Cairo University  
Faculty of Computers and Artificial Intelligent

**CS251 - Software Engineering I**

Project Name

Software Requirements Specifications (SRS)

Team Names

Month & Year

Contents

[Instructions [To be removed] 3](#_Toc101814799)

[Team 3](#_Toc101814800)

[Document Purpose and Audience 3](#_Toc101814801)

[Introduction 3](#_Toc101814802)

[Software Purpose 3](#_Toc101814803)

[Software Scope 3](#_Toc101814804)

[Definitions, acronyms, and abbreviations 3](#_Toc101814805)

[Requirements 4](#_Toc101814806)

[Functional Requirements 4](#_Toc101814807)

[Non Functional Requirements 4](#_Toc101814808)

[System Models 4](#_Toc101814809)

[Use Case Model 4](#_Toc101814810)

[Use Case Tables 5](#_Toc101814811)

[Ownership Report 6](#_Toc101814812)

[Policy Regarding Plagiarism: 6](#_Toc101814813)

# 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 SRS description for a parking garage application. It describes the functions done by the system and use case illustration.

# This document is meant for client and system developers to understand -at a high level- the features and functionality of the application.

# This document is also considered a basis to the contract between the developing company and the client.

# Introduction

## Software Purpose

## This software is an application that is used by drivers to find, allot and pay for parking slots.

## Software Scope

## Software stakeholders: driver, administrator.

## The driver can scan parking lot for empty slots and confirm parking in the slot chosen by the system, and view parking details, and pay the parking fees using credit card services.

## An administrator can display available parking slots and calculate income at any given time.

## Definitions, acronyms, and abbreviations

* **Active Slot Configuration:** The methods used by the system to select an empty slot for a car to park in.
* **First come first served configuration:** A configuration method that will use the first free slot available from the parking garage slots.
* **Best-fit approach:** A configuration where you need to find the slot with the minimum dimension to hold the vehicle.

# Requirements

## Functional Requirements

## The software allows the driver to:

## Sign up with a name, a valid email or phone number and vehicle information ( model name - model year - vehicle dimensions).

## Search for garages by area, and select the closest parking garage.

## Check the availability of an empty parking slot in the garage.

## Confirm parking in the parking slot assigned to the driver by the system if an empty slot was found.

## View charges at any given time by the driver by calculating the time elapsed multiplied by the hourly rate.

## Park-out of the garage to view parking details like slot details, time, car info, and total parking fees.

## Checkout using credit card services by entering their credit card information and confirm payment.

## The software allows the Owner to:

## Register and log into the system using their credentials.

## Add a garage to the system, by filling garage details like (garage’s name - garage’s location - slot selection configuration - number of slots - slot’s dimensions ).

## Display available parking slots and their info.

## Calculate total income and number of vehicles parked at any given point of time, specifying the start and end date within witch income or number of vehicles parked needs to be calculated.

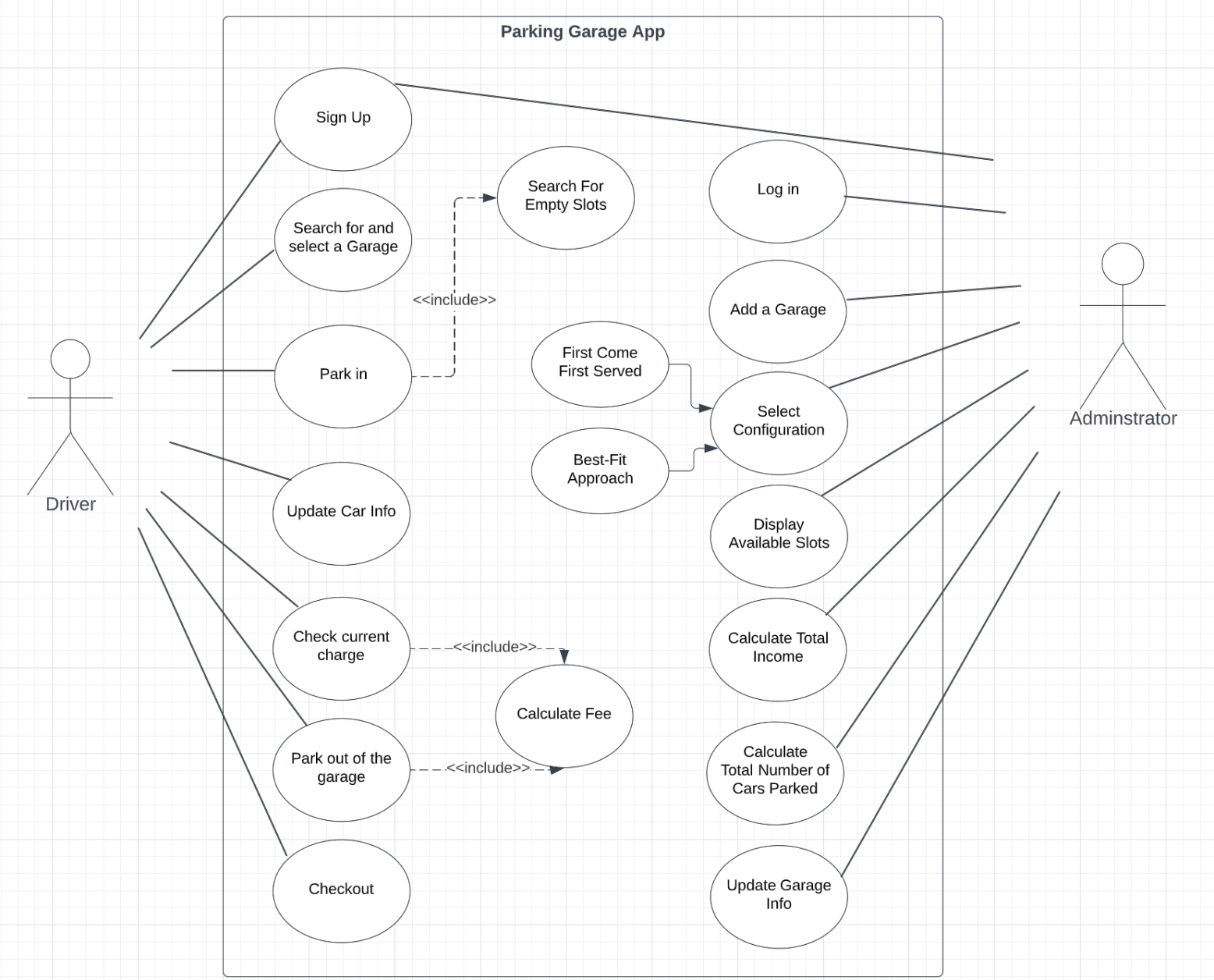
## Non Functional Requirements

|  |  |
| --- | --- |
|  | **Details** |
| **Usability** | * A novice user should be able to use the app with ease within 2 - 5 minutes. |
| **Availability** | * The application is expected to be available 24/7. Down-time shouldn’t exceed 40 minutes a week. |
| **Safety** | * Sensitive information communication “like credit card information”, should be encrypted. |
| **Accessibility** | * Application should provide handicap access, and provide multi-language support. |
| **Performance** | * The Application should load results in 1-3 seconds while simultaneous users are below 100. |
| **Maintainability** | * Software code should be easily maintained by enriching the code with useful comments and eliminating dead codes. |

