Sure-Park Project

Architectural Design Document

Version 1.0

|  |  |
| --- | --- |
| **Team** | **Infinite Challenge(Team 3)** |
| **Members** | Namjin Lee, Jack Oh, Charles Park, Joan Kim, Jaeheon Kim |

**History**

|  |  |  |
| --- | --- | --- |
| **Ver.** | **Date** | **History** |
| 1.0 | 19, June, 2016 | Initial |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[**1.** **System Context** 4](#_Toc454280582)

[**2.** **1st Decomposition** 5](#_Toc454280583)

[**2.1.** **Physical View of 1st Decomposition** 5](#_Toc454280584)

[**2.2.** **Dynamic View of 1st Decomposition** 5](#_Toc454280585)

[**2.3.** **Static View of 1st Decomposition** 6](#_Toc454280586)

[**2.4.** **Entity Catalog** 6](#_Toc454280587)

[**2.5.** **Rationale** 7](#_Toc454280588)

[**3.** **SurePark Manager** 8](#_Toc454280591)

[**4.** **Facility Controller** 10](#_Toc454280592)

[**5.** **Web Service** 10](#_Toc454280593)

[**6.** **Facility Controller to SurePark Manager** 10](#_Toc454280594)

[**6.1.** **How to check if Facility Controller is alive.** 10](#_Toc454280595)

[**7.** **SurePark Manager to Web Service** 11](#_Toc454280596)

[**8.** **Detail Design** 11](#_Toc454280597)

[**8.1.** **FC-SM packet** 11](#_Toc454280598)

[**8.1.1.** **Packet Structure** 11](#_Toc454280599)

[**8.1.2.** **Detailed Packet Scenario** 12](#_Toc454280600)

[**8.2.** **SM-WS packet** 14](#_Toc454280601)

[**8.3.** **SM-WS DB Schema** 14](#_Toc454280602)

[**8.4.** **SurePark Manager Detailed Design** 14](#_Toc454280603)

[**8.5.** **Facility Controller Detailed Design** 14](#_Toc454280604)

[**8.6.** **Web Service Detailed Design** 14](#_Toc454280605)

[**9.** **Future Works** 14](#_Toc454280606)

[**10. Lesson Learned** 14](#_Toc454280607)

# **System Context**

우리의 목표는 안정되고, 확장이 쉽고, 보안이 훌륭한 Sure Park System을 개발하는 것이다. Sure Park System은 H/W Part로 Parking Garage를 제어할 수 있어야 하고, 주차 예약, 주차 감시 및 관리 시스템을 제공해야 한다.

