POLITECNICO
MILANO 1863

Integration Test Plan Document

v. 0.0 -- Software Engineering 2, Academic Year 2016 / 2017

Botta Daniel

Bruschi Agnese

Cano Adriana

Sommario

1. Introduction

1.1 Revision History

Version	Date	Authors	Description
0.0	15/01/2017	Botta Daniel, Bruschi Agnese and Cano Adriana	Initial Version

1.2 Purpose and Scope

This document is the Integration Testing Plan Document of the project PowerEnJoy. Its purpose is to describe all the tests that have to be driven to ensure the correct functioning of the requested system.

We will describe which elements are to test and the prerequisites they need to fulfil before being tested. Moreover, we will also describe the approach of the integration testing: how we intend to test each subsystem, in which order and which tools will be used to do so.

1.3 List of Definitions and Abbreviations

RASD: Requirement Analysis and Specification Document

DD: Design Document

FO: Field Operator

BCE: Boundary Controller Entity Diagram

CC: Credit card

App: Application

1.4 List of Reference Documents

Name	Version	Description
Assignments AA 2016 - 2017.pdf	-	The headlines of the ITPD were descripted in this document.
RASD		Requirement Analysis and Specification Document
DD		Design Document

2. Integration Strategy

In this chapter we will present the outline of our integration strategy. We will display the elements of the system – first the high level components, the we will analyse these components in detail. For each component we will establish the eventual entry conditions, the testing order and its own sub-components testing order (when sub-components are present). Then we will proceed, in the next chapter, to analyse thorough every test.

2.1 Elements to be Integrated

The DD describing the design we proposed for PowerEnJoy carries a high level description of the component [see Architectural Design – chapters 2.1 *Overview – high level components and their interaction*, 2.2 *Component View,* DD] and a low level description [see User Interface, chapter 5.3 *BCE Boundary Controller Entity Diagram*, DD] in which we show the elements of each subsystem.

At high level, the main elements that form the system are:

- Database
- Web Application
- Client Mobile App
- FO App

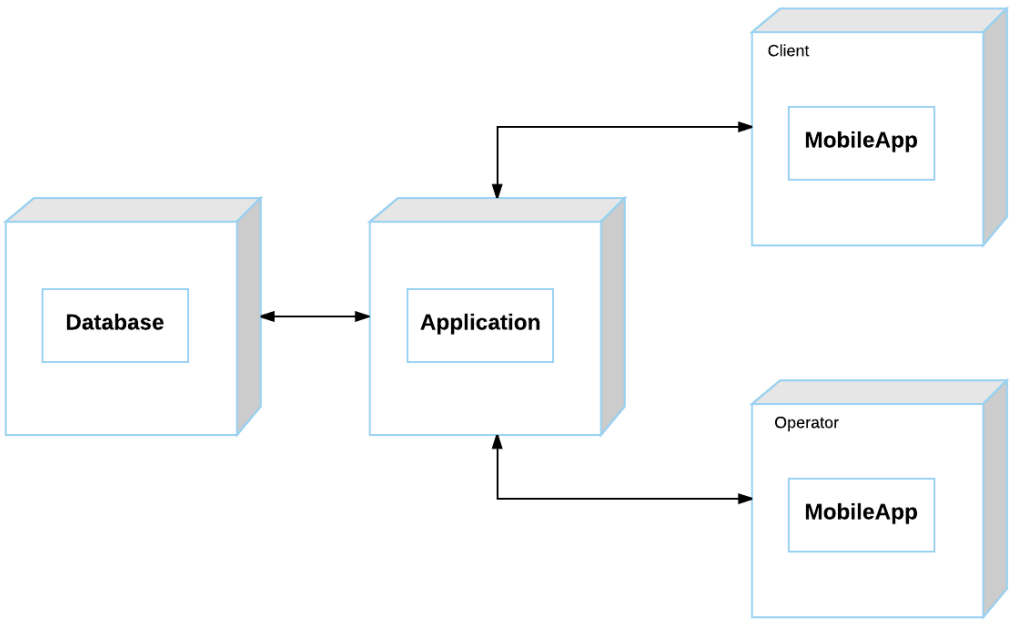


Figure 2.1.a: high level view components

Moreover, we must not forget about the external systems to which we interface, which are:

- Car
- Payment Gateway
- Email Gateway

that we can see in the following image, where the main components are collocated into the tier they belong to.

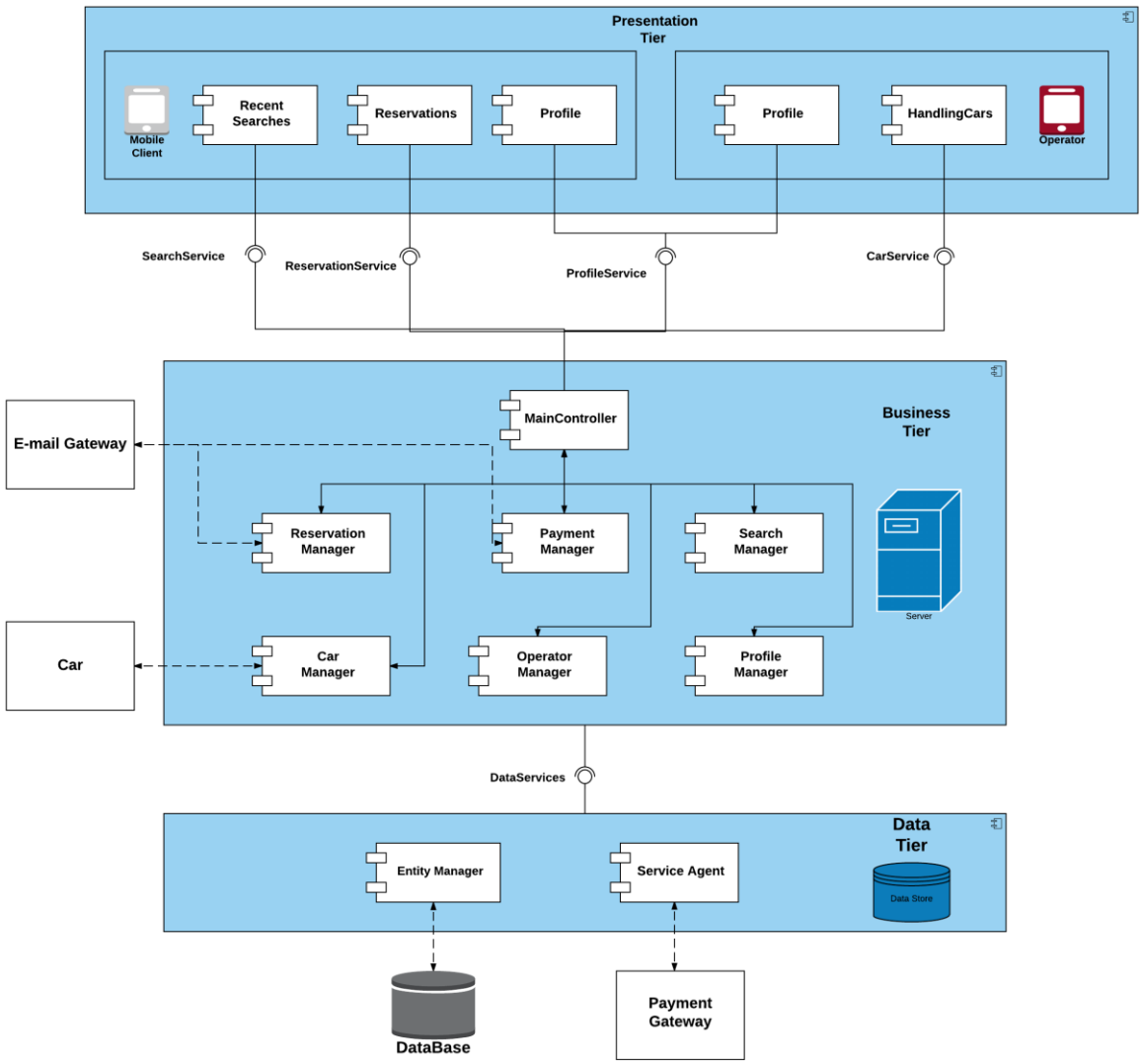


Figure 2.1.b: component view of the system

Based on the reflections and the descriptions made in the DD, we present the main subsystems to test in the following chapter.

