1. **EXECUTIVE SUMMARY**

Pay And Park system is an application that allows users to book a parking slots online. The booking is just a click away, you don’t have to go to some areas and ask for parking.

The Pay and Park System project developed to provide an easy way in finding the parking space for vehicles. These projects help user by analysing the areas where parking is available and details about slots free in that area and then booking to that slot where controller can register online and add their slots. The admin phase updates their details. These will help to reduce the traffic in the parking place and user can search the parking slot easily.

## 2. INTRODUCTION

The Pay and Park System project developed to provide an easy way in finding the parking space for vehicles. These projects help user by analysing the areas where parking is available and details about slots free in that area and then booking to that slot where controller can register online and add their slots. The admin phase updates their details. These will help to reduce the traffic in the parking place and user can search the parking slot easily.

To ensure your transactions are available to download anytime, please sign in or register an account. In this way, drivers are encouraged to shorten parking time so that other drivers are given a reasonable chance of finding parking

**2.1 EXISTING SYSTEM**

The Existing system is manual and highly time consuming and the whole process is time consuming. In case of existing system, the user who wants to book a slot, he/she has to spend more time. They have to search a long time to get parking area for the vehicle. The staff keeps the record of the vehicles as hardcopy. It takes lot of time and maintaining security is very difficult. In today’s world most of the people don’t have time to waste. Parking spaces are very important to cities. A city must have enough parking spaces to provide their residents and their visitors a place to park their vehicles.

## 2.2 PROBLEM DEFINITION

In the existing system, we need to find the location manually for parking it is time consuming and difficult task, The land owner need to manage the location and user need to contact land owner manually. They have to search a long time to get parking area for the vehicle. The staff keeps the record of the vehicles as hardcopy. It takes lot of time and maintaining security is very difficult. In today’s world most of the people don’t have time to waste. Parking spaces are very important to cities. A city must have enough parking spaces to provide their residents and their visitors a place to park their vehicles.

## PROPOSED SYSTEM

The proposed system is design to solve the difficult of existing system. The new system will setup and help to run a Pay and Park Management System. The system contains the details about controllers, user, parking slot, parking fees, duration and booking. This system reduces paper works. It becomes very easy to maintain data and information. Objective of Java Project on Parking System the Purpose of the project is to build an application program to reduce the manual work for managing the Parking Fess, Types, Vehicle, Duration. It tracks all the details about the Duration, Customers Parking Slot.

## OBJECTIVE OF THE PROJECT

Most cities around the world require drivers to pay for the time they occupy a parking spot. In this way, drivers are encouraged to shorten parking time so that other drivers are given a reasonable chance of finding parking. This main objective for developing this project was to pay and park the vehicles through online.

Providing a simple web site for parking vehicles. By using this site, users can easily book and cancel a parking slot at home. It is an easy payment system. The main aim of this project is reducing the risk of finding the parking slots in any parking area. It eliminates the unnecessary travelling of vehicles across the filled parking slots in a city. Objective of Java Project on Parking System The purpose of the project is to build an application program to reduce the manual work for managing the Parking Fees, Types, Vehicles, Duration. It tracks all the details about the Duration, Customers Parking Slots

## SCOPE OF THE PROJECT

* Increases accuracy.
* The whole service is made at one click
* Required data can be easily accessed by the user.
* The proper security of the system is higher than the manual system.
* The proposed system simplifies the task of storage of data and report generation.
* Reduces the time involved.
* User friendly and improved computerized system is more flexible.

## 3. METHODOLOGY

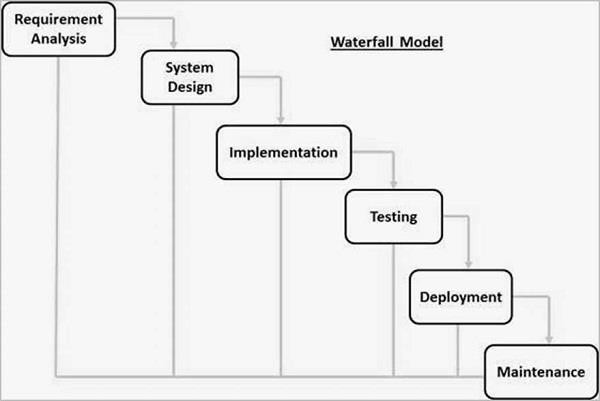
The project was implemented through waterfall model. The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. 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. In this waterfall model, the phases do not overlap.

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The sequential phases in Waterfall model are: −

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document. All necessary software- Net Beans, MySQL, JAVA JSP, and other requirements were downloaded during this phase.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.



## 4. DEVELOPMENT TOOLS

* 1. **INTRODUCTION TO JAVA**

JAVA was developed by Sun Microsystems Inc in 1991, later acquired by Oracle Corporation. It was developed by James Gosling and Patrick Naughton. It is a simple programming language. Writing, compiling and debugging a program is easy in java. It helps to create modular programs and reusable code.

The target of Java is to write a program once and then run this program on multiple operating systems. The first publicly available version of Java (Java 1.0) was released in 1995. Sun Microsystems was acquired by the Oracle Corporation in 2010. Oracle has now the steermanship for Java. In 2006 Sun started to make Java available under the GNU General Public License (GPL). Oracle continues this project called OpenJDK.

Java is defined by a specification and consists of a programming language, a compiler, core libraries and a runtime (Java virtual machine) The Java runtime allows software developers to write program code in other languages than the Java programming language which still runs on the Java virtual machine. The Java platform is usually associated with the Java virtual machine and the Java core libraries.

JAVA is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It is integrated with a number of popular databases, including MySQL, HyperSQL,Apache Derby,Java DB,ObjectDB.The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time. JAVA Syntax is like C and C++.JAVA performs system functions, i.e. from files on a system it can create, open, read, write, and close them. JAVA can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user. You add, delete, modify elements within your database through JAVA. Access cookies variables and set cookies. Using JAVA, you can restrict users to access some pages of your website. It can encrypt data. Five important characteristics make JAVA's practical nature possible −

Simplicity

Efficiency

Security

Flexibility

Familiarity

**FEATURES OF JAVA**

* **Java is a platform independent language:**

Compiler(javac) converts source code (.java file) to the byte code (.class file). As mentioned above, JVM executes the byte code produced by compiler. This byte code can run on any platform such as Windows, Linux, Mac OS etc. Which means a program that is compiled on windows can run on Linux and vice-versa. Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems..

### **Object-Oriented language:**

Object oriented programming is a way of organizing programs as collection of objects, each of which represents an instance of a class.



* **Simple:**

Java is considered as one of simple language because it does not have complex features like Operator overloading, Multiple inheritance, pointers and Explicit memory allocation.

* **Robust Language:**
* Robust means reliable. Java programming language is developed in a way that puts a lot of emphasis on early checking for possible errors, that’s why java compiler is able to detect errors that are not easy to detect in other programming languages. The main features of java that makes it robust are garbage collection, Exception Handling and memory allocation
* **Secure:**

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

* **Distributed:**

Using java programming language, we can create distributed applications. RMI (Remote Method Invocation) and EJB (Enterprise Java Beans) are used for creating distributed applications in java. In simple words: The java programs can be distributed on more than one system that are connected to each other using internet connection. Objects on one JVM (java virtual machine) can execute procedures on a remote JVM..

* **Multithreading:**

Java supports multithreading. Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU..

**Dynamically Typed Language:**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

**4.2 MySQL**

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

**5. MILESTONES**

### **Sprint 1**

Our first meeting was held on 19-1-2022 with the product owner.

Ms. Aswathy M R, it took about an hour to collect necessary details for the project.

### **Sprint 2**

During sprint 2 we started form designing. Development of the admin module was also initiated at this point of time. Adding service provider registration took place in sprint 2 phase. Parallelly, work was being done on the palliative care and health centre modules.

### **Sprint 3**

During sprint 3 the application development where started. The database view and other queries was worked on. Application for the service was being worked on where the user request are evaluated by the admin and service providers . User request and feedback were also developed.

### **5.4 Sprint 4**

Website and Application development were completed . Testing and validation took place during sprint 4.

