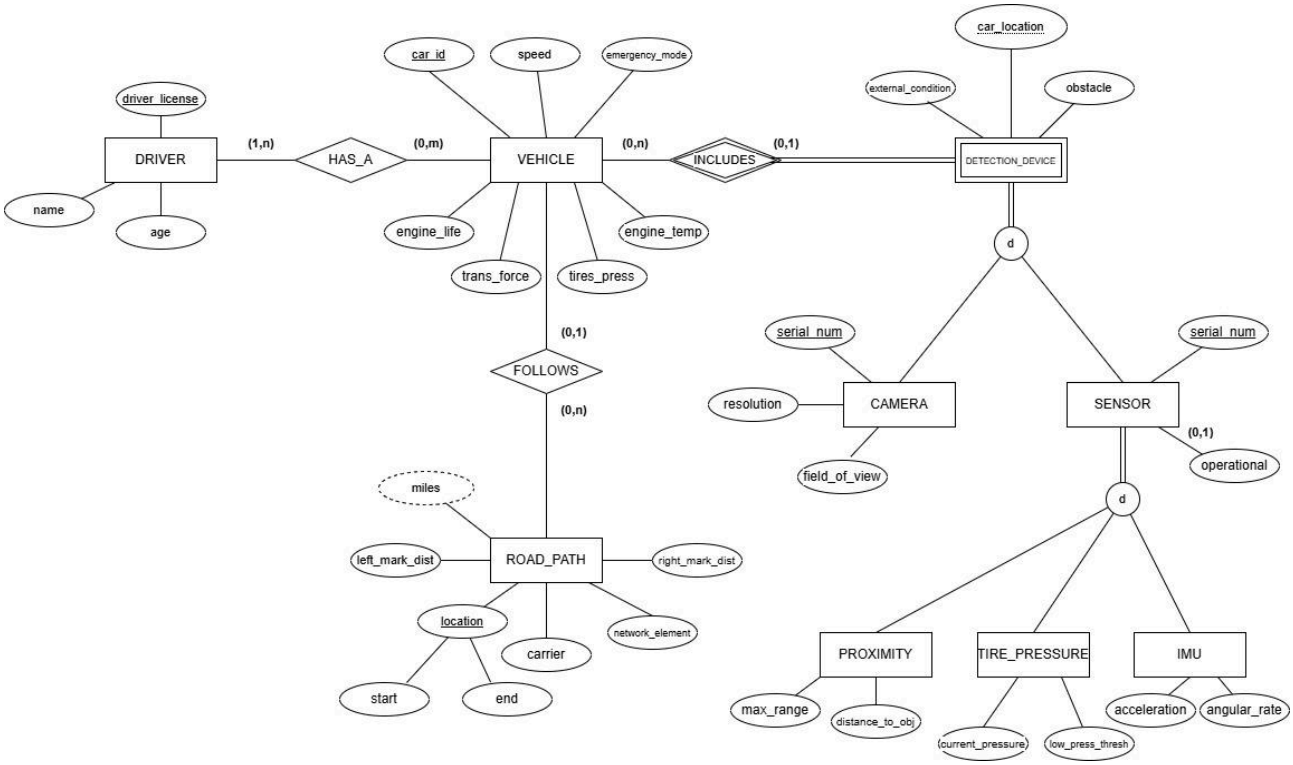
Autonomous Vehicle System

Anwar, John, Daniel, Gabe, Rodolfo, Joseph

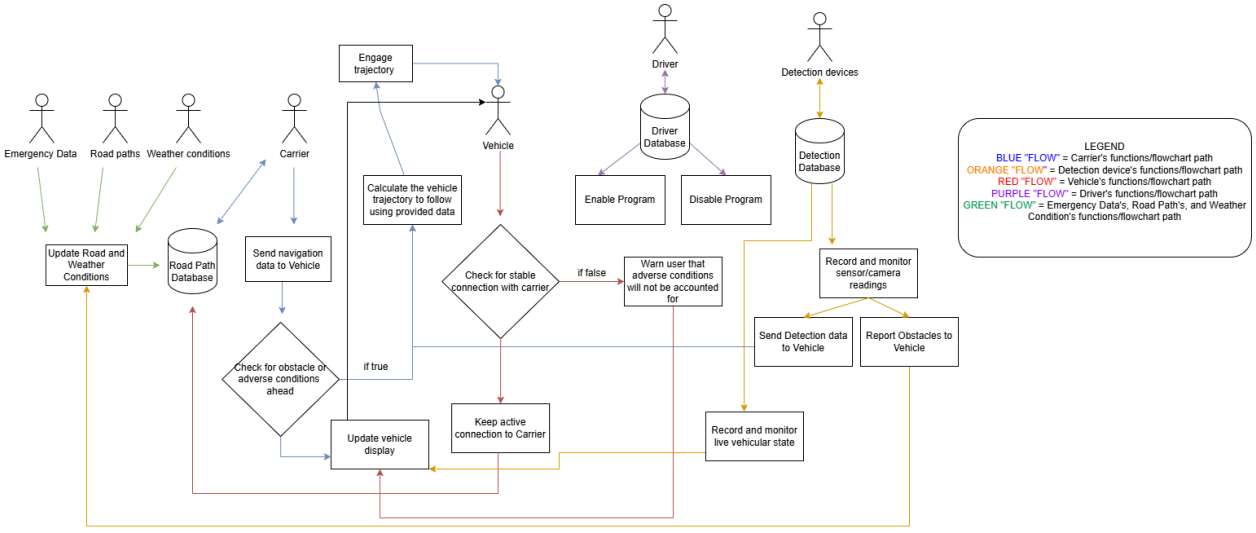
CS 250: Introduction to Software Systems

Professor Umut Can Cabuk

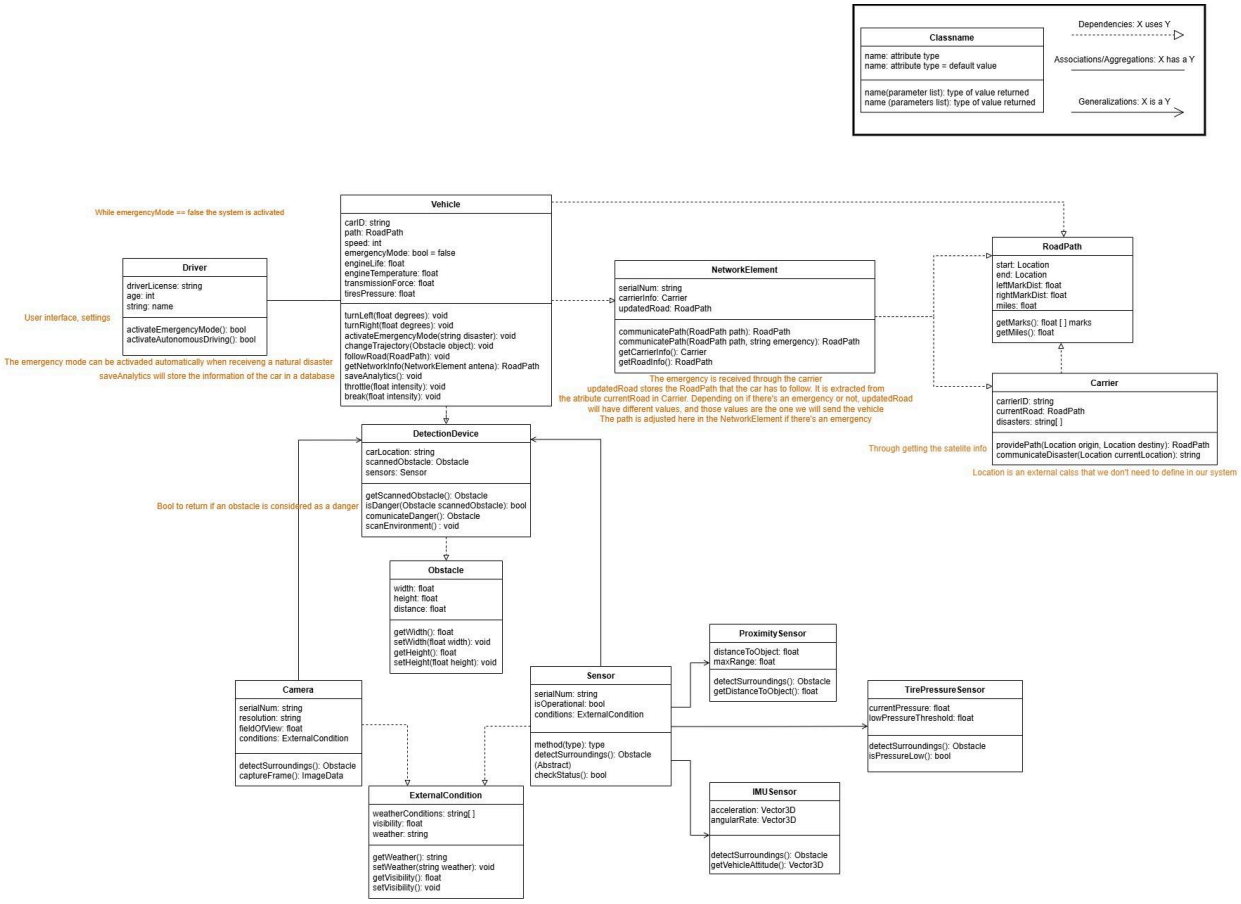
Database Diagram:



Updated SWA Diagram:



Updated UML Diagram:



Design Specification with Modifications:

Following our past versions of the model, we've made edits to reflect how we will store and manage all our data in order to have a functional and efficient program. We've rendered this new diagram that reflects our databases after revising our past design. We decided that our "Navigation Database" was too abstract and vague, so we elected to separate it into "DRIVER", "DETECTION_DEVICE" and "ROAD_PATH". These communicate with each other using an SQL approach in the relational database and store different information that is nicely organized. For example, the "VEHICLE" database follows the "ROAD_PATH" database and is able to retrieve information from it without the queries overloading "ROAD_PATH". Also, "VEHICLE" includes "DETECTION_DEVICE", meaning it uses the information from the database to operate since our program wouldn't be able to navigate without it. This is a conscious decision, as we've decided that dividing the database into separate ones allows for more abstraction which is easier to understand and follow. Additionally, these changes are reflected in our UML diagram and SWA diagram. These additions and alterations to our previous designs get us one step closer to implementing a program that will operate seamlessly and fulfill all the benchmarks from our design specification.

Data Management Strategy:
The Autonomous Vehicle System relies on a single relational database (SQL) to manage all critical data. This centralized approach ensures data consistency and integrity across the tightly coupled entities, which is essential for safe operation.

The database is structured around three primary tables/conceptual entities:

Entry	**Purpose**	**Key Data stored**	**Usage and Updated Frequency**
ROAD_PATH	Stores foundational, geo-spatial, and regulatory data for navigation.	Route segments, GPS coordinates, speed limits, lane geometry, intersection rules, pre-defined maps.	Primarily Read operations by the Vehicle Navigation Logic; updated infrequently by the mapping service.
DETECTION_DEVICE	Stores high-frequency, real-time data from all vehicle sensors.	LiDAR readings, camera object detection (type and position), radar velocity measurements, timestamped event logs.	Primarily Write/Update operations by the sensor fusion component (high volume); Read operations for immediate obstacle avoidance.
DRIVER	Stores persistent data related to the driver, vehicle configuration, and historical performance.	User preferences, emergency contact details, vehicle configuration settings, historical trip logs, maintenance records, authentication tokens.	Balanced Read/Write operations; used during startup, trip planning, and shutdown.

Data Flow Strategy:

1. Guidance: The VEHICLE system initiates a query to the ROAD_PATH table to retrieve the necessary path segments and regulatory information.

2. Perception: Simultaneously, the DETECTION_DEVICE entity is continuously updated with sensor data. The Vehicle Perception Component executes highly localized queries against this real-time data to determine immediate threats and safe movement vectors.

3. Logging and State: All system actions, errors, and trip metadata are persisted to the DRIVER table's log fields, ensuring an immutable record for post-trip analysis or diagnostics.

This relational structure allows for clear foreign key relationships (e.g., the sensor data in DETECTION_DEVICE references the specific vehicle/driver context), enforcing the required strong consistency crucial for safety-critical operations.

Tradeoffs And Discussion For Design Choices: The chosen data management strategy utilizes a single SQL database due to the system's core requirements for strong transactional consistency and handling of complex, tightly coupled relationships between entities including the “VEHICLE”, “DRIVER”, and various “DETECTION_DEVICE” components. SQL is superior to NoSQL alternatives in this context because the integrity of safety-critical vehicle data is more crucial than massive, distributed scalability. The decision to use a single overarching relational database over multiple, distributed databases simplifies operations, guarantees data integrity across related tables within a single transaction, and avoids the complexity and potential synchronization issues associated with distributed data and microservices, which are unnecessary for this system's defined scope.