



**COMSATS University Islamabad,  
Park Road, Chak Shahzad, Islamabad Pakistan**

# **SOFTWARE DESIGN DESCRIPTION**

**(SDD DOCUMENT)**

**for**

**sPARK**  
Version 1.1

***By***

**Muhammad Sawaiz      CIIT/FA16-BCS-027/ISB**

**Bilal Shahid              CIIT/FA16-BCS-037/ISB**

***Supervisor***

**Mr. Asif Muhammad**

***Bachelor of Science in Computer Science (2016-2020)***

# Table of Contents

<b>Revision History .....</b>	<b>iii</b>
<b>1. Introduction .....</b>	<b>1</b>
1.1 Module 1: User Profiling .....	1
1.2 Module 2: Configuration .....	1
1.3 Module 3: Parking Space Allocation .....	1
1.4 Module 4: Automobile Surveillance .....	1
1.5 Module 5: Finance Management .....	1
<b>2. Design methodology and software process model .....</b>	<b>2</b>
2.1 Design Methodology .....	2
2.2 Software Process Model .....	2
<b>3. System overview .....</b>	<b>2</b>
3.1 Architectural design .....	3
3.2 <u>Process flow/Representation</u> .....	4
<b>4. Design models .....</b>	<b>11</b>
4.1 Class Diagram .....	11
4.2 Sequence Diagram .....	12
<b>5. Data design .....</b>	<b>17</b>
5.1 Schema .....	17
5.2 Data dictionary .....	21
<b>6. Algorithm &amp; Implementation .....</b>	<b>23</b>
6.1 addOperator() .....	23
6.2 updateOperator() .....	24
6.3 deleteOperator() .....	25
6.4 viewOperatorDetails() .....	25
6.5 viewOperatorList() .....	25
6.6 viewOperatorActivityRecord() .....	25
6.7 AddVehicle() .....	26
6.8 vehicleAnalysis() .....	26
6.9 addToArchive() .....	27
6.10 viewVehiclesArchive() .....	27
6.11 addMap() .....	28
6.12 viewMap() .....	28
6.13 updateMap() .....	28
6.14 viewParkingLotStatus() .....	28
6.15 updateFee() .....	28
6.16 viewFee() .....	29
6.17 feeCalculation() .....	29
6.18 addCamera() .....	30
6.19 removeCamera() .....	30
<b>7. Software requirements traceability matrix .....</b>	<b>31</b>
<b>8. Human interface design .....</b>	<b>43</b>
8.1 Screen images .....	43
8.2 Screen objects and actions .....	47
<b>9. References .....</b>	<b>52</b>
<b>10. Plagiarism Report .....</b>	<b>52</b>



### **Application Evaluation History**

<b>Comments (by committee)</b> *include the ones given at scope time both in doc and presentation	<b>Action Taken</b>

**Supervised by**  
**Mr. Asif Muhammad**

Signature\_\_\_\_\_

# **1. Introduction**

In this document, we have done the 40% percent implementation of our system's scope and provided the algorithms of our implementation. The implementation includes the backend saving and fetching of data along with the data manipulation, as required. We have also made most of the interfaces of our system and attached the screen shots of some of major interfaces in this document. Furthermore, all the development supporting diagrams have been provided in this document. The modules of the system are given below:

## **1.1 Module 1: User Profiling**

The system has two kinds of users: admins and operators. The users of this system have their accounts and have privileges based on their job. All the privileges can only be accessed after they have signed in to their respective accounts.

## **1.2 Module 2: Configuration**

The administrator operating the system gets the control of system. They can configure map of parking lot and the parking charges as per their demands.

## **1.3 Module 3: Parking Space Allocation**

The incoming vehicles of parking lot will be allotted the right space to get parked. This space will be selected based on number of constraints.

## **1.4 Module 4: Automobile Surveillance**

All the vehicles entering the parking lot will be detected and their size will be analyzed to find the appropriate parking space for them individually. After this, vehicles will be allowed to enter the parking lot. In this, all vehicles will be uniquely identified, and it will be check that the parkers have parked the vehicles in given lane by multi-camera coordination.

## **1.5 Module 5: Finance Management**

Parkers visiting the parking lot can be charged and charges can be based on number of parameters. The parker will be allowed to exit when fee is cleared. The fee with furthermore information will be kept in record for each parker.

## **2. Design methodology and software process model**

### **2.1 Design Methodology**

Object-oriented methodology will be used in design perspective as separate classes will be created. As its understandability and coding is easier. The testing will be easier as bugs can separately be identified between each feature. The code of common functionality modules can be reused among modules.

### **2.2 Software Process Model**

Modified waterfall model will be used as software process methodology. Modified waterfall model is suited for this project as all the milestones are clear before the start of development, and expectancy that the requirements will change during the development is very low. Further upon, system requires a well-planned development on its major modules, and this model is one of the best for management control at beginners' level. Modified waterfall is also beneficial as it allows to switch from one stage to another as per the desired needs.

## **3. System overview**

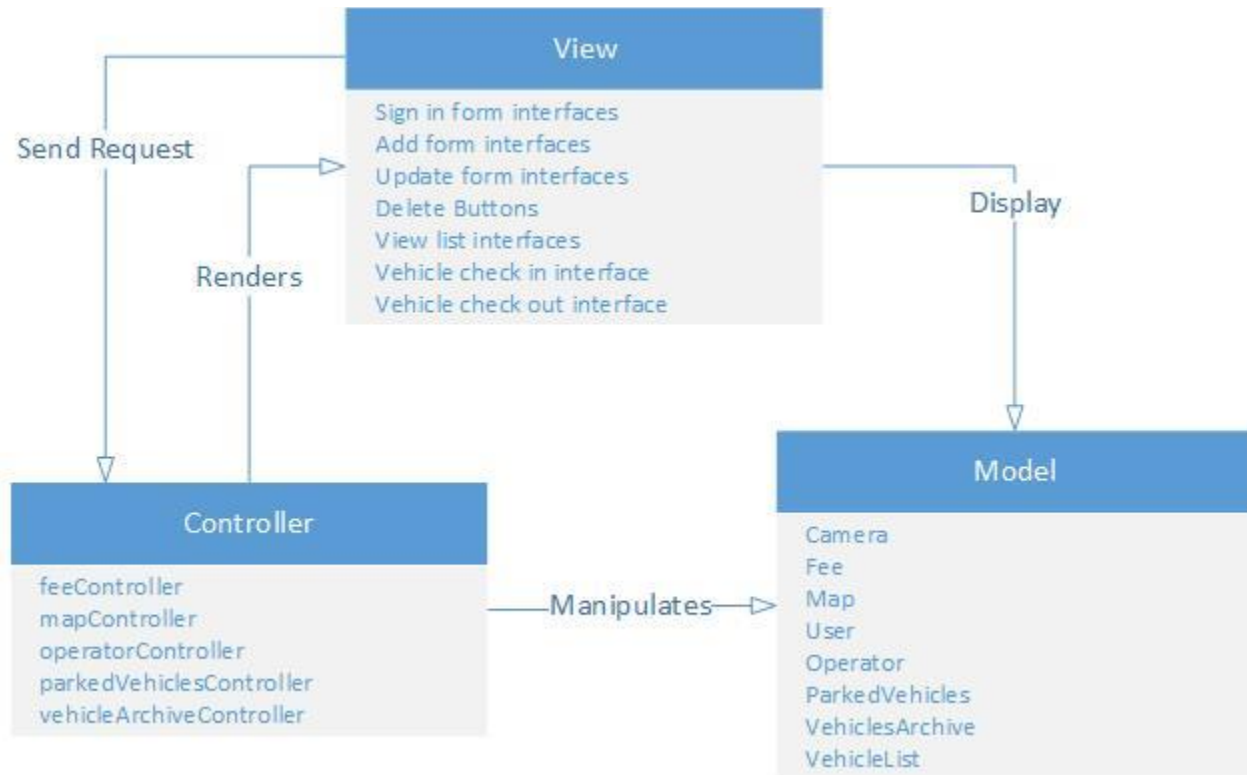
sPARK is designed to be implemented at public parking lots, paid or unpaid, where there is a defined map of parking area and proper boundaries defined for the parking space. The system takes the map of parking lot and add it to the database. Cameras will be used to identify each entering vehicle which must stop at entrance in sight of the cameras so that they can analyze its size and read its vehicle registration plate. System generates a token for the parker to park at a given space which best suites the vehicle according to the algorithm. The algorithm basically works as combination of bin-packing algorithms. The multicamera coordination in the parking lot will be tracking each vehicle entering the parking lot and will confirm that it is parked in the allocated space or not. Systems efficiency will be at best if the parkers parks at the allocated space. In case of negligence, fine is imposed on the parker, but the system performance may be affected due to wrong parking. At the time of leaving the parking lot, the parkers are charged according to the parking time, and if they were charged with any fine for not parking correctly. The parker can leave after clearing the bill.

By default, system is provided with one administrator's account. The admin has control upon creating, updating and deleting operators' account. Admin have control to add and update map, configure parking lot charges, and keep a check on operators' activity. The operators are provided with an already created account from which they are able to control entrance and exit of parking lot. Operator(s) at the entrance manages the allocation of incoming vehicles and operator(s) at the exit manages the fee collection from outgoing vehicles.

### 3.1 Architectural design

The architecture we are following is MVC. This is because the complete project is mainly divided in 3 portions. The 'Model' which stores the data in form of collections, the 'View' which contains the desktop interfaces of project by which user can interact with the system and request for different functionalities of the system, and 'Controller' which is the object-oriented code and algorithms used in project which basically manipulates the model and performs the actual functionality of the system and renders that on to view.

In short, the view sends request to controller to proceed for change and controller displays that change on view by manipulating the model.



**Figure 1: MVC Architecture**

## 3.2 Process flow/Representation

### 3.2.1 Activity Diagram

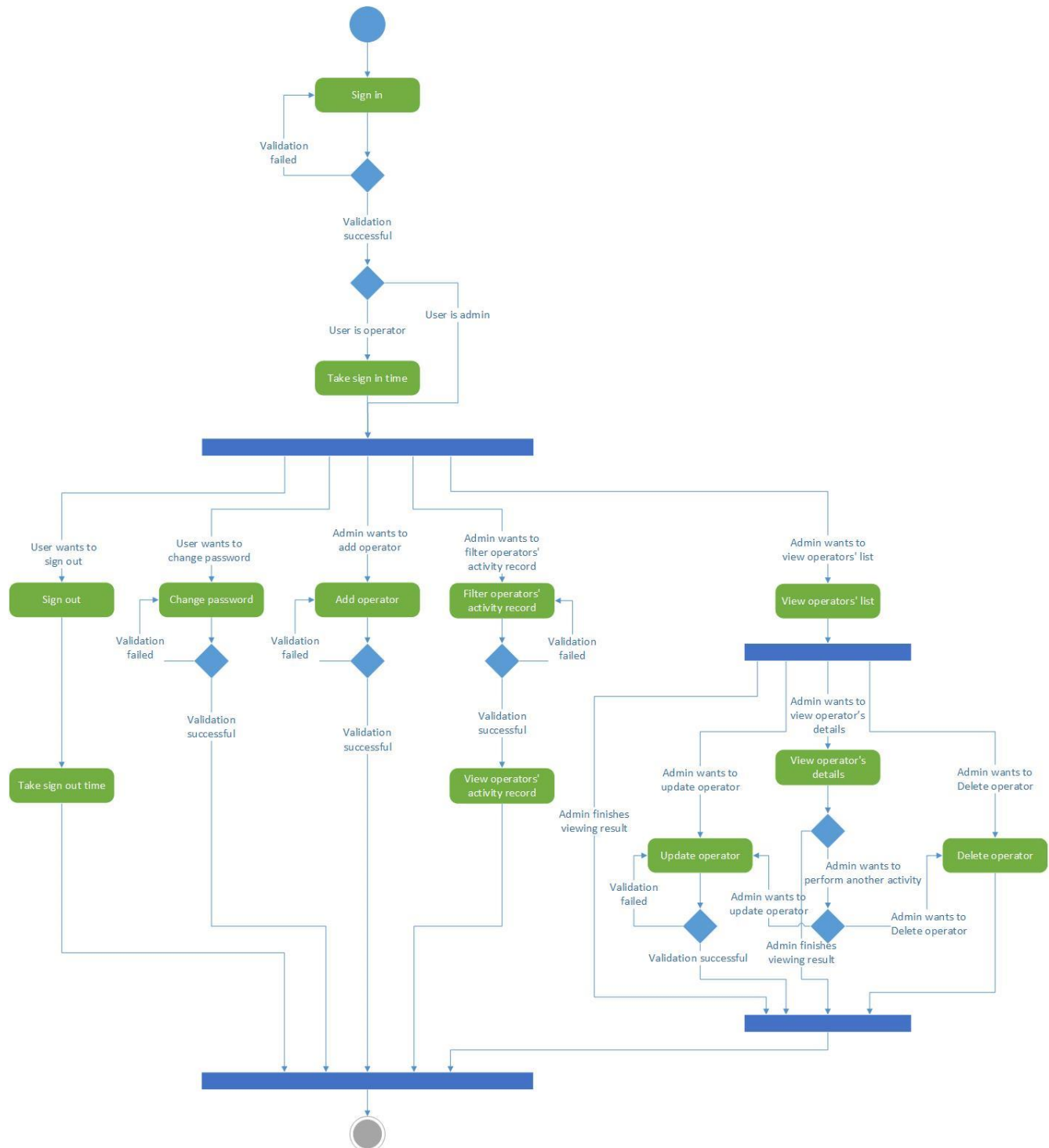
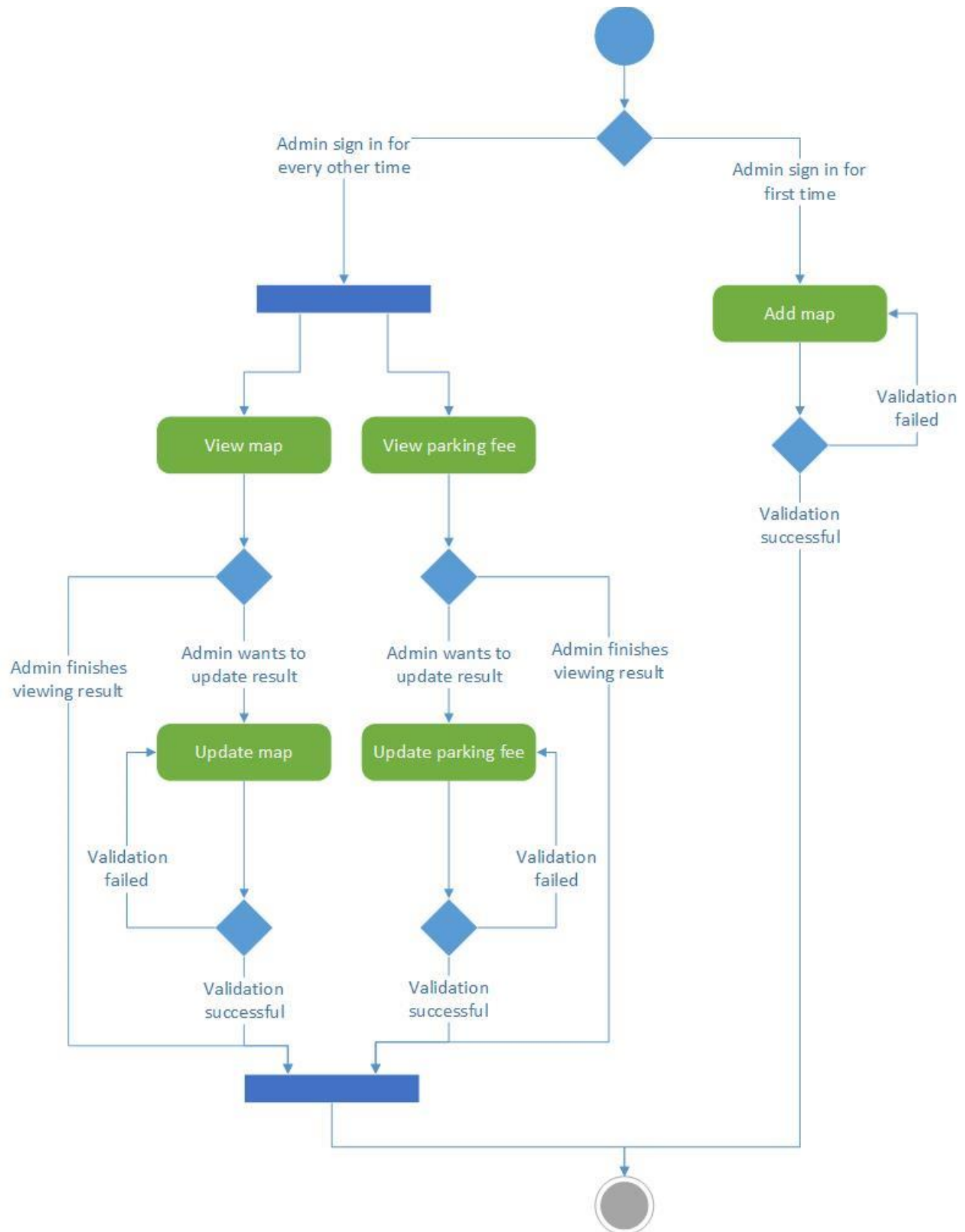
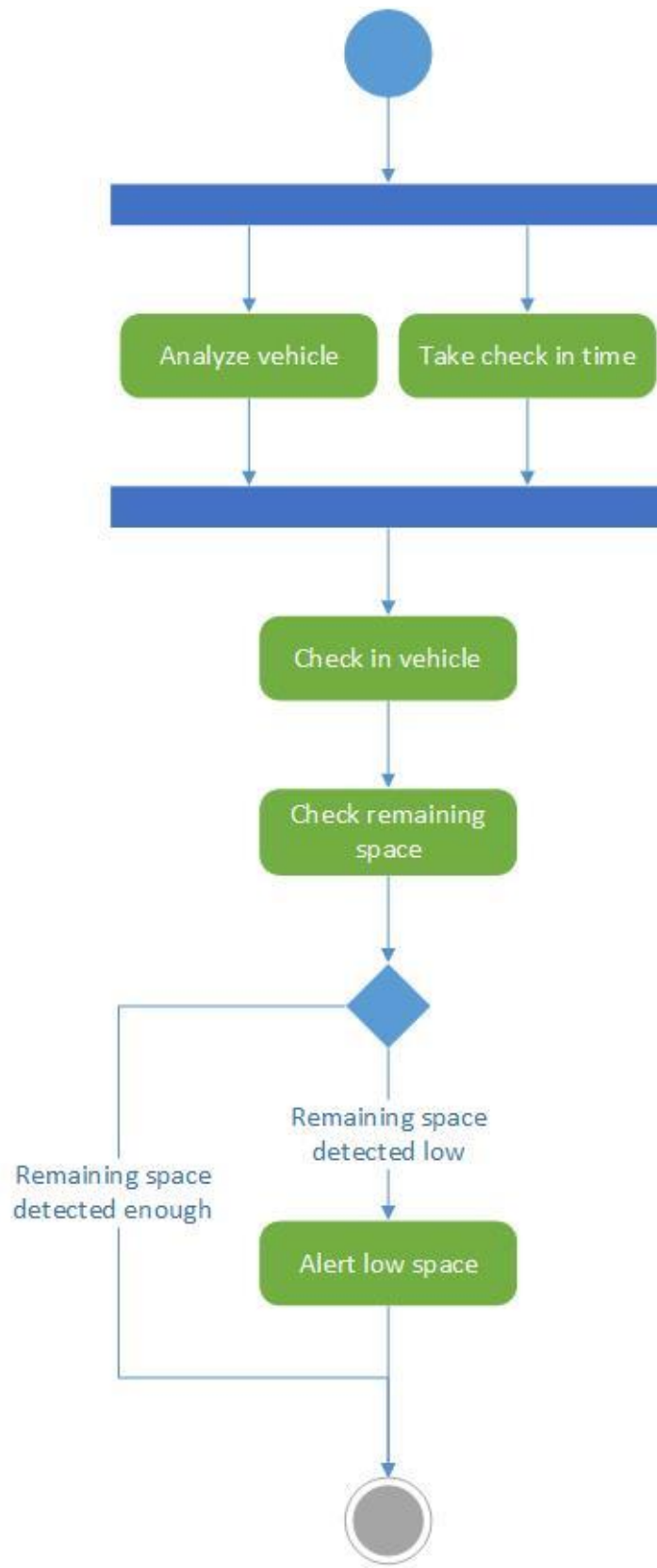
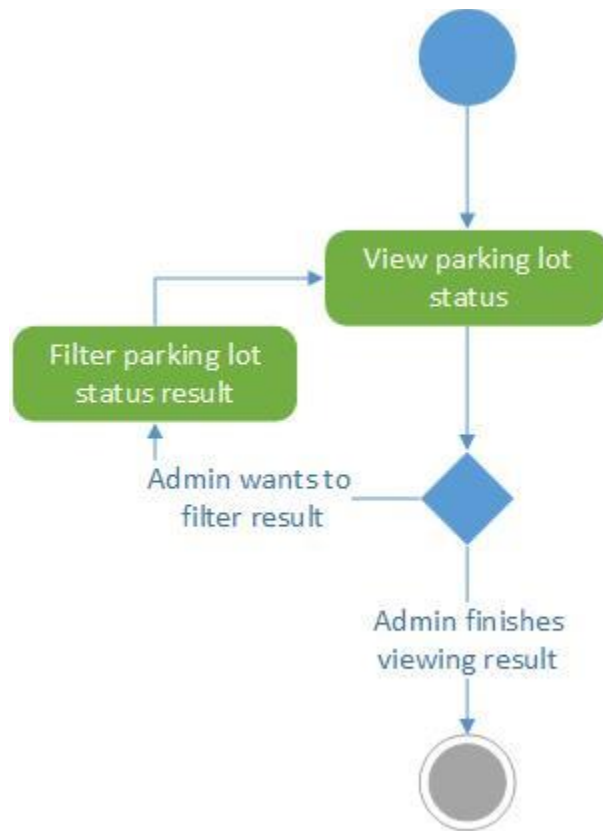


Figure 2: Activity Diagram (User Profiling)

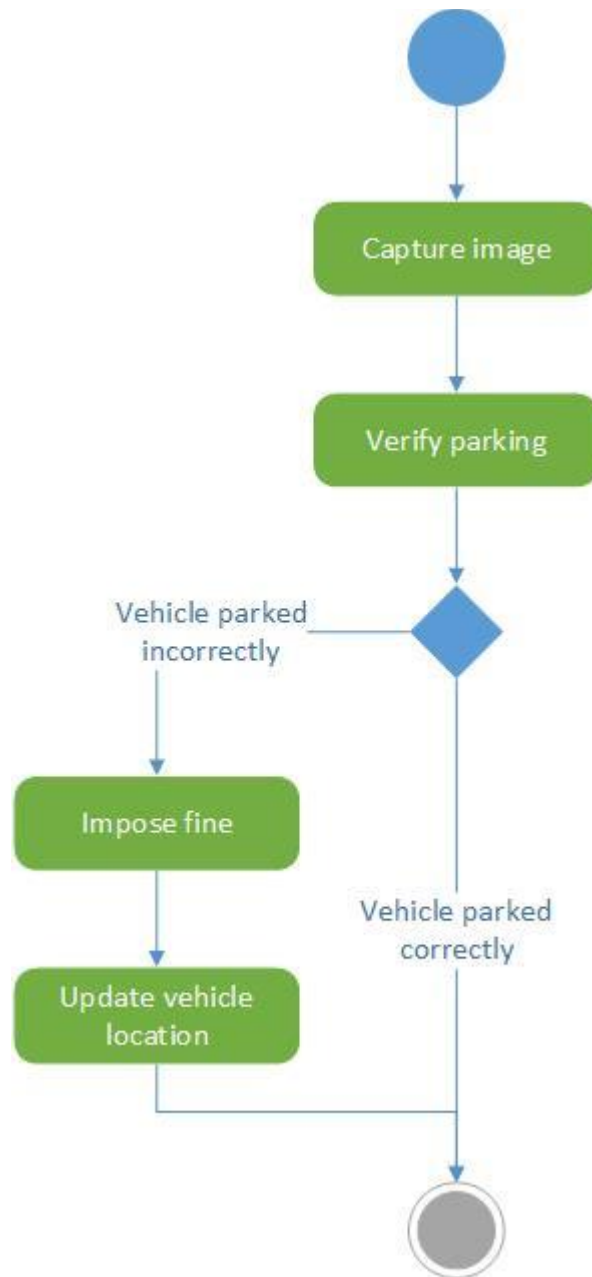


**Figure 3: Activity Diagram (Configuration)**

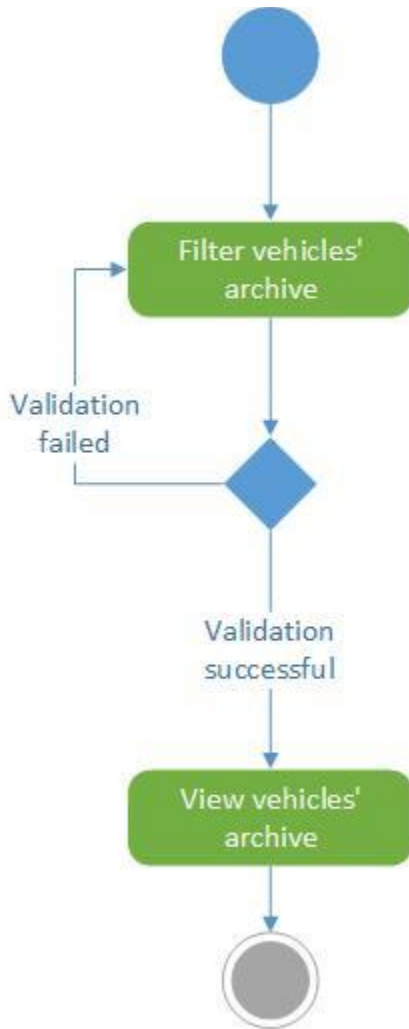
**Figure 4: Activity Diagram (Parking Space Allocation)**



**Figure 5.1: Activity Diagram (Automobile Surveillance - Admin)**



**Figure 5.2: Activity Diagram (Automobile Surveillance - System)**



**Figure 6.1: Activity Diagram (Finance Management - Admin)**

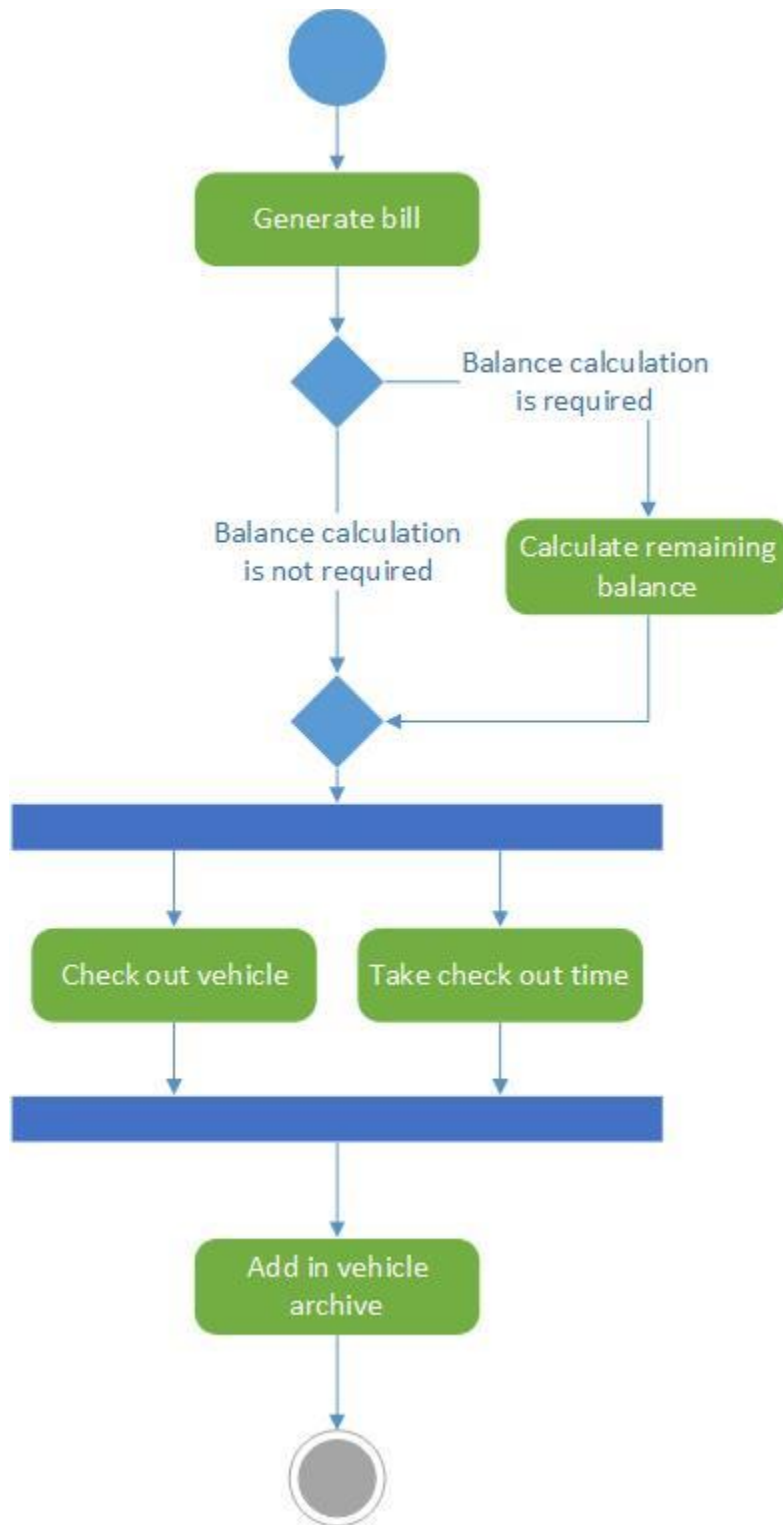


Figure 6.2: Activity Diagram (Finance Management - Operator)

## 4. Design models

### 4.1 Class Diagram

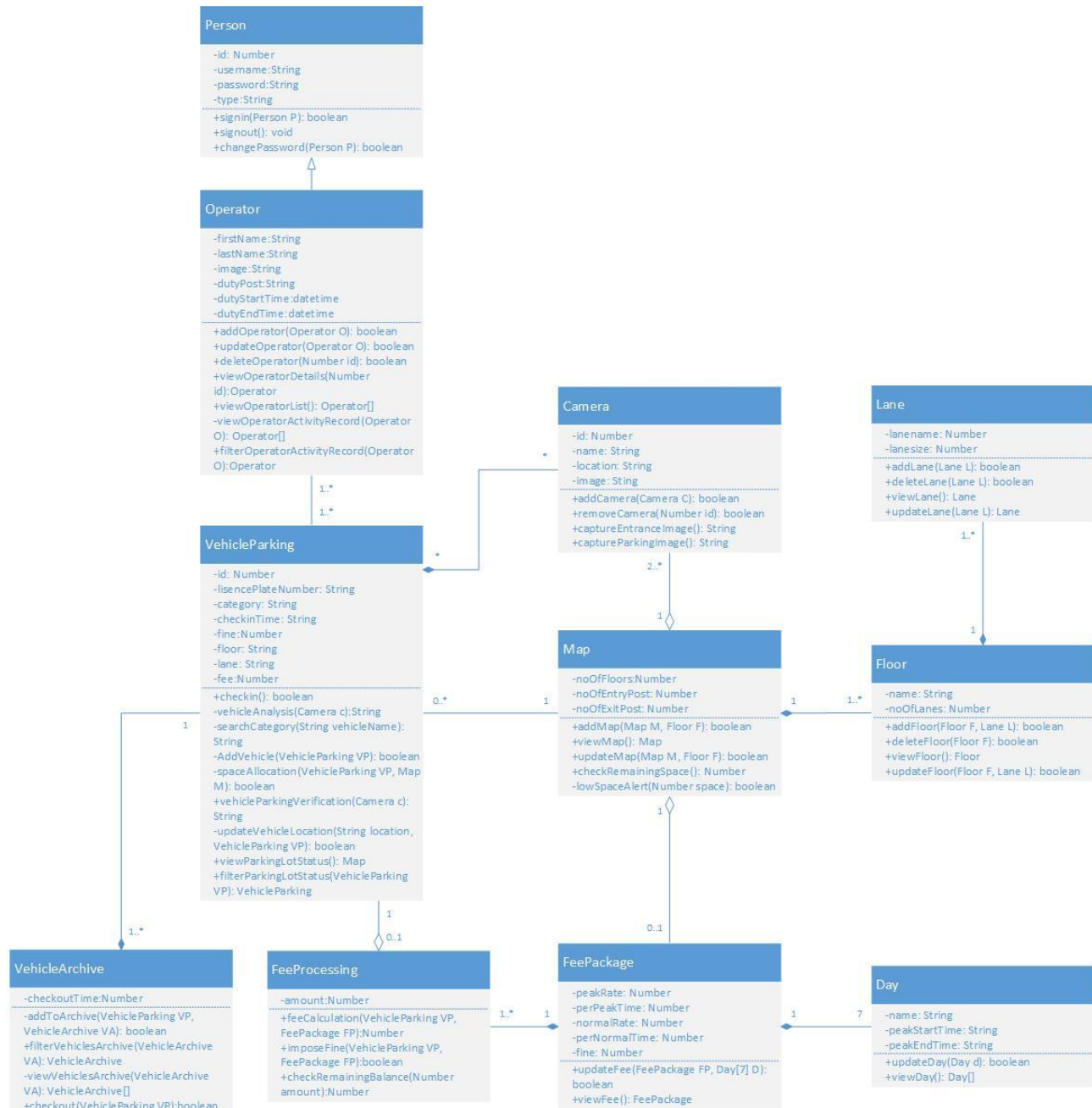


Figure 7: Class Diagram

## 4.2 Sequence Diagram

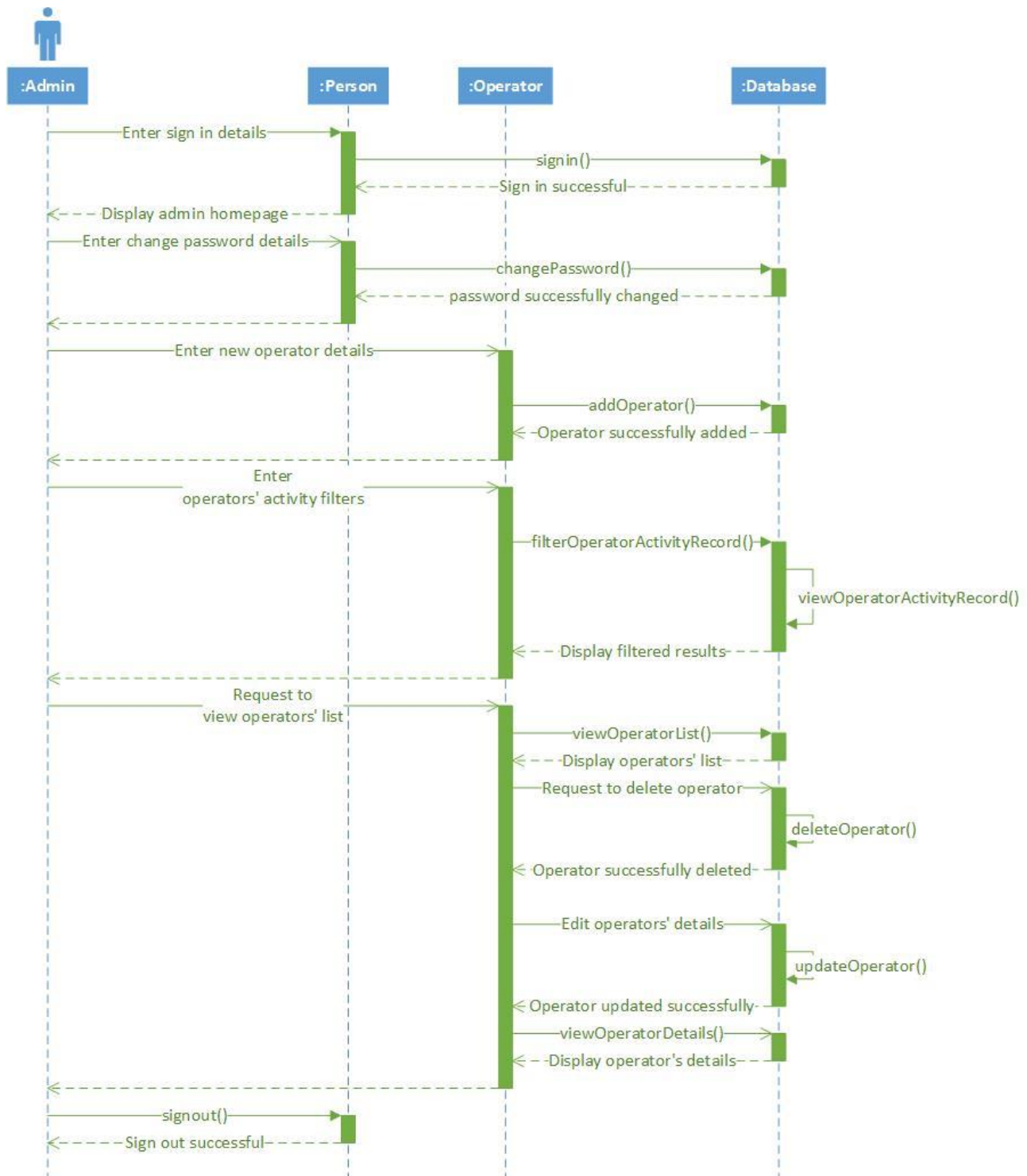


Figure 8: Sequence Diagram (User Profiling - Admin)



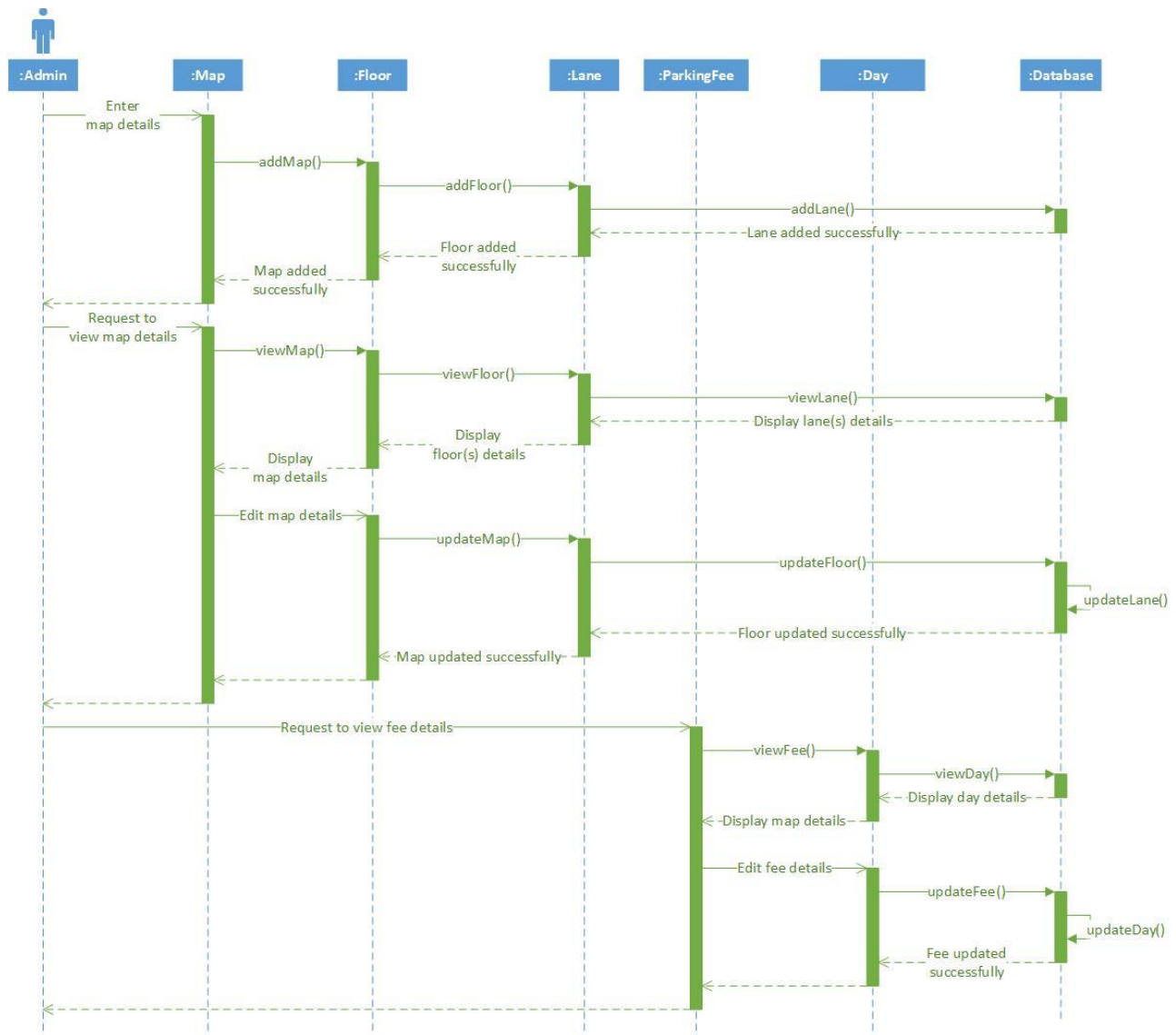


Figure 9: Sequence Diagram (Configuration)

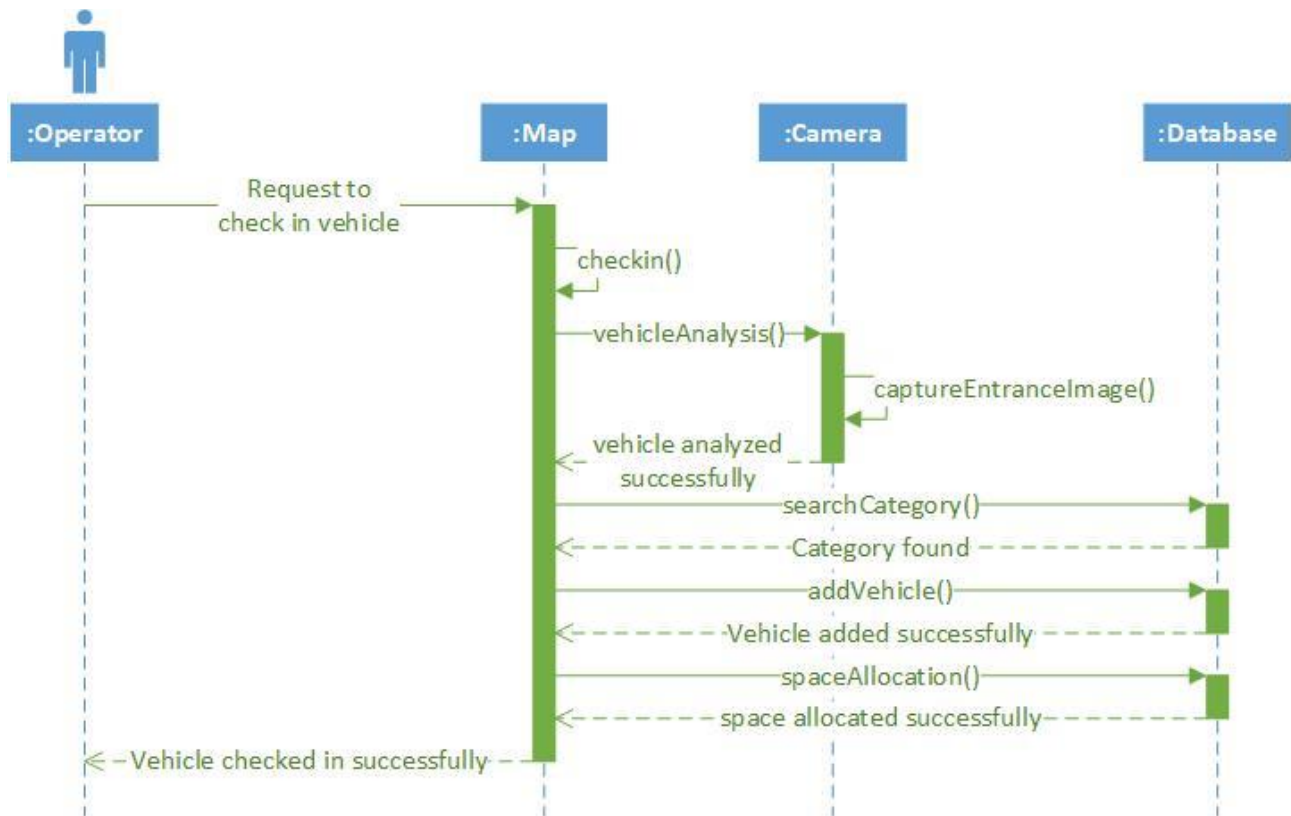


Figure 10: Sequence Diagram (Parking Space Allocation)

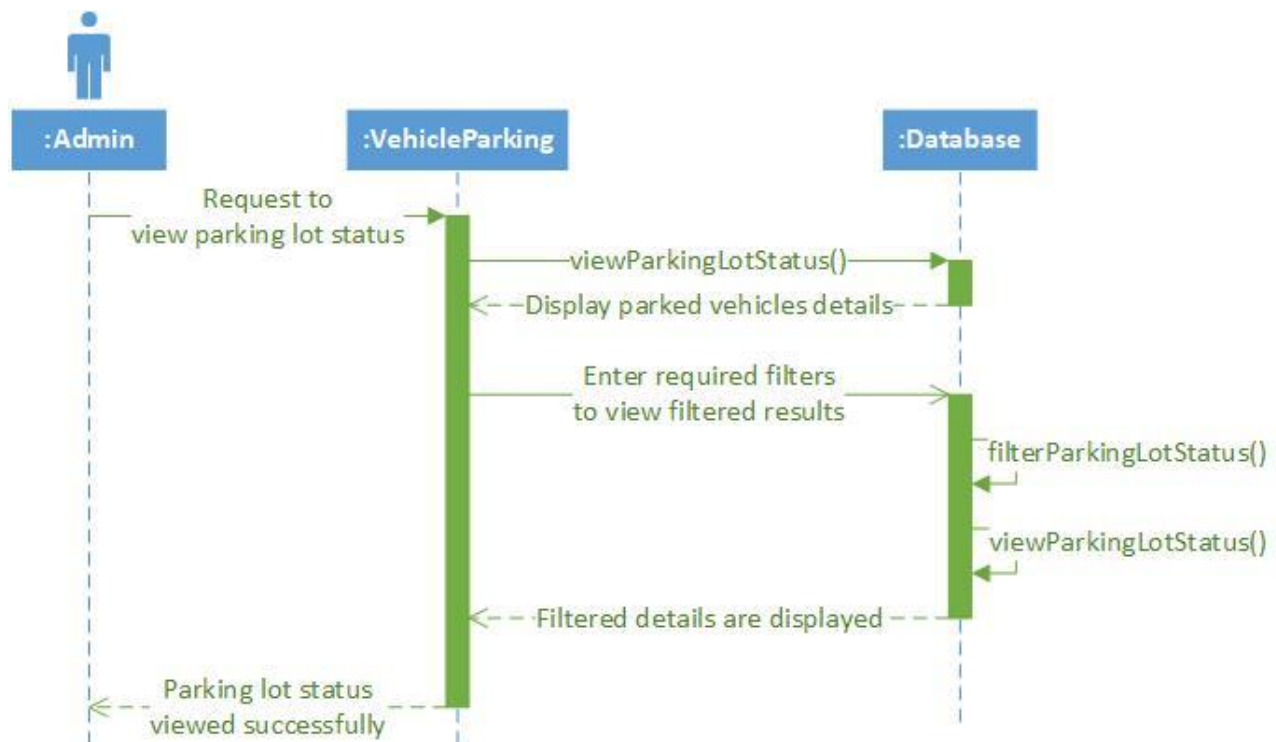


Figure 11.1: Sequence Diagram (Automobile Surveillance - Admin)

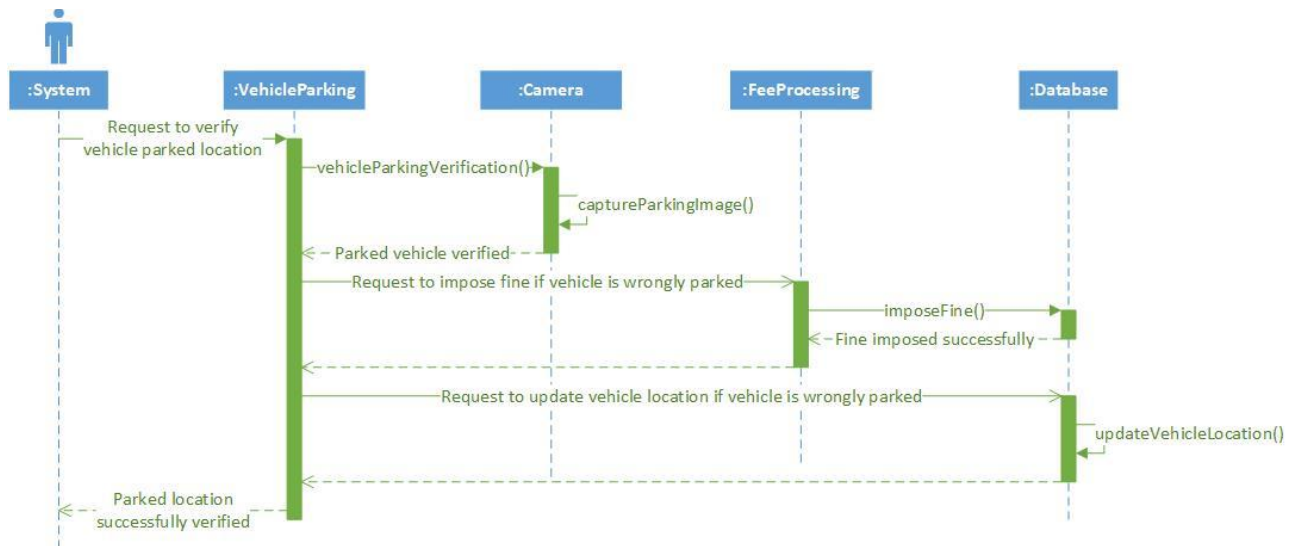


Figure 11.2: Sequence Diagram (Automobile Surveillance - System)

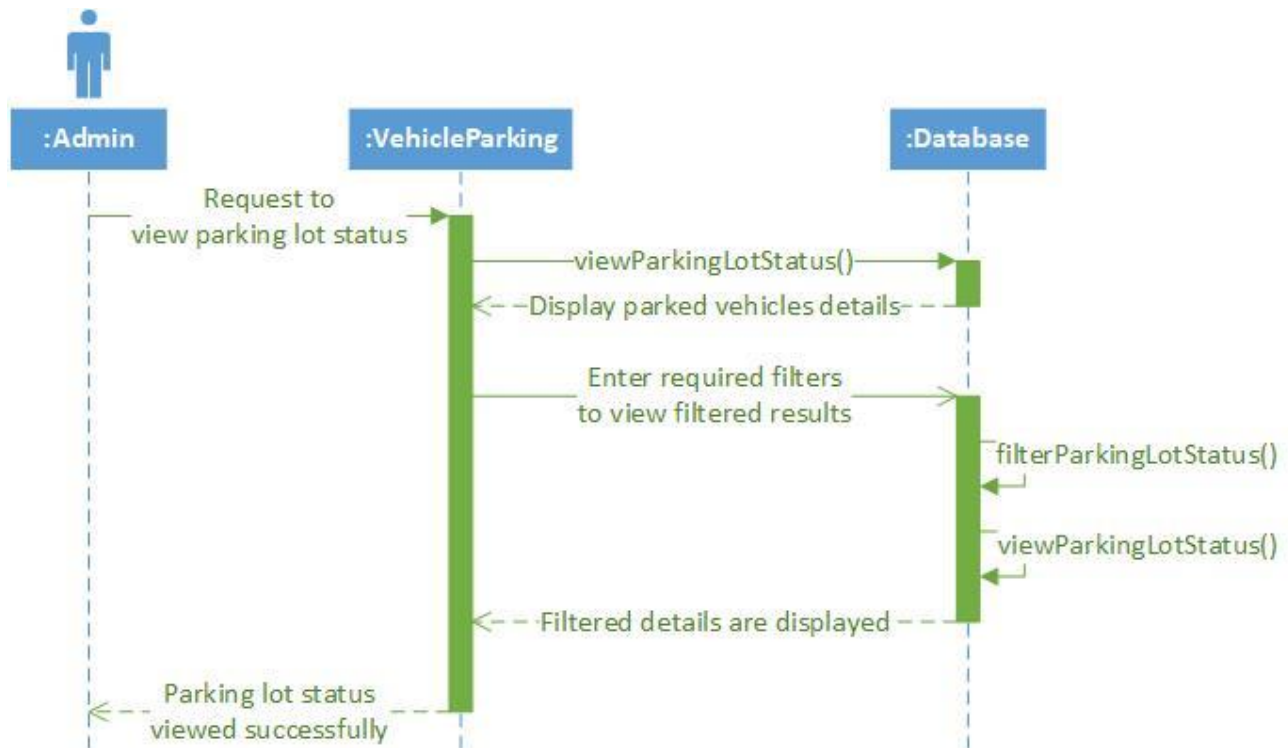
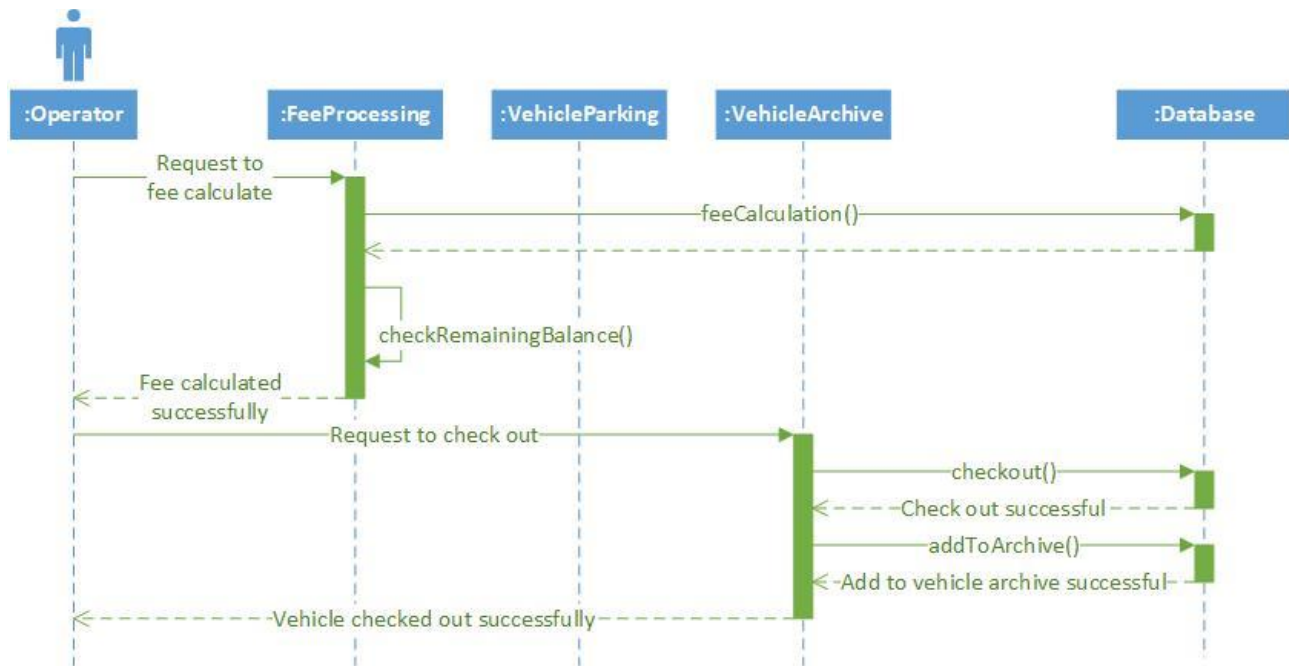


Figure 12.1: Sequence Diagram (Finance Management - Admin)

**Figure 12.2: Sequence Diagram (Finance Management - Operator)**

## 5. Data design

### 5.1 Schema

#### 5.1.1 Vehicle list

```
make: {
  type: String,
  required: true
},
model: [{
  name: {
    type: String,
    required: true
  },
  category: {
    type: String,
    required: true
  }
}]
```

The schema stores information of vehicles with which incoming vehicles will be referenced. First, the schema stores a variable make of type string and secondly schema stores an array on objects, model, that has two variables of both type string, named: name and category. All the variables are constrained as require.

#### 5.1.2 Camera

```
name:{
  type:String
},
location:{
  type:String
}
```

The schema stores the information of cameras connected to the system. Each document has two variables of type string and named: name and location. All the constraints will be implemented on the interface

#### 5.1.3 User

```
username:{
  type:String
},
password:{
  type:String
},
type: {
  type: String
}
```

The schema stores the username, password and type of user whose account is developed on system. All three variables are of type string. The variable constraints are set on system interface.

#### 5.1.4 Operator

```

firstName: {
  type: String
},
lastName: {
  type: String
},
image: {
  type: String
},
duty: {
  dutyPost: {
    type: String
  },
  dutyStartTime: {
    type: String
  },
  dutyEndTime: {
    type: String
  }
},
loggingActivity:[{
  signinTime: {
    type: String
  },
  signoutTime: {
    type: String
  },
  dutyPost: {
    type: String
  }
}]

```

The schema stores the information of operator. This information includes first name, last name, image as string typed variables. The object named duty has string type variables: duty post, which operator is currently allocated, duty start time and duty end time. Another information is an array of objects named loggingActivity also has string type variables: sign time, sign out time and the duty post on which account was used in that time frame. All the required variable constraints are set on interface.

#### 5.1.5 Map

```

floor: [{
  name: {
    type: String
  },
  lane: [{
    name: {
      type: String
    }
  }
}]

```

```

    },
    size: {
      type: Number
    }
  }
}],
post: {
  entrancePost: [{
    name: {
      type: String
    }
  }],
  exitPost: [{
    name: {
      type: String
    }
  }
]}
}

```

The schema stored the information of parking lot map. It has an array of objects named floor. This array stores the information of each floor including name, typed string, and an array of object, lane, having two variables named: name of type string and size of type number. Another information is also the object named post that stores two arrays of entrance posts and exit posts of type string. All the required constraints are set on the interface.

#### 5.1.6 Fee

```

day: [{
  name: {
    type: String,
  },
  peakStartTime: {
    type: String
  },
  peakEndTime: {
    type: String
  }
}],
Rate: {
  peakPrice: {
    type: Number
  },
  normalPrice: {
    type: Number
  },
  perPeakTime: {
    type: Number
  },
  perNormalTime: {
    type: Number
  }
},
fine: {
  type: Number
}

```

```
}
```

The schema stores the information of parking lot fee for each day respective to busy hours and normal hours. It has a variable fine of type number and an object named rate that stores four variables, all of which are type number and named: peakPrice which is price at busy hours, normalPrice which is price at normal hours, perPeakTime which is the time rate for fee increase in busy hours and perNormalTime which is the time rate for fee increase in normal hours. Another is an array of objects name day, that has three variables, all type string, and named: name which is day name, peakStartTime which is busy hours start time and peakEndTime which is busy hours end time, for every respective day. All the required constraints are set on interface.

### 5.1.7 Parked vehicles

```
vehicle: {
  licensePlateNumber: {
    type: String
  },
  category: {
    type: String
  }
},

location: {
  floor: {
    type: String
  },
  lane: {
    type: String
  }
},

checkinTime: {
  type: String
},
fine: {
  type: Number
}
```

The schema stores the information of vehicles of parking lot. This information includes fine of type number, check in time of type string, an object named location which has two variables named floor and lane both of type string and an object named vehicle which also has two variables name licensePlateNumber and category both of which are of type string. The information is all system generated and none of it is user input for this schema. Hence no constraints are applied.

### 5.1.8 Vehicles archive

```
vehicle: {
  licensePlateNumber: {
    type: String
  },
  category: {
    type: String
  }
},
```



```

location: {
  floor: {
    type: String
  },
  lane: {
    type: String
  }
},
checkinTime: {
  type: String
},
checkoutTime: {
  type: String
},
fine: {
  type: Number
},
fee: {
  type: Number
}

```

The schema stores the information of vehicles that visited the parking lot. This information includes fine of type number, fee of type number, check in time of type string, check out time of type string, an object named location which has two variables named floor and lane both of type string and an object named vehicle which also has two variables name licensePlateNumber and category both of which are of type string. The information is all system generated and none of it is user input for this schema. Hence no constraints are applied.

## 5.2 Data dictionary

The table below provides information of attributes of each collection. The attribute information includes its data type and description about each attribute.

**Table 1 Data dictionary**

Collection	Attribute	Data type	Description
Camera	Name	String	Name of camera
Camera	Location	String	Location where a camera is fixed
Fee	Day	Object	Object containing information of fee for a day
Fee	Name	String	Name of day whose fee is stored
Fee	peakStartTime	String	Start time for peak hours of the day
Fee	peakEnd Time	String	Start time for peak hours of the day
Fee	Rate	Object	Object containing information of rate of parking
Fee	peakPrice	Number	Rate for peak hour timings.
Fee	normalPrice	Number	Rate for normal hour timings.
Fee	perPeakTime	Number	Time rate for which price is set for peak hours
Fee	perNormalTime	Number	Time rate for which price is set for normal hours
Fee	Fine	Number	Amount of fine as penalty.

Map	Floor	Array of Objects	One object contains information of one floor
Map	Name	String	Name of floor whose information is stored
Map	Lane	Array of Objects	One object contains information of one lane
Map	Name	String	Name of lane whose information is stored
Map	size	Number	Size of a lane
Map	Post	Object	Object containing information of posts in a parking lot
Map	entrancePost	Array of Objects	One object contains name of one entrance post
Map	name	String	Name of entrance post whose information is stored
Map	exitPost	Array of Objects	One object contains name of one exit post
Map	name	String	Name of exit post whose information is stored
Operator	firstName	String	First name of operator whose information is stored
Operator	lastName	String	Last name of operator whose information is stored
Operator	Image	String	Path of image of operator whose information is stored
Operator	Duty	Object	Object containing information of duty for an operator
Operator	dutyPost	String	Post where operator's duty is fixed.
Operator	dutyStartTime	String	Starting time of operator's duty.
Operator	dutyEndTime	String	Ending time of operator's duty.
Operator	loggingActivity	Object	Object containing information of operator's logging activity.
Operator	signinTime	String	Time at which operator signed into the system.
Operator	signoutTime	String	Time at which operator signed out of the system.
Operator	dutyPost	String	Post from where operator has signed in and signed out of the system.
parkedVehicles	Vehicle	Object	Object containing information of vehicle currently in the parking lot.
parkedVehicles	licensePlateNumber	String	License plate number of the vehicle whose record is saved.
parkedVehicles	Category	String	Category of the vehicle whose record is saved.
parkedVehicles	Location	Object	Object containing the location where a vehicle is parked.
parkedVehicles	Floor	String	Floor where the vehicle is parked.
parkedVehicles	Lane	String	Lane where the vehicle is parked.
parkedVehicles	checkinTime	String	Check in time of the vehicle which is parked.
parkedVehicles	Fine	Number	Fine imposed on a vehicle if it is not parked where system allocated space

			for it
User	Username	String	Unique username of each user necessary to sign in
User	Password	String	Password of user necessary to sign in.
User	Type	String	Type of user which can be admin or operator.
vehiclesArchive	Vehicle	Object	Object containing information of vehicle present in vehicle archive.
vehiclesArchive	licensePlateNumber	String	License plate number of the vehicle present in vehicle archive.
vehiclesArchive	Category	String	Category of the vehicle whose record is saved in archive.
vehiclesArchive	Location	Object	Object containing the location where a vehicle was parked.
vehiclesArchive	Floor	String	Floor where the vehicle was parked.
vehiclesArchive	Lane	String	Lane where the vehicle was parked.
vehiclesArchive	checkinTime	String	Check in time of the vehicle which was parked.
vehiclesArchive	checkoutTime	String	Check out time of the vehicle which was parked.
vehiclesArchive	fine	Number	Amount of fine that was imposed on the vehicle.
vehiclesArchive	fee	Number	Amount of fee that was paid by the vehicle before checkout
vehiclesList	make	String	Company name of the vehicle that is stored
vehiclesList	model	Array of Objects	One object contains information of one model of a company.
vehiclesList	name	String	Name of the model that is stored.
vehiclesList	category	String	Category of the model that is saved.

## 6. Algorithm & Implementation

### 6.1 addOperator()

```

user.findOne({ username: 'Bilal' }, function (err, doc) {
  if (err) {
    res.status(500).send('Error Occurred')
    console.log('error')
  }
  else {
    if (doc) {
      console.log('User Exists')
    }
    else {
      var usr = new user();
      usr.username="Bilal";
      usr.password=usr.hashPassword('test');
      usr.type="Operator";
    }
  }
}

```

```

usr.save(function (err, user) {
  if (err) {
    console.log('DB ERROR')
  }
  else {
    console.log(user)
  }
})
var optr = new operator({
  _id: usr._id,
  firstName: "Bilal",
  lastName: "Shahid",
  image: "image1.jpg",
  duty: {
    dutyPost: 'Post I',
    dutyStartTime: '00:00',
    dutyEndTime: '00:00'
  }
});

optr.save(function (err, data) {
  if (err) {
    console.log(err);
  }
  else {
    console.log(data);
  }
});}

```

## 6.2 updateOperator()

```

user.findOne({ username: 'Sawaiz' }).then(doc => {
  doc.username = "Sawaiz";
  doc.password = "sawaiz";
  doc.save();
  operator.findById(doc._id).then(doc2 => {
    doc2.firstName = "Sawaiz";
    doc2.lastName = "sawaiz";
    doc2.image = "image1.png";
    doc2.duty = {
      dutyPost: "POST II",
      dutyStartTime: "00:00",
      dutyEndTime: "00:00"
    }
    doc2.save();
    res.send("success");

  }).catch(err => {
    console.log(err)
  })
}).catch(err => {
  console.log(err)
})

```

```
}}
```

### 6.3 deleteOperator()

```
user.findByIdAndDelete('5ddc182b3fb61a0761f85887', function (err) {  
  if (err) return err;  
});  
operator.findByIdAndDelete('5ddc182b3fb61a0761f85887', function (err) {  
  if (err) return err;  
  res.send("Successfully deleted")  
});
```

### 6.4 viewOperatorDetails()

```
user.findOne({ username: 'Bilal' }, function (err, result) {  
  if (err) throw err;  
  console.log(result)  
  operator.findById(result._id, function (err, result) {  
    if (err) throw err;  
    console.log(result)  
  });  
});
```

### 6.5 viewOperatorList()

```
user.find ({}, function (err, result) {  
  if (err) throw err;  
  console.log(result)  
  operator.findById(result._id, function (err, result) {  
    if (err) throw err;  
    console.log(result.firstname+ " " + result.lastname)  
    console.log(result.image)  
  });  
});
```

### 6.6 viewOperatorActivityRecord()

```
user.find ({}, function (err, result) {  
  if (err) throw err;  
  console.log(result)  
  operator.findById(result._id, function (err, result) {  
    if (err) throw err;  
    console.log(result.firstname+ " " + result.lastname)  
    console.log(result.image)  
    console.log(result.loggingActivity.signinTime)  
    console.log(result.loggingActivity.signoutTime)  
    console.log(result.loggingActivity.dutyPost)  
  });  
});
```

## 6.7 AddVehicle()

```

vehicle.findOne({ licensePlateNumber: 'qw123' }, function (err, doc) {
  if (err) {
    res.status(500).send('Error Occurred')
    console.log('error')
  }
  else {
    if (doc) {
      console.log('Vehicle Exists')
    }
    else {
      var datetime = new Date()
      var veh = new vehicle({
        licensePlateNumber: "qw123",
        details: {
          floor: 1,
          lane: 2,
          checkinTime: datetime.getDate() + "-" + (datetime.getMonth() + 1) + "-"
+ datetime.getFullYear() + " " + datetime.getHours() + ":" + datetime.getMinutes() + ":" + datetim
e.getSeconds(),
          fine: 0
        },
        category: "Small"
      });
      veh.save(function (err, doc) {
        if (err) {
          console.log('DB ERROR')
        }
        else {
          console.log(doc)}}}}))

```

## 6.8 vehicleAnalysis()

```

try:
    import httplib # Python 2
except:
    import http.client as httplib # Python 3
headers = {"Content-type": "application/json",
           "X-Access-Token": ""}
conn = httplib.HTTPSConnection("dev.sighthoundapi.com",
                                context=ssl.SSLContext(ssl.PROTOCOL_TLSv1))

image_data = base64.b64encode(open("MND691.jpg", "rb").read()).decode()

params = json.dumps({"image": image_data})
conn.request("POST", "/v1/recognition?objectType=licenseplate", params, headers)
response = conn.getresponse()
result = response.read()

obj = json.loads(result)

```

```
print("PLATE NO: "+
obj['objects'][0]['licenseplateAnnotation']['attributes']['system']['string']['name'])
```

## 6.9 addToArchive()

```
parVeh.findOne({ licensePlateNumber: 'qw123' }, function (err, doc) {
  if (err) {
    res.status(500).send('Error Occurred')
    console.log('error')
  }
  else {
    var datetime = new Date();
    var arcveh = new vehArch({
      licensePlateNumber: doc.licensePlateNumber,
      details: {
        floor: doc.details.floor,
        lane: doc.details.lane,
        checkinTime: doc.checkinTime,
        checkoutTime: datetime.getDate() + "-" + (datetime.getMonth() + 1) + "-"
+ datetime.getFullYear() + " " + datetime.getHours() + ":" + datetime.getMinutes() + ":" + datetim
e.getSeconds(),
        fine: doc.details.fine,
        charges: 12345
      },
      category: doc.category
    });

    arcveh.save(function (err, result) {
      if (err) {
        console.log('DB ERROR')
      }
      else {
        console.log(result)
      }
    })
    doc.remove()
  }
}
```

## 6.10 viewVehiclesArchive()

```
vehArch.find({}, function (err, result) {
  if (err) throw err;
  console.log(result)
  res.send("success!!")
});
```

## 6.11 addMap()

**Inputs:** number of entrances, number of exists, number of floors, number of lanes, lane name and size.

**Output:** true

**Steps:**

- Enter number of floors.
- For each floor, enter number of lanes.
- For each lane of each floor, enter lane name and lane size.
- Enter number of parking lot entrances.
- Enter number of parking lot exits.
- Save all the entered information in Map schema.

## 6.12 viewMap()

```
map.find({}, function (err, result) {  
  if (err) throw err;  
  console.log(result)  
  res.send("success!!")  
});
```

## 6.13 updateMap()

**Inputs:** number of entrances, number of exists, number of floors, number of lanes, lane name and size.

**Output:** true

**Steps:**

- If required, update number of floors.
- For each floor, if required, update number of lanes.
- For each lane of each floor, if required, update lane name and lane size.
- If required, update number of parking lot entrances.
- If required, update number of parking lot exits.
- Save all the updated information in Map schema.

## 6.14 viewParkingLotStatus()

```
parkVeh.find({}, function (err, result) {  
  if (err) throw err;  
  console.log(result)  
});});
```

## 6.15 updateFee()

```
fee.findById('5dda89795c0d0e77983c149f').then(doc => {  
  var doc1 = doc.day[0];  
  doc1.peakStartTime = "00:00";  
  doc1.peakEndTime = "00:00";  
  doc.save();  
  res.send("success");  
}).catch(err => {
```



```

    console.log(err)
  })
  fee.findByIdAndUpdate('5dda89795c0d0e77983c149f', {
    Rate: {
      peakPrice: 0,
      normalPrice: 0,
      perPeakTime: 0,
      perNormalTime: 0
    },
    fine: 0
  }, function (err, result) {
    if (err) throw err;
    console.log(result)
  })

```

## 6.16 viewFee()

```

fee.findOne({}, function (err, result) {
  if (err) throw err;
  console.log(result)
  res.send("success!!")
});

```

## 6.17 feeCalculation()

```

parVeh.findOne({ licensePlateNumber: 'qw123' }, function (err, doc) {
  if (err) {
    res.status(500).send('Error Occurred')
    console.log('error')
  }
  else {
    var datetime = new Date();
    var hr = parseInt(datetime.getHours(), 10)
    var min = parseInt(datetime.getMinutes(), 10)
    var cotime = (hr*60)+min

    var str1 = (doc.checkinTime).split(" ");
    var str2 = (str1[1]).split(":");
    var cih = parseInt(str2[0], 10)
    var cim = parseInt(str2[1], 10)
    var citime = (cih*60)+cim

    var pst, pet, pp, np, ppt, pnt, fine, fee

    fee.find ({}, function (err, doc1) {
      for(doc1.day.name in each doc1.day){
        if(doc1.day.name == datetime.getDay()){
          pst = doc.day.peakStartTime
          pet = doc.day.peakEndTime
          pp = parseInt(doc.rate.peakPrice
          np = doc.rate.normalPrice
          ppt = doc.rate.peakPerTime

```

```

        pnt = doc.rate.normalPerTime
        fine = doc.fine
    }
    break
}
})
str1 = (pst).split(":");
pst = ((parseInt(str1[0]),10))*60+ (parseInt(str1[1]),10);
str1 = (pet).split(":");
pet = ((parseInt(str1[0]),10))*60+ (parseInt(str1[1]),10);

if(citime>=pst && citime<=pet){
    fee = (cotime - citime)*pp/ppt
}
else if(citime<pst && citime<pet){
    fee = (cotime - citime)*np/pnt
}
else if(citime>pet && citime<pet){
    fee = (cotime - citime)*np/pnt
}
return fee;
}})

```

### 6.18 addCamera()

```

var camera = new camera();
camera.id=1;
camera.name= "Camera 1";
camera.location= "en1";
camera.save(function (err, user) {
    if (err) {
        console.log('DB ERROR')
    }
    else {
        console.log(camera)
    }
})
})

```

### 6.19 removeCamera()

```

camera.findByIdAndDelete('5ddc182b3fb61a0761f85887', function (err) {
    if (err) return err;
    res.send("Successfully deleted")
});

```

## 7. Software requirements traceability matrix

The table below maps each functional requirement to the class (design component) and its function (component item) which implements that FR. The design component and component items are shown in class diagram.

**Table 2 Requirements Traceability Matrix**

<b>Req. Number</b>	<b>Ref. Item</b>	<b>Design Component</b>	<b>Component Items</b>
FR-1.1	UC-1	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class User Profiling Sequence Diagram
FR-1.2	UC-1	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class User Profiling Sequence Diagram
FR-2.1	UC-2	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class User Profiling Sequence Diagram
FR-3.1	UC-3	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class User Profiling Sequence Diagram
FR-4.1	UC-4	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class User Profiling Sequence Diagram
FR-5.1	UC-5	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class
FR-5.2	UC-5	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class User Profiling Sequence Diagram

FR-5.3	UC-5	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Person Class User Profiling Sequence Diagram
FR-6.1	UC-6	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class Configuration Sequence Diagram
FR-6.2	UC-6	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class, Floor Class Configuration Sequence Diagram
FR-6.3	UC-6	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class, Floor Class, Lane Class Configuration Sequence Diagram
FR-6.4	UC-6	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class Configuration Sequence Diagram
FR-6.5	UC-6	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class Configuration Sequence Diagram
FR-7.1	UC-7	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class, Floor Class, Lane Class Configuration Sequence Diagram
FR-8.1	UC-8	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class Configuration Sequence

			Diagram
FR-8.2	UC-8	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class, Floor Class Configuration Sequence Diagram
FR-8.3	UC-8	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class, Floor Class, Lane Class Configuration Sequence Diagram
FR-8.4	UC-8	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class Configuration Sequence Diagram
FR-8.5	UC-8	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram Map Class Configuration Sequence Diagram
FR-9.1	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-9.2	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-9.3	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-9.4	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence

			Diagram
FR-9.5	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-9.6	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-9.7	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-9.8	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-9.9	UC-9	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-10.1	UC-10	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.1	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.2	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram

FR-11.3	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.4	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.5	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.6	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.7	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.8	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-11.9	UC-11	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-12.1	UC-12	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram

FR-12.2	UC-12	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-13.1	UC-13	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-14.1	UC-14	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-14.2	UC-14	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-14.3	UC-14	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-14.4	UC-14	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-14.5	UC-14	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram
FR-15.1	UC-15	Activity Diagram Class Diagram Sequence Diagram	User Profiling Activity Diagram Operator Class User Profiling Sequence Diagram



FR-16.1	UC-16	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Admin) Activity Diagram VehicleArchive Class Finance Management (Admin) Sequence Diagram
FR-16.2	UC-16	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Admin) Activity Diagram VehicleArchive Class Finance Management (Admin) Sequence Diagram
FR-16.3	UC-16	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Admin) Activity Diagram VehicleArchive Class Finance Management (Admin) Sequence Diagram
FR-16.4	UC-16	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Admin) Activity Diagram VehicleArchive Class Finance Management (Admin) Sequence Diagram
FR-16.5	UC-16	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Admin) Activity Diagram VehicleArchive Class Finance Management (Admin) Sequence Diagram
FR-16.6	UC-16	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Admin) Activity Diagram VehicleArchive Class Finance Management (Admin) Sequence Diagram
FR-17.1	UC-17	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Admin) Activity Diagram VehicleArchive Class Finance Management (Admin) Sequence Diagram
FR-18.1	UC-18	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram FeePackage Class, Fee Class Configuration Sequence Diagram

FR-19.1	UC-19	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram FeePackage Class, Fee Class Configuration Sequence Diagram
FR-19.2	UC-19	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram FeePackage Class, Fee Class Configuration Sequence Diagram
FR-19.3	UC-19	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram FeePackage Class, Fee Class Configuration Sequence Diagram
FR-19.4	UC-19	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram FeePackage Class Configuration Sequence Diagram
FR-19.5	UC-19	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram FeePackage Class Configuration Sequence Diagram
FR-19.6	UC-19	Activity Diagram Class Diagram Sequence Diagram	Configuration Activity Diagram FeePackage Class Configuration Sequence Diagram
FR-20.1	UC-20	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (Admin) Activity Diagram Map Class Automobile Surveillance (Admin) Sequence Diagram
FR-21.1	UC-21	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (Admin) Activity Diagram Map Class Automobile Surveillance (Admin) Sequence Diagram

FR-21.2	UC-21	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (Admin) Activity Diagram Map Class Automobile Surveillance (Admin) Sequence Diagram
FR-21.3	UC-21	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (Admin) Activity Diagram Map Class Automobile Surveillance (Admin) Sequence Diagram
FR-21.4	UC-21	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (Admin) Activity Diagram Map Class Automobile Surveillance (Admin) Sequence Diagram
FR-21.5	UC-21	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (Admin) Activity Diagram Map Class Automobile Surveillance (Admin) Sequence Diagram
FR-22.1	UC-22	Activity Diagram Class Diagram Sequence Diagram	Parking Space Allocation Activity Diagram VehicleParking Class Parking Space Allocation Sequence Diagram
FR-22.2	UC-22	Activity Diagram Class Diagram Sequence Diagram	Parking Space Allocation Activity Diagram VehicleParking Class Parking Space Allocation Sequence Diagram
FR-23.1	UC-23	Activity Diagram Class Diagram Sequence Diagram	Parking Space Allocation Activity Diagram VehicleParking Class Parking Space Allocation Sequence Diagram
FR-24.1	UC-24	Activity Diagram Class Diagram Sequence Diagram	Parking Space Allocation Activity Diagram Map Class Parking Space Allocation Sequence Diagram
FR-24.2	UC-24	Activity Diagram	Parking Space Allocation

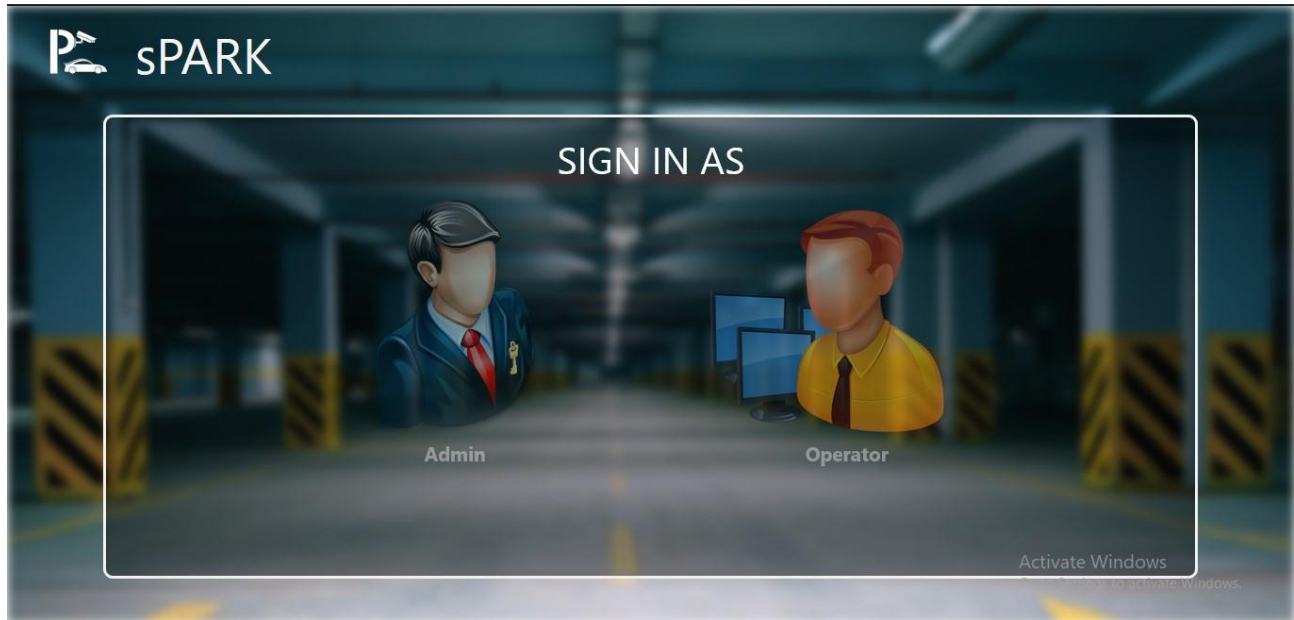
		Class Diagram Sequence Diagram	Activity Diagram Map Class Parking Space Allocation Sequence Diagram
FR-25.1	UC-25	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (System) Activity Diagram VehicleParking Class Automobile Surveillance (System) Sequence Diagram
FR-26.1	UC-26	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (System) Activity Diagram FeeProcessing Class Automobile Surveillance (System) Sequence Diagram
FR-27.1	UC-27	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (System) Activity Diagram VehicleParking Class Automobile Surveillance (System) Sequence Diagram
FR-27.2	UC-27	Activity Diagram Class Diagram Sequence Diagram	Automobile Surveillance (System) Activity Diagram VehicleParking Class Automobile Surveillance (System) Sequence Diagram
FR-28.1	UC-28	Activity Diagram Class Diagram Sequence Diagram	Parking Space Allocation Activity Diagram VehicleParking Class Parking Space Allocation Sequence Diagram
FR-28.2	UC-28	Activity Diagram Class Diagram Sequence Diagram	Parking Space Allocation Activity Diagram VehicleParking Class Parking Space Allocation Sequence Diagram
FR-29.1	UC-29	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram FeeProcessing Class Finance Management (Operator) Sequence Diagram
FR-29.2	UC-29	Activity Diagram Class Diagram	Finance Management (Operator) Activity Diagram FeeProcessing Class

		Sequence Diagram	Finance Management (Operator) Sequence Diagram
FR-30.1	UC-30	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram FeeProcessing Class Finance Management (Operator) Sequence Diagram
FR-30.2	UC-30	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram FeeProcessing Class Finance Management (Operator) Sequence Diagram
FR-31.1	UC-31	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-32.1	UC-32	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-32.2	UC-32	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-33.1	UC-33	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-33.2	UC-33	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-33.3	UC-33	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-33.4	UC-33	Activity Diagram	Finance Management (Operator) Activity Diagram

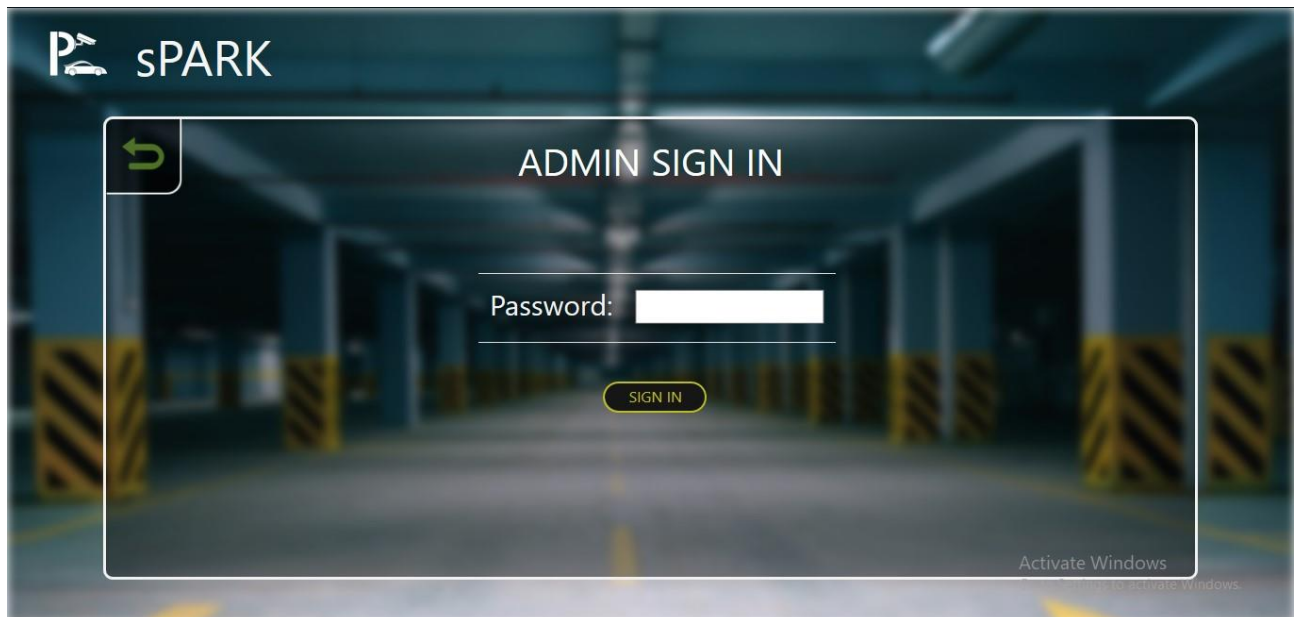
		Class Diagram Sequence Diagram	VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-33.5	UC-33	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class Finance Management (Operator) Sequence Diagram
FR-33.6	UC-33	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class
FR-33.7	UC-33	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Sequence Diagram
FR-33.8	UC-33	Activity Diagram Class Diagram Sequence Diagram	Finance Management (Operator) Activity Diagram VehicleArchive Class

## 8. Human interface design

### 8.1 Screen images



Screen 1: Sign in as



Screen 2: Admin Sign in

## ADD MAP

Enter No Of Floors

Enter No Of Entrance Posts

Enter No Of Exit Posts

NEXT

Activate Windows  
Go to Settings to activate Windows.

Screen 3: Add Map (Screen I)

## ADD MAP

←

ENTER DETAILS FOR FLOOR I

Enter No Of Lanes

GENERATE LANES


LANE	SIZE (meters)

NEXT

Activate Windows  
Go to Settings to activate Windows.

Screen 4: Add Map (Screen II)





- MAP
- PARKING LOT
- OPERATORS
- VEHICLES' ARCHIVE
- CHANGE PASSWORD

SIGN OUT

## UPDATE FEE

Day

Peak Hours Start Time

--:--:--

Peak Hours Start Time

--:--:--

Peak Hours Rate

/

(Rs/min)

Normal Hours Rate

/

(Rs/min)


Fine

Rs

SAVE FEE

Activate Windows  
Go to Settings to activate Windows.

Screen 5: Update Fee



- MAP
- PARKING LOT
- OPERATORS
- VEHICLES' ARCHIVE
- CHANGE PASSWORD

SIGN OUT

## ADD OPERATOR

First Name

Last Name

Username

Password

Image

Browse...

No file selected.

Duty Start Time

--:--:--

Duty End Time

--:--:--

Duty Post

en1



SAVE OPERATOR

Activate Windows  
Go to Settings to activate Windows.

Screen 6: Add Operator

sPARK		OPERATORS' LIST		
MAP PARKING LOT OPERATORS VEHICLES' ARCHIVE CHANGE PASSWORD SIGN OUT	SR#	IMAGE	NAME	
	1		Sawaiz	<a href="#">Details</a> <a href="#">Update</a> <a href="#">Delete</a>
	2		Bilal	<a href="#">Details</a> <a href="#">Update</a> <a href="#">Delete</a>
	3		Omair	<a href="#">Details</a> <a href="#">Update</a> <a href="#">Delete</a>
	4		Jawad	<a href="#">Details</a> <a href="#">Update</a> <a href="#">Delete</a>
	5		Talal	<a href="#">Details</a> <a href="#">Update</a> <a href="#">Delete</a>

Screen 7: Operators' List

sPARK		PARKING LOT STATUS				
MAP PARKING LOT OPERATORS VEHICLES' ARCHIVE CHANGE PASSWORD SIGN OUT	 Refresh		 Filter			
	SR#	FLOOR	LANE	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE
	SR#	FLOOR	Lane	CHECK IN TIME	LICENSE PLATE NUMBER	FINE

Screen 8: Parking Lot Status

## 8.2 Screen objects and actions

### 8.2.1 Screen 1:

It is the first screen of the application which is displayed when the application is launched. It gives user “Sign in as” option to select according to the type of the user i.e. Admin or operator. It has two objects which will allow the user to interact with system which performs the following functionality:

**Object 1: Button (Admin)**

**Functionality:** Displays the sign in page for the admin when pressed.

**Object 2: Button (Operator)**

**Functionality:** Displays the sign in page for operator when pressed.

### 8.2.2 Screen 2:

This screen is displayed for Admin sign in which will allow the admin to enter his/her password and press “Sign in” button to proceed to their portal. Its objects are as follows:

**Object 1: Textfield (Password)**

**Functionality:** Allows the admin to write his/her password here.

**Object 2: Button (Sign in)**

**Functionality:** Proceeds to validation of admin password and moving to admin portal when pressed.

### 8.2.3 Screen 3:

This is the first screen to add map of the parking lot. It allows the admin to enter number of floors, number of entrance floors and number of exit posts of the parking lot. Its objects are as follows:

**Object 1: Textfield (No. of floors)**

**Functionality:** Allows the admin to write number of floors of the parking lot here.

**Object 2: Textfield (No. of entrance posts)**

**Functionality:** Allows the admin to write number of entrance posts of the parking lot here.

**Object 3: Textfield (No. of exit posts)**

**Functionality:** Allows the admin to write number of exit posts of the parking lot here.

**Object 4: Button (Enter)**

**Functionality:** Proceeds to the next screen for entering other information of the parking lot map when pressed.

#### 8.2.4 Screen 4:

This is the second screen to add map of the parking lot. It allows the admin to enter number of lanes and name and size for each lane of the parking lot. Its objects are as follows:

**Object 1: Textfield (No. of lanes)**

**Functionality:** Allows the admin to write number of lanes for a specific floor here.

**Object 2: Button (Generate lanes)**

**Functionality:** It generates entered number of lane fields when pressed, to enter name and size of each lane.

**Object 3: Textfield (lane name)**

**Functionality:** Allows the admin to write name of each lane of one floor here.

**Object 4: Textfield (lane size)**

**Functionality:** Allows the admin to write lane size for each lane of one floor here.

**Object 5: Button (Next)**

**Functionality:** Proceeds to the next screen for entering other information of the parking lot map when pressed.

**Object 6: Button (back icon)**

**Functionality:** Take admin to the previous screen of entering information of the parking lot map when pressed.

#### 8.2.5 Screen 5:

This screen is for updating the fee charges of the parking lot according to the required days and set their fees charges according to the need. Its objects are as follows:

##### 8.2.5.1 Side navigation bar objects:

**Object 1: Button (Map)**

**Functionality:** It opens Map information for the admin at the center right div of this screen, where “Add Fee” form is displayed.

**Object 2: Dropdown (Parking lot)**

**Functionality:** It provides dropdown list showing parking lot functionalities for the admin when pressed.

**Object 3: Dropdown (Operators)**

**Functionality:** It provides dropdown list showing Operators functionalities for the admin when pressed.

**Object 4: Button (Vehicles' Archive)**

**Functionality:** It opens Map information for the admin at the center right div of this screen, where “Add Fee” form is displayed.

**Object 5: Button (Change Password)**

**Functionality:** It opens Changing password form for the admin at the center right div of this screen, where “Add Fee” form is displayed.

**Object 6: Button (Sign out)**

**Functionality:** It signs out the admin from his/her portal.

**8.2.5.2 Update fee form objects:****Object 1: Textfield (day)**

**Functionality:** Allows to enter days for which fee needs to be added.

**Object 3: Textfield (peak hours start time)**

**Functionality:** Allows the admin to enter peak hours start time of the parking lot.

**Object 4: Textfield (peak hours end time)**

**Functionality:** Allows the admin to enter peak hours end time of the parking lot

**Object 5: Textfield (peak hours rate)**

**Functionality:** Allows the admin to enter peak hours rate per unit time.

**Object 6: Textfield (normal hours rate)**

**Functionality:** Allows the admin to enter normal hours rate per unit time.

**Object 7: Textfield (fine)**

**Functionality:** Allows the admin to enter amount of fine of the parking lot.

**Object 8: Button (Save fee)**

**Functionality:** Validates the entered field and updates fee of the parking lot when pressed.

**8.2.6 Screen 6:**

This screen is for adding a new operator account into the system. It allows the admin to enter the required fields and save operator record by pressing ‘Save Operator’ button. Its objects are as follows:

**8.2.6.1 Side navigation bar objects:**

Same as discussed in section 8.2.5.1 of this document.

**8.2.6.2 Add operator form objects****Object 1: Textfield (First Name):**

**Functionality:** Allows the admin to enter first name of the operator here.

**Object 2: Textfield (Last Name):**

**Functionality:** Allows the admin to enter last name of the operator here.

**Object 3: Textfield (Username):**

**Functionality:** Allows the admin to enter username of the operator here.

**Object 4: Textfield (Password):**

**Functionality:** Allows the admin to set password of the operator here.

**Object 5: Browse file (Image):**

**Functionality:** Allows the admin to select file to set image of the operator.

**Object 6: Timefield (Duty start time):**

**Functionality:** Allows the admin to set duty start time of the operator.

**Object 7: Timefield (Duty end time):**

**Functionality:** Allows the admin to set duty end time of the operator.

**Object 8: Dropdown (Duty post):**

**Functionality:** Allows the admin to set duty post of the operator from the dropdown list.

**Object 9: Button (Save operator):**

**Functionality:** Validates the entered field and adds a new operator in the system.

### 8.2.7 Screen 7:

This screen allows the admin to view the list of operators and shows their image and name and also provides buttons to check details, update details or delete record of any operator from the list.

#### 8.2.7.1 Side navigation bar objects:

Same as discussed in section 8.2.5.1 of this document.

#### 8.2.7.2 Operator's list objects:

**Object 1: Button (Details):**

**Functionality:** Display details of specific operator to the admin when pressed.

**Object 2: Button (Update):**

**Functionality:** Display form for updating information of specific operator to the admin when pressed.

**Object 3: Button (Delete):**

**Functionality:** Allows the admin to delete record of a specific operator from the system.

**Object 4: Scroll bar:**

**Functionality:** Allows the admin to move the list of operators up or down through up and down arrows or through scrolling bar itself.

### 8.2.8 Screen 8:

This screen allows the admin to view the parking lot status which shows the floor, lane, check in time, license plate number, fine and size of the vehicles that are currently parked in the parking lot.

#### 8.2.8.1 Side navigation bar objects:

Same as discussed in section 8.2.5.1 of this document.

#### 8.2.8.2 Parking lot status list objects:

**Object 1: Button (Refresh):**

**Functionality:** Allows the admin to refresh the list of parking lot status by pressing it.

**Object 2: Button (Filter):**

**Functionality:** Allows the admin to filter results of the list of parking lot status by pressing it.

## 9. References

1. Scott W. Ambler, Agile Models Distilled: Potential Artifacts for Agile Modeling, <http://agilemodeling.com/artifacts/>, 02 Dec. 2019
2. Ian Sommerville, (2011), Software Engineering, Addison-Wesley, Chapter 6, pp155-157.
3. Ian Sommerville, (2011), Software Engineering, Addison-Wesley, Chapter 5, pp136-138.

## 10. Plagiarism Report