## SYSTEM DESIGN

## MODULE DESCRIPTION

The system provide four modules

* **Admin**
* **User**
* **Controller**

The details about each module mentioned above is as follows:

1. **Admin Module:**

* Admin manage the entire system View User and Controller
* View slots and feedback
* Approve Controller

1. **User Module:**

* In this module an existing user mention the user name and the password and can enter into the home page.
* If the user name and the password didn’t matched the error message will be displayed. And there is new user registration is available in which the new user can mention their details including the user name, email, phone number, location and Unique password.
* User can book slot for parking in their convenient time and date. After the successful booking user will be redirect into home page
* User can give feedback to the parking area

**3. Controller Module:**

Through registration a Controller units can register into this application by providing their Name and other details. Then through login they can enter into their home page.

* Controller units can add slot for parking, by this way user can book the parking slot.
* Controller units can also view the booking, When the user park into the place controller check in the user similarly when the user left the parking area controller will checked out the customer
  1. **DATABASE DESIGN**

### **Login**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATATYPE** | **CONSTRAINTS** | **DESCRIPTION** |
| uname | int | PRIMARY KEY | Login Id |
| pwd | Varchar(100) | Not Null | Username |
| type | Varchar(100) | Not Null | User password |
| status | Varchar(100) | Not Null | User type |

* **Controller Registration**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATATYPE** | **CONSTRAINTS** | **DESCRIPTION** |
| C\_id | Int | PRIMARY KEY | Controller id |
| name | Varchar(50) | Not Null | name |
| gen | Varchar(50) | Not Null | gender |
| addr | Varchar(50) | Not Null | Address |
| phone | Int | Not Null | Phone number |
| email | Varchar(50) | Not Null | Email Id |
| Place | Varchar(50) | Not Null | Place |

* + **User**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATATYPE** | **CONSTRAINTS** | **DESCRIPTION** |
| Uid | Int | PRIMARY KEY | User id |
| name | Int | Not Null | Name |
| gender | Varchar(50) | Not Null | Gender |
| addr | Varchar(50) | Not Null | Address |
| c\_num | Varchar(50) | Not Null | Phone Number |
| email | Varchar(15) | Not Null | Email |

* + **Booking Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATATYPE** | **CONSTRAINTS** | **DESCRIPTION** |
| b\_id | Int | PRIMARY KEY | Booking Id |
| b\_person | Int | FORGIN KEY | Booked Person |
| controller | Varchar(100) | Not Null | Controller name |
| no\_slot | Varchar(25) | Not Null | Slot Number |
| total\_rate | Varchar(50) | Not Null | Date of Booking |
| p\_date | Date | Not Null | Parking Date |
| p\_time | Int | Not Null | Parking Time |
| pay\_status | Varchar(50) |  | Paying Status |
| map | Varchar(50) | Not Null | Google Map |
| slot\_id | Int | Not Null | Slot ID |
| updation | Varchar |  | Updations |

* + **Feedback Table**

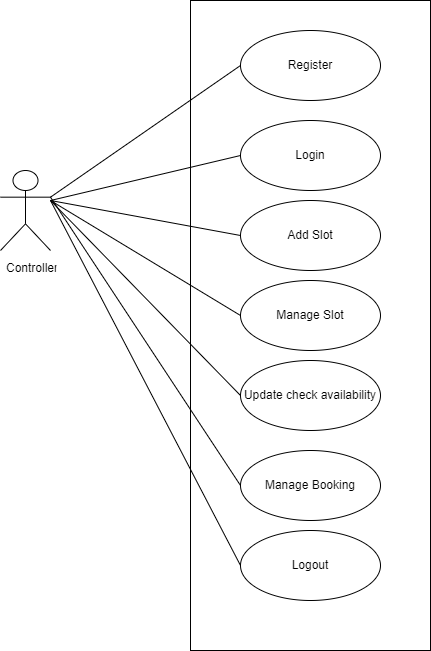
|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATATYPE** | **CONSTRAINTS** | **DESCRIPTION** |
| F\_id | Int | PRIMARY KEY | Feedbak Id |
| name | Varchar(50) | Not Null | User name |
| rating | Varchar(100) | Not Null | rating |
| Comments | Varchar(100 | Not Null | User Comment |

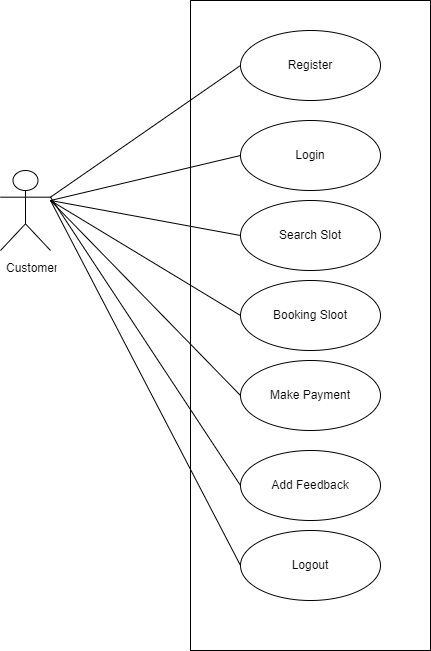
* + **Slot**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATATYPE** | **CONSTRAINTS** | **DESCRIPTION** |
| S\_id | Int | Primary Key | Doctor Id |
| C\_name | Varchar(50) | Foreign Key | Service provider Id |
| C\_email | Varchar(15) | Not Null | Doctor name |
| Veh\_type | Varchar(15) | Not Null | Phone number |
| noofslots | Varchar(50) | Not Null | Email id |
| addr | Varchar(100) | Not Null | Specialization |
| phn | Int | Not Null | Phone Number |
| amount | Int | Not null | Amount for Parking |
| location | Varchar(100) | Not Null | Location |
| district | Varchar(100) | Not Null | District |
| terms | Varchar(100) | Not Null | Terms and Condition |
| map | Varchar(100) | Not Null | Map |

## USE CASE DIAGRAMS

## 





1. **TESTING**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs

Testing can be stated as the process of validating and verifying that the software product such as meets the requirements that guided its design and development, responds correctly to all kinds of inputs, performs its functions within an acceptable time.

## Test Cases

* + 1. **Login**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.NO** | **INPUT** | **EXPECTED**  **RESULT** | **OBSERVED RESULT** |
| **1** | Click Login button with invalid username  and password. | Display “invalid username and password” | Display “invalid username and password” |
| **2** | Click Login button with valid username and  invalid password. | Display “invalid username and password” | Display “invalid username and password” |
| **3** | Click Login  button with each field as blank | Display “Please fill out this field” | Display “Please fill out this field” |
| **4** | Click Login button without verify email | Display “invalid username and password” | Display “invalid username and password” |

* + 1. **Registration**

|  |  |  |  |
| --- | --- | --- | --- |
| SL.NO | INPUT | EXPECTED  RESULT | OBSERVED  RESULT |
| 1 | Click login button with invalid username and  Password | Display “Please fill out this field” | Display ” Please fill out this field” |
| 2 | Click signup button without giving values to  some field | Display “Please fill out this field” | Display “Please fill out this field” |
| 3 | Click signup button with invalid  registration details | Display “successfully registered” | Display “successfully registered” |
| 4 | Click signup button with  invalid email | Display “Include an ‘@’ in the email  address.” | Display “Include an ‘@’ in the email  address.” |
| 5 | Click signup button with already registered  email | Display “already registered” | Display “already registered” |

* 1. **Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case No** | **No of Test Cases run** | **No of Test Cases Successful (%)** | **Pass/Fail** | **Expected Result** | **Actual Result** |
| TC-01 | 8 | 100 | PASS | The details should be stored in the DB. | The details are stored in the DB. |
| TC-02 | 10 | 100 | PASS | Output should be displayed and tables are updated | Output is displayed and tables are updated |
| TC-03 | 5 | 90 | PASS | Output should be displayed and tables must be updated | Output is displayed and tables are updated |
| TC-04 | 7 | 90 | PASS | Output should be displayed and tables must be updated | Output is displayed and tables are updated |

1. **SYSTEM IMPLEMENTATION**

The implementation phase of the software development is concerned with translating design specification into source code. The user tests the developed system and changes are made according to their needs. Our system has been successfully implemented. Before implementation several tests have been conducted to ensure that no errors are encountered during the operation. The implementation phase ends with an evaluation of the system after placing into the operation for a period of time. The following are modules are the ones that are already developed. Their working is as follows:

Registration: Through registration a User and Controller can log in to website

Login: Once the registration is done the user can login to the system. This module will ask the user to provide the username and password. User will be asked to validate whether the details are entered correctly. Once the entry is done correctly database will be checked for entry. If the user is authorized it will be directed to homepage.

