Phase 3

```
# sensors/sensor_simulation.py
import random

def get_vehicle_count():
    return {
        "lane_1": random.randint(0, 10),
        "lane_2": random.randint(0, 10),
        "lane_3": random.randint(0, 10),
        "lane_4": random.randint(0, 10)
    }

# vision/vehicle_counter.py
# Simulated counter – replace with OpenCV logic if using real camera

Def detect_vehicles():
    From sensors.sensor_simulation import get_vehicle_count
    Return get_vehicle_count()


# ai_model/q_learning.py
Import numpy as np
Import random

Class TrafficQAgent:
```

```
  Def __init__(self, state_size, action_size, learning_rate=0.7, discount=0.95, epsilon=1.0, epsilon_decay=0.995):
      Self.q_table = np.zeros((state_size, action_size))
      Self.lr = learning_rate
      Self.gamma = discount
      Self.epsilon = epsilon
      Self.epsilon_decay = epsilon_decay
      Self.action_size = action_size

  Def choose_action(self, state):
      If random.uniform(0, 1) < self.epsilon:
          Return random.randint(0, self.action_size – 1)
      Return np.argmax(self.q_table[state])

  Def learn(self, state, action, reward, next_state):
      Predict = self.q_table[state][action]
      Target = reward + self.gamma * np.max(self.q_table[next_state])
      Self.q_table[state][action] += self.lr * (target – predict)
      Self.epsilon *= self.epsilon_decay

# server/controller.py
From ai_model.q_learning import TrafficQAgent
From vision.vehicle_counter import detect_vehicles

Agent = TrafficQAgent(state_size=16, action_size=4)  # 4 lanes, 4 actions
```

```
Def get_state():
    Data = detect_vehicles()
    Return sum(data.values()) % 16  # Simplified state encoding

Def optimize_signal():
    State = get_state()
    Action = agent.choose_action(state)
    Reward = 10 – state  # Fewer vehicles = higher reward
    Next_state = get_state()
    Agent.learn(state, action, reward, next_state)
    Return action

# server/app.py
From flask import Flask, jsonify
From controller import optimize_signal

App = Flask(__name__)

@app.route('/optimize', methods=['GET'])
Def optimize():
    Signal = optimize_signal()
    Return jsonify({"green_signal_for_lane": signal})

If __name__ == '__main__':
    App.run(debug=True)
```