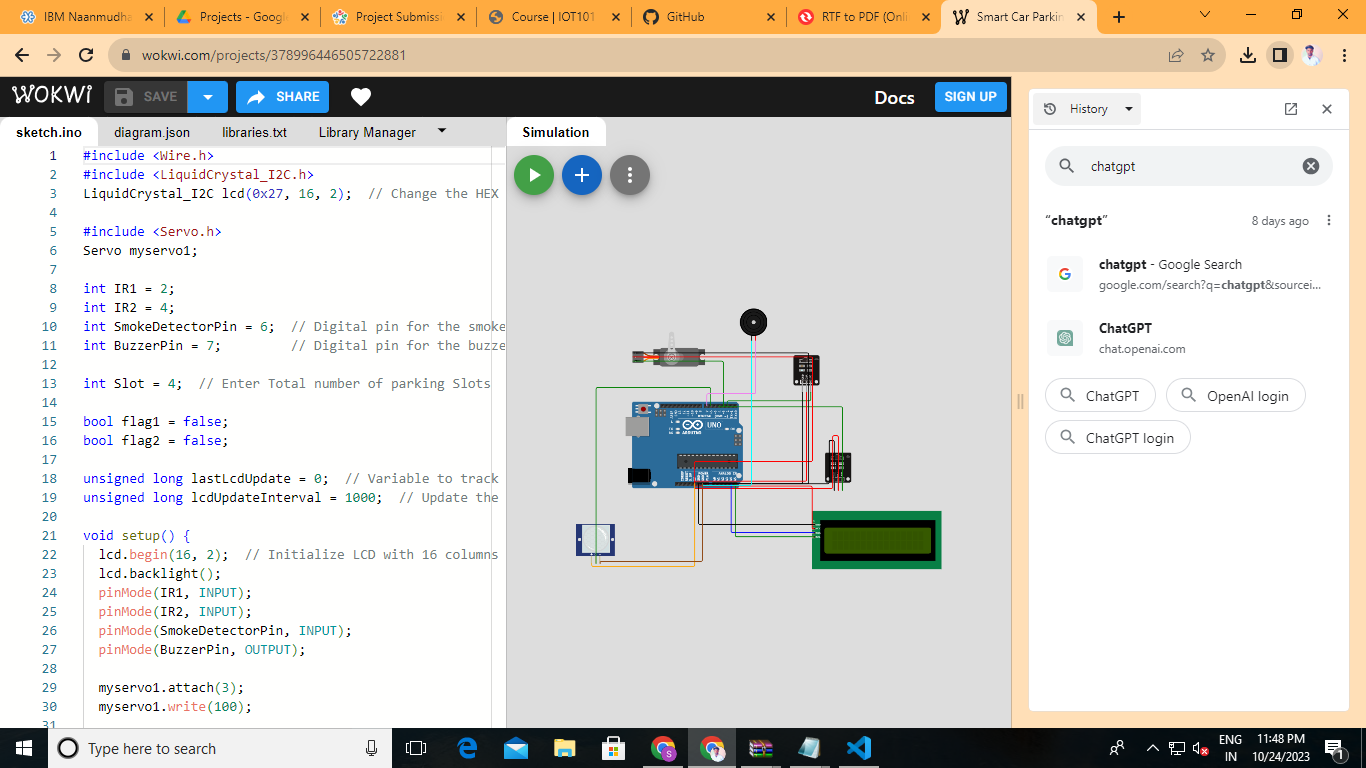
**SMART PARKING**

* Data Collection: Gathering relevant data from various sources, which can include databases, APIs, web scraping, or other means.
* Data Cleaning: Preprocessing and cleaning the data to handle missing values, outliers, and other inconsistencies. This may involve imputation, normalization, and more.
* Exploratory Data Analysis (EDA): Visualizing and analyzing the data to gain insights and understand the relationships between variables. EDA helps in feature selection and engineering.
* Feature Engineering: Creating new features or transforming existing ones to improve the model's performance. This includes one-hot encoding, feature scaling, and creating interaction terms.
* Data Splitting: Splitting the dataset into training, validation, and test sets to train and evaluate the machine learning model. Common splits are 70-30, 80-20, or 90-10.
* Model Selection: Choosing an appropriate machine learning algorithm or model architecture for the task at hand. This may involve considering various factors like the nature of the problem, data size, and available resources.
* Model Training: Training the selected model on the training dataset using optimization techniques such as gradient descent. Hyperparameter tuning may be done to find the best model configuration.
* Model Evaluation: Assessing the model's performance using appropriate metrics, which could include accuracy, precision, recall, F1-score, or others depending on the problem (classification, regression, etc.).
* Model Validation: Fine-tuning the model and validating its performance on the validation dataset. This step may involve adjusting hyperparameters, dealing with overfitting, or choosing different algorithms.
* Model Testing: Evaluating the final model on the test dataset, which it hasn't seen during training or validation. This provides an estimate of the model's generalization performance.
* Model Deployment: Deploying the model into a production environment so that it can make predictions on new, unseen data. This may involve creating APIs or integrating it into a larger system.
* Monitoring and Maintenance: Continuously monitoring the model's performance in the production environment and retraining or updating it as needed to maintain accuracy.
* Interpretability: Understanding the model's decisions and predictions, which is especially important in sensitive domains like healthcare or finance.
* Documentation: Creating documentation for the entire machine learning pipeline, including data sources, preprocessing steps, model architecture, and deployment procedures.
* Reporting and Communication: Communicating the results and insights to stakeholders, which may involve creating reports, presentations, or visualizations.
* Post-Deployment Analysis: Analyzing how the deployed model performs in a real-world setting and making any necessary adjustments.
* These activities are iterative and may be repeated as needed to achieve the desired model performance and maintain its effectiveness over time. The specific tasks and order may vary depending on the project's complexity and requirements.



#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);  // Change the HEX address

#include <Servo.h>

Servo myservo1;

int IR1 = 2;

int IR2 = 4;

int SmokeDetectorPin = 6;  // Digital pin for the smoke detector

int BuzzerPin = 7;         // Digital pin for the buzzer

int Slot = 4;  // Enter Total number of parking Slots

bool flag1 = false;

bool flag2 = false;

unsigned long lastLcdUpdate = 0;  // Variable to track the time of the last LCD update

unsigned long lcdUpdateInterval = 1000;  // Update the LCD every 1000 milliseconds (1 second)

void setup() {

  lcd.begin(16, 2);  // Initialize LCD with 16 columns and 2 rows

  lcd.backlight();

  pinMode(IR1, INPUT);

  pinMode(IR2, INPUT);

  pinMode(SmokeDetectorPin, INPUT);

  pinMode(BuzzerPin, OUTPUT);

  myservo1.attach(3);

  myservo1.write(100);

  lcd.setCursor(0, 0);

  lcd.print("     ARDUINO    ");

  lcd.setCursor(0, 1);

  lcd.print(" PARKING SYSTEM ");

  delay(2000);

  lcd.clear();

**Serial**.begin(9600);  // Start serial communication for debugging

}

void loop() {

  if (digitalRead(IR1) == LOW && !flag1) {

    if (Slot > 0) {

      flag1 = true;

      if (!flag2) {

        myservo1.write(0);

        Slot--;

      }

    } else {

      displayMessage("    SORRY :(    ", "  Parking Full  ");

    }

  }

  if (digitalRead(IR2) == LOW && !flag2) {

    flag2 = true;

    if (!flag1) {

      myservo1.write(0);

      Slot++;

    }

  }

  if (flag1 && flag2) {

    delay(1000);

    myservo1.write(100);

**Serial**.println("Servo returned to initial position.");

    flag1 = false;

    flag2 = false;

  }

  // Update the LCD display with a delay

  if (millis() - lastLcdUpdate >= lcdUpdateInterval) {

    updateLcdDisplay();

    lastLcdUpdate = millis();

  }

  // ... (Rest of your code)

}

void updateLcdDisplay() {

  if (digitalRead(SmokeDetectorPin) == HIGH) {

    displayMessage("   WARNING!   ", " Smoke Detected ");

    digitalWrite(BuzzerPin, HIGH);  // Turn on the buzzer

  } else {

    displayMessage("    WELCOME!    ", "Slot Left: " + String(Slot));

    digitalWrite(BuzzerPin, LOW);   // Turn off the buzzer

  }

}

void displayMessage(const char \*line1, const String &line2) {

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print(line1);

  lcd.setCursor(0, 1);

  lcd.print(line2);

}