POLITECNICO
MILANO 1863

# PowerEnjoy

## Integration Test Plan Document

Version 1.0


**Software Engineering 2 (A.A. 2016/2017)**

- Simone Boglio (mat. 772263)
- Lorenzo Croce (mat. 807833)

# Summary

# 1. Introduction

## 1.1. Revision History

This is the first version of the document. There are not previous versions.

| Revision | Last Edit | Changes |
|---|---|---|
| 1.0 | xx/01/2016 | Document first redaction |

## 1.2. Purpose and Scope

The Integrated Test Planning Document (ITPD) describes the plan to accomplish the integration test. This document is supposed to be written before the integration test happens and takes the architectural description of the software system as a starting point, for this reason it is often redacted in parallel with the Design Document.
It explains to the development team what to test, in which sequence, which tools are needed for testing (if any), which stubs/drivers/oracles need to be developed.

The purpose of integration testing is to verify functional, performance, and reliability requirements of individual software modules of the product when they are combined and tested as a group; i.e., units (or groups of units) are exercised through their interfaces. The aim is to test the modules interactions incrementally, with success and error cases being simulated via appropriate parameter and data inputs.
Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.
Test cases are constructed to test whether all the components interact correctly, for example across procedure calls or process activations.
This is done after testing individual modules, i.e., unit testing; the overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages up to the complete final system (the testing on the complete system is not part of this integration testing phase).

## 1.3. List of Definitions and Abbreviations

The following acronyms are used in this document:

- RASD: Requirements Analysis and Specification Document.
- DD: Design Document.
- DB: DataBase. It is a test Database filled with test data.
- CU: Control Unit of a car.
- API: Application Programming Interface.

The following definitions are used in this document:

- Driver: are considered dummy modules which are always distinguished as calling programs that are handled in bottom up integration testing; they are only used when main programs are under construction.
- Stub: in computer science, test stubs are programs that simulate the behaviors of software components (or modules) that a module undergoing tests depends on. Test stubs are mainly used in incremental testing's top-down approach.
- Oracle: in computing, software testers and software engineers can use an oracle as a mechanism for determining whether a test has passed or failed. The use of oracles involves comparing the output(s) of the system under test, for a given test-case input, to the output(s) that the oracle determines that product should have.

## 1.4. List of Reference Documents

- Specification document: PowerEnjoy project
- Requirements Analysis and Specification Document (RASD)
- Design Document (DD)

# 2. Integration Strategy

## 2.1. Entry Criteria

It is supposed the unit testing phase has already been completed successfully.
In particular, the following components are considered already tested, so we consider them as working:

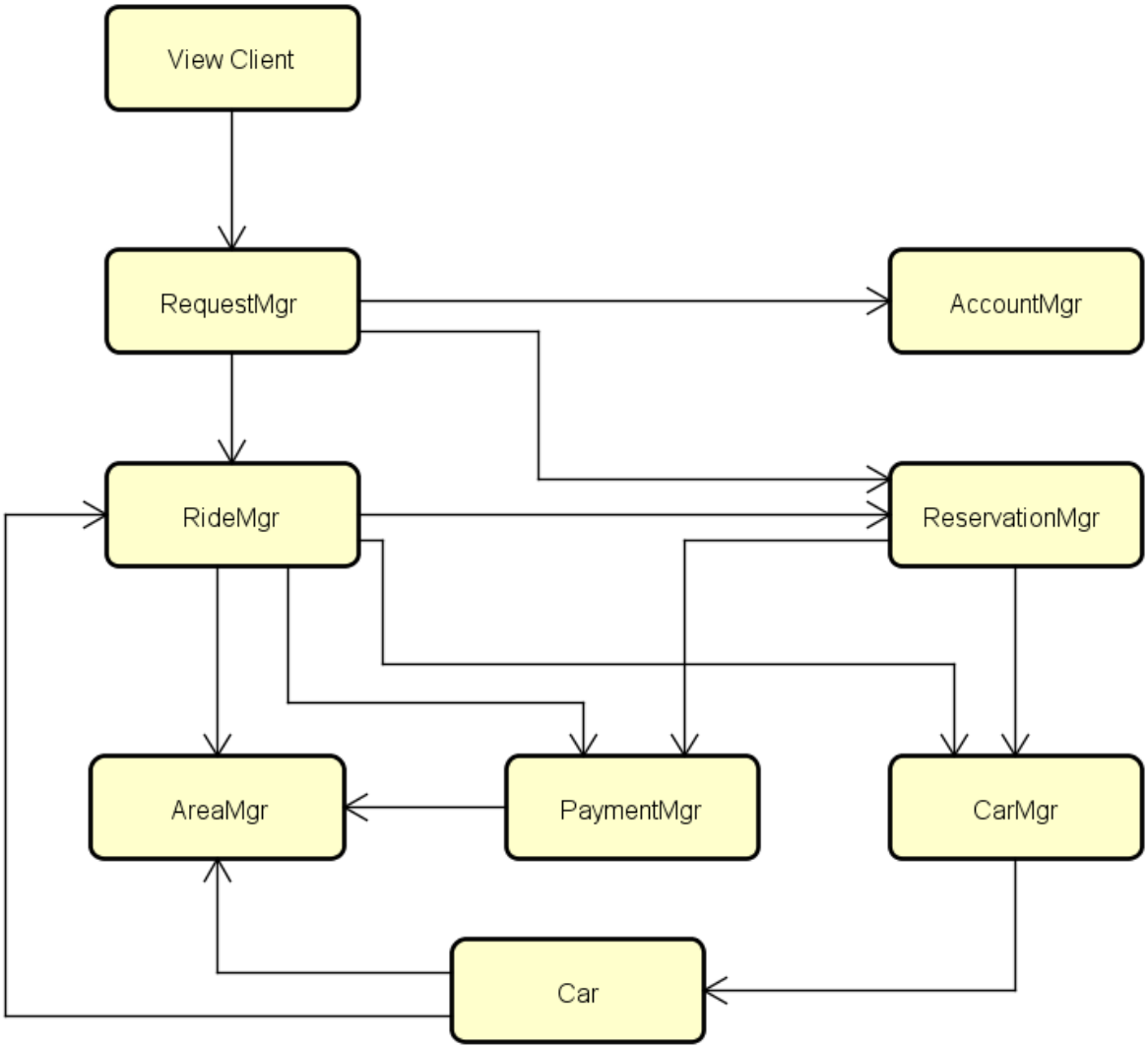
- Area Manager
- DB
- Legacy DB
- Google Map APIs
- External Payment Service APIs
- E-mail service
- SMS service

We consider the car subsystem already tested and worked (CU + Green e-Box + Display).
We will take care of the integration between the central system and the car (for more details you can refer to the DD document)

## 2.2. Elements to be Integrated

### 2.2.1. High Level Components Integration

The picture below shows the high-level components interaction of the PowerEnjoy system



### 2.2.2. Components Integration Table

This is the table of all integrations tests of the whole system, for each test it shows the corresponding paragraph where the test is analysed with much more details.
From test I1 up to test I7 the integration is related to sub-system.
The tests from I8 will be focused in integration of higher level components.

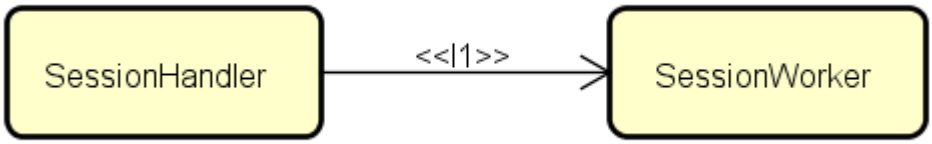
| ID | Integration Test | Paragraphs | | |
|---|---|---|---|---|
| | | Integration Sequence | Integration Test | Test Procedure |
| I1 | Session Handler -> Session Worker | 2.4.1 | 3.1.1 | 3.2.1 |
| I2 | Login Manager -> Account Factory | 2.4.2 | 3.1.2 | 3.2.2 |
| I3 | Request Handler -> Request Worker | 2.4.3 | 3.1.3 | 3.2.3 |
| I4 | Ride Handler -> Ride Worker | 2.4.4 | 3.1.4 | 3.2.4 |
| I5 | Car Handler -> Car Worker | 2.4.5 | 3.1.5 | 3.2.5 |
| I6 | Car Worker -> Car Data | 2.4.5 | 3.1.5 | 3.2.5 |
| I7 | Payment Manager -> Bonus Calculator | 2.4.6 | 3.1.6 | 3.2.6 |
| I8 | View Client -> Request Manager | 2.4.7 | 3.1.7 | 3.2.7 |
| I9 | Request Manager -> Account Manager | 2.4.8 | 3.1.8 | 3.2.8 |
| I10 | Request Manager -> Reservation Manager | 2.4.8 | 3.1.9 | 3.2.8 |
| I11 | Request Manager -> Ride Manager | 2.4.8 | 3.1.10 | 3.2.8 |
| I12 | Reservation Manager -> Car Manager | 2.4.9 | 3.1.11 | 3.2.9 |
| I13 | Reservation Manager -> Payment Manager | 2.4.9 | 3.1.12 | 3.2.9 |
| I14 | Ride Manager -> Reservation Manager | 2.4.10 | 3.1.13 | 3.2.10 |
| I15 | Ride Manager -> Car Manager | 2.4.10 | 3.1.14 | 3.2.10 |
| I16 | Ride Manager -> Area Manager | 2.4.10 | 3.1.15 | 3.2.10 |
| I17 | Ride Manager -> Payment Manager | 2.4.10 | 3.1.16 | 3.2.10 |
| I18 | Car Manager -> Green e-Box (Car) | 2.4.11 | 3.1.17 | 3.2.11 |
| I19 | Payment Manager -> Area Manager | 2.4.12 | 3.1.18 | 3.2.12 |
| I20 | Green e-Box (Car) -> Ride Manager | 2.4.13 | 3.1.19 | 3.2.13 |
| I21 | Green e-Box (Car) -> Area Manager | 2.4.14 | 3.1.20 | 3.2.13 |

## 2.3. Integration Testing Strategy

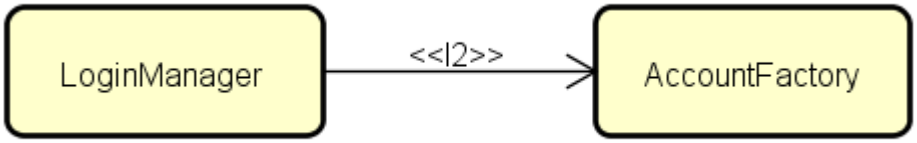
The integration strategy used is the bottom-up approach.
We start from low-level components integration to obtain the sub-system components; next we proceed to integrate the sub-system components among them. We iterate this process until all the components are integrated and the system is complete.

