**SRI SHANMUGHA COLLEGE OF ENGINEERING AND TECHNOLOGY**

**Department of Biomedical Engineering**

**SMART PARKING** Present by 3rd year

S.Swetha-(732721121055)

M.Sindhu-(732721121046)

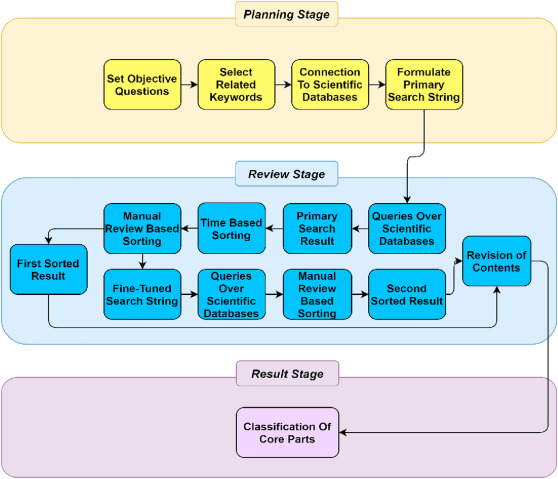
T.Srimathi-(732721121051)

C.Chanthruba-(732721121007)

R.Bharathi-(732721121006) 

**Introduction**:

The goal of parking system project is to reserve parking spot for a car/vehicle before it arrives. One of the most problems that the driver faces is finding a free parking spot, so many driver stopping their cars at the edges of the street. Therefore, we choose this to prevent the frustration of finding a parking spot and they can reserve a spot when they stay at home. It is an IOT based project.



**Necessary steps** :

**Sensor Deployment**: Deploy IoT sensors (e.g., ultrasonic, magnetic, or cameras) in each parking space to detect vehicle presence. Ensure these sensors are robust and can withstand environmental conditions.

**Connectivity**: Establish a reliable network infrastructure (e.g., Wi-Fi, LoRa, or cellular) to connect the sensors to a central server or cloud platform. Ensure data transmission is secure and efficient.

**Data Processing**: Process the data collected from the sensors to determine the availability of parking spaces in real-time. Algorithms and analytics can help in this process.

**User Interface**: Develop a user-friendly mobile app or a web interface where drivers can check the availability of parking spaces, reserve spots, and make payments if necessary.

**Payment Integration**: If your system involves paid parking, integrate payment gateways for users to make payments securely.

**Notifications and Alerts**: Implement a notification system to alert users when a parking space becomes available or when their reservation is about to expire.

**Security Measures:** Ensure the security of the IoT devices and data. Implement encryption, access control, and security protocols to protect against cyber threats.

**Scalability:** Design the system to be scalable to accommodate more parking spaces or additional features in the future.

**Monitoring and Maintenance**: Regularly monitor the system for performance and maintenance issues. Set up alerts for sensor malfunctions or connectivity problems.

**User Feedback and Improvements**: Collect feedback from users to make continuous improvements to the system. Use data analytics to identify patterns and optimize parking space allocation.

**Compliance and Regulations**: Be aware of local regulations and compliance requirements for parking systems, including data privacy and accessibility standards.

**Sustainability**: Consider the environmental impact and energy efficiency of the system. Optimize power usage and minimize environmental footprint.

**Testing and Quality Assurance**: Thoroughly test the system to ensure its reliability, accuracy, and user-friendliness.

**Launch and Marketing**: When everything is in place, launch the system, and promote it to potential users. Effective marketing can help in gaining user adoption.

**Data Analysis and Reporting**: Use data collected from the system to generate reports and insights that can be valuable for urban planning and decision-making.

**Feedback Loop**: Continuously gather feedback from users and administrators to make necessary adjustments and improvements.

**Program:**

#include &lt;Ultrasonic.h&gt;

#define NUM\_PARKING\_SPACES 4

Ultrasonic ultrasonic[NUM\_PARKING\_SPACES] = {

Ultrasonic(2, 3), // Trigger, Echo for Space 1

Ultrasonic(4, 5), // Trigger, Echo for Space 2

Ultrasonic(6, 7), // Trigger, Echo for Space 3

Ultrasonic(8, 9) // Trigger, Echo for Space 4

};

int parkingSpaceStatus[NUM\_PARKING\_SPACES] = {0}; // 0 = vacant, 1 = occupied

void setup() {

Serial.begin(9600);

for (int i = 0; i &lt; NUM\_PARKING\_SPACES; i++) {

pinMode(i \* 2 + 2, OUTPUT); // Initialize LED pins for each parking space

}

}

void loop() {

for (int i = 0; i &lt; NUM\_PARKING\_SPACES; i++) {

long distance = ultrasonic[i].timing();

int status = parkingSpaceStatus[i];

// Adjust the threshold based on your sensor placement and environment

int threshold = 20; // Distance threshold (in cm) to detect a car

if (distance &lt; threshold &amp;&amp; status == 0) {

// Vehicle detected

parkingSpaceStatus[i] = 1;

digitalWrite(i \* 2 + 2, HIGH); // Turn on LED for occupied space

Serial.print(&quot;Space &quot;);

Serial.print(i + 1);

Serial.println(&quot; is occupied.&quot;);

} else if (distance &gt;= threshold &amp;&amp; status == 1) {

// Space vacated

parkingSpaceStatus[i] = 0;

digitalWrite(i \* 2 + 2, LOW); // Turn off LED for vacant space

Serial.print(&quot;Space &quot;);

Serial.print(i + 1);

Serial.println(&quot; is vacant.&quot;);

}

}

}

|  |  |
| --- | --- |
|  |  |
|  |  |

**Preprocess the Dataset:**

**Data Collection:**

Gather data from various sources, such as sensors, cameras, and IoT devices, to monitor parking spaces and vehicle occupancy.

**Data Cleaning:**

Remove duplicate records, missing values, and outliers from the collected data to ensure data quality.

**Data Integration:**

Combine data from different sources to create a unified dataset that provides a comprehensive view of the parking area.

**Data Transformation:**

Convert data into a consistent format and units, ensuring that all data sources are compatible.

**Time Series Analysis:**

Analyze time-based data to identify patterns and trends in parking space occupancy.

**Spatial Analysis:**

Perform spatial analysis to determine the location of parking spaces and their proximity to entrances, exits, or other amenities.

**Data Normalization:**

Normalize numerical data to a common scale, making it easier to compare and analyze.

**Feature Engineering:**

Create new features that can help improve parking space occupancy prediction, such as historical occupancy rates or weather conditions.

**Data Aggregation:**

Aggregate data at different time intervals (e.g., hourly, daily) to provide summary statistics for analysis and visualization.

**Data Labeling:**

Label parking spaces as occupied or vacant based on sensor data, which is essential for training machine learning models.

**Data Validation:**

Continuously validate and monitor the data for consistency and correctness to ensure the system's accuracy.

**Data security:**

Implement data encryption and access control to protect sensitive information, such as license plate data.

**Data Storage:**

Store preprocessed data in a reliable and scalable database or cloud storage for easy access and retrieval.

|  |  |
| --- | --- |
|  | Smart Parking System Market Size & Growth, 2028 | Industry Report |

**Loading the Dataset :**

Loading a dataset for smart parking using IoT typically involves working with data related to parking spots, occupancy, and other relevant information collected through IoT devices. To load such a dataset, you would typically follow these steps:

**Data Source:** Identify the source of your dataset. This could be IoT sensors, cameras, or other devices installed in parking lots.

**Data Collection:** Ensure that data from IoT devices is being collected and transmitted to a central database or storage system.

**Data Format:** Understand the format of the data. It might be in CSV, JSON, or a database format.

**Access and Permissions:** Ensure you have the necessary permissions and access to the dataset, especially if it's collected from a third-party source.

**Data Processing:** Depending on the format and quality of the data, you may need to preprocess it. This could involve cleaning, filtering, and transforming the data to make it usable for analysis.

**Data Loading:** Use tools or programming languages like Python to load the dataset into your analysis environment. Common libraries like Pandas are often used for this.

Here's a simple example in Python for loading a CSV dataset using Pandas:

**Python code**

**import pandas as pd**

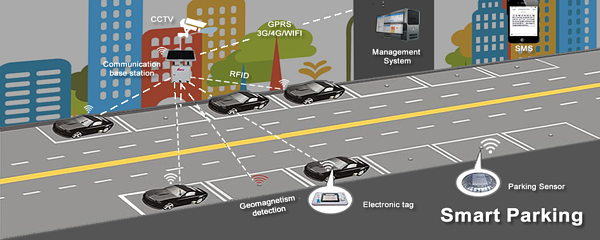
# Load the dataset

df = pd.read\_csv('your\_dataset.csv')

# Now you can work with the dataset using df

**Exploratory Data Analysis (EDA):** Once loaded, perform exploratory data analysis to understand the structure and content of the dataset.

**Analysis and Visualization:** Analyze the data to gain insights into parking patterns, occupancy rates, and other relevant information. You can use libraries like Matplotlib or Seaborn for visualization.



**Conclusion:**

Smart parking solutions offer a comprehensive approach to tackling the challenges associated with parking in today’s urban environments. While initial implementation can be an investment, the long-term benefits in terms of reduced traffic congestion, environmental impact, and improved user experiences make smart parking a valuable addition to modern cities. As technology continue to advance, smart parking systems are likely to evolve and become even more integral to urban planning and sustainability efforts.

|  |  |
| --- | --- |
|  |  |