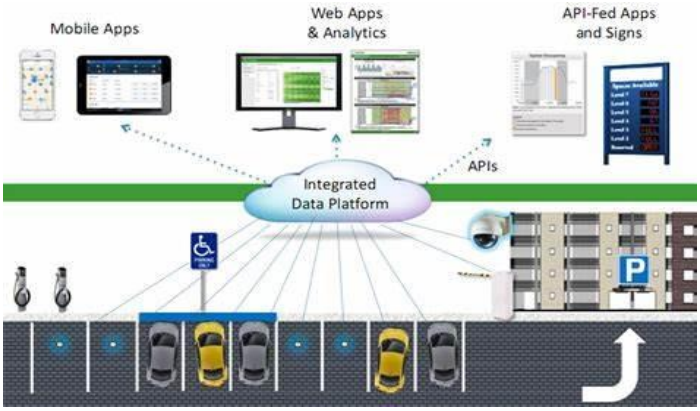
Phase 5: Project Documentation & Submission

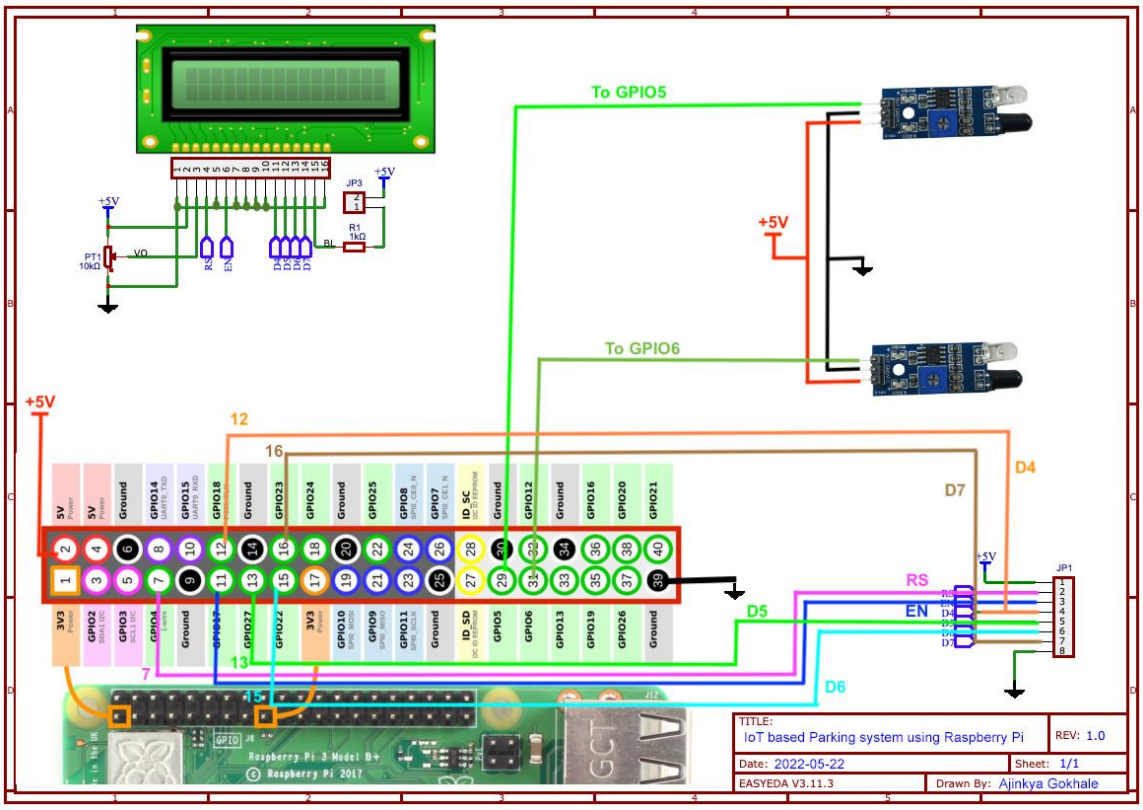
of smartparking system

project's objectives of smartparking:

The objective of a smart parking system is to identify a vehicle's presence or absence in a particular parking space with a high degree of accuracy. This data is then passed on to a system for visualization and analysis, which is available for parking asset managers and/or enforcement officers. The system aims to ease the parking process around malls, schools, and other areas. It helps control parking slot availability and allows drivers to book parking slots in advance. Smart parking systems use IoT devices and sensors to collect real-time data on parking lot occupance.

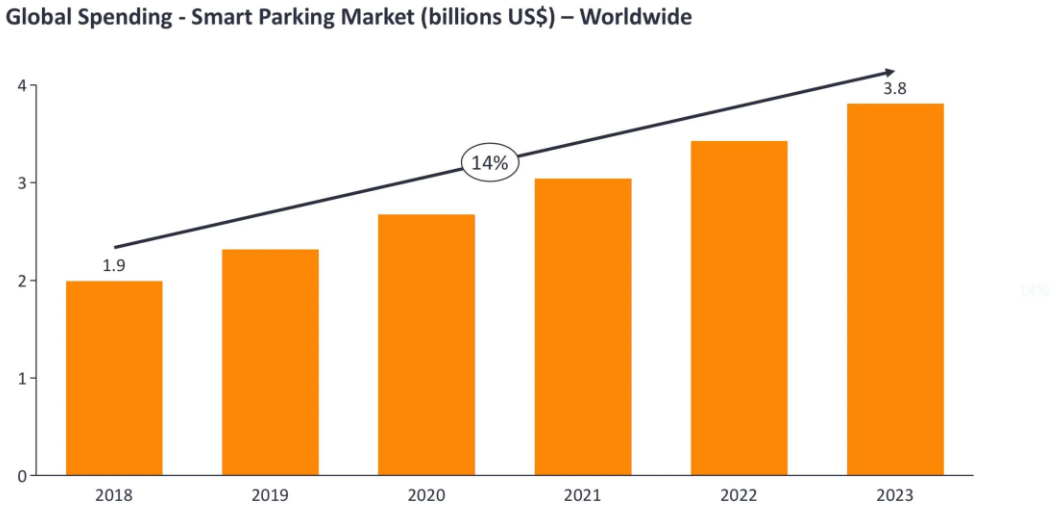


IoT sensor setup:



