Mohammad Ali Jinnah University

Department of Computer Systems Engineering

SMART PARKING REGISTERATION SYSTEM

MORTAZA HASSANI	-	FA19-BECE-0007
ABBAS PARWAZ	-	FA19-BECE-2004
JIBRAN AHMED	-	FA19-BECE-0004
ARSH AFAQ	-	FA19-BECE-0008

Project report submitted to Prof. NAUMAN HAFEEZ ANSARI in partial fulfilment of (DCN) lab project in the semester of (FA11 21)

Table of Contents

INTRODUCTION:

The challenge of parking is particularly important, as most people prefer private car ownership, something that is deeply ingrained in the daily routines of many of us [6]. For this reason, the search for a parking spot in busy towns and cities is a daunting endeavor, leading to time wastage and unwarranted consumption of fuel, and, importantly, contributing to climate change . The issue of parking is significant to the point that it has been raised in discussions of climate change mitigation as well as in political arenas . In the new era, where technology has been accepted as one of the most critical tools for solving some of the challenges faced in the 21st century, especially in urban areas, the issue of parking has not been left behind. With the adoption of the Smart City model in most urban areas, smart digital solutions have emerged. Among them is the smart parking system. [11], may be instrumental in bringing order and sanity to parking lots. According to them, the smart parking system could be customized to combine both technology and human innovations in order to optimize the utilization of scarce resources such as fuel, time, and space.

Abstract:

The Internet of Things (IoT) has come of age, and complex solutions can now be implemented seamlessly within urban governance and management frameworks and processes. For cities, growing rates of car ownership are rendering parking availability a challenge and lowering the quality of life through increased carbon emissions. The development of smart parking solutions is thus necessary to reduce the time spent looking for parking and to reduce greenhouse gas emissions. The principal role of this research paper is to analyze smart parking solutions from a technical perspective, underlining the systems and sensors that are available, as documented in the literature. The review seeks to provide comprehensive insights into the building of smart parking solutions. A holistic survey of the current state of smart parking systems should incorporate the classification of such systems as big vehicular detection technologies. Finally, communication modules are presented with clarity

Spaces allow users to request the application layer, wherein the request will immediately be processed through a network layer [20]. As a way of handling the user request, parking providers are expected to utilize the network layer to process the interaction with the transaction layer Finally, the transaction layer's consensus mechanism protocol and the individual parking provider update the distributed ledger

The Architecture of Smart Parking Systems A smart parking system is an architectural framework that comprises different application platforms integrated into embedded systems. For instance, reserved parking Spaces allow users to request the application layer, wherein the request will immediately be processed through a network layer [20]. Finally, the transaction layer's consensus mechanism protocol and the individual parking provider update the distributed ledger.

Objectives

SPARKAU will turn out to be a time-saver application. It will help many university members in the coming days. Students, faculty members, and admin members of our university will use it. The key objectives of this project are

*To help the people of our university to find a free parking slot

* To solve the daily routine issue of looking for free slots.

*To save the time and fuel of university members To help the university members get rid of this useless frustration caused by the parking problem. ¬ To research on computer vision and enhance our knowledge

Advantages

The main objective of this project is to ease the university students and faculty members in finding a free slot to park their cars It is a smart parking system based on an algorithm that works on the principles of computer vision. The Advantages of the project are given below

*Timesaving: It would save time for both drivers and the administration to search for the available space for the car park.

* Cost-effective: The system does not cost any hardware as it computes the availability of parking slots using this device.

*Fully automated. It is a very efficient way to handle car parking issues rather than human monitoring Timesaving means it would save the time of the drivers to park their cars. Many times faculty and as well as students are getting late for the class. They are in a hurry to park their car. They immediately enter into a parking area to search for every possible slot, but most of the time, they cannot find one. It often gets disappointing and frustrating. It is a cost-effective product. It does not require any hardware and sensors. It increases the overall cost of the product. The maintenance of this hardware would not cost more money.

