Project name – AI Enabled Car Parking Using OpenCV

Team Details -
Khushi Tolani *[Team Leader]*
Amisha Tripathi
Saad Sabahuddin
Raghav Memani

Project Report Format

1. INTRODUCTION

1.1 Project Overview

In congested urban areas, the hunt for available parking spaces during peak hours is a persistent source of frustration and delay for drivers. Parking lots, especially at high-traffic locations like shopping centers, are often overcrowded, leading to an inefficient and stressful experience. To address this common challenge, the project seeks to develop an innovative AI-enabled car parking solution employing OpenCV and Flask. This system's primary objective is to automate the process of identifying and counting vacant parking spaces in real-time, providing an accurate and up-to-the-minute overview of parking availability. The solution stands to greatly enhance urban parking convenience, minimize congestion, and alleviate the frustrations that often plague drivers.

The proposed system harnesses the power of OpenCV to monitor parking space occupancy in real-time using CCTV camera footage and displays the results on a screen with a counting meter. It provides an intuitive visual representation of the parking lot layout, marking open parking spots with green rectangles and occupied spaces with red rectangles. By streamlining the parking process, this innovative approach not only simplifies the parking experience but also contributes to reducing urban traffic congestion and enhancing the overall efficiency of parking in bustling urban areas. Furthermore, there's room for fine-tuning and optimization to continually improve the system's accuracy and effectiveness in different parking scenarios, making it a versatile and adaptable solution for various urban environments.

1.2 Purpose

The primary purpose of this project is to develop an AI-enabled car parking system utilizing OpenCV and Flask to address the persistent challenges associated with parking in busy urban areas. The project's key objectives and purposes can be summarized as follows:

1. Enhance Urban Parking Efficiency:The project aims to significantly improve the efficiency of urban parking by providing real-time information on available parking spaces. By automating the process of identifying and counting vacant parking spots, it empowers drivers to quickly locate open spaces, reducing the time and stress associated with parking in congested areas.

2. Alleviate Driver Frustration:Frustration and delays caused by the difficulty of finding parking spaces during peak hours in urban areas are a common problem. This solution intends to minimize driver frustration by offering accurate, up-to-the-minute data on parking availability, ultimately creating a smoother and less stressful parking experience.

3. Reduce Traffic Congestion:By streamlining the parking process and enabling drivers to find available spaces more efficiently, the project contributes to the reduction of traffic congestion in urban areas.

This, in turn, has a positive impact on traffic flow, reduces carbon emissions, and enhances overall urban mobility.

4. Innovation and Technology Integration:The project leverages cutting-edge technologies, such as computer vision (OpenCV) and web-based applications (Flask), to create a sophisticated and innovative parking solution. It demonstrates the potential of AI in improving daily urban experiences.

5. Adaptability and Scalability: The project's purpose extends beyond a single application. It offers a versatile system that can be adapted and scaled to various parking scenarios, including shopping malls, office buildings, and public spaces, making it a valuable tool for urban planners and facility managers.

In summary, the purpose of this project is to revolutionize urban parking by providing a reliable, real-time parking space monitoring system that reduces driver frustration, minimizes traffic congestion, and showcases the potential of AI and technology to enhance the quality of life in crowded urban environments.

2. LITERATURE SURVEY

2.1 Existing problem

In busy urban areas, the existing problem centers around the arduous task of finding available parking spaces, particularly during peak hours. This issue is a common source of frustration, delays, and inefficiency for drivers. Overcrowded parking lots, notably at locations like shopping malls and commercial centers, exacerbate this problem, leading to several challenges:

- Traffic Congestion: The extended search for parking spaces contributes significantly to traffic congestion in urban areas. Drivers often circle lots or drive slowly while looking for available spaces, disrupting traffic flow and increasing fuel consumption (Shoup, D.C. (2005). Cruising for parking. Transport Policy, 12(6), 479-486).

- Stress and Delays: Finding an open parking spot can be a stressful and time-consuming endeavor, especially during peak hours. This leads to delays, potential missed appointments, and heightened driver frustration (Huang, H., & Guensler, R. (2006). A real-time parking guidance system: Algorithm and field testing. Transportation Research Part C: Emerging Technologies, 14(3), 196-215).

- Inefficiency: Parking lots often exhibit both overutilized and underutilized areas, indicating inefficiencies in parking space allocation. This underutilization of space can be improved through effective parking management (Shoup, D.C. (2011). The high cost of free parking, Updated edition. American Planning Association).

- Environmental Impact: Extended searches for parking spaces result in increased fuel consumption and emissions, negatively affecting air quality and the environment in urban areas (Zhang, X., & Rilett, L. R. (2008). A simulation-based evaluation of the environmental impacts of parking guidance and information systems. Transportation Research Part C: Emerging Technologies, 16(2), 150-166).

2.2 References

- Shoup, D.C. (2005). Cruising for parking. Transport Policy, 12(6), 479-486.

- Huang, H., & Guensler, R. (2006). A real-time parking guidance system: Algorithm and field

testing. Transportation Research Part C: Emerging Technologies, 14(3), 196-215.

- Shoup, D.C. (2011). The high cost of free parking, Updated edition. American Planning Association.

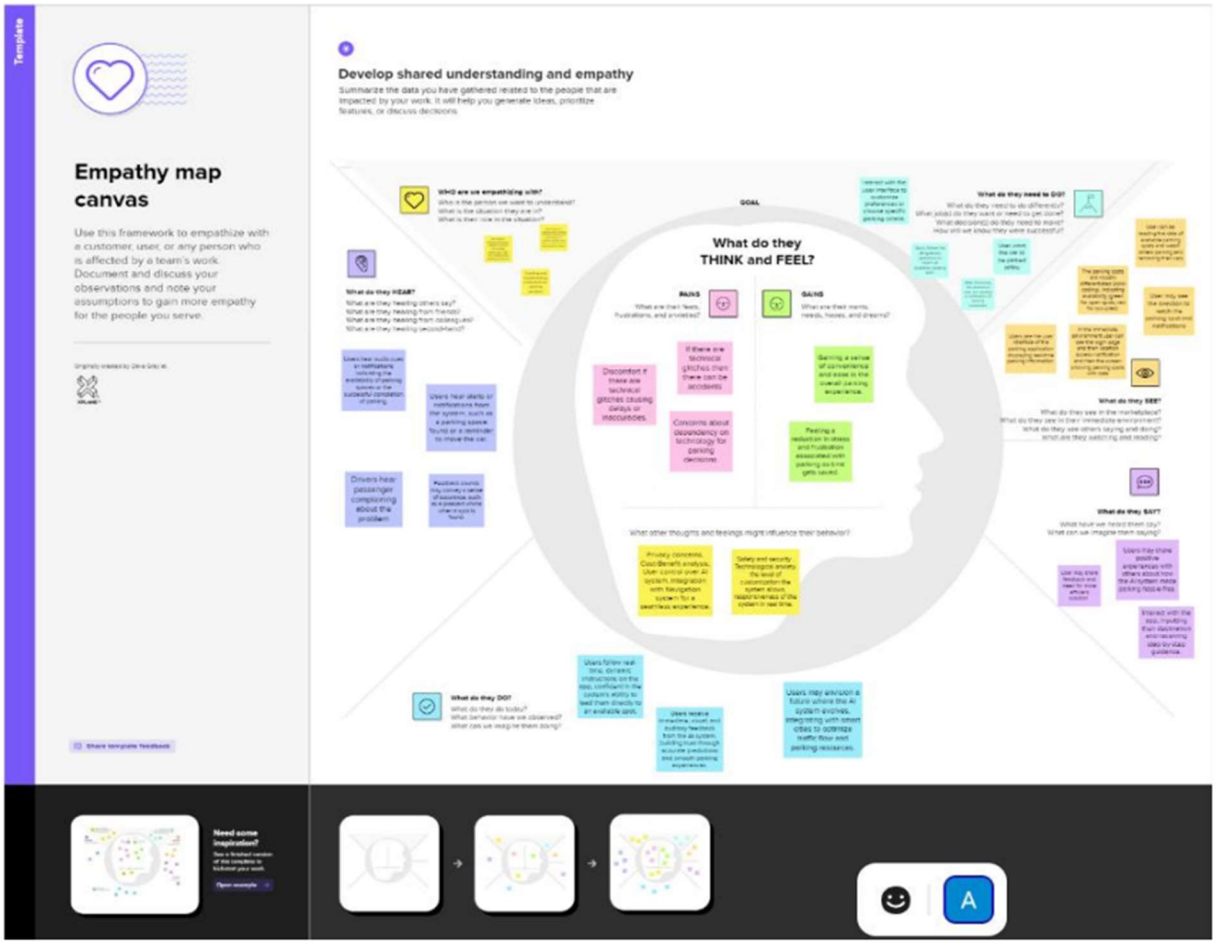
- Zhang, X., & Rilett, L. R. (2008). A simulation-based evaluation of the environmental impacts of parking guidance and information systems. Transportation Research Part C: Emerging Technologies, 16(2), 150-166.

