Cairo University
Faculty of Computers and Information
Department of Computer Sciences

Rakenny
(Parking Recommender System)

Supervised by
Dr. Sabah Sayed
TA. Ahmed Samir

Implemented by

20176035	*Mostafa Khattab*
20178031	*Kareem Hany*
20176034	*Nouran Ahmed*

Graduation Project
Academic Year 2020-2021
Midyear Short Documentation

Table of Contents

Table of Figures

Tables

Abstract

Describing the existing problem

- Parking has been a problem for decades in our community and daily life, and in the past couple of years many garages have been built all around the country with minimal knowledge of their existence so we aspire to create an application that connects the Drivers (Client) and Garages through the services that we provide.
- We aspire to create an application that connects the Drivers (Client) and Garages through the services that we provide.

Tools Used

- **Android:** We used Android as our client side operating system platform; which is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets, It is free and open source software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License.
- **Visual Studio Code:** is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.
- **Flutter:** is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. And we are going to use Flutter Maps library to view flutter maps in flutter apps, by adding googler_maps_flutter package.
- **Google places API:** The Places API is a service that returns information about places using HTTP requests. Places are defined within this API as

establishments, geographic locations, or prominent points of interest. We use it to give users the option to search for nearby Garages to park their cars in.

- **Google Directions API:** The Directions API is a service that uses an HTTP requests to return JSON or XML-formatted directions between locations. You can receive directions for several modes of transportation. For direction calculations that respond in real time to user input, you can use the Directions API.

- **Android Emulator:** The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device. The emulator provides almost all of the capabilities of a real Android device.

- **Maps SDK for Android:** It's a service API provided by Google that enables us to add different maps to our application that we later on use to preview the data visually to users, either that data being the route connecting two consecutive stops or a specific place on the map.

- **FireBase:** it is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

- **Amazon Web Services:** offers a broad set of global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security and enterprise applications. These **services** help organizations move faster, lower IT costs, and scale.

- **Python Flask:** Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

- **Python:** it is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

- **Anaconda Jupiter:** The Jupiter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Once saved, you can share these files with others. NOTE: Python and R language are included by default, but with customization, Notebook can run several other kernel environments.

1. Background

1.1 Main Area of Project

- We want to solve the parking problem in our country by developing an application to find the nearby garages for the client destination so he could park his own vehicle in.

1.2 Motivation

- We will provide our clients with recommendations for parking lots (Garages) in an area of his/her specifications (by address, landmark name or neighborhood), to review, choose and book from available spaces nearby with the tap of a finger saving a lot of time and effort for our clients that is usually spent circulating streets to find a parking spot.

1.3 Beneficiary

- Clients will easily find a garage to the nearby destinations to park their vehicles.
- Garage owners profit will increase.

1.4 main techniques

Waterfall Methodology:

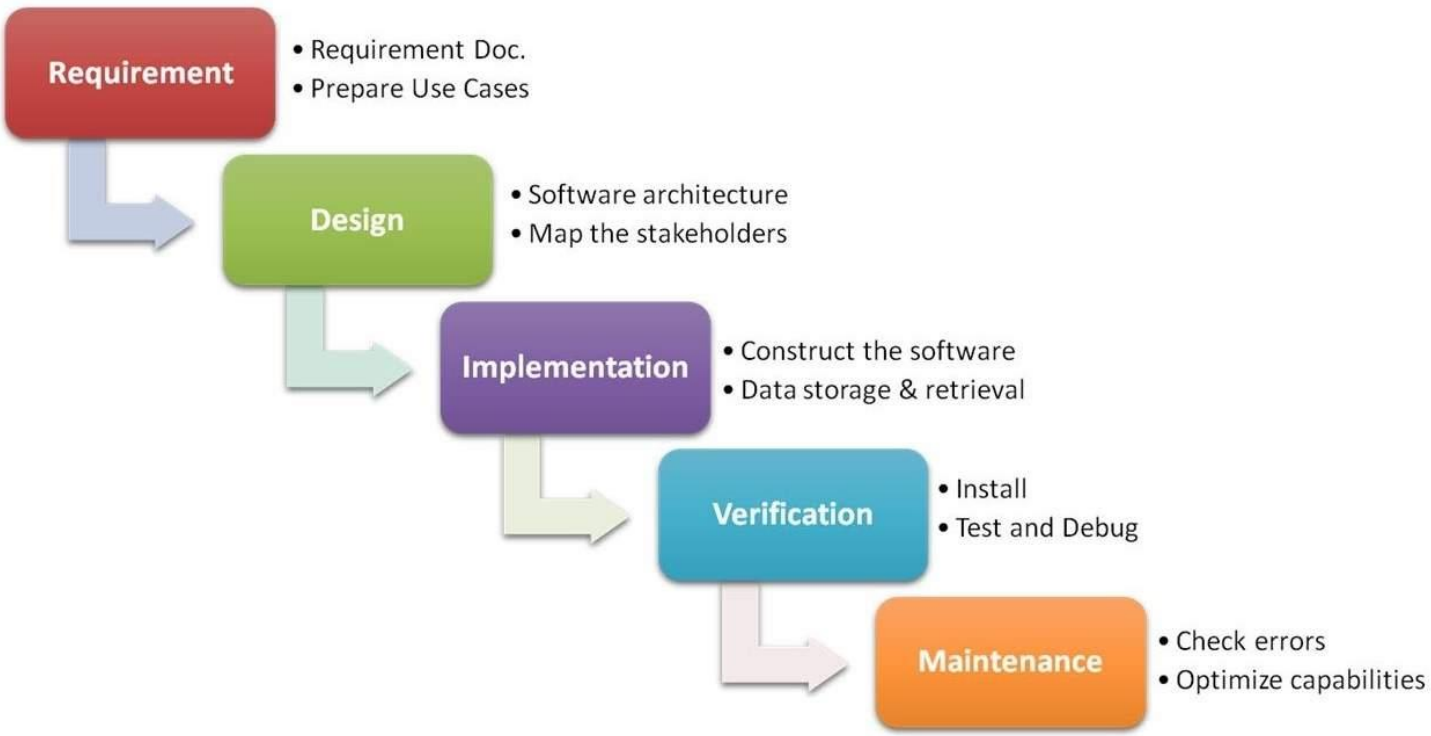

Figure 1

- The Waterfall model is the earliest SDLC approach that was used for software development, the waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete, and the phases don't overlap.
- It consisted of several discrete phases. No phase begun until the prior phase was complete, and each phase's completion is terminal: in the requirement phase we started to search for existing systems and finding features that makes our users satisfied, then after finishing this step and collecting all the required requirements we started to search for the quality attributes and non-functional requirements that should exist in the system as security, Usability and so on.
- Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially: design of system architecture was an input to the coding step after finishing the coding it was input to the test phase.

1.5 Main Application

- Our Application gives recommendations for parking lots (Garage) in a specified area, to review, choose, rate and book from if you would like.

2. Problem Definition

- The main problem that our application is trying to solve is the parking problem we have been facing in our daily life as we go around streets trying to find a suitable parking slot for our vehicle.

3. Related Work

In this part we are going to show an application that is pretty similar to ours and make a comparison between both of them.

Best Parking

- It's an application abroad having similar features as our application.
- The main difference between best parking and our application is that best parking offers to search in parking spot in both garages and lots in the street.

	Best Parking	Rakenny
Searching for Parking	*The client Search for a parking spot where and when he need it. Then Enter his destination by address, landmark name or neighborhood to see available spaces nearby*.	*The client Search a specified Area for Garages and get Recommendations Based on your user experience and garage Data. Review Garage before booking your spot.*
How to book?	*Reserve a spot at select garages and lots with a few quick taps and pay with your smartphone. Browse thousands more parking facilities for pricing and amenity info. Gets turn-by-turn driving directions to your space.*	*Confirm booking in a specific garage, the application will calculate the distance and time from your location and book your spot in a given time in advance.*
Reservations History	*Check history of your previous Parking Passes.*	*You can leave a review after you finish paying. Check your parking history in the app. See where you parked, how long you stayed and what it cost you.*

Table 1

4. Project Specification

4.1 Component Diagram

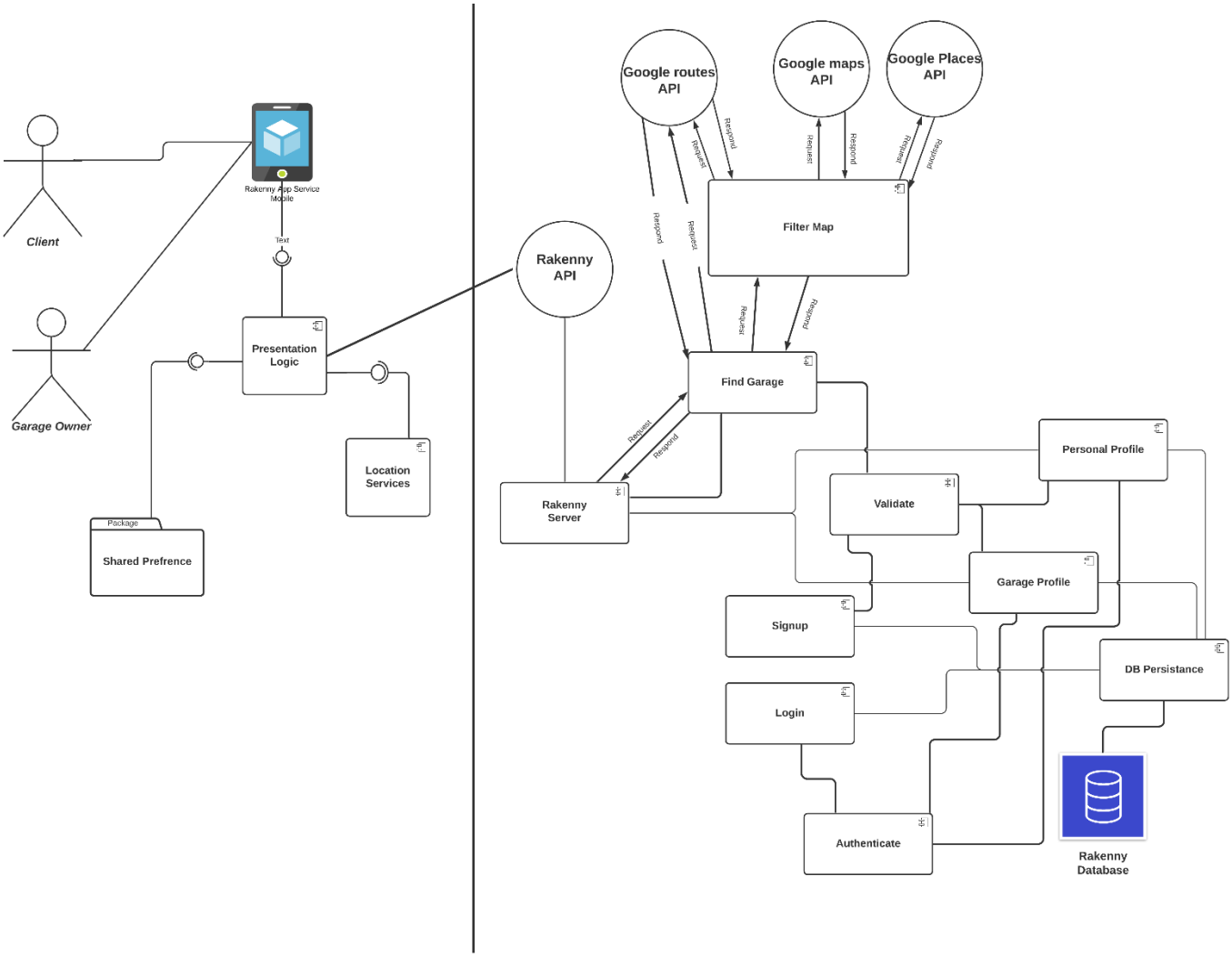

Figure 2

Component Diagram Description:

Users: *consists of*

- ***Clients:*** *User searching for a Garage to reserve a spot in.*
- ***Garage Owners:*** *Users using the application to advertise for their Garage and managing Garage Owner Accounts.*

Presentation: *consists of*

- ***Hardware device (Android Mobile):*** *The application (GUI) that Users will access our services through.*
- ***Presentation Logic (Component):*** *The logic by which Data is represented to the User and passed to the System API.*
- ***Location Services (Component):*** *Component responsible for getting User current location from the Mobile and forwarding it to the System API.*
- ***Shared Preference (File):*** *Personal preference saved on the User's phone to be used for Application personal customization.*

Business: *consists of*

- ***Rakenny Services API:*** *Application Programming Interface for the System's Backend.*
- ***Rakenny Server (Component):*** *The Hard code implementation for the mentioned above API.*
- ***Find Garage (Component):*** *A component responsible for communication with the Filter Map Component to request a List of Garages, or communicate directly with the Google Routes API to request a navigation Map.*
- ***Filter Map (Component):*** *A component that communicates with the three Google APIs to collect Data that are injected into a Neural-Network Algorithm responsible for Filtering and Sorting through a list of candidate Garages to Display in a certain area.*
- ***Google Routes API:*** *Google's Application Programming Interface responsible for Route generating and navigation using a start and end points.*
- ***Google maps API:*** *Google's Application Programming Interface responsible for Map generating using coordinates to display around.*
- ***Google Places API:*** *Google's Application Programming Interface responsible for searching for places on the map using keywords.*
- ***Validate (Component):*** *A component responsible for Validating User input and check it against a list of criteria (e.g. SQLCommands, inappropriate input, form correctness).*
- ***Sign-up (Component):*** *A component responsible for handling using input for Signup, Validating it with the Validate Component and pushing it into the DB Persistence Component to be stored in the Database.*
- ***Authenticate (Component):*** *A component Responsible for Authenticating User password during Login, Change Password and Account Deletion.*

- ***Login (Component):*** *A component Responsible for handling User request for Login ,authenticating it with the Authenticate Component , loading User Data from Database using DB Persistence Component and giving access to the Rakenny Services API*
- ***Personal Profile (Component):*** *A component Responsible for handling all Personal Profile related functionalities (e.g. Edit Profile, Change Password, and Delete Account).*

- ***Garage Profile (Component):*** *A component Responsible for handling all Garage related modification/functionalities (e.g. Edit Garage, Manage Vacancy).*
- ***DB Persistence (Component):*** *A component Responsible for establishing and maintaining a connection with the database, also it contains commands to be executed on the Database (e.g. Select, Insert, Delete etc...).*

- ***Rakenny Database (Database):*** *The Database used in storing all User related information.*

4.2 Stakeholders

- Clients who uses the application.
- Garage owners who owns the garage.
- Developers who implemented the application.
- Doctors who review the application.