HARDWWARE:

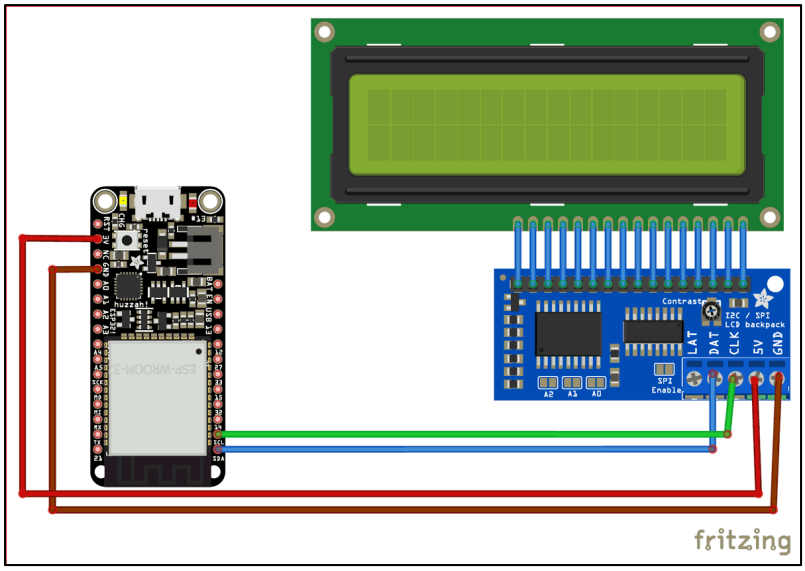
LCD

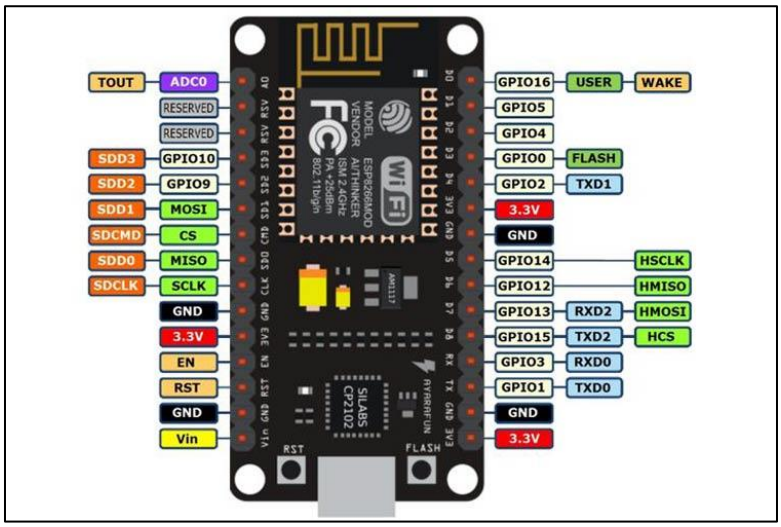
NODE MCU

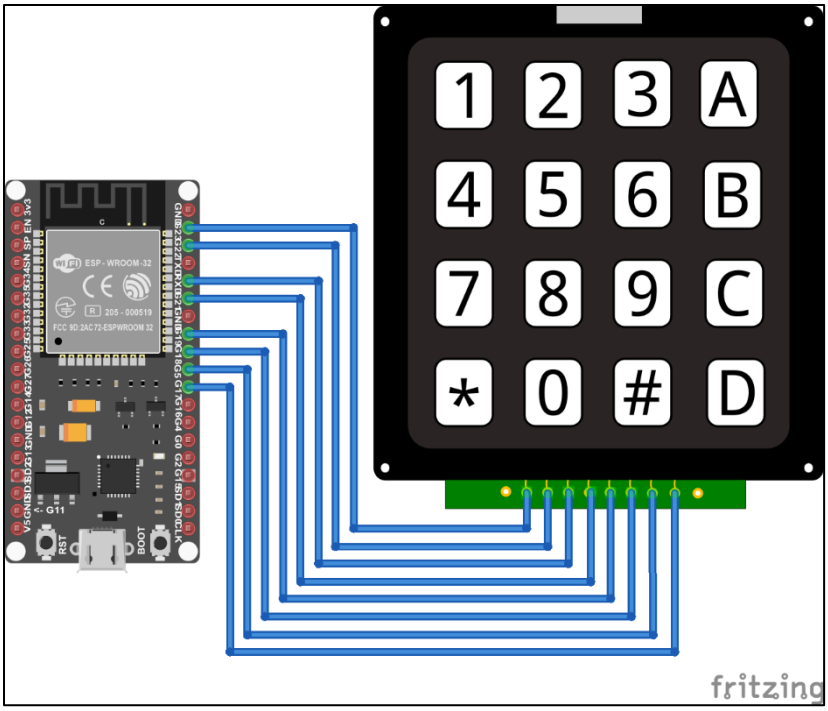
BATTERY

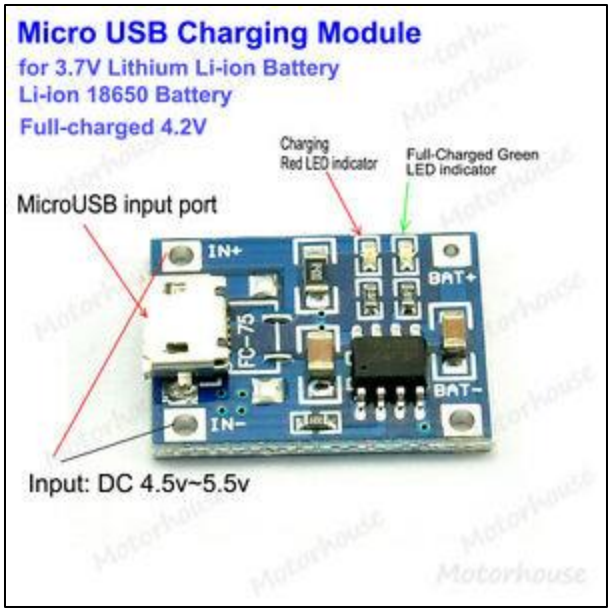
KEYPAD

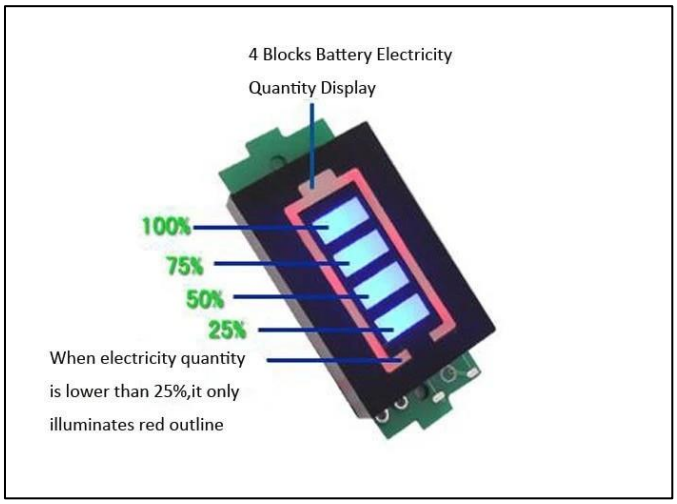
BATTERY INDICATOR

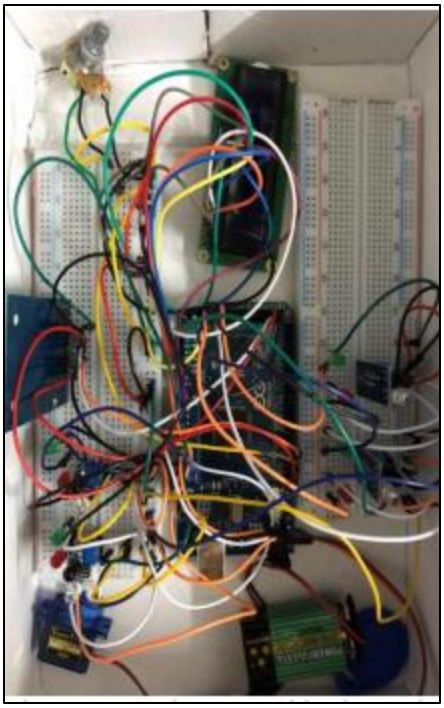





1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D
ESP - WROOM -32
RST
BOOT
fritzing


Micro USB Charging Module
for 3.7V Lithium Li-ion Battery
Li-ion 18650 Battery
Full-charged 4.2V
Charging
Red LED indicator
Full-Charged Green
LED indicator
MicroUSB input port
IN+
IN-
BAT+
BAT-
FC-75
Input: DC 4.5v~5.5v


