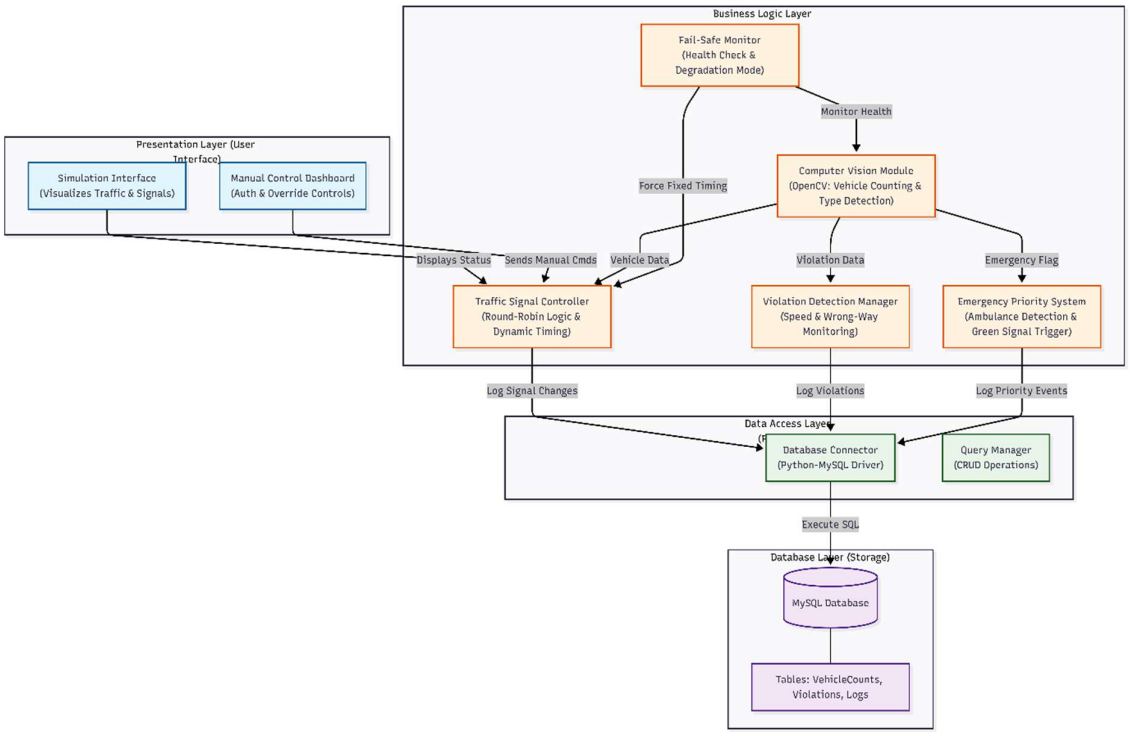
# Software Architecture Design

*Proposed Software Architecture Style*

**Architecture Style: Layered Architecture**

## Glimpse of Layered Architecture Of Traffic Signal Automation:




## Justification of Categorization (Granularity of Components):

The "Smart Traffic Signal Automation System" falls strictly into the **Layered Architecture** category because the software components are organized into horizontal logical layers, where each layer has a specific responsibility and only communicates with the layers directly above or below it.

## The granularity of components is defined as follows:

- **Presentation Layer (UI Granularity):** Contains only components responsible for user interaction and visualization (Simulation & Dashboard). It contains no processing logic.
- **Business Logic Layer (Processing Granularity):** Contains the "brain" of the system (Computer Vision, Signal Control, Safety Checks). It processes raw data but does not know how to store it permanently.
- **Data Access Layer (Persistence Granularity):** Abstracts the database technology. It translates high-level requests (e.g., "Log Violation") into low-level SQL queries.
- **Database Layer (Storage Granularity):** Contains the physical raw data and tables, isolated from the application code.

## Why Layered Architecture is the Best Choice:

Considering the requirements of the Smart Traffic System, this style is superior for the following reasons:

- **Performance (Low Latency):** The system requires signal decisions within **2 seconds**. Layered architecture allows the **Computer Vision Module** to pass data directly to the **Traffic Signal Controller** (in the same layer) via function calls. This is significantly faster than the network overhead required by Microservices or SOA.

- **Maintainability (Separation of Concerns):** If the **MySQL** database schema changes, we only need to update the **Data Access Layer**. The **Computer Vision** and **Presentation** layers remain untouched, reducing the risk of breaking the system during updates.

- **Scalability (Modular Growth):** We can easily upgrade the **Computer Vision Module** (e.g., adding pedestrian detection) without rewriting the User Interface or the Database, as long as the inputs/outputs remain consistent.

- **Reliability (Fail-Safe Implementation):** The **Fail-Safe Monitor** sits within the Logic Layer, allowing it to instantly override the **Traffic Signal Controller** if the Vision module fails. This tight integration ensures the required **3-second fail-safe activation** is met reliably.

## Application Components:

**1. Presentation Layer Components**

- **Simulation Interface:** Visualizes the 4-way intersection, real-time vehicle movement, and signal status (Red/Yellow/Green) for testing and demonstration.
- **Manual Control Dashboard:** Provides a secure login for traffic authorities to override automatic signals or trigger emergency stops.

**2. Business Logic Layer Components**

- **Computer Vision Module:** Uses OpenCV to capture images, detect vehicles, classify types (car/bus), and count traffic density.
- **Traffic Signal Controller:** Implements the core logic for the Round-Robin sequence and calculates Dynamic Timing (15-60s) based on density.
- **Fail-Safe Monitor:** Continuously checks system health; triggers "Fixed-Time Mode" if camera or processing errors are detected.
- **Violation Detection Manager:** Tracks vehicle IDs to detect and flag "Wrong-Way" driving and "Overspeeding" events.
- **Emergency Priority System:** Identifies emergency vehicles (ambulances/fire trucks) and triggers immediate Green signals.

**3. Data Access Layer Components**

- **Database Connector:** Manages the secure python-to-SQL connection using the MySQL driver.
- **Query Manager:** Contains specific CRUD (Create, Read, Update, Delete) functions to log counts, violations, and system events without exposing SQL code to the logic layer.

**4. Database Layer Components**

- **MySQL Database:** The relational storage system hosting tables for VehicleCounts, ViolationRecords, SystemLogs, and AuthUsers.