STUDENT PARKING SLOT SYSTEM MANAGEMENT

CAVITE STATE UNIVERSITY - BACOOR CAMPUS

INTRODUCTION

Due to increasing student enrollment, campus parking has become a challenge, with traditional manual systems proving inefficient. To address this, a Student Parking Slot System using Python is proposed. This system will allow students to register, reserve, and monitor parking slots in real time, improving convenience and space utilization. The project aims to streamline parking, reduce campus traffic, and offer features like user authentication and reservation tracking through a simple, automated solution.

PROJECT OBJECTIVES

This project proposes the development of an automated Student Parking Slot System using Python to improve campus parking management. The system enables students to register, reserve, and monitor parking slots in real time, reducing congestion and optimizing space usage.

Develop an Efficient

Parking Slot Management System Create a user-friendly application using Python to manage and monitor student parking slots on campus in real-time.

Enhance Campus Parking Experience

Improve the overall efficiency of campus parking by minimizing time spent searching for available slots and preventing overcrowding.

Automate Slot Allocation and Booking

Implement features that allow students to check slot availability, reserve parking slots, and receive confirmations to reduce manual intervention.

Booking History and Logs

This feature keeps a detailed record of all parking slot bookings made by students. Each log includes information such as the student ID, booking date and time, slot number, and duration of the reservation.

PROJECT OVERVIEW

The purpose of the Student Parking System Management is to ease the process of assigning, keeping track of, and monitoring parking spaces on a college or university campus. Improving parking lot efficiency, accessibility, and organization particularly for enrolled students the system's main objective. With this system's user-friendly interface, students may monitor real-time availability and reserve parking spaces in advance. From an administrative perspective, the system allows school staff to monitor parking slot utilization, adjust slot status, and create parking activity reports

PROJECT OVERVIEW

General Features: □

Slot Reservation System □
Real-Time Park Slot Monitoring □
User-Friendly Interface □
Vehicle Information Management

Application and Tools to be used: □

Back-end: Python □
Front-end: N/A
□ Database: N/A □
Version Control: Git and GitHub for code tracking and collaboration.

PROJECT FEATURES

In order to make parking slot reservations and monitoring easier and more efficient in a campus setting, the Student Parking System Management project provides an array of crucial features.

▯ Slot Reservation System

Save a slots based on date and time.

Real-Time Park Slot Monitoring

View current availability of all parking slots on a visual indicators.

Vehicle Information Management

Park or Unpark registered vehicles linked to students.

User-Friendly Interface

Simple and Responsive for easy navigation.

SCOPES AND LIMITATIONS

The functionality of each component in the Student Parking System Management project is thoroughly explained in this section, along with the necessary inputs, validation procedures, expected behaviors, and the limits of their existing capabilities.

SCOPES AND LIMITATIONS

Parking Slot Reservation System

Required Inputs: Only specific names of their vehicles.

Data Validation:

o The chosen time slot needs to be accessible within the specified period.

o Entering time must be done during university business hours.

 Expected Behavior:

o Users can book parking slots in real time.

o Once reserved, the slot becomes unavailable to others

. **Limitations:**

o Reservations are limited to one active slots per users.

o No auto-cancellation

SCOPES AND LIMITATIONS

Real Time Park Monitoring

Required Inputs: None (The users only see the visual indicator)

Data Validation:

o When a reservation is made, the data is updated with the backend. o Only the most recent and accurate slot status is shown because of the system.

Expected Behavior:

o A visual interface allows users to view available, reserved, and occupied slots.

o Real-time, automatic slot status updates that don't require page refreshes.

SCOPES AND LIMITATIONS

Vehicle Information Management

□**Required Inputs:** Only specific names of their vehicles.

Data Validation:

o Before a reservation can be made, vehicle name field must be filled out.

