HUMAEIN

FULL-STACK AI/ML DEVELOPER_CASE STUDY

Scrubbing Case Study

Overview

You are asked to design a **Mini RCM Validation Engine**. This engine ingests claims data, validates against provided rule documents, persists adjudication outcomes, and presents results through a secure front-end. Treat this as a **production-minded prototype**.

The system must consist of:

- **Front-end:** Secure login, upload capability, results visualization (charts + tables).
- **Back-end:** API service with a **master table** storing all claims, errors, status, and explanations.
- **Data pipeline:** A modular data pipeline that reads **Technical** and **Medical** adjudication documents (files provided) and orchestrates multiple jobs dynamically. (Data validation, Static rule evaluation, LLM-based data evaluation)

The **data flow**:

- Claims file upload / API data pull (good to have) –> Master table –> validation –> Refined table –> Analytics pipeline (static rules and LLM) –> Metrics table
- All tables feeding into Charts/table into the FE UI

Analytics pipeline

- Must include 2 components - Static rule evaluation and LLM-based rule evaluation

Deliverables

Front-end

- Login screen (username/password).
- File upload interface (Claims file, Technical rules, Medical rules).
- Output views:
    - **Charts:** two waterfall charts (1) claim counts by error category, (2) paid amount by error category.
    - **Results table:** Each claim with status, Error type, explanation, and recommended action.
    - NOTE:
        - Status - Validated / Not validated
        - Error type - No error / Medical error / Technical error / both
        - Explanation - Tactically outline and explain the errors
            - Each error is one bullet
            - Explain why the error has happened based on the rules in the adjudication files
        - Recommendation action - Should be actionable, succinct, and targeted towards corrective action

Back-end

- API endpoints for file ingestion, validation, audit, and health check.

- **Master Table** must include:

  claim_id | encounter_type | service_date | national_id | member_id | facility_id | unique_id | diagnosis_codes | service_code | paid_amount_aed | approval_number | status | error_type | error_explanation | recommended_action

- Must support **multi-tenant config** (switch rule sets/files without code changes).

- The rule engine is separate and configurable (e.g., thresholds can change).

Rule Engine

- Parse **Technical** and **Medical** documents.
- Apply rules deterministically to classify errors.
- Must be modifiable (e.g., thresholds adjustable without code edits).

Final Written Questions

Answer these in ≤500 characters each:

1. What adjudication rules (technical + medical) did you extract from the documents?
2. What are your top five assumptions in implementing this case study?
3. What top five questions would you ask the system architect to improve your submission?

Submission

- Deploy backend + front-end (mock deployment acceptable).
- Submit by email: **careers@humaein.com**
- Subject: [Recruitment] Scrubbing Case Study – Your Name
- Include:
    - Demo link + login credentials
    - Git repository link
    - Admin/health endpoint
    - 5-minute recorded demo walkthrough
    - Answers to final written questions

Evaluation Rubric (100 pts)

- Rule correctness & explanations: 25
- Master table completeness & persistence: 20

- UI clarity (login, charts, tables): 15
- Multi-tenant config & dynamic parsing: 15
- Tests & CI/CD pipeline: 15
- Documentation & assumptions: 10

Pass for next interview: ≥75 overall, with no zero in rule correctness or master table.

Live Interview

- We will ask you to:
    - Show adjudication output for the provided artifacts.
    - Re-run validation on a **new claims file** (same schema, new data).
    - Optionally use **new Technical/Medical rule files** with different thresholds (e.g., higher Paid amount for approval)

Case artefacts

1. Claims file (Make a copy OR download as Excel file)
2. Medical rules file
3. Technical rules file