**Smart Car Parking System - Simulation and Cloud Integration**

**1. Introduction** As part of our B.Tech IT curriculum, we are collaborating with ABC Company to design a Smart Car Parking System. The system will be simulated using Wokwi and VS Code, with Google Sheets as cloud storage for managing parking space data. Users will monitor available parking spaces remotely via an HTML page hosted on GitHub Pages.

**2. System Features**

**2.1 User Interaction via Buttons and Sensors**

* **Button 1: Entry Confirmation**
  + Simulates vehicle entry.
  + The motor opens the parking gate (LED1 simulating the gate).
  + Reduces total available parking spaces by one.
  + The gate remains open for 3 seconds before closing.
* **Button 2: Exit Confirmation**
  + Simulates vehicle exit.
  + The motor opens the gate for exit.
  + Increases total available parking spaces by one.
  + The gate closes automatically after 3 seconds.
* **Button 3: Manual Update**
  + Allows the operator to manually adjust the total number of parking spaces for system error correction or maintenance.

**2.2 Real-Time Monitoring**

* The number of available parking spaces will be displayed:
  + Locally on an LCD connected to the system.
  + Remotely on an HTML page hosted on GitHub Pages.
* The HTML page will allow users to:
  + View the real-time count of free parking spaces.
  + Ensure users can monitor but not modify the parking space count.

**3. Cloud Integration with Google Sheets**

* The system will send and retrieve data from Google Sheets.
* Google Sheets acts as the backend for managing parking space availability.

**4. Scalability Demonstration**

* The system will be tested on three different computers simultaneously.
* Ensures multiple users can access real-time parking availability data without conflicts.

**5. Implementation Tools**

1. **Wokwi** - For simulating the embedded system hardware (buttons, motor, sensors).
2. **VS Code** - For coding and debugging the embedded system.
3. **Google Sheets API** - For integrating cloud storage to manage parking space data.
4. **HTML/JavaScript** - For creating a GitHub-hosted monitoring page.

**6. Tasks**

1. **Simulate the car parking system using Wokwi**
   * Implement button functionalities for entry, exit, and manual updates.
   * Use a servo motor or LED to simulate the gate opening and closing.
2. **Integrate Google Sheets for real-time data synchronization.**
3. **Develop an HTML page to display the real-time count of available parking spaces.**
4. **Demonstrate scalability by running the system on three different computers simultaneously.**

**7. Evaluation Criteria**

1. Correct functionality of the simulation (button operations, motor behavior).
2. Accurate integration with Google Sheets for parking space management.
3. Usability and design of the HTML monitoring page.
4. Successful scalability demonstration across three computers.

**8. Detailed System Design and Architecture**

* System block diagram.
* Communication flow between components.
* Software and hardware interaction layers.
* Data flow from sensors to cloud and HTML page.

**9. Implementation Steps**

* Step-by-step guide to coding and integrating APIs.
* Setting up Wokwi simulation environment.
* Configuring Google Sheets API.
* Designing and hosting the HTML monitoring page.

**10. Challenges and Solutions**

* Potential issues in real-time data synchronization.
* Handling network failures.
* Optimizing cloud API requests for efficient data handling.

**11. Future Enhancements**

* Implementing AI-based parking space prediction.
* Adding IoT sensors for automatic vehicle detection.
* Enhancing security features like RFID access control.

**12. Conclusion** This project will effectively simulate a smart car parking system, integrating embedded system functionalities with cloud storage for real-time data monitoring. The successful implementation will showcase the feasibility of such a system in real-world applications.

**13. Appendices**

* Code snippets and explanations.
* Simulation screenshots.
* Hardware setup diagrams.
* API request/response examples.

**14. References**

* Research papers, documentation, and online resources used in the project.

**15. Additional Notes and Documentation**

* Testing logs and results.
* Team member roles and contributions.
* Feedback from scalability tests.
* Optimization techniques applied during development.