The Sure Park system



< Figure1. System Context >

# **1st Decomposition**

## **Physical View of 1st Decomposition**



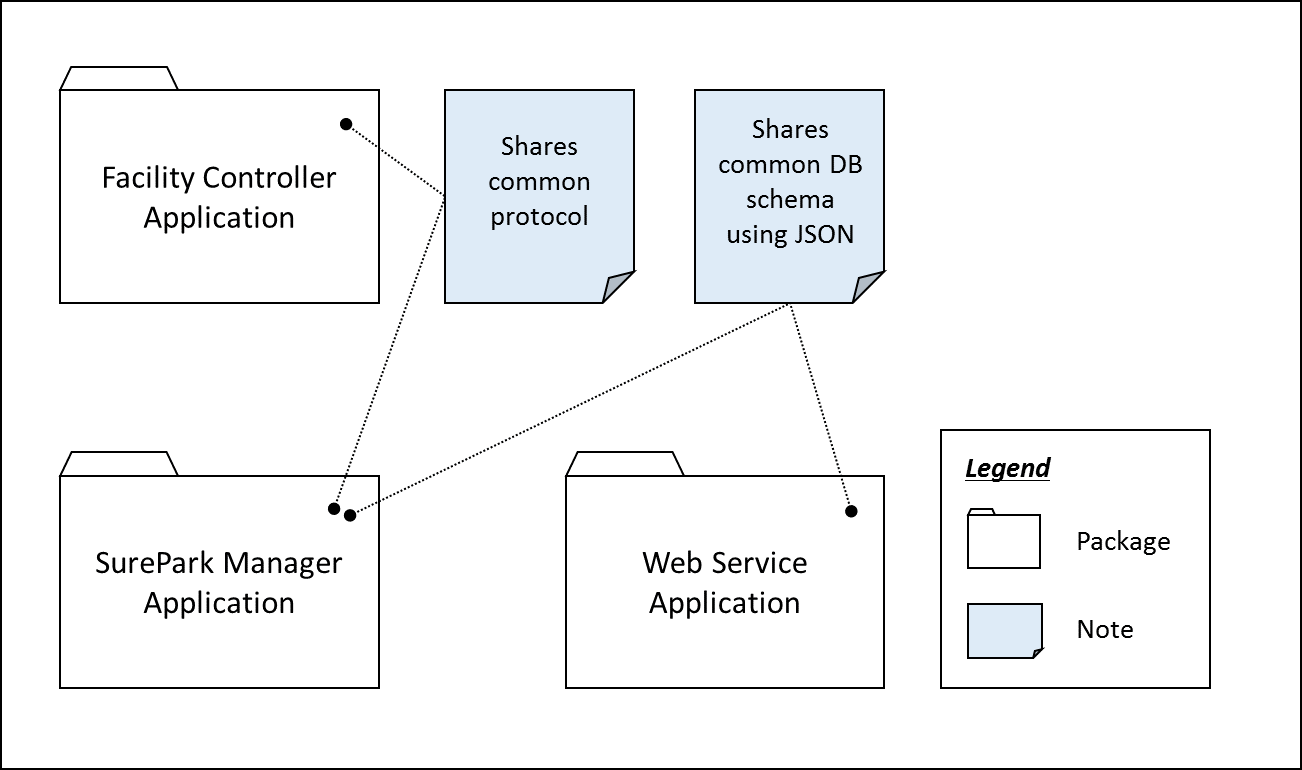
< Figure 2. Physical view of 1st decomposition >

## **Dynamic View of 1st Decomposition**



< Figure 3. Dynamic view of 1st decomposition >

## **Static View of 1st Decomposition**



< Figure 4. Static view of 1st decomposition >

## **Entity Catalog**

|  |  |
| --- | --- |
| **Entity** | **Description** |
| Web Browser | Users, attendants and owner can access their own UI through the web browser provided by the web server. |
| Web Service | Provides users with the functions of sign-up, log in, reservation, monitoring facilities and/or showing parking statistics based on data retrieved from SurePark DB.  Sends information to SurePark Manager for DB updates. |
| Facility Controller | Controls parking facilities; get the status of parking slots, turn on/off LEDs, detect a car at the gates and open/close the gates.  Receives data from SurePark Manager to control LEDs and/or gates.  Sends data to SurePark Manager to update the status of parking slots. |
| SurePark Manager | Handles show-up and no-show scenarios based on DB information.  Updates SurePark DB when a user has signed up, a reservation has been made or facility status has been changed. |
| SurePark DB | Keeps all of the data about users, garages and reservations.  Only can be updated by SurePark Manager. |

## **Rationale**

Modifiability (QA08) is one of the most important QAs of the SurePark system. An engineer needs to scale up the system within a week. We have divided the whole system into 5 parts according to their responsibilities, and applied client-server and repository pattern to connect each parts.