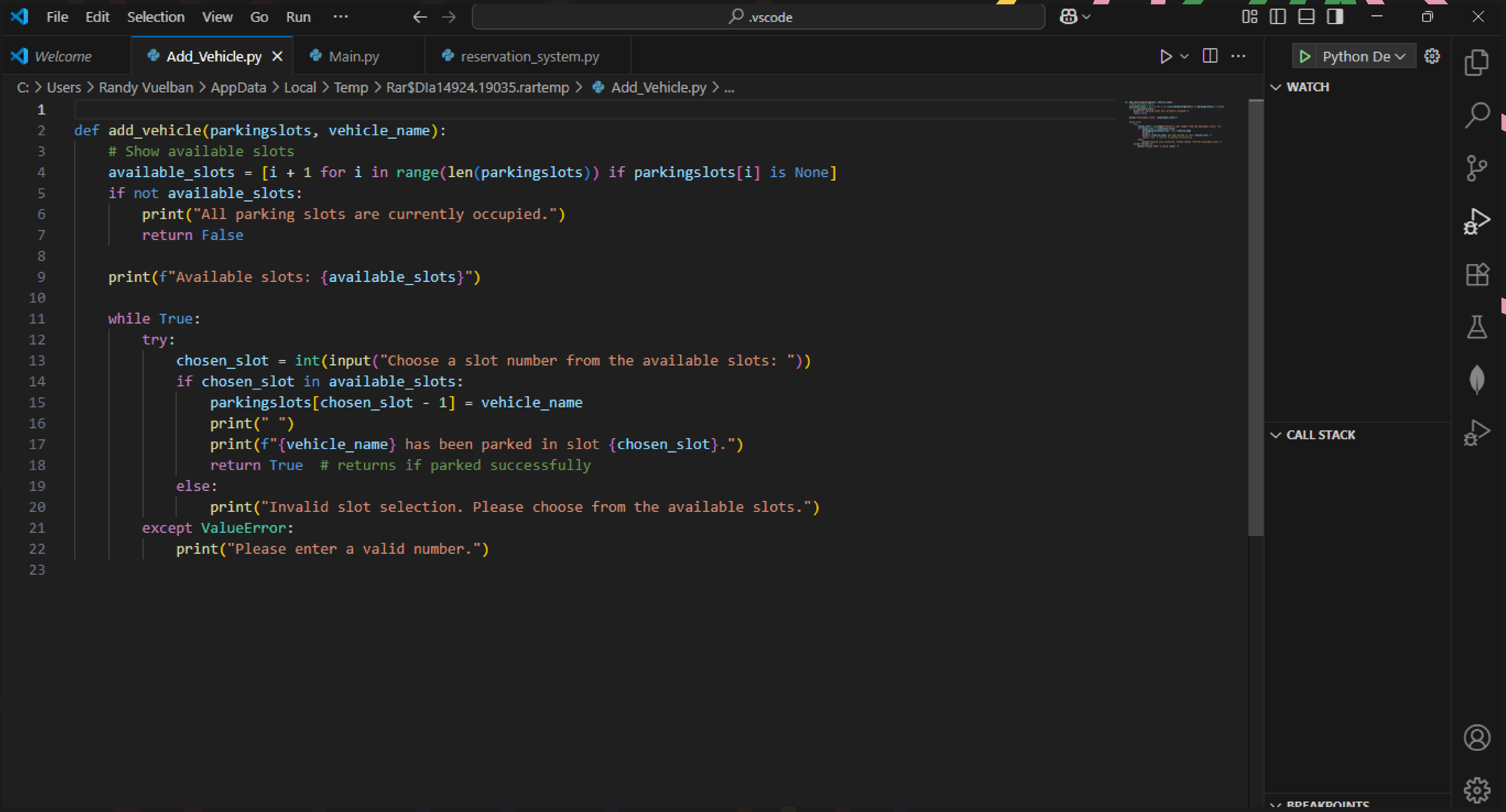
Expected Behavior:

o At the slots, users can park or unpark their registered vehicles. o Parking reservations are only accepted for student vehicles.

Limitations:

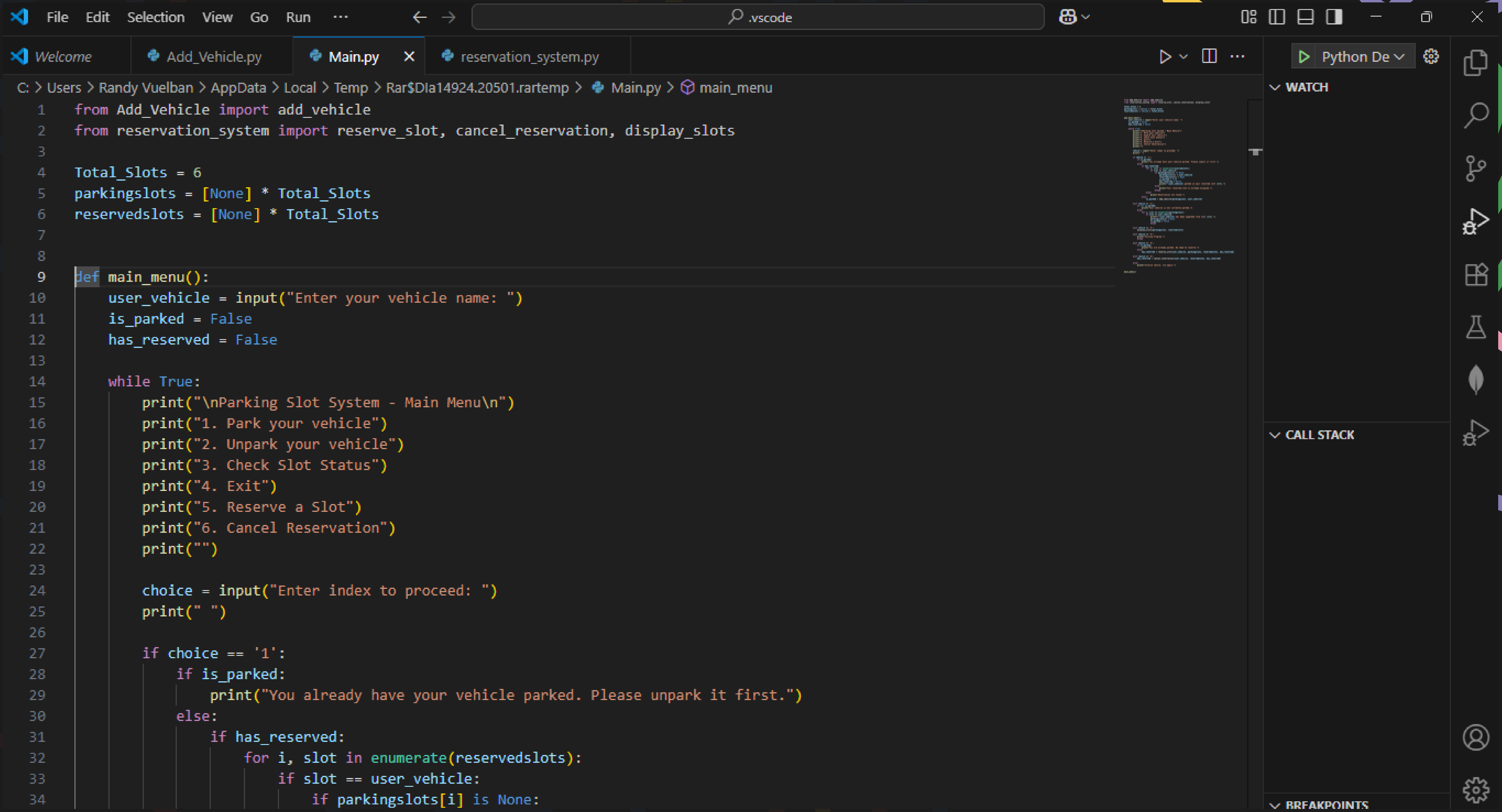
o The present version supports one vehicle per user.

