|  |  |
| --- | --- |
| **­** | **MINISTRY OF EDUCATION AND TRAINING** |

**FPT UNIVERSITY**

|  |  |
| --- | --- |
| **Report 4** | |
| **Parking Guidance System Solution** | |
| **Group 1** | |
| **Group members** | Trần Nguyễn Minh Trung – Team Leader – SE61496  Bùi Phú Hiệp – Team Member – SE61438  Nguyễn Đỗ Phương Huy – Team Member – SE61358 |
| **Supervisor** | Nguyễn Đức Lợi |
| **Ext. Supervisor** | N/A |
| **Capstone Project Code** | PGSS |

- Ho Chi Minh City, Jan, 2017

*This page is intentionally left blank*

# Table of Contents

[Table of Contents 1](#_Toc475972823)

[List of Tables 2](#_Toc475972824)

[List of Figures 2](#_Toc475972825)

[Definitions, Acronyms and Abbreviations 2](#_Toc475972826)

[D. Software – Hardware Design Description 3](#_Toc475972827)

[1. Design Overview 3](#_Toc475972828)

[2. System Architectural Design 3](#_Toc475972829)

[2.1. Hardware Program Architecture Description 3](#_Toc475972830)

[2.2. Desktop Application Architecture Description 3](#_Toc475972831)

[3. Component Diagram 3](#_Toc475972832)

[4. Detailed Description 3](#_Toc475972833)

[4.1. Class Diagram 3](#_Toc475972834)

[4.1.1. Mobile App Class Diagram 3](#_Toc475972835)

[4.1.2. Web API Class Diagram 3](#_Toc475972836)

[4.1.3. MCU Class Diagram 3](#_Toc475972837)

[4.1.4. Parking Lot Class Diagram 3](#_Toc475972838)

[4.1.5. Information LED Display Class Diagram 3](#_Toc475972839)

[4.2. Class Diagram Explanation 3](#_Toc475972840)

[4.3. Interaction Diagram 3](#_Toc475972841)

[4.3.1. Flowchart Diagram (cho phần cứng) 3](#_Toc475972842)

[4.3.2. Activity Diagram (mobile app) 3](#_Toc475972843)

[4.3.3. Sequence Diagram (web api) 3](#_Toc475972844)

[5. Interface 3](#_Toc475972845)

[5.1. Component Interface 3](#_Toc475972846)

[5.2. User Interface Design 3](#_Toc475972847)

[6. Database Design 3](#_Toc475972848)

[6.1. Logical Diagram 3](#_Toc475972849)

[6.2. Data Dictionary 3](#_Toc475972850)

[7. Algorithms 3](#_Toc475972851)

# List of Tables

[Table 1: Definitions, Acronyms and Abbreviations 3](#_Toc475466552)

[Table 2: Raspberry Pi 3 - Specification 6](#_Toc475466553)

[Table 3: Arduino Nano - Specification 8](#_Toc475466554)

[Table 4: RF module nRF24L01+ - Specification 10](#_Toc475466555)

# List of Figures

[Figure 1: PGSS Block Diagram 5](#_Toc475466537)

[Figure 2: Raspberry Pi 3 6](#_Toc475466538)

[Figure 3: Arduino Nano 7](#_Toc475466539)

[Figure 4: Compass Module 3-Axis HMC5883L 8](#_Toc475466540)

[Figure 5: RF module nRF24L01+ 9](#_Toc475466541)

[Figure 6: RF module nRF24L01+ - Specification 10](#_Toc475466542)

[Figure 7: 7-segment LED Display 11](file:///C:\Users\TrungTNM\Documents\-FPT-CAPSTONE-PGSS\Common\Reports\Report%203.docx#_Toc475466543)

[Figure 8: TPIC6B595 Power Logic 8-Bit Shift Register 12](#_Toc475466544)

[Figure 9: TPIC6B595 Pin outs 13](#_Toc475466545)

[Figure 10: RGB LED common anode 14](#_Toc475466546)

[Figure 11: RGB LED common anode pin-out 15](#_Toc475466547)

[Figure 12: TIP122 Transistor 16](#_Toc475466548)

[Figure 13: Overview use case diagram 17](#_Toc475466549)

[Figure 14: Manager Use case diagram 18](#_Toc475466550)

[Figure 15: Conceptual Diagram 27](#_Toc475466551)

# Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Name** | **Definition** |
| PGS | Parking Guidance System |
| Parking area | An area set aside for parking vehicles, aircraft, etc. |
| Parking lot | A place inside parking area that provide space for one vehicle |
| IoT | Internet of Things |
| CCU | Central Control Unit |

Table : Definitions, Acronyms and Abbreviations

# D. Software – Hardware Design Description

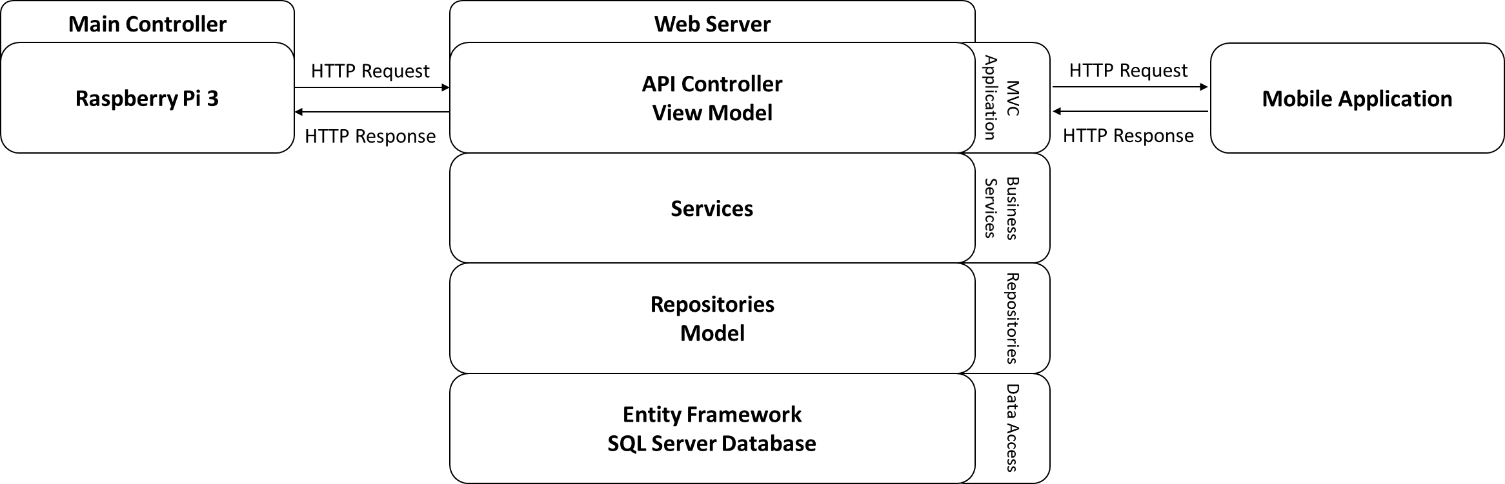
## Design Overview

## System Architectural Design

### Hardware Program Architecture Description

### Mobile Application Architecture Description

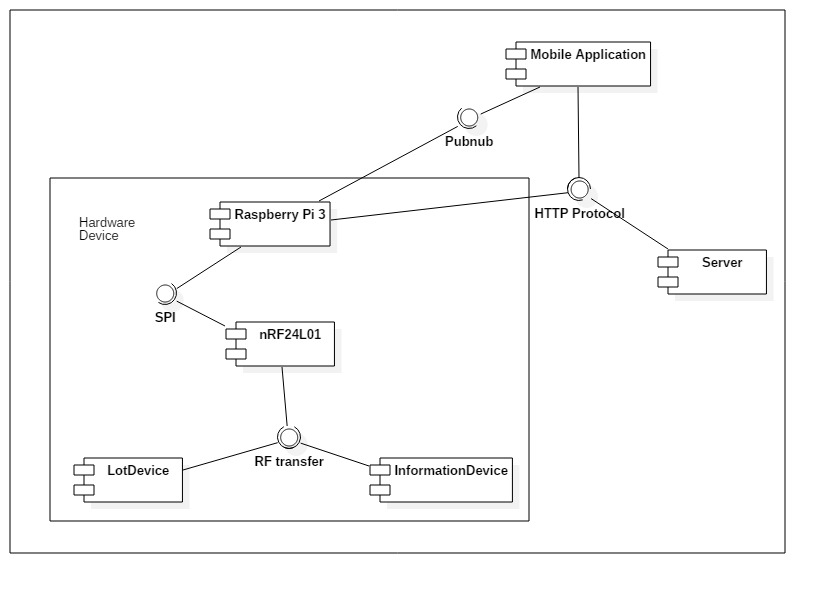
### Web API Architecture Description



**Figure 1: Hardware - Software Connection Architecture Description**

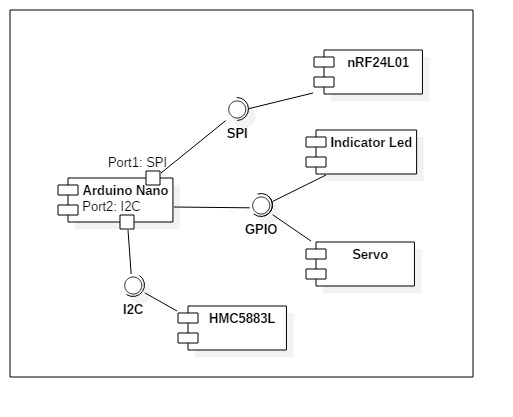
## Component Diagram

### General Component Diagram



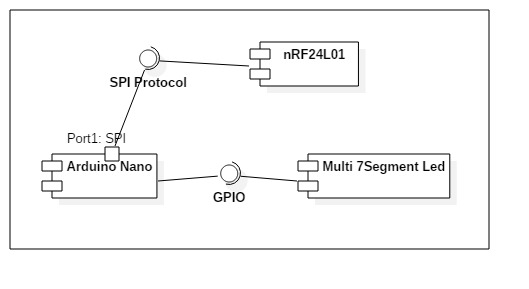
**Figure 2: General Component Diagram**

### Lot Device Diagram



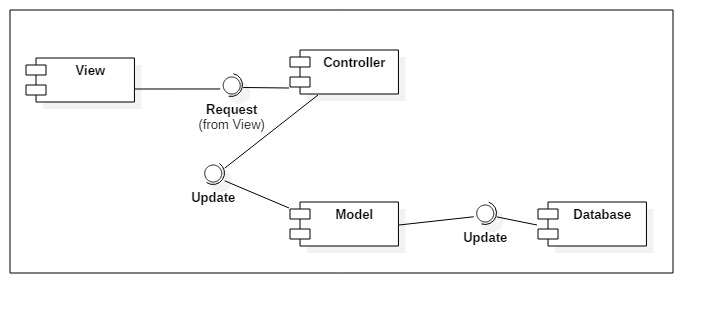
**Figure 3: Lot Device Diagram**

### Information Device Diagram



**Figure 4: Information Device Diagram**

### Server Diagram



**Figure 5: Server Diagram**

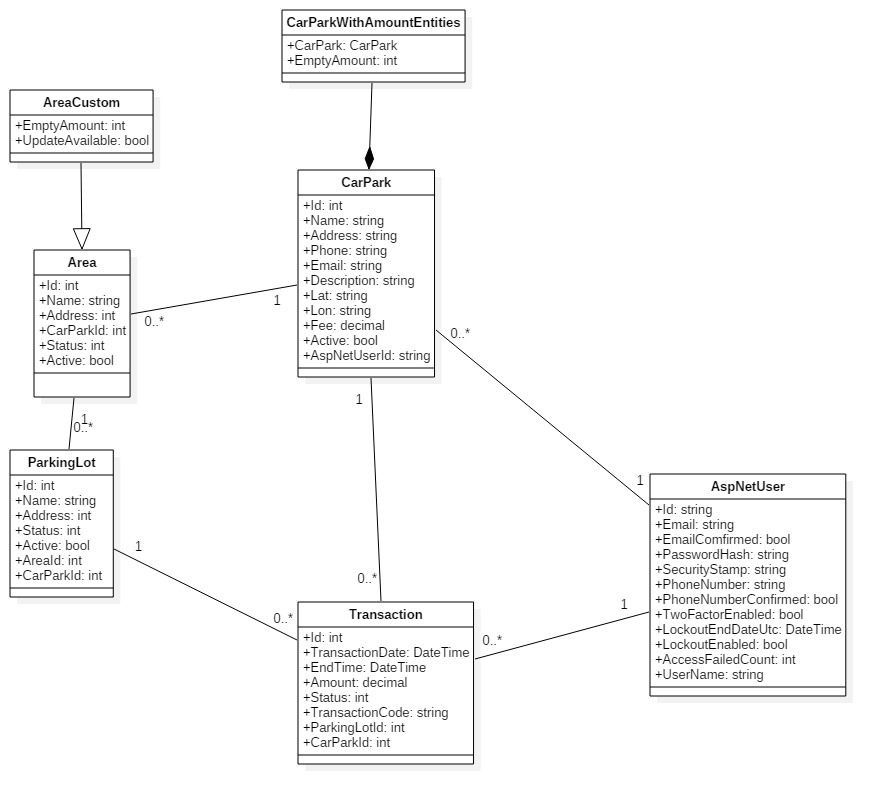
### Component Dictionary

|  |  |
| --- | --- |
| Component Dictionary: Describes components | |
| **Mobile Application** | The application for user on android devices |
| **Server** | The Web API server, which make the interaction between the Raspberry Pi and the Mobile Application. It also save information to the database. |
| **Raspberry Pi 3** | The main controller in the hardware system that control all Lot Device and Information Device in each car park. |
| **Lot Device** | The device in each parking lot that control the indicator led, servo and use the magnetometer sensor to detect car. |
| **Information Device** | The number led board that show the number of empty lot in each area and number of empty lot in car park. |
| **nRF24L01** | The RF module, which help main controller, lot device and information device interact with each other. |
| **Arduino Nano** | The controller in lot device and information device. |
| **Indicator Led** | The RGB led to show the status of parking lot (empty, in used, reserved, disabled). |
| **Servo** | The barrier to prevent other car get in the parking lot when it is reserved. |
| **HMC5883L** | The magnetometer sensor to realize if the metal is near. |
| **Pubnub** | The third party API that help real time interaction. |
| **HTTP Protocol** | A transfer protocol to interact between server, mobile application and Raspberry Pi 3. |
| **SPI** | Synchronous serial communication interface used for short distance communication. |
| **I2C** | The inter-intergrated circuit to communicate between arduino and magnetometer sensor. |
| **GPIO** | The communication between Arduino and led, servo, … |
| **Multi 7Segment Led** | The 7 segment led to indicate number. |
| **View** | The JSON result of the API controller. |
| **Controller** | The API controller in the server. |
| **Model** | The entity model. |
| **Database** | The collection of data. |