4.3 Functional requirements

Client related requirements

- Client must be able to register an account.
- The client must provide his credentials in order to register.
- Client may be able to add his\her profile photo during registering.
- Client must be able to delete his account.
- Client must be able to view/add or remove any number of vehicles in his account
- Client must provide vehicle name in order to register a vehicle.
- Client must provide vehicle type/model in order to register a vehicle.
- Client must provide vehicle plate number in order to register a vehicle.
- Client must be able to select the vehicles he want to unregister from his vehicles table.
- The registered client must able to login.
- Client must be provided by a personal (1) reservation history.
- Client must be able to change his password.
- Client must be able to change his phone number.
- Client may be able to remove or change his\her profile photo.
- The information of the client must be stored in our application database.
- After client's login there will be a profile page displaying his\her credentials.
- Client must be able to log out from his account.

- There should be an AI Algorithm that handles the filtering and sorting through a garage list that will be displayed during booking process.

Booking related requirements

- Client must allow access to location services by our application to locate him\her.
- Client must choose which registered vehicle he\she is using.
- Client may search for the desired destination on the map.
- Client may enter\search the desired destination in the search bar.
- Client must be provided with a list of Garages to choose from.
- Client must be provided with a map shown the garages locations.
- Client must choose a garage to view its information.
- Client can go back to choose another garage to view.
- If the garage charges hourly rate the client must provide the start and end time.
- If the garage charges daily rate the client must provide the start time.
- After finding the suitable garage, client must provide credit card information in order to begin the (2) booking process.
- After reservation, the client must be given a (3) reservation number.
- Client can extend his parking time before the end of the main period.
- Client should be prompted with a notification to rate\review his\her experience.

Garage owner related requirements

- Garage owner must be able to register an account.
- The Garage owner must provide his credentials in order to register.
- The Garage owner must provide the Garage information in order to register a Garage.
- Garage owner may be able to add his\her profile photo during registering.
- Garage owner must be able to delete his account.
- The registered Garage owner must able to login.
- Garage owner must be able to change his password.
- Garage owner must be able to change his phone number.
- Garage owner may be able to remove or change his\her profile photo.
- Garage owner must be able to change Garage Working Hours.
- Garage owner must be able to change Garage Capacity.
- Garage owner must be able to change Garage Contact number.
- Garage owner must be able to change Garage name.
- Garage owner must be able to change Garage fare.
- The information of the Garage owner must be stored in our application database.
- After Garage owner’s login there will be a profile page displaying his\her credentials.

- After Garage owner's login there will be a Garage page displaying Garage credentials.
- After Garage owner's login there will be a Home page displaying Garage statistics.
- Garage owner must be able to log out from his account.

Other requirements

- Client should have (4) help center in the application.
- Client should have (5) About us page in his/her application.

4.4 Non-Functional Requirements

Usability Requirement:

- Mobile app usability makes it easy for the user to become familiar with the user interface (UI).
- Application must warn user if he entered his credentials wrong.
- GUI must have an easily navigated map.
- Users can sometimes take actions within an app that they didn't intend to take. When a user makes a mistake, Mobile apps need to support undo and redo functions.
- The application will help the client in choosing garage in an easy way.

Reliability Requirement:

- Garage information must be up to date.
- Navigation must be updated and reliable.
- Data stored in and retrieved from the database must be the same.
- The system must be up and running 24/7.

Performance Requirement:

- The system should startup in less than 6 sec.
- Login authentication should take less than 10 sec.
- After conformation navigation map should take less than 8 sec.

Scalability Requirement:

- The app should be capable enough to handle almost xxxx users simultaneously without affecting its performance.
- The system should have coverage over Cairo capital Garages.

Robustness Requirement:

- The system must not allow users to sign up with the same credentials twice.
- The reservation function must be synchronized.
- User must not be able to book more than one lot at a time.
- User can't login with wrong credentials.
- User can't login again with the same credentials after deleting his account.

Security Requirement:

- All user's entered information must be encrypted.
- Safety measurements must be taken to secure any leak of any user information.

Accessibility Requirement:

- Adhere to the standard keyboard access methods.
- Provide descriptions and instructions for all accessibility features
- Use the simplest language possible for instructions, prompts and outputs.
- Provide accessible training and support materials.
- Ensure that the user interface and task flow is similar across different functions

Availability Requirement:

- System functionalities must be available at all times.
- All the garages near the desired destination must be all shown on the map

Data integrity:

- Garage information must be up to date and validated.
- User information must be up to date and validated.
- Routes Database must be up to date and validated

Maintainability Requirement:

- The System must be implemented adhering to OOP principles to ensure easy maintenance.

Flexibility Requirement:

- The System must be implemented adhering to OOP principles to ensure easy change.

Visibility Requirement:

- The System must always give Feedback about what operation is being done in wait time.
- The System must give a clear receipt about the fare being deducted from the balance.
- E.g.: if the customer is to book for 4 hours
- (fare1*1)+ (fare2*3) +tax=Total amount to be deducted.

Compatibility Requirement:

- The System must be compatible with Android System.
- The Frontend and the Backend must be compatible to work together for an extended period of time.
- The Backend and the Database must be compatible to work together for an extended period of time.

4.5 Definitions

- Reservation history: A list include all the past reservation the client has made.
- Booking process: the process in which the client goes through to reserve a parking spot for his vehicle.
- Reservation number: a serial number for booking confirmation.
- Help center: a page with instructions (Q&A) on how to use our application.
- About us page: a page containing information about the developers/product owners, their policy & terms and Contact E-mails**.**

4.6 Use Case Diagrams

Sign Up & Login Use Case

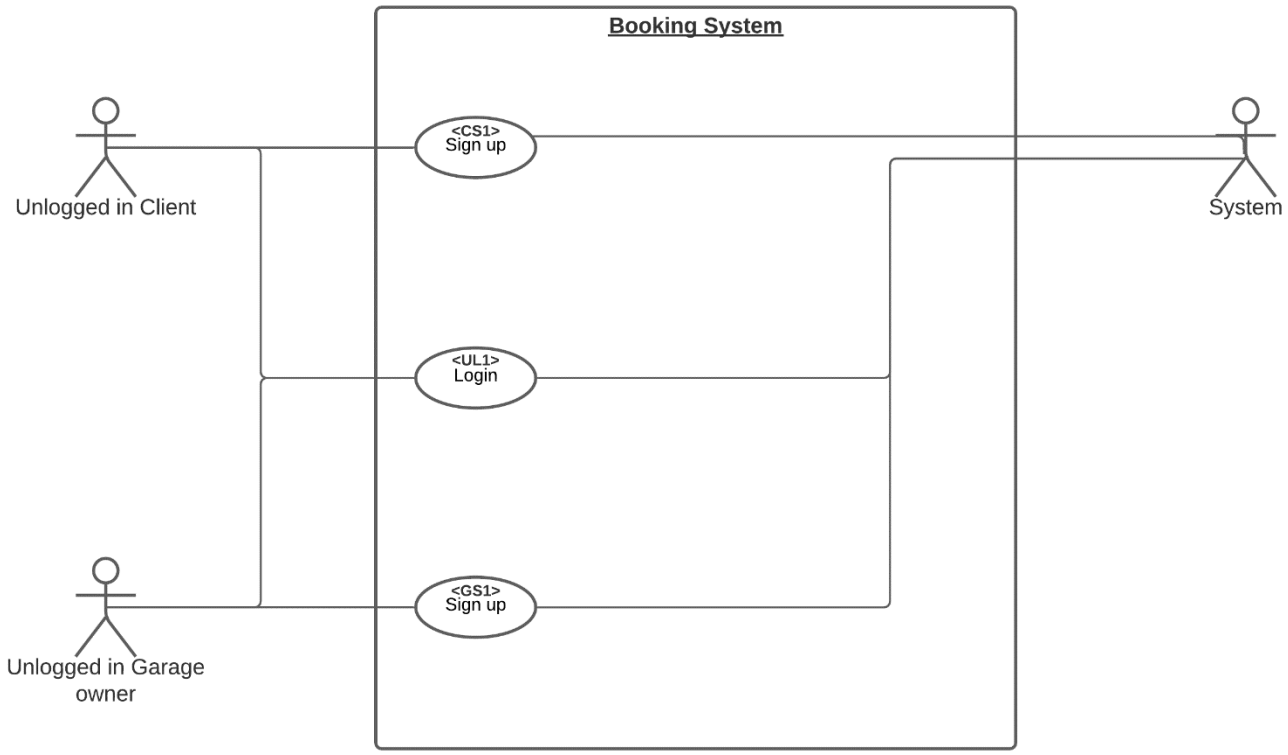

Figure 3

Our Booking system contains three involved actors; Unlogged in client, Unlogged in Garage Owner and System.

1.

Use Case ID:	CS1	
Use Case Name:	Sign up	
Actors:	Unlogged in Client , System	
Pre-conditions:	None	
Post-conditions:	If the use case was successful, the client is now registered into the system. If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1- Client Enter First Name, Last Name, Email address, Phone number and Password.	
		2- System stores data in database
Exceptions:	**User Action**	**System Action**
	1- Client Enter First Name, Last Name, Email address, Phone number and Password.	
		2- Phone number, Email address or password is invalid. 3-System rejects sign up applications
Priority	High	
Notes and Issues:	Email address must be unique. Password must contain only letters and numbers.	

2.

Use Case ID:	GS1	
Use Case Name:	Sign up	
Actors:	Unlogged in Garage owner , System	
Pre-conditions:	None	
Post-conditions:	If the use case was successful, the Garage owner is now registered into the system. If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1- Garage owner Enter First Name, Last Name, Email address, Phone number and Password.	
		2- System stores data in database
	3-Garage owner enters garage	

	name , pricing , Capacity , Location address , Working hours , Contact Number , Number of floors and Photo of garage.	
		4-System stores data in database.
Exceptions:	**User Action**	**System Action**
	1- Garage owner Enter First Name, Last Name, Email address, Phone number and Password.	
		2- Phone number, Email address or password is invalid. 3-System rejects sign up applications
	4-Garage owner enters garage name , pricing , Capacity , Location address , Working hours , Contact Number , Number of floors and Photo of garage.	
		5-Garage contact number, Location address is invalid. 6-System rejects signup applications.
Priority	High	
Notes and Issues:	Email address must be unique. Password must contain only letters and numbers. Garage contact number must be validated. Location address must be in correct formatting inside Egypt.	

3.

Use Case ID:	ULI1	
Use Case Name:	Login	
Actors:	Unlogged in Client , unlogged in garage owner , system	
Pre-conditions:	None	
Post-conditions:	If the use case was successful, the client and Garage owner is now logged into the system. If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1- User Enter Email address and Password.	
		2- System Verify user data
Exceptions:	**User Action**	**System Action**
	1- User Enter Email address and Password.	
		2- Email is invalid and unreadable. 3- System rejects login application
Priority:	High	
Notes and Issues:	Email must be registered in database. Password must be the same as the registered password.	

Booking Process Use Case

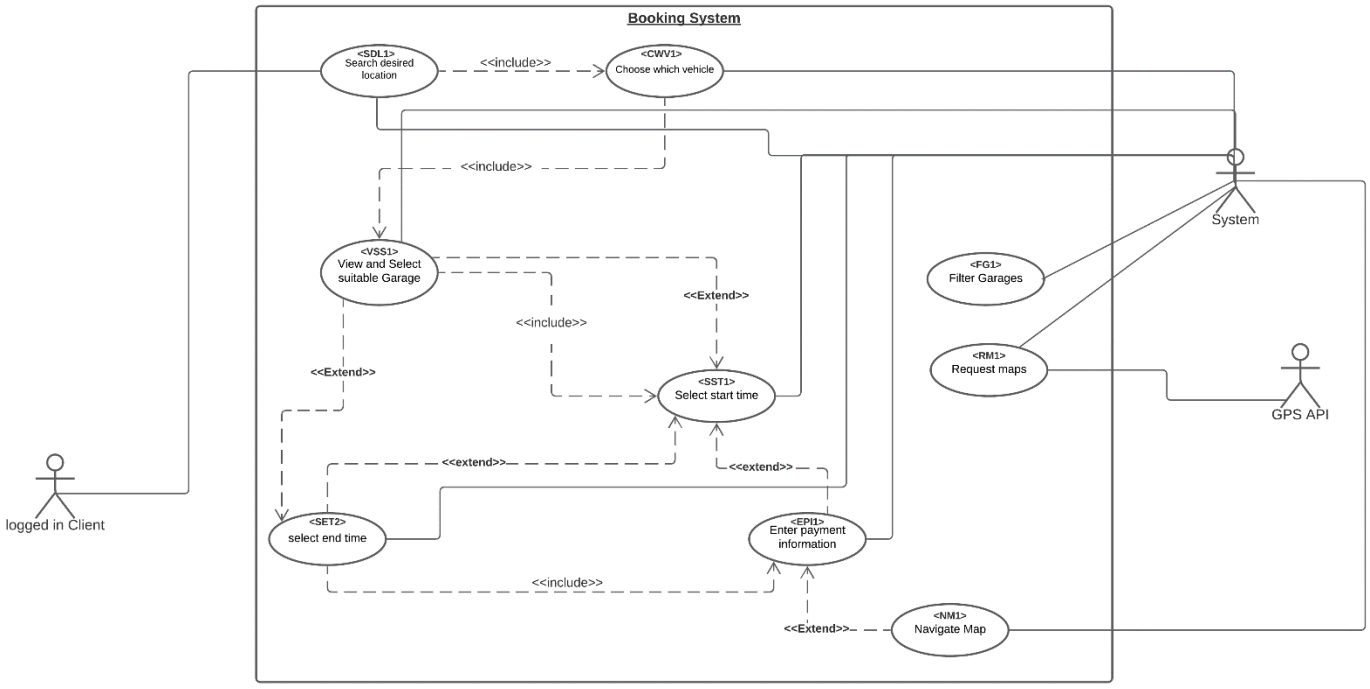

Figure 4

Our Booking system contains three involved actors; logged in Client, GPS API and System.

1.

Use Case ID:	FG1
Use Case Name:	Filter Garage
Actors:	System
Pre-conditions:	None
Post-conditions:	If the use case was successful, the use case will return a filtered and sorted list of garages. If not, the system state remains unchanged.
Flow of events:	**System Action**
	1-the use case takes input from the system and prepares it for feed forwarding. 2-then it injects the input into the system's filtering/sorting algorithm for computation. 3-the use case then return a list of filtered and sorted garages to be represented to the client
Priority	High

Use Case ID:	RM1	
Use Case Name:	Request Maps	
Actors:	System , GPS API	
Pre-conditions:	SDL1	
Post-conditions:	If the use case was successful the System will display client current location on the map If not, the system state remains unchanged.	
Flow of events:	**System Action**	**GPS API Action**
	1-System Requests Map using Coordinates	
		2-GPS API validates coordinates and returns a map.
Exceptions:	**User Action**	**System Action**
	1-System Requests Map using Coordinates	
		2-if coordinates are not in correct format 3- System rejects Request application
Priority:	High	
Notes and Issues:	Coordinates must be in right format or known keyword.	

5.

Use Case ID:	SDL1	
Use Case Name:	Search Desired Location	
Actors:	logged in Client ,system	
Pre-conditions:	Allowing access to location services	
Post-conditions:	If the use case was successful the User will be prompted with a window to choose a vehicle If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1-Client allow access to location services	
		2- System uses (RM1) and displays a map of Client current location
	3- Enter coordinates of desired location	
		4- System Validate user Entries
Exceptions:	**User Action**	**System Action**
	1- Client allow access to location services and Enter coordinates of desired location	
		2- location services access not allowed or coordinates 3- System rejects Search application
Priority:	High	
Notes and Issues:	Coordinates must be in right format or known keyword.	

6.

Use Case ID:	CWV1	
Use Case Name:	Choose which vehicle	
Actors:	logged in Client ,system	
Pre-conditions:	SDL1	
Post-conditions:	If the use case was successful the User will be prompted to garages nearby his/her desired location If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System pulls Client vehicle data from Database and displays Client vehicles to choose from.

	2- Client Chooses which vehicle he/she will be using	
		3-System takes client input.
Exceptions:	**User Action**	**System Action**
		1-System displays Client vehicles to choose from.
	2- Client Chooses which vehicle he/she will be using	
		3-if client chooses to exit the booking process 4- System goes back to Home Page
Priority:	High	
Notes and Issues:	Client has to choose the vehicle he/she will be using.	

7.

Use Case ID:	VSS1	
Use Case Name:	View and select suitable garage	
Actors:	logged in Client ,system	
Pre-conditions:	CWV1	
Post-conditions:	If the use case was successful the User will be prompted with a window to select start time If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System uses (RM1) and displays a map of garages surrounding the destination.
	2- Client Chooses which garage to review.	
		3-System pulls garage information form database and display it to user.
	4-Client confirms garage.	
Exceptions:	**User Action**	**System Action**
		1-System uses (RM1) and displays a map of garages surrounding the destination
	2- Client Chooses which garage to review.	
		3-System pulls garage information form database and display it to user.
	4- Client chooses to go back and	

	view another Garage	
		5-System repeats from step 1
Priority:	High	
Notes and Issues:	Client has to choose a Garage to book a slot.	

8.

Use Case ID:	SST1	
Use Case Name:	Select Start Time	
Actors:	logged in Client ,system	
Pre-conditions:	VSS1	
Post-conditions:	If the use case was successful and garage charges daily Client will be prompted with a windows to enter Payment information, else if Garage charges Hourly client will be prompted with a window to select start time If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts user with a windows to Select Start Time
	2- Client Chooses Start Time	
		3-if Garage charges daily process will go to (EPI1), else if Garage charges hourly process will continue to (SET2)
Exceptions:	**User Action**	**System Action**
		1-System prompts user with a windows to Select Start Time
	2- Client Chooses to go back and select another Garage	
		3-System goes back to (VSS1)
Priority:	High	
Notes and Issues:	Client has to select start time	

9

Use Case ID:	SET2
Use Case Name:	Select End Time
Actors:	logged in Client ,system
Pre-conditions:	SST1
Post-conditions:	If the use case was successful Client will be prompted with a receipt

	page If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts user with a windows to Select End Time
	2- Client Chooses End Time	
		3-System will continue to (SET2)
Exceptions:	**User Action**	**System Action**
		1-System prompts user with a windows to Select End Time
	2- Client Chooses to go back and select another Garage	
		3-System goes back to (VSS1)
Priority:	High	
Notes and Issues:	Client has to select End time	

10.

Use Case ID:	EPI1	
Use Case Name:	Enter Payment Information	
Actors:	logged in Client ,system	
Pre-conditions:	SET2||SST1	
Post-conditions:	If the use case was successful Client will be prompted with a Confirmation Page, System state changes to (NM1) If not, the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts user with a Receipt Page
	2- Client clicks confirm	
		3-System open a windows for user to Enter Credit Card information
	4-Client Enters Credit Card number , expiration date , CCV and Card Holder Name	
		5-System validates Credit Card information 6-System prompts user with a confirmation message
	7-Client clicks continue	

Exceptions:	User Action	System Action
		1-System prompts user with a Receipt Page
	2- Client Chooses to go back and select another Garage	
		3-System goes back to (VSS1)
		1-System prompts user with a Receipt Page
	2- Client clicks confirm	
		3-System open a windows for user to Enter Credit Card information
	4-Client Enters Credit Card number , expiration date , CCV and Card Holder Name	
		5-Credit card information is not valid 6-System goes back to step 1.
Priority:	High	
Notes and Issues:	Client has to Enter valid Card information	

11.

Use Case ID:	NM1	
Use Case Name:	Navigate Map	
Actors:	logged in Client ,system	
Pre-conditions:	EPI1 || CR1	
Post-conditions:	If the use case was successful Client will be prompted with a Review message later in the day If not, the system state remains unchanged.	
Flow of events:	User Action	System Action
		1-System opens a map with navigation instructions for Client to follow
	2-Client follows navigation to destination	
Priority:	High	
Notes and Issues:	If Client choose to exit navigation	

Logged in Client Use Case

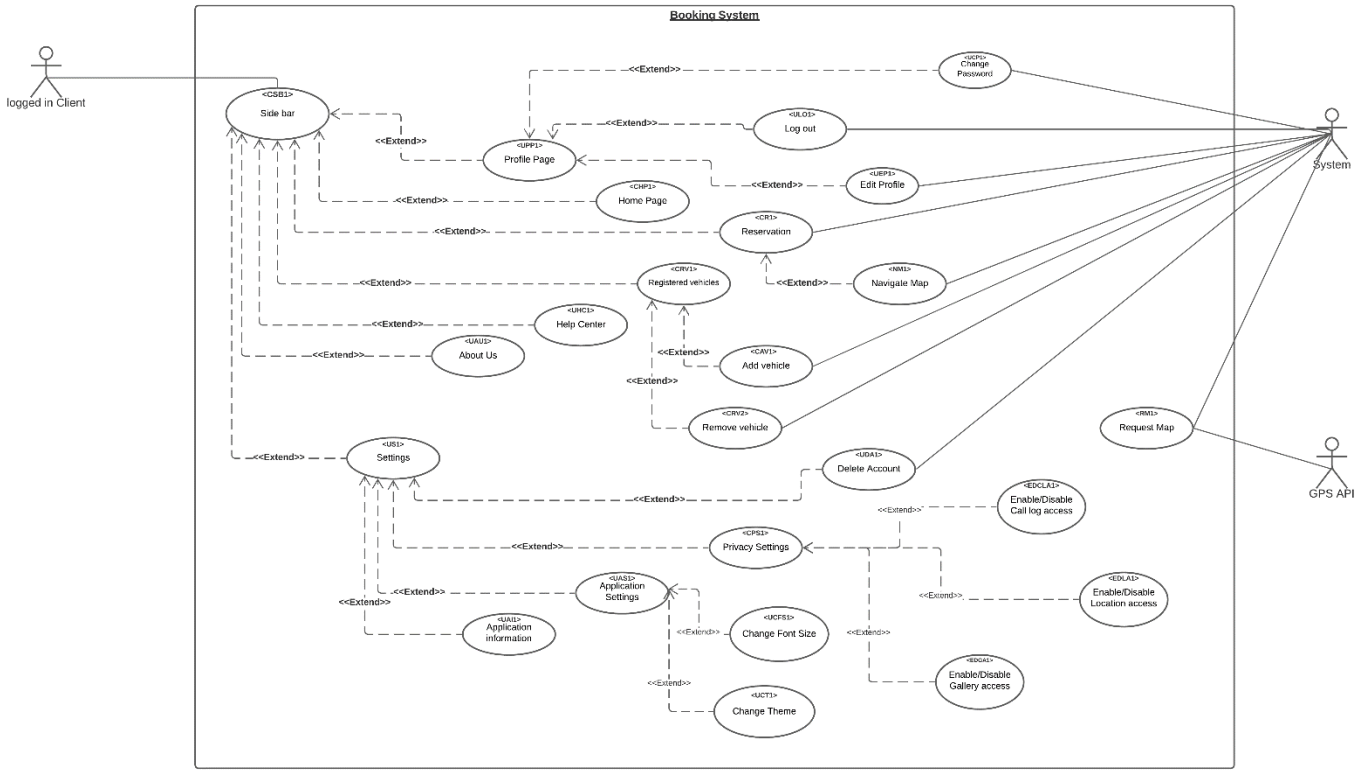

Figure 5

Our Booking system contains three involved actors; logged in client, GPS API and System.

Use Case ID:	CHP1	
Use Case Name:	Home Page	
Actors:	logged in Client ,system	
Pre-conditions:	ULI1	
Flow of events:	**User Action**	**System Action**
		1-System request a map (RM1) using client's current location and opens a page with a map of Client current location and an input field to search for desired location (SDL1)
	2- Client Enters desired location to start booking process	
		3-System triggers (SDL1)
Exceptions:	**User Action**	**System Action**

		1-System request a map (RM1) using client's current location and opens a page with a map of Client current location and an input field to search for desired location (SDL1)
	2- Client chooses to go to Sidebar	
		3-System redirects Garage Owner to (US1)
Priority:	Medium	

35.

Use Case ID:	CSB1	
Use Case Name:	Side Bar	
Actors:	logged in Client ,system	
Pre-conditions:	CHP1	
Post-conditions:	If the use case was successful Client will be redirected to his/her choice, if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts Client with a sidebar containing Profile Page(UPP1), Home Page(CHP1), Reservation, Registered vehicles, Help Center(UHC1), About us(UAU1) and Settings(US1) to choose from
	2- Client Chooses an option	
		3-System redirects Client to desired choice (use case).
Exceptions:	**User Action**	**System Action**
		1-System prompts Client with a sidebar containing Profile Page(UPP1), Home Page(CHP1), Reservation, Registered vehicles, Help Center(UHC1), About us(UAU1) and Settings(US1) to choose from
	2- Client Chooses to close sidebar	
		3-System goes back to (CHP1)
Priority:	High	

36.

Use Case ID:	CR1	
Use Case Name:	Reservation	
Actors:	logged in Client ,system	
Pre-conditions:	CSB1	
Post-conditions:	If the use case was successful Client will be redirected to his/her choice, if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts Client with a Page containing all of his/her past reservation and if a reservation is still in progress he/she will be prompted with a navigate button
	2- Client reviews reservation and chooses to navigate future reservation	
		3-System request a route (RM1) from GPS API using client's current location and desired location and prompts user with a navigation map (NM1)
Exceptions:	**User Action**	**System Action**
		1-System prompts Client with a Page containing all of his/her past reservation and if a reservation is still in progress he/she will be prompted with a navigate button
	2- Client Chooses to go back	
		3-System goes back to (CSB1)
Priority:	High	
Notes and Issues:	User has to have already given access to the location settings.	

37.

Use Case ID:	CRV1	
Use Case Name:	Reserved vehicles	
Actors:	logged in Client ,system	
Pre-conditions:	CSB1	
Post-conditions:	If the use case was successful Client will be redirected to his/her registered vehicle's table, if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts Client with a Page containing all of his/her reserved vehicles with a button to each vehicle used to remove vehicle (CRV2), and a button to add vehicle (CAV1)
	2- Client reviews reserved vehicles and chooses action to be done	
		3-System triggers desired use case (CRV2 || CAV1)
Exceptions:	**User Action**	**System Action**
		1-System prompts Client with a Page containing all of his/her reserved vehicles with a button to each vehicle used to remove vehicle (CRV2), and a button to add vehicle (CAV1)
	2- Client Chooses to go back	
		3-System goes back to (CSB1)
Priority:	Medium	

38.

Use Case ID:	CAV1	
Use Case Name:	Add vehicle	
Actors:	logged in Client ,system	
Pre-conditions:	CRV1	
Post-conditions:	If the use case was successful Client will be redirected to his/her registered vehicle's table, if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts Client with a window containing five input fields to enter (license plate number, vehicle type, vehicle nickname and manufacturer nickname)
	2- Client enters all fields.	
		3-System validates input and stores it in database 4-System redirects Client to edited reserved vehicles page (CRV1)
Exceptions:	**User Action**	**System Action**
		1-System prompts Client with a window containing five input fields to enter (license plate number, vehicle type, vehicle nickname and manufacturer nickname)
	2- Client enters all fields.	
		3-input is invalid 4-System notify Client and asks Client to re-enter his/her car credentials
Priority:	Medium	
Notes and Issues:	Client has to enter all input fields in right format	

39.

Use Case ID:	CRV2	
Use Case Name:	Remove vehicle	
Actors:	logged in Client ,system	
Pre-conditions:	CRV1	
Post-conditions:	If the use case was successful Client will be redirected to his/her registered vehicle's table, if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts user with a window asking him/her if they are sure of taking the next step
	2-Client confirms.	
		3-System deletes desired vehicle from the database 4-System refreshes page to the edited Reserved vehicles table (CRV1)
Exceptions:	**User Action**	**System Action**
		1-System prompts user with a window asking him/her if they are sure of taking the next step
	2-Client reject.	
		3-System redirects user to his/her vehicles table(CRV1)
Priority:	Medium	
Notes and Issues:	Client has to confirm action in-order to continue with the action	

40.

Use Case ID:	CPS1	
Use Case Name:	Privacy settings	
Actors:	logged in Client ,system	
Pre-conditions:	US1	
Post-conditions:	If the use case was successful access state will be switched , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a window with three switches that trigger the Enable/Disable gallery access(EDGA1) , Enable/Disable

		Call log access(EDGA1) and Enable/Disable location access(EDGA1)
	2- Client triggers the desired switch to change access state (enable/disable)	
		3-System triggers ((EDGA1) || ||) and waits for confirmation 4-System changes access state if use case returned true
Exceptions:	**User Action**	**System Action**
		1-System opens a window with three switches that trigger the Enable/Disable gallery access(EDGA1) , Enable/Disable Call log access(EDGA1) and Enable/Disable location access(EDGA1)
	2- Client Chooses to go back	
		3-System redirects Client to (US1)
Priority:	Medium	

41.

