PARKING LOT MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

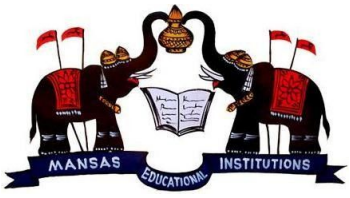
In

COMPUTER SCIENCE AND ENGINEERING

BY

P.V.S.VIKAS
(Roll No: 17331A05C2)
R.V.PAVAN KUMAR
(Roll No: 17331A05E0)

Under the Supervision of
Mr. G.SUVARNA KUMAR SIR
Associate Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MVGR COLLEGE OF ENGINEERING (Autonomous)
VIZIANAGARAM-535005, AP (INDIA)
(Accredited by NBA, NAAC, and Permanently Affiliated to Jawaharlal Nehru Technological University Kakinada)

PARKING LOT MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

Maharaj Vijayaram Gajapathi Raj (MVGR) College of Engineering (Autonomous) Vizianagaram

CERTIFICATE

This is to certify that the project report entitled "**parking lot management system**" being submitted by **P.V.S.VIKAS** ,**R.V.PAVANKUMAR** bearing registered numbers **17331A05C2, 17331A05E0,** respectively, in partial fulfillment for the award of the degree of "**Bachelor of Technology" in Computer Science and Engineering** is a record of bonafide work done by them under my supervision during the academic year 2020-2021.

Dr. C. KALYANA CHAKRAVARTHY, Head of the Department,
Department of CSE,
MVGR College of Engineering, Vizianagaram.

Mr.G.Suvarna kumar ,
Associate professor,
Department of CSE,
MVGR College of Engineering,
Vizianagaram.

External Examiner

ACKNOWLEDGEMENTS

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

We place on record our heartfelt appreciation and gratitude to **Mr.G.suvarna kumar sir** for the immense cooperation and navigation as mentor in bringing out the project work under his guidance. His uncompromising attitude to bring out the best with constructive suggestions inspired us to achieve the results set forth for the project work. We are deeply indebted to him for his excellent, enlightened and enriched guidance.

We also thank **Dr. K. V. L. Raju**, Principal, and **Dr. C. Kalyana Chakravarthy**, Head of the Department, for extending their utmost support and co-operation in providing all the provisions for the successful completion of the project.

We sincerely thank all the members of the staff in the Department of Computer Science & Engineering for their sustained help in our pursuits. We thank all those who contributed directly or indirectly in successfully carrying out this work.

p.v.s.vikas

R.v.pavan kumar

INDEX:

CHAPTER-1

PARKING LOT MANAGEMENT SYSTEM

PROBLEM STATEMENT:

Develop a system for management of multi store parking lot

DESCRIPTION:

A parking lot is a dedicated cleared area that is intended for parking vehicles in most countries where cars are a major mode of transportation,parking lots is a feature of every city and suburban area.

INTRODUCTION:

1.1 Project Objective:

Main Objectives:

To enable drivers to locate and reserve a parking place online through accessing it on web platform

Specific Objectives:

To establish possible solutions to improve on the current Vehicle Parking Reservation system

To design and implement Online Vehicle Parking Reservation system

To make a good research about People's Park and gather all necessary information that helped in designing the new parking reservation system

1.2 Stakeholders

Admin:

mainly responsible for adding /removing floors,parking spots,exit panels ,parking attendant.

Customers:

Customers can book slots by paying for them.

Parking attendant:

Responsible for collecting money and issuing tickets.

System

To display messages on different info panels as well as assigning and removing a vehicle.

1.3 Processes:

Customer:

-> Take ticket
-> Scan ticket
-> Payment

Admin:

-> add/create parking floor
-> add parking spot
-> add/modify parking rate
-> add/removing parking attendant

System:

-> Assign a parking spot to a vehicle
-> removing a vehicle from a parking spot
->showing parking status

Parking Attendant:

-> view account
-> update account
->cash payment

1.4 Issues in Application Development:

As the money flow increases, and more and more individuals and companies expand their ownership of vehicles, the complexities and conflicts of parking swells. While cooperation and coordination are crucial for a smooth functioning of the parking lot areas, there are a number of things that the parking lot owners could do to speed up their processes. Parking management solutions address parking problems and ***fully automate parking operations***.

In the cities, the supply-demand ratio leads to some deep parking issues for the parking area providers. From tracking, processing, checking to management everything seems to be messed up.

Manual Checks: Parking managers perform manually intensive work of counting permit and non-permit cars. There is manual checking of vehicle status and details and handwritten tickets. Such a manual procedure leads to 5% entry errors, further resulting in huge losses to the bottom line.

Paper Records: It is difficult to sieve through the large volumes of information. For accomplishing this task, the parking lot managers have to spend hours searching files for the exact information. So, these paper records create a lot of problems.

High Labor Costs: Reading, writing and entering data is labor-intensive and time consuming. Unnecessary capital expenditure is increased due to the money spent on labor that performs repetitive manual tasks.

Waiting Customers: The outdated mode of management troubles the customers and makes them wait in long queues when they need to enter and exit the parking lot. Due to this, precious time of the clients is wasted, and their sustainability gets shaken.

Unauthorized Access: The parking manager in-charge issue handwritten paper tickets that can be duplicated easily. No security alerts are raised to the authorized personnel if any unauthorized vehicle enters the parking lot.

Now, let us evaluate how Parking space management solutions can help to eradicate these issues and simplify your parking business like never before:

Automated System: Parking lot management system is an automated system that eliminates data errors made by the manpower. RFID speed up the process for checking automobiles with permit parking. Moreover, the automobiles with authorized RFID tags are allowed automatic entry. This enhances the effective use of space and resources.

Cloud-based Reports: Parking lot management solution offers reports to be accessed online by the authorized personnel. The records are all well-maintained on the cloud to be accessed whenever needed. Particular records can be accessed by just pushing in the license plate or parking ticket's number or permit holder's name.

Reduce Labor Costs: ***parking management system*** **i**s fully automatic, and there's really not much to do after employing the same. It requires limited personnel to be kept for performing various parking operations. It's almost like unmanned 24-hour operation. Hence, costs with wireless RFID system are greatly reduced.

Happy Customers: The parking lot management software can identify the vehicles within microseconds. As the RFID reader can read the RFID tags from a

considerable distance, ticket jamming problems are resolved. RFID parking space management system allows faster check-in and check-out of the vehicles under controlled conditions. In this way, customers' time is saved, and they become satisfied with the service forever.

Security Alarms and Alerts: With customized parking space management, parking lot owners can forget unauthorized access of vehicles. Security alerts are raised in real-time and delivered by e-mail or text message to appropriate personnel. RFID technology in parking management solutions is the most secure medium than other networks to perform parking control operations.

CHAPTER 2

2 REQUIREMENT ANALYSIS:

-> The parking lot should have multiple floors where customers can park the cars.

-> The parking lot should have multiple entry and exit points.

-> customers can collect a parking ticket from the entry points and can pay the parking fee on their way out

->customers can pay for the tickets at the automated exit panel or at the parking attendant.

-> customers can pay via both cash and credit cards.

-> customers should als be able to pay the parking fee at the customers info panel on each floor.

->the system should not allow more vehicles than the maximum capacity of parking lot.

->if the parking is full the system should be able to show a message at the entrance panel and on the parking display board on the ground floor.

->each parking floor will have many parking spots specified for electric cars these spots should have an electric panel through which customers can pay and change their vehicles

->the system should support parking for different types of vehicles like car,truck,van,motorcycle etc.

->each floor should have a display board showing any free parking spot for each spot type.

CHAPTER 3

UML DIAGRAMS

3.1 UML Introduction

UML is a graphical modeling language way of designing computer programs and software systems.

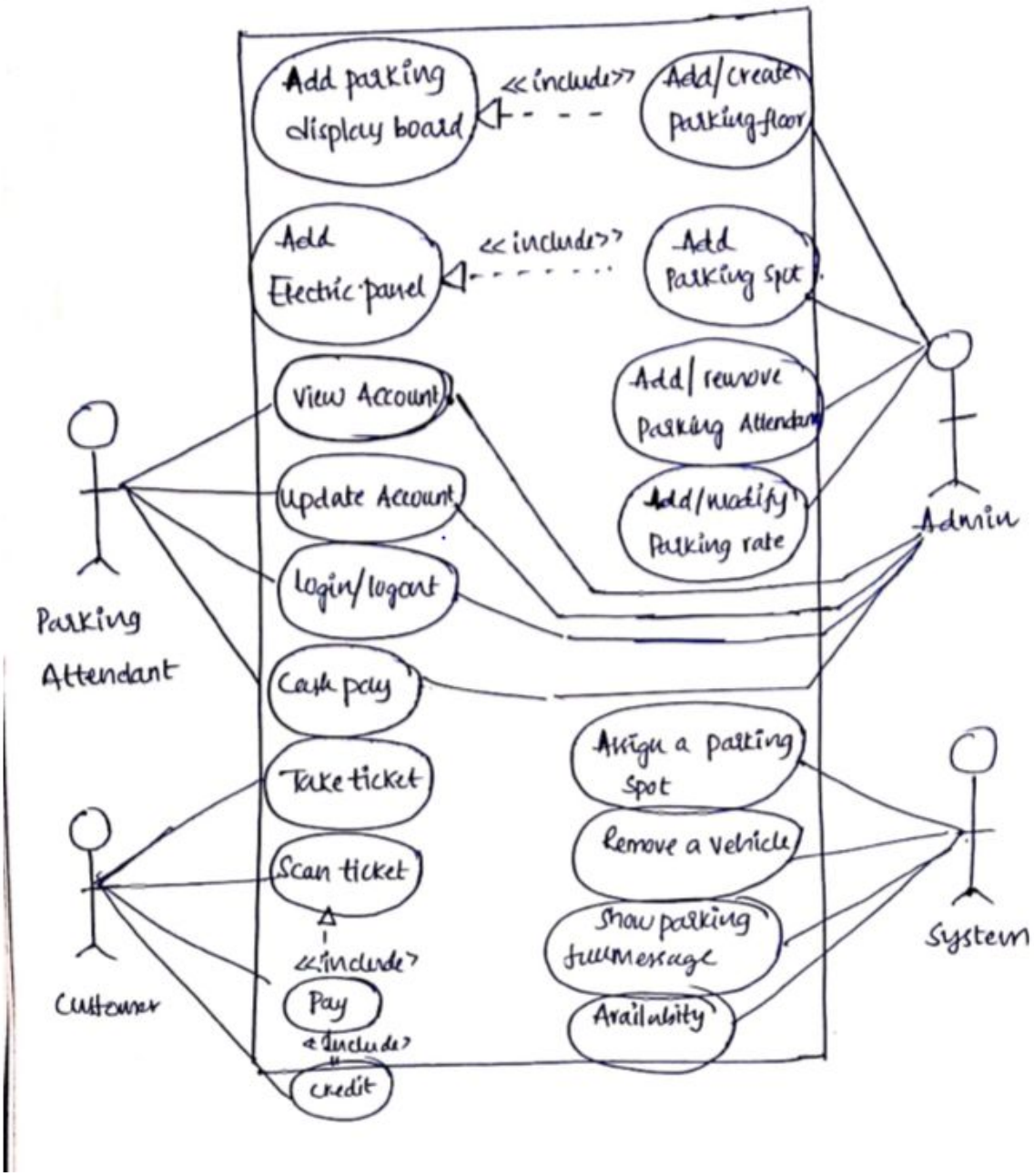
The UML standard provides for several different types of diagrams, each one representing a different way of looking at the system. For example, a class diagram shows the properties and relationships of several classes. A use case diagram shows several different types of users and codifies their needs within the system. Each diagram has a set of defined symbols which represent various concepts of entities within a computer system classes, relationships, users, interfaces, components, and so forth.

This system consists of three
actors namely:

-> User
->Admin
-> Parking Attendant

3.2 Use case

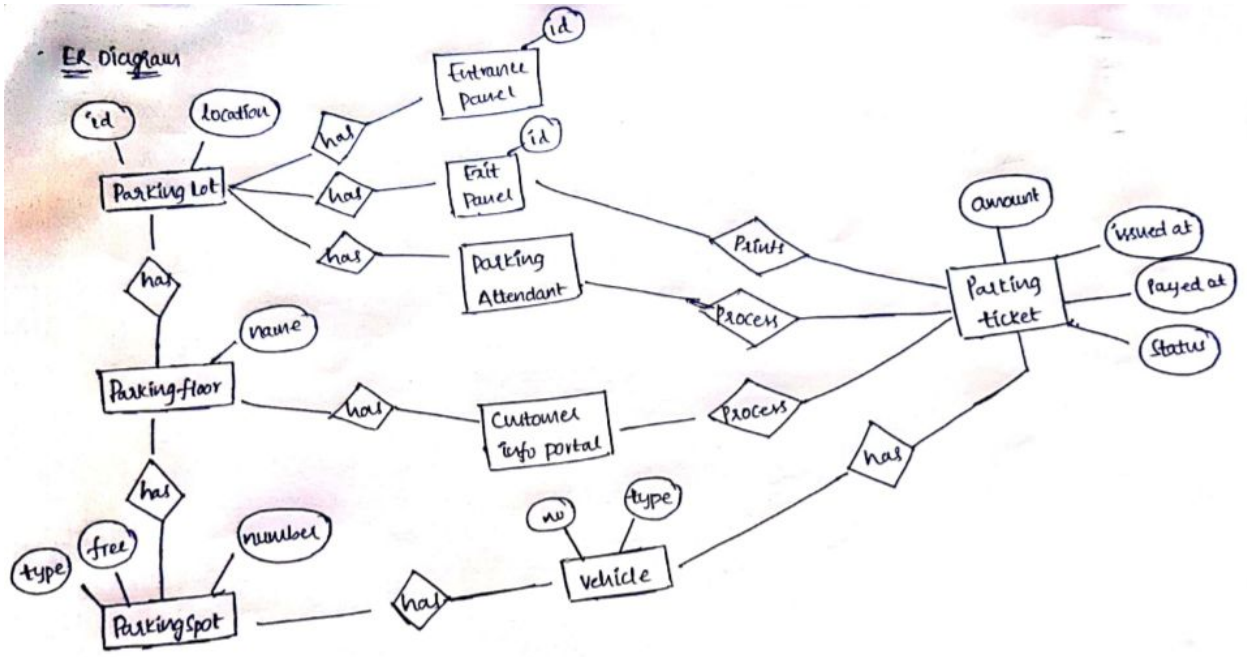
Use case is a list of actions or event steps typically defining the interactions between actor and system to achieve a goal. The actor can be human or other external system.



3.3 Entity Relationship Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. Connecting lines, solid lines that connect attributes to

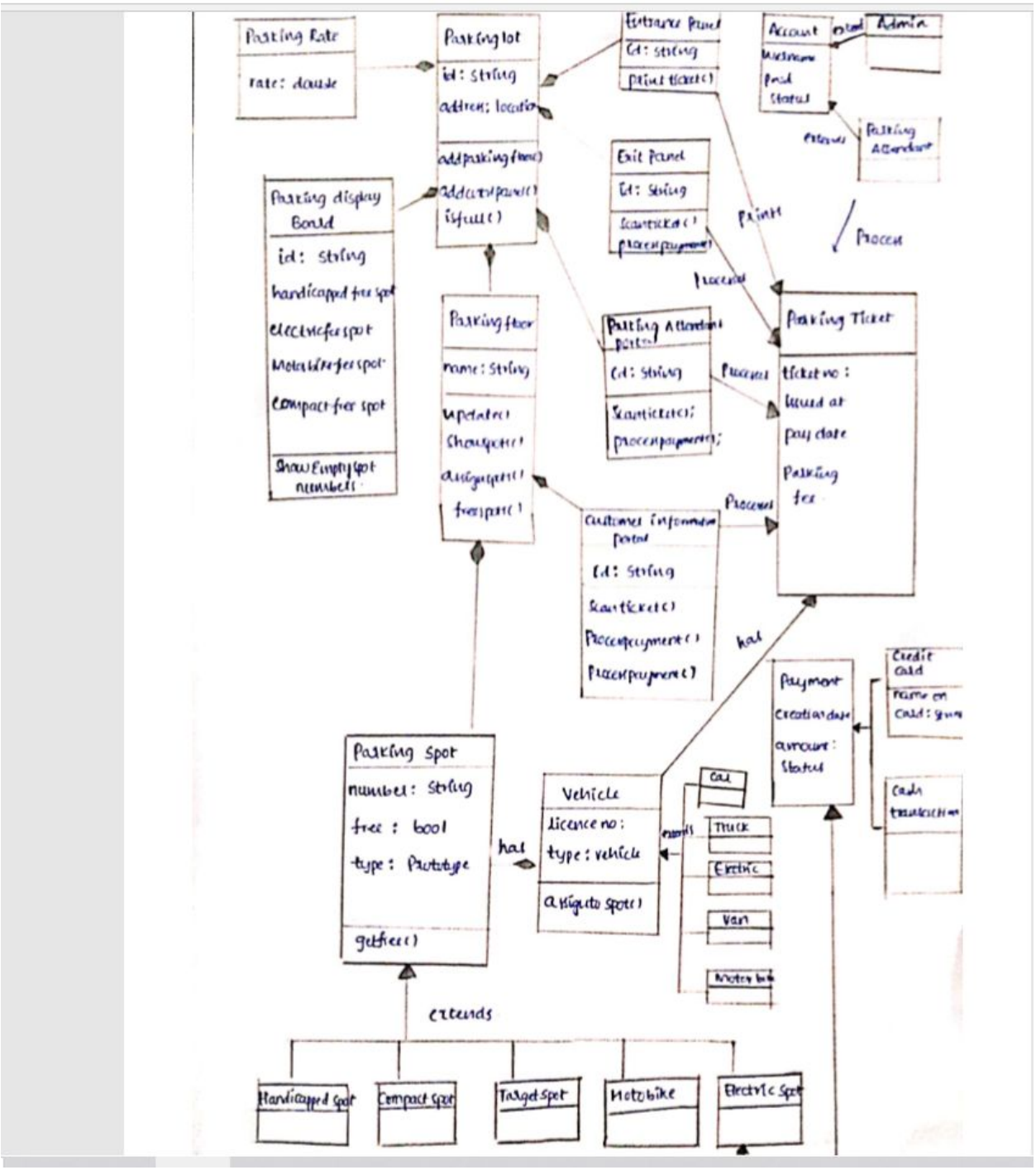
show the relationships of entities in the diagram. **Entity Relationship diagram for Hospital Queue Management System:**



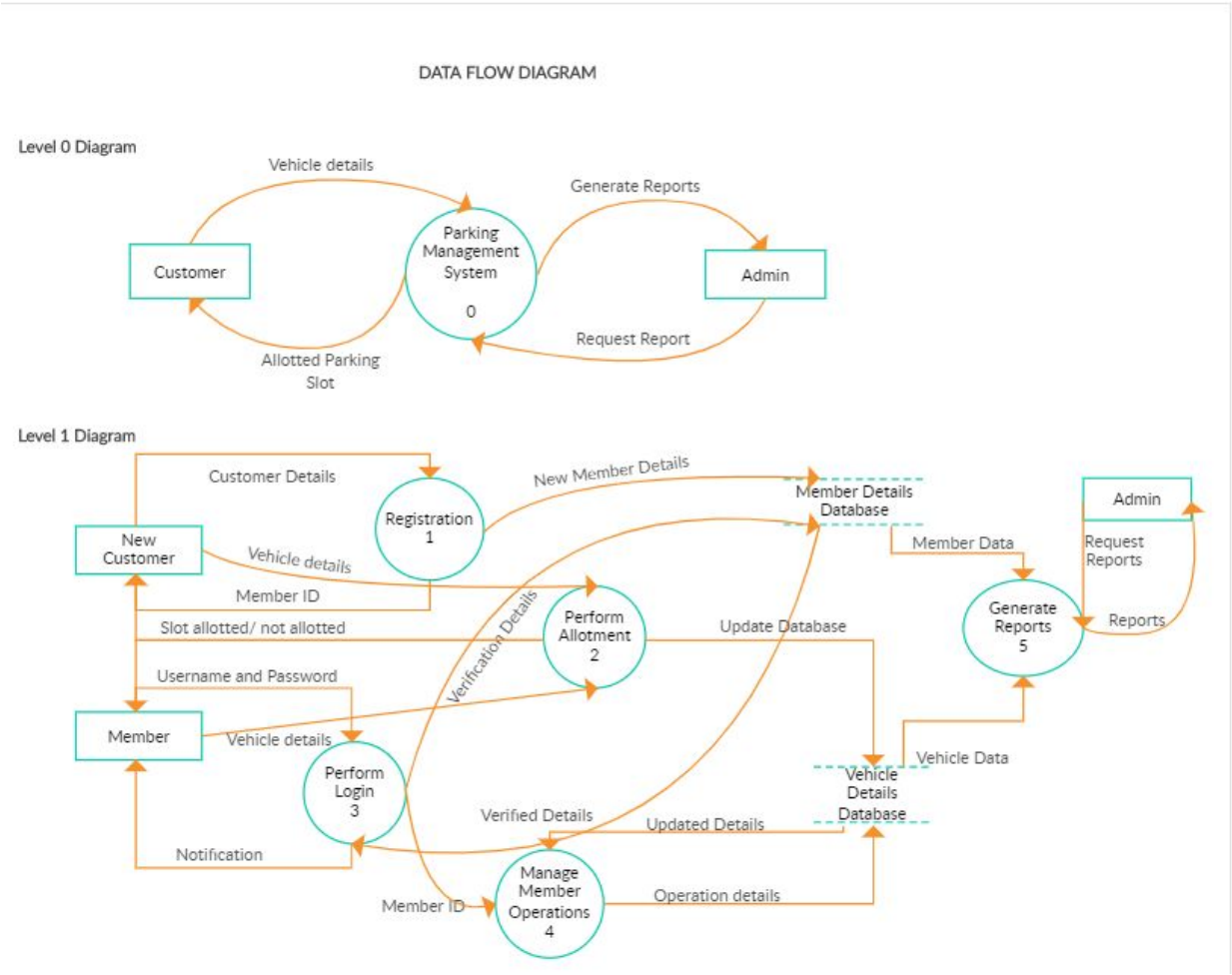
3.4 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

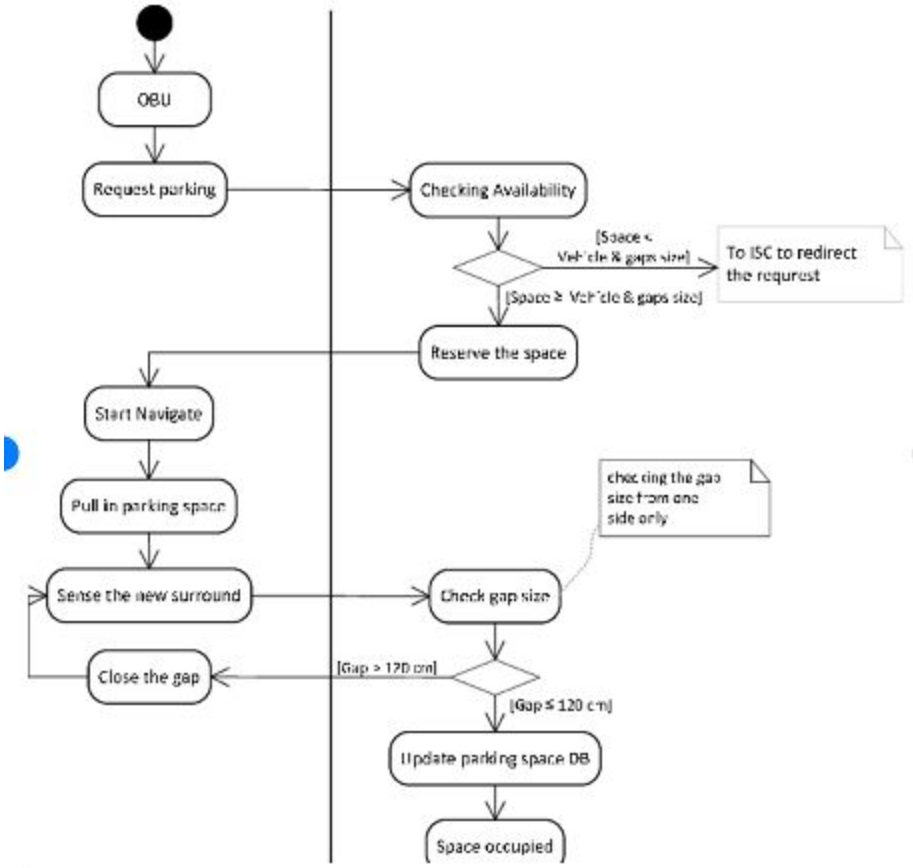
Class Diagram for Parking Lot Management System:



3.5 DATAFLOW DIAGRAM:

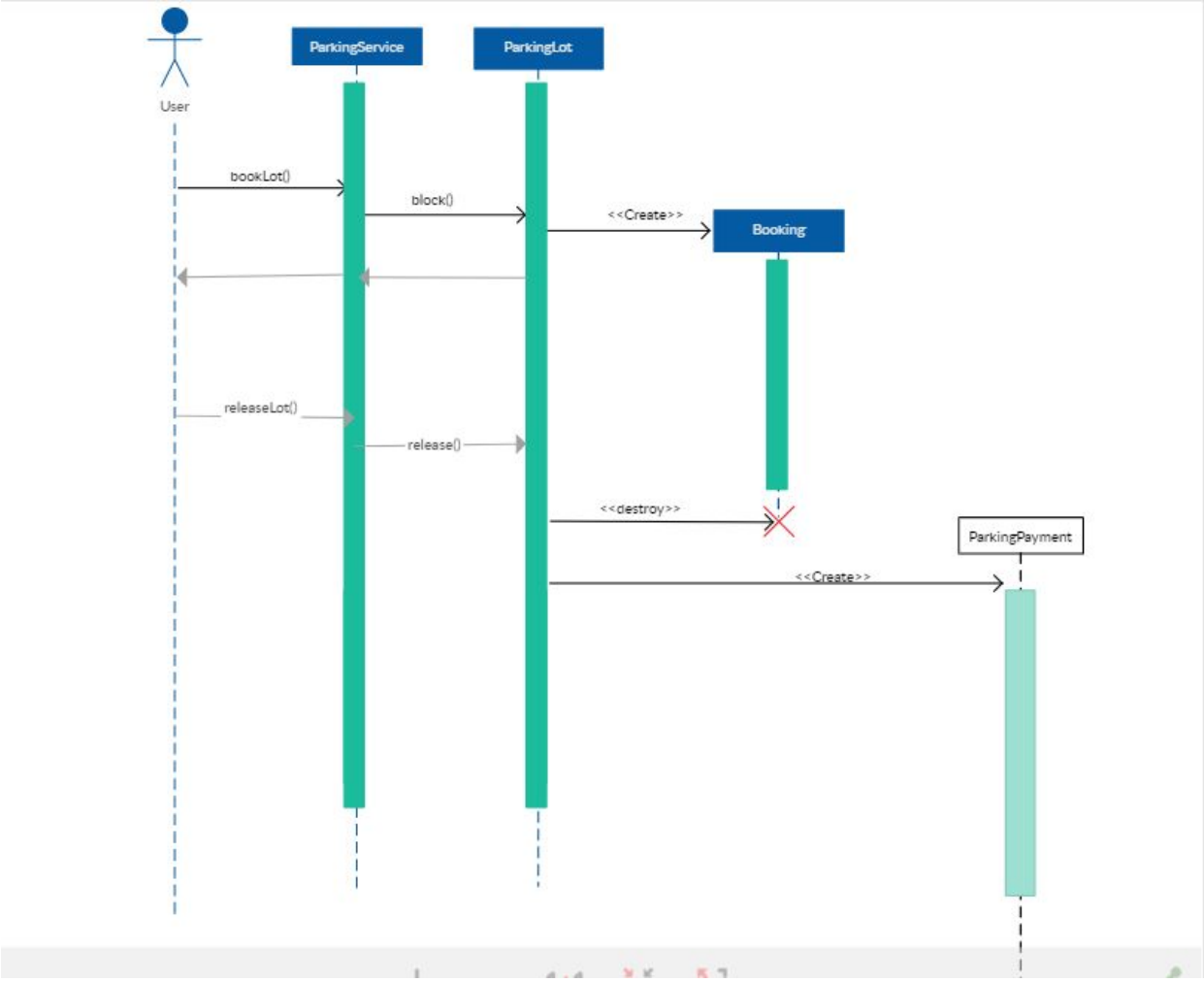


3.6 ACTIVITY DIAGRAM

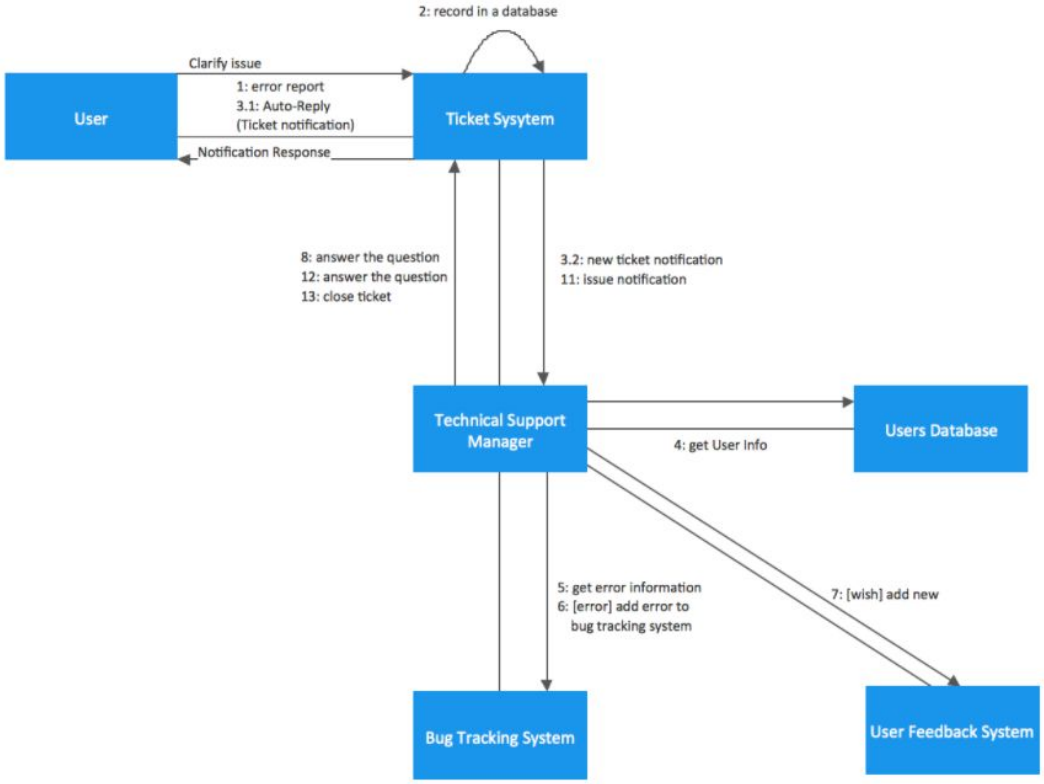


3.7 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.



Collaboration Diagram:



CHAPTER -4

SOFTWARE REQUIREMENT SPECIFICATION

4 Requirements Specifications

After analyzing the data collected, we formulated a number of requirements namely user requirement, system hardware software attribute. These were grouped as user, functional, non-functional and systems requirements.

4.1 User Requirement

The users can be able to search the products based on name. User should be able to see their order history.

4.2 Functional Requirements

There are a lot of software requirements specifications included in the functional requirements , which contains various process, namely Registration, product's search, Division of categories, orders, view the order history, view profile of user, update profile of the user.

1. Registration Process:

Adding bookings: it enables the admin to include new bookings to the system.

Assigning an Id to the products: it assigns an id to the bookings

2. Cancellation:

User can be able to cancel the bookings if he/she wants to cancel..

3. View the order history:

User should be able to see a message like booking successful, after he had order.

User can be able to view their booking history.

4. View user information:

User once registered their details will be saved in the database. User can view their profile that includes their first and last name, phone number, Email, etc.

5. Update user information:

User can be able to update their profile information.

4.3 Non Functional Requirements

There are a lot of software requirements specifications included in the non-functional requirements ,which contains various process, namely Security, Performance, Maintainability and Reliability.

Security:

1. user Identification: The system needs the user to recognise her or himself using the phone
2. Logon ID: Any users who make use of the system need to hold a Login ID and password
3. Modification: Any modification like insert, delete, update, etc. for the database can be synchronized quickly and executed only by the ward administrator.
4. Administrator rights: The administrator can view as well as alter any information in System.

Performance:

5. Response Time: The system provides acknowledgement in just once the product information is checked.
6. Capacity: The system needs to support at least 1000 people at once.
7. User-Interface: The user interface acknowledges within five seconds.

Maintainability:

8. Back-Up: The system offers the efficiency for data backup.
9. Errors: The system will track every mistake as well as keep a log of it

Reliability:

10. Availability: The system is available all the time.

4.4 System Requirement

This section describes the hardware components and software requirements needed for effective and efficient running of the system

Table: 6.0 Hardware Requirements

02	Memory	2 GB RAM
03	Disk Space	500 GB

Table: 6.1 Software Requirements

SL	Software	Minimum System Requirement
01	Operating System	Windows10
02	Runtime Environment	Visual Studio Code or other python compilers

Chapter-5

• 5.UNDERLYING TECHNOLOGIES

The technologies that are used in our system are as follows:-

- IONIC
- HTML
- CSS AND SCSS
- TYPESCRIPT
- ANGULAR

5 IONIC

Ionic framework is an open source UI toolkit for building Performant, high-quality mobile and desktop apps. Using Ionic, One can build an excellent user interface. It has a routing mechanism which provides routes to navigate from one page to another page.

In Ionic, we mainly come across pages, components and services. Ionic page is an angular component. A component has a template and is used to add a completely new element to the app. Any authenticated connections come under the services.

All the pages, components, services come under the src/app folder. Along with these, app folder contains the following:

1. app-routing.module.ts
2. app.module.ts
3. app.component.html
4. app.component.ts
5. app.component.scss
6. app.component.spec

Any new generated page comes under the app-routing.module.ts file. This app-routing.module.ts file is exported and is used in the app.module.ts file. User interface can be designed in the html. User interface can be styled in .scss file. Function definition can be defined in the typescript file.

To generate a page in ionic use the following command: ionic generate page pageName

Prerequisites:

- Node.js must be installed in your system.
- A code editor for writing the code. We are using visual studio code.
- Android studio must be installed(To use the app in android mobiles)
- Xcode must be installed (To use the app in ios mobiles)

Install ionic framework using following command in your command prompt. npm install –g ionic cordova

To start your ionic application use the following

command: ionic start appName

It will ask us to choose a framework. We have selected angular.js framework. And then it will ask us to choose whether we need a blank application or tabs or side menu application.

After that you need to change the directory to your application.

You need to install ionic lab to see the mobile view. Use the following command to install ionic lab.

npm install @ionic/lab –save –dev.

Run the application using ionic serve –l.

You need to add platform to your ionic application using the following

commands: ionic capacitor add Android

ionic capacitor add ios

Build and run the application using following

commands: ionic capacitor build android

ionic capacitor build ios

ionic capacitor run android

ionic capacitor run ios

5.2 HTML

Html is a hyper text mark up language. Html describes the content of web pages. It lets you to design a user interface of your choice. Ionic offers many user interface comp

ike ion-header, ion-toolbar, ion-title, ion-content, ion-list, ion-button, ion-searchbar etc..

Component	**Usage**
ion-header	It gives a header at the top of the page.
ion-toolbar	These are positioned above or below the content.

ion-content	The main content will be displayed here.
ion-title	Here you can give the title of the page.
ion-input	Form data will be displayed using this component.
ion-button	Buttons can be displayed using this component.

Above described are the few components and there are a lot more in ionic.

5.3 CSS and SCSS

Cascading Style Sheets (**CSS**) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. Css is used to specify the layout of web pages. In ionic, Scss files are there instead of Css. But we can write the Css code in the scss file. In scss, there are some additional features when compared to Css. You can style the html components in this scss file. The ionic framework is built with Css, it comes with some default style which we can easily change and modify.

5.4 Typescript

, you will get errors.

Typescript is a superset of JavaScript and adds optional static typing to the language.
It is designed for the development of large applications and transcompiles to JavaScript.

In ionic, typescript files are auto-generated when we are creating a new page or a component. In the typescript file, we need to add a constructor to set up our new routes. We also define the functions which are used in the Html files. This typescript file lets you code and change the functionality of the system as you like. When you are using some components in the typescript file, you must import those components in your typescript file. Otherwise

5.5 Angular

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps. Angular is a blanket term that is used for all the versions of AngularJS (Angular1), i.e., Angular2, Angular4, Angular 5 and Angular 6.

It has the latest and most refined framework till date to design a web application that is dynamic and responsive. Angular's data binding and dependency injection eliminate much of the code currently you write.

CHAPTER 6

6 TESTING

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the code so the application should be thoroughly tested and validated.

6.1 Testing Activities

Testing a large system is a complex activity and like any complex activity. It has to be broke into smaller activities. Thus incremental testing was performed on the project i.e., components and subsystems of the system were tested separately before integrating them to form the subsystem for system testing.

1. **Unit Testing**

The testing of individual software components or modules is called unit testing. Stubs and drivers are two components used for the replacement of a module. In the e-commerce system system, there are many modules like login, the home page, Book Appointment, etc. Firstly we created a login page and our home page is not ready. To verify the validation of the login page you need to redirect to a page and there comes the usage of a stub. Stubs are called programs and they are used temporarily in our application. Stubs follow the top-down approach and drivers follow the bottom-up approach. Drivers are used temporarily when our main application is not ready.

2. **Integration Testing**

In unit testing, we are testing the individual modules. For a complete application, all these individual modules need to be integrated and tested. In our application, Authentication, division of timeslots, Booking an appointment, Grayed out after the limit reached to maximum

appointments are tested individually first. After testing the individual modules, we integrate them into one application and then tested.

3. Validation Testing

The validation testing is nothing but validation success when system functions in a manner that can be reasonably expected by the customer.

4.System Testing

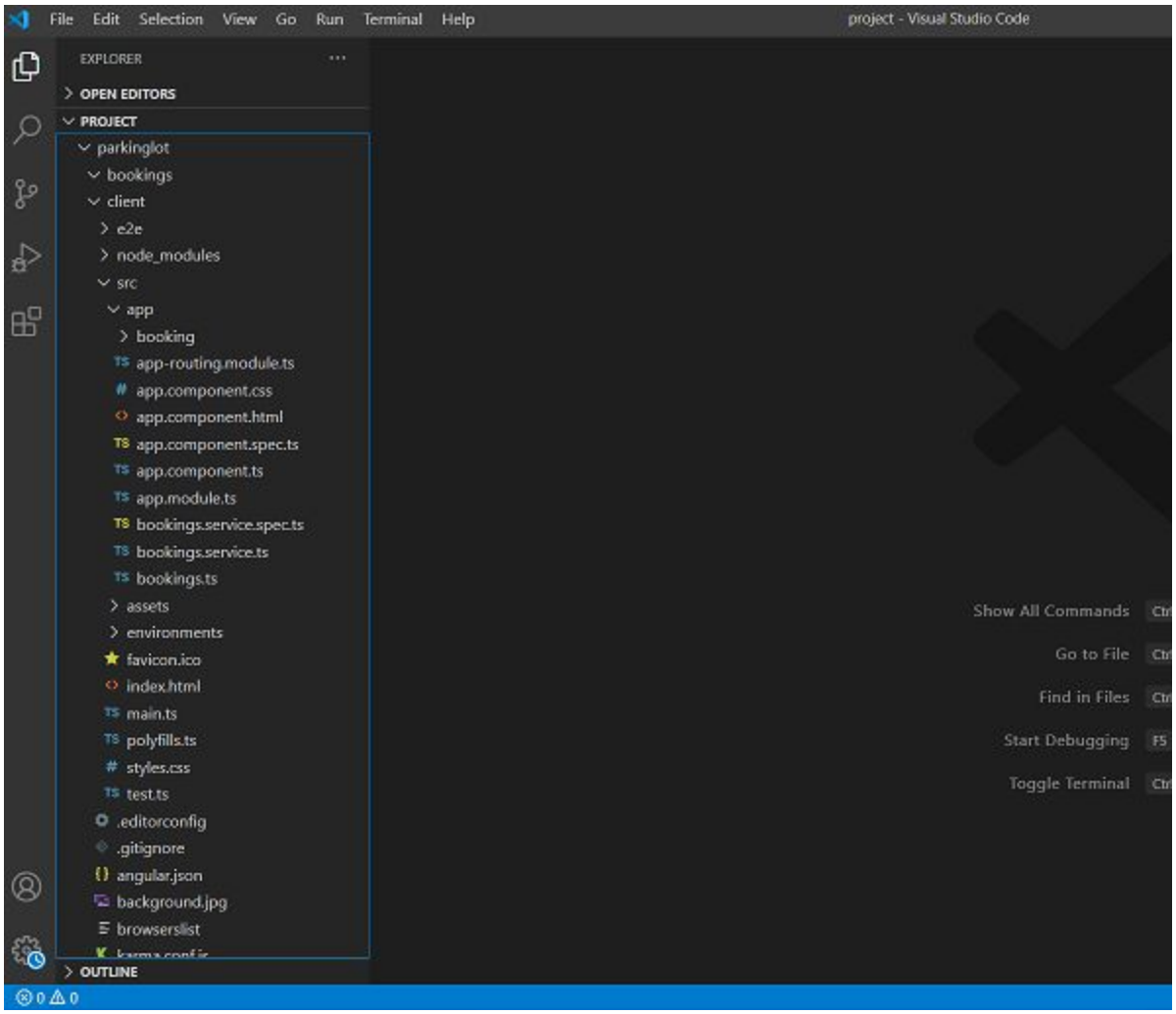
Entire system is tested as per the requirements. Black box testing falls under the category of system testing which we can test without knowing the code or logic.

6.2 Testing Plan

Testing accounts for 45-75% of the typical project effort. It is also one of the most commonly underestimated activities on a project. A test plan is a document that answers the basic questions about your testing effort. It needs to be initiated during the requirements gathering phase of your project and should evolve into a roadmap for the testing phase.

- Test planning enables a more reliable estimate of the testing effort up front.
- It allows the project team time to consider ways to reduce the testing effort without being under time pressure.

FOLDER STRUCTURE:



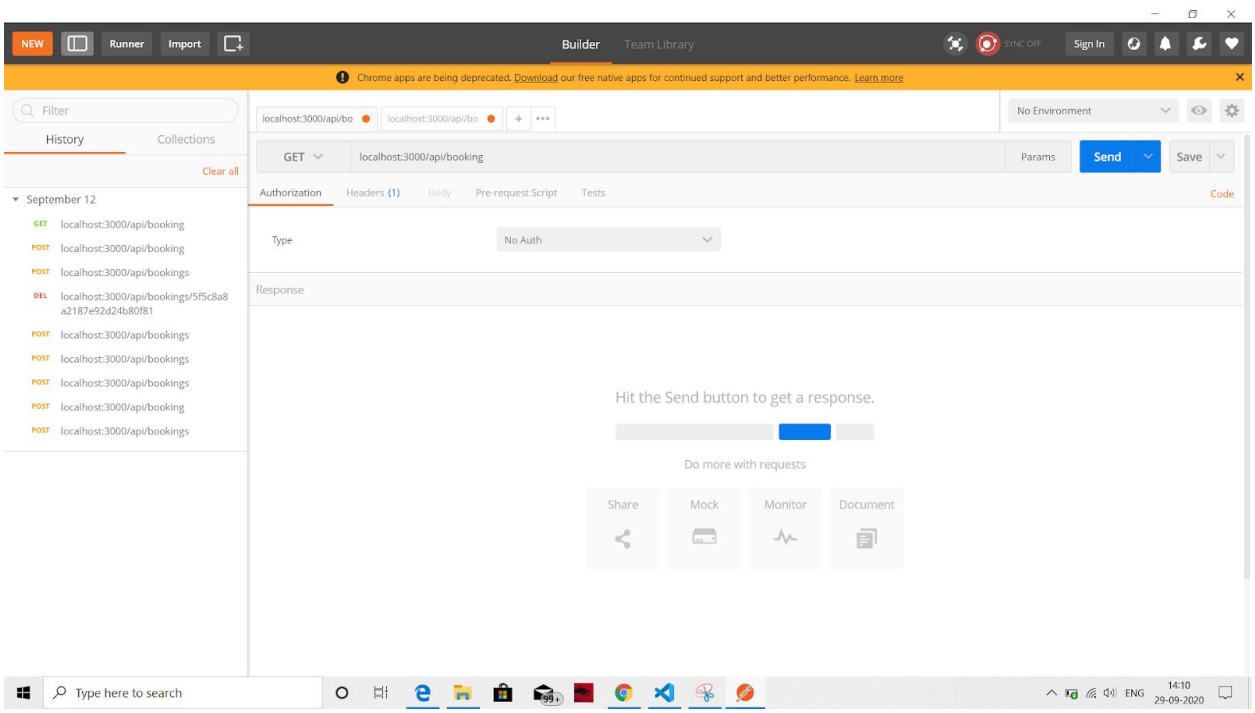
RESULT:

Modules:

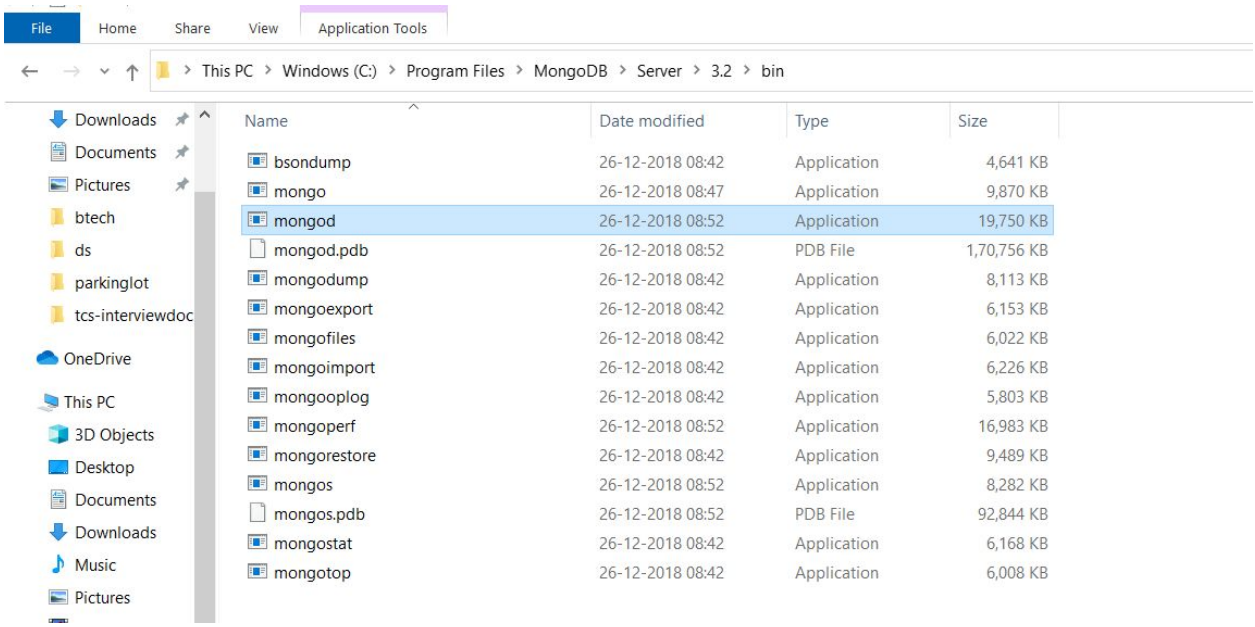
Name	Date modified	Type	Size
bookings	22-03-2020 10:56	File folder	
client	11-09-2020 11:21	File folder	
models	11-09-2020 11:21	File folder	
node_modules	11-09-2020 11:31	File folder	
public	11-09-2020 11:31	File folder	
routes	11-09-2020 11:31	File folder	
app	15-03-2020 12:03	JavaScript Source ...	1 KB
package	19-03-2020 14:04	JSON Source File	1 KB
package-lock	19-03-2020 14:04	JSON Source File	126 KB

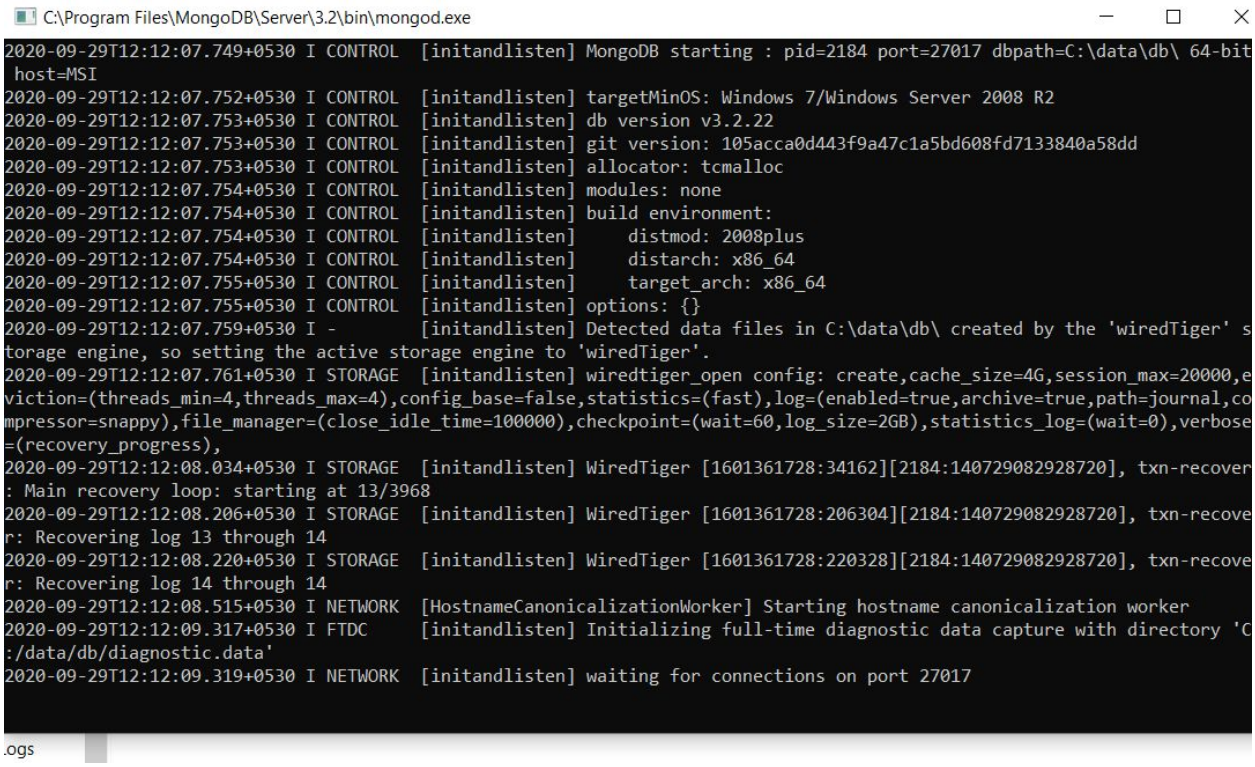
DATABASE CONNECTION:

Postman chrome for sending requests:

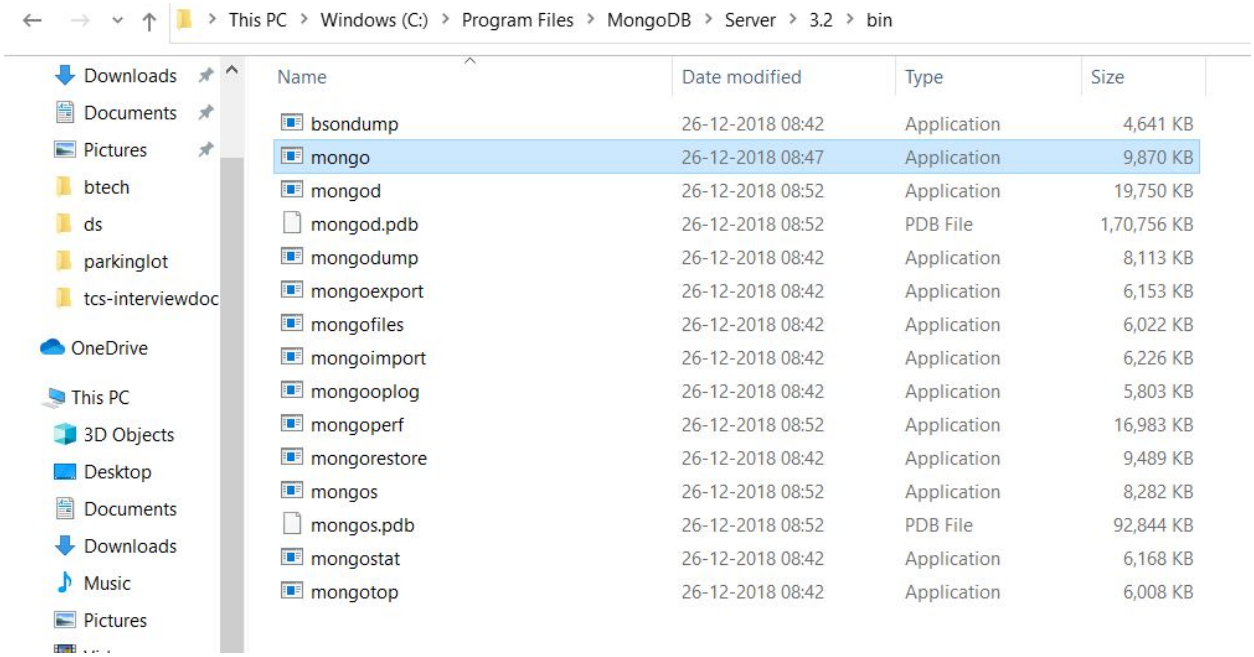


Step-1:

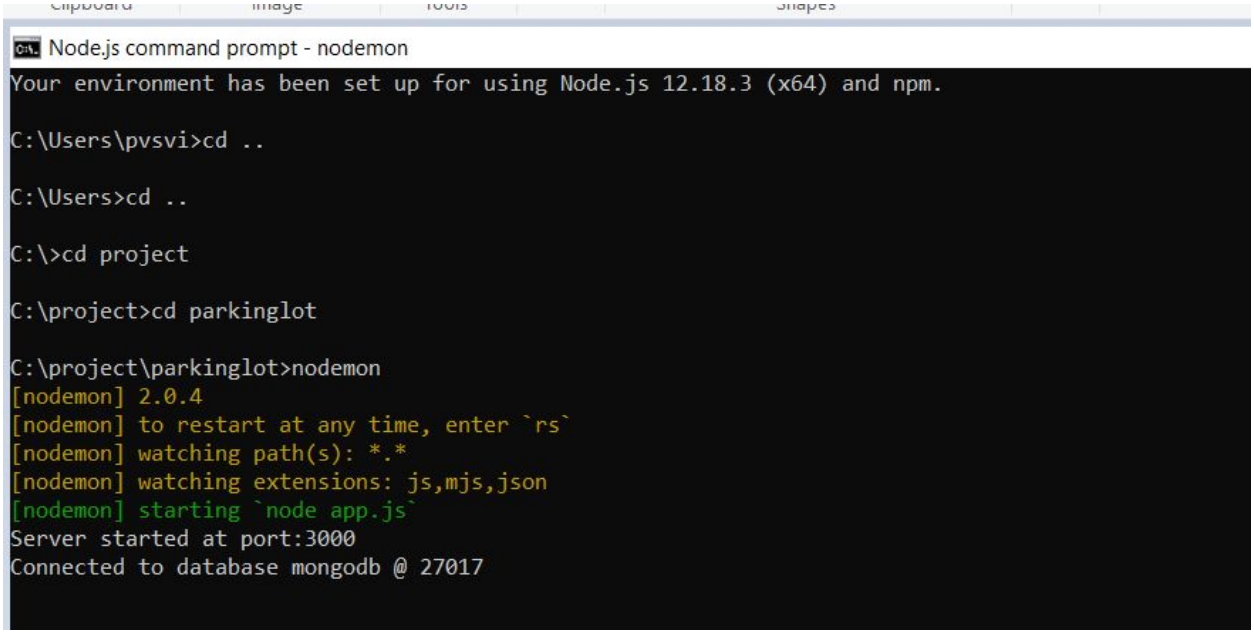

File
Home
Share
View
Application Tools
This PC > Windows (C:) > Program Files > MongoDB > Server > 3.2 > bin
Downloads
Documents
Pictures
btech
ds
parkinglot
tcs-interviewdoc
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Name
Date modified
Type
Size
bsondump 26-12-2018 08:42 Application 4,641 KB
mongo 26-12-2018 08:47 Application 9,870 KB
mongod 26-12-2018 08:52 Application 19,750 KB
mongod.pdb 26-12-2018 08:52 PDB File 1,70,756 KB
mongodump 26-12-2018 08:42 Application 8,113 KB
mongoexport 26-12-2018 08:42 Application 6,153 KB
mongofiles 26-12-2018 08:42 Application 6,022 KB
mongoimport 26-12-2018 08:42 Application 6,226 KB
mongooplog 26-12-2018 08:42 Application 5,803 KB
mongoperf 26-12-2018 08:52 Application 16,983 KB
mongorestore 26-12-2018 08:42 Application 9,489 KB
mongos 26-12-2018 08:52 Application 8,282 KB
mongos.pdb 26-12-2018 08:52 PDB File 92,844 KB
mongostat 26-12-2018 08:42 Application 6,168 KB
mongotop 26-12-2018 08:42 Application 6,008 KB

