

AI ENABLE CAR PARKING USING OPENCV



TEAM ID: Team-593057

COLLEGE NAME: VELLORE INSTITUTE OF TECHNOLOGY

TEAM MEMBERS:

1. Ssathyan S R (TEAM LEADER)
2. Ashish Kumar Pandey
3. Shriyansh Raj Timsina
4. PARTHIB DEY

Mentor(s) Name: Saumya Mohandas

1.INTRODUCTION:

In the realm of smart urban infrastructure, AI-enabled car parking using OpenCV integrates advanced computer vision technology to revolutionize the parking experience. OpenCV (Open-Source Computer Vision Library) is employed to analyse real-time video feeds from parking lots, enabling automated detection of available parking spaces, monitoring occupancy, and facilitating seamless user interactions. This innovative solution enhances operational efficiency, reduces congestion, and introduces a data-driven approach to optimize parking resources.

1.1 Project Overview

S.NO	TOPICS
1	INTRODUCTION
2	LITERATURE SURVEY
3	IDEATION & PROPOSED SOLUTION
4	REQUIREMENTS ANALYSIS
5	PROJECT DESIGN
6	PROJECT PLANNING AND SCHEDULING
7	CODING AND SOLUTIONING
8	PERFORMANCE TESTING
9	RESULTS
10	ADVANTAGES & DISADVANTAGES
11	CONCLUSION
12	FUTURE SCOPE
13	APPENDIX

1.2 PURPOSE:

The purpose of implementing AI-enabled car parking using OpenCV is to leverage computer vision technology for efficient and automated parking management. By analysing real-time video data, the system aims to streamline parking operations, enhance user experience, reduce congestion, and optimize resource utilization, ultimately contributing to smart and sustainable urban mobility.

1. Efficient Resource Utilization:

- Optimizes parking space allocation by accurately identifying and managing available parking spots in real-time.

2. Enhanced User Experience:

- Provides a seamless and user-friendly parking experience with automated space detection, guidance, and streamlined payment processes.

3. Reduced Congestion:

- Mitigates traffic congestion by directing drivers to available parking spaces promptly, minimizing the time spent searching for parking.

4. Data-Driven Insights:

- Utilizes data analytics to offer valuable insights into parking patterns, occupancy rates, and overall parking lot performance.

5. Improved Urban Mobility:

- Contributes to smart city initiatives by promoting sustainable urban mobility through the efficient use of parking resources.

6. Cost Optimization:

- Helps businesses and municipalities optimize parking infrastructure costs by ensuring efficient use of available spaces.

7. Automation of Operations:

- Automates key parking management tasks, including occupancy monitoring, payment processing, and user notifications.

8. Enhanced Security:

- Augments security through surveillance capabilities, ensuring a safer parking environment with real-time monitoring.

9. Environmentally Friendly:

- Supports eco-friendly initiatives by reducing vehicle idling time and associated emissions through quicker parking space allocation.

10.Future-Ready Infrastructure:

- Positions urban areas for future growth and technological advancements by incorporating AI-driven solutions into parking management systems.

2. LITERATURE SURVEY

2.1 Existing problem

1. Accuracy Challenges:

- Difficulty in achieving consistently high accuracy in parking space detection, especially in complex environments with varying lighting conditions and diverse vehicle types.

2. Scale and Integration:

- Scaling AI-enabled parking solutions for large parking lots or integrating them seamlessly with existing infrastructure poses technical and logistical challenges.

3. Cost of Implementation:

- The initial cost of implementing AI-based parking systems, including hardware, software, and installation, may be a barrier for widespread adoption.

4. Maintenance and Reliability:

- Ensuring continuous system reliability and addressing maintenance issues, such as camera malfunctions or algorithm updates, can be demanding.

5. Privacy Concerns:

- Increasing concerns related to privacy and data security, particularly when deploying surveillance technologies that capture and process real-time video data.

6. Integration with IoT:

- Integration challenges with other smart city technologies and IoT devices, hindering the creation of a comprehensive and interconnected urban infrastructure.

7. User Adoption and Awareness:

- Limited awareness and adoption among users, leading to resistance or lack of understanding regarding the benefits and functionality of AI-enabled parking solutions.

8. Regulatory Compliance:

- Navigating complex regulatory environments and ensuring compliance with data protection laws and standards can be a significant challenge.

9. Adaptation to Dynamic Environments:

- Difficulty in adapting to dynamic parking scenarios, such as events or festivals, where parking patterns change rapidly and unpredictably.

