NAAN MUDHALVAN

JEPPIAAR ENGINEERING COLLEGE

PHASE 3

DEPARTMENT OF AI&DS

2ND YEAR

TRAFFIC FLOW OPTIMIZATION

Parking Management following system approach for the city

Team Leader : Kamaleshwaran M

Team Members : Sharun P

Sakthi G

Sanjay B

Sanjay K

Implementation of Project

Title: Parking Management following system approach for the city

Objective

The goal of Phase 3 is to implement the core components of the Smart Parking Management System based on the plans and innovative solutions developed during Phase 2. This includes real-time parking detection, reservation features, dynamic pricing, AI-powered enforcement, and basic data analytics integration.

1. Real-Time Parking Detection System

Overview

Sensors and cameras will be installed in parking areas to detect and update real-time space availability.

Implementation

• IoT ground sensors and cameras deployed in key areas.
• Connectivity to a central system for status update and dashboard reflection.
• Data transmission over Wi-Fi/LoRaWAN to the server for user accessibility.

Outcome

Drivers can view live parking slot availability through a mobile app or digital signage, reducing time and fuel wastage.

2. Reservation and Pricing System

Overview

A digital system for reserving parking slots and charging users dynamically based on time and demand.

Implementation

• Mobile/web application to reserve slots and make digital payments.
• Integration of time-sensitive and location-sensitive pricing algorithms.
• Backend dashboard for managing pricing schemes and slot availability.

Outcome

Enables better space utilization and incentivizes users to park in less crowded areas, enhancing system efficiency.

3. Enforcement via AI & Cameras

Overview

Automated monitoring of illegal or expired parking using number plate recognition cameras and AI logic.

Implementation

• Cameras capture vehicle data and detect parking violations.
• AI cross-verifies with the parking database and triggers alerts.
• Notifications or e-challans sent to violators via SMS or app.

Outcome

Increases law compliance, reduces manual patrolling, and provides seamless enforcement across the city.

4. Data Analytics and Reporting

Overview

Data from parking systems is collected and used for traffic planning, forecasting demand, and improving policies.

Implementation

• Centralized cloud platform for data aggregation.
• Reports generated on occupancy, revenue, peak hours.
• Used by city officials for long-term parking infrastructure decisions.

Outcome

Empowers data-driven decision-making for smart city planning.

5. Security and Privacy Measures

Overview

Protecting user and vehicle data is critical. This phase includes basic data encryption and secure access protocols.

Implementation

• All user data encrypted in transit and at rest.
• Secure login for users and admins.
• Storage on encrypted servers with limited access.

Outcome

Ensures compliance with data privacy standards while building user trust.

6. Testing and Feedback

Overview

Testing the full system including app interface, sensors, enforcement logic, and data dashboards.

Implementation

• Pilot deployment in select parking zones.
• User testing for app experience and accuracy.
• Feedback gathered for refining system performance.

Outcome

Identifies bugs and usability issues, prepares the system for city-wide scaling in Phase 4.

Challenges and Solutions

1. Hardware Failures

- Solution: Regular maintenance and real-time alerts for sensor malfunction.

2. User App Adoption

- Solution: Incentives and awareness campaigns for early users.

3. Privacy Concerns

- Solution: Transparent policies and user consent on data collection.

4. Network Downtime

- Solution: Offline logging on sensors and sync upon reconnection.

