International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 6, Nov-Dec-2026, ISSN (Online): 2395-566X

Iot-Based Smart Parking Reservation System With Automatic Number Plate Recognition And Wrong-Slot Identification

Prof. Asha R[1], Babu Reddy R[2] Gagan Kumar J[3]
Dept. Of Electronics And Telecommunication Engineering Bangalore Institute Of Technology Bengaluru-560004, India

Abstract—The increasing number of vehicles in urban en-vironments has created a demand for efficient parking man-agement solutions. Conventional parking facilities often involve manual supervision, difficulty in locating available spaces, and unauthorized occupation of reserved slots. This paper presents an Internet of Things (IoT)-based smart parking reservation system that integrates vehicle number plate recognition, web-based reservation management, slot occupancy monitoring, and wrong-slot identification.The proposed prototype consists of an Arduino Uno, three infrared (IR) sensors, an entrance camera, a host computer, a web application, a reservation database, a buzzer, and a servo motor. Users register their vehicle details and reserve an available parking slot through the web interface. When a vehicle arrives, its registration plate is captured by the entrance camera and processed using the Plate Recognizer Cloud API. The recognized registration number is verified against active reservation records.Following successful verification, the reserved slot information is transmitted from the host computer to the Arduino Uno through USB serial communication. Individual IR sensors monitor the occupancy of the three parking slots. The controller compares the newly occupied slot with the reserved slot and activates a buzzer when a mismatch is detected. The Arduino Uno also controls a servo motor connected to digital pin D9 for automatic entrance gate operation after successful vehicle and reservation verification. The proposed approach combines vehicle identification and individual slot monitoring to improve reservation compliance and reduce manual intervention. The prototype provides a foundation for developing more comprehensive and scalable parking management systems.

Index Terms—Internet of Things, Smart Parking, Automatic Number Plate Recognition, Plate Recognizer Cloud API, Arduino Uno, Infrared Sensor, Flask, Python, Servo Motor, Wrong-Slot Identification.

I. INTRODUCTION

The growth of vehicle ownership has increased the pressure on parking facilities in urban areas. Educational institutions, hospitals, commercial buildings, shopping complexes, and residential communities require organized parking arrangements to accommodate the increasing number of vehicles.

Conventional parking management frequently depends on manual supervision and physical inspection. Drivers may

spend considerable time searching for vacant spaces, while parking administrators must monitor vehicle movement and slot availability. These activities can lead to unnecessary delays, inefficient space utilization, and conflicts between reserved and occupied parking spaces.

The Internet of Things (IoT) provides a framework for connecting physical devices with software applications to ex-change information and automate monitoring tasks. In parking applications, sensors can provide occupancy information, while web applications can manage reservations and display parking availability.

This paper presents an IoT-based smart parking reservation system that combines vehicle registration number verification with slot-level occupancy monitoring. The proposed prototype is designed around three parking spaces, each equipped with an individual infrared sensor.

A web application developed using Flask allows users to register their vehicle details and select an available parking slot. The reservation information is stored in a database and used during vehicle verification.

At the entrance, a camera captures the vehicle registration plate. The host computer sends the captured image to the Plate Recognizer Cloud API, which processes the image and returns the recognized registration number. The returned number is compared with the active reservation records to retrieve the vehicle's assigned parking slot.

After verification, the host computer communicates the expected slot number to the Arduino Uno through USB serial communication. The Arduino monitors the three IR sensors and identifies new occupancy events.

If the newly occupied slot matches the reserved slot, the occupancy is accepted. If a different slot becomes occupied,

© 2026 IJSRET

the controller activates a buzzer to indicate incorrect parking.

The principal objective is to combine reservation verification and physical slot monitoring rather than treating them as separate operations.

1.1 Problem Statemen

Conventional parking facilities may not provide a reliable connection between a vehicle reservation and the actual parking space occupied by that vehicle. Although a driver may reserve a particular slot, the vehicle can be parked in another space.

This creates inconsistencies between reservation records and actual parking availability. It may also prevent another user from accessing the slot originally assigned to them.

Manual vehicle verification and continuous parking supervi-sion further increase the workload of parking administrators.

The proposed system addresses these issues by recognizing vehicle registration numbers, retrieving the corresponding reservation, monitoring individual parking spaces through IR sensors, and generating an audible warning when an incorrect slot is occupied.

I. OBJECTIVES

The objectives of the proposed system are:

- To develop a web-based parking reservation application.
- To register vehicle details and associate them with parking reservations.
- To capture vehicle registration plates using an entrance camera.
- To recognize registration numbers using the Plate Rec-ognizer Cloud API.
- To verify recognized vehicles against active database reservations.
- To monitor three parking spaces using individual IR sensors connected to an Arduino Uno.
- To transmit reserved slot information through USB serial communication.
- To identify incorrect parking by comparing the expected slot with the newly occupied slot.
- To activate a buzzer when a wrong-slot condition is identified.
- To control the entrance barrier using a servo motor connected directly to the Arduino Uno.

1.2 Scope Of The Project

The proposed implementation focuses on a small-scale parking environment with three parking slots. Each slot is monitored by an individual IR sensor connected to an Arduino Uno.

The web application handles user registration, vehicle details, parking reservations, and reservation records. The host computer performs camera processing, cloud-based number plate recognition, database verification, and communication with the controller.

The Arduino Uno is responsible for monitoring the IR sensor inputs, controlling the buzzer, and operating the entrance gate through a servo motor. The servo signal is connected to digital pin D9.

The system is designed for controlled vehicle entry, where one verified reservation is associated with the next newly detected parking occupancy. A single entrance camera and three binary occupancy sensors cannot independently track multiple vehicles parking simultaneously. Such operation would require additional vehicle-to-slot association mechanisms.

1.3 Literature Survey

Research in smart parking has explored different methods for improving parking availability, vehicle identification, and space management. Existing approaches include sensor-based moni-toring, IoT communication, image processing, and automated reservation applications.

1.4 Sensor-Based Parking Systems

Sensor-based parking systems use electronic devices to determine whether a parking space is occupied. Infrared, ultrasonic, and magnetic sensors are commonly considered for this purpose.

The sensor output can be processed by a microcontroller and transferred to a monitoring application. Such systems provide information about individual parking spaces without requiring continuous visual inspection.

The proposed prototype adopts three IR sensors, with one sensor assigned to each parking slot. The Arduino Uno reads the sensor outputs and determines changes in occupancy.

1.5 Iot-Based Parking Management

IoT-based parking management connects physical sensing equipment with software services. The collected information can be used to update parking status, manage reservations, and support centralized monitoring.Zanella et al. discussed the role of IoT technologies in smart city applications, including communication between connected devices and network-based services [2].

© 2026 IJSRET

The proposed system applies this concept by connecting an Arduino Uno to a host computer and a Flask-based web application.

1.6 Automatic Number Plate Recognition

Automatic Number Plate Recognition (ANPR) identifies vehicles by extracting registration information from images. The process generally involves image acquisition, plate localization, character recognition, and text verification.

The proposed system uses the Plate Recognizer Cloud API to process the captured registration plate image. The returned registration number is compared with the reservation database to identify the corresponding booking.

1. Web-Based Reservation Systems

Web-based parking applications provide an interface for users to view available spaces and reserve parking facilities.

The proposed application uses Flask to process reservation requests and communicate with the database. The reserved slot is associated with the registered vehicle number.

2. Research Gap

Basic occupancy monitoring can identify whether a parking space is vacant or occupied, but it does not establish which vehicle is using the space.

Similarly, reservation management can assign parking spaces without necessarily checking whether the vehicle occupies the assigned slot.

The proposed prototype integrates these functions through the following operations:

- Vehicle registration-based parking reservation.
- Camera-based number plate recognition.
- Database-based reservation verification.
- Individual IR sensor-based slot occupancy monitoring.
- USB serial communication between the host computer and Arduino Uno.
- Buzzer-based wrong-slot warning.
- Arduino-controlled servo motor for automatic entrance gate operation.

The main contribution is the integration of vehicle reservation verification and slot occupancy comparison in a single prototype.

II. PROPOSED SYSTEM ARCHITECTURE

1. System Overview

The proposed architecture contains five major functional units: the user interface, reservation database, vehicle recognition module, hardware controller, and parking occupancy monitoring unit.

The user enters vehicle information through the web application and selects an available parking slot. The reservation is stored in the database.

When the vehicle reaches the entrance, the camera captures its registration plate. The host computer submits the captured image to the Plate Recognizer Cloud API.

The recognized vehicle number is retrieved and checked against the active reservation records. When a valid booking is identified, the expected slot number is transmitted to the Arduino Uno.

The Arduino continuously monitors the three IR sensors. A transition from vacant to occupied indicates a new occupancy event. The controller compares the newly occupied slot with the expected slot and activates the buzzer if they are different.

A. Block Diagram

1. User Interface:

The user interface allows vehicle registration, reservation creation, and viewing of parking availability. Users select an available parking space through the web application.

2. Flask Web Application:

Flask manages web requests, user operations, reservation processing, and database communication.

3. Reservation Database:

The database stores user details, vehicle registration numbers, reserved slot numbers, booking information, and reservation status.

© 2026 IJSRET

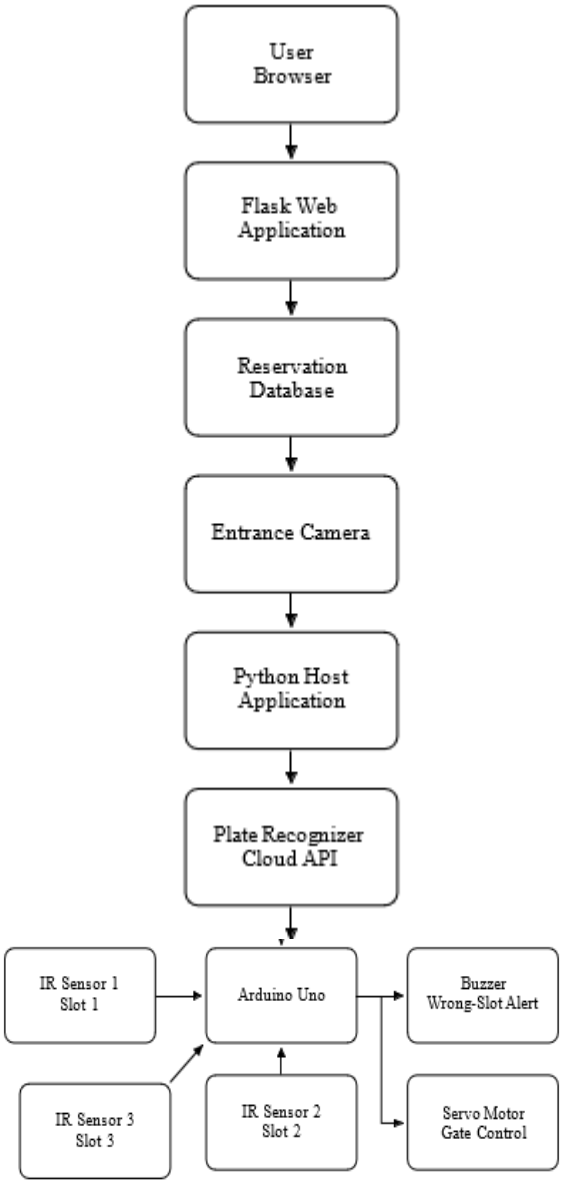

Fig. 1. System architecture of the proposed IoT-based smart parking reservation system.

4. Entrance Camera:
The entrance camera captures the registration plate image when a vehicle reaches the verification area.

5. Plate Recognition Module:
The host computer sends the captured image to the Plate Recognizer Cloud API. The API processes the image and returns recognized plate information.

6. Host Computer:
The host computer performs camera capture, API communication, reservation verification, and serial communication with the Arduino Uno.

7. Arduino Uno:
The Arduino Uno reads the three IR sensor inputs, compares new occupancy events with the reserved slot, and controls the buzzer. It also controls the entrance gate servo motor through digital pin D9.

8. Ir Sensors:
Each IR sensor detects whether its corresponding parking slot is occupied. These sensors provide occupancy information and do not identify vehicle registration numbers.

9.Buzzer:
The Buzzer Generates An Audible Warning When The Newly Occupied Slot Differs From The Reserved Slot.

10:Servo Motor-Based Automatic Gate Control:
The servo motor is used to control the automatic entry barrier of the proposed smart parking system. It is interfaced with the Arduino Uno through a digital control signal. When a vehicle's number plate is successfully recognized and its reservation is verified, an authorized gate-opening command is provided to the controller. The servo motor rotates to the predefined open position, allowing the vehicle to enter the parking area. After the vehicle passes through the entrance, the servo returns to its initial closed position. This mechanism reduces manual gate

1. Working Principle

The complete operation follows these steps:

- The user registers vehicle information through the website.
- The user selects an available parking slot.
- The application stores the reservation in the database.
- The entrance camera captures the vehicle registration plate.
- The host computer sends the image to the Plate Recognizer Cloud API.
- The recognized registration number is retrieved from the API response.
- The host computer searches the database for a valid active reservation.
- The reserved slot number is transmitted to the Arduino Uno using USB serial communication.

© 2026 IJSRET

- After successful vehicle and reservation verification, the Arduino drives the servo motor to open the entrance barrier.
- After the vehicle passes the entrance, the Arduino returns the servo motor to the closed position.
- The Arduino monitors the three IR sensors.
- When a new occupancy event is detected, the controller identifies the corresponding slot.
- The detected slot is compared with the reserved slot.
- A mismatch activates the buzzer and generates a warning.

4. Host-to-Arduino Communication

The host computer communicates with the Arduino Uno through a USB serial connection.

A text-based communication protocol can be used to transmit reservation information.

An example command is:

a. Book,Session01,2

This message indicates that the active session is associated with Slot 2.

The Arduino may return:

b. Ack,Session01

For a wrong-slot condition, an example response is:

This indicates that Slot 2 was expected but Slot 3 became occupied.

No.	Component	Quantity
1	Arduino Uno	1
2	IR Sensor Module	3
3	USB Cable	1
4	Buzzer	1
5	Entrance Camera	1
6	Host Computer	1
7	Connecting Wires	As required
8	Power Supply	As required
9	Servo Motor	1

The Python application and Arduino program must use the same communication format.

I. HARDWARE DESIGN

1. Hardware Requirements

The prototype requires the following hardware components.

1. Arduino Uno

The Arduino Uno is the main hardware controller of the prototype. It is based on the ATmega328P microcontroller and provides digital input and output connections.

The three IR sensor modules are connected to separate digital input pins. The buzzer is connected to a digital output pin. A servo motor is also connected to the Arduino Uno, with its signal wire connected to digital pin D9, to operate the entrance barrier.

The controller receives reservation information from the host computer through USB serial communication. It uses this information to evaluate newly detected parking occupancy events.

2. Infrared Sensors

Infrared sensors detect the presence of an object within their sensing range.One sensor is installed in each parking slot. The sensor output changes according to the presence of a vehicle and the module's output configuration.

The Arduino reads the individual sensor states to determine whether each parking space is vacant or occupied.

The sensors do not recognize registration numbers. Vehicle identification is performed separately by the entrance camera and recognition module.

3. Entrance Camera

The entrance camera captures an image of the vehicle registration plate.The captured image is provided to the host computer, which submits it to the Plate Recognizer Cloud API.The API returns the recognized registration number and as-sociated recognition information. The returned vehicle number is used for reservation verification. TABLE II Arduino Pin Configuration

Device	Arduino Pin
IR Sensor 1	D2
IR Sensor 2	D3
IR Sensor 3	D4
Buzzer	D8
Servo Motor Signal	D9
USB Serial Communication	USB Port

1. Buzzer

The buzzer is used to provide an audible warning during wrong-slot parking.

© 2026 IJSRET

The Arduino activates the buzzer when a newly occupied parking slot does not match the reserved slot.

A suitable driver circuit must be used if the buzzer current requirement exceeds the Arduino output pin capability.

