PROJECT REPORT

TITLE: AI Enable Car Parking Using OpenCV

TEAM ID: Team-591796

Team Size: 1

Team Leader/Member:

SRIKAL KAKULA

21BCE7457

VIT-AP UNIVERSITY Student

srikal.21bce7457@vitapstudent.ac.in

EXTERNSHIPS DETAILS:

Externship Title: Machine Learning and Deep Learning

Application ID: SPS_APL_20230591796

Start-End Date: 08 September 2023 - 22 NOVEMBER 2023

Under The Guidance Of: SMARTBRIDGE

https://www.thesmartbridge.com/

https://smartinternz.com/student-login

INDEX

1. INTRODUCTION

1.1. Project Overview

Car parking is a common problem faced by drivers in busy urban areas. For example, imagine you are driving to a shopping mall during peak hours. As you approach the mall, you notice that the parking lot is full, and several other cars are circling around looking for available spots.

You join the queue of cars, hoping to find an available spot soon. However, as time passes, you realize that the parking lot is overcrowded, and it's becoming increasingly difficult to find a spot. You start to feel frustrated and anxious, knowing that you might be late for your appointment or miss out on a great shopping opportunity.

AI-enabled car parking using OpenCV is a computer vision-based project that aims to automate the parking process. The project involves developing an intelligent system that can identify empty parking spaces and it gives the count of available parking spots. The system uses a camera and OpenCV (Open-Source Computer Vision) library to capture live video footage of the parking lot.

1.2. Purpose

The primary purpose of this project is to address the challenges faced by drivers in urban areas when searching for parking spaces. The prevalent issues of frustration, anxiety, and delays stemming from the lack of efficient parking management inspire the development of an innovative solution. By leveraging the power of artificial intelligence (AI) and computer vision through OpenCV, the project aims to streamline and automate the parking process. Through real-time video processing, the system identifies and counts vacant parking spaces, providing a user-friendly interface via Flask, a web framework. The ultimate goal is to revolutionize the driving and parking experience, contributing to sustainable urban development by reducing search times, frustration, and emissions, while improving traffic management and customer satisfaction.

Moreover, from a business perspective, the project aims to introduce a cost-effective and scalable solution for parking management. By minimizing the need for manual staff and simplifying operations, the AI-powered parking system presents a viable revenue model. Its adaptability to various urban environments ensures broad accessibility, while the modular architecture supports future enhancements and seamless integration with emerging technologies, reflecting a commitment to long-term sustainability and innovation in the field of smart parking solutions.

2. LITERATURE SURVEY

2.1. Existing problem

The rapid increase in vehicle ownership and urbanization has led to a significant shortage of parking spaces in urban areas. This has resulted in a multitude of problems for drivers, including:

- **Extended search times:** Drivers often spend considerable time circling around urban areas, searching for vacant parking spaces. This wasted time can lead to frustration, anxiety, and delays.
- **Traffic congestion:** The process of searching for parking spaces contributes to traffic congestion, as drivers manoeuvre slowly through streets and parking lots. This congestion not only slows down traffic but also increases fuel consumption and emissions.
- **Illegal parking:** Due to the scarcity of parking spaces, drivers often resort to illegal parking practices, such as parking in restricted areas or double parking. This can lead to safety hazards and obstruct the flow of traffic.

The lack of efficient parking management systems further exacerbates these problems. Traditional parking methods, such as manual counting and signage, are often outdated and ineffective. This leads to inaccurate information about available parking spaces, further frustrating drivers and contributing to congestion. Additionally, the lack of real-time parking information prevents drivers from making informed decisions about parking options, leading to inefficient use of parking resources.

2.2. References

1. "Smart Parking System using Python and OpenCV" by International Journal of Research and Analytical Studies (IJRASET)

Summary: This paper presents an image-processing-based smart parking system designed for multistory parking garages, open parking lots, and other scenarios. The system utilizes edge detection and coordinate bound pixel sections to determine whether a parking spot is occupied or not. It is implemented in Python using the OpenCV library and demonstrates the concept of image-to-text conversion using Tesseract.

Link: https://www.ijraset.com/research-paper/smart-parking-system-using-python-and-opencv

2. "AUTOMATING PARKING SYSTEM USING PYTHON AND OPENCV" by International Research Journal on Engineering and Management (IRJMETS)

Summary: This paper describes the development of an automated parking system using Python and OpenCV. The system employs OpenCV's Cascade Classifier to detect and recognize vehicle license plates, enabling the identification of occupied and vacant parking spaces. It highlights the efficiency and accuracy of using OpenCV for image processing tasks in parking management applications.

Link: https://www.academia.edu/43353754/Smart_Car_Parking_System_using_IR_Sensor

3. "EasyChair Preprint Car Parking Space Detection Using OpenCV"

Summary: This study focuses on the development of an AI-enabled car parking system using OpenCV for parking space detection. The system utilizes background subtraction and morphological operations to identify and count empty parking spaces in real-time. The proposed algorithm demonstrates high accuracy and efficiency, making it a promising solution for smart parking applications.

Link: https://easychair.org/publications/preprint_open/hVnT

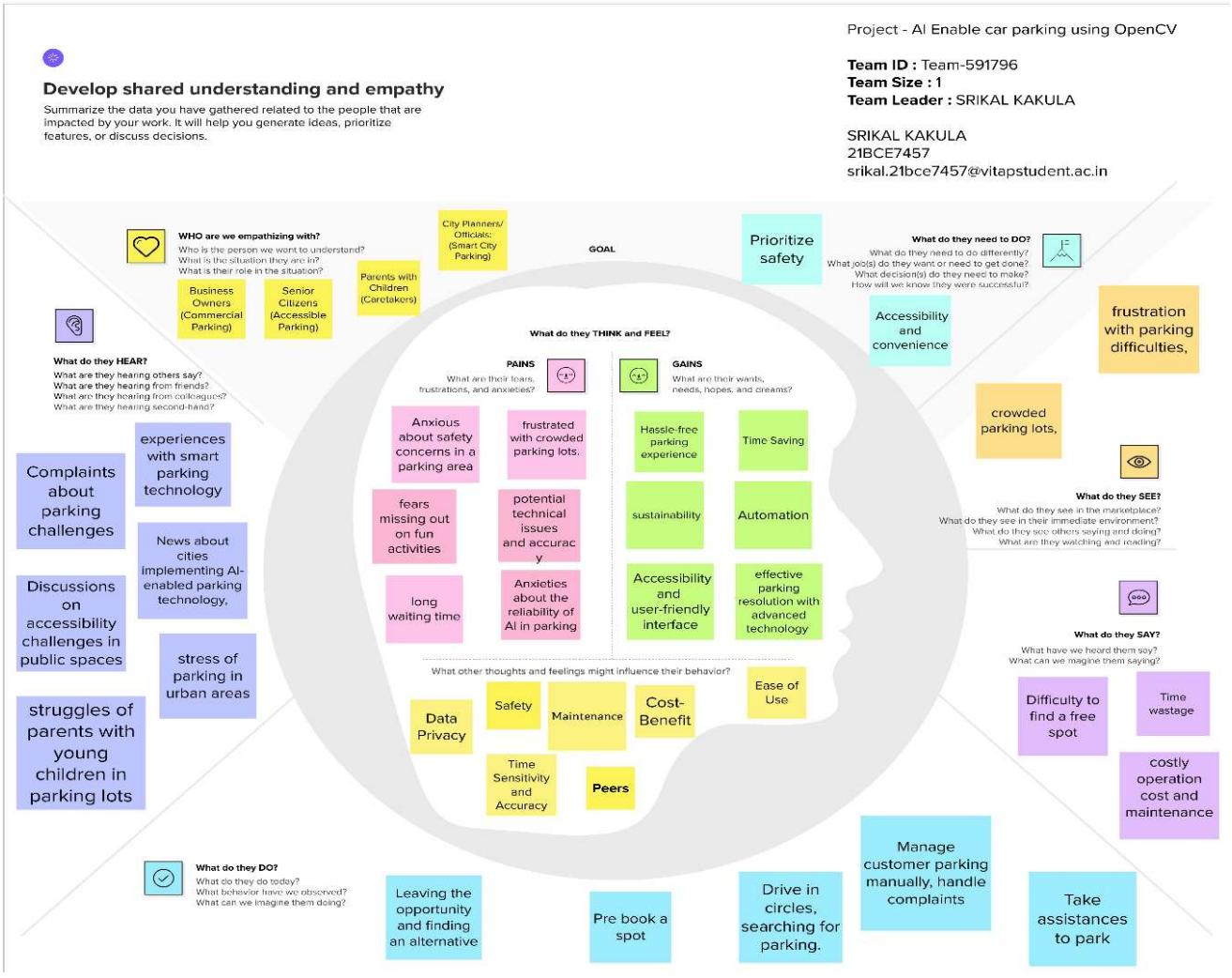
2.3. Problem Statement Definition

Parameter	Description
Problem Statement (Problem to be solved)	The problem at hand revolves around the challenges faced by drivers in crowded urban areas when searching for parking spaces, leading to frustration, anxiety, and potential delays. The need is to streamline and automate the parking process using AI and computer vision.
Idea / Solution description	The proposed solution involves implementing an AI-enabled car parking system using OpenCV, a computer vision library. A camera captures live video footage of the parking lot, and the system employs image processing techniques to identify and count empty parking spaces. The solution integrates with Flask, a web framework, to provide a user interface for accessing the parking information.

Novelty / Uniqueness	The uniqueness of this solution lies in its combination of computer vision, AI, and web technology to create an intelligent parking management system. The use of OpenCV for real-time video processing, coupled with a Flask-based web interface, distinguishes this solution in its holistic approach to address the parking problem.
Social Impact / Customer Satisfaction	AI-powered parking revolutionizes the driving and parking experience, aligning with sustainability goals. It reduces search times, frustration, and emissions. It improves traffic management, reduces congestion, and promotes a sustainable urban environment. The integration of AI in parking solutions enhances customer satisfaction and aligns with sustainability objectives.
Business Model (Revenue Model)	The parking solution powered by AI minimizes the requirement of manual staff for managing parking spaces, simplifying operations and reducing labour expenses. It is designed specifically for parking and effectively manages the parking process, enhancing resource utilization. Its adaptable and user-friendly design is suitable for various urban environments, ensuring easy access and future adaptability.
Scalability of the Solution	The solution is highly scalable, leveraging widely adopted technologies like OpenCV and Flask for easy deployment in diverse urban environments. Its horizontal scalability accommodates additional parking facilities, and a lightweight, web-based interface ensures accessibility across various devices. The modular architecture supports future enhancements and seamless integration with emerging technologies.

