Specification Details**

1. The account has been correctly created in the database with the specified user profile data.
2. The account with the specified username and password has been correctly found in the database.
3. Data have been correctly modified in the DB with the specified new user profile data.
4. Password has been correctly modified in the database.

| | |
|---|---|
| **Test Case Identifier** | I9T2 |
| **Test Item(s)** | DB Manager → Reservation System |
| **Input Specifications** | Test all the reservation System functionalities that involves the DB. |
| **Output Specifications** | Check if the information was correctly read and stored in the DB. |
| **Environmental Needs** | Reservation System Driver, Notification System Stub, Taxi Positioning System Stub. |

**Input Specification Details**

1. **Answer new ride:** Specify a new reservation.
2. **Delete reservation:** Specify an existing reservation to be deleted.
3. **Update reservation:** Specify an existing reservation and the new data associated to this reservation to be updated.
4. **Accept existing ride:** Specify a taxi driver and an existing reservation to simulate the acceptance of a reservation by a taxi driver.

**Output Specification Details**

1. The new reservation was correctly stored in the database.
2. The specified reservation was correctly deleted.
3. The specified reservation was correctly updated with the new data.
4. The specified ride has been correctly associated to the specified reservation.

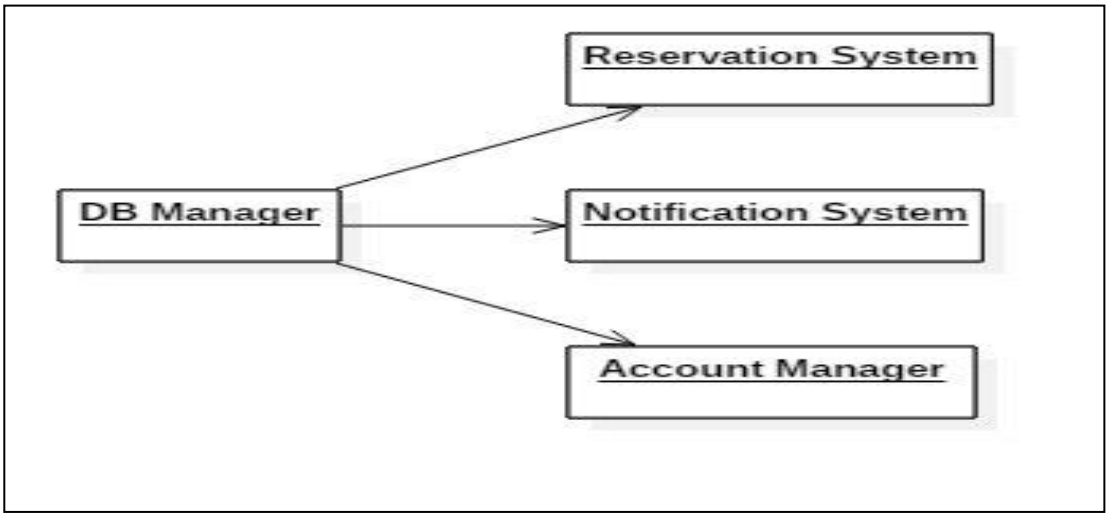
| | |
|---|---|
| **Test Case Identifier** | I9T3 |
| **Test Item(s)** | DB Manager → Notification System |
| **Input Specifications** | Test all the Notification System functionalities that involves the DB |
| **Output Specifications** | Check if the notification was correctly read and stored in the database. |
| **Environmental Needs** | Notification System Driver, Communication System Stub. |

**Input Specification Details**

1. **Notify:** Specify a notification.

**Output Specification Details**

1. Check that the notification has been correctly stored in the database.




### 3.2.2 Integration Test Case I10

| | |
|---|---|
| **Test Case Identifier** | I10T1 |
| **Test Item(s)** | Communication System → Notification System |
| **Input Specifications** | Test the functionalities of the communication system. |
| **Output Specifications** | Check if the notification is correctly sent by socket to the client. |
| **Environmental Needs** | Notification System Driver, Client.<br>I10 Done Successfully. |

**Input Specification Details**

1. **Send Message:** Specify a new notification object as a serializable object.

**Output Specification Details**

1. Check that the notification sent has been correctly received by the client.

| | |
|---|---|
| **Test Case Identifier** | I10T2 |
| **Test Item(s)** | Communication System → Account Manager |
| **Input Specifications** | Test the associated client functionalities and the deassociate client functionalities. |
| **Output Specifications** | Check the functionalities associated are called. |
| **Environmental Needs** | Account Manager Driver, Client, i10 done successfully. |

**Input Specification Details**

1. **Associate Client:** Specify the username of a logged in client.
2. **Deassociate Client:** Specify the username of a logged out client.

**Output Specification Details**

1. The specified username has been correctly associated to the correct client in the communication system manager(the one associated with the client socket).
2. The specified username has been correctly deassociated to the correct client in the communication system manager(the one associated with the client socket).

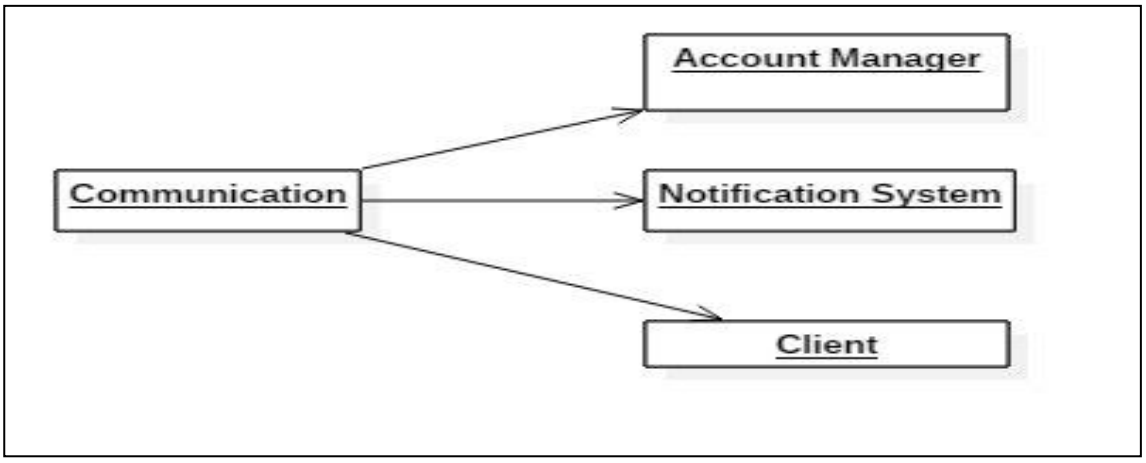
| | |
|---|---|
| **Test Case Identifier** | I10T3 |
| **Test Item(s)** | Communication System → Client |
| **Input Specifications** | Test if the Communication System opens/closes correctly the connection via socket. |
| **Output Specifications** | Check if the account socket is correctly added to the sockets' list. |
| **Environmental Needs** | Client, API Manager Stub. |

**Input Specification Details**

1. **Open Connection:** Open a socket connection with the server.
2. **Close Connection:** Close a socket connection previously open with the server.

**Output Specification Details**

1. Check if the client has been correctly added in the connected clients' list.
2. Check if the client has been correctly removed to the connected clients' list.



### 3.2.3 Integration Test Case I11

| | |
|---|---|
| **Test Case Identifier** | I11T1 |
| **Test Item(s)** | Notification System → Reservation System |
| **Input Specifications** | Test if the notification is sent correctly. |
| **Output Specifications** | Check if the notification was correctly handled by the notification system. |
| **Environmental Needs** | Reservation System Driver, Taxi Positioning System Stub, i10 done successfully. |

