SMART PARKING

- **Parking has always been an issue in metropolitan cities. Let it be a shop or a public place, parking wastes time, fuel and sometimes breaks sweat to get it done.**
- **Drivers tend to circle around the place searching for parking spots. This increases traffic congestions, wastes fuel and increases pollution. It escalates to a risky scale during festival times.**
- **Unable to find proper parking spots, some vehicles are parked in narrow spaces causing traffic jams.**

Designed and Developed by

K.Ponvishnu(911921106003)

P.T.R college of engineering and technology

Contents:

Research Objectives:

- 🕐 **Reduces the required manpower to maintain a parking system.**
- 🕐 **Senses when a car arrives and scans the license number.**
- 🕐 **Checks for available slots and finds the best optimal path the parking slot.**
- 🕐 **Shows directions to the available parking slot.**
- 🕐 **Updates the database with the car license number, entry time and exit time.**

Smartparking mobile app uses:

- **The project proposes to install low cost camera modules in multiple parking lots across the**

city, which streams live image to the corresponding remote server

- **The remote server processes the data from the camera module and decides on the number of vacant parking spaces available in the parking lot**
- **The remote server updates the number of vacant parking slots and number of filled parking slots in a cloud database**
- **The number of vacant parking slots and their location is displayed in a web application accessible to general public and free to use.**
- **The database is updated continuously , ensuring a pristine user experience**

SEGMENTS INVOLVED:

- **Decentralized server (Raspberry Pi) for image processing, computation and network management.**
- **ESP32-CAM hardware setup for wireless image transmission and reception by server.**
- **Object detection and updation of database**
- **Cloud Deployed and completely scalable Website and Cloud Database management.**

TECHNICAL STACK:

- ### Processing server

  Raspberry Pi 4B

  Raspbian 32-bit OS

  SSH access - PuTTy, FileZilla

  Auto run on boot-up

  TCP Sockets

  PIL Library

- ### ESP32-CAM setup

  Arduino IDE

  ESP32 board

  socket library

  TCP Protocol

- ### Website

Node.js(Express)

HTML5/CSS3

MongoDB

Google Maps API services

- **Object detection and updation**

Python

OpenCV library

pyMongo library

Numpy library

OS module

TECHNICAL EXPLANATION:

The Website:

- **The website is a scalable, cloud deployed, responsive web application accessible to general public and free to use.**
- **Depending on the functionality desired, the user can either search for parking lots near their current location(Mode 1) or a desired destination(Mode 2).**
- **The application uses geolocation technology to find the device location and prints out an interactive map with the 10 closest parking lots and the availability.**
- **The user can click on the parking slot in the map which links to google maps for directions to that parking lot from the user location(in Mode 1) or from the destination(in Mode 2)**
- **It uses Node Js as the back end environment and Express as the server technology.**

ESP32 CAM SETUP:

- **ESP32 CAM is used to take pictures in regular intervals and transmit it to a remote server which is in the same WiFi network.**
- **Camera pins are defined , camera settings are configured and is initialised.**
- **ESP32 cam connects to LAN.**
- **ESP32 cam establishes connection with remote server.**
- **A frame is captured from the camera and stored in a camera_fb_t pointer , which holds the pixel data , height , width of the image.**

- **Pixel data is transmitted to the server through TCP protocol.**
- **Since the size of pixel data exceeds the size limit of single TCP socket, pixel data is broken up into chunks and sent individually.**
- **The process repeats for every two seconds approximately**.