Demand for Smart Parking Mobile App:

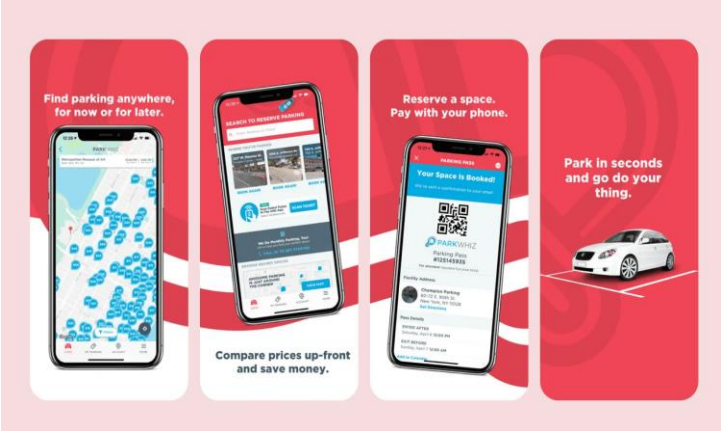
By 2023, the investment in smart parking projects is expected to reach $3.8 billion, witnessing a CAGR of 14% year over year. It's good news because a smart car parking project can address the challenges related to finding the space for a car while making it convenient for users to access everything right from their mobile devices.



IOT Smart Parking System MArket:

Is there a demand for smart parking app development in the market? Let's answer this with some quick statistics.

ParkWhiz is currently one of the most used parking apps. 1,000,000+ Android users are using it, and it has a rating of 4.4 on Google Play Store. Another parking app used widely in the US is the MKE Park App.



Parkwhiz

The number of users of these apps increased significantly in no time. More than 40 million users are accessing over 4,000 parking locations in the US. Recent reports also reveal that around 30% of cars are looking for a nearby parking location using mobile apps.

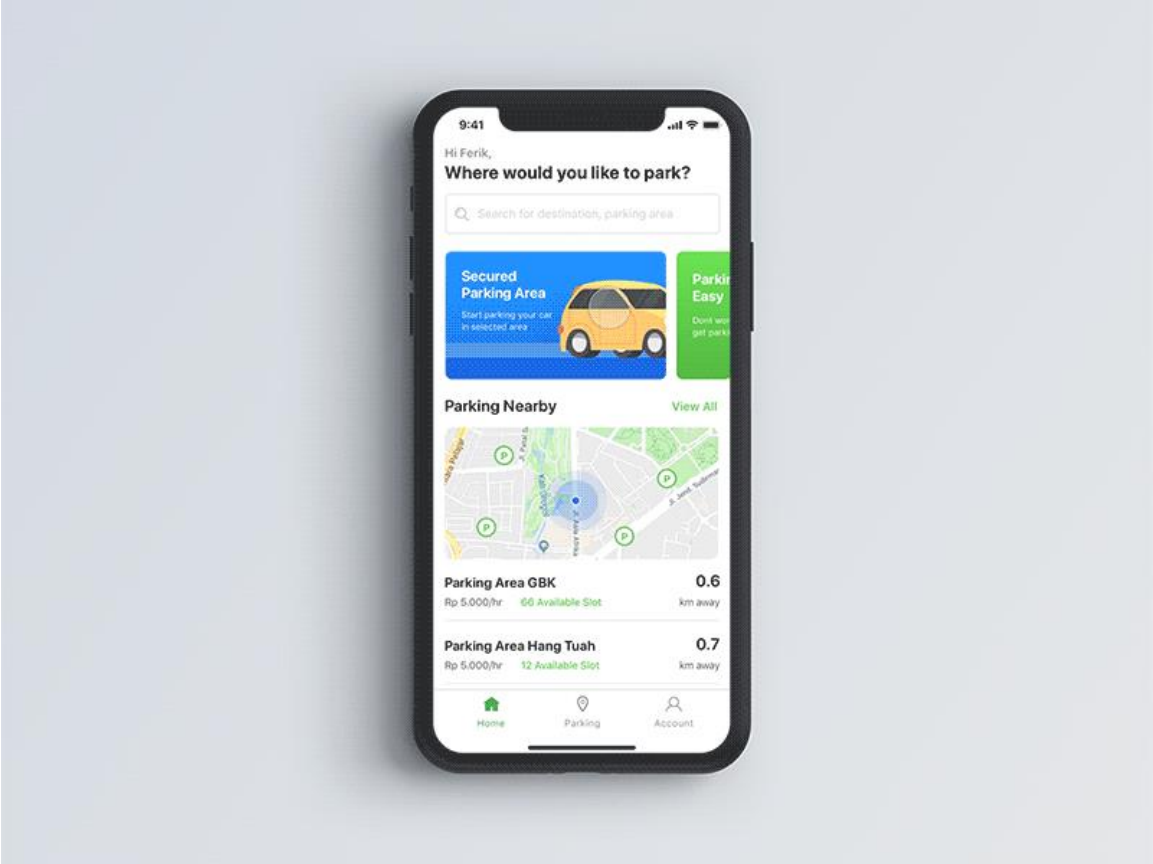
These stats show that IoT-based car parking apps are in demand today. Many startups and businesses are investing in this technology to grab the opportunity in this growing market.

However, there are many questions on this topic. How to develop a smart parking system mobile app? What is the cost to develop a smart car parking system using IoT app development? What are the key features of these apps? And more.

In this article, we have answered almost every question of yours.

Features of Smart Parking System

A smart app for parking must exhibit the following features:



Smart Parking System Development – User Panel Interface Features

Smart Parking System Development - Driver Panel App

Register / Login: Feature to sign up for the app and create an account using an email address or phone number. Another good idea is to allow login using Google Account. This makes the user experience effortless.

