To create a smart parking system using Python and a Raspberry Pi, you will need the following:

- Raspberry Pi
- Ultrasonic sensors (one for each parking space)
- Breadboard and jumper wires
- Power supply for the Raspberry Pi and sensors
- Enclosure for the Raspberry Pi and sensors

Once you have all of your components, follow these steps:

1. Set up the sensors: Mount the ultrasonic sensors in the parking lot, one for each parking space. Make sure to mount the sensors at a height where they can accurately detect the presence of vehicles.
2. Connect the sensors to the Raspberry Pi: Connect the ultrasonic sensors to the Raspberry Pi according to the sensor's datasheet.
3. Install the necessary Python libraries: Install the necessary Python libraries, such as `RPi.GPIO` and `time`.
4. Write a Python script to read the ultrasonic sensors and determine which parking spaces are occupied: Here is a simple example of a Python script for a smart parking system:

```
Python
```

```python
#Libraries
import RPi.GPIO as GPIO
import time

#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)

#set GPIO Pins
GPIO_TRIGGER = 18
GPIO_ECHO = 24

#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2

    return distance

if __name__ == '__main__':
    try:
        while True:
            dist = distance()
            print ("Measured Distance = %.1f cm" % dist)
            time.sleep(1)

        # Reset by pressing CTRL + C
```

```
    except KeyboardInterrupt:
        print("Measurement stopped by User")
        GPIO.cleanup()
```

5.Deploy the Python script to the Raspberry Pi: You can deploy the Python script to the Raspberry Pi using a variety of methods, such as using a USB drive or an SSH connection.

6.Create a user interface: You can create a user interface for your smart parking system in a variety of ways, such as developing a web app or mobile app. The user interface should allow users to view the occupancy status of the parking spaces in real time.

Once you have completed these steps, you will have a working smart parking system. You can now deploy the system to your parking lot and start using it!

Safety guidelines:

- Be sure to mount the ultrasonic sensors in a safe location where they will not be damaged by vehicles or pedestrians.
- Make sure to enclose the Raspberry Pi and sensors in a weatherproof enclosure.
- Be careful not to overload the power supply.
- Do not connect the Raspberry Pi or sensors to a live electrical outlet.

## Wiring

There are four pins on the ultrasound module that are connected to the Raspberry:

- VCC to Pin 2 (VCC)
- GND to Pin 6 (GND)
- TRIG to Pin 12 (GPIO18)
- connect the 330Ω resistor to ECHO.  On its end you connect it to Pin 18 (GPIO24) and through a 470Ω resistor you connect it also to Pin6 (GND).

We do this because the GPIO pins only tolerate maximal 3.3V. The connection to GND is to have a obvious signal on GPIO24. If no pulse is sent, the signal is 0 (through the connection with GND), else it is 1.

If there would be no connection to GND, the input would be undefined if no signal is sent (randomly 0 or 1), so ambiguous.
Here is the structure as a circuit diagram: