# TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

## 1.1 PROJECT OVERVIEW

Smart Parking is an IoT-based project aimed at revolutionizing urban parking. It leverages sensors and connectivity to provide real-time data on parking space availability, significantly reducing the time drivers spend searching for spots. This system includes a user-friendly mobile app and signage for effortless navigation and convenient payments, enhancing the overall parking experience. It not only optimizes space usage but also contributes to environmental sustainability by reducing idling and traffic congestion. Furthermore, it offers valuable data to parking operators for operational efficiency and future planning.

## 1.2 PURPOSE

The purpose of Smart Parking in IoT is to optimize parking space utilization by using sensors and connectivity to provide real-time data on parking availability. This technology enhances the parking experience, reduces congestion, and minimizes the environmental impact of urban parking, ultimately improving urban mobility and user convenience. The Smart Parking System is an innovative solution aimed at optimizing and enhancing the parking experience for both drivers and parking facility operators. This system utilizes advanced technologies to provide real-time information, increase efficiency, and reduce the stress associated with finding parking spaces in congested areas. The project is designed to make parking easier, more convenient, and environmentally friendly.

# CHAPTER 2

# PROBLEM DEFINITION

## 2.1 EXISTING PROBLEM

One significant problem in smart parking systems is the potential for technical glitches and sensor inaccuracies. These issues can result in incorrect

information about parking space availability, leading to driver frustration and wasted time. Additionally, issues related to data privacy and security can arise when collecting and handling user information through mobile apps. Integration challenges with existing parking infrastructure and regulations also pose obstacles to the seamless adoption of smart parking systems, hindering their widespread implementation.

### 2.2 PROBLEM STATEMENT DEFINITION

The problem statement in smart parking revolves around the inefficiencies and challenges associated with traditional parking systems. These issues include extended search times for available parking spaces, resulting in increased traffic congestion, environmental pollution, and driver frustration. Moreover, the inaccurate or unreliable data from parking sensors can lead to confusion and wasted time for users. Security and privacy concerns related to the collection and management of personal data within smart parking applications are also problematic. Additionally, integrating these modern systems with existing infrastructure and regulatory frameworks presents a hurdle. Addressing these issues is critical to realizing the full potential of smart parking solutions for urban mobility and sustainability

# CHAPTER 3

# PROPOSED SOLUTION

### 3.1 PROPOSED SOLUTION

The proposed solution for smart parking involves the implementation of an advanced IoT-based system with robust sensor infrastructure and user-friendly mobile applications. Sensors would accurately monitor parking space availability in real-time, significantly reducing search times for drivers. A comprehensive mobile app would provide real-time parking information, navigation to available spaces, and cashless payment options. Enhanced security measures and transparent data handling practices would address privacy concerns. To facilitate integration, a centralized control system would ensure compatibility with existing infrastructure and regulatory requirements. This holistic approach aims to create a seamless and efficient parking experience, reducing congestion, environmental impact, and driver stress while promoting urban mobility and sustainability.

### 3.2 PROBLEM SOLUTIONFIT

The problem-solution fit in smart parking is evident through the alignment of the challenges faced by traditional parking systems with the proposed innovative solutions. The problems of extended search times, inaccurate data, and security concerns are directly addressed by the implementation of IoT-based sensor systems and user-friendly mobile applications. The solution not only reduces congestion and environmental pollution but also enhances user experience and privacy protection. Moreover, by ensuring compatibility with existing infrastructure and regulations, the proposed system creates a harmonious fit within the urban landscape. This convergence of problems and solutions demonstrates the effectiveness and relevance of smart parking systems in tackling real-world parking issues.

# CHAPTER 4

# REQUIREMENT ANALYSIS

## 4.1 FUNCTIONAL REQUIREMENTS

Functional requirements for a smart parking system ensure that it performs the necessary tasks and meets the needs of users effectively. Here are some key functional requirements for a smart parking system:

1. Vehicle Detection:

The system must accurately detect the presence of vehicles in parking spaces using sensors such as cameras, ultrasonic sensors, or in-ground sensors.

2. Real-time Availability Information:

Provide real-time information on the availability of parking spaces, including the total number of spaces, the number of vacant spaces, and their locations.

3. Reservation and Booking:

Allow users to reserve parking spaces in advance through a mobile app or website, ensuring that they have a guaranteed spot upon arrival.

4. Payment and Billing:

Enable secure and convenient payment methods, such as credit card payments or mobile wallets, and generate invoices or receipts for users.

5. User Authentication:

Implement a system for user authentication to ensure that only authorized individuals can access and use the parking facilities.

6. Navigation and Guidance:

Provide turn-by-turn navigation to guide drivers to available parking spaces within the parking facility.

7. Mobile Application:

Develop a user-friendly mobile application that allows users to find, reserve, and pay for parking spaces easily.

