FSD_03 – Tag & Reference Linking Engine

🧠 Purpose:

This engine takes enriched, classified content segments from **FSD_02** and connects them to:

- Audira's Discovery Tags
- Existing onboarding answers (from 10 fixed + 10 dynamic Qs)
- Each other (i.e. internal document references)
- Gaps in coverage → to power follow-up question generation (FSD_04)

📄 Section Plan for FSD_03:

Section	Description
1. Scope	What linking this engine is responsible for
2. Input Format	Data structures from FSD_02 and onboarding system
3. Matching Engine	Tag assignment, scoring, and reference graph building
4. Discovery Gaps	Detection of missing tag coverage (gap analysis)
5. Output Format	Linked segment structure ready for prompt chain or validator
6. Confidence Routes	Handling low match certainty
7. Cross-Segment Anchors	How references across documents are traced
8. Future Enhancements	Roadmap for deeper logic

Section 1: Scope

🔍 Objective:

This module builds **semantic bridges** between all known knowledge fragments within Audira — aligning discovery tags, document insights, and onboarding answers into one unified memory graph. It is the **central "understanding core"** of the Audira AI brain.

✅ Responsibilities:

1. **Tag Assignment from Segments**
   - Matches enriched segments (from `FSD_02`) to Audira's known **Discovery Tags**
   - Uses embeddings, keyword cues, and metadata logic to suggest or confirm intent
2. **Cross-Linking Between Inputs**

    - Identifies when multiple document blocks refer to the same business idea (e.g. "revenue model" appearing in a table and in text)
3. **Answer-to-Tag Linking**
    - Links onboarding answers from 10 Fixed + 10 Dynamic Questions to the same discovery tag space
    - Enables comparison: *"Was this tag already answered during onboarding?"*
4. **Tag Gap Detection**
    - Builds a "Tag Coverage Map" to identify **which tags exist**, **which are confidently answered**, and **which remain uncovered**
    - This fuels follow-up question generation (via `FSD_04`)
5. **Reference Anchoring**
    - Generates `tag_reference_map`: showing where each tag appears in files, answers, or both

---

❌ Out of Scope:

This module **does not**:

- Generate follow-up questions (FSD_04 does that using the output of this module)
- Decide if the agent is launch-ready (that's handled in FSD_05: Readiness Validator)
- Finalize prompt phrasing (but it powers the prompt engine via tag-linkage)

📎 Internal References:

- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – master source of tag rules
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – question-to-tag mapping
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – prompt path routing from tags
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – defines tag coverage criteria

✅ Here is **FSD_03 – Section 2: Input Format**, carefully structured for downstream tag-linking logic:

Section 2: Input Format

🎯 Purpose:

To unify all business-relevant knowledge sources into a single tag-linkable input stream:

1. Enriched segments from `FSD_02`
2. Onboarding answers (from 10 fixed + 10 dynamic questions)
3. Pre-tagged metadata (from Upload Schema)
4. Discovery Tags Dictionary (master tag schema)

📦 Required Inputs:

📦 A. Enriched Segments (from FSD_02)

Each content segment is expected to follow this format:

```
{
  "segment_id": "seg_01",
  "text": "Net revenue per active user reached $42 in Q1",
  "semantic_role": "kpi_snapshot",
  "embedding": [...],
  "suggested_tags": ["revenue_model"],
  "confidence": 0.91,
  "source_file": "audira_plan_2025.pdf"
}
```

📦 B. Onboarding Answers

From the Fixed and Dynamic Questions collected via the *AGENT ONBOARDING FRAMEWORK*:

```
{
  "answer_id": "ans_003",
  "question_id": "q_10",
  "text": "We charge vendors a 5% commission per transaction.",
  "linked_tags": ["vendor_model", "payout_policy"],
  "source": "onboarding"
}
```

📦 C. Discovery Tags Dictionary

Source-of-truth from *AUDIRA DISCOVERY TAGS DICTIONARY.docx*:

```
{
  "tag_id": "revenue_model",
  "description": "Defines how money is earned (subscriptions, commission,
etc.)",
  "keywords": ["pricing", "earnings", "transaction fee", "per user",
"revenue"],
  "category": "business_model",
  "priority": "core"
}
```

🧩 Enrichment Expected:

- All entries must include embedding vectors (for semantic matching)
- Text-only items (e.g. answers) will be embedded on-the-fly
- Tags are matched using both vector similarity and keyword matching

---

📎 References:

- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* — structure of uploaded docs
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* — onboarding Q/A format
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* — tag keyword + category rules

---

Section 3: Matching Engine

🎯 Purpose:

To intelligently assign Audira Discovery Tags to both:

- **Document segments** (from FSD_02)
- **Onboarding answers** (from 10 fixed + 10 dynamic questions)

This forms the shared **semantic memory graph** used by the agent to understand, reason, and respond.