4 Blocks Battery Electricity
Quantity Display
100%
75%
50%
25%
When electricity quantity
is lower than 25%,it only
illuminates red outline



Hardware used in the Project

Prototype

ESP8266 Code:

```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>
//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and
other helper functions.
#include "addons/RTDBHelper.h"
//Date & Time
#include <NTPClient.h>
#include <WiFiUdp.h>
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);
String formattedDate;
String dayStamp;
String timeStamp;

// Insert your network credentials
#define WIFI_SSID "MAJU-BELL"
#define WIFI_PASSWORD "847549632548"

//#define WIFI_SSID "DevSol2"
//#define WIFI_PASSWORD "M@maju0007$"

// Insert Firebase project API Key
#define API_KEY
"AIzaSyCWqoGDAMXns6JnQcFJk3XJLFbraCZ
LoKs"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://maju-
smartparking-default-rtdb.firebaseio.com/"
//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
int count = 0;
bool signupOK = false;

#include <Keypad.h>
#include <LiquidCrystal_I2C.h>

#include <Wire.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);


const byte n_rows = 4;
const byte n_cols = 4;

char keys[n_rows][n_cols] = {
 {'1','2','3','A'},
 {'4','5','6','B'},
 {'7','8','9','C'},
 {'*','0','#','D'}
};

byte rowPins[n_rows] = {3, 1, D3, D4};
byte colPins[n_cols] = {D5, D6, D7, D8};

Keypad myKeypad = Keypad(
makeKeymap(keys), rowPins, colPins, n_rows,
n_cols);

//
char Data[7];
byte data_count = 0;
char IDC[12];
byte id_count = 0;
char Plate[7];
byte Plate_count=0;



void setup(){
 Serial.begin(115200);
Wire.begin(D2, D1);   //Use predefined PINS
consts
lcd.begin(20,4);      // The begin call takes the
width and height. This
                      // Should match the number provided
to the constructor.

lcd.backlight();      // Turn on the backlight.

lcd.home();

lcd.setCursor(0, 0);  // Move the cursor at origin
lcd.print("MAJU SMART Parking");

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  lcd.setCursor(0,1);
  lcd.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED){
    lcd.setCursor(0,2);
    lcd.print(".");
    lcd.setCursor(0,3);
    lcd.print(WIFI_SSID);
```

```
    delay(300);
  }
  Serial.println();
  clearLCDLine(1);
  clearLCDLine(2);
  clearLCDLine(3);
  lcd.setCursor(0,1);
  lcd.print("Connected with IP:");
  lcd.setCursor(0,2);
  lcd.print(WiFi.localIP());
  delay(1000);

  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;

  /* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")){
  clearLCDLine(1);
  clearLCDLine(2);
  lcd.setCursor(0,1);
    lcd.print("Firebase Connected");
    clearLCDLine(1);
    signupOK = true;
  }
  else{
    Serial.printf("%s\n",
config.signer.signupError.message.c_str());
  clearLCDLine(1);
  clearLCDLine(2);
  lcd.setCursor(0,1);
    lcd.print("Firebase ERROR");
  }
  delay(500);

  /* Assign the callback function for the long
running token generation task */
  config.token_status_callback =
tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);


  timeClient.begin();
  timeClient.setTimeOffset(18000);
    while(!timeClient.update()) {
    timeClient.forceUpdate();
  }

//


  GETF();
}

void loop(){
