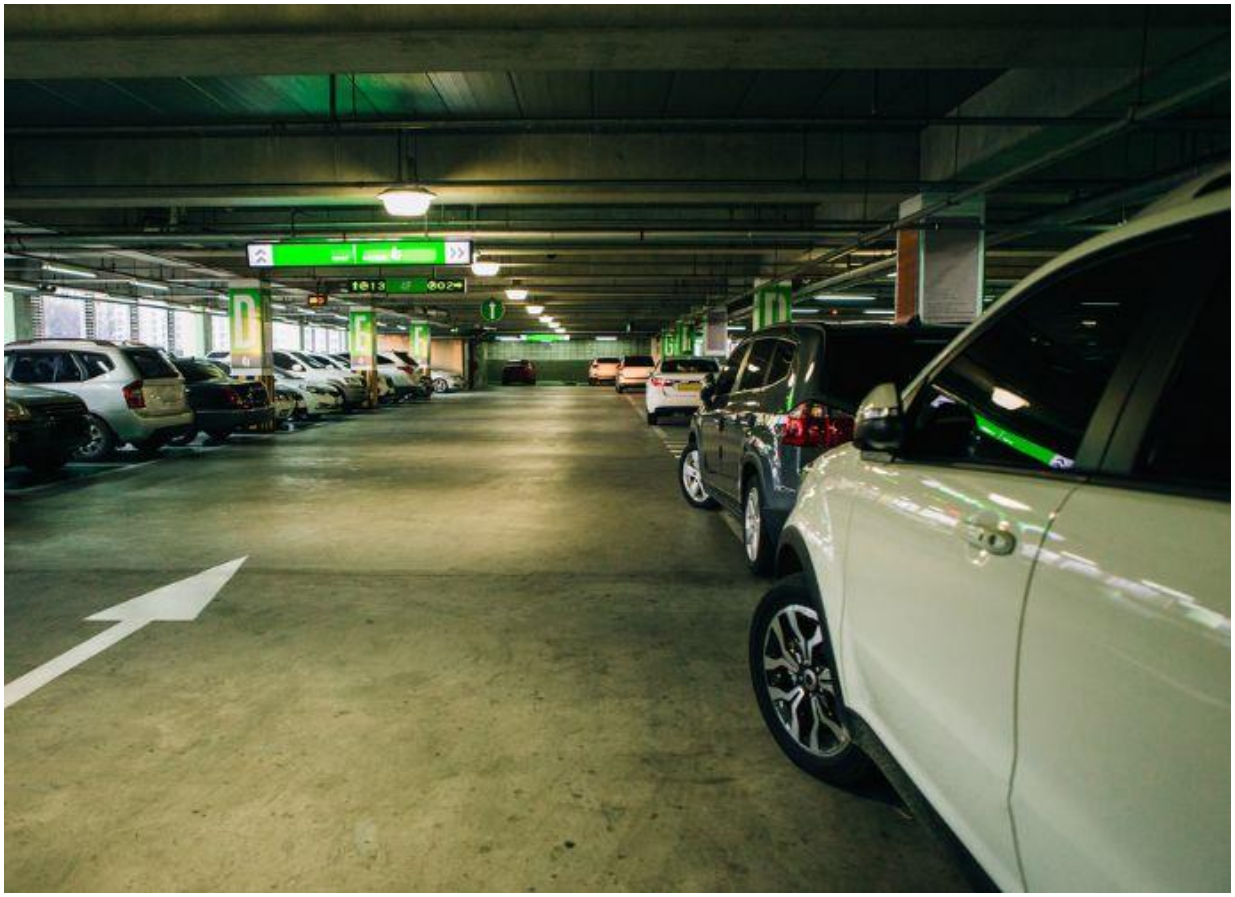
AI ENABLED CAR PARKING USING OPEN CV



DATE	22/11/2023
TEAM ID	592404
TEAM MEMBER – 1	BIPIN SHAH
TEAM MEMBER – 2	KARNAVAT KUSH KAILESHBHAI
TEAM MEMBER – 3	VADODARIA NILANSH CHINTAN
TEAM MEMBER – 4	PRATHAM GUPTA

TOPICS COVERED

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **LITERATURE SURVEY**
   - 2.1 Existing problem
   - 2.2 References
   - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - 3.1 Empathy Map Canvas
   - 3.2 Ideation & Brainstorming
4. **REQUIREMENT ANALYSIS**
   - 4.1 Functional requirement
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams & User Stories
   - 5.2 Solution Architecture
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Technical Architecture
   - 6.2 Sprint Planning & Estimation
   - 6.3 Sprint Delivery Schedule
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   - 7.1 Feature 1
   - 7.2 Feature 2
   - 7.3 Database Schema (if Applicable)
8. **PERFORMANCE TESTING**
   - 8.1 Performace Metrics
9. **RESULTS**
   - 9.1 Output Screenshots
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
    - Source Code
    - GitHub & Project Demo Link

INTRODUCTION

➢ PROJECT OVERVIEW

Despite having AI capabilities, a lot of the auto parking systems in use today are still operated manually. It is challenging to keep track of each parking space's capacity without an automated monitoring system. Finding a spot in the parking lot typically requires drivers to negotiate a convoluted approach. There is always a greater demand for parking spaces than there are available; these issues are particularly prevalent near schools, shopping centers, hospitals, and other densely populated areas.

➢ PURPOSE

The goal is to improve and expedite the process of parking cars in garages or parking lots by integrating AI-powered car parking systems with OpenCV (Open-Source Computer Vision). These systems are able to detect and track vehicles, determine whether parking spaces are available, and efficiently guide drivers to empty places by utilizing computer vision techniques and AI algorithms.

LITERATURE SURVEY

➢ EXISTING PROBLEM

1. **Conventional Parking Systems' Inefficiency:**

   Talk about the shortcomings of human monitoring and the absence of real-time data in typical parking systems.

2. **Detection of Parking Space Difficulties:**

   Draw attention to the difficulties in correctly identifying open parking spaces and differentiating between occupied and unoccupied areas.

3. **Maximizing the Use of Parking Spaces:**

   Examine how wasteful use of resources and traffic may result from existing systems' inability to maximize parking space utilization.

➢ REFERENCES

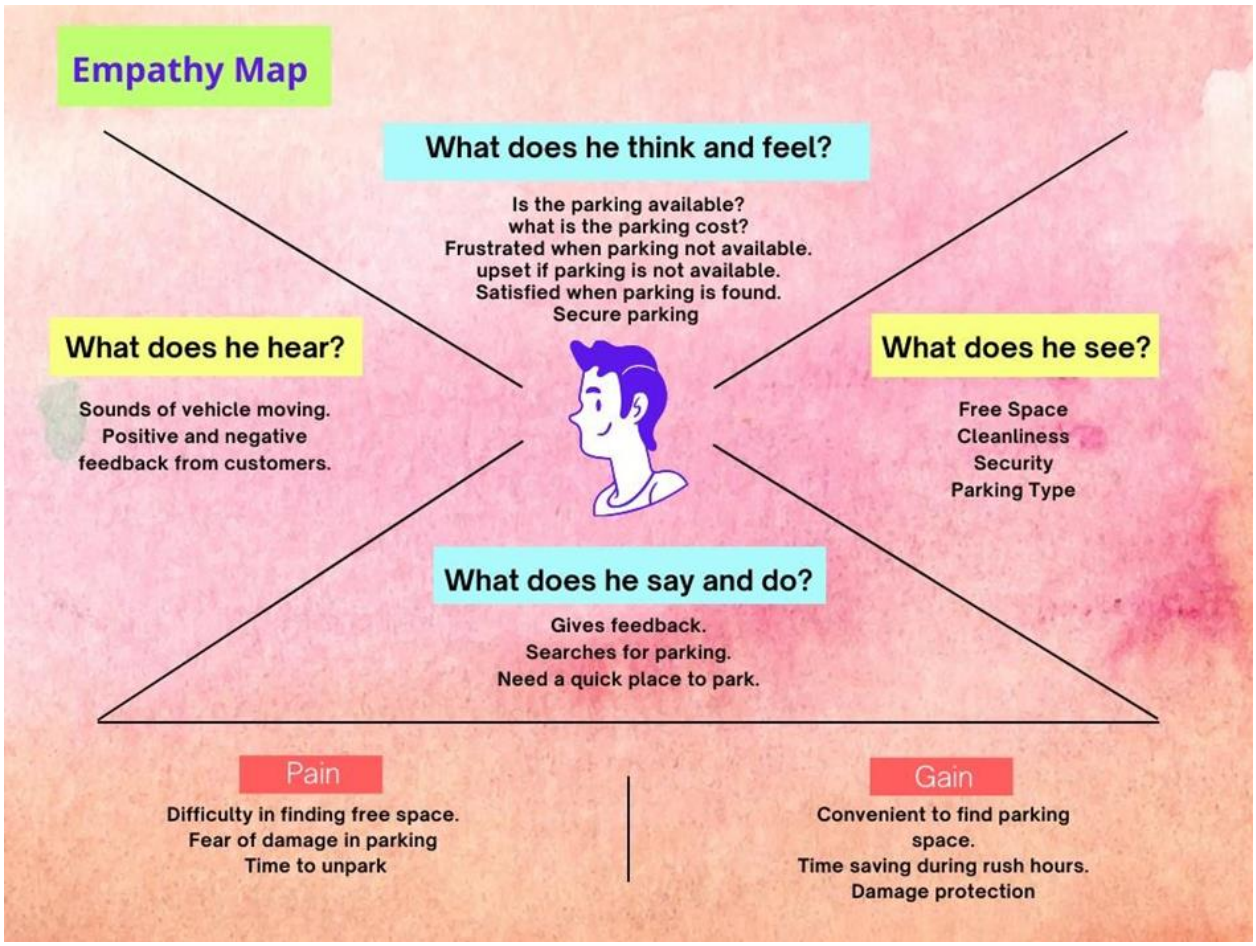
- https://www.analyticsvidhya.com/blog/2021/05/computer-vision-using-opencv-with-practical-examples/
- https://www.aitech.vision/products/ai-smart/ai-smart-parking/#:~:text=AI%2DParking%20allows%20you%20to,of%20events%20to%20the%20A.I
- https://www.workero.com/parking-management-system-guide/
- https://www.claysys.com/blog/artificial-intelligence-in-transportation/
- https://www.sciencedirect.com/science/article/pii/S2666691X21000397
- https://brainstation.io/career-guides/does-machine-learning-require-coding
- https://www.scribd.com/document/650944475/AI-enable-car-parking-1
- https://www.studocu.com/in/document/anna-university/artificial-intelligence/ai-enabled-car-parking-using-open-cv/60396601
- https://towardsdatascience.com/find-where-to-park-in-real-time-using-opencv-and-tensorflow-4307a4c3da03
- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3163473

➢ PROBLEM STATEMENT DEFINATION

It is difficult for the present parking systems to track, identify, and use parking spaces efficiently. This project's objective is to use OpenCV to create an AI-based vehicle parking system that incorporates real-time object identification and tracking to maximize parking space utilization and offer a more seamless parking experience.

