

PowerEnJoy

Design Document

A.Y 2016/2017

Francesco Tinarelli(matr:806146)

Marco Wenzel(matr:878021)

Versione1.1

Purpose

The Design Document’s purpose is to provide a fully specification of the architecture used for requirements implementation.

The architecture of a software system defines that system in terms of computational components and the interactions among those components.

This document permits to identify:

* High-Level architecture components and their interaction
* The runtime behaviour
* Architectural styles and patterns decision

**Scope**

We will project and implement PowerEnJoy, it’s a digital management system for a car-sharing service.

This system is based on mobile application and web application, there are two kind of different people that can use the system: Unregistered user or Registered user.

Unregistered user can only visit the web application and register himself on the system.

The system allows registered user to see the available cars in a specific zone or current position area using GPS and through this service is possible to make a reservation for up to 1 hour.

The user can unlock the reserved car logging on the web application and using “unlock car” button, the system provides this operation if the user is nearby to the reserved car.

The system registered some special safe area called: power grid area which is possible plug the car.

The system during the ride show the current charges money on the car screen.

More over the screen show power grid stations and permits to choose the path for the nearest grid station or ones specified by the user.

In addition to the functionality above, the system should incentivize the virtuous behaviours of the users, in fact, if the user provides to plug the car or in generally take care of the car, the system rewards him with discount.

There is no system before this one.

Glossary

**External Interface:** System interface that permits interaction among other systems.

**API:** Application Programming interface; it’s a common way to communicate with other system, (application running on external interface).???

**Component**: this terminology is used in component diagram to specify a piece of the system software that aims to provide a specific service.

**MVC**: model view controller.

**Tier**: it’s a specific software level, it’s a parameter that permits to identify architectural style.

**Client-Server**: is a multi-tier architecture.

**Rasd**: Requirements analysis and specification document.

Reference Documents

For creating this DD document, we used:

* Design document notions: Design part I.pdf, Design part II.pdf
* Structure of Document: Assignments2016/2017.pdf
* JEE implementations: JavaEEOverview.pdf
* Example: Sample design deliverable.pdf

**Document Structure**

The document is composed by several parts:

1. **Introduction:** brief introduction for describe the document’s scope and purpose.

A glossary for explain most common terms used into this document.

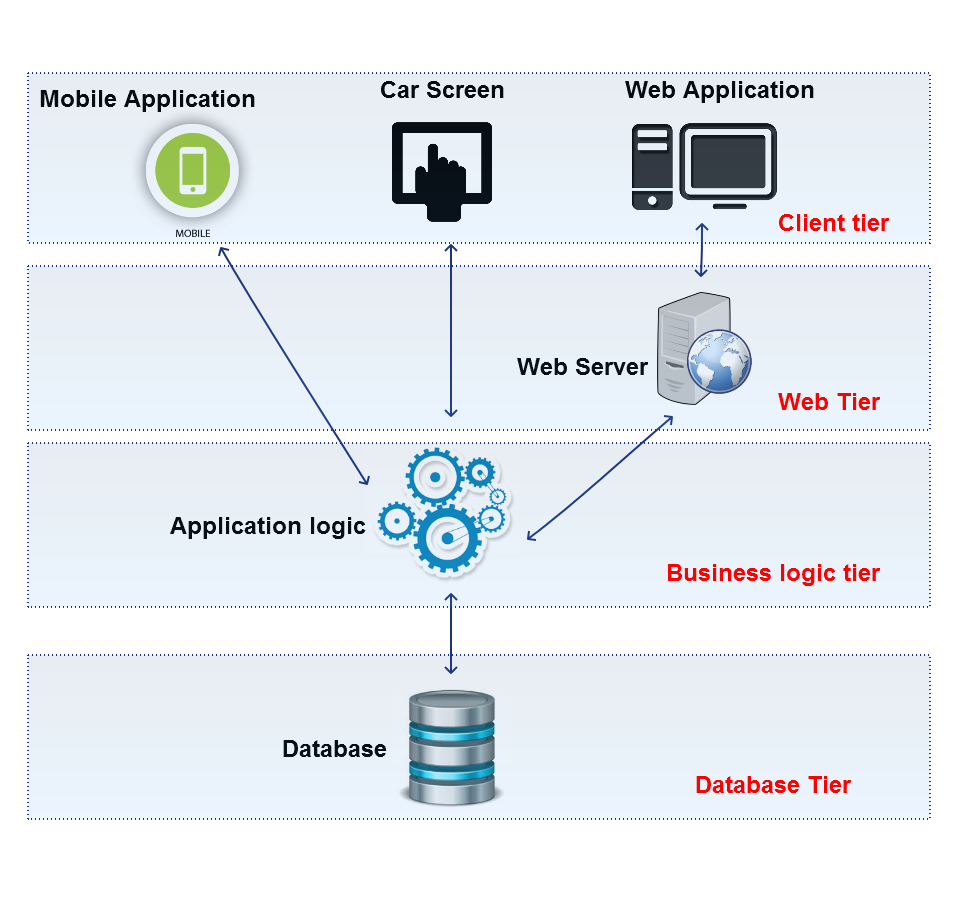
1. **Architectural design:** top-down description, for explain the architectural choice and the interaction among components.