Use Case ID:	EDLA1	
Use Case Name:	Enable/Disable location access	
Actors:	logged in Client ,system	
Pre-conditions:	CPS1	
Post-conditions:	If the use case was successful access state is switched , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1-User triggers use case using his/her privacy settings switch	
		2-System checks access state and switches it
Priority:	Medium	
Notes and Issues:	User has to have already given access to the desired settings.	

42.

Use Case ID:	EDCLA1	
Use Case Name:	Enable/Disable Call Log access	
Actors:	logged in Client ,system	
Pre-conditions:	CPS1	
Post-conditions:	If the use case was successful access state is switched , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1-User triggers use case using his/her privacy settings switch	
		2-System checks access state and switches it
Priority:	Medium	
Notes and Issues:	User has to have already given access to the desired settings.	

Logged in Garage Owner Use Case

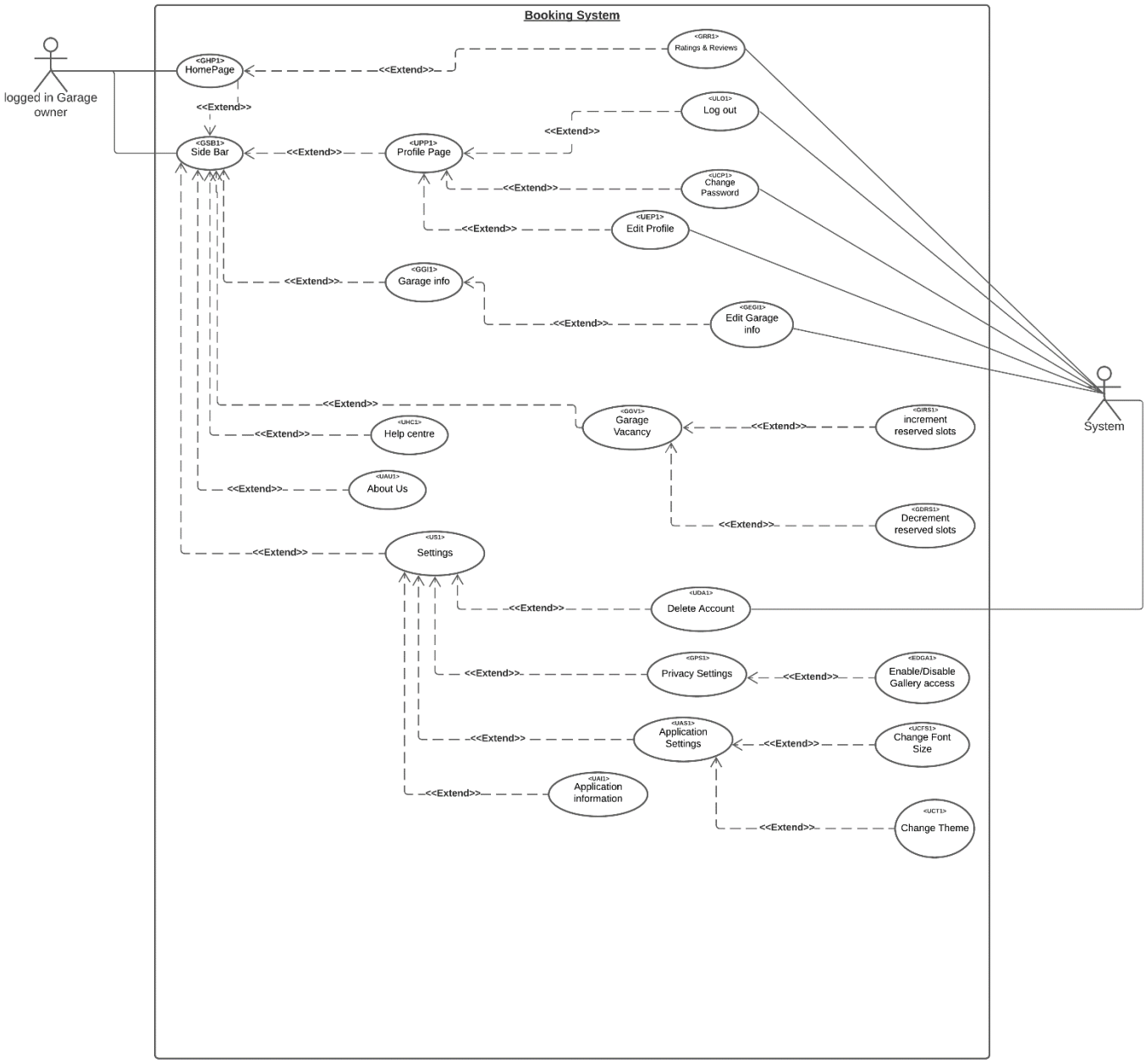

Figure 6

Our Booking system contains three involved actors; logged in Garage Owner and System.

Use Case ID:	GHP1	
Use Case Name:	Home Page	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	None	
Post-conditions:	If the use case was successful Garage Owner will be prompted with the Home Page , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a Home Page for Garage Owner to review statistical data and ratings for the garage to choose from
	2- Client Chooses rating to review	
		3-System pulls review data from database 4-System displays Rating & review for Garage Owner
Exceptions:	**User Action**	**System Action**
		1-System opens a Home Page for Garage Owner to review statistical data and ratings for the garage to choose from
	2- Client Chooses to open Sidebar	
		3-System goes to (GSB1)
Priority:	High	
Notes and Issues:		

13.

Use Case ID:	GRR1	
Use Case Name:	Rating and Reviews	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GHP1	
Post-conditions:	If the use case was successful Garage Owner will be prompted with the Rating&Review that he/she chose to review. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**

		1-System opens a window containing Client name with her/his review and rating.
	2-Garage Owner reads the reviews and clicks back	
		3-System goes back to (GHP1)
Priority:	High	
Notes and Issues:		

14.

Use Case ID:	GSB1	
Use Case Name:	Side Bar	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GHP1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to his/her choice , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System prompts Garage owner with a sidebar containing Profile Page(GPP1), Home Page(GHP1), Garage info(GGI1), Garage Vacancy(GGV1), Help Center(UHC1), About us(UAU1) and Settings(US1) to choose from
	2- Garage owner Chooses an option	
		3-System redirects Garage Owner to desired choice (use case).
Exceptions:	**User Action**	**System Action**
		1-System prompts Garage owner with a sidebar containing Profile Page(GPP1), Home Page(GHP1), Garage info(GGI1), Garage Vacancy(GGV1), Help Center(UHC1), About us(UAU1) and Settings(US1) to choose from

	2- Garage owner Chooses to close sidebar	
		3-System goes back to (GHP1)
Priority:	High	
Notes and Issues:		

Use Case ID:	ULO1	
Use Case Name:	Log Out	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GPP1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to the Login page. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System Log user out 2-System redirects user to login page(ULI1)
Priority:	Medium	
Notes and Issues:	None	

19.

Use Case ID:	GGI1	
Use Case Name:	Garage info	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GSB1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to his\her Garage info page || edit Garage info page , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System will open a window for Garage owner view his/her Garage information(photos, Garage name , capacity,

		availability , pricing , address, working hours, contact numbers and number of floors) 2-and an href to edit his/her garage information
	3-Garage owner click on edit Garage info	
		5-System redirects Garage owner to Edit Garage info page(GEGI1)
Exceptions:	**User Action**	**System Action**
		1-System will open a window for Garage owner view his/her Garage information(photos, Garage name , capacity, availability , pricing , address, working hours, contact numbers and number of floors) 2-and an href to edit his/her garage information
	3- Garage views Garage info and chooses to go back	
		5-System redirects user to Home Page (GHP1)
Priority:	High	
Notes and Issues:	None	

20.

Use Case ID:	GEGI1	
Use Case Name:	Edit Garage info	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GGI1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to his\her edited Garage info Page. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System will open a window for Garage owner to edit his Garage information which consists of : 2-Input boxes containing current data to be edited (Garage name , contact number , Working Hours, Pricing , Capacity and Photos)

	3- Garage owner edit desired data boxes. 4-Garage owner clicks save.	
		5-System checks if input is valid. 6-System stores data input in database. 7-System goes back to edited Garage info Page (GGI1).
Exceptions:	**User Action**	**System Action**
		1-System will open a window for Garage owner to edit his Garage information which consists of : 2-Input boxes containing current data to be edited (Garage name , contact number , Working Hours, Pricing , Capacity and Photos)
	3- Garage owner edit desired data boxes. 4-Garage owner clicks save.	
		5-Data input is invalid. 6-System goes back to (GHP1).
Priority:	High	
Notes and Issues:	Garage owner has to enter valid information while editing.	

21.

Use Case ID:	GGV1	
Use Case Name:	Garage Vacancy	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GSB1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to his\her Garage Vacancy Page. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a window containing (garage name, photos, Garage vacancy(reserved spots/capacity) and two buttons which redirect to () or ()
	2- Garage owner clicks desired button	
		5-System redirects Garage

		owner to desired option
Exceptions:	**User Action**	**System Action**
		1-System opens a window containing (garage name, photos, Garage vacancy(reserved spots/capacity) and two buttons which redirect to () or ()
	3- Garage owner chooses to go back	
		4-System redirects user to (GHP1)
Priority:	High	
Notes and Issues:	None	

22.

Use Case ID:	GIRS1	
Use Case Name:	Increment Reserved Spots	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GGV1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to edited garage vacancy page , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System increments reserved spots entry in database with 1.
Priority:	High	
Notes and Issues:	None	

23.

Use Case ID:	GDRS1	
Use Case Name:	decrement Reserved Spots	
Actors:	logged in Garage Owner ,system	
Pre-conditions:	GGV1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to edited garage vacancy page , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**

		1-System decrements reserved spots entry in database with 1.
Priority:	High	
Notes and Issues:	None	

Use Case ID:	GPS1	
Use Case Name:	Privacy settings	
Actors:	(logged in Garage Owner) ,system	
Pre-conditions:	US1	
Post-conditions:	If the use case was successful access state will be switched , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a window with a switch that triggers the Enable/Disable gallery access(EDGA1)
	2- user triggers the switch to change access state (enable/disable)	
		3-System triggers (EDGA1) and waits for confirmation 4-System changes access state if (EDGA1) returned true
Exceptions:	**User Action**	**System Action**
		1-System opens a window with a switch that triggers the Enable/Disable gallery access(EDGA1)
	2- user Chooses to go back	
		3-System redirects Garage Owner to (US1)

Use Case ID:	UPP1
Use Case Name:	Profile Page
Actors:	(logged in Garage Owner || logged in Client),system
Pre-conditions:	GSB1 || CSB1
Post-conditions:	If the use case was successful User will be redirected to his/her Profile Page , if not the system state remains unchanged.

Flow of events:	**User Action**	**System Action**
		1-System pulls Garage owner information from Database and displays them in profile Page which contains of photo, Full name , Phone number, Email 2-and the system provides three options , Edit Profile (UEP1), Change password (UCP1) and logout(ULO1)
	3- User Chooses an option	
		4-System redirects User to desired choice (use case).
Exceptions:	**User Action**	**System Action**
		1-System pulls User information from Database and displays them in profile Page which contains of photo, Full name , Phone number, Email 2-and the system provides three options , Edit Profile (GEP1), Change password (GCP1) and logout(ULO1)
	2- User Chooses to go back	
		3-System goes back to (GHP1)
Priority:	High	
Notes and Issues:		

16.

Use Case ID:	UEP1	
Use Case Name:	Edit Profile	
Actors:	(logged in Garage Owner || logged in Client),system	
Pre-conditions:	UPP1	
Post-conditions:	If the use case was successful user will be redirected to his\her edited profile page. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System will open a window for user to edit his profile which consists of : 2-Input boxes containing current

		data to be edited (First name , last name , phone number, photo, Email address)
	3- User edit desired data boxes. 4- User clicks save.	
		5-System checks if input is valid. 6-System stores data input in database. 7-System goes back to edited profile page (UPP1).
Exceptions:	**User Action**	**System Action**
		1-System will open a window for user to edit his profile which consists of : 2-Input boxes containing current data to be edited (First name , last name , phone number, photo, Email address)
	3- User edit desired data boxes. 4- User clicks save.	
		5-Data input is invalid. 6-System goes back to (GHP1) or (CHP1).
Priority:	Medium	
Notes and Issues:	User has to enter valid information while editing.	

17.

Use Case ID:	UCP1	
Use Case Name:	Change Password	
Actors:	(logged in Garage Owner || logged in Client),system	
Pre-conditions:	UPP1	
Post-conditions:	If the use case was successful user will be redirected to his\her profile page and the password is successfully changed. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System will open a window for user to Enter his current password.
	2- User Enters current password. 3-user clicks Next	

		4-System verifies the entered password. 5-System open a new window to enter his\her new password twice for confirmation.
	6-user enters the new password twice. 7-user Clicks Save.	
		8-System checks if both entries are the same 9-System updates garage owner password in database.
Exceptions:	**User Action**	**System Action**
		1-System will open a window for user to Enter his current password.
	2- User Enters current password. 3- user clicks Next	
		5-Entered password is wrong. 6-System notify user that the entered password is wrong.
	7- User Re-enters a different password. 8- user clicks save	
		9--System verifies the entered password. 10-System open a new window to enter his\her new password twice for confirmation.
	11- User enters a new password twice.	
		12-Both entries are not the same 13-System notify user that both passwords are not the same.
Priority:	High	
Notes and Issues:	User has to enter a valid password and both new passwords must be the same.	

24.

Use Case ID:	UHC1	
Use Case Name:	Help Center	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	GSB1 || CSB1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to the application Help Center. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a window containing a series of Q&A to help the user use our application
	2- user chooses question to view answer for	
		3-System drop down a window with the answer under the question
Exceptions:	**User Action**	**System Action**
		1-System opens a window containing a series of Q&A to help the user use our application
	2- user chooses to go back	
		4-System redirects user to (GHP1) || ()
Priority:	Low	
Notes and Issues:	None	

Use Case ID:	UAU1	
Use Case Name:	About Us	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	GSB1 || CSB1	
Post-conditions:	If the use case was successful Garage Owner will be redirected to the application About Us Page. , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a window containing brief paragraphs (our scope- our purpose – Contact E-mail)

	2- user views information and chooses to go back	
		3-System goes back go (GHP1)||()
Priority:	Low	
Notes and Issues:	None	

Use Case ID:	US1	
Use Case Name:	Settings	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	GSB1 || CSB1	
Post-conditions:	If the use case was successful User will be redirected to desired Settings page , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a window containing four path options which are Delete Account() , privacy settings(), application setting() and application info()
	2- user chooses path to take	
		3-System redirects user to desired path
Exceptions:	**User Action**	**System Action**
		1-System opens a window containing four path options which are Delete Account() , privacy settings, application setting and application info
	2- user chooses to go back	
		4-System redirects user to (GHP1) || ()
Priority:	High	
Notes and Issues:	None	

Use Case ID:	UDA1	
Use Case Name:	Delete account	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	US1	
Post-conditions:	If the use case was successful User will redirected to Login Page , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System opens a window with two input boxes to confirm his/her password.
	2- user enters password twice	
		3-System authenticate password 4-System deletes user information from database 5-System redirects user to Login page
Exceptions:	**User Action**	**System Action**
		1-System opens a window with two input boxes to confirm his/her password.
	2- user enters password twice	
		3-User entry is incorrect 4-System informs user that his/her entries are wrong and asks them to re-enter them.
Priority:	Medium	
Notes and Issues:	User has to enter his current password twice correctly in order to authenticate	

Use Case ID:	EDGA1	
Use Case Name:	Enable/Disable Gallery access	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	(GPS1) || (CPS1)	
Post-conditions:	If the use case was successful access state is switched , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1-User triggers use case using his/her privacy settings switch	

		2-System checks access state and switches it
Priority:	Medium	
Notes and Issues:	User has to have already given access to the desired settings.	

30.

Use Case ID:	UAS1	
Use Case Name:	Application settings	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	US1	
Flow of events:	**User Action**	**System Action**
		1-System opens a window with a switch that triggers Dark/Light Mode (UCT1) , and three choices of font size (small ,medium ,large) (UCFS1)
	2- user triggers the use case he would like to modify	
		3-System triggers (UCT1) or (UCFS1) and waits for confirmation 4-System modifies application if the use case returns true
Exceptions:	**User Action**	**System Action**
		1-System opens a window with a switch that triggers Dark/Light Mode (UCT1) , and three choices of font size (small ,medium ,large) (UCFS1)
	2- user Chooses to go back	
		3-System redirects Garage Owner to (US1)
Priority:	Medium	

Use Case ID:	UCFS1
Use Case Name:	Change Font Size
Actors:	(logged in Garage Owner || logged in Client) ,system
Pre-conditions:	UAS1
Post-conditions:	If the use case was successful Font size will be changed , if not the system state remains unchanged.

Flow of events:	**User Action**	**System Action**
	1-User triggers use case using his/her application settings choice	
		3-System changes font size to user chosen option
Priority:	Low	

32.

Use Case ID:	UCT1	
Use Case Name:	Change Theme	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	UAS1	
Post-conditions:	If the use case was successful the application theme will be switched (Dark, Light) , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
	1-User triggers use case using his/her application settings switch	
		2-System checks Theme coloring and switches it
Priority:	Low	

33.

Use Case ID:	UAI1	
Use Case Name:	Application information	
Actors:	(logged in Garage Owner || logged in Client) ,system	
Pre-conditions:	US1	
Post-conditions:	If the use case was successful the System will prompt user with his/her application information , if not the system state remains unchanged.	
Flow of events:	**User Action**	**System Action**
		1-System open a window containing information about the current application version.
	2- user reviews the information and chooses to go back	
		3-System redirects user to (US1)
Priority:	Low	

4.7 Class Diagrams

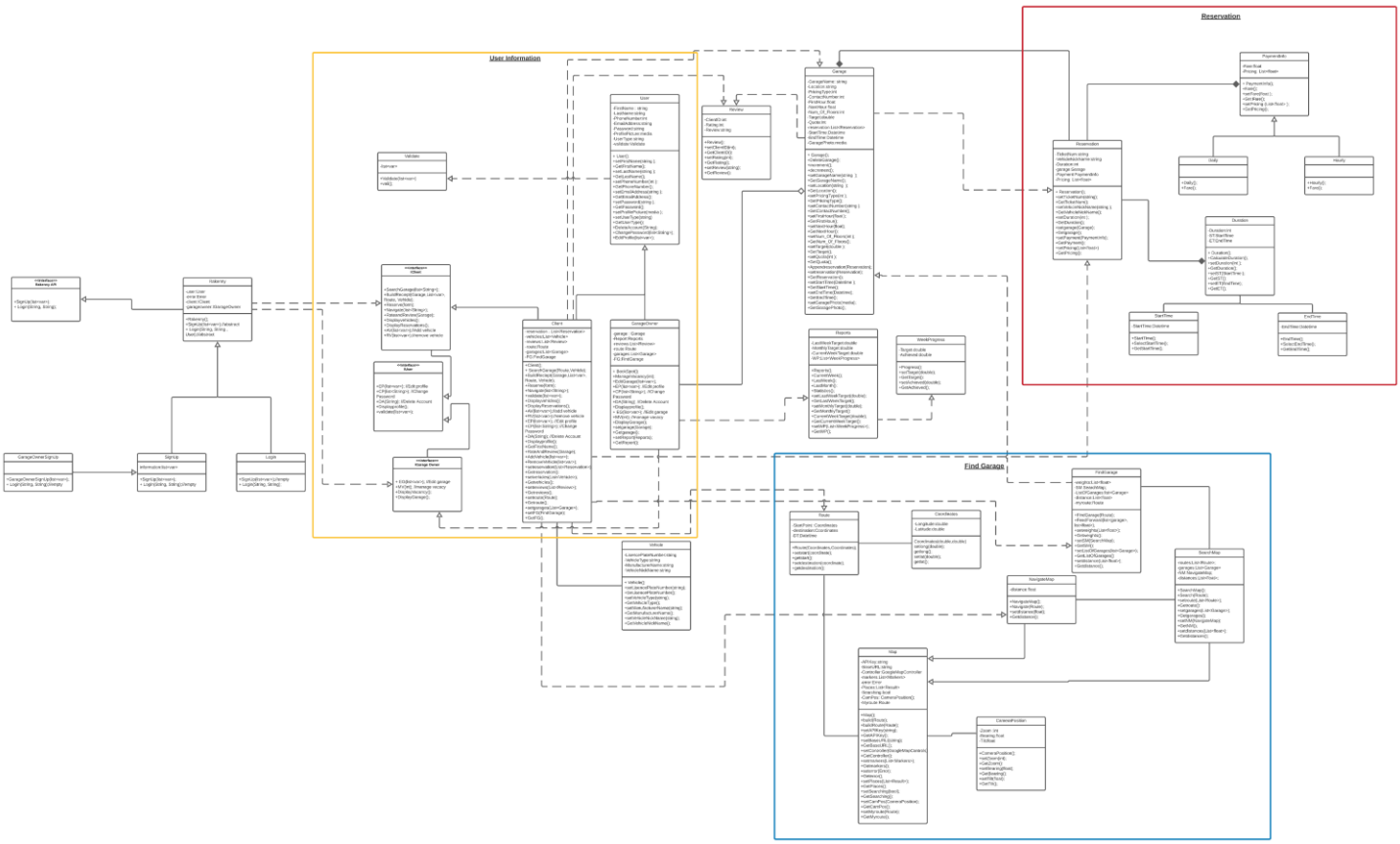

Figure 7

https://lucid.app/lucidchart/invitations/accept/4fc18d82-7388-4b68-91ea-04f5528e2e5f

4.8 Sequence Diagrams

Booking Process

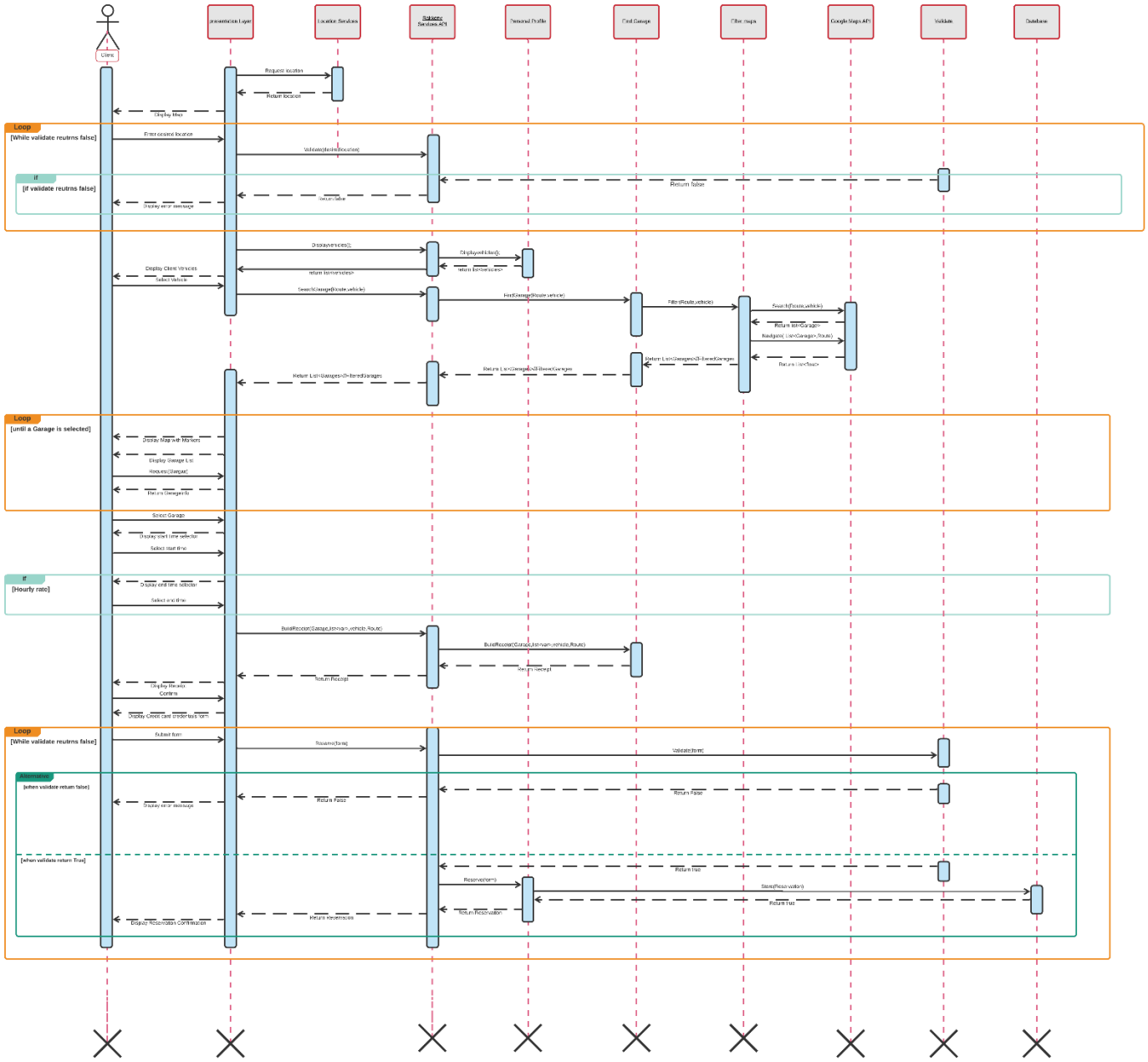

Figure 8

Navigation

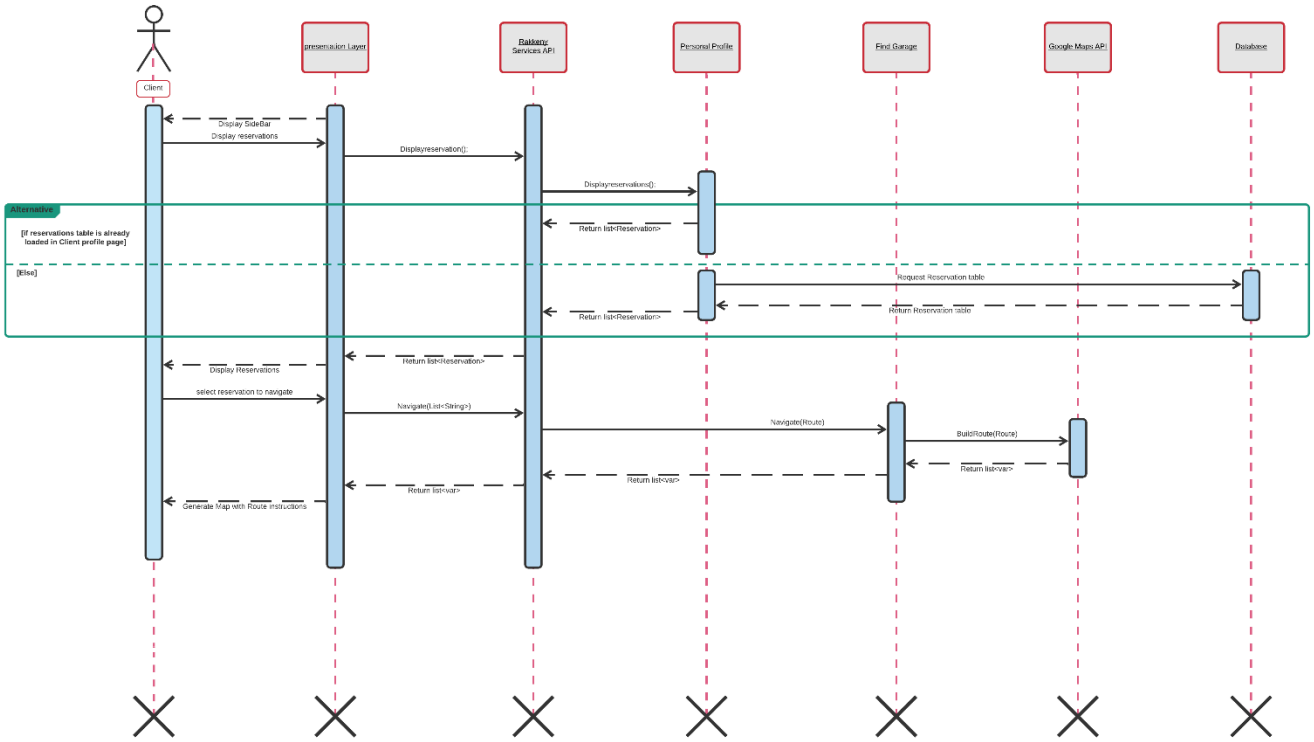

Figure 9

Add or Remove Vehicle

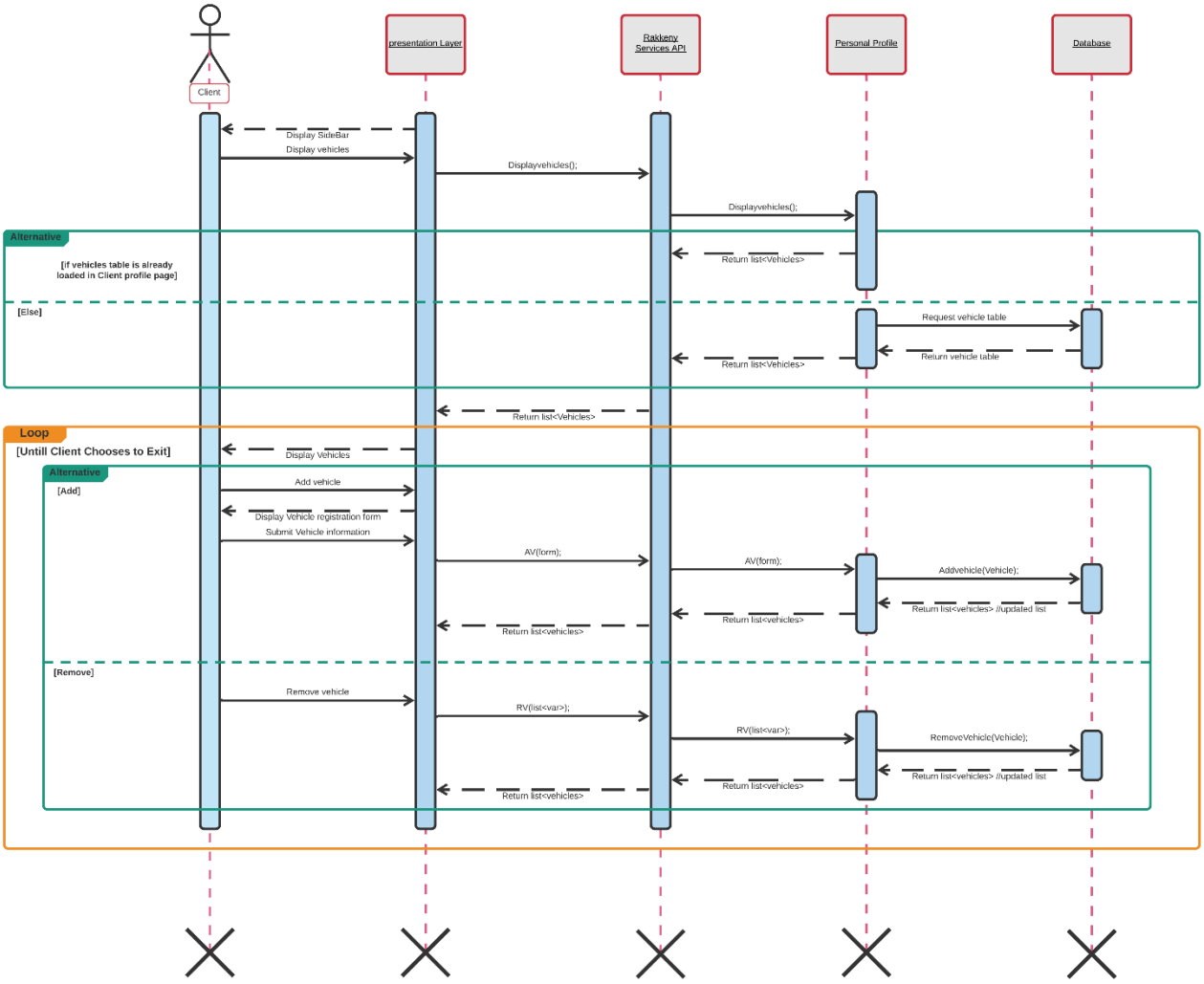

Figure 10

Increment & Decrement

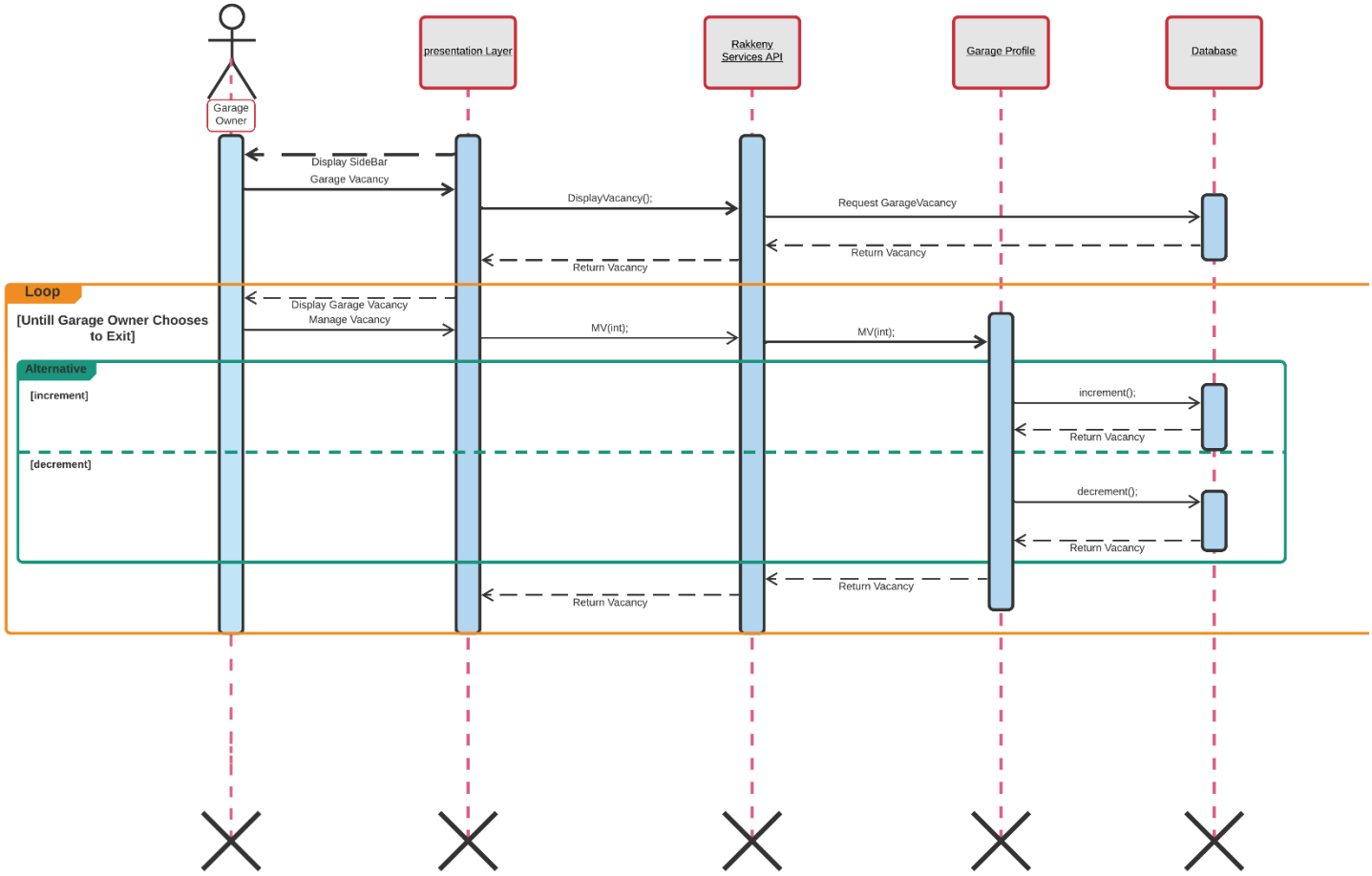

Figure 11

4.9 System GUI Design

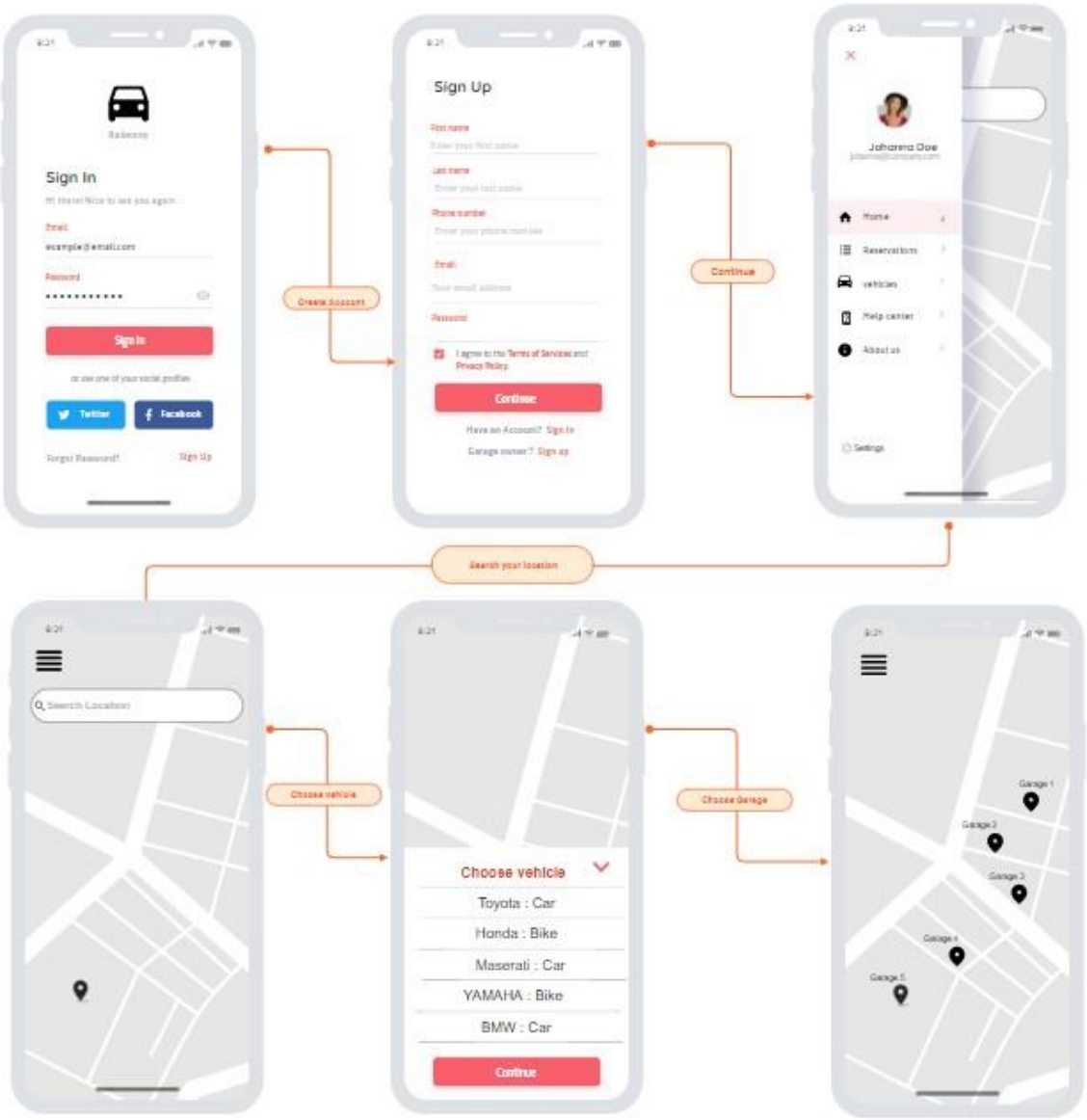

Figure 12

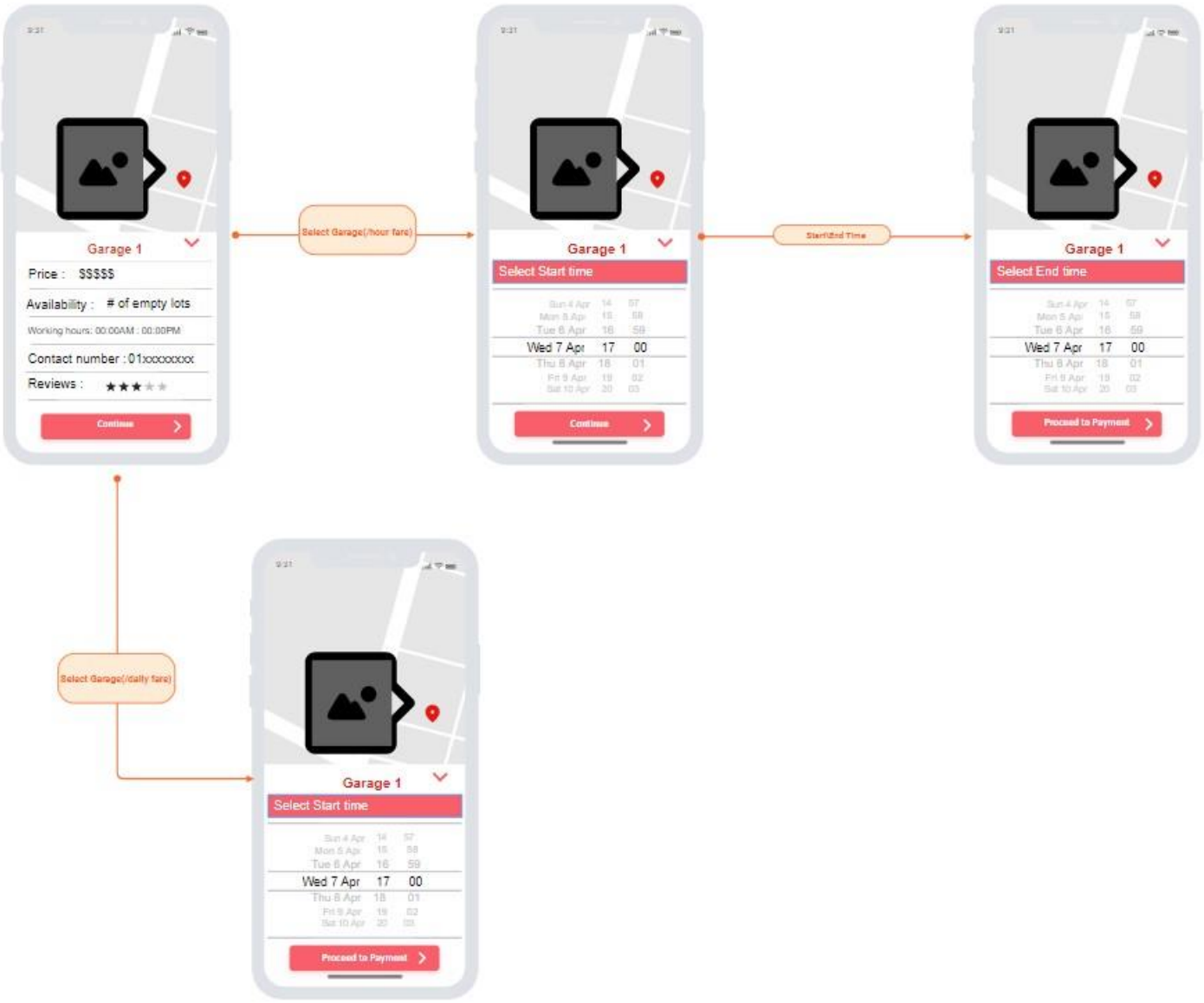
Garage 1
Price : $$$$$
Availability : # of empty lots
Working hours: 00:00AM : 00:00PM
Contact number : 01xxxxxxxx
Reviews :
Continue
Select Garage(/hour fare)
Garage 1
Select Start time
Wed 7 Apr 17 00
Continue
Start/End Time
Garage 1
Select End time
Wed 7 Apr 17 00
Proceed to Payment
Select Garage(/daily fare)
Garage 1
Select Start time
Wed 7 Apr 17 00
Proceed to Payment

