**Cairo University  
Faculty of Computers and Artificial Intelligent** 

**Software Engineering Program**

**SCS252 - Software Modeling**

**Parking Garage application**

Software Design Specifications (SDS)

Team Names

1. Mansour Sherif Mohamed

2. Noor Eldien Mahmoud Abdelhamid

3. Omar Hisham Saad Mohamed

4. Abdelrahman Mohamed Abdelhamed

May 2025

**Contents**

[Team 3](#_Toc198578405)

[Document Purpose and Audience 3](#_Toc198578406)

[System Models 4](#_Toc198578407)

[I. Class diagrams 4](#_Toc198578408)

[Important Algorithm 6](#_Toc198578409)

[II. Sequence diagrams 7](#_Toc198578410)

[● Pick a Free Spot 8](#_Toc198578411)

[Class - Sequence Usage Table 10](#_Toc198578412)

[● Pick a Free Spot 10](#_Toc198578413)

[Ownership Report 11](#_Toc198578414)

[Policy Regarding Plagiarism: 11](#_Toc198578415)

# Team

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20236109 | Noor Eldien Mahmoud Abdelhamid | noormahmoud747@gmail.com | 0 115 879 7683 |
| 20236102 | Mansour Sherif Mohamed | mansoursherif5566@gmail.com | 0 100 672 9980 |
| 20236067 | Omar Hisham Saad Mohamed | omarmekky1973@gmail.com |  |
| 20236055 | Abdelrahman Mohamed Abdelhamed |  | 0 155 866 6278 |

# Document Purpose and Audience

This document provides a description of the static structure of the System to be Implemented, as well as the dynamic behavior between different static structures in the System. It also breaks down complex algorithms used during System Operation and provides detailed Mapping between Structures used in the system, and their contribution to the dynamic behavior of the System.

This document is more curated to the level of understanding of General readers, as it provides a simple graphical representation for all complex operations, as well as a description of complex algorithms. It is intended to be read by Customers, and Managers.

# System Models

## Diagram Description automatically generatedClass diagrams

| **Class ID** | **Class Name** | **Description & Responsibility** |
| --- | --- | --- |
| C.1 | User Interface | The main boundary class that interacts with Users of the System, including the main function and other input / output functions |
| C.2 | Garage Controller | A control class that contains many of the main Garage **functions**, and **keeps track** of the Garage slots, income, and other variables related to the garage. |
| C.3 | Garage | A class that contains the **basic information** related to the Garage, as well as the different controllers of the garage. **Calls different methods of those controllers.** |
| C.4 | Parking Space | A class that contains the **basic information** of a Parking Space, such as: A unique ID, Dimensions of the Parking Space, and The Vehicle Parked Inside it. And has a basic setter used to insert Vehicles into the Space. |
| C.5 | ParkInController | A control class that responsible for finding a free spot and **assigning the Vehicle**. |
| C.6 | SpotFindAlgorithm | An **interface** that is implemented by different concrete algorithms used to find a free parking spot for a Vehicle. |
| C.7 | FirstComeFirstServe | A class that **inherits** from **SpotFindAlgorithm** , and defines the getSpot function to define a spot finding algorithm. |
| C.8 | BestFit | A class that **inherits** from **SpotFindAlgorithm** , and defines the getSpot function to define a spot finding algorithm. |
| C.9 | ParkOutController | A control class **responsible** **for** **parking out**, by calculating its fees and freeing its spot. |
| C.10 | Payment Controller | A control class that simulates payment of Parking Fees via Cash. |
| C.11 | Vehicle | A class that **represents** an **abstraction** of a Vehicle, **including only the parts related to the System**, such as: id, modelName, modelYear, Dimensions, and TimeStamps of arrival and departure |
| C.12 | Date Log | A class responsible for **storing and processing Vehicle arrival and departure times**, where it stores time in milliseconds to be retrieved when calculating time difference in hours during fee calculation. |
| C.13 | Dimensions | A class responsible for **storing** **Vehicle** **and Parking Space** **Width** **and** **Depth** and determining whether a Space fits a particular Vehicle or not. |

### Important Algorithm

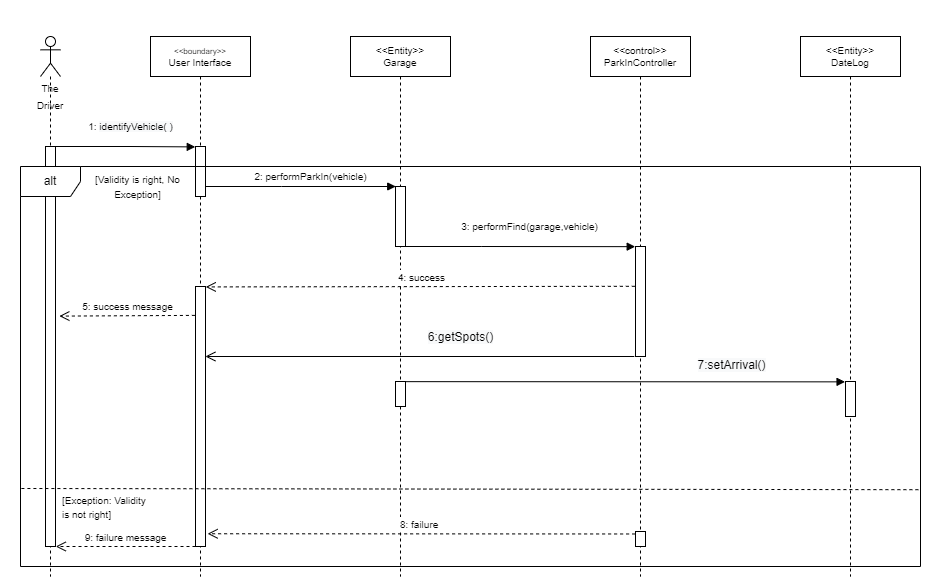
* **First Come, First Serve Spot Finder:**

1. Loops over all parking spaces in the garage
2. If it finds a space whose dimensions fit the Vehicle, it returns its index
3. If the loop is done and no vehicles are found, it returns **-1** to indicate none was found

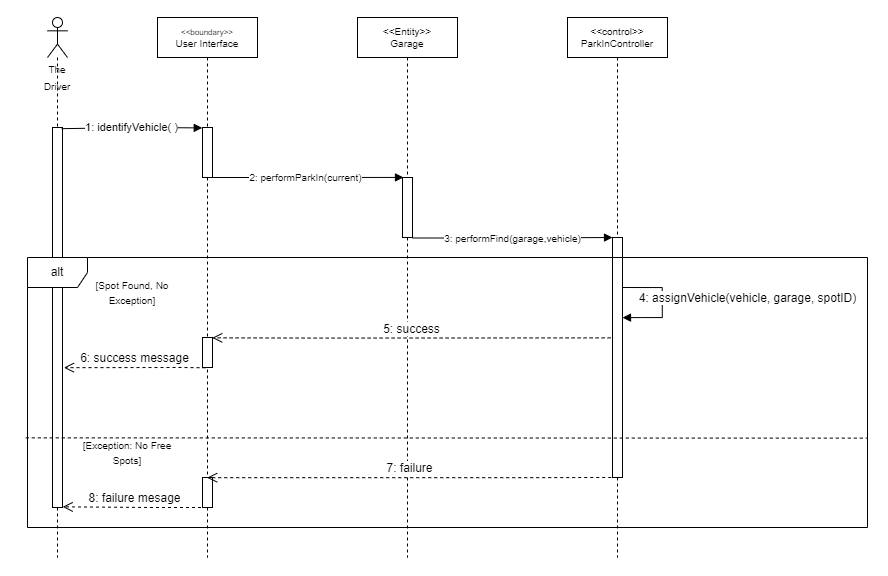
* **Best-Fit Spot Finder:**

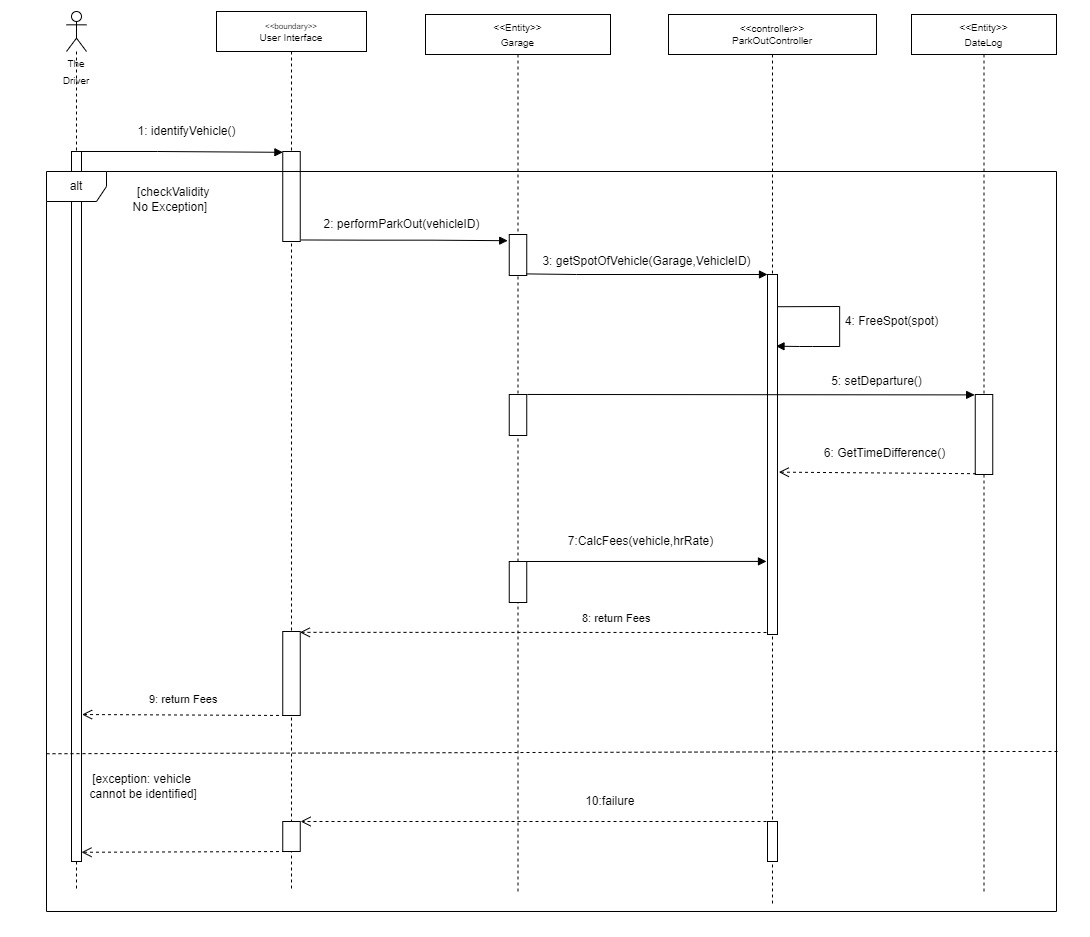
1. Considers the minimum area as the max double value
2. Loops over all parking spaces in the garage looking for the minimum space
3. If it finds a space whose area is smaller than the current minimum area, and the dimensions of that space fit the vehicle, it considers it the new minimum space
4. In the end, returns the index of the minimum space, or **-1** if none was found

## II. Sequence diagrams

* **Park-In**

### Pick a Free Spot (choose approach)



* **Park Out**

### Class - Sequence Usage Table

* **Park-In**

| **Class Name** | **Sequence Diagrams** | **Overall used methods** |
| --- | --- | --- |
| 1. User Interface | 1 | identifyVehicle() |
| 1. Garage | 2,6 | performParkIn(vehicle),  getSpots() |
| 1. ParkInController | 3 | performFind(garage,vehicle), |
| 1. DateLog | 7 | setArrival() |

### Pick a Free Spot

| **Class Name** | **Sequence Diagrams** | **Overall used methods** |
| --- | --- | --- |
| 1. User Interface | 1 | identifyVehicle() |
| 1. Garage | 2 | performParkIn(current), |
| 1. ParkInController | 3,4 | performFind(garage,vehicle),  assignVehicle(vehicle, garage, spotID) |

## 

* **Park-out**

| **Class Name** | **Sequence Diagrams** | **Overall used methods** |
| --- | --- | --- |
| 1. User Interface | 1 | identifyVehicle() |
| 1. Garage | 2, | performParkOut(vehicleID) |
| 1. ParkOutController | 3,4,7 | getSpotOfVehicle(Garage,VehicleID), FreeSpot(spot)  CalcFees(vehicle,hrRate) |
| 1. DateLog | 5,6 | setDeparture() GetTimeDifference() |

# Ownership Report

|  |  |
| --- | --- |
| **Item** | **Owners** |
| [Document Purpose and Audience](https://docs.google.com/document/d/1VKlXl588y6CN30zXPSnnmS9JIKr2I93m/edit#heading=h.1fob9te) | All |
| [Class diagrams](https://docs.google.com/document/d/1VKlXl588y6CN30zXPSnnmS9JIKr2I93m/edit#heading=h.2et92p0) | All |
| [Sequence diagrams](https://docs.google.com/document/d/1VKlXl588y6CN30zXPSnnmS9JIKr2I93m/edit#heading=h.3dy6vkm) | All |

# Policy Regarding Plagiarism:

**Students have collective ownership and responsibility of their project. Any violation of academic honesty will have severe consequences and punishment for ALL team members.**

1. تشجع الكلية على مناقشة الأفكار و تبادل المعلومات و مناقشات الطلاب حيث يعتبر هذا جوهريا لعملية تعليمية سليمة
2. ساعد زملاءك على قدر ما تستطيع و حل لهم مشاكلهم فى الكود و لكن تبادل الحلول غير مقبول و يعتبر غشا.
3. أى حل يتشابه مع أى حل آخر بدرجة تقطع بأنهما منقولان من نفس المصدر سيعتبر أن صاحبيهما قد قاما بالغش.
4. قد توجد على النت برامج مشابهة لما نكتبه هنا أى نسخ من على النت يعتبر غشا يحاسب عليه صاحبه.
5. إذا لم تكن متأكدا أن فعلا ما يعد غشا فلتسأل المعيد أو أستاذ المادة.
6. فى حالة ثبوت الغش سيأخذ الطالب سالب درجة المسألة ، و فى حالة تكرار الغش سيرسب الطالب فى المقرر.