Moreover, a dynamic view architecture.

1. **Algorithm Design:** description using pseudocode the most relevant code part.
2. **Requirements traceability:** Explain how the requirements defined in the RASD are provided through architectural components.
3. **Effort Spent:** how much time we spend for redact this document.

Architectural Design

Overview



We add a tier for (and only) web server because the web site must be dynamic, the services must be dynamically provided from the system, the web pages will be creating in web server.

This architecture represents the devices that need to provide services but among tiers there will be protection component like firewall.

We choose to use Client-Server Architecture, the tiers belong to Server-side are: database, business logic and web server.

This architecture doesn’t want represent a physical division but a conceptual division based on software system.

E.g.: there will be a part of system code nested into every car (and provide several service), but this not represent Client tier because it doesn’t have a directly interaction with user. External Interface can be considered part of Client tier. (???)

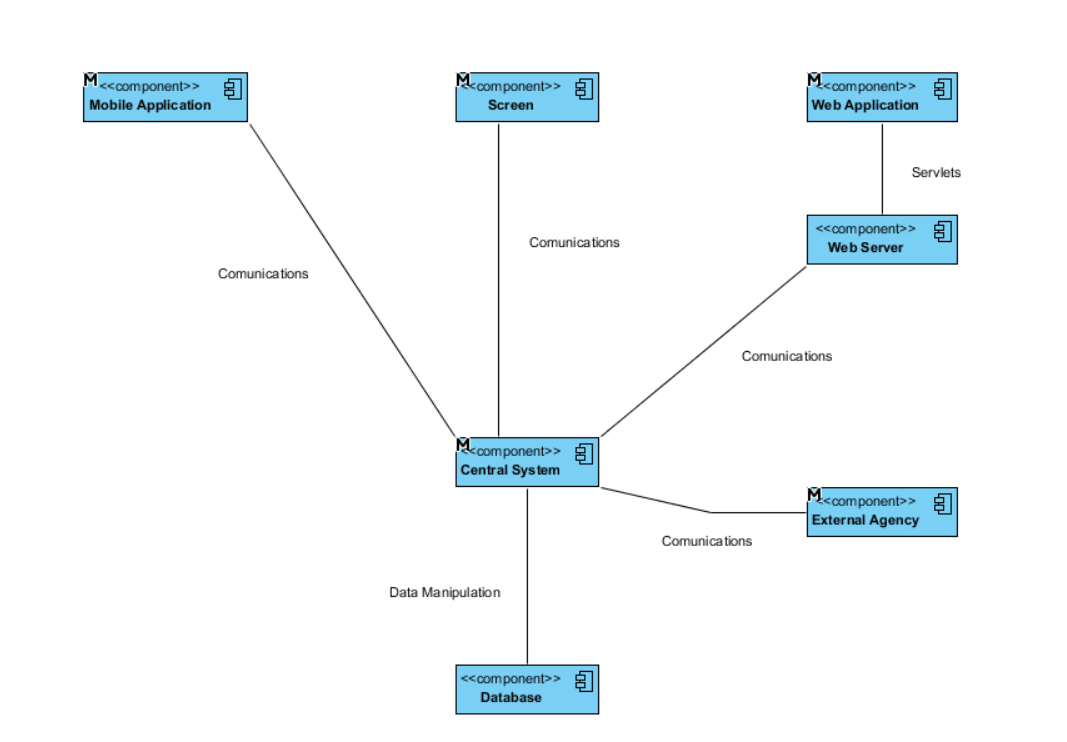
High Level Architecture and their interaction

The view show below is composed by:

* Database: an entity that represents the place where data is stored.
* Business Logic: or also called Central System, is an entity that represents the part of system where data is managed, moreover in this part, system receives/sends data from front-end application.