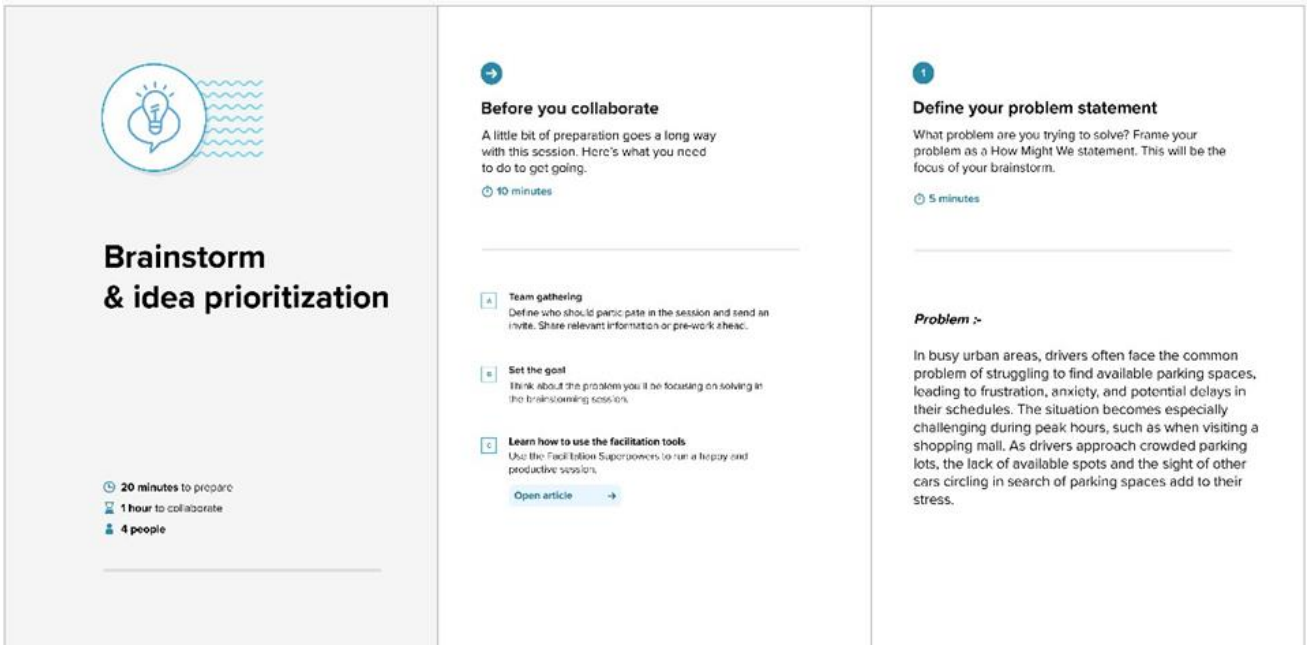
IDEATION AND PROPROSED SOLUTION

➢ EMPATHY MAP CANVAS

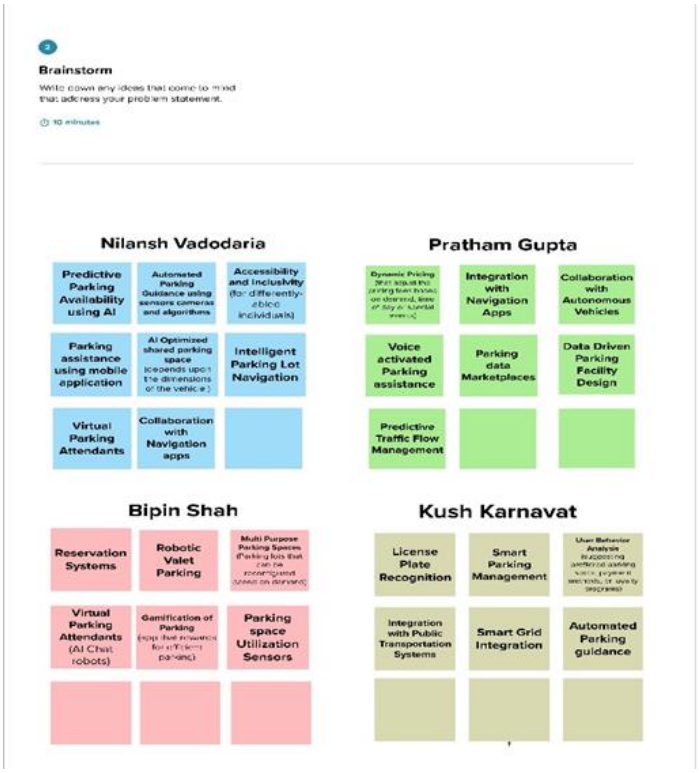


➢ IDEATION AND BRAINSTROMING

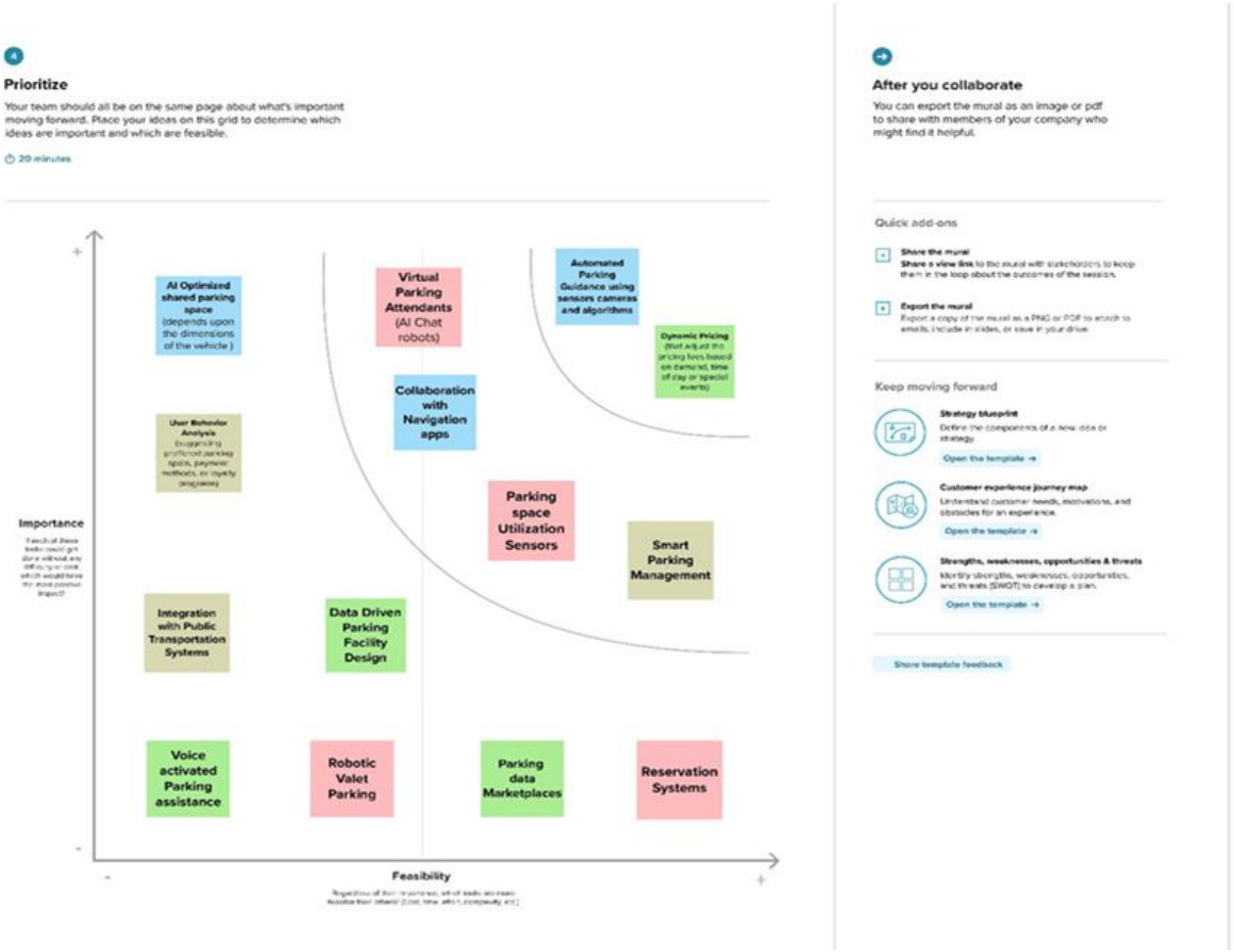
- **STEP 1:**
  Team Gathering, Collaboration and Select the Problem Statement



- **STEP 2:**
  Brainstorm, Idea Listing and Grouping



- **STEP 3:**
  Idea Prioritization



REQUIREMENT ANALYSIS

- **FUNCTIONAL REQUIREMENT**

FR NO.	FUNCTIONAL REQUIREMNT(EPIC)	SUB REQUIREMENT (STORY / SUB TASK)
FR - 1	User authentication	Capture and save facial images of users. Verify the identity of users through facial recognition. Grant entry to the parking facility upon successful authentication.
FR – 2	Parking spot management	Identify the presence of available parking spaces through camera footage. Allocate a parking spot to the user. Maintain real-time updates on parking spot availability.