Home Page: The user is redirected to this page once login done successfully. The user module then sends service request to User or Controller side from there we can book slot and controller can manage the slots and booking

The following are the steps involved in the implementation plan

* Test system with sample data
* Detection and correction of errors
* Make the necessary changes in the system
* Check the existing system
* Installation of hardware and software utilities
* Training and involvement of user personals

## CONCLUSION & FUTURE ENHANCEMENT

The Pay and Park System project developed to provide an easy way in finding the parking space for vehicles. These projects help user by analysing the areas where parking is available and details about slots free in that area and then booking to that slot where controller can register online and add their slots

With following the requirements and activated it without any error and inconsistency. Overall the teaches us the essential skills like:-

• Using system analysis and design techniques Like use case diagram in designing the system.

• Understanding the database handling and Query processing.

• Application development.

•Understanding python program structures.

•Finally we can say that this project performs in the best possible manner.

This project was completed within the time span allotted and I am glad to get this opportunity to do this project. All the knowledge I gained during the system study is completely applied in the design of the mentioned system.

**FUTURE ENHANCEMENT**

* In Future the checking checkout procedure will be automated using computer hardwares
* Live parking update will be available
* Map Location search will be integrated

## Appendix

* 1. **Appendix A**

**11.1.1 Sample Source Code**

Main.py

from flask import Flask

from public import public

from admin import admin

from palliativecare import palliativecare

from healthcare import healthcare

from api import api

import smtplib

from email.mime.text import MIMEText

from flask\_mail import Mail

app=Flask(\_\_name\_\_)

app.secret\_key="1234"

mail=Mail(app)

app.config['MAIL\_SERVER']='smtp.gmail.com'

app.config['MAIL\_PORT'] = 587

app.config['MAIL\_USERNAME'] = 'projectsriss2020@gmail.com'

app.config['MAIL\_PASSWORD'] = 'messageforall'

app.config['MAIL\_USE\_TLS'] = False

app.config['MAIL\_USE\_SSL'] = True

app.register\_blueprint(public)

app.register\_blueprint(admin,url\_prefix="/admin")

app.register\_blueprint(palliativecare,url\_prefix="/palliativecare")

app.register\_blueprint(healthcare,url\_prefix="/healthcare")

app.register\_blueprint(api,url\_prefix="/api")

app.run(debug=True,port=5007,host="192.168.43.67")

#bsbBBapp.run(debug=True,port=5007)

Public.py

from flask import \*

from database import \*

import uuid

public=Blueprint('public',\_\_name\_\_)

@public.route("/")

def index():

session.clear()

data={}

return render\_template("index.html",data=data)

@public.route("/login",methods=['get','post'])

def login():

# session.clear()

data={}

if 'login' in request.form:

uname=request.form['uname']

passw=request.form['passw']

q="SELECT \* FROM `login` WHERE `username`='%s' AND `password`='%s'"%(uname,passw)

res=select(q)

if res:

session['login\_id']=res[0]['login\_id']

lid=session['login\_id']

session['usertype']=res[0]['usertype']

if res[0]['usertype']=="admin":

flash("Welcome Back Admin")

return redirect(url\_for("admin.admin\_home"))

elif res[0]['usertype']=="Pending":

flash("Your Account Is Under Verification, Please Wait For The Confirmation.")

return redirect(url\_for("public.login"))

elif res[0]['usertype']=="Palliative Care":

q="SELECT \* FROM `serviceprovider` WHERE `login\_id`='%s'"%(lid)

res1=select(q)

if res1:

session['serviceprovider\_id']=res1[0]['serviceprovider\_id']

flash("Welcome Back Palliative Care")

return redirect(url\_for("palliativecare.palliativecare\_home"))

elif res[0]['usertype']=="Health Care":

q="SELECT \* FROM `serviceprovider` WHERE `login\_id`='%s'"%(lid)

res1=select(q)

if res1:

session['serviceprovider\_id']=res1[0]['serviceprovider\_id']

flash("Welcome Back Health Care")

return redirect(url\_for("healthcare.healthcare\_home"))

else:

print("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")

flash("Invalid Username Or Password")

return redirect(url\_for("public.login"))

return render\_template("login.html",data=data)

@public.route("/serviceprovider\_registration",methods=['get','post'])

def serviceprovider\_registration():

session.clear()

data={}

if 'serviceprovider' in request.form:

name=request.form['name']

image=request.files['image']

path='static/images/'+str(uuid.uuid4())+image.filename

image.save(path)

place=request.form['place']

phone=request.form['phone']

email=request.form['email']

ward\_no=request.form['ward\_no']

ptype=request.form['ptype']

uname=request.form['uname']

passw=request.form['passw']

q="SELECT \* FROM `serviceprovider` WHERE `ward\_no`='%s' AND `type`='%s'"%(ward\_no,ptype)

res=select(q)

if res:

flash("Sorry.")

else:

q="SELECT \* FROM `login` WHERE `username`='%s'"%(uname)

res=select(q)

if res:

flash("Username Already Exist. Please Choose Anotherone")

return redirect(url\_for("public.serviceprovider\_registration"))

else:

q="INSERT INTO `login` VALUES(NULL,'%s','%s','Pending')"%(uname,passw)

id=insert(q)

q1="INSERT INTO `serviceprovider` VALUES(NULL,'%s','%s','%s','%s','%s','%s','%s','%s')"%(id,name,place,phone,email,path,ward\_no,ptype)

insert(q1)

flash("Regsitration Successfully Completed. Please Wait For The Confirmation To Login.")

return redirect(url\_for("public.login"))

q='select \* from ward'

res=select(q)

data['wardnum']=res

return render\_template("serviceprovider\_registration.html",data=data)

@public.route("/password\_reset",methods=['get','post'])

def password\_reset():

session.clear()

data={}

if 'submit' in request.form:

uname=request.form['uname']

phone=request.form['phone']

email=request.form['email']

cpwd=request.form['password']

password=request.form['cpass']

if cpwd!=password:

flash("PASSWORD DOES NOT MATCH")

else:

q="select \* from login where username ='"+uname+"'"

res=select(q)

if res:

session['lid']=res[0]['login\_id']

if res[0]['usertype']=='Palliative Care':

q="select \* from serviceprovider inner join login using(login\_id) where login\_id='%s' and email='%s'"%(session['lid'],email)

res=select(q)

if res:

q="update login set password='%s' where login\_id='%s'"%(password,session['lid'])

update(q)

flash("PASSWORD UPDATED SUCCESSFULLY")

return redirect(url\_for('public.login'))

else:

flash("Please Input Your Valid Email")

return redirect(url\_for('public.login'))

elif res[0]['usertype']=='Health Care':

q="select \* from serviceprovider inner join login using(login\_id) where login\_id='%s' and email='%s'"%(session['lid'],email)

res=select(q)

if res:

q="update login set password='%s' where login\_id='%s'"%(password,session['lid'])

update(q)

flash("PASSWORD UPDATED SUCCESSFULLY")

return redirect(url\_for('public.login'))

else:

flash("Please Input Your Valid Email")

return redirect(url\_for('public.login'))

else:

flash("Invalid username")

return render\_template('password\_reset.html',data=data)

## Appendix B

* + 1. **Webliography**
* [www.w3schools.com](http://www.w3schools.com)
* [www.w3layout.com](http://www.w3layout.com)
* [www.wampserver.com](http://www.wampserver.com)
* [www.creatly.com](http://www.creatly.com)
* Sourse.android.com

