i.mobilothon 3.0

The Life-Saving Innovation

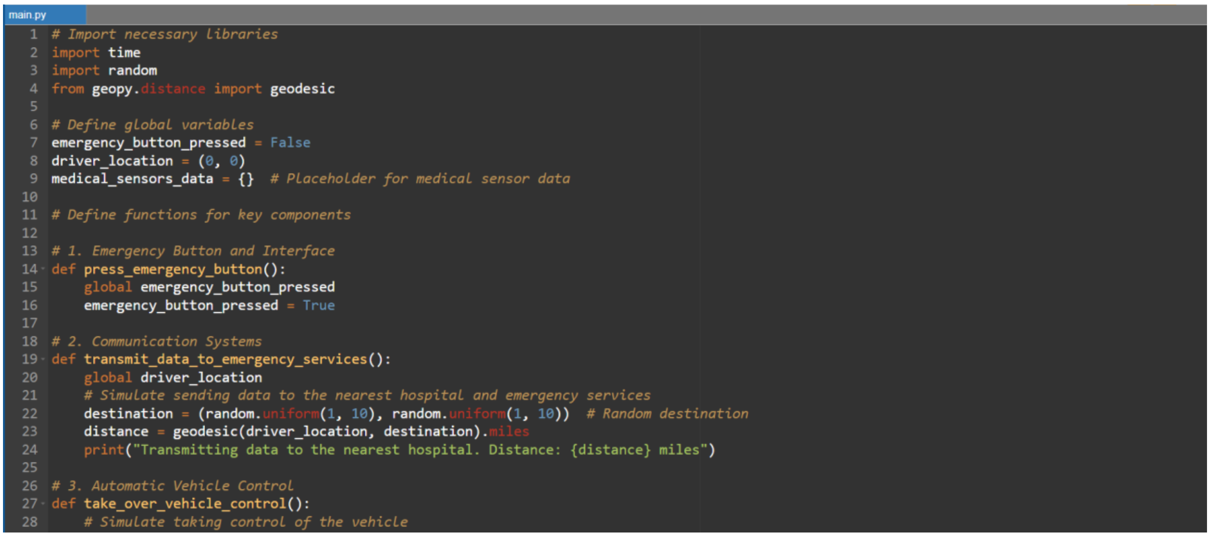
Member Name: Rebecca Bhattacharjee

Team Name: VolkswagenAspirer (Solo member)