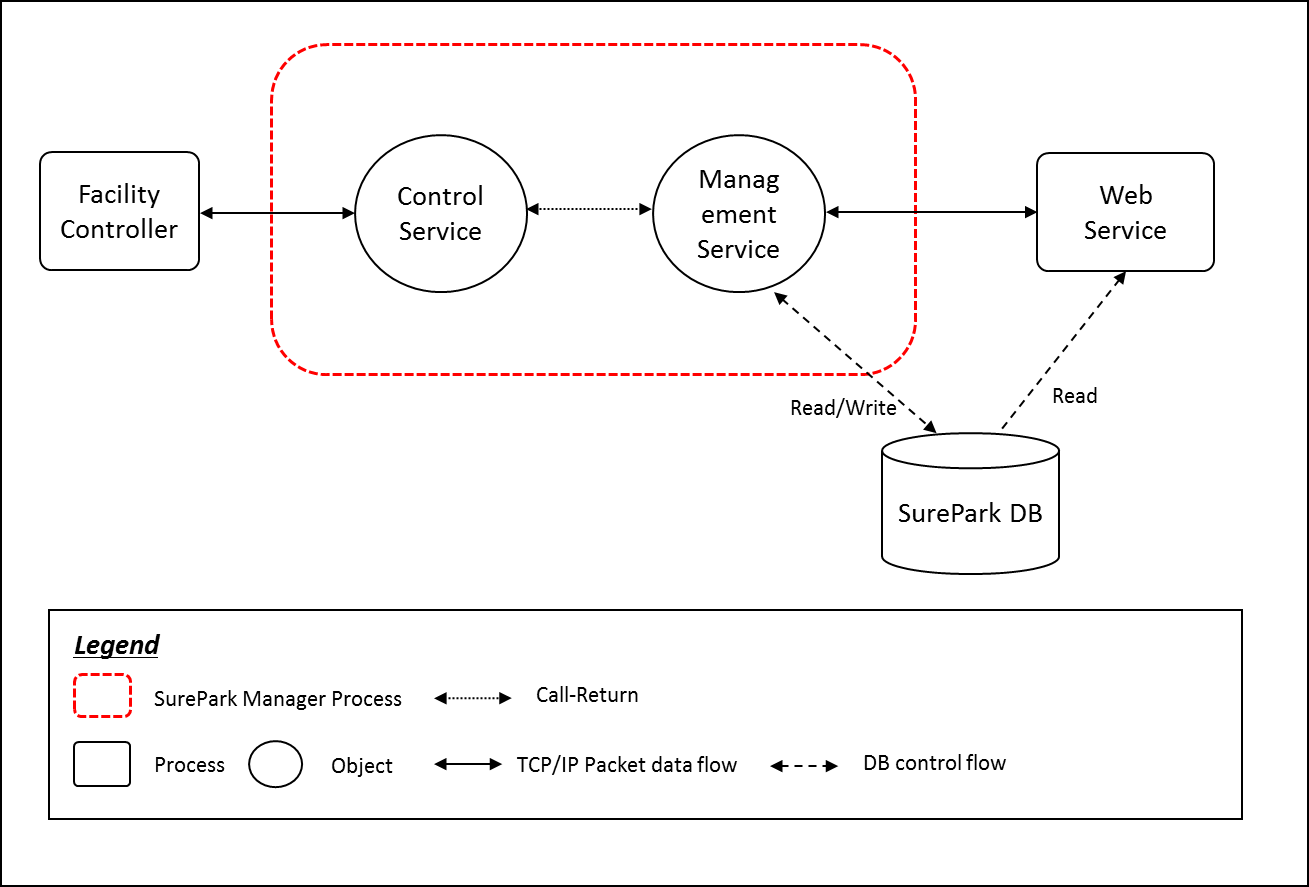
< Figure 5. Architectural patterns of SurePark System >