2.3 Problem Statement Definition

In busy urban areas, finding available parking spaces during peak hours is a common challenge that leads to frustration and delays for drivers. The overcrowding of parking lots, especially at locations like shopping malls, creates a stressful and inefficient experience for motorists. To address this issue, there is a need for an AI-enabled car parking solution that utilizes OpenCV to automate the process of identifying and counting empty parking spaces in real-time, thus enhancing the convenience and efficiency of urban parking for drivers.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Person 1

security should get the maximum priority

Protect user data and ensure compliance with privacy regulations

ensuring that vehicles and pedestrians are protected throughout the parking process

Person 2

It is better to have a camera in the front and back of the car for good accuracy

Parking area size and allotment details must collected for further processing

These systems aim to make parking quicker and more convenient for users, so they should be user-friendly and efficient

Person 3

Cars size may vary for every individual customers so space allotment may differ

There must be different parking area for cars and bikes so that we can improve our accuracy

Be mindful of the data collected by AI systems and ensure it is handled securely and in compliance with privacy regulations

Person 4

Recognise free space after a car left the parking place

Once a parking slot is taken the database must update its value

Regular maintenance and backup systems are essential to prevent failures and ensure continuous operation of the system in parking facilities

Person 5

Person 6

Person 7

Person 8


4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.
Implementing voice commands may be less feasible and of lower importance than other user interfaces
Implementing license plate recognition is both important and feasible, as it enhances security and allows for efficient access control
Parking area size and allotment details must collected for further processing
effectively detecting available parking slots, contributing to efficient space management and convenience for users
Integrating weather and lighting sensors for automated adjustments is less feasible and may have lower importance compared to core functionalities
Displaying parking slot occupancy status is feasible and helpful but may have a lower priority compared to other features
Monitoring real-time vehicle count is essential for this case but may be less feasible due to computational requirements
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)


Implementing voice commands may be less feasible and of lower importance than other user interfaces
Implementing license plate recognition is both important and feasible, as it enhances security and allows for efficient access control
Parking area size and allotment details must collected for further processing
effectively detecting available parking slots, contributing to efficient space management and convenience for users
Integrating weather and lighting sensors for automated adjustments is less feasible and may have lower importance compared to core functionalities
Displaying parking slot occupancy status is feasible and helpful but may have a lower priority compared to other features
Monitoring real-time vehicle count is essential for this case but may be less feasible due to computational requirements
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In busy urban areas, finding available parking spaces during peak hours is a common challenge that leads to frustration and delays for drivers. The overcrowding of parking lots, especially at locations like shopping malls, creates a stressful and inefficient experience for motorists. To address this issue, there is a need for an AI-enabled car parking solution that utilizes OpenCV to automate the process of identifying and counting empty parking spaces in real-time, thus enhancing the convenience and efficiency of urban parking for drivers.
2.	Idea / Solution description	This solution utilizes Open CV for an AI-enabled parking system. It monitors the availability of parking space in real time using CCTV camera footage and counts available spaces by using a counting meter which will be available on the screen The system displays the parking lot with marked spaces (occupied or available) and provides a real-time count of open parking spots. The free space will be marked with green rectangle and the occupied space will be marked with red rectangle. This innovative approach streamlines the parking process, enhances convenience for drivers, and helps reduce congestion and frustration in busy urban areas. Fine-tuning and optimizations can further improve accuracy and effectiveness.
3.	Novelty / Uniqueness	**Real-Time Object Detection and Tracking:** The system's ability to perform real-time object detection and tracking using OpenCV distinguishes it from conventional parking systems. This feature enhances security, minimizes the risk of unauthorized access, and ensures efficient utilization of parking spaces. **Optimized Space Allocation:** The AI-driven approach can intelligently allocate parking spaces, considering factors such as vehicle size and available space. This optimization reduces congestion, improves traffic flow, and enhances the overall parking experience for users. **AI-Based Traffic Flow Analysis:** By analyzing the parking lot's traffic flow, the system can provide insights to optimize parking lot layout, reduce bottlenecks, and improve traffic management.
4.	Social Impact / Customer Satisfaction	**Reduced Traffic Congestion:** By optimizing parking space allocation and traffic flow, the AI-powered system helps reduce traffic congestion, making urban areas more efficient and less stressful for commuters. **Safety and Security:** Real-time object detection and tracking enhance parking lot security, reducing the risk of vehicle theft,

		vandalism, and unauthorized access. **Enhanced Accessibility:** Implementing designated accessible parking spots and providing automated assistance for individuals with disabilities makes parking facilities more inclusive and accessible for all members of the community.
5.	Business Model (Revenue Model)	In our business model, we will introduce a parking tag system. Initially, users must register for parking tags, ensuring they maintain a minimum balance in the tag to use it for parking. When a car is parked, the required parking fee will be deducted automatically from the parking tag, calculated based on the duration of time the car remains in the parking lot. Essentially, the parking tag functions as a user's digital wallet for paying parking fees, with charges proportional to the parking duration.
6.	Scalability of the Solution	**Advanced Machine Learning Models:** Fine-tuning and optimizing the machine learning models used by OpenCV enhance the system's accuracy and effectiveness. As the system scales, continuous refinement of these models ensures that the parking occupancy detection remains precise, providing reliable real-time information to drivers. **User-Friendly Interface:** The scalability of the solution is not limited to its backend architecture. The user interface, displaying parking lot availability with color-coded rectangles (green for available spaces, red for occupied spaces), is intuitive and easy to comprehend. Regardless of the parking lot's size, users can quickly interpret the information, promoting seamless scalability without compromising user experience. **Adaptive Hardware Infrastructure:** The system's architecture is designed to work seamlessly with varying hardware setups. This flexibility allows parking facility operators to scale the solution according to their needs, whether it's a small parking lot or a large urban parking complex. It can run on different hardware configurations, from basic cameras to high-end cameras and servers, depending on the scale of the operation.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Real-Time Monitoring: The system must continuously monitor parking space availability in real-time using CCTV camera footage.

Counting Meter: A counting meter should be displayed on the screen, accurately tallying open parking spaces and providing drivers with the latest count.

Parking Space Identification: The system should identify vacant parking spaces and mark them visually on a screen, distinguishing between occupied and available spaces.

User Interface: The project should have a user-friendly web-based interface using Flask, enabling drivers to access the parking information easily.

Accessibility: The system should be accessible from various devices, such as smartphones, tablets, and desktop computers, ensuring wide usability.

Scalability: The solution should be scalable to accommodate various parking lot sizes and layouts, from small lots to extensive parking structures.

Reliability: The system should maintain a high level of reliability to ensure that real-time data is accurate and consistently available to users.

Data Logging: The project should log and store historical parking data, allowing for analysis and reporting of parking space utilization trends over time.

Security: Ensure that the system is secure, protecting user data and preventing unauthorized access to the monitoring system.

4.2 Non-Functional requirements

Performance: The system must respond swiftly to changes in parking space availability, providing real-time updates to minimize delays for drivers.

Accuracy: The parking space identification and counting mechanisms must be highly accurate to prevent errors and ensure drivers receive reliable information.

Scalability: The system should handle varying numbers of cameras and parking spaces, adapting to different parking environments with ease.

Usability: The user interface must be intuitive and user-friendly, catering to both tech-savvy and non-tech-savvy users.

Compatibility: Ensure that the system is compatible with a range of CCTV camera models and configurations, enabling widespread adoption.

Security: Data security and privacy measures should be in place to safeguard the system from cyber threats and protect user information.

Availability: The system should have minimal downtime to provide continuous access to parking space information.

Maintainability: Regular maintenance and updates should be feasible to ensure the system remains functional and up-to-date.

Cost-Effectiveness: The project should be developed and operated within a reasonable budget, with cost-efficient solutions for hardware and software components.

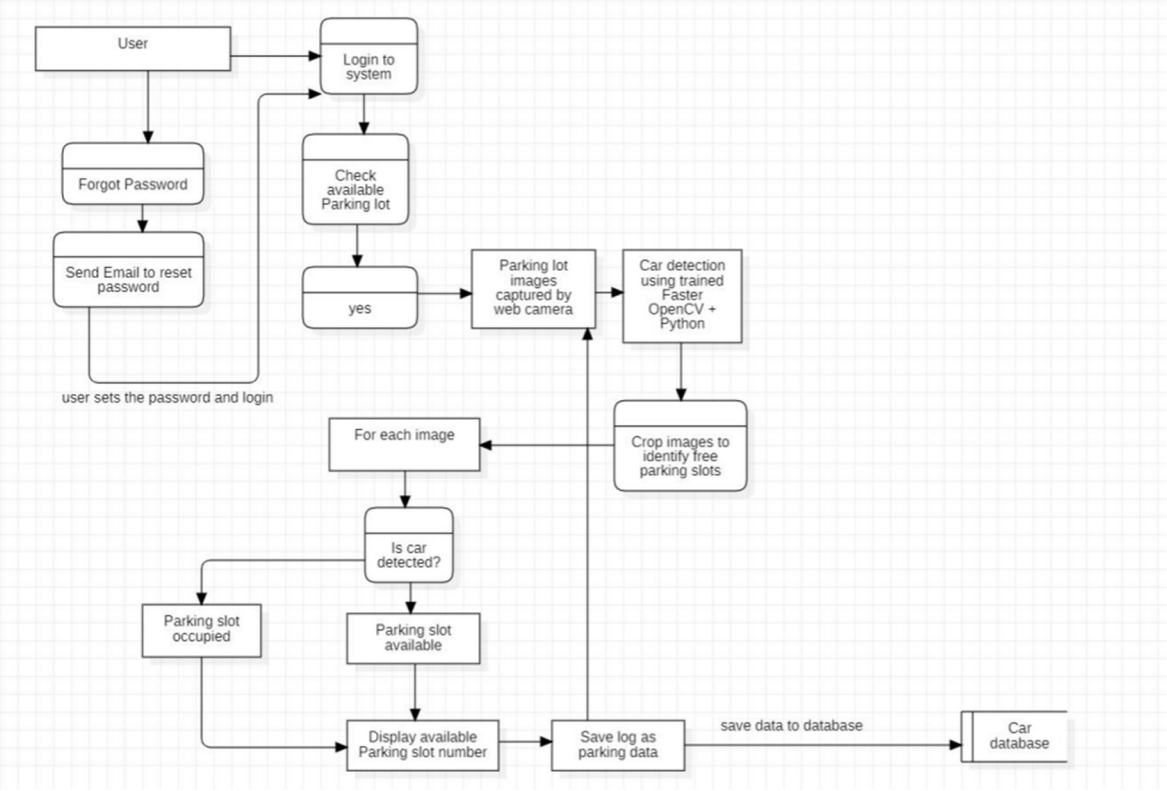
Adaptability: The system should be adaptable to different parking environments, allowing for customization to meet the specific needs of various facilities, such as shopping malls, office complexes,

and public parking lots.