SOURCE CODE :



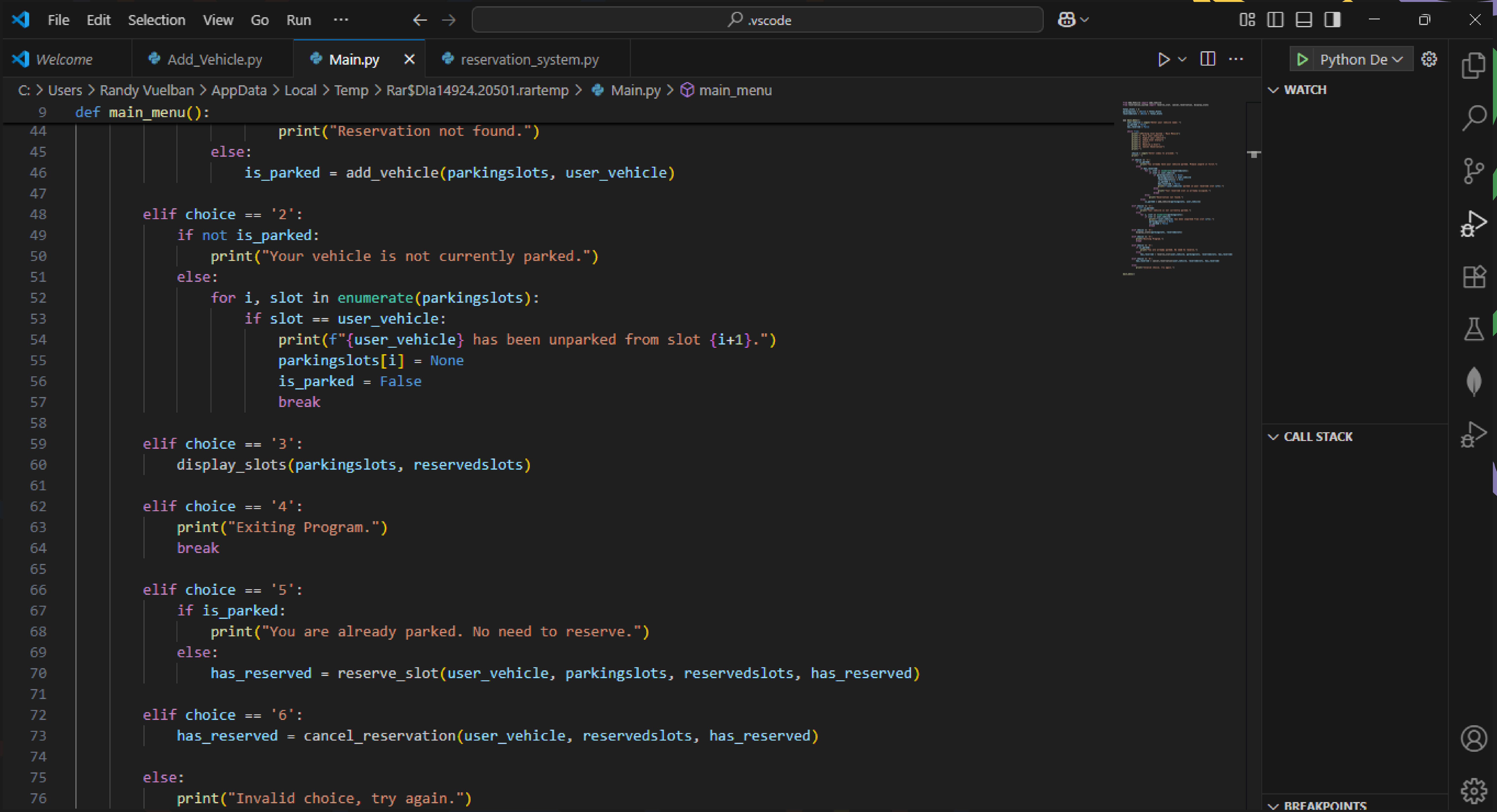
```python

def add_vehicle(parkingslots, vehicle_name):
    # Show available slots
    available_slots = [i + 1 for i in range(len(parkingslots)) if parkingslots[i] is None]
    if not available_slots:
        print("All parking slots are currently occupied.")
        return False

    print(f"Available slots: {available_slots}")

    while True:
        try:
            chosen_slot = int(input("Choose a slot number from the available slots: "))
            if chosen_slot in available_slots:
                parkingslots[chosen_slot - 1] = vehicle_name
                print(" ")
                print(f"{vehicle_name} has been parked in slot {chosen_slot}.")
                return True  # returns if parked successfully
            else:
                print("Invalid slot selection. Please choose from the available slots.")
        except ValueError:
            print("Please enter a valid number.")