# **2nd Decomposition**

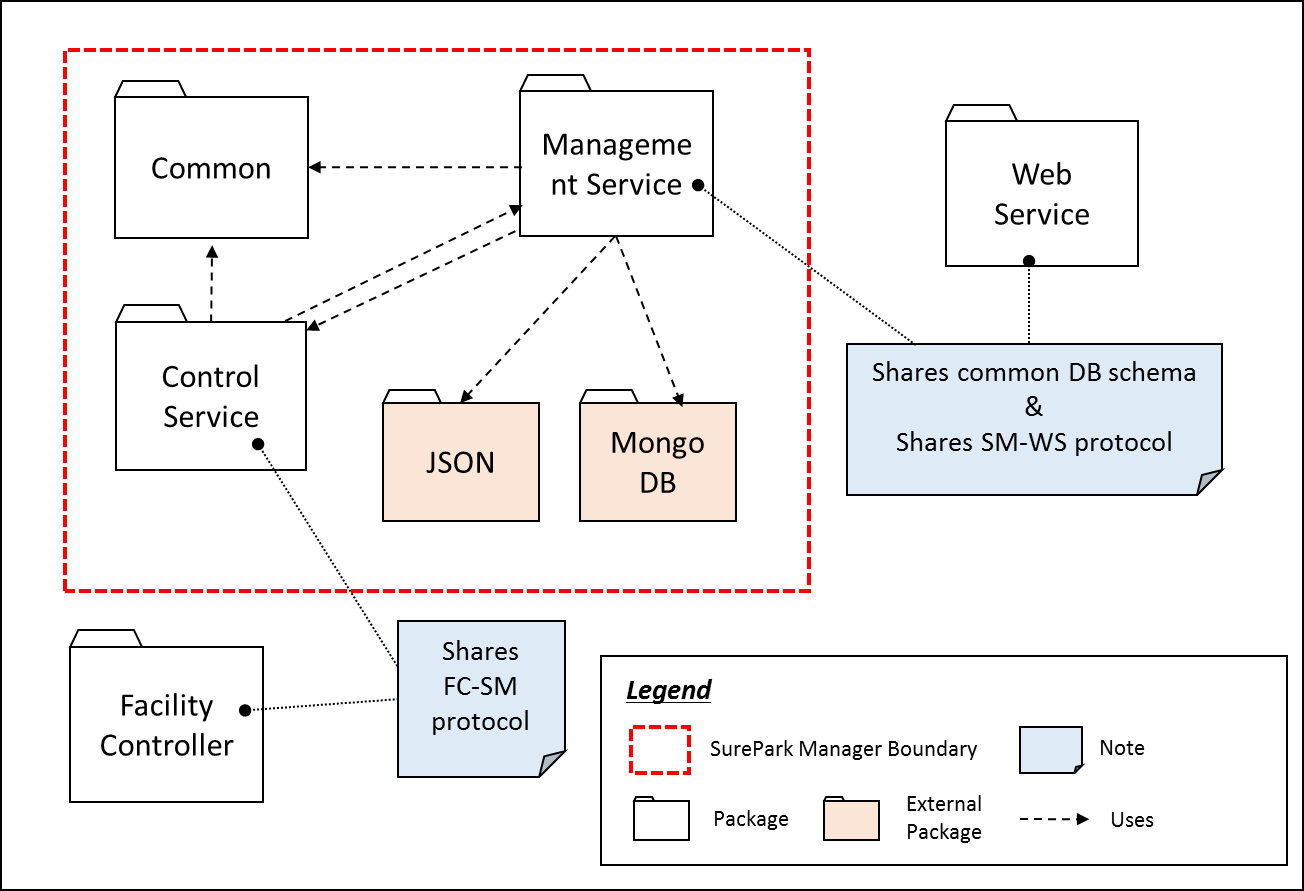
## **SurePark Manager**

### **Dynamic view of SurePark Manager**



< Figure6. Dynamic view of SurePark Manager >

### **Static view of SurePark Manager**



< Figure6. Static view of SurePark Manager >

### **Entity Catalog**

|  |  |
| --- | --- |
| **Entity** | **Description** |
| Common | Common package includes garage and reservation information. Control service and Management service refers common parts. |
| Control Service | Control Service have a responsibility of managing facilities. It try to connect/disconnect facilities following garage information. And it send/receive packets for controlling facilities. |
| Management Service | Management Service have a responsibility of managing DB transaction and it also have a responsibility of managing Web Service program. |

### **Rationale**

The SurePark Manager has two major responsibilities. One thing is handling Facility Controllers and the other is handling Web Service and DB. We’ve divided it ‘Control service’ and ‘Management service’ for increasing cohesion. Control service and Management service call each interface class for reducing coupling (QA08).

