System Design Requirements

Accuracy Requirements

1. **Requirement**: When a traffic sensor reports vehicle counts, the **Traffic Control System (TCS)** shall ensure that the vehicle count data is accurate within an acceptable error range.
    - **Potential Design Solutions**:
        - Implement sensor fusion techniques (e.g., combining loop detectors, cameras, and radar).
        - Use machine learning-based error correction for noisy sensor readings.
        - Calibrate sensors periodically to minimize drift in measurements.
2. **Requirement**: When a traffic signal phase decision is made, the **TCS** shall select the optimal phase based on real-time traffic conditions.
    - **Potential Design Solutions**:
        - Train the RL agent using a well-validated traffic simulator (e.g., SUMO).
        - Use deep reinforcement learning with extensive historical traffic data.
        - Validate decisions against rule-based baselines to ensure improvements.

Robustness Requirements

3. **Requirement**: When a sensor fails or provides inconsistent data, the **TCS** shall maintain traffic control functionality by estimating missing data.
    - **Potential Design Solutions**:
        - Implement Kalman filters or Bayesian networks to infer missing values.
        - Use redundancy (multiple sensors per intersection) to cross-verify data.
        - Apply anomaly detection to flag and correct erroneous sensor readings.
4. **Requirement**: When unexpected traffic conditions occur (e.g., accidents, road closures), the **TCS** shall adapt its signal timings dynamically.
    - **Potential Design Solutions**:
        - Integrate real-time traffic incident reports from external sources (e.g., GPS data, city traffic management).
        - Train the RL agent on diverse scenarios, including rare events, using domain randomization.
        - Use multi-agent RL to enable coordination between adjacent intersections.

Quality of Data Requirements

5. **Requirement**: When collecting traffic flow data, the **TCS** shall filter out noise and erroneous readings before making decisions.
    - **Potential Design Solutions**:
        - Use statistical smoothing techniques (e.g., moving averages, outlier detection).
        - Implement data preprocessing pipelines before feeding into RL models.
        - Employ real-time anomaly detection algorithms to identify faulty sensor readings.
6. **Requirement**: When training the RL model, the **TCS** shall use a diverse dataset that reflects real-world variability in traffic patterns.
    - **Potential Design Solutions**:
        - Collect data across different times of the day, seasons, and weather conditions.

Verifiability Requirements

7. **Requirement**: When an RL model makes a decision, the **TCS** shall log the state, action, and reward for future auditing and debugging.
    - **Potential Design Solutions**:
        - Implement an RL decision logging system with timestamps and traffic conditions.
        - Use interpretable RL techniques (e.g., SHAP values) to explain model decisions.
        - Store logs in a secure, version-controlled database for post-analysis.
8. **Requirement**: When evaluating system performance, the **TCS** shall compare real-world outcomes with expected results to verify correctness.
    - **Potential Design Solutions**:
        - Define key performance metrics (e.g., average waiting time, queue lengths).
        - Use A/B testing by running RL-based control alongside traditional rule-based systems.
        - Regularly validate decisions using real-world traffic video footage.