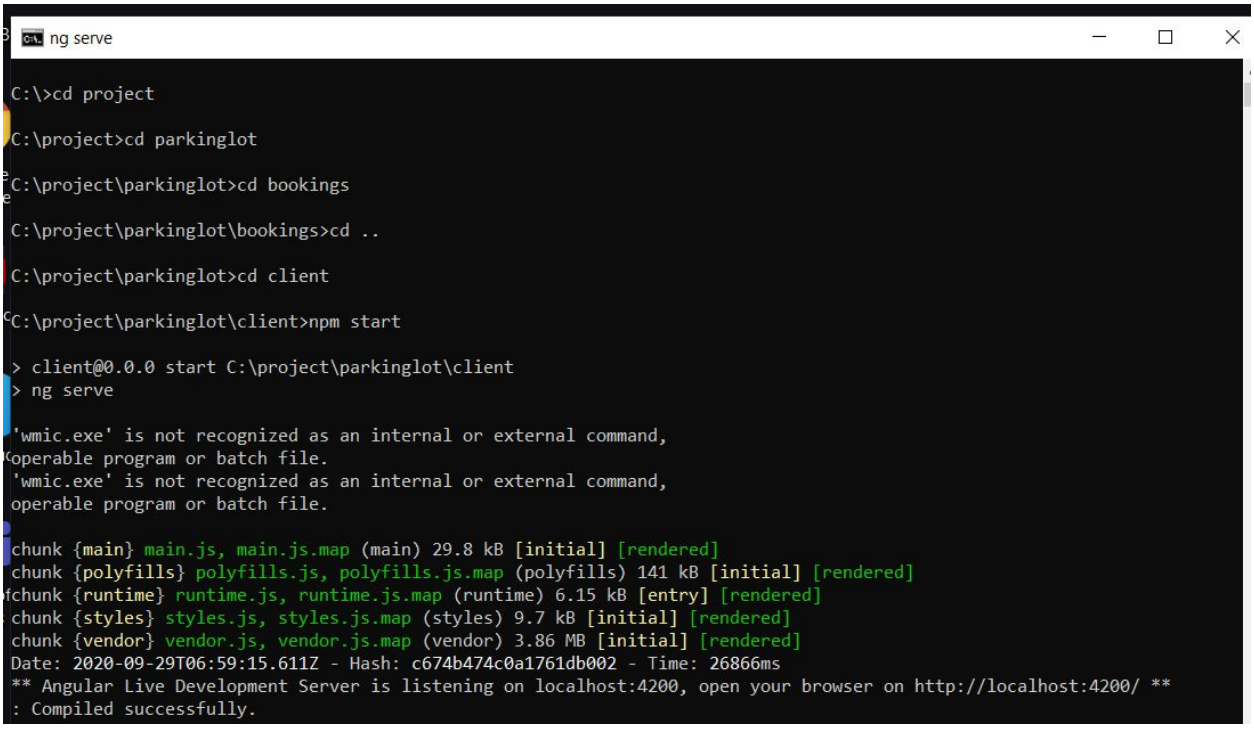

C:\Program Files\MongoDB\Server\3.2\bin\mongod.exe
2020-09-29T12:12:07.749+0530 I CONTROL [initandlisten] MongoDB starting : pid=2184 port=27017 dbpath=C:\data\db\ 64-bit host=MSI
2020-09-29T12:12:07.752+0530 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-09-29T12:12:07.753+0530 I CONTROL [initandlisten] db version v3.2.22
2020-09-29T12:12:07.753+0530 I CONTROL [initandlisten] git version: 105acca0d443f9a47c1a5bd608fd7133840a58dd
2020-09-29T12:12:07.753+0530 I CONTROL [initandlisten] allocator: tcmalloc
2020-09-29T12:12:07.754+0530 I CONTROL [initandlisten] modules: none
2020-09-29T12:12:07.754+0530 I CONTROL [initandlisten] build environment:
2020-09-29T12:12:07.754+0530 I CONTROL [initandlisten] distmod: 2008plus
2020-09-29T12:12:07.754+0530 I CONTROL [initandlisten] distarch: x86_64
2020-09-29T12:12:07.755+0530 I CONTROL [initandlisten] target_arch: x86_64
2020-09-29T12:12:07.755+0530 I CONTROL [initandlisten] options: {}
2020-09-29T12:12:07.759+0530 I - [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2020-09-29T12:12:07.761+0530 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=4G,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),verbose=(recovery_progress),
2020-09-29T12:12:08.034+0530 I STORAGE [initandlisten] WiredTiger [1601361728:34162][2184:140729082928720], txn-recover: Main recovery loop: starting at 13/3968
2020-09-29T12:12:08.206+0530 I STORAGE [initandlisten] WiredTiger [1601361728:206304][2184:140729082928720], txn-recover: Recovering log 13 through 14
2020-09-29T12:12:08.220+0530 I STORAGE [initandlisten] WiredTiger [1601361728:220328][2184:140729082928720], txn-recover: Recovering log 14 through 14
2020-09-29T12:12:08.515+0530 I NETWORK [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2020-09-29T12:12:09.317+0530 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2020-09-29T12:12:09.319+0530 I NETWORK [initandlisten] waiting for connections on port 27017
.ogs

Step-2:



Successful connection:



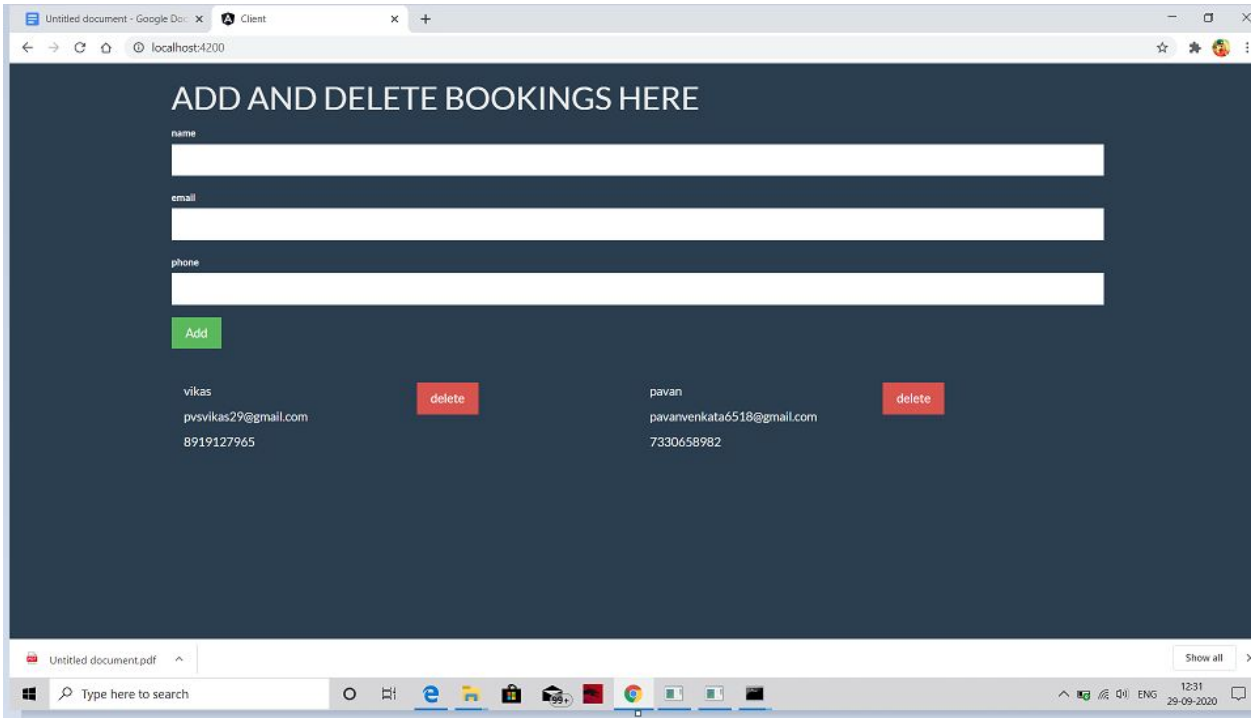


FUNCTION MODULES:

Addbookings

Deletebookings

Deleteby id



CHAPTER -9

SOURCE CODE:

Booking a spot:

Bookingcomponent.html

booking.component.html

parkinglot > client > src > app > booking > booking.component.html > div.container > div > div.col-md-3 > p > p

```
<div class = "container">
  <h1>              ADD AND DELETE BOOKINGS HERE          </h1>
  <form (submit) = "addBookings()">
    <div class = "form-group">
      <label>name</label>
      <input type = "text" [(ngModel)]="name" name="name" class = "form-control">
    </div>
    <div class = "form-group">
     <label>email</label>
     <input type = "text" [(ngModel)] ="email" name="email" class = "form-control">
    </div>
    <div class = "form-group">
      <label>phone</label>
      <input type = "text" [(ngModel)] ="phone" name="phone" class = "form-control">
    </div>
    <div>
      <input type = "submit"  class = "btn btn btn-success" value="Add">
    </div>
  </form>
</div>

<br><br>

<div class = "container">
  <div *ngFor =" let bookings of booking">
      <div class="col-md-3">
      <p>  {{bookings.name}} <p>     <p> {{bookings.email}}</p>     {{bookings.phone}}
      </div>
      <div class="col-md-3">
       <input type="button" (click)= "deleteBookings(bookings.name)" value="delete" class="btn btn-danger">
      </div>
  </div>
</div>
```

TS booking.component.spec.ts ×

parkinglot > client > src > app > booking > TS booking.component.spec.ts > ...

```
1   import { async, ComponentFixture, TestBed } from '@angular/core/testing';
2
3   import { BookingComponent } from './booking.component';
4
5   describe('BookingComponent', () => {
6     let component: BookingComponent;
7     let fixture: ComponentFixture<BookingComponent>;
8
9     beforeEach(async(() => {
10      TestBed.configureTestingModule({
11        declarations: [ BookingComponent ]
12      })
13      .compileComponents();
14    }));
15
16    beforeEach(() => {
17      fixture = TestBed.createComponent(BookingComponent);
18      component = fixture.componentInstance;
19      fixture.detectChanges();
20    });
21
22    it('should create', () => {
23      expect(component).toBeTruthy();
24    });
25  });
26
```

```
import { Component, OnInit } from '@angular/core';
import {BookingsService} from '../bookings.Service';
import { Bookings } from '../bookings';
@Component({
  selector: 'app-booking',
  templateUrl: './booking.component.html',
  styleUrls: ['./booking.component.css'],
  providers:[BookingsService],
})
export class BookingComponent implements OnInit {

   booking: Bookings[];
   bookings: Bookings;
   name:string;
   email:string;
   phone:string;

  constructor(private bookingsService: BookingsService) { }

    addBookings()
    {
      const newBookings ={
        name:this.name,
        email:this.email,
        phone:this.phone

      }
      this.bookingsService.addBooking(newBookings)

    }
```

```
  deleteBookings(id:any)

   {

      var booking = this.booking;

          var booking= this.booking;

          this.bookingsService.deleteBookings(id)

          .subscribe(data =>{

            if(data.n==1)

            {
```

```
            for(var i=0;i < booking.length;i++)
            {
              if(booking[i].name==id)
              {
                    booking.splice(i,1);
              }
            }
          }
        })
  }
ngOnInit(): void {
  this.bookingsService.getBooking()
  .subscribe( booking =>
    this.booking = bool;
```

CHAPTER 10

10 REFERENCES

https://sourceforge.net/projects/dia-installer/

https://online.visual-paradigm.com/

https://ionicframework.com/docs

**********THANK YOU**********