**Input Specification Details**

1. **Notify:** Specify a new reservation notification

**Output Specification Details**

1. Check if the notification was added to the notifications' list.

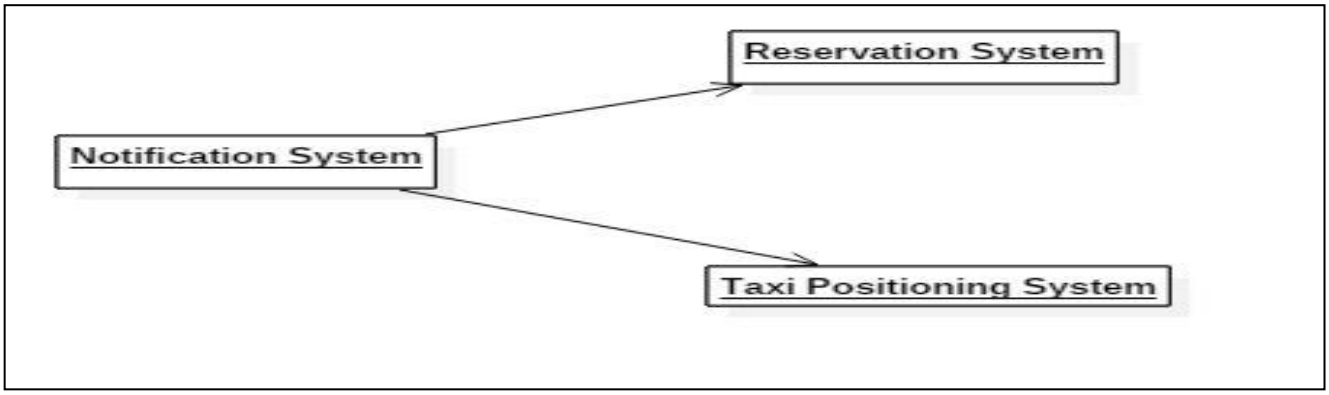
| | |
|---|---|
| **Test Case Identifier** | I11T2 |
| **Test Item(s)** | Notification System → Taxi Positioning System |
| **Input Specifications** | Test if the notification is sent correctly. |
| **Output Specifications** | Check if the notification was correctly handled by the notification system. |
| **Environmental Needs** | Taxi Positioning System Driver, i10 done successfully, i11 done successfully. |

**Input Specification Details**

1. **Notify:** Specify a new positioning notification.

**Output Specification Details**

1. Check if the notification was correctly added to the notifications' list.



### 3.2.4 Integration Test Case I12

| | |
|---|---|
| **Test Case Identifier** | I12T1 |
| **Test Item(s)** | Taxi Positioning System → Reservation System |
| **Input Specifications** | Specify a new reservation |
| **Output Specifications** | Check if the location of the nearest taxi and his zone associated are returned to the reservation system. |
| **Environmental Needs** | Reservation System Driver, I12 done successfully, i10 done successfully. |

**Input Specification Details**

1. **Ask nearest taxi:** Specify a source zone.

**Output Specification Details**

1. Check if the nearest taxi from the source zone is returned.

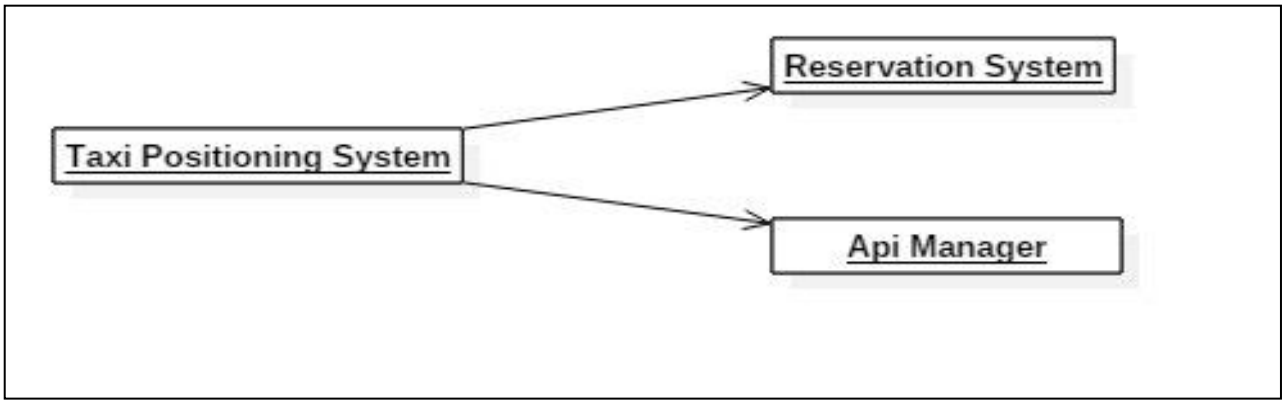
| | |
|---|---|
| **Test Case Identifier** | I12T2 |
| **Test Item(s)** | Taxi Positioning System → API Manager |
| **Input Specifications** | Specify a new location for a taxi. |
| **Output Specifications** | Check if the location was changed correctly. |
| **Environmental Needs** | API Manager Driver, Account Manager Stub, Reservation System Stub, Taxi Driver System Stub, i10 done successfully, i12 done successfully, i11 done successfully. |

**Input Specification Details**

1. **Update Location:** Specify a taxi and the new GPS position for its.