2. Servo Motor-Based Gate Control
A servo motor is physically connected and tested with the Arduino Uno in the prototype. Its signal wire is connected to digital pin D9. The servo controls the entrance barrier by moving between the predefined closed and open positions. After successful vehicle-number recognition and reservation verification, the Arduino commands the servo to open the gate. After the vehicle passes, the servo is returned to the closed position. The servo is controlled directly by the Arduino and is not connected to the Python host computer or to the buzzer.

3. USB Communication
The USB cable connects the Arduino Uno to the host computer.It provides a serial communication channel through which the host computer transmits the reserved slot information and receives controller status messages.

4. Proposed Pin Configuration
The following pin configuration represents the prototype arrangement used for the physical setup.The sensor supply voltage and signal polarity must be checked according to the selected IR modules. All connected modules must share a common ground.

I. SOFTWARE DESIGN

1. Software Requirements
The software components used in the proposed system are listed below.

2. Python Application
Python is used for host-side application development. It handles camera operations, database queries, API requests, reservation verification, and serial communication.Its libraries support the integration of web services, image acquisition, and hardware communication.

TABLE III Software Requirements

No.	Software
1	Python
2	OpenCV
3	Flask
4	Arduino IDE
5	Database Software
6	Plate Recognizer Cloud API
7	Serial Communication Library

3. OpenCV
OpenCV is used for camera access and image acquisition. It can capture images, resize frames, and perform basic image preprocessing.

In this project, the complete number plate recognition task is handled by the Plate Recognizer Cloud API, not by OpenCV alone.

7. Plate Recognizer Cloud API
The Plate Recognizer Cloud API provides the number plate recognition functionality.

The host computer submits the captured vehicle image to the API using an authenticated HTTP request.

The API processes the image and returns a structured response containing recognized registration information.

The application extracts the plate text from the response, normalizes the text, and searches for a matching active reservation in the database.

The recognition result must be checked for validity before the vehicle is associated with a parking reservation.

8. Flask Web Framework
Flask is a Python-based web framework used to implement the reservation interface.

It handles user requests, vehicle registration, slot selection, reservation creation, and display of booking information.

The web application communicates with the database to retrieve parking availability and reservation records.

9. Database Management
The database stores the information needed for parking reservation and vehicle verification.

The principal data fields include:

- User identification.
- Vehicle registration number.
- Reservation identification.
- Reserved slot number.
- Booking date and time.
- Entry and exit information.
- Reservation status.

A simplified database structure is presented in Table IV.

The database must prevent conflicting active bookings for the same slot and reservation period.

© 2026 IJSRET

TABLE IV

Field	Purpose
Reservation ID	Unique booking identifier
Vehicle Number	Vehicle registration
Slot Number	Assigned parking space
Booking Time	Reservation time
Entry Status	Vehicle entry information
Exit Status	Vehicle exit information
Status	Booking state

1. Image Acquisition:
The Camera Captures The Vehicle Registration Plate Image.

2. Image Preparation:
The Host Application Checks The Captured Image And Prepares It For Submission.

3. Api Request:
The Image Is Transmitted To The Plate Recognizer Cloud Api Using An Authenticated Request.

4. Recognition Response:
The Api Returns The Recognized Plate Information In A Structured Response.

5. Text Normalization:
The Application Converts The Extracted Registration Text Into A Consistent Format For Comparison.

6. Reservation Verification:
The Recognized Vehicle Number Is Checked Against Active Reservation Records.

7. Slot Retrieval:
If A Valid Reservation Is Found, The Assigned Parking Slot Is Retrieved.

8. Controller Communication:
The expected slot information is transmitted to the Arduino Uno.

I. System Flowchart
The flowchart in Fig. 2 illustrates the major operations performed by the system.

I. ALGORITHM AND SYSTEM OPERATION
A. Reservation Algorithm
The reservation process is performed as follows:

- Start the web application.
- Accept the user's vehicle registration number.
- Display the available parking slots.
- Allow the user to select a parking slot.
- Check the selected slot's availability for the requested reservation period.
- Store the booking when the slot is available.
- Display the reservation confirmation.
- Request another slot if the selected slot is unavailable.

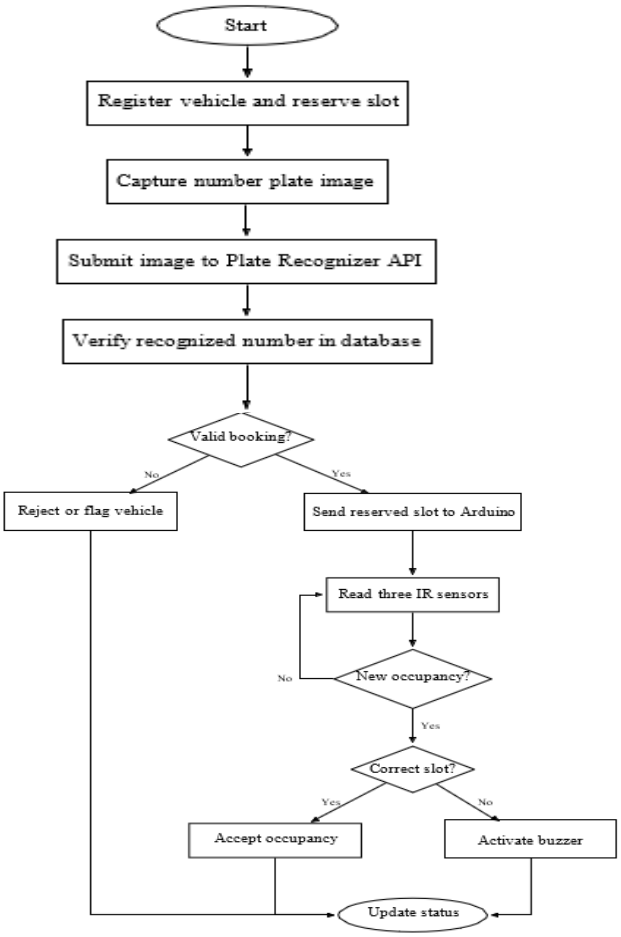

Fig. 2. Operational flowchart of the proposed parking system.

A. Vehicle Verification Algorithm

- Initialize the entrance camera.
- Capture the registration plate image.
- Prepare the image for API submission.
- Send the image to the Plate Recognizer Cloud API.
- Receive the recognition response.
- Extract and normalize the registration number.

© 2026 IJSRET

- Search for an active reservation with the recognized number.
- Retrieve the assigned parking slot if a valid booking exists.
- Transmit the slot information to the Arduino Uno.
- Flag the vehicle if no valid reservation is found.

A. Wrong-Slot Detection Algorithm

- Initialize the three IR sensors.
- Receive the expected slot number.
- Read the current sensor states.
- Compare current readings with previous readings.
- Identify a vacant-to-occupied transition.
- Determine the newly occupied slot.
- Compare the detected slot with the reserved slot.
- Accept the occupancy if both slot numbers match.
- Activate the buzzer if the slot numbers differ.
- Transmit the detection result to the host computer.

I. Mathematical Representation

The proposed system determines parking correctness by comparing the reserved slot with the newly occupied slot.

Let:

- Sr represent the reserved slot.
- So represent the occupied slot.
- V represent the correct parking condition.
- B represent the buzzer output.

The correct parking condition is represented by:

$$V = \begin{cases} 1, & S_r = S_o \\ 0, & S_r \neq S_o \end{cases} \quad (1)$$

The buzzer output is

$$B = \begin{cases} 0, & S_r = S_o \\ 1, & S_r \neq S_o \end{cases} \quad (2)$$

The occupancy state of each parking slot is represented by:

The occupancy state of each parking slot is represented by:For three parking spaces, the occupancy vector is: O = [O1, O2, O3] (4) A new occupancy event is detected when the sensor state changes from vacant to occupied: Ei(t) = { 1, Oi(t − 1) = 0, Oi(t) = 1 0, otherwise (5) The controller uses the new occupancy event to identify the slot that requires comparison with the active reservation.

