Comprehensive AI System for SSM MBRS 2.0 XBRL Filings

Augmented with Qwen – double check with DeepSeek.

**With Built-in LLM Evaluation for Accuracy, Reliability & Continuous Improvement**

Target: Automate FS-MFRS, FS-MPERS, AR, EA, etc. for all Malaysian company types  
Core Tech: Docling + Agentic LLM + XBRL Generator + Evaluation Layer

**📊 Why LLM Evaluation Matters in This Context**

|  |  |
| --- | --- |
| Risk without Evaluation | Business Impact |
| Mis-mapping “Cost of Sales” → wrong concept | Invalid XBRL →rejection by SSM |
| Missing mandatory DEI field (e.g., Reg No) | Filing failure– ERROR in mTool |
| Incorrect context (Instant vs Duration) | XBRL validation fails |
| Hallucinated textBlock content | Legal/compliance risk |
| Silent model drift over time | Costly manual rework |

💡 Key Insight: Standard LLM metrics (e.g., BLEU, ROUGE) are useless here. We need domain-specific, actionable metrics.

**🔍 What to Evaluate – Critical Dimensions**

**Table: LLM Evaluation Dimensions for SSM XBRL AI**

| **Dimension** | **What to measure** | **Why it matters** | **How to measure** |
| --- | --- | --- | --- |
| 1. Concept Accuracy | % of line items mapped to correct taxonomy concept | Wrong concept = invalid XBRL | Compare LLM output vs gold-standard mapping (manual or rule-based) |
| 2. Mandatory Field Coverage | % of mandatory fields (\*in Excel) filled | Missing =ERROR in SSM validation | Count filled vs required DEI + FS fields |
| 3. Context Correctness | % of facts with correct context (as of vs from to) | Wrong context =XBRL schema error | Validate against reporting period logic |
| 4. Numeric Fidelity | Absolute/relative error in extracted numbers | Financial misstatement =regulatory risk | Compare extracted vs ground-truth values |
| 5. TextBlock Faithfulness | % of narrative text faithfully copied (no hallucination) | Legal disclosure must be verbatim | Use exact match or semantic similarity (e.g., BERTS core) |
| 6. Filing-Type Consistency | Correct entry point used (e.g., FS-MFRS vs AR1) | Wrong entry point =wrong taxonomy | Validate against detected company/filing type |

✅ Do NOT measure: Perplexity, generic QA accuracy, or "helpfulness"

**🧪 Evaluation Strategy – From Day 1**

**Step 1: Build a** Gold Standard Test Set

* Curate 20–50 real filings (PDF + correct XBRL)
* Cover:
  + Company types: Private, Public, CLBG, FC
  + Filing types: FS-MFRS, FS-MPERS, AR1, EA2
  + Edge cases: Zero values, notes-only disclosures, multi-currency

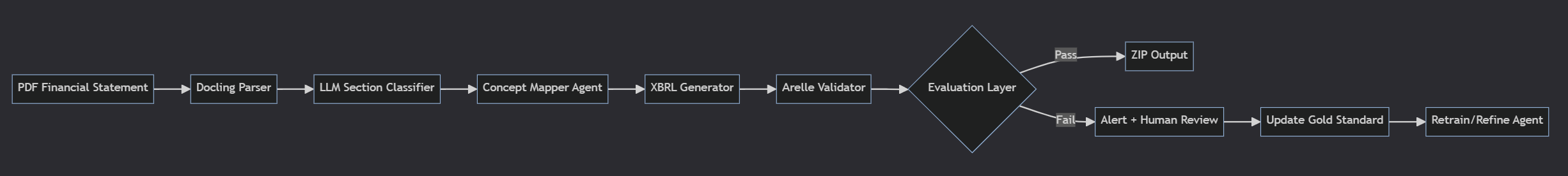
**Step 2: Define** Evaluation Scripts

|  |
| --- |
| # eval\_concept\_accuracy.py |
| def evaluate\_concept\_mapping(predicted\_concept, expected\_concept): |
| return predicted\_concept == expected\_concept |
|  |
| # eval\_numeric\_fidelity.py |
| def evaluate\_numeric\_error(predicted\_value, expected\_value, tolerance=0.01): |
| if expected\_value == 0: |
| return predicted\_value == 0 |
| return abs(predicted\_value - expected\_value) / abs(expected\_value) <= tolerance |

