POLITECNICO
MILANO 1863

Politecnico di Milano
2016-2017 Software Engineering 2 Project:
PowerEnJoy
Integration Testing Plan

Version 1.0

Alessandro Polenghi 879111

Alessandro Terragni 879112

1. Introduction

1.1 Purpose

This document describes the plans to test the integration of the software components of PowerEnjoy.
The purpose of this document is to test the interfaces between the components as described in the Design Document in order to fulfill the functional requirement of the RASD.

1.2 Scope

The system aims to support a car-sharing service that exclusively provide electrical cars.
PowerEnjoy is a service based on mobile application and has a unique target: the clients.
The system allows clients to reserve an electric car via the mobile app, using his GPS position or inserting his position manually to find an electrical car in the same zone.
The clients must be registered to the system to use it.
The registration process is very simple, it requires the credentials, the driving license and the payment data; the system is able to check if this data are valid and it allows the registration only for new users.

1.3 Definitions, acronyms, abbreviations

For the missing definitions, acronyms and abbreviations please refer to paragraph 1.4 of the RASD document and 1.3 of the DD document.

1.4 Reference documents

- Design Document (DD)
- Requirements and Analysis Specification Document (RASD)
- Project Plan (PP)

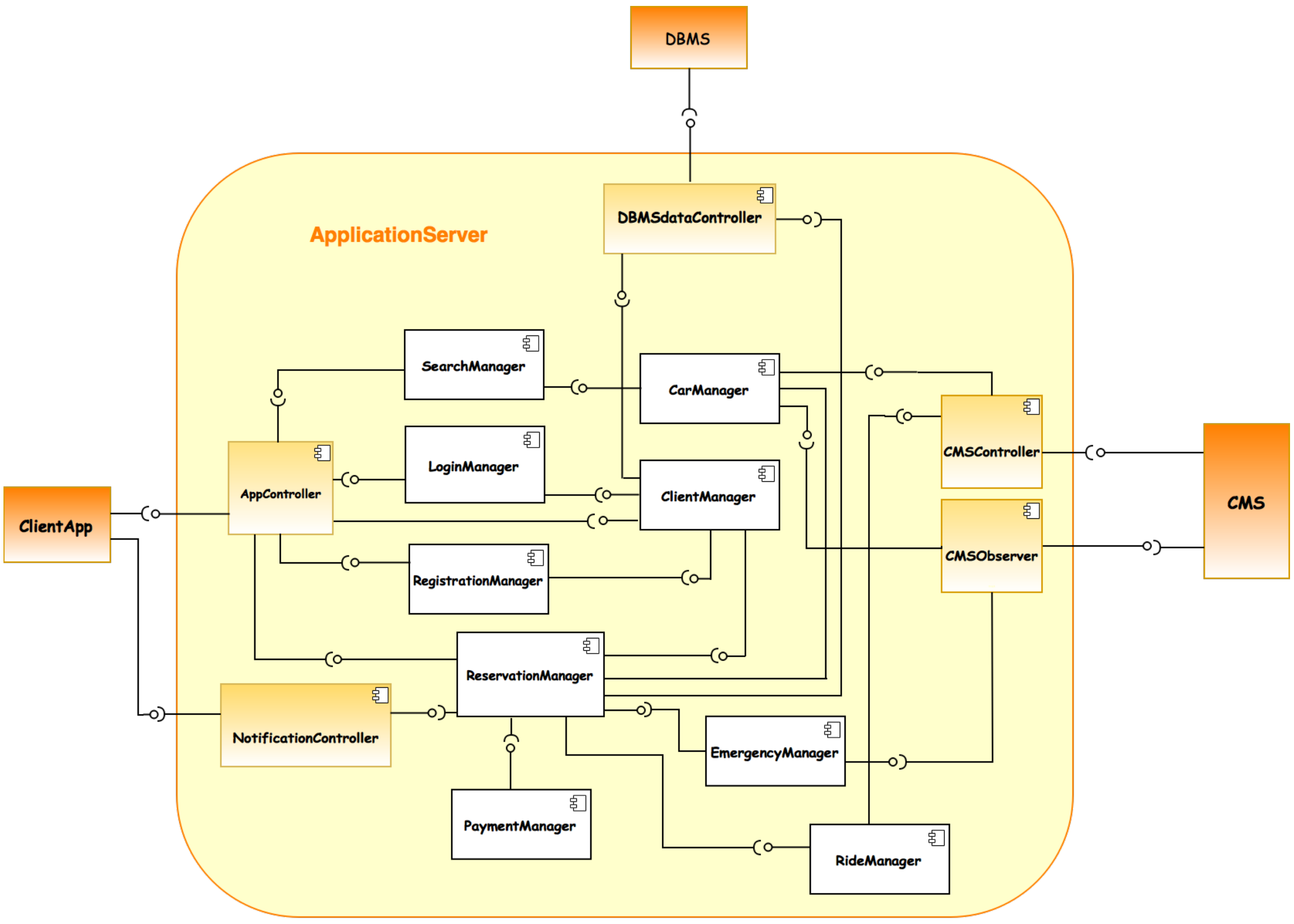
2. Integration Strategy

2.1 Entry criteria

All the involved components should have already been developed and tested individually using unit testing.
The missing ones can easily be replaced by drivers, as described in paragraph 5.

2.2 Elements to be integrated

All the components and the interactions involved in the integration tests, described in the following pages, refer to the ones depicted in the Design Document.



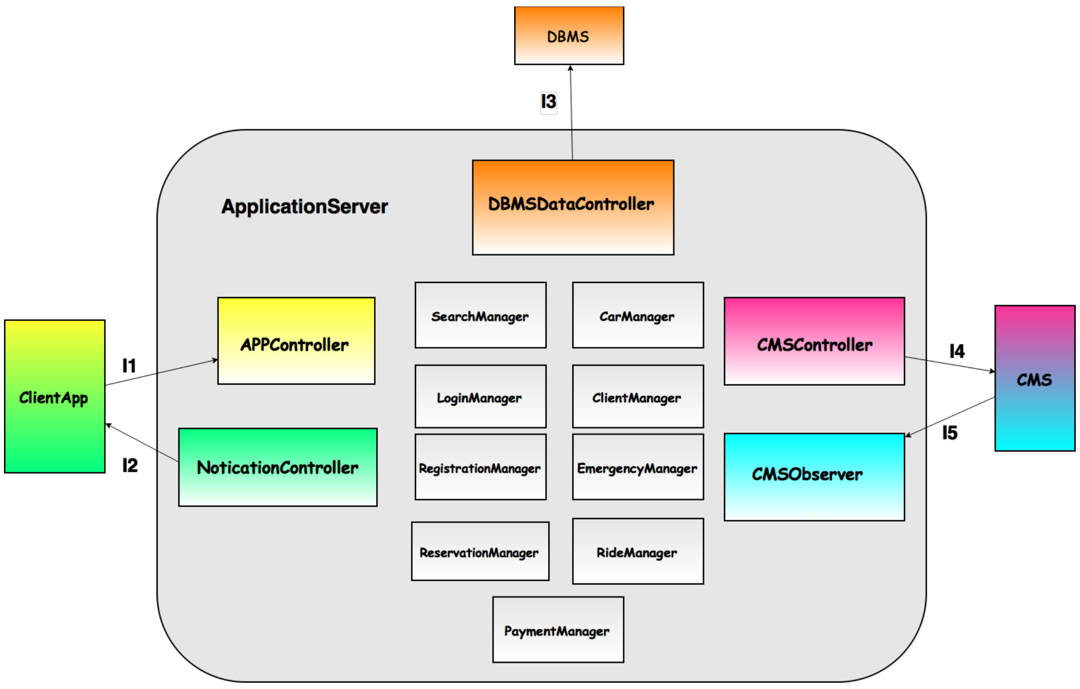
2.3 Integration Testing Strategy

The integration strategy we choose to drive our tests is a hybrid between the bottom-up approach and the Critical Modules approach: focusing on the Integration testing as a risk-reduction activity, in order to deliver any bad news as early as possible

2.4 Sequence of Component/Function Integration

First we decided to test what we consider the most critical parts of our system: the connections between the application server and the external components:

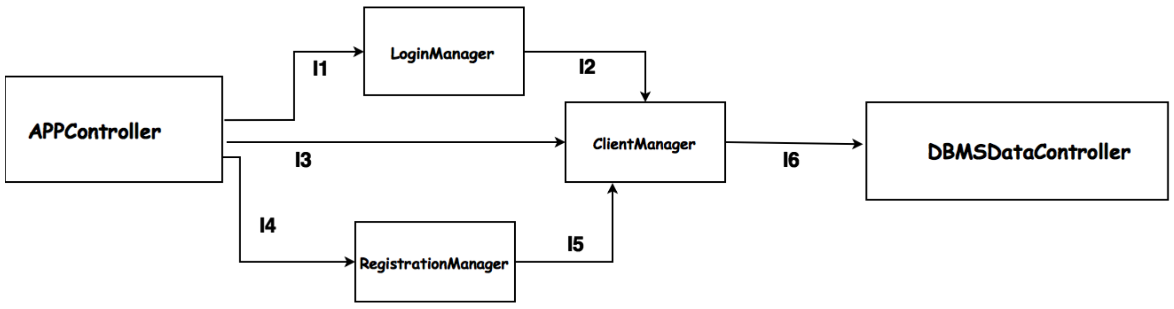
- **Subsystem 1: External interfaces**



- ➢ S1I1: ClientApp -> AppController
- ➢ S1I2: NotificationController -> ClientApp
- ➢ S1I3: DBMSDataController -> DBMS
- ➢ S1I4: CMSController -> CMS
- ➢ S1I5: CMS -> CMSObserver

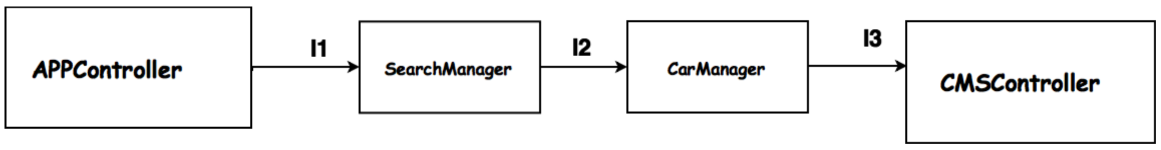
Then we start focusing on the ApplicationServer, dividing the test in subsystems, aggregating correlated functions, starting from the ones that have less component involved, in order to progressively simplify the testing process.

- **Subsystem 2: Login, Registration and edit account**



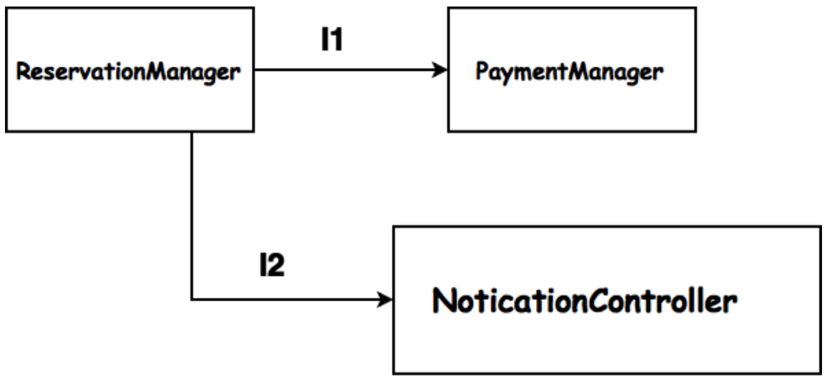
- ➢ S2I1: AppController -> LoginManager
- ➢ S2I2: LoginManager -> ClientManager
- ➢ S2I3: AppController -> ClientManager
- ➢ S2I4: AppController -> RegistrationManager
- ➢ S2I5: RegistrationManager -> ClientManager
- ➢ S2I6: ClientManager -> DBMSDataController

- **Subsystem 3: Search**



- ➢ S3I1: AppController -> SearchManager
- ➢ S3I2: SearchManager -> CarManager
- ➢ S3I3: CarManager -> CMSController

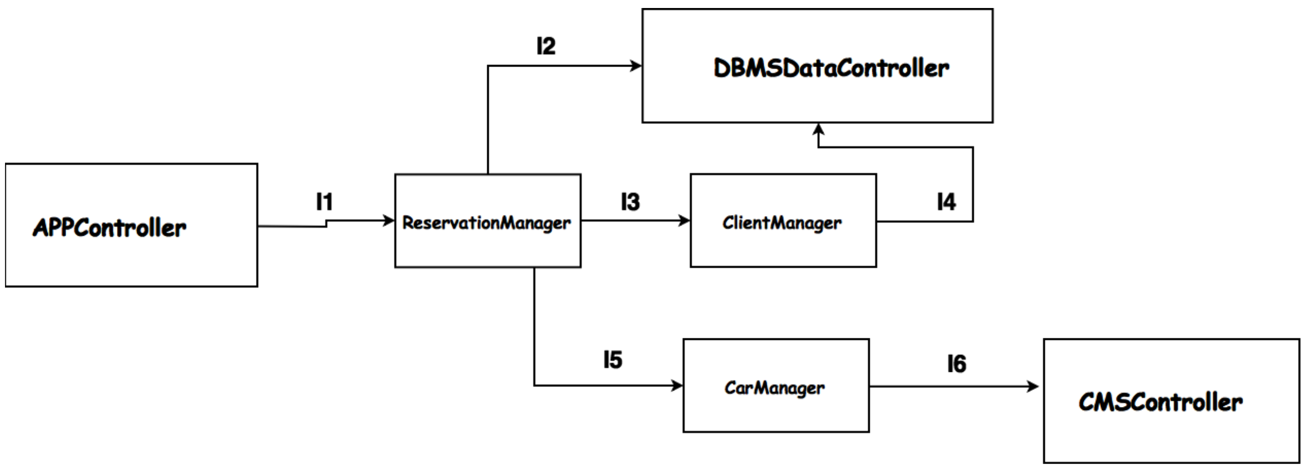
- **Subsystem 4: Payment and Notification**



➢ S4I1: ReservationManager -> PaymentManager

➢ S4I2: ReservationManager -> NotificationController

Subsystem 5: Reservation



➢ S5I1: AppController -> ReservationManager

➢ S5I2: ReservationManager -> DBMSDataController

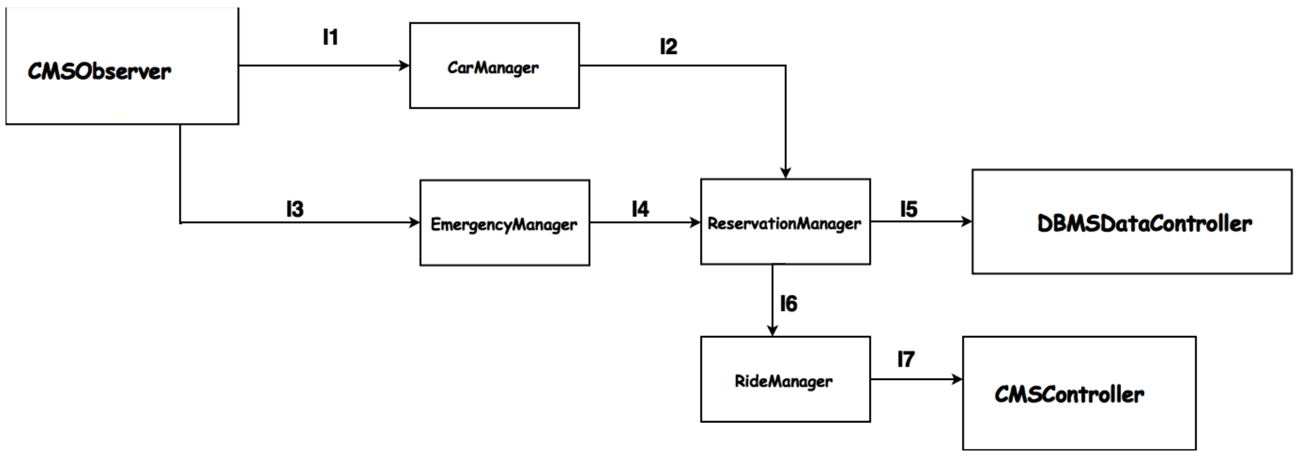
➢ S5I3: ReservationManager -> ClientManager

➢ S5I4: ClientManager -> DBMSDataController

➢ S5I5: ReservationManager -> CarManager

➢ S5I6: CarManager -> CMSController

- **Subsystem 6: Start, Ride, Park and Emergency**



- ➢ S6I1: CMSObserver -> CarManager
- ➢ S6I2: CarManager -> ReservationManager
- ➢ S6I3: CMSObserver -> EmergencyManager
- ➢ S6I4: EmergencyManager -> ReservationManager
- ➢ S6I5: ReservationManager -> DBMSDataController
- ➢ S6I6: ReservationManager -> RideManager
- ➢ S6I7: RideManager -> CMSController

3 Individual Steps and Test Description

3.1 Integration test Subsystem 1

3.1.1 Integration test S1I1

Test items	ClientApp -> AppController
Purprose	To test if AppController is able to handle requests coming from the ClientApp: • Registration • Login • Search a Car • Edit Account • Reserve a car
Input specifications	Create all possible requests that the ClientApp can send to the AppController
Output specifications	Check if the correct methods are called in the AppController
Environmental needs	-

3.1.2 Integration test S1I2

Test items	NotificationController-> ClientApp
Purprose	To test if ClientApp receives and shows correctly the notifications sent by the NotificationController
Input specifications	Create all possible types of notifications that the NotificationController can send
Output specifications	Check if the correct methods are called in the ClientApp
Environmental needs	-

3.1.3 Integration test S1I3

Test items	DBMSDataController -> DBMS
Purprose	To test if the DBMS correctly handles all the requests of the DBMSDataController: • Create data • Remove data • Modify data • Find Data
Input specifications	Create all possible types of request that the DBMSDataController can send
Output specifications	Check if the correct methods are called in the DBMS
Environmental needs	-

3.1.4 Integration test S1I4

Test items	CMSController -> CMS
Purprose	To test if the CMS correctly handles all the requests of the CMSController: • Create data • Remove data • Modify data • Find Data
Input specifications	Create all possible types of request that the CMSController can send
Output specifications	Check if the correct methods are called in the CMS
Environmental needs	-

3.1.5 Integration test S1I5

Test items	CMS -> CMSObserver
Purprose	To test if the CMSObserver correctly handles all the notification from the CMS
Input specifications	Create all possible types of notifications that the CMS can send
Output specifications	Check if the correct methods are called in the CMSObserver
Environmental needs	-

3.2 Integration test Subsystem 2

3.2.1 Integration test S2I1

Test items	AppController -> LoginManager
Purprose	To test if the LoginManager correctly handles all the possible types of requests from the AppController
Input specifications	Create all possible types of requests that the AppController can send to the LoginManager
Output specifications	Check if the correct methods are called in the LoginManager
Environmental needs	-

3.2.2 Integration test S2I2

Test items	LoginManager -> ClientManager
Purprose	To test if the ClientManager correctly handles all the possible types of requests from the LoginManager
Input specifications	Create some login requests
Output specifications	Check if the correct methods are called in the ClientManager
Environmental needs	S2I1 passed

3.2.3 Integration test S2I3

Test items	AppController -> ClientManager
Purprose	To test if the ClientManager correctly handles all the possible types of requests from the AppController: • change password • change email • change payment-information
Input specifications	Create all types of requests that the AppController can send to the ClientManager
Output specifications	Check if the correct methods are called in the ClientManager
Environmental needs	-

3.2.4 Integration test S2I4

Test items	AppController -> RegistrationManager
Purprose	To test if the ClientManager correctly handles the registration requests by the AppController
Input specifications	Create all types of requests that the AppController can send to the RegistrationManager
Output specifications	Check if the correct methods are called in the RegistrationManager
Environmental needs	-

3.2.5 Integration test S2I5

Test items	RegistrationManager -> ClientManager
Purprose	To test if the ClientManager correctly handles the registration requests by the RegistrationManager
Input specifications	Create all types of requests that the RegistrationManager can send to the ClientManager
Output specifications	Check if the correct methods are called in the ClientManager
Environmental needs	S2I4 passed

3.2.6 Integration test S2I6

Test items	ClientManager -> DBMSDataController
Purprose	To test if the DBMSDataController correctly handles all types of requests by the ClientManager: • Get Email of a client • Get Password of a client • Create new Client • Edit password • Edit e-mail • Edit payment-information
Input specifications	Create all types of requests that the ClientManager can send to the DBMSDataController
Output specifications	Check if the correct methods are called in the DBMSDataController
Environmental needs	S2I2, S2I3, S2I5 passed

3.3 Integration test Subsystem 3

3.3.1 Integration test S3I1

Test items	AppController -> SearchManager
Purprose	To test if the SearchManager correctly handles the search requests by the AppController: • Search by address • Search by position
Input specifications	Create all types of requests that the AppController can send to the SearchManager
Output specifications	Check if the correct methods are called in the SearchManager
Environmental needs	-

3.3.2 Integration test S3I2

Test items	SearchManager -> CarManager
Purprose	To test if the CarManager correctly handles the search requests by the SearchManager
Input specifications	Create all types of requests that the SearchManager can send to the CarManager
Output specifications	Check if the correct methods are called in the CarManager
Environmental needs	S3I1 passed

3.3.3 Integration test S3I3

Test items	CarManager -> CMSController
Purprose	To test if the CMSController correctly handles the search requests by the CarManager
Input specifications	Create all types of requests that the CarManager can send to the CMSController
Output specifications	Check if the correct methods are called in the CMSController
Environmental needs	S3I1, S3I2 passed

3.4 Integration test Subsystem 4

3.4.1 Integration test S4I1

Test items	ReservationManager -> PaymentManager
Purprose	To test if the PaymentManager correctly handles the requests by the ReservationManager
Input specifications	Create all types of requests that the ReservationManager can send to the PaymentManager
Output specifications	Check if the correct methods are called in the PaymentManager
Environmental needs	-

3.4.2 Integration test S4I2

Test items	ReservationManager -> NotificationController
Purprose	To test if the NotificationController correctly handles the requests by the ReservationManager
Input specifications	Create all types of requests that the ReservationManager can send to the NotificationController
Output specifications	Check if the correct methods are called in the NotificationController
Environmental needs	-

3.5 Integration test Subsystem 5

3.5.1 Integration test S5I1

Test items	AppController -> ReservationManager
Purprose	To test if the ReservationManager correctly handles the reservation requests by the AppController
Input specifications	Create all types of requests that the AppController can send to the ReservationManager
Output specifications	Check if the correct methods are called in the ReservationManager
Environmental needs	-

3.5.2 Integration test S5I2

Test items	ReservationManager -> DBMSDataController
Purprose	To test if the DBMSDataController correctly handles all types of requests by the ReservationManager: • Set Current Reservation
Input specifications	Create all types of requests that the ReservationManager can send to the DBMSDataController
Output specifications	Check if the correct methods are called in the DBMSDataController
Environmental needs	S5I1 passed

3.5.3 Integration test S5I3

Test items	ReservationManager -> ClientManager
Purprose	To test if the ClientManager correctly handles the requests by the ReservationManager
Input specifications	Create all types of requests that the ReservationManager can send to the ClientManager
Output specifications	Check if the correct methods are called in the ClientManager
Environmental needs	S5I1 passed

3.5.4 Integration test S5I4

Test items	ClientManager -> DBMSDataController
Purprose	To test if the DBMSDataController correctly handles all types of requests by the ClientManager: • Check Account • Chack Payment Info • Set OPC • Manage Account
Input specifications	Create all types of requests that the ClientManager can send to the DBMSDataController
Output specifications	Check if the correct methods are called in the DBMSDataController
Environmental needs	S5I1, S5I3 passed

3.5.5 Integration test S5I5

Test items	ReservationManager -> CarManager
Purprose	To test if the CarManager correctly handles the requests by the ReservationManager
Input specifications	Create all types of requests that the ReservationManager can send to the CarManager
Output specifications	Check if the correct methods are called in the CarManager
Environmental needs	S5I1 passed

3.5.6 Integration test S5I6

Test items	CarManager -> CMSController
Purprose	To test if the CMSController correctly handles the search requests by the CarManager
Input specifications	Create all types of requests that the CarManager can send to the CMSController
Output specifications	Check if the correct methods are called in the CMSController
Environmental needs	S5I1, S5I5 passed

3.6 Integration test Subsystem 6

3.6.1 Integration test S6I1

Test items	CMSObserver -> CarManager
Purprose	To test if the CarManager correctly handles the requests by the CMSObserver
Input specifications	Create all types of requests that the CMSObserver can send to the CarManager
Output specifications	Check if the correct methods are called in the CarManager
Environmental needs	-

3.6.2 Integration test S6I2

Test items	CarManager -> ReservationManager
Purprose	To test if the ReservationManager correctly handles the search requests by the CarManager
Input specifications	Create all types of requests that the CarManager can send to the ReservationManager
Output specifications	Check if the correct methods are called in the ReservationManager
Environmental needs	S6I1 passed

3.6.3 Integration test S6I3

Test items	CMSObserver -> EmergencyManager
Purprose	To test if the EmergencyManager correctly handles the requests by the CMSObserver
Input specifications	Create all types of requests that the CMSObserver can send to the EmergencyManager
Output specifications	Check if the correct methods are called in the EmergencyManager
Environmental needs	-

3.6.4 Integration test S6I4

Test items	EmergencyManager -> ReservationManager
Purprose	To test if the ReservationManager correctly handles the emergency requests by the EmergencyManager
Input specifications	Create all types of requests that the EmergencyManager can send to the ReservationManager
Output specifications	Check if the correct methods are called in the ReservationManager
Environmental needs	S6I3 passed

3.6.5 Integration test S6I5

Test items	ReservationManager -> DBMSDataController
Purprose	To test if the DBMSDataController correctly handles all types of requests, concerned the the starting, the ride, the parking and there emergency by the ReservationManager
Input specifications	Create all types of requests that the ReservationManager can send to the DBMSDataController
Output specifications	Check if the correct methods are called in the DBMSDataController
Environmental needs	S6I1 and S6I2 passed or S6I3 and S6I4 passed

3.6.6 Integration test S6I6

Test items	ReservationManager -> RideManager
Purprose	To test if the RideManager correctly handles the requests by the ReservationManager
Input specifications	Create all types of requests that the ReservationManager can send to the RideManager
Output specifications	Check if the correct methods are called in the RideManager
Environmental needs	S6I1 and S6I2 passed or S6I3 and S6I4 passed

3.6.7 Integration test S6I7

Test items	RideManager -> CMSController
Purprose	To test if the CMSController correctly handles the search requests by the RideManager
Input specifications	Create all types of requests that the RideManager can send to the CMSController
Output specifications	Check if the correct methods are called in the CMSController
Environmental needs	(((S6I1 and S6I2) or (S6I3 and S6I4)) and S6I6) passed

4 Tools and test equipment required

In this section we present the approaches and the tool that can be useful for the integration testing described in the previous sections.

The choose of the tool depends on the of programming language, but assuming that we developed PowerEnjoy in Java EE, a suitable choice is Java EE Arquillian (http://www.jboss.org/arquillian) integration testing tool.

Arquillian is an integration and functional testing platform that can be used for Java middleware testing. With the main goal of making integration (and functional) tests as simple to write as unit tests, it brings the tests to the runtime environment, freeing developers from managing the runtime from within the test.

Arquillian supports integration with Java EE containers like GlassFish and servlet containers, and supports running tests in cloud services. Moreover the container support allows developers to target a variety of technology platforms.

Given that Arquillian is used to test integrated subsystems, sometimes is not possible to completely fulfill the testing needs, in that case manual testing could also be useful to simulate user inputs, especially in case of user interface integration testing.

5 Program stubs and test data required

As we have mentioned before, we are going to adopt a bottom-up strategy.

That's why we will need various drivers in order to call the right method on the components involved in the integration testing.

5.1 Drivers

Here follows a list of all the drivers that will be developed as part of the integration testing phase, together with their specific role.

Each table presents the integration test id that the driver refers to, the simulated functionality provided by the driver and the data needed to perform the test with that driver.