III. IMPLEMENTATION DETAILS

A. Hardware Implementation The hardware arrangement contains an Arduino Uno, three IR sensor modules, a buzzer, and a servo motor. The servo motor is physically interfaced with the Arduino Uno and its signal is connected to digital pin D9. The IR sensors are installed in the three parking spaces. Each sensor is connected to a separate digital input pin of the Arduino. The buzzer is connected to a digital output pin. The servo motor is connected to the Arduino Uno and is used to operate the entrance gate after successful vehicle and reservation verification.The Arduino is connected to the host computer using a USB cable. The controller receives the reserved slot information and evaluates the sensor readings to determine whether the new occupancy matches the reservation. B. Web Application Implementation The web application is developed using Flask. It provides an interface for user registration, vehicle information, slot availability, and reservation operations. When a user submits a reservation request, the application checks the availability of the selected slot. If the slot is available, the application stores the reservation details in the database. C. Number Plate Recognition Implementation The entrance camera is accessed through the host computer. The captured image is submitted to the Plate Recognizer Cloud API. The API returns the recognized registration information. The host application extracts the recognized plate text and compares it with the registered vehicle number in the reservation database. The reservation is considered valid only when the recognized vehicle number matches an active booking. The API response must be checked for unsuccessful requests, low-confidence recognition, and missing plate information. D. Serial Communication Implementation After successful reservation verification, the host computer transmits the expected slot number to the Arduino Uno. An example command is:

BOOK,SESSION01,2

The Arduino receives the command and stores the expected slot for the active session. When a new occupancy event is detected, the Arduino sends the result to the host computer. E. System Integration The web application, database, recognition API, host computer, and Arduino controller operate as interconnected modules.

The web application manages the reservation. The recognition module identifies the vehicle. The database verifies the booking. The Arduino monitors parking occupancy. The buzzer provides an immediate audible indication of incorrect slot usage. The Arduino also controls the servo motor for automatic gate opening and closing.

© 2026 IJSRET

IV. TESTING AND PERFORMANCE EVALUATION

Testing is necessary to verify the operation of the proposed system. The following test cases can be used during prototype evaluation

A. Functional Test Cases B. Performance Parameters The prototype can be evaluated using the following parameters. 1. Number Plate Recognition Accuracy This parameter measures the percentage of correctly recognized registration numbers.

$$Ap = Nc\ Nt \times 100 \quad (6)$$

where Nc is the number of correctly recognized plates and Nt is the total number of test plates.

2. Slot Detection Accuracy This parameter represents the percentage of correct occupancy detections.

$$As = Sc\ St \times 100 \quad (7)$$

where Sc represents correct slot detections and St represents total slot detection tests.

3. Wrong-Slot Detection Rate This parameter evaluates the ability of the controller to identify incorrect slot occupation.

$$Dw = Wd\ Wt \times 100 \quad (8)$$

where Wd represents detected wrong-slot events and Wt represents total wrong-slot test events.

4. System Response Time Response time is the time between the detection of an event and the corresponding system response.

$$Tr = T2 - T1 \quad (9)$$

where T1 is the event detection time and T2 is the response generation time. C. Experimental Results Actual measurements must be entered after testing the prototype. The experimental values should be obtained from repeated hardware tests. No performance percentage or response time is assumed in this paper.

V. DESIGN REQUIREMENTS AND USE-CASE ANALYSIS

1. Functional Requirements

The functional requirements describe the operations that the proposed prototype is expected to perform. They provide a basis for dividing the system into software and hardware modules and for preparing the test cases presented later in this paper.

The first requirement is vehicle registration. The application must accept the vehicle registration number and associate it with a user record. The registration number is the principal identifier used during the entrance verification process. The application should store the registration number in a consistent format so that it can be compared with the text returned by the recognition service.

The second requirement is reservation creation. A user must be able to view the parking spaces and select an available slot. Before saving a booking, the application must check the selected slot against existing active reservations. A reservation should not be created if another active booking already occupies the same slot for an overlapping reservation period.

The third requirement is vehicle verification. The entrance camera captures a registration plate image, and the host application submits the image to the Plate Recognizer Cloud API. The recognized text is compared with the registered vehicle number and the active reservation records. A valid result must identify the booking and its assigned slot.

The fourth requirement is slot occupancy monitoring. Each of the three parking spaces is monitored by a separate IR sensor. The Arduino Uno reads the sensor states and detects a change from vacant to occupied. The controller uses the active expected slot received from the host computer to determine whether the newly occupied space is correct.

The fifth requirement is warning generation. If the newly occupied slot does not match the reserved slot, the Arduino activates the buzzer. The host application may also receive a status message from the controller so that the event can be recorded for later review.

2. Non-Functional Requirements

In addition to the functional requirements, the prototype should consider reliability, usability, maintainability, and re-sponse time. These requirements influence the way the software and hardware modules are designed.

Reliability: The system should avoid accepting an incom-plete or invalid recognition response as a successful vehicle verification. Sensor readings should be interpreted consistently, and communication errors should not be treated as successful commands.

Usability: The web interface should present the available parking spaces in a clear manner. Registration and reservation forms should request only the information needed for the booking process. The user should be able to identify whether a reservation has been created successfully.

© 2026 IJSRET

Maintainability: The application should separate reservation management, API communication, database access, and serial communication into independent software functions. This separation makes it easier to modify one module without redesigning the complete system.

Response time: The system should process a vehicle arriving at the entrance without unnecessary delay. The total time includes image capture, network communication with the recognition service, database verification, serial transmission, and the controller response. These components should be measured separately during experimental testing.

Availability: The web application, host computer, camera, and network connection must be available for the complete recognition workflow. If the cloud recognition service cannot be reached, the system should report that verification is incomplete rather than assuming that the vehicle is authorized.

TABLE V Functional Test Cases

Parameter	Value
Number of parking slots	3
Number of IR sensors	3
Controller	Arduino Uno
Camera recognition	measured sucessfully
Slot detection accuracy	measured sucessfully
Wrong-slot detection rate	measured sucessfully
Average response time	measured sucessfully
Buzzer operation	Tested successfully
Servo motor operation	Tested successfully

TABLE VI Experimental Evaluation

No.	Test	Input Condition	Expected Result	Status
1	Vehicle registration	User submits vehicle details	Vehicle details are stored	Tested successfully
2	Slot reservation	User selects an available slot	Reservation is created	Tested successfully
3	Slot availability	User selects an unavailable slot	Conflicting reservation is rejected	Tested successfully
4	Camera capture	Vehicle reaches entrance	Registration image is captured	Tested successfully
5	Number recognition	Valid image is submitted	Plate information is returned	Tested successfully
6	Reservation verification	Recognized number matches booking	Assigned slot is retrieved	Tested successfully
7	Correct parking	Reserved Slot 2 becomes occu-pied	Occupancy is accepted	Tested successfully
8	Wrong parking	Reserved Slot 2, Slot 3 becomes occupied	Buzzer is activated	Tested successfully
9	Vehicle departure	Occupied slot becomes vacant	Slot status is updated	Tested successfully
10	Servo gate operation	Valid vehicle authorization	Servo opens and closes the gate	Tested successfully

3. Primary Use Cases

The principal actors in the prototype are the parking user and the parking system operator. The user interacts with the web application to register a vehicle and create a reservation. The system operator may inspect reservation and occupancy information during prototype operation.

The main use cases are summarized below.

1. **Register vehicle:** The user enters the vehicle number and the required account information. The application validates the submitted fields and stores the vehicle record.

2. **View slot availability:** The user requests the current parking status. The application displays the status in-formation available from the reservation and occupancy records.

3. **Create reservation:** The user selects a slot and a reservation period. The application checks for a conflict and stores the booking when the selected slot is available.

4. **Verify arriving vehicle:** The camera captures the plate image, the recognition service returns the plate text, and the application searches for a matching active reservation.

5. **Check parking position:** After a valid reservation is identified, the Arduino monitors the three IR sensors and compares the new occupancy event with the expected slot.

© 2026 IJSRET

6. **Generate warning:** If the detected slot differs from the reserved slot, the buzzer is activated to indicate a wrong-slot event.

7. **Record departure:** When a slot changes from occupied to vacant, the occupancy change can be reported to the host application for updating the corresponding parking record. The exact association between a departing vehicle and a booking must be maintained by the application logic.

4. Operating Assumptions

The prototype is intended for a controlled, small-scale parking environment. It assumes that a vehicle is presented to the entrance camera in a position where the registration plate can be captured. It also assumes that the camera is connected to the host computer and that the host computer has access to the recognition API.

The present design associates one verified reservation with the next newly detected parking occupancy. This assumption is important because the three IR sensors provide occupancy states but do not identify the vehicles that cause those states. If two or more vehicles enter and park at nearly the same time, the current arrangement cannot reliably determine which occupancy event belongs to which recognized vehicle. The prototype therefore requires controlled operation during evaluation. Multi-vehicle tracking is considered a future enhancement.

VI. DETAILED RESERVATION AND DATABASE DESIGN

1. Vehicle Registration Data

The vehicle registration record connects a user account with the registration number that is expected to be recognized at the entrance. The registration number should be stored in a normalized form. For example, leading and trailing spaces should be removed, and letter case should be made consistent before comparison. The normalization process must not change the actual sequence of letters and digits.

A registration record should be checked for missing values before it is saved. The application should also define how duplicate vehicle numbers are handled. If the same registration number is already associated with an account, the application

should avoid creating conflicting records unless the project requirements explicitly permit multiple users to share a vehicle.

2. Reservation Record

A reservation record links a user, a vehicle, a parking slot, and a booking period. A unique reservation identifier is useful for distinguishing separate bookings made by the same user. The record should contain the selected slot number, booking date and time, expected entry information, exit information, and the current reservation status.

A reservation can pass through several logical states. These states may include pending, active, vehicle verified, parked, completed, and cancelled. The exact state names depend on the implementation, but the transitions should be consistent. For example, a reservation should not be treated as completed merely because a vehicle has been recognized at the entrance. Completion should be associated with the departure or with an explicit booking completion operation.

3. Slot Availability Check

The availability check is performed before a reservation is inserted into the database. Let a requested reservation use slot s during the interval $[t_s, t_e]$. An existing reservation for the same slot conflicts with the requested reservation if the two time intervals overlap and the existing record is still active.

For two intervals $[t_1, t_2]$ and $[t_3, t_4]$, an overlap exists when

$$t_1 < t_4 \text{ and } t_3 < t_2. \quad (10)$$

The application should perform the conflict check immediately before saving the new reservation. In a multi-user environment, the check and insertion should be handled as one database transaction so that two requests cannot reserve the same slot based on the same earlier availability result.

4. Logical Database Entities

The database can be organized around four logical entities: users, vehicles, reservations, and parking slots. The user entity stores account information. The vehicle entity stores the registration number associated with a user. The reservation entity stores the booking details and references the vehicle and slot. The slot entity represents the parking spaces supported by the installation.

The relationship between these entities is described as follows. A user may have one or more registered vehicles, depending on the application policy. A vehicle may be associated with multiple reservations over time, but it should not have conflicting active reservations if the application permits only one active booking per vehicle. Each reservation refers to one vehicle and one parking slot. A parking slot may be referenced by many historical reservations, but overlapping active reservations for that slot must be prevented.

This logical structure is independent of a particular database product. The project may use a lightweight local database during prototype development. The selected database should support consistent inserts, updates, and queries for reservation verification.

© 2026 IJSRET

TABLE VII Logical Database Entities

Entity	Representative attributes
User	User ID, name, contact details, account credentials
Vehicle	Vehicle ID, user ID, registration number
Slot	Slot ID, slot number, current occupancy state
Reservation	Reservation ID, vehicle ID, slot ID, start time, end time, status

5.Reservation Verification Query

After the recognition service returns a plate number, the host application normalizes the returned text and searches the reservation records. The query must identify a record whose vehicle number matches the recognized value and whose reservation status and time period indicate that the booking is valid.

A simplified logical query is shown below. It is illustrative and must be adapted to the actual table and column names used in the implementation.

```
SELECT RESERVATION_ID, SLOT_NUMBER FROM
RESERVATIONS
WHERE NORMALIZED_VEHICLE_NUMBER = ?
AND STATUS = 'ACTIVE'
AND START_TIME <= CURRENT_TIMESTAMP AND
END_TIME >= CURRENT_TIMESTAMP;
```

The use of a parameterized query is important. User-supplied and API-returned text should be passed as query parameters rather than combined directly into an SQL statement. This reduces the risk of query manipulation and makes the database operation more predictable.

6. Consistency Between Reservation And Occupancy

Reservation status and physical occupancy are related but different types of information. A reservation indicates that a slot has been assigned to a vehicle for a period. An IR sensor indicates whether an object is currently detected in the corresponding parking space. A reserved slot may remain physically vacant before the vehicle arrives, and an occupied slot may not correspond to a valid reservation.

The application should therefore avoid using the reservation table alone as proof of physical occupancy. Similarly, sensor occupancy alone should not be interpreted as proof that the occupying vehicle is authorized. The proposed system combines these two information sources during the wrong-slot check.

VII. DETAILED MODULE DESIGN AND INTERACTION

1. User Interface Module

The user interface is the entry point for reservation operations. It should provide forms for user registration, vehicle registration,

and slot selection. The available parking spaces should be presented using a consistent status representation so that the user can understand whether a slot can be selected.

Form validation should be performed before a request is submitted to the server. Required fields should not be left empty, and the vehicle registration number should be checked for a reasonable format. Server-side validation is also required because client-side validation alone can be bypassed.

After a reservation is submitted, the interface should display whether the booking was accepted or rejected. If the slot is no longer available, the user should receive a clear message and should be asked to select another available slot.

2. Flask Application Module

The Flask application receives browser requests and coordinates the reservation workflow. Its routes can be grouped according to their responsibilities: account operations, vehicle operations, slot availability, reservation creation, reservation history, and administrative monitoring.

The application should keep database operations separate from presentation templates. This improves readability and reduces the chance that a change to the page layout will affect reservation logic. Input validation, authentication checks, and error handling should be performed in the server-side route or in supporting functions.

The Flask application also provides the information required by the host-side recognition process. After a recognized plate is received, the application or its supporting Python module can retrieve the active reservation and return the expected slot number. The implementation should ensure that only a valid active booking is returned.

3.Camera Acquisition Module

The camera acquisition module is responsible for obtaining an image of the vehicle registration plate at the entrance. Camera placement should provide a clear view of the plate while allowing the vehicle to approach in a controlled manner. The camera should be mounted securely to reduce changes in angle and vibration.

OpenCV may be used to access the camera, read frames, and perform basic image preparation. It is not the number

© 2026 IJSRET

plate recognition engine in this design. The recognition task is performed by the Plate Recognizer Cloud API. This distinction is important when describing the software architecture and evaluating the source of recognition errors.

Before submitting an image, the host application can check whether a frame was captured successfully and whether the image contains usable visual information. If the camera does not return a valid frame, the application should request another capture or report that image acquisition failed.

4. Cloud Recognition Module

The recognition module transfers the captured image to the Plate Recognizer Cloud API using the API's supported request format and authentication method. The host application then processes the response and extracts the recognized registration number.The response should be checked before the plate text is used. Possible conditions include a successful response with a plate value, a successful response without a usable plate value, an authentication failure, a network timeout, or a service-side error. These cases should be handled separately because they have different causes and may require different corrective actions.

The recognized value should be normalized before comparison with the database. If the API provides a confidence value, the application may use a configured threshold as an additional quality check. The threshold should be selected through testing; a specific value should not be assumed without experimental evidence.

5. Arduino Sensor Monitoring Module

The Arduino Uno receives the expected slot number from the host computer and monitors the digital outputs of the three IR sensors. Each sensor corresponds to one physical parking slot. The software must account for the electrical output convention of the selected sensor modules because some modules produce an active-low output while others may be configured differently.A stable sensor state should be established before a transition is accepted as a parking event. A short period of repeated sampling or a debounce interval can reduce false transitions caused by electrical noise or unstable sensor output. The selected interval should be validated with the actual sensors and installation.The controller compares the previous occupancy vector with the current occupancy vector. A vacant-to-occupied transition identifies a newly occupied slot. The slot index is then compared with the expected slot received from the host application.