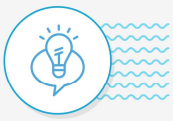
3. IDEATION & PROPOSED SOLUTION

3.1. Empathy Map Canvas



3.2. Ideation & Brainstorming

Templ



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

Project - AI Enable car parking using OpenCV

Team ID : Team-591796
Team Size : 1
Team Leader : SRIKAL KAKULA

SRIKAL KAKULA
21BCE7457
srikal.21bce7457@vitapstudent.ac.in

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

The existing parking management systems encounter significant challenges in optimizing space utilization and providing an efficient and user-friendly experience for both drivers and facility operators. Conventional methods, often relying on manual monitoring or outdated technologies, fall short in accurately identifying and managing available parking spaces in real-time. This results in underutilized parking spaces, increased traffic congestion, and suboptimal resource allocation

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

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!

SRIKAL KAKULA	Person 1	Person 2	Person 3
Integrate license plate recognition for personalized parking access. Develop a data analytics dashboard for facility optimization.	Integrate biometric authentication for enhanced parking security and access control.	By studing traffic behaviour to predict future events	a mini device which move around the parking arear to find the free spots with integration with open cv
Mobile application to find a free spot near by using gps	Implement a feedback loop through the app for reporting inaccuracies in parking availability	Integrate IoT sensors for real-time occupancy and environmental monitoring	Access the nearest free parking area using gps and OpenCV video footage to give map to the free spot
automated alerts and notifications to alert facility operators to potential issues, such as illegally parked vehicles or maintenance needs, and ensure prompt resolution	AI parking markering the vechicle entry/exit flow	reservation system for pre-booking parking spaces, reducing congestion.	Combine a mobile app with the OpenCV system to provide users with real-time information on available parking spaces and make parking easier
A parking allocation app that assigns parking spaces based on visitors' preferred time slots and utilizes an advanced algorithm to efficiently manage the parking spaces based on the duration of parking.	Install a camera-based system at parking lot entrances and exits to monitor parking space occupancy in real-time using OpenCV	Enhance space occupancy detection with smart parking sensors and OpenCV	keeping a smart video analysis of footage to dtect cars
Use machine learning algorithms to forecast peak parking hours and adjust pricing dynamically to encourage off-peak usage, optimizing parking space utilization.			Drones with cameras, AI, and ML algorithms like YOLO are used to find the best parking spot near the car in real-time.

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 and 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.

Access & Optimization

- Integrate license plate recognition for personalized parking access.
- Implement a reservation system for pre-booking parking spaces to reduce congestion.
- Utilize machine learning algorithms to dynamically adjust pricing for optimized space utilization

Advance Tech

- Implement biometric authentication to enhance parking security.
- space occupancy detection by integrating smart parking sensors with OpenCV
- Utilize drones with cameras, AI, and ML algorithms like YOLO for real-time parking spot optimization

Real Time Navigation

- Create a mobile application using GPS for finding nearby free parking spots
- Combine the mobile app with OpenCV for real-time updates on available parking spaces
- GPS and OpenCV video footage to guide users to the nearest free parking area.

Inovation

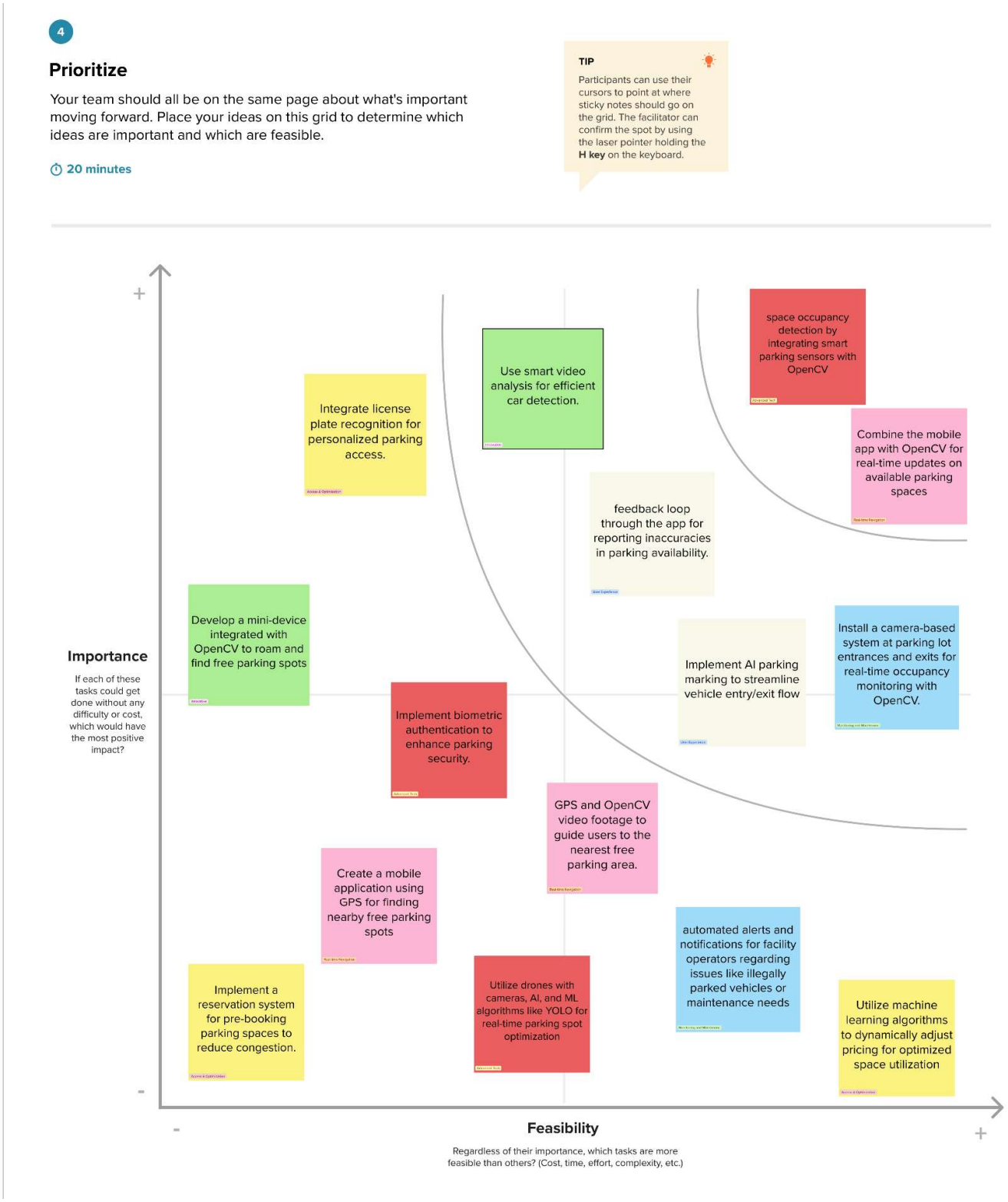
- Use smart video analysis for efficient car detection.
- Develop a mini-device integrated with OpenCV to roam and find free parking spots

Monitoring

- Install a camera-based system at parking lot entrances and exits for real-time occupancy monitoring with OpenCV.
- automated alerts and notifications for facility operators regarding issues like illegally parked vehicles or maintenance needs

User Experience

- feedback loop through the app for reporting inaccuracies in parking availability.
- Implement AI parking marking to streamline vehicle entry/exit flow



4. REQUIREMENT ANALYSIS

4.1. Functional requirement

- **Parking Space Detection:** The system must accurately detect and identify empty parking spaces in real-time using computer vision algorithms.
- **User Interface:** The solution should provide a user-friendly interface accessible via web browsers. It should display real-time parking availability, allowing users to easily navigate and find vacant spots.

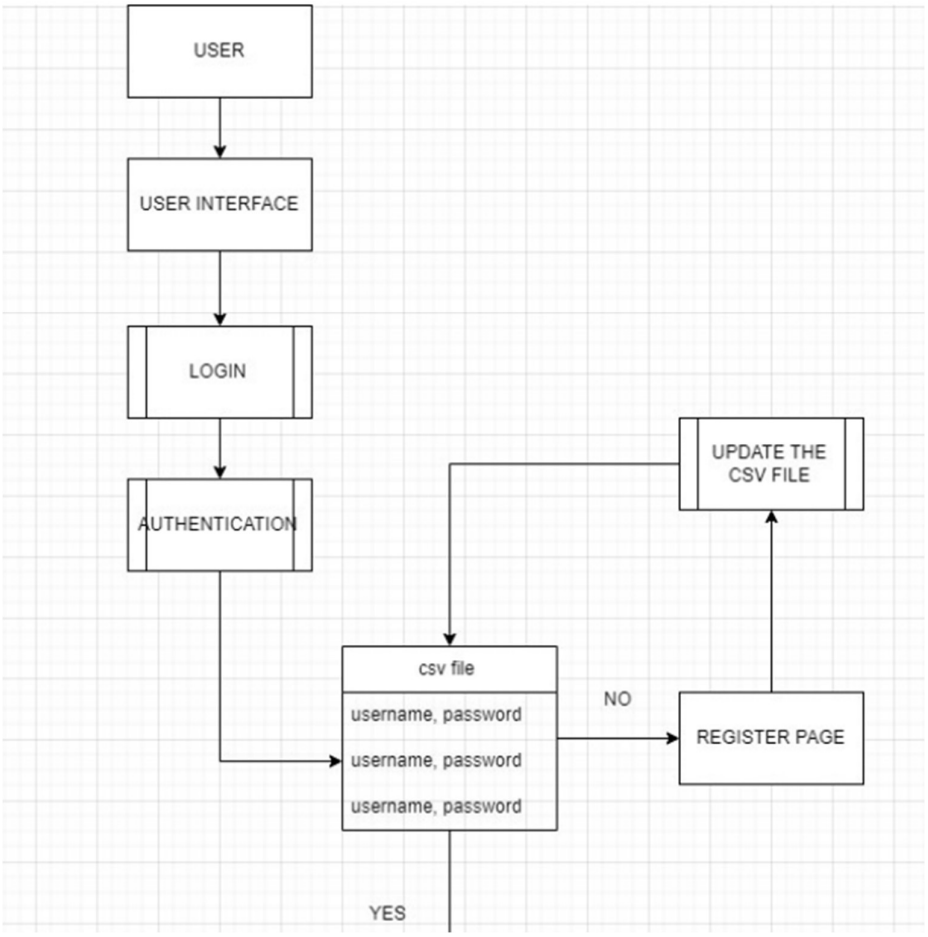
- **Real-time parking space detection:** The system should be able to identify and count vacant parking spaces in real time using image processing techniques.
- **Integration with parking infrastructure:** The system should be able to integrate with existing parking infrastructure, such as parking lot sensors and signage, to provide comprehensive parking information.