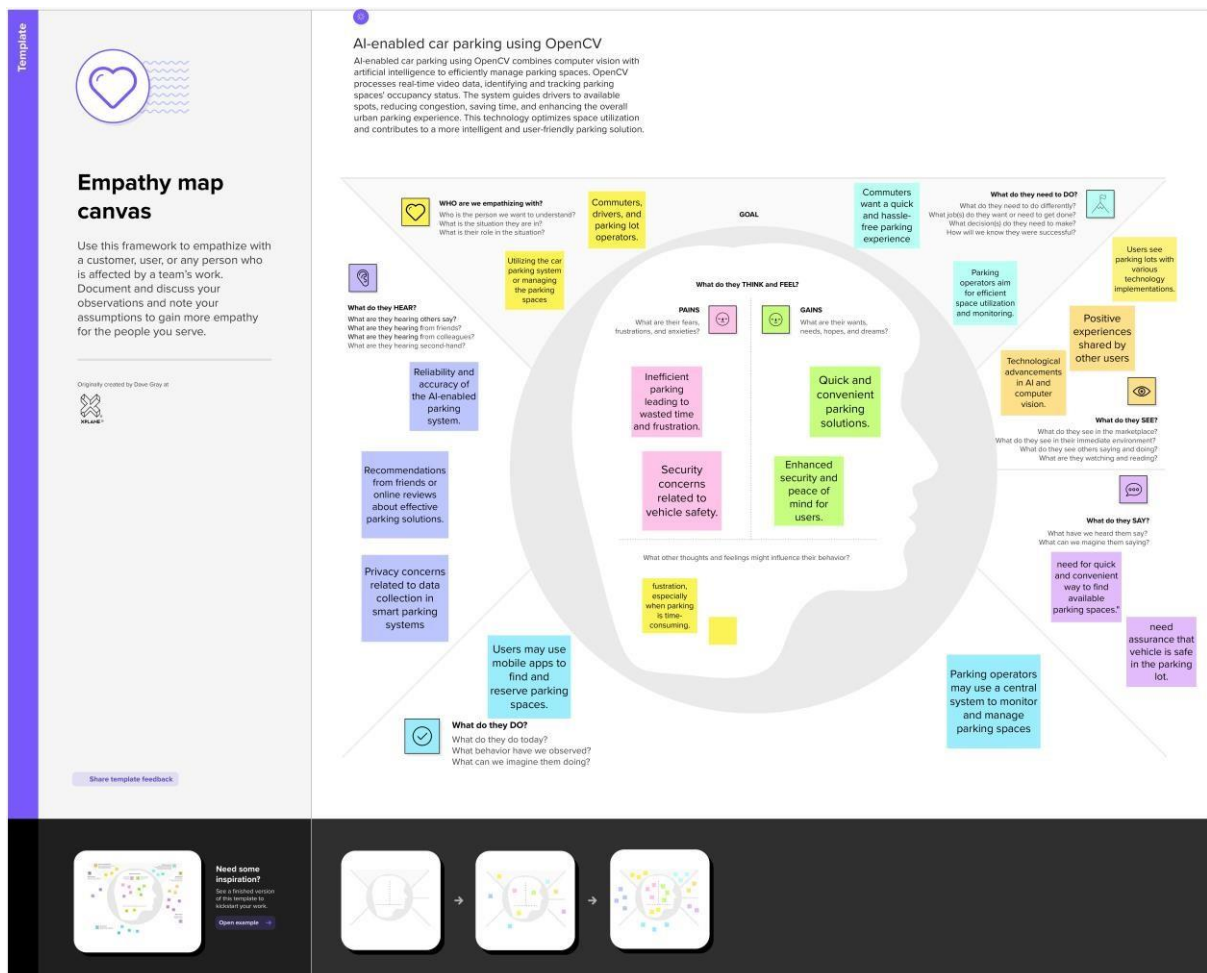
10. Energy Consumption:

- Addressing energy consumption concerns, particularly in the case of continuously operating surveillance systems, to align with sustainability goals

2.2 Problem Statement Definition

Develop an AI-driven car parking system using OpenCV to address the challenges of accurate and scalable parking space detection, integration with existing infrastructure, high initial implementation costs, and privacy concerns. The solution should prioritize user-friendly interfaces, seamless integration with smart city technologies, and compliance with regulatory frameworks while aiming to enhance overall urban mobility and parking management efficiency.

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Step 1: Brainstorm, Idea Listening and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

SSATHYAN S R

Easy to use and visually appealing user interface

Deep learning methods like CNN to build and train model

Develop community for the app so that users can contribute to the validation process

PARTHIB DEY

Multilingual website/app service to users

Retrain model with new datasets and update algorithms

Use ensemble learning strategies like bagging and boosting for better accuracy

ASHISH PANDEY

Use heatmap for correlation visualization

Use box plot for identifying any outliers

Improve accuracy and check for performance degradation

SHRIYANSH

Use flask framework for web development

Add extra features for the model

Optimized model for better accuracy

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

CNN to build and train model

Improve accuracy and check performance degradation

Regularly retrain model with new training datasets and updating algorithms

Easy to use and visually appealing UI

Step 2: Idea Prioritization

4

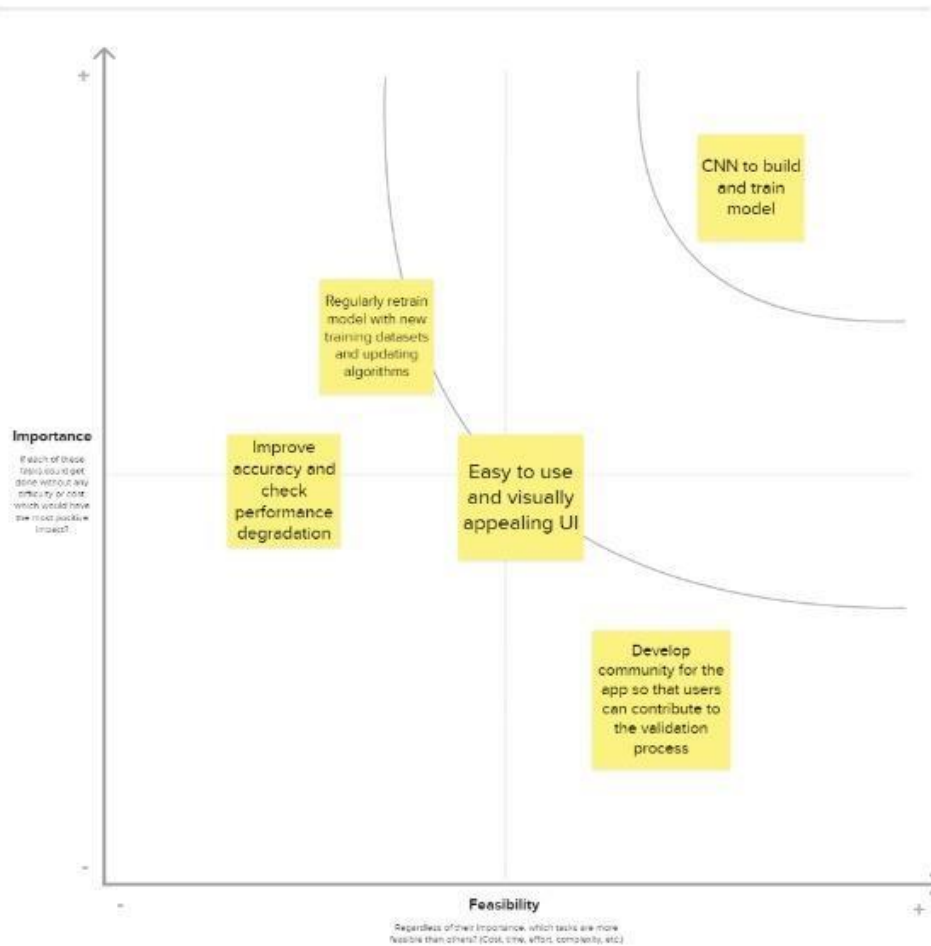
Prioritize

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.