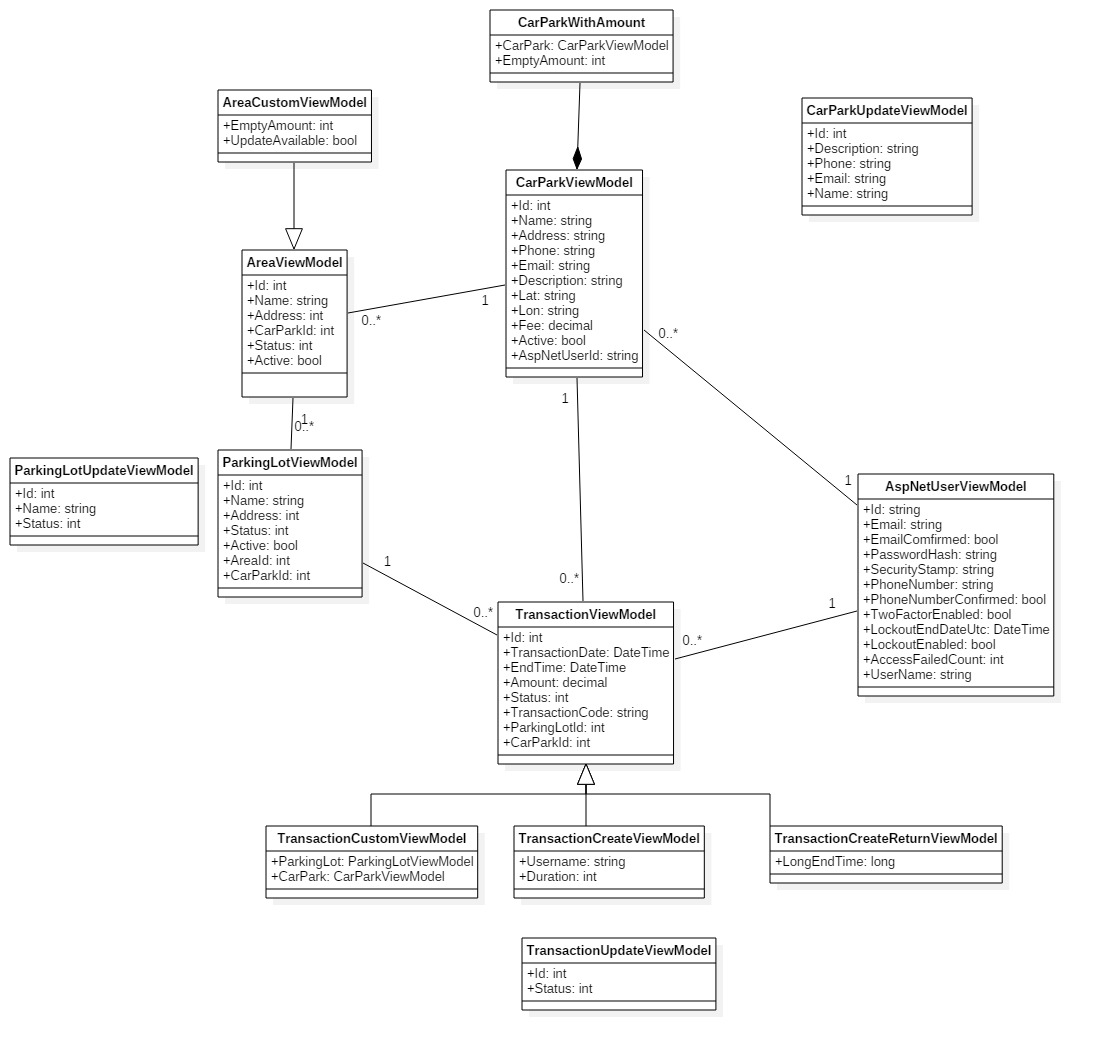
## Detailed Description

### Web API Detailed Description

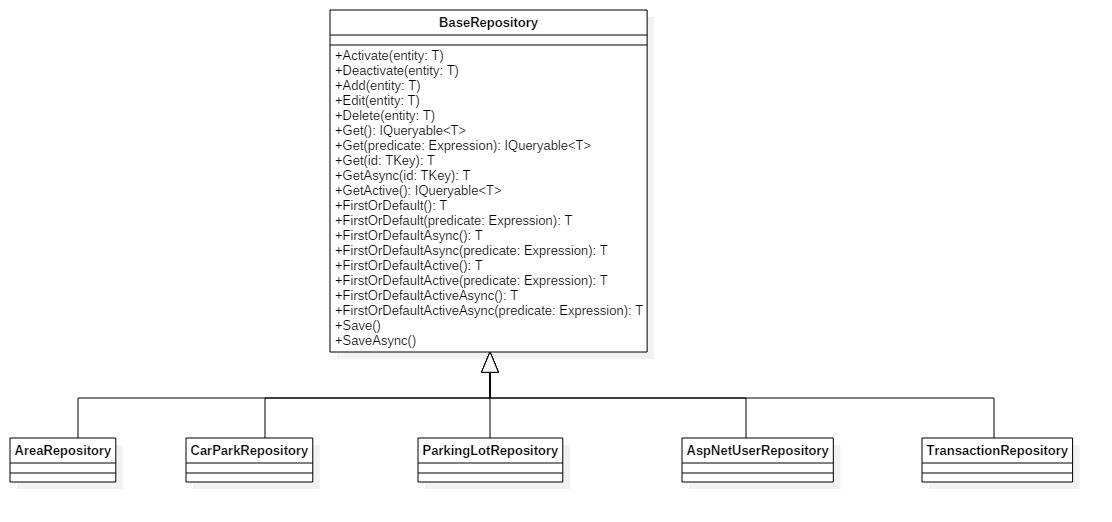
#### Class Diagram



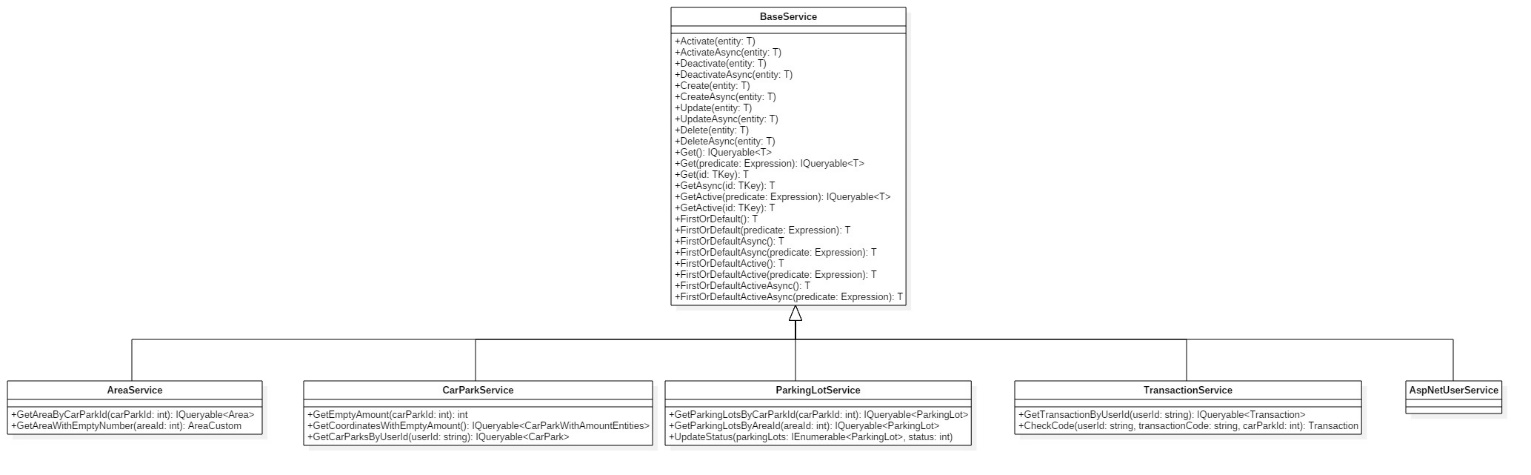
**Figure 6: Model Class Diagram**



**Figure 7: Vỉew Model Class Diagram**



**Figure 8: Repository Class Diagram**



**Figure 9: Service Class Diagram**

#### Class Diagram Explanation

##### Model Class Diagram

**CarPark**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each car park |
| Name | string | public | Name of car park |
| Address | string | public | Address of carpark |
| Phone | string | public | Phone number of car park |
| Email | string | public | Email of car park’s owner |
| Description | string | public | Information of car park |
| Lat | string | public | Latitude of a car park |
| Lon | string | public | Longitude of a car park |
| Fee | decimal | public | Amount of money each hour |
| Active | bool | public | Value show that the car park is deleted or not |
| AspNetUserId | string | public | Id of the owner |

**CarParkWithAmountEntities**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| CarPark | CarPark | public | The car park entity |
| EmptyAmount | int | public | The number of empty lot in car park |

**Area**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each area |
| Name | string | public | Name of area |
| Address | int | public | Bit address of a area |
| CarParkId | int | public | Id of car park which hold area |
| Status | int | public | The number indicate the status of area |
| Active | bit | public | Value show that the area is deleted or not |

**AreaCustom**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| EmptyAmount | int | public | The number of empty lot in area |
| UpdateAvailable | bool | public | The boolean value check that area is available for update |

**ParkingLot**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each parking lot |
| Name | string | public | Name of parking lot |
| Address | int | public | Bit address of a parking lot |
| Status | int | public | The number indicate the status of parking lot |
| Active | bit | public | Value show that the parking lot is deleted or not |
| AreaId | int | public | Id of area which hold parking lot |
| CarParkId | int | public | Id of car park which hold parking lot |

**Transaction**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each transaction |
| TransactionDate | string | public | The time customer book the parking lot |
| EndTime | int | public | End of the booking time |
| Amount | int | public | The total fee of transaction |
| Status | bit | public | The number indicate status of the transaction |
| TransactionCode | string | public | The code for the user to enter in the car park to unlock the booked parking lot |
| ParkingLotId | int | public | Id of area which hold parking lot |
| CarParkId | int | public | Id of car park which hold parking lot |

**AspNetUser**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | string | public | Unique Id for each user |
| Email | string | public | Email of the user |
| EmailConfirm | bool | public | Check if the user confirm the email |
| PasswordHash | string | public | Hash of the user password, don’t save exact password |
| SecurityStamp | string | public |  |
| PhoneNumber | string | public | The number of the user |
| PhoneNumberConfirmed | bool | public | Check if the number is confirmed |
| TwoFactorEnabled | bool | public |  |