8. Alerts and Notifications:

Send notifications to users regarding the status of their reservations, parking availability, or reminders when their parking time is about to expire.

9. Integration with Vehicle Systems:

Offer integration with in-vehicle systems, enabling drivers to interact with the parking system through their vehicle's dashboard.

10. Access Control:

Implement access control mechanisms such as barriers, gates, and automatic license plate recognition (ALPR) systems to control entry and exit to the parking facility.

11. Security:

Ensure the security of user data, payment information, and the system itself to protect against cyber threats and unauthorized access.

12. Reporting and Analytics:

Provide reporting and analytics tools to monitor parking usage, revenue, and operational performance, allowing for data-driven decision-making.

13. Maintenance and Health Monitoring:

Monitor the health of sensors and other equipment to proactively identify and address maintenance needs, reducing downtime.

14. Emergency Procedures:

Establish procedures for handling emergencies, such as vehicle breakdowns or security incidents, to ensure the safety and well-being of users.

15. Scalability:

Design the system to be scalable, allowing for the addition of more sensors or parking spaces as needed to accommodate increased demand.

16. Accessibility:

Ensure that the system is accessible to all users, including those with disabilities, by providing accessible parking spaces and user interfaces.

17. Customer Support:

Offer customer support channels, such as a helpline or chat support, to assist users with any issues or inquiries they may have.

18. Data Storage and Retrieval:

Store historical data related to parking transactions, reservations, and usage patterns for future analysis and reporting.

These functional requirements are essential for the successful implementation and operation of a smart parking system, ensuring a seamless and efficient experience for both parking facility operators and users.

## 4.2 NONFUNCTIONAL REQUIREMENTS

**Non-functional requirements in a smart parking system describe the qualities or attributes that are critical to its performance, security, scalability, and** overall user experience. These requirements ensure that the system meets certain quality standards and constraints. Here are some non-functional requirements for a smart parking system:

1. Performance:

Response Time: Specify the maximum allowable response time for various system operations, such as reserving a parking space or retrieving availability information.

Throughput: Define the system's capacity to handle concurrent user requests and parking transactions.

Scalability: Ensure the system can easily scale to accommodate increased user and parking space demands.

2. Reliability:

Availability: Specify the minimum uptime and availability percentage, ensuring that the system is operational for the majority of the time.

Fault Tolerance: Define how the system should handle hardware or software failures to maintain service continuity.

3. Security:

Data Security: Specify measures to protect user data, payment information, and system data from unauthorized access and breaches.

Authentication and Authorization: Describe the authentication and authorization mechanisms to ensure only authorized users access the system.

Privacy: Ensure that user privacy is maintained, especially when collecting and storing personal information.

4.Usability:

User Interface Design: Define user interface design standards to ensure the application is user-friendly and easy to navigate.

Accessibility: Ensure the system is accessible to individuals with disabilities, complying with accessibility standards.

5. Compliance:

Regulatory Compliance: Specify any legal or industry-specific regulations that the system must adhere to, such as data protection laws or accessibility standards.

6. Interoperability:

Integration: Specify the ability of the system to integrate with other relevant systems, such as traffic management systems, payment gateways, or navigation services.

7. Maintainability:

Modularity: Define how the system is organized into modules or components to facilitate updates and maintenance.

Documentation: Ensure comprehensive and up-to-date documentation for system maintenance and troubleshooting.

8. Scalability:

Horizontal Scalability: Define the ability of the system to scale out by adding more servers or nodes to handle increased load.

Vertical Scalability:Specify how the system can scale up by upgrading individual components to handle increased demands.

9. Performance Testing:

Load Testing: Specify the testing procedures to validate the system's performance under various loads and concurrent user interactions.

10. Disaster Recovery and Backup:

Define procedures and requirements for data backup and disaster recovery, including off-site backups and recovery time objectives (RTO).

11. Environmental Considerations:

Specify any environmental constraints, such as operating temperatures or weather resistance, if applicable to the physical components of the system.

12. Cost Constraints:

Define cost constraints related to system development, maintenance, and operation, considering budget limitations.

13. Geographic Considerations:

Specify any geographical constraints or requirements, such as location-specific regulations or the need for multiple language support.

14. Network Requirements:

Define the network requirements, including the necessary bandwidth, network protocols, and security measures.