Outcomes of Phase 3

1. Functional Real-Time Parking Detection System.

2. Operational Reservation and Pricing Platform.

3. AI-Based Monitoring and Enforcement System in Use.

4. Secured Data Handling and Analytics Integration.

5. Testing Reports and User Feedback Collected.

Next Steps for Phase 4

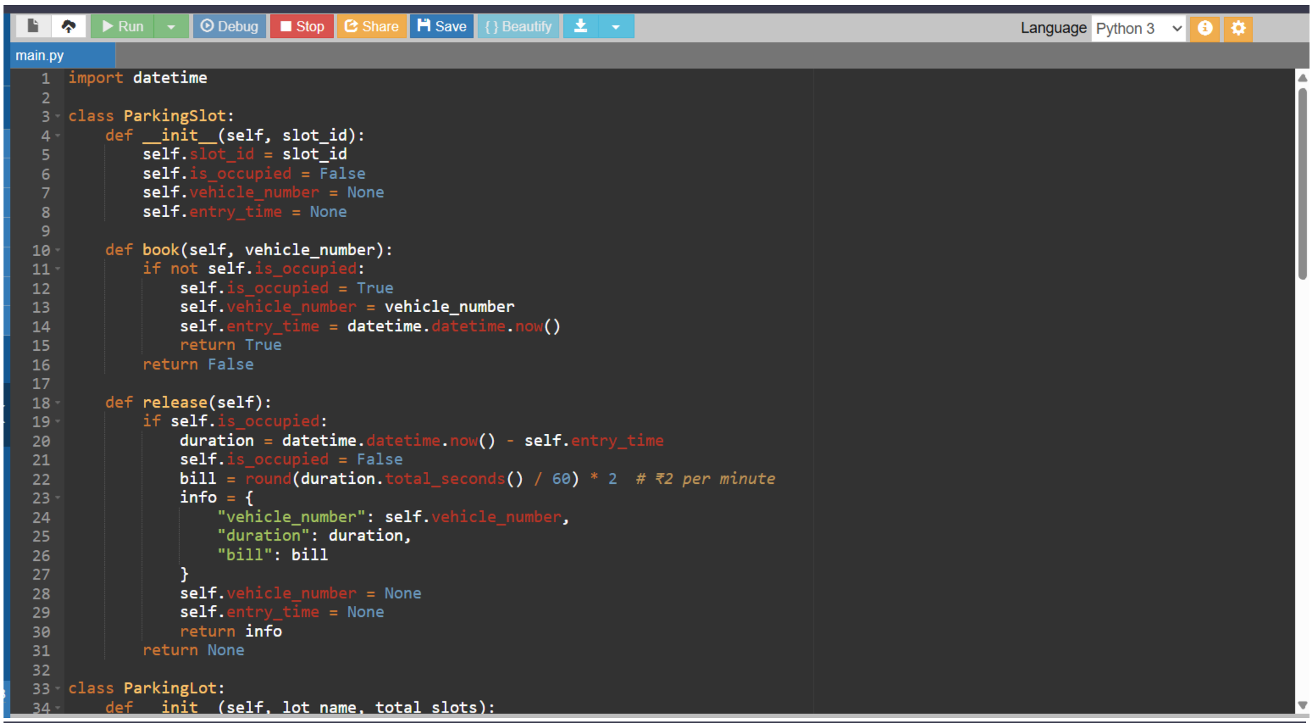
1. Expand Deployment to More Zones Based on Feedback.