FR – 3	Payment and billing	Compute parking charges based on the duration of parking. Facilitate payment through diverse methods such as credit cards and mobile wallets.
FR – 4	Security and surveillance	Surveil the parking area through cameras to identify any suspicious activities. Notify security personnel promptly in the event of a security breach.
FR – 5	System maintenance and support	Perform routine maintenance and updates to the system. Extend customer support services to address any issues encountered by users.

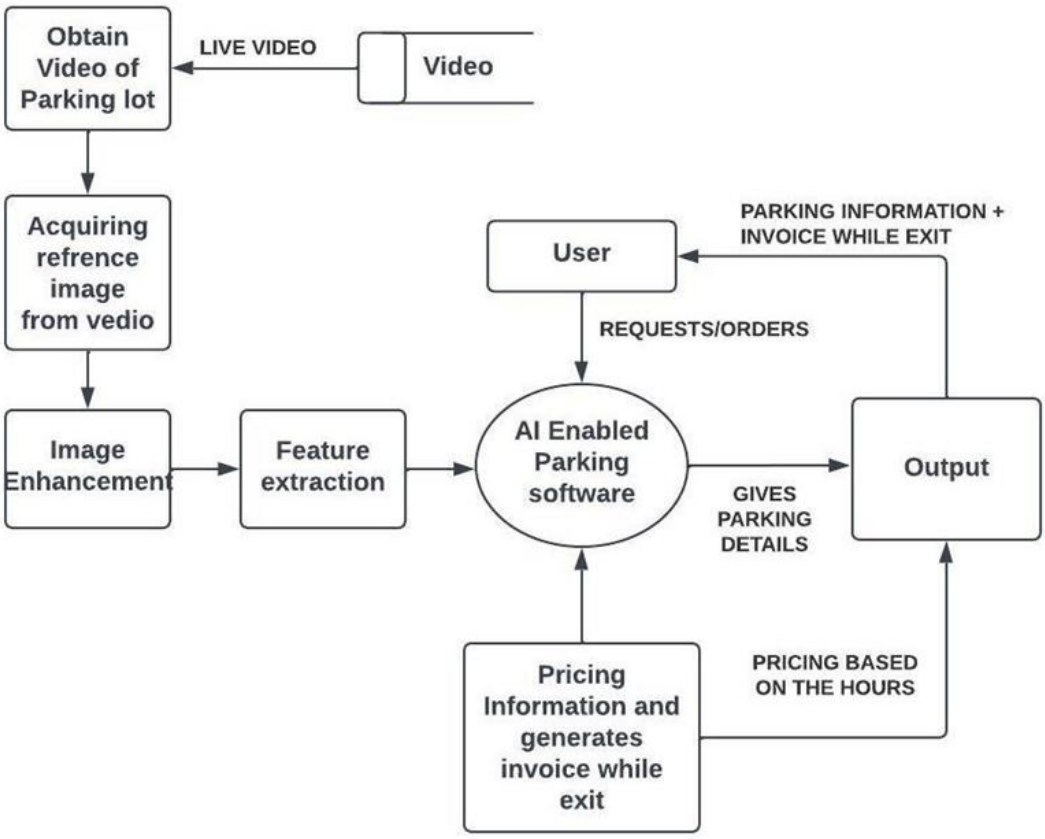
➢ NON – FUNCTIONAL REQUIREMENT

FR NO.	NON – FUNCTIONAL REQUIREMENT	DESCRIPTION
FR-1	Usability	The system should offer ease of use and a user-friendly experience, delivering explicit instructions to drivers on parking locations and presenting available spots in an intuitive fashion.
FR-2	Security	The system should ensure security and possess the capability to thwart unauthorized entry to the parking lot.
FR-3	Reliability	The system must demonstrate reliability and precision in detecting and identifying available parking spots. .

FR-4	Performance	The system must operate with swiftness to promptly detect and identify available parking spots in real-time, enabling drivers to rapidly locate a parking space. Moreover, it should have the capacity to manage multiple vehicles entering and exiting the parking lot concurrently.
FR-5	Availability	The system should be adaptable to various vehicle types, encompassing cars, trucks, and motorcycles, and possess the capability to accommodate different vehicle sizes.
FR-6	Scalability	The system should have the capacity to manage an extensive number of parking spaces and should be readily scalable to incorporate additional spaces in the future.

PROJECT DESIGN

➢ DATA FLOW DIAGRAM



➢ USER STORIES – TYPE 1

Customer (Driver) Functional Requirement (Epic)

STORY NUMBER	USER STORY / TASK	ACCEPTANCE CRITERIA	PRIORITY	RELEASE
ACP-1	As a driver, I want the AI-enabled car parking system to assist me in finding an available parking spot when I enter the parking area.	-	High	Sprint-1

ACP-2	As a driver, I want the system to guide me accurately into the parking space using OpenCV for visual cues on my in-car display.	-	High	Sprint-1
ACP-3	As a driver, I want the AI system to provide real-time information on parking spot availability through a mobile app or in-car display.	I can see real-time parking spot availability.	High	Sprint-1
ACP-4	As a driver, I want to reserve a parking space in advance through the AI system and have it guaranteed when I arrive.	-	Medium	Sprint-2
ACP-5	As a driver, I want a seamless and contactless payment and exit process using OpenCV to recognize my license plate.	-	High	Sprint-1
ACP-6	As a driver with accessibility needs, I want the AI system to offer features that assist me in parking, such as guidance for accessible parking spaces.	-	Low	Sprint-2
ACP-7	As a driver, I want the system to provide guidance on leaving the parking space safely.	-	Medium	Sprint-2

➢ USER STORIES – TYPE 2

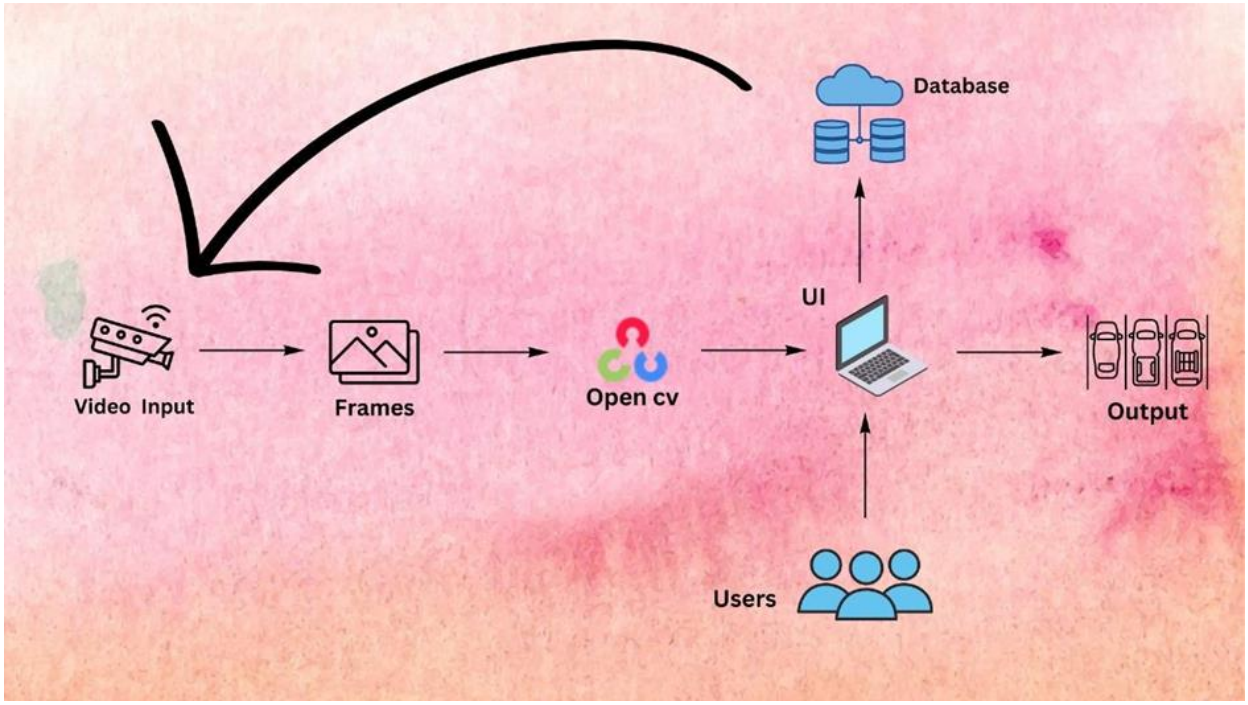
Parking Lot Owner/Manager Functional Requirement (Epic)

STORY NUMBER	USER STORY / TASK	ACCEPTANCE CRITERIA	PRIORITY	RELEASE
ACP-8	As a parking lot owner, I want the AI system to provide real-time data and analytics on parking spot occupancy, revenue, and trends.	I can view reports and analytics on the parking facility's performance.	High	Sprint-1
ACP-9	As a parking facility owner, I want the AI system to notify me of any maintenance or security issues detected by OpenCV in the parking area.	I receive real-time alerts for maintenance and security issues.	High	Sprint-1
ACP-10	As a parking facility manager, I want to customize parking rates and rules through the AI system.	I can modify parking rates and rules through the management interface.	Medium	Sprint-2