```

SOURCE CODE:



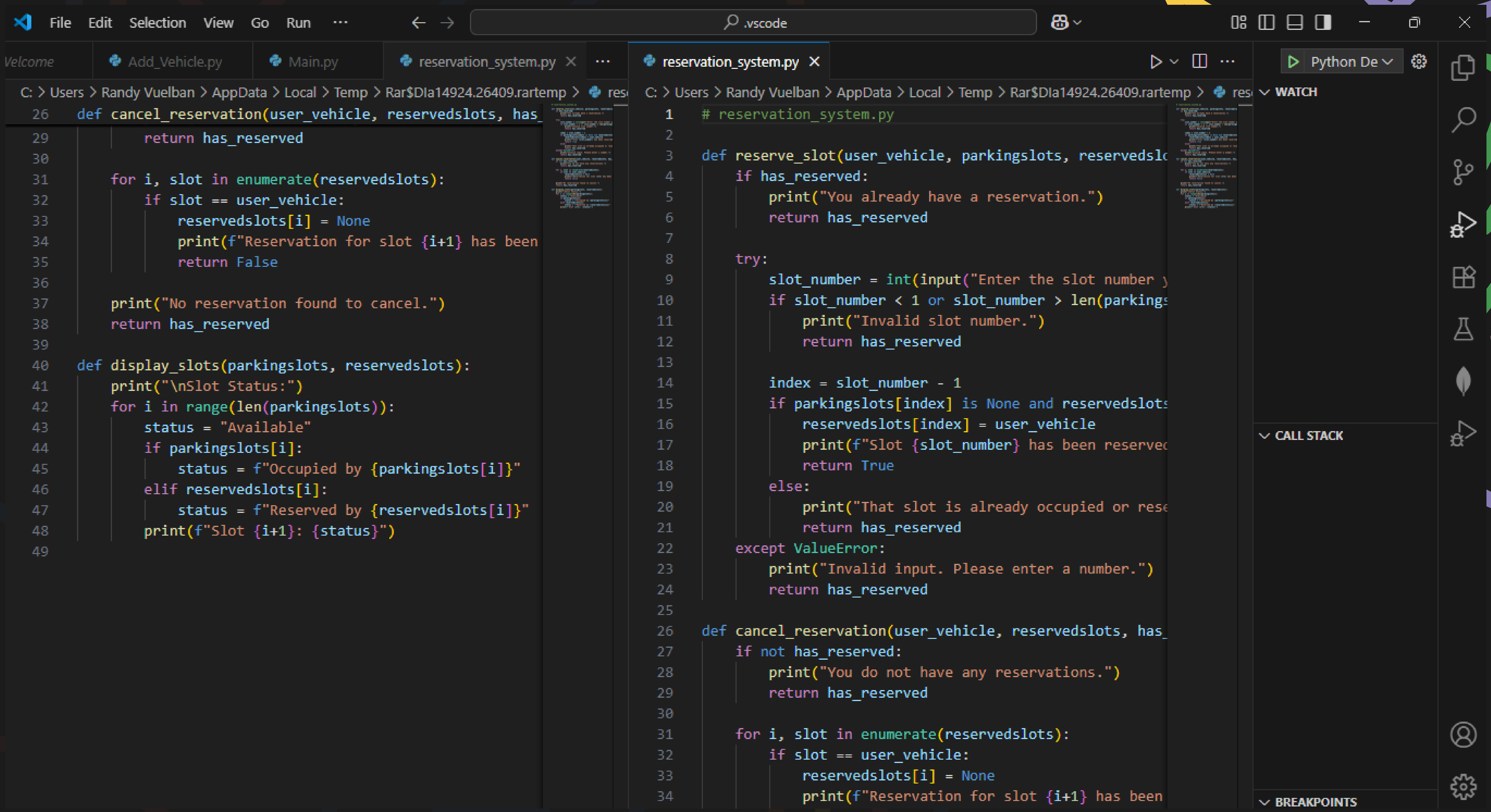
```python
from Add_Vehicle import add_vehicle
from reservation_system import reserve_slot, cancel_reservation, display_slots

Total_Slots = 6
parkingslots = [None] * Total_Slots
reservedslots = [None] * Total_Slots


def main_menu():
    user_vehicle = input("Enter your vehicle name: ")
    is_parked = False
    has_reserved = False

    while True:
        print("\nParking Slot System - Main Menu\n")
        print("1. Park your vehicle")
        print("2. Unpark your vehicle")
        print("3. Check Slot Status")
        print("4. Exit")
        print("5. Reserve a Slot")
        print("6. Cancel Reservation")
        print("")

        choice = input("Enter index to proceed: ")
        print(" ")

        if choice == '1':
            if is_parked:
                print("You already have your vehicle parked. Please unpark it first.")
            else:
                if has_reserved:
                    for i, slot in enumerate(reservedslots):
                        if slot == user_vehicle:
                            if parkingslots[i] is None:
```