3.3 Proposed Solution

S. No	Parameter	Description
1.	Problem Statement (Problem to be solved)	To find the free parking slot in a minimum distance from a starting point.
2.	Idea / Solution description	The idea of this project to find the difference between empty slot and occupied slot and given numbers for each slot in ascending order to find minimum distance unoccupied slot.
3.	Novelty / Uniqueness	By above approach the parking slot is segregated as occupied and unoccupied slots
4.	Social Impact / Customer Satisfaction	This technique will reduce the time taken park their car and thereby improving the customer satisfaction.
5.	Business Model (Revenue Model)	Generates revenue through subscriptionbased parking management services, offering real-time space availability, automated payments, and analytics for optimized parking operations.
6.	Scalability of the Solution	The outcome of this project will be very helpful in parking management system.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

1. Parking Space Detection:

- The system must accurately identify and monitor available parking spaces in real-time using OpenCV computer vision algorithms.

2. User Interface:

- Provide a user-friendly interface for drivers to access real-time parking availability information, receive guidance, and complete payment transactions.

3. Automated Guidance System:

- Implement an automated guidance system that directs drivers to the nearest available parking spaces based on real-time data.

4. Integration with Payment Systems:

- Integrate with payment gateways for seamless and automated payment processing, supporting various payment methods.

5. Occupancy Tracking:

- Track and display the occupancy status of the parking lot, including the number of available and occupied spaces.

6. Data Analytics and Reporting:

- Implement data analytics capabilities to generate reports on parking patterns, usage trends, and system performance for informed decision-making.

7. Security and Surveillance:

- Incorporate security features, such as surveillance cameras, to ensure the safety of parked vehicles and the parking environment.

8. Scalability:

- Design the system to scale effectively, accommodating various parking lot sizes and configurations without compromising performance.

9. Real-time Alerts and Notifications:

- Provide real-time alerts and notifications to users regarding parking availability, successful payments, and other relevant information.

10. Integration with Smart City Infrastructure:

- Ensure seamless integration with existing smart city infrastructure, such as traffic management systems and IoT devices.

11. Privacy Controls:

- Implement privacy controls to safeguard user data and adhere to data protection regulations.

12. Adaptability to Dynamic Environments:

- Enable the system to adapt to dynamic parking scenarios, such as events or peak traffic periods.

13. Compliance with Regulatory Standards:

- Ensure compliance with local and national regulations related to data privacy, security, and parking management.

14. Energy-Efficiency Measures:

- Implement energy-efficient measures, such as intelligent camera activation and deactivation, to minimize energy consumption.

15. System Maintenance and Updates:

- Facilitate easy maintenance, troubleshooting, and regular updates to address system issues and incorporate improvements.

4.2 Non-Functional requirements

1. Performance:

- The system must process and analyze video data in real-time, ensuring low latency in parking space detection and guidance.

2. Scalability:

- The solution should scale horizontally to accommodate varying parking lot sizes and handle increased user loads during peak hours or events.

3. Reliability:

- The system must operate reliably with minimal downtime, ensuring continuous availability for users.

4. Usability:

- The user interface should be intuitive, requiring minimal training for both drivers and parking management personnel.

5. Security:

- Implement robust security measures to safeguard user data, prevent unauthorized access, and ensure the integrity of the parking system.

6. Privacy:

- Ensure that the system adheres to privacy regulations, anonymizes data when necessary, and protects the privacy of individuals using the parking facility.

7. Compatibility:

- The solution should be compatible with various devices, browsers, and operating systems to maximize accessibility for users.

8. Maintainability:

- Design the system for ease of maintenance, allowing for quick troubleshooting, updates, and enhancements without significant disruption.

9. Interoperability:

- Ensure interoperability with other smart city technologies and infrastructure, fostering a cohesive and integrated urban ecosystem.

10. Compliance:

- The system must comply with relevant industry standards, data protection laws, and regulations governing parking management solutions.

11. Energy Efficiency:

- Implement energy-efficient practices, such as optimizing camera usage and processing algorithms, to minimize environmental impact.

12. Reliability:

- Ensure that the system can handle various environmental conditions, such as changes in lighting, weather, and traffic volume.

13. Response Time:

- The system should provide quick response times for user queries, alerts, and guidance to enhance the overall user experience.

