TALKING GARAGE

By

Sudhin Sarker Bishal

(151-35-985)

A thesis submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Software Engineering

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

Fall - 2018

APPROVAL

This **Project** titled "**Talking Garage**", submitted by **Sudhin Sarker Bishal, 151-35-985** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc in Software Engineering and approved as to its style and contents.

BOARD OF EXAMINERS

Dr. Touhid Bhuiyan
Professor and Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Chairman

Mohammad Khaled Sohel
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Md. Shohel Arman
Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Mr. Shafiul Hasan
Managing Director
Vivacom Solution, Dhaka

External Examiner

DECLARATION

I hereby declare that I have taken this thesis under the supervision of **Md. Shohel Arman, Lecturer, Department of Software Engineering, Daffodil International University,** I also declare that neither this thesis nor any part of this has been submitted elsewhere for award of any degree.

সুধীন সরকার বিশাল

Sudhin Sarker Bishal
ID:151-35-985
Batch: 16th
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

Certified by:

24-12-18

Md. Shohel Arman
Lecturer
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

ACKNOWLEDGEMENT

First of all, I am grateful to The Almighty for making me eligible to complete this project. Then I would like to thank my supervisor Md. Shohel Arman, Lecturer, Department of Software Engineering. I am extremely grateful and indebted to him for his expert, sincere and valuable guidance and encouragement extended to motivation, enthusiasm and immense knowledge. I wish to express my sincere thanks to Dr. Touhid Bhuiyan, Professor & Head, Department of Software Engineering for his constant encouragement. Lastly, I would like to thank to my parents, for their unconditional support, love and without them I would not have come so far.

Table of Contents

List of Figures

List of Tables

Chapter 1:
Introduction

1.1 Overview

Traffic jam is a massive problem in Dhaka and the city faces it every moment. Whether it is day or night this problem haunts us like nightmares. There are many reasons for this situation.

Street parking is one of the major reasons for this traffic mess. Well, for a country like ours this situation is a matter of much worries. But we can fix this situation. Many steps had been taken in past but all those steps have their own flaws. Some steps have created other problems like cost and management.

TalkingGarage a smart car parking system has been designed considering all those situations and system flaws. With the help of technology this system can fix all the issues with parking and add a great initiative in solving traffic problem. TalkingGarage has been designed in a way any kind of place can the turned and controlled as a parking lot with a managed system of billing and payments.

The system has two parts:

1. User Application
2. Parking lot device

Users can find available parking space within 1.5 km with the location-based search. After getting their desired parking they can reserve the space. A RFID card will be issued for every user. This card will be used to check in and out after they get into the parking lot. System will trace and collect user's time of staying in the parking lot for creating bills. Billing system is postpaid.

1.2 Purpose

1.2.1 Background

Before setting up my mind to go for this system, I noticed some problems in existing parking lots and the system they are using. Also, the problems of illegal street parking which was affecting the traffic problems. I thought to bring mobility in the same system but just in a different way.

Use unused spaces to reduce the problems of street parking.

Bring the situation in a managed system.

Reduce the hassles of drivers to find spaces to park.

1.2.1 Benefits & Beneficiaries

This system will mostly beneficial for the drivers because searching for a good place to keep their cars is a big hassle. This system takes all his/her tension to park it in a safe place.

Secondly, the parking space owner. This system is suitable to fit in kind of parking spaces whether it is a commercial building or an apartment. They can use their spaces to make a great income out of it.

1.2.2 Goals

The main goal of this system is to solve the problem of street parking and make a new scope of business in our country.

1.2.3 Challenges

It's a common thing in performing an activity is to face some difficulties. There is no work exist without challenge. Similarly, this project might face some challenges:

1. Authority might be unable to handle the application.
2. Lot or car owner might have lack of knowledge about technology.
3. Lack of electricity and internet in the parking lot.
4. Willingness of users to use this application.
5. Expenses.

If we can overcome these challenges, it will be much fruitful for both of the car owners or drivers and lot or garage owner. I strongly believe that, it will not be so tough task to solve these problems.

1.3 Stakeholders

There are many members associate with this project. They have helped to develop the system directly or indirectly.

Internal Stakeholders:

- Owner
- Employees
- Manger

External Stakeholders:

- Car owners
- Parking space owners

1.4 Proposed System Model

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time. SDLC includes a detailed plan for how to develop, alter, maintain, and replace a software system.

SDLC involves several distinct stages, including planning, design, building, testing, and deployment. Popular SDLC models include the waterfall model, spiral model, and Agile model.

Figure 1.1: Spiral model

For this project I have selected spiral model. Because, this is the most flexible of all the SDLC models. The spiral model is similar to the iterative model in its emphasis on repetition. The spiral model goes through the planning, design, build and test phases over and over, with gradual improvements at each pass.

As this system is real-time, it needs to be checked over and over again for avoiding any kind of system failure. I found spiral model the most appropriate SDLC model for this project.

1.5 Project schedule

A. Initial Step

Table 1.1: Initial Project Schedule Table

Serial	Work Description	Start (date)	End (date)	Total days
1	Idea Finding	01/06/2018	08/06/2018	7

2	Feasibility study	08/06/2018	12/06/2018	4
3	Available Source Check	13/06/2018	18/06/2018	5
4	Mind Mapping	19/06/2018	26/06/2018	7
5	Similar site analysis	27/06/2018	30/06/2018	3
			Total Days	26

B. Idea Proposal

Table 1.2: Idea Proposal

Serial	Work Description	Start (date)	End (date)	Total days
1	Idea Finding with Supervisor	01/08/2018	04/08/2018	3
2	Feasibility study With Supervisor	05/08/2018	07/08/2018	2
3	Feature Discussion with Supervisor	08/08/2018	10/08/2018	5
4	Work Flow Maintenance	11/08/2018	16/08/2018	2
5	SDLC Selection	17/08/2018	19/08/2018	2
			Total Days	14

C. Requirement Gathering

Table 1.3: Requirement Gathering

Serial	Work Description	Start (date)	End (date)	Total days
1	System Flow sketch	20/08/2018	28/08/2018	8
2	Requirement gathering for proposed system	29/09/2018	05/09/2018	7
3	Requirement Collection	06/09/2018	08/09/2018	2
4	SRS	09/09/2018	11/09/2018	2

5	All requirement and Information	12/09/2018	16/09/2018	4
			Total Days	23

D. Physical System Design

Table 1.4: Physical System Design

Serial	Work Description	Start (date)	End (date)	Total days
1	Designing Prototype	17/08/2018	22/08/2018	5
2	GUI (Graphical user Interface)	23/08/2018	05/08/2018	12
3	Process Design	06/08/2018	08/08/2018	2
			Total Days	19

E. Logical System design

Table 1.5: Logical System design

Serial	Work Description	Start (date)	End (date)	Total days
1	Use case Diagram Design	09/09/2018	10/09/2018	1
2	Data Flow Diagram Design (level-0)	11/09/2018	12/09/2018	1
3	Data Flow Diagram Design (level-0)	13/09/2018	14/09/2018	1
4	Data Flow Diagram Design (level-1)	15/09/2018	16/09/2018	1
5	Entity Relationship Diagram	18/09/2018	19/09/2018	1
			Total Days	5

F. Development Phase

Table 1.6: Development Phase

Serial	Work Description	Start (date)	End (date)	Total days
1	Build Maps Module	20/10/2018	27/10/2018	7
2	Build Lot Module	28/10/2018	03/11/2018	7
3	Build Booking Module	04/11/2018	12/11/2018	8
4	Multiple Auth	13/11/2018	16/11/2018	3
5	Database Integration	17/11/2018	19/11/2018	2

6	Hardware Implementation	20/11/2018	25/11/2018	5
7	Billing Module	26/11/2018	28/11/2018	2
			Total Days	34/130

G. System Testing

Table 1.7: System Testing

Serial	Work Description	Start (date)	End (date)	Total days
1	Separate Module Test	28/11/2018	30/11/2018	2
2	Boundary Value Testing	1/12/2018	3/12/2018	3
3	Functionality Test	4/12/2018	6/12/2018	2
			Total Days	7

Chapter 2:
Software Requirement Specification

Requirement elicitation in my project is to find out what we need to full fill this project. This is very first step to build this project. Talking Garage is a web based and IoT application to maintain parking. Two stakeholders are in this application.

a) Parking-Lot Owner
b) Car owner/driver or user

For those stakeholders there have many functionalities and non-functionalities. I will discuss in SRS subsection. Feature and quality requirement of lot owners and drivers are also included in this section.

Studying similar project: Most of the requirements are collected by studying similar projects. I observed their projects carefully and make my project advance. I observed their system not way of making. Some common feature is in our system such as finding nearest parking space but I have provided some extra and powerful feature that they don't have. Instant booking, RFID, Check In/Out, Realtime space availability etc. I have tried my best to make this system smarter than others.

Names of applications I have studied:

- ParkingKoi
- JustPark

Experts opinion: I talked about my project with many web developers and tried to understand what they have suggested me. Some of they have real life experienced in this sector. They suggested some key things. And that was very important for this project. A security expert mark down some issues that we might face.

Brainstorming: Except similar projects and expert's opinion everything I did in this project is brainstorming. I wrote down everything in a scratch book and reviewed all the term very carefully. Then I sorted out the best outcome and tried to compare with our real life. I have discussed mostly based on Advanced feature ideas, programming capability.

2.1 Functional Requirements

The whole project has to develop on the basis of the following requirements.

- End user must be registered in system.
- In login process, system will fetch data from database and confirm identification for multiple user.

- Activities will create for particular user. Activities will be sortable and time notable.
- System will fetch map data and place them in maps with user location. After reservation a countdown timer will start.
- Application will auto refresh while user is checking in and out.
- User cannot book other lots while he already checked in a parking-lot.
- Every reservation should have to be cancelled if the user is unable to reach in 30 minutes and user cannot book while this he has already booked.
- Each request from the CheckIn or CheckOut devices must have to be checked if they are giving correct API key or not.

Table 2.1: Functional Requirements

FR-01	**User registration**
Description	End user must be registered in system.
Stakeholders	User

FR-02	**User authentication**
Description	In login process, system will fetch data from database and confirm identification for multiple user.
Stakeholders	User

FR-03	**Activities**
Description	Activities will create for particular user. Activities will be sortable and time notable.
Stakeholders	User

FR-04	**Map data fetching**
Description	System will fetch map data and place them in maps with user location. After reservation a countdown timer will start.
Stakeholders	User

FR-05	**Map page**
Description	Application will auto refresh while user is checking in and out.
Stakeholders	User

FR-06	**Book parking space**
Description	User cannot book other lots while he already checked in a parking-lot.
Stakeholders	User

FR-07	Reservation time
Description	Every reservation should have to be cancelled if the user is unable to reach in 30 minutes and user cannot book while this, he has already booked
Stakeholders	User

FR-08	Device requests
Description	Each request from the CheckIn or CheckOut devices must have to be checked if they are giving correct API key or not.
Stakeholders	User

FR-09	Member Information
Description	This module helps admin to register alumni members. Admin is able to maintain all the information of alumni members
Stakeholders	User

FR-10	Developer's access
Description	Developer would the ability access all data in testing purpose
Stakeholders	N/A

FR-11	User Access
Description	Only registered users have access to enter system. System will not give access to view another page which is guard by other type user.
Stakeholders	User

FR-12	User interface
Description	User friendly interface and easy system should be built. Non-technical people can use this system easily.
Stakeholders	N/A

2.2 Non-Functional Requirements

- Easy to use. Comfortable for various device & browsers.
- SQLSERVER default capacity.
- During the process, to update any type of data response time will not more than one second.

- Password sorting hash, SQL query, protect SQL injection, CSRF, Encryption.
- Timers should start in no time.
- Run smoothly in cross platform. IIS is recommended.

Table 2.2: Non-Functional Requirements

NFR-01	**Devices**
Description	Easy to use. Comfortable for various device & browsers.
Stakeholders	N/A

NFR-02	**Latency**
Description	During the process, to update any type of data response time will not more than one second.
Stakeholders	N/A

NFR-03	**Encryption**
Description	Password sorting hash, SQL query, protect SQL injection, CSRF, Encryption.
Stakeholders	N/A

NFR-04	**Time**
Description	Timers should start in no time.
Stakeholders	N/A

NFR-05	**Server**
Description	Run smoothly in IIS
Stakeholders	N/A

2.3 Performance Requirements

It's very important to maintain the performance of the project. To ensure a good performance, this project have to meet some requirements which will ensure a good performance.

2.3.1 Speed and Latency Requirements

While inserting or viewing the system in the browser, system need a minimum amount of speed to perform the task.

Table 2.3: Speed and Latency Requirements

SLR-01	**Map load time**
Description	The map should be loaded between 5-10 seconds minimum.
Stakeholders	N/A

SLR-02	**Booking process**
Description	Booking process should start immediately
Stakeholders	N/A

SLR-03	**Check in process**
Description	Check-in process should work immediately
Stakeholders	N/A

SLR-04	**Check out process**
Description	Check-out process should work immediately
Stakeholders	N/A

2.3.2 Precision and Accuracy Requirements:

System have to ensure the precision and accuracy of the data.

Table 2.4: Precision and Accuracy Requirements

PAR-01	**Map data**
Description	The map data must have to accurate for errorless location
Stakeholders	N/A

PAR-02	**Availability data**
Description	Availability data must have to accurate
Stakeholders	N/A

PAR-03	**Device data**
Description	Data from the devices must have to accurate
Stakeholders	N/A

PAR-04	**Billing info**
Description	Billing information should be correct
Stakeholders	N/A

2.3.3 Capacity Requirements

System will manage all the data.

Table 2.5: Capacity Requirements

CPR-01	**Database storage**
Description	All the data should be stored in database
Stakeholders	N/A

2.4 Dependability Requirements

By the terms of dependability, it does not mean that this project is totally rely on something. Here, dependability means the running time of this project.

2.4.1 Reliability & Availability Requirements

In order to support global and smooth operations the system must be available around the clock. On the other hand, most of services in this system are not mission-critical.

Table 2.6: Reliability & Availability Requirements

RAR-01	**System availability**
Description	The system must be available 24 hours in a day
Stakeholders	N/A

RAR-02	**System update**
Description	The system must be updated regularly
Stakeholders	N/A

RAR-03	**Report generation**
Description	The system must generate report and other things in time
Stakeholders	N/A

2.4.2 Robust and fault tolerance Requirements

In every system, there will have some person for destroying something. System will have to handle this type of person easily

Table 2.7: Robust and Fault Tolerance Requirements

RFTR-03	**Multiple requests**
Description	Sometimes multiple user can over access to this system. The system can handle multiple user access
Stakeholders	N/A

2.4.3 Safety critical Requirements

There are no specific safety critical requirements

2.5 Maintainability and supportability

To look after or maintain and support the project some person has to associate with this project.

2.5.1 Maintenance Requirements

Table 2.8: Maintenance Requirements

MR-01	**System update**
Description	System helps to update the accounts information and user info at any time.
Stakeholders	N/A

2.5.2 Supportability Requirements

SRS-1. In order to understand the system's behavior on a technical support required by the system operator.
SRS-2. System malfunction has occurred and the system operator has to find the exact point of the time when this happened.

SRS-3. System produces wrong results and the developers must be able to reproduce the data flow through the system.

SRS-4. Hacker tried to breach the system's security mechanisms and the system operator must understand what he did.

2.5.3 Adoptability Requirements

There are no specific adaptability Requirements.

2.6 Security Requirements

To get access to this system or a specific module the system must provide a central authentication mechanism. In order to prevent anyone to exploit stolen participants all participants password must be encrypted in hash process.

2.6.1 Access Requirements

To get access to the system, the system provides authorization/authentication way. This system uses various modules.

Table 2.9: Access Requirements

AR-01	**Security services**
Description	The system is designed in way that allows all modules to access a mechanism that provides security services.
Stakeholders	N/A

2.6.2 Integrity Requirements

To protect credentials of user from being stolen, all passwords are stored in encrypted form. The Requirements significantly reduces the value of stolen user credentials, it's not easy to decrypt the password.

2.6.3 Privacy Requirements

The system provides a protection of the database in the server. However, the system will have to increment this level of protection because of the personal data mode available on the system & the larger share of people that will be having access to it through the system's registration. The user's privacy will be granted by the limited access that the log in process is going to give to the database.

Table 2.10: Privacy Requirements

PR-01	**Data Protection**
Description	All data will be protected
Stakeholders	N/A

2.7 Usability and Human Integrity Requirements

2.7.1 Ease of Use Requirements

The system is easy to use and can easily be understandable.

Table 2.11: Ease of Use Requirements

EUR-01	**Usability**
Description	The system must be usable for participants with all associate stakeholders.
Stakeholders	N/A

2.7.2 Understandability and Politeness Requirements

Table 2.12: Understandability and Politeness Requirements

UPR-01	**Understandable**
Description	The system is understandable for both of user. The system will not use any term that is not specified in this system.
Stakeholders	N/A

2.7.3 Accessibility Requirements

There are no specific accessibility requirements.

2.8 Look and Feel Requirements

There should not exist any unnecessary things on this project.

2.8.1 Appearance Requirements

It should be clear to a user how to book parking space and check-in to this system.

Table 2.13: Appearance Requirements

APR-01	**Understandable**
Description	Mandatory sections must be identifiable.
Stakeholders	N/A

2.8.2 Style Requirements

User interface will be web based. For styling the interface and making lucrative, I need to use CSS3 and JavaScript.

Table 2.14: Style Requirements

STR-01	**Styling**
Description	The styling of the elements of the web-based user interface will be defined using CSS3 and JavaScript.
Stakeholders	N/A

2.9 Operational and Environmental Requirements

Operational and environmental requirements is very important because this project may not work in every environment and its operation may not accurate in every time.

2.9.1 Expected Physical Requirements

There are no specific expected physical requirements.

2.9.2 Requirement for Interfacing with Adjacent System

There is no specific interfacing with adjacent system requirements

2.9.3 Release Requirements

This project will be released according to the project schedule.

2.10 Legal Requirements

Fraudulent data and engaging third party software or third person is totally prohibited.

2.10.1 Compliance Requirements

Compliance requirements are only guidelines for compliance with the hundreds of laws and regulations applicable to the specific type assistance used by the recipient, and their objectives are generic in nature due to the large number of federal programs. Each compliance requirement is identified by a letter, in alphabetical order.

2.10.2 Standard Requirements

To comply with the Open Standards Requirement, an "open standard" must satisfy the following criteria. If an "open standard" does not meet these criteria, it will be discriminating against open source developers.

- **No Intentional Secrets**: The standard MUST NOT withhold any detail necessary for interoperable implementation. As flaws are inevitable, the standard MUST define a process for fixing flaws identified during implementation and interoperability testing and to incorporate said changes into a revised version or superseding version of the standard to be released under terms that do not violate the OSR.

- **Availability**: The standard MUST be freely and publicly available (e.g., from a stable web site) under royalty-free terms at reasonable and non-discriminatory cost.

- **No Agreements**: There MUST NOT be any requirement for execution of a license agreement.

Chapter 3:
System Analysis

3.1 Use case diagram

The following diagram has been depicted with two users. The relationship of different node with these two users clarify the system in brief.

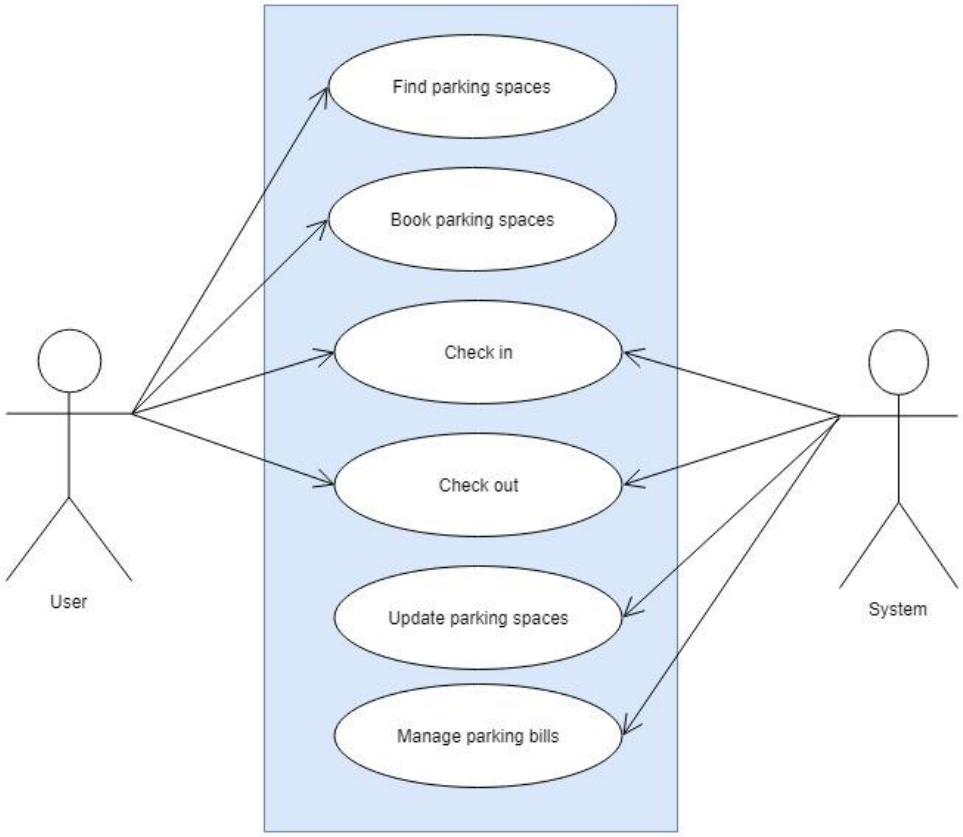

Figure 3.1: Use case diagram

3.1.1 Find parking spaces

Table 3.1: Find parking spaces

Use case title	Find parking spaces.
Goal	Finding the nearest parking lots around 1.5 km.
Preconditions	User should have to be registered.
Success End Condition	If available then show parking lot(s) in the list with distance and available space.
Failed End Condition	Map will not work correctly and no data will be shown.
Primary Actors: **Secondary Actors:**	Users/Drivers N/A
Trigger	Location based search starts.
Description / Main Success Scenario	• User will log in to his/her account • Map data will start loading and initializing • The location-based search will start automatically • If spaces are available near 1.5 km then it will be shown on the list.
Alternative Flows	N/A
Quality Requirements	N/A

3.1.2 Book parking spaces

Table 3.2: Book parking spaces

Use case title	Book parking spaces.
Goal	Reserve a space in a particular lot.
Preconditions	User should have to be registered.
Success End Condition	If the user is not checked in anywhere then book a space and 30 minutes countdown timer.
Failed End Condition	Map will not work correctly and no data will be shown and the timer will show “00:00” and will not refresh automatically.
Primary Actors: **Secondary Actors:**	Users/Drivers N/A
Trigger	30 minutes countdown timer starts.
Description / Main Success Scenario	After use case 3.1.1 is completed, user will get a page where he will be shown a countdown timer.
Alternative Flows	N/A
Quality Requirements	N/A

3.1.3 Check-in

Table 3.3: Check-in

Use case title	Check-in
Goal	Let the system know that user has arrived in the lot.
Preconditions	User should have to be registered.
Success End Condition	Reservation timer will stop and the page will refresh and checkpoint page will appear with a count up timer.
Failed End Condition	Map will not work correctly and no data will be shown.
Primary Actors: **Secondary Actors:**	Users/Drivers, System N/A
Trigger	Timer will start.
Description / Main Success Scenario	After the completion of use case 3.1.1 and 3.1.2, when the user will have arrived to his reserved parking lot, there he will find a device "CheckOut Machine". After tapping his RFID card, the device will send a ping to the server. If server approves the request, on the application site user will see that the browser is refreshing and a timer has started which will tell the user that how long he is been staying in the lot.
Alternative Flows	N/A
Quality Requirements	N/A

3.1.4 Update parking spaces

Table 3.4: Update parking spaces

Use case title	Update parking spaces
Goal	Server updates parking lot number of available spaces.
Preconditions	N/A
Success End Condition	All the user will see the number of available spaces left in a particular parking lot.
Failed End Condition	Map will not work correctly and no data will be shown.
Primary Actors: **Secondary Actors:**	System N/A
Trigger	N/A
Description / Main Success Scenario	After the completion of use case 3.1.1, 3.1.2 and 3.1.3 the server will update the available space left in the parking lot.
Alternative Flows	N/A
Quality Requirements	N/A

3.1.5 Check-out

Table 3.5: Check-out

Use case title	Check-out
Goal	Let the system know that user has left the lot.
Preconditions	User should have to be registered.
Success End Condition	Check in timer will stop and get back to use case 3.1.1
Failed End Condition	Map will not work correctly and no data will be shown.
Primary Actors: **Secondary Actors:**	Users/Drivers, System N/A
Trigger	Timer will stop and page will refresh.
Description / Main Success Scenario	Check in timer will stop and get back to use case 3.1.1
Alternative Flows	N/A
Quality Requirements	N/A

3.1.6 Manage parking bills

Table 3.6: Manage parking spaces

Use case title	Manage parking bills
Goal	See all the information about check ins and outs and charges for the stay.
Preconditions	User should have to be registered.
Success End Condition	Billing page will appear.
Failed End Condition	Billing page will not appear.
Primary Actors: **Secondary Actors:**	Users/Drivers N/A
Trigger	N/A
Description / Main Success Scenario	If user wants to check their monthly parking bills and payments then they will click “My account” Option.
Alternative Flows	N/A
Quality Requirements	N/A

3.2 Activity Diagram

Following activity diagrams are precisely depicting the flow of the different state of the project.

3.2.1 Location-Based Search Activity

This diagram shows how the system searches for nearby parking lots.

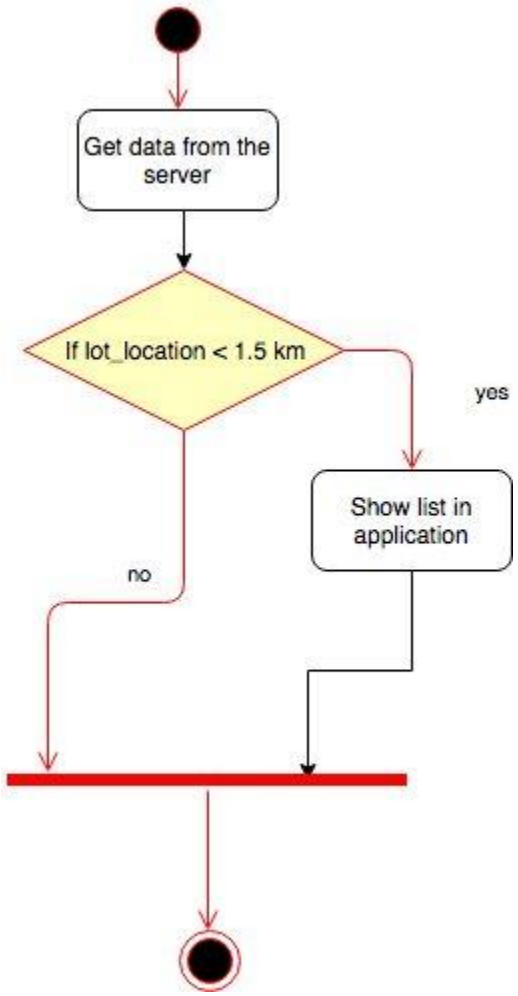

Figure 3.2: Search activity diagram

3.2.2 Booking Activity

This diagram shows how user books a parking lot.

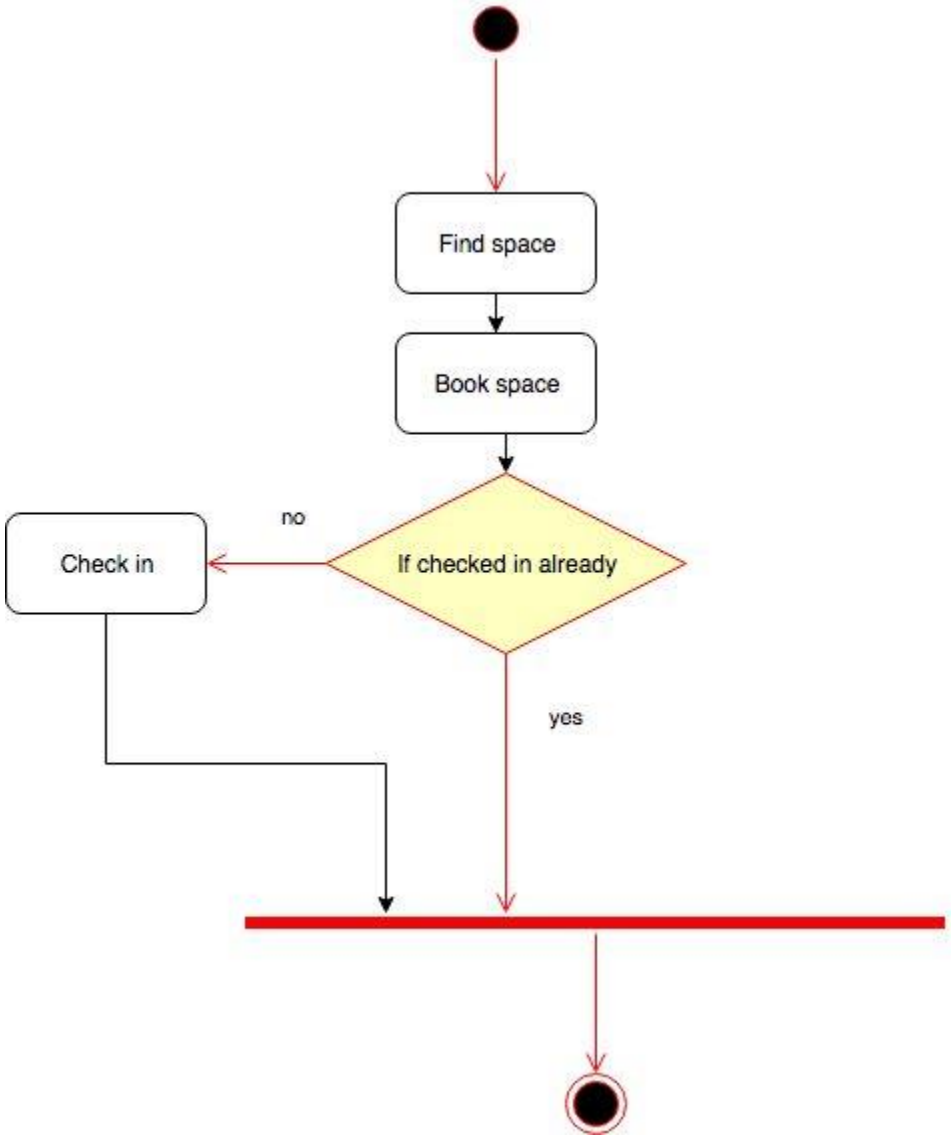

Figure 3.3: Booking activity diagram

3.2.3 Check-in Activity

This diagram shows how user checks in to a lot.

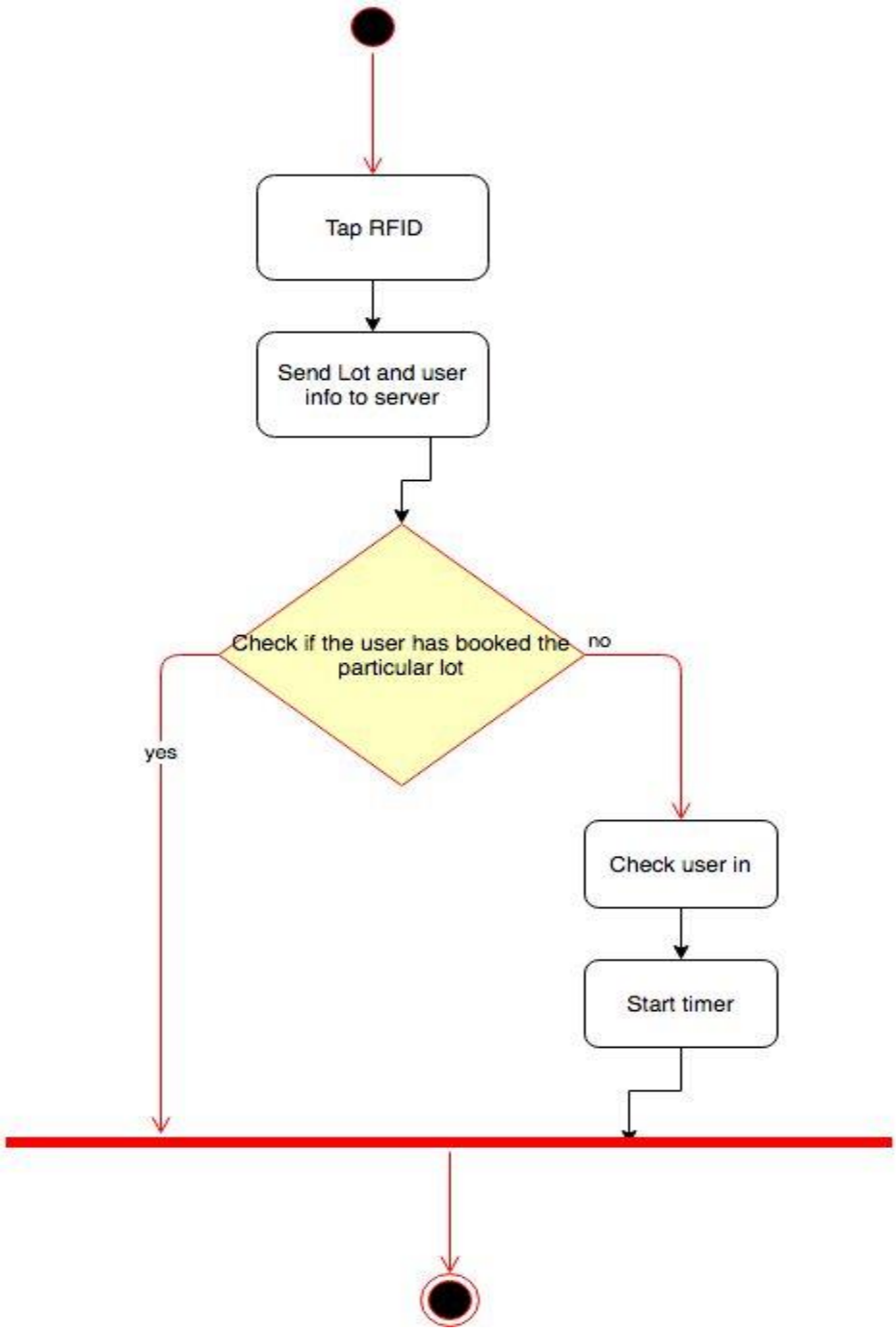

Figure 3.4: Check-in activity

3.2.4 Check-out Activity

This diagram shows how user checks out from a lot.

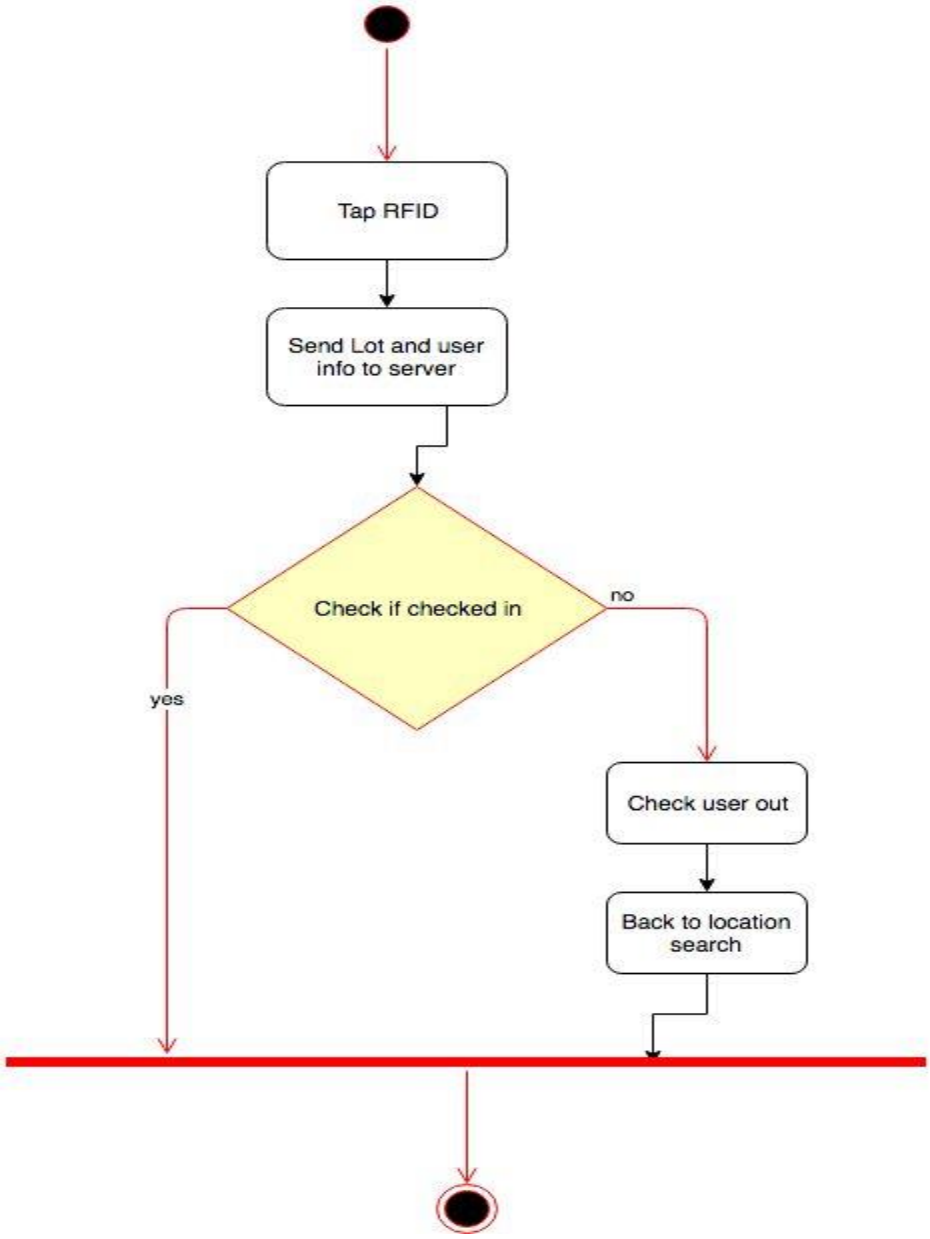

Figure 3.5: Check-out diagram

3.3 Sequence Diagram

Data should be flowed sequentially in a project. The following sequential diagrams show the data, in which the data are flowing sequentially.

3.3.1 Finding parking space

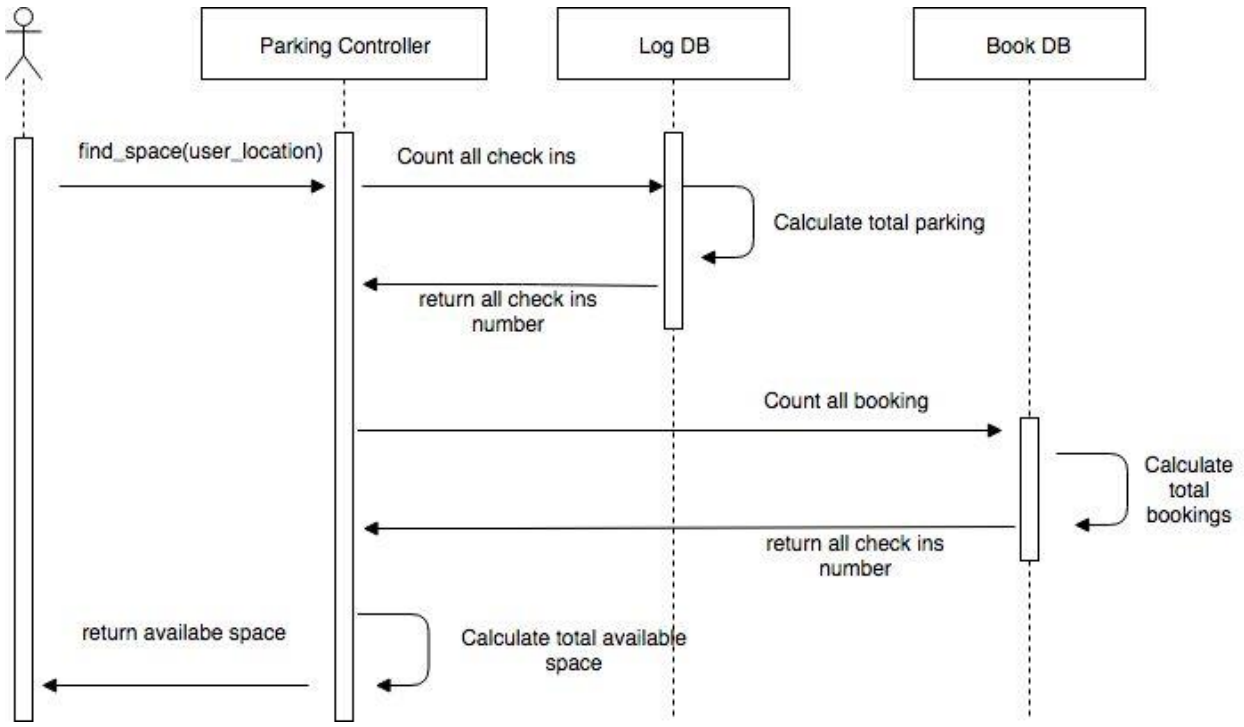

Figure 3.6: Finding parking space

3.3.2 Booking parking space

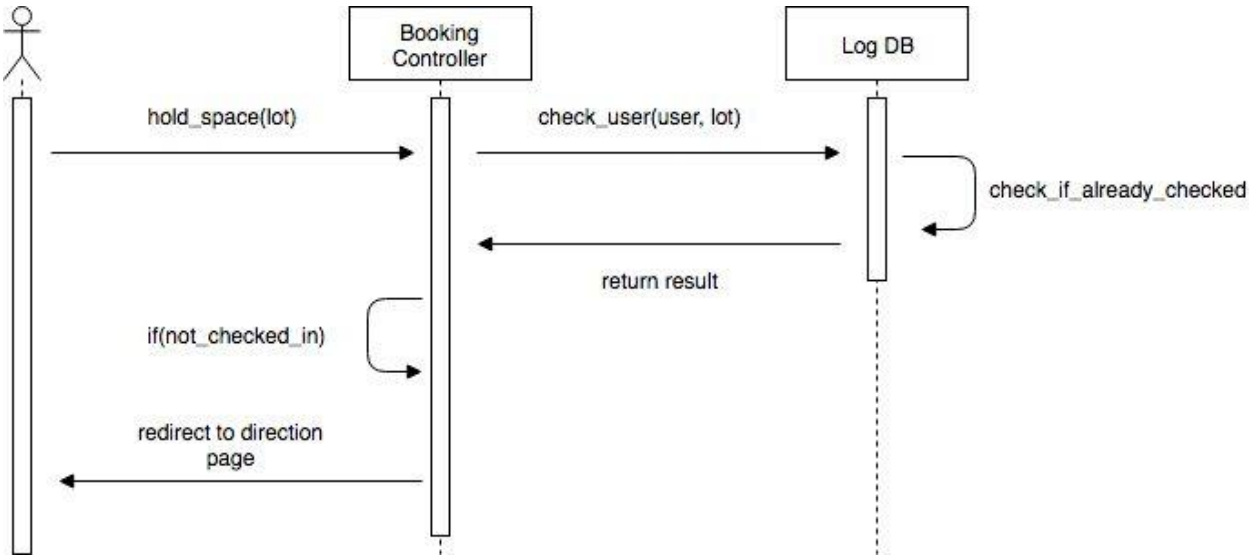

Figure 3.7: Booking parking space

Chapter 4:
System Design Specification

4.1 Development tools and technology

Without using tools, development of software is impossible. There are many tools that I have used to develop this software.

4.1.1 User interface technology

User interface (UI) is everything designed into a system view that which person's associates with this system may like the interface of this system.

4.1.1.1 jQuery

jQuery is a JavaScript library. jQuery greatly simplifies JavaScript programming. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery is the perfect choice. jQuery is built for designers and developers alike. We've designed all of our plugins to get you up and running quickly while being flexible enough to evolve with your needs and solve a plethora of use cases.

4.1.1.2 Bootstrap (CSS Framework)

CSS is a language that describes the style of an HTML document. CSS describes how HTML elements should be displayed. Build responsive, mobile-first projects on the web with the world's most popular front-end component library. Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mix INS, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.
After adding some classes to existing elements in the HTML-code and altering some CSS code such as removing some values for width given in pixels the site was changing depending on the width of the window. The bootstrap code is included minified, which means that white spaces are removed to make the file size smaller and therefore make the load time faster for the file which improves the load time for the whole page.
The main design that bootstraps ads without specifically adding design to elements is that when hovering over a link. This is fixed with some simple CSS-code added to the CSS-file, unless the bootstrap CSS-file is included after the original, then bootstrap will override the custom ones and the changes will not be seen. Having some basic knowledge about how Bootstrap works before starting to use it would increase the efficiency and speed one might achieve the goal one has in mind for including bootstrap into the project.

4.1.1.3 ASP.NET MVC 4

For developing this system, I have use ASP.NET Framework and C# as my programming language. The ASP.NET MVC is a web application framework developed by Microsoft, which implements the model-view-controller (MVC) pattern. It is open-source software, apart from the ASP.NET Web Forms component which is property.

4.1.1.4 Arduino

For the hardware implementation, I have used Arduino as my development board. Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world. Arduino uses C and C++ to program.

4.1.2 Implementation tools and platform

The order of execution may vary depending upon the person developing the plan. Some people do better with looking at lots of tools and asking themselves "How can I use these tools to accomplish my goals and which ones do I use?" While others may look at tactics that have been tried and proven successful and determine which tactics best apply to them and their goals. And, many starts with developing a sound strategy, then determine which tactics and tools best suits their needs to accomplish their goals.

4.1.2.1 IDE

I have used Microsoft Visual Studio 2013 to develop the application part. For Hardware programming I have used Arduino IDE and CodeBlocks. Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source control systems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer). Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS.

For Hardware programming I have used Arduino IDE and CodeBlocks. The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino board. The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the "Wiring" project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

For API tests I have used Postman.

4.1.2.2 Web Server

Internet Information Services (IIS, formerly Internet Information Server) is an extensible web server created by Microsoft for use with the Windows NT family.

IIS supports HTTP, HTTP/2, HTTPS, FTP, FTPS, SMTP and NNTP. It has been an integral part of the Windows NT family since Windows NT 4.0, though it may be absent from some editions (e.g. Windows XP Home edition), and is not active by default. The first Microsoft web server was a research project at the European Microsoft Windows NT Academic Centre (EMWAC), part of the University of Edinburgh in Scotland, and was distributed as freeware. However, since the EMWAC server was unable to handle the volume of traffic going to Microsoft.com, Microsoft was forced to develop its own web server, IIS.

4.1.2.3 Database Server

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

4.2 Database Design

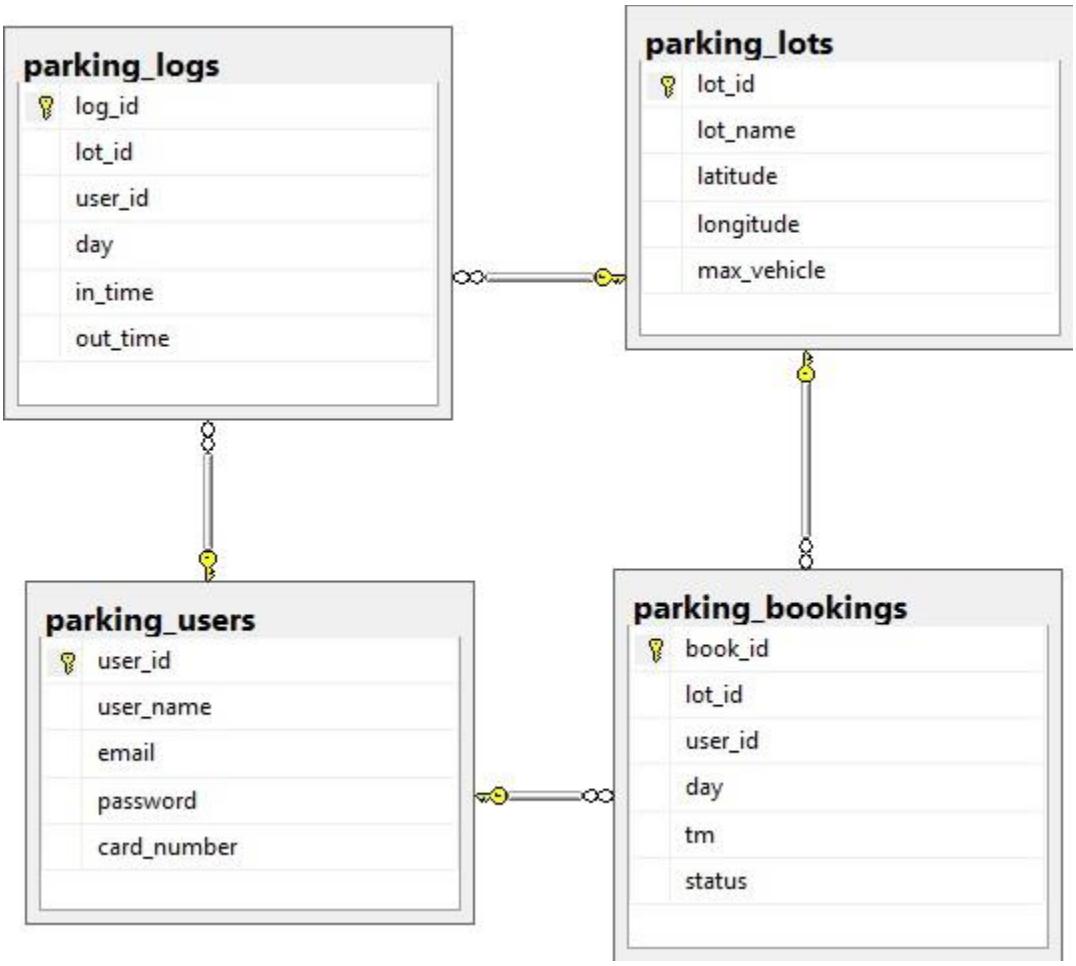

Figure 4.1: Database design

4.3 Class Diagram

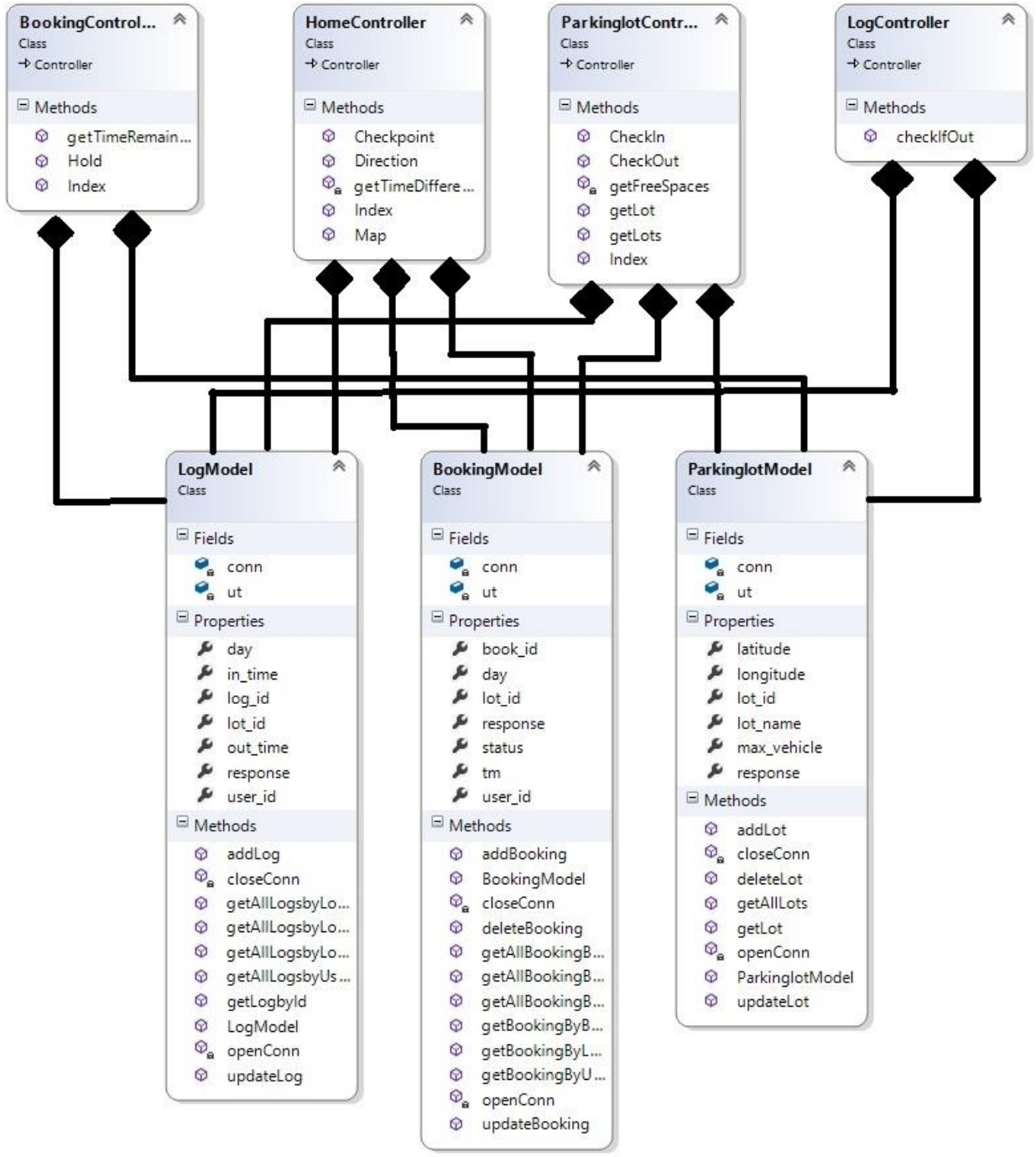

Figure 4.2: Class diagram

4.4 Device Design Diagram

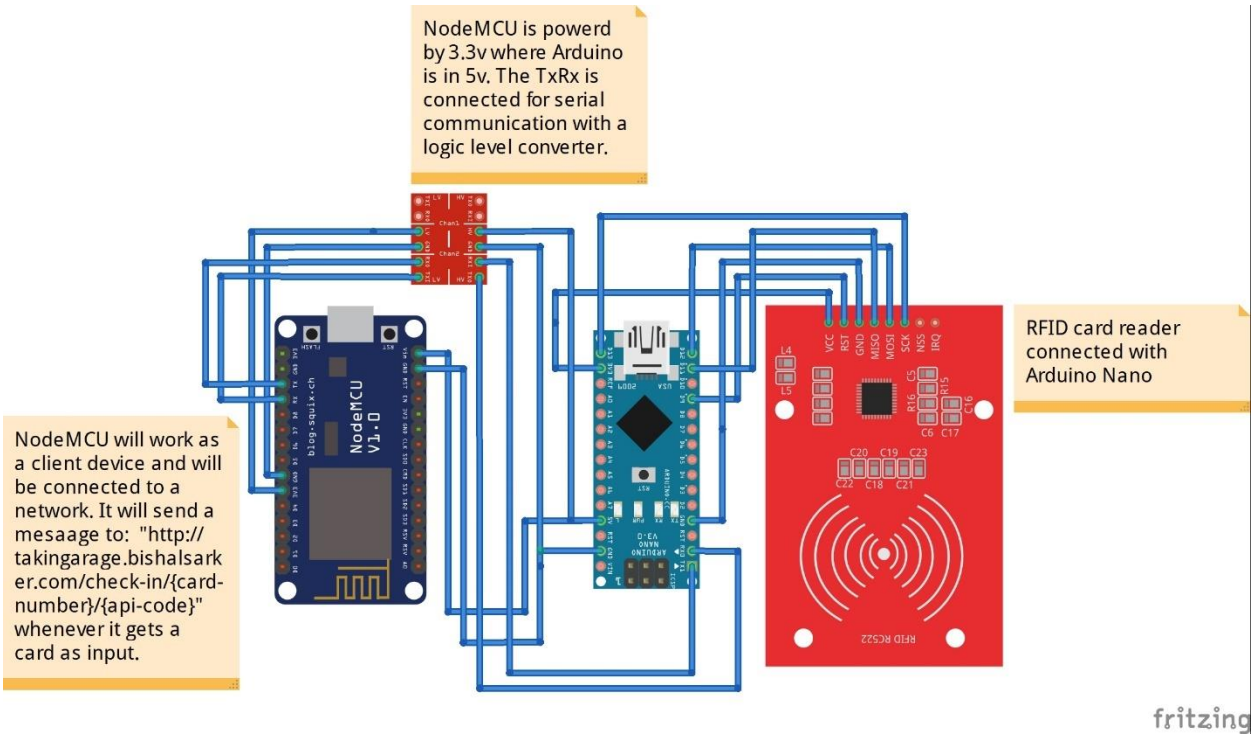

Figure 4.3: Device design diagram

Chapter 5:
System Testing

5.1 Testing Features

Feature testing is the process of making changes in software system to add one or more new features or to make modifications in the already existing features. Each of these features is said to have a characteristic that is designed to be useful, intuitive, and effective.

5.1.1 Features to be tested

Table 5.1: Features to be tested

Features	Priority	Description
Registration	2	To use this application, one has to register himself/herself.
Log In	2	Login as authenticated user.
Log Out	2	Logout from the system.
Find parking space	3	User finds nearest parking lots.
Book parking space	3	User book a space from the lot.
Check-in	3	After reaching to the parking lot user taps the RFID card on the CheckIn device to check in.
Check-out	3	User taps RFID to check out.
Billings and Payments	3	User will find all his booking and check in history with a monthly bill.
Database	2	Access to database is frequently needed operation. So, this technical feature should be tightly in control for management system

Here,

Low Priority = 1

Medium Priority = 2

High Priority = 3

This is on the basis of how the feature is impacting on the system. If any of the features failed what effect will cost the system.

5.2 Testing Strategies

A testing strategy is a general approach to the testing process rather than a method of devising particular system or component tests. Different testing strategies may be adopted depending on the type of system to be tested and the development process used.

5.2.1 Test Approach

A test approach is the test strategy implementation of a project, defines how testing would be carried out. Test approach has two techniques:

Proactive - An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
Reactive - An approach in which the testing is not started until after design and coding are completed.

5.2.1.1 Equivalence class partitioning

In considering the inputs for our equivalence testing, the following types will be used:
Legal Input values: Test values within boundaries of the specification equivalence classes. This will be input data the program expects and is programmed to transform into usable values.
Illegal Input Values: Test equivalence classes outside the boundaries of the specification. This will be input data the program may be presented, but that will not produce any meaningful output.

5.2.1.2 Boundary value analysis

The acceptable range of values for this application was set by the development team. At the time of testing developer will define the boundary value & generate test case for performing the boundary value analysis.

5.2.1.3 White box testing

White box testing is a software testing method in which the internal structure /implementation of the item being tested is known to the tester.

The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming knowhow and the implementation knowledge is essential.

5.2.1.4 Pass/Fail Criteria

The entrance criteria for each phase of testing must be met before the next phase can commence. Now the criteria for pass and fail are given below.

- According to the given scenario the expected result needs to take place then the scenario will be considered as pass otherwise that criteria should be failed.
- If an item tested 10 times, 9 times perfectly worked and single time do not work properly then it will consider as fail case.
- System crash will be considered as fail case.
- After submitting a query in the system, if expected page won't appear then it will be considered as fail case.

5.3 Testing Schedule

Table 5.2: Testing schedule

Test Phase	**Time**
Test Plan Creation	1 week
Test specification creation	2 weeks
Unit Testing	Developing time
Component testing	1 week
Test Phase	**Time**
Integration Testing	1 week
Use case validation	1 week
User interface testing	1 week
Load testing	1 week
Performance Testing	1 week
Release to Production	1 week

5.4 Test Environment

Testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured. For test environment, key area to set up includes

- System and applications
- Test data
- Database server
- Front end running environment
- Client operating system
- Browser
- Hardware includes Server Operating system
- Network
- Documentation required like reference documents/configuration guides/installation guides/ user manuals

5.5 Test Cases

It is impossible to build a system without any fault. Sometimes, this fault makes software implementation failure. If we test the system before executing the system it will help us to find the fault of the system. For testing the system, we need to write some test cases.

5.5.1 Log In

Table 5.3: Log in

Test case #1	**Test case name:** Log In
System: Talking Garage	**Subsystem:** User data
Designed By: Bishal	**Designed Date:** 20.11.18
Executed by:	**Executed date**
Short Description: The user is registered and trying to log in to the Talking Garage website when the system will check validity	
Pre-conditions: **1.** When any users try to go home page or any page, they will be asked to login first. **2.** Assume that Username is 'admin' and password 'password'	

Step	**User name**	**Password**	**Expected Response**	**Pass/ Fail**	**Comment**
1	Shuvo	124	Wrong username and password	Pass	Working correctly
2		admin	Username can't be blank	Pass	Working correctly

3	admin	admin	Invalid password	Pass	Working correctly
4	Password	admin	Invalid username	Pass	Working correctly
5	admin		Password can't be blank	Pass	Working correctly
6			Username and password can't be blank	Pass	Working correctly
7	b@gmail.com	password	Invalid username	Pass	Working correctly
8	shuvo@gmail.com	password	Invalid username.	Pass	Working correctly
9	…..	Sldjf	Invalid username & password	Pass	Working correctly
10	Admin	Password	Redirecting to maps	Pass	Working correctly
11	Khan	khan@fasdff	Invalid username and password	Pass	Working correctly
Post conditions: User will successfully log in in the system					

5.5.2 Find Parking Space

Table 5.4: Find parking space

Test case #2	**Test case name:** Find Parking Space
System: Talking Garage	**Subsystem:** N/A
Designed By: Bishal	**Designed Date:** 20.11.18
Executed by:	**Executed date**
Short Description: User has logged into the system and searching for an available parking lot to park his car.	
Pre-conditions: • User must have to be logged in • User has to open device's location service • User can't see this page if he has booked a space or checked in.	

Step	Action	Expected Response	Pass/ Fail	Comment
1	Access maps page without log in.	Login page will appear.	Pass	Working correctly
2	Access maps page without starting location service	No data will appear.	Pass	Working correctly
Post conditions: User will successfully find parking lots				

5.5.3 Book Parking Space

Table 5.5: Book parking space

Test case #3	**Test case name:** Book Parking Space
System: Talking Garage	**Subsystem:** N/A
Designed By: Bishal	**Designed Date:** 20.11.18
Executed by:	**Executed date**
Short Description: After a suitable space is been found user has to click "Hold space" option to reserve a space then a 30 minutes countdown timer will start.	
Pre-conditions: • User must have to be logged in • User has to open device's location service • User can't see this page if he has checked in.	

Step	Action	Expected Response	Pass/ Fail	Comment
1	Access booking page without log in.	Login page will appear.	Pass	Working correctly
2	Access booking without starting location service	Direction Information will appear.	Pass	Working correctly
3	Access maps while on booking page	Page redirects to bookings	Pass	Working correctly
4	Timer ends time	Page redirects to maps	Pass	Working correctly

Post conditions: User will successfully book a space in the parking lot

5.5.4 Check-in

Table 5.6: Check-in

Test case #4	**Test case name:** Check-in
System: Talking Garage	**Subsystem:** N/A
Designed By: Bishal	**Designed Date:** 20.11.18
Executed by:	**Executed date**
Short Description: User books a space, system will give user 30 minutes to reach in the lot. If user fails to tap the RFID card in the device installed in the lot, reservation will be cancelled.	
Pre-conditions: • User must have to be logged in • User has to open device's location service • User can't see this page if he has not booked a space.	

Step	**Action**	**Expected Response**	**Pass/ Fail**	**Comment**
1	Tap RFID before time is finished	Successfully checked in and start counting time of stay.	Pass	Working correctly
2	Tap RFID after time is finished	Maps page will appear	Pass	Working correctly
Post conditions: User will successfully be checked in the parking lot				

5.5.5 Check-out

Table 5.7: Check-out

Test case #5	**Test case name:** Check-out
System: Talking Garage	**Subsystem:** N/A
Designed By: Bishal	**Designed Date:** 20.11.18
Executed by:	**Executed date:**
Short Description: If user wants to leave, he will tap the card to check out. After successful checkout the time elapsed page will refresh and get to maps.	
Pre-conditions: • User must have to be logged in • User has to open device's location service • User has to tap card to checkout.	

Step	**Action**	**Expected Response**	**Pass/ Fail**	**Comment**
1	Tap RFID card	Successfully checked out and back to maps page	Pass	Working correctly
Post conditions: User will successfully be checked out from the parking lot				

Chapter 6:
User Manual

6.1 The Maps

After login, the first page user will see is the Maps page. This is too easy to understand. I have used Mapbox – an open source mapping api. All the parking spaces will be marked with a blue marker and the user location with black dot. To zoom in or out you will get a "plus" and a "minus" button. Also, a direction button to change the direction. You can also use mouse wheel for zooming.

If you are not viewing the black dot please check that the "User Location Service" is turned on and allowed for TalkingGarage.

On the right-side bar in PC and bottom of the page in Mobile, you will find the "Location-based search results" and "My account" options for billing.

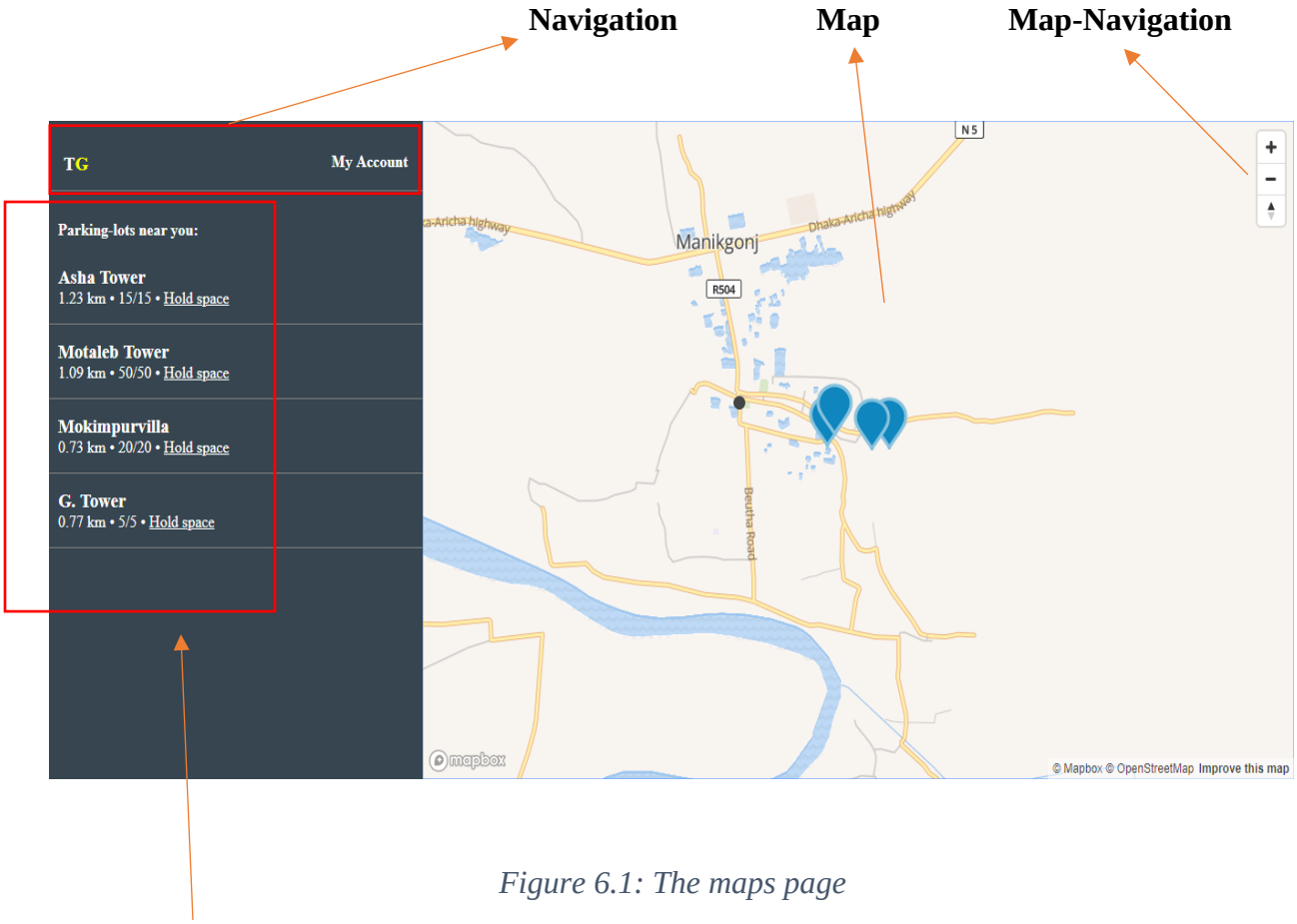

Figure 6.1: The maps page

6.2 Searching for a parking space

It is too easy to search for a parking space. Searching is not manual in TG like other sites. You will find available parking spaces near your 1.5 km. So, while you move, TG will move along with you and find if there is an available parking space near you. Make sure the “Location” service is turned on your device. Otherwise, the system will be unable to calculate.

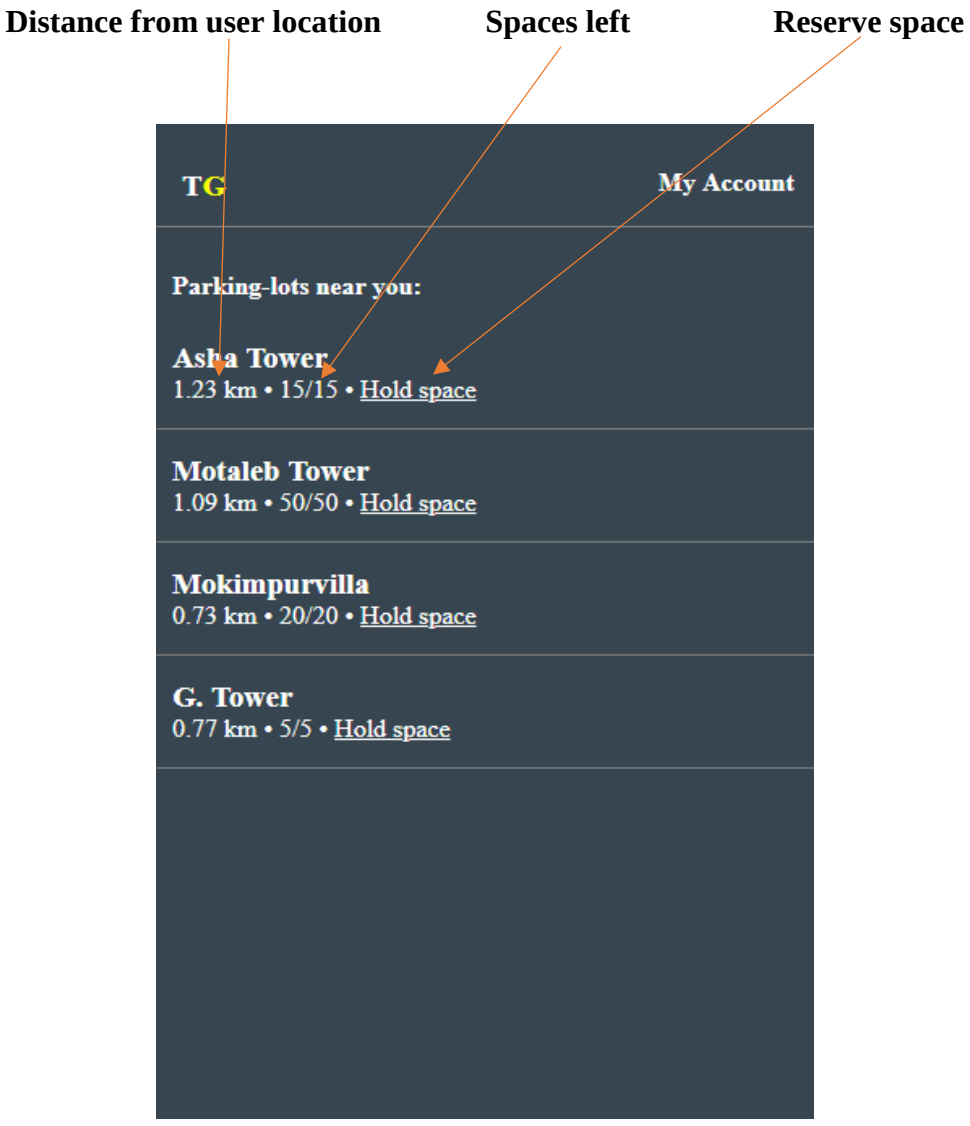

Figure 6.2: Searching parking spaces

6.3 Booking a parking space

To book a parking space user has to click “Hold space” option after he/she has got a suitable parking lot. After that, user will be redirected to the Direction page. User will get the space reserved for 30 minutes. If user is unable to reach to the lot and check in the user will be redirected to the Maps page.

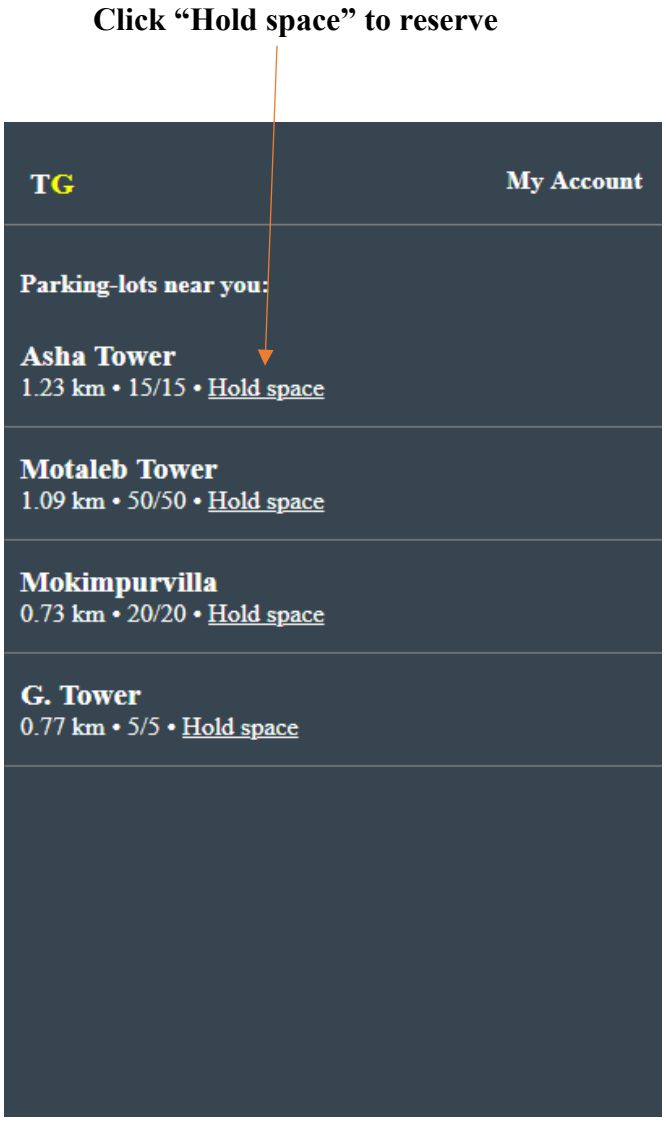

Figure 6.3: Booking a space

6.4 Check in to a parking space

After following the instruction 6.3, user will have to reach to his/her desired lot. There user will find a device to check in. User will have to tap his/her RFID card in that device. The device will send user info to the server to check. If the credentials match, user will be checked in and in the application he/she will redirected from the Direction page to Checkpoint page.

Figure 6.4: Tapping RFID to the device

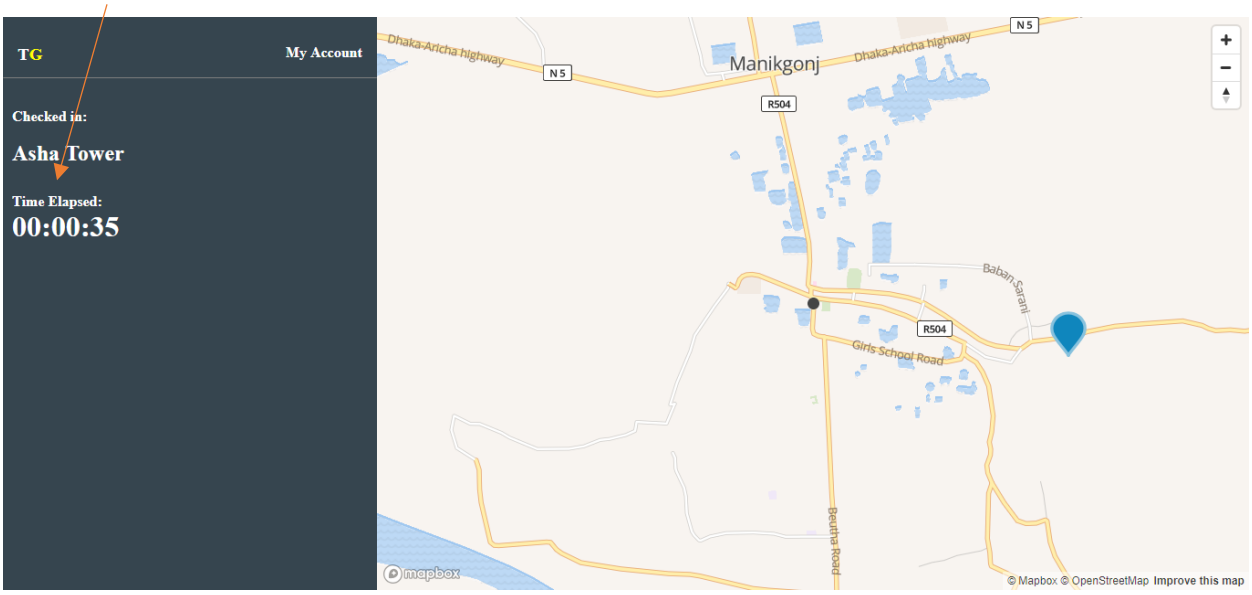

Figure 6.5: Checkpoint page

6.5 Check out from a parking space

If user wants to leave, he/she has to tap the card on the other device. In the application user will be redirected from Checkpoint page to the Maps page.

6.6 Billings and payments

For billing and payments, user has to go to “My Account” option.

Chapter 7:
Project Summary

7.1 Conclusion

This project has been started from June. From that beginning, I have to work hard to know the user requirements clearly. After that I proposed a design to them by help of my supervisor. They appreciated and said to start developing the project. Then I started to develop the project. From then I gradually develop the project. To build a real-time system with device integration, I think storing the data in database neatly is very important. That's why I did this first and made a relationship with the tables. After that I designed the UI. This project's UI is very simple and clean which is very helpful for the user's experience. Then I started coding and executing the project. If I did not test this project there will stay some bug on this project which will ruin the full project. That why give importance to test this project and then I solved some bug which I got after testing this project.

7.2 Limitations

It is very hard to develop something without any limitations. This project has some limitations too. Limitations are: -

- Lack of the supply of electricity and internet connection to the parking lot.
- Device requests are not encrypted.
- Not fully responsive for all the devices.
- Maps needs a good amount of internet speed.

7.3 Obstacles and Achievements

To walk in the good way, one has to face many obstacles. By facing obstacles, one will get some achievements. To store the real-time data from the devices and to get the data in a correct format so that it can be processed was an obstacle for me. Although I have done it by taking help from my supervisor, friends and by searching the solution from google. Some obstacles and achievement are as:

- **Scope Change:** Sometimes I was asked to add some features. Then I had to redesign the system. It made me sometimes hopeless.

- **Resource Deprivation:** In some cases, I did not get proper resource to handle that situation.

- **Lack of Stakeholder's Engagement:** Making people get interested with this new kind of feature and to manage the garage owners with good amount of profit was too hard to get.

7.4 Future Scope

By working with this project, I have learnt many things and meet with some great person. This project will give me some opportunity to work with this type of similar project.

REFERENCES

1. Freeman, A. (2012). *Pro ASP.NET MVC 5 (5th ed.)*. Apress.

2. SM (2018). *Getting started with Arduino Mini*. Received from https://www.arduino.cc/en/Guide/ArduinoMini

3. Wikipedia. *Radio-frequency identification*. Received from https://en.wikipedia.org/wiki/Radio-frequency_identification

4. Rui Santos. *Security Access using MFRC522 with Arduino*. Received from https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/

5. Ravi Singh Jaiswar (2017). *Spiral Model of SDLC*. Received from https://www.utest.com/articles/spiral-model-of-sdlc

6. PiLabsBD (2018). *Smart Car Parking System*. Received from https://www.pilabsbd.com/portfolio-items/car_parking/

7. Abhirup Khanna (2016). *IoT based Smart Parking System*. Received from https://www.researchgate.net/publication/303842610_IoT_based_Smart_Parking_System

8. W3Schools. *HTML5 Introduction*. Received from https://www.w3schools.com/html/html5_intro.asp

9. Julien Dubois (2010). *Introduction to jQuery*. Received from https://www.oracle.com/technetwork/java/intro-jquery-166845.html

10. Mapbox. *How Mapbox Works*. Received from https://www.mapbox.com/help/how-mapbox-works-overview/