Navigation and search: The app should provide suggestions about the nearby parking locations along with the number of vacancies. They can then check the distance to the location and start navigating towards it.

User profile: After signing up, the user should be able to edit his profile. This allows him to add his profile photo, contact number, email address, and other information.

Booking: Integrate a feature that allows users to book a slot whenever required. There should also be options to book a slot on a weekly or monthly basis. This is good for the users who park their cars on a regular basis in that area. While booking a slot, the driver should also be able to add the expected arrival and departure times.

Payment: With this option, they should be able to pay online using debit/credit cards or mobile banking apps. People nowadays like to avoid cash tinkers.

Push notifications: Push notifications are great when it comes to showing the current offers and discounts to the app users. You can also notify them about app updates and the time when their parking time is about to end. This will help them avoid the fine or additional payment.

Parking history: An option to check the history of bookings should be there so that the user can check things whenever required. This feature will also include the payment history and mode of payment.

Multiple cities: If the user has traveled to any other city, he should also be able to use the app to find a nearby parking place.

Waiting list: If there are no parking vacancies in a certain location, the user should be able to check the waiting time to book a slot.

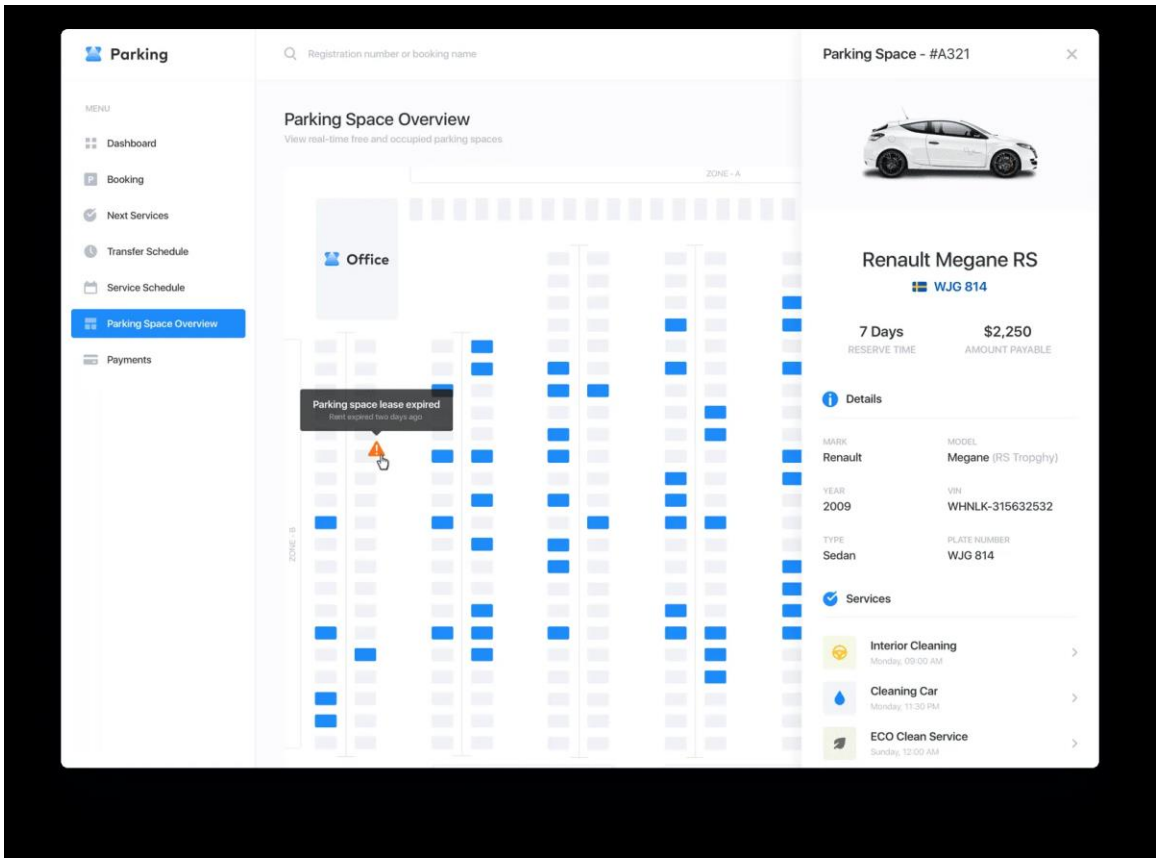
Pricing: Before booking a slot, the pricing should be shown. You can include a feature to show the full rate card for advance bookings weekly or monthly.

Multilingual support: The app should support multiple languages so that users don't have to get stuck and use the app in the language of their choice.

Help/Support: It is an important feature to support and help the customers if there are any issues. They should be able to contact customer service via chat or call.

Rating & Review: The user should have the option to rate his experience and write a review. Reviews are good, helping other users to decide which parking system is right for them. Parking owners can also see the review and improve the experience for more users.

Smart Parking System Development – Admin Panel Features:



Smart Parking System Software - Admin Panel

User Management: This will allow the admin to manage the users from the backend.

Revenue Control: Only the administrator can access the revenue flow of the app and the charges related to both the user and the parking provider.

Manage Earnings: The admin should be able to manage the earnings of the parking provider, commissions,etc.

Approve/Reject driver profile: When a new user signs up to the app, the admin receives a notification and can check his activities. In case there are any fraudulent activities, the admin can block or ban the user from using the app further.

Bookings Manager: A unified panel or option to manage all the bookings made using the mobile app.

Manage offers: The management of promotions, coupon codes, discounts, etc. should be managed by the admin.

Add new locations: If any new parking provider joins the app, the admin can add the new location in the app. Moreover, if the services are expanded to more cities, this can also be processed from the admin panel.

Reports: Option to see analytics and reports to understand whether the app is growing, what are the performance metrics, things to improve for further growth, etc.

Parker Panel Features

Register/login: Signing up for new parking providers.

Add ID Proof: Uploading relevant documents for credibility.

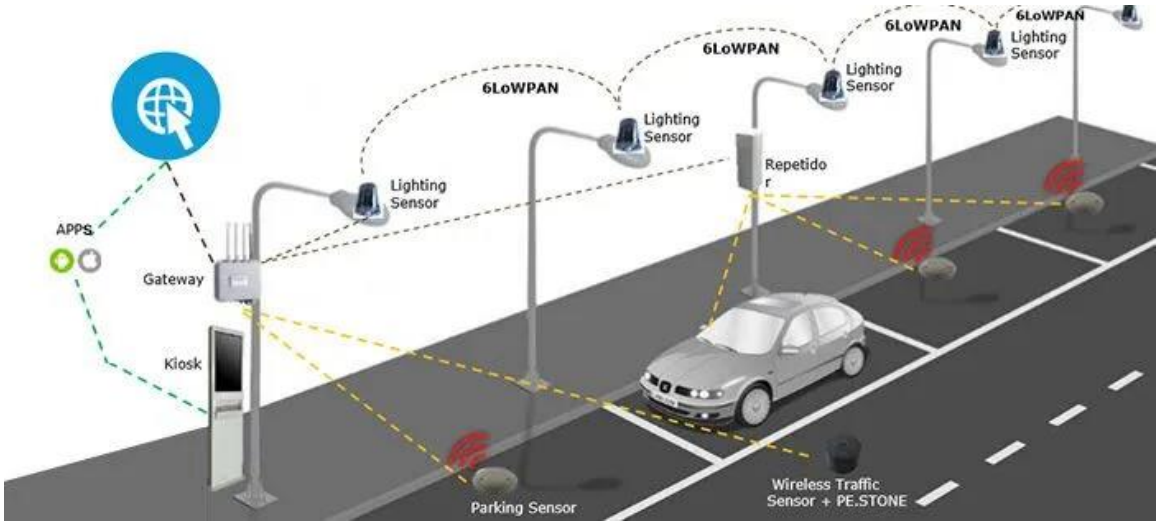
Contact customers: Contact number and email address for customers to reach out.

Receive payment from customers: Get payments from customers with their preferred payment options.

Check reviews and ratings: For parking providers to check the reviews and ratings posted by the customers. They should also be able to reply.

Accept or decline parking request: When the customer submits a request, the parker should be able to accept or reject it.

Smart Parking Management System Works:



Smart Car Parking system works

The app will connect the driver to the parking location listed in the app. When the driver opens the app, this is it will work.

Using sensors, networking, and data analytics, a smart parking system built on the Internet of Things is intended to manage and optimize parking spots effectively.

Sensor Positioning: In the parking complex, sensors are put in at strategic locations or in each parking place. These sensors may be cameras, magnetic sensors, infrared sensors, ultrasonic sensors, or other sorts.

Parking Data collection – The sensors continually track each parking space's condition. They gather information about whether a space is filled or empty. This data is sent to a central server or cloud platform through a wireless network, such as Wi-Fi, Bluetooth, or cellular networks.

Processing of data– The central server or cloud platform receives and processes data from every sensor. It analyses the data to decide which parking spaces are vacant and which are filled. Filtering away mistakes or erroneous readings may also be a part of the procedure.

User Interface (Mobile App) – Drivers are given a user interface, generally in the form of a website or a mobile app. With the help of this interface, users can instantly check the availability of parking spots, book a spot, and, if necessary, make payments.

Notification and Alert – Users may get messages from the system regarding bookings, reminders, and parking availability. This can shorten the time spent looking for parking spaces and help people schedule their parking in advance.

Payment Integration – The ability to pay for parking using an app or website is provided by certain smart parking systems that incorporate payment processing. Utilizing pre-paid accounts or linking to payment gateways may be necessary for this.

Security and surveillance – To increase the safety of the parking facility, many smart parking systems also contain security and surveillance features like cameras and alarms.

Data analytics – The system accumulates a considerable amount of data over time. Analytics may be applied to this data to improve parking garage management. For instance, it may be used to determine peak usage periods, space usage, and money production.

Search: The user can find near by parking facilities in the app by enabling GPS or turning the location tracking on.

Book and pay: Once the choice is made, he can pay online and book the slot.

Drive: The app will show the map navigation towards the parking slot after booking. The driver just needs to follow the route and reach the destination.

Find Your Car – Users can find other parking spaces by just registration numbers, as parking slots are enabled with OCR capabilities.

Hardware Required for Smart Car Parking Sensors:

Smart parking systems use hardware and Sensors to determine if parking spots are occupied. The selection of parking spot sensors may be influenced by elements including price, precision, and the particular needs of the smart parking App Development. Parking space sensors come in the following popular varieties:

Smart Parking System Development - Hardwares

Beacons for IOT smart parking app development:

Smartphones in the area can receive signals that Bluetooth Low Energy (BLE) beacons can emit. To aid customers in navigating and locating available parking spaces, these beacons can be installed in parking spaces or across the parking complex. Mobile app users can get proximity-based notifications and directions to the closest open parking.

Ultrasonic Sensors:

Ultrasonic sensors use sound waves to identify vehicles. The sensor monitors the amount of time it takes for the ultrasonic waves to return after they bounce off objects and are emitted by the device. Based on this time, the sensor detects whether the parking place is occupied or available.

Magnetometer Sensors:

Magnetic sensors pick up on modifications in the magnetic field brought on by the presence of a car. These sensors are especially helpful for seeing big metal items like vehicles and may be put in the ground or on the pavement.

Infrared Sensors:

Infrared sensors pick up an automobile's heat emissions. They may cover many parking spaces and are frequently mounted on poles or overhead structures.

Digital cameras:

Images of parked cars are among the visual data that video cameras record. These photographs may be examined, and the occupancy can be determined using sophisticated image processing and computer vision techniques.

Ground- or Surface-Mounted Sensors:

These are weight or pressure sensors that have been put on the parking space's surface or just below the pavement. They determine if a vehicle's weight occupies a space.

Sensors with lidar:

Lidar sensors make a 3D map of the environment using laser technology and may be used to find cars.

Sensor-Based Smart Parking Vs IOT Smart Parking System

Sensor Based Parking System Sensor-Based Parking System

Sensor Types:

– To detect parking spot occupancy, sensor-based systems often use a variety of sensors, including ultrasonic, magnetic, infrared, or cameras.

Detection Accuracy:

– Since these systems were created specifically to identify parking spot occupancy, they can do so with great accuracy.

Limited Connectivity:

– Sensor-based systems typically don't have many robust connection features. They may transfer data locally and detect occupancy but frequently lack more advanced IoT features.

Cost-Effective:

– Because these systems are task-focused, they may be less expensive to deploy and maintain in terms of sensors.

Scalability:

– When moving to larger facilities, they might need to make additional infrastructure adjustments.

Quick Implementation:

– Because of their simplicity, sensor-based systems are usually put into place more quickly.

Sensor Types-

IoT-based systems can employ a number of sensors, including IoT sensors like beacons as well as sensors used in sensor-based systems.

Connectivity

– Wi-Fi, Bluetooth, cellular networks, and other IoT communication protocols are frequently used by IoT-based devices to provide their wide connection features.

Data Integration

– By integrating with other IoT devices and systems, these systems may become a part of a larger ecosystem for smart cities or urban mobility.

Real-Time Data

-IoT-based technologies enable remote monitoring and management of parking facilities and give real-time data.

Analytics and Insights

– Internet of Things (IoT) solutions are better suited for gathering and analyzing information on parking patterns, user behavior, and facility performance.

Scalability

– IoT-based solutions are more flexible and can be adjusted to different types and sizes of facilities.

User Interaction and Navigation

– They can help users with their navigation across mobile apps, enhancing their overall user experience.

Cost considerations

– Because of their improved connection and data analytics features, IoT systems could be more expensive to deploy. IoT systems could be more expensive to deploy because of their improved connection and data analytics features.

In conclusion, sensor-based smart parking systems tend to be easier to use and more affordable for routine monitoring of parking spot occupancy. IoT-based solutions provide a wider range of functions, such as data analytics, connection, user engagement, and system integration. The decision between the IOT and Sensor-based Smart Parking App Development is based on the parking facility's individual objectives and budget and the need for a more comprehensive, integrated urban mobility solution.

Process for Smart Parking System Development:

Research the smart parking industry and market size

Start with some market research to understand the current state of car parking systems, competitors, the budget required to build the app, marketing strategies, etc. You can then decide whether to make it a local app or a global app, look for investors, and make other important decisions.

PArking System Development Company:

If you don't have an in-house team of dedicated developers, then it will be great to choose a reliable web development company that can build a smart parking system project in Java or any other coding language. Share your expectations and requirements with them and get things done.

Finalize the features for smart parking system development:

Please include all the features mentioned above in this article. However, if you are unsure about any specific attribute, you can exclude it. Finalize all the elements to understand which technologies will be required further in case some features need third-party integration.

UI/UX Design of Smart Car Parking System:

The UI/UX design of the app will play a vital role in the app's success. The navigation in the app should be simple so that users don't find it challenging to find the required options. The connection between each screen should be smooth and consistent across the entire app. In short, the users should like the UX and feel at ease while using it.

Smart Car Parking System and Mobile App Testing:

Once the app has been developed (both backend and front-end), it is time to test it. There should not be any glitches or issues with any features. The aim of testing is to find flaws in the UX so that these can be avoided before launch.

Releasing of Smart Parking Mobile App and Software:

When the mobile app is tested and ready, it can be released on app marketplaces, including Google Play Store and App Store. The app release should be a part of the marketing campaigns to receive initial installations quickly.Also, specific policies must be followed to list an app on marketplaces. Taking care of these things right from the time of the development phase will save you the time required at the end moment.

Support for Smart Parking Software and Apps:

Smart car parking system and mobile app needs maintenance and bug fixes every now and then. You can let a web development company take charge or hire a dedicated developer. If any customer has an issue with the app, he can report it, and the developers will resolve it.

Smart Car Parking App Development Cost:

The answer is that it depends on varied factors, such as the features to be integrated within the app, the experience and expertise of the team (developers, designers, testers) working on it, the country of operation, technical specifications, etc.

Moreover, if the app is built for both Android and iOS, then the cost will also increase.

Smart Parking System Development

Regardless of all the factors, here is a breakdown of the cost of developing a smart vehicle parking system using IoT:

Development cost: $20,000 – $70,000

UI/UX design cost: $5,000 – $15,000

Testing cost: $5,000 – $15,000

Total cost: $30,000 – $1,00,000

Since the cost of developers and designers varies based on the country of their residence, on average, the smart parking system cost will be between $50,000 to $1,00,000.

Wrapping up:

The development of a car parking system using IoT is not a trend but a need of the hour. Not many companies or startups have grabbed the opportunity yet, but those who have invested in the plan are experiencing massive benefits. Apps like ParkMobile, ParkMe, and ParkNow are just a few examples of successful mobile apps for smart parking

Raspberry Pi integration:

Parking slots have become a widespread problem in urban development. In this context, the growth of vehicles inside the university's campus is rapidly outpacing the available parking spots for students and staff as well. This issue can be mitigated by the introduction of parking management for the smart campus which targets to assist individually match drivers to vacant parking slots, saving time, enhance parking space utilization, decrease management costs, and alleviate traffic congestion. This paper develops an IoT Raspberry Pi-based parking management system (IoT-PiPMS) to help staff/students to easily find available parking spots with real-time vision and GPS coordinates, all by means of a smartphone application. Our system composes of Raspberry Pi 4 B+ (RPi) embedded computer, Pi camera module, GPS sensor, and ultrasonic sensors. In the IoT-PiPMS, RPi 4 B+ is used to gather and process data input from the sensors/camera, and the data is uploaded via Wi-Fi to the Blynk IoT server. Ultrasonic sensors and LEDs are exploited to detect the occupancy of the parking spots with the support of the Pi camera to ensure data accuracy. Besides, the GPS module is installed in the system to guide drivers to locate parking areas through the Blynk App. that discovers parking spaces availability over the Internet. The system prototype is fabricated and tested practically to prove its functionality and applicability. According

to the results, the IoT-PiPMS can effectively monitor the occupancy of outdoor parking spaces in the smart campus environment, and its potency in terms of updating the data to the IoT server in real-time is also validated.

Code implementation:

Here is a simple Python code for a smart parking system using OpenCV and a webcam. The code is composed of two scripts: selector and detector. The selector script selects the coordinates of the parking spaces and saves them into a file. The detector script gets the coordinates from the file and decides if the spot is available or not.

The selector:

Now, let's get to the coding part. First we need to build the selector. We start with importing the modules we need (this is available for both selector and detector)

```
import cv2

import csv
```

After that we can start working on getting the image on which we will select the parking spots. For that we can take the first frame provided by the webcam, save it and use the picture to select the spots. The following code works like this:

Opens the video stream in image variable; suc determines if the stream was opened successfully.

Writes the first frame into frame0.jpg.

The stream is released and all windows are closed.

The new saved picture is read in img variable.

```
VIDEO_SOURCE = 1

cap = cv2.VideoCapture(VIDEO_SOURCE)

suc, image = cap.read()

cv2.imwrite("frame0.jpg", image)

cap.release()

cv2.destroyAllWindows()

img = cv2.imread("frame0.jpg")
```

Now that we have saved the first frame and opened it in the img variable we can use selectROIs function to mark our parking spots. ROIs are defined as regions of interest and represents a portion of the image on which we will apply different functions and filters to get our results.

Back to selectROIs function, this will return a list (type: numpy.ndarray) which contains the numbers we need to assemble images with their boundaries as our ROIs.

```
r = cv2.selectROIs('Selector', img, showCrosshair = False, fromCenter = False)
```

Our list will be saved in r variable. The parameters given to cv2.selectROIs function are the following:

'Selector' is the name of the window that will let us select the ROIs.

img is the variable containing the image on which we want to select.

showCrosshair = Flase removes the center lines inside the selection. This can be set to True as there's no impact on the result.

fromCenter = False is quite an important parameter, as if this would have been set to True the proper selection would have been much harder.

After selecting all of the parking spots, it's time to write them into a .csv file. To do that, first, we need to transform our r variable into a python list. For this we can use rlist = r.tolist() command.

After we have our proper data, we can save it into the .csv file we will use further.

```
with open('data/rois.csv', 'w', newline='') as outf:
  csvw = csv.writer(outf)
  csvw.writerows(rlist)
```

The detector:

Now that we're done with selecting parking spots, it's time to do some image processing. The way I approached this was:

Get coordinates from the .csv file.

Build a new image out of it.

Apply Canny function available in OpenCV.

Count the white pixels inside the new image.

Establish a pixel range within the spot would be occupied.

Draw a red or green rectangle on the live feed.

For all of these operations we need to define a function to be applied for each spot. This is how the function looks like:

```
def drawRectangle(img, a, b, c, d):
    sub_img = img[b:b + d, a:a + c]
    edges = cv2.Canny(sub_img, lowThreshold, highThreshold)
    pix = cv2.countNonZero(edges)
    if pix in range(min, max):
        cv2.rectangle(img, (a, b), (a + c, b + d), (0, 255, 0), 3)
```

```
        spots.loc += 1
    else:
        cv2.rectangle(img, (a, b), (a + c, b + d), (0, 0, 255), 3)
```

Now that we have the function we need, all we have to do is to apply it to every set of coordinates in the .csv file, right? Could be, but we can make it even better.

First of all we have some parameters which would be nice if we could adjust them in real time while the script is running, also via a GUI. To do that we need to build a few track bars. Fortunately, OpenCV gives us the opportunity to do that without extra libraries.

First we need a callback function which does nothing but is required in as a parameter for creating track bars with OpenCV. Actually the callback parameter has a well defined purpose but we don't use it here. To read more about this visit the OpenCV docs.

```
def callback(foo):
    pass
```

Now we need to create the track bars. To do so we will use cv2.namedWindow and cv2.createTrackbar functions.

```
cv2.namedWindow('parameters')
cv2.createTrackbar('Threshold1', 'parameters', 186, 700, callback)
cv2.createTrackbar('Threshold2', 'parameters', 122, 700, callback)
cv2.createTrackbar('Min pixels', 'parameters', 100, 1500, callback)
cv2.createTrackbar('Max pixels', 'parameters', 323, 1500, callback)
```

Now that we created the GUI to operate the parameters there is one thing left. This would be the number of available spots in the image. Which is defined in the drawRectangle as spots.loc. This is a static variable which must be defined at the beginning of our program. The reason why this variable is static is because we want every drawRectangle function we call to write it's result in the same global variable, and not a separate variable for each function. This prevents the returned number of available spaces to be higher than the actual number of available spaces.

To implement this we just have to create the spots class with it's loc static variable.

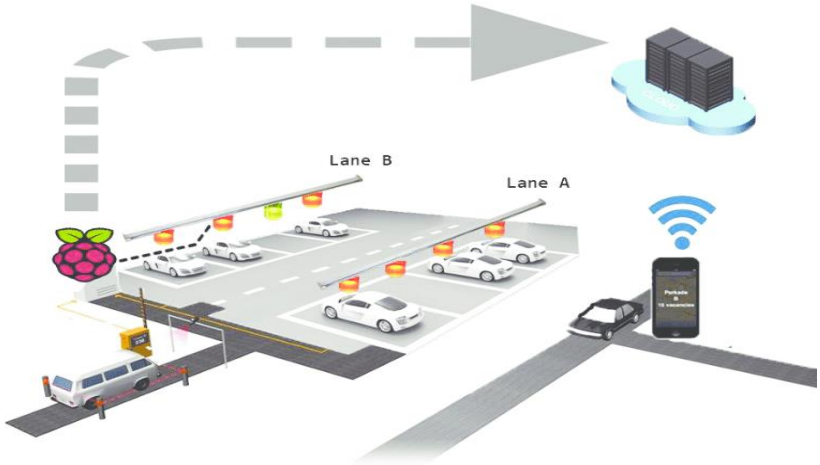
```
class spots:
    loc = 0
```

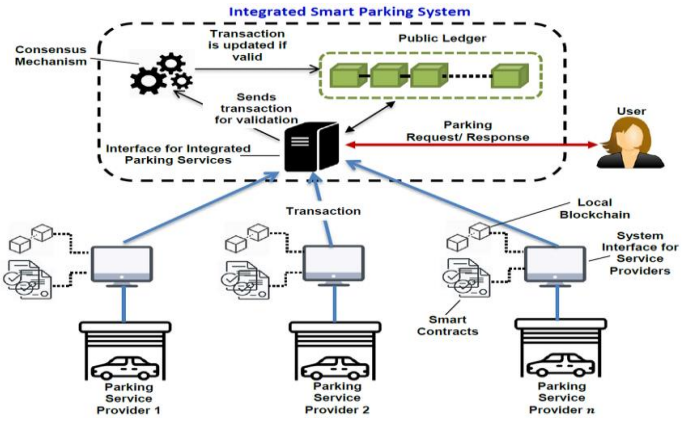
Now that we're almost ready, we just need to get the data from the .csv file, convert all its data into integers and apply our built functions in an infinite loop.

```
with open('data/rois.csv', 'r', newline='') as inf:
    csvr = csv.reader(inf)
    rois = list(csvr)
```

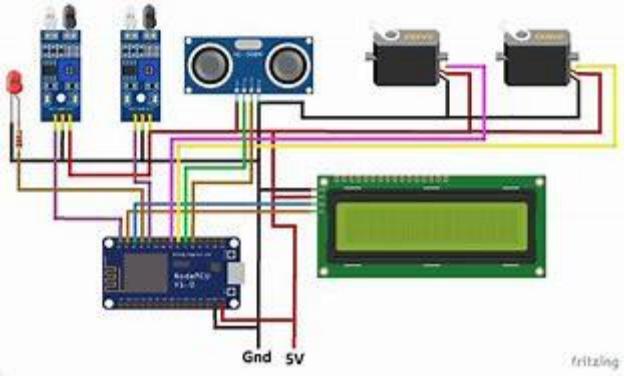
```
rois = [[int(float(j)) for j in i] for i in rois]
VIDEO_SOURCE = 1
cap = cv2.VideoCapture(VIDEO_SOURCE)
while True:
    spots.loc = 0
    ret, frame = cap.read()
    ret2, frame2 = cap.read()
    min = cv2.getTrackbarPos('Min pixels', 'parameters')
    max = cv2.getTrackbarPos('Max pixels', 'parameters')
    lowThreshold = cv2.getTrackbarPos('Threshold1', 'parameters')
    highThreshold = cv2.getTrackbarPos('Threshold2', 'parameters')
    for i in range(len(rois)):
        drawRectangle(frame, rois[i][0], rois[i][1], rois[i][2], rois[i][3])
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(frame, 'Available spots: ' + str(spots.loc), (10, 30), font, 1, (0, 255, 0), 3)
    cv2.imshow('Detector', frame)
    canny = cv2.Canny(frame2, lowThreshold, highThreshold)
    cv2.imshow('canny', canny)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

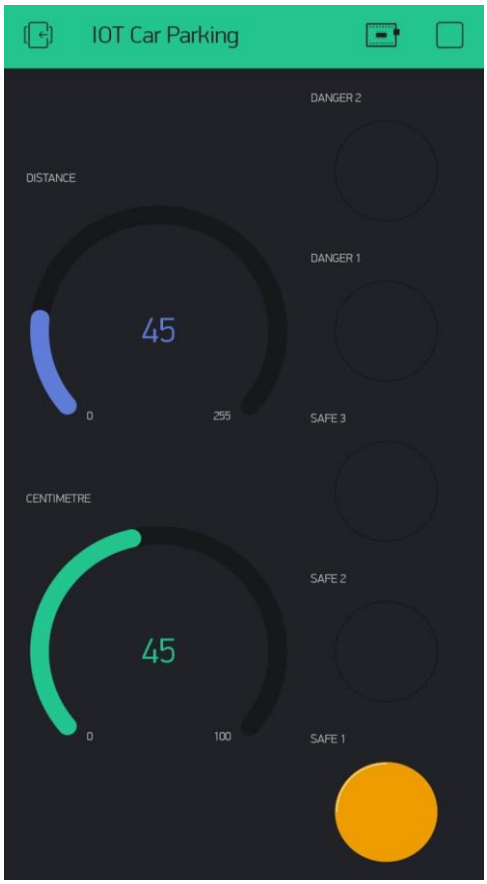
diagrams for smart parking system:

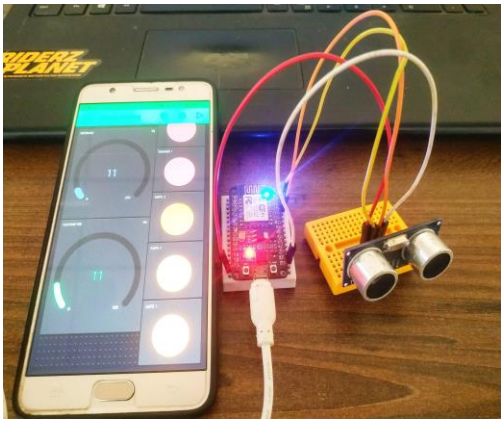




schematics for smart parking system:


Gnd
5V
fritzing


IOT Car Parking
DANGER 2
DISTANCE
DANGER 1
45
0
255
SAFE 3
CENTIMETRE
SAFE 2
45
0
100
SAFE 1



Real-time parking availability system:

Basic Components and Development Process for a Real-Time Parking Availability System

The parking inventory tracking system typically comprises three components:

a parking management server

IoT hardware platforms (sensors and cameras)

a mobile app

Parking Management Server

The parking management server monitors all the parking spaces:

tracks parking spot availability

predicts availability in real-time

manages metering and billing for parking users

A prediction module can use AI technology, such as neural networks, to predict parking occupancy in different areas of the city based on the historical data in each parking region.

IoT Hardware:

An IoT hardware platform based on sensors or cameras detects the presence or absence of a vehicle. It connects parking structures to the server and transmits the data.

Mobile App:

A mobile app connects the system with the users so drivers can find a vacant space quickly and easily using clear and simple directions. Generally, such an application shows the driver a real-time view of available and taken parking spots via a simple and comprehensive interface, allowing a parking space reservation and automatically tracking billing based on when the driver's vehicle is logged entering and leaving the parking space.

Location-Based Technology;

To improve the parking experience you can use location-based technology in the app. Examples of such technology are geofencing and iBeacons. A geofence feature implemented in the app allows giving automatic alerts about how many spots are available in the lot to an app user driving into the area. iBeacons, which are low-energy Bluetooth devices, can detect an app user in the vicinity of the parking lot area and notify this user about the available parking spots.

Benefits of the Smart Parking System:

Real-time parking availability system development opens the door to opportunities and innovations. Cloud technology allows automated sharing of availability information with drivers, dynamic pricing implementation based on the parking utilization rate, and efficient parking space utilization. Smart parking reservation systems create a better customer experience and lead to lower city congestion and higher parking revenues.

1. INTRODUCTION

More than half of the world population lives in the urban areas so the cities have reached its full occupancy. As a result, number of vehicles in the cities is also increased. Car parking system would have more appreciated in placeless of higher demands such like Theatre, Shopping malls and in some crowded place. The devices could be tracked, controlled or monitored using remote computers connected through the Net. In IOT objects are connected to each other and exchange information from internet. Our cloud based smart parking organized the parking lot. It helps user to find a vacant space in parking slot. It saves user's time as well as vehicle fuel. An infrared (IR) sensor is used at each slotin parking; ittells the space availability which can be easily seen in mobile application through internet.It may bedefined as connecting things present in the physical world with sensors and then connecting them to a network through wired or wireless means.

The solution proposed in this paper utilizes the architecture ofthe cloud server is in a way that an unrestricted number of slots may be added without any change in the code. The associated mobile application can run on Windows,Android and iOS. Moreover, the code can be recycled for multiple boards making the proposed solution cost effective, adaptable and versatile. Followed by the

developments in sensor technology, man modern cities have opted for deploying various IoT based systems in and around the cities for the purpose of monitoring

2. ARCHITECTURE:

A. Sensors:

The Infra-Red photoelectric reflective module, with range of 0.02 to 0.4 meters was used. It requires 5Volts and less than 2mA of current. The required powers provided by the raspberry pi board. The output goes low in the presence of an obstacle.

B. Raspberry- Pi:

The Raspberry has the following collections: a power source, an Ethernet cable and to 12 sensors. A voltage divider network is used to connect the sensors to the GPIO pins of the Raspberry Pi as they give an output of 5V but the raspberry pi accepts inputs of 3.3V. The device is registered using Web API to be recognized as a part of the network. Using the sensor, we know the presence of the vehicle, is available on parking slot or not. This is enabled by checking the distance of the object in the slot from the sensor.

C. Cloud Database:

Information regarding registered applicant, status of the parking slot, and time duration of parking is maintained in server. The information regarding availability of parking slot is shared among all users who use mobile application. Database server is updated from two terminals. One is from sensors and System. Cloud computing offers cheap and effective ways of connecting, tracking, and managing devices from anywhere over the Net. By the use of built-in applications systems could monitor and control things on a real-time basis through remote locations. The processed data from the Web jobs and Web APIs is stored on the firebaseDatabase. The function ofthe Web job is to send the Queued data from the Service. Cloud provides a scalable approach.

D. Mobile Application:

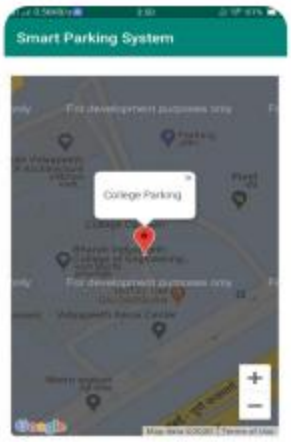
The mobile application for the users is developed using Android Studio Tools on Microsoft Visual Studio. Angular JavaScript was used so that the application may be ported across Android, Windows and IOS platforms. It reads outthe number of empty parking bays in the user's preselected preferred parking level or zone. The application has an error reporting facility which may lead to the recognition of faulty sensors. This draft facilitates user to login and registration for new user. During login user has to note preregistered username and password. For registration user has to provide name, password, E-mail IDetc.After successful login there will be a desk window at the left most corner of main window. When user selects desk window there will three options available to him which are profile, booking and checkout.

3. PROCESS:

There are some steps to park a vehicle in parking slot, such as follows

1. Android app:

Using this app, we get the information about the vacant parking slot; using this app know aboutthe actual location of the parking slot. You pay your bill via through this app. New users are also registered easily.



2. Database/Display/payment:

Using display, we got the information about the user. When user allot the slot and they ale not arrivewithin time, after10 minutes allocated slot will be cancelled. Payment system in metropolitan areas searching parking, slot is very critical in peak hours. In our system, user can book the parking slot in advance like theatre tickets booking online by mobile phone. We can book parking spaces by mobile apps and websites.

4. CONCLUSION:

Work proposed in this system addresses an issue of parking in smart cities. The system is implemented using low cost IR sensors, Raspberry pi model 3b for real time data processing, E-parking mobile application motor. The developed system contributes real time information of availability of parking slots in parking area and allows users to book parking slot from remote locations by using mobile application and also provides user authentication. The developed system is tested for different cases such as single user booking, multiple users booking, user trying to book reserved slot and user authentication.