14. Documentation:

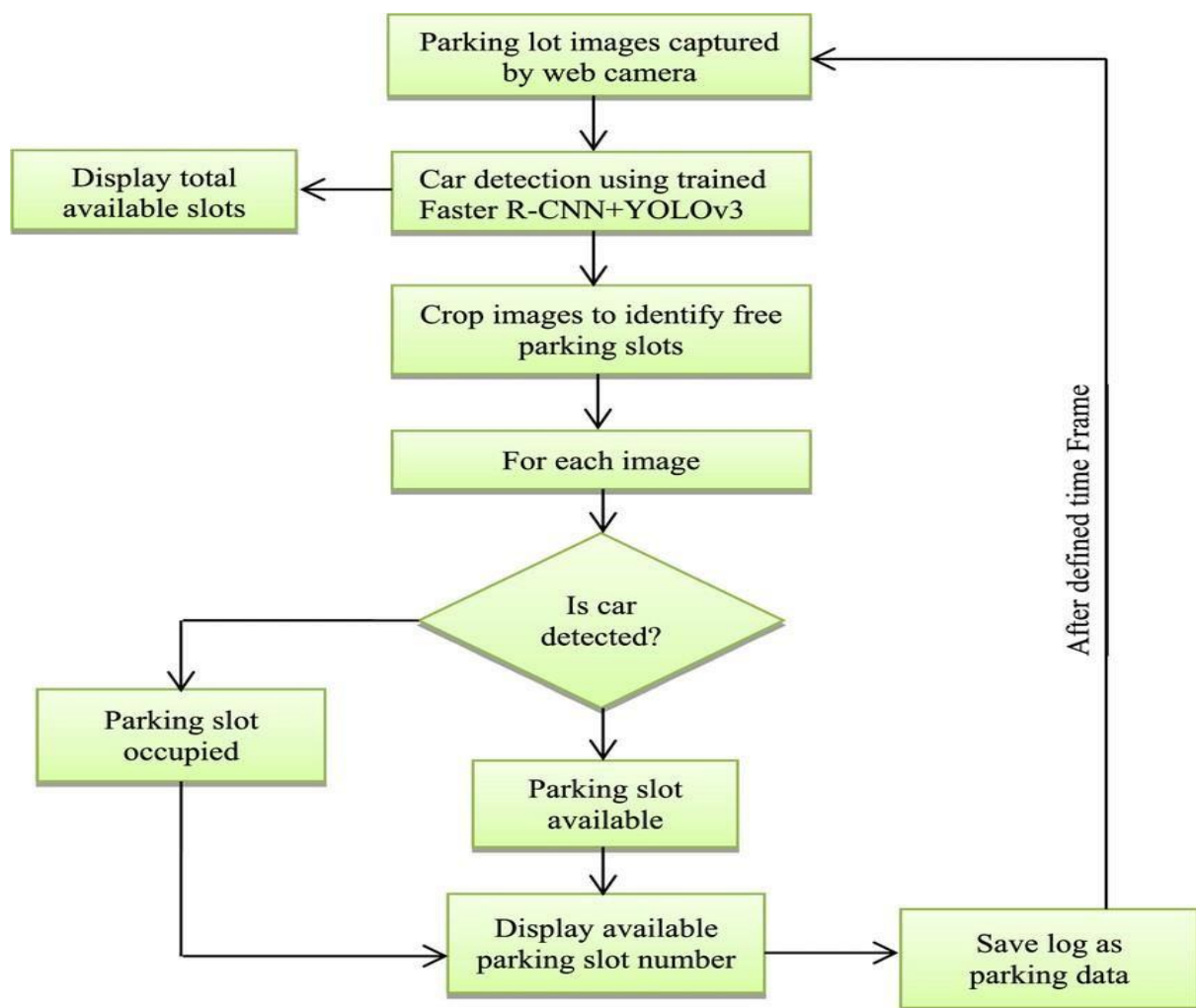
- Comprehensive documentation should be available for administrators, developers, and end-users to understand system functionality, configuration, and troubleshooting procedures.

15. Regulatory Compliance:

- The system must comply with relevant laws and regulations related to parking management, surveillance, and data handling in the specific geographic location of deployment.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



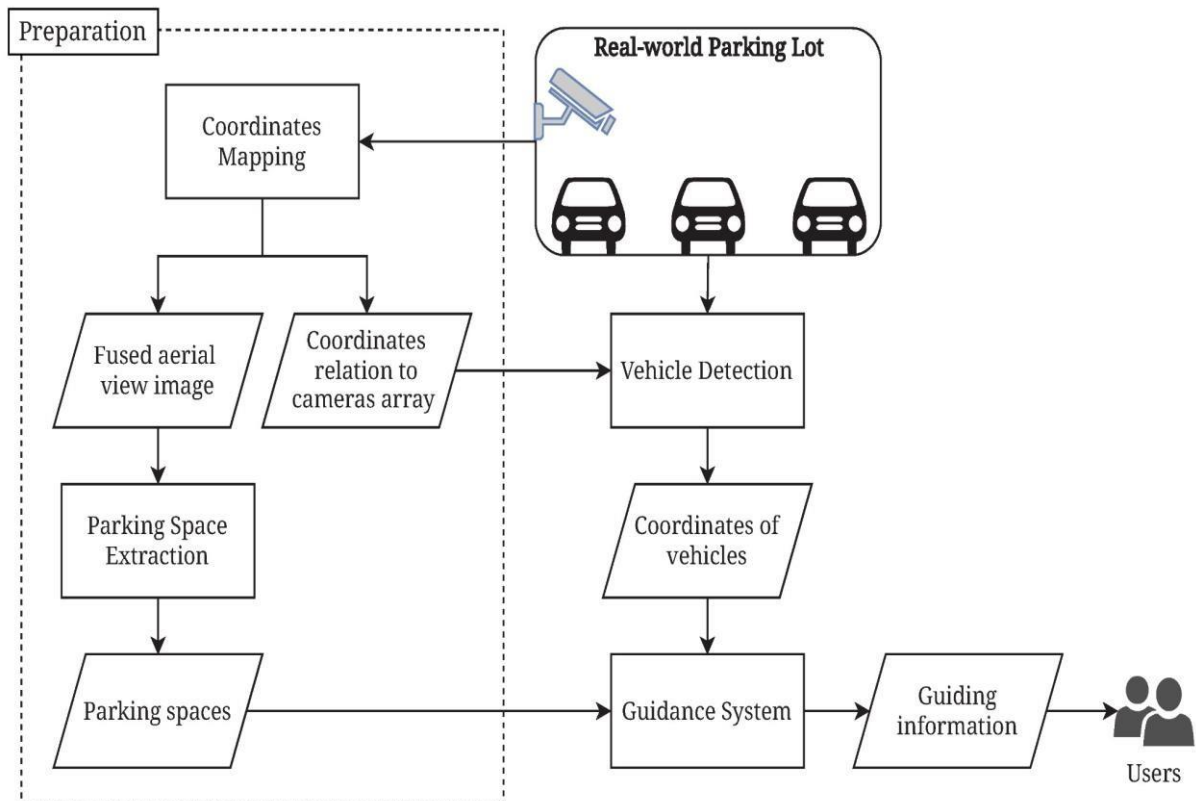
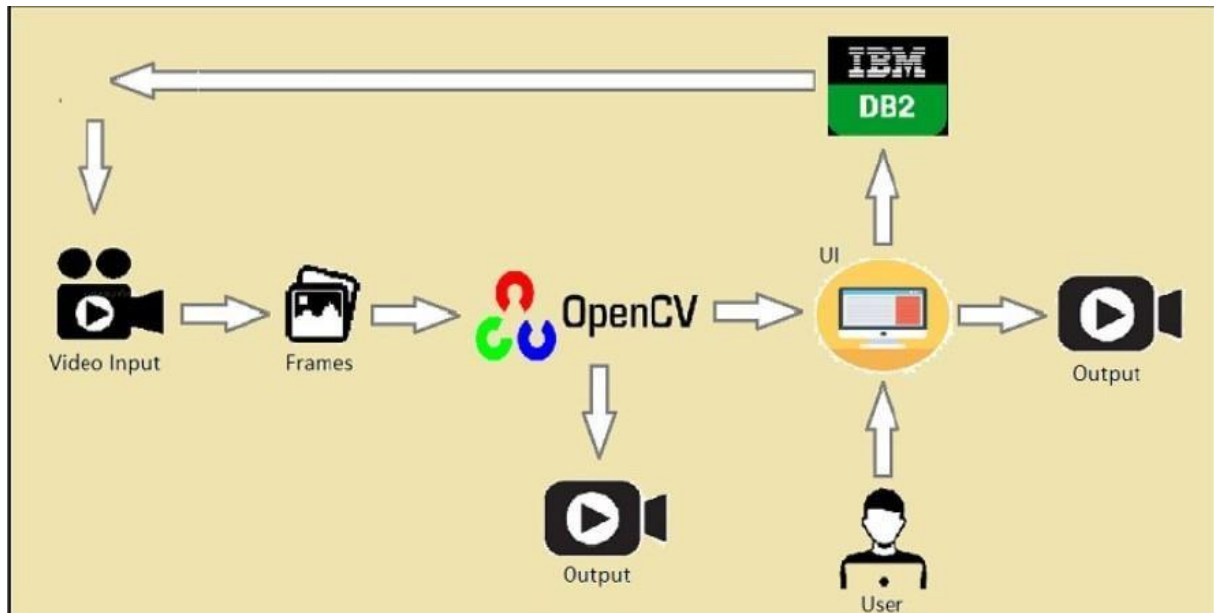
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	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 Face book	I can register & access the dashboard with Face book 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	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 Chatbot for 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 an administrator ,I can view the database of the registered users	I can check and verify the persons who are the registered their mail ids and information's	Medium	Sprint - 4