## Bibliography

1. System Analysis And Design : Elias M Award

2.Introduction To Database Management System : Naveen Prakash

3.Database Management System : Raghu Ramakr

**UI Design**

Fig 1. Home page

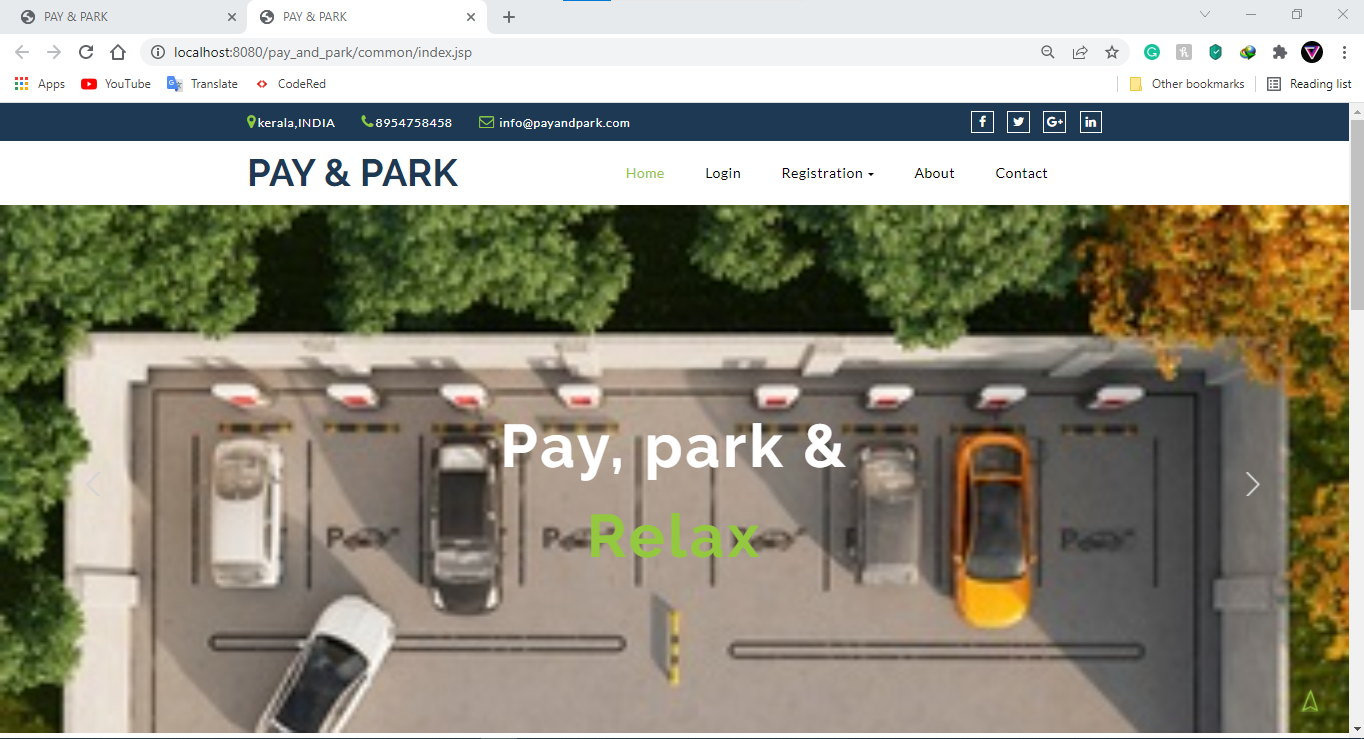


Fig 2. User Registration

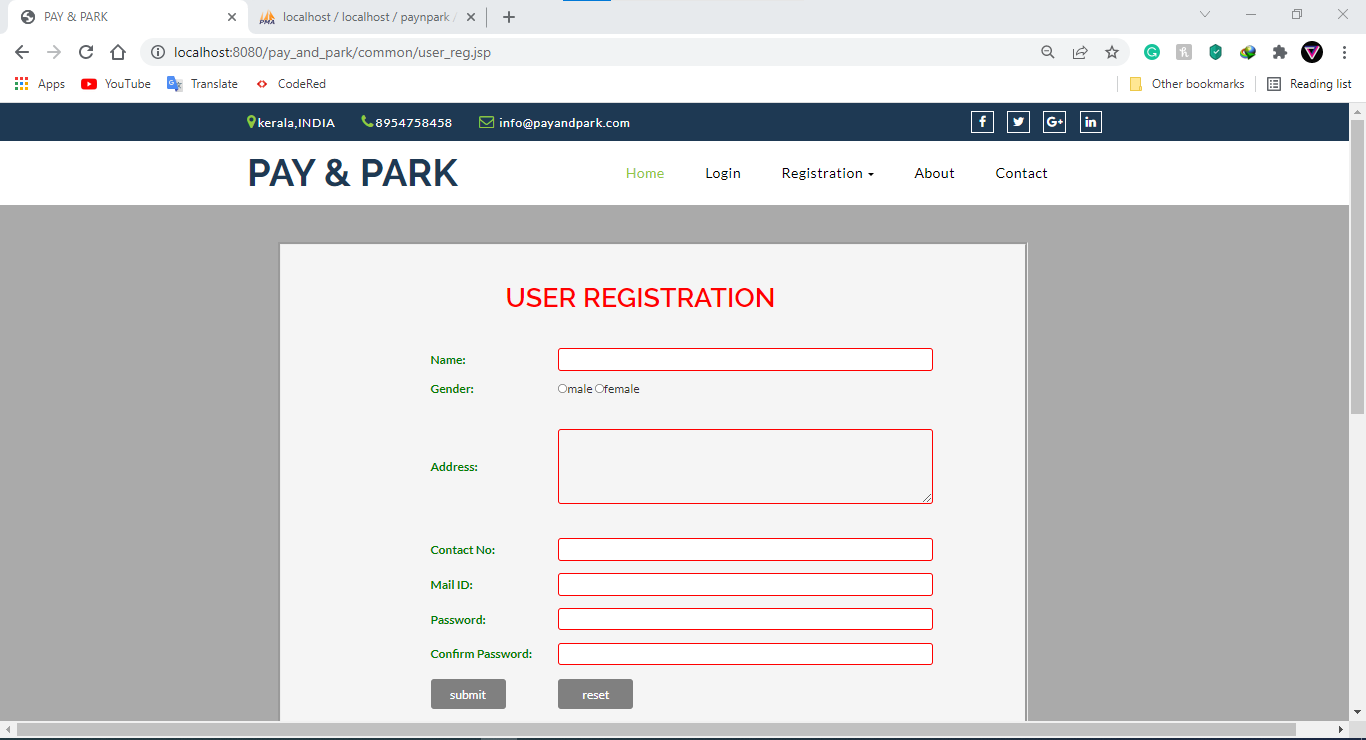


Fig 2. Registration page for controller

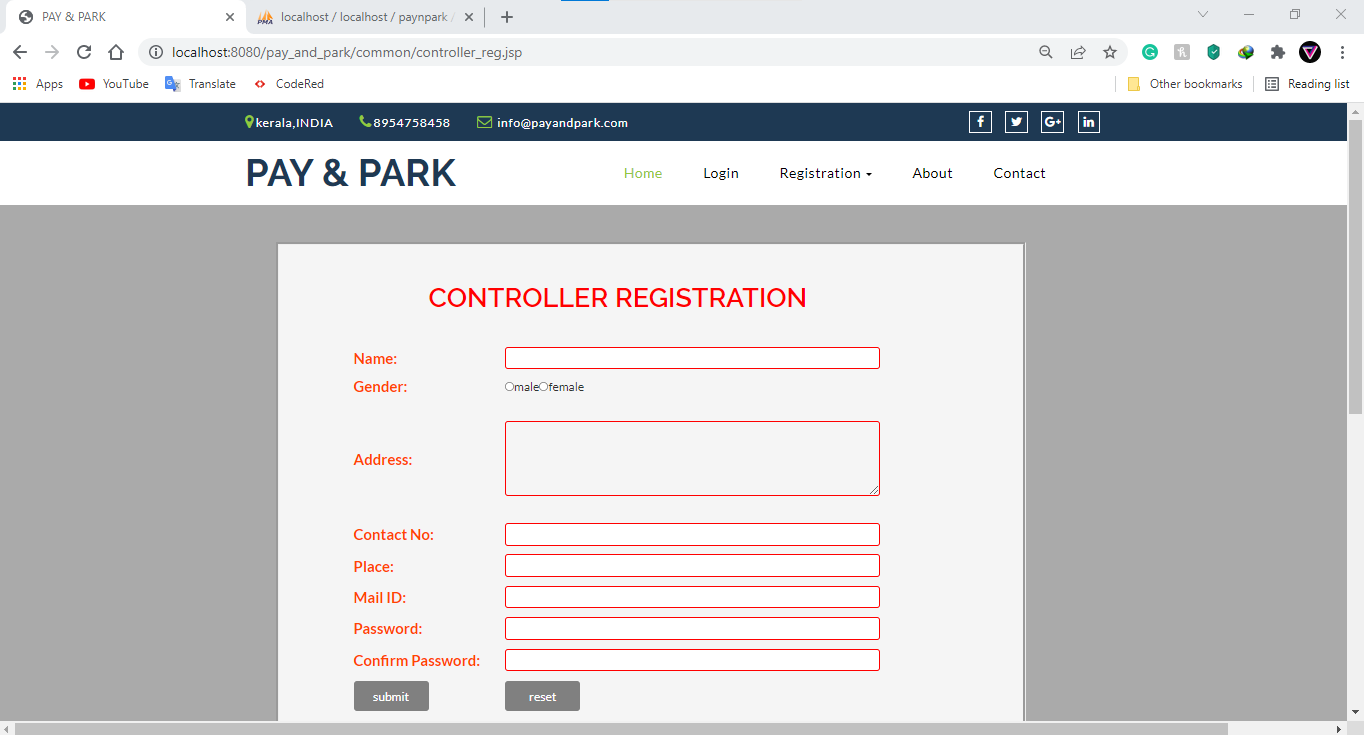


Fig 3 Login

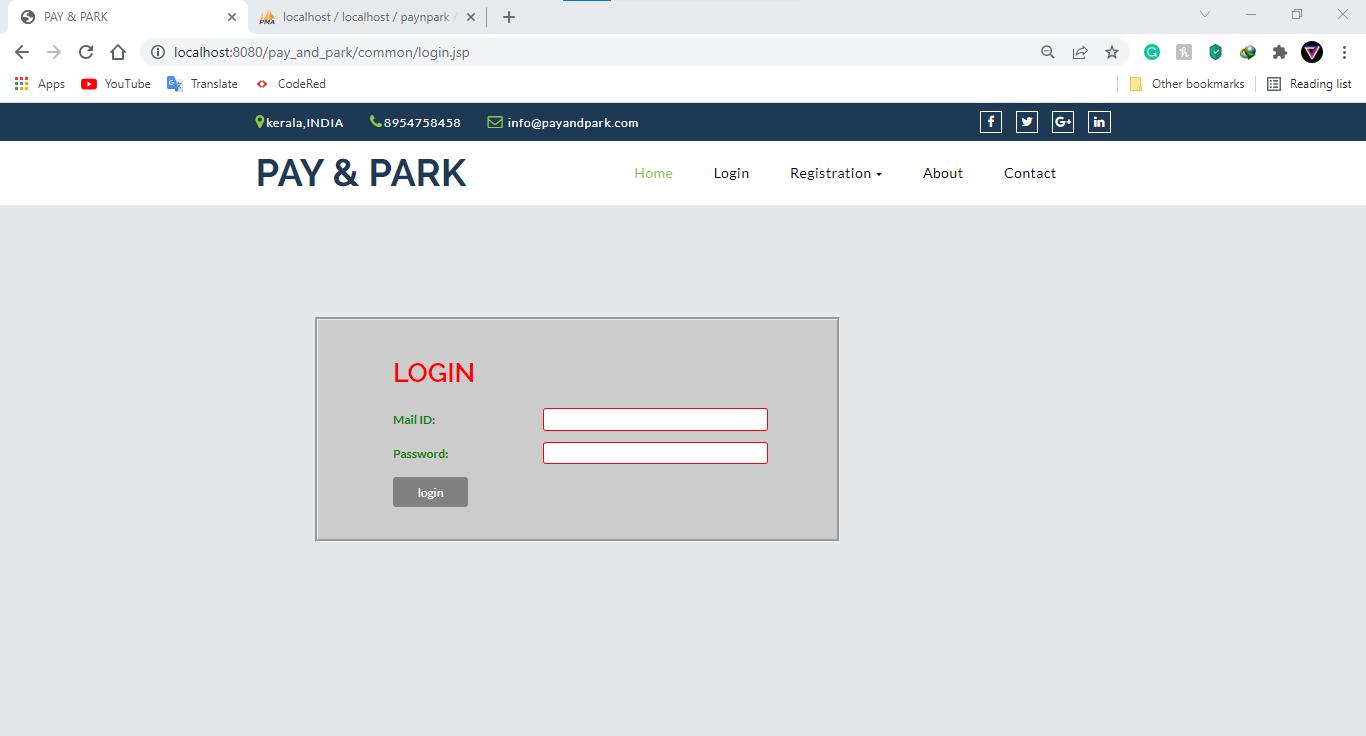


Fig 4. Password reset form.

