Smart Parking Report

1. Introduction:

   The project aimed to develop a smart parking system that can recognize and direct vehicles to specific parking slots using the ESP32 platform.

2. System Components and Overview:

   2.1 **Proximity Sensors：**

   - Used to detect the arrival of a vehicle.
   - proximity sensors were positioned at the entrance of the parking lot.

   2.2 **Camera & Image Recognition:**

   - Cameras were employed to capture images of vehicle license plates (using small game vehicles for simulation).
   - These images were subsequently processed via Python-based image recognition libraries.

   2.3 **Database Storage – Firebase:**

   - The recognized license number is sent to Firebase for storage.
   - Within Firebase, there are two primary collections:
     1. Parking Spots: Information regarding all parking slots and their current occupants.
     2. Registered Vehicles: Details of all vehicles stored in the system.

   2.4 **Smart Parking Server**:

   - Scans all the parking spaces continuously.
   - If a spot is available, the incoming vehicle is directed to that location.
   - An additional camera and set of proximity sensors, at the parking entrance, validate the vehicle's authenticity.

   2.5 **LED Indicators:**

   - Blue: Denotes an available parking space.
   - Yellow: Indicates a parking space that is legally occupied.
   - Red: Represents an illegally occupied spot, i.e., a vehicle that hasn't been assigned to that particular spot.

3. System Workflow:

i. **Vehicle Arrival:** Proximity sensors, detect the incoming vehicle at the entrance.

ii. **License Plate Recognition:** The camera snaps the license plate, and its number is processed using Python-driven image recognition techniques.

iii. **Database Update:** The identified license number is transmitted and stored in Firebase.

iv. **Parking Allocation:** The server reviews the parking area for any free slots. When identified, the vehicle is directed accordingly.

v. **Vehicle Verification:** After reaching the assigned parking slot, a second round of verification is conducted using the camera and proximity sensors to ensure the right vehicle occupies the spot.

vi. **LED Status Indication:** Depending on the vehicle's parking status, LEDs (Blue/Yellow/Red) display the spot's current occupancy status.

vii. **Vehicle Departure:** Upon a vehicle's departure, the proximity sensors note the change, prompting the server to update Firebase. The associated LED reverts to blue, signaling a vacant slot.

4. Implementation Details:

- **Hardware:** The ESP32, along with connected peripherals like sensors, cameras, and LEDs, was programmed using the Arduino IDE.
- **Software:** Image recognition and server was executed with Python libraries, with Firebase serving as the backend database.

5. Conclusion:

The smart parking system efficiently administers parking spots through a blend of proximity sensors, image recognition, and real-time database updates. This fusion ensures optimal space utilization and offers users a smooth parking experience.