2.1.1 Main Components

Users components:

- User: Recent searches, reservation and Profile.
- FO: Profile and Handling Cars.

Business tier components:

- Main controller
  Controller responsible for routing the requests coming from clients and FOs to the right manager in the business tier.

- Profile manager
  Controls the profiles of all the users of the system, checks the correctness of the information inserted upon registration or changed at some point later on, accesses the users' info through the data tier.
- Reservation manager
  Controls the reservations made through the application. It handles the changes of the states of the reservation and communicates with the car manager for the updates of the car, with the payment manager to finalize the fee that the user needs to pay, and with the E mail gateway.
- Search manager
  Controls the searches made by the users and shows them the result of the search. Thus it also communicates with the Car Manager.
- Car manager
  Handles the cars that PowerEnJoy offers to its clients. It is responsible for the updates of the states of the car as reservations happen. It's the only manager that is allowed to access to the car entities in the database.
- Operator manager
  It is in control of the management of the FOs. It communicates with the Car Manager.
- Payment manager
  It is in control of the payment after a reservation is done. It retrieves all the necessary information from the other components and it then communicates a third party to finalize the payment, the Payment Gateway.

The data tier components:

- Entity manager
  It is in control of the entities stored in the database. It handles the insertion and removal of data from the database as well as information retrieval when asked by other components.
- Service agent
  This component is the one responsible for the communication with the external services (i.e. payment gateway, e-mail gateway).

Other components:

The following components play a very important role in the system we designed. For this reason, we here present them and describe which controllers and components of our system will be responsible of representing them in the integration.

- **E-mail Gateway**
  Makes possible the e-mail service to the system, so that e-mails can be sent to the users for reservations and payments (once the payment is done, a recipe is sent to the user via mail).it is all treated by the **Reservation manager**
- **Car**
  The **car manager** handles all the possible interactions with the physical cars and the retrieval of the cars information from the database. Consequently, we will solely focus on the integration of the car manager in our system from now on.
- **Payment Gateway**
  It is handled by the **Payment Manager** and the **Service Agent**.

2.1.2 Sub-components

In our system, there are Managers which do not own sub-components: the ones listed below. They will be treated in the higher part of our testing process, but they will not appear in the low-level integration testing.

- Operator Manager: it handles the FO. Its main function is communicating with the car manager to receive the list of cars to treat them and modify their status.
- Payment Manager
- Entity Manager
- Service Agent
- Search Manager

As a sample for the components of the subsystems, we refer to the BCE diagrams [see User Experience, chapter 5.3 *BCE*, DD] reported below. These diagrams show the managers, their eventual sub-components and their connections with the entities (of the database, from which they pick up information) and the web interface (represented by the boundaries).

In fact, from the diagrams we can derive the components that have sub-systems, and they are described below:

- Profile Manager is divided into Sign-In manager and Registration Manager. The first manages the already registered users, the second provides to the registration. The Profile Manager decides to which manager delegate the management of the user based on the request of the client (registration or sign in, on the initial page of the app). These two component will communicate with the Entity Manager. Plus, there are two more elements, which are CC Controller and Driving Licence Controller, which works for the Registration manager and also are used every time the data given for the CC or the Driving Licence is changed. They interface with outside databases to check the correctness of the data given by the user, thus they will connect to the Service Agent.

In our BCE Diagrams the Main Controller is transparent, but it is still always in charge of directing a request from the client to all the correct managers in the business tier.

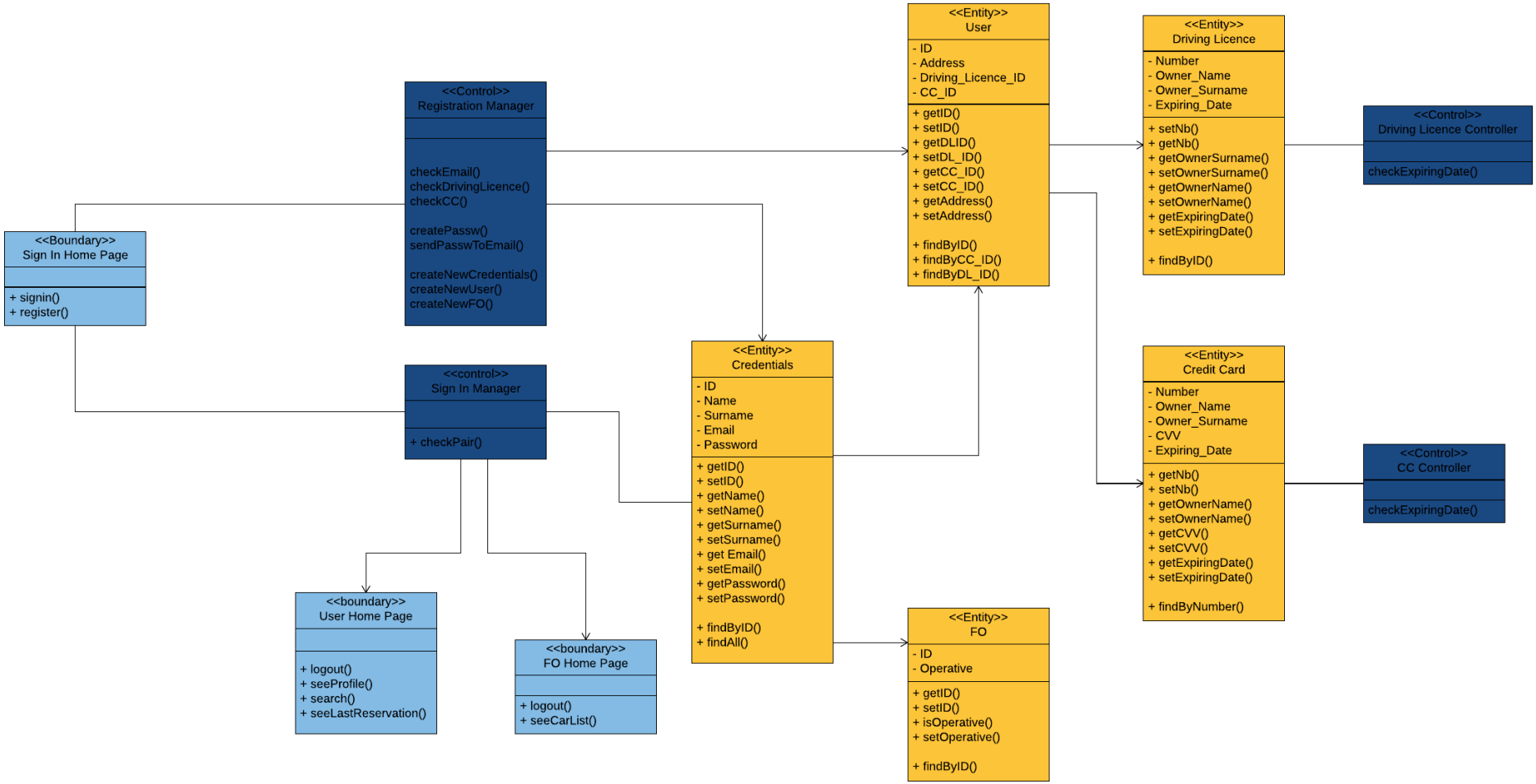


Figure 2.1.c: BCE Registration / Log in

- **Reservation Manager:** the reservation manager is made of sub-components that handle the different part of the registration. The proper **Reservation Manager** is responsible of the actual reservation and of the necessary controls and communicates with the e-mail gateway to send the reservation code. As sub-components we find **Reserved Car Manager** and **Reservation Payment Manager**. The first, once a car is reserved, is responsible of communicating with the car manager to change the state of the car throughout the course of the reservation and to lock/unlock the car after receiving the request from the same car manager. The second one is responsible for communicating with the payment manager (and hence with the payment gateway).

