PARKING SPOT INDICATOR IN VICNITY

Harshitha Chitimireddy

08-08-2024

ABSTARCT

This project focuses on developing a web-based Parking Spot Indicator system for efficient parking space management. Utilizing image processing techniques, it provides real-time updates on parking space availability within designated areas. The goal is to create an organized and dynamic parking environment, addressing challenges related to temporary parking spaces. Software requirements include Windows XP/7/8/10, Visual Studio Code, MySQL, Python, and the CV2 module. Hardware requirements comprise an Intel i3 processor, 1.1 GHz speed, 2 GB RAM (minimum), 2 GB disk space (minimum), and standard peripherals.

The methodology involves installing cameras to capture live footage of parking lots. Images are processed to identify Regions of Interest (ROIs) representing parking spaces. A neural network-based car detection module tracks and detects vehicles within ROIs, generating virtual lines for efficient parking. An admin interface monitors parking activities and providing real time updates on vacant spaces. Outcomes include a dynamic pricing model, and accessibility features. The system is designed with scalability and interoperability. The timeline outlines project execution stages. The conclusion highlights the web application's utility in addressing parking difficulties at mega-events, providing real-time updates and an admin interface for monitoring. Overall, the Parking Spot Indicator with Vicinity offers a professional solution for effective parking space management.

1.0 PROBLEM STATEMENT

In urban areas, finding a parking spot quickly and efficiently is a significant challenge for drivers, leading to wasted time, increased fuel consumption, and heightened stress levels. This problem is exacerbated by the lack of real-time information about available parking spots in the vicinity of a driver's location. Additionally, congestion in parking areas can lead to traffic bottlenecks and environmental pollution.

The Parking Spot Indicator in Vicinity system aims to address these issues by providing drivers with real-time information about available parking spots near their current location. By utilizing a combination of sensors, GPS, and mobile technology, the system will detect and display open parking spots on a user-friendly mobile app or car dashboard interface. The solution will help drivers make informed decisions, reduce search time, alleviate traffic congestion, and contribute to a greener environment by reducing unnecessary driving.

This system could be implemented in various environments, such as shopping malls, airports, public parking lots, and even street parking areas, making it a versatile solution for modern urban challenges.

2.0 INTRODUCTION

In an era characterized by rapid urbanization and burgeoning population growth, the demand for efficient and intelligent urban infrastructure becomes imperative. Parking management, a critical component of urban planning, is confronted with challenges such as congestion, inadequate space utilization, and the need for real-time monitoring. This project endeavors to address these challenges through the implementation of a sophisticated Parking Spot Indicator with Vicinity, designed to elevate the standard of parking space management.

The essence of this initiative lies in leveraging cutting-edge technologies, specifically image processing and computer vision, to create a dynamic and responsive system. Aimed at optimizing parking space utilization, the project targets temporary parking areas, where conventional parking systems are often absent. By utilizing a web-based interface, this system seeks to provide users with real-time insights into parking space availability within a designated vicinity.

The foundation of the project rests on a meticulously chosen set of software and hardware requirements. From operating systems to development tools, databases, and hardware specifications, each component has been thoughtfully selected to ensure seamless integration and optimal performance. The utilization of the CV2 module, coupled with neural network-powered car detection, forms the technological backbone, facilitating accurate identification and tracking of vehicles within specified regions. The outcomes envisioned from this project extend beyond more space optimization.

The incorporation of a dynamic pricing model, data analytics for future urban planning and accessibility features underscores a holistic approach toward transforming the parking landscape.

Furthermore, scalability and interoperability have been integral considerations, ensuring adaptability to evolving technological landscapes.

3.0 OBJECTIVE

The primary aim of this project is to revolutionize the parking experience by developing an efficient and affordable smart parking solution. Using advanced image processing techniques, the system will swiftly detect the presence of vehicles in parking lots and provide real-time updates on the number and location of available parking spaces.

3.1 RESEARCH GAPS OF EXISTING METHODS

The literature on smart parking systems, while diverse and insightful, reveals certain research gaps that warrant further exploration. Parikh et al.'s "Park Indicator" introduces a reservation system for parking spots, but there is a lack of discussion on the scalability and adaptability of such systems, particularly in high-density urban areas (Parikh et al., n.d. [1]). Reinhard Hössinger et al.'s real-time model for short-term parking zones provides valuable insights, yet there is a research gap in understanding the adaptability of such models to

various urban settings and the integration of predictive analytics for future parking demand (Reinhard Hössinger et al., n.d. [2]).

"Sarfsense" by Sarfraz Nawaz et al. focuses on smartphone-based sensing for on-street parking, but there is a need for further investigation into the privacy and security implications of utilizing personal devices for parking management (Sarfraz Nawaz et al., n.d. [3]). Selcuk Demir et al.'s study on selecting suitable parking lot sites in megacities addresses strategic placement, yet research gaps exist in exploring the socio-economic impact of these parking lot selections on local communities and businesses (Selcuk Demir et al., n.d. [4]).

Xie et al.'s image-based parking solutions for shared bicycles highlight the emerging trend of shared mobility but lack a comprehensive exploration of potential challenges related to image quality, occlusions, and the integration of such systems with existing urban infrastructure

(Xie et al., n.d. [5]). Yanfeng Geng et al.'s work on a new "Smart Parking" system infrastructure touches on practical aspects, yet there is a research gap in understanding the long-term sustainability and maintenance challenges associated with large-scale implementation (Yanfeng Geng et al., n.d. [6]).

3.2 PROPOSED METHODOLOGY

The project envisions a solution to the prevalent issue of inefficient parking space utilization through the development of a real-time parking spot indicator system. The proposed methodology outlines a systematic approach to address the challenges associated with finding available parking spaces, particularly in high-occupancy areas.

The first phase involves a thorough understanding of the problem and the scope of the project. This includes defining the targeted parking areas and specifying the desired user experience. A comprehensive literature review follows, focusing on existing methodologies and technologies in the domain of smart parking systems. The aim is to identify strengths and weaknesses of prior approaches to ensure the proposed methodology addresses and improves upon existing drawbacks. With a clear understanding of the problem and insights from the literature review, the system architecture is designed. This encompasses the identification of suitable hardware components, such as surveillance cameras, and the definition of software components, including image processing algorithms.

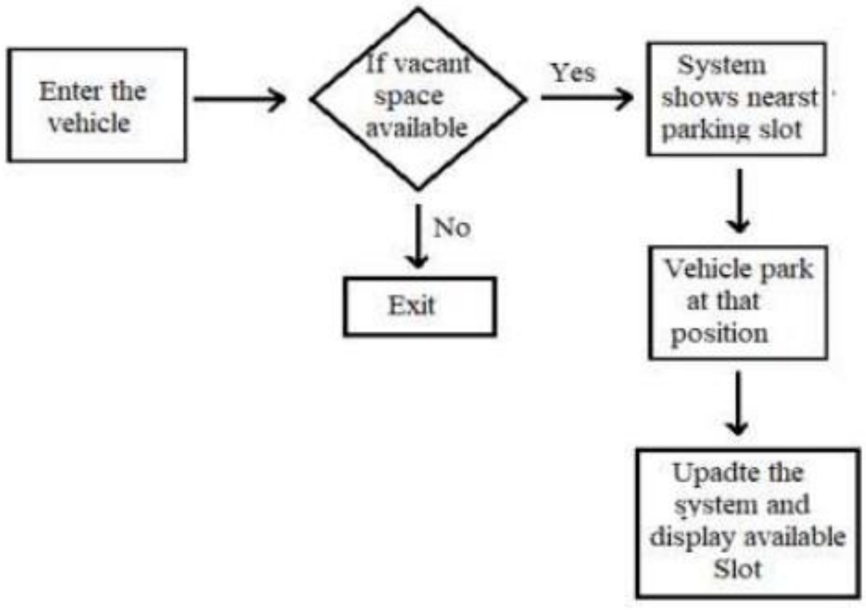

Enter the vehicle
If vacant space available
Yes
System shows nearst parking slot
No
Exit
Vehicle park at that position
Upadte the system and display available Slot

Figure 1:Flow diagram of proposed solution

The next step involves the careful selection and placement of surveillance cameras to capture comprehensive images of the parking area. Considerations are made for factors like lighting conditions, weather variations, and potential obstructions. Image preprocessing techniques are then implemented to enhance the quality and clarity of the captured images, addressing challenges such as variations in lighting, shadows, and occlusions. The core of the system lies in the development of a vehicle detection module using image processing algorithms.Techniques like object detection or background subtraction are explored to accurately identify the presence of vehicles in the captured images. Algorithms for counting available parking spaces within the detected Region of Interest (ROI) are developed, with potential integration of machine learning models for improved accuracy.

3.3 OBJECTIVES

1. Capture and Detect Vehicle Presence:

Develop an image processing technique to accurately capture and detect the existence of vehicles in a parking lot using surveillance cameras.

2. Count and Display Available Parking Space:

Implement a system that counts and displays the number of available parking spaces in real-time, providing valuable information to drivers entering the parking lot.

3. Optimized Parking:

Aim to optimize parking space utilization, ensuring that users find the best available spot, saving time and resources for both individual drivers and commercial entities.

4. Traffic Flow Improvement:

Reduce traffic congestion by providing accurate information on available parking spaces, minimizing the need for drivers to circle in search of an open spot.

5. Environmental Impact Reduction:

Address the environmental impact of parking by decreasing the time spent searching for parking, ultimately reducing vehicle emissions and environmental footprint.

6. Cost Efficiency:

Decrease management costs through automation, reducing the need for manual intervention in parking management processes.

7. Data Analytics for Future Planning:

Utilize data collected through the smart parking system for comprehensive analytics, providing insights into usage patterns, peak hours, and popular parking spots for informed urban planning.

8.Scalability and Interoperability:

Design the smart parking solution with scalability and interoperability, allowing for easy expansion to meet growing demands and ensuring compatibility with future technologies.

9.Real-Time Wrong Parking Detection:

Implement a module for real-time detection of wrongly parked cars, providing administrators with information on misplaced vehicles.

10.Web Application Development:

Develop a web application with user and admin interfaces, allowing users to access real-time parking information and administrators to monitor and manage the parking system.

