Content

1. Introduction

1. Purpose and Scope

In our project, the Integration Testing Plan Document is an important part, which can guarantee all the subcomponents be integrated consistently.

The main purpose of this document is to make up the system when the subcomponents need to be revised according to the user's requirements or other unexpected reasons. The Integration Testing Plan Document can give us a clear and simple way to organize all the testing of subcomponents. We will provide these sections as follows: - To introduce the integration testing subsystems and their subcomponents involved in the integration activity. - The Elements to be Integrated and their entry criteria. - A description of the Integration Testing Strategy. - The Sequence of Components needs to be integrated, including Software Integration Sequence and Subsystem Integration Sequence. - Individual steps and test description and their input data and the expected output. - A description of performance analysis. - All the tools used in testing, and a description of the operating environment that perform all the tests.

2. Definitions and Abbreviations

1. Definitions

- Subsystem: a single, pre-defined work environment and a high-level functional unit of the system.
- Subcomponent: Implementation of subsystem functions

2. Abbreviations

- RASD: Requirements Analysis and Specification Document
- DD: Design Document
- ITPD: Integration Test Plan Document
- API: Application Programming Interface
- GUI: Graphical User Interface
- DBMS: Database Management System
- GPS: Global Positioning System
- SDK: Software Development Kit
- Req: Requirement
- App: Application

3. Reference Documents

- Requirements Analysis and Specification Document
- Design Document

- Assignments AA 2016-2017
- Integration testing example document
- IEEE standard on requirement engineering

2. Integration Strategy

1. Entry Criteria

For doing this part we should ensure that all units of project work can commence to achieve the corresponding unit test quality objectives and output the corresponding test report.

Besides, the following documents should have been completed, reviewed, approved to do unit testing phase, for instance, the Requirements Analysis and Specification Document and the Design Document. W.R.T integration testing phase, we assume that the project has already being code-complete. And there are no missing features or media elements. The product satisfies the performance and memory requirements specified by the Functional Spec. All priority bugs have been fixed and closed. Internal documentation has been updated to reflect the current state of the product.

Also, we should as possible as keeping the percentage of completion of every component with respect to its functionalities as:

- 100% for the Data Persistence component
- At least 90% for the **Car Management** subsystem
- At least 90% for the **Reservation and Billing Management** subsystem
- At least 80% for the **System Administration and Account Management** subsystems
- At least 70% for the **Client Application**

2. Elements to be Integrated

As we presented before in the design document our system is composed by many components and units. So in this section, we will list all the main components which will be integrated in this phase. And we concern the integration phase as two levels of abstraction: - High-level components integration testing - Lower-level components integration testing

For high-level integration testing, as we introduced 3-tier structure to build our system(PowerEnjoy) in Design Document so we will follow the structure to process the testing, 1. presentation tier (mobile client, web client, web server), 2. logic tier (application server), 3. persistence tier (DB server).

At lower-level integration testing, we decided to integrate those components which are highly depending on one another to offer the higher level functionalities of PowerEnjoy. In this case, these components will be involved : Ride

insertReservation(reservation)	
Input	*Effect*
A null parameter	A NullArgumentException is raised.
A reservation with an id already existent in the database	An InvalidArgumentValueException is raised.
lksqnflkdjflkds	lkfjsdflkjsdlkfjsd

controller, Bill controller, Reservation controller, Economic controller. And we assume that the car controller and charge station and the interaction between car and charge station will be well integration tested by third part.

3. Integration Testing Strategy

The goal of integration testing is to uncover errors that may arise when two or more components are integrated together. As stated before, unit tests are performed and validated before conducting the integration tests. The latter provides assurance that bugs arise from the integrations and not from the components (units) themselves.

We opted for a bottom-up strategy because of many reasons. First, we are using a lot of already-developed or commercially available components that interact directly with our low level components. Since those components are already tested, they represent reliable building blocks to integrate from. Second, using a bottom-up approach removes the need to develop stubs that are needed in a top-down approach. In addition to that, our main datasources (database and car parameters) are already tested which removes the need for stubs. Furthermore, the low-level modules are the most critical in our application (model, localization and car communication) and almost all other components depend on them. So the bottom-up approach ensures that these components are integrated first making sure that they are bug-free earlier. In case errors are uncovered, they can be fixed early in the process.