SOURCE CODE :



```python
def main_menu():
                    print("Reservation not found.")
                else:
                    is_parked = add_vehicle(parkingslots, user_vehicle)

        elif choice == '2':
            if not is_parked:
                print("Your vehicle is not currently parked.")
            else:
                for i, slot in enumerate(parkingslots):
                    if slot == user_vehicle:
                        print(f"{user_vehicle} has been unparked from slot {i+1}.")
                        parkingslots[i] = None
                        is_parked = False
                        break

        elif choice == '3':
            display_slots(parkingslots, reservedslots)

        elif choice == '4':
            print("Exiting Program.")
            break

        elif choice == '5':
            if is_parked:
                print("You are already parked. No need to reserve.")
            else:
                has_reserved = reserve_slot(user_vehicle, parkingslots, reservedslots, has_reserved)

        elif choice == '6':
            has_reserved = cancel_reservation(user_vehicle, reservedslots, has_reserved)

        else:
            print("Invalid choice, try again.")
```

SOURCE CODE :

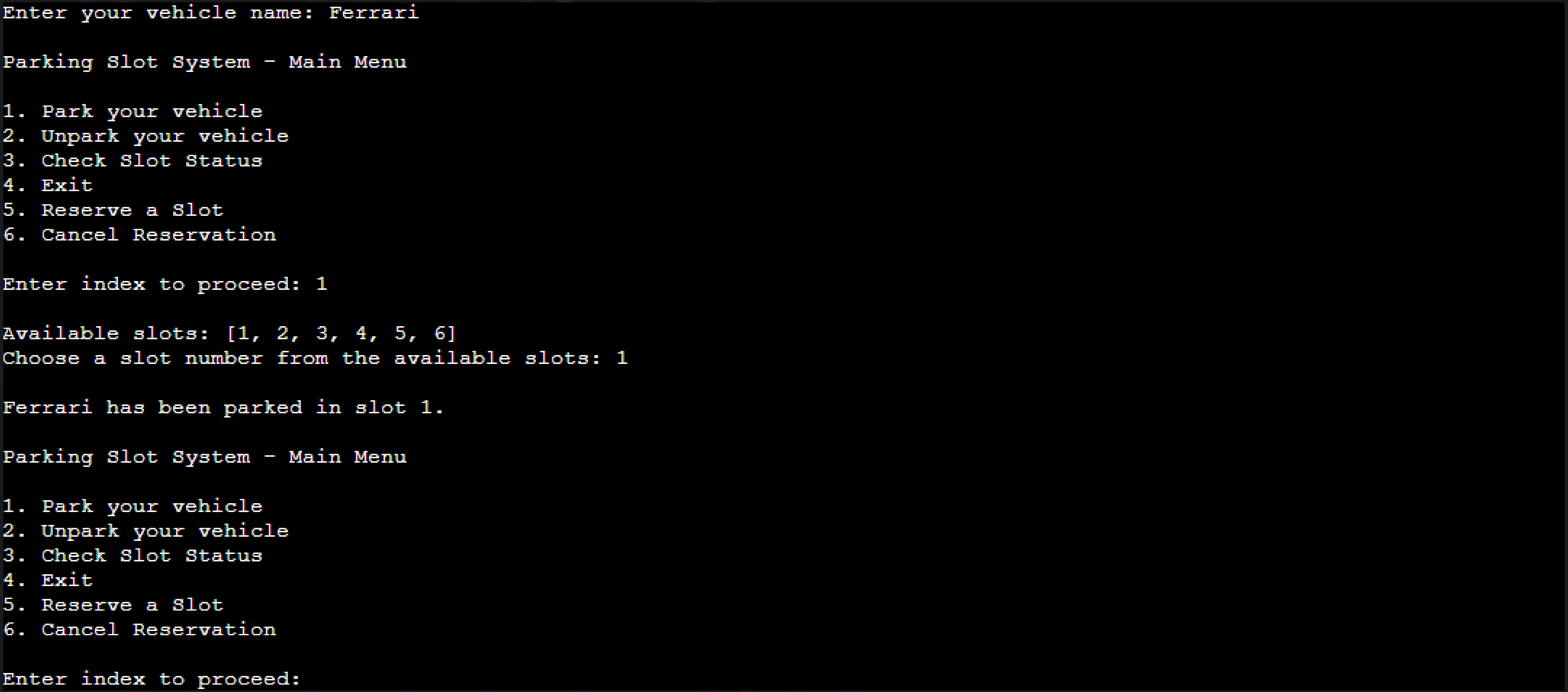


```
def cancel_reservation(user_vehicle, reservedslots, has_
        return has_reserved

    for i, slot in enumerate(reservedslots):
        if slot == user_vehicle:
            reservedslots[i] = None
            print(f"Reservation for slot {i+1} has been
            return False

    print("No reservation found to cancel.")
    return has_reserved

def display_slots(parkingslots, reservedslots):
    print("\nSlot Status:")
    for i in range(len(parkingslots)):
        status = "Available"
        if parkingslots[i]:
            status = f"Occupied by {parkingslots[i]}"
        elif reservedslots[i]:
            status = f"Reserved by {reservedslots[i]}"
        print(f"Slot {i+1}: {status}")

```

```
# reservation_system.py

def reserve_slot(user_vehicle, parkingslots, reservedslo
    if has_reserved:
        print("You already have a reservation.")
        return has_reserved

    try:
        slot_number = int(input("Enter the slot number y
        if slot_number < 1 or slot_number > len(parkings
            print("Invalid slot number.")
            return has_reserved

        index = slot_number - 1
        if parkingslots[index] is None and reservedslots
            reservedslots[index] = user_vehicle
            print(f"Slot {slot_number} has been reservec
            return True
        else:
            print("That slot is already occupied or rese
            return has_reserved
    except ValueError:
        print("Invalid input. Please enter a number.")
        return has_reserved

def cancel_reservation(user_vehicle, reservedslots, has_
    if not has_reserved:
        print("You do not have any reservations.")
        return has_reserved

    for i, slot in enumerate(reservedslots):
        if slot == user_vehicle:
            reservedslots[i] = None
            print(f"Reservation for slot {i+1} has been
```

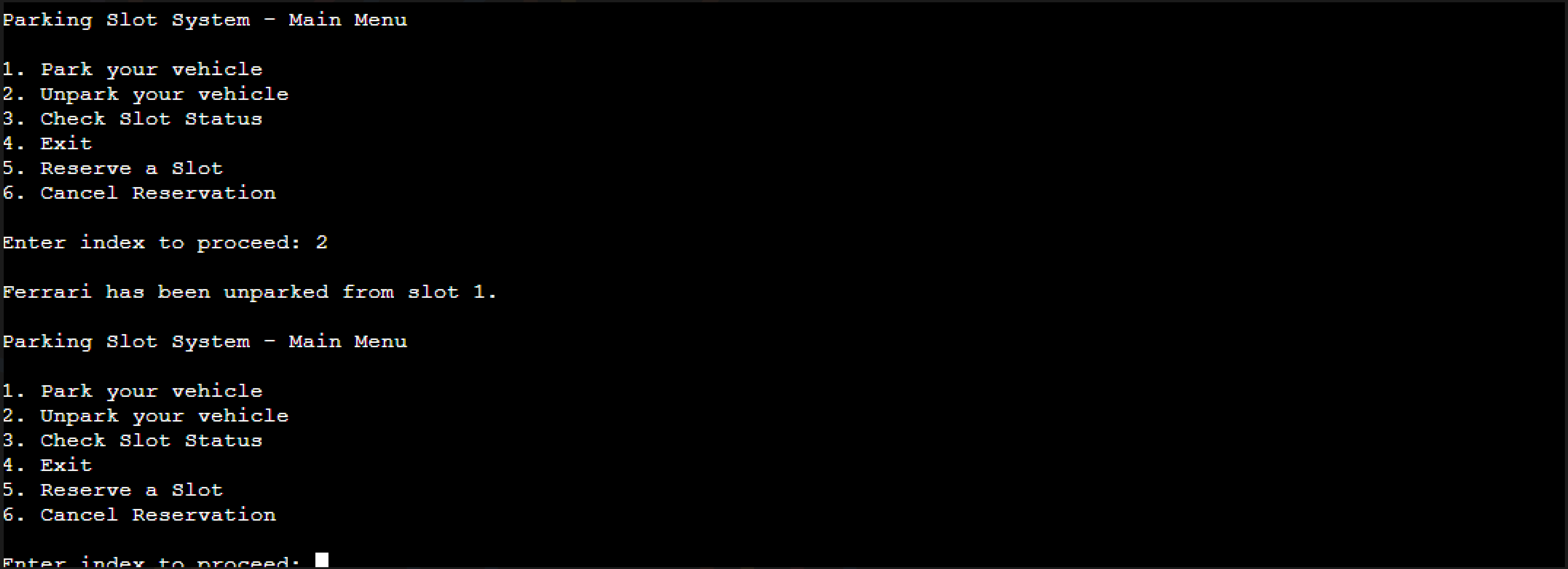
OUTPUT :

ENTERING OR PARKING YOU CAR



```
Enter your vehicle name: Ferrari

Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed: 1

Available slots: [1, 2, 3, 4, 5, 6]
Choose a slot number from the available slots: 1

Ferrari has been parked in slot 1.

Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed:
```