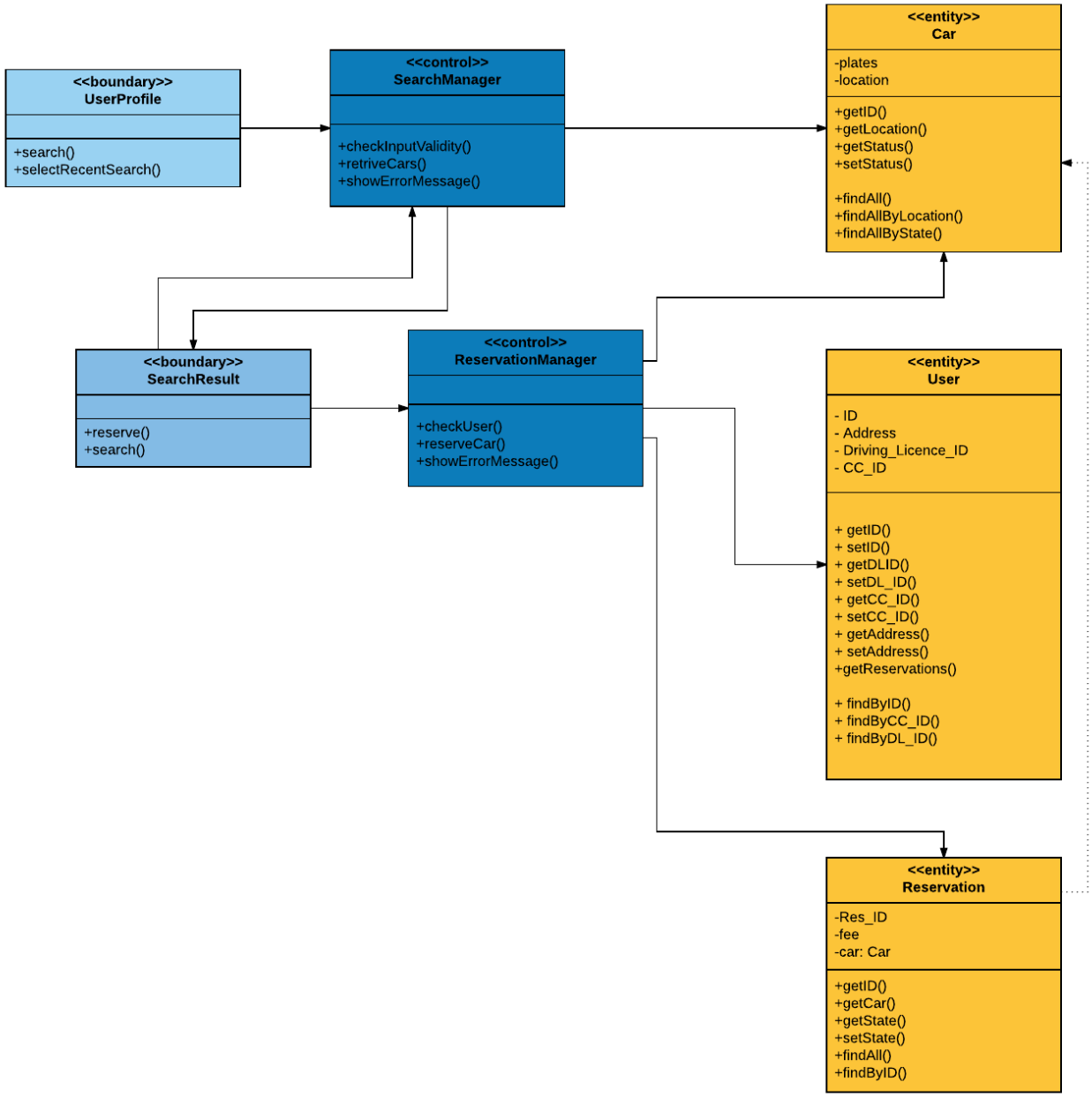


Figure 2.1.d: BCE Search / Reservation

- Car manager: it is formed by Car Status Manager, which interacts with the Registration Manager and the FO Manager. Moreover, it is also composed by the Car Locking System, which is itself composed by the Car Keypad Controller, the one that is in charge of the keypad and allows to unlock the car, and the Car Lock Controller, which inform the system when it is necessary to lock the car. The Car Analysis System is in charge of retrieving information from the sensors in the car (battery, passengers…) while the Car Charging System is the system responsible of calculating the charge of the bill.

2.2 Entry Criteria

The entry criteria are the assumptions required to insert an element in the integration test. Overall, before starting the tests these statements need to be achieved:

- RASD and DD must be completed. DD gives a complete description of the overall components of the elements of the system, the ones which will be tested

In a more detailed degree, we can assume the following subsystems to be already completed in this percentage/in this area:

- CAR, PAYMENT GATEWAY, EMAIL GATEWAY: their software is considered acquired already completed and functioning. The only testing required is the integration testing with the other high level subsystems, to ensure that the communication works correctly.

- DATABASE: The database requires a sample of entities which will allow the search [see the entities required in the database in Management of Persistent Data, chapter 4.3 *Conceptual Design*, DD] of the right information by the managers which will be allowed to have access to the database.
  Our decision is to settle: 10 cars, 2 FOs, 5 clients to start the testing.

- WEB APP, CLIENT MOBILE APP, FO APP:
  a) High level testing:
     The communication systems that allow the subsystems here considered to interact with each other, and with the stubs representing the lower level components are completed.

  b) Low level testing:
     Unit test of the components of the subsystem has already been done. Each function has been tested.

2.3 Integration Testing Strategy

In the DD we proceeded describing our project architecture from the highest level (figure 2.1.a) to the lowest (figures 2.1.c and 2.1.d), in a progressive degree of detail. Thus we have proceeded in a top-down fashion, and the links between our subsystems are clear. For this reason, we decided to also adopt a **top-down** integration test strategy, starting from the elements that interact directly with the user up until the database, which is the deeper element of our analysis.