**Output Specification Details**

1. The taxi location has been correctly updated and the taxi has been put in the correct zone.



### 3.2.5 Integration Test Case I13

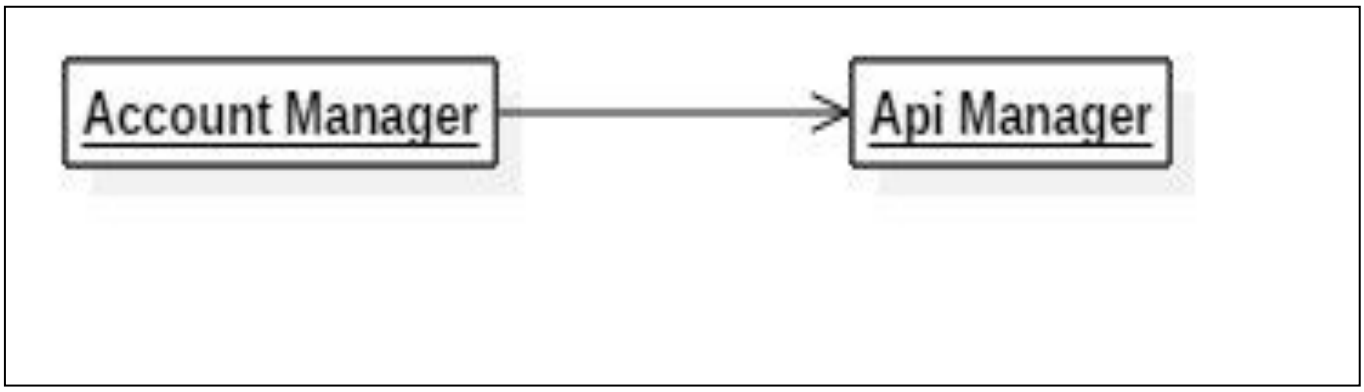
| | |
|---|---|
| **Test Case Identifier** | I13T1 |
| **Test Item(s)** | Account Manager → API Manager |
| **Input Specifications** | Test all account manager operations |
| **Output Specifications** | Check if all the operations are handled correctly. |
| **Environmental Needs** | API Manager Driver, Reservation System Stub, Taxi Driver System Stub, i10 done successfully, i11 done successfully, i12 done successfully, i13 done successfully. |

**Input Specification Details**

1. **Create User:** Specify a new user to be created.
2. **Login User:** Specify user credentials: first wrong credentials and after right credentials.
3. **Logout User:** Specify the username of a user to be logged out.
4. **Modify Profile Data:** Specify a user and the new information to be updated.
5. **Update Password:** Specify the old and the new password, first with wrong old password, then with right old password.

**Output Specification Details**

1. The user has been correctly created in the database.
2. In the first case(with wrong credentials) an exception is thrown and correctly handled; in the second case(with right credentials) the user has successfully logged in.
3. The user has successfully logged out.
4. The information has been correctly updated in the database.
5. In the first case(with wrong credentials) an exception is thrown and correctly handled; in the second case(with right credentials) the password has successfully been changed.



### 3.2.6 Integration Test Case I14

| | |
|---|---|
| **Test Case Identifier** | I14T1 |
| **Test Item(s)** | Reservation System → Taxi Driver System |
| **Input Specifications** | A typical taxi driver operation |
| **Output Specifications** | Check if the operations are handled correctly by the reservation system |
| **Environmental Needs** | Taxi Driver System Driver, i10 done successfully, i11 done successfully, i12 done successfully, i13 done successfully. |

**Input Specification Details**

1. **Accept Existing Ride:** Specify the ID of a reservation and the username of the taxi that has accepted that reservation.