## 2.4. Sequence of Component/Function Integration

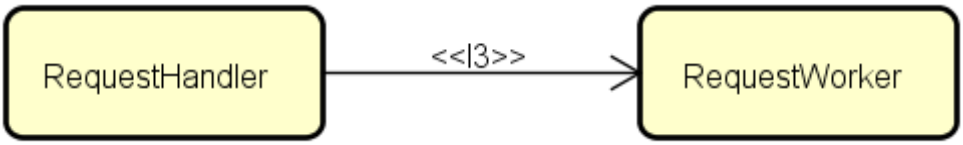
### 2.4.1. Sequence of Integration: Request Manager Sub-System



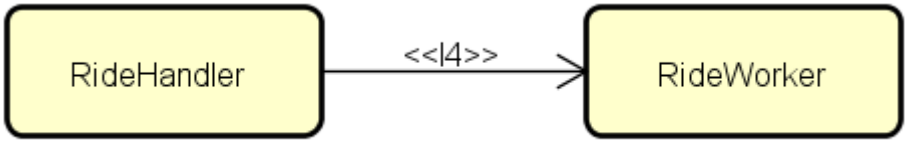

### 2.4.2. Sequence of Integration: Account Manager Sub-System



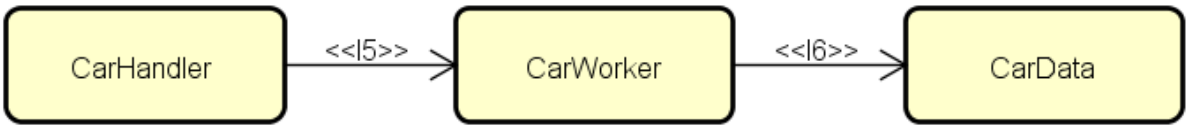

### 2.4.3. Sequence of Integration: Reservation Manager Sub-System



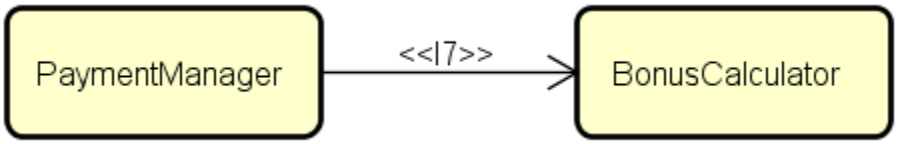

### 2.4.4. Sequence of Integration: Ride Manager Sub-System



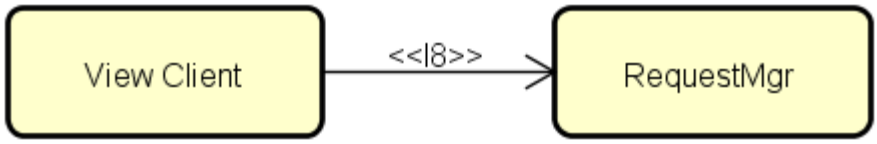
### 2.4.5. Sequence of Integration: Car Manager Sub-System



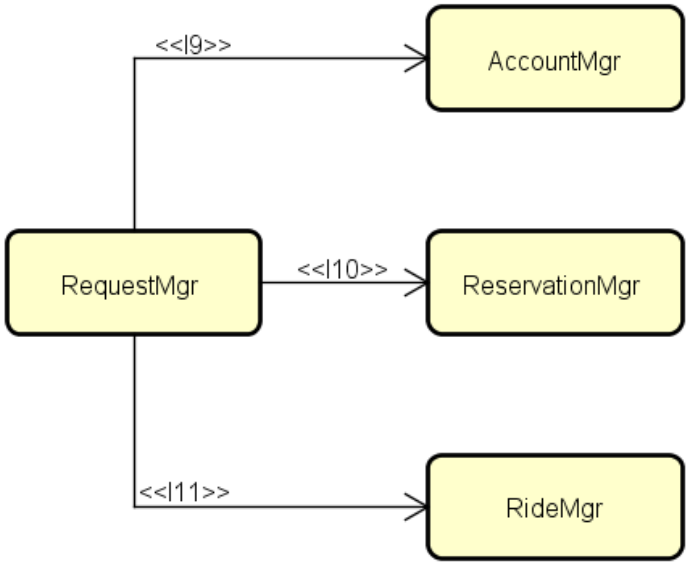

### 2.4.6. Sequence of Integration: Payment Manager Sub-System



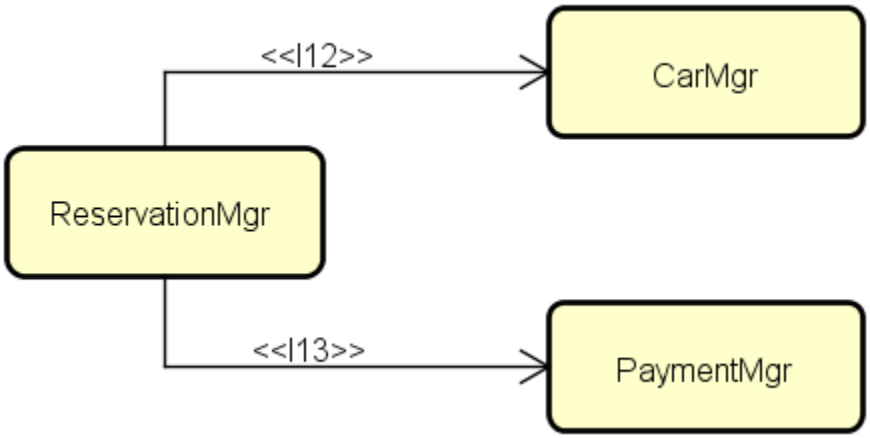

### 2.4.7. Sequence of Integration: View Client



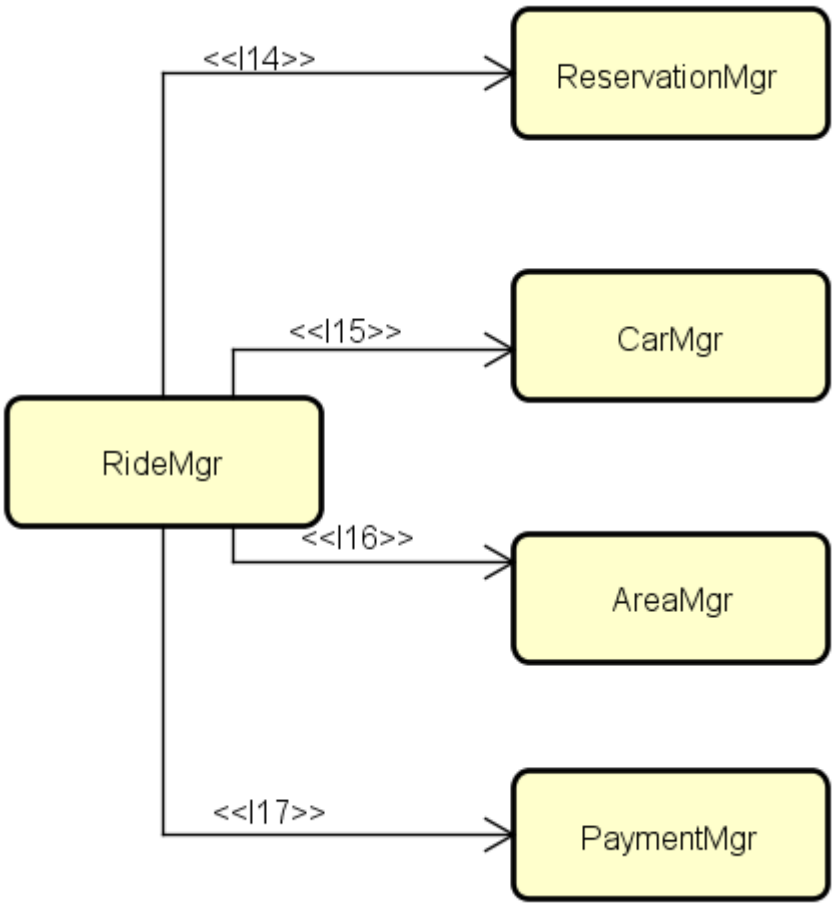

### 2.4.8. Sequence of Integration: Request Manager



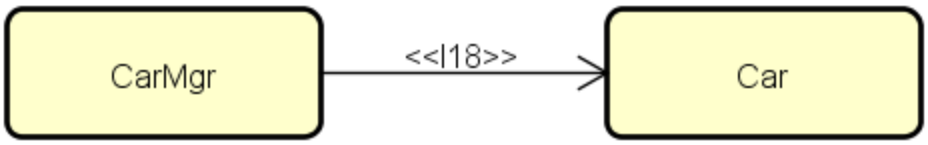
### 2.4.9. Sequence of Integration: Reservation Manager



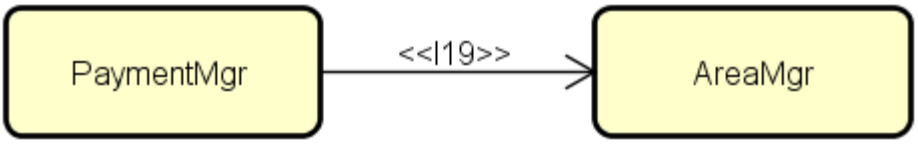

### 2.4.10. Sequence of Integration: Ride Manager



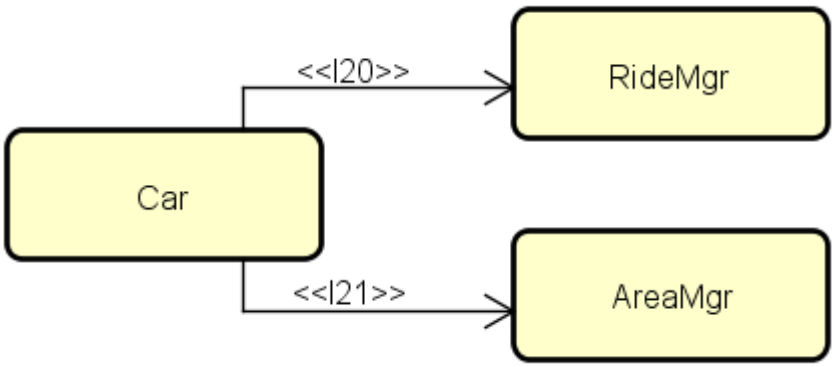
### 2.4.11. Sequence of Integration: Car Manager