15. Redundancy:

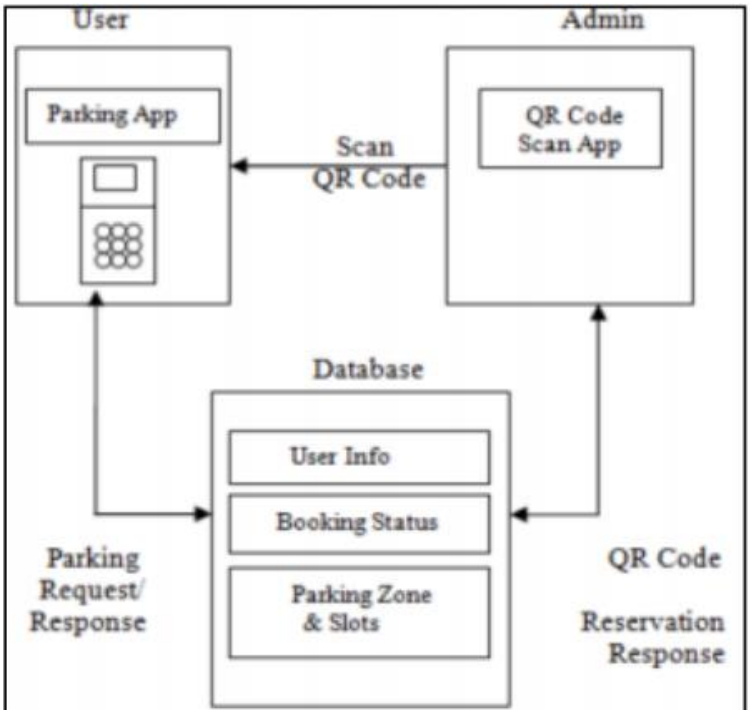
Specify the level of redundancy required for critical components to ensure system availability and minimize downtime.

These non-functional requirements are essential for shaping the quality and performance of a smart parking system and help ensure it meets the expectations of users and stakeholders while maintaining security and reliability.

# CHAPTER 5

# PROJECT DESIGN

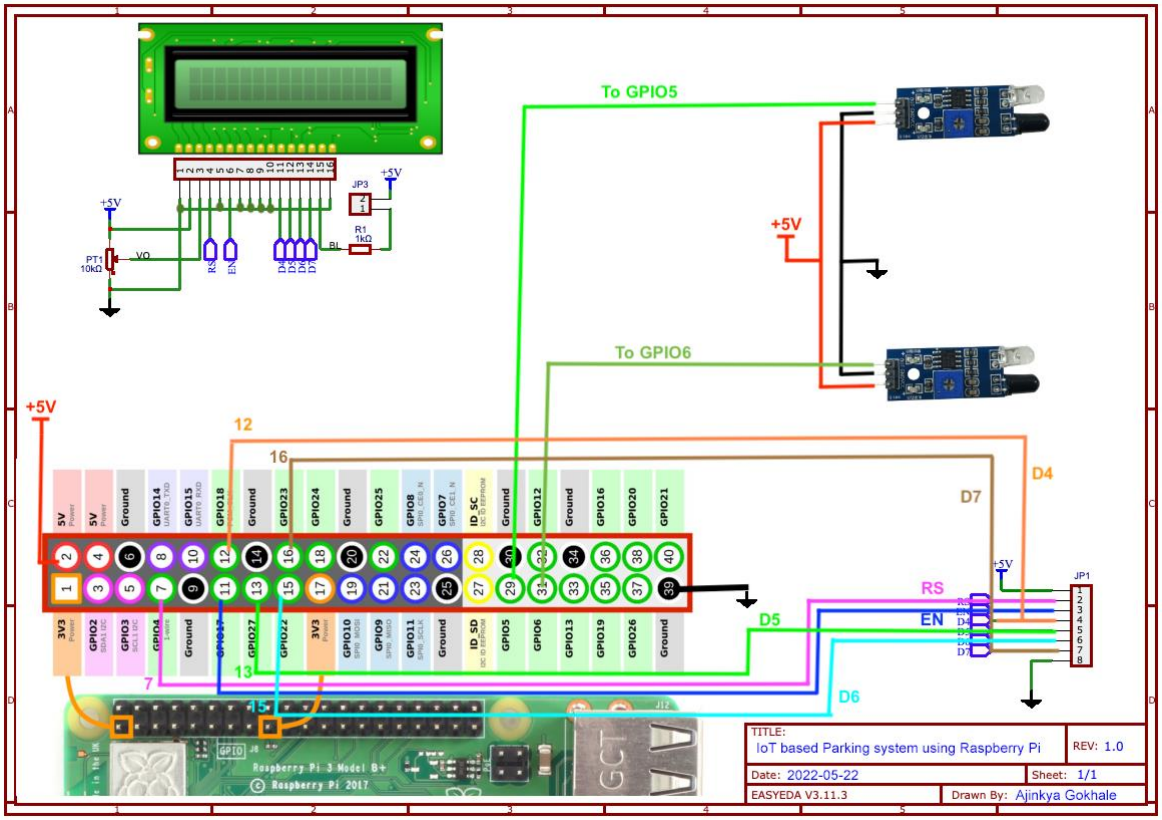
## 5.1 DATA FIOW DIAGRAM




## 5.2 SOLUTION AND TECHNICAL ARCHITECTURE

In a smart parking system, the solution architecture typically involves a combination of hardware and software components. Sensors, cameras, and IoT devices detect parking space occupancy and relay data to a central server. The server processes and stores this information, which is then accessible to users through mobile apps or web interfaces. Payment gateways handle transactions, and navigation systems guide users to available spots. Security protocols, cloud-based storage, and data analytics are essential technical components. The architecture aims to optimize parking efficiency, enhance user experience, and ensure data security.

## 5.3 CIRCUIT DIAGRAM




# 6.DESCRIPTION OF THE PROJECT

## 6.1 PROJECT OBJECTIVES

The project objectives in a smart parking initiative include optimizing parking space utilization, reducing congestion, and improving user experience. It aims to provide real-time parking availability information, enable reservation and payment options, enhance navigation within parking facilities, and ensure data security. Additionally, the project seeks to reduce environmental impact by minimizing unnecessary vehicle circulation. Ultimately, the objectives are to streamline parking operations, increase user convenience, and contribute to more efficient urban mobility while leveraging technology to create a smarter and more sustainable parking ecosystem.

## 6.2 IOT SENSOR PLATFORM

An IoT sensor platform in smart parking involves deploying a network of sensors throughout parking facilities to monitor and manage parking spaces efficiently. These sensors detect vehicle presence and occupancy status, transmitting real-time data to a central server. They can use technologies like ultrasonic sensors, cameras, or in-ground sensors. The platform also often includes wireless communication protocols (e.g., LoRa or Wi-Fi) for data transmission and a cloud-based infrastructure for data storage and processing.

The goal is to provide accurate and up-to-date parking information, enabling users to find available spots easily and improving parking facility management.

### 6.3 IOT SENSOR DEVELOPMENT

IoT sensor development in smart parking involves creating and deploying sensors designed to monitor parking space occupancy and provide real-time data. These sensors use various technologies like ultrasonics, magnetometers, or cameras to detect vehicle presence. They are typically integrated with wireless communication modules (e.g., LoRa or Wi-Fi) for data transmission. The development process includes hardware design, firmware development, and integration with cloud-based platforms. These sensors play a crucial role in improving parking efficiency, reducing congestion, and enhancing the overall smart parking experience for both users and operators.

### 6.4 MOBILE APP DEVELOPMENT

Mobile app development in smart parking focuses on creating user-friendly applications that allow drivers to easily find, reserve, and pay for parking spaces. These apps typically incorporate real-time parking availability information, navigation features, and secure payment gateways. Development involves designing an intuitive user interface, ensuring compatibility with various mobile platforms (iOS, Android), and integrating with back-end systems that manage parking space data. The goal is to enhance user convenience, reduce search time, and streamline the parking process, contributing to a more efficient and connected smart parking ecosystem.

# 7.CODING AND SOLUTIONING

### 7.1 PYTHON SCRIPT

```
import random

import time

parking_spots = [0, 0, 0, 0, 0]

def get_parking_status():

 return [random.choice([0, 1]) for _ in range(len(parking_spots))]

while True:

  parking_spots = get_parking_status()
```

```
  print("Sending data to the cloud:", parking_spots)

 time.sleep(10)  # Simulated data update every 10 seconds
```

7.2 SOLUTIONING

OUTPUT:

Sending data to the cloud: [0, 1, 0, 1, 0]

Sending data to the cloud: [0, 0, 1, 0, 1]

Sending data to the cloud: [1, 0, 1, 0, 0]

Sending data to the cloud: [1, 1, 0, 1, 0]

Sending data to the cloud: [0, 1, 1, 0, 1]

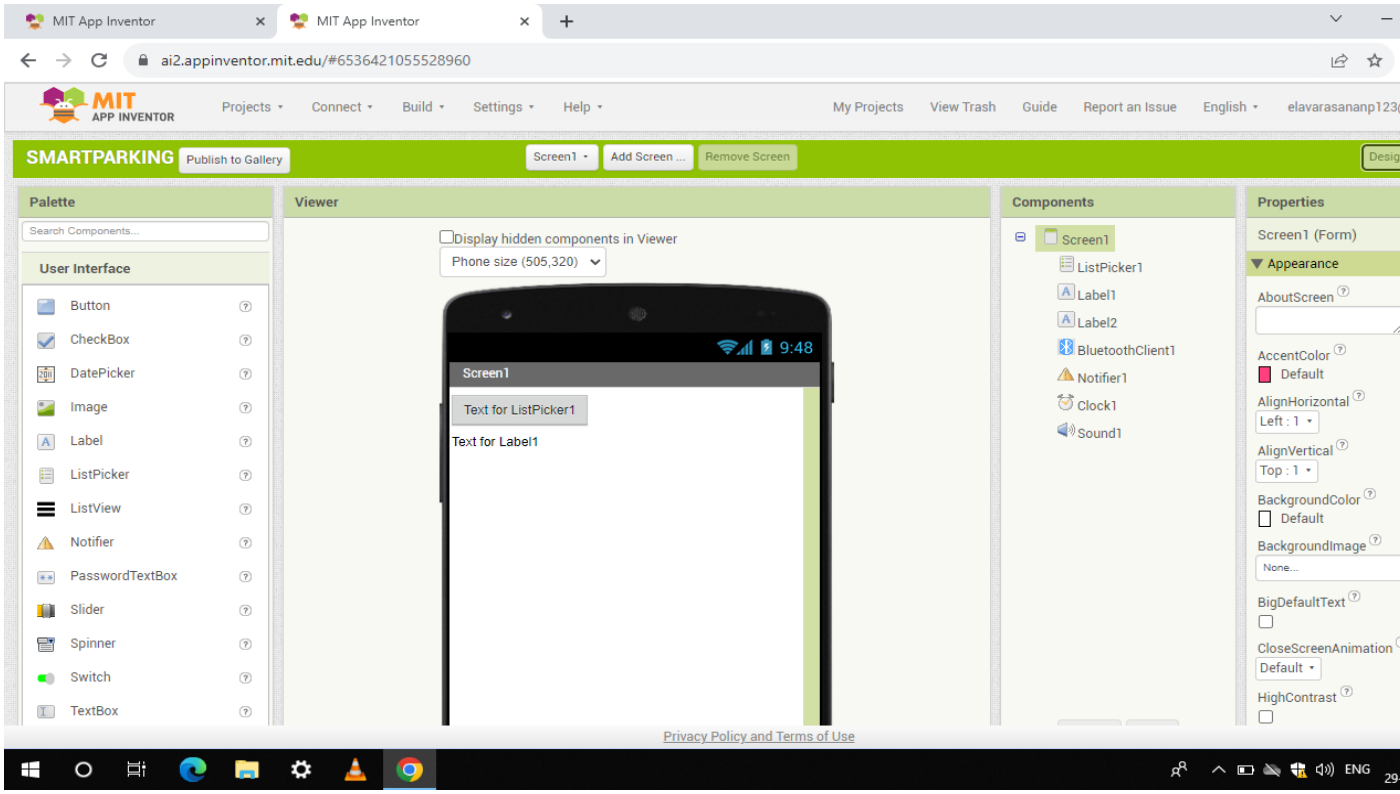
The provided code simulates a basic smart parking system. It randomly generates the occupancy status (0 for vacant, 1 for occupied) of five parking spots and sends this data to the cloud every 10 seconds. Here's how the output would look:The code continuously updates the parking spot occupancy status with random values and prints them to the console every 10 seconds.

# 8.SCREENSHOTS OF USER INTERFACE

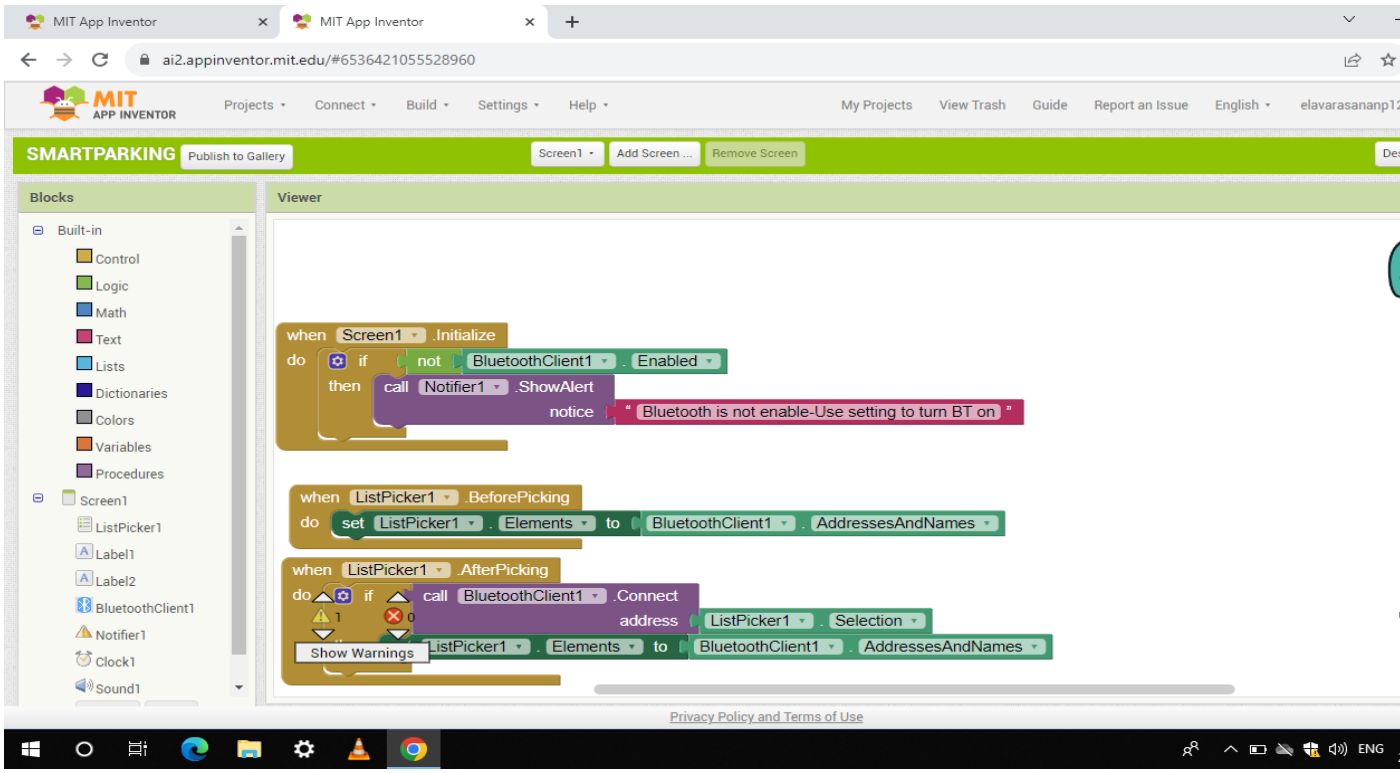
8.1 USER INTERFACE

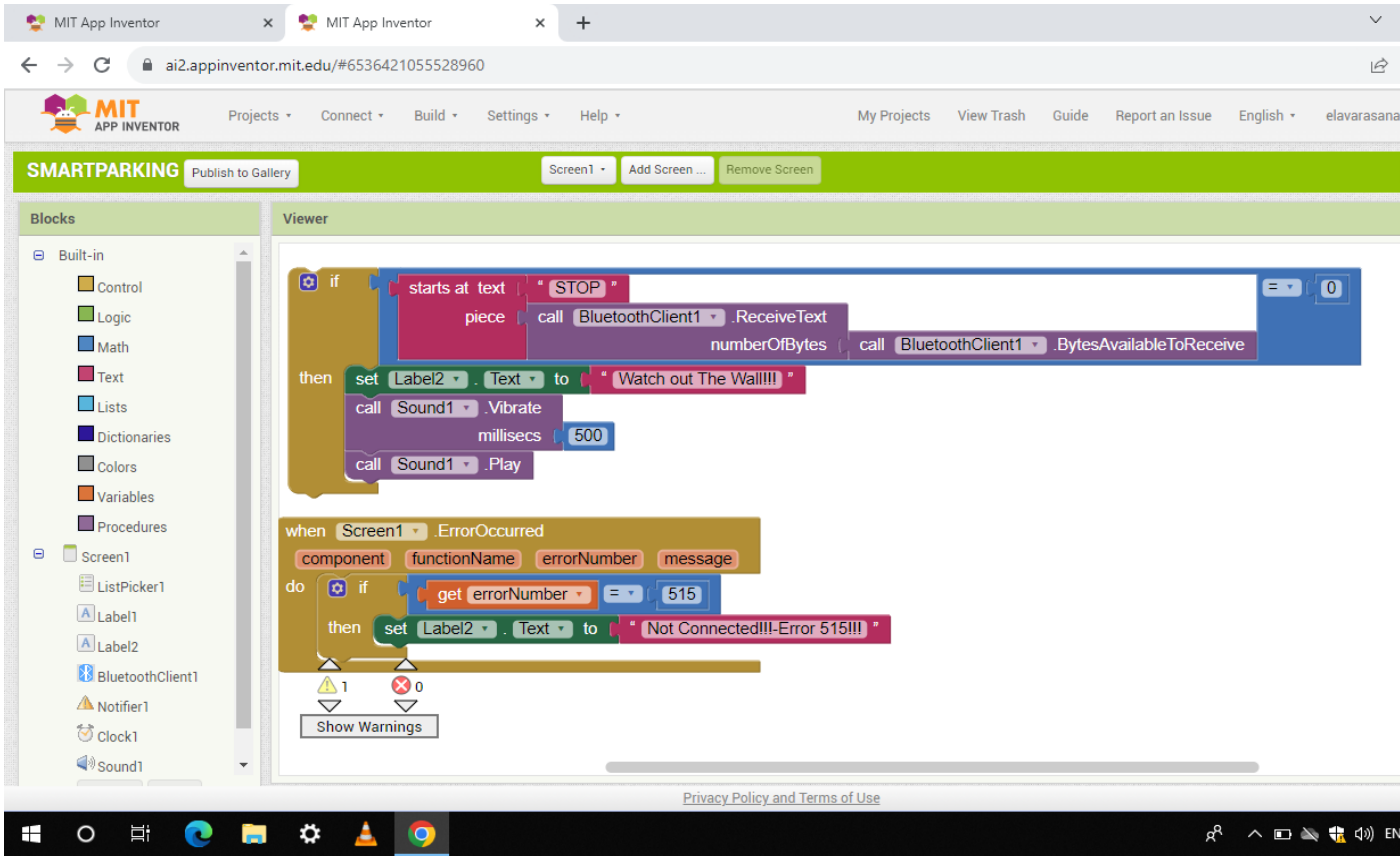
The user interface (UI) in a smart parking system is a crucial component that provides an interactive platform for users to access and utilize parking services. It typically includes a mobile app or web application that offers features such as real-time parking space availability, reservation options, navigation guidance, and secure payment methods. The UI should be intuitive, user-friendly, and responsive, allowing users to easily find and manage parking, ultimately enhancing the overall parking experience and contributing to reduced congestion and improved urban mobility.

8.2 SCREENSHOTS


MIT App Inventor
ai2.appinventor.mit.edu/#6536421055528960
SMARTPARKING
Publish to Gallery
Screen1
Add Screen ...
Remove Screen
Palette
Viewer
Components
Properties
Button
CheckBox
DatePicker
Image
Label
ListPicker
ListView
Notifier
PasswordTextBox
Slider
Spinner
Switch
TextBox
Text for ListPicker1
Text for Label1
Privacy Policy and Terms of Use

## 8.3 BLOCKS


MIT App Inventor
ai2.appinventor.mit.edu/#6536421055528960
SMARTPARKING
Publish to Gallery
Blocks
Viewer
when Screen1 .Initialize
do if not BluetoothClient1 . Enabled
then call Notifier1 .ShowAlert
notice "Bluetooth is not enable-Use setting to turn BT on"
when ListPicker1 .BeforePicking
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
when ListPicker1 .AfterPicking
call BluetoothClient1 .Connect
address ListPicker1 . Selection
Show Warnings
ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
Privacy Policy and Terms of Use




# 9.PUBLIC AWARENESS

## 9.1 PUBLIC AWARNESS

Public awareness of smart parking is essential to maximize its benefits. Initiatives such as informational campaigns, community outreach, and media coverage can educate the public about the advantages of smart parking systems. These benefits include reduced traffic congestion, decreased fuel consumption, lower emissions, and enhanced urban mobility. Public awareness efforts can also highlight how smart parking improves the overall quality of life, reducing the stress associated with finding parking spaces. To gain public support, it is crucial to communicate the environmental, economic, and convenience-related advantages, encouraging users to adopt smart parking solutions and contribute to more sustainable and efficient urban environments.

# 10.CONTRIBUTES OF SMART PARKING MITIGATION

## 10.1 CONTRIBUTES OF SMART PARKING

Smart parking systems contribute significantly to urban efficiency and sustainability. They reduce traffic congestion and emissions by helping drivers quickly locate available parking spaces, minimizing the time spent searching for parking. This leads to lower fuel consumption and improved air quality. Additionally, these systems optimize parking space utilization, maximizing revenue for parking operators. Real-time data collection and analysis enable better urban planning decisions, improving traffic flow and city infrastructure. The convenience of reservation and payment through mobile apps enhances the overall user experience, making cities more accessible and user-friendly. Ultimately, smart parking systems play a vital role in creating greener, more efficient, and livable urban environments.

# 11.ADVANTAGES & DISADVANTAGES

## 11.1 ADVANTAGES

- Smart parking offers numerous advantages, including reduced congestion, enhanced urban mobility, and environmental benefits by minimizing circling for parking.
- It optimizes space utilization, increasing revenue for parking operators. Real-time data aids in traffic management and planning, improving overall city infrastructure.
- Reservation and payment via mobile apps enhance user convenience, while data-driven insights facilitate urban development decisions. Smart parking contributes to greener, more efficient, and accessible urban environments.

## 11.2 DISADVANTAGES

- Smart parking systems come with several potential disadvantages.
- These include the high initial setup costs for hardware and software integration.
- Maintenance and technical issues may lead to downtime or service disruptions.
- User privacy and data security concerns arise due to the collection of personal information.

- Additionally, some users may find the reliance on technology challenging, leading to barriers in adoption.
- Lastly, a loss of jobs in traditional parking management roles may result from automation.
- Balancing the advantages and disadvantages is essential for successful implementation.

# 12.CONCLUSION

In conclusion, smart parking systems offer a promising solution to the pervasive urban challenge of parking inefficiency. These systems, powered by IoT technology and data-driven insights, provide numerous benefits, including reduced congestion, enhanced user experience, and improved environmental sustainability. By optimizing space utilization and streamlining the parking process, they contribute to more efficient urban mobility. However, smart parking also faces challenges such as initial implementation costs and privacy concerns. Striking the right balance between the advantages and disadvantages is key to ensuring that these systems fulfill their potential and continue to evolve as a crucial component of smarter, greener, and more accessible urban environments.

# 13.FUTURE SCOPE

The future scope for smart parking is promising, with ongoing advancements in technology and urban planning. These systems will increasingly integrate with autonomous vehicles, allowing for seamless parking and drop-off/pick-up experiences. Artificial intelligence and machine learning will improve predictive analytics for parking availability. Moreover, expanding the use of renewable energy sources for charging stations and sensor networks can make smart parking more sustainable. Smart cities will continue to adopt these solutions, reducing traffic congestion, emissions, and overall urban stress. The future will likely see enhanced user interfaces, increased accessibility, and more efficient space management, making smart parking an integral part of urban transportation systems.

# APPENDIX

# SOURCE CODE

RASPBERRY PI INTEGRATION

```
import time
import RPi.GPIO as GPIO
import time
import os,sys
from urllib.parse import urlparse
import paho.mqtt.client as paho
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
define pin form
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1
LCD_RS = 7
LCD_E  = 11
LCD_D4 = 12
LCD_D5 = 13
LCD_D6 = 15
LCD_D7 = 16
slot1_Sensor = 29
slot2_Sensor = 31
GPIO.setup(LCD_E, GPIO.OUT)  # E
```

```
GPIO.setup(LCD_RS, GPIO.OUT) # RS
GPIO.setup(LCD_D4, GPIO.OUT) # DB4
GPIO.setup(LCD_D5, GPIO.OUT) # DB5
GPIO.setup(LCD_D6, GPIO.OUT) # DB6
GPIO.setup(LCD_D7, GPIO.OUT) # DB7
GPIO.setup(slot1_Sensor, GPIO.IN)
GPIO.setup(slot2_Sensor, GPIO.IN)
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line
LCD_LINE_3 = 0x90# LCD RAM address for the 3nd line
def on_connect(self, mosq, obj, rc):
        self.subscribe("Fan", 0)
    def on_publish(mosq, obj, mid):
    print("mid: " + str(mid))
mqttc = paho.Client()
mqttc.on_connect = on_connect
mqttc.on_publish = on_publish
url_str = os.environ.get('CLOUDMQTT_URL', 'tcp://broker.emqx.io:1883')
url = urlparse(url_str)
mqttc.connect(url.hostname, url.port)
```

Function Name :lcd_init()

Function Description : this function is used to initialized lcd by sending the different commands

```
def lcd_init():
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)
```

Function Name :lcd_byte(bits ,mode)

Fuction Name :the main purpose of this function to convert the byte data into bit and send to lcd port

```
def lcd_byte(bits, mode):
GPIO.output(LCD_RS, mode) # RS
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
    GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
    GPIO.output(LCD_D5, True)
```

```
  if bits&0x40==0x40:
    GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
    GPIO.output(LCD_D7, True)
lcd_toggle_enable()
 GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x01==0x01:
    GPIO.output(LCD_D4, True)
  if bits&0x02==0x02:
    GPIO.output(LCD_D5, True)
  if bits&0x04==0x04:
    GPIO.output(LCD_D6, True)
  if bits&0x08==0x08:
    GPIO.output(LCD_D7, True)
  lcd_toggle_enable()
```

Function Name : lcd_toggle_enable()

Function Description:basically this is used to toggle Enable pin

```
def lcd_toggle_enable():
 time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
```

```
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)
Function Name :lcd_string(message,line)
Function  Description :print the data on lcd
def lcd_string(message,line):
 message = message.ljust(LCD_WIDTH," ")
  lcd_byte(line, LCD_CMD)
   for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(0.5)
lcd_string("Car Parking ",LCD_LINE_1)
lcd_string("System ",LCD_LINE_2)
time.sleep(0.5)
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
delay = 5
while 1:
    rc = mqttc.loop()
  slot1_status = GPIO.input(slot1_Sensor)
  time.sleep(0.2)
  slot2_status = GPIO.input(slot2_Sensor)
```

```
time.sleep(0.2)
if (slot1_status == False):
 lcd_string("Slot1 Parked  ",LCD_LINE_1)
 mqttc.publish("slot1","1")
 time.sleep(0.2)
else:
  lcd_string("Slot1 Free  ",LCD_LINE_1)
  mqttc.publish("slot1","0")
  time.sleep(0.2)
   if (slot2_status == False):
 lcd_string("Slot2 Parked  ",LCD_LINE_2)
 mqttc.publish("slot2","1")
 time.sleep(0.2)
else:
  lcd_string("Slot2 Free  ",LCD_LINE_2)
  mqttc.publish("slot2","0")
  time.sleep(0.2)
```