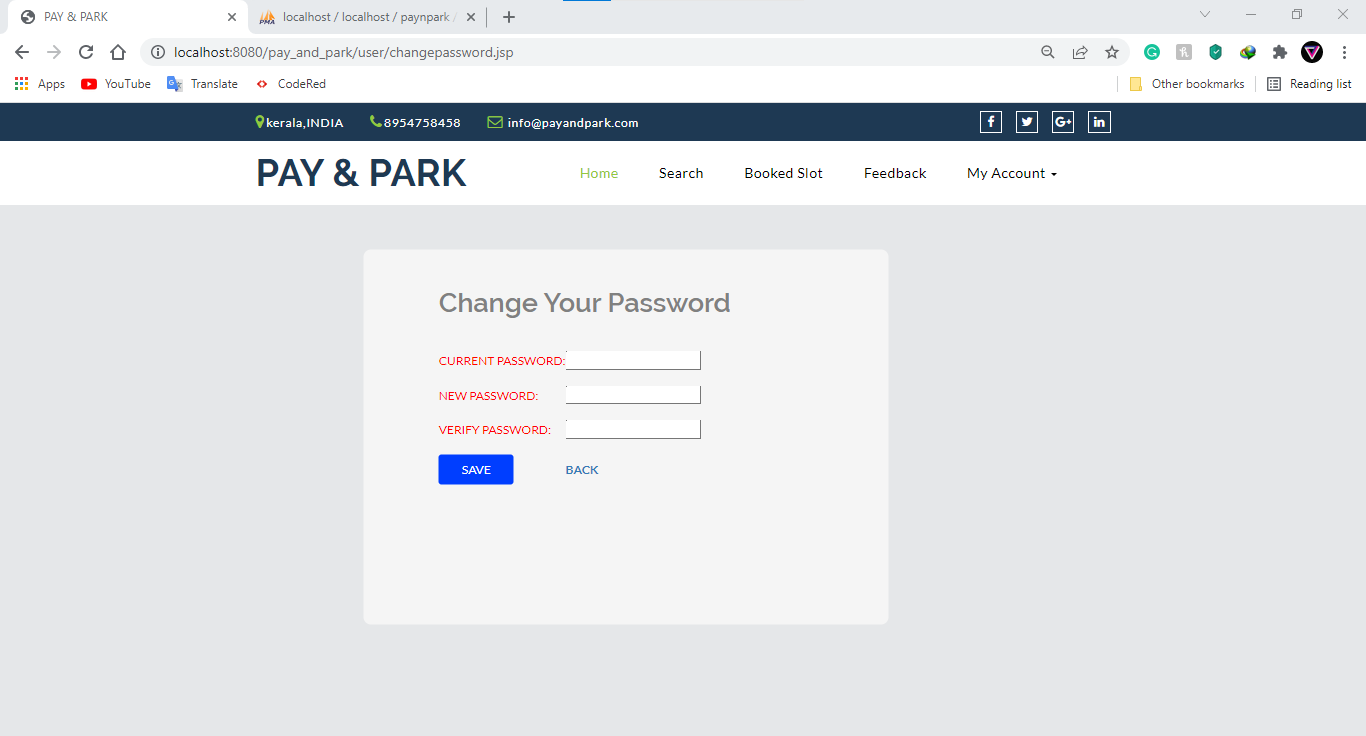


Fig 5. Home page of Admin

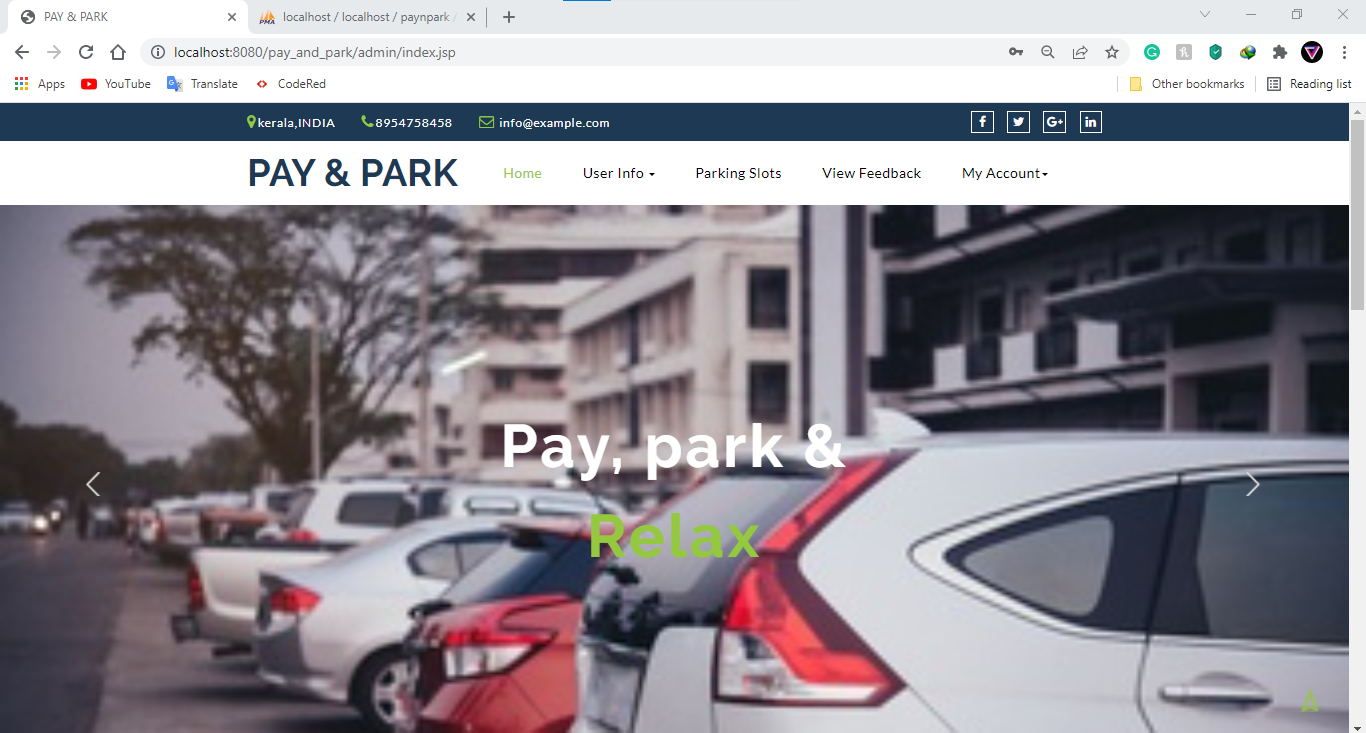


Fig 6. Controller Approval

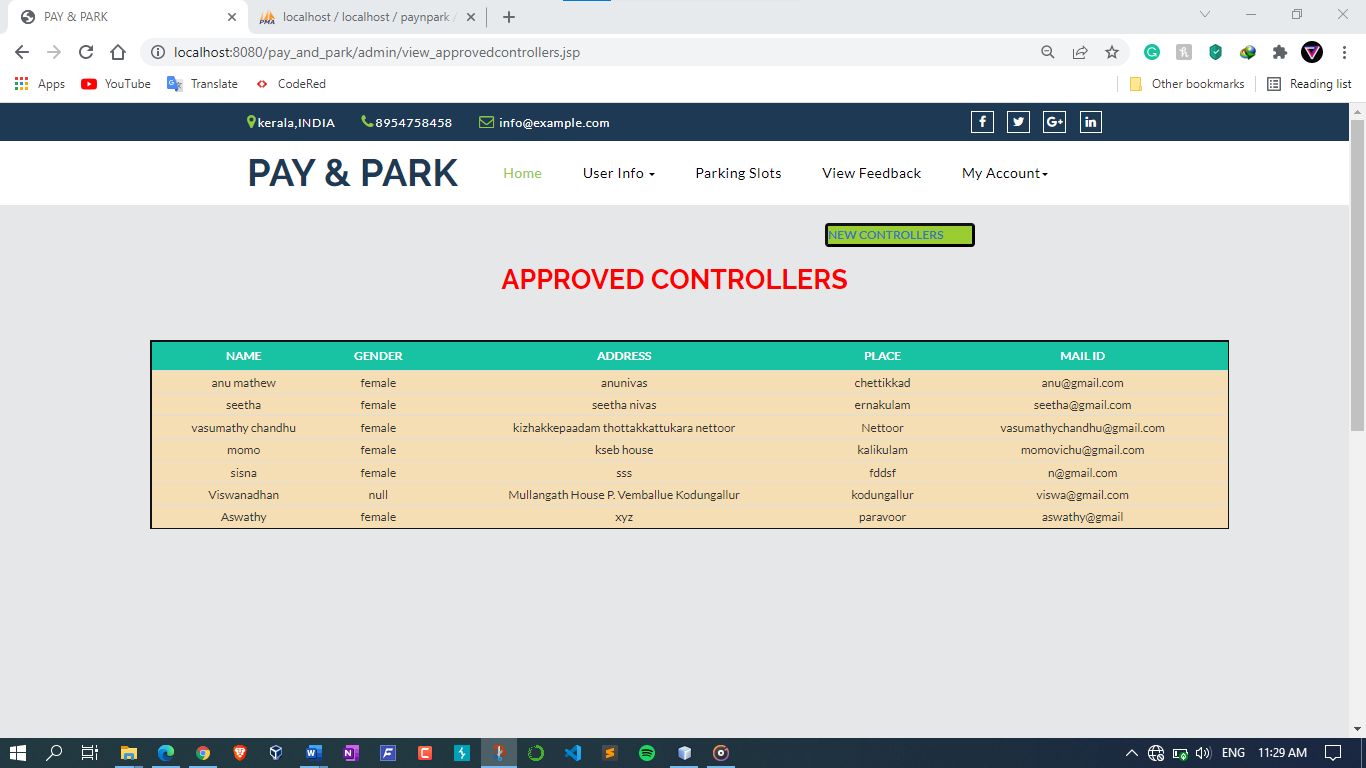


Fig 7. View Slots

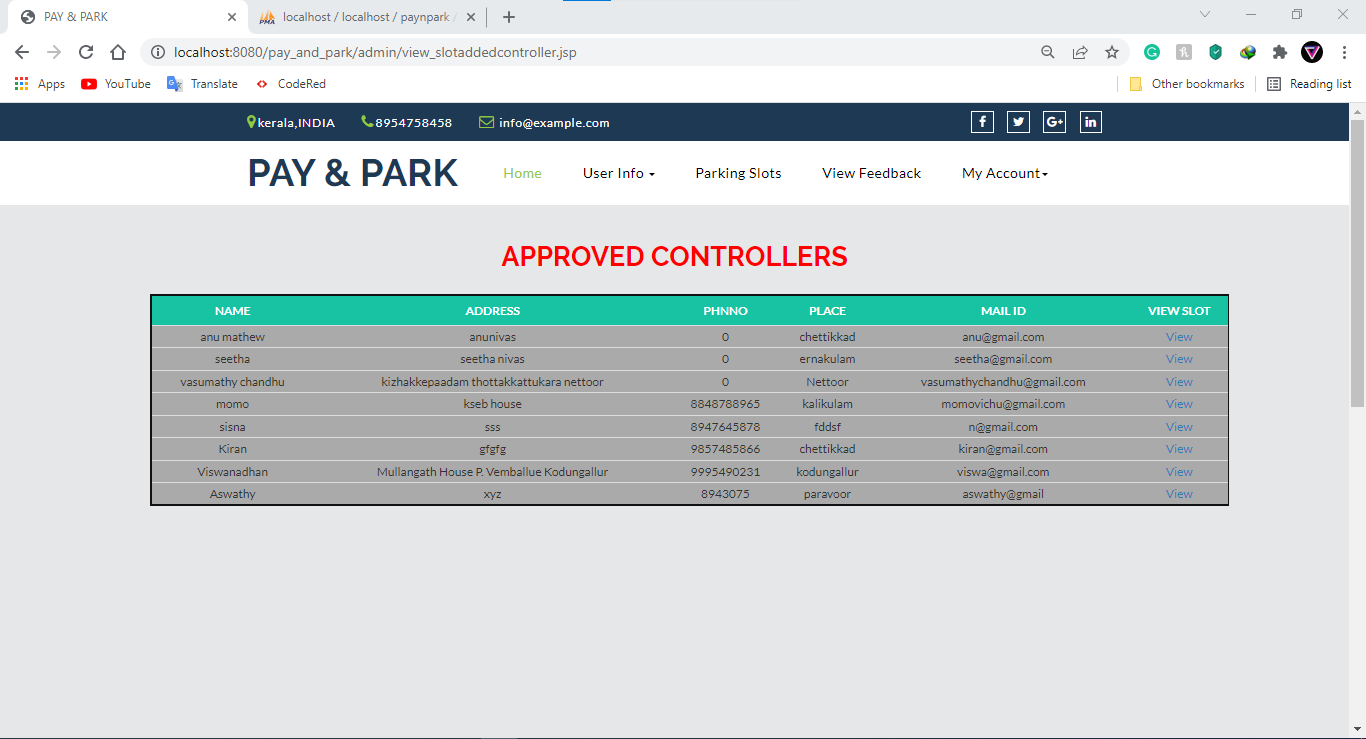


Fig 8. View Feedbacks

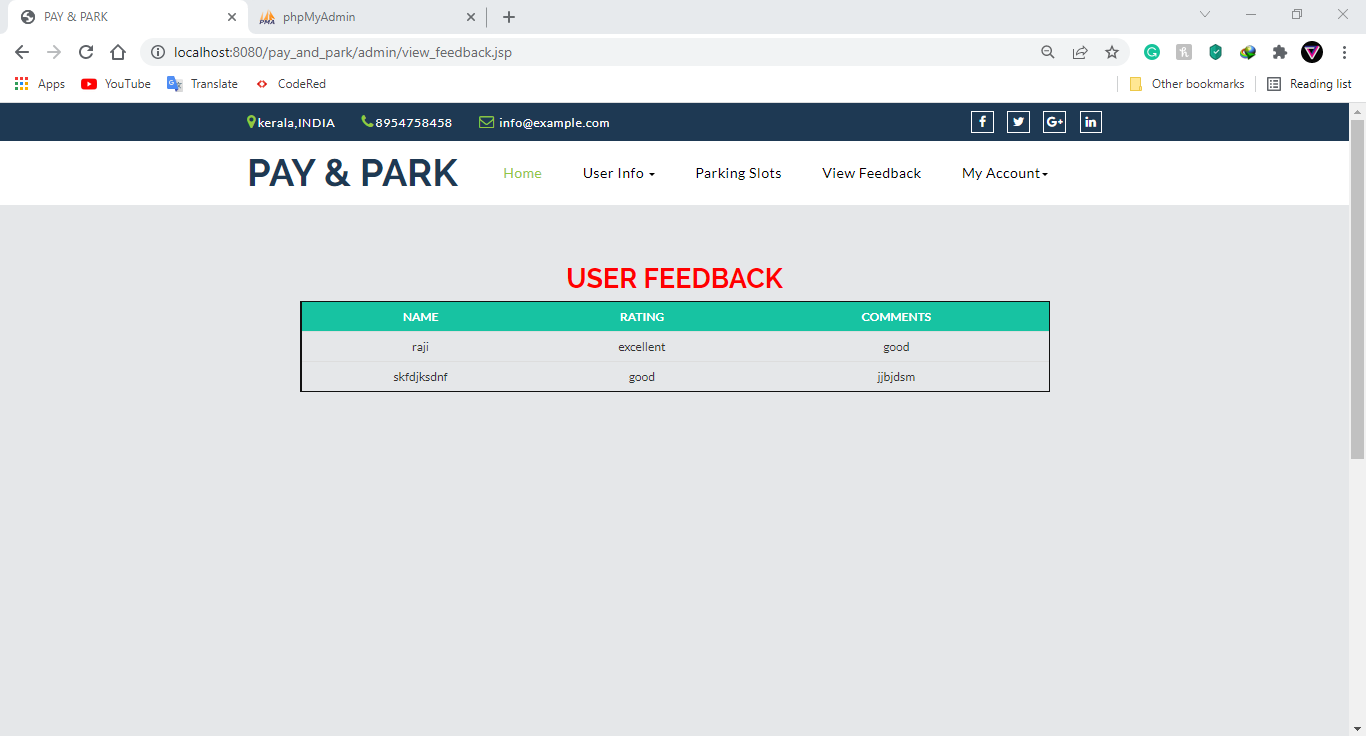


Fig 9 Search Slot

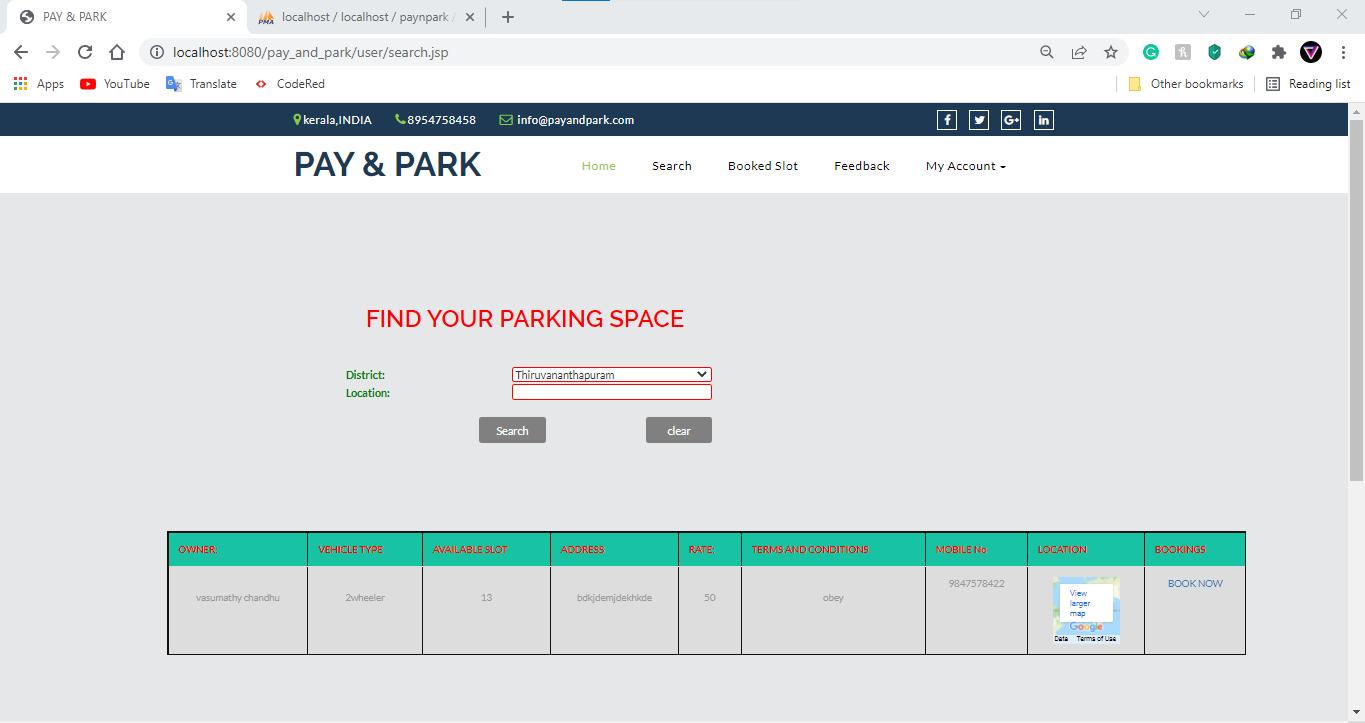


Fig 10. Booking