Compliance: Ensure that the project complies with relevant data protection and privacy regulations to safeguard user data and maintain legal and ethical standards.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



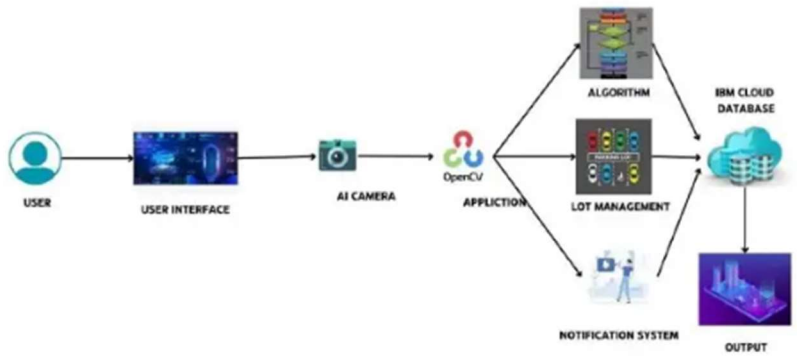
User Stories

Use the below template to list all the user stories for the product.

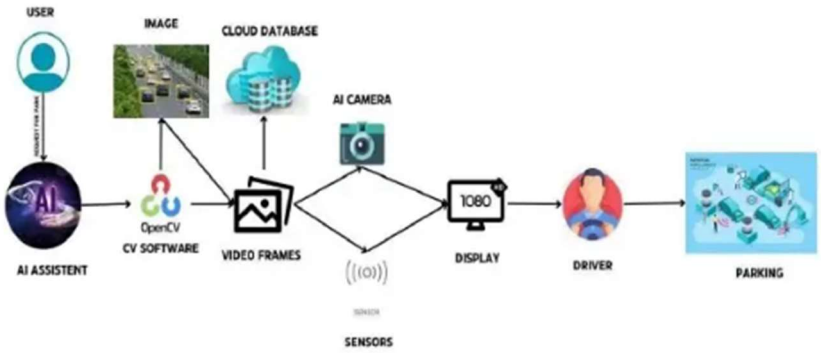
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register the app with email account	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register & access user profile/account with Gmail account	High	Sprint-1
	Requesting/ conferrer	USN-6	As a conferrer I can request vacant parking space to park my car and see the number of slots available	I can get information about parking rates	High	Sprint-2
Customer (Web user)	Profile	USN-7	As a user I can see registration page ,login page and request page to see available slots and camera footage screen where I can check availability of parking spots in real time	I can login through email and social media account for registration	Medium	Sprint-2
Customer Care Executive	Help desk/ user support	USN-8	As a customer care executive I can solve the queries of the users	I can reply to their queries and solve their related problems	High	Sprint -3
Administrator	Registration	USN-9	As a administration or I can view the database of registered users	I can check and verify the persons who are the registered their mail id's and information	Medium	Sprint -4
	Dash board	USN-10	As an administrator, I can view how many members requested for what trouble occurs in parking vehicle	I can Check the numbers of requirements and monitor the availability	Low	Sprint-4
Camera Screen	User Interface	USN-11	Car parking spots available number displayed, Spots available highlighted with green colour and car number changing continuously , screen shows the CCTV footage coverage of Parking area	I can get information of number of slots available and area in which slots available on the screen	High	Sprint-5

5.2 Solution Architecture

USER SIDE :



AI SIDE :



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

Technical Architecture:

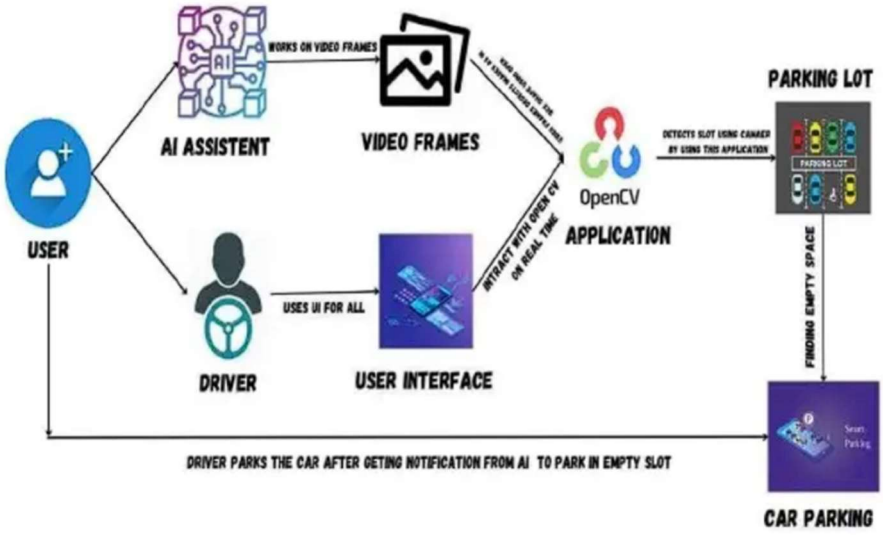


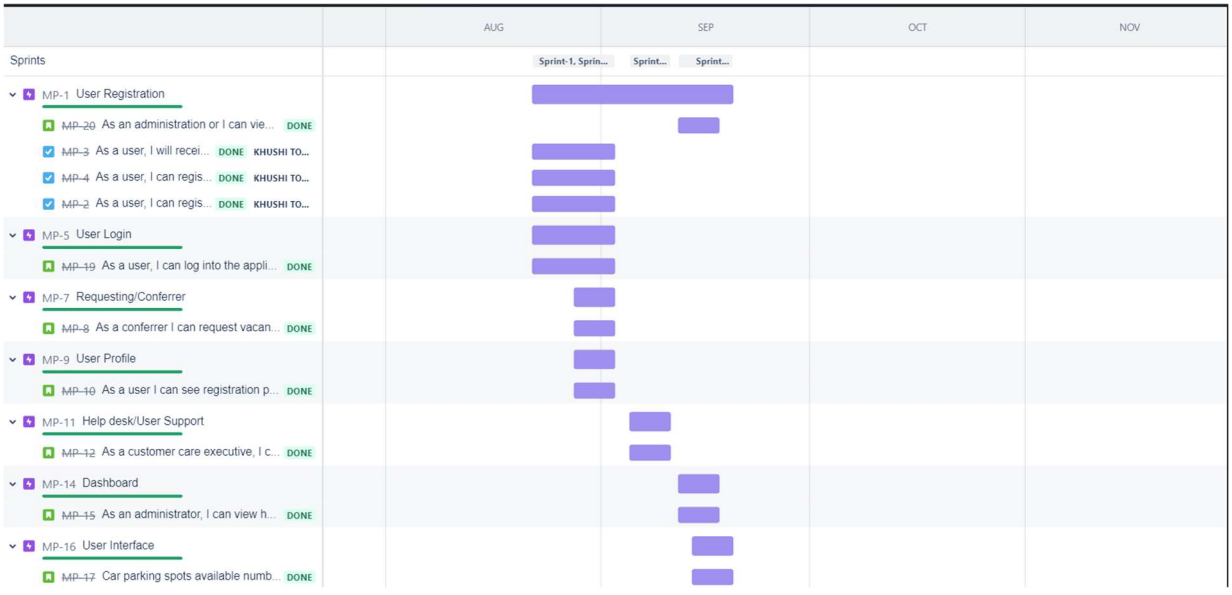
Table-1 : Components & Technologies:

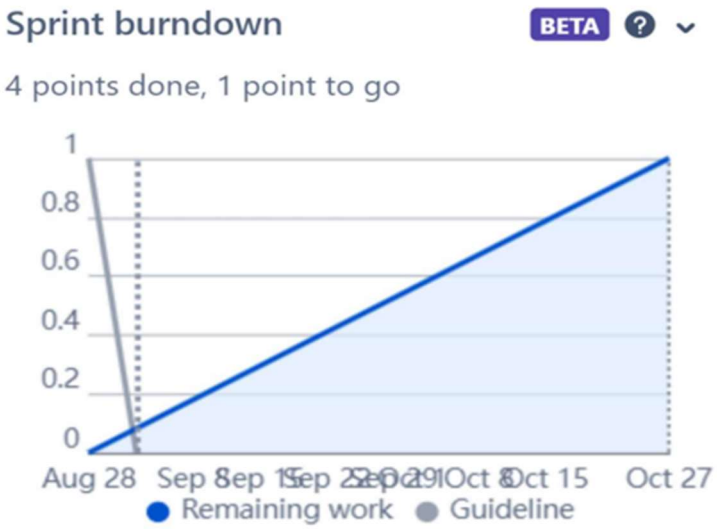
S.No	Component	Description	Technology
1.	User Interface	How user interacts with application eg web app	HTML,BootStrap
2.	Application Logic-1	Framework used for design the software	Python , python-flask
3.	Application Logic-2	Access the software in the car by the driver to detect spot	Python, Open CV
4.	Application Logic-3	Open cv is an open source platform for providing real time computer vision technology	Open CV
5.	Machine Learning Model	Uses test and trained data images and video to learn the environment	DNN (Deep Neural Network) model
6.	Infrastructure (server / cloud)	Application Development on Local system / cloud	Python-Flask