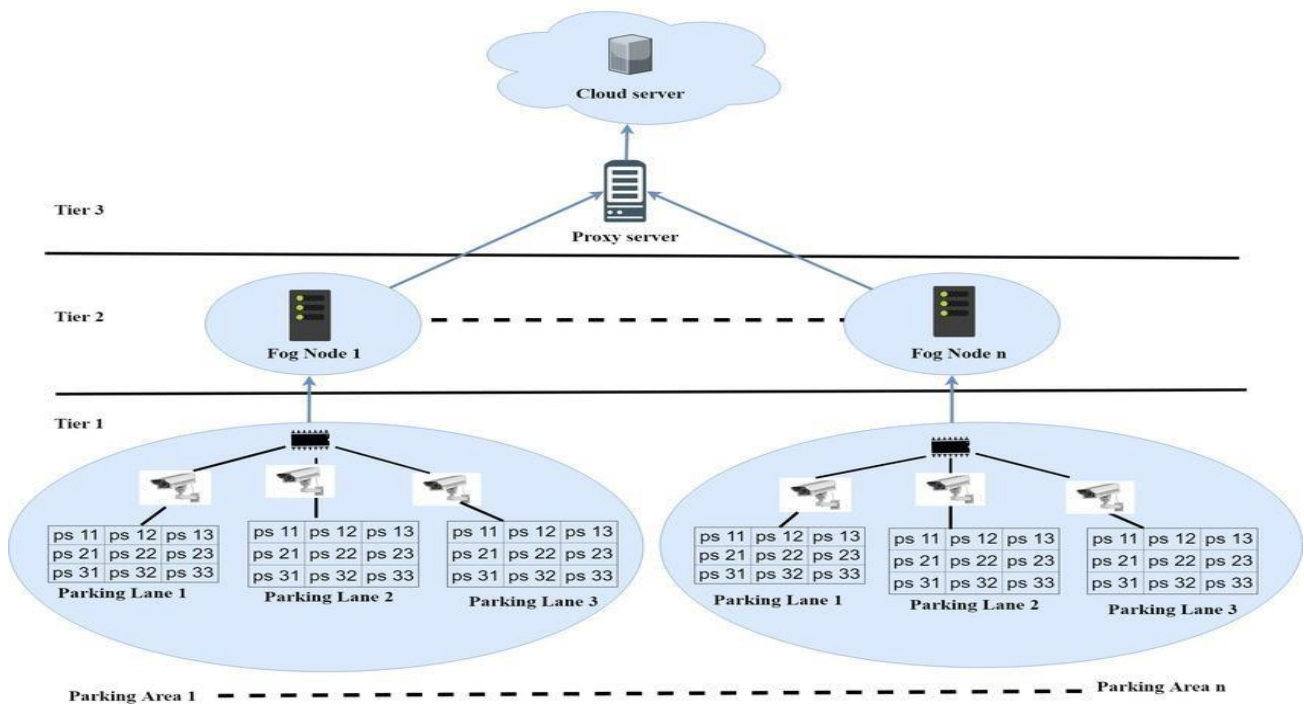
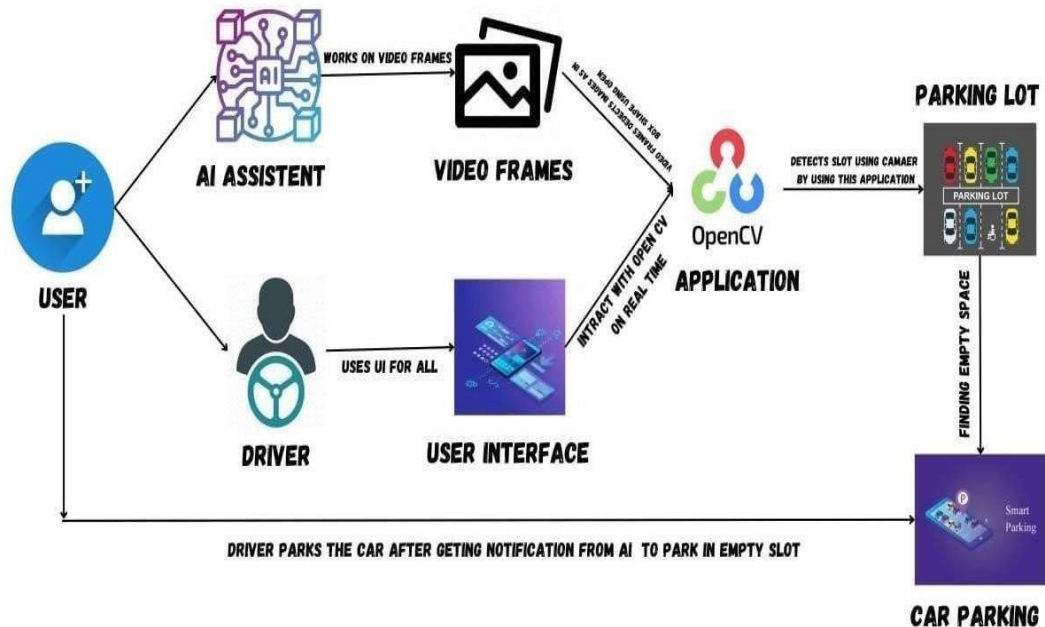
	Dash board	USN10	As an administrator ,I can view how many members requested for what trouble occurs in parking a	I can check the number of requirements and monitor the availability.	Low	Sprint - 4
			vehicle			
chatbot	User interface	USN11	In addition to the customer care executive I can solve all the queries of the customer as well as the conferrer	I can reply to all the questions which are asked by the users that are related to the service we provided	Medium	Sprint - 4