# 

# System Models

## Use Case Model

****

## 

## Use Case Tables

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC1 | |
| Use Case Name: | Login | |
| Actors: | Garage owner or driver | |
| Pre-conditions: | User has an account in the system and wants to login | |
| Post-conditions: | User enters home page of the system | |
| Flow of events: | **User Action** | **System Action** |
| 1- User enters his email or phone number and password. |  |
|  | 2- System validates from the entered data like:  -email or phone number  -password |
|  | 3- System verifies that the entered data from the user is correct |
| 4- User enters the home page of the system |  |
|  |  |
| Exception 1:  One or more input field is invalid | **User Action** | **System Action** |
| 1- User enters his email or phone number and password. |  |
|  | 3-system clarify which input is invalid like:  a) not matching email or phone number  b) wrong password |
|  |  | 5- System clears fields and asks the user to re-enter the data correctly. |
|  | Use case flow continues from step 1 |  |
| Includes: |  | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC2 | |
| Use Case Name: | Sign up | |
| Actors: | Garage owner | |
| Pre-conditions: | \_ | |
| Post-conditions: | Owner profile is added to the system. | |
| Flow of events: | **User Action** | **System Action** |
| 1- User clicks on garage owner button |  |
|  | 2- System displays log in form and sign-up button |
| 3- User clicks on sign-up button |  |
|  | 4- System ask user to fill registration form. |
| 5- User fills: full name, email, age, password, and repeated password fields and then click the sign up button. |  |
|  |  | 6- System validates from the entered data .  7- System adds a garage owner account to the system and displays the owner home page to the user. |
| Exception1:  One or more input field is invalid. | **User Action** | **System Action** |
| 5- User fills: full name, email, age, password, and repeated password fields and then click the sign up button. |  |
|  | 8- system clarify which data is invalid like:   1. Invalid Full name 2. Invalid email 3. Invalid age 4. Weak password 5. Password in password field and repeated password field don’t match |
|  | 8- The system asks the user to enter the data correctly. |
| Use case flow continue from step 5 |  |
| Includes: |  | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC3 | |
| Use Case Name: | Add Garage | |
| Actors: | Garage owner | |
| Pre-conditions: | Owner has signed up to the system. | |
| Post-conditions: | A new garage is created and added to the system. | |
| Flow of events: | **User Action** | **System Action** |
| 1-User clicks on add garage option. |  |
|  | 2-System displays garage details form. |
| 3- User enters garage details: (Name , City, Area ,Capacity ,Slots and slot selection configuration) . |  |
|  | 4- System assigns special id to the garage then adds the garage to the system and displays “Garage Added!” to the owner. |
| Exceptions: | **User Action** | **System Action** |
|  |  |  |
| Includes: |  | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC4 | |
| Use Case Name: | Search for and select a garage | |
| Actors: | Driver | |
| Pre-conditions: | The driver login to the application and choose his car and input its dimensions. | |
| Post-conditions: | The system displays the garages that best fit his search. | |
| Flow of events: | **User Action** | **System Action** |
| 1- The driver clicks “Search for Garage” button and inputs city and area location. |  |
|  | 2- System searchs for the nearest garages that match the entered location and display these garages and their location to the driver. |
| 3- Driver selects the garage they want to drive up to. |  |
|  | 4- System shows the driver garage details (name, slots, address, empty slots), and the directions to the chosen garage. |
| Includes: |  | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC5 | |
| Use Case Name: | Park-in | |
| Actors: | Driver | |
| Pre-conditions: | Driver has signed up with vehicle. | |
| Post-conditions: | Driver starts a parking session. | |
| Flow of events: | **User Action** | **System Action** |
| 1- Driver clicks Park-in button upon arrival to the selected garage. |  |
|  | 2- System receives Park-in request, and displays a vehicle information form for confirmation. |
| 3- Driver confirms the existing vehicle information. |  |
|  | 4- System searches for and selects a slot based on the active slot configuration chosen by the garage owner, then displays selected slot ID and asks for park-in confirmation.  5-System marks the selected slot as occupied and starts a 5-minute timer for the driver to confirm Park-in. |
| 5-Driver clicks the Confirm button. |  |
|  |  | 6- System creates a new parking session and the starts a timer for the session. |
| Exception1:  Driver updates vehicle information | **User Action** | **System Action** |
|  | 2- System receives Park-in request, and displays a vehicle information form for confirmation. |
| 3-Driver updates the vehicle information fields with their current vehicle information and confirms new information. |  |
|  |  | 4-System updates the info and displays “Vehicle information Updated!” |
|  | Flow continues from step 4. |  |
| Exception2:  5 minutes pass before the Driver confirms parking. | **User Action** | **System Action** |
|  | 6- System frees the selected slot and exits the park in screen back to the driver’s home page. |
| Includes: | 2- Update vehicle info use case is invoked by the system when the system demands confirmation on the vehicle info and the driver selects the update option. | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC6 | |
| Use Case Name: | Park-out | |
| Actors: | Driver | |
| Pre-conditions: | Driver has parked in the slot assigned by the system. | |
