FSD_02: Segment & Classify Engine.

🧠 Purpose of FSD_02

This module receives raw content blocks from `FSD_01` (e.g. paragraphs, tables, images, audio transcripts) and performs:

- **Segmentation**: Breaks large blocks into meaningful sub-segments
- **Classification**: Tags each segment with type, intent, and potential tag categories
- **Contextual role detection**: Determines if a block is explanatory, summary, instruction, etc.
- **Pre-linking enrichment**: Adds embeddings and context markers used in FSD_03

📎 References:

- [AUDIRA FILE & DATA UPLOAD SCHEMA]
- [AUDIRA PRODUCT BLUEPRINT]
- [AUDIRA PROMPT CHAIN & LLM LOGIC FLOW]
- [AUDIRA AGENT ONBOARDING FRAMEWORK]

📄 FSD_02 Section Plan

Section	Description
1. Scope	What this module handles vs. doesn't
2. Input Format	Expected input (from FSD_01)
3. Segmentation Rules	How content blocks are split
4. Classification Types	What types of content are detected
5. Enrichment Logic	Adding embedding, tags, references
6. Output Schema	Structured output passed to next module
7. Confidence Handling	Low-confidence fallback routes
8. Future Enhancements	Optional add-ons and open-source logic

Section 1: Scope

🔍 Module Objective:

This module transforms raw input blocks — received from FSD_01 (Multi-format Bulk Input Handling) — into **semantically segmented, labeled, and enriched content units**, which can later be understood, referenced, and reasoned over by downstream Audira modules.

✅ Responsibilities:

The Segment & Classify Engine is responsible for:

1. **Sub-segmenting Complex Blocks**
   - Breaks paragraphs into logical sub-segments (e.g. bullet points, disclaimers, terms)
   - Splits tables by row or section when required for individual meaning
2. **Content-Type Classification**
   - Identifies the structural class of each block:
     `text_paragraph`, `table`, `image_caption`, `voice_transcript`, etc.
3. **Intent & Role Classification**
   - Infers business-relevant function:
     `summary`, `policy clause`, `financial figure`, `instruction`, etc.
4. **Triggering Tag Candidates**
   - Suggests possible matches to Audira's Discovery Tags Dictionary (cross-referenced against *AUDIRA DISCOVERY TAGS DICTIONARY.docx*)
5. **Segment Metadata Enrichment**
   - Adds unique IDs, context lineage (e.g. page, parent block), position metadata
6. **Routing to Next Stage**
   - Prepares clean handoff to FSD_03 (Reference Linking) and the Prompt Pipeline

---

❌ Out of Scope:

This module **does not**:

- Perform long-range document linking (handled in FSD_03)
- Make decisions about agent simulation, tone, or qualification (FSD_04/FSD_05)
- Finalize tag confidence scoring (initial candidate tagging only)

📎 Internal References:

- Input schema defined in *AUDIRA FILE & DATA UPLOAD SCHEMA*
- Prompt-ready tag format aligned with *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW*
- Tag match dictionary defined in *AUDIRA DISCOVERY TAGS DICTIONARY*

Section 2: Input Format

(From `FSD_01: Multi-format Bulk Input Handling Engine`)

📥 Expected Input Schema

Each file is converted into one or more **structured content blocks** with consistent metadata. This module expects an array of such blocks, each having the following fields:

```
{
  "block_id": "blk_001",
  "file_id": "doc_2025_05_001",
  "block_type": "paragraph" | "table" | "image" | "caption" | "transcript",
  "text": "...",
  "page_number": 3,
  "position": { "x": 74, "y": 112 },
  "metadata": {
    "source_format": "pdf",
    "lang": "en",
    "extracted_from": "table_2",
    "confidence": 0.91
  }
}
```

🧠 Business-Ready Input Assumptions

Type	Must Include	Example
Text	Text content, page number, lang	Paragraphs, bullet lists, terms
Tables	Row-wise structure, headers	KPI tables, pricing, timelines
Images	OCR output (if text-based), alt or caption fallback	Signature scan, flowcharts
Voice	Transcript + speaker ID (optional)	Meeting summary, voice memo
Mixed	Nested structure with tags	PDF with text + tables + annotations

---

📎 Based On:

- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA*
- 🧩 Compatible with *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* for downstream use
- 🗂 Used as reference for enrichment logic in Section 5

Section 3: Segmentation Rules

🎯 Purpose:

Segmentation breaks large or compound content blocks into meaningful **atomic segments** for classification and downstream reference linking. This enables Audira to reason over fine-grained elements like rows, clauses, or bullets — not just whole paragraphs or images.