    formattedDate =
timeClient.getFormattedDate();
  lcd.setCursor(0,3);
  lcd.print(formattedDate);
 char myKey = myKeypad.getKey();

    if (myKey != NULL && data_count !=7){
    lcd.setCursor(0,1);
    lcd.print("Enter Plate Number:");
      Data[data_count] = myKey;
      lcd.setCursor(data_count,2);
      lcd.print(Data[data_count]);
      data_count++;
    }
    delay(100);

    if (data_count == 7 && id_count == 0){
      clearLCDLine(1);
      clearLCDLine(2);
    }

    if (data_count == 7 && myKey != NULL &&
id_count !=12){
    lcd.setCursor(0,1);
    lcd.print("Enter ID Card:");
      IDC[id_count] = myKey;
      lcd.setCursor(id_count,2);
      lcd.print(IDC[id_count]);
      id_count++;
    delay(100);
    }
    if (data_count == 7 && id_count == 12){
      clearLCDLine(1);
      clearLCDLine(2);
      lcd.setCursor(4,2);
      lcd.print("Confirm [ok]?");
      delay(300);
    }
        if (data_count == 7 && id_count == 12 &&
myKey != NULL){
      if(myKey == '#'){
        clearLCDLine(2);
        data_count = 0;
        id_count = 0;
        POSTF();
        delay(500);
        clearLCDLine(2);
      }

    }
}
void clearLCDLine(int line)
```

```
{
      lcd.setCursor(0,line);
      for(int n = 0; n < 20; n++)
      {
            lcd.print(" ");
      }
}
void POSTF ()
{
  char cstr[16];
//  itoa(Data, cstr, 10);
   char sourced[32];


strcpy(sourced,"PDB/");
strcat(sourced,Data);
strcat(sourced,"/plate");


   char sourcedID[32];
strcpy(sourcedID,"PDB/");
strcat(sourcedID,Data);
strcat(sourcedID,"/stuid");

   char sourcedtime[32];
strcpy(sourcedtime,"PDB/");
strcat(sourcedtime,Data);
strcat(sourcedtime,"/intime");

   if (Firebase.ready() && signupOK && (millis()
- sendDataPrevMillis > 15000 ||
sendDataPrevMillis == 0)){
   sendDataPrevMillis = millis();
   Serial.println("------------------------------------");
   Serial.println("Set JSON test...");

   if (Firebase.RTDB.setString(&fbdo, sourcedID,
IDC))
   {
      lcd.setCursor(4,2);
      lcd.print("Database Post");
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
   }
   else
   {
      lcd.setCursor(4,2);
      lcd.print("Database Failed");
      Serial.println("FAILED");
      Serial.println("REASON: " +
fbdo.errorReason());
      Serial.println("------------------------------------
");
      Serial.println();
   }


   // Write an Float number on the database path
test/float
   if (Firebase.RTDB.setString(&fbdo, sourced,
Data)){
     Serial.println("PASSED");
     Serial.println("PATH: " + fbdo.dataPath());
     Serial.println("TYPE: " + fbdo.dataType());
   }
   if (Firebase.RTDB.setString(&fbdo,
sourcedtime, formattedDate)){
     Serial.println("PASSED");
     Serial.println("PATH: " + fbdo.dataPath());
     Serial.println("TYPE: " + fbdo.dataType());
   }
   else {
     Serial.println("FAILED");
     Serial.println("REASON: " +
fbdo.errorReason());
   }
  }
 count++;
}

void GETF ()
{
   if (Firebase.RTDB.getString(&fbdo,
"PDB/MH")) {

       lcd.setCursor(5,2);
     lcd.print(String(fbdo.stringData()).c_str());

  } else {
   lcd.setCursor(5,2);
   lcd.print("GET ERROR");
   Serial.println(fbdo.errorReason());
  }
  delay(1000);
  clearLCDLine(2);
}
```