🧠 Matching Logic Flow:

1. Text Preprocessing

- Normalize casing, punctuation, currency, common formatting
- Tokenize into subphrases and filter stop words
- Handle OCR noise if detected (from scanned PDFs)

2. Vector-Based Similarity Matching

- Compare each segment/answer embedding to every tag keyword set embedding
- Compute cosine similarity scores

```
{
  "tag": "revenue_model",
  "cosine_score": 0.88
}
```

3. Keyword Overlap Boost

- Matches increase in weight if raw keyword matches are found in text
- Includes keyword synonyms from *DISCOVERY TAGS DICTIONARY*

4. Contextual Hint Matching

- Use metadata: segment role (`kpi_snapshot`, `payout_clause`, etc.)
- If the segment is a `table_row` under a section titled "Commission Plan," boost `vendor_model` and `revenue_model`

5. Final Confidence Scoring

Combined score =
`0.6 * embedding_score + 0.3 * keyword_match + 0.1 * context_boost`

📦 Matching Output Example:

```
{
  "segment_id": "seg_01",
  "matched_tag": "revenue_model",
  "confidence": 0.87,
  "match_method": "embedding+keyword",
  "source": "file_block"
}
```

⚠️ Multiple Tags per Input:

Each segment or answer can link to multiple tags (ranked by score)

📎 References:

- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – keywords + embeddings
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – for source metadata
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – for downstream routing

---

Section 4: Discovery Gaps

🎯 Purpose:

To identify **missing or weakly covered discovery tags** based on:

- Onboarding answers (Fixed + Dynamic Qs)
- Auto-tagged document segments (from FSD_02)
- Cross-match against the full *DISCOVERY TAGS DICTIONARY*

This ensures that **no critical business area is missed**, enabling generation of AI follow-up questions in FSD_04.

🧩 Gap Detection Workflow:

1. Build Tag Coverage Index

For each tag in the dictionary, track:

```
json
CopyEdit
{
  "tag_id": "payout_policy",
  "covered_by_answers": true,
  "covered_by_file_blocks": false,
  "best_confidence": 0.72,
  "sources": ["onboarding_q7"],
  "confidence_class": "moderate"
```

}

2. Apply Thresholds for Gap Flagging

- `Fully Covered` → Confidence > 0.85 from at least one trusted source
- `Partially Covered` → 0.65–0.85 from one or multiple sources
- `Uncovered` → No reliable source (or below 0.65)

3. Track Discovery Source Types

Tags are flagged as:

- `found_in_answers_only`
- `found_in_files_only`
- `redundant_cross-confirmed`
- `not_found` → triggers FSD_04 dynamic Q generation

📦 Output Example:

```
json
CopyEdit
{
  "tag_id": "vendor_risk",
  "status": "uncovered",
  "sources": [],
  "suggested_action": "trigger_question",
  "priority": "high"
}
```

📊 Coverage Snapshot Per Business:

This module generates a **coverage map** per consumer, showing:

- % of tags covered
- % covered only from files
- % needing clarification
- Tags with conflicting signals (e.g. answers vs. document)

📎 References:

- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – defines full tag set
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – provides baseline answer sources

- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – uses tag coverage % for launch qualification

---

Section 5: Output Format

📦 Unified Output Structure

This module outputs two key artifacts:

A. Tagged Knowledge Objects

For each file segment or onboarding answer:

```
json
CopyEdit
{
  "source_type": "file_segment" | "onboarding_answer",
  "source_id": "seg_01",
  "text": "Vendors are paid every 14 days via ACH.",
  "matched_tags": [
    {
      "tag_id": "payout_policy",
      "confidence": 0.91,
      "match_method": "embedding+keyword",
      "source": "text + metadata"
    }
  ]
}
```

B. Tag Coverage Map

For the entire consumer profile:

```
json
CopyEdit
{
  "consumer_id": "csm_4021",
  "tag_coverage": [
    {
      "tag_id": "revenue_model",
      "status": "fully_covered",
      "best_confidence": 0.93,
      "sources": ["seg_01", "onboarding_q10"]
    },
    {
```

```
      "tag_id": "vendor_risk",
      "status": "uncovered",
      "best_confidence": null,
      "sources": [],
      "suggested_action": "trigger_question"
    }
  ],
  "coverage_stats": {
    "total_tags": 85,
    "fully_covered": 61,
    "partially_covered": 14,
    "uncovered": 10
  }
}
```

🎯 Used By:

- **FSD_04**: to generate follow-up Qs for uncovered/partial tags
- **Prompt Chain**: to inject relevant tag context into LLM behavior
- **Validator Module**: for launch decision logic
- **Agent Memory Builder**: for long-term semantic recall

📎 References:

- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* — defines minimum coverage %
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* — routes prompts using tag anchors
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* — defines all tag metadata

---

Section 6: Confidence Routes

🎯 Purpose:

To determine how the system behaves when tag matches fall below trusted thresholds — so that uncertainty is **flagged, explained, and traceable**, not silently passed forward.