6 Buzzer Alert Module

The buzzer is the local warning device for incorrect parking. When the controller identifies a mismatch, it activates the buzzer according to the alert logic programmed in the Arduino. The alert duration and repetition pattern should be selected so that the warning is noticeable without causing unnecessary continuous activation.

The buzzer output should be returned to its inactive state after the alert condition is cleared or after the configured alert interval has elapsed. If the selected buzzer draws more current than the Arduino output pin can safely provide, an appropriate driver stage should be used. The buzzer rating and the driver circuit must be checked before final hardware assembly.

7. Module Interaction Sequence

The modules interact in a defined sequence. First, the user creates a reservation through the web interface. The database stores the vehicle number and assigned slot. When the vehicle arrives, the camera provides an image to the host computer. The host computer submits the image to the cloud recognition service and receives the plate result.

The host application verifies the result against the active reservation data. If a valid reservation is found, the assigned slot is sent to the Arduino. The controller acknowledges the

command and waits for a new occupancy event. When one of the sensors indicates a new occupied state, the controller compares the corresponding slot with the expected slot. A mismatch causes the buzzer to operate and may be reported to the host application.

This sequence separates vehicle identification from physical occupancy detection. The camera and recognition API identify the vehicle at the entrance, while the IR sensors identify the parking space that becomes occupied.

VII.COMMUNICATION PROTOCOL AND ERROR HANDLING

1. Serial Message Structure

The USB serial connection provides a simple communication path between the host computer and the Arduino Uno. A text-based protocol is suitable for the small number of messages exchanged by the prototype. Each message should use a fixed field order and a line terminator so that the receiver can identify the end of a command.

The reservation command can contain a command name, a session identifier, and a slot number. For example:

Book,Session01,2

The Arduino can acknowledge receipt using:

Ack,Session01

© 2026 IJSRET

A wrong-slot message can contain the expected slot and the detected slot:

Wrong,2,3
The message indicates that Slot 2 was expected while Slot

Became Occupied
The session identifier helps the host application associate a controller response with the reservation that initiated the command.

2. Message Validation
The Arduino should validate every received command before using its contents. It should check the command name, the number of fields, the session identifier, and whether the slot number is within the supported range. A malformed command should be rejected and should not replace the currently stored expected slot.

The host application should also validate messages received from the Arduino. An acknowledgement should be associated with the correct session, and a wrong-slot message should contain valid slot identifiers. Unexpected or incomplete mes-sages should be logged for diagnosis rather than interpreted as successful parking events.

3. Communication Failure Conditions
Serial communication may fail if the USB cable is dis-connected, the wrong serial port is selected, the baud rate differs between the host and controller, or the Arduino is reset during operation. The host application should detect a missing acknowledgement within a defined timeout period.

If the command is not acknowledged, the host application should not assume that the Arduino has received the reservation. It may retry the command a limited number of times and then report a communication failure. The retry count and timeout duration should be selected during implementation testing.

4. Recognition And Network Errors
The cloud recognition process depends on a working internet connection and valid API credentials. A network timeout or authentication failure must not be treated as a valid recognition result. The application should display an appropriate status and allow an operator to follow the project's defined manual verification procedure, if such a procedure is provided.

The application should also distinguish between a plate that was not detected and a plate that was detected but did not match an active reservation. These are different outcomes: the first is a recognition or image-quality issue, while the second is a reservation verification failure.

5.Sensor Fault Considerations
An IR sensor may remain in one state because of a wiring fault, incorrect alignment, obstruction, or component failure. The controller can identify some abnormal conditions by checking whether a sensor state remains unchanged for an unexpectedly long period, but a binary digital sensor does not provide enough information to diagnose every possible fault.

During installation, each sensor should be tested indepen-dently. The installer should confirm its detection range, output polarity, stable vacant state, and occupied state. The physical placement should be checked to ensure that the sensor responds to a vehicle in the intended parking position.

VIII.SECURITY, PRIVACY, AND DATA MANAGEMENT

1. User Account Protection
The web application stores user and vehicle information. Ac-count credentials should not be stored as plain text. Passwords should be protected using a suitable password-hashing method provided by the application framework or a trusted security library. Authentication checks should be applied to pages that display private reservation information.

The application should validate all form inputs on the server. A user should be permitted to view or modify only the records that belong to that account, unless the user has an authorized administrative role. Session handling should be configured carefully so that private pages are not accessible to an unauthenticated visitor.

2. Api Credential Management
The Plate Recognizer API credential is required for requests to the cloud recognition service. It should be stored outside publicly accessible source files, such as in a protected envi-ronment configuration. The credential should not be printed in application logs or included in screenshots and public project repositories.The host application should use the authentication method specified by the API provider. If the credential is exposed, it should be replaced according to the provider's account management procedure.

3. Database Protection
Database operations should use parameterized statements or the parameter-binding features of the selected database library. This is particularly important for vehicle registration numbers and other values received from forms or external

© 2026 IJSRET

services.Access to the database file or database server should be restricted to the application processes and authorized administrators. Regular backups can reduce the risk of losing reservation records during development and demonstration.

4. Vehicle Image And Personal Data

Vehicle registration numbers and captured plate images may be associated with identifiable individuals. The project should collect only the information needed for reservation and verification. Access to stored images and recognition responses should be limited to authorized project operators.

A retention policy should be defined for captured images and logs. If images are not required after verification, the application should avoid retaining them indefinitely. The demonstration and testing process should use data responsibly and should not expose API credentials or personal information in the research paper.

5. Trust Boundaries

The architecture contains several trust boundaries: the user's browser communicates with the Flask server, the host computer communicates with the cloud API, and the host computer exchanges serial messages with the Arduino. Each boundary should validate the information it receives.

The browser must not be treated as a trusted source of slot availability. The server must repeat the availability check before saving a reservation. Similarly, the Arduino should validate serial commands rather than accepting arbitrary text as a valid slot assignment. These checks help prevent inconsistent system states.

IX. DETAILED EXPERIMENTAL METHODOLOGY

1. Purpose Of Evaluation

The evaluation is intended to determine whether the individ-ual modules and the complete prototype perform their specified functions. The testing process should begin with individual component checks and proceed to integrated system tests. This approach makes it easier to identify whether a failure originates in the camera, recognition service, database, serial link, sensor circuit, or buzzer output.No numerical performance result is assumed in this paper. The values in the experimental evaluation table must be filled only after measurements have been collected from the physical prototype.

2. Hardware Inspection

Before functional testing, the Arduino Uno, IR sensors, buzzer, USB cable, and power connections should be inspected. The sensor supply voltage and signal wiring must match the specifications of the selected modules. The common ground connection should be verified.

Each IR sensor should then be tested separately. The test should record the output state when the parking space is vacant and when a vehicle or suitable test object is placed within the detection range. The test should be repeated at different positions within the intended detection area to identify alignment problems.

3. Web Application Testing

The web application should be tested using valid and invalid registration data. The test should verify that required fields are checked, vehicle information is stored correctly, and the application displays the expected response.

Reservation tests should include selecting an available slot, selecting a slot that already has an overlapping active reservation, and submitting incomplete reservation details. The database should be inspected after each test to confirm that records have been inserted or rejected correctly.

4. Recognition Testing

Recognition tests should use a set of registration plate images captured with the actual entrance camera. The images should represent the lighting, distance, angle, and visibility conditions expected during the demonstration.

For each test image, the experiment should record whether the camera captured a usable image, whether the API returned a plate result, whether the returned text matched the physical registration number, and the time required to receive the response. Images that produce no result should be recorded separately from images that produce an incorrect result.

The test set should include repeated captures of the same vehicle so that variation between captures can be observed. The number of test images and the conditions under which they were captured should be reported with the final results.

5. Reservation Verification Testing

Reservation verification should be tested with at least three conditions: a recognized plate associated with an active reservation, a recognized plate with no active reservation, and a recognized plate whose reservation is outside its valid time period.

The expected outcome is that only a matching active reserva-tion returns an assigned slot. An unknown vehicle or an expired booking should not be treated as an authorized reservation. The test should also confirm that a failed verification does not unintentionally overwrite an existing active session on the Arduino.