- **CMS Driver**

Integration Test ID	S1I5
Driver Name	CMS Driver
Simulated Component	CMS
Simulated Functionalities	Notifications send to the CMSObserver, as they were activated by triggers in CMS
Data needed	Set of • Malfunction Data • Starting engine Data • Plugging charge station Data

- **ClientApp Driver**

Integration Test ID	S1I1
Driver Name	ClientApp Driver
Simulated Component	ClientApp
Simulated Functionalities	• Registration • Login • Search a Car • Edit Account • Reserve a car
Data needed	Set of • Registration Data • Login Data • Searching Parameters • Account Parameters • Reservation Data

- **SearchManager Driver**

Integration Test ID	S3I2
Driver Name	SearchManager Driver
Simulated Component	SearchManager
Simulated Functionalities	Searching input of a car
Data needed	Position or address

- **LoginManager Driver**

Integration Test ID	S2I2
Driver Name	LoginManager Driver
Simulated Component	LoginManager
Simulated Functionalities	Login
Data needed	Client access data

- **RegistrationManager Driver**

Integration Test ID	S2I5
Driver Name	RegistrationManager Driver
Simulated Component	RegistrationManager
Simulated Functionalities	Registration
Data needed	New client data

- **ReservationManager Driver**

Integration Test ID	S4I1 S4I2 S5I2 S5I3 S5I5 S6I5 S6I6
Driver Name	ReservationManager Driver
Simulated Component	ReservationManager
Simulated Functionalities	• Payment request (S4I1) • Send notification request (S4I2) • Query request to DBMS about reservation (S5I2) and ride (S6I5) • Call method checkAccount() (S5I3) • Call method associateOPC() (S5I3) • Call methods about Timeouts and Fines (S5I5) • Call methods about Ride (S6I6)
Data needed	• Payment Data (S4I1) • Notification Data (S4I2) • Reservation Data (S5I2) • Ride Data (S5I5),(S6I6) • OPC Data (S5I3) • Timeout and Fines Data (S5I5)

- **CarManager Driver**

Integration Test ID	S3I3 S5I6 S6I2
Driver Name	CarManager Driver
Simulated Component	CarManager
Simulated Functionalities	• Queries about car (S3I3,S5I6,S6I2) • Queries about pitStops and fines (S6I2)
Data needed	• Car Data • Fine Data • PitStop Data

- **ClientManager Driver**

Integration Test ID	S2I6 S5I4
Driver Name	ClientManager Driver
Simulated Component	ClientManager
Simulated Functionalities	• Call queries on DBMSDataController (S2I6,S5I4) • Call checkPaymentInfo(S2I6)
Data needed	• Client Data (S2I6,S5I4) • Payment Data (S2I6) • OPC Data (S5I4)

- **PaymentManager Driver**

Integration Test ID	S4I3
Driver Name	PaymentManager Driver
Simulated Component	PaymentManager
Simulated Functionalities	Payment receipt notification
Data needed	Payment receipt

- **EmergencyManager Driver**

Integration Test ID	S6I4
Driver Name	EmergencyManager Driver
Simulated Component	EmergencyManager
Simulated Functionalities	Emergency procedure notification
Data needed	Data coming from the CMS about the emergency

- **RideManager Driver**

Integration Test ID	S6I7
Driver Name	RideManager Driver
Simulated Component	RideManager
Simulated Functionalities	Transfer Ride Data to the CMS
Data needed	Ride Data

- **NotificationController Driver**

Integration Test ID	SI12
Stub Name	NotificationController Driver
Simulated Component	NotificationController
Simulated Functionalities	Send notifications to the clientApp
Data needed	set of notifications Data

- **CMScontroller Driver**

Integration Test ID	SI12
Stub Name	CMScontroller Driver
Simulated Component	CMScontroller
Simulated Functionalities	Perform queries on the CMS
Data needed	-

- **DBMScontroller Driver**

Integration Test ID	SI12
Stub Name	DBMScontroller Driver
Simulated Component	DBMScontroller
Simulated Functionalities	Perform queries on the DBMS
Data needed	-

6 Appendix

6.1 Effort Spent

Alessandro Polenghi

- 22\12 2h
- 23\12 3h
- 28\12 1.30h
- 29\12 1h
- 10\1 2h
- 11\1 1h
- 13\1 0.30h

Alessandro Terragni

- 22\12 2h
- 23\12 4.30h
- 27\12 4h
- 10\1 1.30h
- 11\1 2.30h
- 13\1 0.30h