🧱 Core Segmentation Logic:

Input Type	Segmentation Logic	Examples
Paragraphs	Split by bullet points, numbered lists, newline distance, or punctuation weight	A policy clause with sub-points becomes 4 separate blocks
Tables	Split by row, section header, or merged cells	Each row in a pricing or KPI table becomes a `table_row` segment
Images	Split only if annotated (e.g. OCR-detected labels, diagram legends)	Infographic with 3 labeled parts → 3 image-text pairs
Voice Transcripts	Split by speaker turns or long pauses	Interview audio = multiple `transcript_segment`s
Multi-format (PDF pages)	Compound blocks split by layout zones (top/bottom, header/body)	Scanned report → `summary_box`, `metrics_table`, `disclaimer_text`

🧠 Enhancement Flags:

- `segmentable: true/false` — Flags blocks that should not be split (e.g. legal terms)
- `segmentation_method`: `bullet`, `table_row`, `layout_split`, `ocr_zone`, etc.

📎 References:

- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* (defines base `block_type`)
- 📘 *AUDIRA PRODUCT BLUEPRINT* (notes need for fine-grain understanding)
- 🧩 Used to fuel pre-linking in FSD_03 and prompt slicing in *PROMPT CHAIN & LLM LOGIC FLOW*

Section 4: Classification Types

🧠 Purpose:

Every segment created in Section 3 is classified into:

- A **Content Type** (what it is structurally)
- A **Semantic Role** (what it means in context)
- A **Potential Intent Tag** (what it's related to functionally)

This allows later modules (FSD_03, Prompt Chain) to connect the segment to user intent, discovery tags, and agent behavior.

🧱 Classification Layers:

Layer	Key	Examples
Content Type	`text_paragraph`, `table_row`, `image_caption`, `voice_segment`, `ocr_zone`, etc.	"This is a row in a financial summary table."
Semantic Role	`policy_clause`, `financial_kpi`, `feature_description`, `customer_instruction`, `faq_response`, `deadline_notice`	"This is a pricing condition clause."
Intent Category (Pre-Tag)	`revenue_model`, `vendor_terms`, `payment_flow`, `crm_usage`, `employee_roles`, etc.	"This likely relates to vendor payout cycle."

🧠 Classification Method:

- Embedding + similarity to pre-trained labeled sets
- Pattern-based detection (e.g. `%`, currency, date)
- Heuristic rules (e.g. "starts with *to be eligible…* → policy")

🧪 Confidence Threshold:

Each classification output includes:

```
json
CopyEdit
"classification": {
  "type": "table_row",
  "semantic": "kpi_snapshot",
  "intent_tag": "revenue_model",
  "confidence": 0.88
}
```

Low-confidence outputs are routed to fallback logic (defined in Section 7).

📎 References:

- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* (intent category match)
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* (influences which prompts are triggered)
- 📘 *AUDIRA PRODUCT BLUEPRINT* (supports AI memory structuring)

Section 5: Enrichment Logic

🎯 Purpose:

This step enriches each classified segment with additional metadata, semantic context, and embedding representations. This is what allows the downstream system (FSD_03: Reference & Tag Linker) to link related knowledge, resolve cross-references, and create prompt-aware agent memory.

🧱 Enrichment Layers:

Enrichment	Description	Output Format
UUID (Segment ID)	Global unique identifier per atomic segment	`seg_8730fd29...`
Lineage	File ID, Page #, Parent Block ID, Split Origin	`{ file_id, page: 2, parent_block: "blk_001", source: "table_3" }`

LLM Embedding Vector	Multi-layer embedding (OpenCLIP, LLaMA2, SentenceTransformers)	`embedding: [0.023, -0.344, ...]`
Enrichment Flags	Flags like `summarizable`, `prompt_sensitive`, `rag_indexable`, `tag_suggested`	`prompt_sensitive: true`
Cross-link Anchors	Placeholder for future internal references (e.g. "see KPI snapshot above")	`anchor_id: "kpi_section_q1"`
Related Tags	Early-stage suggested discovery tags for refinement	`suggested_tags: ["recurring_revenue", "vendor_payout"]`

🧠 How This Helps:

- Anchors memory → enhances *agent simulation* (Agent Simulation Test Kit)
- Creates backward references to file blocks (e.g. when answering: "What does this vendor clause mean?")
- Boosts *tag confidence scoring* when documents repeat a motif (see: DISCOVERY TAGS DICTIONARY)

📎 References:

- 📘 *AUDIRA PRODUCT BLUEPRINT* → Agent memory / reusable knowledge layer
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* → Embedding-based prompt routing
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* → Source block lineage requirements

---

Section 6: Output Schema

📦 Output Object Per Segment

Each processed and enriched segment is delivered as a self-contained JSON object, ready to be used by:

- FSD_03: Tag & Reference Linker
- Prompt Chain & RAG Layer
- Agent Memory Store
- Validator System (for tag gap coverage scoring)

📦 Output Format (Per Segment):

```
{
  "segment_id": "seg_0b2e30...",
  "file_id": "doc_2025_05_001",
  "block_type": "table_row",
  "semantic_role": "kpi_snapshot",
  "text": "Net recurring revenue in Q1 was $40,120",
  "page": 3,
  "position": { "x": 80, "y": 220 },
  "parent_block": "blk_001",
  "lineage": {
    "origin": "table_3",
    "split_method": "row"
  },
  "embedding": [0.024, -0.733, 0.115, ...],
  "suggested_tags": ["revenue_model", "quarterly_kpi"],
  "enrichment_flags": {
    "prompt_sensitive": true,
    "rag_indexable": true
  },
  "confidence": 0.91,
  "anchor_id": "kpi_section_q1"
}
```

📎 Data Dictionary:

Field	Purpose
`segment_id`	Global ID for agent memory & retrieval
`semantic_role`	Functional purpose: `instruction`, `policy_clause`, `feature`, etc.
`embedding`	Multi-model vector for matching/tagging
`enrichment_flags`	Used by simulation, prompt, and retrieval flows
`suggested_tags`	Used for downstream discovery validation
`anchor_id`	Used for cross-reference or summary cue

🔗 Output Sent To:

- ✅ FSD_03: for tag linkage + cross-segment reasoning
- ✅ Prompt Trigger System: via *PROMPT CHAIN & LLM LOGIC FLOW*
- ✅ Validator: for *coverage score analysis* in *PRE-LAUNCH VALIDATOR SPEC*

📄 FSD_02 – Section 7: Confidence Handling

🎯 Purpose:

Not every segment will yield high-confidence classifications, especially in:

- Ambiguous or noisy input
- Scanned tables with OCR errors
- Mixed-language content
- Handwritten or diagrammatic text

This section ensures such cases are **systematically flagged, routed, and monitored**, not ignored or lost.

🧠 Confidence Ranges & System Behavior:

Confidence Score	System Behavior
`> 0.85`	✅ Auto-accepted for downstream processing
`0.65 - 0.85`	⚠️ Marked as `low_confidence = true`; passed forward but flagged
`< 0.65`	❌ Routed to fallback handler for reprocessing, admin validation, or isolation

---

🧩 Fallback Handlers:

1. **Retry with Alternate Model**
   e.g. If SentenceTransformer fails, retry with OpenCLIP
2. **Agent Review Pool**
   Segments with `low_confidence = true` are added to a review queue in simulation
3. **Tag Suggestion Suppression**
   Suggested tags are disabled from contributing to auto-discovery if score < 0.65
4. **Audit Logging**
   All low-confidence segments are written to a `confidence_log.json` for downstream monitoring or dashboard alerts

---

🛠 Config Flags:

```
"confidence_handling": {
  "allow_low_confidence_segments": true,
  "retry_alternate_model_on_fail": true,
  "log_uncertain_tags": true,
  "auto_route_to_validation": true
}
```

📎 References:

- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* — tag confidence readiness
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* — scenario for fallback testing
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* — defines metadata for confidence & segment trust score

---

Section 8: Future Enhancements

🚀 Planned Improvements

Enhancement	Description	Justification
Multilingual Classification Layer	Add language-aware segment analysis	Critical for global SMB clients using Arabic, French, Hindi, etc.
Visual + Text Fusion Classification	Use hybrid models for diagrams or scanned handwriting	Helps classify visual-rich uploads like contracts or invoices
Knowledge-Weighted Scoring	Boost segments that match past high-performing tags or industries	Improves prompt targeting and simulation accuracy
Feedback Loop Injection	Enable agent behavior to refine classification in real-time	Allows corrections by the agent itself during interaction
Explainability Mode	Add reasons behind classification (e.g. “due to currency symbols”)	Enhances debugging, trust, and admin auditability

🧩 Tooling Suggestions (All Open Source / MIT)

Task	Suggested Stack
Semantic classification	`sentence-transformers`, `OpenCLIP`, `spaCy`
OCR preprocessing	`Tesseract`, `LayoutParser`, `PaddleOCR`
Tag clustering	`BERTopic`, `Faiss`, `Haystack`
Metadata validation	`pydantic`, `cerberus`
Explainability	`LIME`, `SHAP`, `Captum`

🔗 Related Enhancements In:

- *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* → For hybrid prompt routes
- *AUDIRA AGENT SIMULATION TEST KIT* → To simulate explainable outputs
- *AUDIRA DISCOVERY TAGS DICTIONARY* → Expand with real-time enriched tags

✅ That concludes the full **FSD_02 – Segment & Classify Engine**.