AI Enable Car Parking Using OpenCV

Project Report

Team Id.: Team-592467

Team Members

1. Boga. Vivek (Team Leader)
2. M Sakthi Sorna maheswar
3. Bijjula Sai Srujan Reddy
4. Thalir Mahizhnan D

Mentor's Name: Saumya Mohandas

Table of Contents:

1. INTRODUCTION **3**
1.1 Project Overview 3
1.2 Purpose 4
2. LITERATURE SURVEY **4**
2.1 Existing problem 4
2.2 References 4
2.3 Problem Statement Definition 5
3. IDEATION & PROPOSED SOLUTION **5**
3.1 Empathy Map Canvas 5
3.2 Ideation & Brainstorming 6
3.3 Proposed solution 7
4. REQUIREMENT ANALYSIS **8**
4.1 Functional requirement 8
4.2 Non-Functional requirements 8
5. PROJECT DESIGN **9**
5.1 Data Flow Diagrams & User Stories 9
5.2 Solution Architecture 10
6. PROJECT PLANNING & SCHEDULING **11**
6.1 Technical Architecture 11
6.2 Sprint Planning & Estimation 12
6.3 Sprint Delivery Schedule 13
7. CODING & SOLUTIONING **16**
7.1 Feature 1 16
8. PERFORMANCE TESTING **26**
8.1 Performance Metrics 26
9. RESULTS **27**
9.1 Output Screenshots 27
10. ADVANTAGES & DISADVANTAGES **28**
ADVANTAGES: 28
DISADVANTAGES: 30
11. CONCLUSION **31**
12. FUTURE SCOPE **31**
13. APPENDIX **31**
Source Code 31
GitHub & Project Demo Link 31

1. INTRODUCTION

The AI-enabled car parking project using OpenCV aims to revolutionize parking management through the integration of computer vision and machine learning. The goal is to automate parking space detection, optimize allocation, and enhance user experience. With features such as real-time information updates, security monitoring, and scalability, the project strives to make parking more efficient, convenient, and secure, contributing to the advancement of smart transportation solutions.

1.1 Project Overview

The AI-enabled car parking project using OpenCV in urban areas is a pioneering initiative designed to tackle the escalating challenges of parking congestion in densely populated cities. The project's core objective is to deploy advanced technologies, specifically Artificial Intelligence (AI) and computer vision through OpenCV, to create an intelligent parking system that can adeptly navigate the complexities of urban parking scenarios.

At the heart of this initiative is the implementation of automated parking space detection. Leveraging OpenCV's sophisticated computer vision algorithms, the system will continuously analyze real-time video feeds or images from urban parking areas, providing an accurate and up-to-date representation of vacant and occupied parking spaces. This technological advancement forms the foundation for a more efficient and responsive urban parking infrastructure.

To enhance the user experience for urban drivers, the project will introduce an intuitive mobile application or web platform. This user-friendly interface aims to empower drivers with real-time information about available parking spaces, guiding them through the intricate urban landscape and minimizing the time spent searching for suitable parking spots.

Anticipated benefits of the project in an urban context include improved traffic flow, optimized urban space utilization, enhanced mobility, and increased user convenience. By reducing congestion and streamlining the parking process, the project aspires to contribute to the creation of smarter, more sustainable, and user-centric urban environments. In conclusion, the AI-enabled car parking project using OpenCV in urban areas represents a transformative step towards addressing the intricacies of parking management in the dynamic and densely populated urban landscape.

1.2 Purpose

The AI-enabled car parking project, using OpenCV, aims to address parking congestion in densely populated cities by introducing an intelligent system that uses advanced algorithms for real-time analysis of parking spaces. The project provides a user-friendly interface, enhancing the overall experience for urban drivers. It also optimizes parking resource allocation, considering factors like peak traffic times and density. The system also prioritizes security by incorporating surveillance cameras and anomaly detection systems. The project's scalability and integration with existing urban infrastructure ensure compatibility with various parking facilities. The ultimate goal is to contribute to smarter, more sustainable urban environments, improving traffic flow, optimizing space utilization, and enhancing mobility.

2. LITERATURE SURVEY

2.1 Existing problem

The AI-enabled car parking project in urban areas, using OpenCV, faces several challenges, including integrating the system with diverse urban infrastructure, addressing data privacy concerns, and ensuring user accessibility. The reliability of computer vision algorithms in dynamic urban environments is also a concern.

The project also faces security vulnerabilities from surveillance cameras and anomaly detection systems. Cost constraints and public awareness are additional challenges. Regulatory compliance with traffic and parking regulations is crucial, requiring adaptation to legal frameworks and approvals. Addressing these challenges is essential for the successful and sustainable implementation of the AI-enabled car parking project in urban areas. Continuous collaboration, monitoring, and adaptability are key to mitigating these issues and maximizing the project's effectiveness.

2.2 References

- D. Rianto, I. M. Erwin, E. Prakasa and H. Herlan, "Parking slot identification using local binary pattern and support vector machine", *Proc. Int. Conf. Comput. Control Informat. Appl. (ICINA)*, pp. 129-133, Nov. 2018.
- S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi and T. Faughnan, "Smart surveillance as an edge network service: From Harr-cascade SVM to a lightweight CNN", *Proc. IEEE 4th Int. Conf. Collaboration Internet Comput. (CIC)*, pp. 256-265, Oct. 2018
- N. Chen, Y. Chen, E. Blasch, H. Ling, Y. You and X. Ye, "Enabling smart urban surveillance at the edge", *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, pp. 109-119, Nov. 2017
- T. Lin, H. Rivano and F. Le Mouël, "A survey of smart parking solutions", *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3229-3253, Dec. 2017.
- S. Nurullayev and S.-W. Lee, "Generalized parking occupancy analysis based on dilated convolutional neural network", *Sensors*, vol. 19, no. 2, pp. 277, 2019.