4.0 SYSTEM DESIGN & IMPLEMENTATION

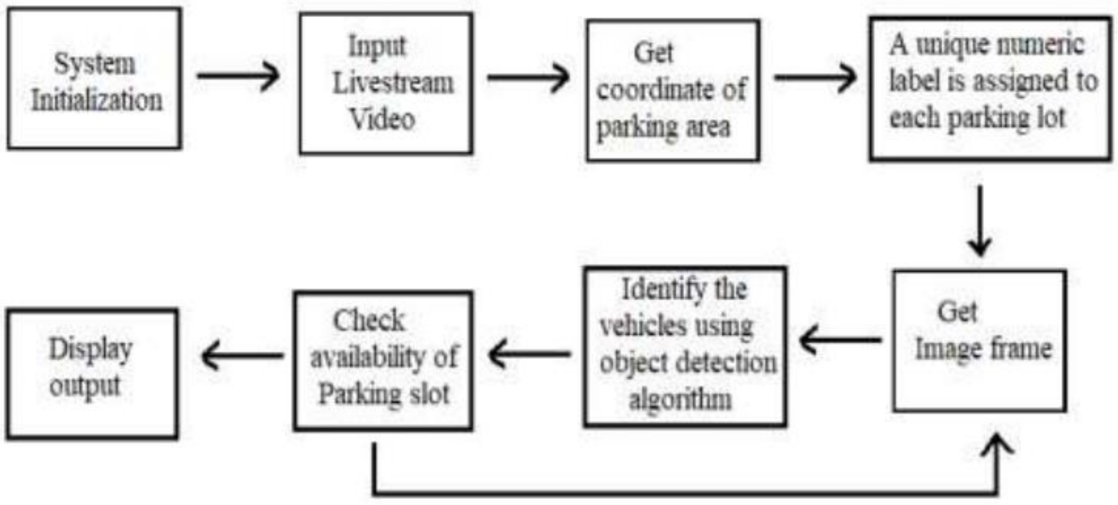
4.1 System Design:

Web-Based Interface: Develop a user-friendly web-based interface for real-time insights. Ensure responsive design for accessibility on various devices.

CV2 Module Integration: Utilize the CV2 module with neural networkpowered car detection for accurate identification and tracking of vehicles. This forms the core of the system's technological backbone.

Data Analytics: Incorporate data analytics for future urban planning. Analyze parking trends, user behavior, and traffic patterns to enhance overall urban infrastructure planning

Scalability and Interoperability: Ensure the system is scalable to adapt to evolving technological landscapes and future urban developments. Consider interoperability with other smart city systems for a holistic approach.



System Initialization
Input Livestream Video
Get coordinate of parking area
A unique numeric label is assigned to each parking lot
Get Image frame
Identify the vehicles using object detection algorithm
Check availability of Parking slot
Display output

Figure-2: Flow chart of parking system detection

4.2 Implementation:

4.2.1 Software Requirements:

• Operating system: WINDOWS XP/7/8/10

• Tool used: vscode

• Database: MYSQL

• python

• CV2 Module(OpenCV-Python is a library of Python bindings designed to solve computer vision problems)

4.2.2 Hardware Requirements:

• Processor: Intel i3

• Speed: 1.1 GHz

• RAM: 2 GB (Minimum)

• Disk space: 2 GB (Minimum)

4.2.3 Testing &Test Cases:

Rigorous testing is essential, including live testing in a parking environment.

Test the system's responsiveness, accuracy in detecting parking spaces, and the effectiveness.

Test Cases for Login Page:

• Valid Credentials: Provide a valid username and password and expect a successful login, redirecting to the home page.

• Invalid Username: Input an incorrect username with a valid password and expect an error message indicating an invalid username.

• Invalid Password: Input a valid username with an incorrect password and expect an error message indicating an incorrect password.

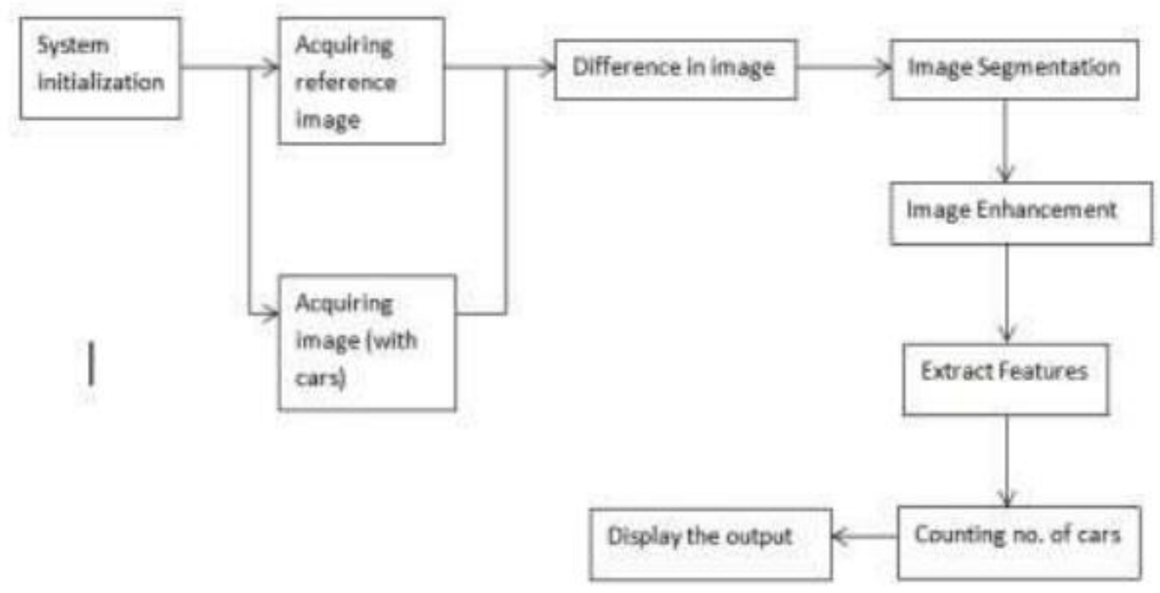


Figure-3: System Implementation

5.0 ALGORITHM OF THE PROPOSED SYSTEM

• Capture Live Video Stream:

The system initiates by obtaining a live video stream of the parking lot through a camera.

• Image Capture on Car Movement:

Images are captured dynamically when a car either enters or exits the parking lot, allowing the system to focus on relevant moments.

• Conversion to Gray Scale:

The RGB images acquired are transformed into gray scale images, simplifying subsequent processing steps.

• Calibration:

Calibration is conducted to refine the focus of the algorithm. Firstly, the system identifies and selects the coordinates defining the boundaries of the parking lot. This step effectively crops any extraneous areas beyond the parking lot, optimizing the computational resources. Secondly, the algorithm identifies and selects the coordinates of a single parking slot, effectively dividing the parking lot into uniform slots.

• Binary Conversion:

Each block within the defined parking slots undergoes a conversion from gray scale to binary. Subsequently, an inverse binary transformation is applied to accentuate the presence of a car, represented in white, against the background of the parking area in black.

This algorithmic framework ensures a systematic and efficient process for detecting parking space occupancy. By dynamically capturing and processing live video streams, converting images to gray scale, and calibrating to the specific parking lot geometry, the system optimizes the accuracy and responsiveness of parking space detection. The binary conversion further enhances the clarity of car presence within individual parking slots, facilitating effective monitoring of parking lot occupancy.

6.0 OUTCOMES

Data Analytics for Future Planning:

Utilize the data collected through the smart parking system for comprehensive analytics. This data-driven approach can offer insights into usage patterns, peak hours, and popular parking spots, enabling informed decision-making for future urban planning and infrastructure development.

Integration with Navigation Apps:

Collaborate with popular navigation applications to seamlessly integrate smart parking information. This ensures that drivers receive real-time updates on parking availability as part of their route planning, promoting a more efficient and hassle-free commuting experience.

Accessibility Features:

Incorporate features that cater to individuals with special needs, such as designated accessible parking spaces and user interfaces designed for those with disabilities. This ensures inclusivity and compliance with accessibility standards.

Scalability and Interoperability:

Design the smart parking solution with scalability and interoperability in mind. This allows for easy expansion to accommodate growing demands and ensures compatibility with future technologies and innovations in the smart city ecosystem

Social Integration for Community Engagement:

Introduce social integration features within the smart parking application, allowing users to share real-time parking availability updates with their social networks. This fosters a sense of community engagement, encouraging users to contribute to a collective effort in optimizing parking resources.

Energy-Efficient Lighting Controls:

Consider integrating the smart parking system with lighting controls based on occupancy. Implementing energy-efficient lighting that adjusts based on the number of vehicles present can contribute to sustainability goals, reducing energy consumption in the parking facility.

7.0 CONCLUSION

In conclusion, the development and implementation of our Smart Parking System, CAPSuIP (Counting Available Parking Space using Image Processing), represent a significant stride in addressing the multifaceted challenges associated with parking space availability and management. By harnessing advanced image processing techniques, CAPSuIP not only detects the presence of vehicles in parking lots but also provides real-time information on the number and location of available parking spaces.

A meticulous review of existing parking detection methods guided our approach, identifying the critical need for a cost-effective, accurate, and user-friendly solution. CAPSuIP is designed to meet these requirements, offering a simplified yet robust approach to parking management.

With well-defined objectives centered on optimizing parking, reducing traffic congestion, minimizing environmental impact, and enhancing user experience, CAPSuIP integrates features such as real-time wrong parking detection, dynamic pricing, and accessibility considerations, augmenting its functionality and usability. The experimental details and methodology provide a clear roadmap for project development, encompassing software and hardware requirements, and ensuring the execution of a technologically advanced and user-centric solution.

Anticipated outcomes, including the implementation of a dynamic pricing model, data analytics for future planning, integration with navigation apps, and scalability, underscore the forward-thinking approach embedded in CAPSuIP. These outcomes not only contribute to effective parking management but also align with broader goals of sustainability, efficiency, and inclusivity.

Looking ahead, the successful execution of this project holds the promise of a positive impact on urban mobility, offering a tangible solution to the persistent challenge of finding suitable parking spaces. By providing real-time information to drivers, reducing search times, and incorporating intelligent features, CAPSuIP exemplifies the transformative potential of technology in enhancing everyday experiences. The web application, featuring user and admin interfaces, represents a meaningful step toward creating a more connected and efficient parking ecosystem.

8.0 PSUEDOCODE

8.1 PYTHON CODE:

8.1.1) main.py

```
from flask import Flask, render_template

import csv

from flask import Flask, render_template, redirect, request, session
from flask import Flask, render_template

import firebase_admin

import random

from flask import Flask, request, jsonify

from firebase_admin import credentials, firestore, initialize_app

import os

from google.cloud.firestore_v1 import FieldFilter

from werkzeug.utils import secure_filename

import smtplib

from email.mime.text import MIMEText

from flask import render_template, session, redirect, url_for

import firebase_admin

import random

from flask import Flask, request

from firebase_admin import credentials, firestore

cred = credentials.Certificate("key.json")

firebase_admin.initialize_app(cred)

app=Flask(__name__)

app.secret_key="SlotBooking@1234"

app.config['upload_folder']='/static/upload

ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])

app.config['UPLOAD_FOLDER'] = 'static/upload'

sender = "dhanu.innovation@gmail.com"

password = "dkgppiexjwbznzcv"
```

```
def send_email(subject, body, sender, recipients, password):
 msg = MIMEText(body)
 msg['Subject'] = subject
 msg['From'] = sender
 msg['To'] = ', '.join(recipients)
 with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp_server:
 smtp_server.login(sender, password)
 smtp_server.sendmail(sender, recipients, msg.as_string())
 print("Message sent!")
@app.route('/')
def homepage():
 try:
 return render_template("index.html")
 except Exception as e:
 return str(e)
@app.route('/usermainpage')
def usermainpage():
 try:
with open('freespace.csv') as csvfile:
 csvreader = csv.reader(csvfile)
 data = [row[0] for row in csvreader]
 new_data = [30 - int(x) for x in data]
 print("Data : ", data)
 print("New Data : ", new_data)
 return render_template('mainpage.html', data=data, new_data=new_data)
 except Exception as e:
 return str(e)
@app.route('/index')
def indexpage():
 try:
 return render_template("index.html")
 except Exception as e:
 return str(e)
@app.route('/logout')
def logoutpage():
 try:
 return render_template("index.html")
 except Exception as e:
 return str(e)
@app.route('/loginpage', methods=["POST","GET"])
def loginpage():
 try:
 msg = ""
if request.method == 'POST':
 uname = request.form['uname']
 pwd = request.form['pwd']
 db = firestore.client()
 print("Uname : ", uname, " Pwd : ", pwd)
 newdb_ref = db.collection('newuser')

 dbdata = newdb_ref.get()

 data = []

 flag = False

 for doc in dbdata:

 data = doc.to_dict()

 if (data['UserName'] == uname and data['Password'] == pwd):

 flag = True

 session['userid'] = data['id']

 break

 if (flag):

 return redirect(url_for("usermainpage"))

 else:

 msg = "UserName/Password is Invalid"

 return render_template("loginpage.html", msg=msg)

 else:

 return render_template("loginpage.html", msg=msg)

 except Exception as e:

 return render_template("loginpage.html", msg=e)

@app.route('/registerpage', methods=["POST","GET"])

def registerpage():

try:

 msg = ""

 if request.method == 'POST':
```

```
print("Add New Register page")

fname = request.form['fname']

lname = request.form['lname']

uname = request.form['uname']

pwd = request.form['pwd']

email = request.form['email']

phnum = request.form['phnum']

address = request.form['address']

id = str(random.randint(1000, 9999))

json = {'id': id,

'FirstName': fname,'LastName': lname,

'UserName': uname, 'Password': pwd,

'EmailId': email, 'PhoneNumber': phnum,

'Address': address}

db = firestore.client()

newuser_ref = db.collection('newuser')

id = json['id']

newuser_ref.document(id).set(json)

msg = "New User Added Success"

return render_template("registerpage.html", msg=msg)

else:

return render_template("registerpage.html", msg=msg)

except Exception as e:

return str(e)
```

```
@app.route('/contact', methods=["POST","GET"])

def contact():

 try:

 return render_template("contact.html")

 except Exception as e:

 return str(e)

if __name__ == '__main__':

 app.run(host ="localhost", port=int(5000), debug=True)
```

8.1.2) data_training.py

```
from itertools import count

import cv2

import pickle

try:

 with open('parking_area','rb') as f:

 parking_list = pickle.load(f)

except:

 parking_list = []

width = 45

height = 150

def draw_parking(events,x,y,flags,params):

 if events == cv2.EVENT_LBUTTONDOWN:

 parking_list.append((x,y))

 if events == cv2.EVENT_RBUTTONDOWN:
```

```
for i, pos in enumerate(parking_list):
 x1,y1 = pos
 if x1<x<x1+width and y1<y<y1+height:
 parking_list.pop(i)
 with open("parking_area","wb") as f:
 pickle.dump(parking_list,f)
while True:
 image = cv2.imread("parking.jpg")
 image_resize = cv2.resize(image,(640,480))
 for pos in parking_list:
 cv2.rectangle(image_resize,pos,(pos[0]+width,pos[1]+height),(0,255,0),2)
 cv2.imshow("parking",image_resize)
 cv2.setMouseCallback("parking",draw_parking)
 cv2.waitKey(0)
```

8.1.3) parking_detection.py

```
import cv2
import pickle
import numpy as np
import csv
try:
 with open("parking_area","rb") as f:
 parking_list = pickle.load(f)
except:
 parking_list = []
def freeSpace(spaceCount):
data = [spaceCount]
 with open('freespace.csv',mode='w',newline='') as f:
 writer = csv.writer(f,delimiter=',')
 writer.writerow(data)
def checking(video_resize):
 spaceCount = 0
 for pos in parking_list:
 x,y = pos
 video = video_resize[y:y+height,x:x+width]
 count = cv2.countNonZero(video)
 if count < 1500:
 spaceCount += 1
 color = (0,255,0)
 tickness = 2
 else:
 color = (0,0,255)
 tickness = 2
 cv2.rectangle(video_pos,pos,(x+width,y+height),color,tickness)
 cv2.rectangle(video_pos,(40,30),(300,80),(180,0,180),-1)
 cv2.putText(video_pos,f'Free
Slots:{spaceCount}/{len(parking_list)}',(50,60),cv2.FONT_HERSHEY_SIMP
LEX,0.9,(255,255,255),2)
```

```
 freeSpace(spaceCount)
width = 45
height = 150
capture = cv2.VideoCapture("parking.mp4")
while True:
 if capture.get(cv2.CAP_PROP_POS_FRAMES) ==
capture.get(cv2.CAP_PROP_FRAME_COUNT):
 capture.set(cv2.CAP_PROP_POS_FRAMES,0)
 success , image = capture.read()

 video_pos = cv2.resize(image,(640,480))

 imageGray = cv2.cvtColor(video_pos,cv2.COLOR_BGR2GRAY)
 cv2.imshow("gray",imageGray)

 imageBlur = cv2.GaussianBlur(imageGray,(3,3),1)
 cv2.imshow("blur",imageBlur)

 imageThreshold =
cv2.adaptiveThreshold(imageBlur,255,cv2.ADAPTIVE_THRESH_GAUSSIA
N_C,cv2.THRESH_BINARY_INV,25,16)
 cv2.imshow("thershold",imageThreshold)

 imageMeadianBlur = cv2.medianBlur(imageThreshold,5)
```

```
cv2.imshow("mblur",imageMeadianBlur)

kernel = np.ones((3,3),np.uint8)

imageDilate = cv2.dilate(imageMeadianBlur,kernel,iterations=1)

cv2.imshow("dilate",imageDilate)

checking(imageDilate)


cv2.imshow("image",video_pos)

cv2.waitKey(10)
```

9.0 SCREENSHOTS

9.1 Main Page

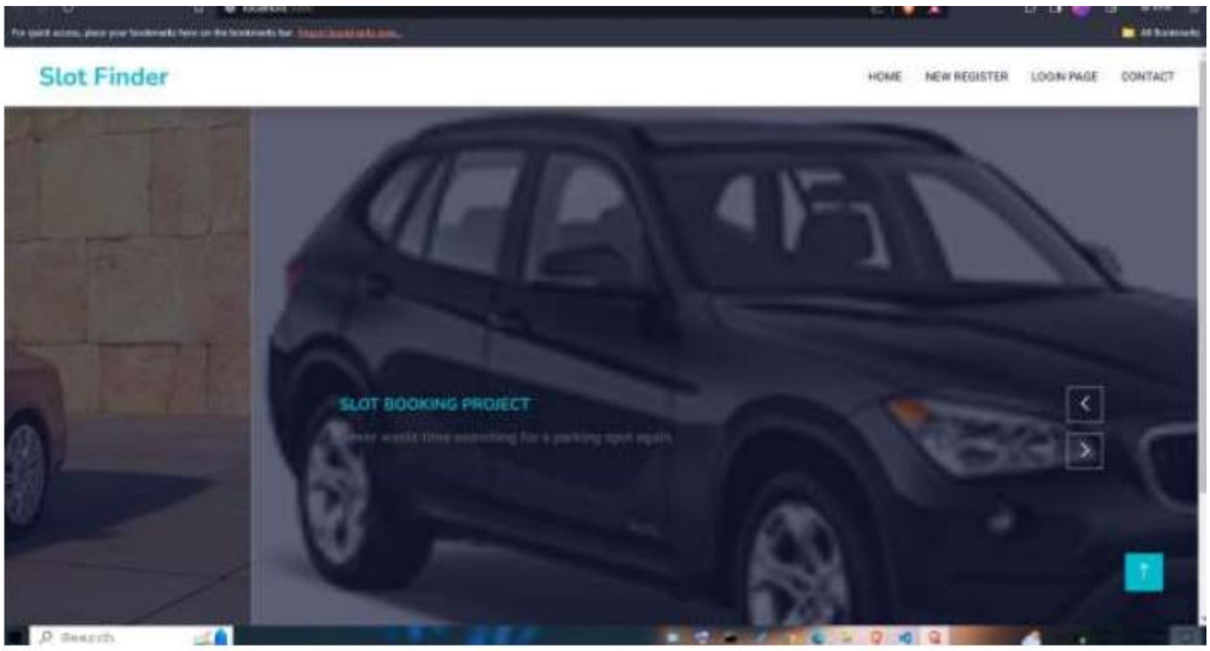


Figure-4

9.2 Login Page

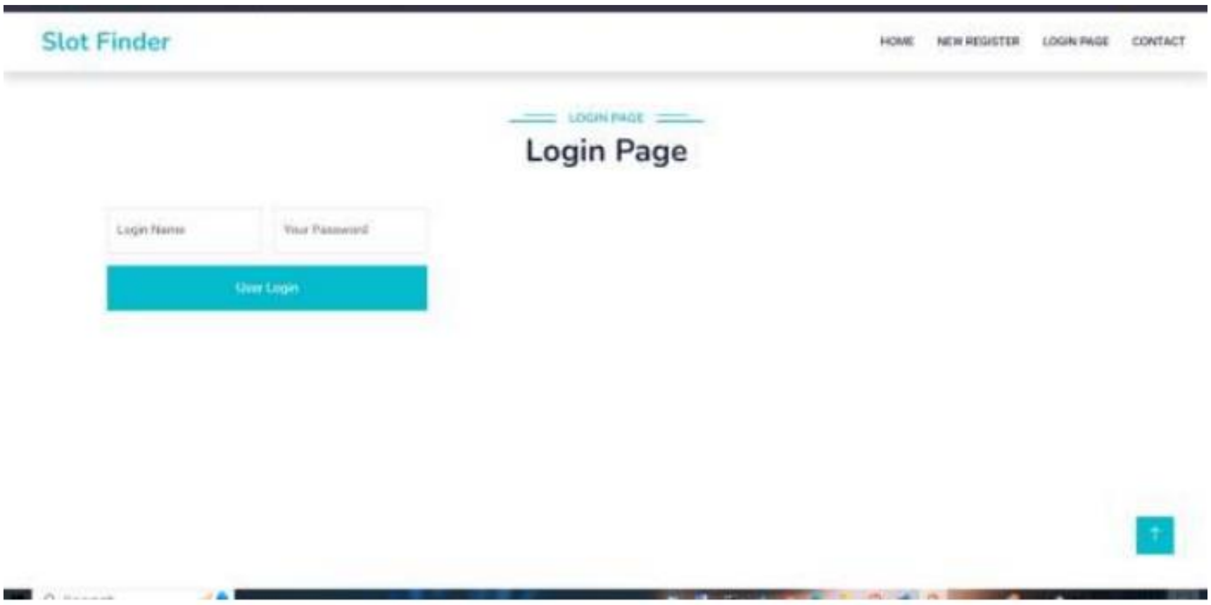


Figure-5

9.3 New Registration Page

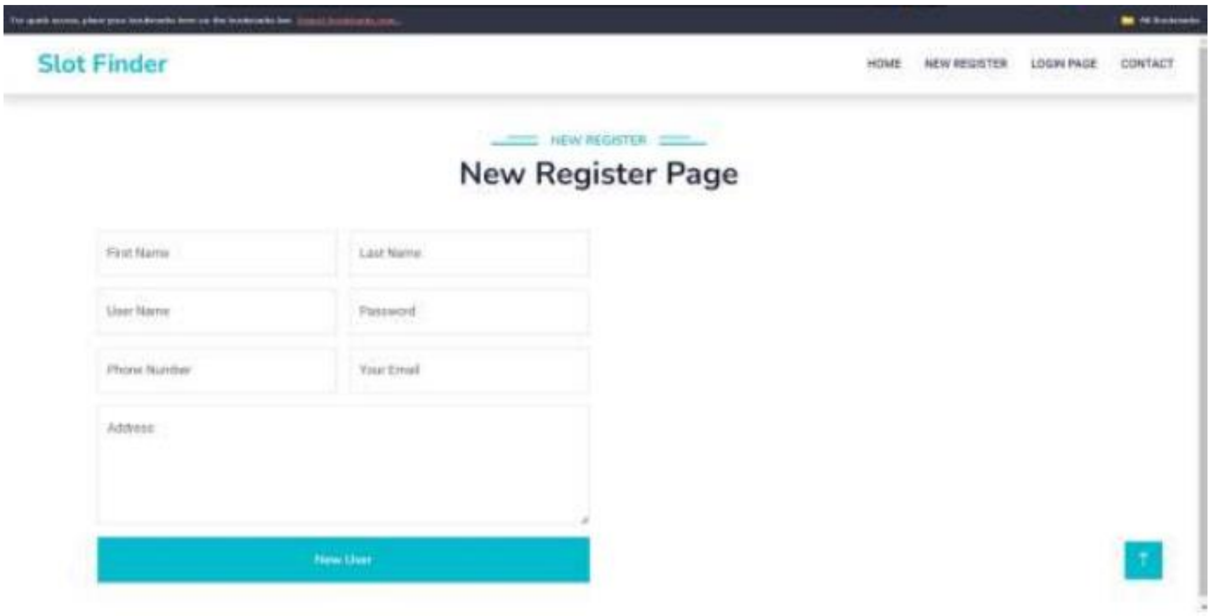
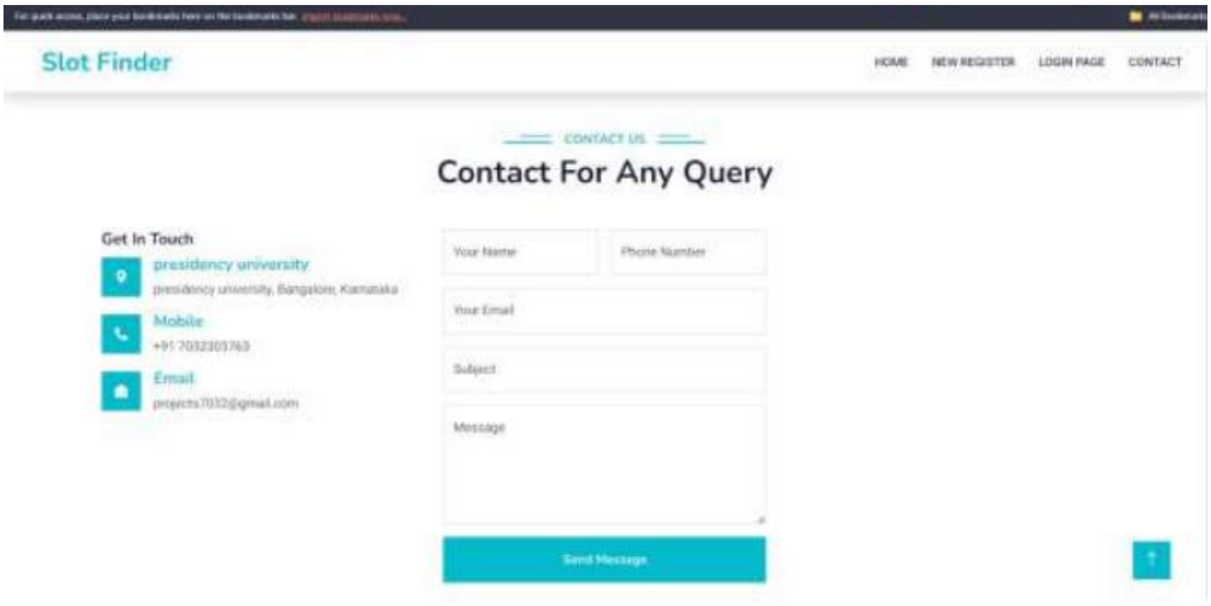


Figure-6

9.4 Contacting Page



Contact For Any Query

Get In Touch

presidency university
presidency university, Bangalore, Karnataka

Mobile
+91 7032303763

Email
projects7032@gmail.com

Your Name | Phone Number

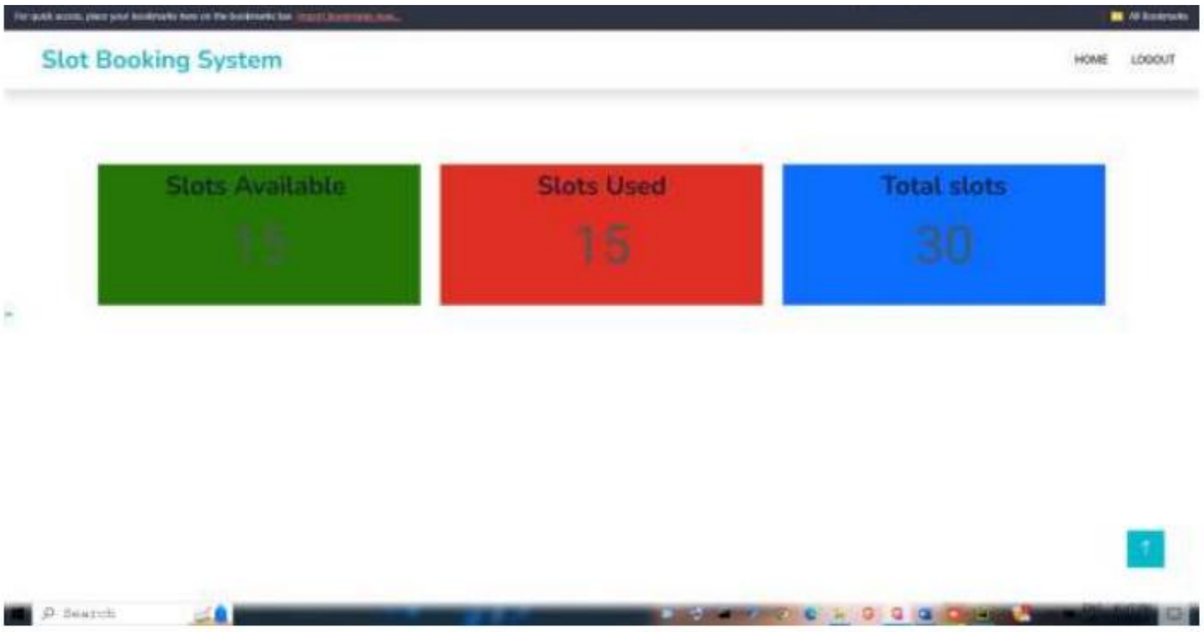
Your Email

Subject

Message

Figure-7

9.5 Dashboard



Slots Available	Slots Used	Total slots
15	15	30

Figure-8