Source code:

# Import necessary libraries and modules

# 1. Emergency Button and Interface

class UserInterface:

def \_\_init\_\_(self):

self.button\_pressed = False

def press\_button(self):

self.button\_pressed = True

# 2. Communication Systems

class CommunicationSystem:

def \_\_init\_\_(self):

self.location = None

self emergency\_message = None

def transmit\_location(self, location):

self.location = location

def transmit\_emergency\_message(self, message):

self.emergency\_message = message

# 3. Automatic Vehicle Control

class VehicleAutomation:

def \_\_init\_\_(self):

self.control\_enabled = False

def enable\_control(self):

self.control\_enabled = True

def disable\_control(self):

self.control\_enabled = False

def maneuver\_to\_safe\_location(self, location):

# Implement the vehicle maneuvering logic here

pass

# 4. Emergency Services Coordination

class EmergencyServicesCoordinator:

def \_\_init\_\_(self):

self.emergency\_status = "No emergency"

def update\_emergency\_status(self, status):

self.emergency\_status = status

def send\_data\_to\_emergency\_services(self, data):

# Implement data transmission to emergency services here

pass

# 5. Medical Sensors in Seatbelt

class MedicalSensor:

def \_\_init\_\_(self):

self.vital\_signs = None

def monitor\_vital\_signs(self, data):

self.vital\_signs = data

# 6. Data Processing and Analysis

class DataProcessor:

def analyze\_data(self, data):

# Implement data analysis and decision-making logic here

pass

# 7. User Authentication and Security (Not implemented in this basic prototype)

# 8. Power Supply and Redundancy (Not implemented in this basic prototype)

# 9. User Feedback and Alerts (Not implemented in this basic prototype)

# 10. Testing and Safety Regulations (Not implemented in this basic prototype)

# 11. App and Backend Support (Optional) (Not implemented in this basic prototype)

# 12. User Education and Training (Not implemented in this basic prototype)

# Simulate a driver pressing the emergency button

if \_\_name\_\_ == "\_\_main\_\_":

ui = UserInterface()

communication\_system = CommunicationSystem()

vehicle\_automation = VehicleAutomation()

emergency\_coordinator = EmergencyServicesCoordinator()

medical\_sensor = MedicalSensor()

data\_processor = DataProcessor()

ui.press\_button()

communication\_system.transmit\_location("123.456, 789.012")

communication\_system.transmit\_emergency\_message("Driver needs assistance")

# If an emergency is detected, enable vehicle control

if data\_processor.analyze\_data(medical\_sensor.vital\_signs):

vehicle\_automation.enable\_control()

vehicle\_automation.maneuver\_to\_safe\_location(communication\_system.location)

# Update emergency status and send data to emergency services

emergency\_coordinator.update\_emergency\_status("Emergency detected")

emergency\_coordinator.send\_data\_to emergency\_services(

communication\_system.emergency\_message

)