### 2.4.12. Sequence of Integration: Payment Manager



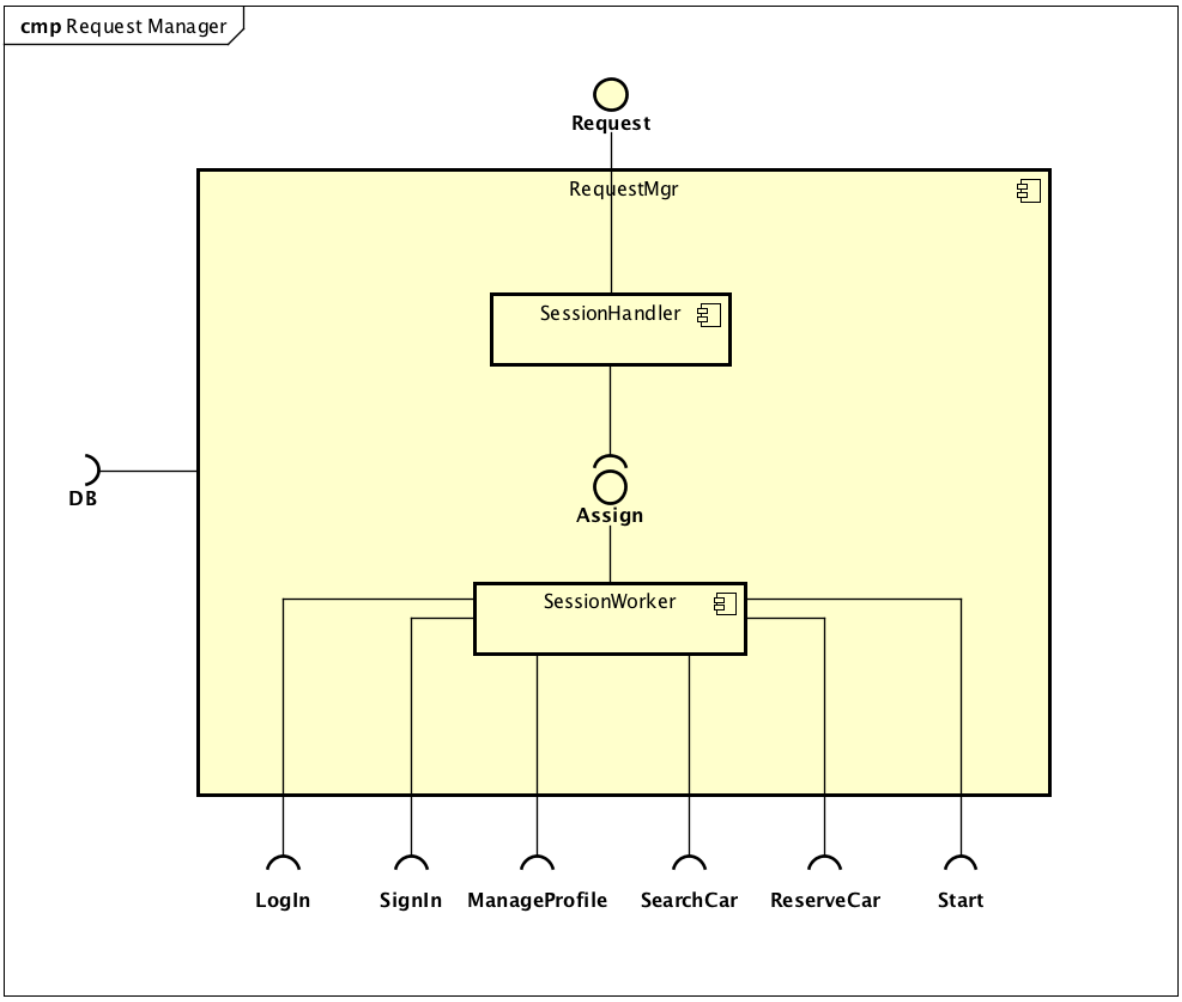

### 2.4.13. Sequence of Integration: Car System



# 3. Individual Steps and Test Description

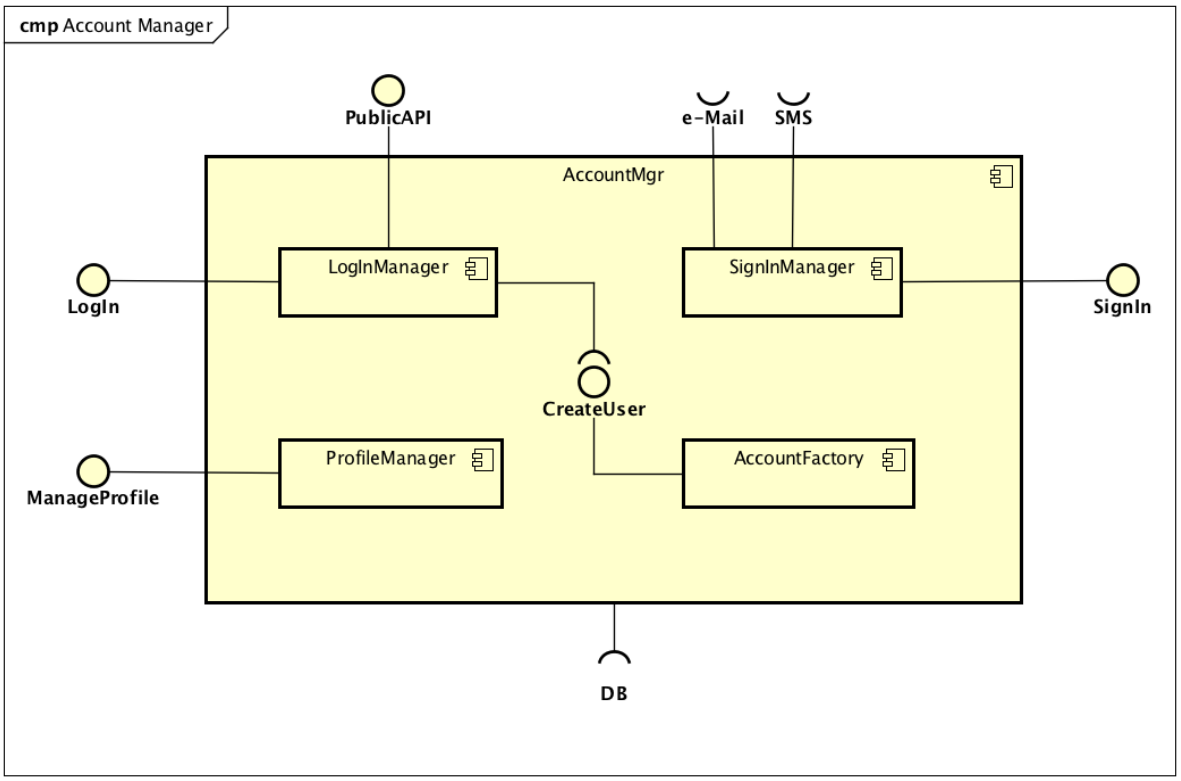
## 3.1. Test Case Specification

### 3.1.1. Integration Test: Request Manager Sub-System



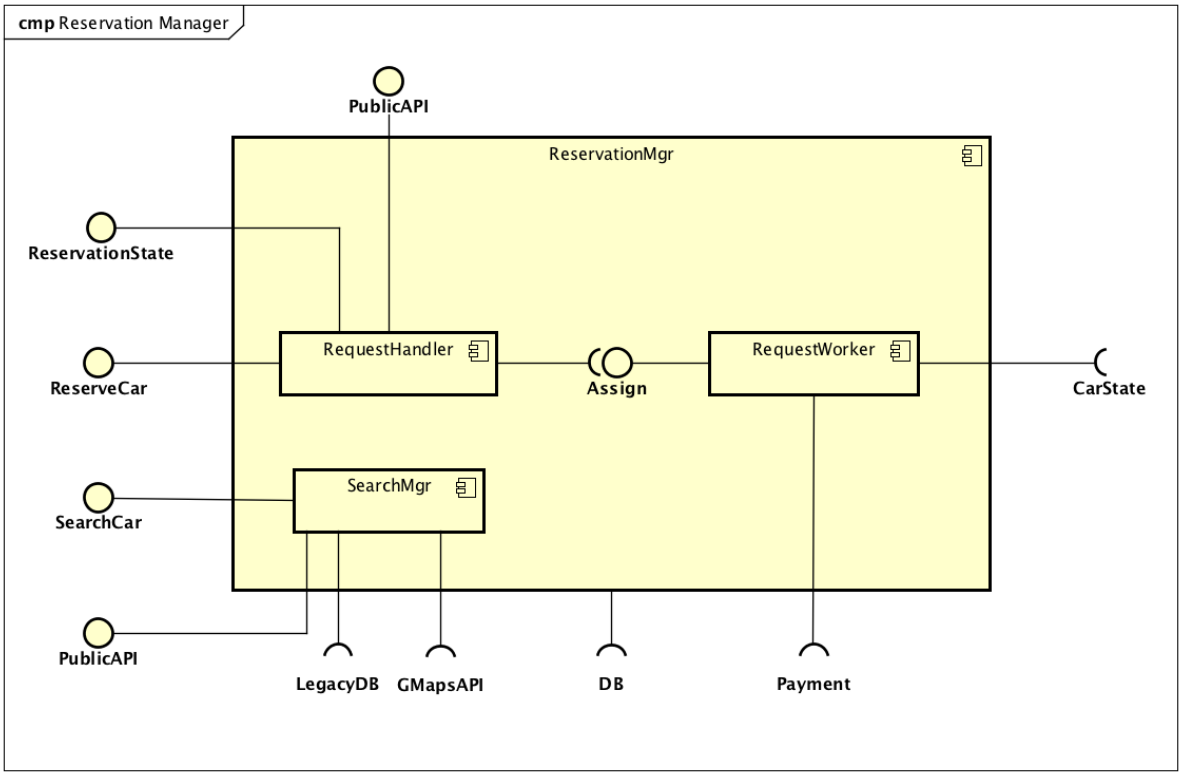

| | |
|---|---|
| **Test case ID** | I1 |
| **Test item(s)** | Session Handler -> Session Worker |
| **Input specification** | Create typical Session Handler input |
| **Output specification** | Check if the correct functions are called in the Session Worker |
| **Environmental needs** | • Client driver<br>• Account Manager stub<br>• Reservation Manager stub<br>• Ride Manager stub<br>• DB |

### 3.1.2. Integration Test: Account Manager Sub-System



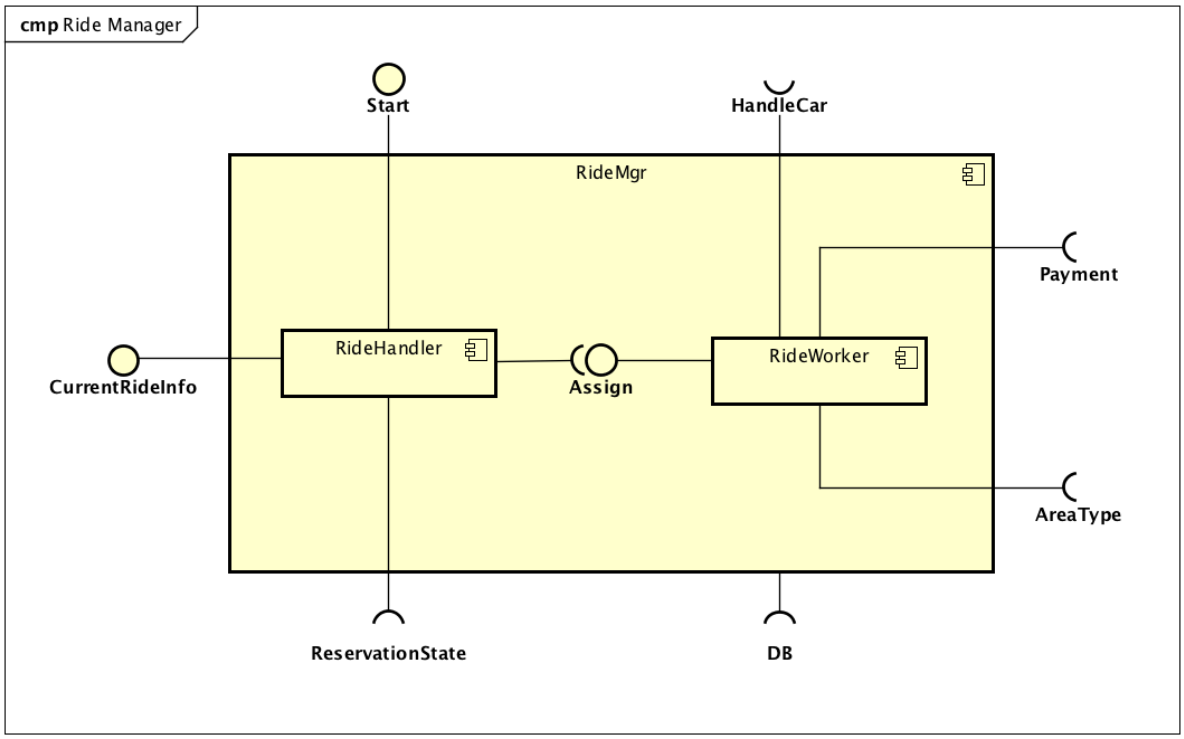

| | |
|---|---|
| **Test case ID** | I2 |
| **Test item(s)** | Login Manager -> Account Factory |
| **Input specification** | Create typical Login Manager input |
| **Output specification** | Check if the correct methods are called in the Account Factory |
| **Environmental needs** | • Request Manager driver<br>• DB |

### 3.1.3. Integration Test: Reservation Manager Sub-System



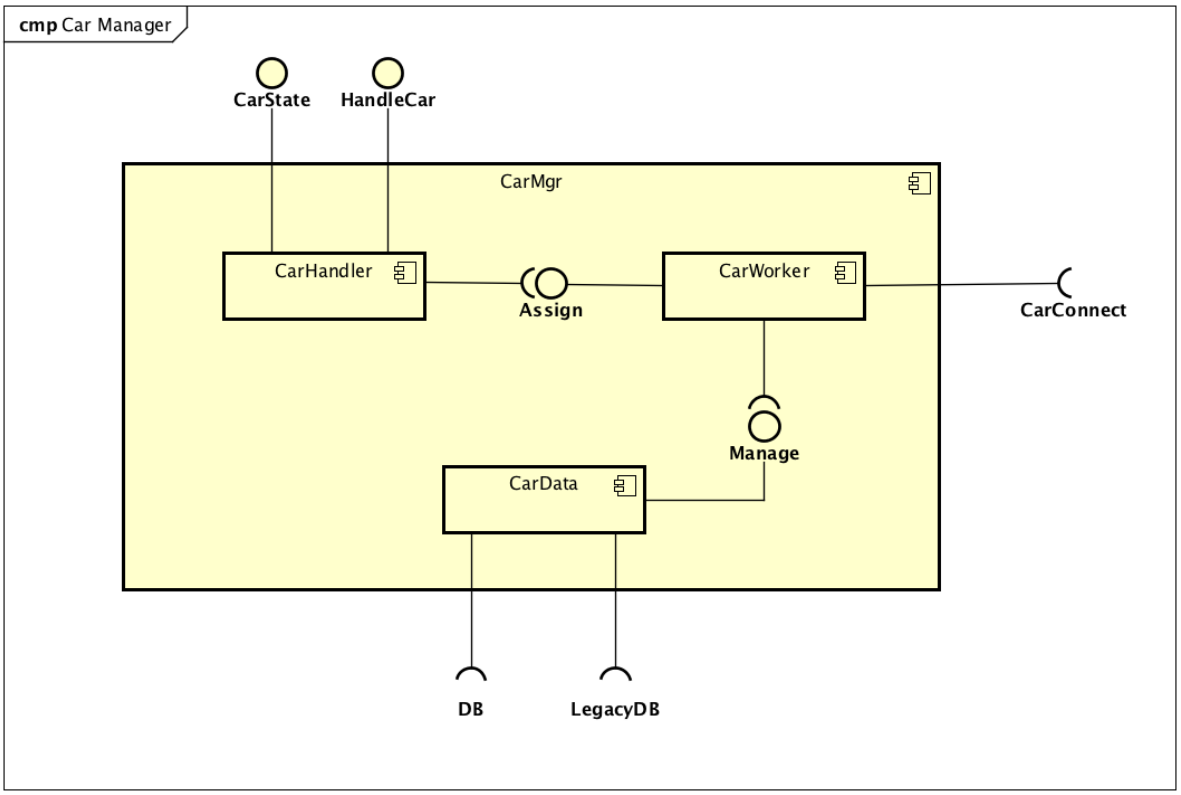

| | |
|---|---|
| **Test case ID** | I3 |
| **Test item(s)** | Request Handler -> Request Worker |
| **Input specification** | Create typical Request Handler Input |
| **Output specification** | Check if the correct methods are called in the Ride Worker |
| **Environmental needs** | • Request Manager driver<br>• Ride Manager driver<br>• Payment Manager stub<br>• Car Manager stub<br>• DB |