## **Facility Controller**

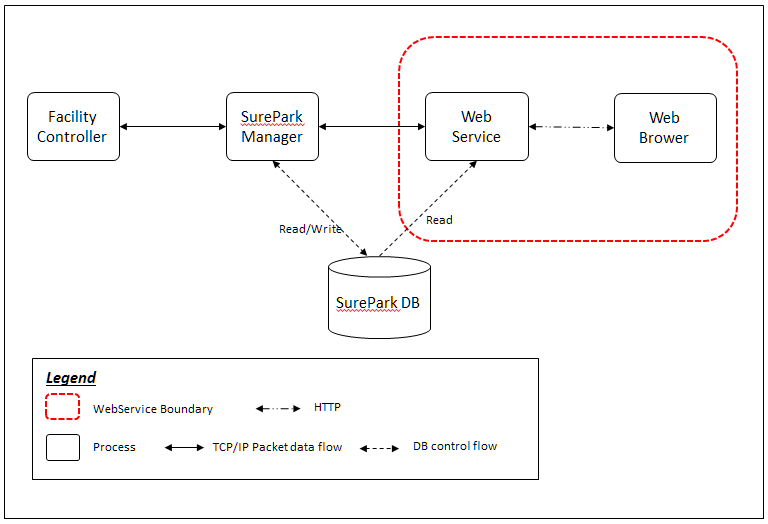


|  |  |
| --- | --- |
| **Modifiability(QA08)** | **Perspective: Static** |
| Architectural Pattern | Layered |
| Rationale | we’ve divided that use interfaces between DeviceControl/Observer and Devices.  So, We reduce dependency. Finally, Developers can design scale up/out the system easy. |

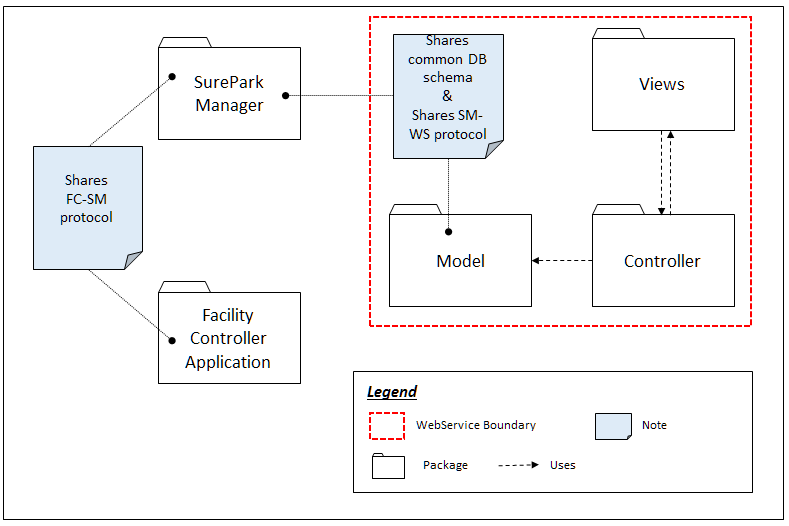
|  |  |
| --- | --- |
| **Entity** | **Description** |
| Comm | Communication module between surepark manager and facility controller |
| Device Control | Module for Non feed/back device control(exit/entry gate LED, exit/entry servo on/off) |
| Observer | This module observes state of devices.  When the state changed, then send message to superpark manager |
| Non f/b device interface | Set states of exit/entry gate LED, exit/entry servo and stall LEDs by deviceControl |
| f/b device interface | Get states of stallsensors and exit/entry sensors when observer request |
| Non f/b device | Control ON/OFF or OPEN/CLOSE(exit/entry gate LED, exit/entry servo and stall LEDs) |
| f/b device | Sensing value of stallsensors and exit/entry sensors |

## **Web Service**

### **Dynamic view of Web Service**



### **Static view of Web Service**



### **Entity Catalog**

|  |  |
| --- | --- |
| **Entity** | **Description** |
| Views | The View is the representation of the user interface. Views are implemented using html5 and css that renders an HTML page. |
| Controller | Controller handles the incoming HTTP request and route it if necessary.  The Controller receives user requests, and translates them into actions that the Model should take. Then it selects the appropriate View to handle the response. |
| Model | Model contains the transaction with other process which are DB read/write and communication with SureParkManager process. |

### **Rationale**

## **Facility Controller(FC) to SurePark Manager(SM)**

### **How to check if Facility Controller is alive**

#### Dynamic behavior of FC-SM packet



<figure X,

#### Rationale

We need to check the status of the Facility Controller. The SurePark Manager communicate with the Facility Controller through TCP/IP. Communication could be failed because of the many reasons. We’ve considered heartbeat tactic and ping-echo tactic to detect this kinds of faults. Ping-echo sends packets each other but heartbeat sends a packet one way. Since the SurePark Manager has to control several Facility Controllers, we’d better reduce network traffic. So we choose heartbeat tactic for availability (QA02).

## **SurePark Manager to Web Service**

### **How to resist attack from an attacker?**



#### Rationale

SurePark Manager and WebService use MongoDB for database. It supports a number of authentication mechanisms that clients can use to verify their identity. Only verified user can access database.

# **Detail Design**

## **FC-SM packet**

### **Packet Structure**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Start Symbol** | **Facility Id** | **Code** | **Value** | **End Symbol** |
| 1byte($) | 4byte | 1byte | Variable length | 1byte(\n) |

* Start Symbol: Start point of valid packet.
* Facility Id: Assigned the facility Id.
* Code: Indicate what kind of packet is. I means “Information”. S means “Slot Status”. E means “Entry Gate”. X means “Exit Gate”, L means “LED”.
* Value: It depends on “Code”. Please refer to “detailed packet scenario”.
* End Symbol: End point of valid packet.

### **Detailed Packet Scenario**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information | After connection, the SurePark Manager has to send information to the Facility Controller. The facility controller has to control slot number by this information. Max slot number is 10 by technical constraint.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | $ | Facility Id | I | Slot number | \n |   Ex) $1001I4\n (Garage 1001 consists of 4 stalls.) |
| Heartbeat | Basically, Facility controller send heartbeat packet at every 10 seconds. Timer for which heartbeat restarts after sending all packet. For example, if the facility controller send a slot status packet, it reset the timer for heartbeat and recount that time.  The SurePark manager recognizes if the facility controller is alive through all received packets.   |  |  |  | | --- | --- | --- | | $ | Facility Id | \n |   Ex) $1001\n |
| Open Entry Gate | This packet request to open entry gate to facility controller.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | $ | Facility Id | E | 1 | \n |   Ex) $1001E1\n (Garage 1001 has to open the entry gate.) |
| Turn on LED | This packet request to open entry gate to facility controller. Slot index designates 0 ~ Slot Number-1.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | $ | Facility Id | L | Slot index | \n |   Ex) $1001L2\n (Garage 1001 has to turn on the LED No.2.) |
| Slot Status | Whenever a slot status has changed, facility controller send this packet to SurePark manager. Slot index depends on slot number of information packet.   * Value 0: slot is opened. * Value 1: slot is occupied. * Value 2: slot is broken.  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | $ | Facility Id | S | S0 | S1 | … | SN | \n |   Ex) $1001S1001\n (Slot 0 and 3 are occupied.) |
| Exit Gate | After a car gets out of garage, the facility controller send this packet to SurePark manager. If a car is just going out of garage, not parked, the facility controller send it with value 2.   * Value 1: A car which pared at garage went out of garage. * Value 2: A car which is not parked at garage went out.  |  |  |  |  |  | | --- | --- | --- | --- | --- | | $ | Facility Id | X | 1 | \n |   Ex) $1001X1\n (A car which pared at garage went out of garage.)  Ex) $1001X2\n (A car which is not parked at garage went out.) |

## **SM-WS packet**

Communication between SureParkManger and WebService use SMWS packet. The packet structure is used JSON for sending, and the all packet data is encapsulated to string. The communication use request-response mechanism. WebService request services, and SureParkManager do services and response services.

And the communication use notify mechanism also. SureParkManager notify events to WebService, but no response for this mechanism.

1. Request packet structure

|  |  |
| --- | --- |
| **Request command** | **Data** |
| String | JSON Object |
| JSON Object | |

1. Response packet structure

|  |  |
| --- | --- |
| **Response command** | **Acknowledge** |
| String | String |
| JSON Object | |

1. Notify packet structure

|  |  |
| --- | --- |
| **Request command** | **Data** |
| String | JSON Object |
| JSON Object | |

1. Commands

|  |  |
| --- | --- |
| **Command** | **Data** |
| newGarage | When new garage is installed, Web Service add to the garage’s information to SureParkManager. SureParkManager update garages database.  - Request packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | garageName | String | Save garage name | | garageNumber | String | garage ID | | slotNum | String | Number of parking slot | | slotStatus | JSON Object | status of the each parking slot | | gracePeriod | String | gracePeriod value | | parkingFee | String | Parking fee per hour | | garageIP | String | garage IP address | | isAvailable | String | If this garage available |   Ex) {"newGarage": {"garageName":"AAAAA", "garageId":”1001”, "slotNumber":”4”, "slotStatus":["Open","Open","Open","Open"],  "gracePeriod":"90", "parkingFee":"30", "garageIP":"127.0.0.1", "isAvailable":true}}  - Response packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Data** | | Acknowledge | String | “OK” or “FAIL” |   Ex) {"newGarage": "OK"} or {"newGarage": "FAIL"} |
| newReservation | When a driver makes a reservation, WebService send new reservation to SureParkManger. SureParkManager update reservations database.  - Request packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | userID | String | user id who are made reservation | | reservationTime | String | reservation date/time | | cardInfo | String | card information | | confirmInformation | String | confirmation number | | gracePeriod | String | gracePeriod value | | parkingFee | String | parking fee per hour | | usingGarage | String | garage name to park |   {"newReservation": {"userID":"jack", “reservationTime”:”2016-06-22T14:30:00.000Z”, "cardInfo":"1111-\*\*\*\*-\*\*\*\*-4444", "confirmInformation":"A1234", "gracePeriod":"90", "parkingFee":"30", "usingGarage":"Sure-Park"}}  - Response packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Data** | | Acknowledge | String | “OK” or “FAIL” |   Ex) {" newReservation ": "OK"} or {" newReservation ":"FAIL"} |
| cancelReservation | When a driver cancels a reservation, WebService send cancel reservation to SureParkManger. SureParkManager update reservations database.  - Request packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | userID | String | user id who want cancel reservation | | confirmInformation | String | confirmation number | | usingGarage | String | garage name to park |   Ex) {"cancelReservation": {"userID":"jack", "usingGarage":"Sure-Park", "confirmInformation":"A1234"}}  - Response packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Data** | | Acknowledge | String | “OK” or “FAIL” |   Ex) {" cancelReservation ": "OK"} or {" cancelReservation ":"FAIL"} |
| newUser | When new user signed up, Web Service send new user sign up information to SureParkManager. SureParkManager update users database.  - Request packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | userID | String | user ID | | userPassword | String | user Password | | userType | String | classify driver, attentant and owner | | userName | String | user name | | userEmail | String | user email address | | displayName | String | user nick name |   Ex) {"newUser": {"userID":"myID", "userPassword":"myPassword", "userType":"user", "userName":"myName", "userEmail": “myEmail@email.com", "displayName":"MY NAME"}}  - Response packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Data** | | Acknowledge | String | “OK” or “FAIL” |   Ex) {" newUser ": "OK"} or {" newUser ":"FAIL"} |
| parkingCar | When a driver who has reservation is show up with confirm information, Web Service send information to SureParkManager. SureParkManager update reservation database, and control facilities.  - Request   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | userID | String | user id who are made reservation | | usingGarage | String | confirmation number | | confirmInformation | String | garage name to park |   Ex) {"parkingCar": {"userID":"myID", "usingGarage":"Sure-Park", "confirmInformation":"A1234"}}  - Response packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Data** | | Acknowledge | String | “OK” or “FAIL” |   Ex) {" parkingCar ": "OK"} or {" parkingCar ":"FAIL"} |
| updateSlotStatus | When slot status is changed, Web Service need to update slot status. SureParkManger notify updateSlotStatus event to WebService.  - Notify packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | garageName | String | garage name to update | | garageNumber | String | garage number to update |   Ex) {"updateSlotStatus":{"garageName":"Sure-Park", "garageNumber":"1001"}} |
| wrongParking | When facility controller detect a driver parked wrong slot, , Web Service need to update slot status. SureParkManger notify wrongParking event to WebService.  - Notify packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | garageName | String | garage name which has wrong parking | | garageNumber | String | garage number which has wrong parking |   Ex) {" wrongParking ":{"garageName":"Sure-Park", "garageNumber":"1001"}} |
| detectFailure | When SureParkManager detect facilities failure, Web Service need to notice failure to attendants. SureParkManger notify detectFailure event to WebService.  - Notify packet   |  |  |  | | --- | --- | --- | | **Data Field** | **Data type** | **Description** | | garageName | String | garage name which detect failure | | garageNumber | String | garage number which detect failure |   Ex) {" detectFailure ":{"garageName":"Sure-Park", "garageNumber":"1001"}} |