5.2 Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



S. No	Component	Description	Technology
1.	User Interface	User Interface is used by user in mobile application or In Build in car display itself	HTML, CSS, JavaScript / Angular JS / React JS etc.
2.	User Logic-1	Framework used for design the software	Python, python-flask
3.	User Logic-2	Access the software in the car by the driver to detect spot	Python, Open CV
4.	Application Logic-1	Open CV is an open-source platform for providing real time computer vision technology	Open CV
5.	Data Base	Contains images and video frames stores in data base	MySQL, NoSQL, etc.
6.	Cloud Data Base	Data Base Service on cloud	IBM DB2, IBM Cloud etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or local File system

8.	External API-1	They make it easy for developers to store manage and deploy container images	Container registry
10.	Machine Learning Model	Uses test and trained data images and video to learn the environment	Object recognition models, etc.
11.	Infrastructure (server / cloud)	Application Development on Local system / cloud	Local, cloud Foundry, python-flask, etc.

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	Ssathyan S R
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Parthib Dey

Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	Ashish Pandey
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Shriyansh
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	Ssathyan S R
	Dashboard	USN-6	As a admin i can view all the details of vehicles parked	2	Medium	Parthib Dey
Sprint -2	Help desk/customer support	USN-7	As a customer care executive, I can solve queries of customers	2	Medium	Ashish Pandey
Sprint-3	Chatbot	USN-8	Chatbot can solve queries of customers and conferer	2	Low	Shriyansh

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	10	1 week	21 Oct 2023	28 Oct 2023	10	29 Oct 2023
Sprint-2	15	1 week	1 Nov 2023	08 Nov 2023	14	11 Nov 2023
Sprint-3	20	1 week	12 Nov 2023	19 Nov 2023	18	20 Nov 2023

Velocity:

Average velocity = $(10+14+18)/3 = 14$

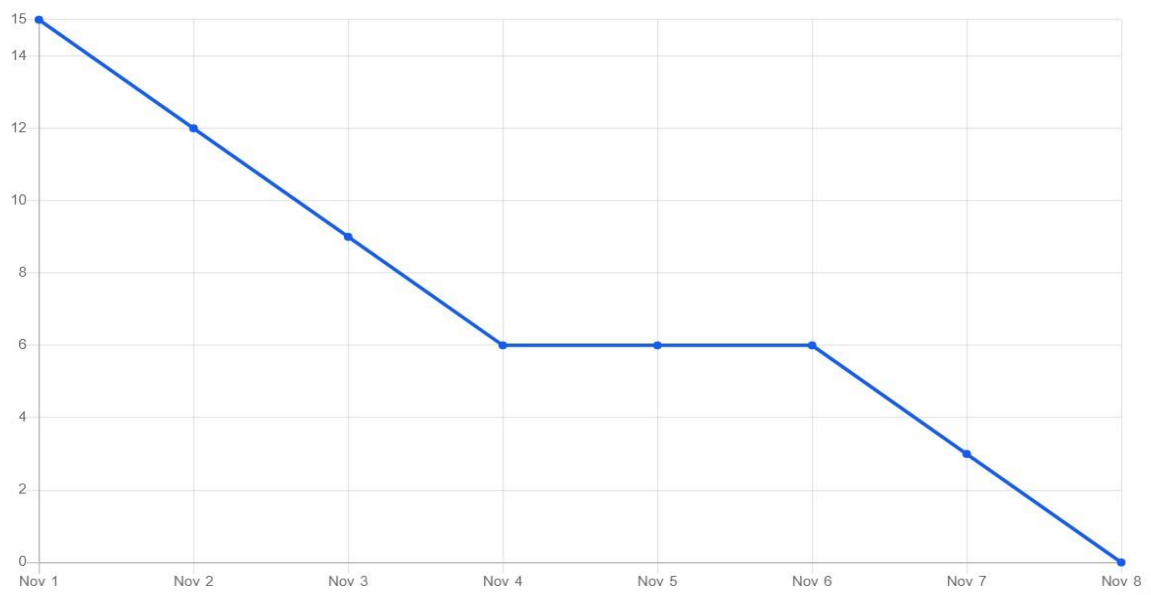
Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time. Sprint 1