➢ SOLUTION ARCHITECTURE

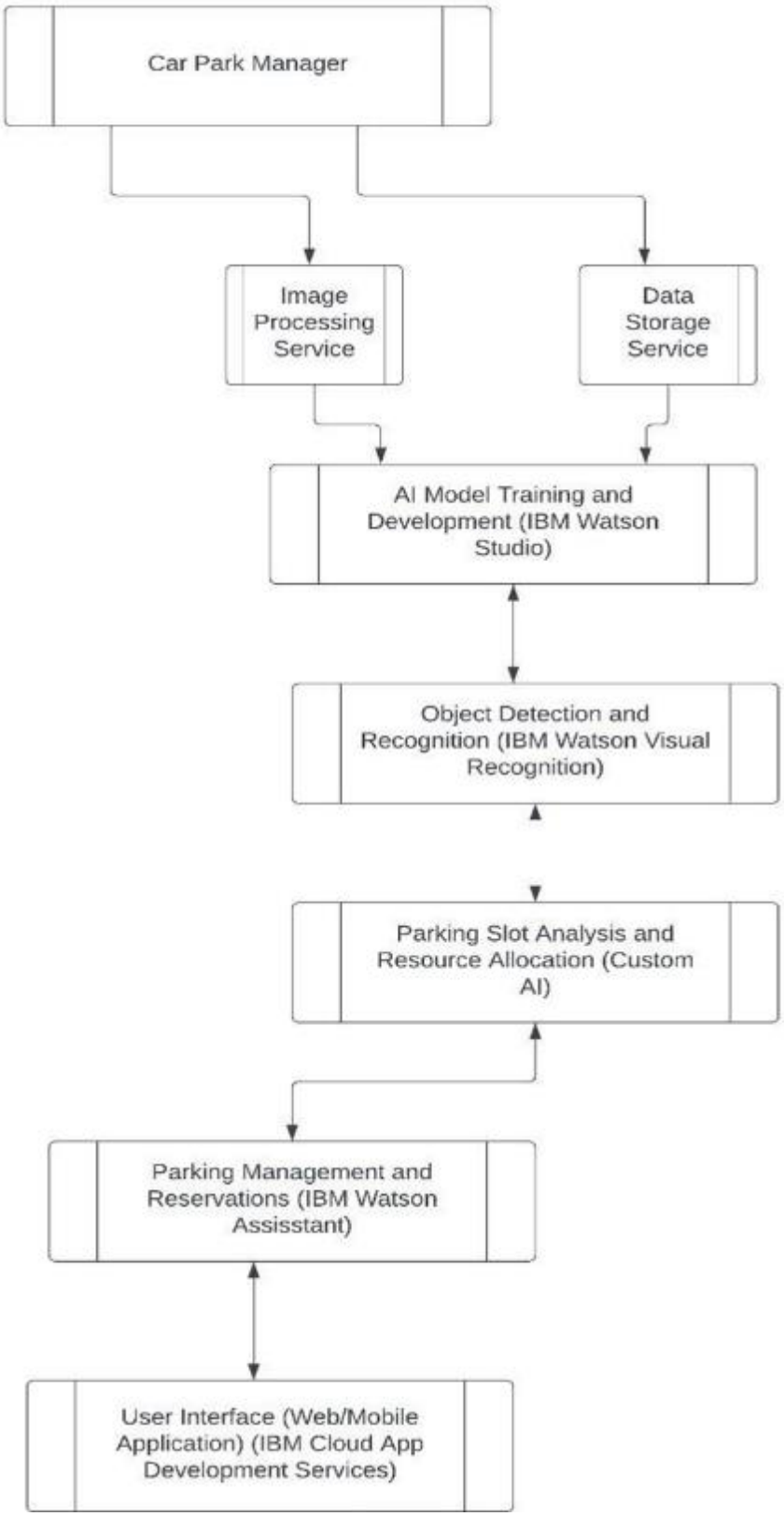
Solution architecture is a complex process – with many sub-processes – that bridgesthe gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of thesoftware to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managedand delivered.



PROJECT PLANNING AND SCHEDULING

➢ TECHNICAL ARCHITECTURE



Components & Technologies for AI-Enabled Car Parking System

SL. NO.	COMPONENT	DESCRIPTION	TECHNOLOGY
1.	User Interface	How users interact with the system	Web UI, Mobile App, Chatbot, etc.
2.	Object Detection and Tracking	Logic for car detection and tracking using OpenCV	Python, OpenCV
3.	License Plate Recognition	Logic for license plate recognition	Python, OpenCV, OCR libraries
4.	Database	Data Type, Configurations, etc.	MySQL, NoSQL, etc.
5.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant, etc.
6.	File Storage	File storage requirements	IBM Block Storage, Other Storage Service, or Local Filesystem
7.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
8.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
9.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud	Local Server Configuration, Cloud Server Configuration, Local, Cloud Foundry, Kubernetes, etc.

Application Characteristics for AI-Enabled Car Parking System

SL. NO.	CHARACTERSTICS	DESCRIPTION	TECHNOLOGY
1.	Open-Source Frameworks	List the open-source frameworks used	OpenCV, TensorFlow, PyTorch, etc.
2.	Security Implementations	List all the security / access controls implemented	Encryption, Access Controls, etc.
3.	Scalable Architecture	Justify the scalability of architecture (e.g., use of microservices, distributed processing)	Cloud Services, Microservices, etc.
4.	Availability	Justify the availability of the application (e.g., use of load balancers, redundant servers)	Load Balancers, Redundancy, etc.
5.	Performance	Design considerations for system performance (e.g., requests per second, caching, use of CDNs)	Caching, CDN, Load Optimization, etc.

➢ SPRINT PLANNING AND ESTIMATION

SPRINT	FUNCTIONAL REQUIREMENT	USER STORY NUMBER	USER STORY / TASK	STORY POINTS	PRIORITY
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	2	Medium
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High
	Dashboard	USN-6	As a user, I want the system to notify me of available parking spaces and guide me to the nearest one.	3	High
	Dashboard	USN-7	As a user, I want the system to notify me of available parking spaces and guide me to the nearest one.	2	Medium
	Dashboard	USN-8	As a user, I want the system to notify me of available parking spaces and guide me to the nearest one.	3	High

➢ SPRINT DELIVERY SCHEDULE

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	20	5 Days	24 Oct 2023	28 Oct 2023	20	28 Oct 2023
Sprint-2	20	5 Days	29 Oct 2023	02 Nov 2023	15	03 Nov 2023
Sprint-3	20	5 Days	03 Nov 2023	07 Nov 2023	18	07 Nov 2023
Sprint-4	20	6 Days	08 Nov 2023	13 Nov 2023	20	14 Nov 2023
Sprint-5	25	6 Days	14 Nov 2023	19 Nov 2023	22	20 Nov 2023

CODING AND SOLUTIONING

➢ FEATURE 1 - ROI (Region of interest)

A region of interest (ROI) is a portion of an image that you want to filter or operate on in some way.

➢ FEATURE 2 – VOD (Video Object Detection)

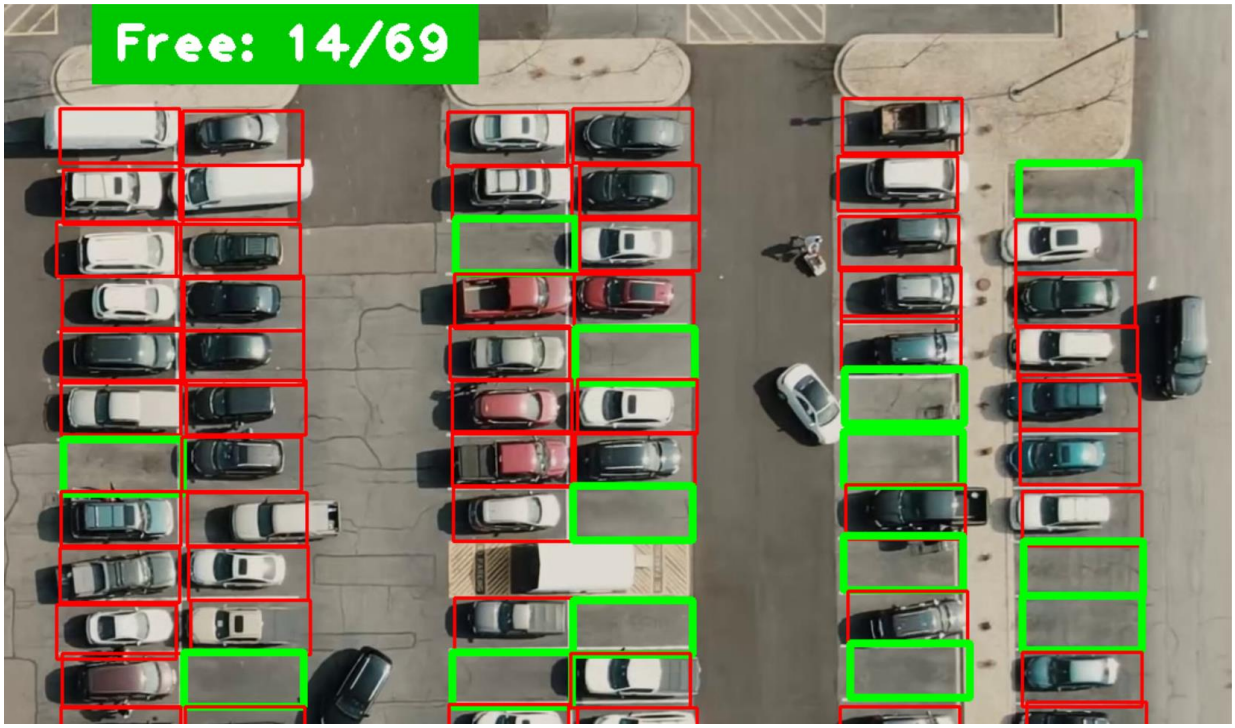
Video Object Detection (VOD) mimics the human visual cortex. It allows machines to analyze video frame by frame and identify the objects present within them. Thus, object detection in video works similarly to AI image recognition.