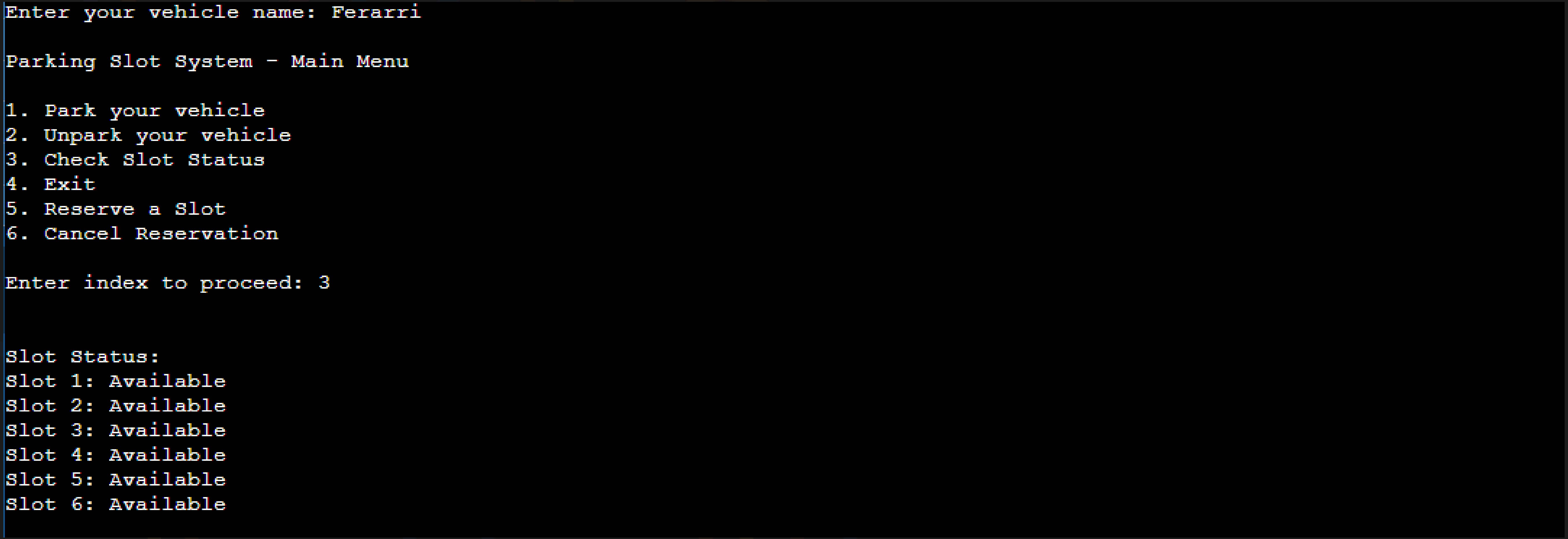
OUTPUT :
UNPARKING YOUR CAR



```
Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed: 2

Ferrari has been unparked from slot 1.

Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed:
```

OUTPUT :

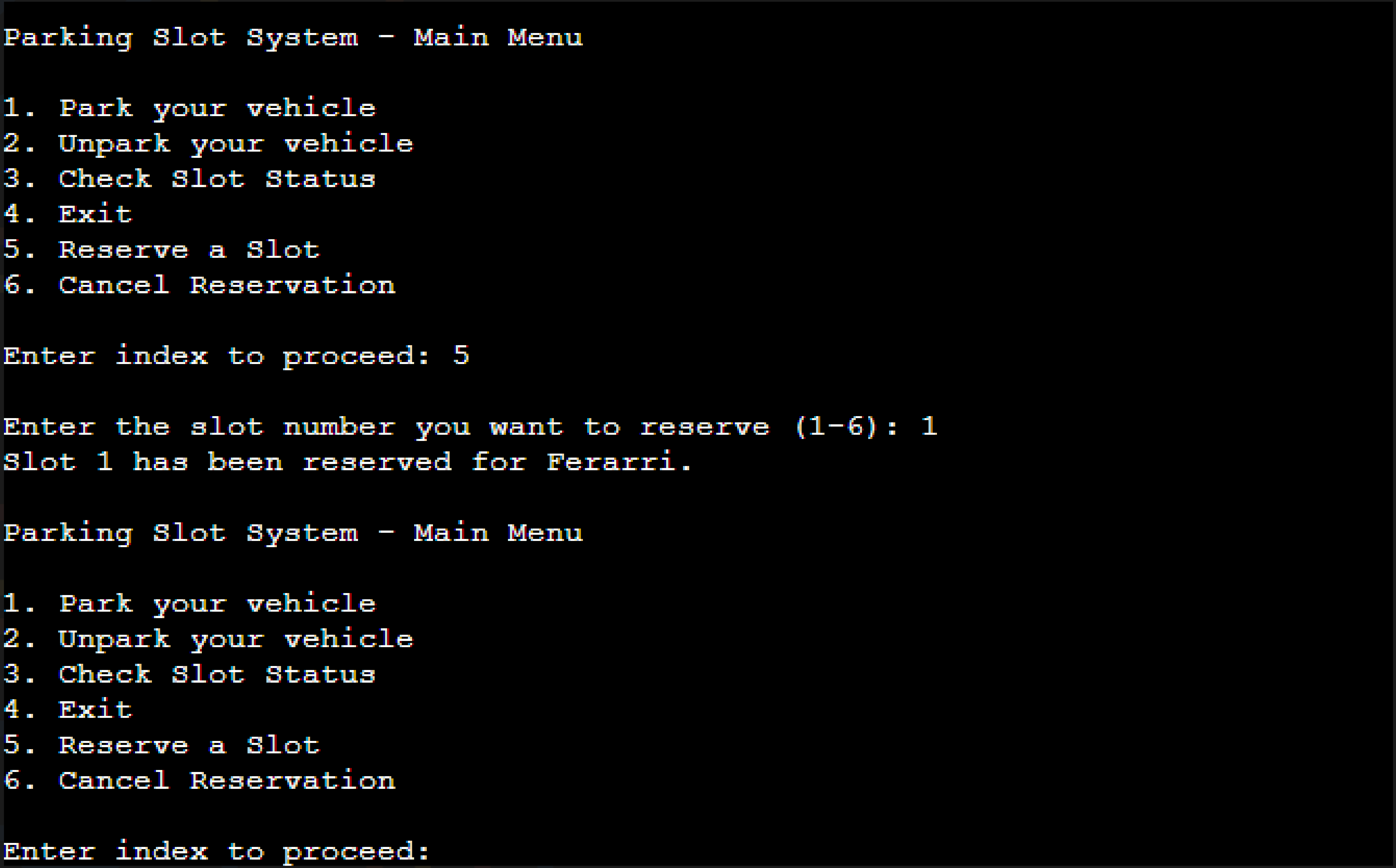
CHECKING SLOTS STATUS



```
Enter your vehicle name: Ferarri

Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed: 3


Slot Status:
Slot 1: Available
Slot 2: Available
Slot 3: Available
Slot 4: Available
Slot 5: Available
Slot 6: Available
```

OUTPUT :

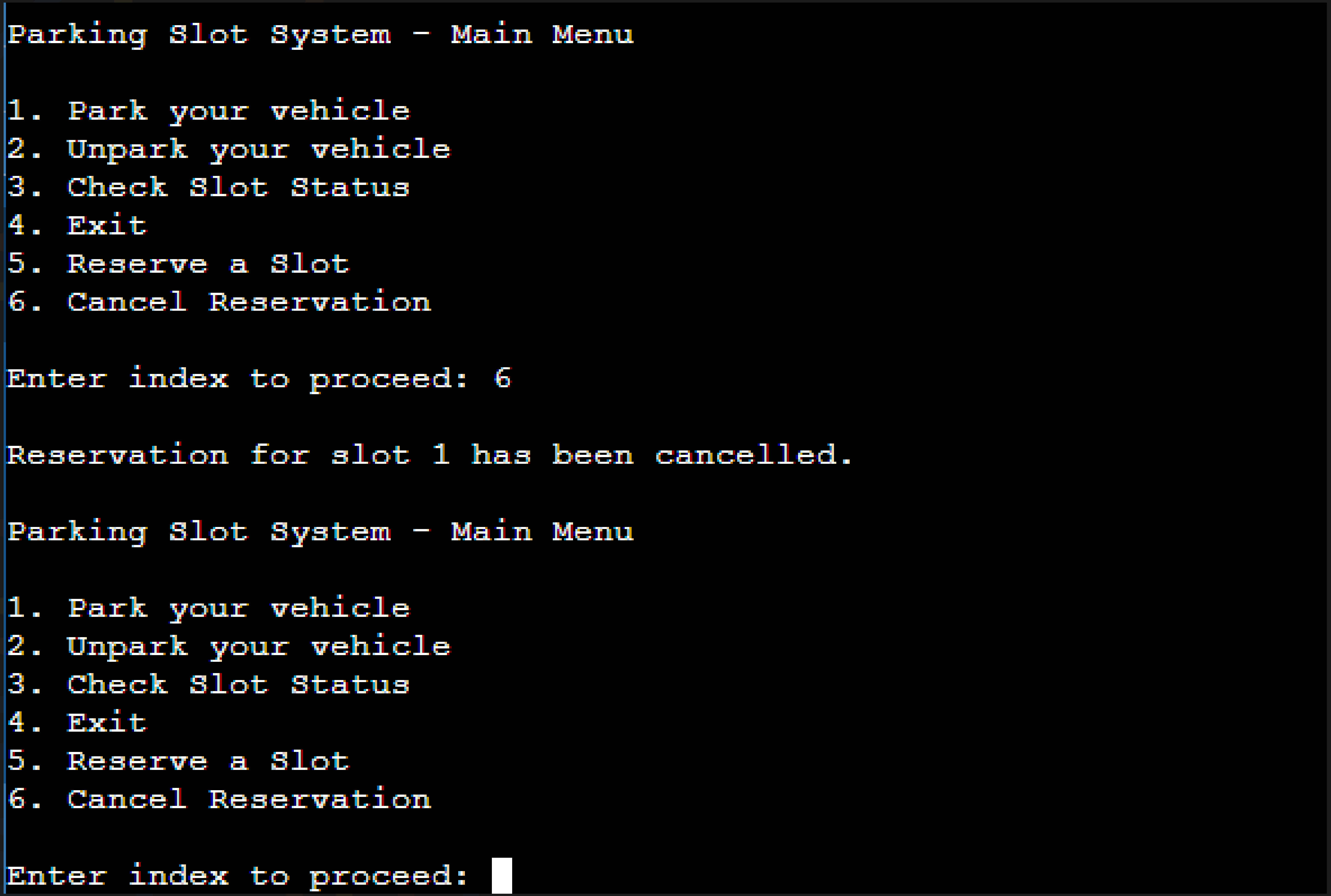
RESERVE A SLOT



```
Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed: 5

Enter the slot number you want to reserve (1-6): 1
Slot 1 has been reserved for Ferarri.

Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed:
```

OUTPUT :

CANCEL RESERVATION



```
Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed: 6

Reservation for slot 1 has been cancelled.

Parking Slot System - Main Menu

1. Park your vehicle
2. Unpark your vehicle
3. Check Slot Status
4. Exit
5. Reserve a Slot
6. Cancel Reservation

Enter index to proceed:
```

THANK YOU FOR LISTENING!

MEMBERS:
Randy Jr. Vuelban
John Michael Antalan
Charlie Plandiano