### 3.1.4. Integration Test: Ride Manager Sub-System




| | |
|---|---|
| **Test case ID** | I4 |
| **Test item(s)** | Ride Handler -> Ride Worker |
| **Input specification** | Create typical Ride Handler input |
| **Output specification** | Check if the correct functions are called in the Ride Worker |
| **Environmental needs** | • Request Manager driver<br>• Car driver<br>• Area Manager stub<br>• Payment Manager stub<br>• Car Manager stub<br>• DB |

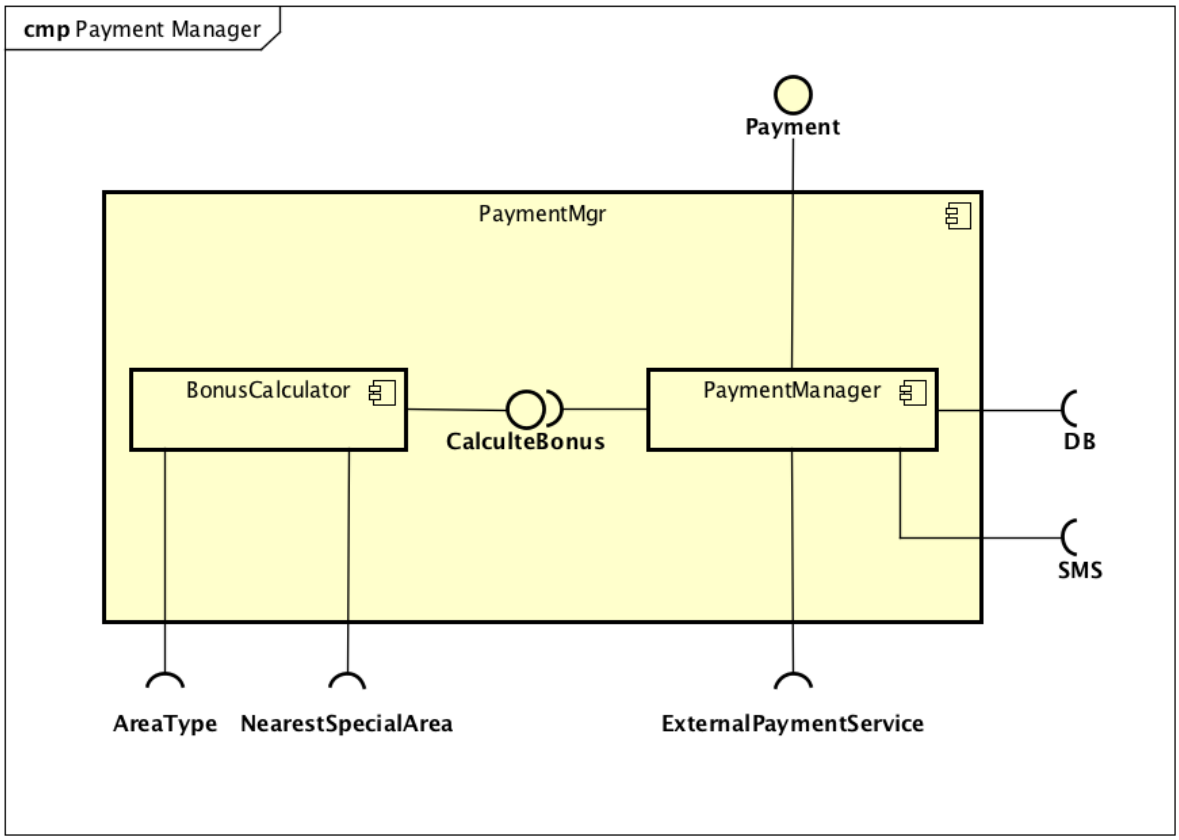
### 3.1.5. Integration Test: Car Manager Sub-System




| **Test case ID** | I5 |
|---|---|
| **Test item(s)** | Car Handler -> Car Worker |
| **Input specification** | Create typical Car Handler input |
| **Output specification** | Check if the correct methods are called in the Car Worker |
| **Environmental needs** | • Reservation Manager driver<br>• Ride Manager driver<br>• Car stub<br>• DB |

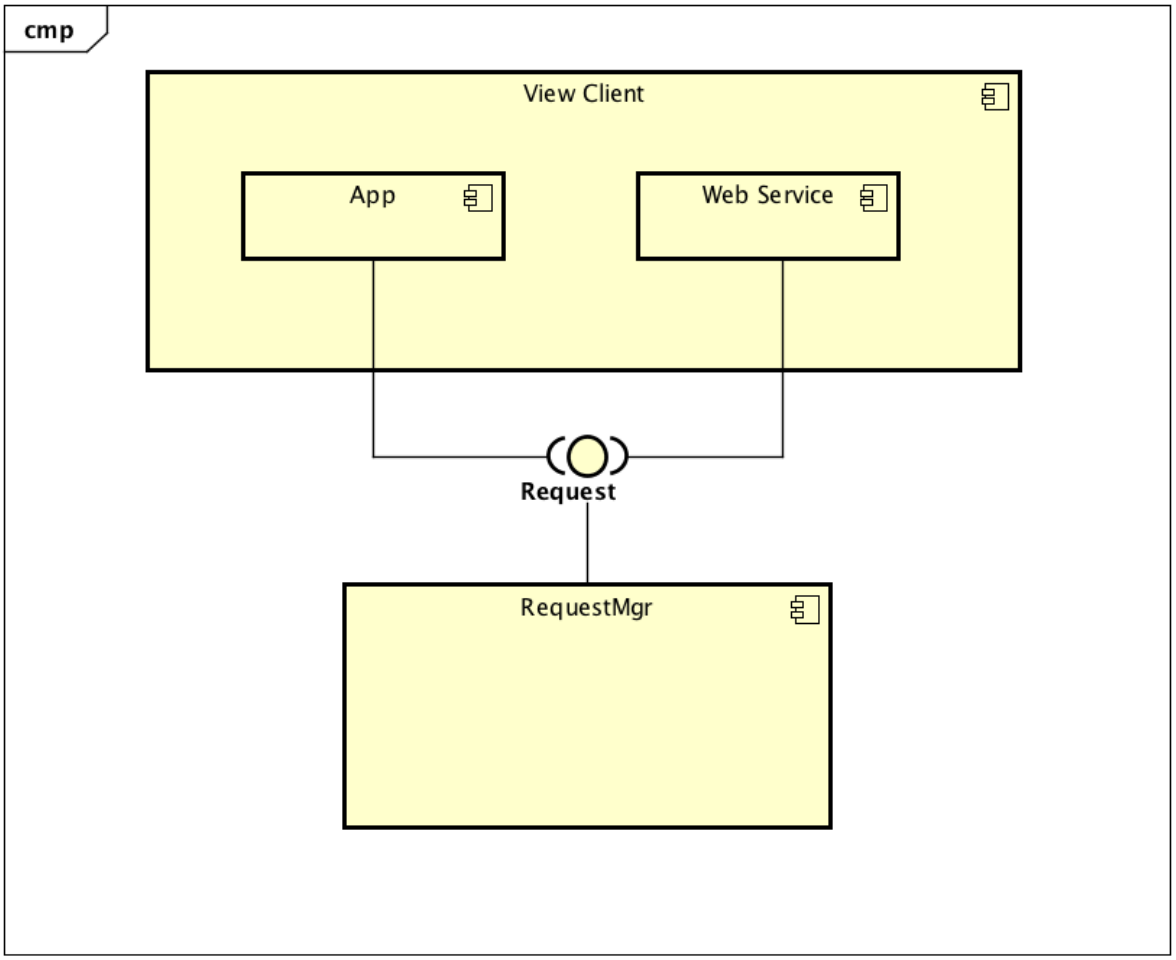
| **Test case ID** | I6 |
|---|---|
| **Test item(s)** | Car Worker -> Car Data |
| **Input specification** | Create typical Car Worker input |
| **Output specification** | Check if the correct methods are called in the Car Data |
| **Environmental needs** | • I5 succeeded<br>• DB<br>• Legacy DB |

### 3.1.6. Integration Test: Payment Manager Sub-System



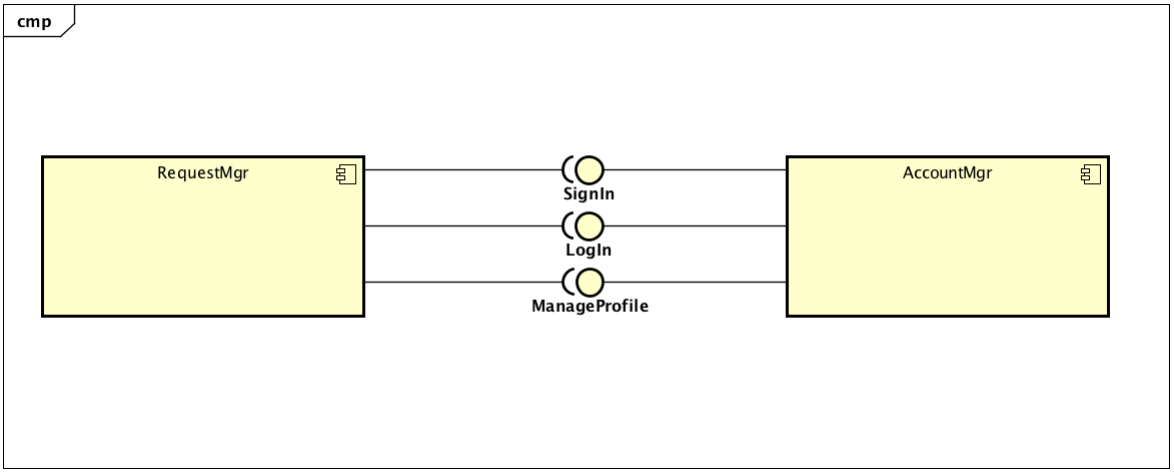

| | |
|---|---|
| **Test case ID** | I7 |
| **Test item(s)** | Payment Manager -> Bonus Calculator |
| **Input specification** | Create typical Payment Manager input |
| **Output specification** | Check if the correct methods are called in the Bonus Calculator |
| **Environmental needs** | • Ride Manager driver<br>• Reservation Manager driver<br>• Area Manager stub<br>• SMS service stub<br>• External Payment service stub<br>• DB |

### 3.1.7. Integration Test: Client and Request Manager



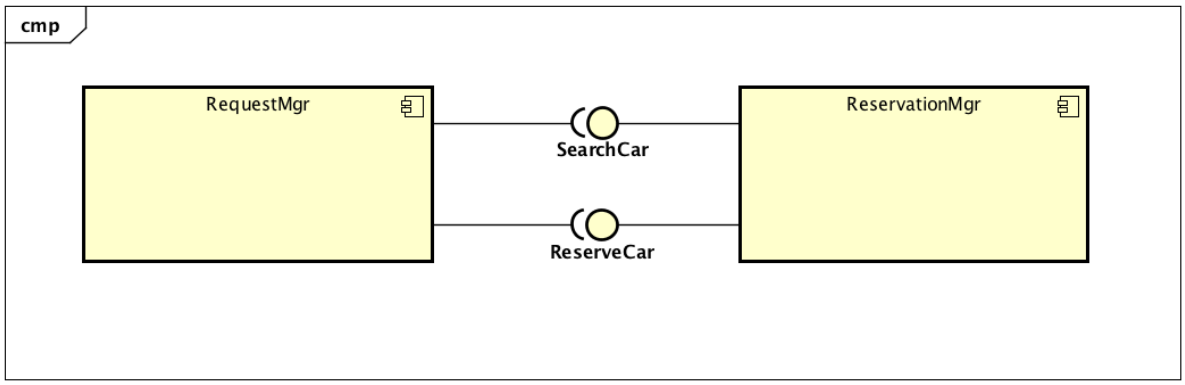

| | |
|---|---|
| **Test case ID** | I8 |
| **Test item(s)** | View Client -> Request Manager |
| **Input specification** | Create typical Client input |
| **Output specification** | Check if the correct functions are called in the Request Manager |
| **Environmental needs** | • I1 succeeded<br>• Account Manager stub<br>• Ride Manager stub<br>• DB |