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	OpenCV, Flask	Technology of Opensource framework
2.	Scalable Architecture	Application Servers	Flask
3.	Availability	Local Website	Flask
4.	Performance	Real Time analysis with high accuracy	Open cv

6.2 Sprint Planning & Estimation





Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	5	5 Days	22 Aug 2023	02 Sept 2023	5	02 Sept 2023
Sprint-2	5	5 Days	28 Aug 2023	02 Sept 2023	5	02 Sept 2023
Sprint-3	3	5 Days	05 Sept 2023	10 Sept 2023	3	10 Sept 2023
Sprint-4	5	5 Days	12 Sept 2023	17 Sept 2023	5	17 Sept 2023
Sprint-5	2	5 Days	14 Sept 2023	19 Sept 2023	2	19 Sept 2023

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

AV = (5+5+3+5+2)/5 = 20/5 = **4**

6.3 Sprint Delivery Schedule

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	
Sprint-1		USN-4	As a user, I can register for the application through Gmail	1	Medium	
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	
Sprint-2	Requesting/conferrer	USN-6	As a conferrer I can request vacant parking space to park my car and see the number of slots available	1	High	
Sprint-2	Profile	USN-7	As a user I can see registration page, login page and request page to see available slots and camera footage screen where I can check availability of parking spots in real time	2	Medium	
Sprint -3	Help desk/ user support	USN-8	As a customer care executive, I can solve the queries of the users	3	High	
Sprint -4	Registration	USN-9	As an administration or I can view the database of registered users	2	Medium	
Sprint-4	Dash board	USN-10	As an administrator, I can view how many members requested for what trouble occurs in parking vehicle	3	Low	
Sprint-5	User Interface	USN-11	Car parking spots available number displayed, Spots available highlighted with green color and car number changing continuously, screen shows the CCTV footage coverage of Parking area	2	High	

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Flask:

- Web framework used for building web applications.
- Flask Basics: Click here

If you are using VScode, follow the below steps to download the required packages:

- Open VScode prompt.
- Type "pip install opencv-python" and click enter.
- Type "pip install cvzone" and click enter.
- Type "pip install Flask" and click enter

If you are using VScode IDE, you can install the packages through the command prompt and follow the same syntax as above.

Project objectives:

By the end of this project, you will:

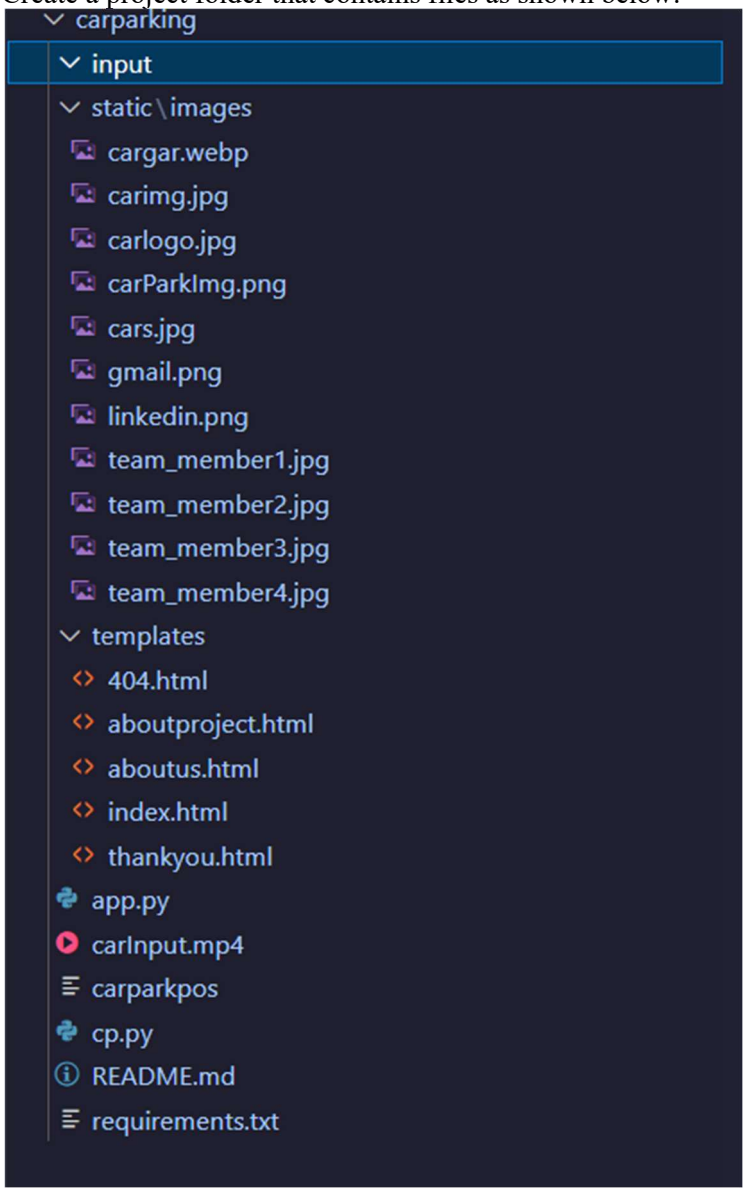
- Know fundamental concepts and techniques of computer vision (OpenCV).
- Gain a broad understanding of image thresholding.

Project Flow:

- Data Collection ○ Download the video
- Video Processing and object detection ○ Import required libraries ○ Checking for parking space ○ Looping the video ○ Frame processing and empty parking slot counters
- Application building ○ Build HTML
    - ○ Build python script for Flask

Project Structure:

Create a project folder that contains files as shown below:



1) Input folder will contain the uploaded video file which is then used by the ML model to do further Image processing.

2) The static folder contains a folder “image” which contains all the images used in the website.
3) The templates folder contains all the HTML files used in the website.
4) app.py is the main flask file from which we will be able to run the website on the local device.
5) Carinput.mp4 is the video on which the model is been trained on.

Milestone – 1: Data Collection

Activity 1: Download the video on the local device

Click the below link to download the video for AI-Enabled Car Parking using OpenCV. Click here.

Milestone – 2: ROI (Region of interest)

ROI stands for Region of Interest, which refers to a specific rectangular portion of an image or video frame that is used for processing or analysis.

Activity 1: Create a python file

Create a Python file in the directory and name it ‘cp.py’. This Python file is used for creating ROI and deleting ROI.

Activity 2: Importing the required packages

- **Opencv:** OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It supports multiple languages including Python, Java, and C++.
- **Pickle:** The pickle library in Python is used for serialization and de-serialization of Python objects. Serialization is the process of converting a Python object into a stream of bytes that can be stored in a file or sent over a network. De-serialization is the process of converting the serialized data back into a Python object.

```
import pickle
import cvzone
```

Activity 3: Define ROI width and height

- Calculating the ROI width and height (manually width and height are calculated and given as 107 & 48).
- In Python, try and except are used for error handling, to catch and handle exceptions that may occur during program execution. The try block is used to enclose the code that may raise an exception, and the except block is used to define what should happen if an exception is raised.

```
try:
    cap = cv2.VideoCapture('carparking/input/' + video_name)
    width, height = 107, 48
```

Activity 4: Select and deselect ROI

- A function is defined as mouseClick. As parameters we are passing events (mouse action), x (ROI starting point), y (ROI ending point), flags (Boolean flag), and params (other parameters).

- In 1st if condition: After the left click from the mouse, the starting and ending points will be added to posList by append method.
- In 2nd if condition: If ROI is selected in the unwanted region. Then we can remove that unwanted ROI by right-clicking from the mouse.
- Python objects are converted into a stream of bytes and stored in carparkpos file.

```
def mouseclick(events, x, y, flags, params):
    if events == cv2.EVENT_LBUTTONDOWN:
        poslist.append((x,y))
    if events==cv2.EVENT_RBUTTONDOWN:
        for i,pos in enumerate( poslist):
            x1,y1=pos;
            if x1<x<x1+width and y1<y<y1+height:
                poslist.pop(i)
    with open('carparkpos','wb') as f:
        pickle.dump(poslist,f)
```

Activity 5: Denote ROI with BBOX

- Reading the image with imread() method from cv2.
- All ROIs are saved in posList (refer activity 4).
- The rectangle is created as BBOX with the starting value (x) and ending value (y) from posList by rectangle() method. The parameters give image source, starting value, ending value, (starting value + width, ending value + height), (color), and thickness.
- For displaying the image imshow() method is used.
- setMouseCallback() function is used to perform some action on listening to the event which we have defined in activity 4.

```
while True:
    image = cv2.imread('carparking/images/carParkImg.png')
    cv2.rectangle(image,(50,190),(157,240),(255,0,255),2)

    for pos in poslist:
        cv2.rectangle(image, pos, (pos[0]+width,pos[1]+height), (255, 0, 255), 2)

    plt.imshow(image)
    cv2.setMouseCallback("image", mouseclick)
    cv2.waitKey(1)
```

Milestone - 3: Video processing and Object detection

Create a new Python file to perform video processing, object detection, and counters.

Activity 1: Import the required packages

Importing the required libraries to a new Python file.

```
import cv2
import pickle
import cvzone
import numpy as np
```

Activity 2: Reading input and loading ROI file

- VideoCapture() method from cv2 is used to capture the video input. Download the input video from milestone 1.
- Load the carparkpos file by the load() method from the pickle library. The carparkpos file is created from milestone 2. The ROI values are presented in the carparkpos file.
- Define width and height which we have used in milestone 2.

```
cap = cv2.VideoCapture('carparking/input/' + video_name)
width, height = 107, 48

with open('carparking/carparkpos', 'rb') as f:
    poslist = pickle.load(f)
```

Activity 3: Checking for parking space

- The function is defined with the name checkparkingspace. As a parameter, the image has to be passed.
- Taking the position from the position list and saving it to variables x and y.
- Cropping the image based on ROI (x and y value).
- To count the pixels from ROI, countNonZero() method is used from cv2. The BBOX (rectangle) is drawn in red color if the pixel count is lesser than 5000 else it is drawn in green color.
- The count of green color is displayed in the frame by putTextRect() method from the cvzone library.

```
def checkparkingspace(imgproceed):
    spacecouter=0;
    for pos in poslist:
        x,y=pos
        imgcrop=imgproceed[y:y+height,x:x+width]
        count = cv2.countNonZero(imgcrop)
        cvzone.putTextRect(image,str(count),(x,y+height-3),scale=1,thickness=2,offset=0,colorR=(0,0,255))

        if count>5000:
            color=(0,255,0)
            thickness=5
            spacecouter+=1
        else:
            color=(0,0,255)
            thickness=2
        cv2.rectangle(image, pos, (pos[0] + width, pos[1] + height), color, thickness)
    cvzone.putTextRect(image, f'Free:{spacecouter}/{len(poslist)}',(100, 50), scale=3, thickness=5, offset=20, colorR=(0, 200, 255))
```

Note: In this activity, function has been defined. But we have not called it.

Activity 4: Looping the video

- Here we calculate the total number of frames in the given video and we loop to each and every flame of the video to draw the rectangles around the parking slot. If no frames are left it will break.

```
while frame_count > 0:
    success, image = cap.read()

    if not success:
        break
```

Activity 5: Frame Processing and empty parking slot counters

- The captured video has to be read frame by frame. The read() method is used to read the frames.
- Opencv reads the frame in BGR (Blue Green Red).
- Converting BGR frame to grayscale image. The grayscale image is blurred with the GaussiamBlur() method from cv2.
- The adaptiveThreshold() method is used to apply a threshold to the blurred image. And again blur is applied to the image. Click here to understand about adaptiveThreshold().
- The dilate() method is used to compute the minimum pixel value by overlapping the kernel over the input image. The blurred image after thresholding and kernel are passed as the parameters.
- The checkparkingspace function is called with a dilated image. As per activity 3, we will get the count of empty parking slots.
- Display the frames in the form of video. The frame will wait for 10 seconds and it'll go to the next frame.

```
imggray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
imageblur = cv2.GaussianBlur(imggray, (3, 3), 1)
imgThreshold = cv2.adaptiveThreshold(imageblur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 25, 16)
imahemedian = cv2.medianBlur(imgThreshold, 5)
kernel = np.ones((3, 3), np.uint8)
imagedilate = cv2.dilate(imahemedian, kernel, iterations=1)

checkparkingspace(imagedilate)
cv2.imshow("output",image)

if cv2.waitKey(10) & 0xFF == ord('q'):
    break
frame_count -= 1
```

Milestone - 4: Application building

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he/she has to navigate to detect button. Then the video will be showcased on the UI.

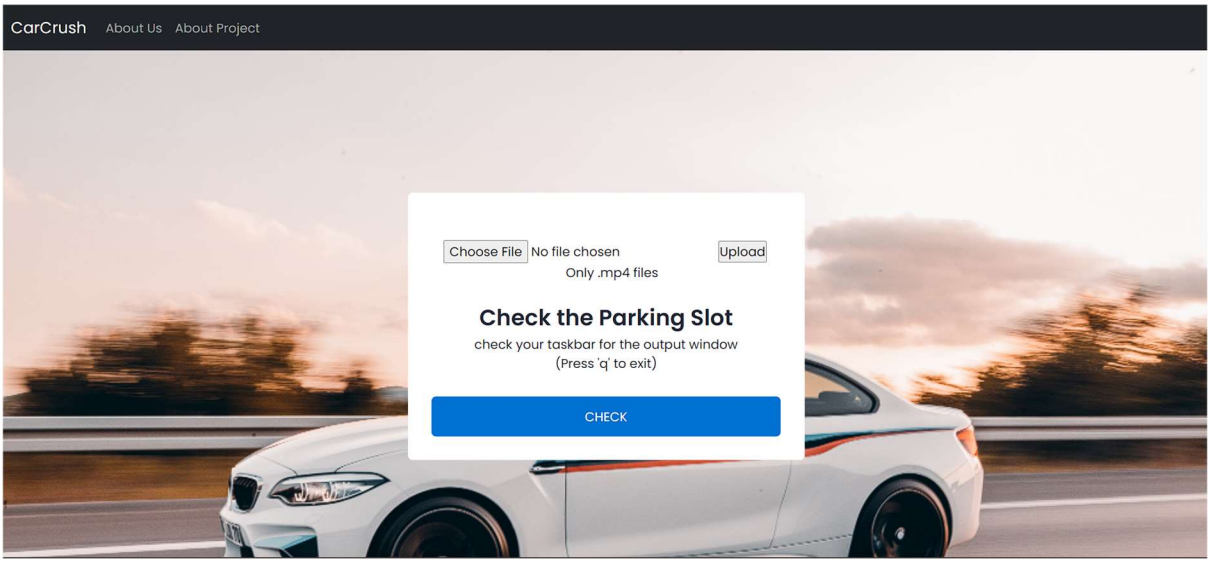
This section has the following tasks

- Building HTML Pages
- Building server-side script

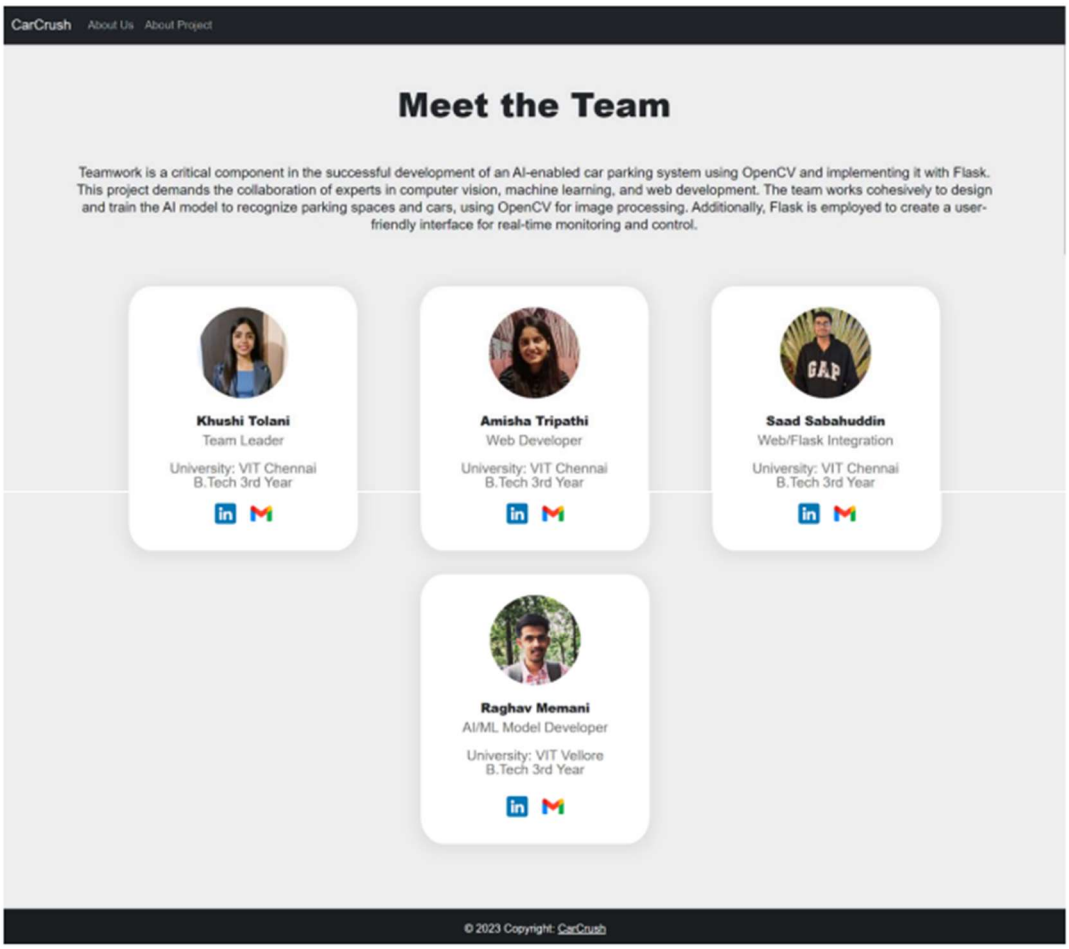
Activity1: Building Html Pages:

For this project, we have created 5 HTML files and saved them in the templates folder. Let's see what HTML pages look like:

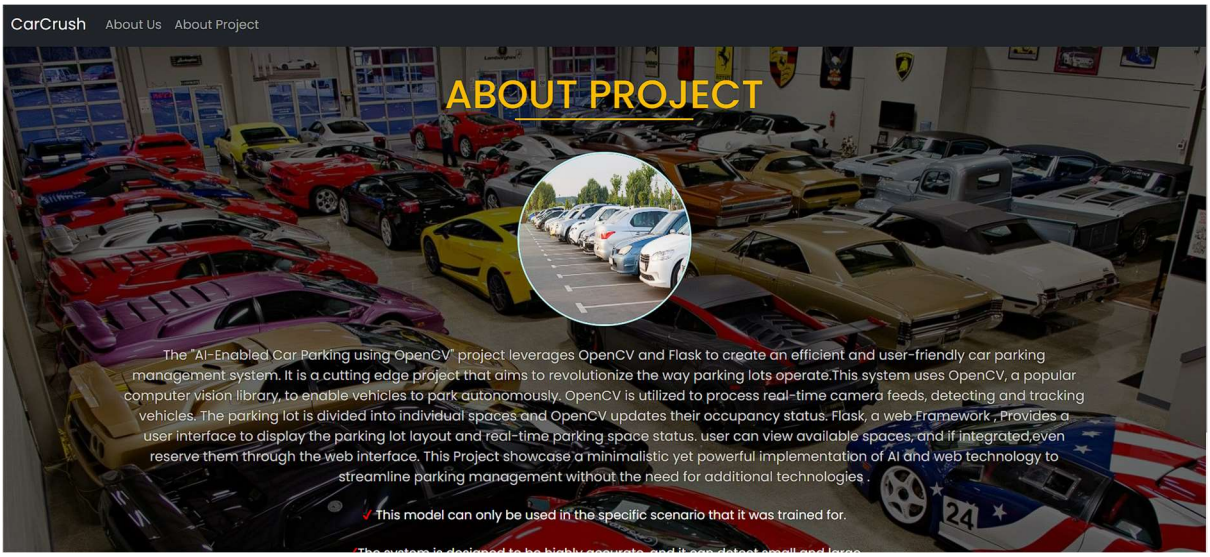
1) **main page:** It contains the home page button(CarCrush), About us, About Project, and upload button where you can upload your video(for the same scenario) and check how many slots are empty/full.



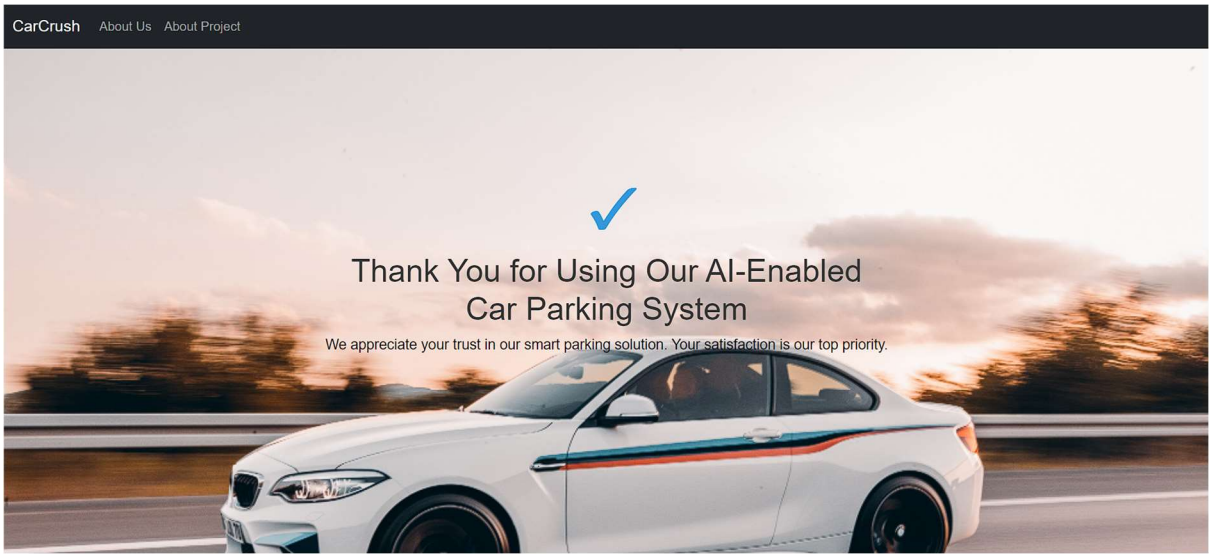
2) **About us:** In this we have given a brief description and the role played by each of them in the project, contact information.



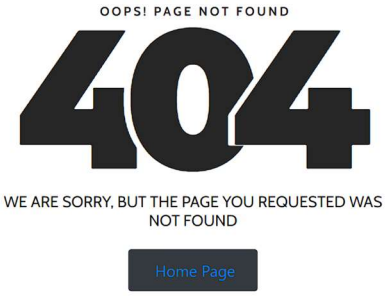
3) **About Project:** In this we have a brief explanation of our project and how it can be used further by integrating it with different technologies, some draw backs of this model/system.



4) **Thankyou page:**



5) **Error 404 page:** This is a customized 404 page not found page which is shown when you search for a directory which doesn't exist in the server.



- We have also added message box which helps the user to understand the error in a better manner.

Activity 2: Build Python code:

- Import the libraries

```
from flask import Flask, render_template, request, flash
import cv2
import pickle
import cvzone
import numpy as np
```

- Importing the flask module into the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as an argument.

```
app = Flask(__name__)
app.secret_key = 'your_secret_key'
```

Render HTML page:

- Here we will be using the declared constructor to route to the HTML page that we have created earlier. In the above example, the '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the HTML page will be rendered.

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/aboutus')
def aboutus():
    return render_template('aboutus.html')

@app.route('/aboutproject')
def aboutproject():
    return render_template('aboutproject.html')

@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html')
```

- create a decorator to route to '/aboutus goes to the About Us page, /aboutproject goes to the About Project page, errorhandler(404) goes to 404.html page.

Complete liv_pred() function:

```
def liv_pred():
    try:
        cap = cv2.VideoCapture('carparking/input/' + video_name)
        width, height = 107, 48

        with open('carparking/carparkpos', 'rb') as f:
            poslist = pickle.load(f)

        def checkparkingspace(imgproceed):
            spacecouter=0;
            for pos in poslist:
                x,y=pos
                imgcrop=imgproceed[y:y+height,x:x+width]
                count = cv2.countNonZero(imgcrop)
                cvzone.putTextRect(image,str(count),(x,y+height-3),scale=1,thickness=2,offset=0,colorR=(0,0,255))

                if count>5000:
                    color=(0,255,0)
                    thickness=5
                    spacecouter+=1
                else:
                    color=(0,0,255)
                    thickness=2
                cv2.rectangle(image, pos, (pos[0] + width, pos[1] + height), color, thickness)
            cvzone.putTextRect(image, f'Free:{spacecouter}/{len(poslist)}',(100, 50), scale=3, thickness=5, offset=20, colorR=(0, 200, 255))
```

```
    while frame_count > 0:
        success, image = cap.read()

        if not success:
            break

        imggray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        imageblur = cv2.GaussianBlur(imggray, (3, 3), 1)
        imgThreshold = cv2.adaptiveThreshold(imageblur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 25, 16)
        imahemedian = cv2.medianBlur(imgThreshold, 5)
        kernel = np.ones((3, 3), np.uint8)
        imagedilate = cv2.dilate(imahemedian, kernel, iterations=1)

        checkparkingspace(imagedilate)
        cv2.imshow("output",image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
        frame_count -= 1