The interaction between Central System and other front-end application is made in a synchronous way for providing service about the ride.

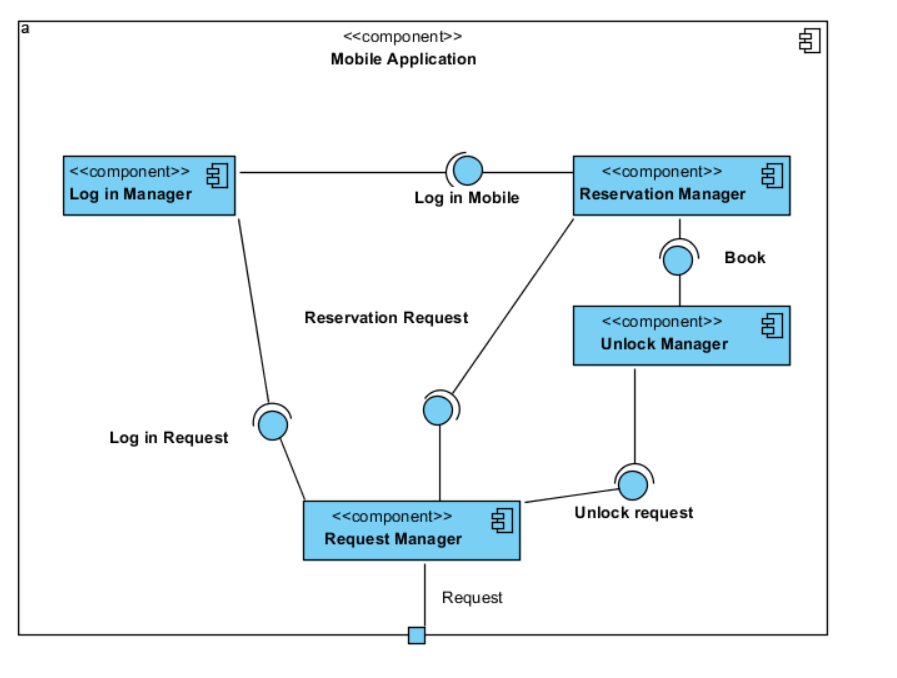
* Web Server: an entity that aims to create web pages dynamically.
* Web Application: front end application that provide the reservation, payment history, registration and log-in service.
* Mobile Application front end application that provide the reservation, unlocking and log-in car service.
* Car screen: front end application that permits to see the grid station positions and calculates the path from the nearest one.



The “M” indicate that components will be expanded.