Our approach will be designed following the architecture schema presented here:

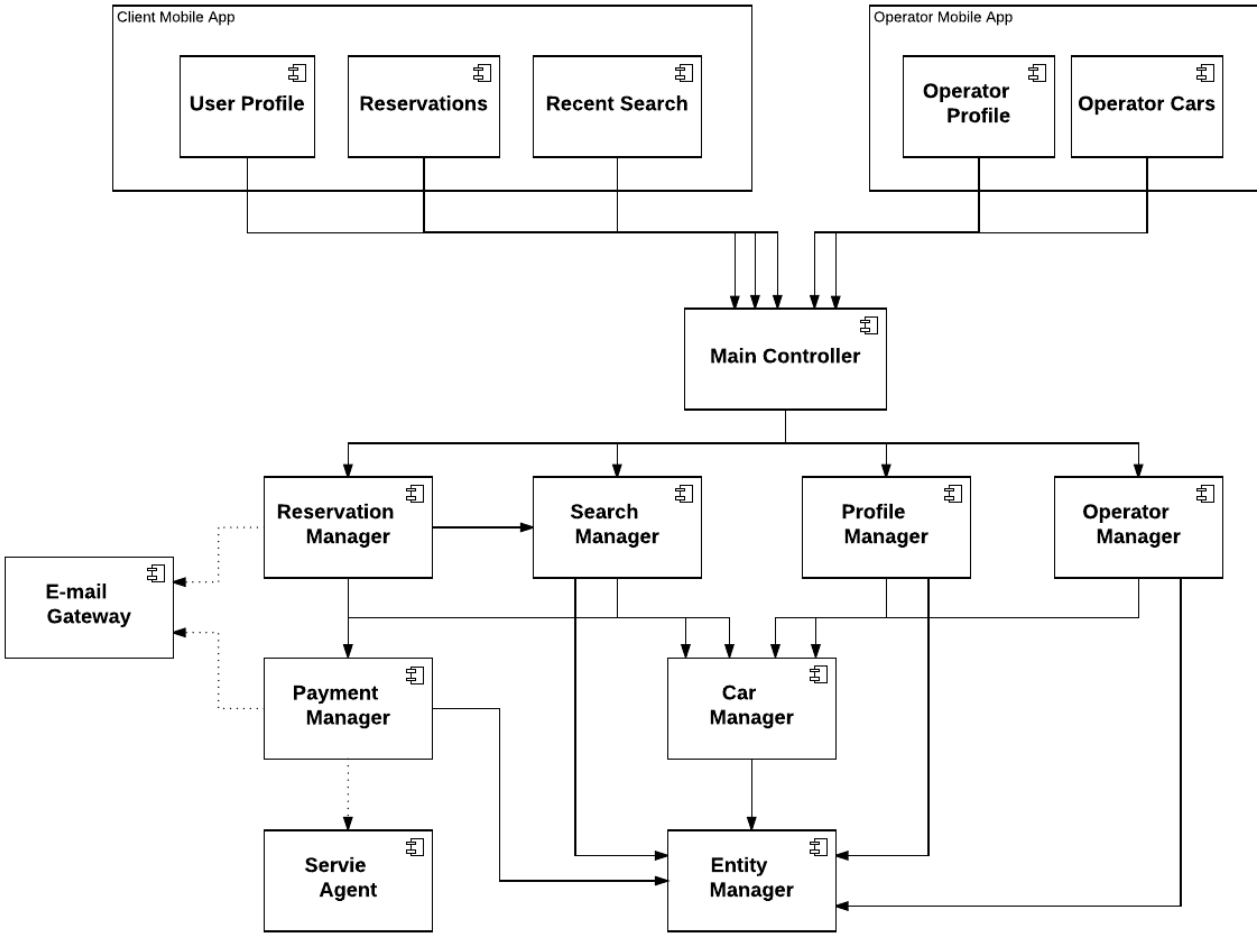


Figure 2.4.a

We will proceed using a **breadth-first** strategy, thus we will first test all the elements that belongs to the same level.

During the tests, it may be possible that some elements, that belong to a deeper level, will be needed to complete the test. These components will be replaced by **Stubs**: Stubs are temporary replacements of systems that are not yet integrated, which give the same output of the actual products. They will be also used to represent external systems.

2.4 Sequence of Component/Function Integration

2.4.1 Subsystem Integration Sequence.

Customer tier:

The first connections to be tested are the ones that link together all the first level components, which are the Client web app and the FO web app, with the Web Application.

1. User Management:
   User profile, Reservation and Recent Search are the three main components of the user management system: they all need to send requests to the Main Controller (in the business tier).

2. FO Management:
   Operator Profile and Operator Cars need to send requests to the Main Controller to be processed.

Business tier:

3. Main controller: To integrate the business tier, we start from the main controller, which is the one that receives all the requests from the upper tier and is in charge of assigning tasks to all the other controllers and subsystems in the business tier. We already tested in the points above the connection with the upper tier (client and FO) thus, the remaining connections to be tested are the ones with the managers in the business tier:
   3.1 Reservation
   3.2 Search
   3.3 Profile
   3.4 Operator

4. Inner communication: testing between elements that belong all to the business tier. Their connection to the Main Controller has already been tested.

   4.1 Reservation manager: the reservation manager has to be:
   - Connected to the Search Manager, because every reservation is made after a previous search.
   - Able to access the database: in this case we will refer to a Stub of the Entity Manager.
   - Able to connect to the payment manager: the Payment Manager will be represented by the Payment Manager Stub.
   - Able to connect to the car manager: the Car Manager will be represented by a Car Manager Stub.

   4.2 Search manager: after the user accesses to the system, the first service to be provided is the search. The Search Manager has to:

- **Communicate with the Car Manager:** the car manager is the only one that is allowed to pick information about the cars from the database. For this reason the Search Manager will send its requests to the Car Manager (in this case, the Car Manager Stub).

4.3 **Profile manager:** the profile if the first thing to be created when a user registers: the Profile Manager gets the request from the Main and stores the data into the database, to which he can later on access. The testing has to make sure that the Profile Manager can:

- Access the database
- Modify the database
- **Access to the outside services:** it is necessary to control the well-functioning of its connection to the Service Agent Stub.

4.4 **Operator manager:** the operator manager has to be able to:

- **Communicate with the Car Manager Stub,** to allow the FO to have access to the cars information and status, and to modify them.

Data tier:

5. **Database Management:** the Entity Manager receives requests from components of the business tier:
   5.1 Payment Manager
   5.2 Search Manager
   5.3 Profile Manager
   5.4 Operator Manager
   5.5 Car Manager

Now that all the internal tests are done, the last high level testing phase consists in testing the elements that communicate with an external system.

E-mail Gateway and Payment Gateway carry links to outside services that need to cooperate with the PowerEnJoy system. As we already stated in the previous chapters, these connections will be treated by Service Agent and Payment Manager. Their inner functions are assumed to always work correctly, thus the tests will focus on the capability of our system managers to perform the correct actions.

6. Email Gateway
7. Payment manager
8. **Car Manager:** Now that all the system has been set up, it's indispensable for the car software interacting with the system to retrieve and send the right data.
   - Send the right number of passengers sitting in the car
   - Send the right percentage of battery left
   - Send the position signalled by the GPS
   - Send the code inserted in the keypad
   - Send the total charge

2.4.2 Software Integration Sequence.

For the composed subsystems, we will display the sequence in which all the sub-components will be tested.

- **Profile Manager**
  1. CC controller
  2. Registration
  3. Sign in

Sequence:

- **Reservation Manager**
  1. Reservation Manager
  2. Reserved car Manager
  3. Reservation payment Manager

- **Car manager**
- **IN CHE ORDINE?**
  1. Car Status Manager: is responsible to check the status of the car (from the database);
  2. Car Locking System: order the car software to lock or unlock the car, and checks the locks: It is composed by;
     a. Car Keypad Controller: it retains the inserted numbers and communicates them to the System;
     b. Car Lock Controller
  3. Car Analysis System: it's in charge of the sensors. The sensor that checks the number of passengers, the charge of the car battery, the GPS.
  4. Car Charging System: calculates the fee and communicate with the

3. Individual Steps and Test Description

For each step of the integration protocol descripted above, we will list below the types of integration tests needed [which were derived from the description of the component integration in chapter 2.4.] and then proceed to describe them in detail.

3.1 High Level Integration Tests

3.1.1 User Management

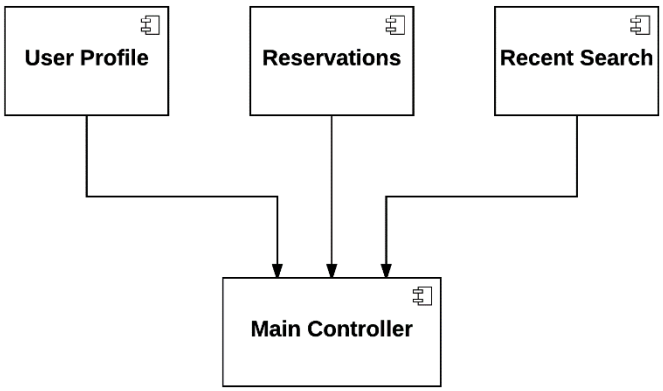


Figure 1: User management

ID	Integration Test
IT1	User Profile → Main Controller
IT2	Reservations → Main Controller
IT3	Recent Search → Main Controller

3.1.2 Operator Management

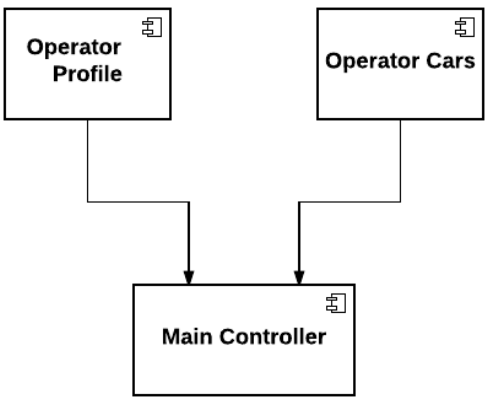


Figure 2: Operator Management

ID	Integration Test
IT4	Operator Profile → Main Controller
IT5	Operator Cars → Main Controller

3.1.3 Main Controller Management

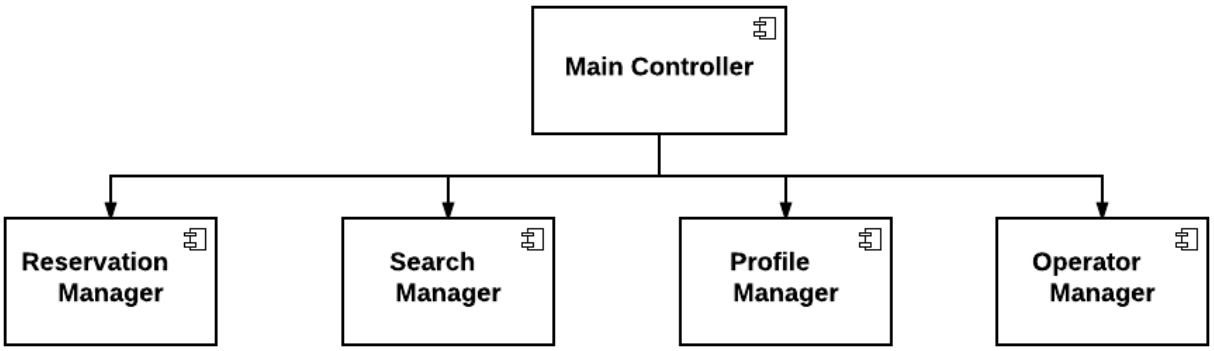


Figure 3: Main Controller Management

ID	Integration Test
IT6	Main Controller → Reservation Manager
IT7	Main Controller → Search Manager
IT8	Main Controller → Profile Manager
IT9	Main Controller → Operator Manager

3.1.4 Inner communication

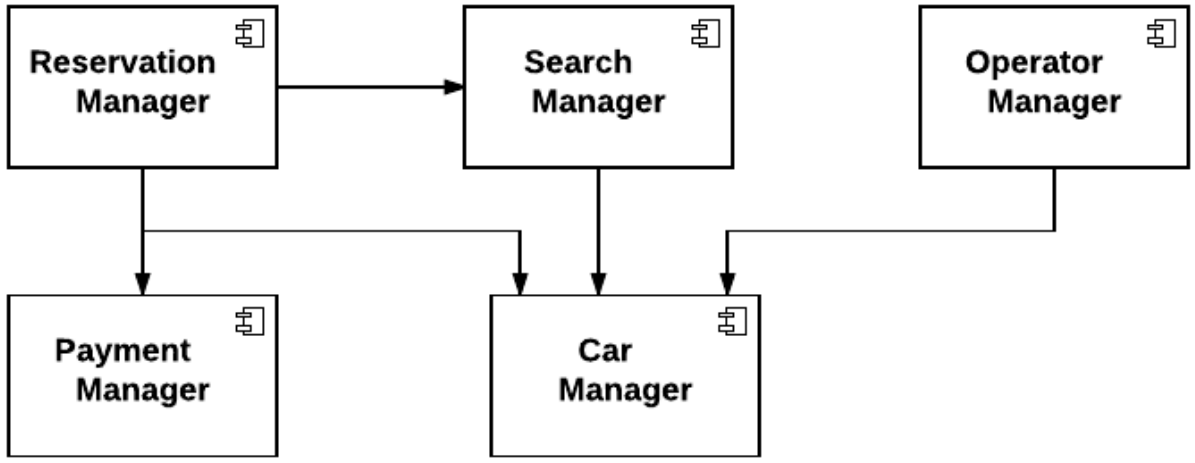


Figure 4: Inner Communication

ID	Integration Test
IT10	Reservation Manager →Search Manager
IT11	Reservation Manager → Payment Manager
IT12	Reservation Manager, Search Manager, Operator → Car Manager

3.1.5 Database Management

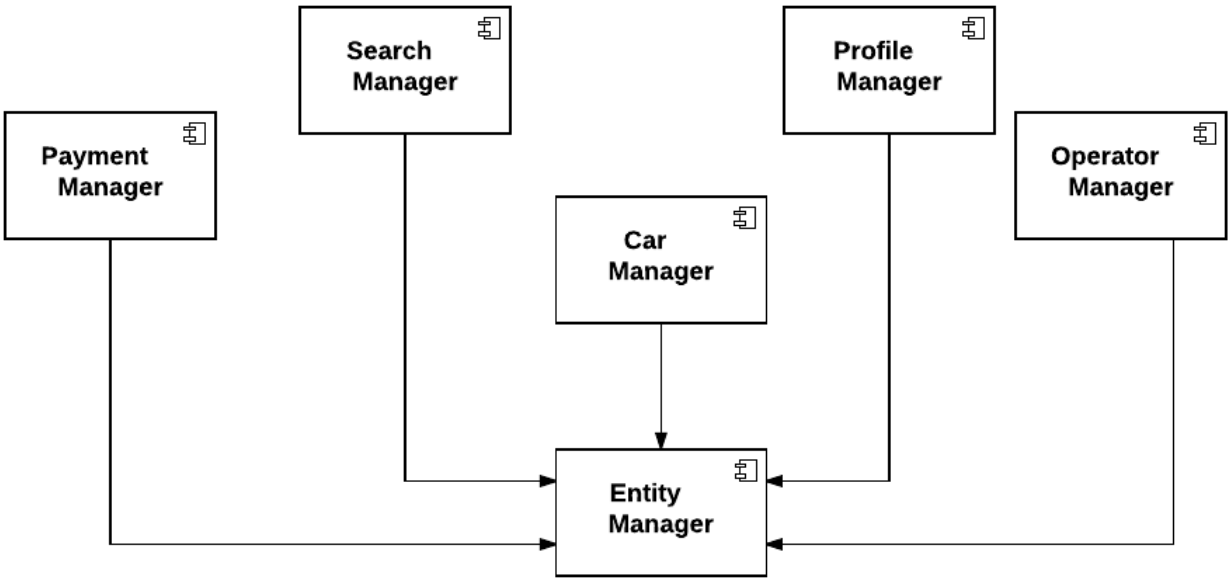


Figure 5: Entity Manager

ID	Integration Test
IT13	Payment Manager, Search Manager, Profile Manager, Operator Manager, Car Manager → Entity Manager

