Cairo University  
Faculty of Computers and Artificial Intelligent

**CS251**

**Software Engineering I**

Project Name

Software Design

Team Names

Month & Year

Contents

[Instructions [To be removed] **Error! Bookmark not defined.**](#_Toc101814919)

[Team 3](#_Toc101814920)

[Document Purpose and Audience 3](#_Toc101814921)

[System Models 3](#_Toc101814922)

[I. Class diagrams 4](#_Toc101814923)

[Important Algorithm 5](#_Toc101814924)

[II. Sequence diagrams 5](#_Toc101814925)

[Class - Sequence Usage Table 10](#_Toc101814926)

[Ownership Report 18](#_Toc101814927)

[Policy Regarding Plagiarism: 18](#_Toc101814928)

# 

# Team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Name** | | **Email** | | **Mobile** |
| 20200752 | Rana AbdulSalam Mohamed | | ranslam@gmail.com | | 01156263699 |
| 20200689 | Esraa Mahmoud Mohamed Khalifa | | Esraakhalifa5@gmail.com | | 01226320539 |
| 20200742 | Zeyad Mohamed Ali | | zeyadmohamedali@gmail.com | | 01100304690 |
| 20200648 | Youssef Hassan Mostafa | Yousef52hm@gmail.com | | 01203143471 | |

# Document Purpose and Audience

# This document includes SDS illustrations for a parking garage application. It illustrates user interactions and describes the functions done by the system in dynamic fashion.

# This document is meant for client and system developers, but mainly system developers as it sets a structure for and facilitates the software implementation phase.

# System Models

## Class Diagram edit 2.jpgI. Class diagrams

| **Class ID** | **Class Name** | **Description & Responsibility** |
| --- | --- | --- |
| C1 | Display | * This is a **boundary** class that contains a set of buttons, when clicked they perform a certain function and displays system response to the user. |
| C2 | MainSystem | * This is a **controller**that holds information about the entities within the system, and is responsible for communicating commands to between our boundary and entity classes, in addition to creating entity objects and performing system calculations |
| C3 | Garage | * This is an **entity** class that holds the important details of the garage and slots, and deals with all the functionalities that happens within the garage by both driver and owner. |
| C4 | Slot | * This is an **entity** class that holds the information of each slot and deals with the reservations and calculations that are associated with parking. |
| C5 | Vehicle | * This is an **entity** class that holds and manipulates information about the drivers and their vehicles. |
| C6 | Payment | * This is a **controller** class that holds banking information and performs communications between the system and bank services providers. |

**Principles**

Our class respects the SOLID principles to a great extent, and we can see that in each class, where each class includes attribute an functions that are only pertinent to the class and the name it holds.

We can also see that functions that could be done in 2 or more methods in application are made into classes that inherit the common method from the abstract parent class and implements it differently in each child class so that it follow the open closed principle, where adding a new method of applying this function is as easy as creating a new class therefore open for extension, but closed for modification.

Additionally we find that our class diagram also respects the interface segregation principle in the sense that no interface or abstract parent class have functions that are not needed by some or all of their children classes or classes that implement them.

**Design Patterns**

Yes, our class diagram involves design patterns.

We used the strategy design pattern in the Search class, where we removed the search algorithm and from its host class and put it in a separate class so that in the same programming context there might be different algorithms (i.e. strategies), which can be selected in runtime. This pattern is based on Open/closed principle. We don’t need to modify the context [closed for modification], but can choose and add any implementation [open for extension]. We created the Search interface as an abstraction and then we have two different implementations (strategies) the BestFit strategy and the FirstFit strategy

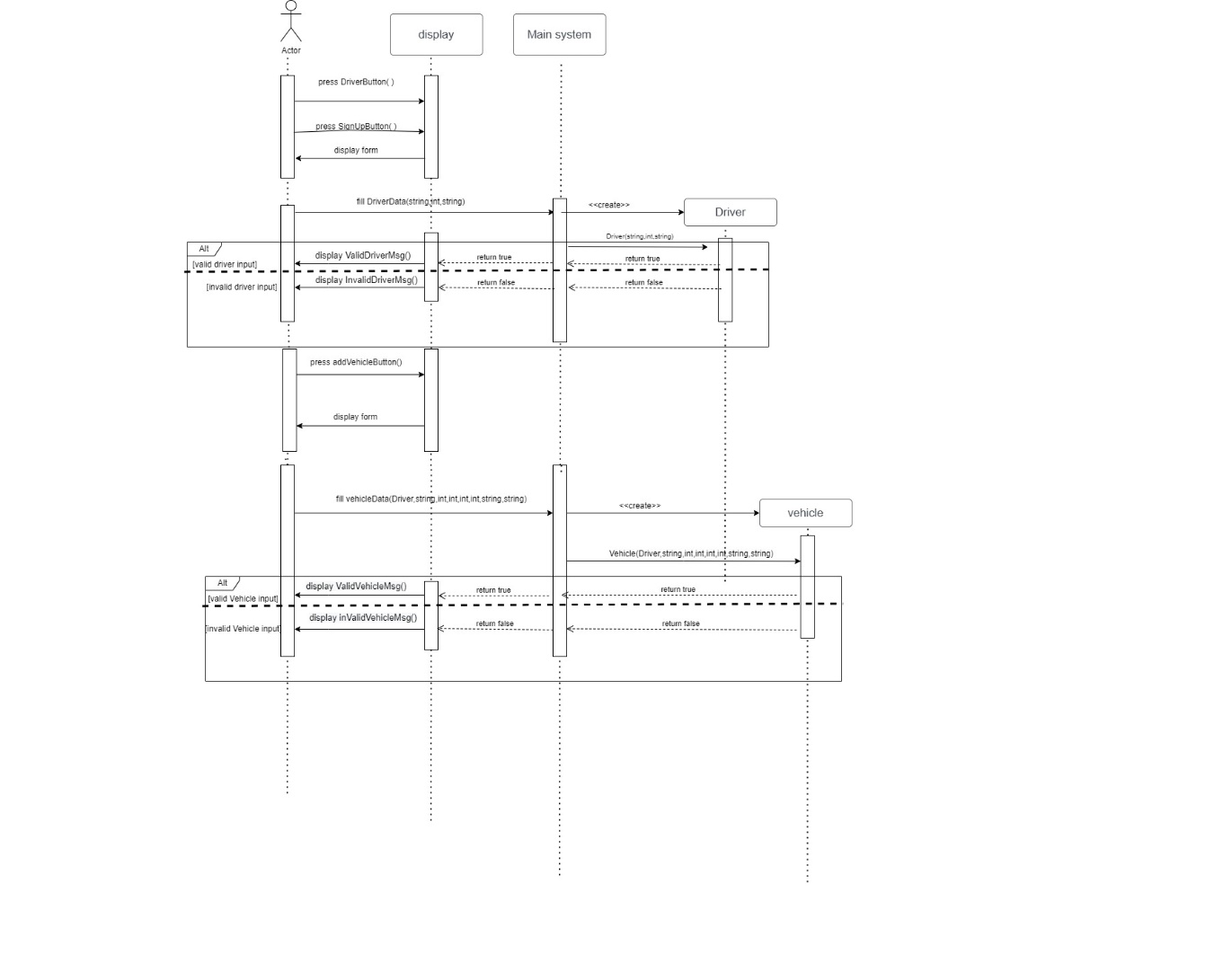
### Important Algorithm

### In Garage class there is a function called “searchSlot” this function searchs for available slots for the vehicle according to the garage configuration, if the configuration of the garage is:

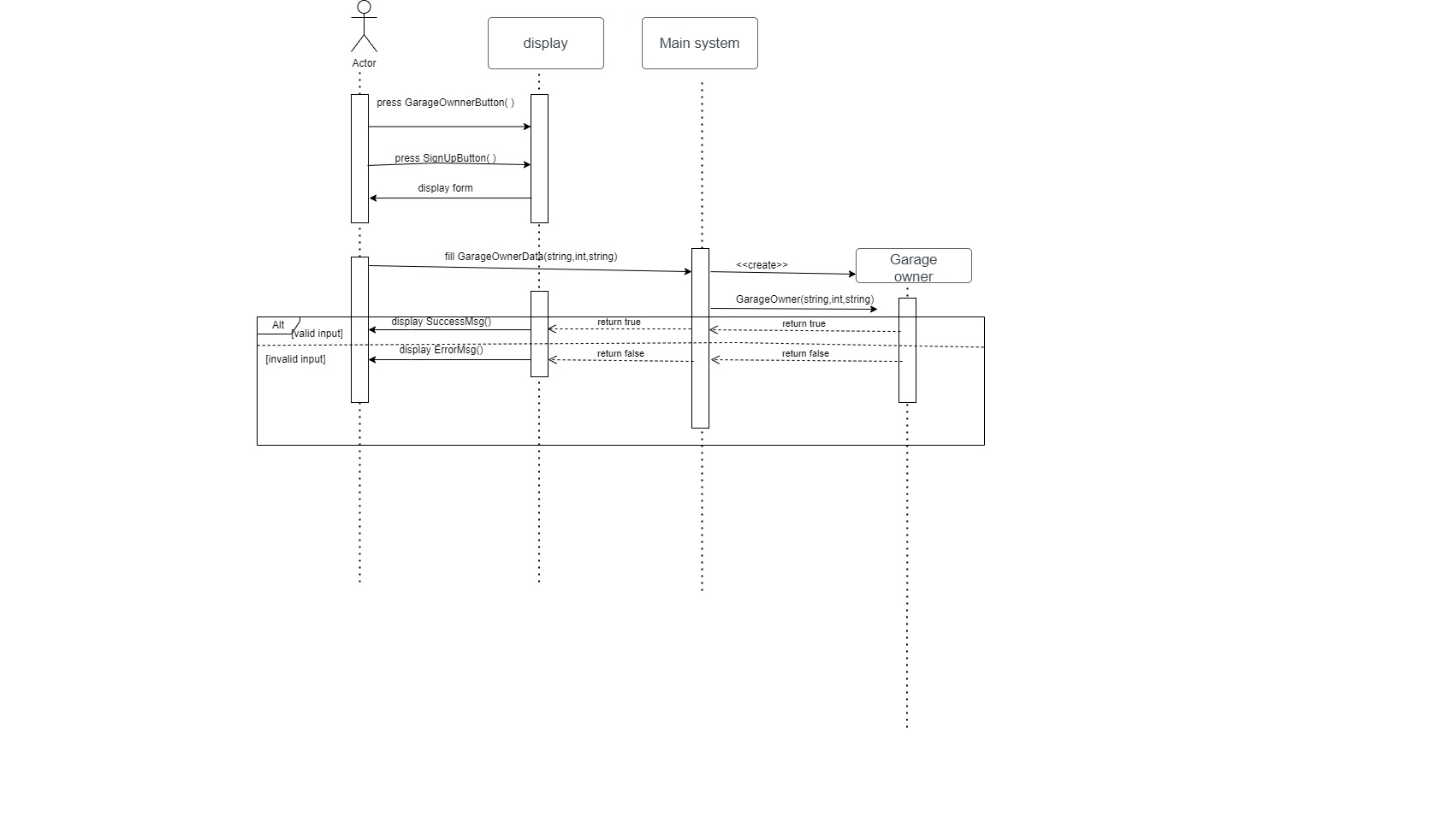
### First come first served: the function will search automatically for the first suitable slot to the vehicle.

### Best-fit configuration: the function will find the best fit slot to the vehicle by finding slot has the average demotions of motorcycle width of 25~45 inches and depth 75~100 inches or the average demotions of car width of 68~78 and depth of 156~197.

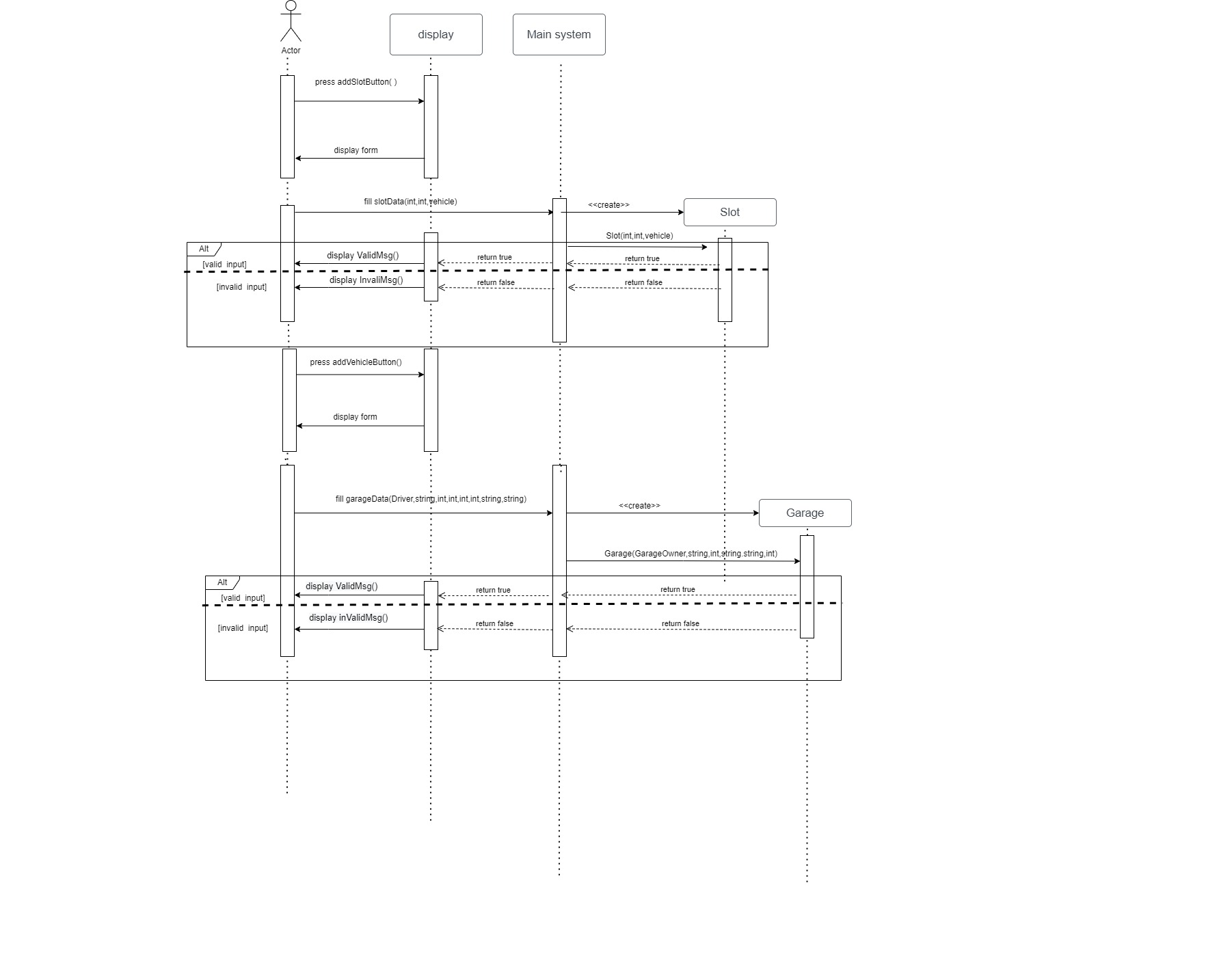
## II. Sequence diagrams



Sequence ID: UC1



Sequence ID: UC2

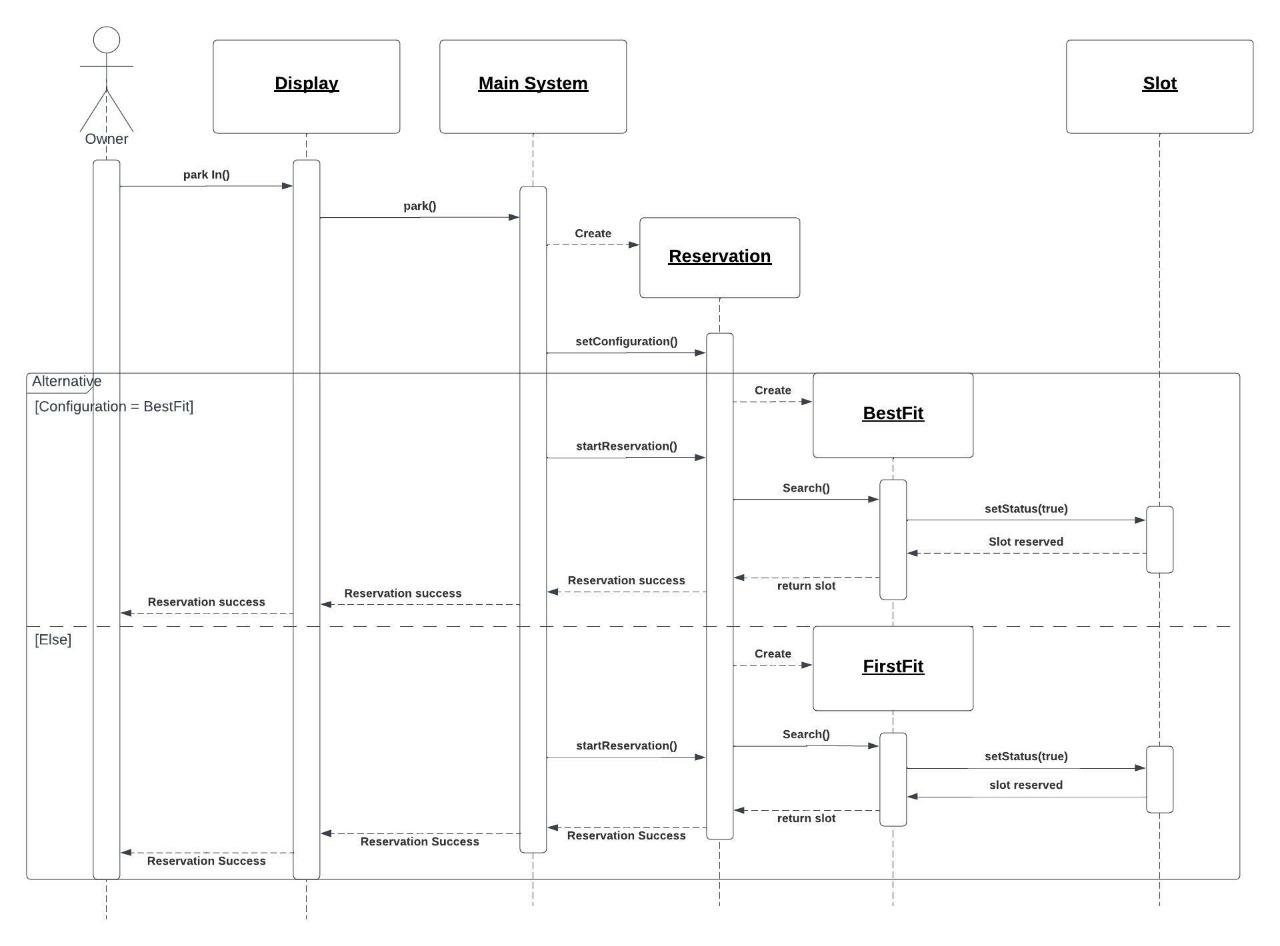
**Sequence ID: UC3**

A picture containing chart

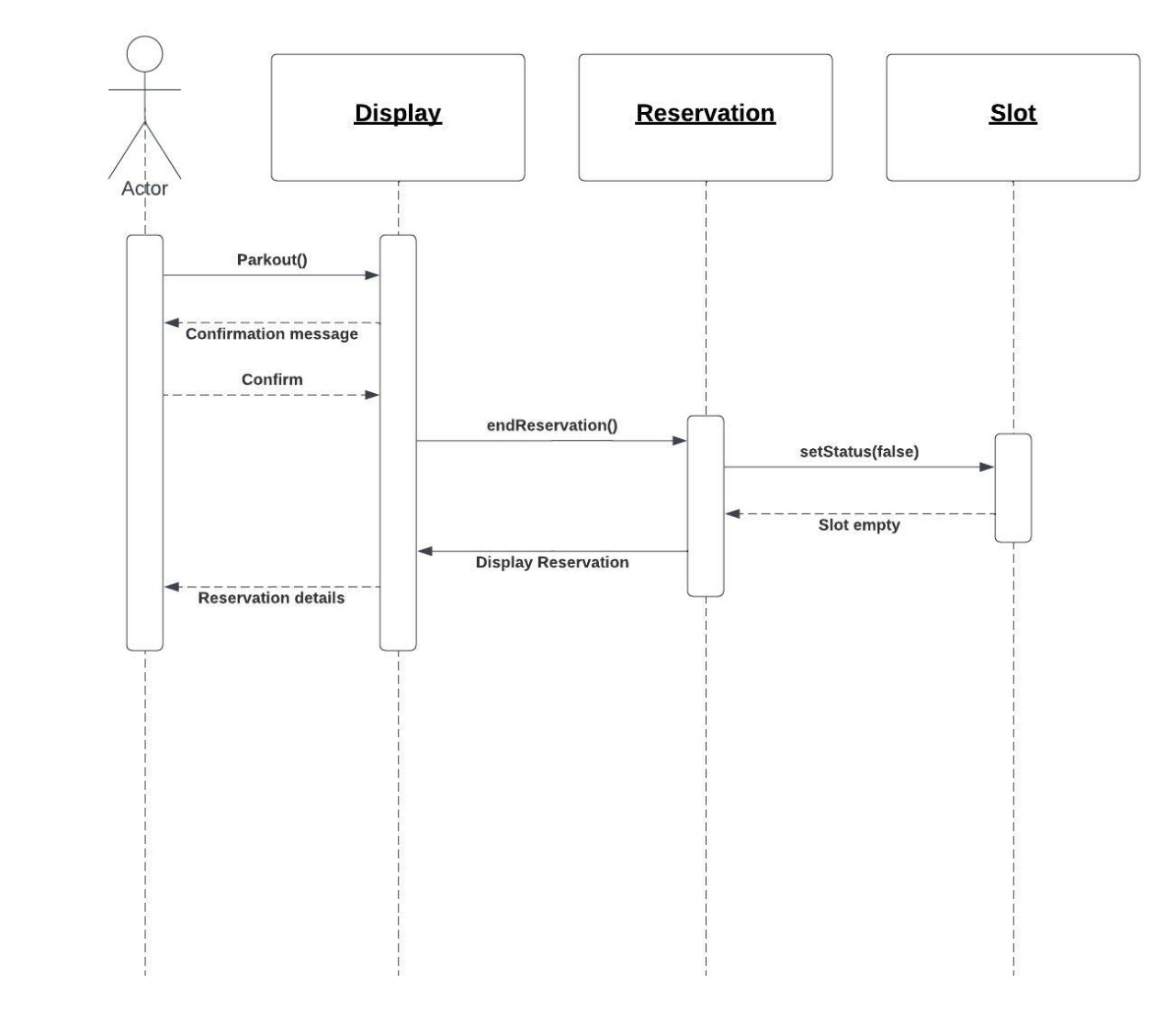
Description automatically generated

Sequence ID: UC4

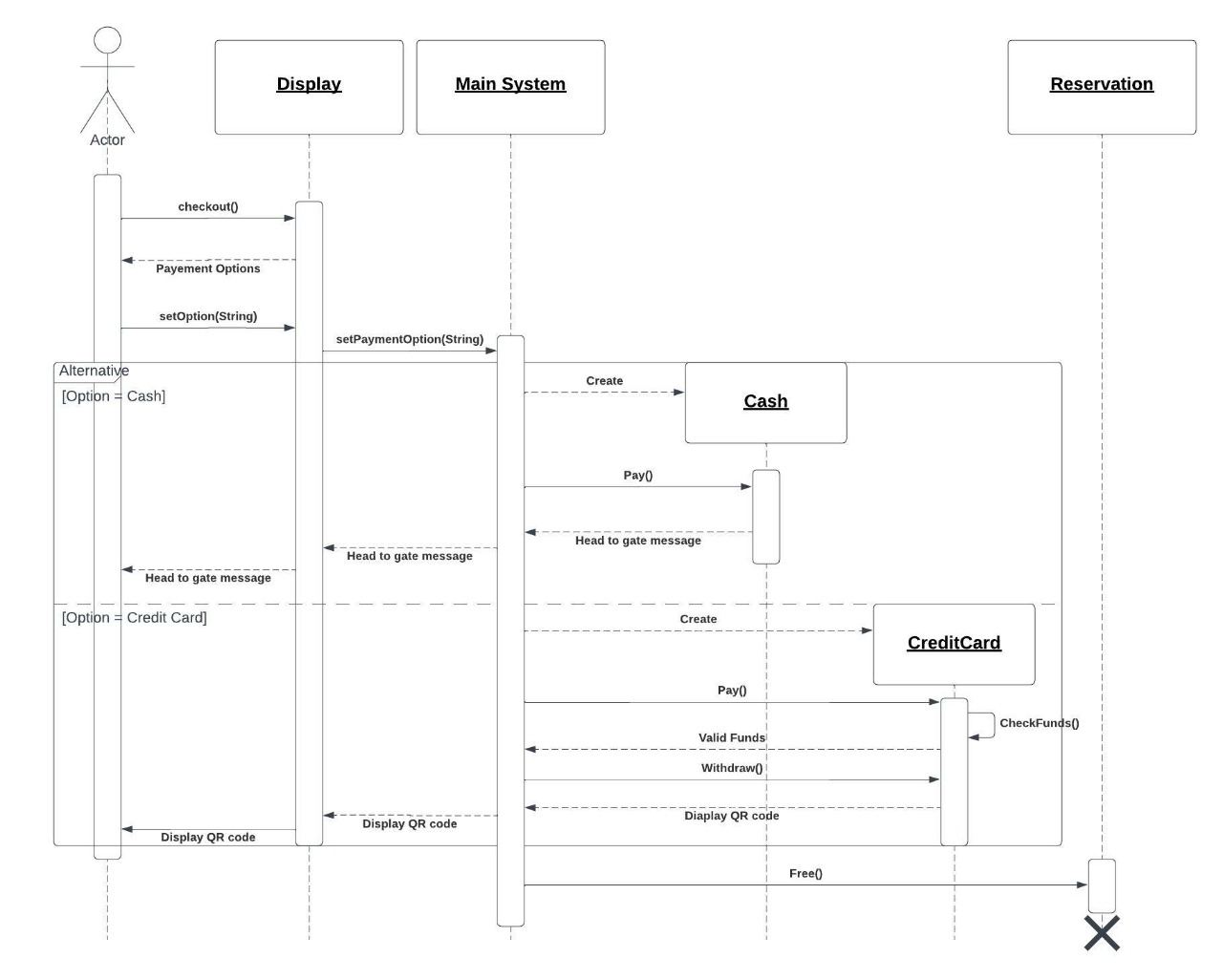
### 



Sequence ID: UC5



Sequence ID: UC6

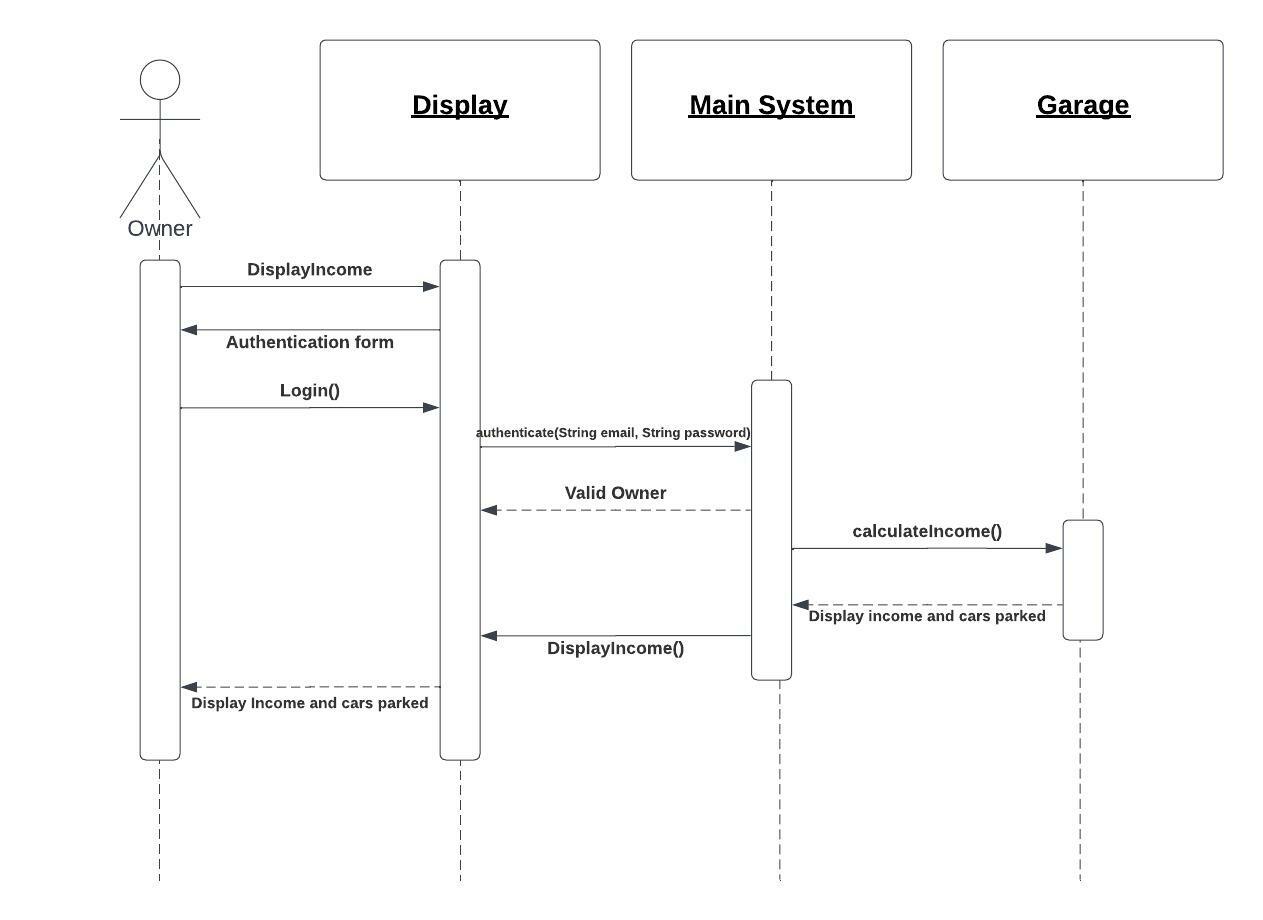


Sequence ID: UC7

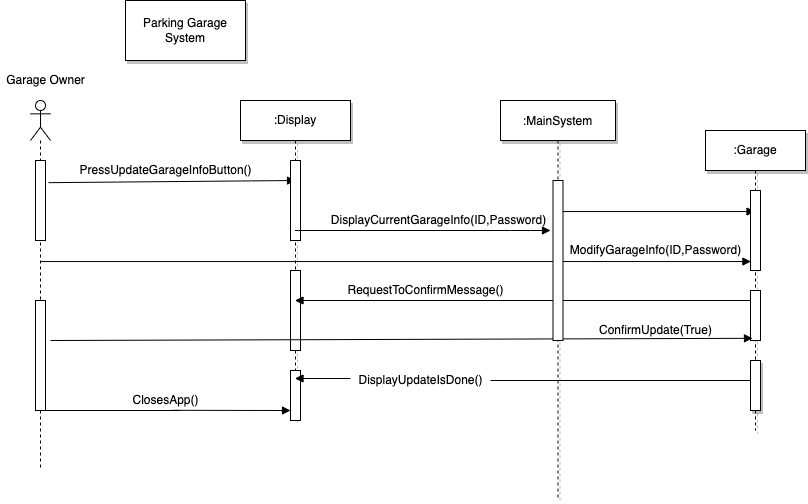
Diagram

Description automatically generated

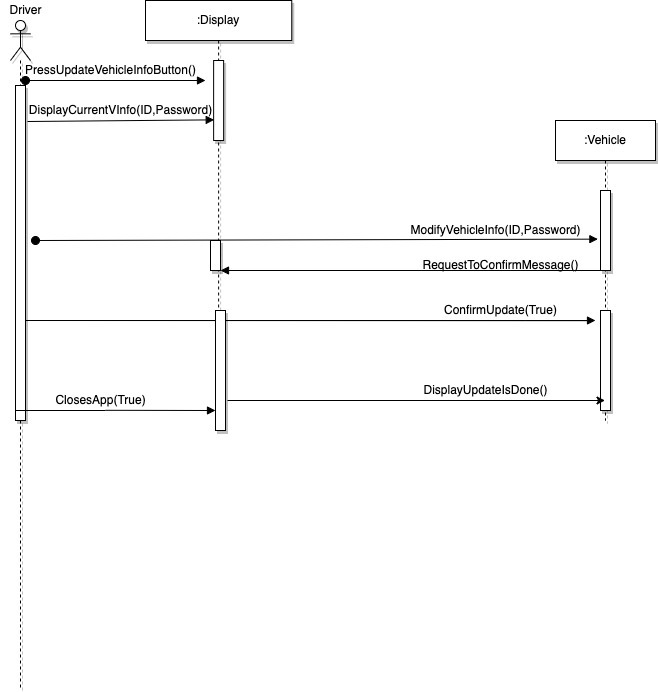
Sequence ID: UC8



Sequence ID: UC9



Sequence ID: UC10



Sequence ID: UC11

### Class - Sequence Usage Table

| **Class Name** | **Sequence Diagrams** | **Overall used methods** |
| --- | --- | --- |
| Display | UC1,UC2, UC3, UC4, UC5, UC6, UC7, UC8, UC9, UC10, UC11 | GarageOwnerButton()  signUpButton()  SuccessMsg()  ErrorMsg()  addGarage()  addSlots()  search\_For\_Garage()  show\_Garage\_location()  show\_Instructions()  ParkIn()  getVehicleForm()  confirmVehicleInfo()  confirmSlot()  parkOut()  checkout()  displayResInfo()  check\_Current\_Charge()  show\_Fees()  Login()  displayIncome()  UpdateGarageInfoButton()  REquestToConfirmMessage()  DisplayUpdateIsDOne()  CloseApp()  PressUpdateVehicleInfoButton()  DisplayCurrentVInfo() |
| MainSystem | UC1,UC2, UC3, UC4, UC5, UC6, UC7, UC9, UC10, | GarageOwnerData()  addGarage()  Search()  ConfirmGarage()  Park()  stopRes()  GetpaymentOptions()  paymentOption()  fillPaymentForm()  Authenticate()  DisplayCurrentGarageInfo() |
| Garage | UC2,UC3,UC5, UC9, UC10 | Garage()  addSlots()  SearchSlot()  Stop()  confirmationTimer()  calculateIncome()  ModifyGarageInfo()  ConfirmUpdate() |
| Slot | 3UC, UC5, UC6, UC7, UC8 | Slot()  SetStatus()  Reserve()  stopTimer()  free()  calcFees() |
| Payment | UC7 | CheckFunds()  Withdraw() |
| Vehicle | UC1,UC11 | Vehicle()  ModifyVehicleInfo()  ConfirmUpdate() |

# Ownership Report

|  |  |
| --- | --- |
| **Item** | **Owners** |
| Class Diagram | *All team members* |
| Sequence Diagrams: UC1, UC2 and UC3 | *Youssef* |
| UC4 and UC8 | *Zeyad* |
| UC5, UC10 and UC11 | *Esraa* |
| UC6, UC7 and UC9 | *Rana* |
| Code Implementation: Park-in | *Zeyad and Youssef* |
| Code Implementation: Display | *Rana and Esraa* |

# 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 ،وفى حالة تكرار الغش سيرسب الطالب فى المقرر.