| LockoutEndDateUtc | DateTime | public |  |
| LockoutEnabled | bool | public |  |
| AccessFailedCount | int | public | The number of time access failed |
| UserName | string | public | Username of the user |

##### View Model Class Diagram

**CarParkViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each car park |
| Name | string | public | Name of car park |
| Address | string | public | Address of carpark |
| Phone | string | public | Phone number of car park |
| Email | string | public | Email of car park’s owner |
| Description | string | public | Information of car park |
| Lat | string | public | Latitude of a car park |
| Lon | string | public | Longitude of a car park |
| Fee | decimal | public | Amount of money each hour |
| Active | bool | public | Value show that the car park is deleted or not |
| AspNetUserId | string | public | Id of the owner |

**CarParkWithAmount**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| CarPark | CarParkViewModel | public | The car park entity |
| EmptyAmount | int | public | The number of empty lot in car park |

**CarParkUpdateViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Id of update car park |
| Description | string | public | New description |
| Phone | string | public | New phone number |
| Email | string | public | New email |
| Name | string | public | New name |

**AreaViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each area |
| Name | string | public | Name of area |
| Address | int | public | Bit address of a area |
| CarParkId | int | public | Id of car park which hold area |
| Status | int | public | The number indicate the status of area |
| Active | bit | public | Value show that the area is deleted or not |

**AreaCustomViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| EmptyAmount | int | public | The number of empty lot in area |
| UpdateAvailable | bool | public | The boolean value check that area is available for update |

**ParkingLotViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each parking lot |
| Name | string | public | Name of parking lot |
| Address | int | public | Bit address of a parking lot |
| Status | int | public | The number indicate the status of parking lot |
| Active | bit | public | Value show that the parking lot is deleted or not |
| AreaId | int | public | Id of area which hold parking lot |
| CarParkId | int | public | Id of car park which hold parking lot |

**ParkingLotUpdateViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each parking lot |
| Name | string | public | New name if change |
| Status | int | public | New status if change |

**Transaction**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Unique Id for each transaction |
| TransactionDate | string | public | The time customer book the parking lot |
| EndTime | int | public | End of the booking time |
| Amount | int | public | The total fee of transaction |
| Status | bit | public | The number indicate status of the transaction |
| TransactionCode | string | public | The code for the user to enter in the car park to unlock the booked parking lot |
| ParkingLotId | int | public | Id of area which hold parking lot |
| CarParkId | int | public | Id of car park which hold parking lot |

**TransactionCustomViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| ParkingLot | ParkingLotViewModel | public | ParkingLot’s model |
| CarPark | CarParkViewModel | public | CarPark, where transaction is booked |

**TransactionCreateViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Username | string | public | Username of the customer |
| Duration | int | public | Time of booking |

**TransactionCreateReturnViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| LongEndTime | long | public | DateTime in number format |

**TransactionUpdateViewModel**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | int | public | Id of the transaction |
| Status | int | public | The integer indicate the status of transaction |

**AspNetUser**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | string | public | Unique Id for each user |
| Email | string | public | Email of the user |
| EmailConfirm | bool | public | Check if the user confirm the email |
| PasswordHash | string | public | Hash of the user password, don’t save exact password |
| SecurityStamp | string | public |  |
| PhoneNumber | string | public | The number of the user |
| PhoneNumberConfirmed | bool | public | Check if the number is confirmed |
| TwoFactorEnabled | bool | public |  |
| LockoutEndDateUtc | DateTime | public |  |
| LockoutEnabled | bool | public |  |
| AccessFailedCount | int | public | The number of time access failed |
| UserName | string | public | Username of the user |

##### Repository Class Diagram

**BaseRepository**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Type** | **Visibility** | **Description** |
| Activate | void | public | Set the active attribute of current model to True |
| Deactivate | void | public | Set the active attribute of current model to False |
| Add | void | public | Add a model to database |
| Edit | void | public | Edit a input model |
| Delete | void | public | Delete a input model in database |
| Get | IQueryable<T> | public | Get a list of model with condition in database |
| Get | T | public | Get a model by Id in database |
| GetAsync | T | public | Get a model by Id in database asynchronous |
| GetActive | IQueryable<T> | public | Get a list of model with condition in database asynchronous |
| FirstOrDefault | T | public | Get first model in a list with condition in database |
| FirstOrDefaultAsync | T | public | Get first model in a list with condition in database asynchronous |
| FirstOrDefaultActive | T | public | Get first model in a list of active item with condition in database |
| FirstOrDefaultActiveAsync | T | public | Get first model in a list of active item with condition in database asynchronous |
| Save | void | public | Save a current state of entity to database |
| SaveAsync | void | public | Save a current state of entity to database asynchronous |

##### Service Class Diagram

**Base Service**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Type** | **Visibility** | **Description** |
| Activate | void | public | Set the active attribute of current model to True |
| ActivateAsync | void | public | Set the active attribute of current model to True asynchronous |
| Deactivate | void | public | Set the active attribute of current model to False |
| DeactivateAsync |  |  | Set the active attribute of current model to False asynchronous |
| Create | void | public | Create a new item in database |
| CreateAsync | void | public | Create a new item in database asynchronous |
| Update | void | public | Update a item in database |
| UpdateAsync | void | public | Update an item in database asynchronous |
| Delete | void | public | Delete an item in database |
| DeleteAsync | void | public | Delete an item in database asynchronous |
| Get | IQueryable<T> | public | Get a list of model with condition in database |
| Get | T | public | Get a model by Id in database |
| GetAsync | T | public | Get a model by Id in database asynchronous |
| GetActive | IQueryable<T> | public | Get a list of model with condition in database asynchronous |
| GetActive | T | public | Get active model by Id in database if exist |
| FirstOrDefault | T | public | Get first model in a list with condition in database |
| FirstOrDefaultAsync | T | public | Get first model in a list with condition in database asynchronous |
| FirstOrDefaultActive | T | public | Get first model in a list of active item with condition in database |
| FirstOrDefaultActiveAsync | T | public | Get first model in a list of active item with condition in database asynchronous |

**AreaService**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Type** | **Visibility** | **Description** |
| GetAreaByCarParkId | IQueryable<Area> | public | Get the area with input car park id |
| GetAreaWithEmptyNumber | AreaCustom | public | Get the area with empty number of lot by area id |

**CarParkService**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Type** | **Visibility** | **Description** |
| GetEmptyAmount | int | public | Get the number of empty lot by car park id |
| GetCoordinatesWithEmptyAmount | IQueryable  <CarParkWithAmountEntities> | public | Get the list of car park with coordinate and number of empty lot |
| GetCarParksByUserId | IQueryable<CarPark> | public | Get the list of car park by AspNetUserId (owner of car park) |

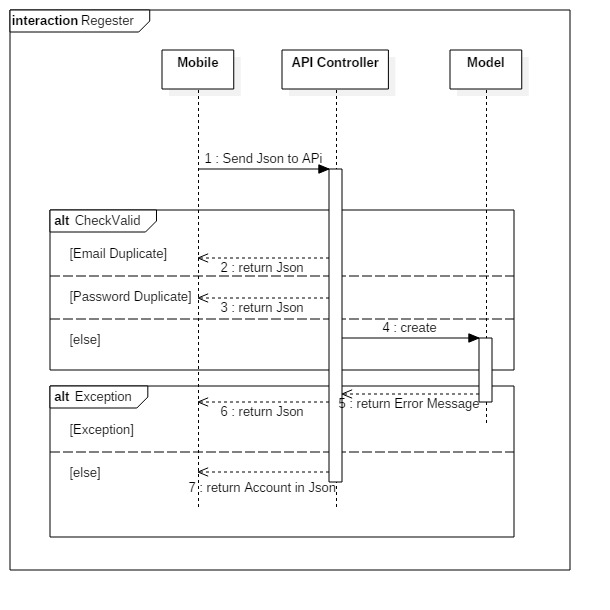
**ParkingLotService**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Type** | **Visibility** | **Description** |
| GetParkingLotsByCarParkId | IQueryable<ParkingLot> | public | Get the list of parking lot in the car park |
| GetParkingLotsByAreaId | IQueryable<ParkingLot> | public | Get the list of parking lot in the area |
| UpdateStatus | void | public | Change the status of a list parking lot |