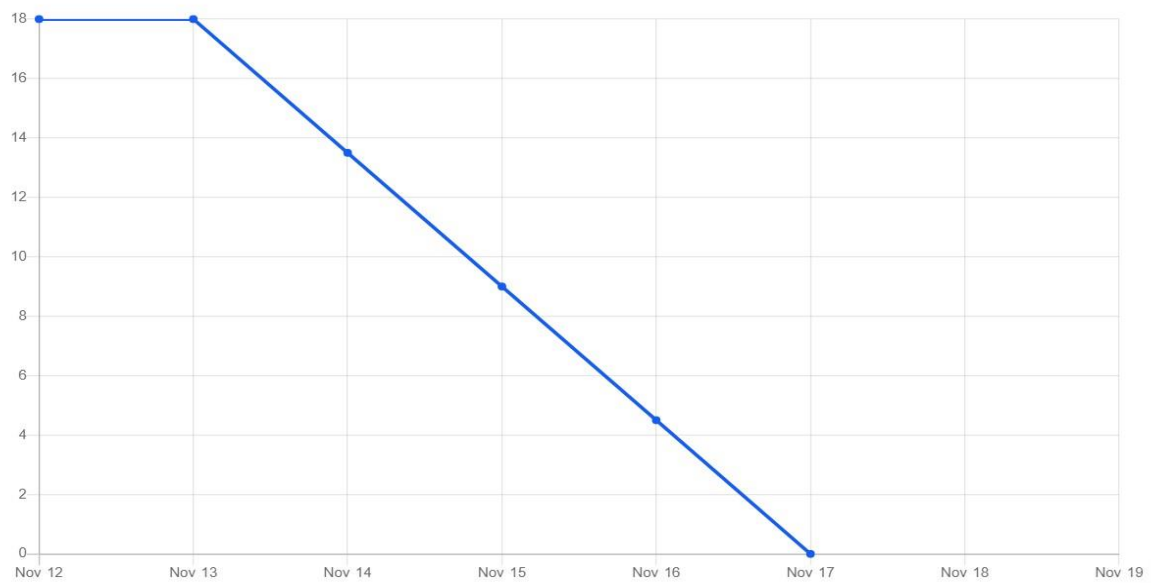
Sprint 2

Burndown Chart



Sprint 3

Burndown Chart



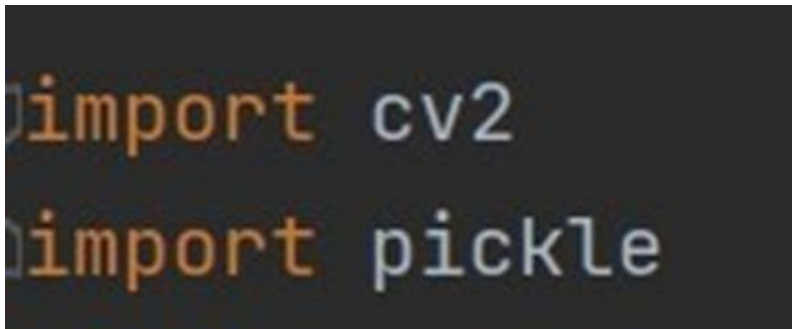
7.Coding and solutioning:

Activity 1: Create a python file

Create a python file in the directory and name it as 'selectingROI.py'. This python file is used for creating ROI and deleting ROI.

Activity 2: Importing the required packages

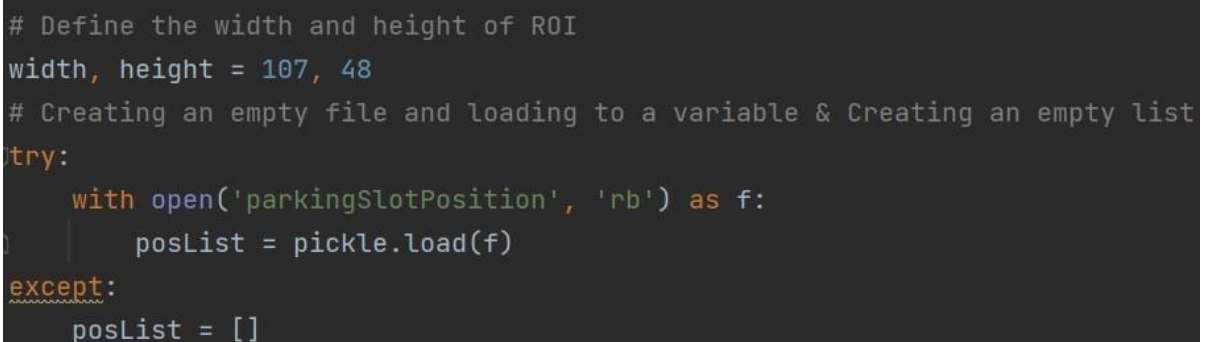
- OpenCV: OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It supports multiple languages including python, java C++.
- Pickle: The 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 cv2
import pickle
```

Activity 3: Define ROI width and height

- Calculating the ROI width and height (manually width and height are calculated and given as 107 & 48).
- An empty file parkingSlotPosition is created to save all the ROI values. Try and except combo is used.
- 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.



```
# Define the width and height of ROI
width, height = 107, 48
# Creating an empty file and loading to a variable & Creating an empty list
try:
    with open('parkingSlotPosition', 'rb') as f:
        posList = pickle.load(f)
except:
    posList = []
```


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 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 unwanted region. Then we can remove that unwanted ROI by right click from the mouse.
- Python object are converted into a stream of bytes and stored in `parkingSlotPosition`

```
def mouseClick(events, x, y, flags, params):  
    # Adding ROI values to posList  
    if events == cv2.EVENT_LBUTTONDOWN:  
        posList.append((x, y))  
    # Removing unwanted ROI from posList  
    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)  
    # Saving the posList values to parkingSlotPosition file  
    with open('parkingSlotPosition', '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. As 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 the event which we have defined in activity 4.

```

while True:
    img = cv2.imread('carParkImg.png')
    for pos in posList:
        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), (255, 255, 255), 2)

    cv2.imshow("Image", img)
    cv2.setMouseCallback("Image", mouseClicked)
    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 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 parkingSlotPosition file by load() method from pickle library. The parkingSlotPosition file is created from milestone 2. The ROI values are presented in the parkingSlotPosition file.
- Define width and height which we have used in milestone 2.

```

# Video feed
cap = cv2.VideoCapture('carParkingInput.mp4')
# Loading the ROI from parkingSlotPosition file
with open('parkingSlotPosition', 'rb') as f:
    posList = pickle.load(f)
# Define width and height
width, height = 107, 48

```

Activity 3: Checking for parking space

- The function is defined with the name as carParkingSpace. As a parameter image has to be passed.
- Taking the position from position list and saved to variable 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 green color if the pixel count is lesser than 900 else it is drawn in red color.
- The count of green color is displayed in the frame by putTextRect() method from cvzone library.