    cap.release()
    cv2.destroyAllWindows()
```

- Main Function: Used to run the current module.

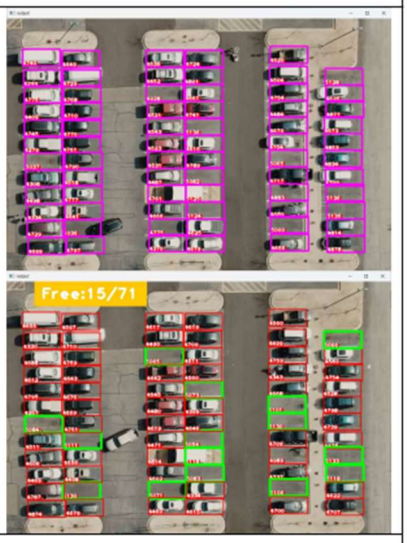
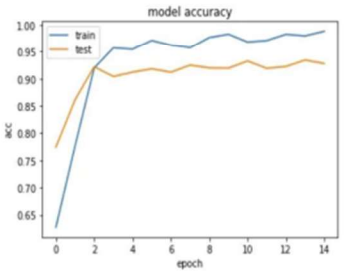
```
if __name__ == "__main__":
    app.run(debug=True)
```

Activity 3: Run the application

- Navigate to the folder where your python script is.
- Now type the “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the check button to see the result/prediction.

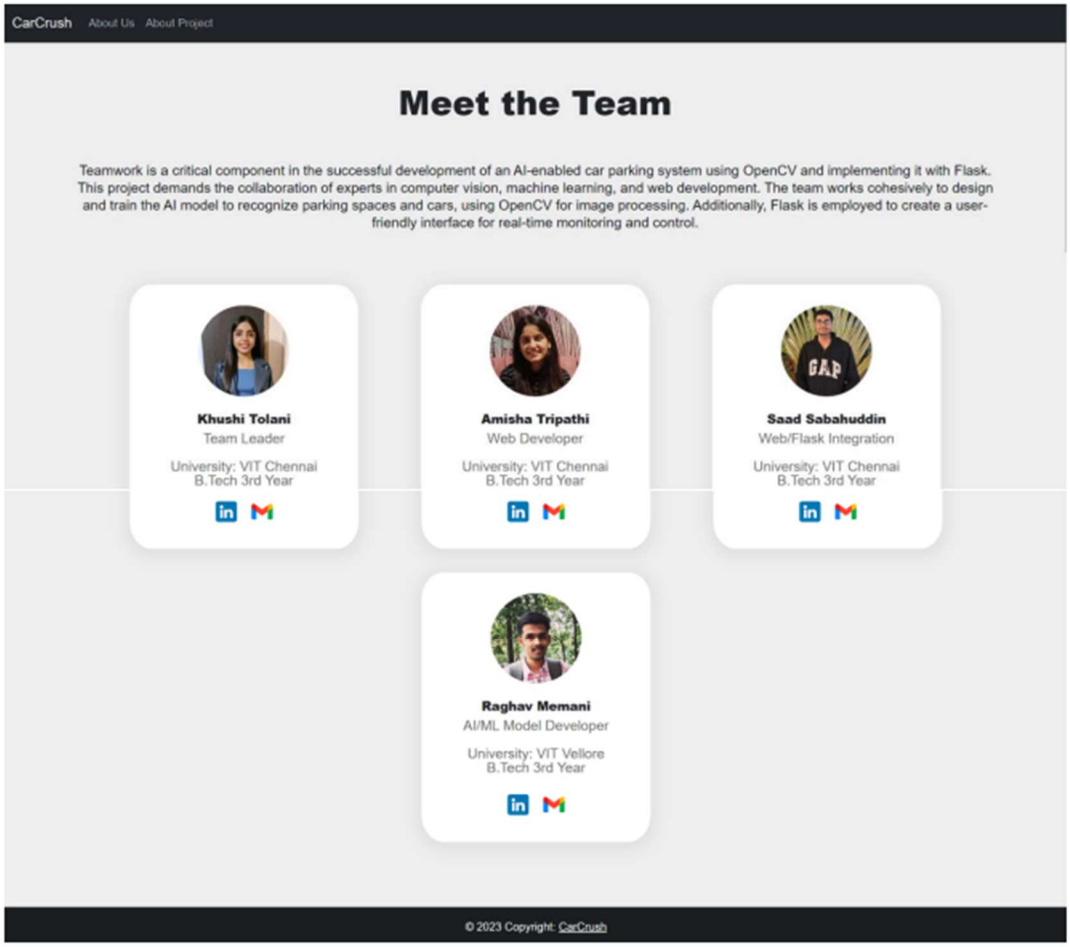
```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 135-972-108
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

8. PERFORMANCE TESTING

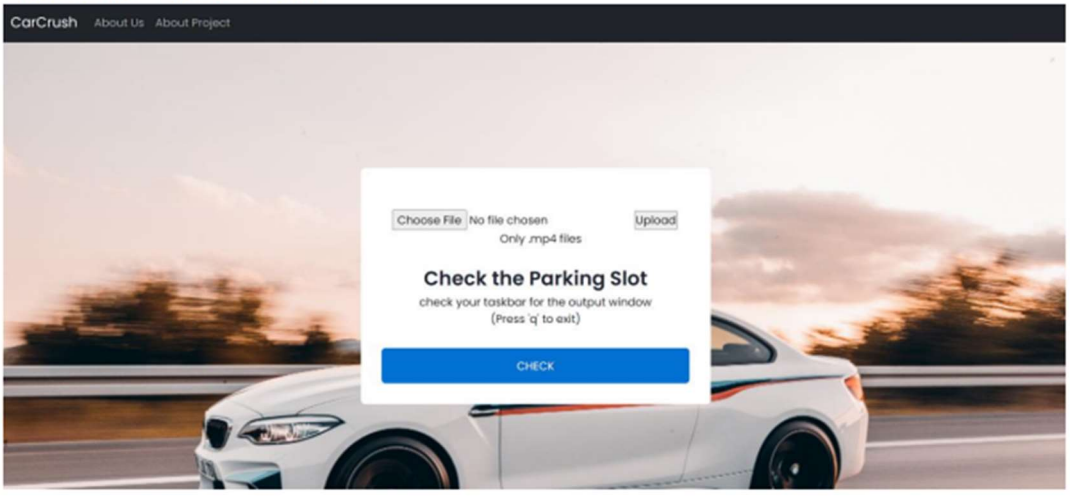
S.No.	Parameter	Values	Screenshot
1.	Model Summary	-	
2.	Accuracy	Training Accuracy – 98% Validation Accuracy – 94%	 CNN model test and train accuracies

