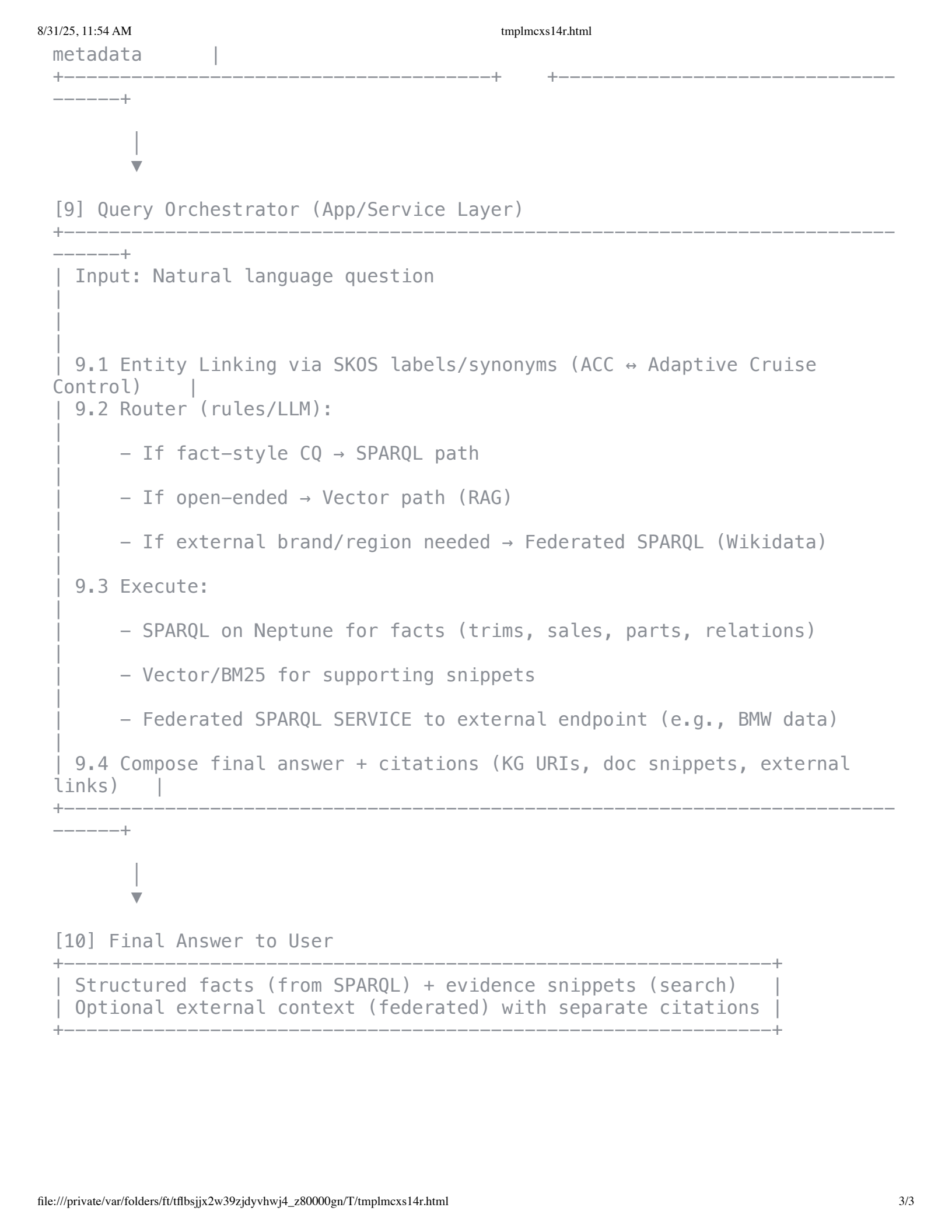
```
[0] Competency Questions (CQs)
    └─ Define must-answer questions (e.g., trims with ACC & sales by
region/date)

       |
       ▼

[1] Sources  (your 8 inputs)
+-------------------------------------------------------------+
| 1.1 Sales DB    | 1.2 R&D PDFs | 1.3 Scanned PDFs (OCR)     |
| 1.4 CAD/Images  | 1.5 XML/JSON | 1.6 Reviews (Twitter/web)  |
| 1.7 Model/Parts | 1.8 Patents  |                            |
+-------------------------------------------------------------+
Notes: raw landing → S3/DBFS "Bronze"
Tools: Airbyte/NiFi/Glue jobs; Textract/Tesseract; Apache Tika; web
scrapers

       |
       ▼

[2] Ingest & Clean ("Silver")
+-------------------------------------------------------------+
| 2.1 Schema unification & parsing (CSV/JSON/XML)             |
| 2.2 PDF text extraction + OCR text                          |
| 2.3 Basic QA: nulls, types, date formats                    |
| 2.4 Entity cues: model codes, trim names, part numbers      |
+-------------------------------------------------------------+
Tools: Spark/Databricks/Glue; Great Expectations for data tests

       |
       ▼

[3] Conform ("Gold" analytics layer)
+-------------------------------------------------------------+
| 3.1 Conformed tables:                                       |
|     - dim_vehicle_model, dim_trim, dim_feature              |
|     - dim_part, bridge_trim_feature, bridge_part_fitment    |
|     - fact_sales, dim_market_region                         |
|     - dim_document/img/patent + mention bridges             |
| 3.2 De-dup + entity resolution (match models/trims/parts)   |
+-------------------------------------------------------------+
Tools: Spark/SQL; dbt for models & tests

       |
       ▼

[4] Ontology & Vocabulary
+-------------------------------------------------------------+
| 4.1 Ontology classes/properties (RDF/RDFS/OWL)              |
|     VehicleModel, Trim, Feature, Part, SaleRecord, Region ... |
| 4.2 SKOS concepts for labels/synonyms (e.g., ACC ↔ Adaptive…) |
```

```
| 4.3 Map CQs → required classes/relations                          |
+-------------------------------------------------------------------+
Tools: Protégé (authoring), OWL/RDFS; SKOS for terms

        |
        ▼

[5] RDF Mapping & Triple Generation
+-------------------------------------------------------------------+
| 5.1 R2RML/RML mapping Gold tables → RDF triples                   |