### 3.1.8. Integration Test: Request Manager and Account Manager



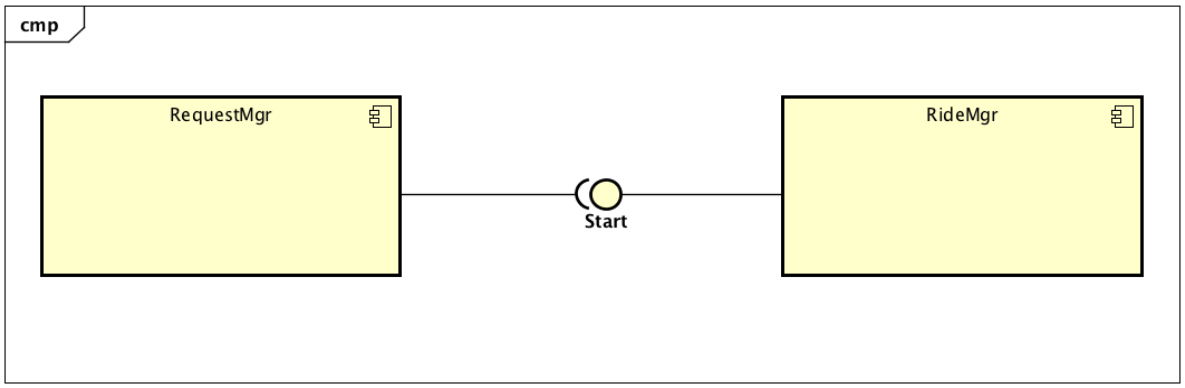

| | |
|---|---|
| **Test case ID** | I9 |
| **Test item(s)** | Request Manager -> Account Manager |
| **Input specification** | Create typical Request Manager input |
| **Output specification** | Check if the correct functions are called in the Account Manager |
| **Environmental needs** | • I1 succeeded<br>• I2 succeeded<br>• SMS service stub<br>• E-Mail service stub<br>• DB |

### 3.1.9. Integration Test: Request Manager and Reservation Manager



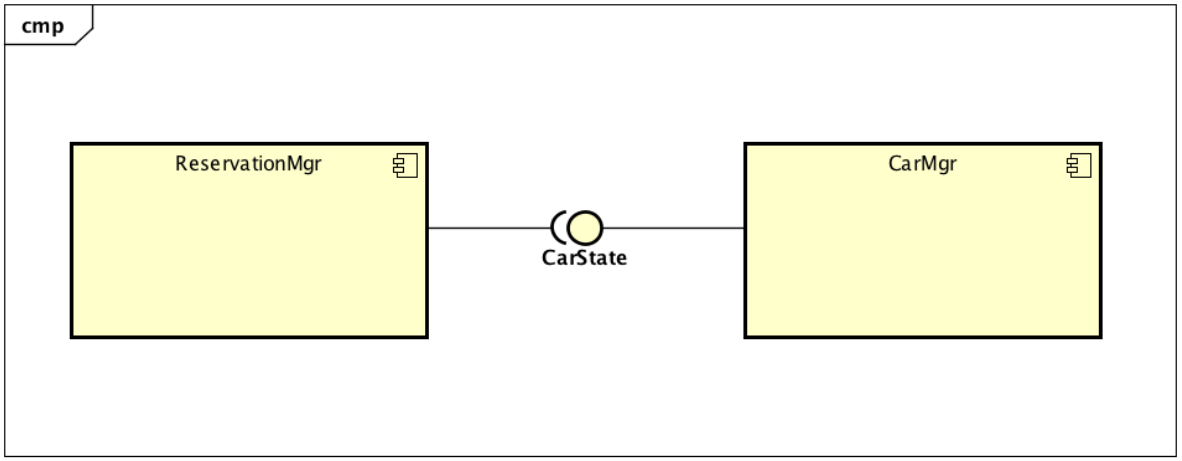

| | |
|---|---|
| **Test case ID** | I10 |
| **Test item(s)** | Request Manager -> Reservation Manager |
| **Input specification** | Create typical Request Manager input |
| **Output specification** | Check if the correct functions are called in the Reservation Manager |
| **Environmental needs** | • I1 succeeded<br>• I3 succeeded<br>• Car Manager stub<br>• DB |

### 3.1.10. Integration Test: Request Manager and Ride Manager



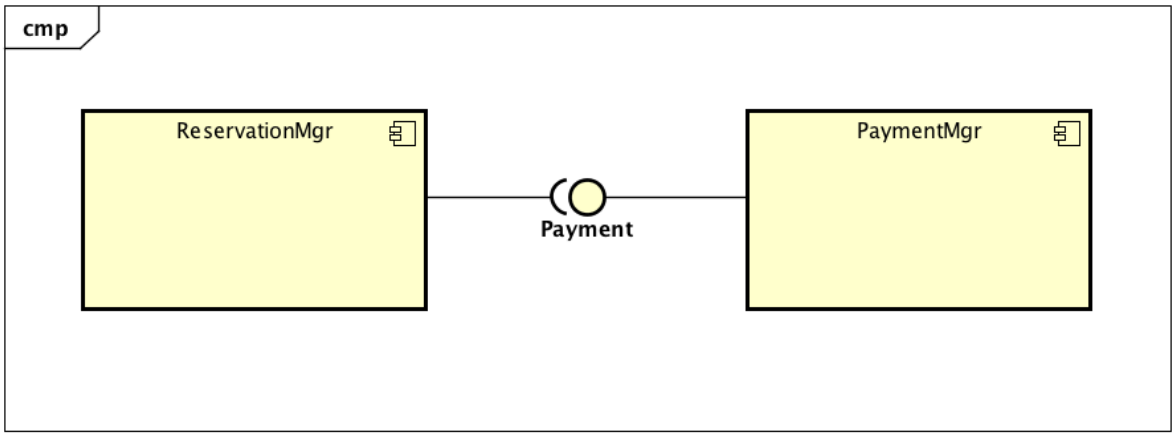

| | |
|---|---|
| **Test case ID** | I11 |
| **Test item(s)** | Request Manager -> Ride Manager |
| **Input specification** | Create typical Request Manager input |
| **Output specification** | Check if the correct functions are called in the Ride Manager |
| **Environmental needs** | • I1 succeeded<br>• I4 succeeded<br>• Car Manager stub<br>• Reservation Manager stub<br>• Payment Manager stub<br>• Area Manager stub<br>• DB |

### 3.1.11. Integration Test: Reservation Manager and Car Manager



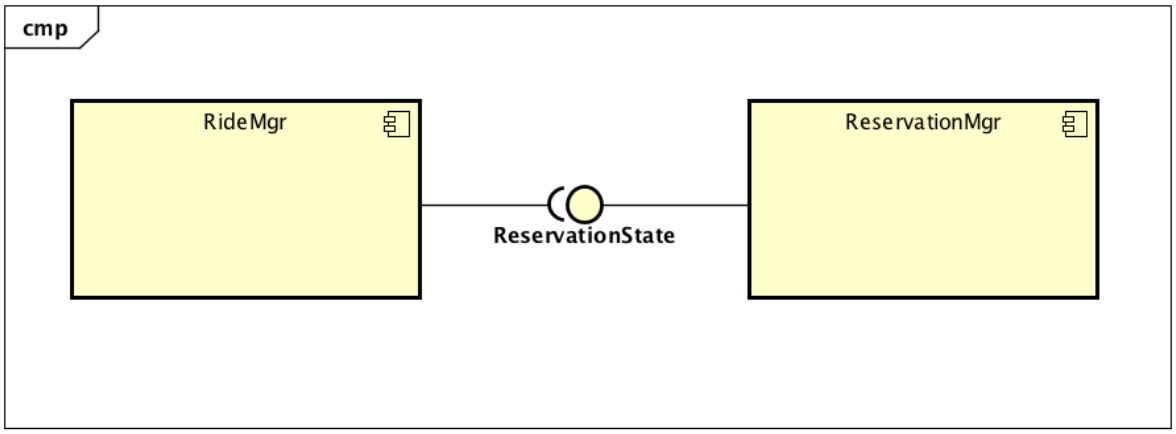

| | |
|---|---|
| **Test case ID** | I12 |
| **Test item(s)** | Reservation Manager -> Car Manager |
| **Input specification** | Create typical Reservation Manager input |
| **Output specification** | Check if the correct functions are called in the Car Manager |
| **Environmental needs** | • I3 succeeded<br>• I5 succeeded<br>• I6 succeeded<br>• Car stub<br>• DB<br>• Legacy DB |

### 3.1.12. Integration Test: Reservation Manager and Payment Manager



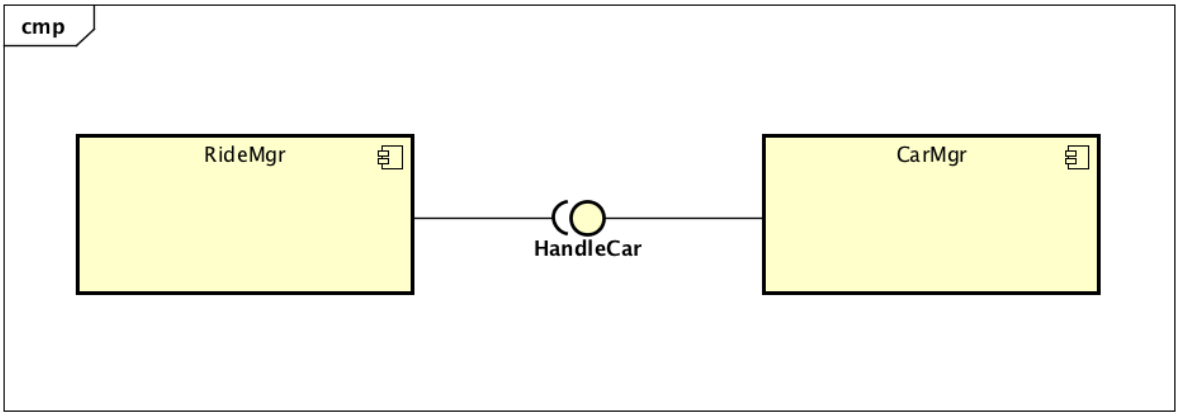

| | |
|---|---|
| **Test case ID** | I13 |
| **Test item(s)** | Reservation Manager -> Payment Manager |
| **Input specification** | Create typical Reservation Manager input |
| **Output specification** | Check if the correct functions are called in the Payment Manager |
| **Environmental needs** | • I3 succeeded<br>• I7 succeeded<br>• Area Manager stub<br>• SMS service stub<br>• External Payment service stub<br>• DB |