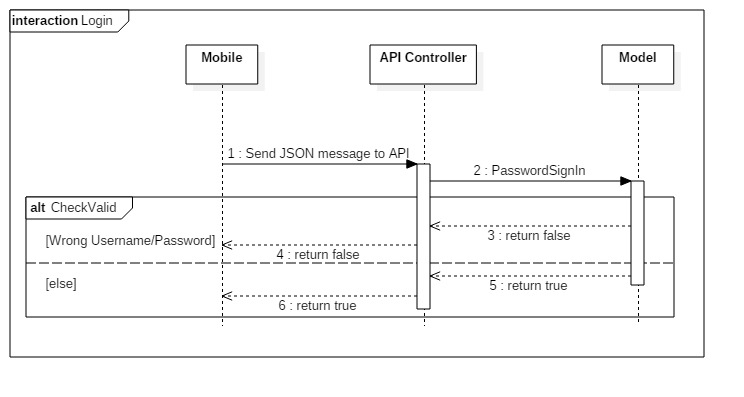
**Transaction Service**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Type** | **Visibility** | **Description** |
| GetTransactionByUserId | IQueryable<Transaction> | public | Get the list of transaction create by a user |
| CheckCode | Transaction | public | Get the transaction base on user, transaction code and car park |

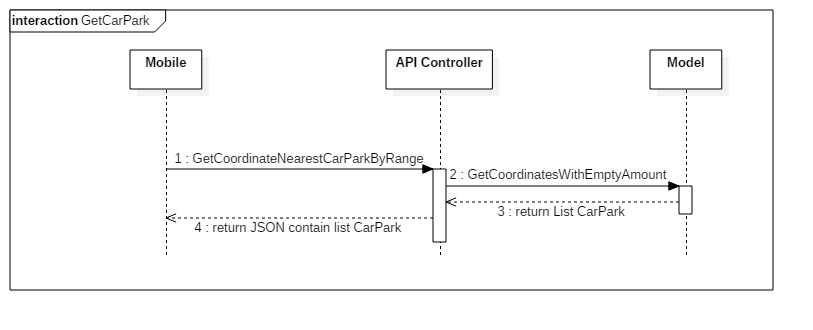
#### Interaction Diagram



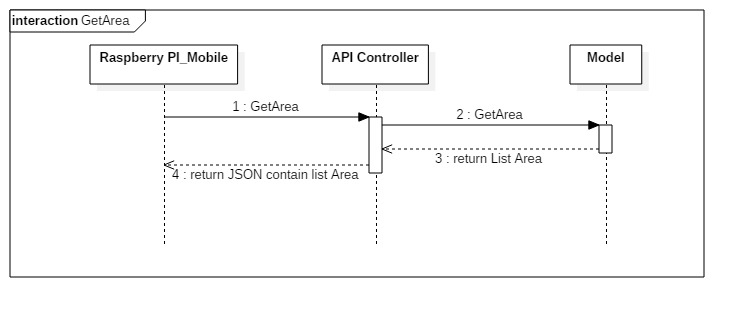
**Figure 10: Register Diagram**



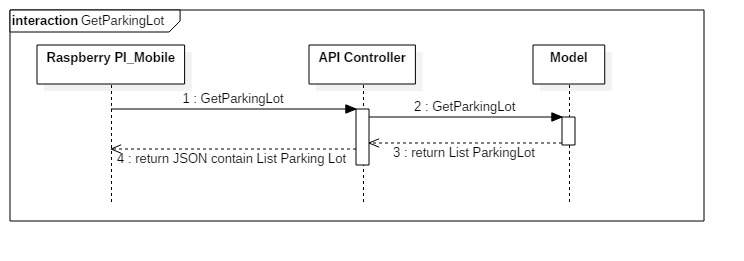
**Figure 11: Login Diagram**



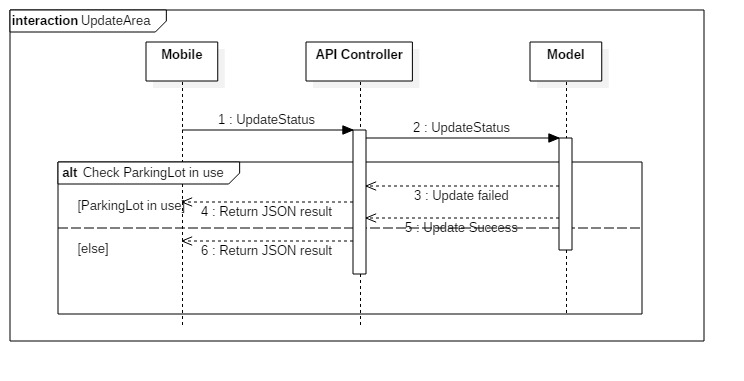
**Figure 12: GetCarPark Diagram**



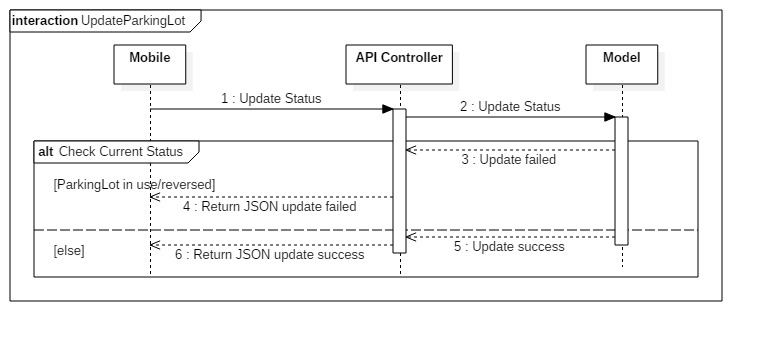
**Figure 13: GetArea Diagram**



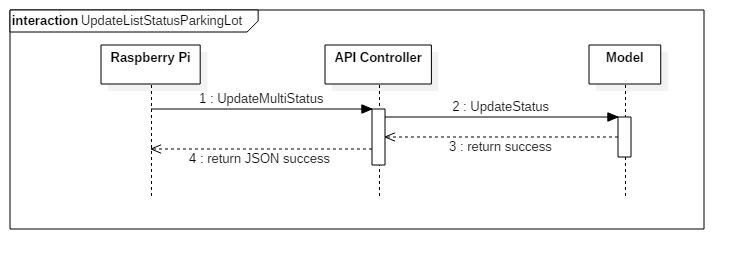
**Figure 14: GetParkingLot Diagram**



**Figure 15: UpdateArea Diagram**



**Figure 16: UpdateParkingLot Diagram**



**Figure 17: UpdateListStatusParkingLot Diagram**

### Mobile Detailed Description

#### Class Diagram

#### Class Diag+ram Explanation

#### Interaction Diagram

### Embedded Detailed Description

#### Class Diagram

#### Class Diagram Explanation

#### Interaction Diagram

### Hardware Detailed Description

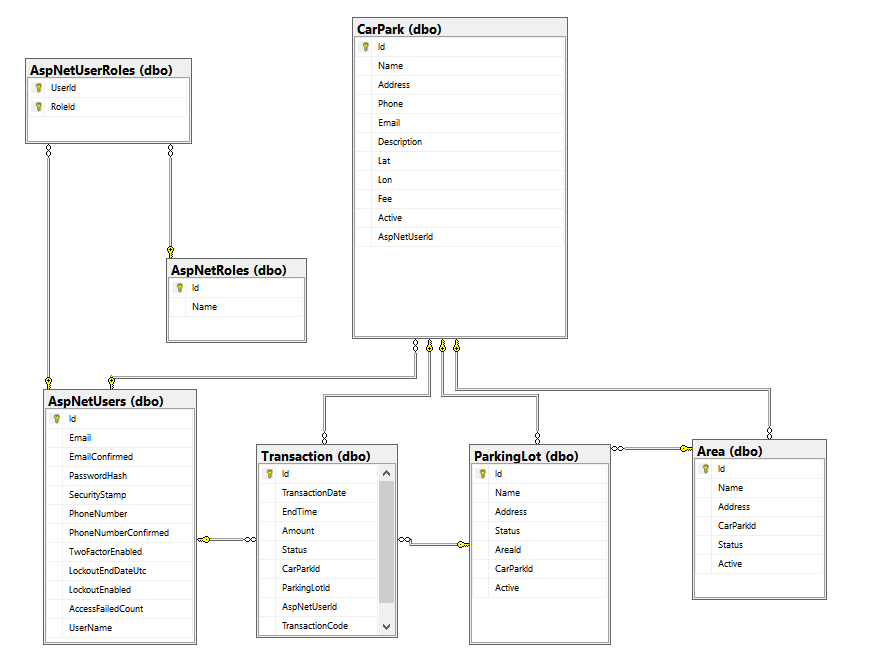
## Interface

### Component Interface

### User Interface Design

## Database Design

### Logical Diagram



### Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name** | **Column Name** | **Is Nullable** | **Data Type** |
| **Area** | Id | NO | int |
| Name | NO | nvarchar |
| Address | YES | int |
| CarParkId | NO | int |
| Status | NO | int |
| Active | NO | bit |
| **ParkingLot** | Id | NO | int |
| Name | NO | nvarchar |