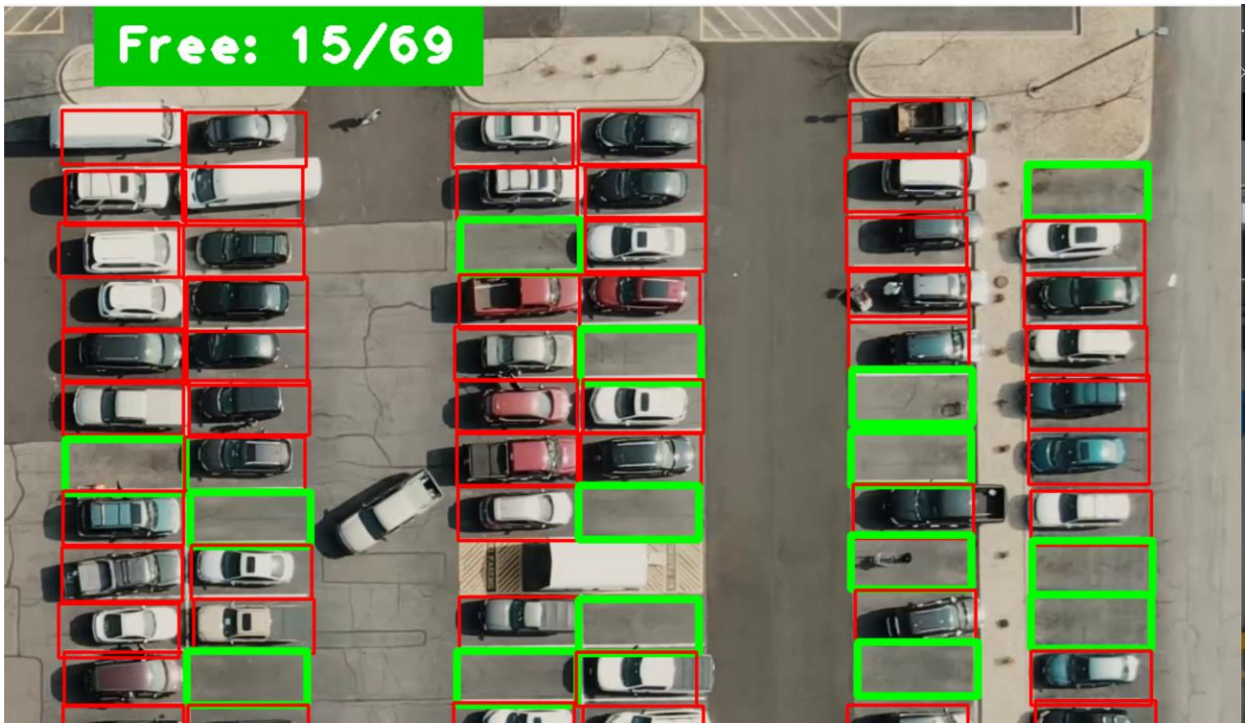
- **DATABASE SCHEMA**

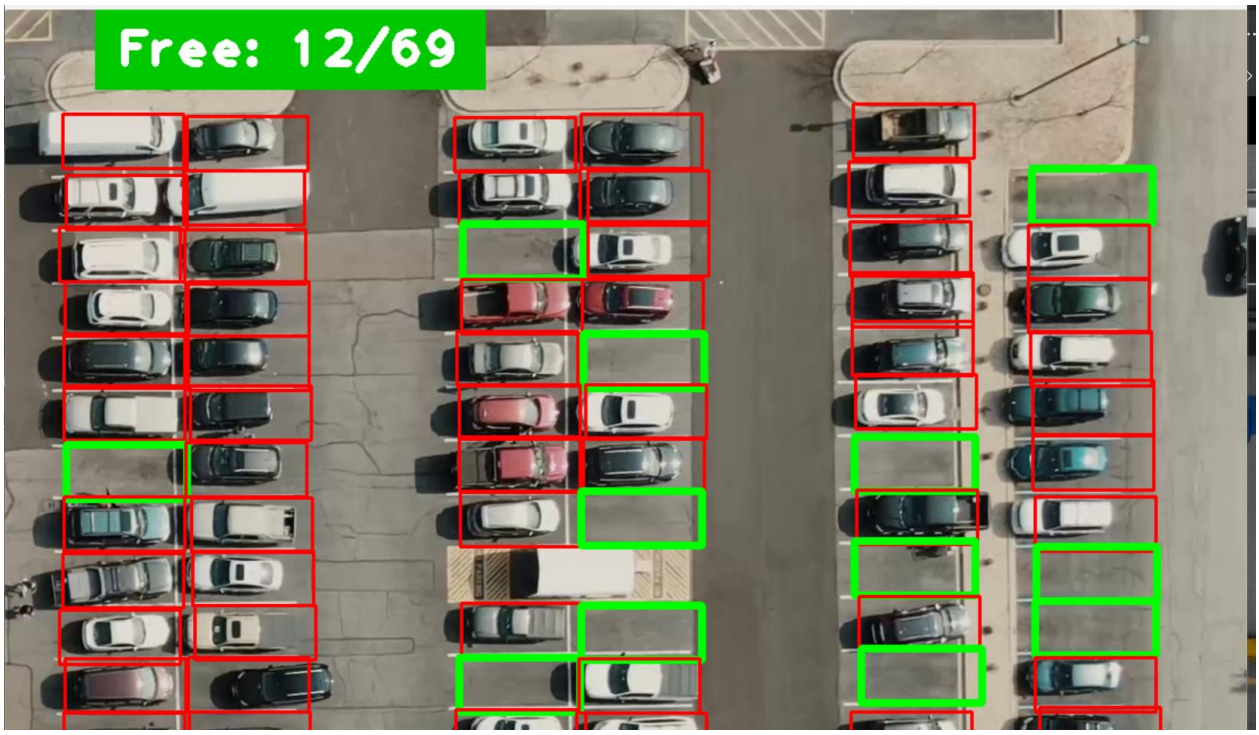
```
postgres=# SELECT * FROM public."REGISTER";
 name |                email                | password
------+-------------------------------------+----------
 b    | bipin@gmail.com                     | 1
 B    | Bipin.Shah2021@Vitstudent.Ac.In     | 12345
(2 rows)
```

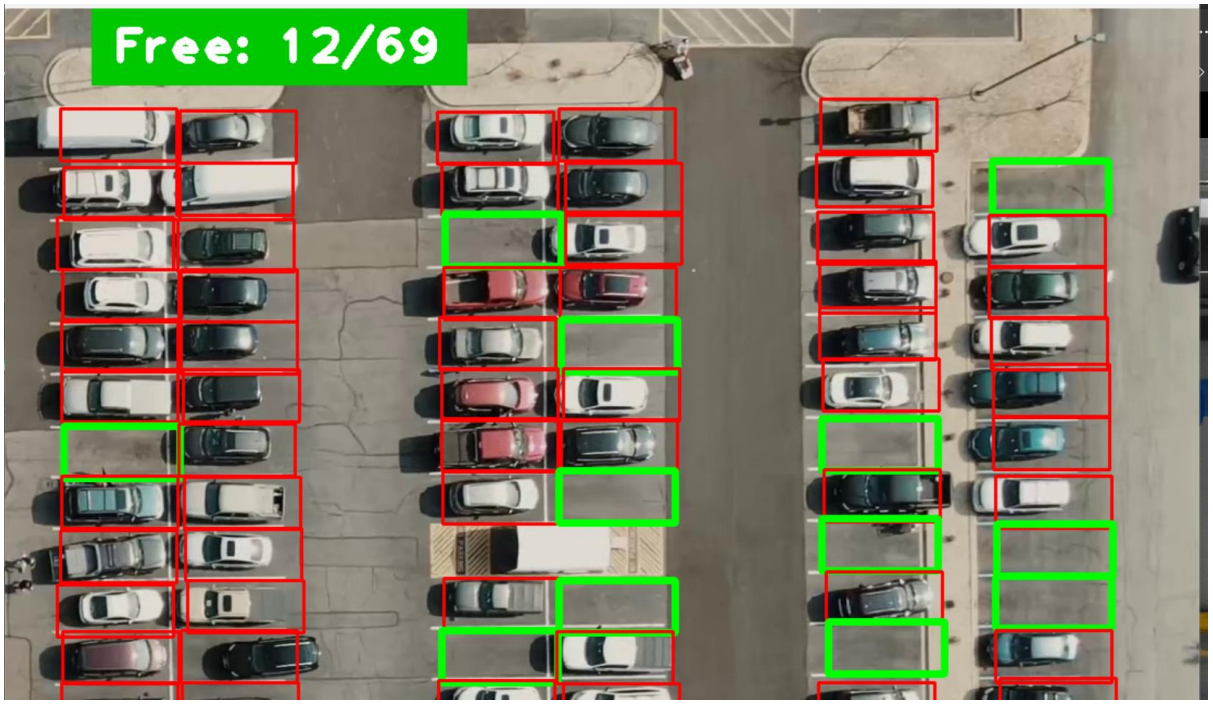
PERFORMANCE TESTING

- **PERFORMANCE METRICS**



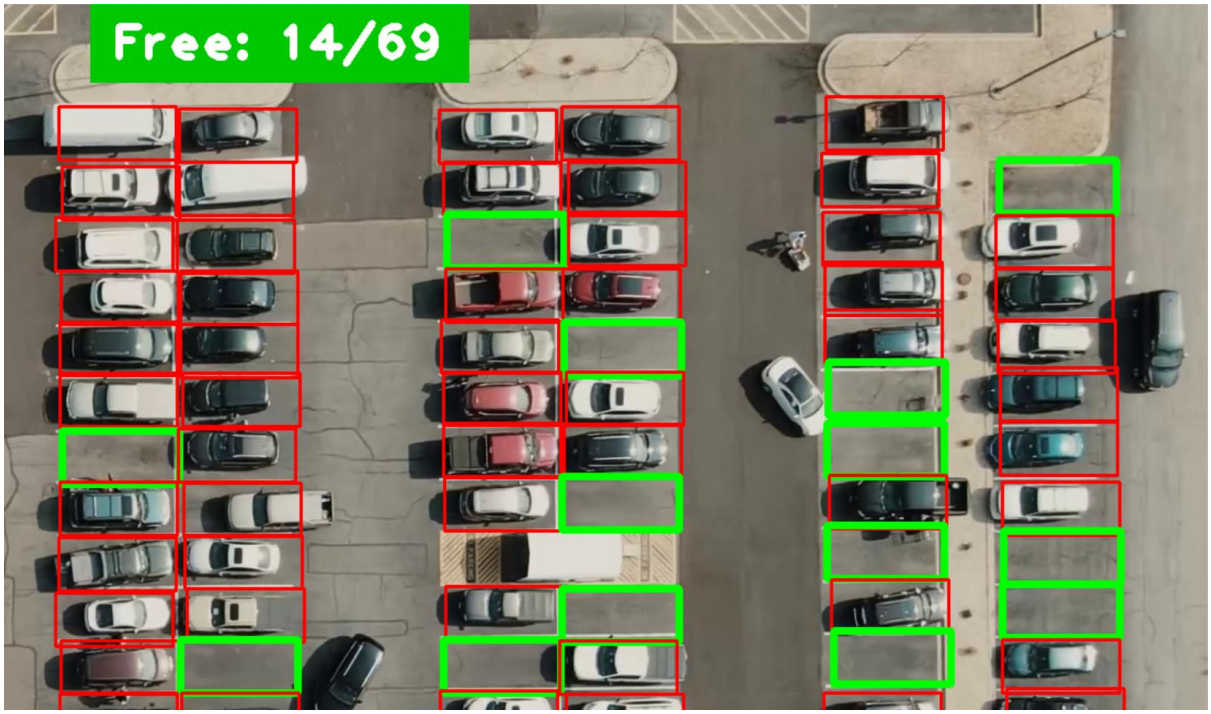

Free: 15/69

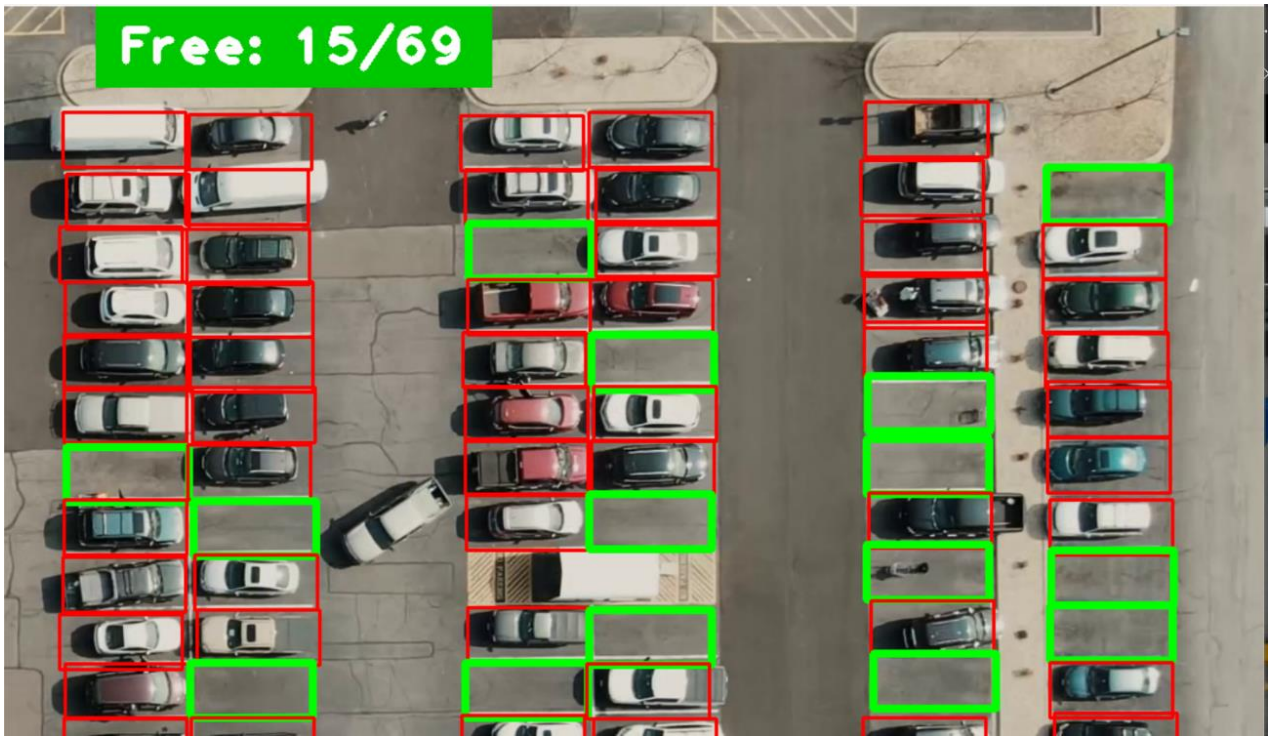

Free: 12/69

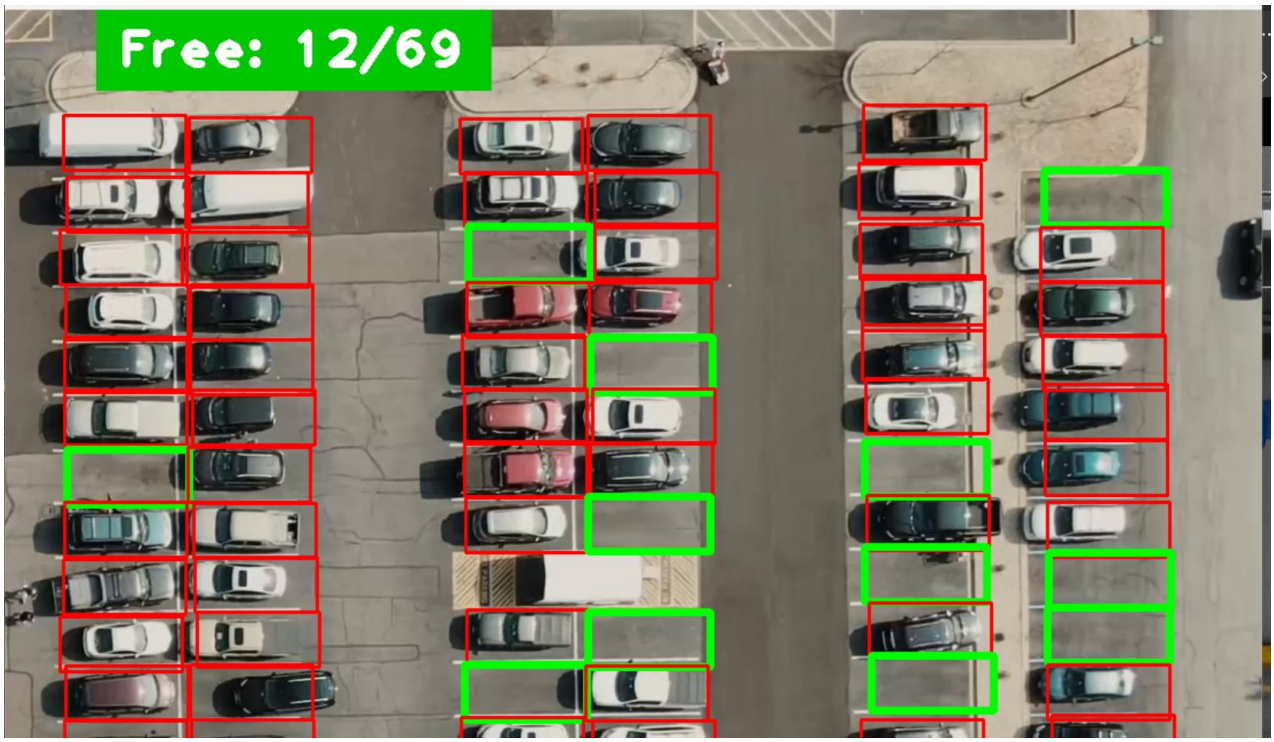


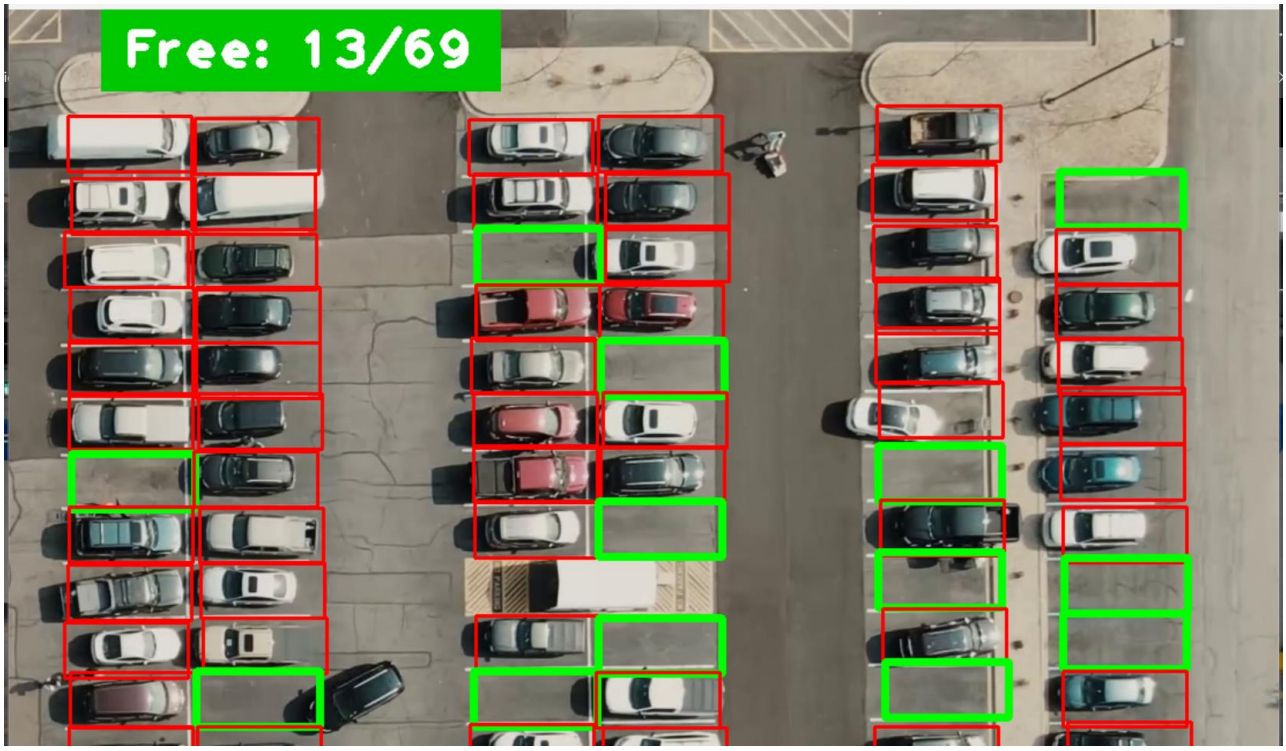
RESULTS

- **OUTPUT SCREENSHOTS**




Free: 15/69


Free: 12/69



ADVANTAGES AND DISADVANTAGES

➢ ADVANTAGES

- **Efficiency:** AI-powered systems using OpenCV can efficiently and accurately detect and track vehicles, optimizing the parking process and reducing the time it takes for drivers to find a spot.
- **Space Optimization:** These systems can analyse parking space availability in real-time, leading to better space utilization and maximizing the number of vehicles that can be accommodated.
- **Improved Traffic Flow:** By guiding drivers to available parking spaces, AI-enabled systems can contribute to smoother traffic flow within parking lots, reducing congestion and improving overall efficiency.
- **Enhanced Security:** OpenCV allows for advanced surveillance, contributing to improved security through the monitoring and analysis of activities within the parking facility.
- **User Convenience:** Drivers benefit from a more streamlined parking experience as they are directed to available spots

quickly, reducing the frustration associated with searching for parking.