2. Refine Pricing and Detection Algorithms.

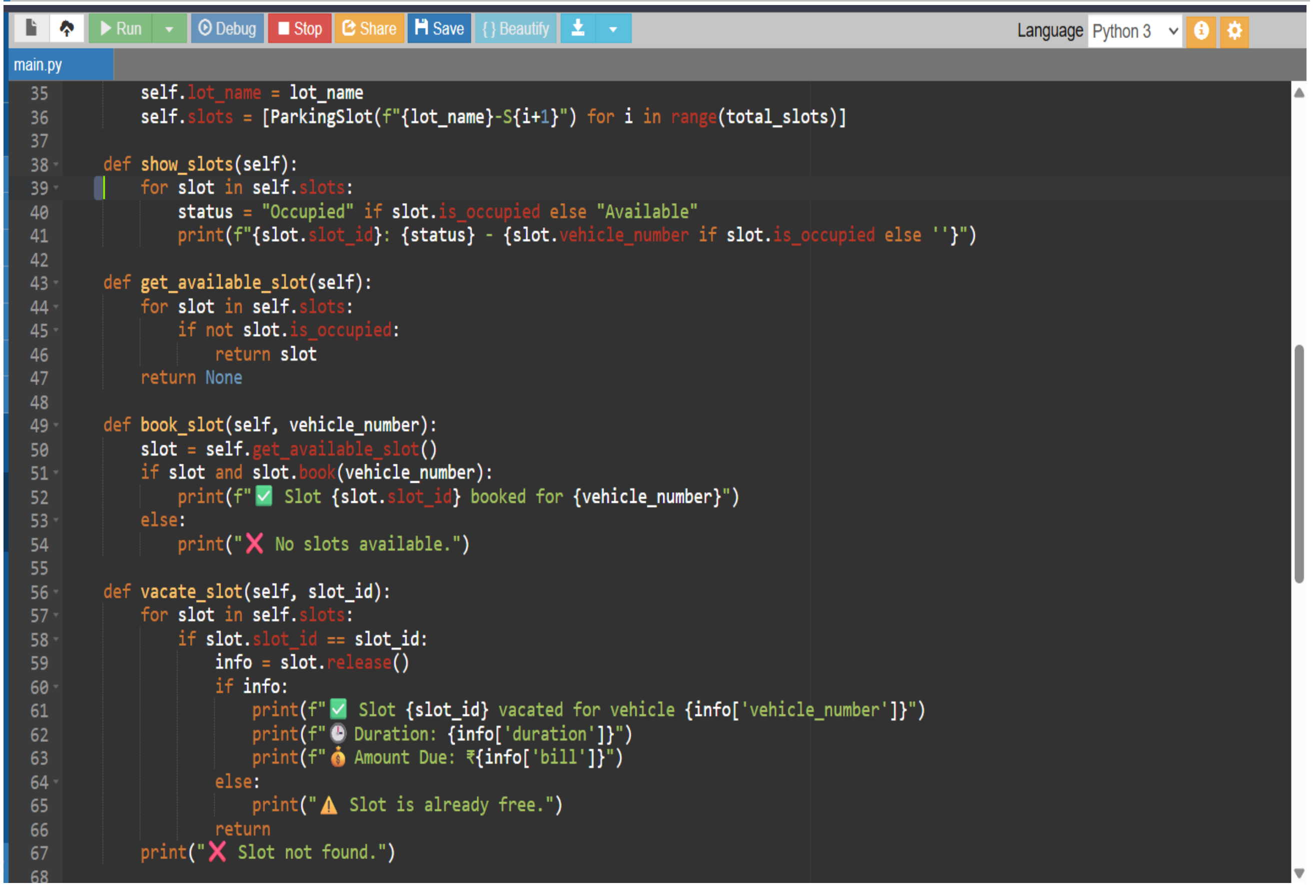
3. Add Multilingual Support and Voice Commands in App.