🧠 Confidence Handling Logic:

Confidence Range	Action
`≥ 0.85`	✅ Accept and assign tag confidently
`0.65 – 0.84`	⚠️ Assign tag with `low_confidence = true`; flag for review
`< 0.65`	❌ Do not assign; log for fallback routing in FSD_04

📦 Example Output With Low Confidence:

```
{
  "segment_id": "seg_91",
  "matched_tag": "data_retention_policy",
  "confidence": 0.68,
  "match_method": "embedding_only",
  "low_confidence": true,
  "audit_note": "Match surfaced via vector proximity but no keyword hit."
}
```

🔄 Optional Fallback Routes:

1. **Retry With Alt Model**
   e.g. OpenCLIP instead of SentenceTransformers
2. **Trigger Context-Aware Prompt**
   FSD_04 may auto-generate a clarifying question:
   "Do you store customer data permanently, or is it deleted over time?"
3. **Log to Audit Trail**
   Low-confidence matches are recorded to `tag_link_audit.json` for transparency
4. **Suppress From Launch Decision**
   Pre-Launch Validator will **exclude weak tags** from coverage calculations

---

🔎 Confidence Logging Schema:

```
{
  "segment_id": "seg_02",
  "tag_attempted": "compliance_risk",
  "confidence_score": 0.61,
  "reason": "OCR noise; match too vague",
  "handled_by": "fallback_routing"
}
```

📎 References:

- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – defines what counts as “covered”
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* – includes test cases for low-match outputs
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – enables “fallback prompt” behavior

---

Section 7: Cross-Segment Anchors

🎯 Purpose:

To identify and establish **semantic or contextual links** between different content segments that:

- Refer to the same topic using different language
- Represent continuation or elaboration (e.g. intro → table → conclusion)
- Form a multi-block view of the same discovery tag

This creates **“reference chains”** the agent can use to cite multiple parts of a document or to track evolving logic across a file.

🧷 Anchor Types:

Anchor Type	Description	Example
Soft Semantic Anchor	Two blocks with high embedding similarity, no explicit link	"We earn via commission" ↔ "5% platform fee charged per transaction"
Structural Anchor	Blocks derived from the same table or document section	Table header → Table row (KPI table)
Explicit Reference Anchor	Direct text cues	“See vendor payout chart below” ↔ “Vendor Payout Chart (Table 4)”
Multi-block Tag Anchor	Multiple segments assigned the same tag within same file	Segments 4, 8, and 13 all tagged `vendor_model`

🧠 How Anchors Are Created:

- Use cosine similarity between segment embeddings
- Use metadata from *AUDIRA FILE & DATA UPLOAD SCHEMA* (e.g. `parent_block`)
- Detect linking phrases (e.g. “as detailed above”, “continued in next table”)
- Reinforced when matched to the **same discovery tag**

📦 Anchor Object Format:

```
{
  "anchor_id": "anchor_vendor_model_q1",
  "segments": ["seg_04", "seg_08", "seg_13"],
  "tag": "vendor_model",
  "anchor_type": "multi_block_tag",
  "confidence": 0.91,
  "source_file": "vendor_pricing_q1.pdf"
}
```

📎 References:

- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – structural lineage for segment relationships
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – used to validate tag-based anchors
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – uses anchors to inject multiple segment memories

---

Section 8: Future Enhancements

🚀 Enhancements Roadmap

Enhancement	Description	Why & When
Tag Overlap Heatmap	Visualize which tags often co-occur in a customer’s data	Helps refine tag clusters & agent persona training
Temporal Linking	Track changes in the same tag across time (e.g. Q1 vs Q2 revenue)	Enables trend-aware simulation and smarter follow-ups
Explainable Matching Output	Show rationale for each match: keywords, embeddings, file cues	Increases trust for admins; useful in edge cases
Industry-Weighted Scoring	Adjust tag scoring based on customer’s industry vertical	Increases tagging precision in sector-specific vocabularies

Crowd-Verified Tag Memory	Pull common tag-anchor links from other agents/clients (optionally)	Powers faster startup memory and transfer learning across SMBs

---

🧩 Suggested Open-Source Tools (All MIT or permissive):

Function	Tool
Cross-document linking	`Haystack`, `LlamaIndex`, `DeepLake`
Vector DB & clustering	`FAISS`, `Chroma`, `Weaviate`
Visualization	`NetworkX`, `Plotly`, `D3.js`
Explainability	`LIME`, `SHAP`, `Captum`
Tag inference from embeddings	`BERTopic`, `KeyBERT`, `FastEmbed`

📎 Linked Specs:

- *DISCOVERY TAGS DICTIONARY* – extended with `related_tags` and `temporal_validity`
- *PRE-LAUNCH VALIDATOR* – may factor anchor density into readiness score
- *PROMPT CHAIN LOGIC* – may pull all segments in an anchor cluster into long-prompt format