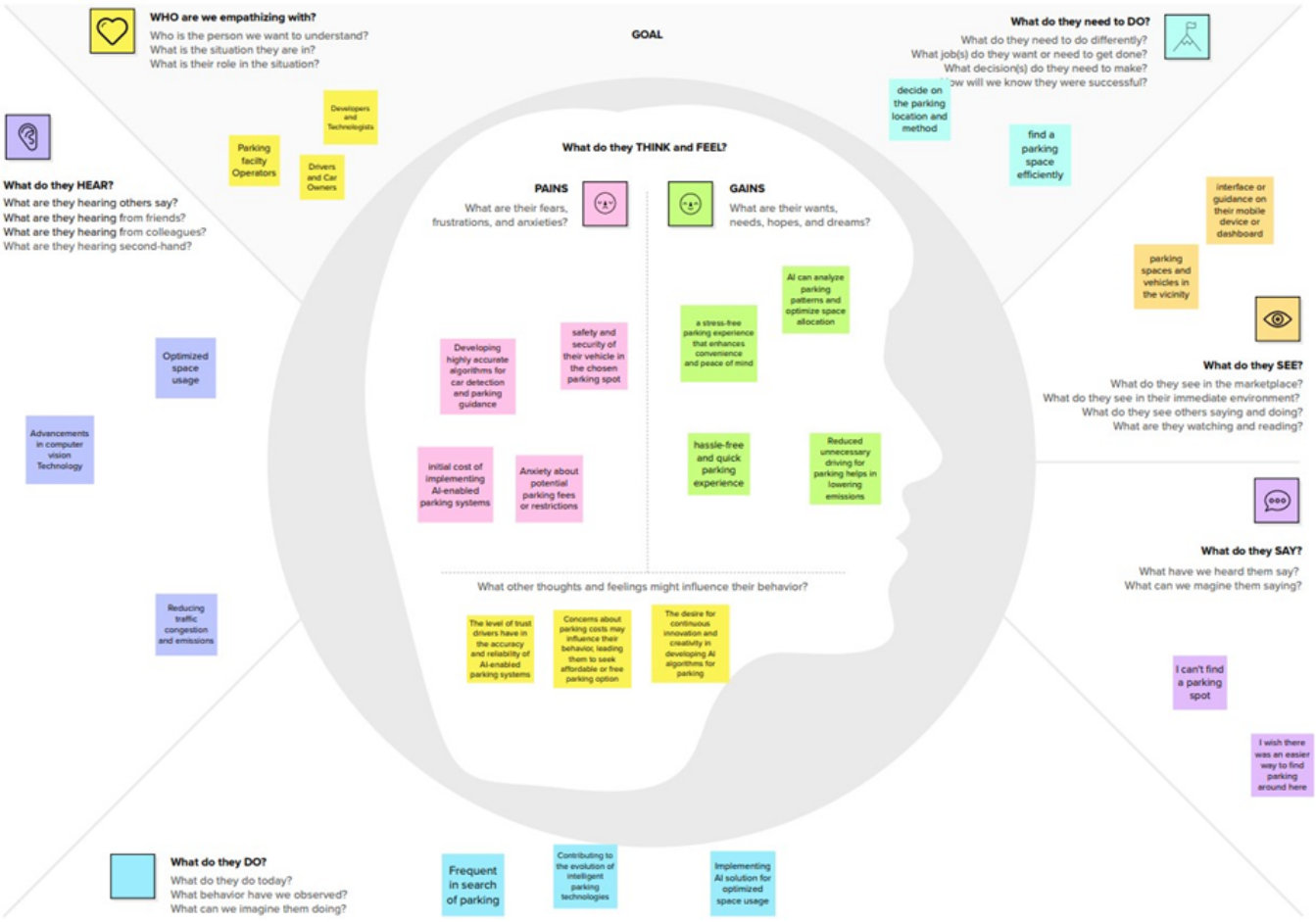
2.3 Problem Statement Definition

In response to the escalating challenges of congestion and inefficiency within urban parking facilities, this project proposes the development and implementation of an AI-enabled car parking system using OpenCV. Urban areas face heightened frustration among drivers due to the lack of intelligence in existing parking systems, leading to wasted time and increased traffic congestion.

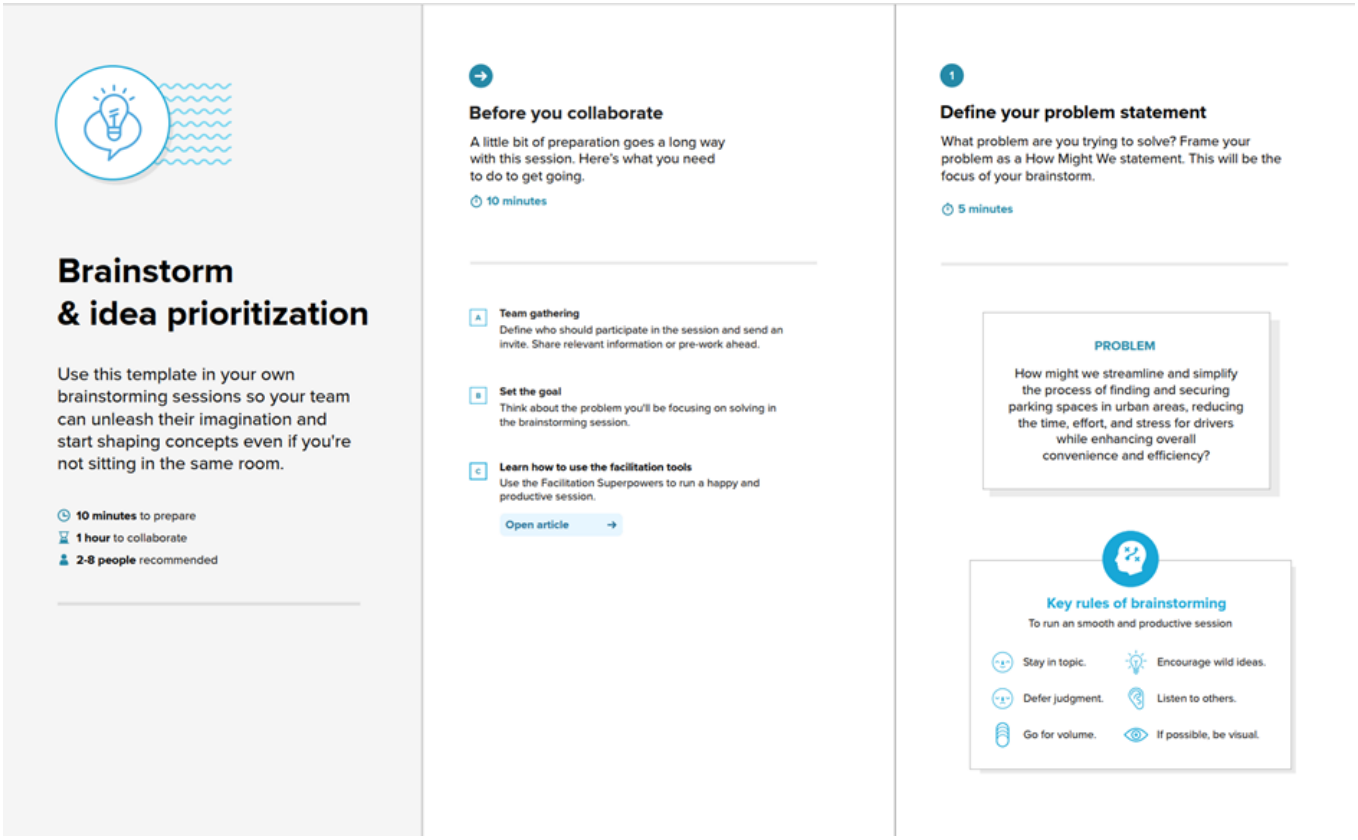
The project aims to improve urban parking by using advanced computer vision algorithms to accurately detect and allocate spaces. It aims to enhance resource utilization, improve traffic flow, and create smarter, user-centric environments. The integration of OpenCV will provide a scalable solution for optimizing parking space allocation and improving urban mobility.