➢ DISADVANTAGES

- **Cost:** Implementing AI-enabled systems can involve significant upfront costs for hardware, software, and installation, which may be a barrier for some organizations.
- **Maintenance Challenges:** Keeping the system up and running requires regular maintenance, and technical issues may arise, leading to downtime and potential disruptions.
- **Dependency on Technology:** The system's effectiveness relies on the proper functioning of the technology, and any technical glitches or failures could lead to disruptions in the parking process.
- **Privacy Concerns:** The use of surveillance technology raises privacy concerns, as it involves monitoring and analysing the movements of vehicles and potentially individuals within the parking area.
- **Adaptation Period:** Users, especially in areas with traditional parking methods, may require time to adapt to and trust the AI-enabled system, leading to a potential learning curve and initial resistance.

CONCLUSION

AI-enabled car parking systems utilizing OpenCV bring notable benefits, including increased efficiency and improved user experiences. While cost and privacy concerns exist, the overall impact on optimizing parking processes and enhancing security showcases the potential of this technology to revolutionize urban parking landscapes. With continued advancements and careful consideration of challenges, AI with OpenCV holds promise for a more streamlined and convenient future in parking solutions.

FUTURE SCOPE

The future scope of AI-enabled car parking using OpenCV holds significant potential for further advancements and widespread integration. As technology continues to evolve, several key areas present opportunities for future development:

- **Smart Cities Integration:** AI-enabled parking systems can be integrated into broader smart city initiatives. Collaborative efforts to synchronize traffic management, transportation networks, and parking solutions can contribute to more efficient urban planning.
- **Enhanced Machine Learning Algorithms:** Future iterations of AI algorithms can be refined to improve accuracy in vehicle detection, space optimization, and real-time analysis. This can lead to even more precise and reliable parking guidance systems.
- **IoT Integration:** Internet of Things (IoT) connectivity can be incorporated to enable communication between vehicles and parking infrastructure. This can facilitate dynamic parking space allocation and provide real-time updates to drivers about space availability.
- **Autonomous Vehicle Integration:** As autonomous vehicles become more prevalent, AI-enabled parking systems can adapt to accommodate self-parking features. Integration with autonomous vehicle technology could revolutionize parking processes and maximize efficiency.
- **Augmented Reality (AR) Guidance:** Implementing AR technology can provide drivers with intuitive visual cues, making it even easier to locate and navigate to available parking spaces within a parking facility.
- **Energy Efficiency:** Future systems can incorporate energy-efficient features, such as smart lighting and ventilation control, to optimize resource consumption in parking facilities based on real-time usage patterns.
- **Data Analytics for Urban Planning:** The data generated by AI-enabled parking systems can be harnessed for urban planning purposes. Analytics on parking patterns and demand can aid in designing more efficient and sustainable city infrastructures.

APPENDIX

➢ SOURCE CODE

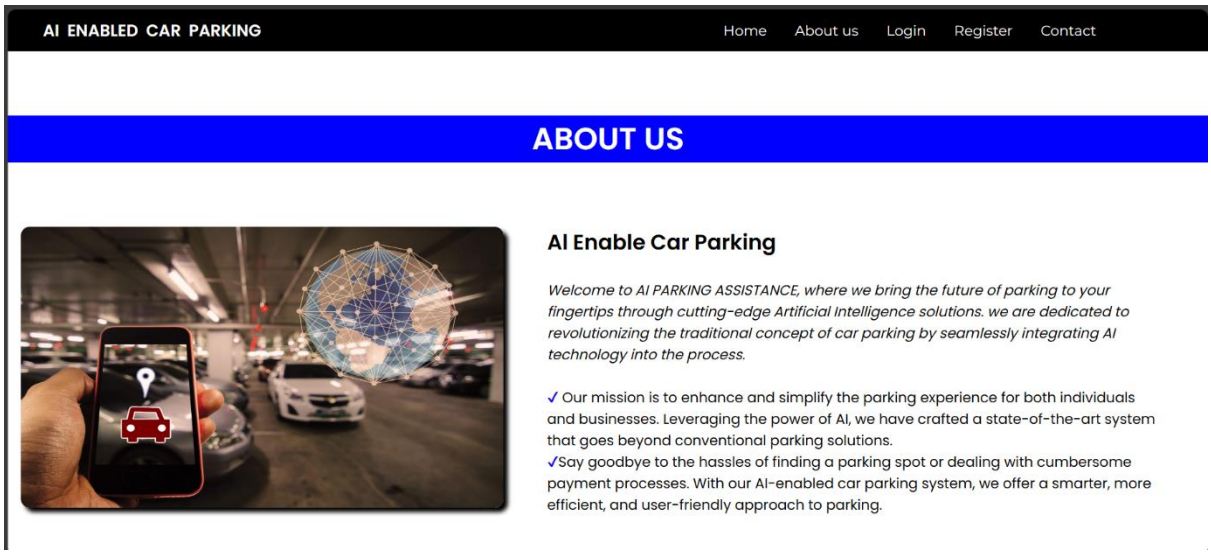
- **ROI CODE:**

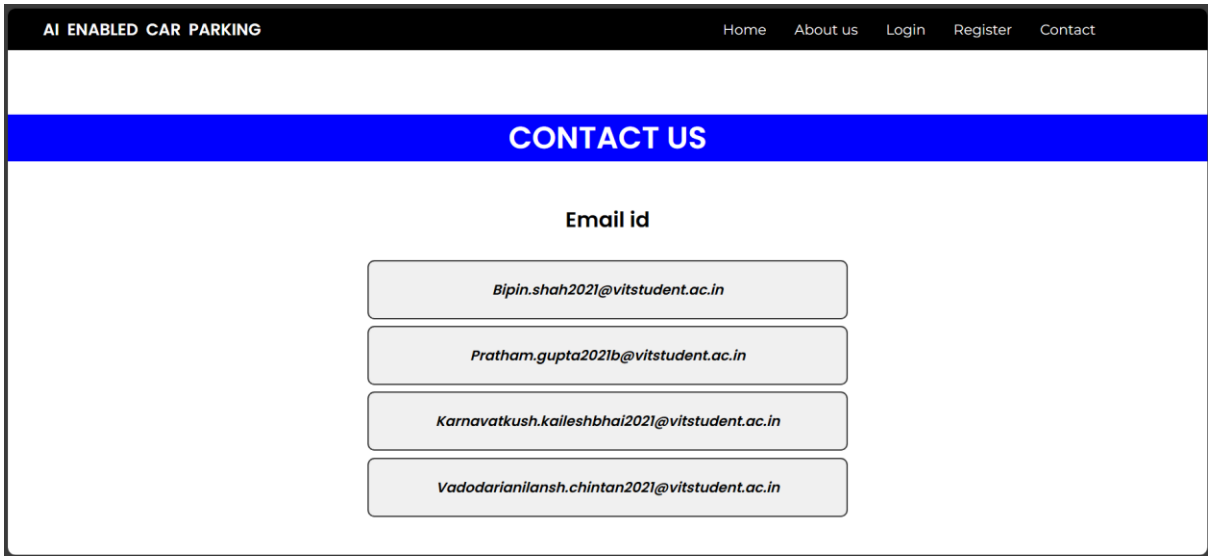
```
import cv2
import pickle

width, height = 107, 48

try:
    with open('CarParkPos', 'rb') as f:
        posList = pickle.load(f)
except:
    posList = []


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("parkingSlotPosition", 'wb') as f:
        pickle.dump(posList, f)


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

    cv2.imshow("Image", img)
    cv2.setMouseCallback("Image", mouseClick)
    cv2.waitKey(1)
```