| 5.2 Mint URIs (ex:/model/M1, ex:/trim/T101, …)                    |
| 5.3 Serialize as N-Triples/Turtle                                 |
+-------------------------------------------------------------------+
Output: RDF files (triples)

        |
        ▼

[6] Validation & Reasoning
+-------------------------------------------------------------------+
| 6.1 SHACL shapes (quality gates):                                 |
|     - Trim has exactly one belongsTo; has modelYear (gYear)       |
|     - SaleRecord has forTrim, soldIn, saleDate, quantity:int      |
| 6.2 Optional OWL reasoning/consistency check                      |
+-------------------------------------------------------------------+
Tools: SHACL engine (Jena/TopBraid/pySHACL); HermiT/ELK reasoner

        |
        ▼

[7] Load Graph DB (Neptune or Neo4j)
+-------------------------------------------------------------------+
| 7.1 Bulk load RDF into Graph DB                                   |
| 7.2 Set indexes as needed                                         |
| 7.3 Expose SPARQL endpoint                                        |
+-------------------------------------------------------------------+
Graph DB: Amazon Neptune (SPARQL)  (or Neo4j + n10s for RDF)

        |
        |──────────────────┐
        |                  |
        ▼                  ▼

[8A] Text/Search Index (for snippets)      [8B] Vector Index (RAG)
+-----------------------------------+    +-------------------------------
------+
| 8A.1 Ingest doc text (R&D/OCR/manual)|    | 8B.1 Embed chunks
(OpenAI/Instructor) |
| 8A.2 BM25/keyword index (OpenSearch) |    | 8B.2 kNN index
(OpenSearch/Qdrant)    |
| 8A.3 Store docId ↔ entity links      |    | 8B.3 Store chunk→entity
```


metadata
[9] Query Orchestrator (App/Service Layer)
Input: Natural language question
9.1 Entity Linking via SKOS labels/synonyms (ACC ↔ Adaptive Cruise Control)
9.2 Router (rules/LLM):
- If fact-style CQ → SPARQL path
- If open-ended → Vector path (RAG)
- If external brand/region needed → Federated SPARQL (Wikidata)
9.3 Execute:
- SPARQL on Neptune for facts (trims, sales, parts, relations)
- Vector/BM25 for supporting snippets
- Federated SPARQL SERVICE to external endpoint (e.g., BMW data)
9.4 Compose final answer + citations (KG URIs, doc snippets, external links)
[10] Final Answer to User
Structured facts (from SPARQL) + evidence snippets (search)
Optional external context (federated) with separate citations