Component Level Architecture and their interaction

Mobile Application



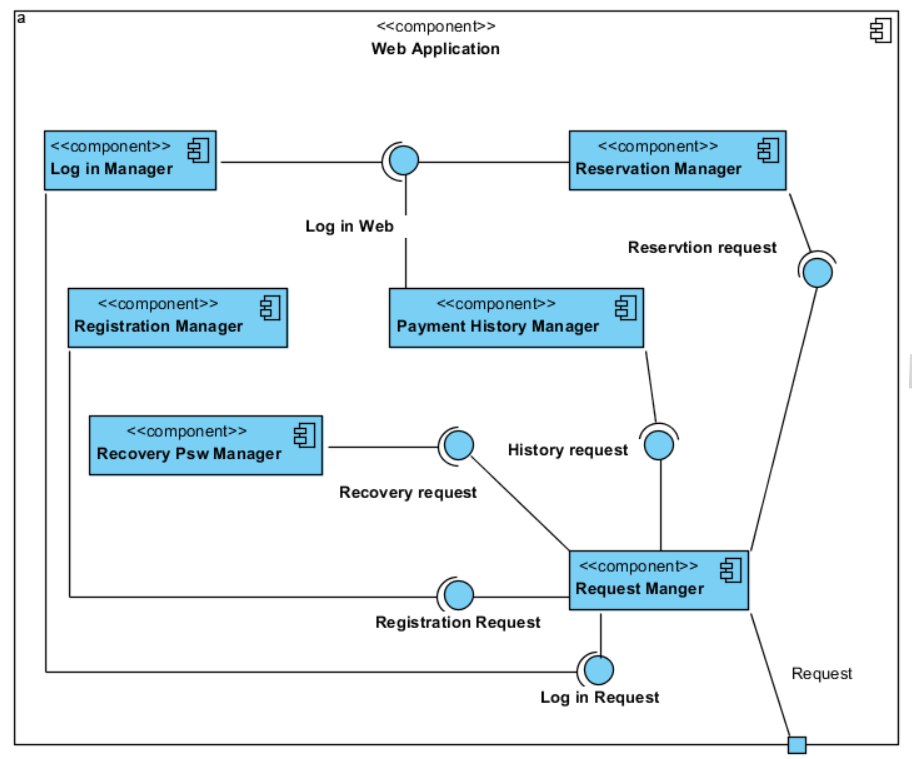
**Log in manager:** manage the log-in service, control syntactical requirements.

**Reservation Manager:** manage the reservation service, provides the map with the available car.

**Unlock Manager:** manage the unlock service, controls if exist a valid reservation.

**Request Manager:** manage the incoming request of service from the service manager and sends the request to the Central System.

Web Application



**Log-in manager:** manage the log-in service, control syntactical requirements.

**Reservation Manager:** manage the reservation service, provides the map with the available car.

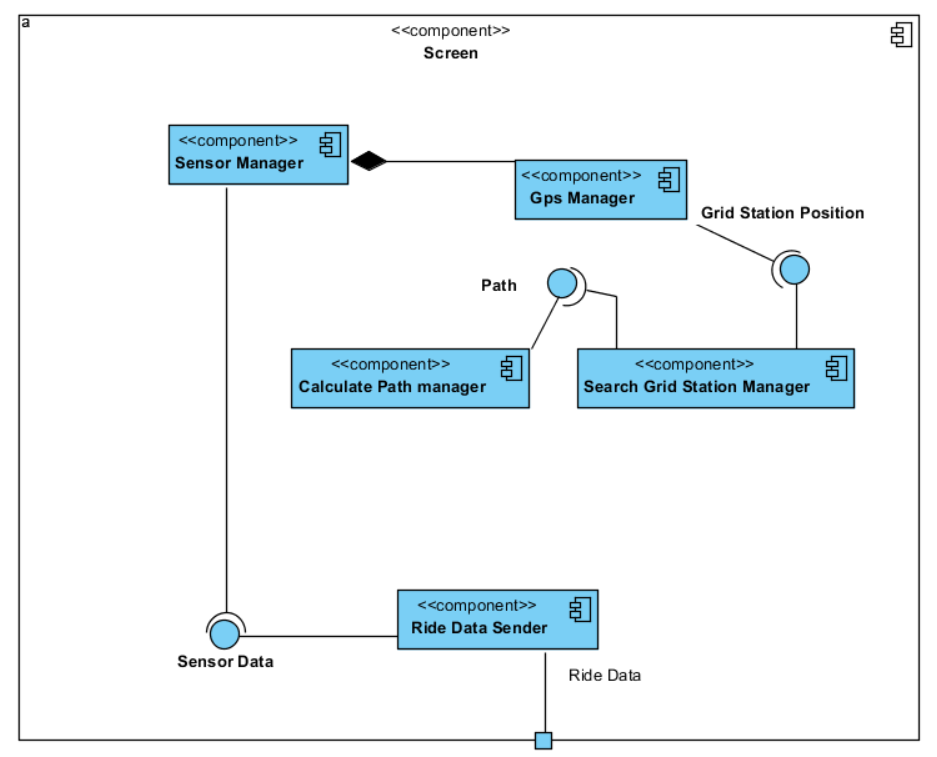
**Registration Manager:** manage the registration service, control syntactical requirements.

