# SMART PARKING USING IOT

Team Leader name and Members:

VIGNESH SEKAR.S (Leader)

[vigneshsekar2003s@gmail.com](mailto:vigneshsekar2003s@gmail.com)

SYED ALTHAF.F

[althafkhankhan19@gmail.com](mailto:althafkhankhan19@gmail.com)

CHRISTY MATHAI

[christymathai008@gmail.com](mailto:christymathai008@gmail.com)

VASANTHARAM.R

vasantharam321@gmail.com

Developing a smart parking system using python involves a combination of various web development technologies and hardware components.Here’s an overview of the technology stack and components build a smart parking system:

Web framework:

You’ll need a web framework to build the backend of your smart parking system.Flasks and Django are

Popular choices for python-based web development

. . Use web development technologies like HTML, CSS, and JavaScript, and consider using a web framework for efficiency.

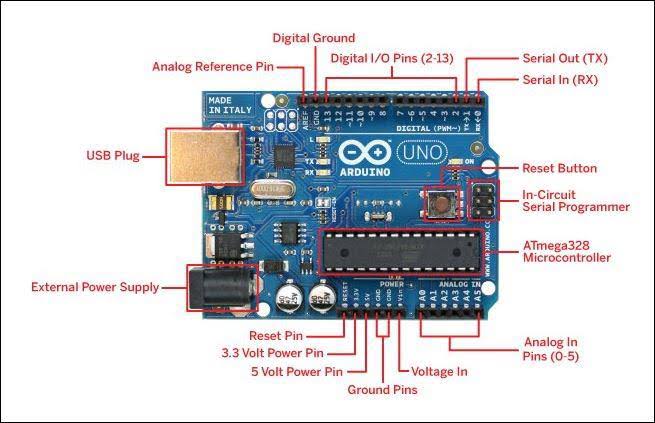
**HTML/CSS**: Design the dashboard's layout and style using HTML and CSS.

**JavaScript**: Implement interactivity for real-time updates, charts, and user management.

**Web Framework:** You can use popular frameworks like React, Angular, or Vue.js for a more organized and responsive interface

1. **Mobile App:**

Develop a mobile app to reserve parking spots, make payments, and receive notifications. Use cross-platform mobile app development frameworks like React Native or Flutter to streamline app development for both Android and iOS.



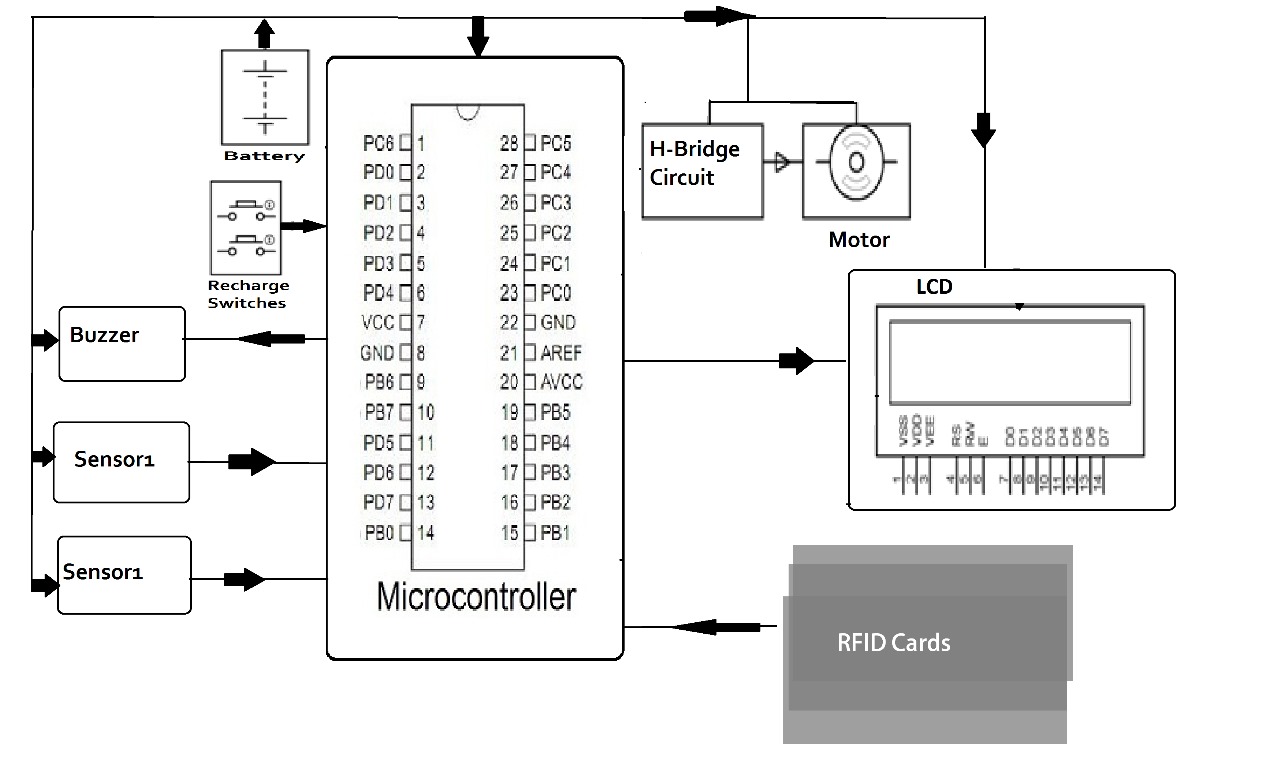
1. **Online Reservation System:**

Implement a web-based reservation system for students to check parking spot availability and make reservations. This system can be integrated with the mobile app and can be developed using standard web technologies.

* + **HTML/CSS**: Design the reservation interface.
  + **JavaScript:** Develop interactive features, such as selecting a parking spot and specifying the reservation duration.
  + **Backend:** Implement reservation logic on the server side, making use of frameworks like Express.js (Node.js) or Django (Python).

1. **Payment Gateway Integration:**

If you include a payment system, you'll need to integrate a payment gateway into your web app for processing payments. Popular payment gateways often provide APIs for this purpose. Here's a simplified example using Python and Flask:



1. **Real-time Updates:**

Use web development technologies to ensure real-time updates on parking spot availability, reservation confirmation, and payment status. You can achieve this with technologies like WebSocket for real-time communication between the server and clients.

**WebSocket**: Implement WebSocket communication to push real-time updates to the web and mobile clients when a parking spot's status changes

1. **User Authentication and Management:**

For user authentication and management, you can create user registration and login systems within the mobile app and web interface. Use web development technologies for user interfaces and backend logic

1. **Data Analytics and Reporting:**

Utilize web technologies to create data analytics and reporting features for administrators. You can use JavaScript libraries for data visualization and reporting tools.

**Data Visualization Libraries**: Integrate libraries like Chart.js or D3.js to display parking utilization statistics and trends.

**Backend**: Develop APIs for fetching historical parking data and generating reports

# Mobile App Development

To connect your IoT Smart Parking System with a mobile app, need to create APIs that allow the mobile app to interact with the backend system. Here's a stepby-step guide on how to achieve this

1. **Develop Backend APIs:** 
   * Create a set of API endpoints on your server to handle various functionalities of the Smart Parking System, such as user authentication, parking spot availability, reservations, and payments. You can use a web framework like Express.js (Node.js) or Django (Python) to develop these APIs.

1. **User Authentication:** 
   * Allow users to register and log in to the mobile app.
   * Create API endpoints for user registration and login.
   * Implement token-based authentication for secure access to the app.

1. **Parking Spot Availability:**

Develop an API endpoint to provide real-time information about parking spot availability.

* + The mobile app can query this endpoint to display available parking spots to users.

**4.Mobile App Development:**

Develop the mobile app using a cross-platform framework like React Native or Flutter to ensure compatibility with both Android and iOS.

Implement user interfaces for registration, login, parking spot selection, reservations, and payment processing.

**5.API Integration:**

Use HTTP requests (e.g., GET, POST, PUT, DELETE) in the mobile app to communicate with the backend APIs.

Handle API responses in the app to update the user interface and provide feedback

**Program:**

Creating a complete mobile app for an IoT Smart Parking System is a complex task that requires a significant amount of code and development effort. I can provide you with a simplified example of a Python program using the Kivy framework to create a basic user interface for a mobile app. Please note that this example is a

To create a Python mobile app using the Kivy framework, follow these steps:

1. Install Kivy if you haven't already. You can do this using pip:

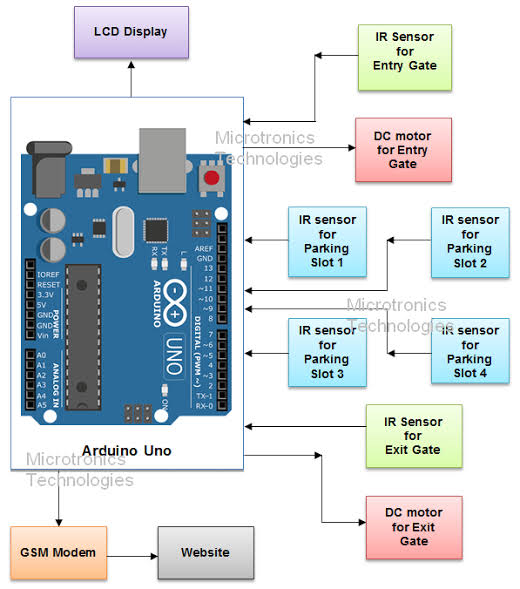
pip install kivy

1. Create a Python script for mobile app. This script will serve as a basic user interface for accessing the parking system features:

PROGRAM

1. from kivy.app import App
2. from kivy.uix.boxlayout import BoxLayout from kivy.uix.label import Label from kivy.uix.button import Button
4. class SmartParkingApp(App): def build(self):
5. layout = BoxLayout(orientation='vertical')
7. # Create labels and buttons for different functionalities label1 = Label(text="Welcome to Smart Parking") label2 = Label(text="Available Parking Spots: 10") reserve\_button = Button(text="Reserve a Spot") payment\_button = Button(text="Make a Payment")
9. # Bind functions to buttons
10. reserve\_button.bind(on\_release=self.reserve\_spot) payment\_button.bind(on\_release=self.make\_payment)
12. layout.add\_widget(label1) layout.add\_widget(label2) layout.add\_widget(reserve\_button) layout.add\_widget(payment\_button)
14. return layout
16. def reserve\_spot(self, instance):
17. # Implement reservation logic here print("Reserving a parking spot...")

SmartparkingApp().run()



Conclution:

This code provides a very basic user interface for the Smart Parking System. For a complete app, that would need to design more advanced UI components, implement user authentication, handle responses from the server, and manage the app's navigation flow