Figure 13

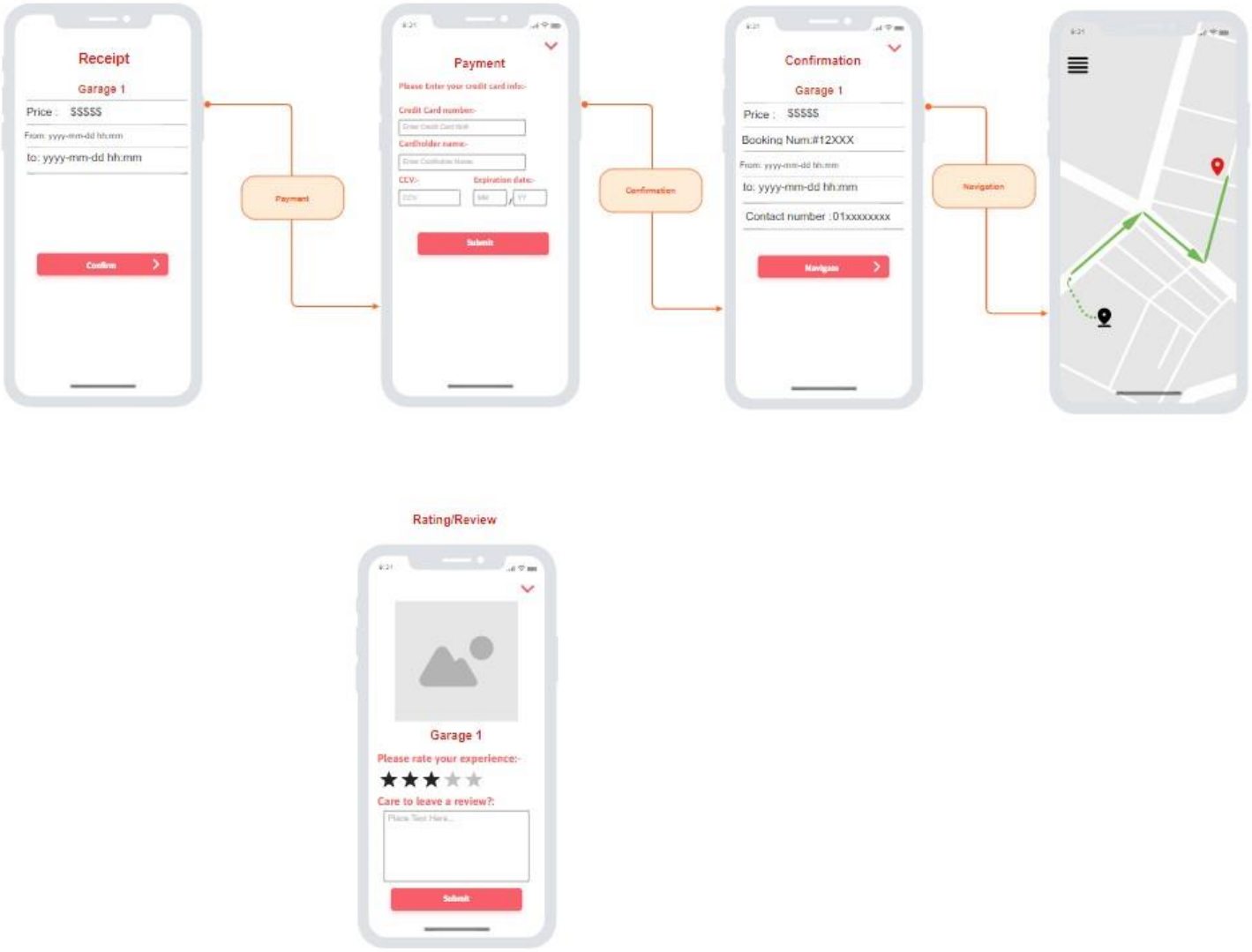
Receipt
Garage 1
Price : $$$$$
From: yyyy-mm-dd hh:mm
to: yyyy-mm-dd hh:mm
Confirm
Payment
Payment
Please Enter your credit card info:-
Credit Card number:-
Cardholder name:-
CCV:-
Expiration date:-
Submit
Confirmation
Confirmation
Garage 1
Price : $$$$$
Booking Num:#12XXX
From: yyyy-mm-dd hh:mm
to: yyyy-mm-dd hh:mm
Contact number :01xxxxxxxx
Navigate
Navigation
Rating/Review
Garage 1
Please rate your experience:-
Care to leave a review?:
Submit

Figure 14

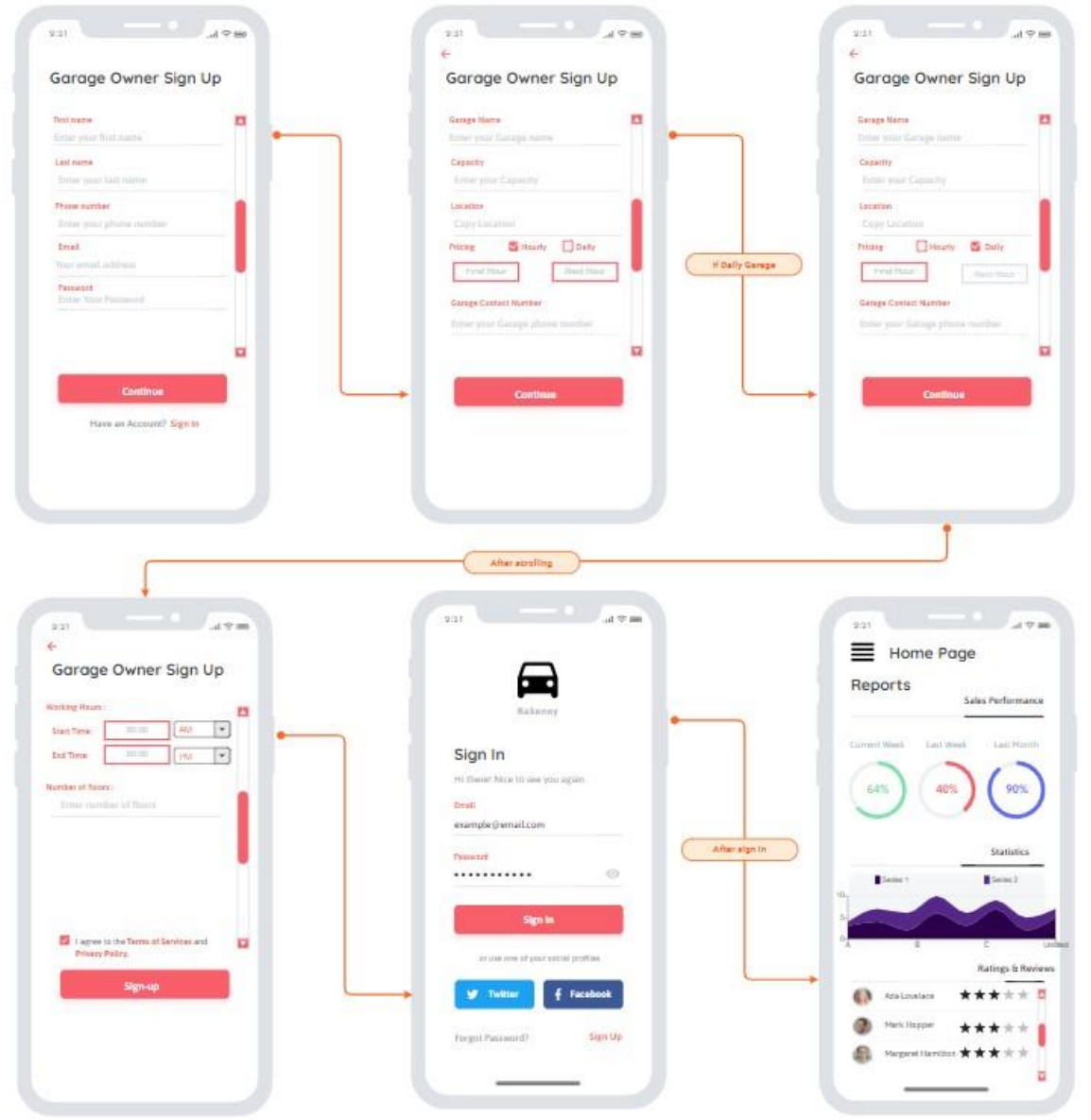
Garage Owner Sign Up
Continue
Have an Account? Sign In
Garage Owner Sign Up
Continue
If Daily Garage
Garage Owner Sign Up
Continue
After scrolling
Garage Owner Sign Up
Sign-up
Sign In
Sign In
Twitter
Facebook
Forgot Password?
Sign Up
After sign in
Home Page
Reports
Sales Performance
64%
40%
90%
Statistics
Ratings & Reviews

Figure 15

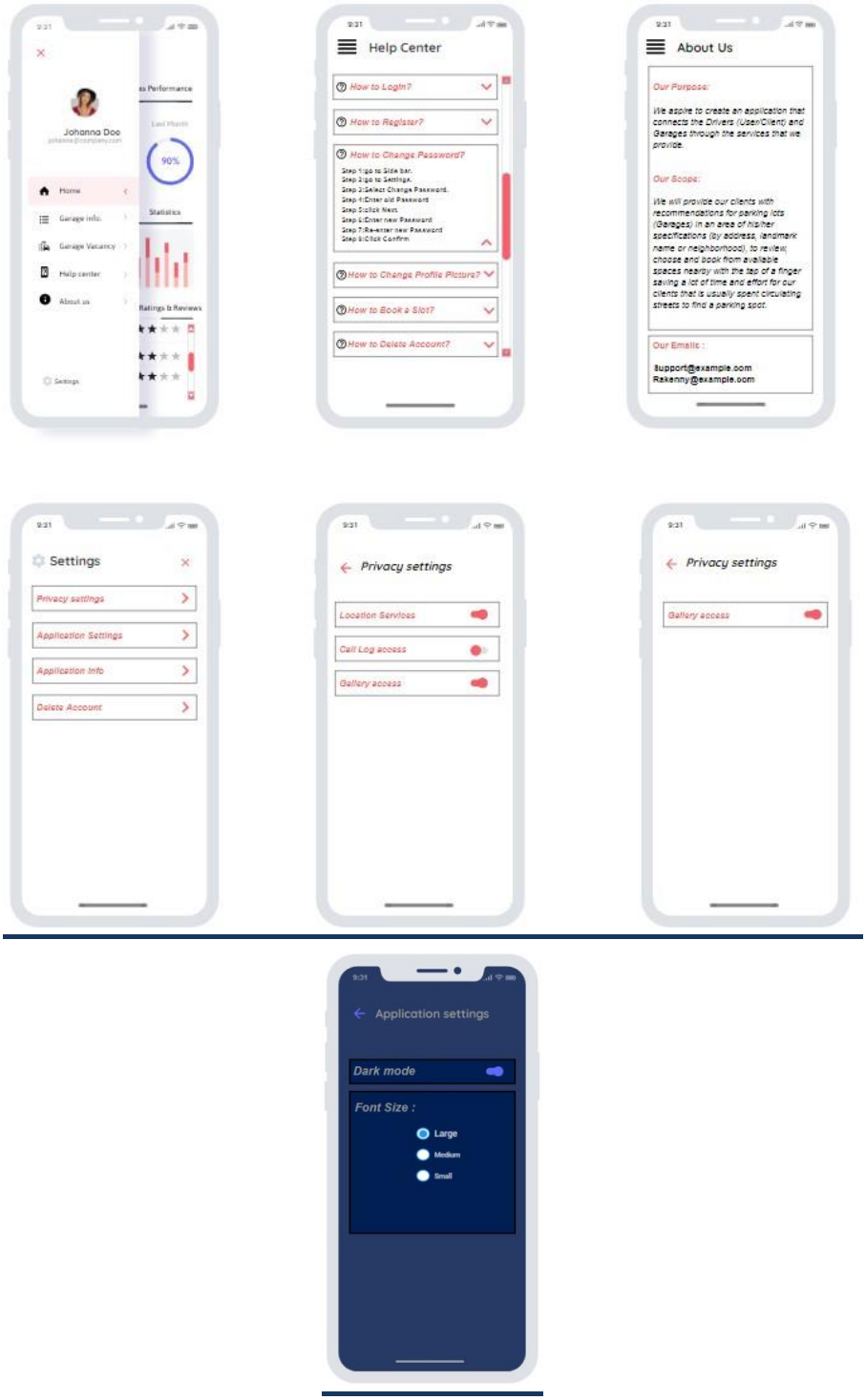
Johanna Doe
90%
Home
Garage info.
Garage Vacancy
Help center
About us
Settings
Help Center
How to Login?
How to Register?
How to Change Password?
How to Change Profile Picture?
How to Book a Slot?
How to Delete Account?
About Us
Our Purpose:
We aspire to create an application that connects the Drivers (User/Client) and Garages through the services that we provide.
Our Scope:
We will provide our clients with recommendations for parking lots (Garages) in an area of his/her specifications (by address, landmark name or neighborhood), to review, choose and book from available spaces nearby with the tap of a finger saving a lot of time and effort for our clients that is usually spent circulating streets to find a parking spot.
Our Emails :
Support@example.com
Rakenny@example.com
Settings
Privacy settings
Application Settings
Application Info
Delete Account
Privacy settings
Location Services
Call Log access
Gallery access
Privacy settings
Gallery access
Application settings
Dark mode
Font Size :
Large
Medium
Small

Figure 16

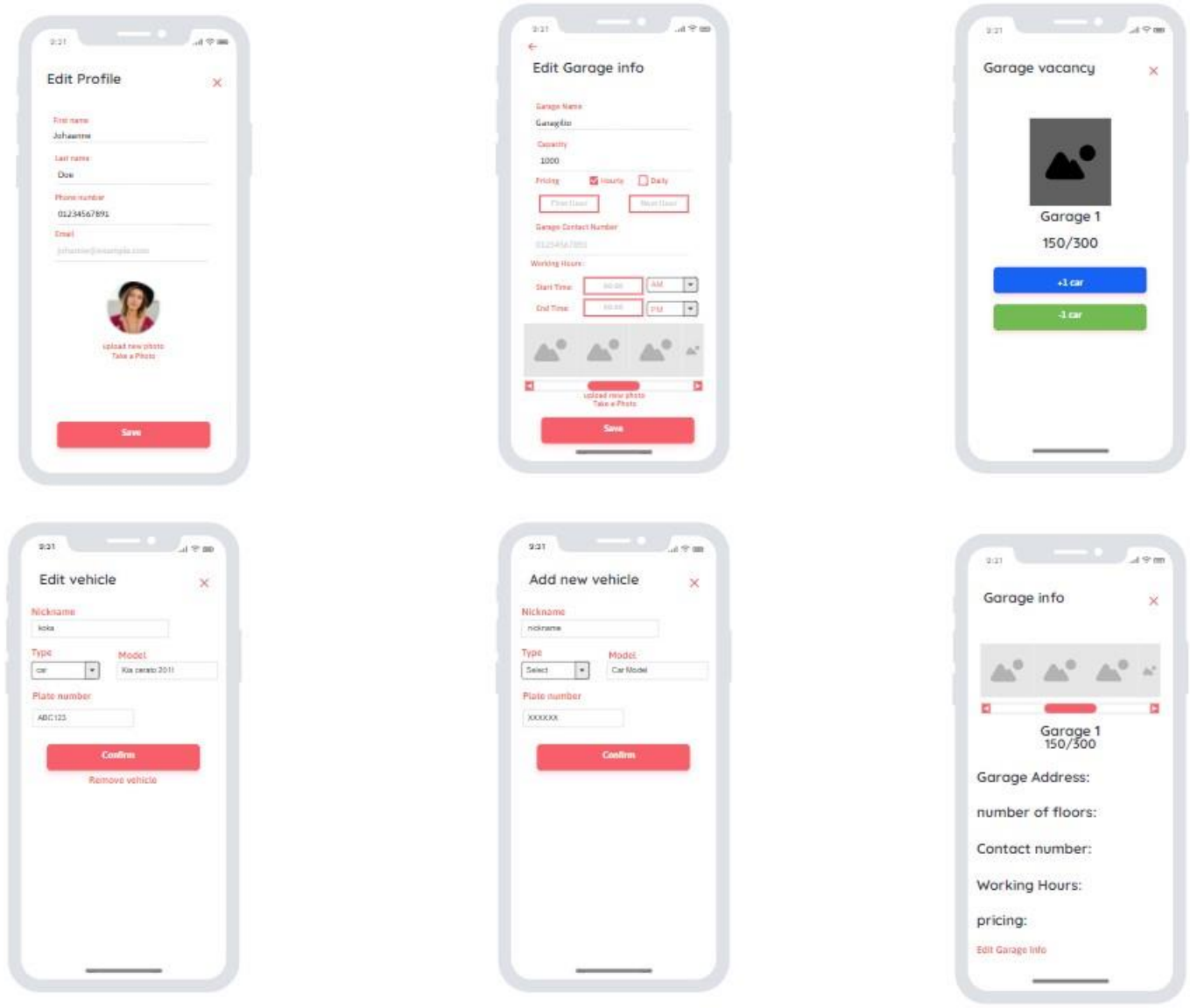
Edit Profile
Save
Edit Garage info
Save
Garage vacancy
Garage 1
150/300
+1 car
-1 car
Edit vehicle
Nickname
Type
Model
Plate number
Confirm
Remove vehicle
Add new vehicle
Nickname
Type
Model
Plate number
Confirm
Garage info
Garage 1
150/300
Garage Address:
number of floors:
Contact number:
Working Hours:
pricing:
Edit Garage Info

Figure 17

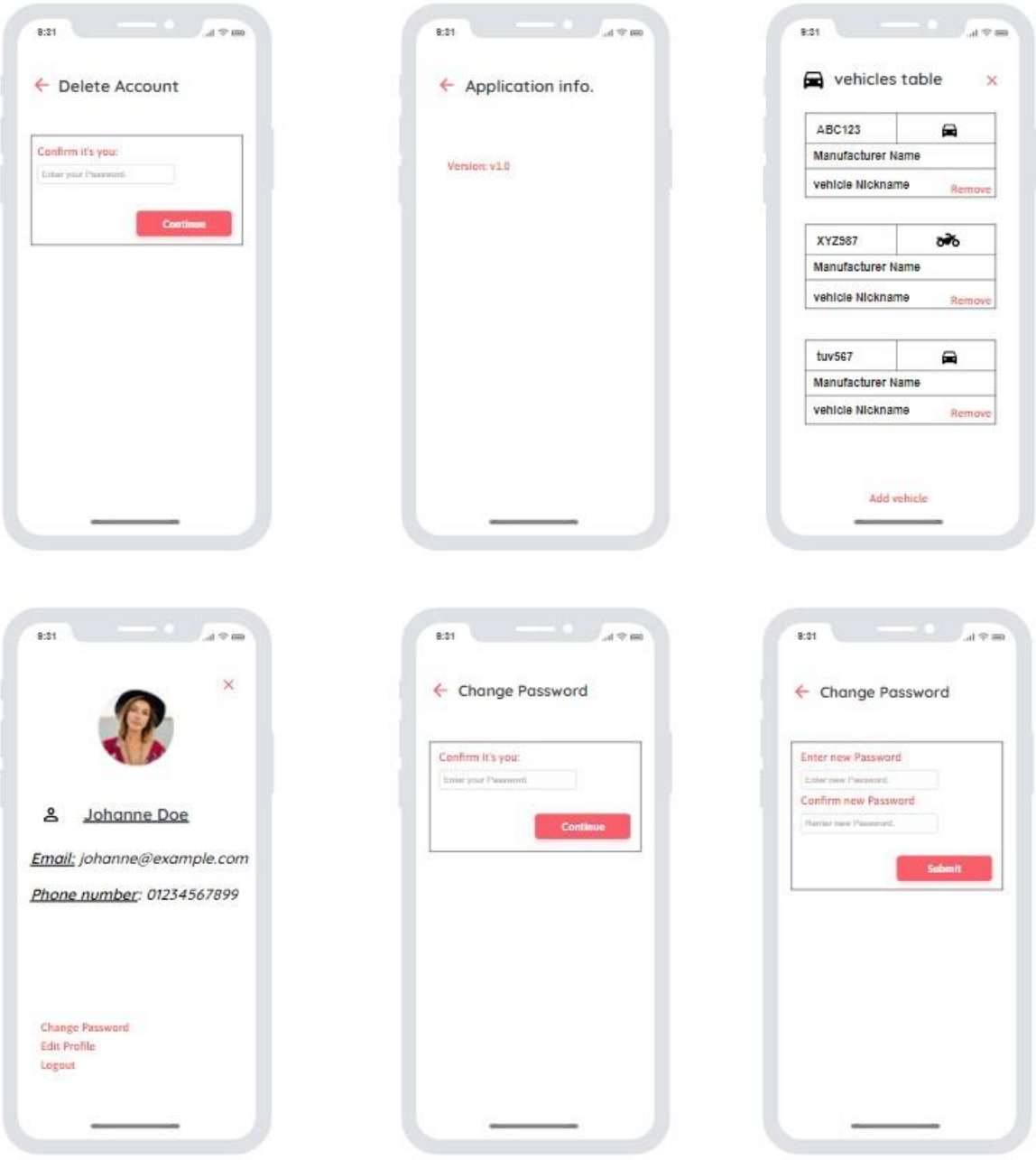
Delete Account
Confirm it's you:
Continue
Application info.
Version: v1.0
vehicles table
ABC123
Manufacturer Name
vehicle Nickname
Remove
XYZ987
Manufacturer Name
vehicle Nickname
Remove
tuv567
Manufacturer Name
vehicle Nickname
Remove
Add vehicle
Johanne Doe
Email: johanne@example.com
Phone number: 01234567899
Change Password
Edit Profile
Logout
Change Password
Confirm it's you:
Continue
Change Password
Enter new Password
Confirm new Password
Submit

Figure 18

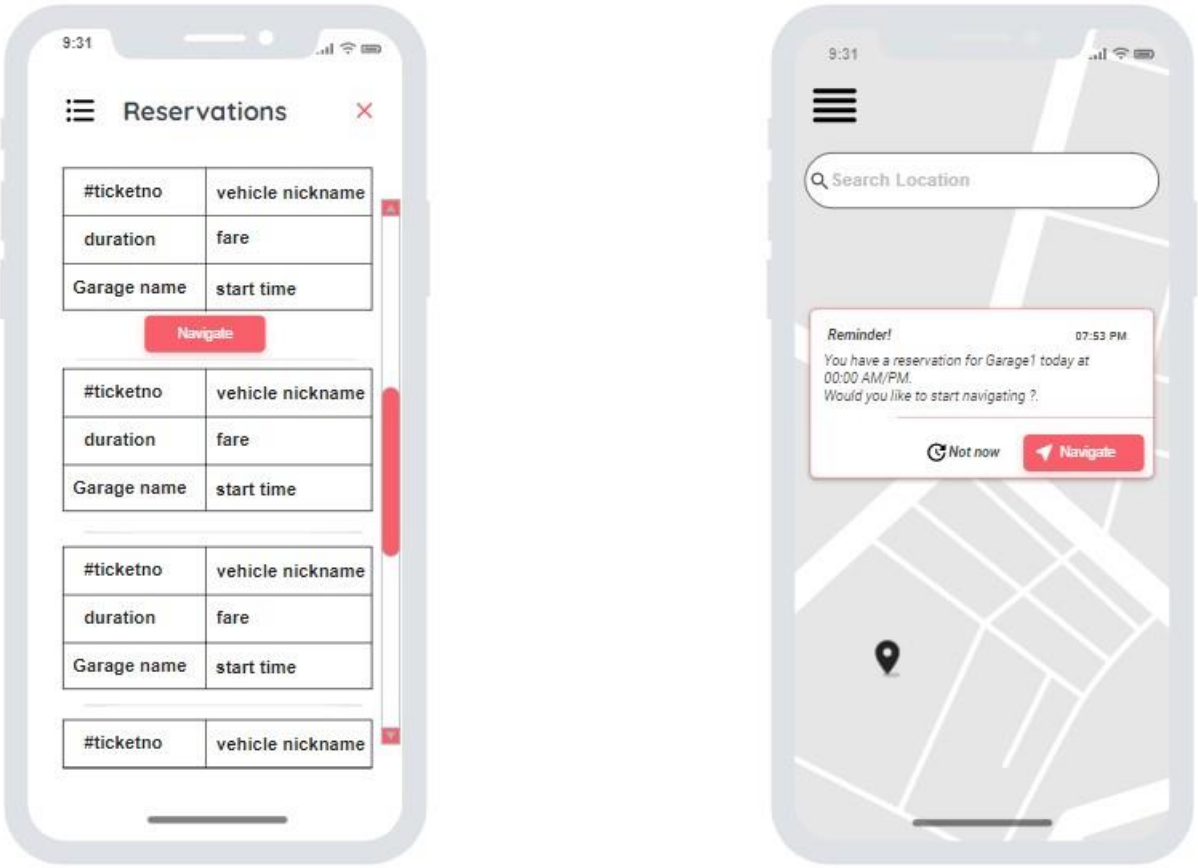
9:31
Reservations
#ticketno
vehicle nickname
duration
fare
Garage name
start time
Navigate
#ticketno
vehicle nickname
duration
fare
Garage name
start time
#ticketno
vehicle nickname
duration
fare
Garage name
start time
#ticketno
vehicle nickname
9:31
Search Location
Reminder!
07:53 PM
You have a reservation for Garage1 today at 00:00 AM/PM.
Would you like to start navigating ?.
Not now
Navigate

Figure 19

5. Work Plan

Task	Task Title	Description	Task status	Days To Complete
1	Problem Definition	Define the problem we are going to work on	12/10/2020	20
2	Idea generation	Generte many ideas as possible for the problem	12/13/2020	15
3	Survey for the project idea	Write a survey to compare between ideas	12/18/2020	7
4	Analysing the ideas	Analysing the ideas we developed	12/27/2020	14
5	Accepting the idea	Accepting which idea we are going to work on	1/12/2021	6
6	Technologies we want to learn	Choosing which technology we are going to use	1/26/2021	20
7	Collecting data	Collecting all the data related to our idea	2/1/2021	40
8	Design & Architecture	Designing the Architecture for our system	3/17/2021	20
9	Documentation	Documenting everything that is needed for representing our idea in details	3/24/2021	30
10	Front-end implementation	Start to implement the front-end of the application	6/5/2021	90
11	Database implementaion	Start to implement database for the application	6/22/2021	90
12	Backend implemantation	Start to implement the back-end of the application	7/15/2021	90
13	Testing the application	Testing the whole application	8/7/2021	35
14	Finish implementation	Complete the whole Implementation	8/14/2021	0

Figure 20

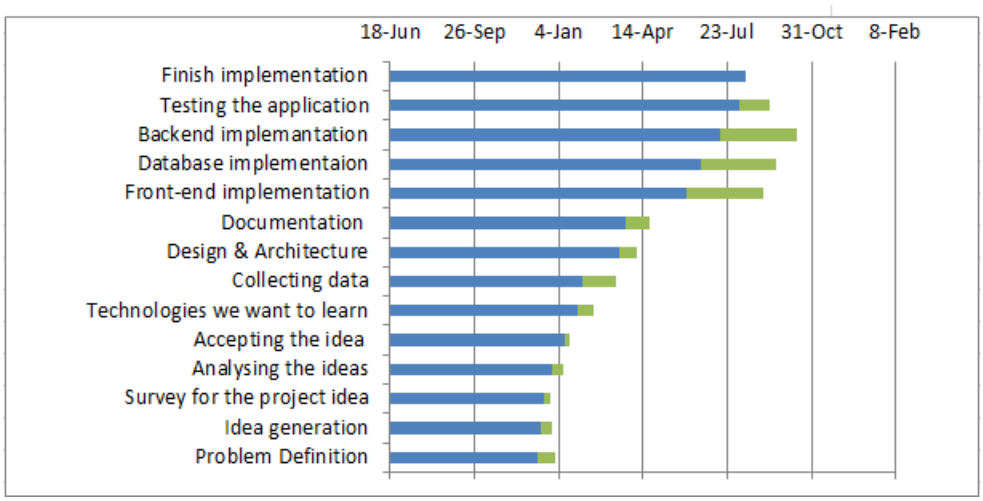

Figure 21