## **SM-WS DB Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **collection** | **name** | **type** | **description** |
| **users** | userID | string | Save user ID |
| userPassword | string | Save user Password |
| userType | string | classify driver, attentant and owner |
| userName | string | Save user name |
| userEmail | string | Save user email address |
| registerationTime | Date | Save user registeration date/time |
| displayName | string | Save user nick name |
| **garages** | garageName | string | Save garage name |
| garageNumber | integer | garage ID |
| slotNumber | integer | Number of parking slot |
| slotStatus | JSON Object | status of the each parking slot |
| updateTime | JSON Object | Update time of parking slot |
| gracePeriod | integer | gracePeriod value |
| parkingFee | integer | Parking fee per hour |
| garageIP | string | garage IP address |
| isAvailable | Boolean | If this garage available |
| **reservations** | userID | string | user id who are made reservation |
| reservationTime | Date | reservation date/time |
| cardInfo | string | card information |
| confirmInformation | string | confirmation number |
| gracePeriod | integer | gracePeriod value |
| reservationStatus | string | reservation status |
| parkingFee | integer | parking fee per hour |
| usingGarage | string | garage name to park |
| usingGarageNumber | integer | garage number tp park |
| usingSlot | string | slot numer to park |
| parkingTime | Date | parking date/time |
| leaveTime | Date | leave date/time |
| chargingFee | integer | charge fee |

## **SurePark Manager Detailed Design**

* + 1. Static Perspective – Class diagram(notation UML2.0)
       1. Management Service



Data Access

Interface

Business Logic

4.4.1.1.1 Rationale

Define the relation among classes. To emphasis modifiability, layer pattern is used. Each layer’s interface use java interface and the reason is dependency inversion principle(DIP).

4.4.1.1.2 Responsibility

|  |  |
| --- | --- |
| **Layer name** | **Responsibility** |
| Interface | This layer communicate with external thread, package and process.  Delivers request from Web service to Facility controller, and notify from Facility controller.  Delivers information to business logic. |
| Business Logic | This layer analysis request, and delivers information to Data Access.  Check reserved driver is show up or not. If driver no show, this layer update database using Data Access. |
| Data Access | This layer access database with authentication.  Information from Business Logic transact to database. |

* + 1. Dynamic perspective
       1. Sequence Diagram (UC01: Reserve parking spaces)



* + - 1. Sequence Diagram (UC02: Show up scenario)



* + - 1. Sequence Diagram (UC04 : Get out the garage and charge scenario)



* + - 1. Sequence Diagram (UC06: Monitoring scenario for attendants)



## **Facility Controller Detailed Design**





## **Web Service Detailed Design**







# **Future Works**

- Add new algorithm for statistics (plugin type)

- SurePark Manager – connection fail scenario

- 몽고DB를 리플리케이션 (for availability)

# **10. Lesson Learned**

- COTS 이용 시 적용 및 수정의 어려움을 느낄 수 있었음.

- TOGGL 등의 개발 툴 사용법을 배웠음.

- perspective의 분리가 감명 깊었음.

- Interface의 중요성.

- Iteration의 중요성

- 제일 critical한 부분을 최대한 빨리 확인해야 한다. 라는 부분의 중요성.

-