```
def checkParkingSpace(imgPro):
    spaceCounter = 0
    for pos in posList:
        x, y = pos
        # Crop the image based on ROI
        imgCrop = imgPro[y:y + height, x:x + width]
        # Counting the pixel values from cropped image
        count = cv2.countNonZero(imgCrop)
        if count < 900:
            color = (0, 255, 0)
            thickness = 5
            spaceCounter += 1
        else:
            color = (0, 0, 255)
            thickness = 2
        # Draw the rectangle based on the condition defined above
        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
    # Display the available parking slot count / total parking slot count
    cvzone.putTextRect(img, f'Free: {spaceCounter}/{len(posList)}', (100, 50), scale=3,
                       thickness=5, offset=20, colorR=(0, 200, 0))
```

Activity 4: Looping the video

- Current frame position is compared with total number of frames. If it reaches the maximum frame, again the frame is set to zero. So, continuously it'll loop the frame.
- cv2.CAP_PROP_POS_FRAMES = Current frame
- cv2.CAP_PROP_FRAME_COUNT = Total no. of frame

```
while True:
    # Looping the video
    if cap.get(cv2.CAP_PROP_POS_FRAMES) == cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
```

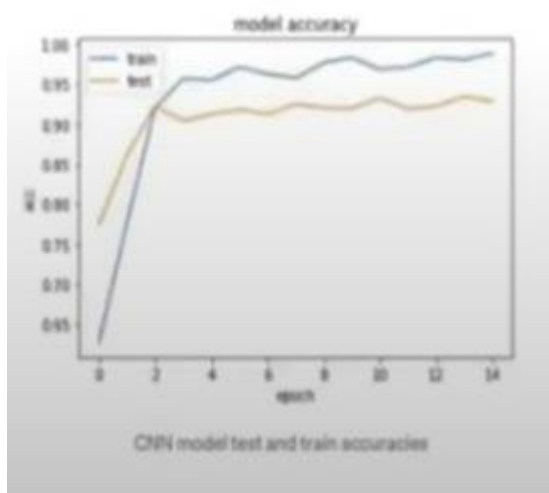
Activity 5: Frame Processing and empty parking slot counters

- 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 gray scale image. And the gray scale image is blurred with GaussianBlur() method from cv2.

- The adaptiveThreshold() method is used to apply threshold to the blurred image. And again blur is applied to the image.
- The dilate() method is used to computing 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 dilated image. As per the activity 3 we will get the count of empty parking slot.
- Display the frames in form of video. The frame will wait for 10 seconds and it'll go to next frame.

```
while True:
    # Looping the video
    if cap.get(cv2.CAP_PROP_POS_FRAMES) == cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    # Reading frame by frame from video
    success, img = cap.read()
    # Converting to gray scale image
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1) # Applying blur to image
    # Applying threshold to the image
    imgThreshold = cv2.adaptiveThreshold(imgBlur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                         cv2.THRESH_BINARY_INV, 25, 16)
    imgMedian = cv2.medianBlur(imgThreshold, 5) # Applying blur to image
    kernel = np.ones((3, 3), np.uint8)
    imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)
    # Passing dilate image to the function
    checkParkingSpace(imgDilate)
    cv2.imshow("Image", img)
    cv2.waitKey(10)
```

8. Performance testing:



Training Accuracy – 98%

Validation Accuracy -94%

9.Results:



10. ADVANTAGES & DISADVANTAGES:

Advantages:

1. Efficient Parking Management:

- AI-powered systems can efficiently manage parking spaces by accurately detecting and tracking vehicles in real-time using computer vision.

2. Automated Parking Assistance:

- OpenCV can be used to implement automated parking assistance, helping drivers find available parking spots more easily.

3. Reduced Traffic Congestion:

- Improved parking efficiency can contribute to reduced traffic congestion as drivers spend less time searching for parking spaces.

4. Enhanced Safety:

- Computer vision can be used to monitor parking areas for safety, detecting and alerting authorities to any unusual activities or incidents.

5. Optimized Space Utilization:

- AI algorithms can help optimize parking space utilization by precisely positioning vehicles, allowing for more efficient use of available space.

6. Integration with Other Systems:

- OpenCV can be integrated with other smart city systems for better overall traffic and urban management.

Disadvantages:

1. Cost of Implementation:

- Implementing AI-enabled parking systems can be expensive, involving the cost of hardware, software, and maintenance.

2. Technical Challenges:

- Computer vision systems can face challenges such as weather conditions, poor lighting, or occlusions that may affect their accuracy and performance.

3. Privacy Concerns:

- AI systems that involve video surveillance raise privacy concerns, as they may capture and analyze data related to individuals without their consent.

4. Maintenance and Downtime:

- Like any technology, AI-enabled parking systems may require regular maintenance, and downtime for maintenance or system updates could impact their availability.

5. Dependence on Technology:

- Reliance on AI systems means that if there are technical issues or malfunctions, it could disrupt the parking management process.

6. Limited Adaptation in Older Infrastructure:

- Older parking structures may not be easily retrofitted with the necessary technology, limiting the applicability of AI-enabled systems.

7. Energy Consumption:

- Running complex computer vision algorithms continuously may require significant computational power, leading to increased energy consumption.

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 liability , transportation mobility and environment sustainability. The Internet of Things integrates the hardware, software and network connectivity that enable objects to be sensed and remotely controlled across existing network. Such integration allows users to monitor available and unavailable parking spots that lead to improved efficiency, accuracy and economic benefit.

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. With the ever-growing field of Internet of Things many concepts can be interfaced along with our system. For residential and domestic parking system the device can be interfaced with Home Automation system which can control the various home appliances by sensing whether the user is arriving or departing from the parking space. For instance, if the user has arrived then the module will sense the presence and will send information about arrival to the Home automation system which can accordingly switch on the selected appliances like HVAC (Heating Ventilation and Air Conditioning) units, Coffee maker, toaster, Wi-Fi routers etc. For commercial parking system the device can be interfaced with a module which can sense the arrival of employee and can switch on his computer and HVAC systems and accordingly switch off the appliances when the employee departs. The system can also be used to track the reporting and departing time of the employee for all days with precision thus acting as an attendance system. Thus, many such modules can be interfaced with our system to provide better facility, security, and optimization of electricity and resources with the principle idea of flawless fleet management system.

13.Appendix:

Github repo: <https://github.com/smartinternz02/SI-GuidedProject-613434-1700634588>

DEMO LINK: IN GIT REPO