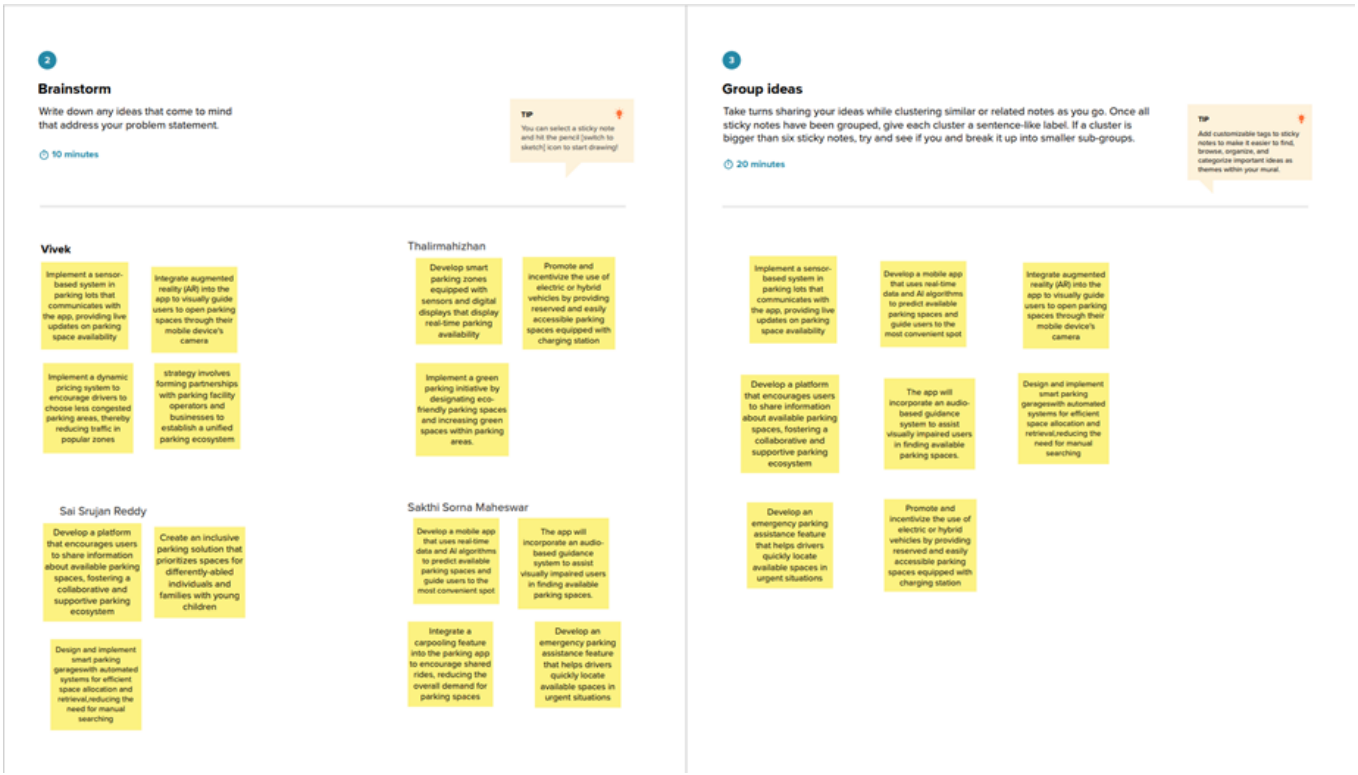
3. IDEATION & PROPOSED SOLUTION

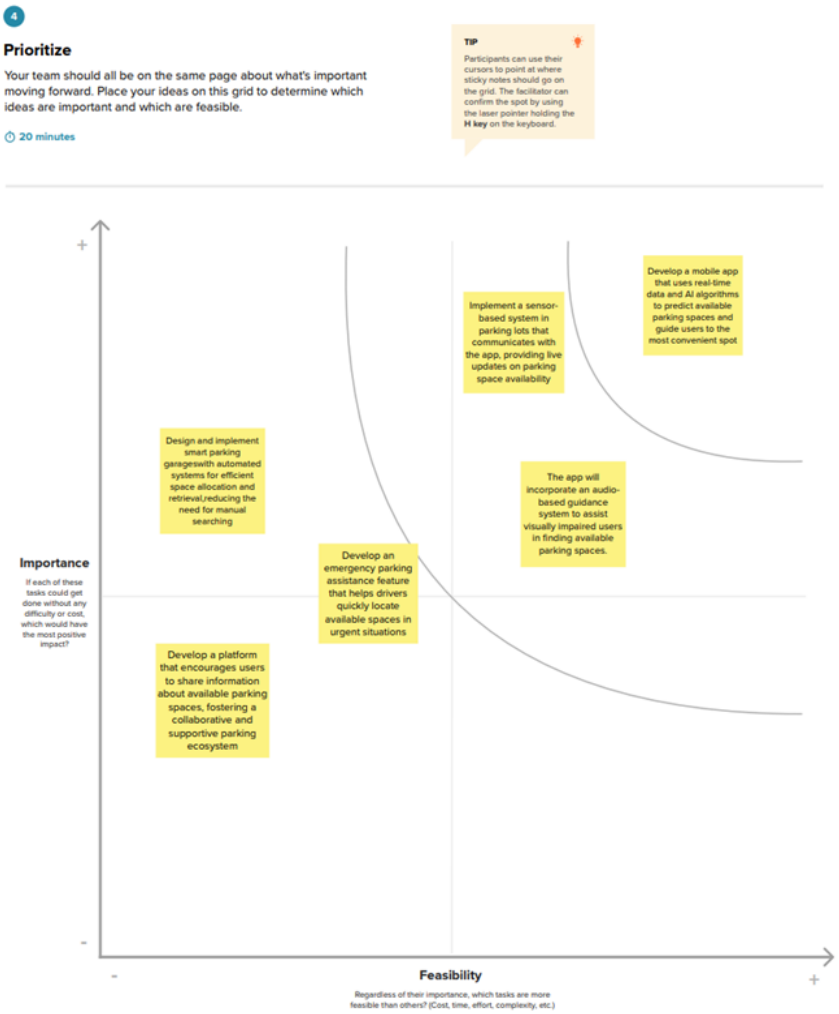
3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming







3.3 Proposed solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Locating an available parking space at the shortest distance from an initial point.
2.	Idea / Solution description	The aim of this project is to distinguish between vacant and occupied parking slots, assigning sequential numbers to each slot in ascending order to identify the nearest unoccupied slot with the minimum distance.
3.	Novelty / Uniqueness	With the method described above, the parking slots are categorized into occupied and unoccupied slots.
4.	Social Impact / Customer Satisfaction	Implementing this method will decrease the time required to park their vehicle, ultimately enhancing customer satisfaction.
5.	Business Model (Revenue Model)	The outcome of this project could be applied in public spaces, enabling them to attain a high level of accuracy.
6.	Scalability of the Solution	The result of this project will prove highly beneficial for the parking management system.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

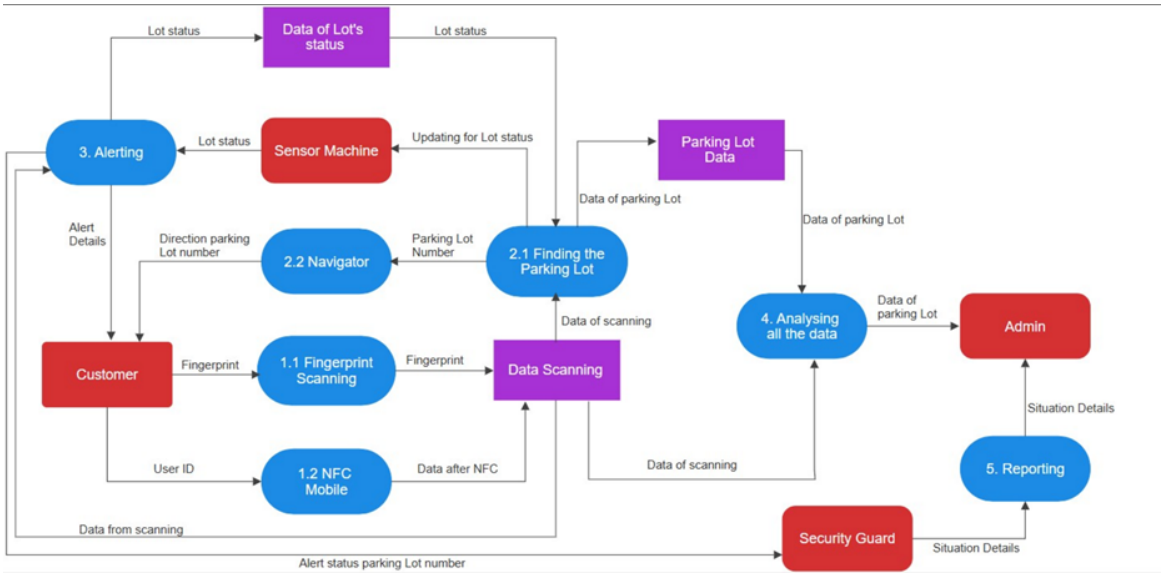
FR No.	Functional Requirement (Epic)	Sub-Requirement (Story/Task)
1	User Application	Upload of the input file
2	Object Detection	The system should be able to detect the presence of a car in a parking spot.
3	Parking Monitoring	The system should be able to monitor the parked cars and detect any illegal activities, such as double parking or parking in a handicap spot.
4	Real-Time Monitoring	The system should provide real-time updates on parking availability and other relevant information to drivers and parking lot staff.
5	User-Friendly Interface	The system should have a user-friendly interface that is easy to use and understand, to ensure a smooth and hassle-free parking experience for drivers.
6	Fault-Tolerance	Implement fault-tolerant mechanisms to handle unexpected system failures or disruptions, ensuring the continuous operation of critical functionalities and minimizing downtime.
7	Data Privacy and Compliance	Ensure compliance with data protection regulations and implement measures to safeguard user privacy, particularly in the collection and processing of real-time data.
8	Reporting and Analytics	Provide reporting and analytics features to analyze parking usage patterns, system performance, and user interactions, supporting continuous improvement and optimization efforts.

4.2 Non-Functional requirements

NFR No.	Non-Functional Requirement	Description
1	Usability	The system should be user-friendly and intuitive, with a simple and easy-to-use interface that is accessible to all users.
2	Security	The system should be designed with robust security features, to ensure the privacy and safety of drivers and their vehicles, and to prevent unauthorized access and data breaches
3	Reliability	The system should be reliable and stable, with high availability and minimal downtime.
4	Performance	The system should be able to process data quickly and accurately, with minimal delay and high efficiency.
5	Availability	The system should be highly available, with minimal downtime and interruption to the parking service
6	Scalability	The system should be able to handle a large number of parking spots and users, and be easily scalable as the demand increases

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

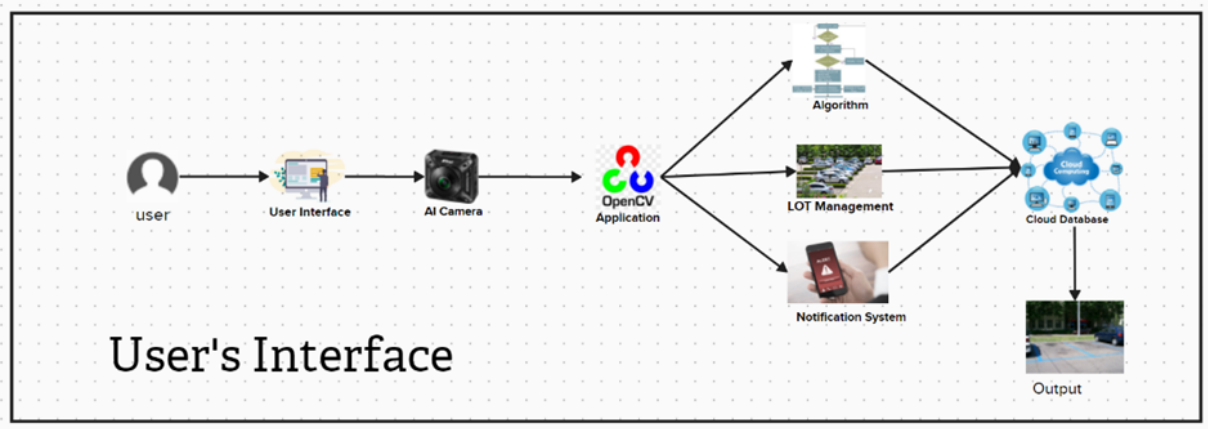


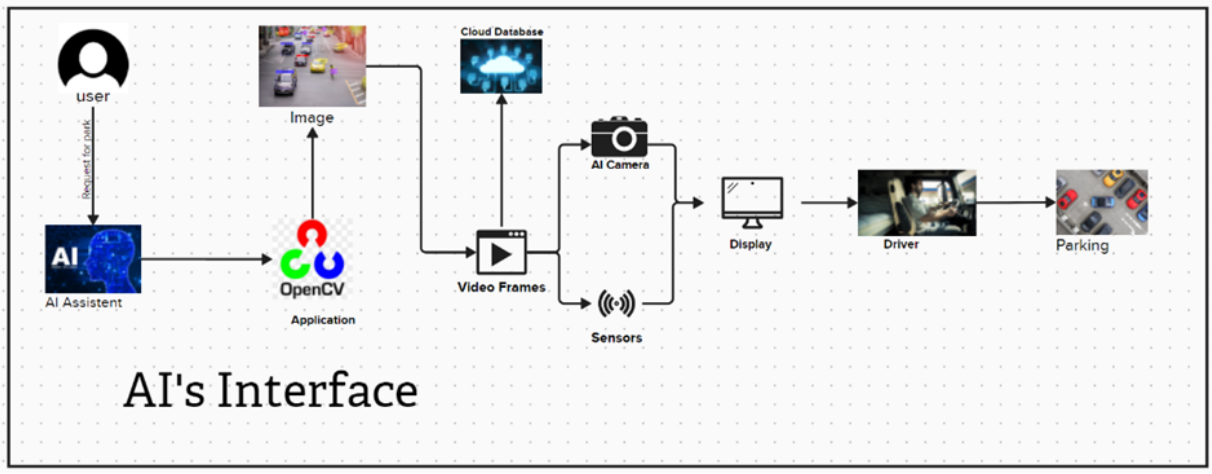
User Type	Functional Requirement (Epic)	User Story Member	User Story/Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register the app with email account	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register to 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

	Profile	USN-7	As a user I can see registration page, login page and Chabot 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 (Web User)	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
Customer Care Executive	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
Administrator	Dashboard	USN-10	As an administrator, I can view how many members requested for what trouble occurs in parking a vehicle	I can check the number of requirement s and monitor the availability	Low	Sprint-4
chatbot	User Interface	USN-11	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

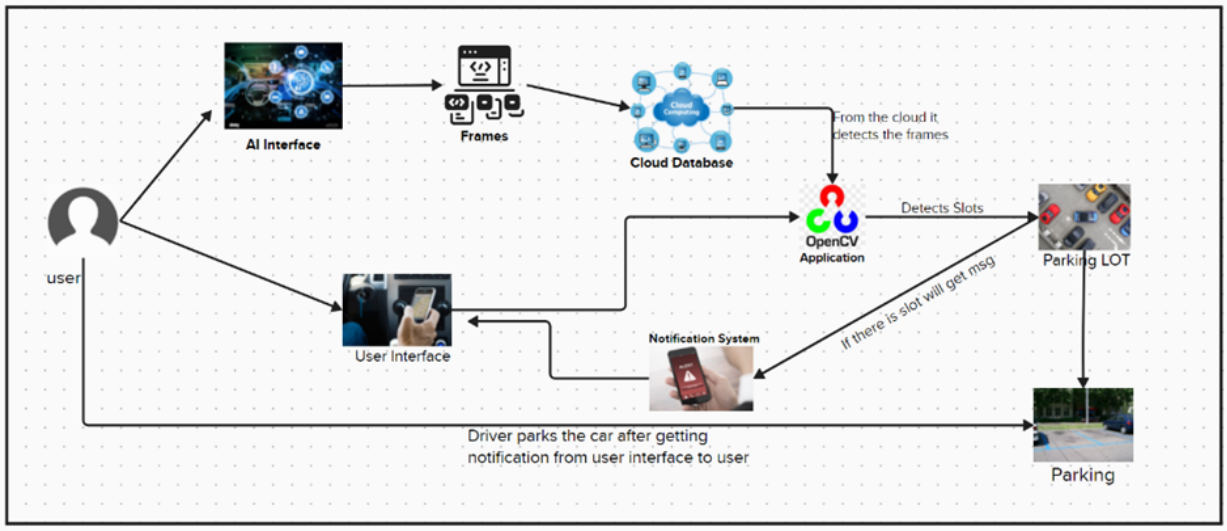
The parking management system uses real-time data sources, cameras, and sensors to capture video footage and provide information on parking space availability, vehicle movements, and environmental conditions. The data processing layer processes raw data to extract meaningful information for parking guidance and decision-making. The AI-based system uses processed data to make decisions, optimize parking allocation, and provide guidance to drivers. The user interface (UI) layer interacts with end-users, providing a user-friendly interface for accessing parking information. The communication layer facilitates data exchange between different system components, including data transmission from sensors to the AI system. The security and privacy layer ensures data security and user privacy. The database management store and manages relevant data for historical analysis, reporting, and system optimization. The system allows seamless integration with external systems like payment gateways, traffic management, and smart city platforms.





6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Table-1: Components & Technologies:

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	OpenCV is an open source platform for providing real time computer vision technology	Open CV
8	External API-1	They make it easy for developers to store manage and deploy container images	Container registry
9	Machine Learning Model	Uses test and trained data images and video to learn the environment	Object recognition models, etc.
10	Infrastructure (Server/Cloud)	Application Development on Local system / cloud	Local, cloud Foundry, python-flask, etc.

Table-2: Application Characteristics:

S. No.	Component	Description	Technology
1	Open-Source Frameworks	The utilization of open-source frameworks to build core components of the system.	Open CV, Tensor Flow, Django, MQTT, SQLite, React
2	Security Implementation	Measures implemented to ensure the security of the application and user data.	Encryption, Secure Storage, User Authentication, API Key, Firewalls, etc.
3	Scalable Architecture	Design considerations and implementations for a scalable and flexible architecture	Microservices Architecture, Load Balancing, Auto-Scaling, Elastic Scaling
4	Availability	Measures taken to ensure continuous system operation and high availability.	Redundancy, Content Delivery networks (CDN), Backup Systems and Data.
5	Performance	Techniques and strategies employed to optimize system performance.	Caching Mechanisms, Database Indexing, Content Compression, Regular Monitoring, Performance Tuning

6.3 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint-1	Project setup and Infrastructure	USN-1	Prepare the necessary tools and frameworks to initiate the AI-enabled Car Parking Project.	3	High	Vivek
Sprint-1	Development environment	USN-2	Collect diverse data essential for training the OpenCV learning model	2	Medium	Srujan
Sprint-2	Data collection	USN-3	Organize and pre-process collected datasets, splitting them into training and validation sets.	2	Medium	Thalir
Sprint-2	Data pre-processing	USN-4	Explore OpenCV architectures, choose the best model for the Car Parking Project.	4	Medium	Thalir
Sprint-3	Model development	USN-5	Train and evaluate the selected OpenCV model with pre-processed datasets.	4	High	Mahesh
Sprint-3	Training	USN-6	Try and implement various augmentation techniques to improve the model's efficiency.	5	High	Mahesh
Sprint-4	Model deployment and Integration	USN-7	Deploy the OpenCV model as a web service. Integrate its API into a user-friendly interface.	3	Medium	Srujan
Sprint-5	Testing and quality assurance.	USN-8	Thoroughly test the model and web interface, tune hyperparameters, and optimize using testing results and user feedback	3	Medium	Vivek

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

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	5	3 Days	28 Oct 2023	30 Oct 2023	20	30 Oct 2023
Sprint-2	6	4 Days	31 Oct 2023	3 Nov 2023	18	4 Nov 2023
Sprint-3	9	1 Week	4 Nov 2023	10 Nov 2023	19	11 Nov 2023
Sprint-4	3	4 Days	11 Nov 2023	14 Nov 2023	18	14 Nov 2023
Sprint-5	3	3 Days	15 Nov 2023	17 Nov 2023	16	18 Nov 2023

Velocity:

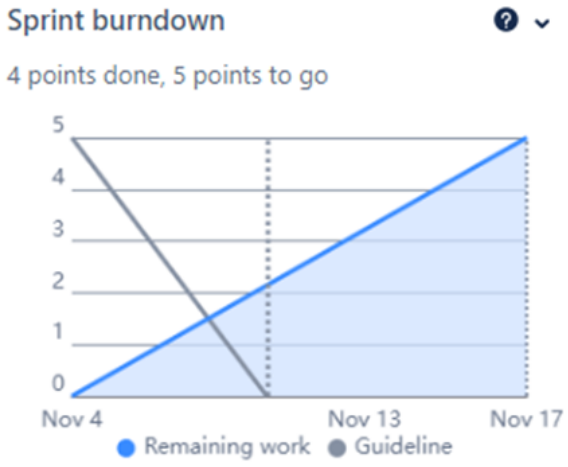
Imagine we have a 29-days sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

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

$$AV = 21/26 = 0.80$$

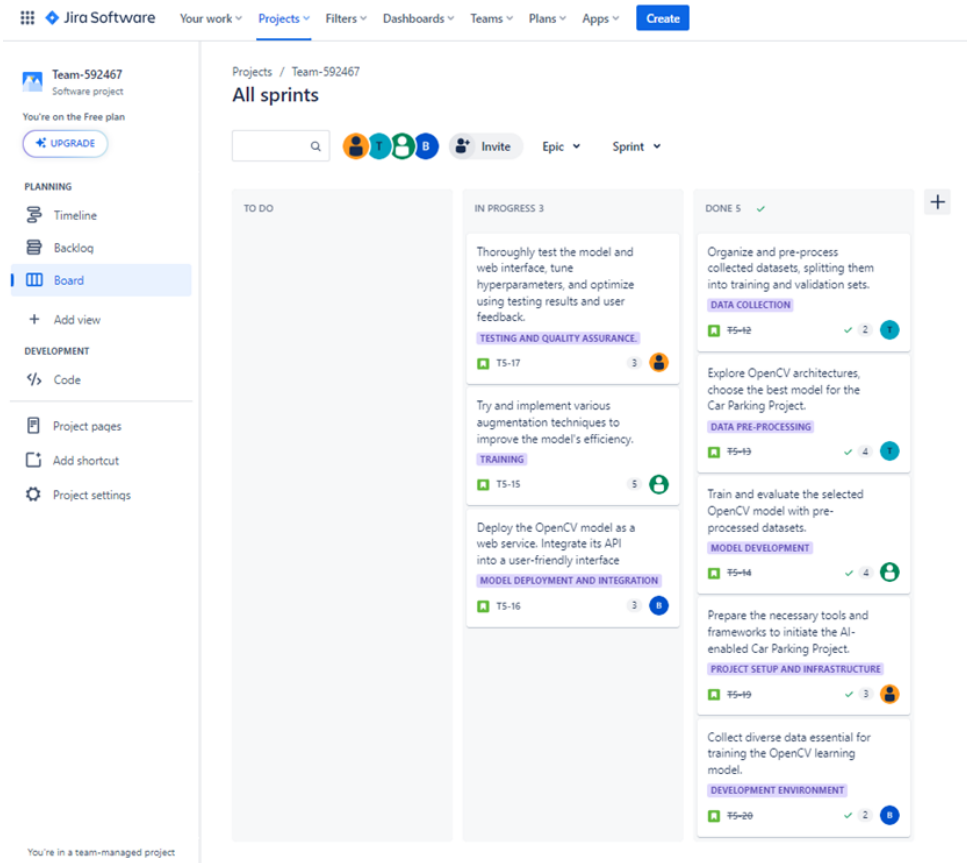
Burndown Chart:

A burndown 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.

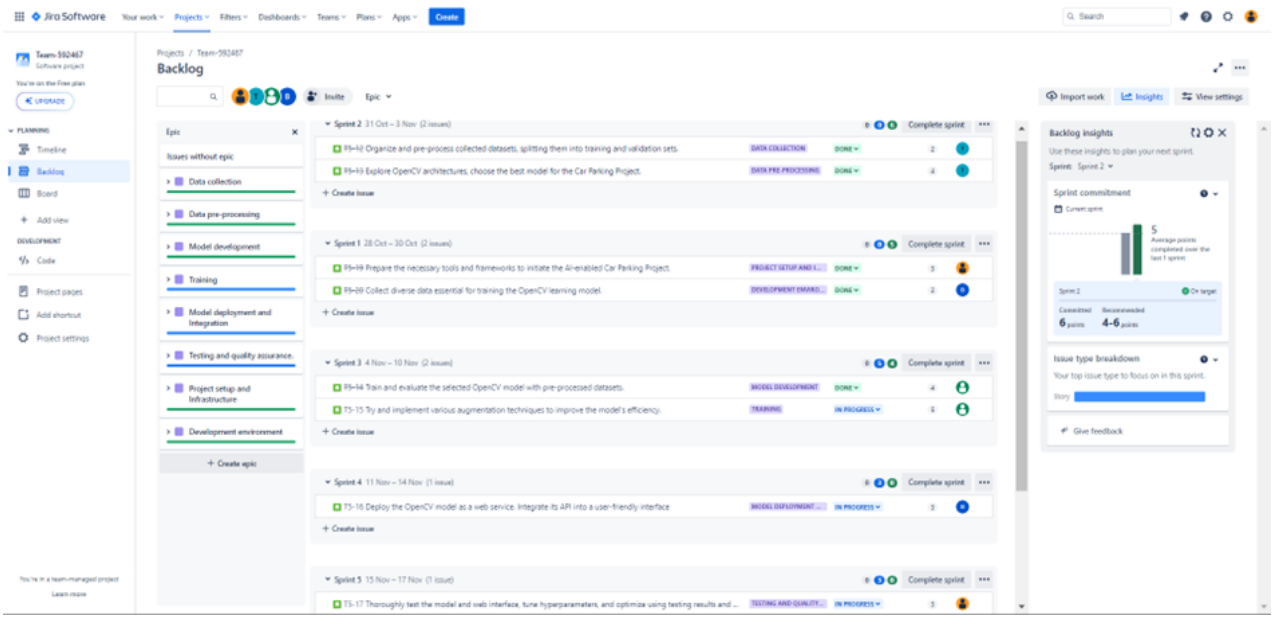


Board section:

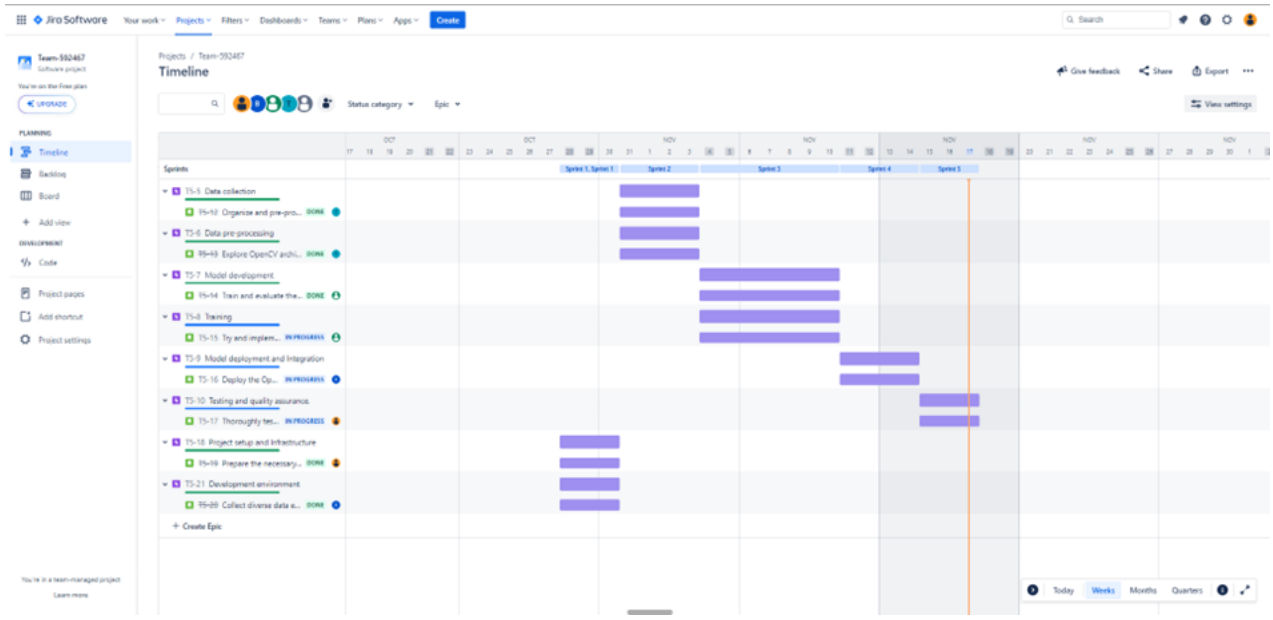
We have completed sprint 1 and 2. So we can see the remaining tasks on board.



Backlog section:



Timeline:



7. CODING & SOLUTIONING

7.1 Feature 1

Pre-requisites:

1. To complete this project you must have the following software versions and packages.

- PyCharm (Download: https://www.jetbrains.com/pycharm/)
- Python 3.7.0 (Download: https://www.python.org/downloads/release/python-370/)
- Numpy
- cvzone

2. To make a responsive python script you must require the following packages.

Flask:

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

If you are using anaconda navigator, follow below steps to download required packages:

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

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

Project objectives:

By the end of this project you will:

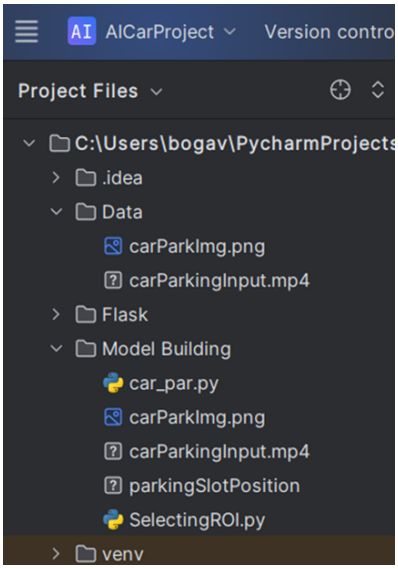
● Know fundamental concepts and techniques of computer vision (openCV). ● Gain a broad understanding of image thresholding.

Project Flow:

- Data Collection
  - Download the 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

Project Structure:

Create a project folder which contains files as shown below:



● The Dataset folder contains the video and image file

● For building a Flask Application we needs HTML pages stored in the **templates** folder,CSS for styling the pages stored in the static folder and a python script **app.py** for server side scripting
● Model Building folder consists of car.py & selectingROI.py python script

Milestone – 1: Data Collection

Activity 1: Download the dataset

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

Milestone – 2: ROI (Region of interest)

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

Activity 1: Create a python file

Create a python file in the directory and name it 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 deserialization 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.

```
width, height = 107, 48

try:
    with open('CarParkPos', '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 an unwanted region. Then we can remove that unwanted ROI by right clicking from the mouse.
● Python object are converted into a stream of bytes and stored in parkingSlotPosition File.

```
def mouseClick(events, x, y, flags, params):
    if events == cv2.EVENT_LBUTTONDOWN:
        posList.append((x, y))
    if events == cv2.EVENT_RBUTTONDOWN:
        for i, pos in enumerate(posList):
            x1, y1 = pos
            if x1 < x < x1 + width and y1 < y < y1 + height:
                posList.pop(i)

    with open('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 to 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, 0, 255), 2)

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

Milestone - 3: Video processing and Object detection

Create a new python file to perform video processing, object detection and counters. **Activity 1: Import the required packages**

Importing the required libraries to the new python file.

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

Activity 2: Reading input and loading ROI file

● The 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.

```
cap = cv2.VideoCapture('carParkingInput.mp4')
width, height = 103, 43
with open('parkingSlotPosition', 'rb') as f:
    posList = pickle.load(f)
```

Activity 3: Trackbar Initialization

```
def empty(a):
    pass


cv2.namedWindow("Vals")
cv2.resizeWindow("Vals", 640, 240)
cv2.createTrackbar("Val1", "Vals", 25, 50, empty)
cv2.createTrackbar("Val2", "Vals", 16, 50, empty)
cv2.createTrackbar("Val3", "Vals", 5, 50, empty)
```

- empty: A placeholder function for the trackbar callback.
- Three trackbars (Val1, Val2, and Val3) are created to adjust parameters interactively.

Activity 4: Checking for parking space

```
def checkSpaces():
    spaces = 0
    for pos in posList:
        x, y = pos
        w, h = width, height

        imgCrop = imgThres[y:y + h, x:x + w]
        count = cv2.countNonZero(imgCrop)

        if count < 900:
            color = (0, 200, 0)
            thic = 5
            spaces += 1

        else:
            color = (0, 0, 200)
            thic = 2

        cv2.rectangle(img, (x, y), (x + w, y + h), color, thic)

        cv2.putText(img, str(cv2.countNonZero(imgCrop)), org: (x, y + h - 6), cv2.FONT_HERSHEY_PLAIN, fontScale: 1,
                    color, thickness: 2)

    cvzone.putTextRect(img, text: f'Free: {spaces}/{len(posList)}', pos: (50, 60), thickness=3, offset=20,
                       colorR=(0, 200, 0))
```

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

Activity 5: 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:

    # Get image frame
    success, img = cap.read()
    if cap.get(cv2.CAP_PROP_POS_FRAMES) == cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, value: 0)
```

Activity 6: Frame Processing and empty parking slot counters

● The captured video has to be read frame by frame. The read() method is used to read the frames.

● Opencv reads the frame in BGR (Blue Green Red).

● Converting BGR frame to grayscale image. And the gray scale image is blurred with GaussiamBlur() method from cv2.

● The adaptiveThreshold() method is used to apply a threshold to the blurred image. And again blur is applied to the image. Click here to understand about adaptiveThreshold().

● The dilate() method is used to compute the minimum pixel value by overlapping the kernel over the input image. The blurred image after thresholding and kernel are passed as the parameters.

● The checkParkingSpace function is called with a dilated image. As per activity 3 we will get the count of empty parking slots.

● Display the frames in the form of video. The frame will wait for 10 seconds and it'll go to the next frame.

```
while True:

    # Get image frame
    success, img = cap.read()
    if cap.get(cv2.CAP_PROP_POS_FRAMES) == cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, value: 0)
    # img = cv2.imread('img.png')
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, ksize: (3, 3), sigmaX: 1)
    # ret, imgThres = cv2.threshold(imgBlur, 150, 255, cv2.THRESH_BINARY)

    val1 = cv2.getTrackbarPos( trackbarname: "Val1", winname: "Vals")
    val2 = cv2.getTrackbarPos( trackbarname: "Val2", winname: "Vals")
    val3 = cv2.getTrackbarPos( trackbarname: "Val3", winname: "Vals")
    if val1 % 2 == 0: val1 += 1
    if val3 % 2 == 0: val3 += 1
    imgThres = cv2.adaptiveThreshold(imgBlur, maxValue: 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                     cv2.THRESH_BINARY_INV, val1, val2)
    imgThres = cv2.medianBlur(imgThres, val3)
    kernel = np.ones( shape: (3, 3), np.uint8)
    imgThres = cv2.dilate(imgThres, kernel, iterations=1)

    checkSpaces()
    # Display Output

    cv2.imshow( winname: "Image", img)
    # cv2.imshow("ImageGray", imgThres)
    # cv2.imshow("ImageBlur", imgBlur)
    key = cv2.waitKey(1)
    if key == ord('r'):
        pass
```

Milestone - 4: Application building

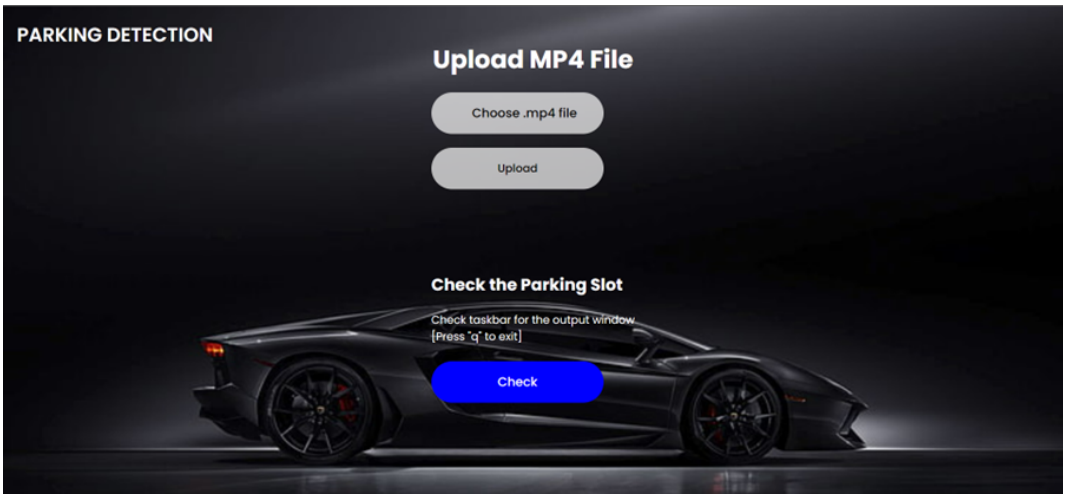
In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the users where he/she has to navigate to the detect button. Then the video will be showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script

Activity1: Building Html Pages:

For this project we have created HTML files and saved them in the templates folder. Let's see how those html pages looks like:



Activity 2: Build Python code:

- Import the libraries

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

app = Flask(__name__)
app.secret_key = 'a'


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

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

● Here we will be using the declared constructor to route to the HTML page that we have created earlier. In the above example, the '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the HTML page will be rendered.
● Create a decorator to route to '/predict_live'. Go to the milestone 3 and copy the entire code and paste it inside the function liv_pred.
● Main Function: Used to run the current module.

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

Activity 3: Run the application

● Open the pycharm from the start menu

● Navigate to the folder where your python script is.

● Now type the "python app.py" command

● Navigate to the localhost where you can view your web page.

● Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
 * 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: 608-656-177
```

8. PERFORMANCE TESTING

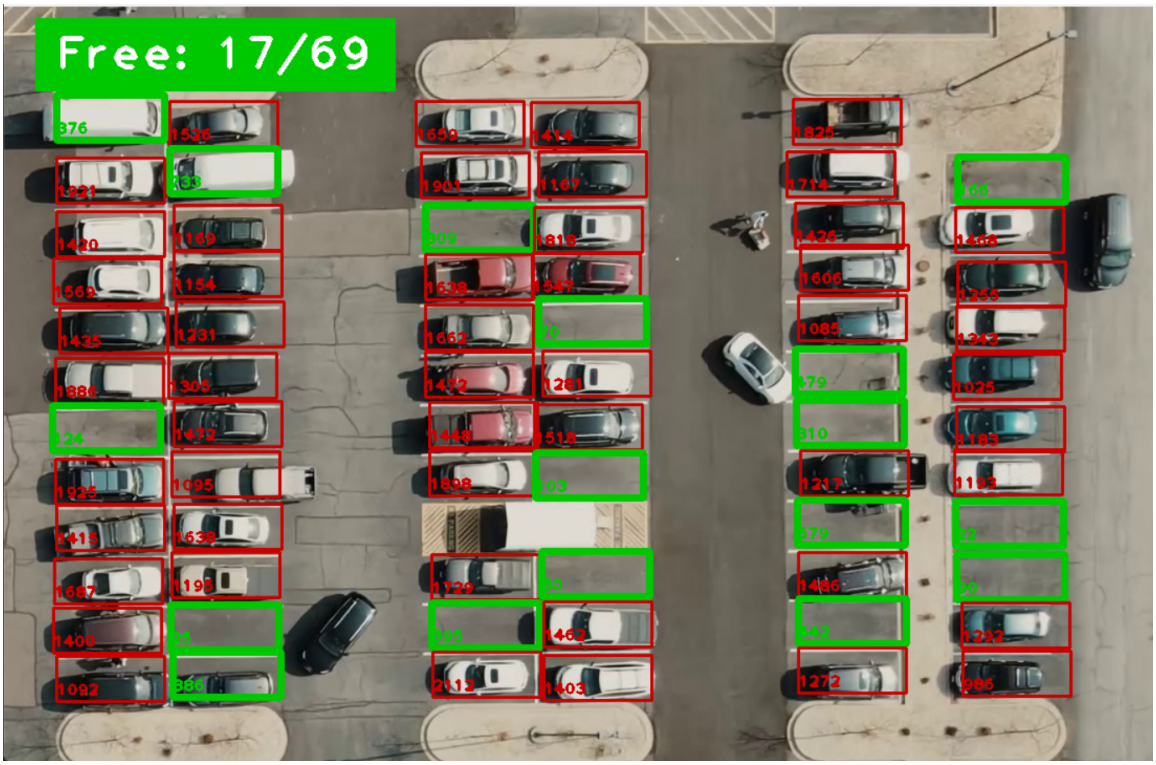
8.1 Performance Metrics

Project team shall fill the following information in the model performance testing template.

S. No.	Parameter	Values	Screenshot
1.	Model Summary	-	 
2.	Accuracy	Training Accuracy – 97% Validation Accuracy -94%	

9. RESULTS

9.1 Output Screenshots



10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

Recent increments in car ownership and urbanization, coupled with poor city planning, have contributed greatly to the rising parking problems across the US[6]. Fortunately, the adoption of AI in smart parking has completely revolutionized how we look for parking spaces. Here are a few advantages of smart parking for drivers and business owners.

Less fuel consumption

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

A reduction of search traffic on the streets

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

Reduced parking stress

Looking for a good parking spot in a congested part of the city is simply overwhelming. You might find yourself driving across the same street several times, only to end up parking far away from your destination.

AI-based smart parking solutions incorporate smart parking technologies with IoT devices so drivers can find parking spots easily using their smartphones and computers. This way, they can see all parking spots available in the area they plan to travel to long before they get there. Some parking lots even allow you to reserve a space. This means you don't have to drive around looking for a space to park.

Benefits of AI-based smart parking for businesses:

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

The result is that some drivers opt to drive longer distances for safer and more accessible parking lots, which means less business for the better-situated but poorly managed parking lots. Fortunately, by incorporating smart parking solutions in their management process, parking lot managers can improve their lots' efficiency, boost customer satisfaction and ultimately boost profits. Here are a few other benefits parking lot businesses stand to gain from leveraging smart parking solutions.

Improved parking experience:

AI-based smart parking solutions leverage collected data to provide specialized services that ultimately lead to stress-free parking experiences. Besides letting nearby drivers know if there are available spots in the parking lot, managers can also install digital signs that receive real-time data from parking lot management software to direct drivers to their parking spots. This eliminates frustration among drivers trying to find a spot within the parking lot, thus improving customer satisfaction.

Also, the mere fact that drivers don't have to drive around the parking lot looking for a free space means fewer emissions, which could improve the air quality in indoor parking lots.

Pinpoint inefficiencies in parking lot management:

Managing a parking lot takes a lot of work. For instance, parking lot managers often have to check the parking duration of certain vehicles or deal with irregularly and illegally parked vehicles. By leveraging real-time information from connected on-site devices, they can accurately determine the parking duration of all vehicles.

This comes in handy, especially in parking lots offering short-time parking services like supermarkets. With this data, the managers can monitor excessive parking durations, which in most cases imply unauthorized use of their parking areas. They can also monitor spaces that stay empty for extended periods, which could imply an issue with the spot in question.

Optimized usage of the facility:

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

DISADVANTAGES:

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

High cost of installation

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

Regular maintenance requirements

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

11. CONCLUSION

In conclusion, the objectives of this project have been achieved. The hassle in searching for available parking slots has been completely eliminated. The designed system could be applied everywhere due to its ease of usage and effectiveness. It facilitates the problems of urban livability , transportation mobility and environment sustainability. 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 systems 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 a commercial parking system the device can be interfaced with a module which can sense the arrival of an 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

Source Code

https://drive.google.com/drive/folders/1soiy9b3qM05T6RdFHZwJl8XgGPrwGIkx?usp=sharing

GitHub & Project Demo Link

GitHub
https://github.com/smartinternz02/SI-GuidedProject-613965-1699092622
Project Demo Link
https://drive.google.com/file/d/1IjjVkY_LRETgIKd_JTu4pjNjbr86wAZL/view?usp=sharing