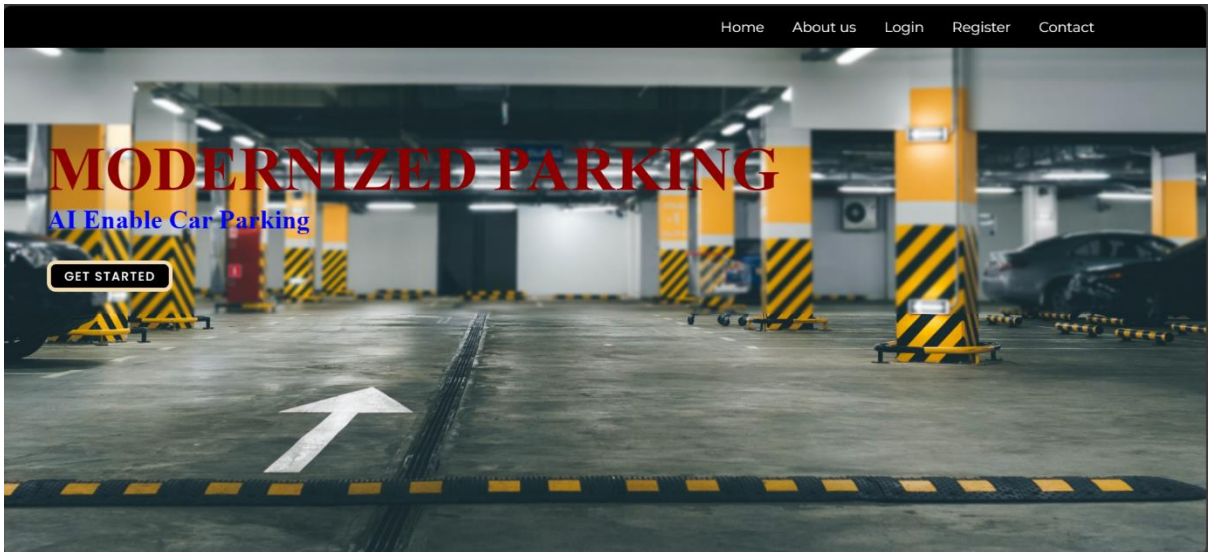
- **FLASK CODE:**

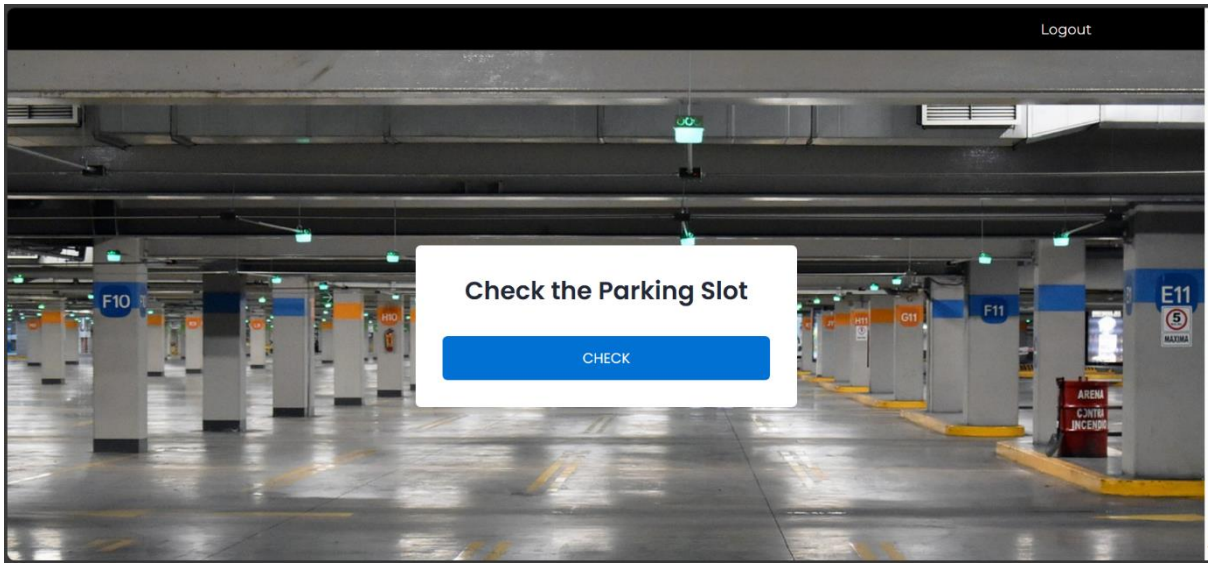
```
            def liv_pred():
                # Video feed
                current_directory = os.getcwd()

            # Assuming the video file is in the same directory as your Python
script
                video_filename = "carParkingInput.mp4"

            # Construct the full file path
                video_file_path = os.path.join(current_directory,
video_filename)
                print(video_file_path)
                cap = cv2.VideoCapture(video_file_path)
                file_path = os.path.join(current_directory, "parkingSlotPosition")
                with open(file_path, 'rb') as f:
                    posList = pickle.load(f)
                width, height = 107, 48

                def checkParkingSpace(imgPro):
                    spaceCounter = 0
                    for pos in posList:
                        x, y = pos
                        imgCrop = imgPro[y:y + height, x:x + width]
                        count = cv2.countNonZero(imgCrop)
                        if count < 900:
                            color = (0, 255, 0)
                            thickness = 5
                            spaceCounter += 1
                        else:
                            color = (0, 0, 255)
                            thickness = 2
                        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height),
color, thickness)
                    cvzone.putTextRect(img, f'Free: {spaceCounter}/{len(posList)}',
(100, 50), scale=3, thickness=5, offset=20,
                                       colorR=(0, 200, 0))

                while True:
                    if cap.get(cv2.CAP_PROP_POS_FRAMES) ==
cap.get(cv2.CAP_PROP_FRAME_COUNT):
```

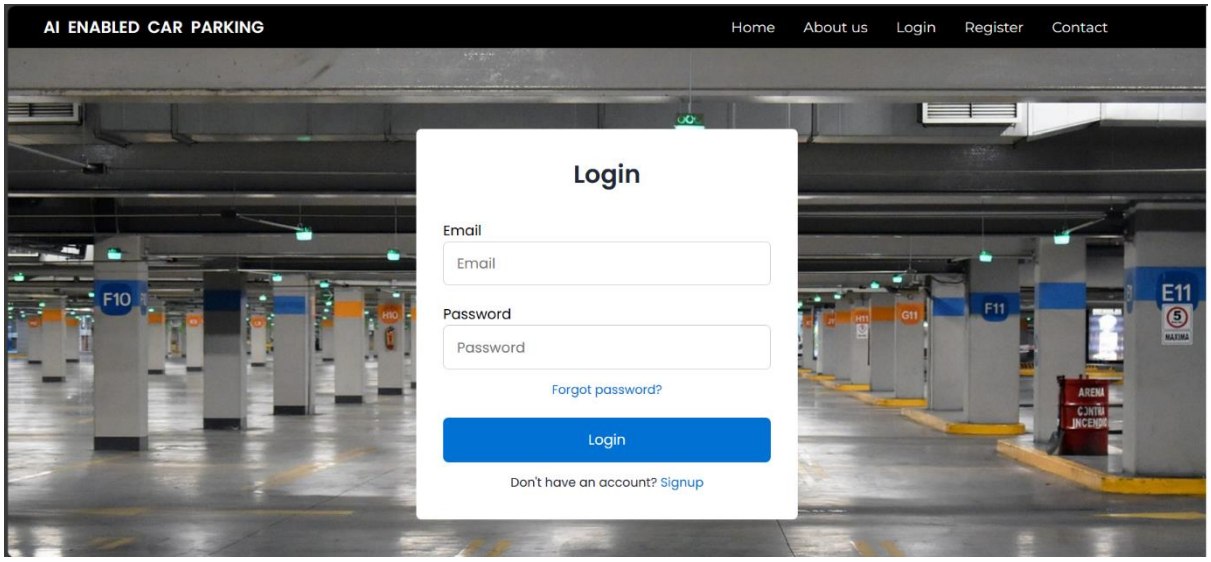
```
            cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        success, img = cap.read()
        if not success:
            break
        imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
        imgThreshold = cv2.adaptiveThreshold(imgBlur, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,
                                             25, 16)
        imgMedian = cv2.medianBlur(imgThreshold, 5)
        kernel = np.ones((3, 3), np.uint8)
        imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)
        checkParkingSpace(imgDilate)
        cv2.imshow("Image", img)

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

    cap.release()
    cv2.destroyAllWindows()
    return render_template('index.html')
```

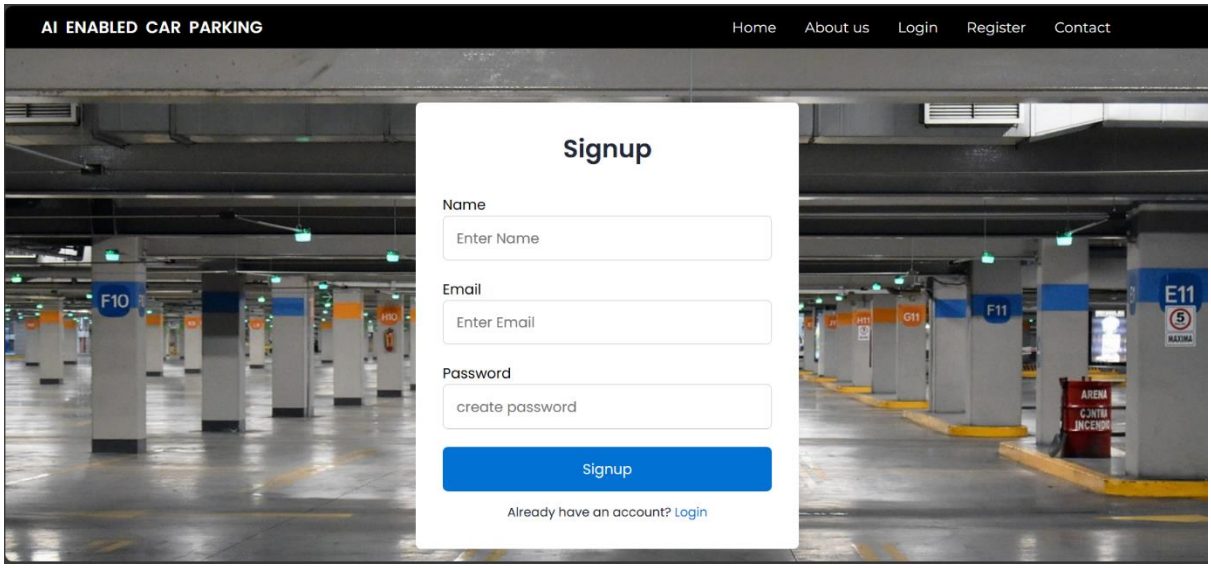



AI ENABLED CAR PARKING
Home
About us
Login
Register
Contact
CONTACT US
Email id
Bipin.shah2021@vitstudent.ac.in
Pratham.gupta2021b@vitstudent.ac.in
Karnavatkush.kaileshbhai2021@vitstudent.ac.in
Vadodarianilansh.chintan2021@vitstudent.ac.in


Home
About us
Login
Register
Contact
MODERNIZED PARKING
AI Enable Car Parking
GET STARTED


Logout
Check the Parking Slot
CHECK
F10
F11
E11





➢ GITHUB AND PROJECT DEMO LINK

- **GITHUB –** https://github.com/smartinternz02/SI-GuidedProject-608975-1698742830
- **PROJECT DEMO -** https://drive.google.com/file/d/1t22vnb6eTUy486sP3RaShA4EK-N06FBR/view?usp=sharing