**Recovery Psw Manager:** manage the password recovery service, control syntactical requirements.

**Payment History Manager:** manage the payment history service, permits to show the reservation timer (and the unresolved payment).???

**Request Manager:** manage the incoming request of service from the service manager and sends the request to the web server.

Car Screen Application



**Sensor Manager:** manage all components inside the car, permits to capture passenger number, battery level and car position.

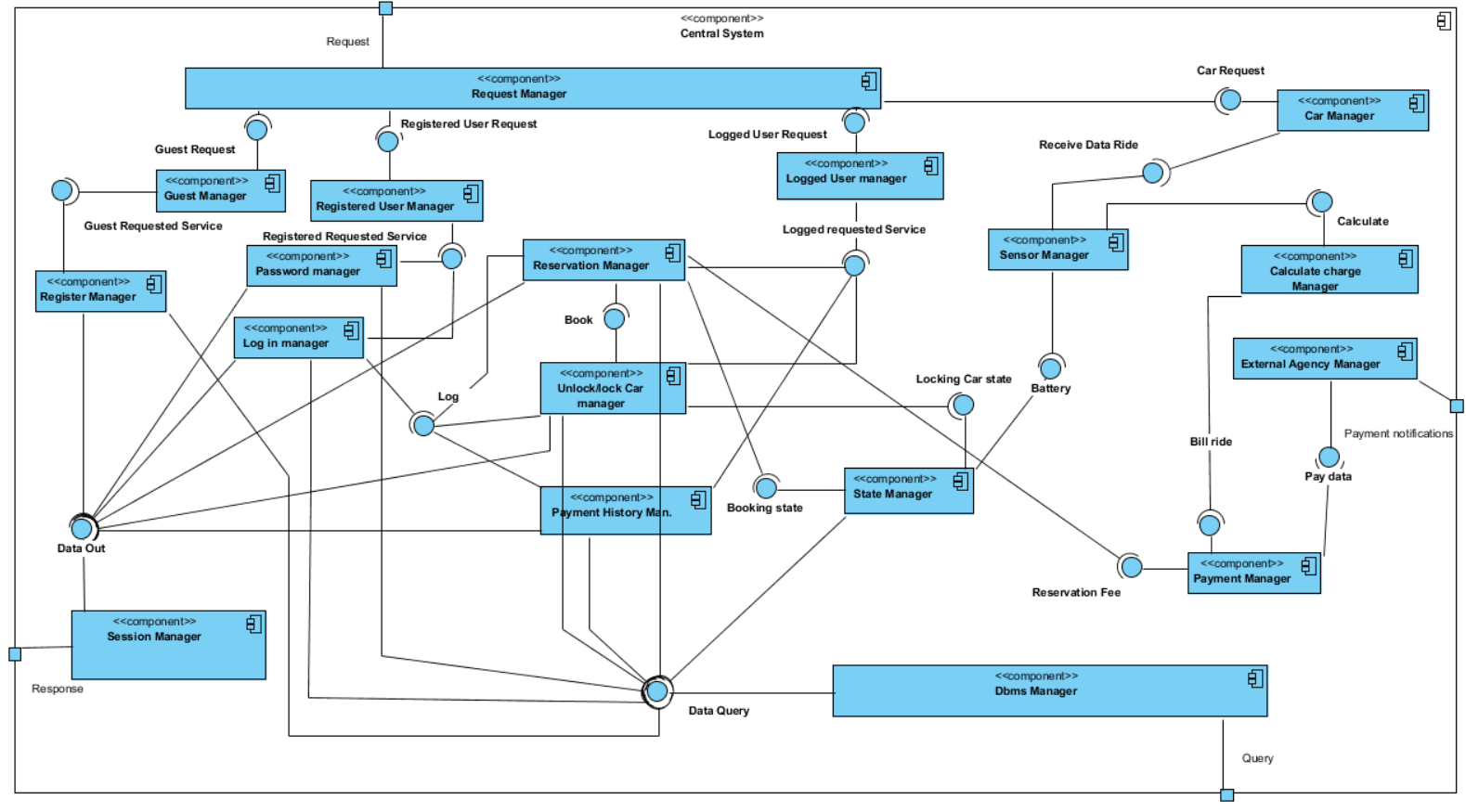
**Gps Manager:** manage all incoming data from gps and permits to show on the car screen.