### 3.1.13. Integration Test: Ride Manager and Reservation Manager



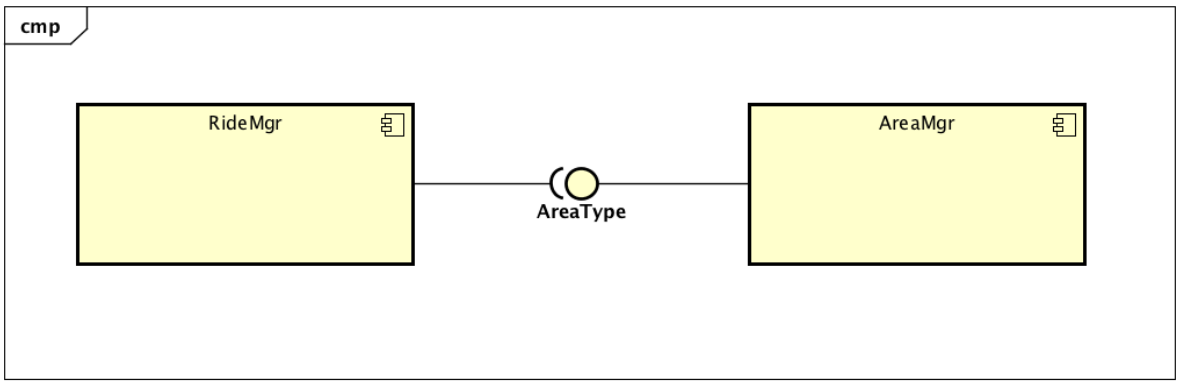

| | |
|---|---|
| **Test case ID** | I14 |
| **Test item(s)** | Ride Manager -> Reservation Manager |
| **Input specification** | Create typical Ride Manager input |
| **Output specification** | Check if the correct functions are called in the Reservation Manager |
| **Environmental needs** | • I3 succeeded<br>• I4 succeeded<br>• Car Manager stub<br>• DB |

### 3.1.14. Integration Test: Ride Manager and Car Manager



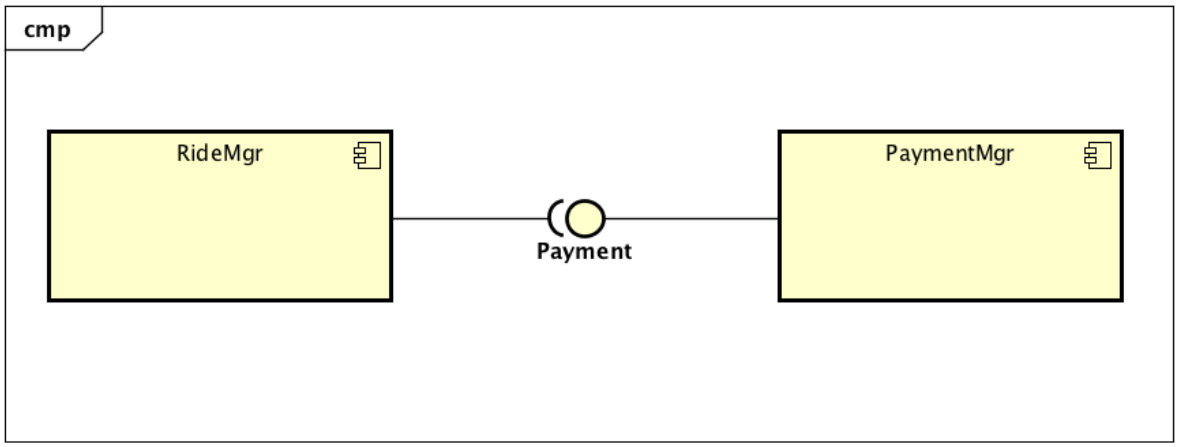

| | |
|---|---|
| **Test case ID** | I15 |
| **Test item(s)** | Ride Manager -> Car Manager |
| **Input specification** | Create typical Ride Manager input |
| **Output specification** | Check if the correct functions are called in the Car Manager |
| **Environmental needs** | • I4 succeeded<br>• I5 succeeded<br>• I6 succeeded<br>• Car stub<br>• DB<br>• Legacy DB |

### 3.1.15. Integration Test: Ride Manager and Area Manager



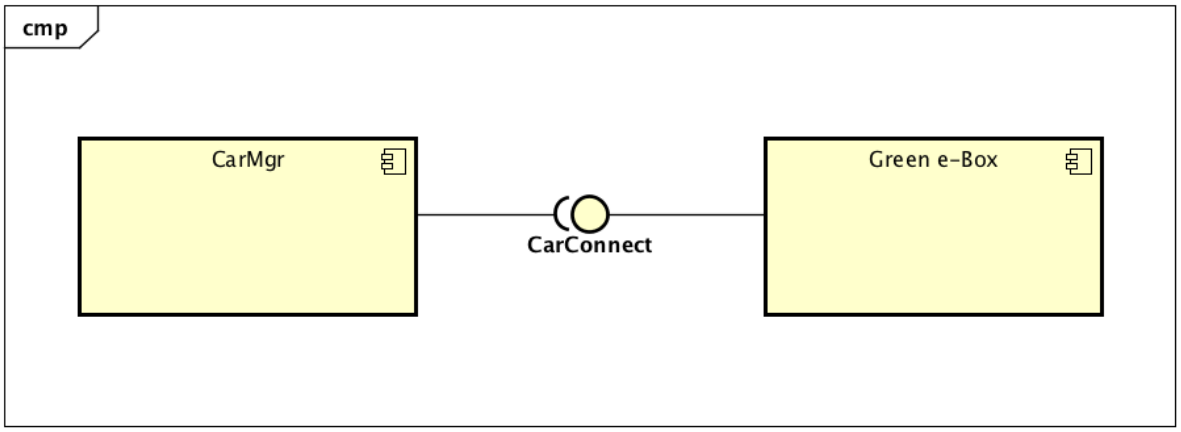

| | |
|---|---|
| **Test case ID** | I16 |
| **Test item(s)** | Ride Manager -> Area Manager |
| **Input specification** | Create typical Ride Manager input |
| **Output specification** | Check if the correct functions are called in the Area Manager |
| **Environmental needs** | • I4 succeeded<br>• DB |

### 3.1.16. Integration Test: Ride Manager and Payment Manager



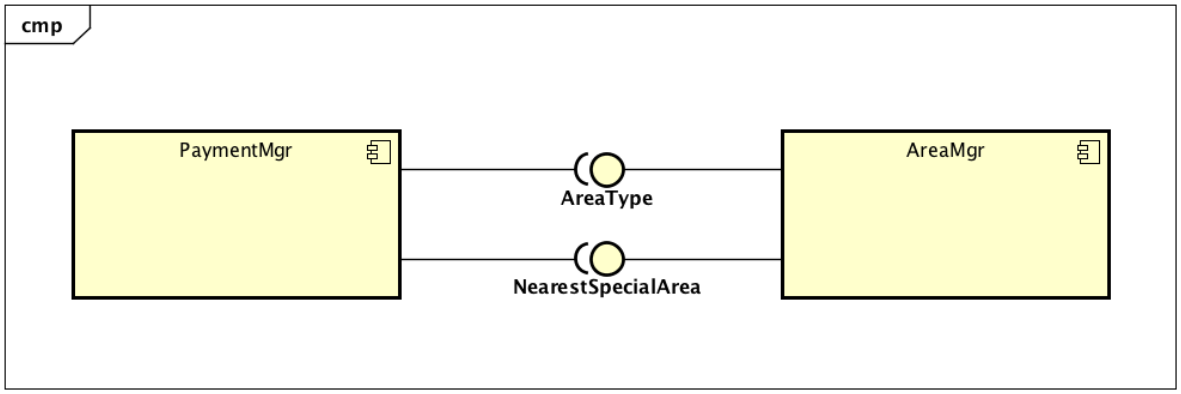

| | |
|---|---|
| **Test case ID** | I17 |
| **Test item(s)** | Reservation Manager -> Payment Manager |
| **Input specification** | Create typical Ride Manager input |
| **Output specification** | Check if the correct functions are called in the Payment Manager |
| **Environmental needs** | • I4 succeeded<br>• I7 succeeded<br>• Area Manager stub<br>• SMS service stub<br>• External Payment service stub<br>• DB |

### 3.1.17. Integration Test: Car Manager and Car System



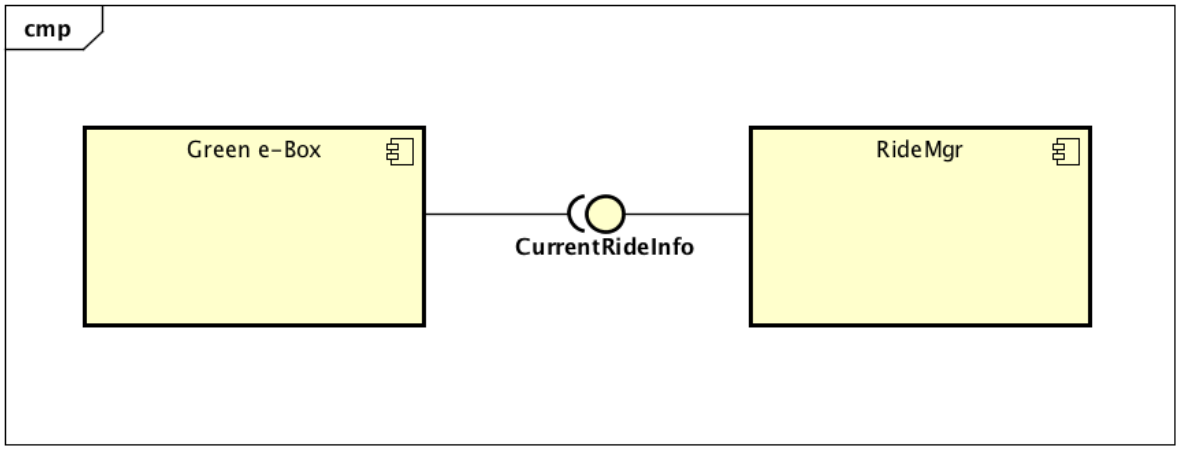

| | |
|---|---|
| **Test case ID** | I18 |
| **Test item(s)** | Car Manager -> Green e-Box (Car) |
| **Input specification** | Create typical Car Manager input |
| **Output specification** | Check if the correct functions are called in the Green e-Box |
| **Environmental needs** | • I5 succeeded<br>• I6 succeeded<br>• Car Display stub<br>• Car CU stub<br>• Area Manager stub<br>• Ride Manager stub |

### 3.1.18. Integration Test: Payment Manager and Area Manager



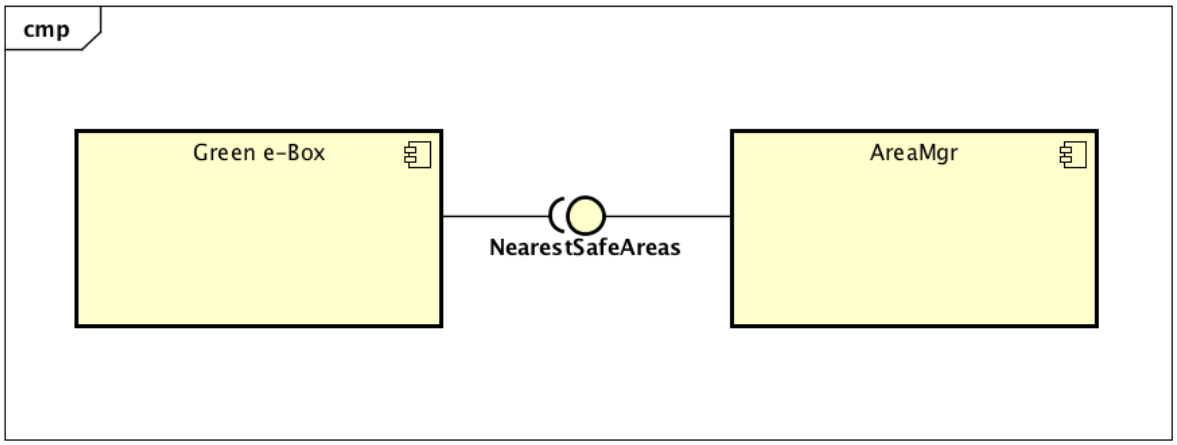

| | |
|---|---|
| **Test case ID** | I19 |
| **Test item(s)** | Payment Manager -> Area Manager |
| **Input specification** | Create typical Payment Manager input |
| **Output specification** | Check if the correct functions are called in the Area Manager |
| **Environmental needs** | • I7 succeeded<br>• DB |

### 3.1.19. Integration Test: Car System and Ride Manager




| | |
|---|---|
| **Test case ID** | I20 |
| **Test item(s)** | Green e-Box (Car) -> Ride Manager |
| **Input specification** | Create typical Green e-Box input |
| **Output specification** | Check if the correct functions are called in the Ride Manager |
| **Environmental needs** | • I4 succeeded<br>• DB |

### 3.1.20. Integration Test: Car System and Area Manager




| | |
|---|---|
| **Test case ID** | I21 |
| **Test item(s)** | Green e-Box (Car) -> Area Manager |
| **Input specification** | Create typical Green e-Box input |
| **Output specification** | Check if the correct functions are called in the Area Manager |
| **Environmental needs** | • DB |