**Step 3: Run Evaluation** After Every Change

* Integrate into CI/CD pipeline (GitHub Actions, GitLab CI)
* Block deployment if:
  + Concept accuracy < 98%
  + Mandatory coverage < 100%
  + Numeric error > 1%

**🏗️ Updated Architecture with Evaluation Layer**



**Evaluation Layer Components:**

1. Accuracy Checker: Validates concept mapping
2. Completeness Checker: Ensures all mandatory fields filled
3. Fidelity Checker: Compares numbers/text to ground truth
4. Regression Monitor: Tracks metrics over time

**📈 Evaluation Dashboard (Example)**

**Evaluation Dashboard (Example)**

| **Metric** | **Target** | **Current** | **Status** |
| --- | --- | --- | --- |
| Concept Accuracy | ≥98% | 99.2% | ✅ |
| Mandatory Coverage | 100% | 100% | ✅ |
| Numeric Error Rate | ≤1% | 0.3% | ✅ |
| TextBlock Faithfulness | ≥95% | 96.7% | ✅ |
| Filing-Type Consistency | 100% | 100% | ✅ |

📉 Alert if: Any metric drops by >2% vs last run

**🔄 Continuous Improvement Loop**

1. Daily: Run eval on new filings
2. Weekly: Review failures → update prompts/rules
3. Monthly: Expand gold standard with new edge cases
4. On Failure: Auto-create Jira ticket with:
   * Input PDF snippet
   * Expected vs actual output
   * Suggested fix

**🧰 Implementation: Add Evaluation to Phase 1 Code**

**File:** eval/eval\_pipeline.py

|  |
| --- |
| from eval\_concept\_accuracy import evaluate\_concept\_mapping |
| from eval\_numeric\_fidelity import evaluate\_numeric\_error |
| import json |
|  |
| def run\_evaluation(predicted\_xbrl\_path, ground\_truth\_path): |
| # Load predicted facts |
| predicted = load\_xbrl\_facts(predicted\_xbrl\_path) |
| # Load ground truth |
| truth = load\_gold\_standard(ground\_truth\_path) |
|  |
| results = { |
| "concept\_accuracy": [], |
| "numeric\_error": [] |
| } |
|  |
| for fact\_id in truth: |
| pred = predicted.get(fact\_id) |
| if not pred: |
| results["concept\_accuracy"].append(False) |
| continue |
|  |
| # Check concept |
| results["concept\_accuracy"].append( |
| evaluate\_concept\_mapping(pred["concept"], truth[fact\_id]["concept"]) |
| ) |
|  |
| # Check number |
| if truth[fact\_id]["type"] == "monetary": |
| results["numeric\_error"].append( |
| evaluate\_numeric\_error(pred["value"], truth[fact\_id]["value"]) |
| ) |
|  |
| return { |
| "concept\_accuracy": sum(results["concept\_accuracy"]) / len(results["concept\_accuracy"]), |
| "numeric\_error\_rate": 1 - (sum(results["numeric\_error"]) / len(results["numeric\_error"])) |
| } |

**File:** main.py **(Updated)**

|  |
| --- |
| # ... after generating XBRL ... |
| from eval.eval\_pipeline import run\_evaluation |
|  |
| eval\_results = run\_evaluation( |
| "output/instance.xml", |
| "gold\_standard/MIHCM\_gold.xml" |
| ) |
|  |
| # Enforce thresholds |
| if eval\_results["concept\_accuracy"] < 0.98: |
| raise Exception("Concept accuracy below threshold!") |
| if eval\_results["numeric\_error\_rate"] > 0.01: |
| raise Exception("Numeric error too high!") |
|  |
| print("✅ Evaluation passed:", eval\_results) |

**📁 Project Structure with Evaluation**

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ Final Checklist with Evaluation**

|  |  |
| --- | --- |
| **STEP** | **DONE** |
| ✅ Set up dev environment (Win/Ubuntu) |  |
| ✅ Install Docling, LangChain, Arelle |  |
| ✅ Build Phase 1 pipeline (FS-MFRS) |  |
| ✅ Create gold standard test set |  |
| ✅ Implement evaluation scripts |  |
| ✅ Integrate eval into main pipeline |  |
| ✅ Set up CI/CD with eval gates |  |
| ✅ Design eval dashboard |  |

**📥 Deliverables**

This blueprint includes:

* Realistic evaluation dimensions (not vanity metrics)
* Actionable thresholds (98% concept accuracy, etc.)
* Code-ready evaluation scripts
* Architecture diagram with eval layer
* Continuous improvement loop