Outcome

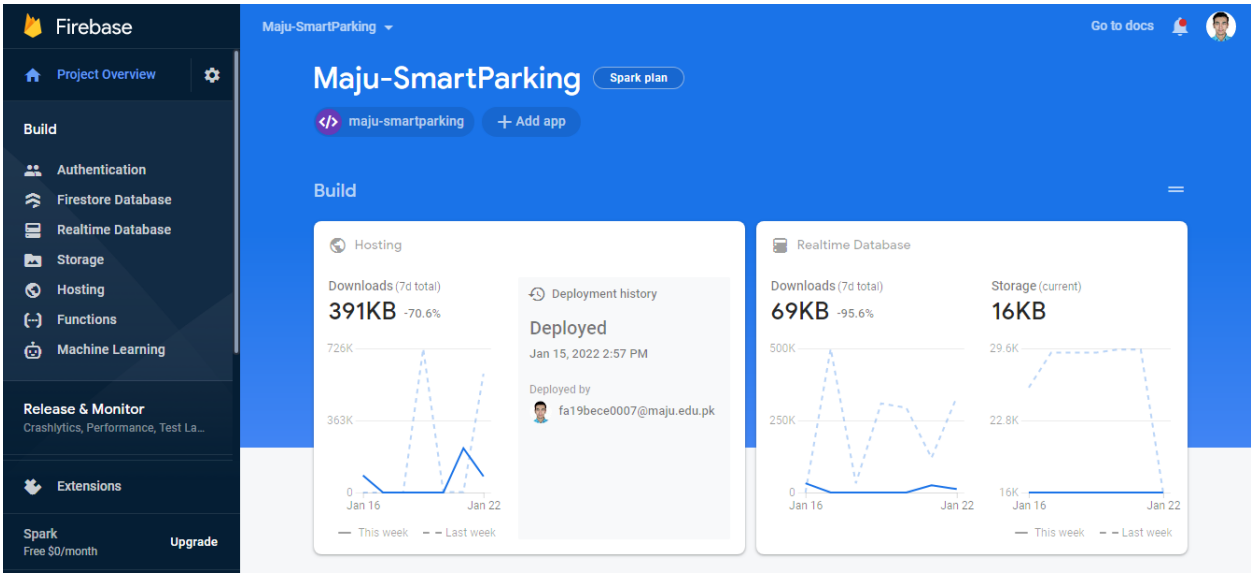
The process flow of the smart parking system when users start initialization through mobile application. Users are required to key-in important details such as name, vehicle's plate number, contact number and duration they want to park for. Once registered, users are taken into the next window where availability of slots based on real time. Red indicates the slots are occupied whereas green indicates free occupancy, thus users can choose to reserve them. Then, user will be sent a unique code which later on they have to scan at the entrance of parking bay within the time given. The unique code differentiates between the mobile users and normal users. Normal users go through the normal ticketing process. Once the car enters in the parking bay, it starts to calculate the time and also parking charges. Else, wait for the car arrival within the time allocated

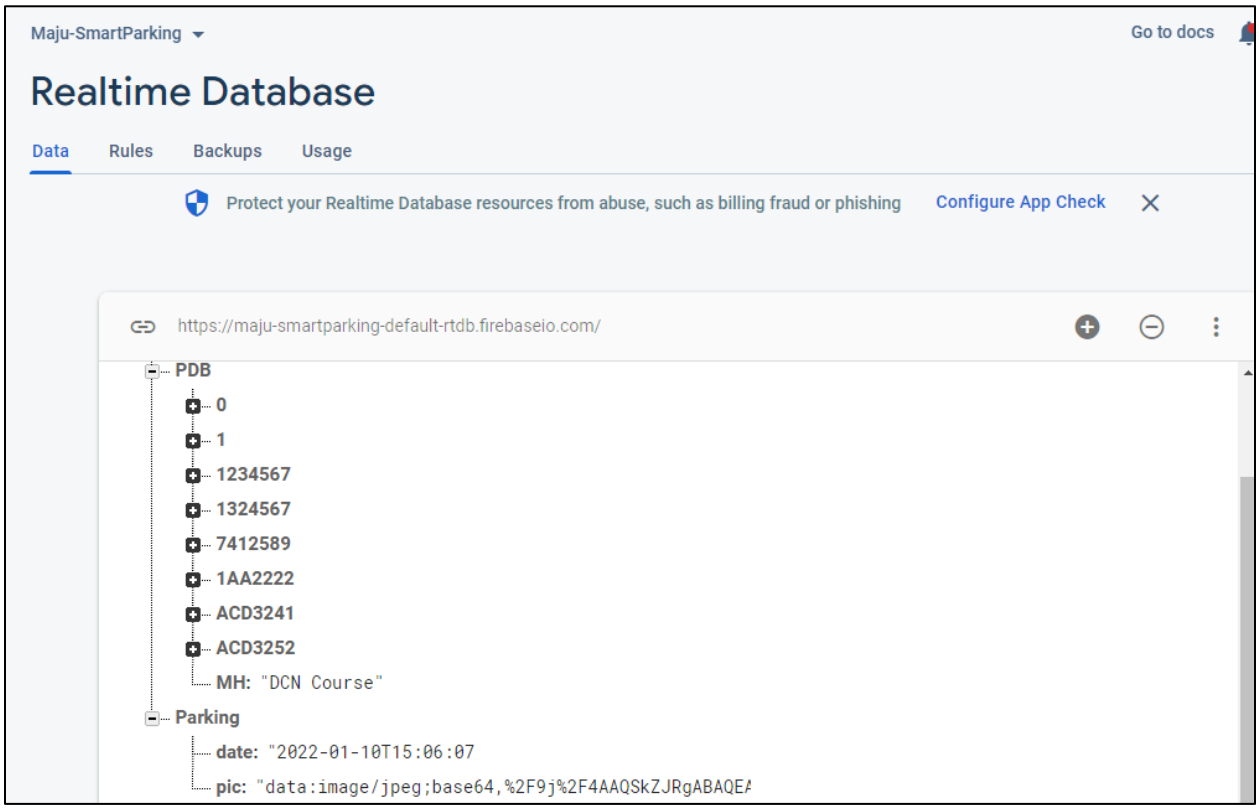
Google Firebase:

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

Realtime Database:

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in Realtime.





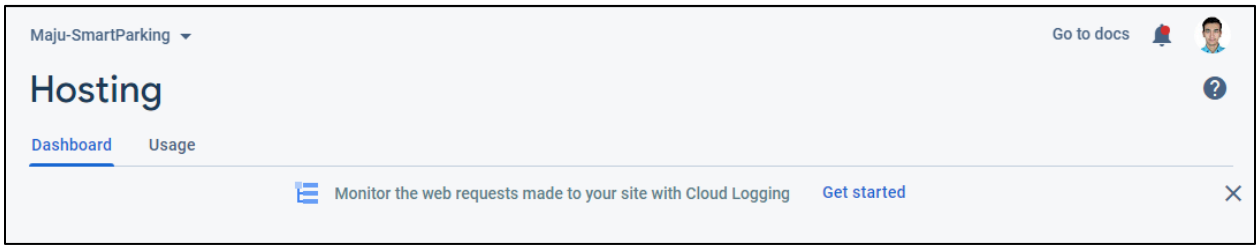
What is Firebase Hosting?

Firebase hosting is a Google hosting service which provides static web content to the user in a secure, fast, free and easy way.

Why Firebase Hosting?

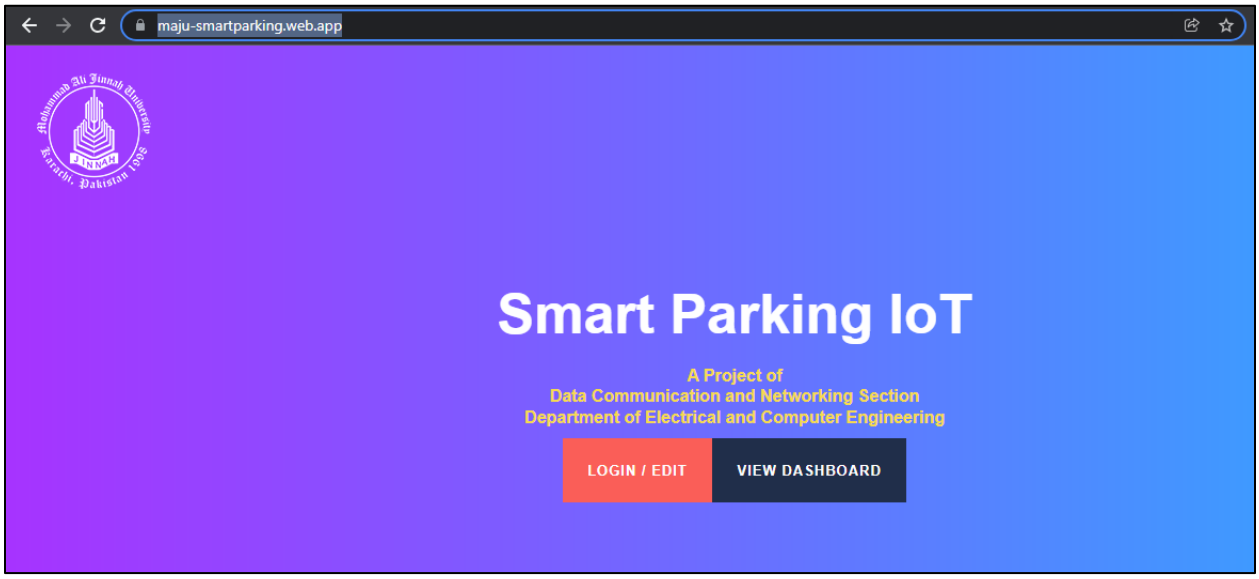
Most of the web hosting will charge you or will be slow if they are free, also you have to pay extra for getting an SSL certificate to convert your website to a secure one with https.

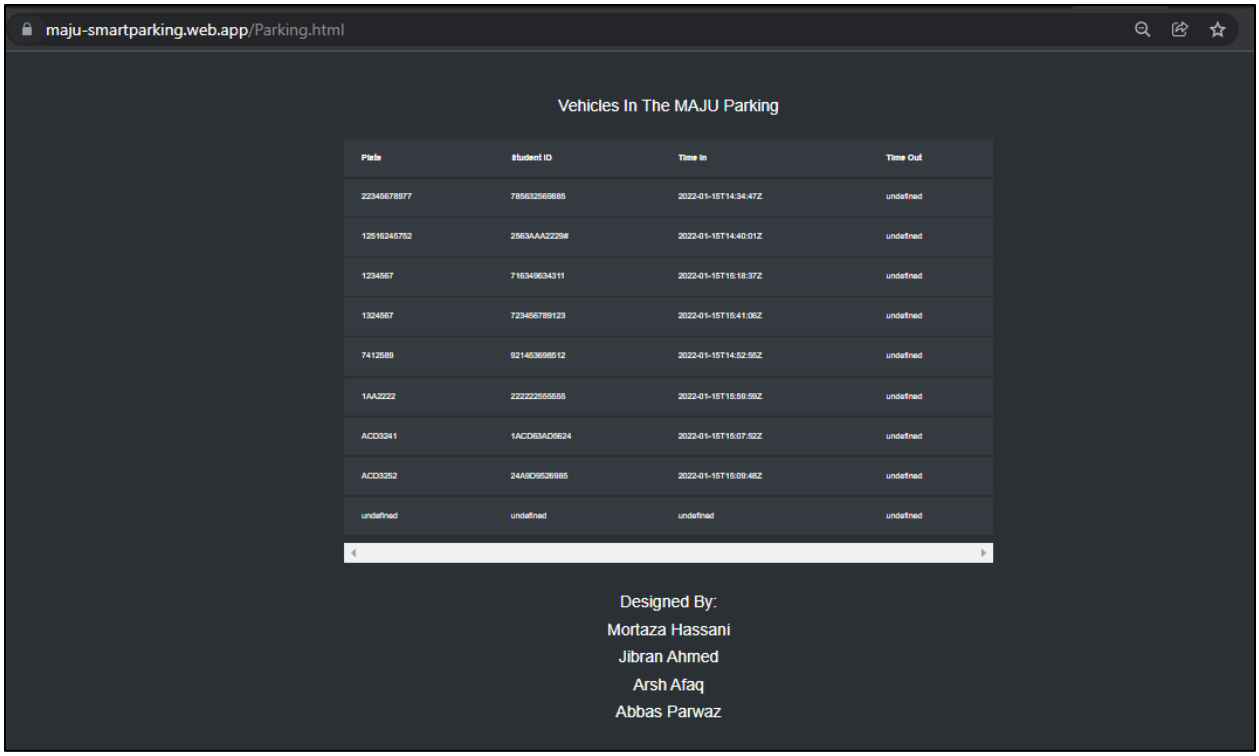
Firebase hosting is free. So, it won't cost you anymore. It by default provides SSL certificate and offers an impressive speed across multiple geographic locations without the need for a separate CDN on top.



Website:

https://maju-smartparking.web.app/





CONCLUSION:

The smart parking system can be realized as a means to solve parking issues both in the current era and in the future. Furthermore, IoT-enabling techniques need to be given maximum attention, ensuring that they are at the center of planning smart parking systems. In line with this, there exist several alternatives to the mechanism of modernizing the setup of parking lots and to the implementation of smart functionality. There are those that can be efficiently installed and there are those that are quite challenging. Regardless, the implementation has to allow drivers to acquire actual information on parking online and on the remaining parking spaces. The parking process in the city has to be addressed fully—in an efficient, real-time, and cost-effective manner