**Output Specification Details**

1. The reservation has been correctly associated with the specified taxi driver.

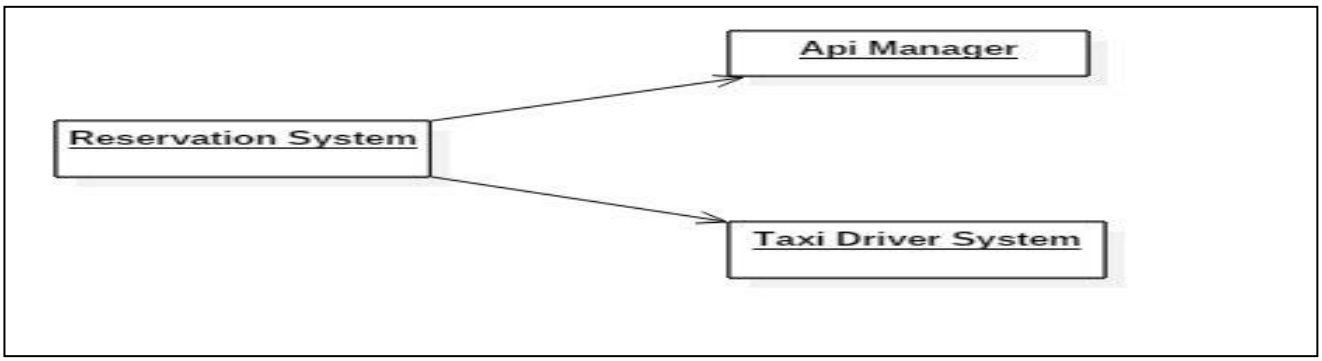
| | |
|---|---|
| **Test Case Identifier** | I14T2 |
| **Test Item(s)** | Reservation System → API Manager |
| **Input Specifications** | A typical taxi driver operation |
| **Output Specifications** | Check if all the operations are handled correctly by the reservation system. |
| **Environmental Needs** | API Manager Driver, Taxi Driver System Stub, i10 done successfully, i11 done successfully, i12 done successfully, i13 done successfully, i14 done successfully. |

**Input Specification Details**

1. **Answer new ride:** Specify a new reservation.(both types).
2. **Delete reservation:** Specify the reservation to be deleted(first with an istantaneous reservation, then with a future reservation).
3. **Update reservation:** Specify the reservation to be updated and the data necessary for the update.(first with an istantaneous reservation, then with a future reservation and a wrong date, at the end with a future reservation and a correct date)

**Output Specification Details**

1. A new reservation has been correctly handled by the reservation system.
2. If the reservation was an istantaneous one, an exception is thrown and it's correctly handled. Otherwise, if it hasnt been converted in an istantaneous one yet, a new reservation has correctly been deleted from the database and from the reservations list.
3. If the reservation was an istantaneous one, an exception is thrown and it's correctly handled. Otherwise a new reservation has correctly been updated.




### 3.2.7 Integration Test Case I15

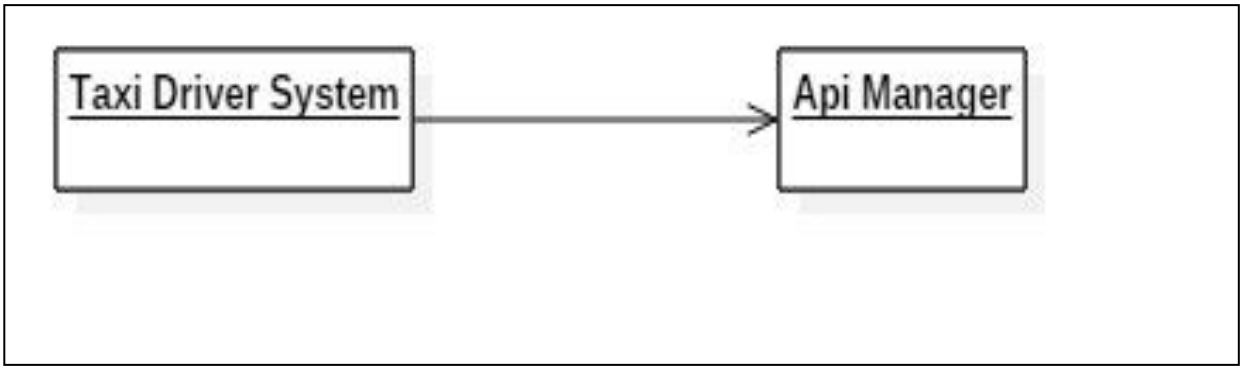
| | |
|---|---|
| **Test Case Identifier** | I15T1 |
| **Test Item(s)** | Taxi Driver System → API Manager |
| **Input Specifications** | Test all the taxi driver system functionalities. |
| **Output Specifications** | Check if if all the operations are handled correctly by the taxi driver system. |
| **Environmental Needs** | API Manager Driver, i10 done successfully, i11 done successfully, i12 done successfully, i13 done successfully, i14 done successfully, i15 done successfully. |