4. Sequence of Component/Function Integration

```
Software Integration Sequence & Subsystem Integration Sequence
```

3. Individual Steps and Test Description

1. External System

1. DBMS, Model

	insertReservation(reservation)
Input	Effect
A null parameter	A NullArgumentException
By means of Model, A reservation with an id already existent in the database	An InvalidArgumentValueE
Formally valid arguments	By means of Model, an entr

	deleteReservation(reservation)
Input	Effect
A null parameter	A NullArgumentException
By means of Model, A reservation with an id already existent in the database	An InvalidArgumentValueE
Formally valid arguments	By means of Model, an entr

2. Google Maps Component, Localization

3. CarComponents, Model

2. Car Management System

1. Localization, Model

2. Car Component, Model

3. Car Component, Bill Component

4. Car Component, Localization Component

3. Operations Management System

1. Reservation Component, Model

- insertReservation
- delete reservation
- update reservation

2. Bill Component, Model

- update bill queues
- get bill information

updateReservationList(reservationList)

Input	Effect
A null parameter	A NullArgumentException is raised.
An empty array	A NullArgumentException is raised.
An array containing somenull values	An InvalidArgumentValueException is raised.
An array of non-null, but inexistent reservations	An InvalidArgumentValueException is raised.
An array of valid and existing reservations	By means of Model, the corresponding entries in the data

updateBillQueues(billQueue)

Input	Effect
A null parameter	A NullArgumentException is raised.
An empty array	An InvalidArgumentValueException is raised.
An array containing somenull values	A NullArgumentException is raised.
An array of non-nullqueues, but containing null values	An InvalidArgumentValueException is raised.
A non-empty array of valid bill queues	By means of Model, The content of the queues is u

Table 1: My caption

getBillInfo(UserId, ride)

Input	Effect
A null user id	A NullArgumentException is raised.
An invalid user id	An InvalidArgumentValueException is raised.
A non-existing ride	A NullArgumentException is raised.
A valid user id and ride	By means of Model, Th bill information can be get from database.

startRide(carId, currentLocation)

Input	Effect
A null location	A NullArgumentException is raised.
A non-existing location	An InvalidArgumentValueException is raised.
A location far from the current location	An InvalidArgumentValueException is raised.
A current location	By means of Model, the corresponding location in the database ar

updateRideInfo(CarId,RideInfoAvailable)

Input	Effect
A null location	A NullArgumentException is raised.
A non-existing CarId	An InvalidArgumentValueException is raised.
A set of valid parameters	By means of Model, the corresponding ride information in the database are upd

3. Ride Component, Model

- startRide
- updateRideInfo
- getRideInfo
- endRide

4. Reservation Component, Car Component

- ReservationCar
- existsAvailableCar
- Car accept reservation
- Car refused reservation

\begin{table}[]

CarDriverRefusedReservation(CarId)

Input	Effect
An invalid CarId	An InvalidArgumentValueException is raised.
A valid CarId	the car driver refuse reservation and by means of model, the database update instantly

\end{table #### 5. Reservation Component, Bill Component - checkBillInfo

- ConfirmBillInfo

getRideInfo(CarId,RideInfoAvailable)

Input	Effect
A non-existing CarId	An InvalidArgumentValueException is raised.
A set of valid parameters	By means of Model, Returns the stored ride information in the database

endRide(CarId, currentLocation)

Input	Effect
A null location	A NullArgume
A non-existing CarId	An InvalidArg
A valid CarId and current-Location, the Car is on a ride and current Location is inside city	By means of M

ReservationCar(passengerId, passengerLocation, destination)

Input	Effect
A null parameter	A NullArgumentException is raised.
A passengerId not cor-rectly formatted	An InvalidArgumentValueException is raised.
A passenger Location whose coordinates are invalid	An InvalidArgumentValueException is raised.
A destination whose coordinates are invalid	An InvalidArgumentValueException is raised.
A passengerLocation outside the city	An InvalidArgumentValueException is raised.
A valid set of parameters	A new reservation is created and handled

existsAvailableCar(reservation, location)

Input	Effect
A null parameter	A NullArgumentException is raised.
An inexistent location	An InvalidArgumentValueException is raised.
A zone with invalid fields	An InvalidArgumentValueException is raised.
A valid set of parameters	Returns true if a Car driver is available to serve the reservation, false otherwise.