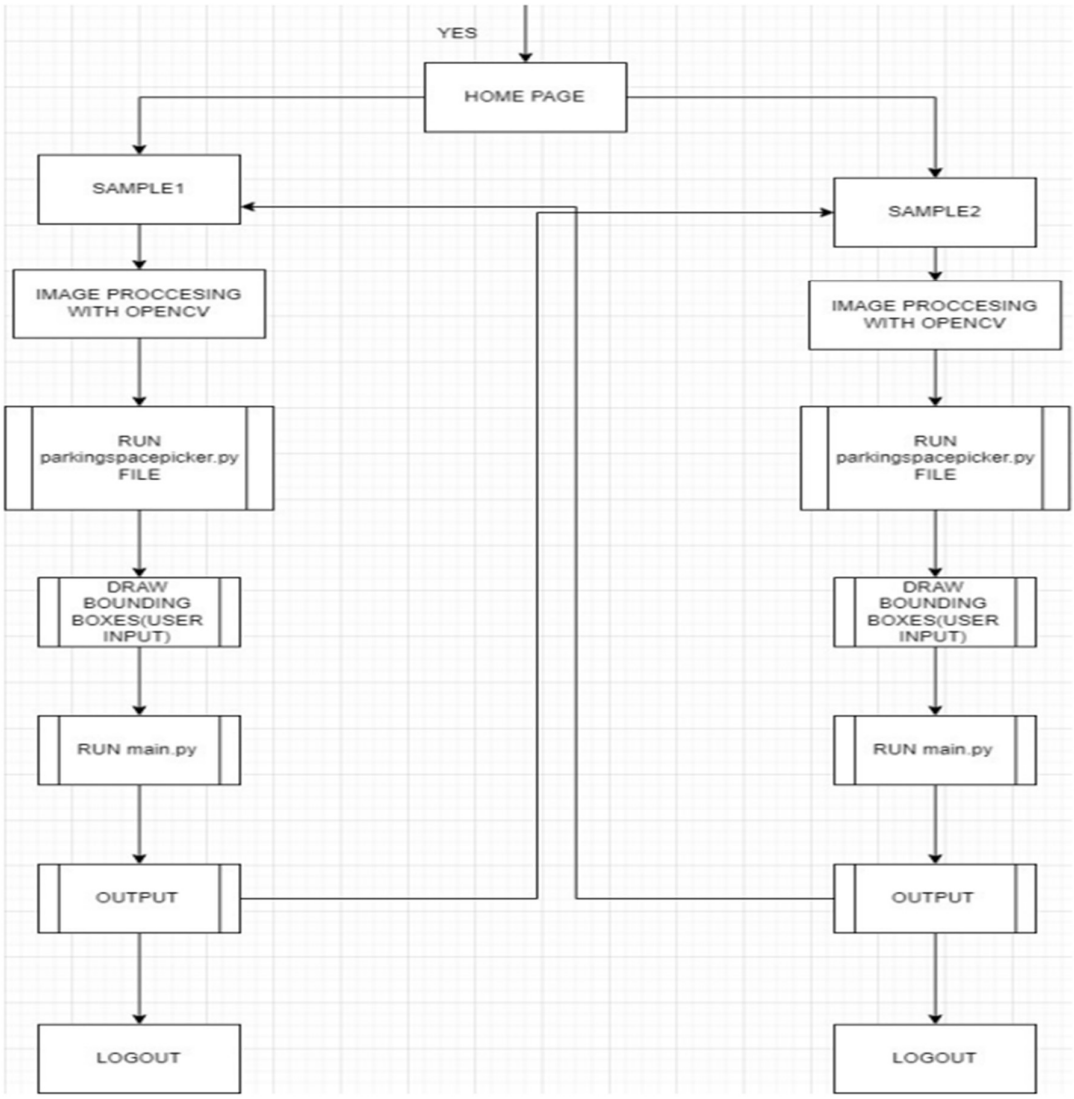
4.2. Non-Functional requirements

- **Performance:** The system should be able to process real-time video footage efficiently and provide parking information with minimal latency.
- **Accuracy:** The system should have a high degree of accuracy in identifying vacant parking spaces, minimizing false positives and false negatives.
- **Scalability:** The system should be scalable to accommodate different parking lot sizes and environments.
- **Security:** The system should implement robust security measures to protect parking data and prevent unauthorized access.
- **Reliability:** The system should be reliable and operate continuously with minimal downtime.
- **Usability:** The system should be easy to use and understand for drivers of all technical backgrounds.
- **Maintainability:** The system should be well-documented and easy to maintain for software developers.
- **Adaptability:** The system should be adaptable to future technologies and advancements in computer vision and AI.

5. PROJECT DESIGN

5.1. Data Flow Diagrams & User Stories

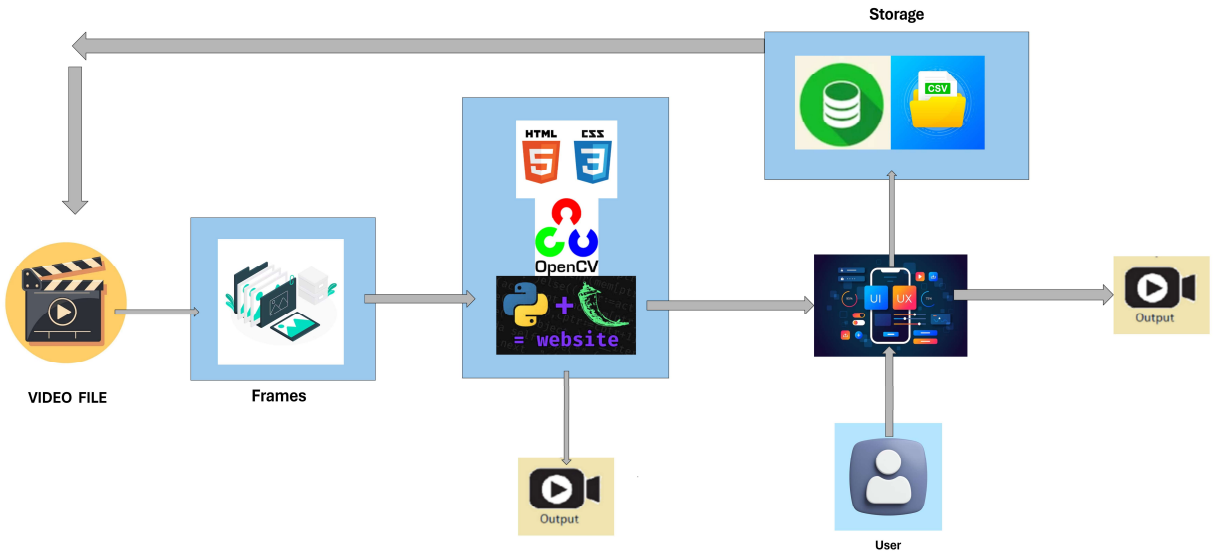




User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Login	USN-1	As a user, I want to log into the AI-enabled car parking system securely using reliable authentication mechanisms to prevent unauthorized access.	Authorized login	High	Sprint 1
Customer (Web user)	Registration	USN-2	As a user, i want to register if i dont have the correct login credentials	Register a new user if not a user	High	Sprint 1
Customer (Web user)	Registration	USN-3	As a user, there should not be any similar user names in	No duplicate usernames	High	Sprint 1

			the database to avoid confusion			
Customer (Web user)	Home Dashboard	USN-4	As a user i want to access all the sample programs easily	Able to run the project sample files	Medium	Sprint 2
Customer (Web user)	Home Dashboard	USN-5	As a user, i want to know the problem statement, advantage and disadvantages of the project	Get project information	Medium	Sprint 2
Customer (Web user)	Home Dashboard	USN-6	As a user, i want more examples of the project	More than 1 sample project files	Low	Sprint 3
Customer (Web user)	Home Dashboard	USN-7	As a user i want a safe and secure use of the platform without any errors	Authenticated platform	High	Sprint 3
Customer (Web user)	Features	USN-8	As a user, i want my data to be safe and protected	secure(use sha256 encryption)	High	Sprint 1
Customer (Web user)	Features	USN-9	As a user, i want the user interface to be easy to use and faster to process	User friendly interface	Medium	Sprint 1
Customer (Web user)	Logout	USN-10	As a user, i need a option to logout the web application	Logout option	Medium	Sprint 1
Customer (Web user)	Features	USN-11	As a user i want the i option to choose the bounding boxes as per my requirement	Able to give bounding boxes to the video file	High	Sprint 3
Customer (Web user)	Features	USN-12	As a user, i want to view the original file of the predicted file	View original file of the sample file	Medium	Sprint 2
Customer (Web user)	Features	USN-13	As a user i want to run the individual files of the opencv code	An option to select and execute specific OpenCV files	High	Sprint 2
Customer (Web user)	Features	USN-14	As a user i want to undo the bounding boxes with are wrongly allocated	Option to edit the bounding boxes	High	Sprint 3
Administrator	Features	ADM-1	As a admin, i should protect the data of user and provide the fast and resilient code to predict the empty spaces	Best performance of the project	High	Sprint 3

5.2. Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1. Technical Architecture

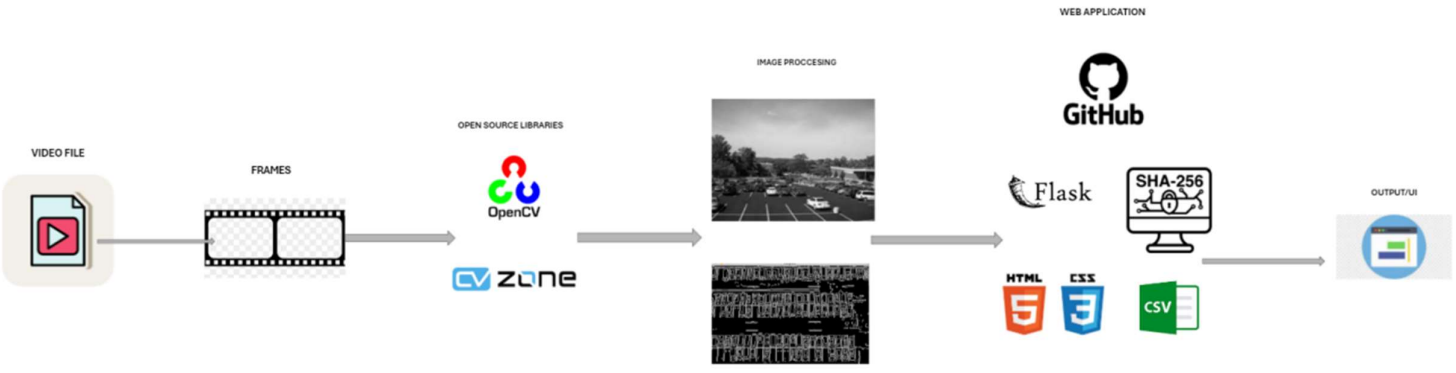
Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1	User Interface	Web UI rendering	PYTHON ,FLASK
2	Video file reading	Video feed and parking space visualization	OpenCV, cvzone
3	Application Logic-1	Parking space occupancy analysis	Python
4	Application Logic-2	File storage and loading of parking positions	Python, pickle
5	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud:	Local Server Configuration, Cloud Server Configuration(GitHub)
6	Database	Data storage for parking space positions	Pickle file
7	User Interface	Front-end markup language for web interface & Styling language for web interface	HTML, CSS,
9	Login Credentials	Storage of user login information	CSV File

10	Password Encryption	Encryption method for password storage	SHA-256

Table-2: Application Characteristics:

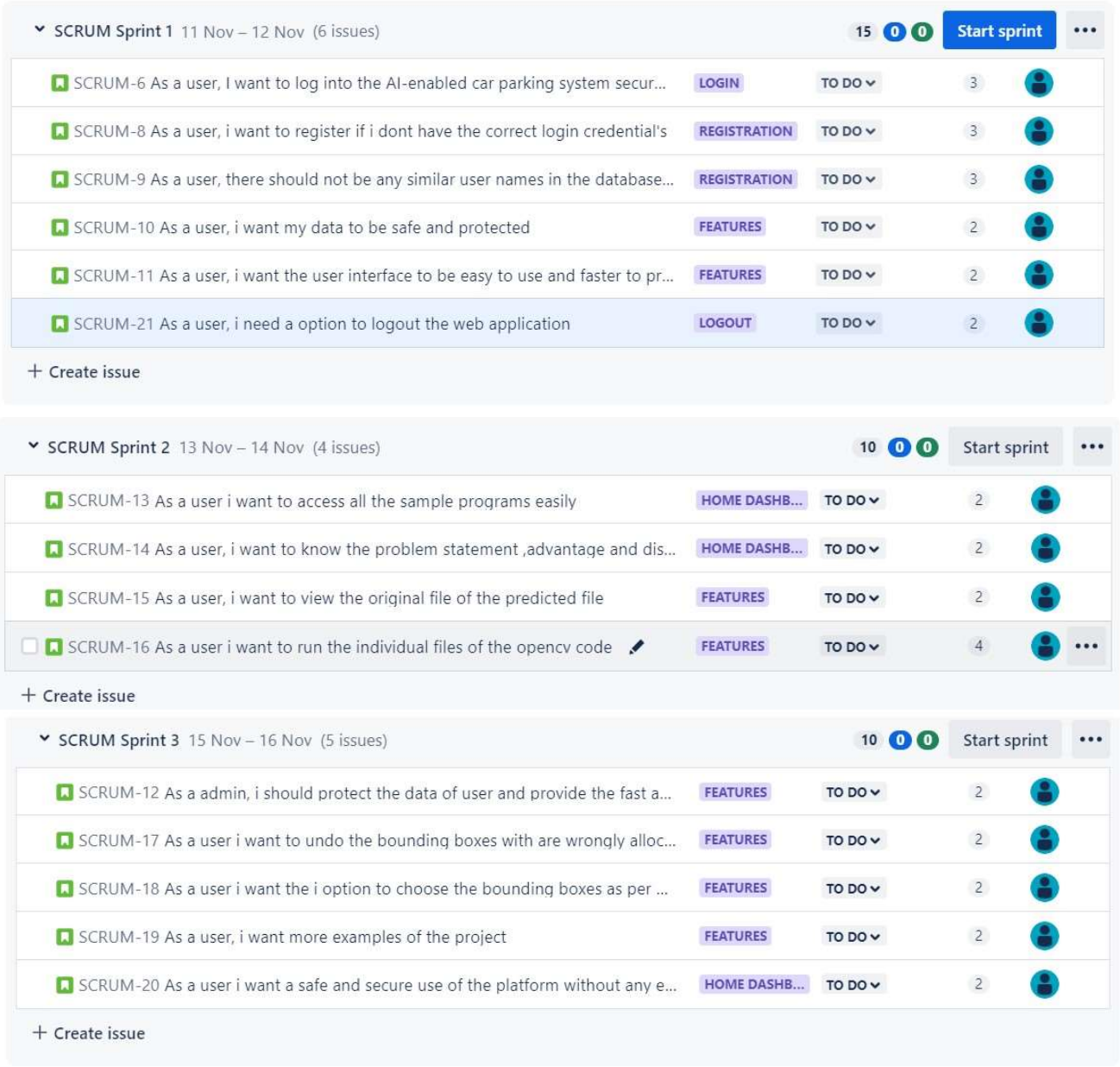
S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Utilization of open-source frameworks for image processing and UI development	OpenCV, cvzone, HTML, CSS,Flask
2.	Security Implementations	Implementation of SHA-256 encryption for password storage and access control for parking space information	Python (hashlib library for SHA-256), Authentication mechanisms
3.	Scalable Architecture	The architecture is designed for scalability through modularized image processing logic, enabling adaptability to varying parking lot sizes and configurations	Python (Modular code structure)
4.	Availability	Ensured availability through the use of load balancers and distribution across multiple servers, minimizing downtime and enhancing fault tolerance	Load balancers (Not explicitly implemented in the provided code, but suggested for availability)
5.	Performance	Designed for optimal performance, considering factors such as requests per second, implementing caching mechanisms, and utilizing Content Delivery Networks (CDNs) for efficient content delivery	Caching (Not explicitly implemented in the provided code), Efficient algorithm design



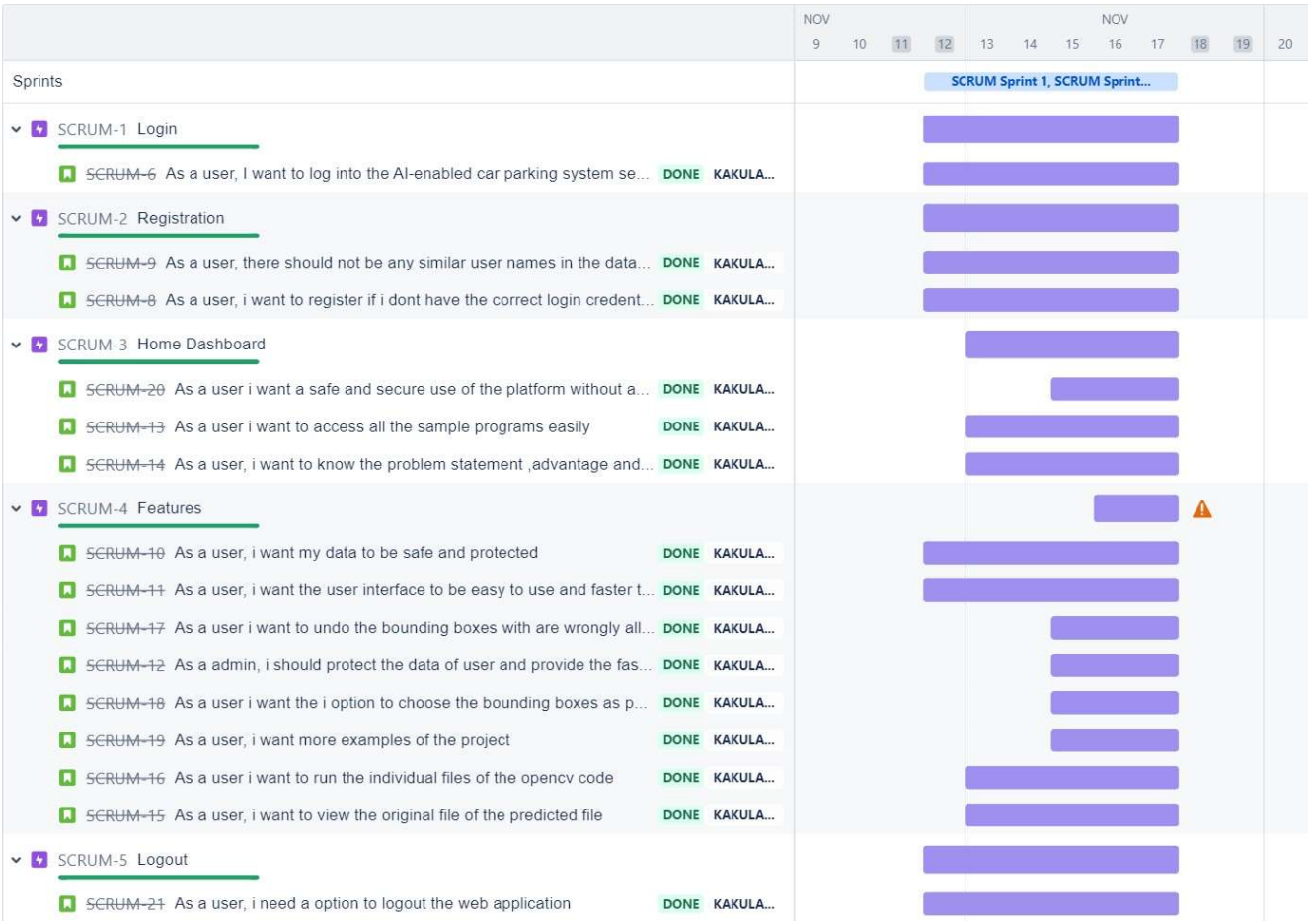
6.2. Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Login	USN-1	As a user, I want to log into the AI-enabled car parking system securely using reliable authentication mechanisms to prevent unauthorized access.	3	High	SRIKAL KAKULA
Sprint 1	Registration	USN-2	As a user, i want to register if i dont have the correct login credentials	3	High	SRIKAL KAKULA
Sprint 1	Registration	USN-3	As a user, there should not be any similar user names in the database to avoid confusion	3	High	SRIKAL KAKULA
Sprint 2	Home Dashboard	USN-4	As a user i want to access all the sample programs easily	2	Medium	SRIKAL KAKULA
Sprint 2	Home Dashboard	USN-5	As a user, i want to know the problem statement ,advantage and disadvantages of the project	2	Medium	SRIKAL KAKULA
Sprint 3	Home Dashboard	USN-6	As a user, i want more examples of the project	2	Low	SRIKAL KAKULA
Sprint 3	Home Dashboard	USN-7	As a user i want a safe and secure use of the platform without any errors	2	High	SRIKAL KAKULA
Sprint 1	Features	USN-8	As a user, i want my data to be safe and protected	2	High	SRIKAL KAKULA
Sprint 1	Features	USN-9	As a user, i want the user interface to be easy to use and faster to process	2	Medium	SRIKAL KAKULA
Sprint 1	Logout	USN-10	As a user, i need a option to logout the web application	2	Medium	SRIKAL KAKULA
Sprint 3	Features	USN-11	As a user i want the i option to choose the bounding boxes as per my requirement	2	High	SRIKAL KAKULA
Sprint 2	Features	USN-12	As a user, i want to view the original file of the predicted file	2	Medium	SRIKAL KAKULA

Sprint 2	Features	USN-13	As a user i want to run the individual files of the opencv code	4	High	SRIKAL KAKULA
Sprint 3	Features	USN-14	As a user i want to undo the bounding boxes with are wrongly allocated	2	High	SRIKAL KAKULA
Sprint 3	Features	ADM-1	As a admin, i should protect the data of user and provide the fast and resilient code to predict the empty spaces	2	High	SRIKAL KAKULA