© 2026 IJSRET

6. Wrong-Slot Detection Testing
Wrong-slot detection should be evaluated by creating a reser-vation for each supported slot and then producing occupancy transitions at the corresponding sensor inputs. For a correct-slot test, the sensor associated with the reserved slot should change from vacant to occupied. For a wrong-slot test, a different sensor should change to occupied.

The test should record the expected slot, the sensor that changed state, the buzzer response, and the message received by the host computer. Each condition should be repeated to check whether the result is consistent. Tests should also include a case in which no new occupancy event occurs, to confirm that the buzzer is not activated without a detected mismatch.

7. Integrated System Testing
After individual tests are complete, the full sequence should be evaluated. A user creates a reservation, the entrance camera captures the registered vehicle, the cloud service returns the plate result, the database verifies the booking, and the host computer sends the expected slot to the Arduino. The controller then monitors the sensor inputs and reports the occupancy result.

The integrated test should record failures at each stage. For example, a successful API response does not prove that the database verification succeeded, and a successful serial acknowledgement does not prove that the correct sensor was activated. Each stage must be checked independently.

8. Measurement Procedure
The number plate recognition accuracy can be measured by comparing the recognized text with the known registration number for every valid test image. The slot detection accuracy can be measured by comparing the controller's reported slot with the slot physically occupied during each test. The wrong-slot detection rate can be measured using deliberate wrong-slot trials.

The response time should be measured using timestamps captured at the beginning and end of a clearly defined operation. For example, API response time can be measured from request submission to response receipt, while end-to-end response time can be measured from image capture to the controller's final status message. The paper should state which definition is used for every reported time.

1. Result Recording Template
The following table provides a format for recording ex-perimental observations. The entries are intentionally left for measurement during physical testing.

X. DISCUSSION

1. Integration Of Vehicle Identity And Slot Occupancy
The main design feature of the proposed system is the combination of vehicle identity verification and slot-level occupancy monitoring. These functions address different parts of the parking process. The recognition module determines which registered vehicle has arrived, while the IR sensorsdetermine which parking space has changed its occupancy state.Using only a reservation application would assign a slot but would not establish whether the vehicle uses that slot. Using only occupancy sensors would show which spaces are occupied but would not identify whether the occupying vehicle has a valid booking. The proposed workflow links the two operations by transmitting the expected slot after successful vehicle verification.

2. Role Of The Cloud Recognition Service
The use of the Plate Recognizer Cloud API separates the recognition engine from the local hardware controller. The host computer captures the image and communicates with the API, while the Arduino performs the sensor monitoring and buzzer control. This division reduces the processing responsibility placed on the microcontroller.However, cloud-based recognition introduces a dependency on internet connectivity, API availability, and valid credentials. Recognition time may also vary with network conditions. These factors should be included in the experimental evaluation rather than being treated as constant system properties.

3. Role Of The Arduino Controller
The Arduino Uno provides a compact controller for reading the three sensor inputs and operating the buzzer. Its responsibili-ties are intentionally limited to real-time sensor-state monitoring and comparison with the expected slot received from the host computer.The host computer performs the tasks that require network access and database operations. This division creates a clear interface between the software and hardware portions of the prototype. It also means that the Arduino depends on receiving a valid slot command before it can perform the wrong-slot comparison.

4. Interpretation Of Prototype Scope
The current design is a small-scale prototype and should not be interpreted as a complete commercial parking man-agement platform. It demonstrates the integration of a web reservation workflow, cloud-based number plate recognition, serial communication, occupancy sensors, and a local warning device.

© 2026 IJSRET

The three-slot configuration is suitable for demonstrating the core operation. Larger installations would require additional sensors, a more comprehensive controller architecture, stronger vehicle-to-slot association, and additional measures for moni-toring the health of the network and hardware components.

5. Practical Deployment Considerations

A practical installation must consider the physical envi-ronment in which the camera and sensors operate. Camera placement, illumination, vehicle speed, plate orientation, and environmental exposure can affect image quality. Sensor placement and calibration influence the reliability of occupancy detection.

TABLE VIII Suggested Experimental Observation Sheet

Trial	Test condition	Expected outcome	Observed outcome	Remarks
1	Active reservation and correct plate	Assigned slot retrieved	recorded sucessfully	recorded sucessfully
2	Plate with no active reservation	Verification rejected	recorded sucessfully	recorded sucessfully
3	Correct slot occupancy	No wrong-slot alert	recorded sucessfully	recorded sucessfully
4	Incorrect slot occu-pancy	Buzzer activated	recorded sucessfully	recorded sucessfully
5	API/network unavail-able	Verification reported in-complete	recorded sucessfully	recorded sucessfully
6	USB serial link dis-connected	Communication failure reported	recorded sucessfully	recorded sucessfully
7	Sensor state transition	Correct slot event iden-tified	recorded sucessfully	recorded sucessfully

The system should also be installed so that wiring is protected and the controller is supplied with a stable power source. The buzzer should be positioned so that the warning can be heard by the intended operator without creating unnecessary disturbance. These factors should be examined before moving from a laboratory prototype to a larger parking facility.

6. Reproducibility Of Evaluation

For the results to be reproducible, the final paper should identify the hardware configuration, sensor model, camera arrangement, software versions, API configuration used for testing, and the number of trials performed. The recognition test images should be captured under documented conditions, and the definition of each performance metric should be stated clearly.

The final results section should distinguish successful tests from tests that were not completed because of network, camera, or hardware faults. Reporting these conditions provides a more accurate picture of the prototype than presenting only successful demonstrations.

I. Advantages And Applications

1. Advantages

The proposed system provides the following advantages:

- Supports advance parking reservations.
- Uses vehicle registration numbers for booking verification.
- Provides automatic number plate recognition through a cloud API.
- Monitors individual parking slot occupancy.
- Compares reserved and occupied slot numbers.
- Provides an audible wrong-slot warning.
- Supports automatic entrance gate operation using an Arduino- controlled servo motor.
- Reduces the need for continuous manual monitoring.
- Uses commonly available hardware components.
- Supports future expansion to additional parking spaces.

2. Applications

The system can be adapted for use in:

- Educational institutions.
- Shopping complexes.
- Hospitals.
- Office buildings.
- Apartment complexes.
- Commercial parking areas.
- Private parking facilities.

2. Limitations

The current prototype has certain limitations that must be considered.

- The initial hardware configuration supports only three parking slots.
- Number plate recognition performance depends on image quality, lighting, camera angle, and registration plate visibility.
- The Plate Recognizer Cloud API requires internet con-nectivity and valid API access.

© 2026 IJSRET

- IR sensors detect occupancy but cannot identify the vehicle registration number.
- The system requires a reliable association between the verified vehicle and its subsequent parking occupancy.
- Multiple simultaneous parking events cannot be indepen-dently associated with different vehicles using only one entrance camera and three binary IR sensors.
- Sensor readings may be affected by installation position, environmental conditions, and sensor calibration.
- The host computer is required for the recognition and web application functions.

3. Future Enhancements

1. Expansion of Parking Capacity
The prototype can be extended to support additional parking spaces by installing more sensors and modifying the controller software.

2. Improved Vehicle Tracking
Additional cameras or bay-level identification methods can be introduced to associate vehicles with their respective parking spaces.This improvement would help support multiple vehicles entering and parking at the same time.

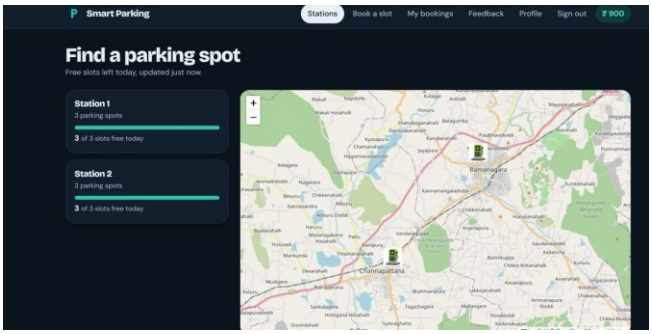

Fig. 3. Smart parking station dashboard showing available slots and map view

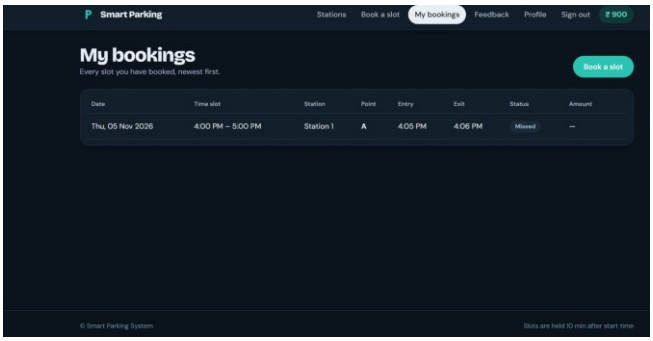

Fig. 4. Slot booking interface showing individual parking slots and availability.