**Input Specification Details**

1. **Accept ride:** Specify a taxi driver username and a reservation.
2. **Set State:** Specify a taxi driver username and a new taxi driver state.
3. **Associate taxi:** Specify a taxi driver username and a taxi.

**Output Specification Details**

1. The taxi driver and the reservation have been correctly associated in the reservation system.
2. The taxi driver state has been correctly updated.
3. The specified taxi has been correctly associated to the specified taxi driver in the taxi driver list.



### 3.2.8 Integration Test Case I16

| | |
|---|---|
| **Test Case Identifier** | I16T1 |
| **Test Item(s)** | API Manager → Client |
| **Input Specifications** | A typical operation made by the client |
| **Output Specifications** | Check if the correct APIs are called |
| **Environmental Needs** | Client, i10 done successfully, i11 done successfully, i12 done successfully, i13 done successfully, i14 done successfully, i15 done successfully, i16 done successfully. |

**Input Specification Details**

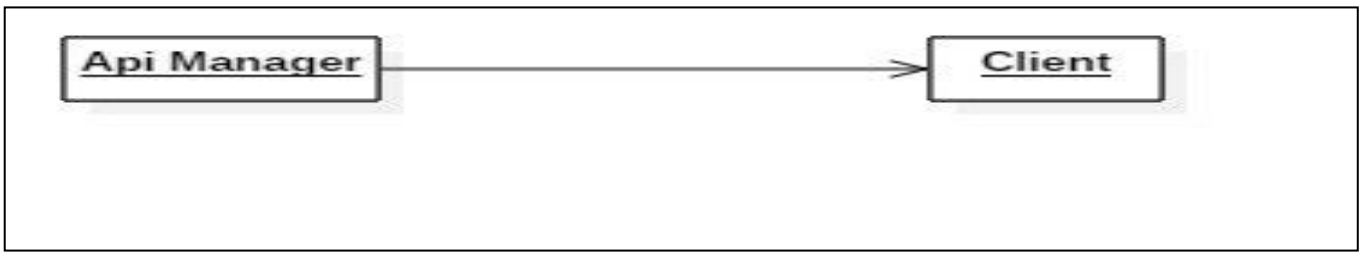
1. **Set state:** Specify a new state and the name of a taxi driver.
2. **Update GPS Location:**Specify a new GPS location and the name of the taxi driver.
3. **Answer Reservation:** Specify the username and the ride ID.
4. **Update Password:** Specify the new and old password and the username of the profile to update.
5. **Occasional Taxi Reservation:** Specify the gps source position, the destination and the username of the taxi driver that accepted the request.
6. **Arrived Destination:** Specify the username of the taxi driver, the ride id and the time taken to satisfy the ride.
7. **Choose Taxi:** Specify the username of the taxi driver and the taxi ID.
8. **Store Ride Data:** Specify the username of the taxi driver, the number of kms and the price.
9. **Get Information:** Specify the type of the information requie, the username of the user that asks the information and the user password.
10. **Arrived Client:** Specify the username of the taxi driver, the ido f the ride and the time.
11. **Pick up Client:** Specify the username and the ride id.
12. **Login:** Specify the username and the password of the client.
13. **Update Reservation:** Specify the username, the id reservation, the reservation date, the reservation time, the reservation source and the reservation destination.
14. **Delete Reservation:** Specify the reservation id, the username and the password.
15. **Logout:** Specify the username of the client.
16. **Registration:** Specify all the data needed for the registration.
17. **Taxi Reservation:** Specify all the data needed for the taxi reservation.

**Output Specification Details**

If the operation has been executed successfully , all the return information are incluede in the XML document and i fan error occorre, information about the type of errori s returned to the client.

1. Check that the state of the specified taxi driver has been correctly updated in the taxi driver system.
2. Check that the Position of the specified taxi was correctly updated in the taxi positioning system.
3. Check that the reservation ansie has been correctly handled by the taxi driver system.
4. Check that the password has been correctly changed.
5. Check that the occasional taxi reservation has been correctly stored in the database and the state of the taxi driver that took care of this ride is set to busy.

6. Check that the time needed to arrive at destination has been correctly stored in the database.
7. Check that the specified taxi has correctly been associated to the taxi driver.
8. Check that the ride data has been correctly saved in the database.
9. Check that the requested information are correctly read by the DB.
10. Check that the time needed to arrive to the client is saved in the DB.
11. Check that the information are stored correctly in the DB.
12. Check that the login has been corrrectly handled.
13. Check that the reservation has correctly been updated and any error has been correctly handled.
14. Check that the reservation has correctly been deleted.
15. Check that the client has been removed to the client list in the socket.
16. Check that all the client information has been correctly stored in the DB.
17. Check that all the information of the taxi reservation has been correctly stored in the DB.




## 4. Tools and Equipment Required

We assume that we used only manual testing because we don't know any particolar automated tool that can be used to speed up the integration testing.

## 5. Programs Stubs and Tests Data Required

We identified all drivers and stubs needed in each integration test explained above. In the database there must be some accounts defined to test the database functionalities, for example login, modify profile data, ecc… We also tested the socket connection with a test program for each platform we support that tested the correct functionalities of the socket in the local network and in the real network. This allows us to test in particolar the interfaces open connecion and close connection of the communication system and to add clients in its connected clients component.

Now we will recap all the drivers and stubs used in the integration testing. They are reported here just for a better readability of the document.

## 5.1 Drivers

### 5.1.1 Account Manager Driver

**Interfaces :**

1. **Create User**
2. **Login**
3. **Modify Profile Data**
4. **Update Password**

### 5.1.2 Reservation System Driver

**Interfaces :**

1. **Answer new Ride**
2. **Delete Reservation**
3. **Update Reservation**
4. **Accept Existing Ride**

### 5.1.3 Notification System Driver

**Interfaces :**

1. **Notify**

### 5.1.4 API Manager Driver

**Interfaces :**

1. **Set State**
2. **Update GPS Location**
3. **Answer Reservation**
4. **Update Password**
5. **Occasional Taxi Reservation**
6. **Arrived Destination**
7. **Choose Taxi**
8. **Store Ride Data**
9. **Get Information**
10. **Arrived Client**
11. **Pick up Client**
12. **Login**
13. **Update Reservation**
14. **Delete Reservation**
15. **Logout**
16. **Registration**
17. **Taxi Reservation**

### 5.1.5 Taxi Positioning System Driver

**Interfaces :**

1. **Update Location**
2. **Ask Nearest Taxi**

## 5.2 Stubs

### 5.2.1 Communication System Stub

**Interfaces :**

1. **Open Connection**
2. **Close Connection**
3. **Associate Client**
4. **Deassociate Client**
5. **Send Message**

### 5.2.2 Notification System Stub

**Interfaces :**

1. **Notifier**

### 5.2.3 Taxi Positioning System Stub

**Interfaces :**

1. **Update Location**
2. **Ask Nearest Taxi**

### 5.2.4 Taxi Driver System Stub

**Interfaces :**

1. **Accept Ride**
2. **Set State**
3. **Associate Taxi**

### 5.2.5 API Manager Stub

**Interfaces :**

1. **Set State**
2. **Update GPS Location**
3. **Answer Reservation**
4. **Update Password**
5. **Occasional Taxi Reservation**
6. **Arrived Destination**

7. **Choose Taxi**
8. **Store Ride Data**
9. **Get Information**
10. **Arrived Client**
11. **Pick Up Client**
12. **Login**
13. **Update Reservation**
14. **Delete Reservation**
15. **Logout**
16. **Registration**
17. **Taxi Reservation**

### 5.2.6 Account Manager Stub

**Interfaces :**

1. **Create User**
2. **Login User**
3. **Logout User**
4. **Modify Profile Data**
5. **Update Password**

### 5.2.7 Reservation System Stub

**Interfaces :**

1. **Accept Existing Ride**
2. **Answer new Ride**
3. **Delete Reservation**
4. **Update Reservation**

Total Working Hours:

Ivan Antozzi 15 hours

Riccardo Giambona 15 hours