| Post-conditions: | Driver gets parking details and calculated fees. | |
| Flow of events: | **User Action** | **System Action** |
| 1- Driver clicks on the park-out button. |  |
|  | 2- System Displays a “Are you sure you want to park-out now?” confirmation message to the driver. |
| 3- Driver confirms park-out. |  |
|  | 4- System stops timer, and displays parking information: parking date, start and end times, car details, parking slot, total fees and a checkout button. |
| 5- User clicks the checkout button. |  |
| Exceptions1:  Driver confirms park-out but doesn’t checkout | **User Action** | **System Action** |
| 3- Driver confirms park-out. |  |
|  | 4- System waits for 2 minutes if the driver doesn’t park-out the system charges the driver 10EGP, and resumes the timer |
| Includes: |  | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC7 | |
| Use Case Name: | Checkout | |
| Actors: | Driver | |
| Pre-conditions: | Driver has parked out of the parking slot. | |
| Post-conditions: | Driver pays parking fees and exits garage. | |
| Flow of events: | **User Action** | **System Action** |
| 1- Driver clicks the checkout button. |  |
|  | 2- System prompts the driver with credit card and cash payment options. |
| 3- Driver selects payment option. |  |
|  | 4- System displays “please pay your fees at the gate” message if cash is chosen and prompts the driver with credit card number and password fields if credit card payment is chosen. |
| 5- Driver enters card number and password, or heads to the gate. |  |
|  |  | 6- System authenticates input and checks If account has sufficient funds, then performs withdrawal and displays “Payment Done!” and a QR code if credit card payment is chosen. |
|  | 8- Driver scans QR code at the gate to verify credit card payment, or pays fees in cash to the owner who lifts the gate for the driver to exit. |  |
|  |  | 9- System marks the slot as empty upon driver’s departure. |
| Exception 1:  Driver enters wrong credit card number or password | **User Action** | **System Action** |
| 7- Driver enters car number and password. |  |
|  | 8- System displays “Wrong credit card information, please try again” message to the user and empties the fields. |
| Exception 2:  Driver checks out and doesn’t park out | **User Action** | **System Action** |
| 1- driver clicks the checkout button. |  |
|  | 8- System waits for 2 minutes if the driver doesn’t park out of the garage the system charges the driver 10EGP, and resumes the timer. |
| Includes: | UC7 - Check Current Charge | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC8 | |
| Use Case Name: | Check current charge | |
| Actors: | Driver | |
| Pre-conditions: | The driver parked his car in the slot. | |
| Post-conditions: | The current charge is displayed to the driver. | |
| Flow of events: | **User Action** | **System Action** |
| 1-Driver opens the application and click button “Check Current Charge”. |  |
|  | 2- The system calculates the time spent by the driver in the parking and multiplies it by the hourly rate. |
|  | 3- The system displays the charge to the driver in EPG. |
| 4- The driver sees the charge. |  |
| Exceptions: | **User Action** | **System Action** |
| 1-Driver opens the application and click button “Check Current Charge”. |  |
|  | 2- The system calculates the time spent by the driver in the parking if the time is less than or equal 15 minutes the system shows that the charge is 0.0 EGP.  3- If the elapsed minutes exceed 15 minutes the system will consider it as one hour. |
| Includes: |  | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC9 | |
| Use Case Name: | Calculate total income / Calculate total number of cars parked | |
| Actors: | Owner | |
| Pre-conditions: | Owner is logged into their account. | |
| Post-conditions: | Income and number of cars parked of the specified period is displayed to the owner. | |
| Flow of events: | **User Action** | **System Action** |
| 2- Owner clicks on “Display total income” button in home page. |  |
|  | 2- System prompts the user with email and password fields to authenticate. |
| 3- Driver enter their email and password. |  |
|  | 4-System will authenticate account then calculate the number of vehicles that have parked-out from the garage and sums up their fares to calculate total income.  5- The system displays the sum as total income as well as the number of cars that have parked out. |
| Includes: |  | |
| Notes and Issues: |  | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC10 | |
| Use Case Name: | Update Garage Information | |
| Actors: | Garage Owner | |
| Pre-conditions: | The owner has already created a garage. | |
| Post-conditions: | Garage information is changed. | |
| Flow of events: | **User Action** | **System Action** |
| 1- Garage owner clicks the update garage info button. |  |
|  | 2- System receives update request and displays a form of the existing garage info for the owner with the editing enabled. |
| 3- The owner updates the fields that need to be changed and submits the form to the system. |  |
|  | 4- System displays a confirmation message and asks owner to enter their password for validation. |
| 5-The owner enters the password and clicks confirm. |  |
|  |  | 6-System updates the garage info and displays “Update Done!” |
| Exceptions: | **User Action** | **System Action** |
|  |  |
|  |  |
| Includes: |  | |
| Notes and Issues: |  | |

# 

# Ownership Report

|  |  |
| --- | --- |
| **Item** | **Owners** |
| Functional and Non-functional Requirements | *Rana and Esraa* |
| Use Case Diagram | *Youssef and Zeyad* |
| UC1, UC2 and UC3 | *Youssef* |
| UC4 and UC8 | *Zeyad* |
| UC5 and UC10 | *Esraa* |
| UC6, UC7, and UC9 | *Rana* |

# 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.