## 3.2. Test Procedures

### 3.2.1. Test procedure: Request Manager Sub-System

| **Identifier** | TP1 |
|---|---|
| **Purpose** | This test must verify that Request manager:<br>• Can handle client request<br>• Can handle user request<br>• Can choose the correct component to delegate the incoming request<br>• Can send back the result of the request |
| **Procedure steps** | Execute I1 |

### 3.2.2. Test procedure: Account Manager Sub-System

| **Identifier** | TP2 |
|---|---|
| **Purpose** | This test must verify that Account Manager:<br>• Can add a new account<br>• Can check if incoming login credentials are correct<br>• Can create a user's session<br>• Can modify a profile if the new values are adequate<br>• Can send back to Request Manager the result of the request |
| **Procedure steps** | Execute I2 |

### 3.2.3. Test procedure: Reservation Manager Sub-System

| **Identifier** | TP3 |
|---|---|
| **Purpose** | This test must verify that Reservation Manager:<br>• Can handle the incoming request<br>• Can allow the research/reserve of a car<br>• Can create a list of available cars in according with the user's parameters<br>• Can create a worker (thread) to manage the reservation<br>• Can allow only one reservation for the same user at the same time<br>• Can send back to Request Manager the result of the request |
| **Procedure steps** | Execute I3 |

### 3.2.4. Test procedure: Ride Manager Sub-System

| **Identifier** | TP4 |
|---|---|
| **Purpose** | This test must verify that Ride Manager:<br>• Can handle the incoming request<br>• Can create a worker (thread) to manage the ride<br>• Can provide the current ride information<br>• Can start the ride when unlock request is received<br>• Can send back to Request Manager the result of the request |
| **Procedure steps** | Execute I4 |

### 3.2.5. Test procedure: Car Manager Sub-System

| **Identifier** | TP5 |
|---|---|
| **Purpose** | This test must verify that Car Manager:<br>• Can handle the incoming request<br>• Can create a worker (thread) for every car to manage the latter<br>• Can assign the incoming request to the correct worker<br>• Can retrieve information about every car<br>• Can send back the result of the request |
| **Procedure steps** | Execute I6 after I5 execution |

### 3.2.6. Test procedure: Payment Manager Sub-System

| **Identifier** | TP6 |
|---|---|
| **Purpose** | This test must verify that Payment Manager:<br>• Can handle the incoming request<br>• Can retrieve information about the position of the car<br>• Can calculate bonus about the specific ride<br>• Can apply the payment using the external payment service<br>• Can send back the result of the operation |
| **Procedure steps** | Execute I7 |

### 3.2.7. Test procedure: View Client interaction

| **Identifier** | TP7 |
|---|---|
| **Purpose** | This test must verify that View Client:<br>• Can send request from mobile App to Request Manager<br>• Can send request from Web Service to Request Manager<br>• Can receive result from Request Manager to the correct interface about its request |
| **Procedure steps** | Execute I8 |

### 3.2.8. Test procedure: Request Manager interaction

| **Identifier** | TP8 |
|---|---|
| **Purpose** | This test must verify that Request Manager:<br>• Can use Account Manager for the SignIn/ManageProfile/Login request received from the View Client<br>• Can use Reservation Manager for the SearchCar/ReserveCar request received from the View Client<br>• Can use Ride Manager for the Start request received from the View Client<br>• Can send back to View Client the result of the request |
| **Procedure steps** | Execute I9, I10, I11 |

### 3.2.9. Test procedure: Reservation Manager interaction

| **Identifier** | TP9 |
|---|---|
| **Purpose** | This test must verify that Reservation Manager:<br>• Can use Car Manager to know/change the state of a car<br>• Can user Payment Manager to apply the fee |
| **Procedure steps** | Execute I12, I13 |

### 3.2.10. Test procedure: Ride Manager interaction

| | |
|---|---|
| **Identifier** | TP10 |
| **Purpose** | This test must verify that Ride Manager:<br>• Can use Reservation Manager to check/change the state of a reservation<br>• Can use Car Manager to manage the information of a car<br>• Can use Area Manager to check in which area a car is located<br>• Can use Payment Manager to apply the payment for the ride |
| **Procedure steps** | Execute I14, I15, I16, I17 |

### 3.2.11. Test procedure: Car Manager interaction

| | |
|---|---|
| **Identifier** | TP11 |
| **Purpose** | This test must verify that Car Manager:<br>• Can access to Car (Green e-Box) if it's necessary to retrieve information about a car |
| **Procedure steps** | Execute I18 |

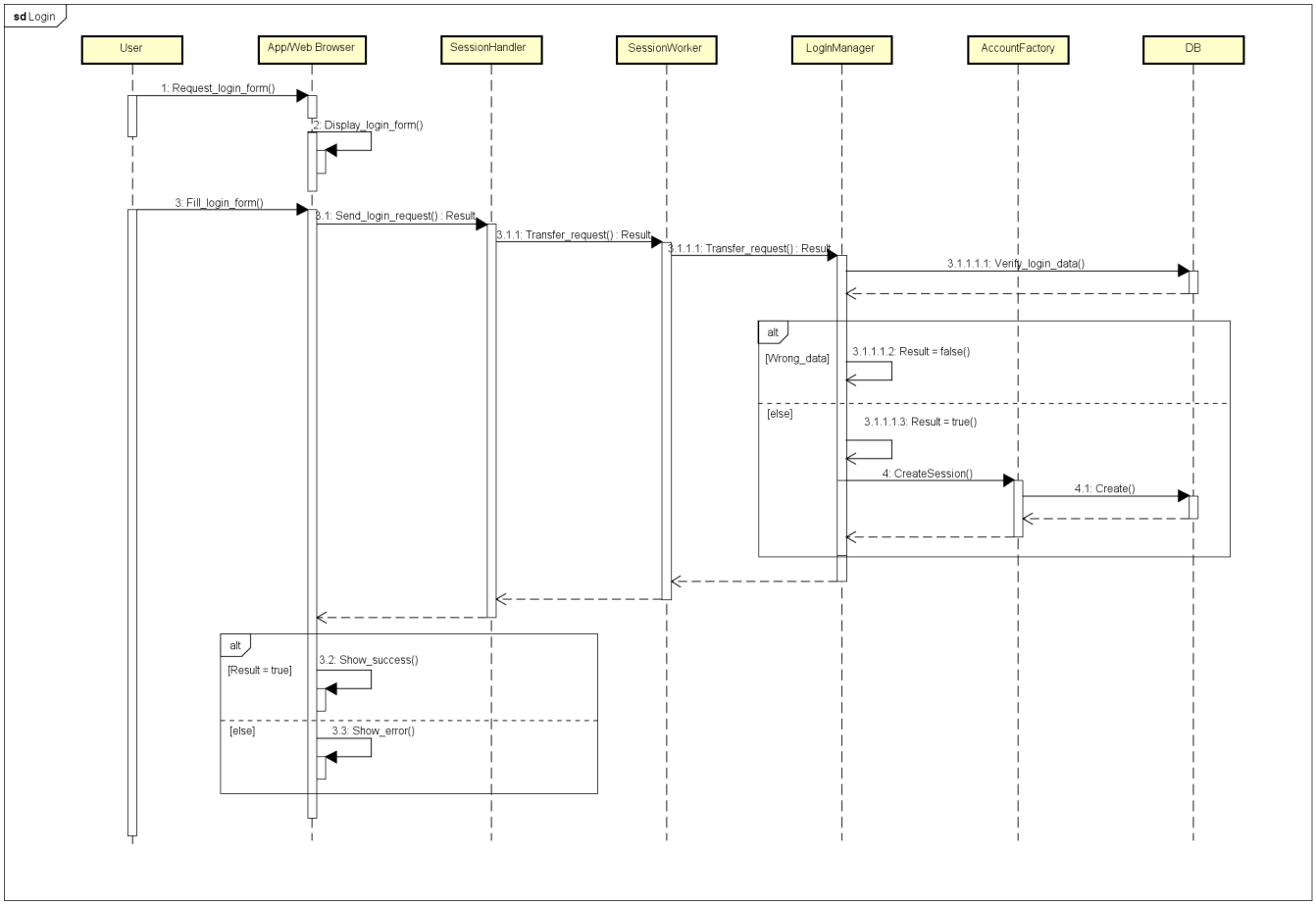
### 3.2.12. Test procedure: Payment Manager interaction

| | |
|---|---|
| **Identifier** | TP12 |
| **Purpose** | This test must verify that Payment Manager:<br>• Can use Area Manager to check in which area the is located<br>• Can send to the External Payment Service the data for the payment |
| **Procedure steps** | Execute I19 |

### 3.2.13. Test procedure: Car (Green e-box) interaction

| | |
|---|---|
| **Identifier** | TP13 |
| **Purpose** | This test must verify that Car:<br>• Can use Ride Manager to retrieve information about the ride<br>• Can use Area Manager to know the safe areas near the current position of the car |
| **Procedure steps** | Execute I20, I21 |

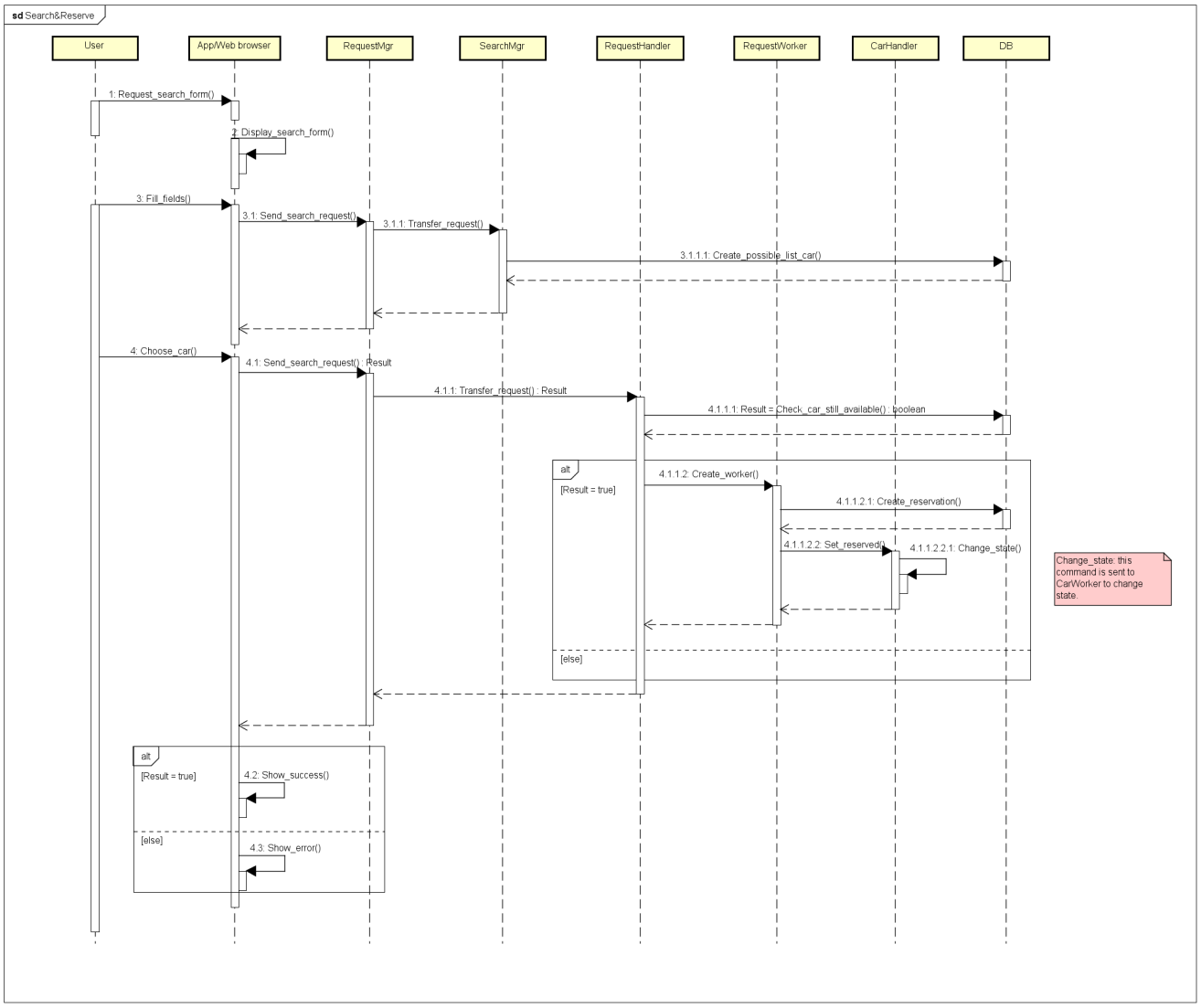
### 3.2.14. Test procedure: Login request




| **Identifier** | TP14 |
|---|---|
| **Purpose** | This test must verify that all the necessary components for the Login operation are, when the provided credential are correct, correctly integrated in order to reach a correct result of the Login procedure. |
| **Procedure steps** | Execute in order: I8, I1, I9, I2 |

| **Identifier** | TP15 |
|---|---|
| **Purpose** | This test must verify that all the necessary components for the Login operation are, when the provided credential aren't correct, correctly integrated in order to reach a correct result of the Login procedure. |
| **Procedure steps** | Execute in order: I8, I1, I9 |

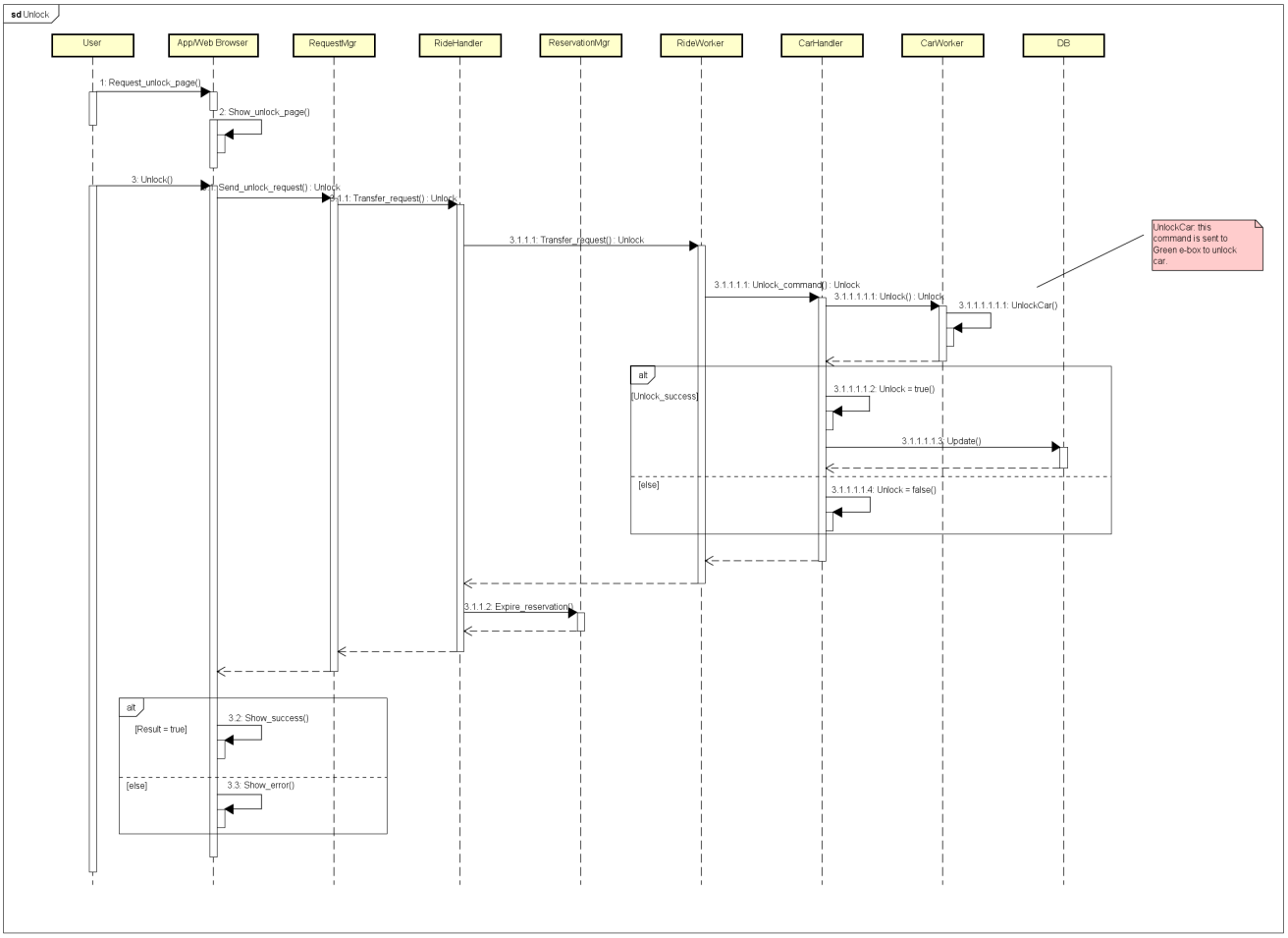
### 3.2.15. Test procedure: Search & Reserve request




| **Identifier** | TP16 |
|---|---|
| **Purpose** | This test must verify that all necessary components for the Search and Reserve operation are correctly integrated, when the car selected from the user is still available at the reservation instant, in order to reach a correct result of the procedure. |
| **Procedure steps** | Execute in order: I8, I10, I3, I12, I5 |

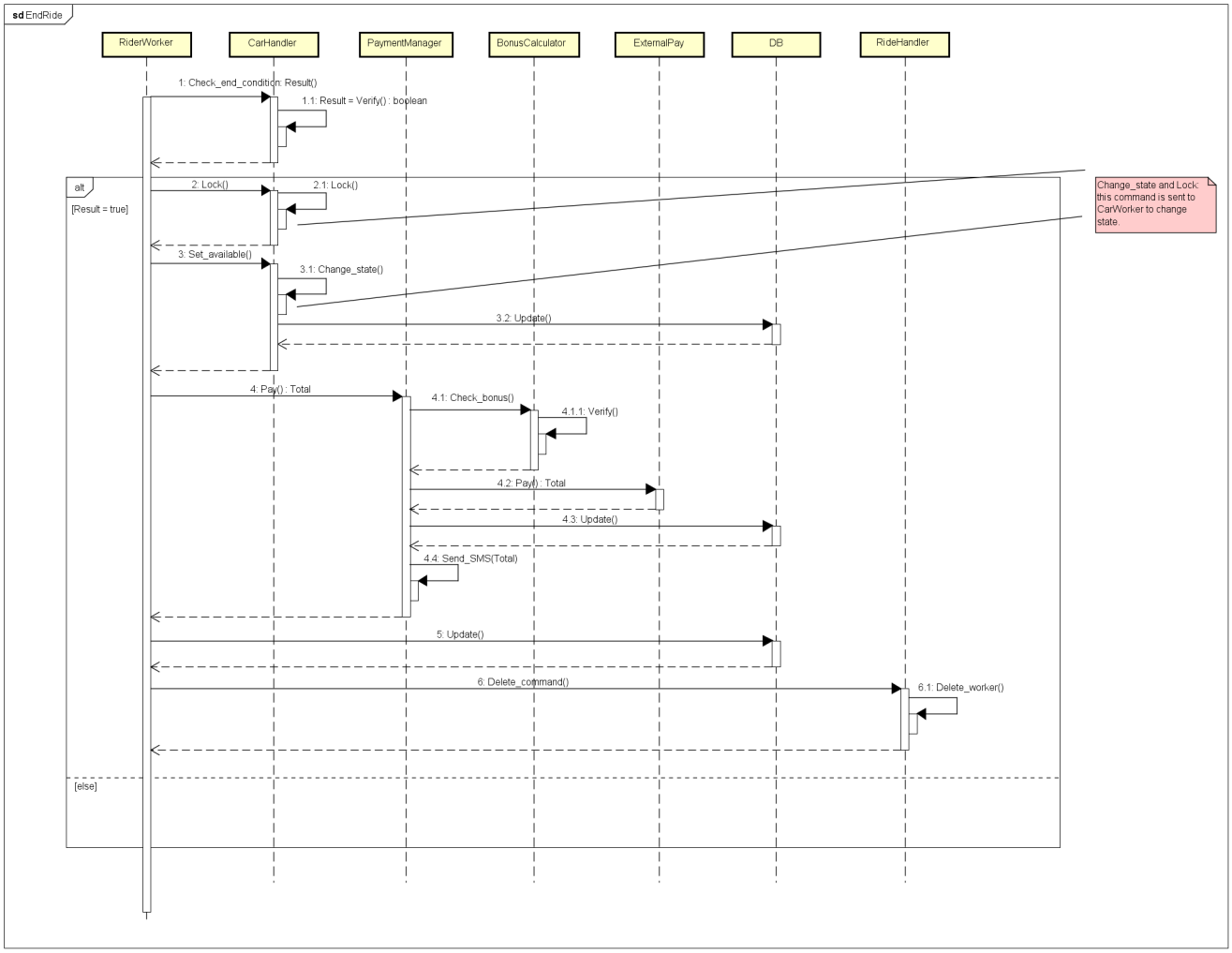
| **Identifier** | TP17 |
|---|---|
| **Purpose** | This test must verify that all necessary components for the Search and Reserve operation are correctly integrated, when the car selected from the user is no longer available at the reservation instant, in order to reach a correct result of the procedure. |
| **Procedure steps** | Execute in order: I8, I10 |

## 3.2.16. Test procedure: Unlock request




| | |
|---|---|
| **Identifier** | TP18 |
| **Purpose** | This test must verify that all necessary components for the Unlock operation are correctly integrated in order to reach a correct result of the procedure. |
| **Procedure steps** | Execute in order: I8, I11, I4, I15, I5, I18, I14 |

## 3.2.17. Test procedure: End Ride request




| | |
|---|---|
| **Identifier** | TP19 |
| **Purpose** | This test must verify that all necessary components for the End Ride operation are correctly integrated, when the end condition are verified, in order to reach a correct result of the procedure. |
| **Procedure steps** | Execute in order: I15, I5, I17, I7 |

# 4. Tools and Test Equipment Required

To perform in a reliable way the integration testing, the following tools and testing environments are needed. To accomplish the integration testing at sub-component level (e.g. integration of the components in the Account Manager) we are going to use the integration framework of Mockito's library. We made this choice because this tool offers the possibility to implement the stubs and the drivers needed for the testing. We are going to use Mockito also in the integration testing of mobile clients.

To perform the integration testing at component level (e.g. integration of Ride Manager with Car Manager) we are going to use the Arquillian framework because we have several runtime units that are going to run in different software containers inside the same virtual machine and this framework provides the tools to test software containers integration. We take in consideration manual testing to test the integration between the web client and the other components of the system.

The testing environment consists in a virtual machine for the application logic and another one for the DBMS running both Ubuntu Server 16.04 LTS. Concerning the client side, we need a mobile terminal running Android 5.0 (or above), another mobile terminal running iOS 10.0 (or above) and a desktop PC running Google Chrome.

# 5. Program Stubs and Data Required

## 5.1. Stubs

- Account Manager
- Reservation Manager
- Ride Manager
- Car Manager
- Area Manager
- Payment Manager
- Car (Green e-Box)

## 5.2. Drivers

- Client
- Car
- Request Manager
- Ride Manager
- Reservation Manager

## 5.3. Data required

To perform a correct integration testing it is necessary to implement a dummy DB and Legacy DB populated with the following data:

- Some registered user data
- A set of reservations for the cars
- A set of cars
- A set of area zones
- A set of rides

All this data elements must match together to perform correctly the integration testing. For more details refer to the RASD document specifications.

# 6. Appendix

## 6.1. References

Testing tool:

- Arquillian: http://arquillian.org/guides/
- Mockito: http://site.mockito.org

Materials from Wikipedia:

- Integration testing: https://en.wikipedia.org/wiki/Integration_testing
- Oracle: https://en.wikipedia.org/wiki/Oracle_(software_testing)
- Test stub: https://en.wikipedia.org/wiki/Test_stub
- Mock object: https://en.wikipedia.org/wiki/Mock_object
- Software testing: https://en.wikipedia.org/wiki/Software_testing

## 6.2. Software and Tools Used

- Microsoft Office Word: to redact and format this document.
- Astah Professional 7.1 (http://astah.net/editions/professional): to create all diagrams.

## 6.3. Effort Spent

- Simone Boglio: 20 hours.
- Lorenzo Croce: 16 hours.