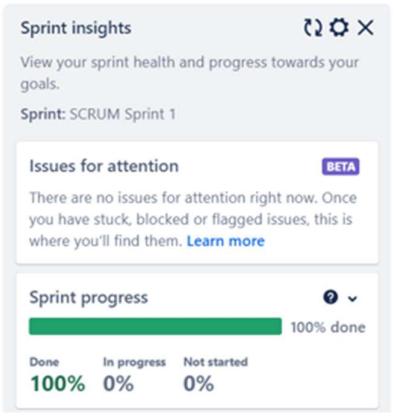
6.3. Sprint Delivery Schedule

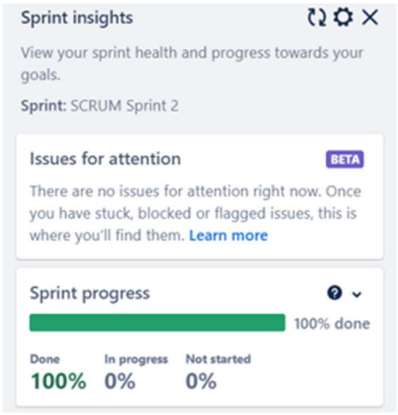


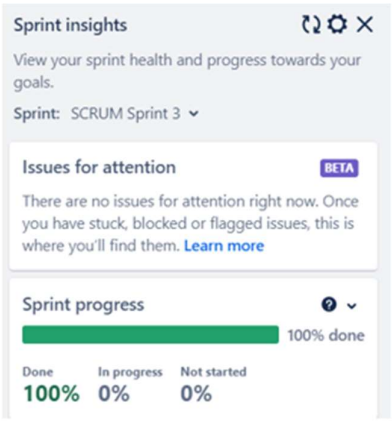
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	15	5 Days	12 Nov 2023	16 Nov 2023	15	17 Nov 2023
Sprint-2	10	4 Days	13 Nov 2023	16 Nov 2023	10	17 Nov 2023
Sprint-3	10	4 Days	15 Nov 2023	18 Nov 2023	10	17 Nov 2023

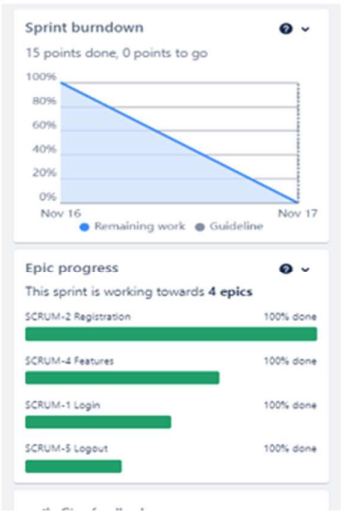
Velocity: $AV = \frac{sprint\ duration}{velocity}$

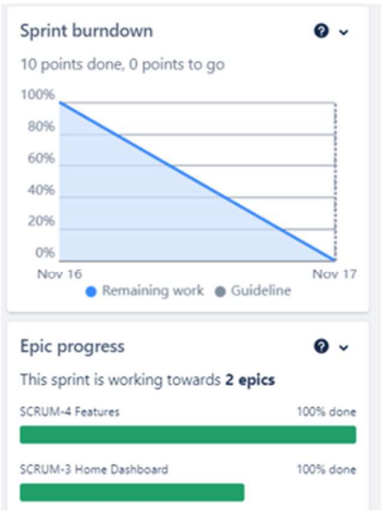
AV = 15+10+10 / 5+4+4 = 35/13 = 2.69

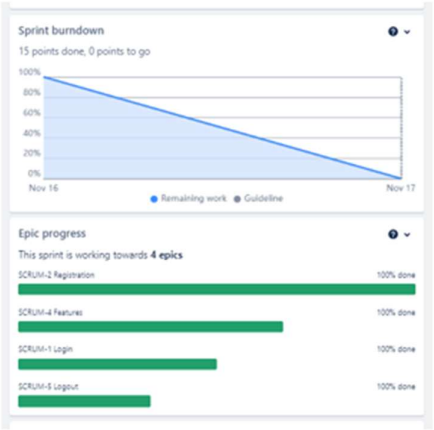












7. CODING & SOLUTIONING

7.1. Project Workflow:

- **Data Collection**

  Download the dataset

- **ROI (Region of interest)**

  Create python file

  Import required libraries

  Define ROI width and height

  Select and deselect ROI

  Denote ROI with BBOX

- **Video Processing and object detection**

  Import required libraries

  Reading input and loading ROI file

Checking for parking space

Looping the video

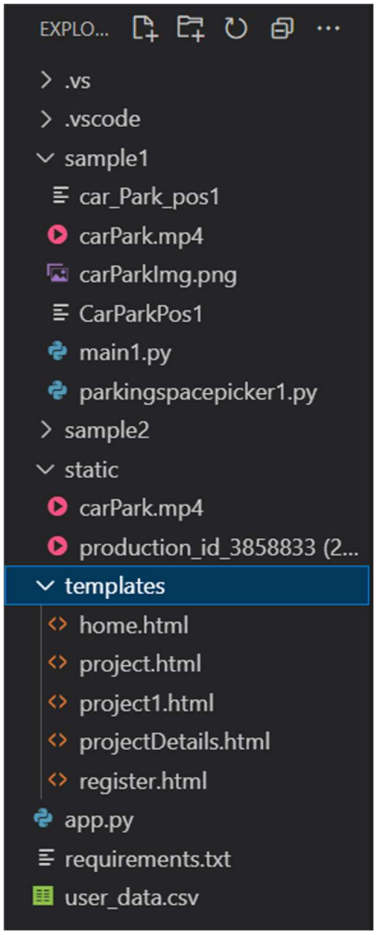
Frame processing and empty parking slot counters

- **Application building**

  Build HTML

  Build python script for Flask

7.2. Project Structure:



The Project folder contains:

.vs: This file contains project configuration settings.

.vscode: This file contains Visual Studio Code configuration settings.

sample1: This folder contains the following files:

carPark.mp4: A video file of a parking lot.

carParkimg.png: An image of the parking lot.

main1.py: The main Python script for the project.

parkingspacepicker1.py: A Python script that implements the parking space detection algorithm.

sample2: This folder contains the following files:

static: A folder containing static resources, such as images and CSS files.

templates: A folder containing HTML templates for the user interface.

app.py: The main Python script for the production code.

requirements.txt: A file that lists the Python dependencies for the project.

user_data.csv: A CSV file containing user data.

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.

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++.

Parkingspacepicker.py

Pickle: The pickle library in Python is used for serialization and de-serialization of Python objects.

```
parkingspacepicker1.py
sample1 > parkingspacepicker1.py > ...
1   import cv2
2   import pickle
```

- In this, the code imports the necessary libraries: OpenCV (cv2) for image processing and computer vision, and pickle for serializing and deserializing Python objects.
- Here, the width and height variables are set to define the dimensions of the rectangular region that will be marked as a parking space.
- The code attempts to load previously saved parking space positions from a file (CarParkPos1).
- If the file exists and can be loaded, the posList variable is assigned the loaded positions.
- If the file doesn't exist or an error occurs during loading, an empty list is assigned to posList.
- This function, mouseClick, is the callback function for mouse events.
- It appends the coordinates (x, y) to posList when the left mouse button is clicked (EVENT_LBUTTONDOWN).

```
4
5   width, height = 107, 48
6
7   try:
8       with open('sample1\CarParkPos1', 'rb') as f:
9           posList = pickle.load(f)
10  except:
11      posList = []
12
13  #posList = []
14
15
16  def mouseClick(events, x, y, flags, params):
17      if events == cv2.EVENT_LBUTTONDOWN:
18          posList.append((x, y))
19      if events == cv2.EVENT_RBUTTONDOWN:
20          for i, pos in enumerate(posList):
21              x1, y1 = pos
22              if x1 < x < x1 + width and y1 < y < y1 + height:
23                  posList.pop(i)
24      with open('sample1\CarParkPos1', 'wb') as f:
25          pickle.dump(posList,f)
```

- It removes a parking space if the right mouse button is clicked (EVENT_RBUTTONDOWN) and the click is within the bounds of an existing parking space.
- After any modification to posList, it saves the updated list to the file CarParkPos1 using pickle.

```
27  while True:
28      img = cv2.imread('sample1\carParkImg.png')
29      for pos in posList:
30          cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), (255, 0, 255), 3)
31      cv2.namedWindow("Image", cv2.WINDOW_NORMAL)  # Create window with freedom of dimens
32      cv2.resizeWindow("Image", 1000, 1000)              # Resize window to specified dim
33      cv2.imshow("Image", img)
34      key = cv2.waitKey(10) & 0xFF
35      if key == 27:  # 27 is the ASCII code for the 'Esc' key
36          break
37      cv2.setMouseCallback("Image", mouseClick)
38      cv2.waitKey(10)
```

- The main loop of the program. It continuously displays the image (carParkImg.png) with rectangles drawn around the parking spaces based on the positions stored in posList.
- The user can click on the image to add or remove parking spaces.
- Pressing the 'Esc' key (27 in ASCII) breaks out of the loop and terminates the program.
- The mouse callback (cv2.setMouseCallback) is set for the "Image" window to handle mouse clicks.
- Overall, this code allows the user to interactively mark and unmark parking spaces on an image using the OpenCV library and save the positions for future use.

Main1.py

```
main1.py
sample1 > main1.py > ...
1   import cv2
2   import pickle
3   import cvzone
4   import numpy as np
```

- The code imports the necessary libraries: OpenCV (cv2) for image processing, pickle for serializing and deserializing Python objects, cvzone for additional drawing and text functions, and numpy for numerical operations.
- This line initializes a video capture object (cap) that reads frames from the specified video file (carPark.mp4).
- The code loads the previously saved parking space positions from the file (CarParkPos1) using pickle.
- The width and height variables are set to define the dimensions of the rectangular region that will be marked as a parking space.

```
6   # video feed
7   cap = cv2.VideoCapture('sample1\carPark.mp4')
8
9   with open('sample1\CarParkPos1', 'rb') as f:
10      posList = pickle.load(f)
11
12  width, height = 107, 48
```

```
14
15  def checkParkingSpace(imgPro):
16
17      spaceCounter = 0
18      for pos in posList:
19          x, y = pos
20
21          imgCrop = imgPro[y:y + height, x:x + width]
22          # cv2.imshow("imgCropped",imgCrop)
23          ##cv2.imshow(str(x * y), imgCrop)
24          count = cv2.countNonZero(imgCrop)
25          cvzone.putTextRect(img,str(count),(x,y+height-3),scale=1,thickness=2,offset=0,colorR=(0,0,255))
26
27          if count<750:
28              color =(0,255,0)
29              thickness = 4
30              cvzone.putTextRect(img, str(count), (x, y + height - 3), scale=1, thickness=2, offset=0, colorR=(0,255,0))
31              spaceCounter+=1
32          else:
33              color = (0,0,255)
34              thickness = 2
35          cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
36
37      cvzone.putTextRect(img,f'Free: {spaceCounter}/{len(posList)}',(100,50),scale=3,thickness=5,offset=20,colorR=(0,255,0))
```

- This function, checkParkingSpace, processes the provided image (imgPro).
- It checks each parking space's occupancy, draws rectangles around the spaces, and displays the count of occupied pixels.
- It also updates the spaceCounter and shows the total number of free parking spaces on the image.

```
39  while True:
40
41      if cap.get(cv2.CAP_PROP_POS_FRAMES) == cap.get(cv2.CAP_PROP_FRAME_COUNT):
42          cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
43      success, img = cap.read()
44      imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
45      imgBlur = cv2.GaussianBlur(imgGray,(3,3),1)
46      imgThreshold = cv2.adaptiveThreshold(imgBlur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,25,16)
47      imgMedian = cv2.medianBlur(imgThreshold,5)
48      kernel = np.ones((3,3),np.uint8)
49      imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)
50
51
52      checkParkingSpace(imgDilate)
53
```

```
    cv2.imshow("Image", img)
    key = cv2.waitKey(10) & 0xFF
    if key == 27:  # 27 is the ASCII code for the 'Esc' key
        break
    #cv2.imshow("ImageBlur", imgBlur)
    #cv2.imshow("ImageThreshold", imgThreshold)
    #cv2.imshow("ImageMedian", imgMedian)

```

- This is the main loop of the program. It continuously reads frames from the video feed, processes each frame to enhance parking space visibility (imgDilate), and calls the checkParkingSpace function to analyze and display parking space information. The loop terminates if the 'Esc' key is pressed.
- The commented lines (#cv2.imshow...) show additional images (e.g., blurred, thresholded) that can be displayed for debugging purposes. Uncomment these lines if you want to visualize those intermediate steps.