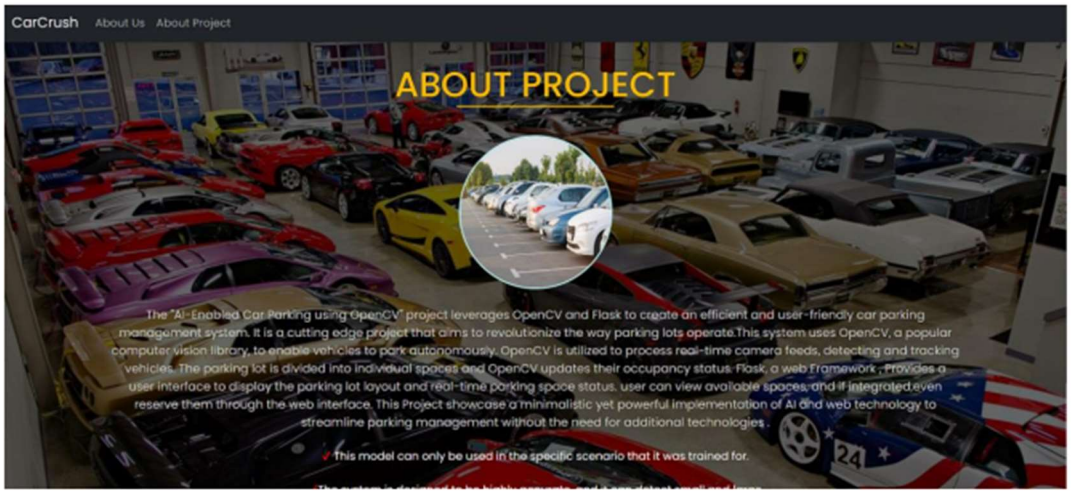
9. RESULTS

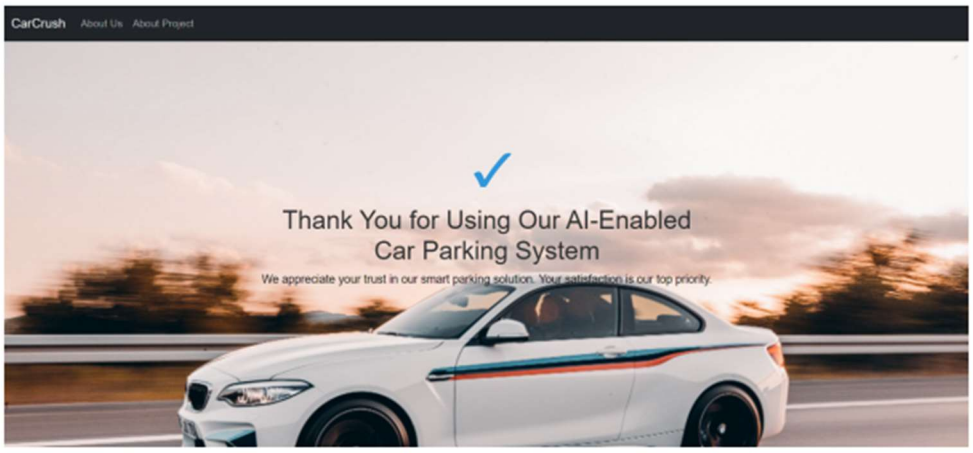
9.1 Output Screenshots

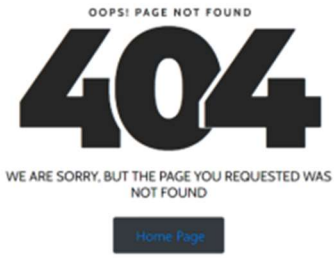


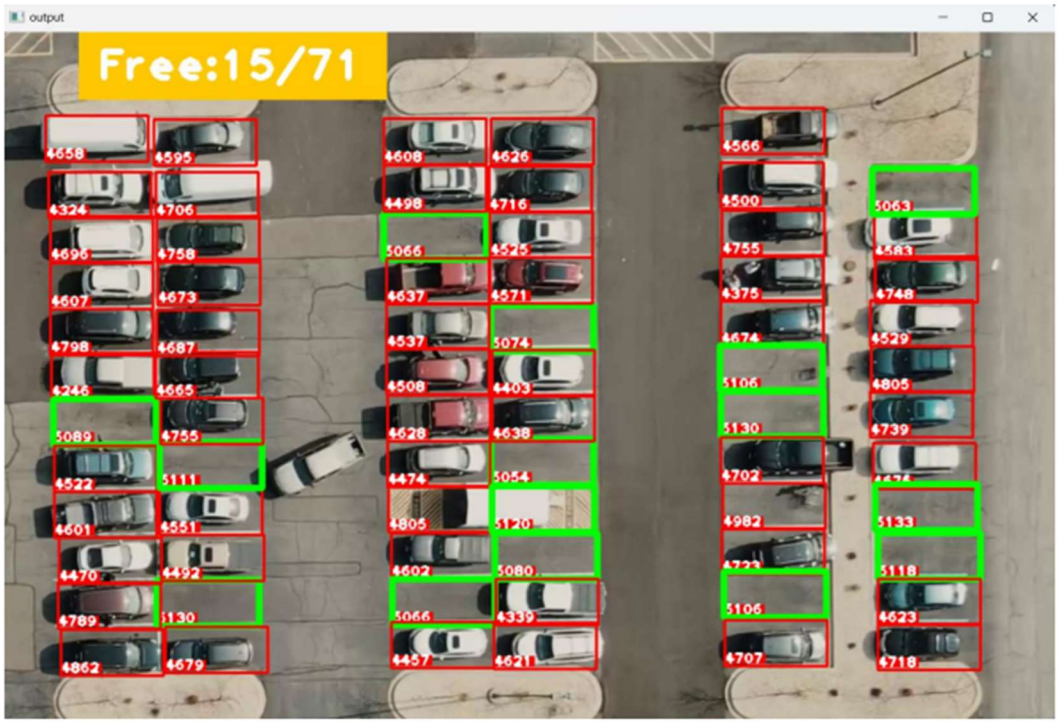
Click on the 'Check' button.




CarCrush
About Us
About Project
ABOUT PROJECT
The "AI-Enabled Car Parking using OpenCV" project leverages OpenCV and Flask to create an efficient and user-friendly car parking management system. It is a cutting edge project that aims to revolutionize the way parking lots operate.This system uses OpenCV, a popular computer vision library, to enable vehicles to park autonomously. OpenCV is utilized to process real-time camera feeds, detecting and tracking vehicles. The parking lot is divided into individual spaces and OpenCV updates their occupancy status. Flask, a web framework , Provides a user interface to display the parking lot layout and real-time parking space status. user can view available spaces, and if integrated,even reserve them through the web interface. This Project showcase a minimalistic yet powerful implementation of AI and web technology to streamline parking management without the need for additional technologies .
This model can only be used in the specific scenario that it was trained for.


CarCrush
About Us
About Project
Thank You for Using Our AI-Enabled Car Parking System
We appreciate your trust in our smart parking solution. Your satisfaction is our top priority.


OOPS! PAGE NOT FOUND
404
WE ARE SORRY, BUT THE PAGE YOU REQUESTED WAS NOT FOUND
Home Page



10. ADVANTAGES & DISADVANTAGES

Recent increments in car ownership and urbanization, coupled with poor city planning, have contributed greatly to the rising parking problems across the US. Fortunately, the adoption of AI in smart parking has completely revolutionized how we look for parking spaces. Here are a few advantages of smart parking for drivers and business owners.

Less fuel consumption

Driving around looking for a parking space consumes a lot of fuel. And considering the current fuel prices, that's a lot of money going down the drain.Smart parking solutions provide easy access to parking spots, thereby saving precious resources such as time, fuel, and space.

A reduction of search traffic on the streets

Nearly a third of traffic in urban areas is created by drivers looking for a parking spot. By leveraging AI-based smart parking solutions, municipalities can manage and reduce search traffic on busy streets. Smart parking solutions not only minimize search traffic but also smoothen traffic flow. The result? People spendless time looking for a parking spot since they already know where to get one.

Reduced parking stress

Looking for a good parking spot in a congested part of the city is simply overwhelming. You might find yourself driving across the same street several times, only to end up parking far away from your destination.AI-based smart parking solutions incorporate smart parking technologies with IoT devices so drivers can find parking spots easily using their smart phones and computers. This way, they can see all parking spots available in the area they plan to travel to long before they get there. Some parking lots even allow you to reserve a space. This means you don't have to drive around looking for a space to park.

Benefits of AI-based smart parking for businesses:

Surface parking lots take up 5% of all urban land in the US [7]. This means more competition among parking lot businesses. You'd think parking lots around busy streets get a lot of business, but sadly, that's not the case. In most cases, drivers can't find these parking lots, and when they do, there are tons of irregularly parked cars, making the parking lot inefficient.The result is that some drivers opt to drive longer distances for safer and more accessible parking lots, which

means less business for the better-situated
but poorly managed parking lots. Fortunately, by incorporating smart parking solutions in their management process, parking lot managers can improve their lots' efficiency, boost customer satisfaction and ultimately boost profits. Here area few other benefits parking lot businesses stand to gain from leveraging smart parking solutions.

Improved parking experience:
AI-based smart parking solutions leverage collected data to provide specialize dservices that ultimately lead to stress-free parking experiences. Besides letting nearby drivers know if there are available spots in the parking lot, managers canal so install digital signs that receive real-time data from parking lot management software to direct drivers to their parking spots.This eliminates frustration among drivers trying to find a spot within the parking lot, thus improving customer satisfaction Also, the mere fact that drivers don't have to drive around the parking lookingfor a free space means fewer emissions, which could improve the air quality inindoor parking lots.

Pinpoint inefficiencies in parking lot management:

Managing a parking lot takes a lot of work. For instance, parking lot managers often have to check the parking duration of certain vehicles or deal with irregularly and illegally parked vehicles. By leveraging real-time information from connected on-site devices, they can accurately determine the parking duration of all vehicles.This comes in handy, especially in parking lots offering short-time parking services like supermarkets. With this data, the managers can monitor excessive parking durations , which in most cases imply unauthorized use of their parking areas. They can also monitor spaces that stay empty for extended periods, which could imply an issue with the spot in question.

Optimized usage of the facility:

Through data collected from various sensors installed around the streets
and parking lot, businesses can monitor which areas have the highest and lowest parking traffic. With this data, they can better determine where to expand
or cut back operations. Sensor data also enables businesses to monitor misuse of emergency access roads and dedicated parking spots

DISADVANTAGES:
Smart parking solutions present a lot of advantages for both drivers and businesses. But they also present a few drawbacks that might cause some people to put back or even avoid incorporating them altogether. Here are some of the drawbacks of incorporating smart parking solutions.

High cost of installation
Numerous systems and technologies go into building an effective smart parking lot management system. Things like sensors, cameras, automated ticketing machines, and software cost a lot of money to install. Unfortunately, some businesses can't afford these systems, making it nearly impossible to in corporate the system.

Regular maintenance requirements
Despite being automated, smart parking management systems require regular maintenance to ensure smooth operation. The frequency of maintenance all comes down to the system in question, but most systems require monthly maintenance. This further racks up the running costs of the parking lot, which might have a significant impact on profits

11. CONCLUSION

As conclusion, the objectives of this project have been achieved. The hassle in searching for available parking slots has been completely eliminated. The designed system could be applied everywhere due to its ease of usage and effectiveness.It facilitates the problems of urban livability , transportation mobility and environment sustainability. This solution utilizes Open CV for an AI-enabled parking system. It monitors the availability of parking space in real time using CCTV camera footage and counts available spaces by using a counting meter which will be available on the screen The system displays the parking lot with marked spaces (occupied or available) and provides a real-time count of open parking spots. The free space will be marked with green rectangle and the occupied space will be marked with red rectangle. This innovative approach streamlines the parking process, enhances convenience for drivers, and helps reduce congestion and frustration in busy urban areas. Fine-tuning and optimizations can further improve accuracy and effectiveness.

12. FUTURE SCOPE

The smart parking management system can be broadly applied for many future applications. Apart from its basic role of parking management of cars it can also be applied for plane and ship and fleet management.The future scope of the AI-Enabled Car Parking System utilizing OpenCV and Flask is brimming with possibilities for further development and enhancement. Beyond its initial implementation, this project can take on a multi-faceted role in shaping the future of urban mobility. Integration with other smart city technologies, like traffic management systems, opens the door to a comprehensive urban mobility ecosystem, enabling dynamic traffic rerouting based on real-time parking availability data. Predictive analytics can be incorporated to forecast parking space availability, assisting drivers in planning their journeys and minimizing the need for last-minute parking spot searches. Further development may include the integration of payment gateways for cashless and contactless parking payments, making the parking process even more convenient and streamlined. Smart parking reservations can become a reality, allowing users to reserve parking spaces in advance, particularly in high-demand areas. Moreover, the system can evolve to offer valuable insights into the environmental impact of parking, contributing to a more environmentally sustainable urban environment and aiding city planners in making data-driven decisions. Accessibility can be enhanced through adaptive user interfaces, customized for different user groups, such as drivers with disabilities, to improve overall accessibility. Continued improvements in security, machine learning optimization, and user feedback integration are also viable directions for future development, ensuring the system remains at the cutting edge of technology and user experience. With global scalability in mind, this project can address parking challenges in various urban settings worldwide, enhancing urban mobility and sustainability. The possibilities for further development are extensive, making this project an exciting step toward the future of urban parking and mobility.

13. APPENDIX

Source Code

GitHub & Project Demo Link

https://github.com/smartinternz02/SI-GuidedProject-600003-1697570742

https://youtu.be/9Js_XhVlEMQ