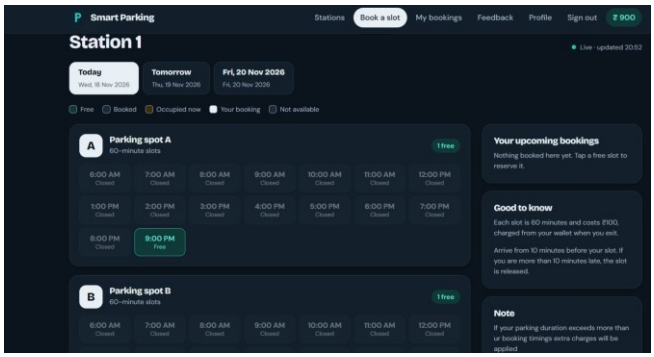

Fig. 5. My bookings page displaying reservation and parking session details.

3. Enhanced Recognition
Recognition performance can be improved through better camera positioning, controlled illumination, image quality monitoring, and suitable recognition settings.

4. Mobile Application
A mobile application can be developed to provide reservation management, slot availability, and booking notifications.

5. Payment Integration
A payment gateway can be introduced in a future version to support online parking fee collection. This functionality is outside the scope of the current prototype.

1. Cloud-Based Monitoring
Cloud-based services can be introduced for centralized monitoring and management of multiple parking locations.

2. Prototype And Application Screenshots
This section presents the implemented web interface and physical prototype. The screenshots demonstrate the station dashboard, reservation interface, booking history, user profile and wallet, serial communication test, and hardware arrange-ment.

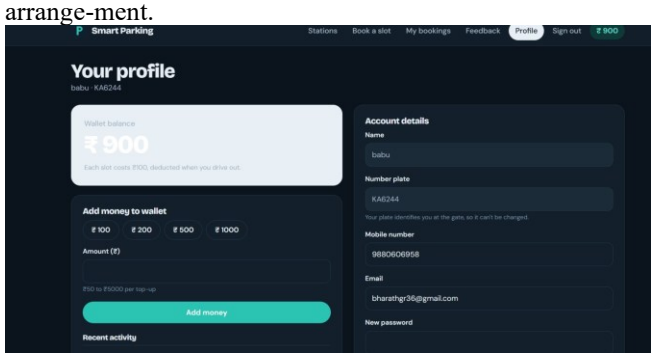

Fig. 6. User profile page with wallet balance and account details

© 2026 IJSRET

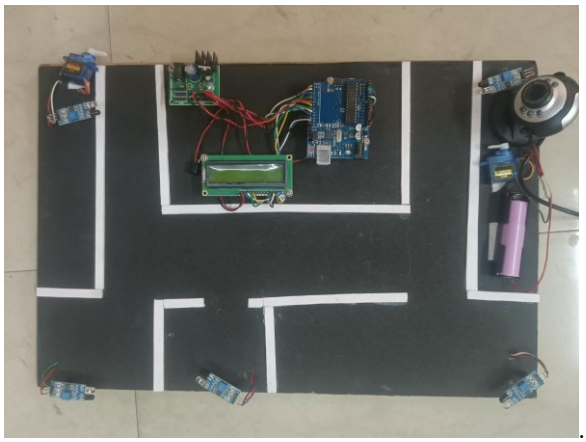

Fig. 7. Physical prototype showing Arduino Uno, sensors, display, buzzer, and servo components.

XI. CONCLUSION

This paper presents an iot-based smart parking reservation system integrating automatic number plate recognition and wrong-slot identification. The proposed prototype combines a flask-based web appli-cation, a reservation database, an entrance camera, the plate recognizer cloud api, an arduino uno, three ir sensors, a buzzer, and a servo motor.

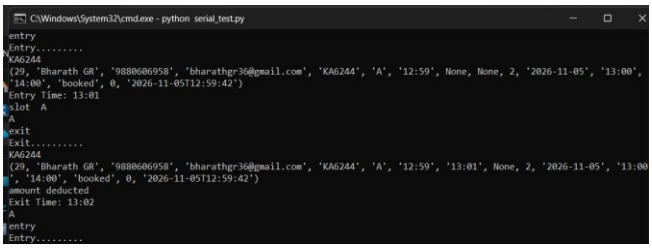

Fig. 8. Serial monitor test demonstrating entry and exit event processing.

Users reserve parking spaces using their vehicle registration numbers. When a vehicle reaches the entrance, the camera captures its registration plate and the cloud recognition service returns the recognized vehicle number.

The recognized number is verified against the active reservation database. After successful verification, the expected parking slot is sent to the Arduino Uno through USB serial communication.

The Arduino monitors the individual IR sensors and compares newly detected occupancy with the expected parking slot. When a mismatch occurs, the buzzer generates an audible warning. The Arduino-controlled servo motor operates the entrance gate after successful vehicle verification.

The proposed design demonstrates how vehicle recognition, web-based reservation management, and sensor-based occupancy monitoring can be integrated into a small-scale parking prototype. Further development can improve vehicle tracking, recognition reliability, and large-scale deployment.

REFERENCES

1. C. Biyik, Z. Allam, G. Pieri, S. Moroni, R. O'Fallon, S. Shatnawi, and S. W. Nam, "Smart parking systems: Reviewing the literature, architecture and ways forward," Smart Cities, vol. 4, no. 2, pp. 623–642, 2021.
2. A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, Feb. 2014.
3. A. G. Khoso, N. Zahid, and I. Khan, "Enhanced IoT-enabled smart parking system with RFID integration and line assistance," Transportation Research Procedia, vol. 84, pp. 346–353, 2025, doi: 10.1016/j.trpro.2025.03.082.
4. S. Patel and A. Mehta, "Smart Parking System Using IoT with Real-Time Slot Monitoring," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 13, no. 5, pp. 421–428, 2022.
5. R. Kumar and K. Reddy, "IoT-Based Smart Parking System with Reservation and Access Control," International Journal of Engineering Trends and Technology (IJETT), vol. 71, no. 3, pp. 115–121, 2023.
6. A. Elfaki, M. Ahmed, and S. Hassan, "Design and Implementation of an Intelligent Smart Parking System Using IoT," Journal of Smart Cities and Society, vol. 2, no. 1, pp. 45–53, 2023.
7. M. Rajabioun and P. Ioannou, "Smart Parking Systems: Technologies, Challenges, and Future Trends," IEEE Intelligent Transportation Systems Magazine, vol. 13, no. 2, pp. 43–56, 2021.
8. J. Miao, "An IoT-Based Smart Parking System Using Cloud Computing," IEEE Access, vol. 7, pp. 117394–117404, 2019.
9. M. Khoso, S. A. Memon, and A. A. Shaikh, "Enhanced IoT-Enabled Smart Parking System with RFID and Sensor Integration," Sustainable Cities and Society, vol. 92, Art. no. 104463, 2025.
10. R. Singh and P. Verma, "Web-Based Smart Parking Reservation System for Smart Cities," International Journal of Computer Applications, vol. 184, no. 9, pp. 25–31, 2022.
11. L. Zhang, Y. Chen, and H. Li, "IoT-Based Smart Parking Management System with Real-Time

© 2026 IJSRET

Monitoring," Sensors, vol. 23, no. 6, Art. no. 3124, 2023.
12. A. Sharma and V. Gupta, "Smart Parking System Using ESP32 and RFID Technology," International Journal of Emerging Technologies and Innovative Research (IJETIR), vol. 10, no. 4, pp. 189–195, 2023.

© 2026 IJSRET