| Address | NO | int |
| Status | NO | int |
| AreaId | YES | int |
| CarParkId | NO | int |
| Active | NO | bit |
| **Transaction** | Id | NO | int |
| TransactionDate | NO | datetime |
| EndTime | YES | datetime |
| Amount | NO | money |
| Status | NO | int |
| CarParkId | NO | int |
| ParkingLotId | NO | int |
| AspNetUserId | NO | nvarchar |
| TransactionCode | YES | nvarchar |
| **CarPark** | Id | NO | int |
| Name | NO | nvarchar |
| Address | NO | nvarchar |
| Phone | YES | nvarchar |
| Email | YES | nvarchar |
| Description | YES | nvarchar |
| Lat | NO | nvarchar |
| Lon | NO | nvarchar |
| Fee | YES | money |
| Active | NO | bit |
| AspNetUserId | YES | nvarchar |
| **AspNetRoles** | Id | NO | nvarchar |
| Name | NO | nvarchar |
| **AspNetUserRoles** | UserId | NO | nvarchar |
| RoleId | NO | nvarchar |
| **AspNetUsers** | Id | NO | nvarchar |
| Email | YES | nvarchar |
| EmailConfirmed | NO | bit |
| PasswordHash | YES | nvarchar |
| SecurityStamp | YES | nvarchar |
| PhoneNumber | YES | nvarchar |
| PhoneNumberConfirmed | NO | bit |
| TwoFactorEnabled | NO | bit |
| LockoutEndDateUtc | YES | datetime |
| LockoutEnabled | NO | bit |
| AccessFailedCount | NO | int |
| UserName | NO | nvarchar |

## Algorithms

### GetDistance formula

To calculate the long distance with more accuracy, we must calculate the distance in the circle by using Haversine formula

is the haversine function

*d* is the distance between the two points

*r* is the radius of the sphere,

*φ*1, *φ*2: latitude of point 1 and latitude of point 2, in radians

*λ*1, *λ*2: longitude of point 1 and longitude of point 2, in radians

By apply the inverse haversine or by using the arcsine, we can calculate d: