**1️⃣ UserService (Authentication & User Management) 👤**

🔹 **Handles**: User registration, authentication (JWT), role-based access control (RBAC).  
🔹 **Tech Stack**:

* **.NET 9 Web API**
* **Identity Framework**
* **MS SQL Server** (for structured user data)
* **MongoDB** (for user activity logs or session data)

🔹 **Endpoints**:

* **POST /api/users/register** → Register a new user
* **POST /api/users/login** → Authenticate user & generate JWT
* **GET /api/users/{id}** → Get user details

📌 **Use Case**: A traffic control officer logs into the system to monitor and control intersections.

**2️⃣ TrafficService (Traffic Light Control & AI Processing) 🚦**

🔹 **Handles**:

* Smart traffic light control based on real-time data
* AI-based optimization of signal timings
* Emergency vehicle prioritization

🔹 **Tech Stack**:

* **.NET 9 Web API**
* **ML.NET** (for AI-based traffic prediction)
* **RabbitMQ** (for real-time alerts)
* **MongoDB** (to store intersection and signal history for AI processing)

🔹 **Endpoints**:

* **GET /api/traffic/status/{intersectionId}** → Get current light status
* **POST /api/traffic/update** → Change traffic light timing
* **POST /api/traffic/emergency** → Prioritize emergency vehicle passage

📌 **Use Case**: AI detects high congestion at an intersection and dynamically adjusts the traffic lights.

**3️⃣ VehicleService (Vehicle Tracking & Traffic Analytics) 🚗**

🔹 **Handles**:

* GPS-based vehicle tracking
* Logs traffic flow data
* Provides analytics on vehicle movements

🔹 **Tech Stack**:

* **.NET 9 Web API**
* **MS SQL Server** (for structured vehicle data)
* **Redis** (for caching frequently accessed vehicle data)
* **MongoDB** (for real-time vehicle tracking logs and unstructured vehicle-related data)

🔹 **Endpoints**:

* **POST /api/vehicles/register** → Register a new vehicle
* **GET /api/vehicles/{licensePlate}** → Get vehicle details
* **GET /api/vehicles/realtime/{location}** → Get real-time vehicle count at a location

📌 **Use Case**: The system detects an unusual number of vehicles in an area, indicating a traffic jam.

**4️⃣ NotificationService (Alerts & Messaging) 🔔**

🔹 **Handles**:

* Sends alerts when accidents or high congestion occur
* Notifies emergency services about blocked roads
* Uses RabbitMQ for event-driven notifications

🔹 **Tech Stack**:

* **.NET 9 Worker Service**
* **RabbitMQ** (for message queuing)
* **MongoDB** (for storing historical notifications and events)

🔹 **Functionality**:

* Listens for **TrafficJamDetected** event from **TrafficService**
* Sends notifications via email/SMS

📌 **Use Case**: If an accident is detected, a notification is sent to the traffic control center and emergency services.

**5️⃣ GatewayAPI (Ocelot API Gateway) 🚪**

🔹 **Handles**:

* Acts as a single entry point for all microservices
* API routing, authentication, and load balancing

🔹 **Tech Stack**:

* **Ocelot API Gateway**

🔹 **Functionality**:

* Routes /users/\* requests to **UserService**
* Routes /traffic/\* requests to **TrafficService**
* Routes /vehicles/\* requests to **VehicleService**

📌 **Use Case**: A mobile app calls /traffic/status/123 and gets routed to **TrafficService** transparently.

**6️⃣ Infrastructure Services**

A. **SQL Server (Persistent Storage)** 🔹 **Handles**:

* Stores user data, vehicle data, and traffic logs (for structured relational data)

B. **RabbitMQ (Message Broker)** 🔹 **Handles**:

* Asynchronous event-driven communication between services
* Example: **TrafficService** detects congestion → sends event to **NotificationService**

C. **Redis (Caching)** 🔹 **Handles**:

* Caches frequently accessed traffic and vehicle data for fast retrieval
* Reduces SQL Server load

D. **MongoDB (Document-Based NoSQL Database)** 🔹 **Handles**:

* Stores unstructured or semi-structured data like traffic light signal history, user activity logs, real-time vehicle data, etc.
* Example: **TrafficService** stores traffic light patterns and AI-generated data in MongoDB.

**🌐 Microservices Communication Flow**

1️⃣ **VehicleService** detects high congestion and sends an event to **RabbitMQ**.  
2️⃣ **TrafficService** listens to the event and adjusts traffic lights.  
3️⃣ **NotificationService** picks up the event and sends an alert.  
4️⃣ The **API Gateway** ensures smooth communication across all services.

**Updated Database Use Case with MongoDB**

* **MongoDB** enhances flexibility for handling dynamic, unstructured data (vehicle tracking, traffic light patterns, real-time logs).
* **MS SQL Server** remains for relational data (e.g., user data, vehicle registrations).
* **Redis** ensures that high-frequency queries, such as current vehicle counts or traffic light status, are served quickly.