4. Increase Integration with Public Transit and Other Smart City Services.

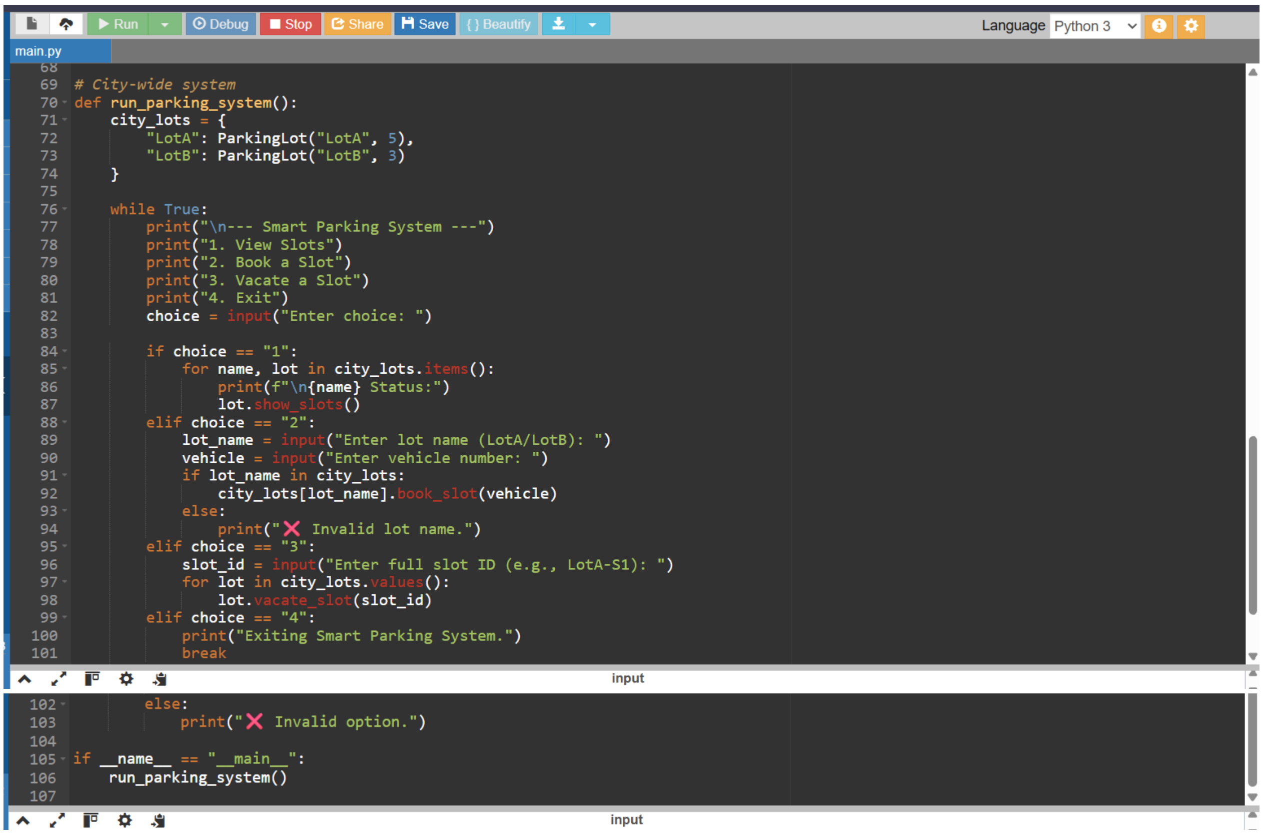
Source Code



```python
import datetime

class ParkingSlot:
    def __init__(self, slot_id):
        self.slot_id = slot_id
        self.is_occupied = False
        self.vehicle_number = None
        self.entry_time = None

    def book(self, vehicle_number):
        if not self.is_occupied:
            self.is_occupied = True
            self.vehicle_number = vehicle_number
            self.entry_time = datetime.datetime.now()
            return True
        return False

    def release(self):
        if self.is_occupied:
            duration = datetime.datetime.now() - self.entry_time
            self.is_occupied = False
            bill = round(duration.total_seconds() / 60) * 2  # ₹2 per minute
            info = {
                "vehicle_number": self.vehicle_number,
                "duration": duration,
                "bill": bill
            }
            self.vehicle_number = None
            self.entry_time = None
            return info
        return None

class ParkingLot:
    def __init__(self, lot_name, total_slots):
```



```python
        self.lot_name = lot_name
        self.slots = [ParkingSlot(f"{lot_name}-S{i+1}") for i in range(total_slots)]

    def show_slots(self):
        for slot in self.slots:
            status = "Occupied" if slot.is_occupied else "Available"
            print(f"{slot.slot_id}: {status} - {slot.vehicle_number if slot.is_occupied else ''}")

    def get_available_slot(self):
        for slot in self.slots:
            if not slot.is_occupied:
                return slot
        return None

    def book_slot(self, vehicle_number):
        slot = self.get_available_slot()
        if slot and slot.book(vehicle_number):
            print(f"✅ Slot {slot.slot_id} booked for {vehicle_number}")
        else:
            print("❌ No slots available.")

    def vacate_slot(self, slot_id):
        for slot in self.slots:
            if slot.slot_id == slot_id:
                info = slot.release()
                if info:
                    print(f"✅ Slot {slot_id} vacated for vehicle {info['vehicle_number']}")
                    print(f"🕒 Duration: {info['duration']}")
                    print(f"💰 Amount Due: ₹{info['bill']}")
                else:
                    print("⚠️ Slot is already free.")
                return
        print("❌ Slot not found.")

```



```python
# City-wide system
def run_parking_system():
    city_lots = {
        "LotA": ParkingLot("LotA", 5),
        "LotB": ParkingLot("LotB", 3)
    }

    while True:
        print("\n--- Smart Parking System ---")
        print("1. View Slots")
        print("2. Book a Slot")
        print("3. Vacate a Slot")
        print("4. Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            for name, lot in city_lots.items():
                print(f"\n{name} Status:")
                lot.show_slots()
        elif choice == "2":
            lot_name = input("Enter lot name (LotA/LotB): ")
            vehicle = input("Enter vehicle number: ")
            if lot_name in city_lots:
                city_lots[lot_name].book_slot(vehicle)
            else:
                print("❌ Invalid lot name.")
        elif choice == "3":
            slot_id = input("Enter full slot ID (e.g., LotA-S1): ")
            for lot in city_lots.values():
                lot.vacate_slot(slot_id)
        elif choice == "4":
            print("Exiting Smart Parking System.")
            break
        else:
            print("❌ Invalid option.")

if __name__ == "__main__":
    run_parking_system()
```