Table 2: My caption

CarDriverAcceptedReservation(CarId)

Input	Effect
An invalid CarId	An InvalidArgumentValueException is raised.
A valid CarId	the car driver accept reservation and by means of model, the database update instantly

checkBillInfo(reservation, ride)

Input	Effect
A null parameters	A NullArgumentException is raised.
An invalid parameters	An InvalidArgumentValueException is raised.
A non-existing ride or reservation	A NullArgumentException is raised.
A valid reservation and ride	From the reservation and ride, the corresponding bill can be check by us

ConfirmBillInfo(reservation, ride)

Input	Effect
A null parameters	A NullArgumentException is raised.
An invalid parameters	An InvalidArgumentValueException is raised.
A non-existing ride or reservation	A NullArgumentException is raised.
A valid reservation and ride	From the reservation and ride, the corresponding bill needs to be confir

RideOfReservation(CarDriver

Input	Effect
A null parameter	A NullArgumentException is raised.
A passenger location whose coordinates are invalid	An InvalidLocationException is raised.
An inexistent car driver	An InvalidArgumentValueException is raised.
A valid car driver and a reservation	Returns the reservation information and ride informati

PassengerChangedDestination(CarDriver, reservation)

Input	Effect
A null parameter	A NullArgumentException is raised.
A change with an invalid destination	An InvalidLocationException is raised.
A change with a valid destination	Returns the reservation information and ride information of new des

6. Reservation Component, Ride Component

- ride of reservation
- passenger Changed Destination
- PassengerInterruptRide

7. Bill Component, Ride Component

- CalculateBill
- GetStoredBillInfo

8. Ride Component, Localization Component

- MonitorCurrentLocation
- GetStoredRideRoute

9. Localization Component, Economic Component

- RemindEconomicInfo
- CalculateEconomicRide
- MonitorEconomicRide

PassengerInterruptRide(CarDriver, reservation)

Input	Effect
A null parameter	A NullArgumentExce
Passengers don't want to go to the destination and asked interrupt ride immediately	Returns the current r

CalculateBill(ride)

Input	Effect
A null parameters	A NullArgumentException is raised.
An invalid parameters	An InvalidArgumentValueException is raised.
A non-existing ride	A NullArgumentException is raised.
A valid ride	According to the formulas we write on DD, the bill can be calculated

GetStoredBillInfo(UserId)

Input	Effect
A null location	A NullArgumentException is raised.
A location whose coordinates are invalid	A InvalidArgumentException is raised.
A non-existing car id	A NullArgumentException is raised.
UserId is valid	User can get the bill information which stored in the database

MonitorCurrentLocation(CarId, locatio

Input	Effect
A null location	A NullArgumentException is raised.
A location whose coordinates are invalid	An InvalidArgumentValueException is raised.
A non-existing car id	A NullArgumentException is raised.
CarId is valid, location is inside city	By GPS ,its status is set to available and its location is written i

GetStoredRideRoute(CarId,location)

Input	Effect
A null location	A NullArgumentException is raised.
A location whose coordinates are invalid	A InvalidArgumentException is raised.
A non-existing car id	A NullArgumentException is raised.
CarId is valid, location is stored in the database	From the location user can get the ride route exists in th

Table 3: My caption

RemindEconomicInfo(UserId)

Input	Effect
A non-existing User id	A NullArgumentException is raised.
UserId is valid	System will remind the economic information when the user make a reservation an

CalculateEconomicRide(UserId, locat

Input	Effect
A null location	A NullArgumentException is raised.
A location whose coordinates are invalid	A InvalidArgumentException is raised.
A non-existing User id	A NullArgumentException is raised.
UserId is valid	The ride can be paid economically because of sharing cars, the p

Table 4: My caption

MonitorEconomicRide(UserId, location)	
Input	Effect
A null location	A NullArgumentException is raised.
A location whose coordinates are invalid	A InvalidArgumentException is raised.
A non-existing User id	A NullArgumentException is raised.
UserId is valid	The economic ride information can be monitor on the user's appl

4. User Management System

1. User Component, Model

2. User Component, Car Component

3. User Component, Bill Component

4. User Component, Reservation Component

4. Tools and Test Equipment Required

5. Program Stubs and Test Data Required

6. Effort Spent

Reda Aissaoui

- 08/01/2017 2h
- 11/01/2017 3h