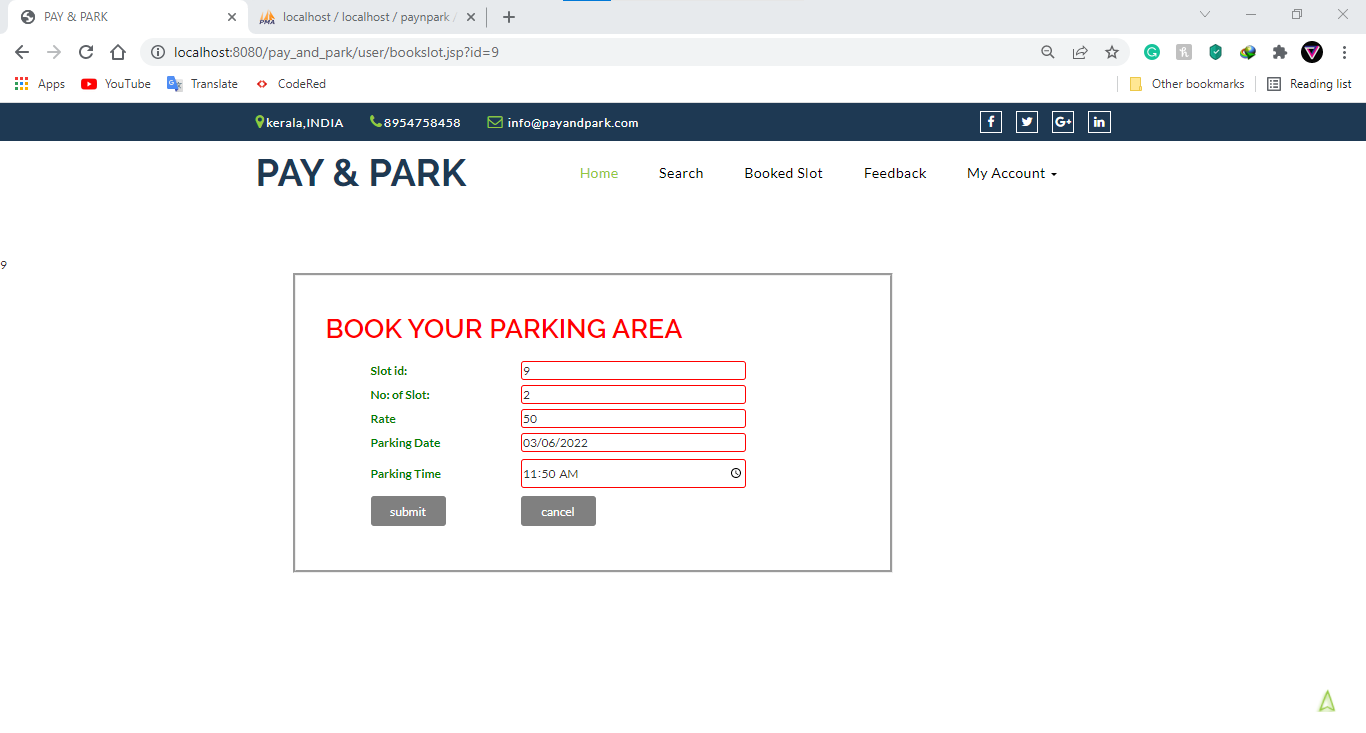


Fig 11 Payment Page

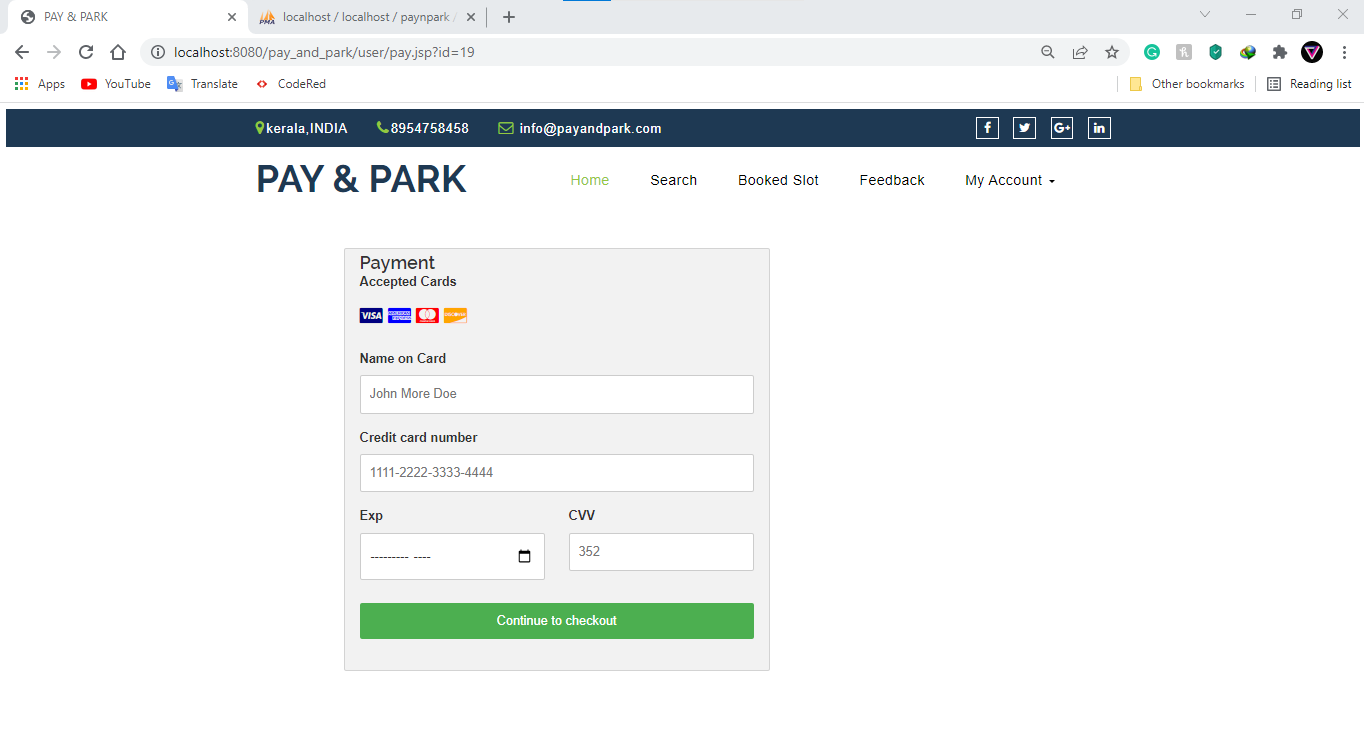


Fig 12. Controller Booked Details

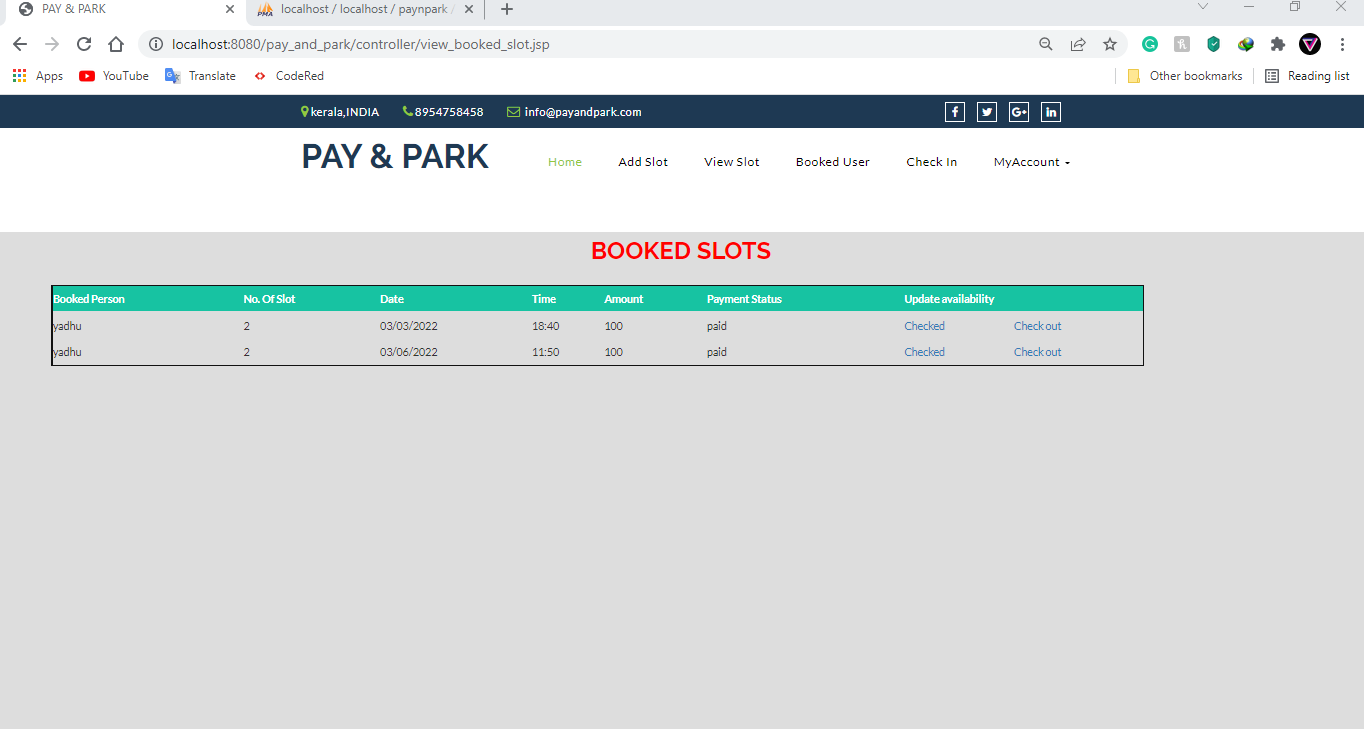


Fig 13. Checked in Details.