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

Frame Processing and empty parking slot counters

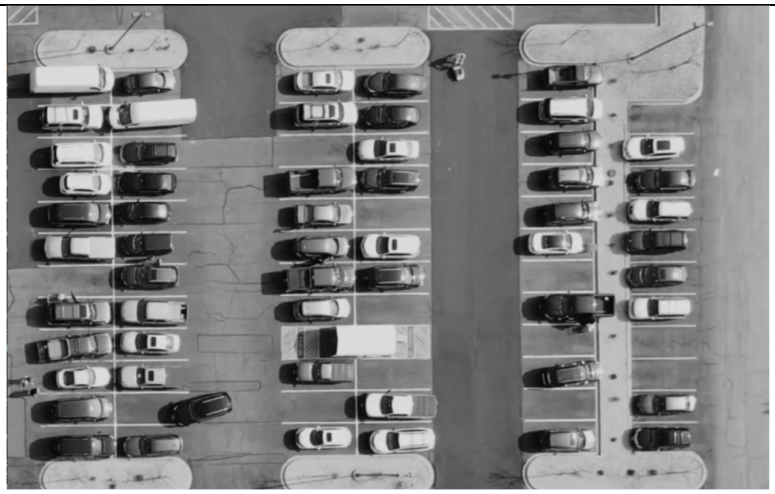
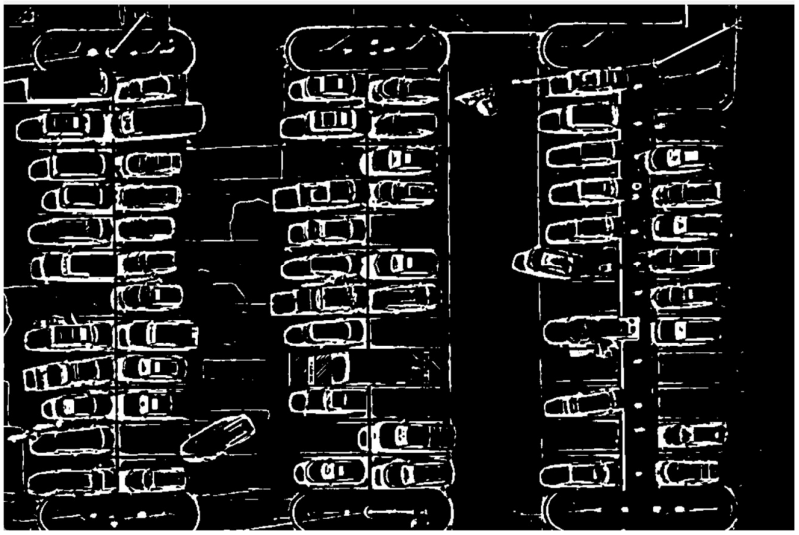
- The captured video has to be read frame by frame. The read() method is used to read the frames.
- Opencv reads the frame in BGR (Blue Green Red).
- Converting BGR frame to gray scale image. And the gray scale image is blurred with
- GaussiamBlur() method from cv2.
- The adaptiveThreshold() method is 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 .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.

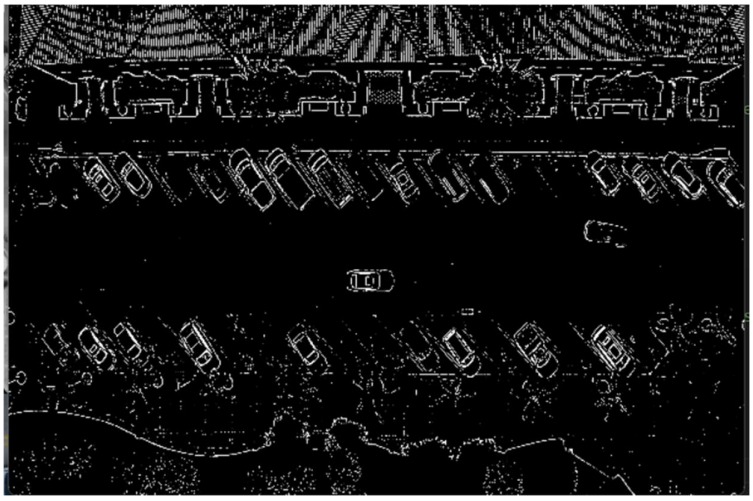
Application building

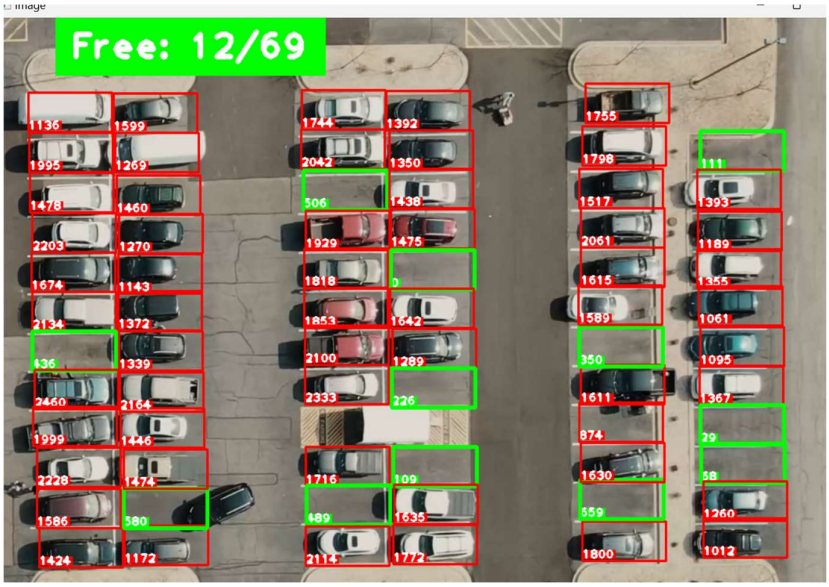
- This Flask application includes user authentication, registration, and various routes to execute external Python scripts.
- the Flask app is created, and a secret key is set for session security. A CSV file is used to store user data, and its existence is checked, creating it if not found. Uses SHA256
- There are routes (/project, /run_main, /run_picker, /project1, /run_main2, /run_picker2) associated with the project pages and subprocess execution. These routes execute external Python scripts using subprocess.run and return the output.
- This Flask application provides user registration, authentication, and interaction with external Python scripts, facilitating user access to various project pages.

8. PERFORMANCE TESTING

8.1. Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	**Model Summary**	**Sample-1** Blur Gray Image	
		Threshold image & Dilate image	
		Bounding boxes	

		Sample-2 Blur Gray Image	
		Threshold image & Dilate image	
		Bounding boxes	

2.	**Accuracy**	**Sample-2** If count < 2170 then it's a free space with green box else occupied space with red box **Sample-1** count<750: then it's a free space with green box else occupied space with red box	 

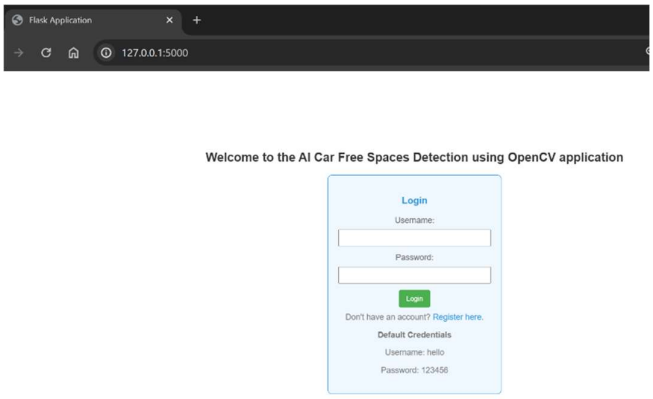
```
if count < 2170:
    color = (0, 255,
    thickness = 5
    spaceCounter += 1
else:
    color = (0, 0, 25
    thickness = 2
```

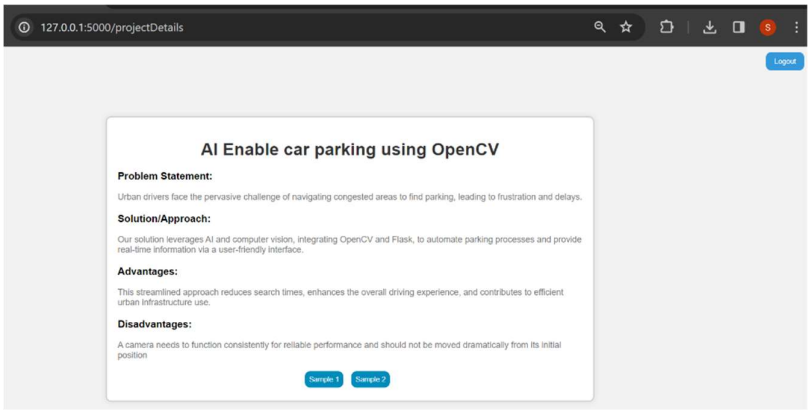
```
if count<750:
    color =(0,255,0)
    thickness = 4
    cvzone.putTextRect(img, str
    spaceCounter+=1
else:
    color = (0,0,255)
    thickness = 2
cv2.rectangle(img, pos, (pos[0]
```

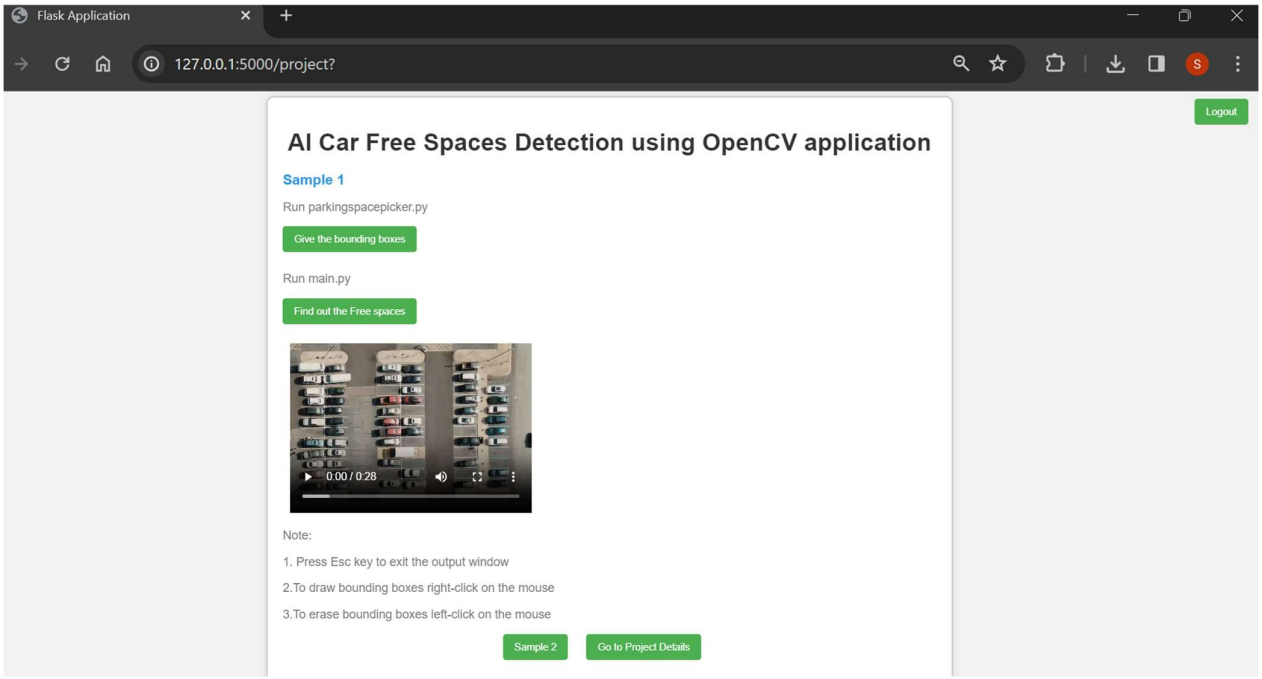
14.RESULTS

14.1. Output Screenshots

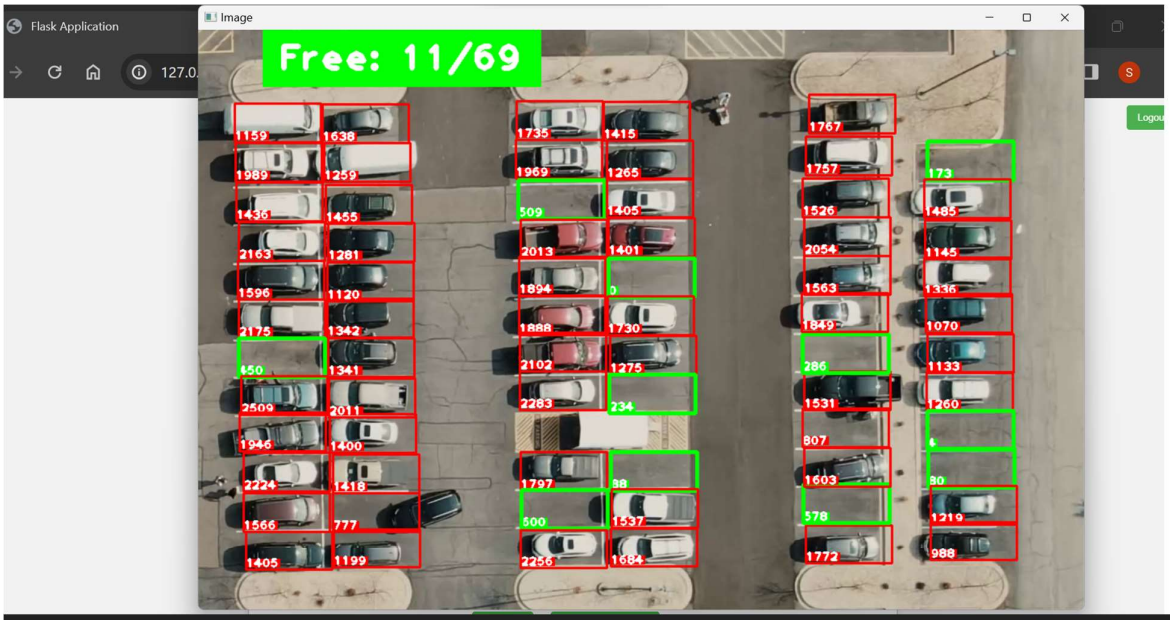
```
ModuleNotFoundError: No module named 'flask'
PS C:\Users\srika\Desktop\AI enable car parking using OpenCV_Project Folder\04-Project Development\flask Application> &
9.exe "c:/Users/srika/Desktop/AI enable car parking using OpenCV_Project Folder/04-Project Development/flask Applicatio
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 750-007-783
```

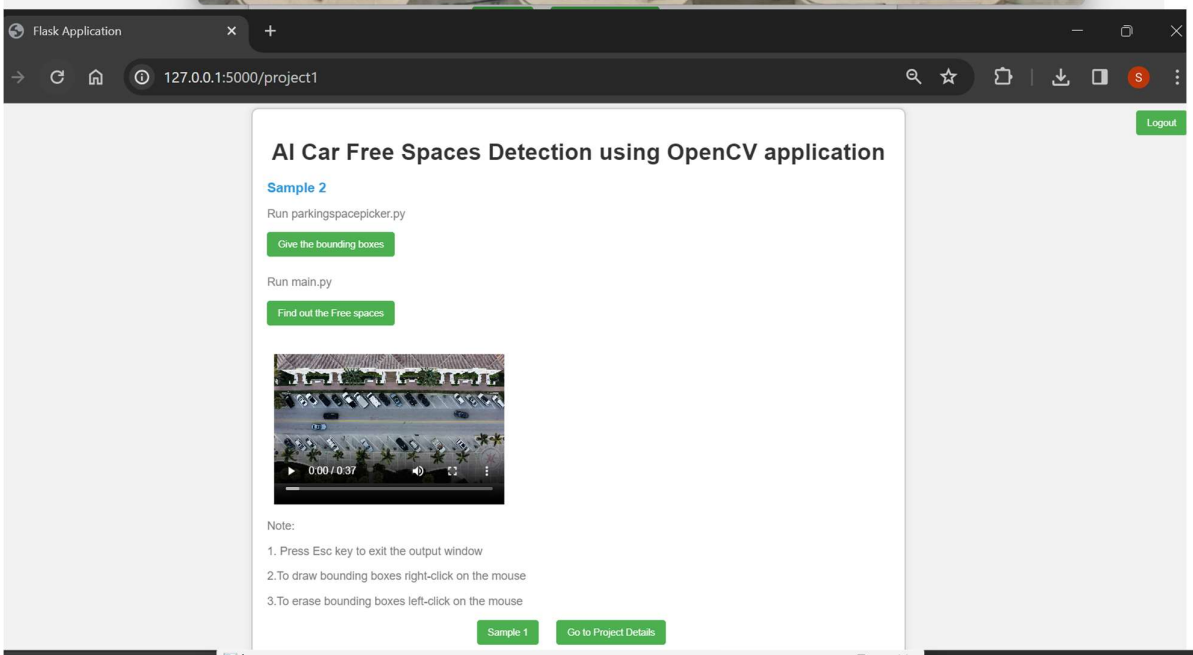

Flask Application
127.0.0.1:5000
Welcome to the AI Car Free Spaces Detection using OpenCV application
Login
Username:
Password:
Login
Don't have an account? Register here.
Default Credentials
Username: hello
Password: 123456

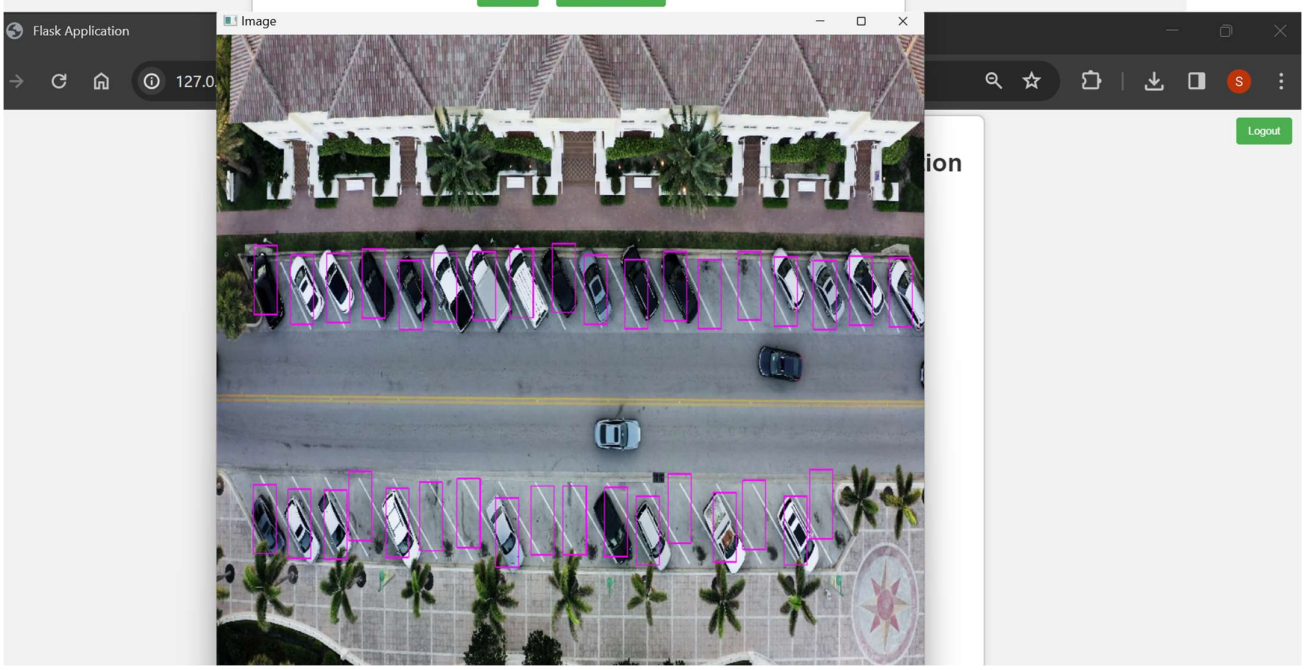

127.0.0.1:5000/projectDetails
Logout
AI Enable car parking using OpenCV
Problem Statement:
Urban drivers face the pervasive challenge of navigating congested areas to find parking, leading to frustration and delays.
Solution/Approach:
Our solution leverages AI and computer vision, integrating OpenCV and Flask, to automate parking processes and provide real-time information via a user-friendly interface.
Advantages:
This streamlined approach reduces search times, enhances the overall driving experience, and contributes to efficient urban infrastructure use.
Disadvantages:
A camera needs to function consistently for reliable performance and should not be moved dramatically from its initial position
Sample 1
Sample 2

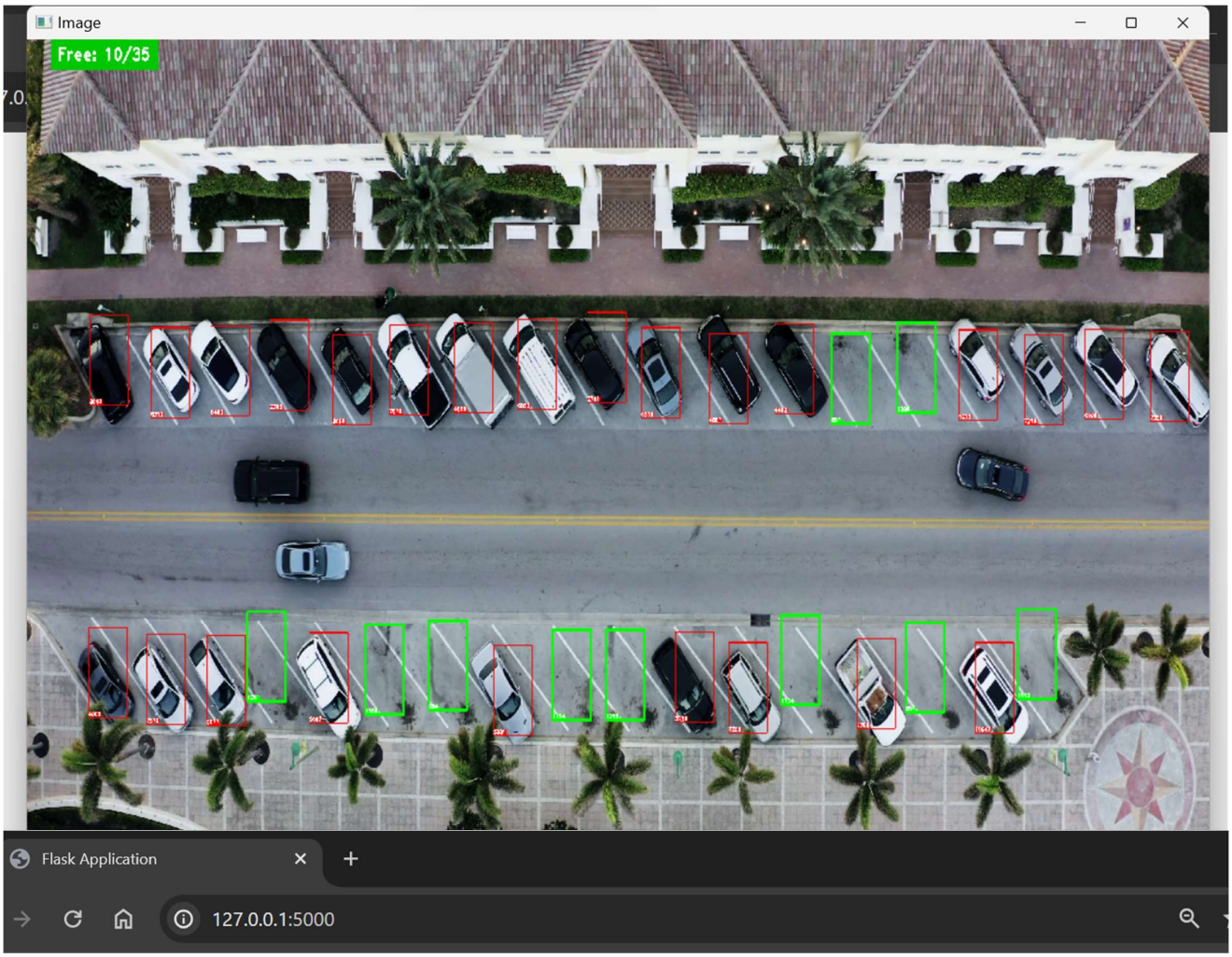

Flask Application
127.0.0.1:5000/project?
Logout
AI Car Free Spaces Detection using OpenCV application
Sample 1
Run parkingspacepicker.py
Give the bounding boxes
Run main.py
Find out the Free spaces
0:00 / 0:28
Note:
1. Press Esc key to exit the output window
2.To draw bounding boxes right-click on the mouse
3.To erase bounding boxes left-click on the mouse
Sample 2
Go to Project Details


Flask Application
Image
Logout


Flask Application
Image
Free: 11/69
Logout


Flask Application
127.0.0.1:5000/project1
Logout
AI Car Free Spaces Detection using OpenCV application
Sample 2
Run parkingspacepicker.py
Give the bounding boxes
Run main.py
Find out the Free spaces
0:00 / 0:37
Note:
1. Press Esc key to exit the output window
2.To draw bounding boxes right-click on the mouse
3.To erase bounding boxes left-click on the mouse
Sample 1
Go to Project Details


Flask Application
Image
Logout



Welcome to the AI Car Free Spaces Detection using OpenCV application

Login

Logout successful

Username:

Password:

Login

Don't have an account? Register here.

Default Credentials

Username: hello

Password: 123456

15.ADVANTAGES & DISADVANTAGES

Advantages:

Efficient Parking Management: The AI-enabled car parking system efficiently manages parking spaces, reducing search times and improving overall traffic flow.

User-Friendly Interface: The integration of Flask provides a user-friendly interface, allowing users to access parking information easily.

Sustainability: The system aligns with sustainability goals by reducing emissions, minimizing search times, and promoting a more efficient use of parking spaces.

Disadvantages:

Initial Setup Complexity: Implementing the AI-enabled parking system may involve initial setup complexity, including camera installation, system configuration, and integration with existing infrastructure.

Dependency on Technology: The system's effectiveness relies on the proper functioning of technology components such as cameras and image processing algorithms. Technical failures could impact system performance.

16.CONCLUSION

In conclusion, the AI-enabled car parking system using OpenCV and Flask offers a comprehensive solution to the challenges of parking in crowded urban areas. By leveraging computer vision and AI technologies, the system streamlines parking processes, reduces frustration, and contributes to sustainable urban development. While there are initial challenges and costs associated with implementation, the long-term benefits in terms of improved traffic management and enhanced user experience make it a valuable solution for modern urban environments.

17.FUTURE SCOPE

The future scope of the project involves potential enhancements and expansions:

Integration with IoT: Incorporating Internet of Things (IoT) devices for real-time data collection and analysis to further optimize parking management.

Machine Learning Improvements: Continuous improvement of the AI algorithms for better accuracy in detecting parking spaces and adapting to changing parking patterns.

Mobile Application: Developing a mobile application for users to access parking information, receive notifications, and navigate to available spaces more conveniently.

Data Analytics: Implementing advanced data analytics to derive insights into parking usage patterns, helping urban planners make informed decisions

18.APPENDIX

https://easychair.org/publications/preprint/hVnT

https://youtu.be/caKnQlCMIYI?si=sUJ9mKg1Y6ZRldou

https://olgarose.github.io/ParkingLot/

https://towardsdatascience.com/find-where-to-park-in-real-time-using-opencv-and-tensorflow-4307a4c3da03

https://www.scribd.com/document/650944475/AI-enable-car-parking-1

Source Code (app.py)

```
from flask import Flask, render_template, request, redirect, url_for, flash
import csv
from werkzeug.security import generate_password_hash, check_password_hash
import subprocess

app = Flask(__name__)
app.secret_key = 'your_secret_key'  # Change this to a secure key in a production environment
CSV_FILE = 'user_data.csv'

try:
    with open(CSV_FILE, 'r') as file:
        pass
except FileNotFoundError:
    with open(CSV_FILE, 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['username', 'password'])
############################################################
@app.route('/', methods=['GET', 'POST'])
def home():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        if authenticate_user(username, password):
            # flash('Login successful', 'success')
            # return redirect(url_for('project'))
            return redirect(url_for('projectDetails'))
        else:
            flash('Login failed. Check your username and password.', 'error')
    return render_template('home.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        if not user_exists(username):
            hashed_password = generate_password_hash(password, method='sha256')
            add_user(username, hashed_password)
            flash('Registration successful. You can now login.', 'success')
            return redirect(url_for('home'))
        else:
            flash('Username already exists. Choose a different username.', 'error')

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

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

@app.route('/run_main', methods=['GET'])
def run_main():
    result = subprocess.run(['python', 'sample1\main1.py'], capture_output=True, text=True)
    return result.stdout

@app.route('/run_picker', methods=['GET'])
def run_picker():
    result = subprocess.run(['python', 'sample1\parkingspacepicker1.py'], capture_output=True, text=True)
    return result.stdout
##################################################
@app.route('/project1')
def project1():
    return render_template('project1.html')

@app.route('/run_main2', methods=['GET'])
def run_main2():
    result = subprocess.run(['python', 'sample2\main2.py'], capture_output=True, text=True)
    return result.stdout

@app.route('/run_picker2', methods=['GET'])
def run_picker2():
    result = subprocess.run(['python', 'sample2\parkingspacepicker2.py'], capture_output=True, text=True)
    return result.stdout
##################################################
def read_csv():
    with open(CSV_FILE, 'r', newline='') as file:
        reader = csv.reader(file)
        return {rows[0]: rows[1] for rows in reader if len(rows) == 2}

def user_exists(username):
    existing_data = read_csv()
    return username in existing_data

def write_csv(data):
    with open(CSV_FILE, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(data)

def update_csv(data):
    with open(CSV_FILE, 'w', newline='') as file:
        writer = csv.writer(file)
        for username, password in data.items():
            writer.writerow([username, password])
@app.route('/submit', methods=['POST'])
def submit():
    username = request.form['username']
    password = generate_password_hash(request.form['password'], method='sha256')
```

```
    existing_data = read_csv()
    if username in existing_data:
        return render_template('register.html', message='Username already exists. Please choose another username.')

    existing_data[username] = password
    update_csv(existing_data)

    return redirect('/')

@app.route('/logout', methods=['POST'])
def logout():
    flash('Logout successful', 'success')
    return redirect(url_for('home'))
def authenticate_user(username, password):
    with open(CSV_FILE, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if row['username'] == username and check_password_hash(row['password'], password):
                return True
    return False

def user_exists(username):
    with open(CSV_FILE, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if row['username'] == username:
                return True
    return False

def add_user(username, password):
    with open(CSV_FILE, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([username, password])

if __name__ == '__main__':
    app.run(debug=True)
    # app.run(host="0.0.0.0",port=5000)
```

GitHub & Project Demo Link

Assignment Link: https://github.com/smartinternz02/SI-GuidedProject-581020-1694582482/tree/main

Project File Link: https://github.com/smartinternz02/SI-GuidedProject-615294-1699448464

Project Video Link: https://drive.google.com/file/d/1NmhLKLMrDYl5ZXTUdVFraw2R31faffed/view?usp=sharing

***** The End *****