PROJECT REPORT

ON

Parking and Space Management

A Project report submitted in partial fulfillment of requirements for the award of the Degree of

Degree of Bachelor of Technology in Computer Science & Engineering

Submitted By :

Aditi Maheshwari (University Roll No.1900270120004)

Anany Srivastava (University Roll No.1900270120007)

Arpit Rajput (University Roll No.1900270120010)

Deepak Gupta (University Roll No.1900270120018)

SUBMITTED TO:

Department of Computer Science & Engineering

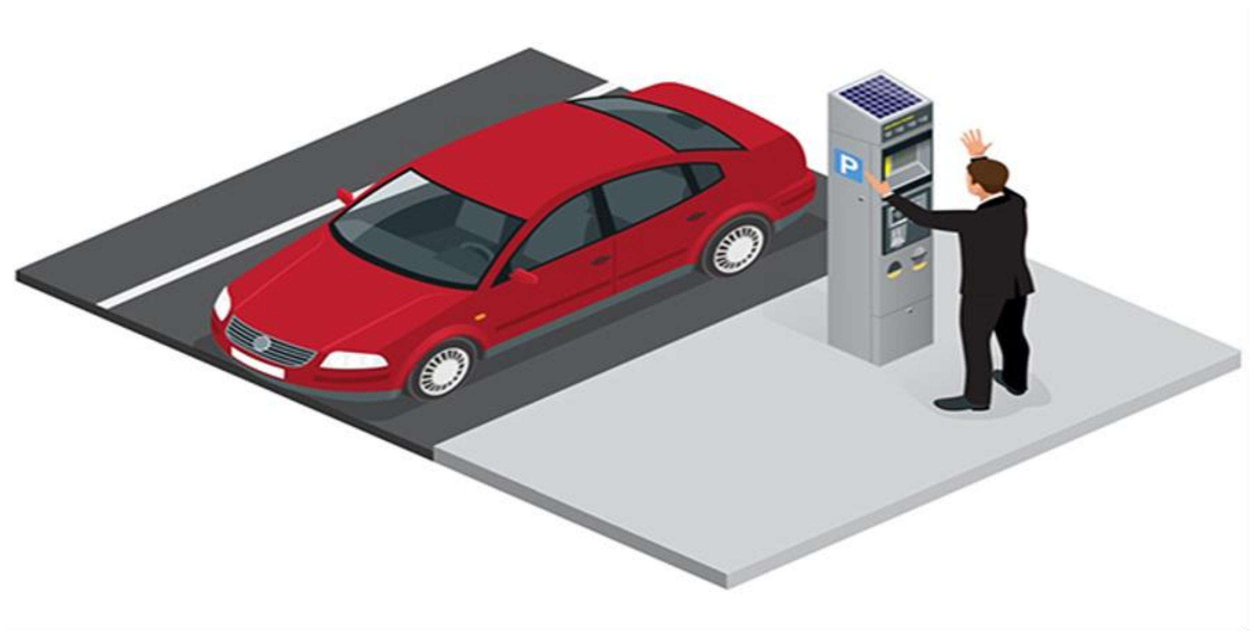
AJAY KUMAR GARG ENGINEERING COLLEGE, GHAZIABAD, UP-201009

Table of Contents :

List of Figures :

Abstract -

As urbanization continues to rise, the need for efficient and safe parking solutions has become more important. In recent years, automatic parking management systems have been developed to address this need. This paper presents a comprehensive review of automatic parking management systems. It discusses the various types of automatic parking systems, the components of an automatic parking management system, and the benefits of implementing such a system. The paper also presents a case study of an automatic parking management system implemented in a commercial building in Singapore. The case study evaluates the effectiveness of the system and highlights the challenges faced during the implementation process. The results of the study suggest that automatic parking management systems are an effective solution for managing parking in urban areas.



Chapter 1: Introduction

Parking has become a major issue in urban areas due to the increasing number of vehicles. Finding a parking space can be a challenging task for many people, and it often leads to congestion on the roads. This problem can be solved by implementing an automatic parking management system. Automatic parking management systems use technology to manage parking spaces and make the process of finding a parking spot more efficient.

The purpose of this research paper is to provide a comprehensive review of automatic parking management systems. The paper will discuss the various types of automatic parking systems, the components of an automatic parking management system, and the benefits of implementing such a system. The paper will also present a case study of an automatic parking management system implemented in a commercial building in Singapore. The case study will evaluate the effectiveness of the system and highlight the challenges faced during the implementation process.

Types of Automatic Parking Systems:

There are several types of automatic parking systems available in the market. Some of the most common types of automatic parking systems are:

1. **Fully Automated Parking System:** This type of system uses sensors to detect the presence of a vehicle and then moves it to a designated parking spot using a robotic arm. This system is ideal for areas where space is limited.

2. **Semi-Automated Parking System:** This type of system requires the driver to park the vehicle on a platform, which is then moved to a designated parking spot using a robotic arm.

3. **Mechanical Parking System:** This type of system uses a series of lifts, conveyors, and turntables to move the vehicle to a designated parking spot.

Components of an Automatic Parking Management System:

An automatic parking management system consists of several components. Some of the most important components of an automatic parking management system are:

1. **Parking Space Detection System:** This system uses sensors to detect the presence of a vehicle in a parking spot.
2. **Parking Guidance System:** This system guides the driver to an available parking spot using LED lights or signage.
3. **Payment System:** This system allows the driver to make payment for parking using a variety of payment methods, such as credit card, cash, or mobile payment.
4. **Parking Management System**: This system manages the parking spaces and provides real-time information on the availability of parking spots.

• Background and Context:

The concept of automated parking systems dates back to the early 1900s when the first automated parking garage was built in Paris. The garage used a series of lifts and conveyors to move vehicles to a designated parking spot. However, the system was not fully automated and required human operators to operate the machinery.

In the 1920s, a fully automated parking system was developed in the United States. The system used a series of lifts, conveyors, and turntables to move vehicles to a designated parking spot. The system was designed to be used in urban areas where space was limited.

In the 1950s, a mechanical parking system was developed in Japan. The system used a series of lifts and turntables to move vehicles to a designated parking spot. The system was designed to be used in high-density urban areas where space was at a premium.

In the 1970s, the first fully automated parking garage was built in Germany. The garage used a series of sensors and robotic arms to move vehicles to a designated parking spot. The system was designed to be used in areas where space was limited and parking was in high demand.

In the 1990s, automated parking systems became more popular in Japan. The systems were used in commercial buildings, shopping centers, and residential buildings. The systems were designed to be user-friendly and easy to operate.

In the 2000s, automated parking systems became more popular in other parts of the world, including Europe and North America. The systems were used in a variety of settings, including airports, train stations, and public parking garages.

Today, automated parking systems continue to evolve and improve. New technologies, such as autonomous vehicles and artificial intelligence, are being integrated into automated parking systems to make them more efficient and user-friendly. As urbanization continues to rise, automated parking systems will play an increasingly important role in managing parking in urban areas.

Developments in automated parking systems:

Automated parking systems have come a long way since the early 1900s, and today, they incorporate advanced technologies to provide more efficient and user-friendly parking solutions. Here are some of the recent developments in automated parking systems:

1. **Smart Parking Systems:** Smart parking systems use advanced technologies such as IoT (Internet of Things), cloud computing, and artificial intelligence to manage parking spaces. These systems use sensors and cameras to detect available parking spaces and provide real-time data to users. Smart parking systems also provide guidance to drivers, allowing them to quickly and easily find a parking spot.

2. **Robotic Parking Systems:** Robotic parking systems use robotic arms and conveyors to park vehicles. These systems are fully automated and require no human intervention. Robotic

parking systems are ideal for areas where space is limited and parking is in high demand.

3. **Automated Valet Parking:** Automated valet parking systems use autonomous vehicles to park and retrieve cars. These systems are currently being tested in several locations around the world, and they have the potential to revolutionize the parking industry.

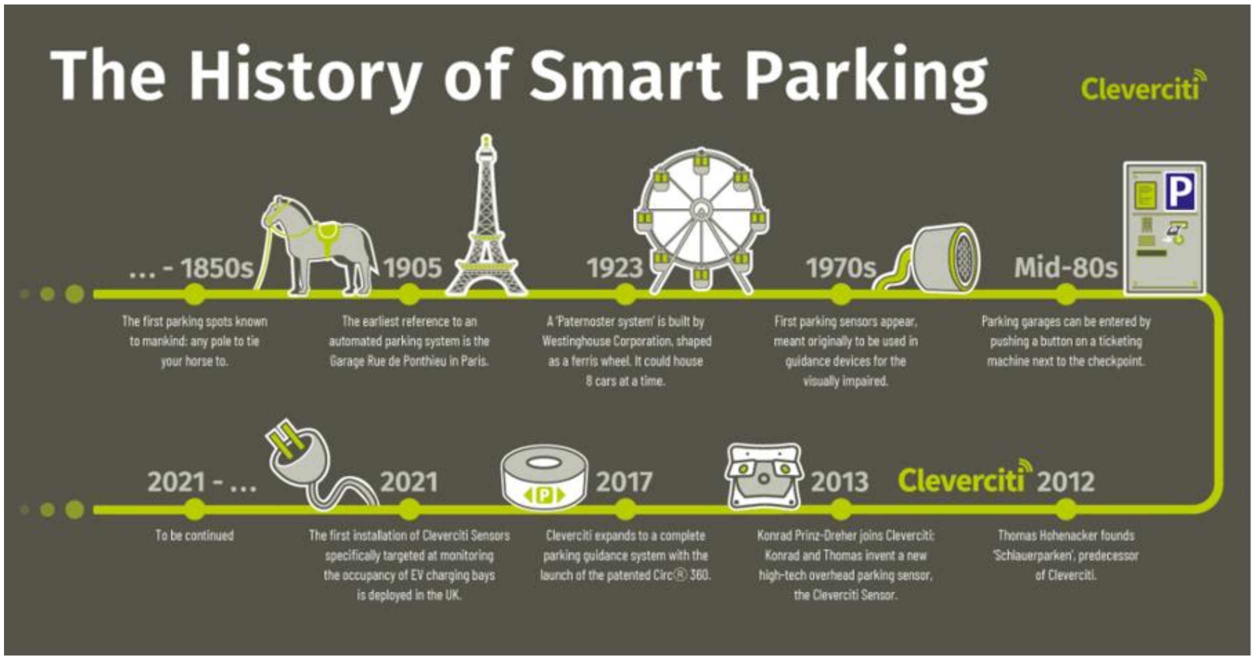
4. **Multi-Level Parking Systems**: Multi-level parking systems use a combination of lifts, conveyors, and turntables to park vehicles in multiple levels. These systems are ideal for areas where space is limited and parking is in high demand.

5. **Solar-Powered Parking Systems:** Solar-powered parking systems use solar panels to generate energy, which is then used to power the parking system. These systems are environmentally friendly and can help reduce energy costs.

6. **Mobile Parking Applications:** Mobile parking applications allow users to find and reserve parking spaces in real-time using their smartphones. These applications provide information on parking availability, pricing, and location, making it easier for drivers to find a parking spot.

Conclusion:

Automated parking systems have evolved significantly over the years, incorporating advanced technologies to provide more efficient and user-friendly parking solutions. These systems offer several benefits, including improved parking efficiency, increased revenue, reduced congestion, and enhanced security. As urbanization continues to rise, automated parking systems will play an increasingly important role in managing parking in urban areas.


The History of Smart Parking
Cleverciti
... - 1850s
The first parking spots known to mankind: any pole to tie your horse to.
1905
The earliest reference to an automated parking system is the Garage Rue de Ponthieu in Paris.
1923
A 'Paternoster system' is built by Westinghouse Corporation, shaped as a ferris wheel. It could house 8 cars at a time.
1970s
First parking sensors appear, meant originally to be used in guidance devices for the visually impaired.
Mid-80s
Parking garages can be entered by pushing a button on a ticketing machine next to the checkpoint.
2021 - ...
To be continued
2021
The first installation of Cleverciti Sensors specifically targeted at monitoring the occupancy of EV charging bays is deployed in the UK.
2017
Cleverciti expands to a complete parking guidance system with the launch of the patented Circ® 360.
2013
Konrad Prinz-Dreher joins Cleverciti: Konrad and Thomas invent a new high-tech overhead parking sensor, the Cleverciti Sensor.
Cleverciti 2012
Thomas Hohenacker founds 'Schlauerparken', predecessor of Cleverciti.

• Purpose and Objective:

This software will be extending to all types of parking having cameras and would be specifically focusing on large parking's of MNCs as well as that of shopping malls. Besides parking it will be extending its roots in the warehouses of various Multinationals where efficient storage is a major issue for the higher turnover.

The project can be further extended to private parking as well when camera functioning is proper and certain costing are reduced with the help of advanced technologies and sensors.

Scope and Limitations of the Project:

While Automatic License Plate Reading (ALPR) technology has been successful in recognizing and reading license plates, there are some common problems that can affect the accuracy of the system. Here are some of the main problems in license plate reading:

1. **Poor Image Quality:** The accuracy of ALPR systems depends on the quality of the license plate image captured by the camera. Poor lighting conditions, image blur, and reflection can all affect the image quality and make it difficult for the software to read the license plate.

2. **License Plate Obstruction:** License plates can be obstructed by objects such as bike racks, trailer hitches, or even dirt or snow. When the license plate is partially or completely obstructed, it can be difficult for the software to read the plate accurately.

3. **License Plate Location:** The location of the license plate on a vehicle can also affect the accuracy of the system. If the license plate is located in an unusual position, such as on the front bumper or under a spoiler, it may be difficult for the software to recognize the plate.

4. **License Plate Style:** Different states and countries have different license plate styles and formats, which can affect the accuracy of the software. Some plates may have non-standard fonts or characters, making them more difficult to recognize.

5. **Vehicle Speed**: ALPR systems are designed to read license plates at high speeds, but excessive speed can still affect the accuracy of the software. If the vehicle is moving too quickly, the software may not have enough time to capture a clear image of the license plate.

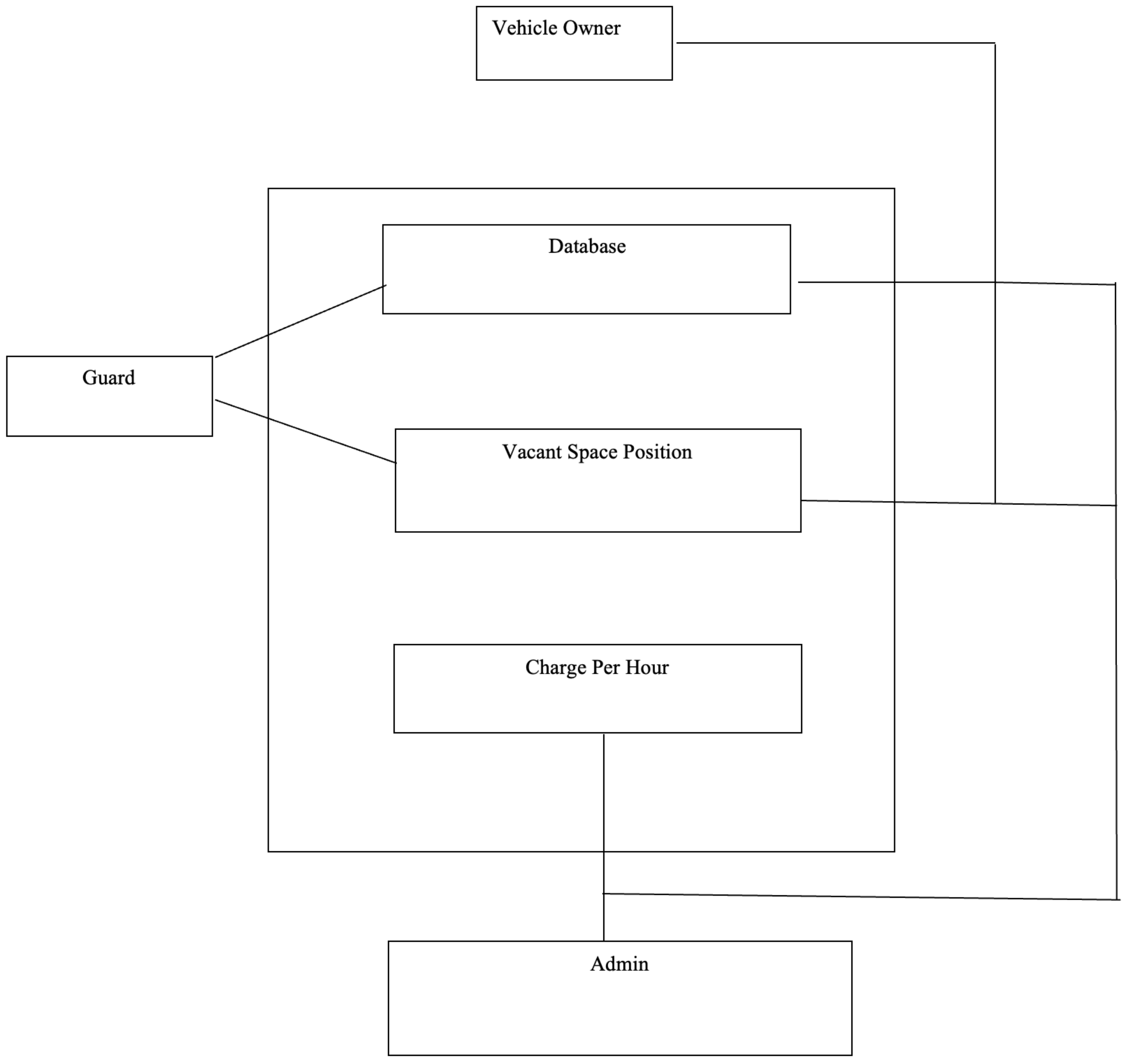
6. **Database Accuracy:** The accuracy of the database used for comparison can also affect the accuracy of the ALPR system. If the database is outdated or incomplete, it may not be able to match the license plate accurately.

In conclusion, there are several challenges that can affect the accuracy of the ALPR system, including poor image quality, license plate obstruction, unusual plate locations, license plate styles, vehicle speed, and database accuracy. These problems can be mitigated by using high-quality cameras, updating the software regularly, and ensuring that the database is accurate and up-to-date.

Chapter 2 : Software Requirements Specifications(SRS)

2.0 Overall Description :

2.1 System Environment :



The Parking and Space Management has three active actors and one cooperating system.

The Vehicle owner, Guard accesses the database only through a secured channel that is via the system installed. Any vehicle owner can access the database having the required details of the owned vehicle through the system installed in the premises only.

The Admin or the Developer has the secured right to access the charge and database of the system for any type of updating or correction.

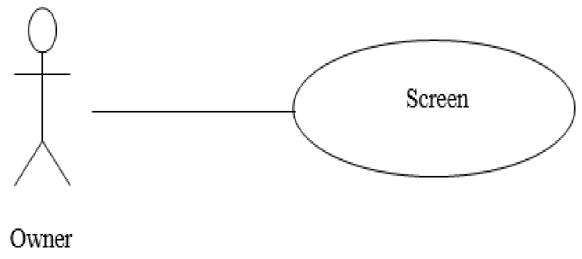
2.2 Functional Requirements Specification

This section outlines the use cases for each of the active users separately. The vehicle owner has two use cases, Guard has one test case and the admin is the supreme user of the software.

2.2.1 Vehicle Owner Use Case

Use case: Providing details

Diagram :



Brief Description

The vehicle owner gets his/her vehicle number plate or license plate identified by the system two times firstly when the vehicle enters and second time when the vehicle exits.

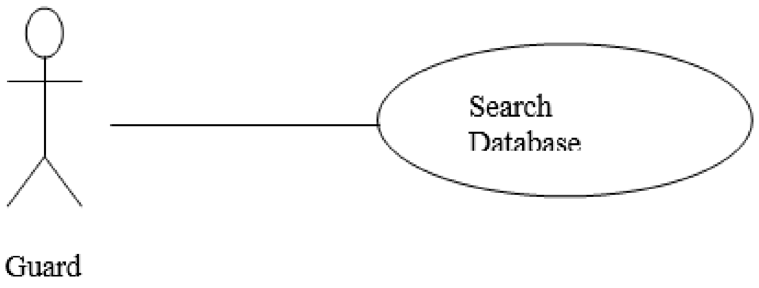
ınitial Step-By-Step Description

1. The vehicle owner drives through the driveway
2. The owner sees the system near barricade.
3. The owner selects whether he/se is new or not.
4. If the owner selects ‘new’ the details provided by the owner are stored in database.
5. If the owner selects ‘existing’ the details are searched and deleted.
6. The system shows the vacant space position on screen.

2.2.2 Guard Use Case

Use case: Search Database

Diagram:



Brief Description

The guard searches for the exiting details of the vehicle via license plate.

Initial Step-By-Step Description

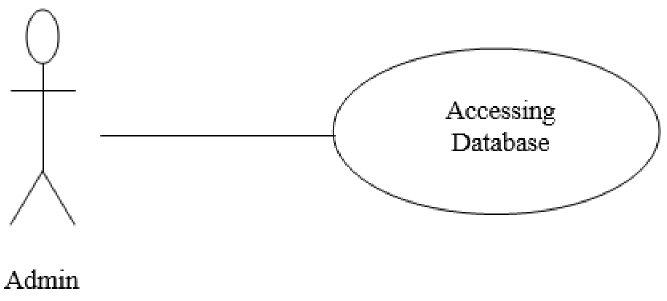
Before this use case can be initiated, the vehicle to be searched should be present in the database.

1. The guard choses his/her option.
2. The System requests for the license plate to be searched.
3. The guard enters or provides the desired license plate and searches for it.
4. The System searches in the database and if found details are printed.

2.2.3 Admin Use Case

Use case: Access database

Diagram:



Brief Description

The admin has principal control over the database and can make any type of updating to it as well.

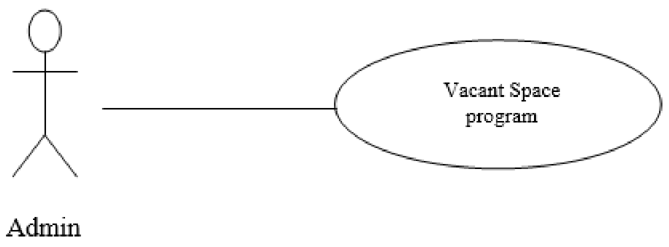
Initial Step-By-Step Description

Before this use case can be initiated, the admin must be aware of the password for accessing the database.

1. The system shows the starting options.
2. The admin selects the admin option.
3. The system asks for the admin password and if password is entered correct then it further shows the option.
4. The admin further selects the option for the database accessing and does the required functioning.

Use case: Accessing vacant space position

Diagram:



Brief Description

The author or developer accesses the program for the vacant space for any type of updation or functioning.

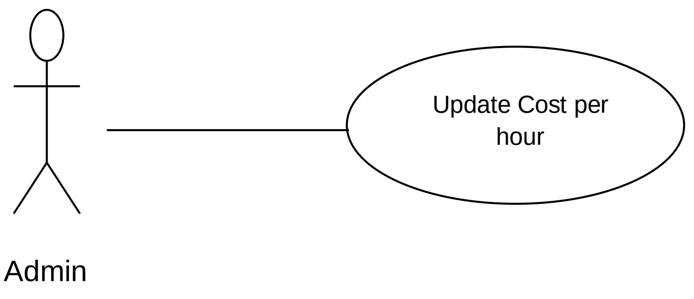
Initial Step-By-Step Description

Before this use case can be initiated, the Admin must know the password to open Admin options.

1. The system shows the starting options.
2. The admin selects the admin option .
3. The system asks for the admin password and if password is entered correct then it further shows the option.
4. The admin further accesses the vacant space program for updating it.

Use case: Update Cost of per hour

Diagram:



Brief Description

The admin updates the cost per hour for the parking.

Initial Step-By-Step Description

Before this use case can be initiated, the admin must know the password to open Admin options.

1. The system shows the starting options.
2. The admin selects the admin option.
3. The system asks for the admin password and if password is entered correct then it further shows the option.
4. The admin further accesses the cost per hour and simply updates it .

2.3 User Characteristics

The Vehicle owner is expected to be English literate and be able to use a touch screen and use buttons. The main screen of the system will have three options for the respective users which are selected as per the user.

The Guard is expected to interact and be able to use buttons operate systems swiftly.

The admin is expected to be Windows literate and to be able to use button, pull-down menus, and similar tools.

2.4 Non-Functional Requirements

The Vehicle database will be on a server with high-speed Internet capability. The physical machine to be used will be determined by the company but should have a minimum of windows XP installed.. The software developed here assumes the use of a tool such as PyCharm or any other IDE for smooth functioning. The speed of the Vehicle owner connection will depend on the hardware used rather than characteristics of this system.

The admin can access the database from his or her own PC depending on the speed of the respective system.

3.0. Requirements Specification

3.1 External Interface Requirements

The System gets activated whenever an input is detected which then shows the welcome screen which consists of three buttons for different users.

The user touches the respective option and then moves further as per the program.

It consists of a backend database which can be superiorly accessed by the admin only by providing the right admin password ensuring the privacy and confidentiality of the data.

3.2 Functional Requirements

The Logical Structure of the Data is contained in Section 3.3.1.

3.2.1 Providing Details

Use Case Name	Providing Details
Trigger	The Vehicle owner enters the premises.
Precondition	The System is started for use.
Basic Path	1. The Vehicle owner chooses the vehicle owner option on the starting screen. 2. The vehicle owner then chooses whether he/she is a new or an existing owner. 3. The owner selects the new vehicle option and then provides the desired details. 4. The data is then stored in database

Alternative Paths	In step 2, if the Owner chooses to be an existing user . 3. The Owner is provided with an option to delete his/her details on departure or not 4. The system then acts accordingly and returns to step 4. In step 2, if the Owner selects the new user option and leaves any field blank then the owner is asked to fill the specific field.
Postcondition	The database is updated.
Exception Paths	The Owner may exit the system or may break the barrier.
Other	The Owner information includes name, phone number address, email and license plate scanned.

3.2.2 Search Database

Use Case Name	Search Database
Trigger	The user selects guard as an option on the welcome screen.
Precondition	The details to be searched are present.
Basic Path	This user selects the guard option in the welcome screen which initiates system to request for the car details of the specific license plate .
Alternative Paths	If the license plate entered is not present in database it will request for the right license plate.
Postcondition	The details are shown
Exception Paths	The attempt may be abandoned at any time.
Other	None

3.2.3 Access database

Use Case Name	Access Database
Trigger	The admin selects the option to access database.
Precondition	The admin enters the right password to enter in the admin options.
Basic Path	1. The admin selects the access database option. 2. The system then asks for the type of updation to be made or only to show the database. 3. Any updation is further displayed in database.
Alternative Paths	If in step 2, if only view option is selected then no updation is made and simply database is displayed.
Postcondition	The admin has updated the database.
Exception Paths	The admin may abandon the operation at any time.
Other	None.

3.2.4 Access vacant positioning

Use Case Name	Access vacant positioning
Trigger	The admin selects for the parking space position in the admin options.
Precondition	The admin enters the right password.
Basic Path	1. The admin selects the parking space position option. 2. The system then confirms for the option and shows the parking space at the current time. 3. Any updation made by Admin is further reflected.
Alternative Paths	In step 2 if the confirmation is not given then the system moves back to the main menu and starts again from the step1 .
Postcondition	The admin is provided with the authorization to view and make any changes.

Exception Paths	The admin may abandon the operation at any time.
Other	None

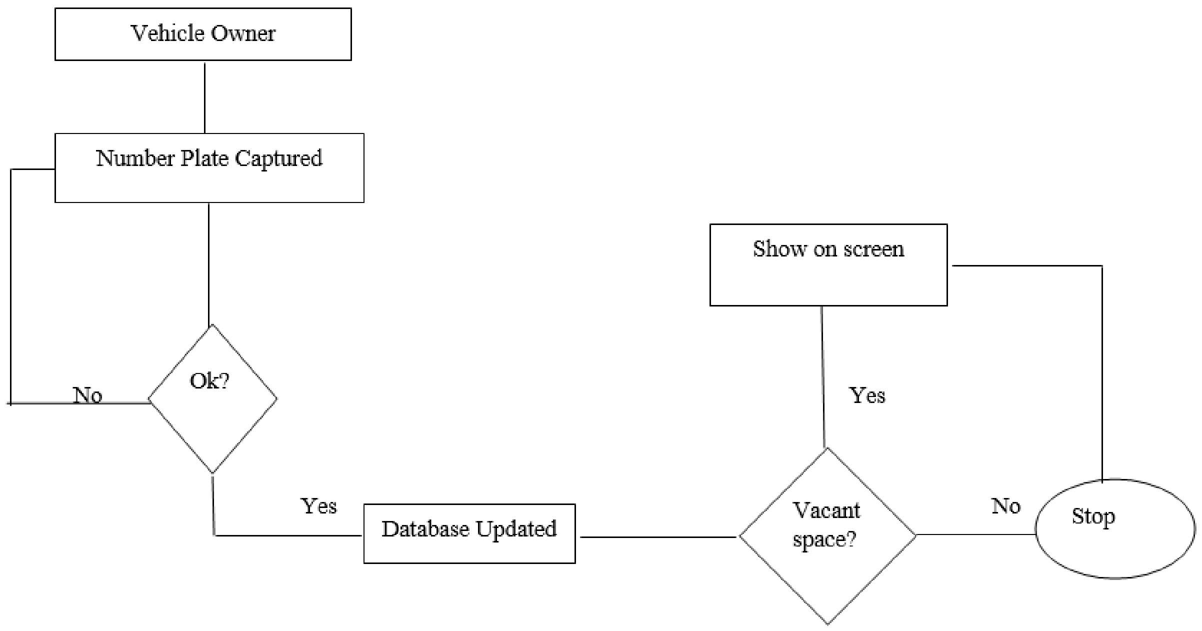
3.2.5 Update Cost per hour

Use Case Name	Update Cost per hour
Trigger	The admin selects for the updation of cost per hour option in admin options.
Precondition	The admin enters the right password.
Basic Path	1. The admin selects the Update cost per hour option. 2. The system then shows the current cost per hour. 3. The updation is made by admin and stored further in the system.
Alternative Paths	In step 3, if no updation is made then the previous value is only taken.
Postcondition	The cost per hour charge is updated.
Exception Paths	The admin may abandon the operation at any time.
Other	None.

3.3 Detailed Non-Functional Requirements

3.3.1 Logical Structure of the Data

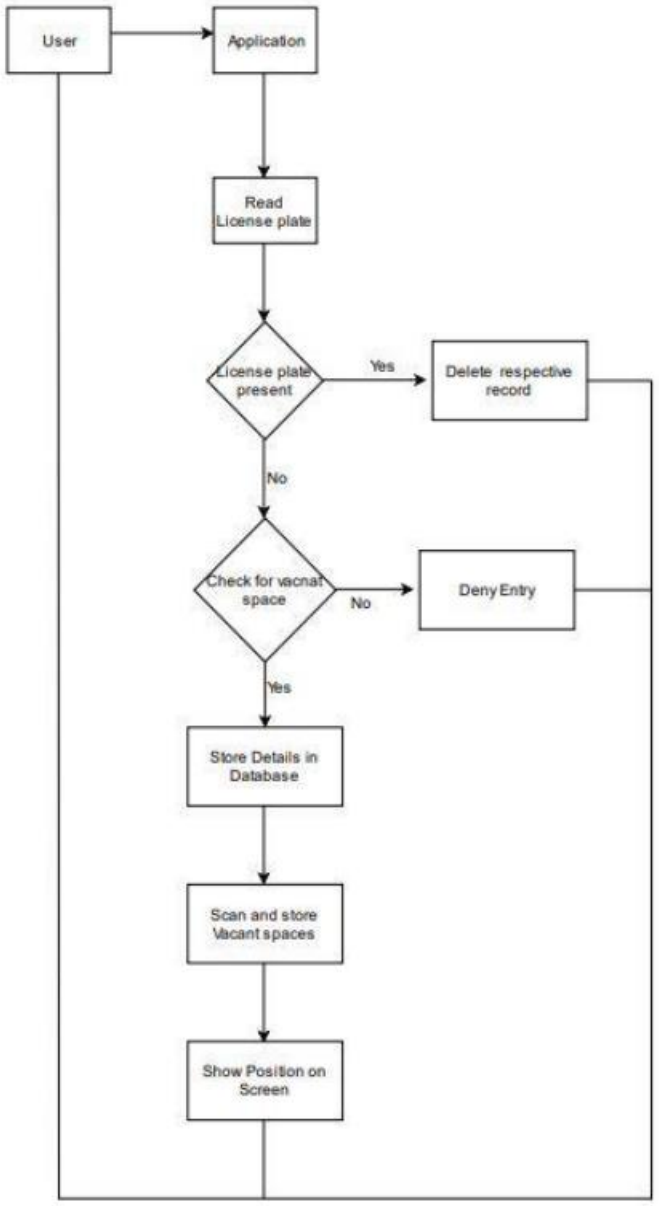
The logical structure of the data to be stored in the database is given below.



Logical Structure of the data storage

Chapter 3 : Software Design Document (SDD)

- **Deployment Diagram –**



Deployment Diagram

This project is basically aimed towards finding and allocating appropriate parking spaces to vehicles in a given area efficiently. It uses various domains of Artificial Intelligence such as- Computer Vision, Object Recognition and Object Tracking.

The block diagram for the project is shown in Figure 1. The purpose of this project is to develop such a system using artificial intelligence which reduces human interaction and somewhat makes the parking process easy and efficient to handle. This project can also be extended for proper management of goods in a warehouse.

The work involves the following:

1. **Number Plate Recognition and reading** - A fixed camera adjusted at a suitable location will click the picture of the number plate of the vehicle which would be read and stored accordingly.

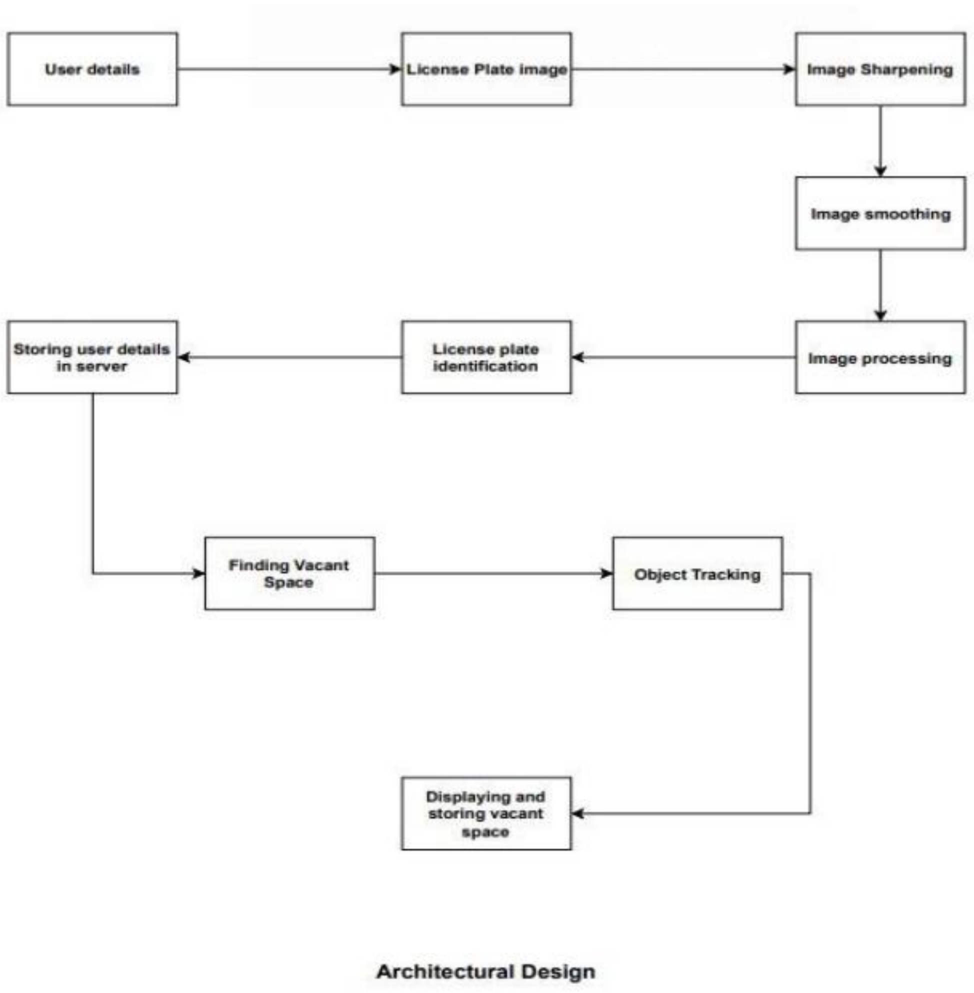
2. **Storing the details into the database** - The number plate of the vehicle along with other details of the vehicle owner such as - name, phone number etc. will be stored in the database.

4. **Finding vacant space** - Efficient object tracking algorithms will be used to find the best and most efficient space for the vehicle in the parking lot.

● Architectural Design

1. System Architecture –

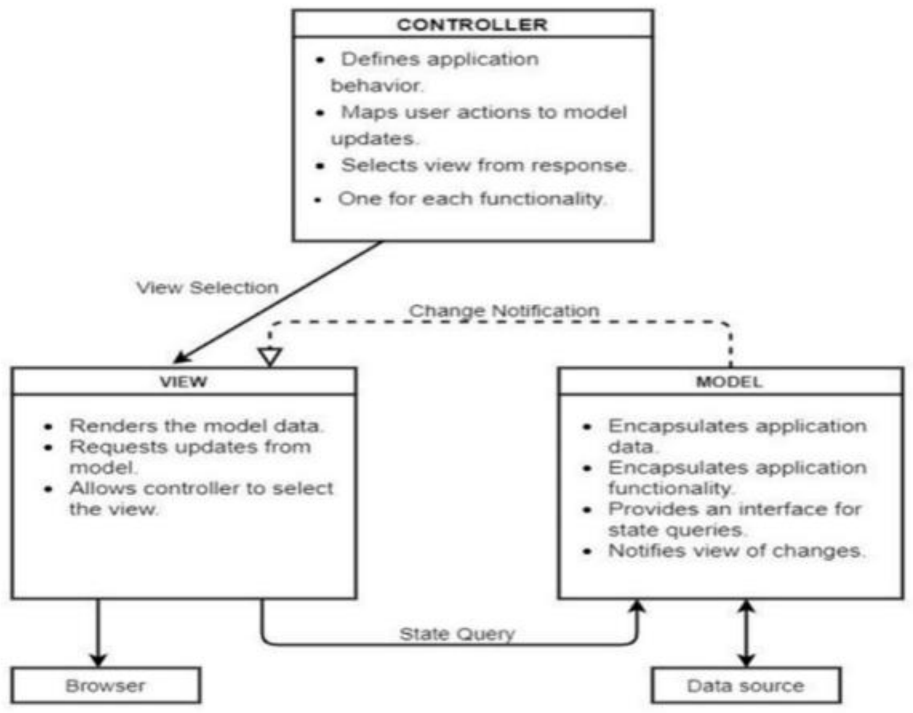
The architecture of the proposed system is as displayed in the figure below. The major components of the architecture are as follows: user database, license plate image, image sharpening, image smoothing, image processing, license plate identification, storing user detail in server, object tracking and displaying and storing vacant space.



Architectural Design

2. Model View Controller Architecture for GUI

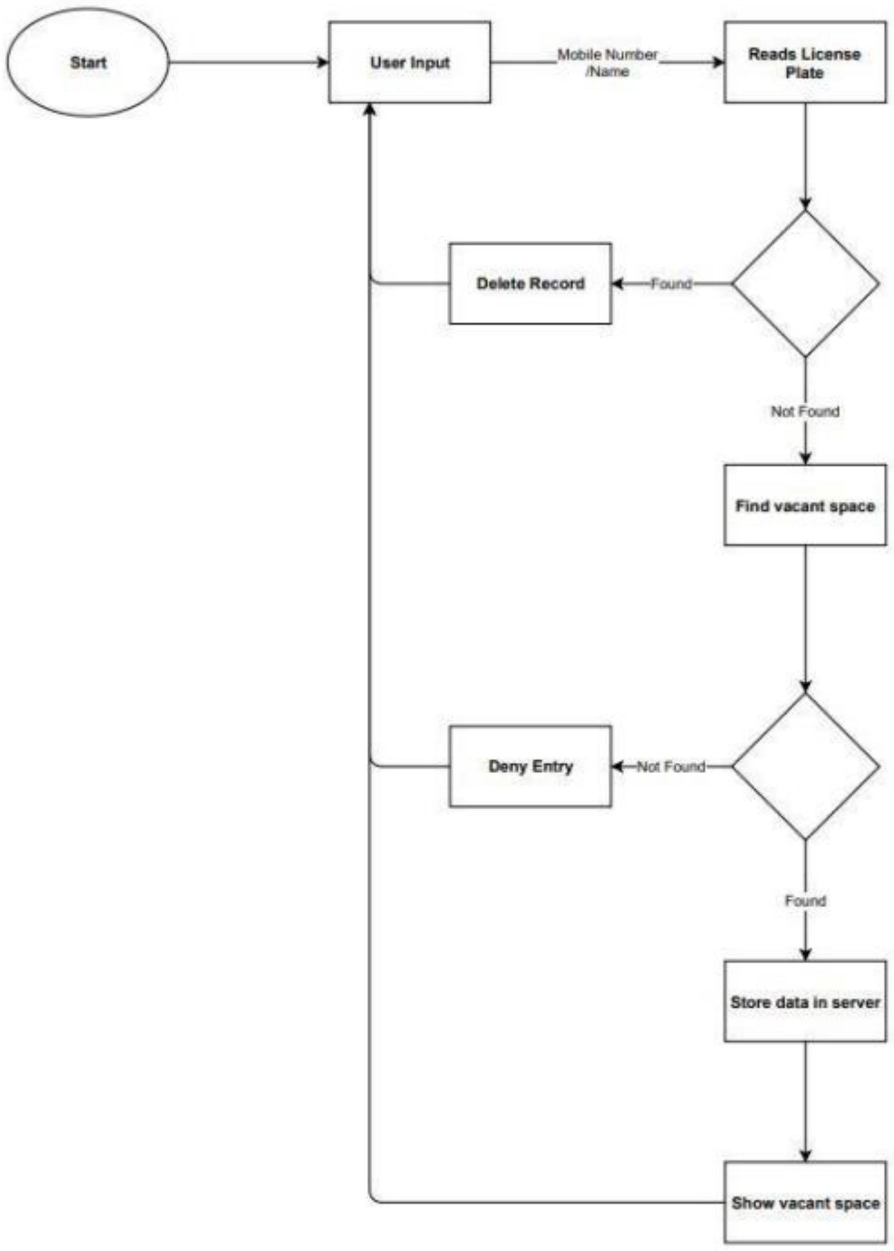
The following figure describes the process of MVC architecture. Here there are three components which are model, view and controller. The model component is responsible for the operation and management of the data. The view component is responsible for the presentation of the data to the user. The controller component is responsible for user interaction.



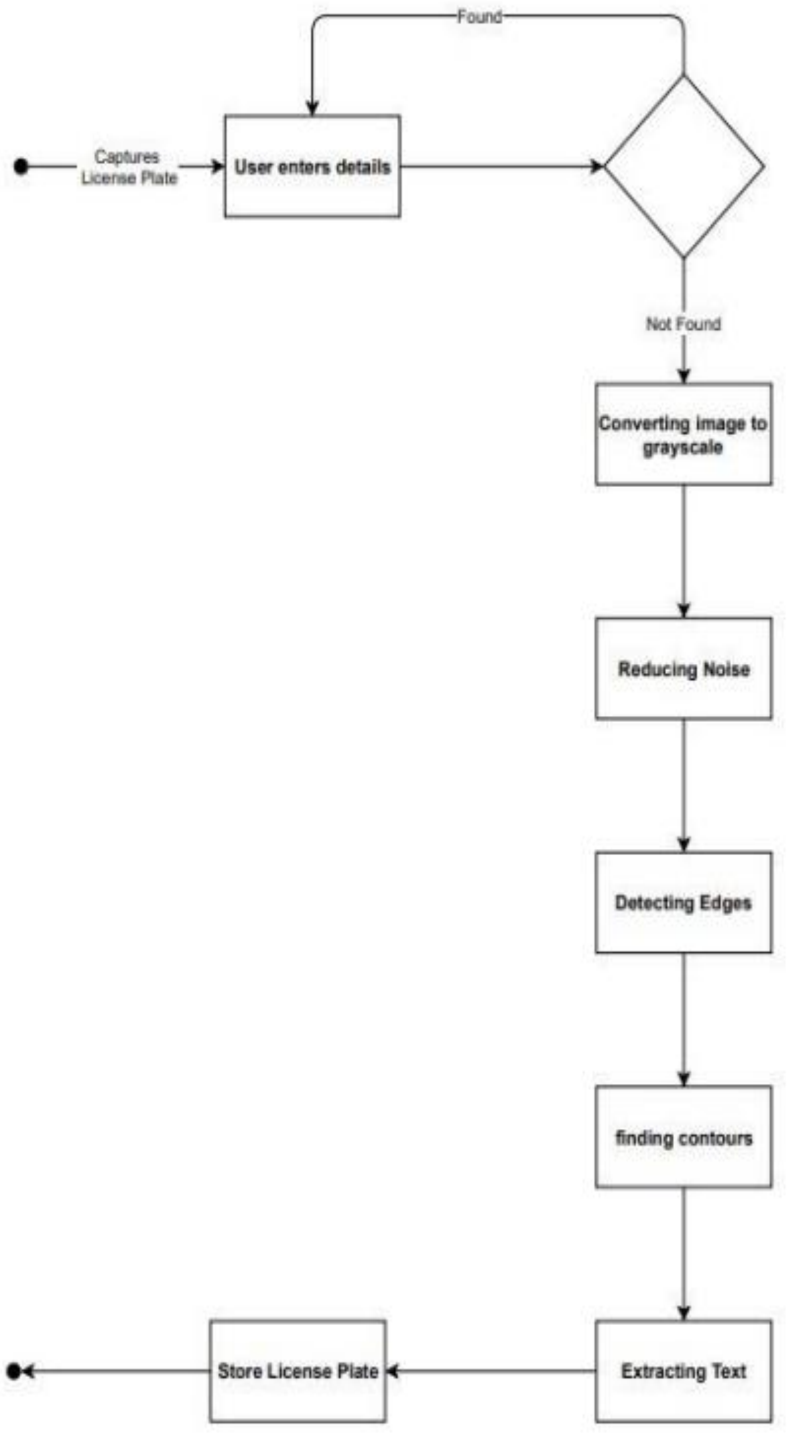
CONTROLLER
• Defines application behavior.
• Maps user actions to model updates.
• Selects view from response.
• One for each functionality.
View Selection
Change Notification
VIEW
• Renders the model data.
• Requests updates from model.
• Allows controller to select the view.
MODEL
• Encapsulates application data.
• Encapsulates application functionality.
• Provides an interface for state queries.
• Notifies view of changes.
State Query
Browser
Data source

Model View Controller Architecture

3. Activity Diagram –



Activity Diagram(User Entry)



Activity Diagram(License Plate Reading)

The activity diagram for user login and license plate reading is shown in the above figures respectively. The diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. For example, if a user wants to login, the input in this activity will be the user details such as - user’s name, mobile number etc. and image of license plate. Once the license plate is read, an object tracking algorithm is executed to find a vacant space and if the vacant space is found, the data entered is stored in the server and vacant space is displayed on the screen. If vacant space is not found, entry is denied. If after reading, the respective record is found, the record is deleted.

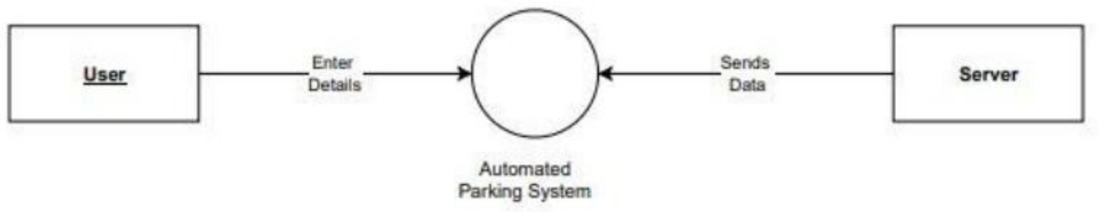
4. Data Flow Diagram -

The entire working or the flow of the data can be divided into three groups for better understanding.
They are:-

1. DFD-L0
2. DFD-L1
3. DFD-L2

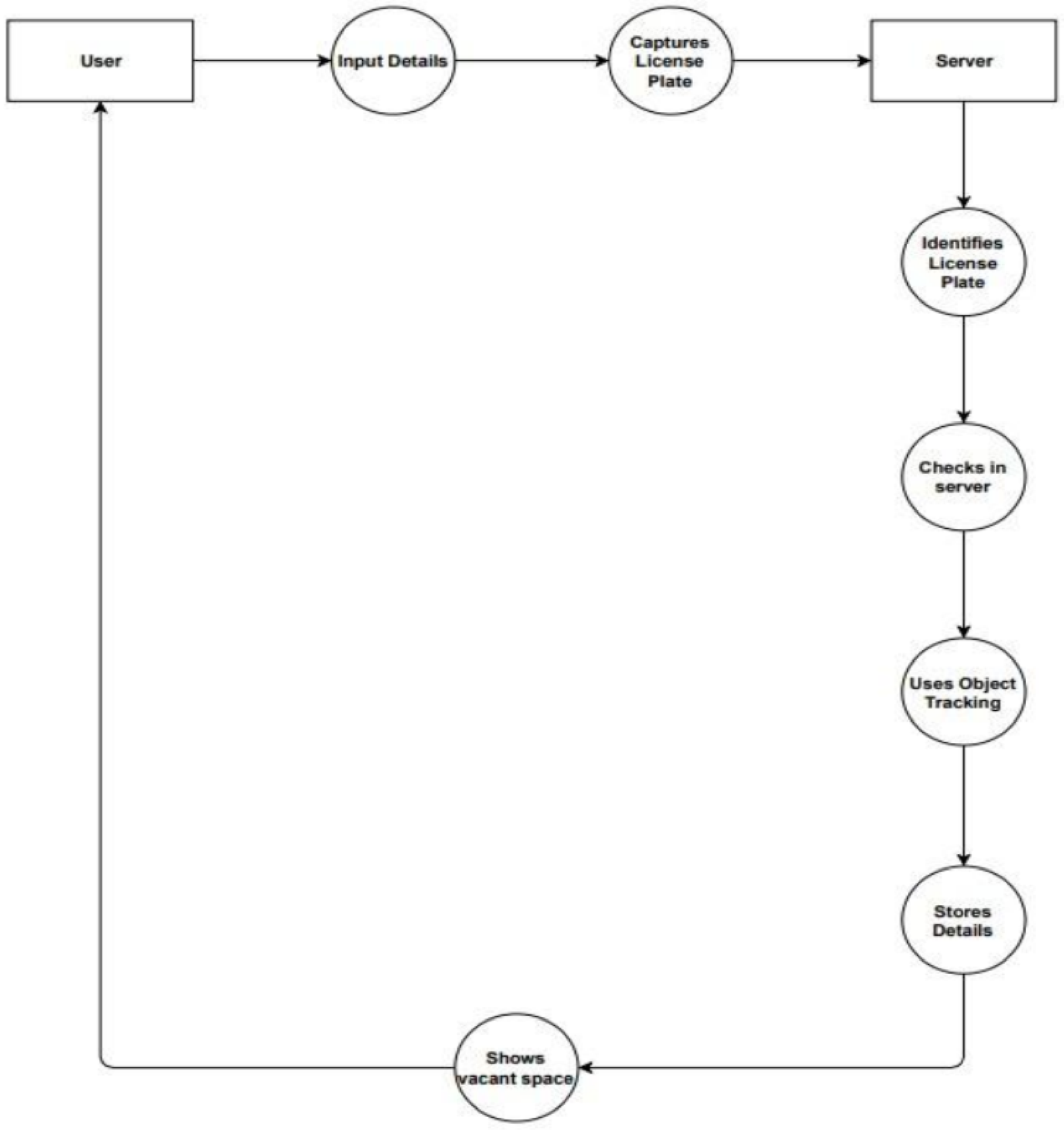
This is the initial idea for the flow of the data. The data has to be flown from user to server and from server to the user for the prediction of the disease by entering details and sending the data. Communication is done between the user and the server.

DFD -L-0 :-



DFD L-0

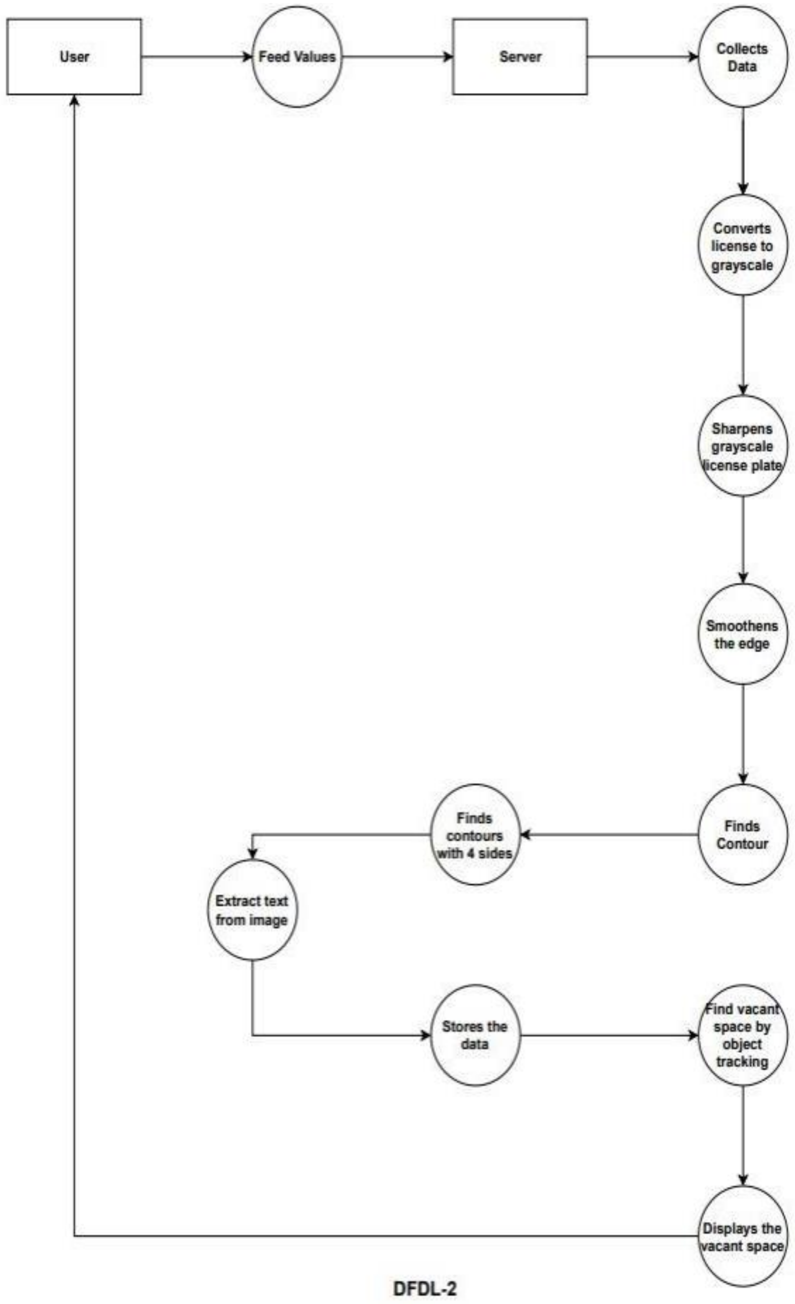
DFD-L-1 :-



DFD L1

This is the process or the idea where the user enters his or her details alongside the image of their license plate is also captured. All this data is collectively sent to the server where the license plate is identified and checked in the server. Afterwards, object tracking is used in order to find the vacant space and then that vacant space is shown along with storing of details in the server.

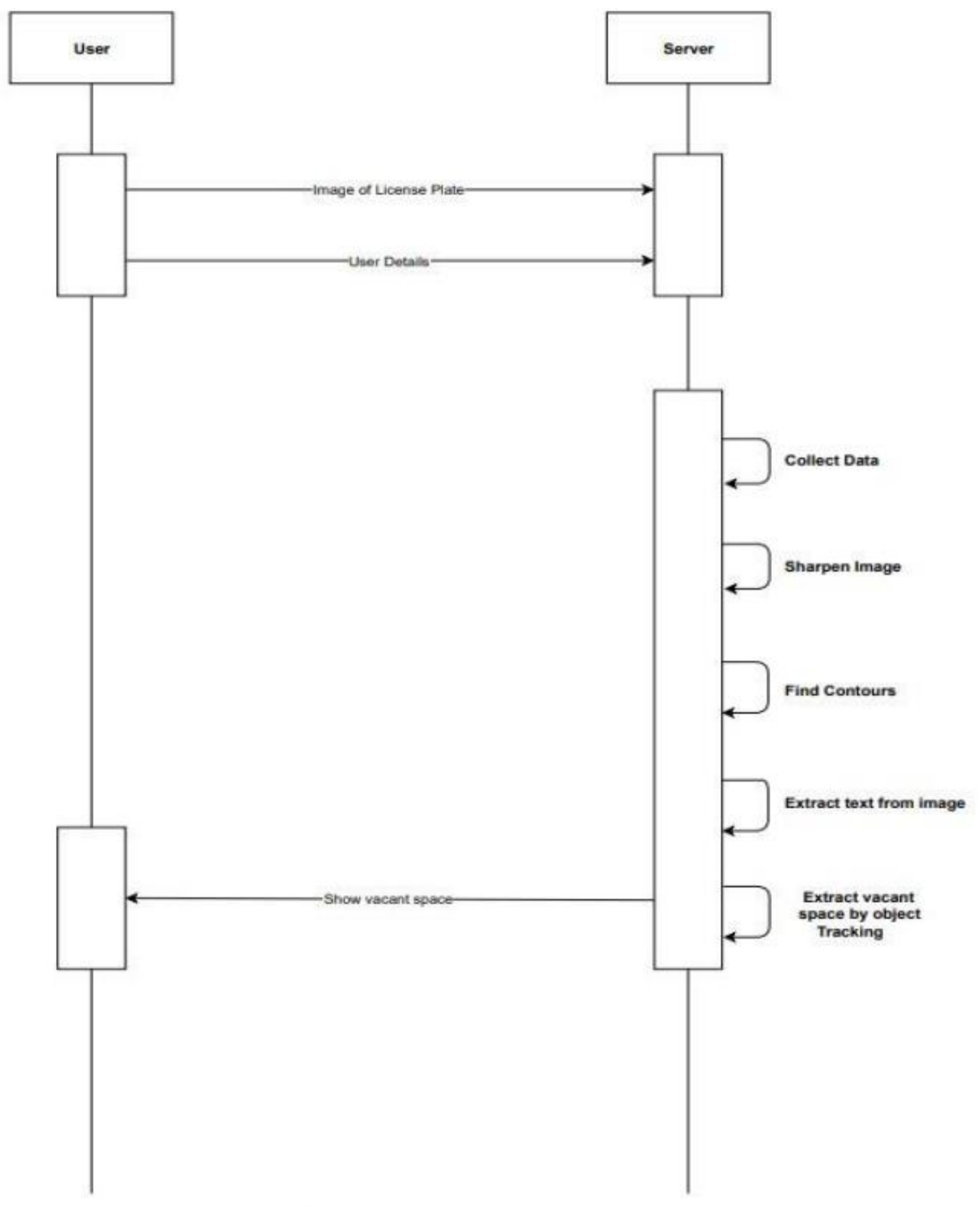
DFD-L-2:-


User
Feed Values
Server
Collects Data
Converts license to grayscale
Sharpens grayscale license plate
Smoothens the edge
Finds Contour
Finds contours with 4 sides
Extract text from image
Stores the data
Find vacant space by object tracking
Displays the vacant space
DFDL-2

DFD L2

The license plate image is flown from user to the server, the image is converted into a gray-scale image where it is sharpened and smoothened to find contour with four sides. From the most suitable contour, The text is extracted and is stored in the server as license plate number and afterwards object tracking is used to find the vacant space.

5. Sequence Diagram –



Sequence Diagram

Sequence Diagram

6. Data Structure Design -

There is only one database embedded in this application, which stores the user information and using text extracted from license plate image, respective data is stored. The various fields in the table of this database are the name of the user, in -time, license plate of the user, phone number of the user, address of the user, email-id of the user, and then the user’s details will be uploaded which will be used for future recognition.

6.1 User Information Database

This database contains only one table which is basically required for the login activity of the user. The table has the following fields:

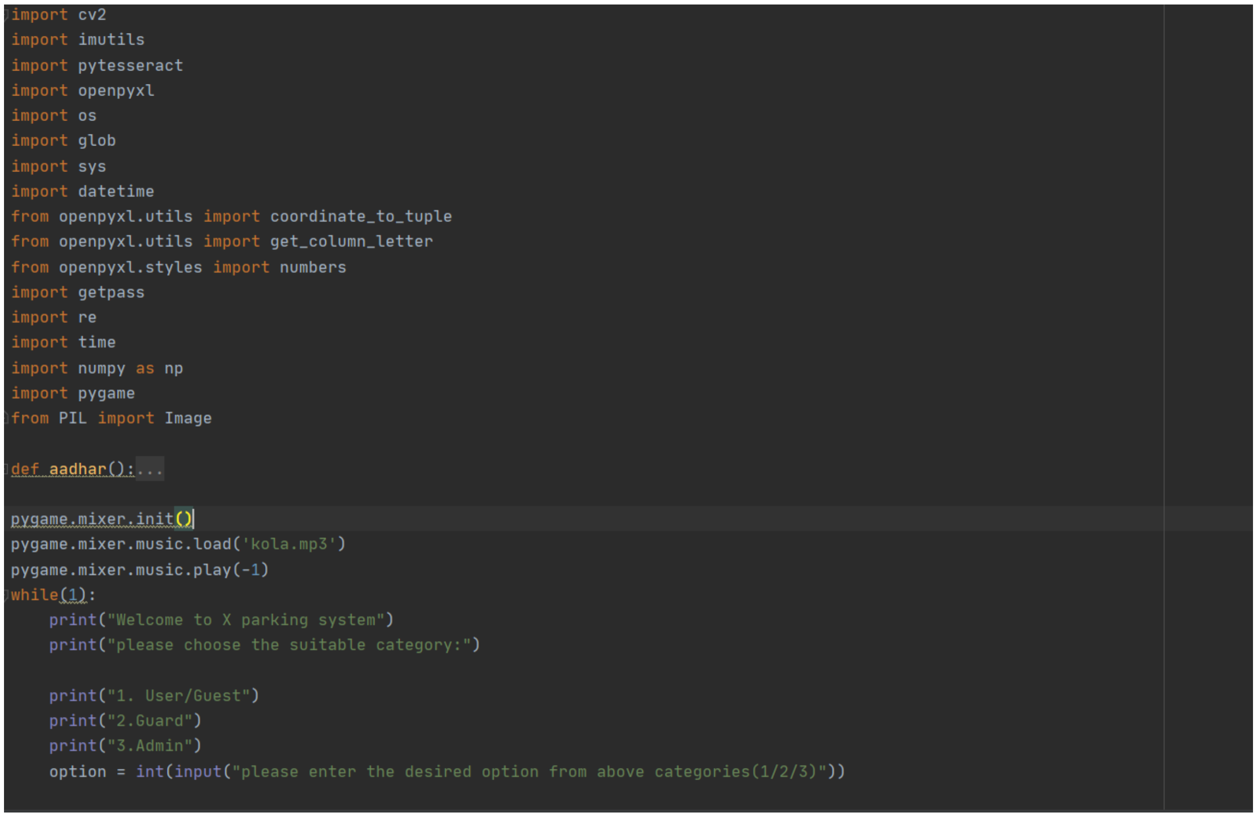
- License plate number
- Name
- Phone No.
- Address
- Email-Id
- Username
- In-time

Following is the table for the above user information database:

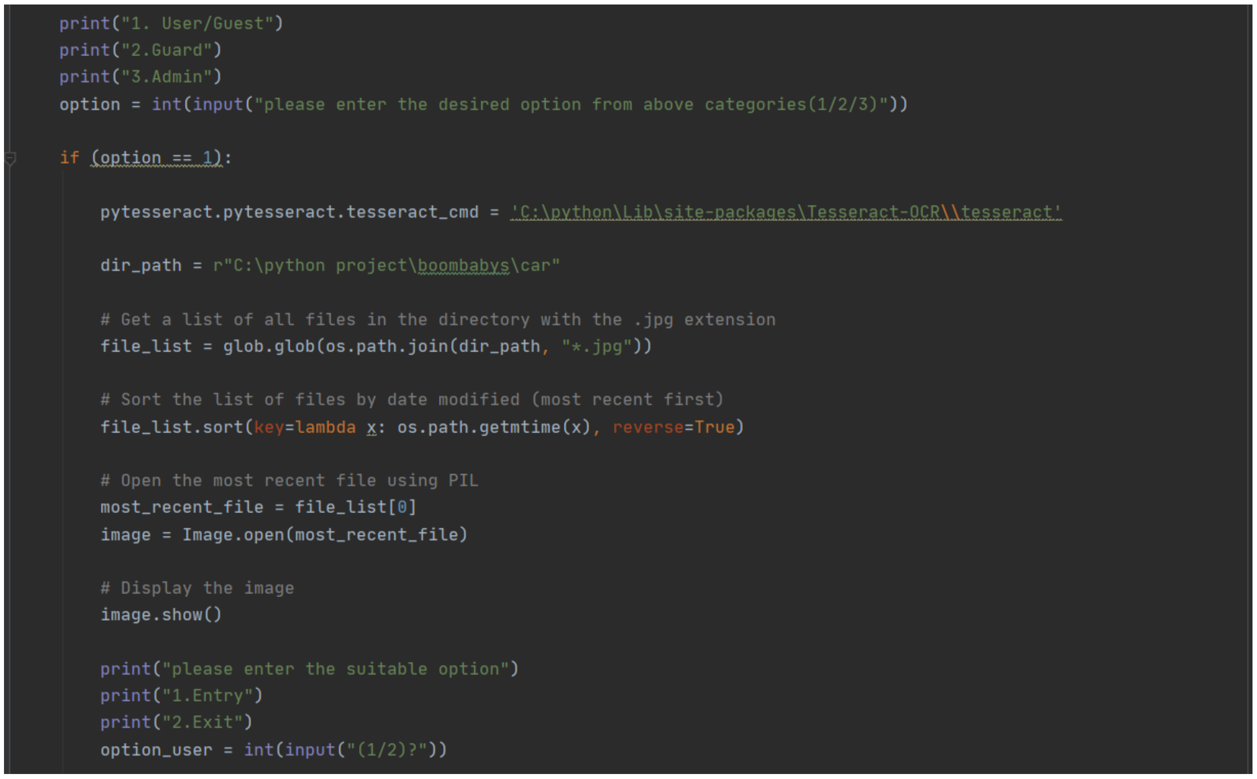
Field	**Type**	**Description**
License plate number	Text	Primary Key
Name	Text	Name of user
Contact Number	BigInt	User Contact Number
Address	Text	User Home Address
Email	Text	User Email id
In-time	Date/Time	In-time of the user

Chapter 4 : Implementation and System Integration

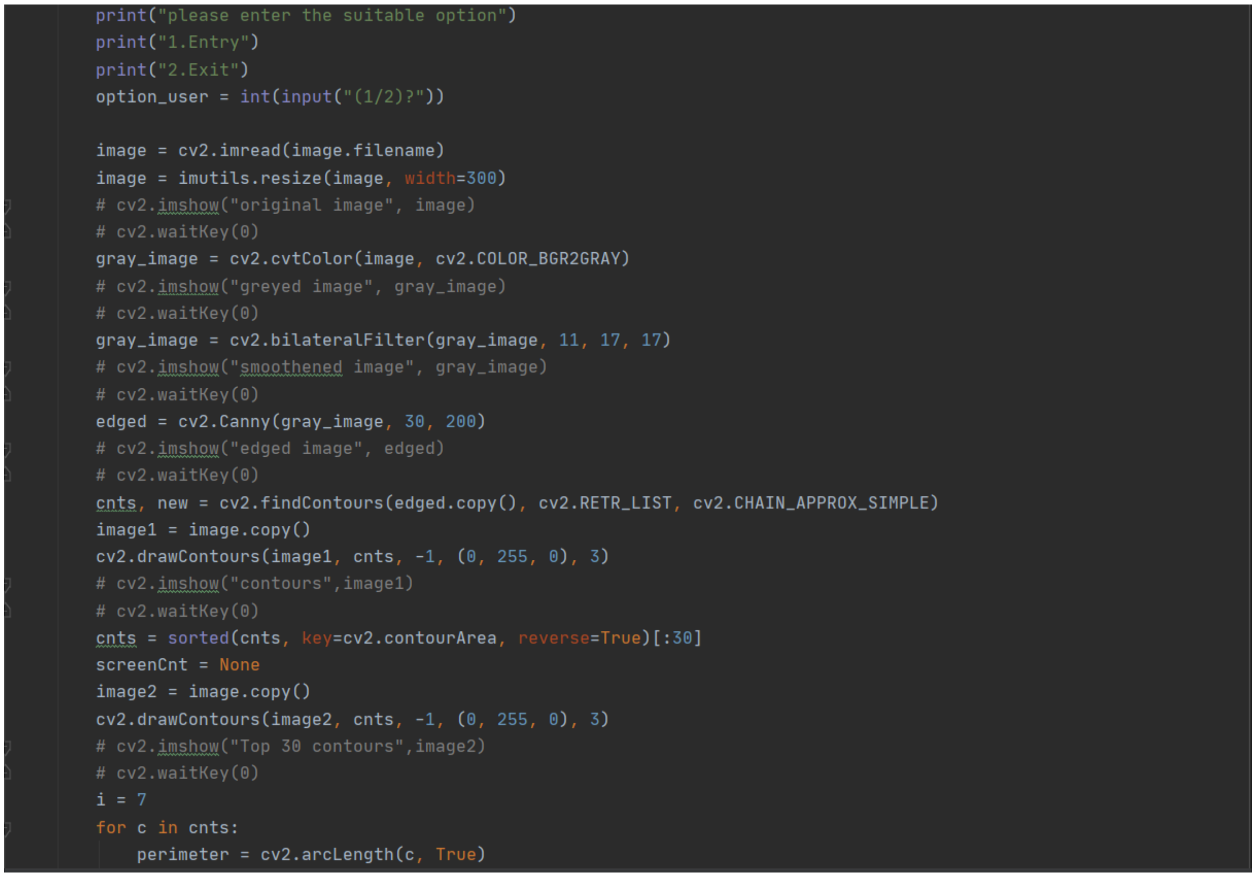
Code Walkthrough :



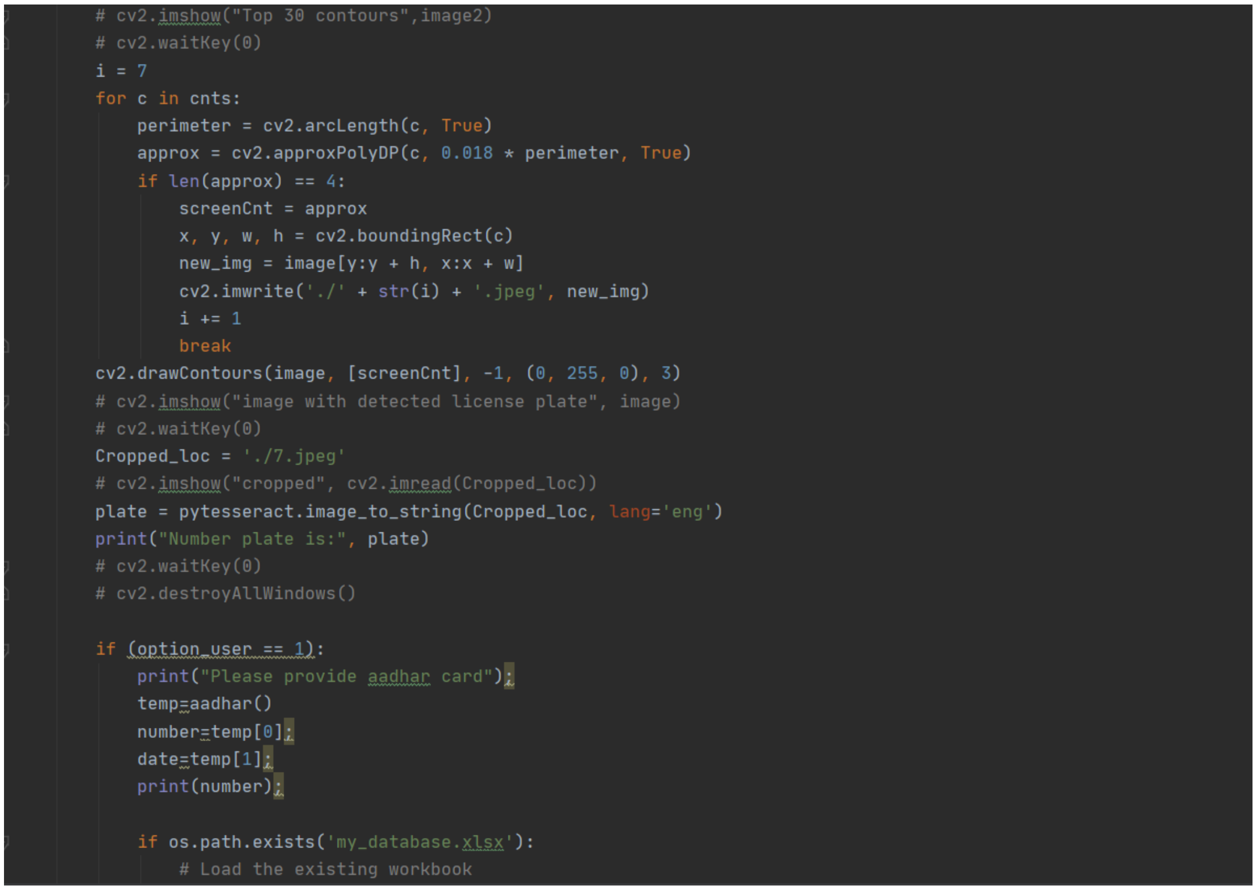
```python
import cv2
import imutils
import pytesseract
import openpyxl
import os
import glob
import sys
import datetime
from openpyxl.utils import coordinate_to_tuple
from openpyxl.utils import get_column_letter
from openpyxl.styles import numbers
import getpass
import re
import time
import numpy as np
import pygame
from PIL import Image

def aadhar():...

pygame.mixer.init()
pygame.mixer.music.load('kola.mp3')
pygame.mixer.music.play(-1)
while(1):
    print("Welcome to X parking system")
    print("please choose the suitable category:")

    print("1. User/Guest")
    print("2.Guard")
    print("3.Admin")
    option = int(input("please enter the desired option from above categories(1/2/3)"))
```



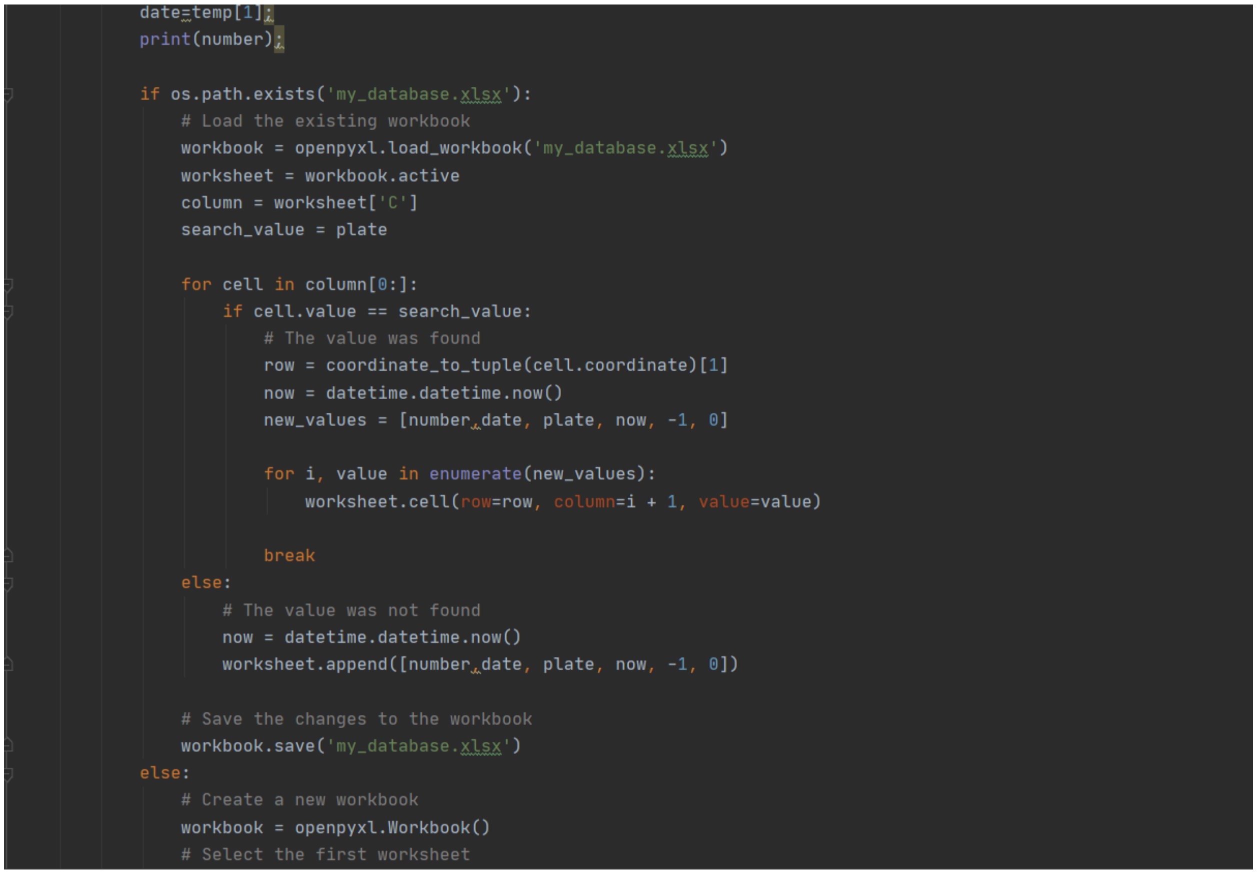
```
print("1. User/Guest")
print("2.Guard")
print("3.Admin")
option = int(input("please enter the desired option from above categories(1/2/3)"))

if (option == 1):

    pytesseract.pytesseract.tesseract_cmd = 'C:\python\Lib\site-packages\Tesseract-OCR\\tesseract'

    dir_path = r"C:\python project\boombabys\car"

    # Get a list of all files in the directory with the .jpg extension
    file_list = glob.glob(os.path.join(dir_path, "*.jpg"))

    # Sort the list of files by date modified (most recent first)
    file_list.sort(key=lambda x: os.path.getmtime(x), reverse=True)

    # Open the most recent file using PIL
    most_recent_file = file_list[0]
    image = Image.open(most_recent_file)

    # Display the image
    image.show()

    print("please enter the suitable option")
    print("1.Entry")
    print("2.Exit")
    option_user = int(input("(1/2)?"))
```



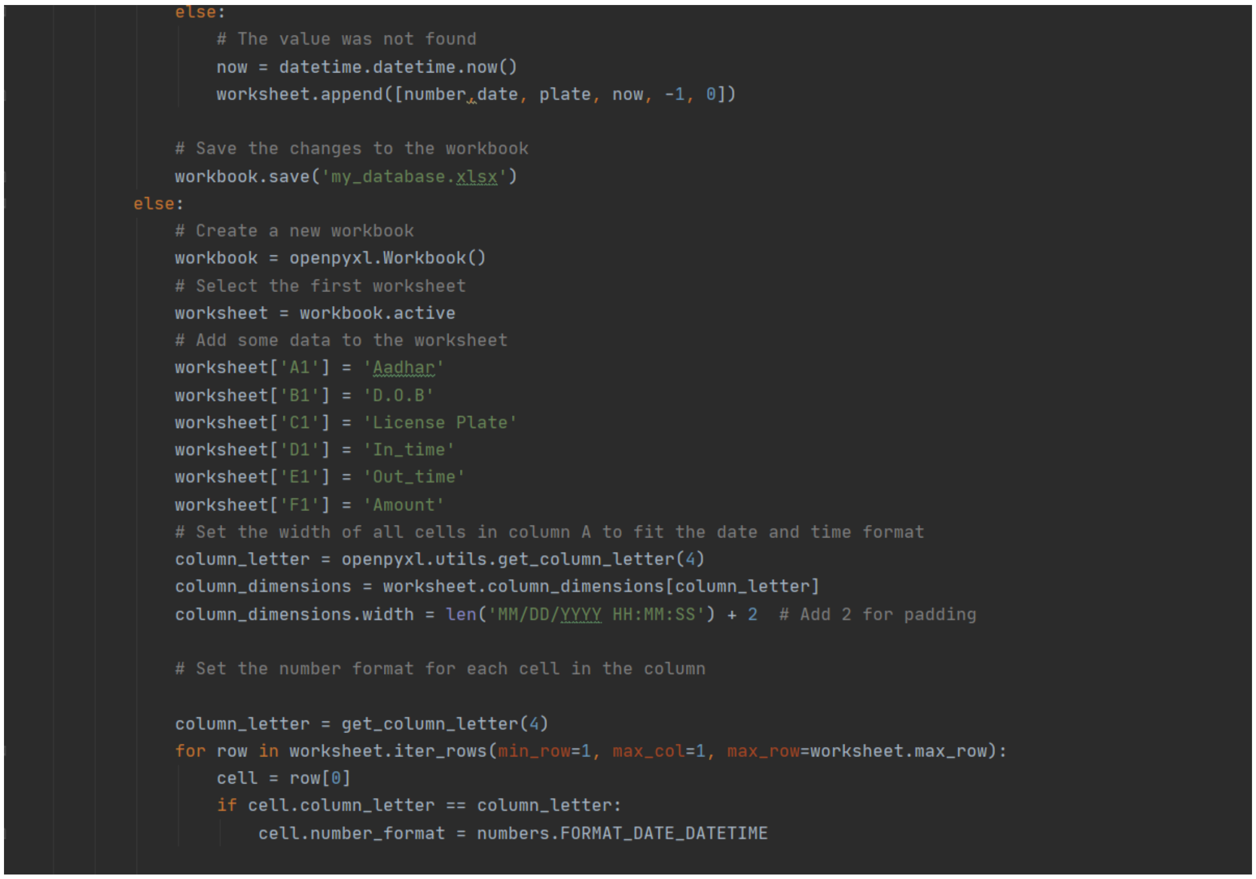
```
    print("please enter the suitable option")
    print("1.Entry")
    print("2.Exit")
    option_user = int(input("(1/2)?"))

    image = cv2.imread(image.filename)
    image = imutils.resize(image, width=300)
    # cv2.imshow("original image", image)
    # cv2.waitKey(0)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # cv2.imshow("greyed image", gray_image)
    # cv2.waitKey(0)
    gray_image = cv2.bilateralFilter(gray_image, 11, 17, 17)
    # cv2.imshow("smoothened image", gray_image)
    # cv2.waitKey(0)
    edged = cv2.Canny(gray_image, 30, 200)
    # cv2.imshow("edged image", edged)
    # cv2.waitKey(0)
    cnts, new = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    image1 = image.copy()
    cv2.drawContours(image1, cnts, -1, (0, 255, 0), 3)
    # cv2.imshow("contours",image1)
    # cv2.waitKey(0)
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:30]
    screenCnt = None
    image2 = image.copy()
    cv2.drawContours(image2, cnts, -1, (0, 255, 0), 3)
    # cv2.imshow("Top 30 contours",image2)
    # cv2.waitKey(0)
    i = 7
    for c in cnts:
        perimeter = cv2.arcLength(c, True)
```



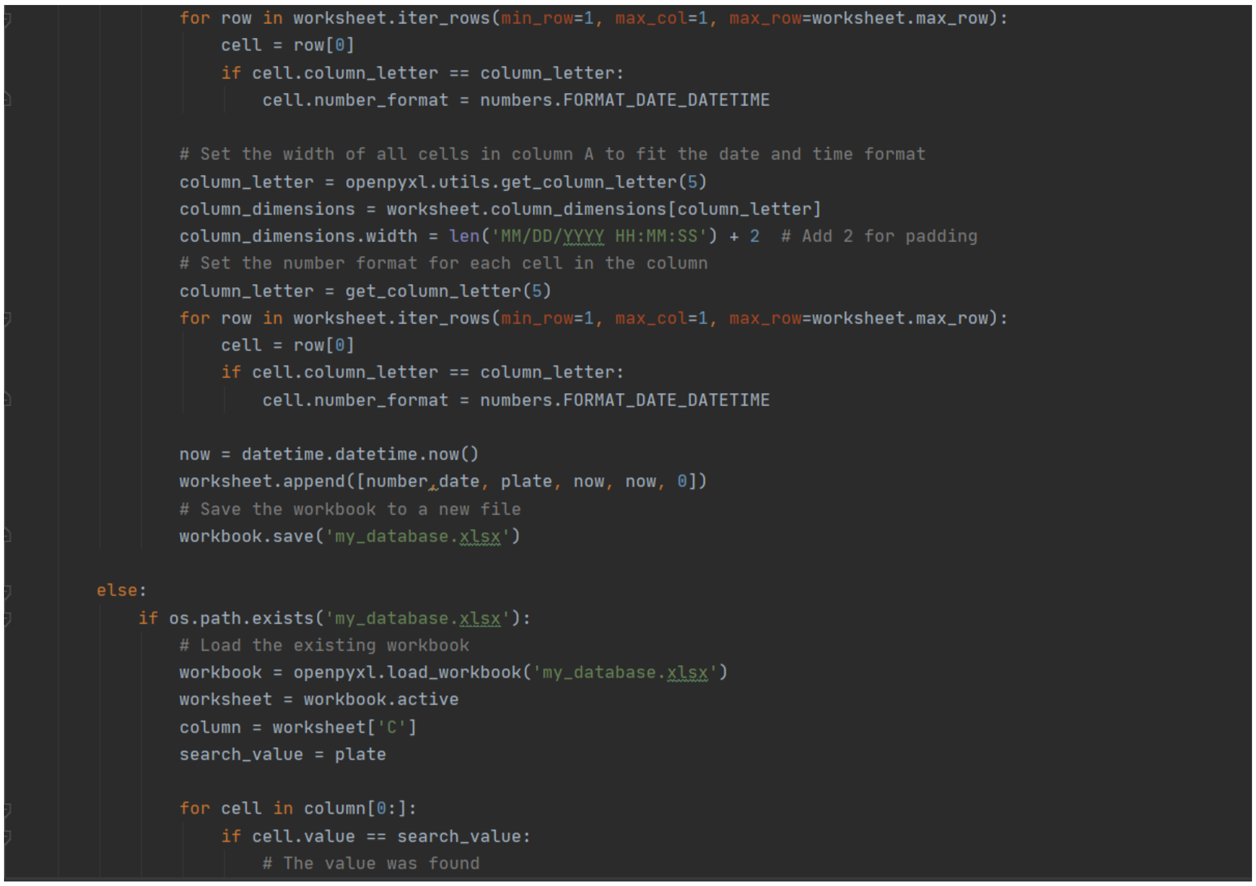
```
    # cv2.imshow("Top 30 contours",image2)
    # cv2.waitKey(0)
    i = 7
    for c in cnts:
        perimeter = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.018 * perimeter, True)
        if len(approx) == 4:
            screenCnt = approx
            x, y, w, h = cv2.boundingRect(c)
            new_img = image[y:y + h, x:x + w]
            cv2.imwrite('./' + str(i) + '.jpeg', new_img)
            i += 1
            break
    cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3)
    # cv2.imshow("image with detected license plate", image)
    # cv2.waitKey(0)
    Cropped_loc = './7.jpeg'
    # cv2.imshow("cropped", cv2.imread(Cropped_loc))
    plate = pytesseract.image_to_string(Cropped_loc, lang='eng')
    print("Number plate is:", plate)
    # cv2.waitKey(0)
    # cv2.destroyAllWindows()

    if (option_user == 1):
        print("Please provide aadhar card");
        temp=aadhar()
        number=temp[0];
        date=temp[1];
        print(number);

        if os.path.exists('my_database.xlsx'):
            # Load the existing workbook
```



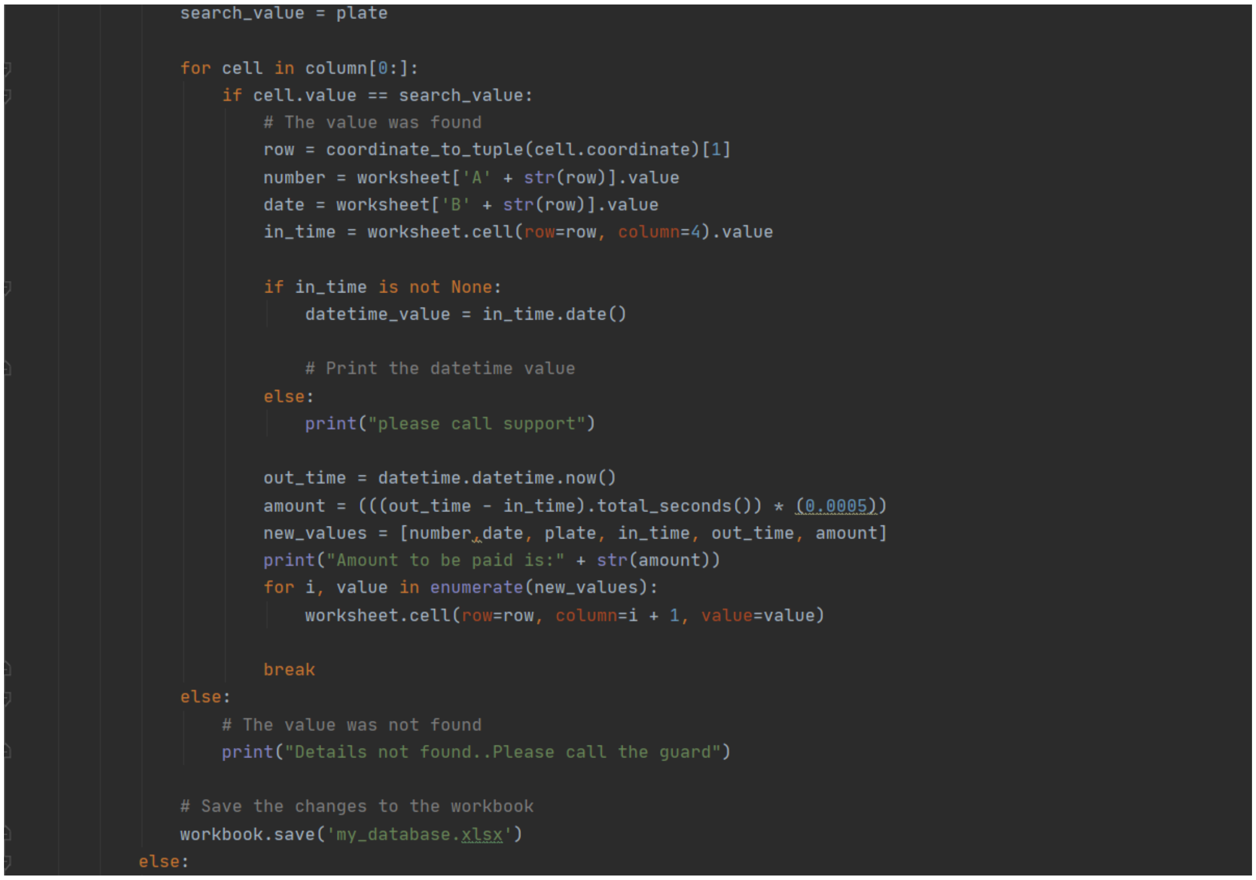
```
        date=temp[1];
        print(number);

        if os.path.exists('my_database.xlsx'):
            # Load the existing workbook
            workbook = openpyxl.load_workbook('my_database.xlsx')
            worksheet = workbook.active
            column = worksheet['C']
            search_value = plate

            for cell in column[0:]:
                if cell.value == search_value:
                    # The value was found
                    row = coordinate_to_tuple(cell.coordinate)[1]
                    now = datetime.datetime.now()
                    new_values = [number,date, plate, now, -1, 0]

                    for i, value in enumerate(new_values):
                        worksheet.cell(row=row, column=i + 1, value=value)

                    break
            else:
                # The value was not found
                now = datetime.datetime.now()
                worksheet.append([number,date, plate, now, -1, 0])

            # Save the changes to the workbook
            workbook.save('my_database.xlsx')
        else:
            # Create a new workbook
            workbook = openpyxl.Workbook()
            # Select the first worksheet
```



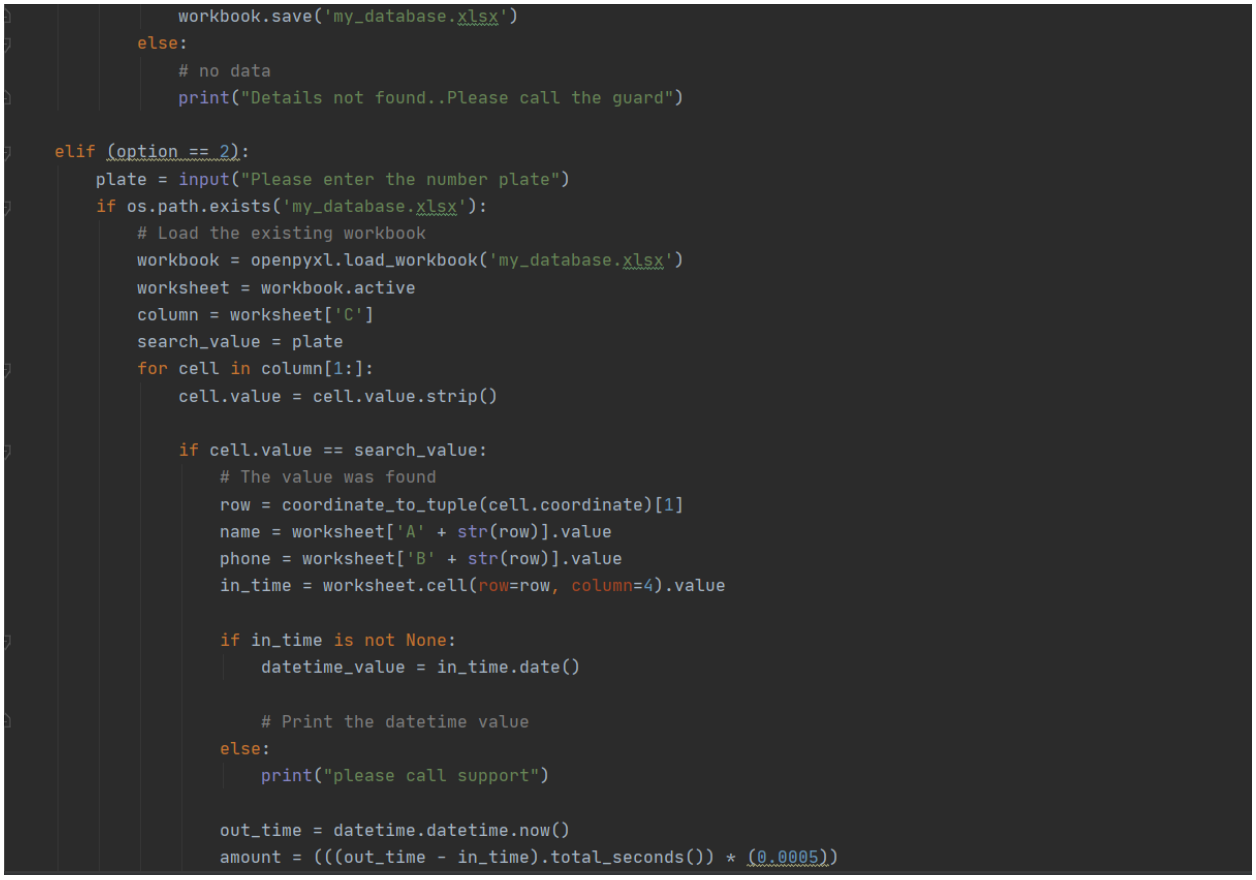
```
                else:
                    # The value was not found
                    now = datetime.datetime.now()
                    worksheet.append([number,date, plate, now, -1, 0])

                # Save the changes to the workbook
                workbook.save('my_database.xlsx')
            else:
                # Create a new workbook
                workbook = openpyxl.Workbook()
                # Select the first worksheet
                worksheet = workbook.active
                # Add some data to the worksheet
                worksheet['A1'] = 'Aadhar'
                worksheet['B1'] = 'D.O.B'
                worksheet['C1'] = 'License Plate'
                worksheet['D1'] = 'In_time'
                worksheet['E1'] = 'Out_time'
                worksheet['F1'] = 'Amount'
                # Set the width of all cells in column A to fit the date and time format
                column_letter = openpyxl.utils.get_column_letter(4)
                column_dimensions = worksheet.column_dimensions[column_letter]
                column_dimensions.width = len('MM/DD/YYYY HH:MM:SS') + 2  # Add 2 for padding

                # Set the number format for each cell in the column

                column_letter = get_column_letter(4)
                for row in worksheet.iter_rows(min_row=1, max_col=1, max_row=worksheet.max_row):
                    cell = row[0]
                    if cell.column_letter == column_letter:
                        cell.number_format = numbers.FORMAT_DATE_DATETIME
```



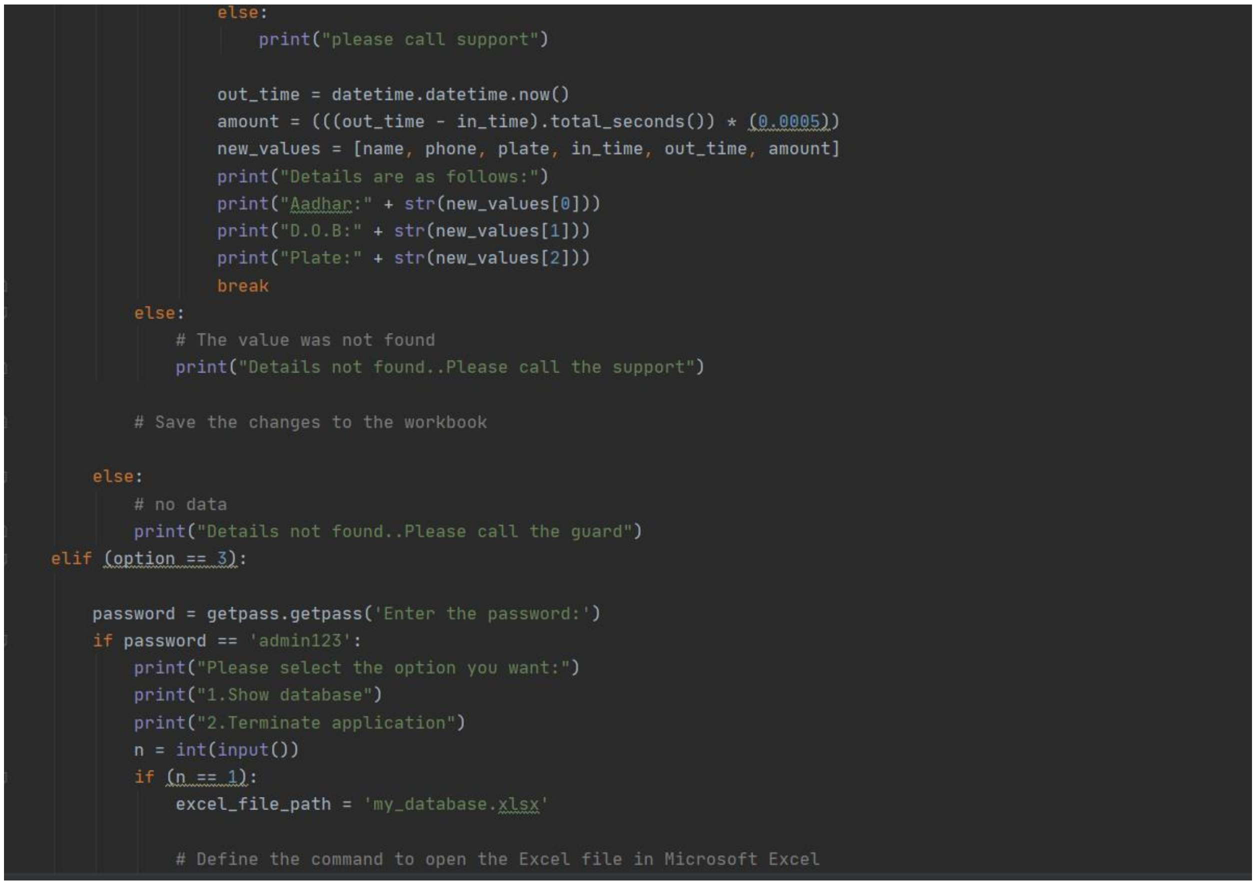
```
                for row in worksheet.iter_rows(min_row=1, max_col=1, max_row=worksheet.max_row):
                    cell = row[0]
                    if cell.column_letter == column_letter:
                        cell.number_format = numbers.FORMAT_DATE_DATETIME

                # Set the width of all cells in column A to fit the date and time format
                column_letter = openpyxl.utils.get_column_letter(5)
                column_dimensions = worksheet.column_dimensions[column_letter]
                column_dimensions.width = len('MM/DD/YYYY HH:MM:SS') + 2  # Add 2 for padding
                # Set the number format for each cell in the column
                column_letter = get_column_letter(5)
                for row in worksheet.iter_rows(min_row=1, max_col=1, max_row=worksheet.max_row):
                    cell = row[0]
                    if cell.column_letter == column_letter:
                        cell.number_format = numbers.FORMAT_DATE_DATETIME

                now = datetime.datetime.now()
                worksheet.append([number,date, plate, now, now, 0])
                # Save the workbook to a new file
                workbook.save('my_database.xlsx')

        else:
            if os.path.exists('my_database.xlsx'):
                # Load the existing workbook
                workbook = openpyxl.load_workbook('my_database.xlsx')
                worksheet = workbook.active
                column = worksheet['C']
                search_value = plate

                for cell in column[0:]:
                    if cell.value == search_value:
                        # The value was found
```



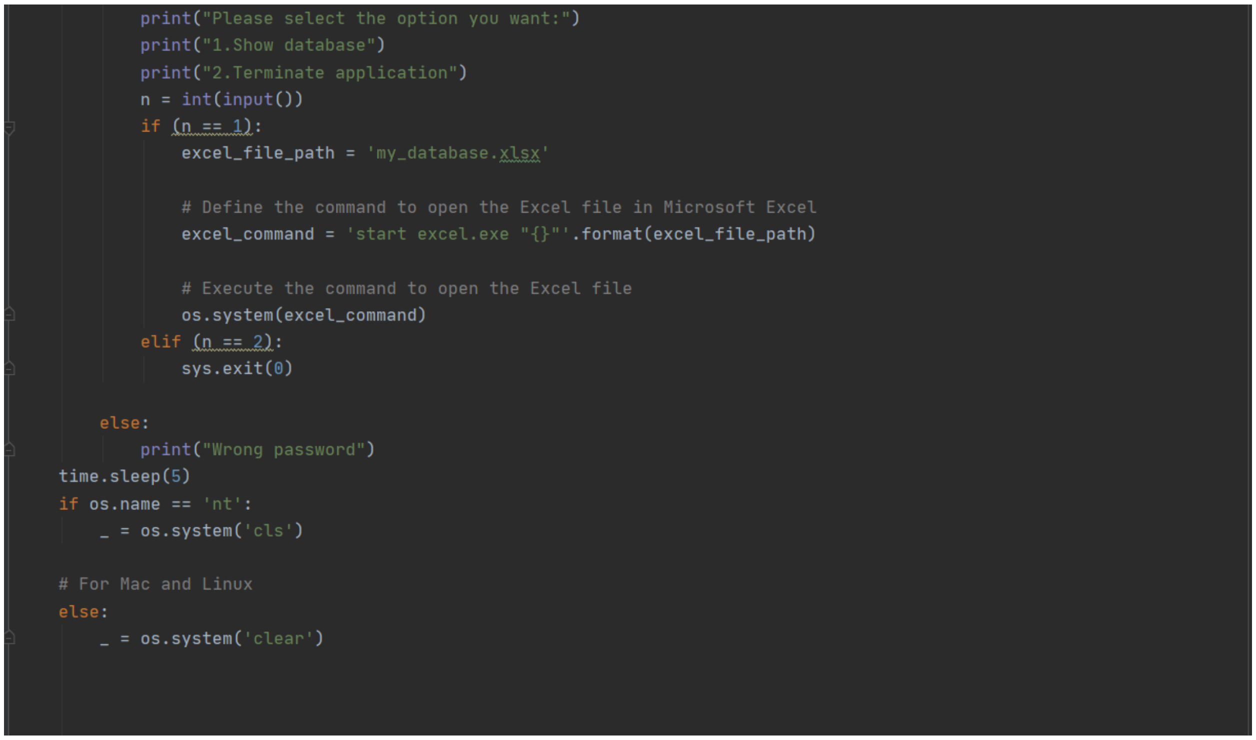
```
                search_value = plate

                for cell in column[0:]:
                    if cell.value == search_value:
                        # The value was found
                        row = coordinate_to_tuple(cell.coordinate)[1]
                        number = worksheet['A' + str(row)].value
                        date = worksheet['B' + str(row)].value
                        in_time = worksheet.cell(row=row, column=4).value

                        if in_time is not None:
                            datetime_value = in_time.date()

                            # Print the datetime value
                        else:
                            print("please call support")

                        out_time = datetime.datetime.now()
                        amount = (((out_time - in_time).total_seconds()) * (0.0005))
                        new_values = [number,date, plate, in_time, out_time, amount]
                        print("Amount to be paid is:" + str(amount))
                        for i, value in enumerate(new_values):
                            worksheet.cell(row=row, column=i + 1, value=value)

                        break
                else:
                    # The value was not found
                    print("Details not found..Please call the guard")

                # Save the changes to the workbook
                workbook.save('my_database.xlsx')
            else:
```



```
                workbook.save('my_database.xlsx')
            else:
                # no data
                print("Details not found..Please call the guard")

    elif (option == 2):
        plate = input("Please enter the number plate")
        if os.path.exists('my_database.xlsx'):
            # Load the existing workbook
            workbook = openpyxl.load_workbook('my_database.xlsx')
            worksheet = workbook.active
            column = worksheet['C']
            search_value = plate
            for cell in column[1:]:
                cell.value = cell.value.strip()

                if cell.value == search_value:
                    # The value was found
                    row = coordinate_to_tuple(cell.coordinate)[1]
                    name = worksheet['A' + str(row)].value
                    phone = worksheet['B' + str(row)].value
                    in_time = worksheet.cell(row=row, column=4).value

                    if in_time is not None:
                        datetime_value = in_time.date()

                        # Print the datetime value
                    else:
                        print("please call support")

                    out_time = datetime.datetime.now()
                    amount = (((out_time - in_time).total_seconds()) * (0.0005))
```

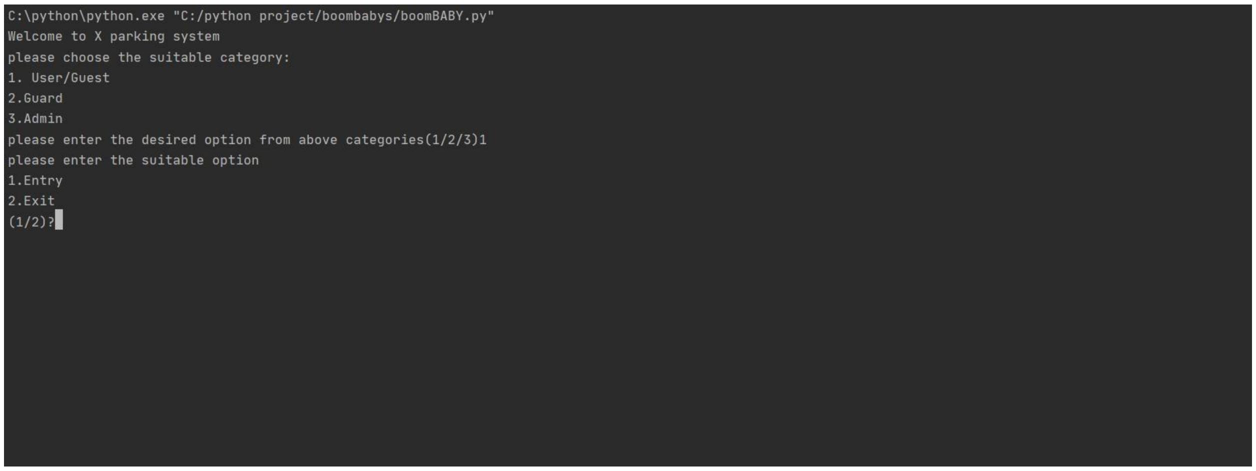


```
                    else:
                        print("please call support")

                    out_time = datetime.datetime.now()
                    amount = (((out_time - in_time).total_seconds()) * (0.0005))
                    new_values = [name, phone, plate, in_time, out_time, amount]
                    print("Details are as follows:")
                    print("Aadhar:" + str(new_values[0]))
                    print("D.O.B:" + str(new_values[1]))
                    print("Plate:" + str(new_values[2]))
                    break
            else:
                # The value was not found
                print("Details not found..Please call the support")

            # Save the changes to the workbook

        else:
            # no data
            print("Details not found..Please call the guard")
    elif (option == 3):

        password = getpass.getpass('Enter the password:')
        if password == 'admin123':
            print("Please select the option you want:")
            print("1.Show database")
            print("2.Terminate application")
            n = int(input())
            if (n == 1):
                excel_file_path = 'my_database.xlsx'

                # Define the command to open the Excel file in Microsoft Excel
```



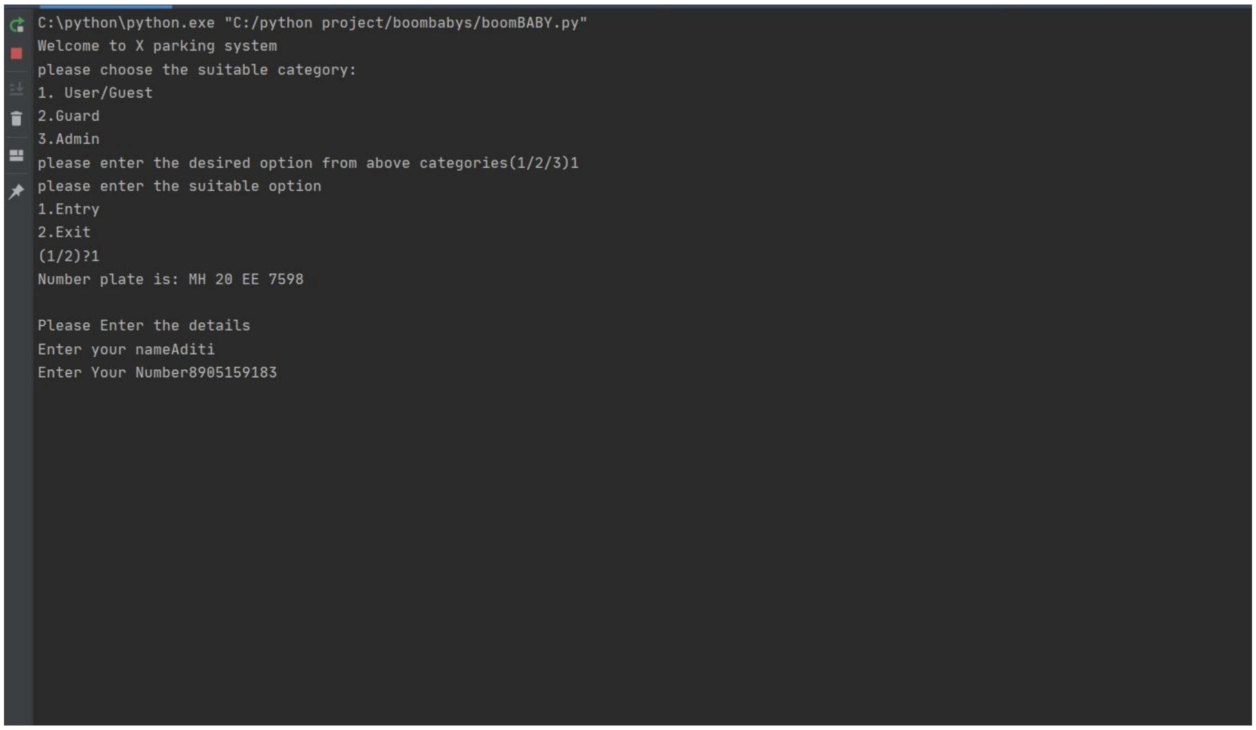
```
        print("Please select the option you want:")
        print("1.Show database")
        print("2.Terminate application")
        n = int(input())
        if (n == 1):
            excel_file_path = 'my_database.xlsx'

            # Define the command to open the Excel file in Microsoft Excel
            excel_command = 'start excel.exe "{}"'.format(excel_file_path)

            # Execute the command to open the Excel file
            os.system(excel_command)
        elif (n == 2):
            sys.exit(0)

    else:
        print("Wrong password")
time.sleep(5)
if os.name == 'nt':
    _ = os.system('cls')

# For Mac and Linux
else:
    _ = os.system('clear')
```

Chapter 5 : OUTPUT and SCREENSHOTS:



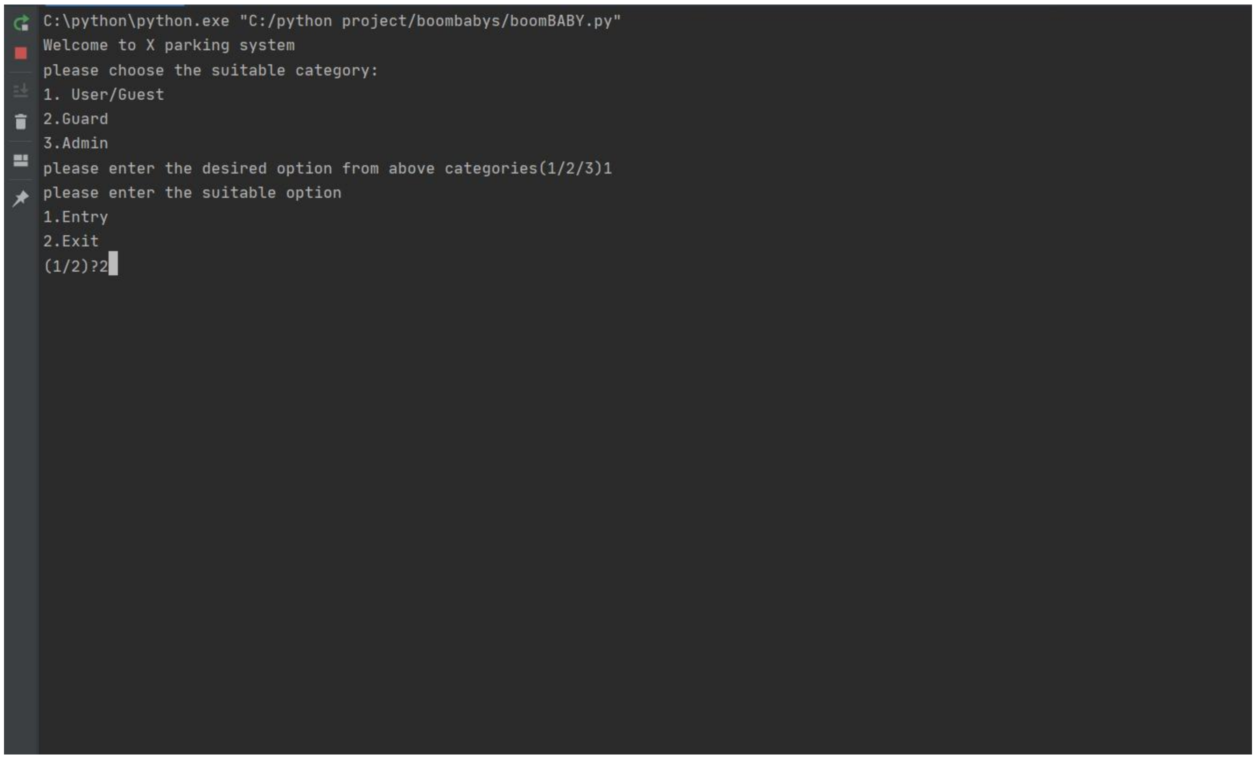
```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)1
please enter the suitable option
1.Entry
2.Exit
(1/2)?
```

Output 1 : Welcome Screen for user



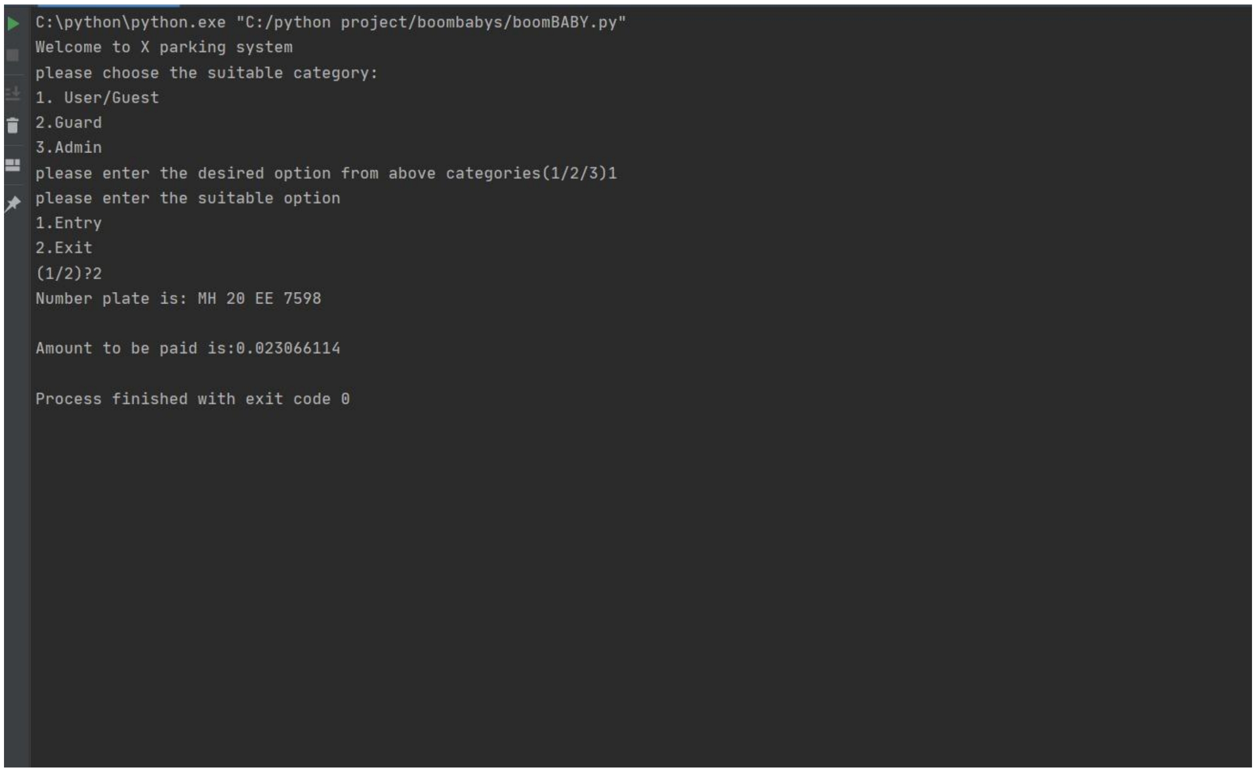
```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)1
please enter the suitable option
1.Entry
2.Exit
(1/2)?1
Number plate is: MH 20 EE 7598

Please Enter the details
Enter your nameAditi
Enter Your Number8905159183
```

Output 2 : User making Entry



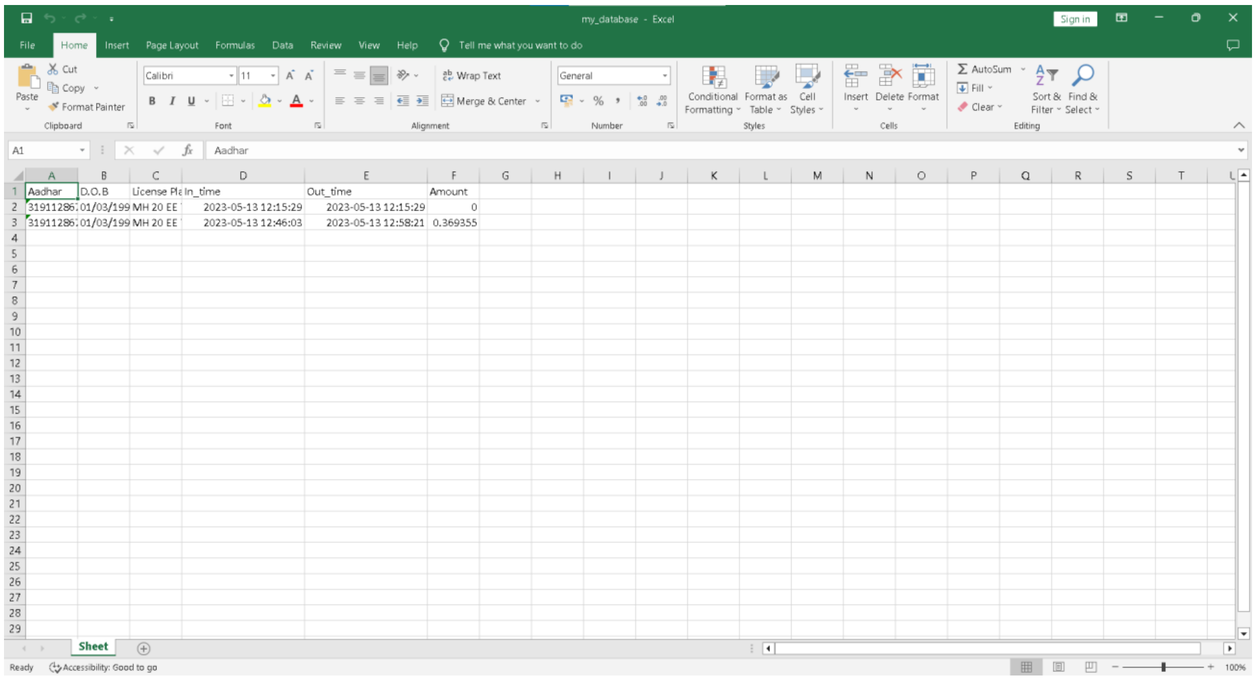
```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)1
please enter the suitable option
1.Entry
2.Exit
(1/2)?2
```

Output 3 : User Exiting



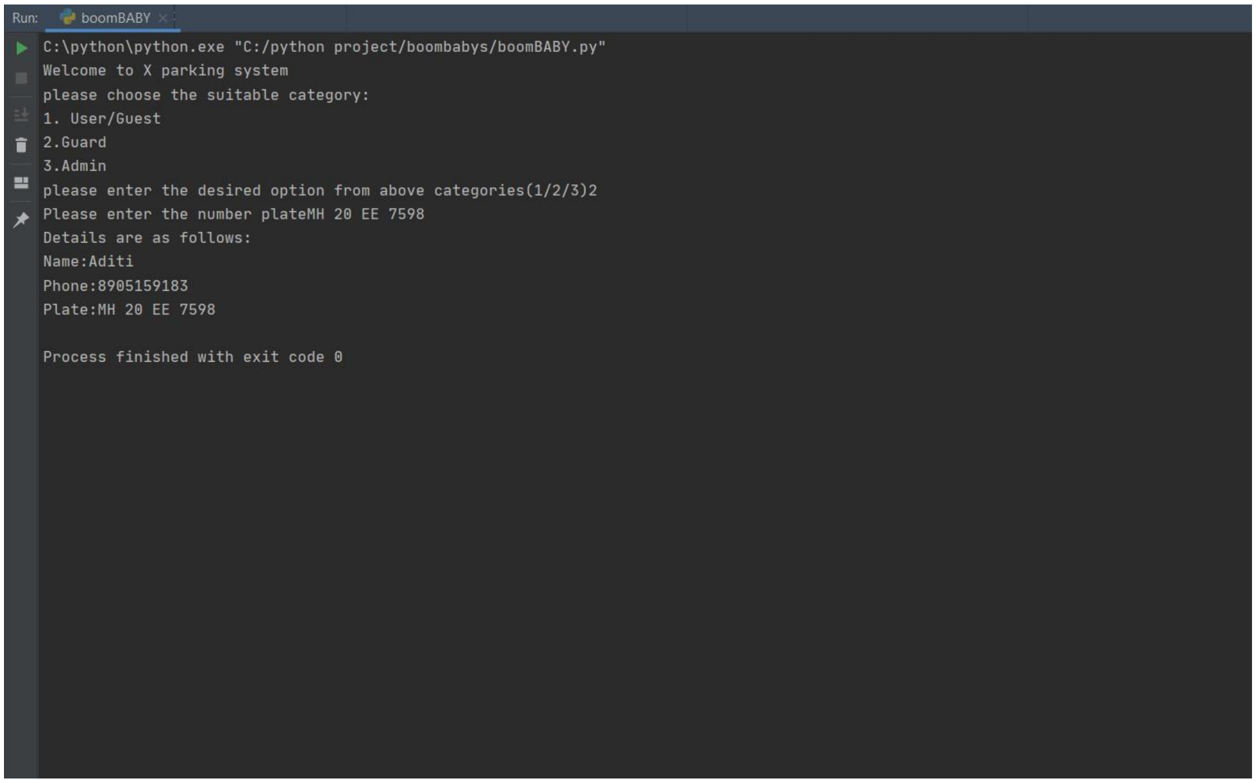
```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)1
please enter the suitable option
1.Entry
2.Exit
(1/2)?2
Number plate is: MH 20 EE 7598

Amount to be paid is:0.023066114

Process finished with exit code 0
```

Output 4 : Amount to be paid



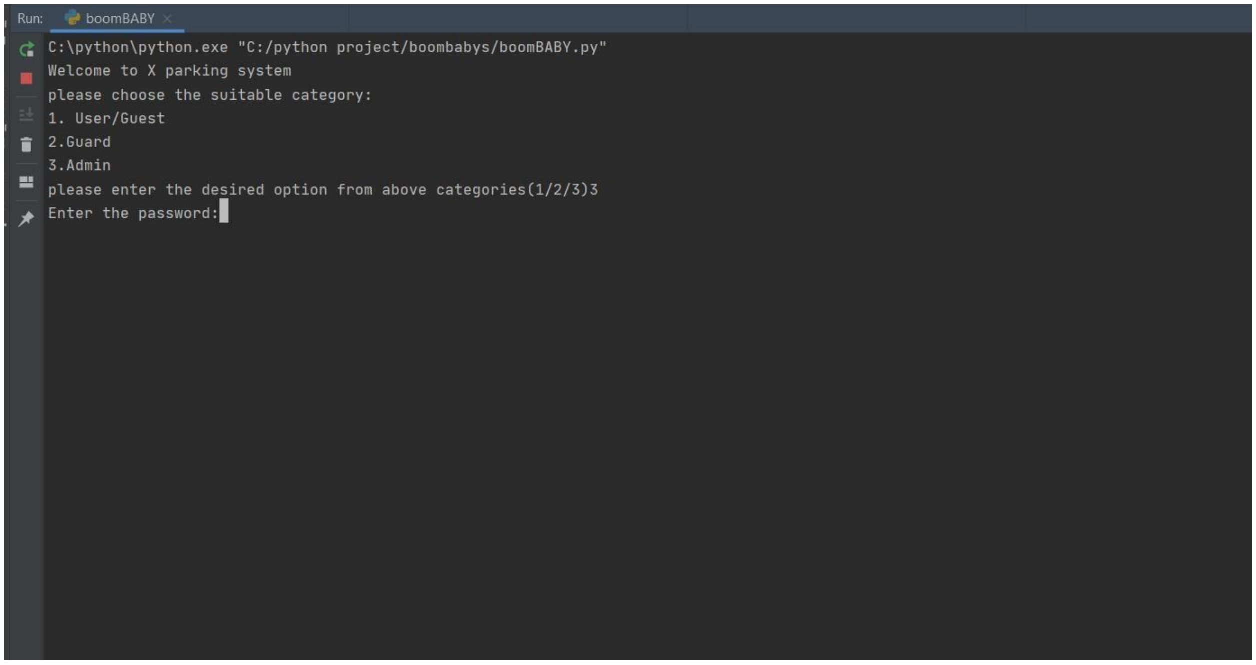
Aadhar	D.O.B	License Pla	In_time	Out_time	Amount
31911286	01/03/199	MH 20 EE	2023-05-13 12:15:29	2023-05-13 12:15:29	0
31911286	01/03/199	MH 20 EE	2023-05-13 12:46:03	2023-05-13 12:58:21	0.369355

Output 5 : Database



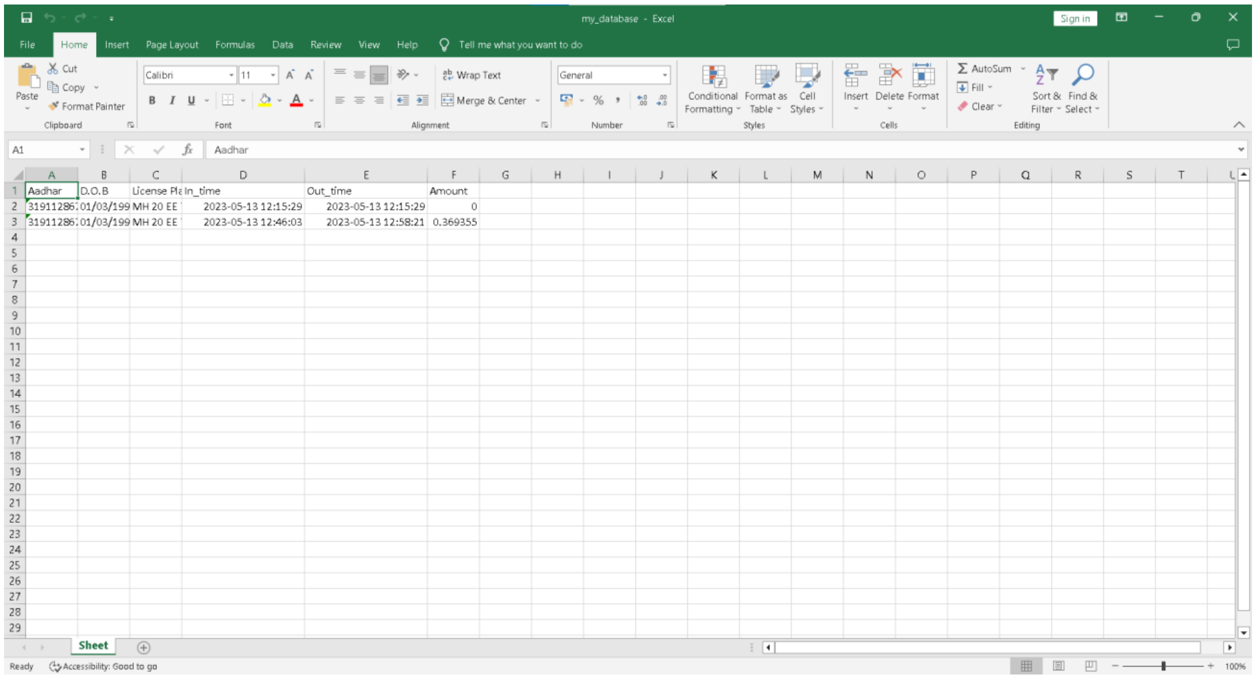
```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)2
Please enter the number plateMH 20 EE 7598
Details are as follows:
Name:Aditi
Phone:8905159183
Plate:MH 20 EE 7598

Process finished with exit code 0
```