**Search Grid Station manager:** manage the car gps data and permits to search the nearest grid station or another one specified by the user.

**Calculate Path manager:** permits to calculate and update on screen the distance from the car to the specific station.

**Ride Data Sender:** send all data ride to the Central System for calculating the charge ride and creates the bill ride.

Business Logic Application

****

**Request Manager:** manage the incoming request from the front-end applications, manages a buffer for these requests and recognizes the type of the user request, sends this request to specific managers.

**Guest Manager/Registered User Manager/Logged User Manager/Car Manager:** manage requests from the same type of user, manages a buffer for these requests and send these request to the specific manager that provides the requested service.

**Register Manager:** Provide registration service and check the requirements validity, this component manages the single user request (create a query) and sending to dbms manager for executing this one.

Then dbms response, the manager provides requirements control.

**Password Manager:** Provide password recovery service and check the requirements validity, this component manages the single user request (create a query) and sending to dbms manager for executing this one.

Then dbms response, the manager provides requirements control.

**Log in Manager:** Provide log in service and check the requirements validity, this component manages the single user request (create a query) and sending to dbms manager for executing this one.

**Reservation Manager:** Provide reservation service, calculate the reservation time and check the requirements validity, this component manages the single user request (create a query) and sending to dbms manager for executing this one.

Then dbms response, the manager provides requirements control.

**Unlock Car Manager:** Provide unlocking service and check the requirements validity, this component manages the single user request (create a query) and sending to dbms manager for executing this one.

Then dbms response, the manager provides requirements control.

**Payment History Manager:** Provide payment history service and check the requirements validity, this component manages the single user request (create a query) and sending to dbms manager for executing this one. (???incosistency)

Then dbms response, the manager provides requirements control.

**Sensor Manager:** Receives and manages all sensor data for permits other manager to use them.

**Calculate Charge Manager:** Manage all ride data provided from Sensor manager and creates the bill ride keep in mind the possible discounts or fee.

**Payment Manager:** Manage all bill ride incoming reservation manager and calculate charge manager, prepares the bill notification to send to external agency.

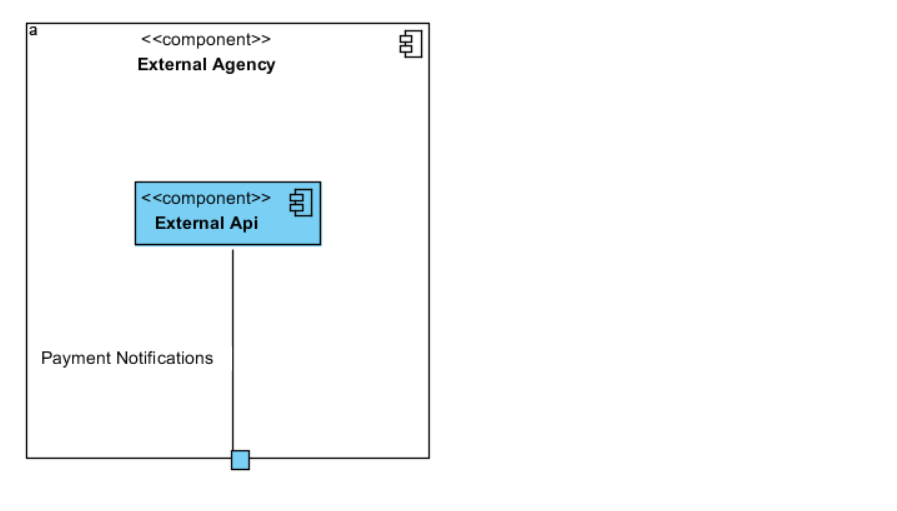
**External Agency Manager**: Manages the connection with External Agency Interface System and notified it with bill notifications.

**State Manager:** Manage the state of the car whenever occur.

**Dbms Manager:** Receives query from service manager and interrogate the database, sends the database response to the correspond service manager.

**Session Manager:** Manages all outgoing response request through received from the service managers an output buffer and send this one to the respective front-end applications.

**External Agency**

****

**External Api:** Manages the incoming notifications from Central System.

Deployment Diagram