3.1.6 Third Parties Components - Email Gateway

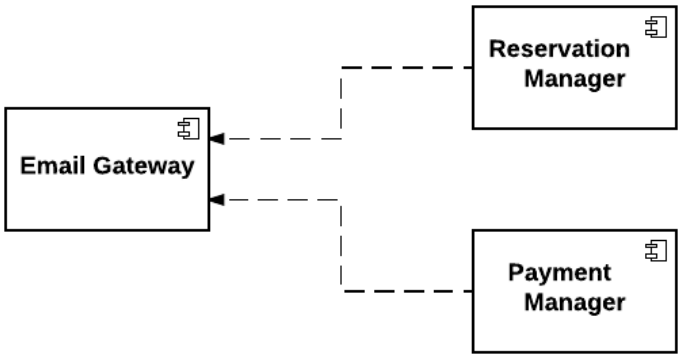


Figure 6: Email Gateway

ID	Integration Test
IT14	Reservation Manager → Email Gateway
IT15	Payment Manager → Email Gateway

3.1.7 Third Parties Components – Payment Gateway

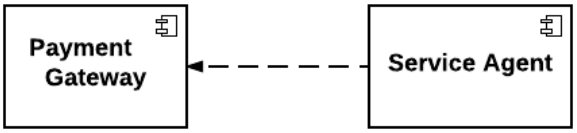


Figure 7: Payment Gateway

ID	Integration Test
IT16	Service Agent → Payment Gateway

3.1.8 Third Parties Components - Car Manager:

Now that all the system has been set up, it's indispensable for the car software interacting with the system to retrieve and send the right data.

- Send the right number of passengers sitting in the car
- Send the right percentage of battery left
- Send the position signalled by the GPS
- Send the code inserted in the keypad
- Send the total charge

ID	Integration Test
IT17	Car Manager → Car Software

3.2 High Level Integration Tests Details

3.2.1 User Management

Test	IT1
Test Components	User Profile || Main Controller
Input specification	Requests from the client regarding the profile, such as registration, log-in, log-out or update of the information
Output specification	The right function calls routed to the main controller, who then directs the call to the consecutive manager. A form is sent back for the client to fill in, different for each request.

Test	IT2
Test Components	Reservation || Main Controller
Input specification	A list of all the reservations of the user in question is requested
Output specification	Request transmitted to the main controller, a list of the reservations is sent back

Test	IT3
Test Components	Recent Search || Main Controller
Input specification	The list of the most recent searches is requested by the client
Output specification	Request transmitted to the main controller, the list of the searches is then sent back if there are any, otherwise a message for the lack of the searches.

3.2.2 Operator Management

Test	IT4
Test Components	Operator Profile || Main Controller
Input specification	Requests from the operator regarding the profile, such as registration, log-in, log-out
Output specification	The right function calls routed to the main controller. A form is sent back specific for any case that the client can fill in.

Test	IT5
Test Components	Operator Cars || Main Controller
Input specification	Requests from the operator to view the car statuses, update the fixing status of the cars
Output specification	The right function calls routed to the main controller, who then directs the call to the consecutive manager. A list of the cars that need fixing is sent back or a message notifying the acknowledgment of the changes.

3.2.3 Main Controller Management

Test	IT6
Test Components	Main Controller || Reservation Manager
Input specification	Requests from the main controller regarding the reservation of a car, in case of a new reservation, the time of the reservation is included
Output specification	The right function calls, a list of the reservations is sent back, information for the specific reservation is sent back or the creation or failure of creation of a reservation. Database is updated in case of a reservation generation.

Test	IT7
Test Components	Main Controller || Search Manager
Input specification	Requests from the main controller regarding the search of a PowerEnjoy car
Output specification	The right function calls and a list of available cars is sent back. In the case of no available cars, a message informing of such thing is sent back.

Test	IT8
Test Components	Main Controller || Profile Manager
Input specification	Requests from the main controller to the profile with the information of which tuple to modify, view or create.
Output specification	The right function calls. Forms generated depending on the incoming request, information requested or messages notifying cases of success or failures.

Test	IT9
Test Components	Main Controller || Operator Manager
Input specification	Requests from the main controller to the operator manager with the information of which tuple to modify, view or create.
Output specification	The right function calls. Depending on the request, forms are generated, information is sent back or a responding message is sent.

3.2.4 Inner Communication

Reservation Management

Test	IT10
Test Components	Reservation Manager || Search Manager
Input specification	Requests from the reservation manager regarding the searches done by the client.
Output specification	The right function calls. Verification of the search done by the client

Test	IT11
Test Components	Reservation Manager || Payment Manager
Input specification	Payment request at the end of the service.
Output specification	The right function calls. Information of a successful payment is sent back or the lack of it signaling a problem.

Car Management

Test	IT12
Test Components	Reservation Manager|| Search Manager|| Operator Manager || Car Manager
Input specification	Requests from the different managers directed at the Car Manager, regarding the state of cars or updating information about cars.
Output specification	The right function calls. Information requested is sent back and information update is acknowledged.

3.2.5 Database: Entity Management

Test	IT13
Test Components	Payment Manager||Search Manager|| Profile Manager||Operator Manager|| Car Manager || Entity Manager
Input specification	Requests from the different managers directed at the Entity Manager, requesting permission to retrieve or update the data in the database.
Output specification	The right function calls. The requested information is sent back and changes are made to the database is the case of update.

3.2.6 Third Parties Components - Email Gateway

Test	IT14
Test Components	Email Gateway || Reservation Manager
Additional Tools	Email address
Input specification	Reservation Manager asks to the Email Gateway to send an email to the specified address.
Output specification	By accessing to the email address, the mail results received.

Test	IT15
Test Components	Email Gateway || Payment Manager
Additional Tools	Email address
Input specification	Payment Manager (which has supposedly received a message of confirmation of payment from the Service Agent) asks to the Email Gateway to send an email to the specified address with a file (the receipt).
Output specification	By accessing to the email address, the mail results received.

3.2.7 Payment Gateway

Test	IT16
Test Components	Payment Gateway || Service Agent
Additional Tools	Most common payment methods: VISA, Maestro, MasterCard, PayPal. Bank account to which the card is linked.
Input specification	1. Payment Gateway requests to pay with a VISA card. 2. Payment Gateway requests to pay with MasterCard. 3. Payment Gateway requests to pay with a Maestro card. 4. Payment Gateway requests to pay with PayPal.
Output specification	1, 2, 3, 4: Service Agent returns "Payment Confirmed". If the user accesses its bank account profile online, the sum charged results subtracted from the previous amount.

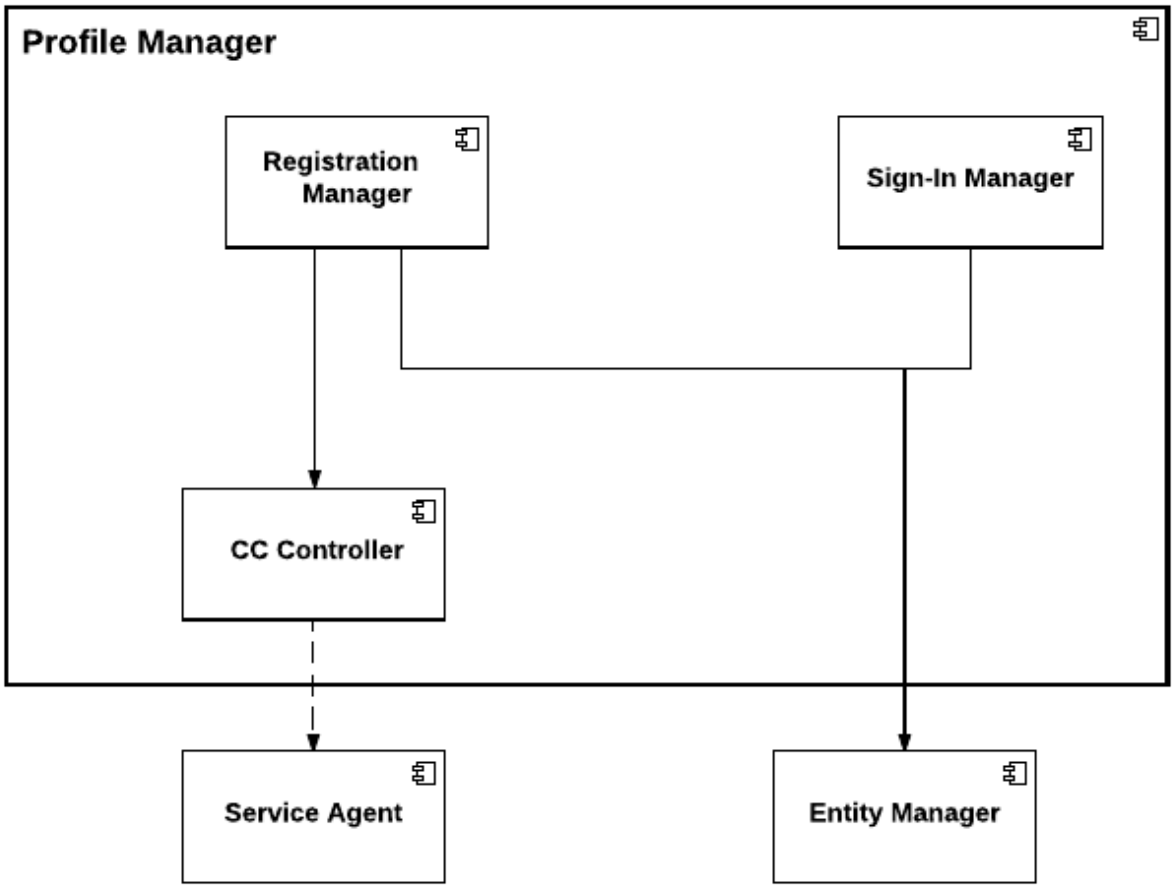
3.2.8 Car Manager

Test	IT17
Test Components	Car Manager || Car system
Additional Tools	Car || Sensors || 5 people
Input specification	1. Request from the Car Manager of the number of people present in the car (i.e. 0, 2, 5) 2. Request from the Car Manager of the state of the charge: the car is charged at different levels (i.e. 20%, 50%, 83%) 3. Request from the Car Manager of the position of the car 4. Code tapped on the keypad 5. Request from the Car Manager of the total time the customer drove. 6. Request from the Car Manager of the status of the car (Locked/Unlocked). 7. The Car Manager orders to Unlock the Car. 8. The Car Manager orders to Lock the Car. 9. The Car Manager asks to temporary lock the Car.
Output specification	1. the Car Manager receives the correct number of passengers. 2. the Car Manager receives the correct percentage of charge. 3. the Car Manager receives the correct position of the car. 4. the Car Manager receives the correct code on the keypad. 5. the Car Manager receives the correct time of usage. 6. the Car Manager receives the correct status of the car. 7. the car is unlocked. 8. the car is locked. 9. the car is temporary locked.

3.3 Low Level Integration Tests

As we stated previously [Chapters 2.1.2 and 2.4.2] Profile Manager is composed by different subsets.

3.3.1 Profile Manager

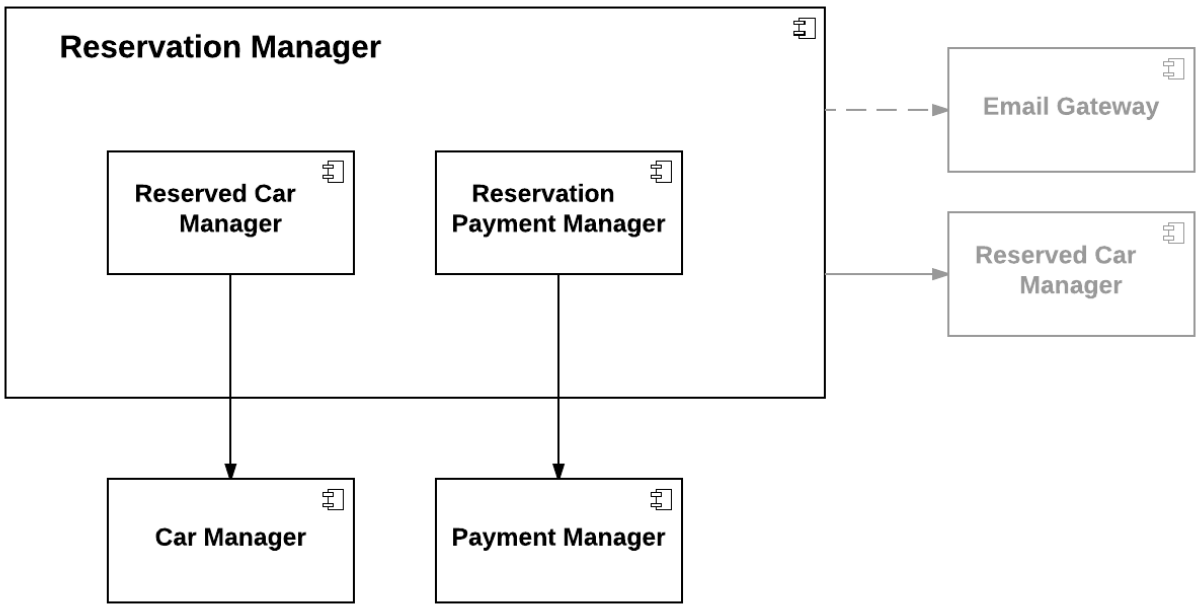


The Profile Manager has a CC Controller that interfaces with the Service Agent, which is linked to the Payment Gateway, that allows the system to check if a payment card is legal and its number is valid. If the user signs-in, the Main Controller should send the request to the Sign-n Manager that will return the forms to compile, and will later check if the pair email – password belongs into the database by calling the Entity Manager. Otherwise, if the client decides to register, the Main Controller will send the request to the Registration Manager, who will send the form and control the inputs. If all the inputs are valid, it calls the Entity Manager to save the info in the database.

They do not add more capabilities to the previous version of the Profile Manager which was displayed: in fact, the tests to run do not change. However, it is important to check that the Main Controller directs the requests to the correct manager, and that these managers can communicate with the components they are supposed to communicate with.