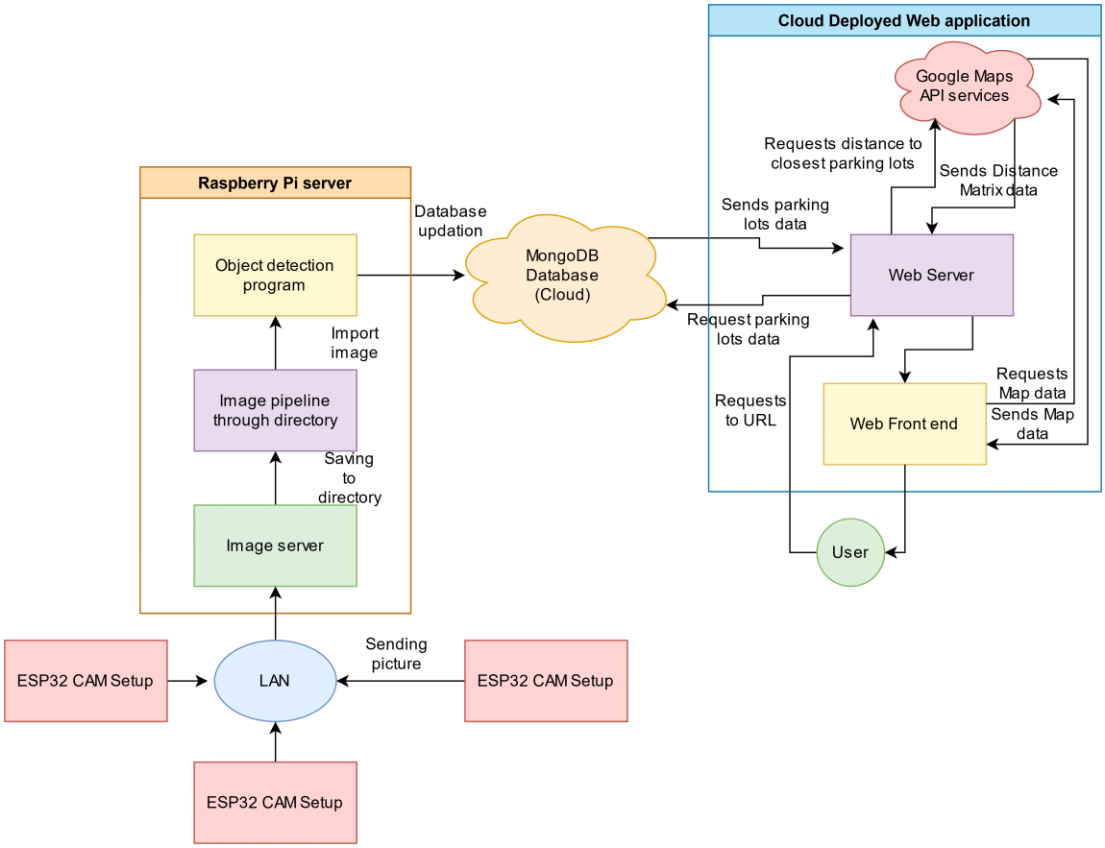
The Server:

- **Raspberry Pi 4B is used as local server for image reception, processing, slot computation and updation to cloud.**
- **PuTTy is used to connect wirelessly to the Pi over SSH and FileZilla was used to transfer the file over SFTP.**
- **The ESP32 image reception and object detection python scripts run on boot-up. This was implemented by modifying the .bashrc script to execute the python scripts on boot-up or when a terminal is launched.**
- **The ESP32 script receives pixel data of the image as a chunk of bytes. It is stored in a byte array and is converted into a JPEG image using Pillow library.**
- **The script then stores each image under proper naming convention (ESP_XX_CHN_X_time) and stores in the respective directory**
- **The Pi is connected to the same local network as the ESP32 CAMs through WiFi. internet is also enabled to perform updation to the cloud.**

Object detection and computation:

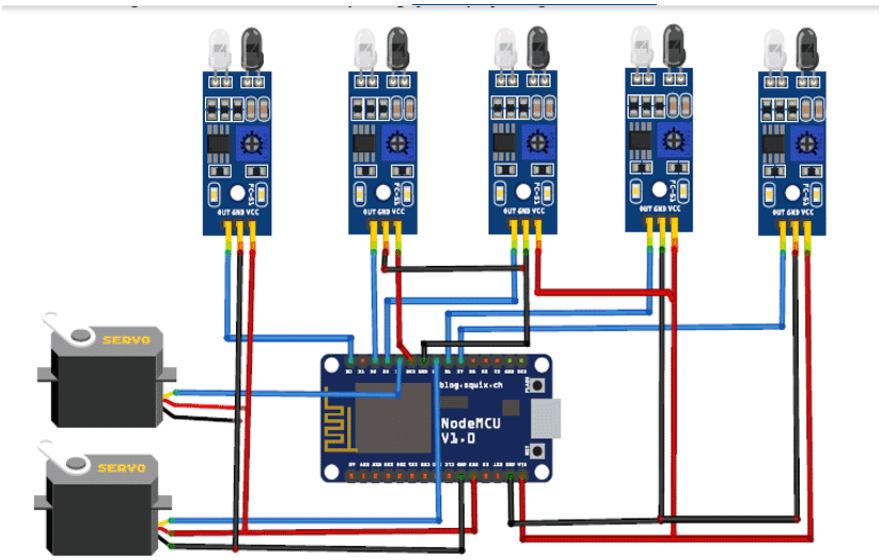
- **It processes the Images obtained from parking lots one after the other and identifies the bounding boxes of the objects (cars and bikes) utilizing a pretrained model(MobileNet SSD model) in OpenCV DNN module.**
- **Based on Intersection Over Union calculation between the predicted bounding boxes and manually drawn bounding boxes representing parking areas Occupancy of respective parking lots gets updated in the server.**
- **After processing, the image gets deleted automatically to avoid reprocessing.**

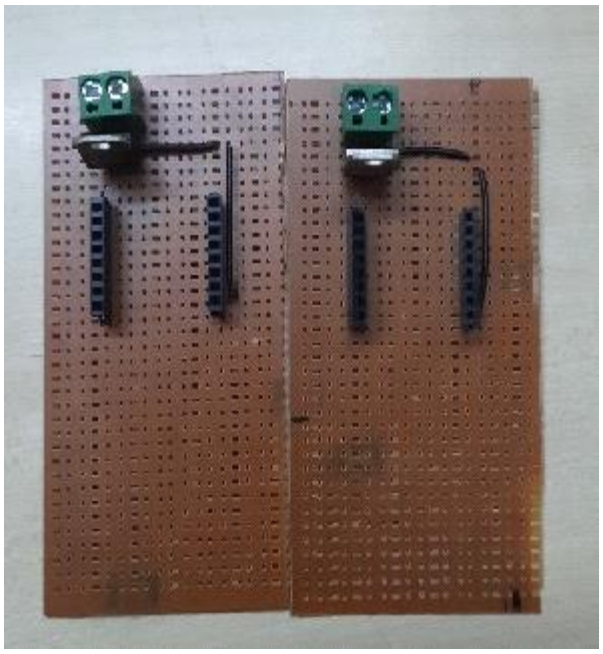
Flowchart of system:



Gallery:

ESP32 module:

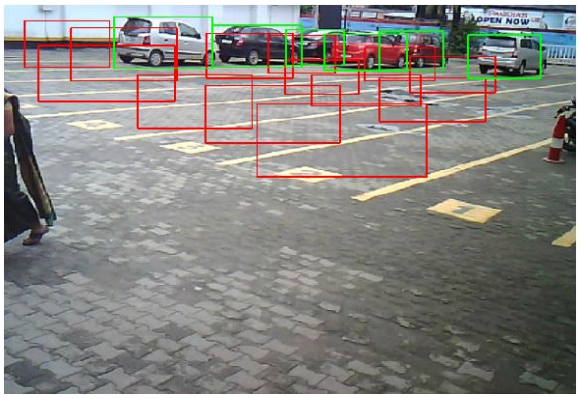


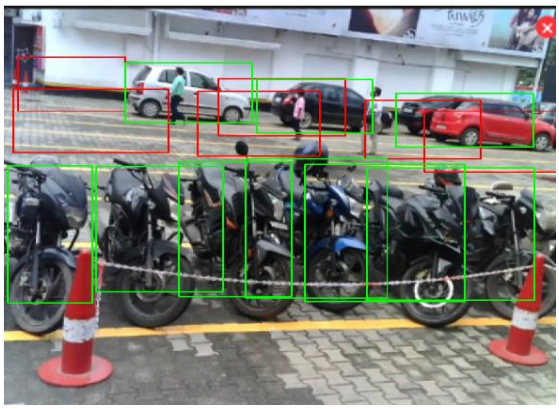


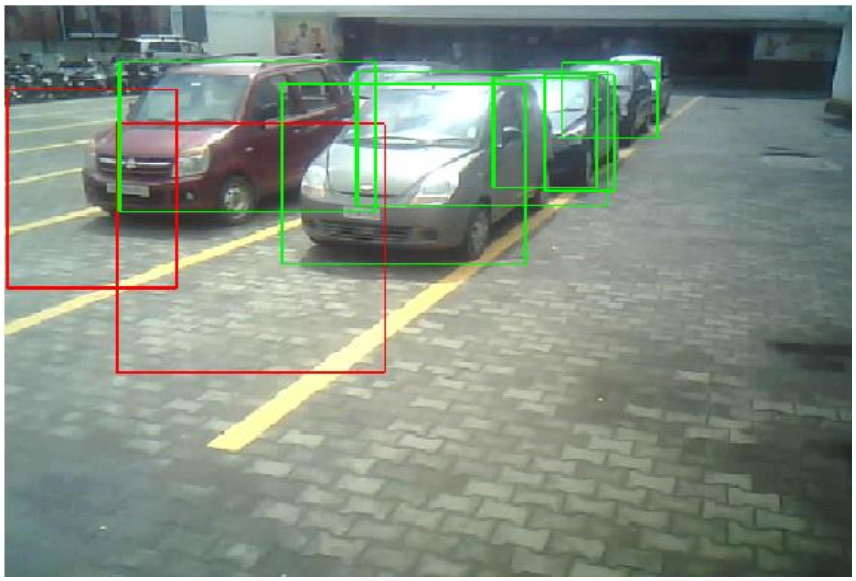
Programming:

```cpp
#include <Ultrasonic.h>

Ultrasonic sensor1(GPIO_TRIGGER1, GPIO_ECHO1);
Ultrasonic sensor2(GPIO_TRIGGER2, GPIO_ECHO2);
// Add more sensors if needed

void setup() {
  Serial.begin(115200);
}

void loop() {
  long distance1 = sensor1.read();
  long distance2 = sensor2.read();
  // Read distances from more sensors if needed

  // Process distance data and manage parking spaces here

  delay(1000); // Delay for better readability
}
```

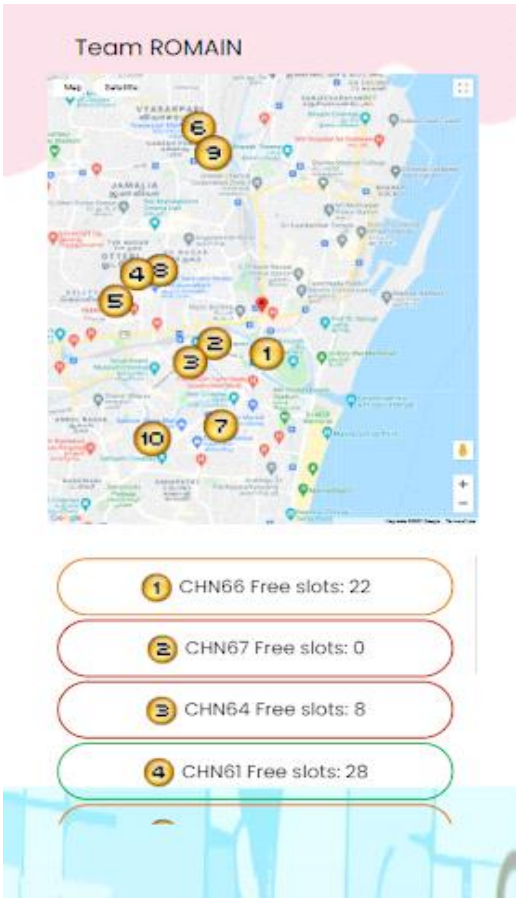
Object detection:

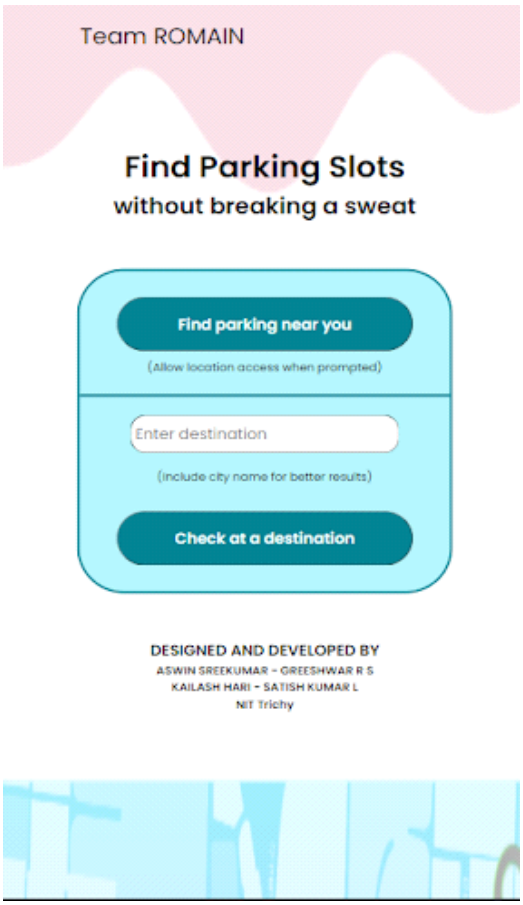


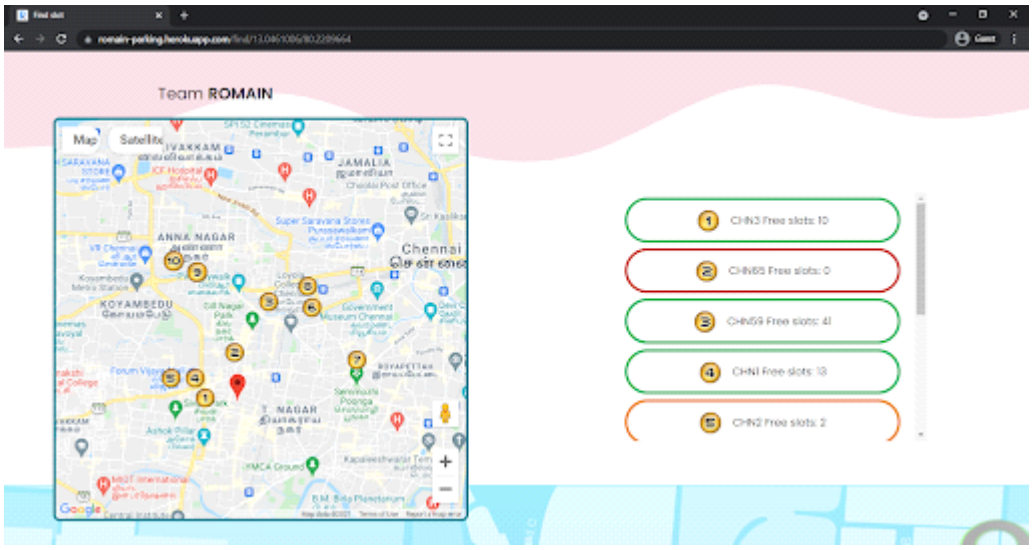


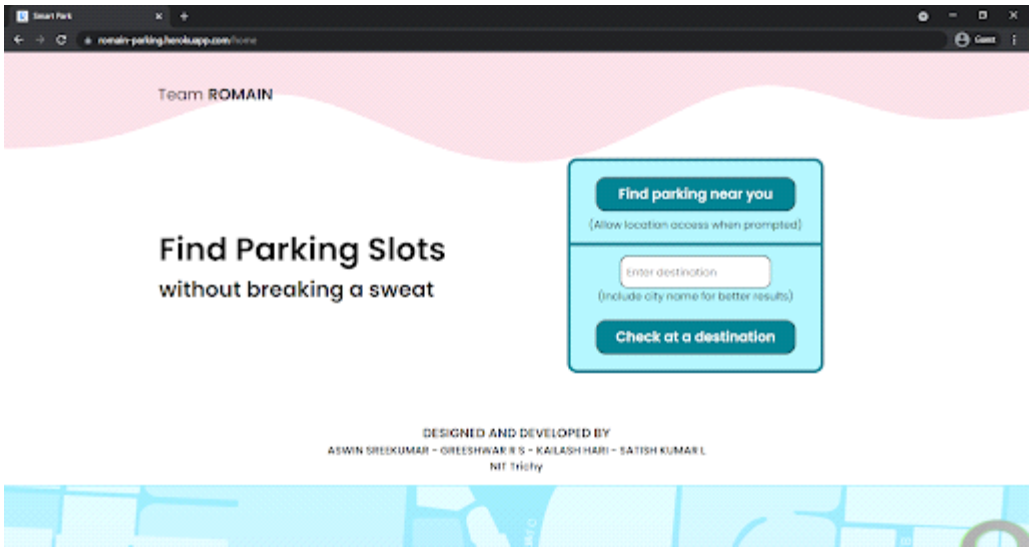


Mobile & app design:


Team ROMAIN
1 CHN66 Free slots: 22
2 CHN67 Free slots: 0
3 CHN64 Free slots: 8
4 CHN61 Free slots: 28


Team ROMAIN
Find Parking Slots
without breaking a sweat
Find parking near you
(Allow location access when prompted)
Enter destination
(Include city name for better results)
Check at a destination
DESIGNED AND DEVELOPED BY
ASWIN SREEKUMAR - GREESHWAR R S
KAILASH HARI - SATISH KUMAR L
NIT Trichy


Team ROMAIN
Map
Satellite
Chennai
1 CHN3 Free slots: 10
2 CHN65 Free slots: 0
3 CHN59 Free slots: 41
4 CHN1 Free slots: 13
5 CHN2 Free slots: 2



Program to smart parking system in Python:

```
class OurParkingSystem:
    def __init__(self, big, medium, small):
        self.sp = [0,big,medium,small]


    def addCar(self, carType):
        if(self.sp[carType] >0 ):
            self.sp[carType] -= 1
            return True
        return False


ps = OurParkingSystem(2, 0, 1)
print(ps.addCar(3))
print(ps.addCar(2))
print(ps.addCar(3))
print(ps.addCar(1))
print(ps.addCar(1))
print(ps.addCar(1))
```

Input:

```
ps.addCar(3)
ps.addCar(2)
ps.addCar(3)
ps.addCar(1)
ps.addCar(1)
ps.addCar(1)
```

Output:

```
True
False
False
True
True
False
```