Output 6 : Guard’s Screen



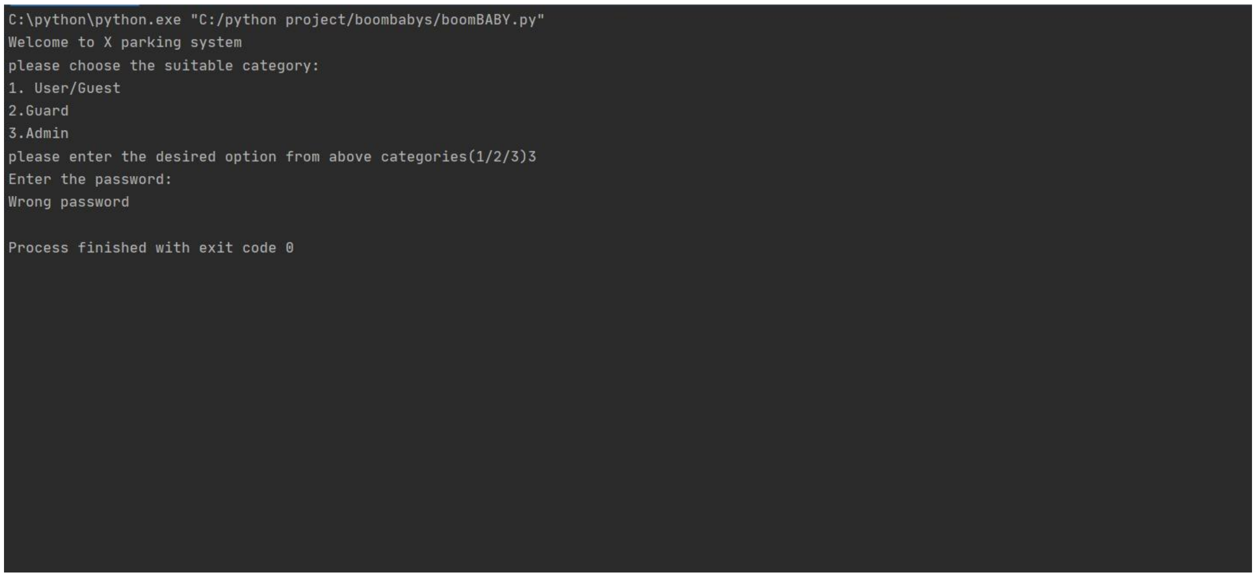
```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)3
Enter the password:
```

Output 7 : Admin Screen



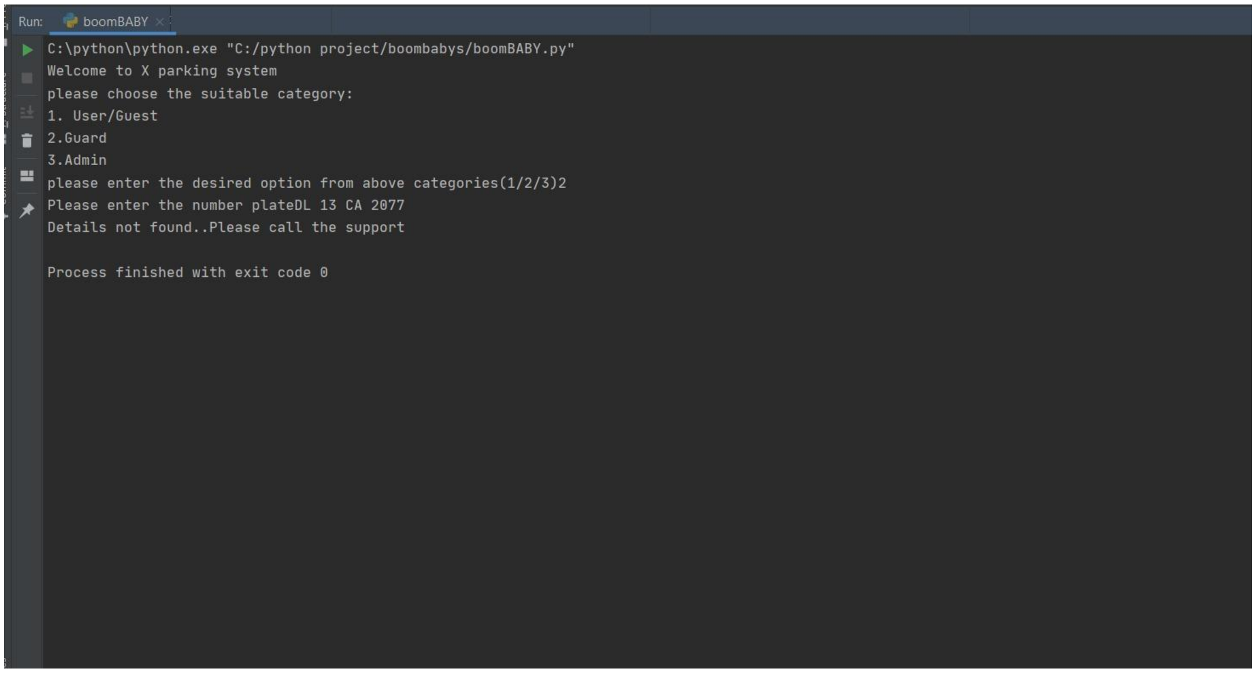
Aadhar	D.O.B	License Pla	In_time	Out_time	Amount
3191128€	01/03/199	MH 20 EE	2023-05-13 12:15:29	2023-05-13 12:15:29	0
3191128€	01/03/199	MH 20 EE	2023-05-13 12:46:03	2023-05-13 12:58:21	0.369355

Output 8 : Database displayed after correct password is entered



```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)3
Enter the password:
Wrong password

Process finished with exit code 0
```

Output 9 : When wrong password is entered



```
C:\python\python.exe "C:/python project/boombabys/boomBABY.py"
Welcome to X parking system
please choose the suitable category:
1. User/Guest
2.Guard
3.Admin
please enter the desired option from above categories(1/2/3)2
Please enter the number plateDL 13 CA 2077
Details not found..Please call the support

Process finished with exit code 0
```

Output 10 : When wrong License Plate is entered

Chapter 6 : Conclusion and Future Scope

Effectiveness and Need of Automated Parking :

Automated parking systems have become increasingly popular in recent years due to the benefits they offer. Here are some of the effectiveness and needs of automated parking:

1. **Increased parking capacity:** Automated parking systems can increase parking capacity in crowded urban areas. Since automated parking systems use a compact, vertical design, they require less space than traditional parking lots. This means that more vehicles can be parked in the same amount of space, increasing parking capacity.

2. **Reduced labor costs:** Automated parking systems require less labor than traditional parking lots. Since vehicles are parked and retrieved automatically, there is no need for parking attendants. This reduces labor costs and can result in significant cost savings for parking lot operators.

3. **Improved efficiency:** Automated parking systems are more efficient than traditional parking lots. Since vehicles are parked and retrieved automatically, the process is faster and more streamlined. This can reduce the amount of time it takes for drivers to park and retrieve their vehicles, resulting in a more pleasant parking experience.

4. **Improved safety:** Automated parking systems can improve safety in parking lots. Since there are no parking attendants, there

is less risk of accidents or injuries involving parking attendants. Additionally, since the parking process is automated, there is less risk of damage to vehicles during the parking process.

5. **Environmental benefits:** Automated parking systems can have environmental benefits. Since they require less space than traditional parking lots, they can reduce the amount of land used for parking. Additionally, since they are more efficient, they can reduce the amount of time vehicles spend idling, resulting in reduced emissions.

In conclusion, automated parking systems are effective and necessary in modern urban areas. They increase parking capacity, reduce labor costs, improve efficiency and safety, and offer environmental benefits. With the continued growth of urban areas and the increasing demand for parking, automated parking systems will continue to be an important part of parking management solutions.

Conclusion :

Automated parking systems are becoming increasingly popular as cities continue to grow and the need for efficient and effective parking solutions becomes more apparent. These systems offer a range of benefits, including increased parking efficiency, reduced congestion, and enhanced user experience. They also have the potential to provide valuable data insights that can inform future planning and decision-making.

From the analysis of various studies and research papers, it is evident that automated parking systems have already demonstrated their effectiveness in a range of settings. For example, they have been successfully implemented in residential and commercial buildings, as well as in public spaces such as airports and shopping centers. These systems have also been shown to improve accessibility for people with disabilities and reduce the time and effort required to park a vehicle.

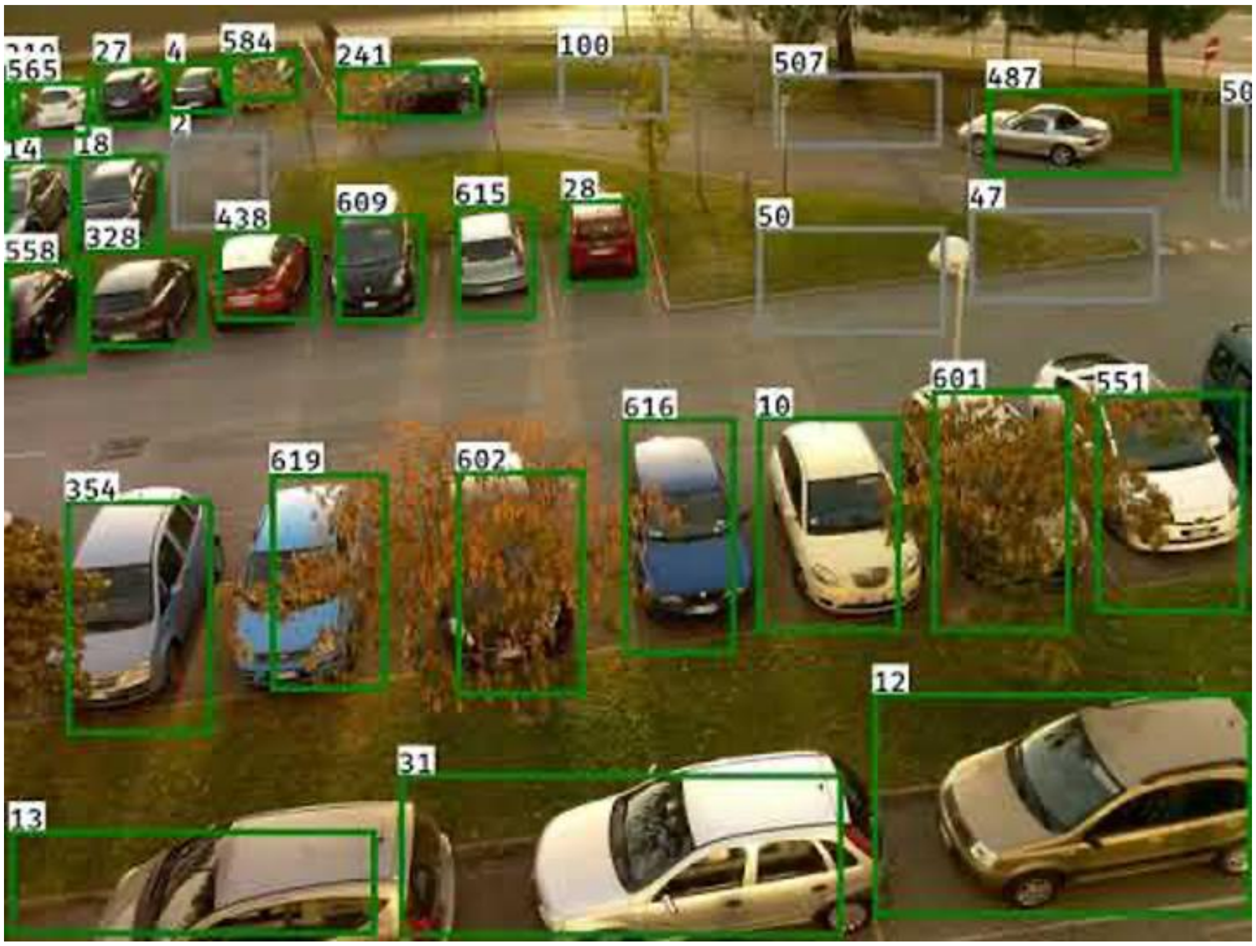
Moreover, the development of autonomous vehicles and smart cities is likely to increase the demand for automated parking systems. This presents an exciting opportunity for further innovation and research in this area. For example, researchers could explore the potential of using artificial intelligence and machine learning to enhance the performance of automated parking systems. These technologies could be used to optimize the allocation of parking spaces, reduce waiting times, and improve the accuracy and reliability of the systems.

Another area of future research could be the integration of automated parking systems with existing infrastructure. Many cities already have established parking structures and systems in place, and there may be challenges associated with integrating automated systems with these existing structures. Further research is needed to address these challenges and ensure that automated parking systems can be seamlessly integrated into existing infrastructure.

In addition, the user experience of automated parking systems could be improved through the development of more intuitive and user-friendly interfaces. Researchers could explore the potential of using augmented reality and other advanced technologies to enhance the user experience and provide more personalized and tailored parking solutions.

Finally, there is also a need for further research on the environmental impact of automated parking systems. While these systems can reduce congestion and improve parking efficiency, they also require significant amounts of energy to operate. Researchers could explore ways to reduce the energy consumption of these systems and minimize their environmental impact.

In conclusion, automated parking systems offer a range of benefits and have the potential to revolutionize parking and transportation systems. Further research and innovation in this area are needed to realize their full potential and address the challenges associated with their implementation. As the demand for efficient and effective parking solutions continues to grow, the development of automated parking systems will likely play a key role in meeting this need.



References

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3274917

https://www.mdpi.com/2032-6653/12/4/200

https://aip.scitation.org/doi/pdf/10.1063/1.3526208#:~:text=LPR%20algorithm%20consists%20of%20the,character%20segmentation%20and%20character%20recognition.

https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/

https://www.westfaliaparking.com/

Chapter 7 : Appendices

Appendices A : Technology Used

The process of Automatic License Plate Reading (ALPR) involves several steps, including:

1. Image Capture: ALPR systems use cameras to capture images of license plates on vehicles. The cameras can be mounted on vehicles, stationary poles, or other structures.

2. Image Preprocessing: Once the image is captured, the ALPR software processes it to improve the image quality and remove any noise or distortion. This preprocessing step includes image stabilization, contrast enhancement, and noise reduction.

3. License Plate Detection: After the image is preprocessed, the ALPR software searches for license plates within the image. This process involves detecting the edges of the license plate, segmenting the characters, and verifying the plate's orientation.

4. Character Segmentation: Once the license plate is detected, the ALPR software segments the characters within the plate. This process involves separating each character from the plate's background and identifying its position within the plate.

5. Character Recognition: After the characters are segmented, the ALPR software uses Optical Character Recognition (OCR) to recognize each character. The OCR process involves analysing the shape and pattern of each character and comparing it to a database of known character patterns.

6. Data Processing: Once the characters are recognized, the ALPR software combines them to form the license plate number. The software then compares the license plate number to a database of known license plate numbers to determine if there is a match.

7. Alerting and Reporting: If the ALPR system detects a match, it can alert law enforcement officers, parking attendants, or other personnel in real-time. The system can also generate reports on the number of vehicles that have been scanned, the number of matches detected, and other relevant data.

In conclusion, the ALPR process involves several steps, from image capture to alerting and reporting. The accuracy and efficiency of the ALPR system depend on the quality of the image capture, the effectiveness of the pre-processing and character recognition algorithms, and the reliability of the database used for comparison. ALPR technology has several benefits, including improved safety, reduced labour costs, and real-time data analysis, but concerns regarding privacy and data protection must also be considered.

Appendices B : Mathematical or theoretical background

Algorithms for automatic license plate reading :

There are several algorithms used in Automatic License Plate Reading (ALPR) systems that enable accurate and efficient recognition of license plates. Here are some of the common algorithms used in ALPR:

1. Edge Detection: Edge detection algorithms are used to detect the edges of the license plate in an image. The algorithm analyzes the image to identify the edges of the plate, and then segments the plate from the rest of the image.

2. Character Segmentation: Character segmentation algorithms are used to separate the characters of the license plate from the background. This algorithm involves analyzing the plate image to identify each character's location and size, and then separating them from the background.

3. Optical Character Recognition (OCR): OCR algorithms are used to recognize the characters on the license plate. This algorithm involves analyzing the shape and pattern of each character and comparing it to a database of known character patterns. OCR algorithms use machine learning techniques such as convolutional neural networks to improve the accuracy of character recognition.

4. Template Matching: Template matching algorithms are used to match the recognized characters against a database of known license plate numbers. This algorithm involves comparing the recognized characters to a template of known characters to determine the license plate number.

5. Machine Learning: Machine learning algorithms are used to improve the accuracy of ALPR systems. Machine learning algorithms use large datasets to learn and improve their performance over time. This allows ALPR systems to improve their accuracy and recognition speed over time.

In conclusion, ALPR systems use several algorithms to accurately and efficiently recognize license plates. Edge detection, character segmentation, OCR, template matching, and machine learning algorithms are all used to improve the accuracy of the system. The use of these algorithms helps ALPR systems to recognize license plates in real-time, providing benefits such as improved safety, reduced labour costs, and real-time data analysis.

Appendices C : Additional Details

Some additional details to consider when implementing an automated license plate reading system:

Image preprocessing techniques like thresholding, blurring, and edge detection can improve the accuracy of license plate recognition.

License plate recognition can be a computationally intensive task, especially when processing large volumes of images. To improve performance, you can use techniques like parallel processing and GPU acceleration.

In addition to recognizing the license plate characters, you can also use other features like the plate color and shape to improve the accuracy of the system.

You may need to handle variations in license plate formats across different countries and regions. One approach is to train the system on a diverse set of license plate images from different regions.

You can integrate the license plate reading system with a database to store and retrieve information about registered vehicles. This can be useful for applications like parking management, toll collection, and law enforcement.

Bio - Data of Team Members :

Individual Contributions in the Project :

1. **Aditi Maheshwari - Involved in license plate reading algorithm Research, Synopsis, Database management and Report, Research Paper, Connecting Excel File with Python Code, Aadhar card reading**

2. **Anany Srivastava – Involved in Project Planning and Organization, Presentation, SRS Document, SDD Document, License Plate Reading Algorithm, Implementation, Project Report, Research Paper, Aadhar Card Reading**

3. **Arpit Rajput – Database Formation, Presentation, Research Paper, Object Tracking Research**

4. **Deepak Gupta – Final Report, Frequent File Accessing, Database Formation, Password Encryption**

Acknowledgements :

We would like to express our gratitude to the following individuals for their invaluable contributions to this project:

We would also like to thank the Python community for creating and maintaining the Python programming language and the many open source libraries that we used in this project. In particular, we would like to thank:

I would like to express my heartfelt thanks to my project guide, Mr. Anuj Kumar Dwivedi, for their invaluable guidance, support, and encouragement throughout the development of this project. Their expertise and knowledge have been instrumental in shaping the direction and scope of this project, and I am grateful for their unwavering support and patience

Finally, we would like to thank our families and friends for their unwavering support and encouragement throughout the development of this project. Their support has been invaluable to us, and we are grateful for their love and encouragement.

Glossary :

Term	**Definition**
Active Image	The image of the current car which has to be parked.
Admin	Person in charge of the functioning and has access of all the functioning of the software.
Database	Collection of all the information monitored by this system.
Driver	Person who is the owner and driving the vehicle
Field	A cell within a form.
Historical Database	The existing membership database
Member	A member of the already existing database of the cars
Guard	Anyone capturing the photo of car
Camera	Devices which would be used to click the image of the incoming cars.
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
Stakeholder	Any person with an interest in the project who is not a developer.
User	Driver or admin or guard