3.3.2 Reservation Manager

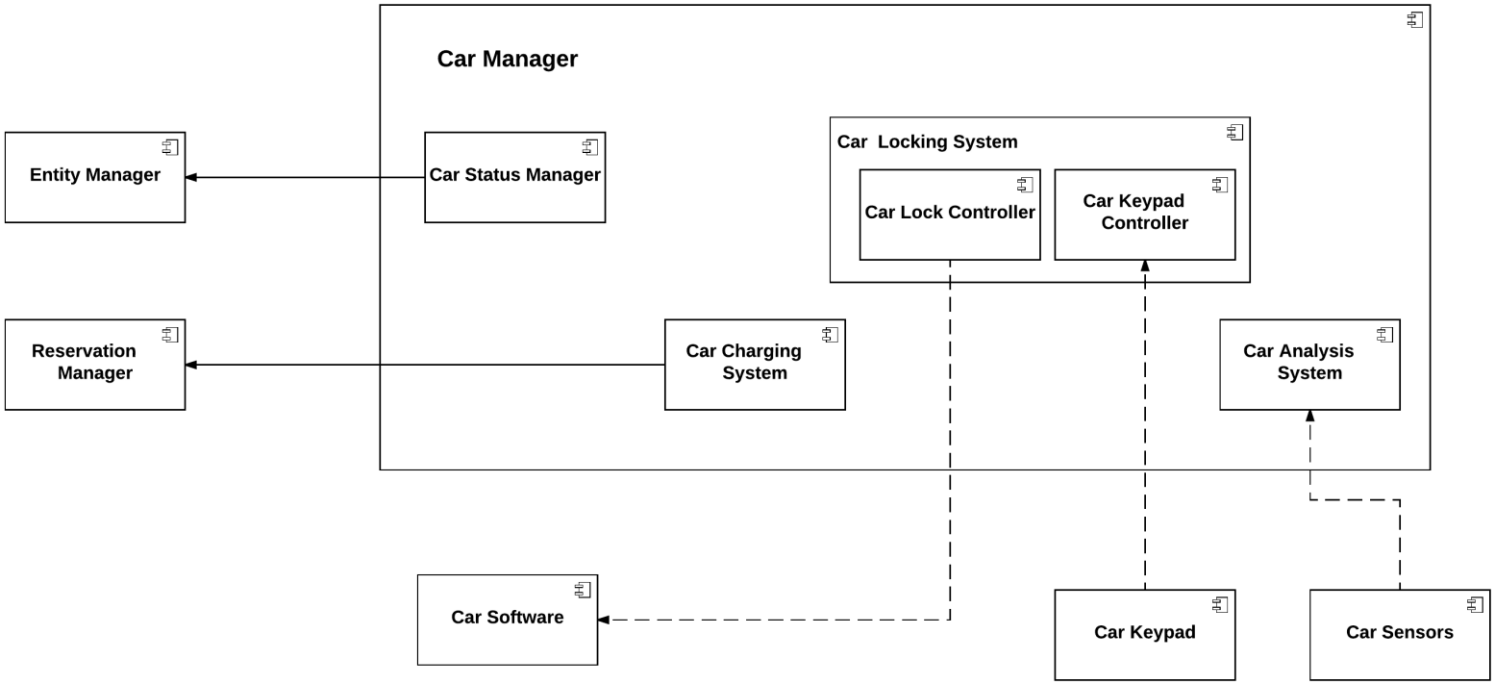
Reservation Manager has two sub-components, which are in charge of the communication with the database (through the car manager) and the confirmation of the payment (through the Payment Manager).



The same concept expressed for Profile Manager can be applied to Reservation Manager and Car Manager.

3.3.3 Car Manager

Car Manager components were fully described in chapter 2.1.2. here we present the structure they form:



It's important to underline that these elements help to simplify the tasks by dividing them between more components. In this way, it is more likely that the single elements may be reused for other purposes, and it will also be easier to add features (as to add more components) or changing the pre-existing features.

4. Tools and Test Equipment Required

In this chapter we identify all tools and test equipment needed to accomplish the integration.

Tools:

As stated into the DD, the PowerEnJoy mobile App is designed to run on iOS, Android and Windows Phone. Furthermore, FOs are given an Android tablet to work.

For these reasons, the tools required to run the tests are:

- at least one Android smartphone
- at least one iPhone
- at least one Windows Phone
- at least one Android tablet

Moreover, we also need:

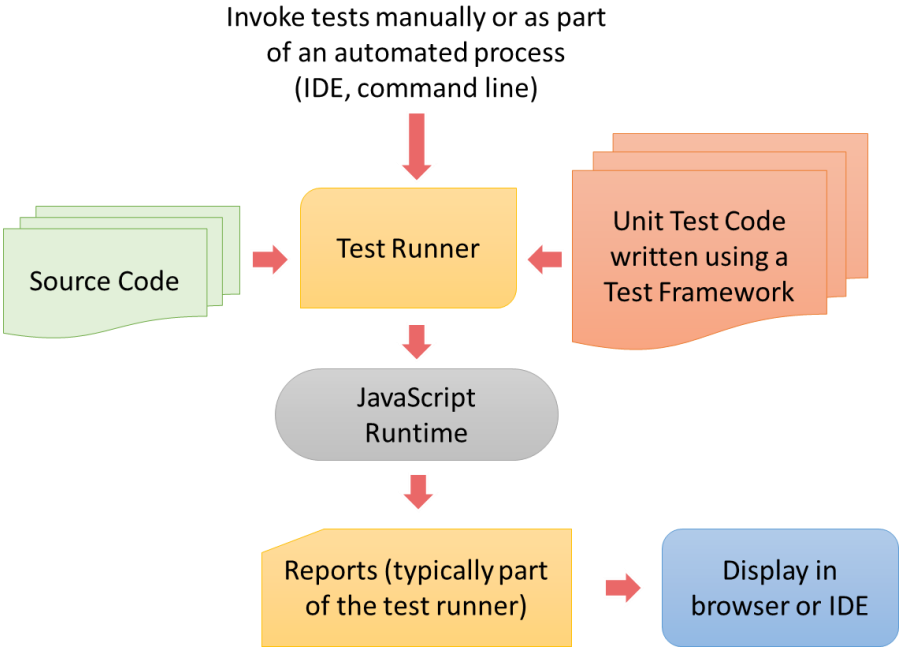
- a Bank account
- one MasterCard
- one PayPal account
- one Maestro credit card
- one Visa credit card
- at least one email account
- at least one car with the running software

Test equipment:

In order to test the various components of PowerEnJoy, we are going to utilize a set of testing environments suitable for **Apache Cordova**:

- For the ***runtime*** we will use **Chrome V8**. This tool compiles and executes JavaScript source code, handles memory allocation for objects and garbage collects objects it no longer needs.
- For the ***framework testing*** we will use **Mocha** along with libraries like **Chai** and **assert.js**. Mocha is a JavaScript test framework running on node.js, featuring browser support, asynchronous testing, test coverage reports, and use of any assertion library.

- For the ***test runner*** we will use **Karma**. It's a simple tool that allows to execute JavaScript code in multiple real browsers, making testing very efficient.



5. Program Stubs and Test Data Required

As we have mentioned earlier, we have chosen a top-down approach. This means we will utilize stubs for testing. All of the main programs calls are replaced with stubs through which the methods are tested.

Only later a stub is replaced with a call to a real method, that may refer to other newly created stub methods as well: only items that have passed the prescribed verification processes can be used in the integration.

Here follows a list of all the stubs that will be created for testing:

- **Reservation Manager Stub:** This stub will simulate the Reservation Manager component and invoke its methods. Interaction between **Search Manager** and **Payment Manager** components will be tested.
- **Payment Manager Stub:** This stub will simulate the Payment Manager component and invoke its methods. Interaction with **Entity Manager** component will be tested.
- **Profile Manager Stub:** This stub will simulate the Profile Manager component and invoke its methods. Interaction between **Car Manager** and **Entity Manager** components will be tested.
- **Operator Manager Stub:** This stub will simulate the Operator manager component and invoke its methods. Interaction between **Car Manager** and **Entity Manager** components will be tested.
- **Car Manager Stub:** This stub will simulate the Car Manager component and invoke its methods. Interaction with **Entity Manager** component will be tested.
- **Entity Manager Stub:** This stab will simulate the Operator Manager component and invoke its methods. It is required for testing all of the other components.

6. Effort Spent

Cano Adriana

Bruschi Agnese 14 h

Botta Daniel 6h