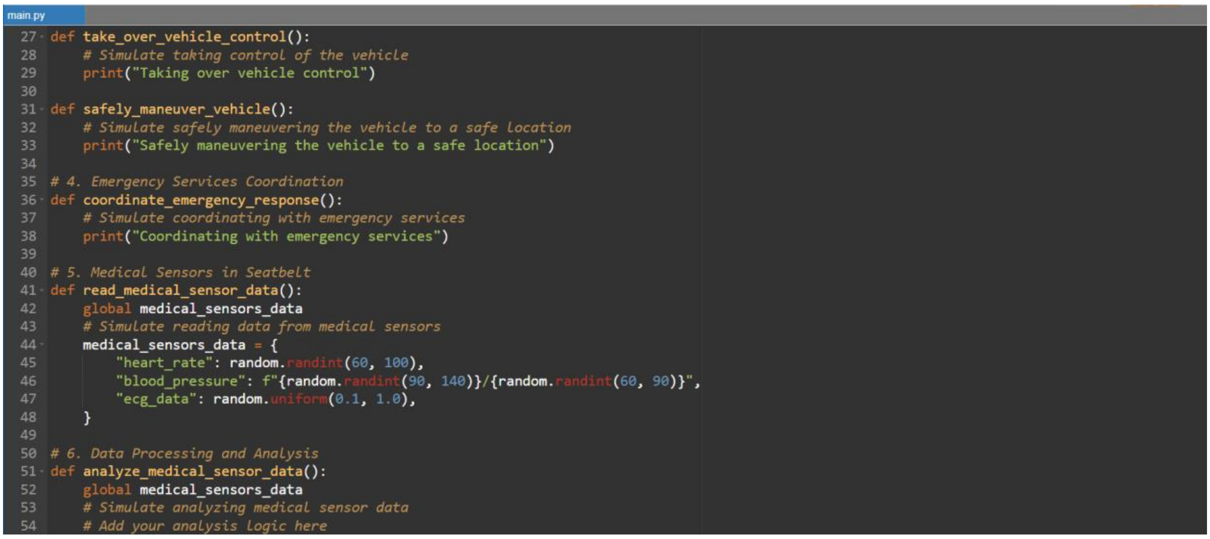
Round 3: Prototype Submission

Source Code

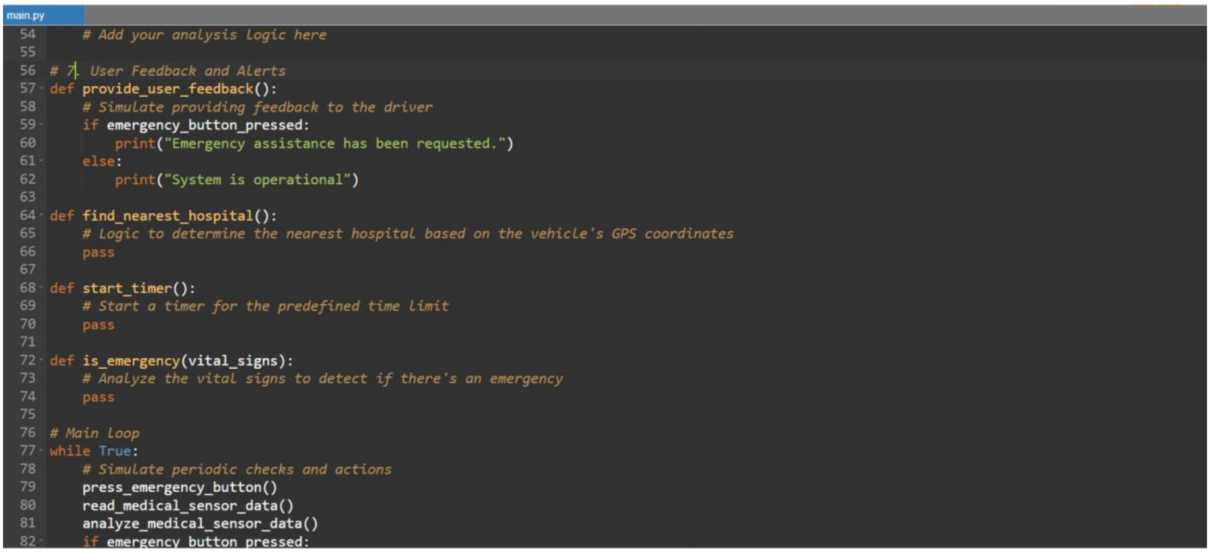
Creating a source code for such a complex and safety-critical system requires extensive knowledge in software development, embedded systems, and real-time communication. This kind of system would be highly regulated and subject to rigorous safety standards. Below is a simplified and high-level pseudocode representation of the logic for the described system:



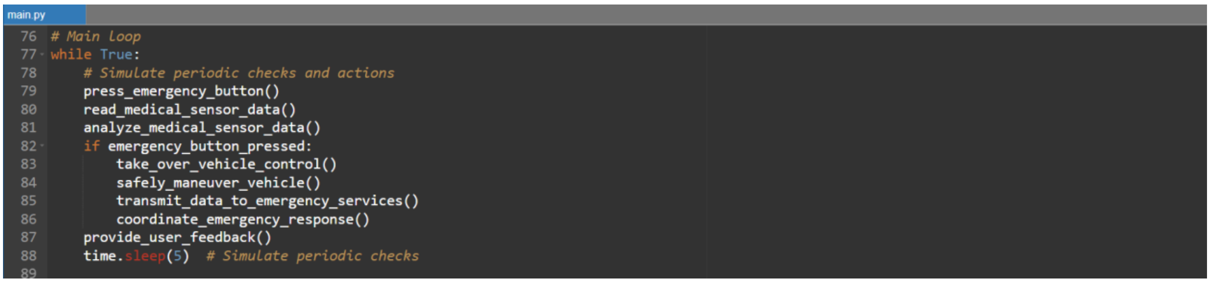
```python
# Import necessary libraries
import time
import random
from geopy.distance import geodesic

# Define global variables
emergency_button_pressed = False
driver_location = (0, 0)
medical_sensors_data = {}  # Placeholder for medical sensor data

# Define functions for key components

# 1. Emergency Button and Interface
def press_emergency_button():
    global emergency_button_pressed
    emergency_button_pressed = True

# 2. Communication Systems
def transmit_data_to_emergency_services():
    global driver_location
    # Simulate sending data to the nearest hospital and emergency services
    destination = (random.uniform(1, 10), random.uniform(1, 10))  # Random destination
    distance = geodesic(driver_location, destination).miles
    print("Transmitting data to the nearest hospital. Distance: {distance} miles")

# 3. Automatic Vehicle Control
def take_over_vehicle_control():
    # Simulate taking control of the vehicle
```



```python
def take_over_vehicle_control():
    # Simulate taking control of the vehicle
    print("Taking over vehicle control")

def safely_maneuver_vehicle():
    # Simulate safely maneuvering the vehicle to a safe location
    print("Safely maneuvering the vehicle to a safe location")

# 4. Emergency Services Coordination
def coordinate_emergency_response():
    # Simulate coordinating with emergency services
    print("Coordinating with emergency services")

# 5. Medical Sensors in Seatbelt
def read_medical_sensor_data():
    global medical_sensors_data
    # Simulate reading data from medical sensors
    medical_sensors_data = {
        "heart_rate": random.randint(60, 100),
        "blood_pressure": f"{random.randint(90, 140)}/{random.randint(60, 90)}",
        "ecg_data": random.uniform(0.1, 1.0),
    }

# 6. Data Processing and Analysis
def analyze_medical_sensor_data():
    global medical_sensors_data
    # Simulate analyzing medical sensor data
    # Add your analysis logic here
```



```python
    # Add your analysis logic here

# 7. User Feedback and Alerts
def provide_user_feedback():
    # Simulate providing feedback to the driver
    if emergency_button_pressed:
        print("Emergency assistance has been requested.")
    else:
        print("System is operational")

def find_nearest_hospital():
    # Logic to determine the nearest hospital based on the vehicle's GPS coordinates
    pass

def start_timer():
    # Start a timer for the predefined time limit
    pass

def is_emergency(vital_signs):
    # Analyze the vital signs to detect if there's an emergency
    pass

# Main Loop
while True:
    # Simulate periodic checks and actions
    press_emergency_button()
    read_medical_sensor_data()
    analyze_medical_sensor_data()
    if emergency_button_pressed:
```



```python
# Main Loop
while True:
    # Simulate periodic checks and actions
    press_emergency_button()
    read_medical_sensor_data()
    analyze_medical_sensor_data()
    if emergency_button_pressed:
        take_over_vehicle_control()
        safely_maneuver_vehicle()
        transmit_data_to_emergency_services()
        coordinate_emergency_response()
    provide_user_feedback()
    time.sleep(5)  # Simulate periodic checks
```

This pseudocode provides a simplified overview of the system's logic and the key components involved. In a real-world implementation, we would need to use specific programming languages and libraries for hardware interactions, real-time processing, and safety-critical systems. Additionally, extensive testing, validation, and adherence to safety standards would be crucial for a functional and safe implementation of this idea.