Technical Interview Exercise: Mining and Structuring Drug Indications from Labels

Objective

Develop a **microservice-based application** that extracts drug indications from **DailyMed** drug labels, maps them to standardized medical vocabulary (ICD-10 codes), and provides a **queryable API**. The implementation must be in **Python, Node.js, or .NET** and follow **enterprise-grade software principles**, including:

- **Test-Driven Development (TDD)**
- **Clean Architecture** (separation of concerns, layered design)
- **High Code Quality** (readability, maintainability, modularity)
- **Scalability & Performance Considerations**
- **Dockerized Deployment** (with `docker-compose` for execution)

Additionally there is a JSON document **dupixent.json**. It contains details about the **Dupixent MyWay Copay Card.** The JSON has a mix of structured and free-text eligibility details.

Requirements

1. Core Features

Data Extraction

- Scrape or parse **DailyMed** drug labels for **Dupixent**.
    - Extract relevant sections describing **indications**.
- Parse the MyWay Copay Card
    - Extract structured information
    - Infer missing details using rule-based transformations or generative AI
    - Standardize the format according to the example output included in this document

Indication Processing & Mapping

- Map extracted indications to **ICD-10** codes using an open-source dataset.
- Handle edge cases like:
    - **Synonyms** (e.g., "Hypertension" vs. "High Blood Pressure").
    - **Drugs with multiple indications**.
    - **Unmappable conditions**.

Eligibility Processing & Mapping

- Use generative AI to parse and summarize the **EligibilityDetails** text field into structured requirements.
- Convert **free-text eligibility conditions** into structured JSON key-value pairs.

Structured Data Output

- Store structured drug-indication mappings in a **database or NoSQL store**.
- Make mappings queryable via an API.

2. Enterprise-Grade API

- **Develop a Web API** using **.NET (C#), Python (FastAPI/Flask), or Node.js (Express/NestJS)**.
- Implement **CRUD operations**:
    - Create, read, update, and delete drug-indication mappings.
- **Authentication & Authorization**
    - Users should be able to register and log in.
    - Implement role-based access control.
- Include **Swagger or Postman workspace** for API testing.
- Ensure consistent data types (e.g., `true/false`, numbers as strings).
- Implement validation rules for missing or ambiguous data.
- Provides an endpoint (`/programs/<program_id>`) returning structured JSON.
- Supports querying program details dynamically.

3. Data & Storage Layer

- Use a **database** (SQL or NoSQL) to store:
    - Drug-indication mappings
    - User authentication data
- **DO NOT use Entity Framework, Dapper, or Mediator in .NET implementations.**

4. Business Logic Layer

- Keep **business rules** independent of the API and data layers.
- Implement validation logic for incoming data.

5. Testing & Quality

- Follow **TDD**: write unit tests **before** implementation.
- Cover:
    - Data extraction and processing logic.
    - API endpoints.

    - Business rules.
    - Authentication flows.
- Ensure **high test coverage**.

Deliverables

1. **GitHub Repository** containing:
    - **Source code** for the full project.
    - **Unit tests** for API, business logic, and data handling.
    - **README.md** with detailed setup and execution instructions.
2. **README.md must include:**
    - **Step-by-step setup** for running the project.
    - **API documentation**.
    - **Sample output** of the system.
    - **Scalability considerations**.
    - **Potential improvements & production challenges**.
3. **Dockerized Deployment**
    - Project must be runnable using `docker-compose up` as the only setup step.

Evaluation Criteria

Category	Description
Clean Architecture	Separation of concerns, modularity, maintainability.
Test-Driven Development	Unit tests for API, business logic, data layer.
Code Quality	Readability, documentation, adherence to best practices.
Functionality	API correctness, data extraction accuracy, ICD-10 mapping.
Scalability & Design	Consideration for large-scale use, error handling.
Dockerization	Ability to launch project using only `docker-compose up`.
Presentation (Interview)	Clear walkthrough of code, choices, and trade-offs.

Bonus Points

- **AI Extraction Improvement**:
    - Use an **LLM** (GPT, Claude, etc.) to extract more nuanced eligibility criteria.
    - For example, detect **age limits**, **geographic restrictions**, or **insurance conditions** from free-text fields.
- **Data Enrichment**:
    - Implement additional **rules-based logic** for missing fields.
    - Example: If **expiration date** is missing, assume **default end-of-year**.
- **Performance Optimization**:
    - Preprocess and **cache structured data** for API efficiency.
    - Allow filtering results dynamically based on parameters.
    - Implement **rate-limiting and security best practices**.

Submission & Interview

1. **Submit your GitHub repo link**.
2. **Prepare a Zoom presentation**:
    - Walk through your **user story**, **architecture**, and **technical decisions**.
    - Demo API functionality.
    - Answer code review questions.

Good luck!

Expected Output Structure For CoPay

Unset


```
{
    "program_name": "Dupixent MyWay Copay Card",
    "coverage_eligibilities": ["Commercially insured"],
    "program_type": "Coupon",
    "requirements": [
        {
            "name": "us_residency",
            "value": "true"
        },
        {
            "name": "minimum_age",
            "value": "18"
        },
        {
            "name": "insurance_coverage",
            "value": "true"
        },
        {
            "name": "eligibility_length",
            "value": "12m"
        }
    ],
    "benefits": [
        {
            "name": "max_annual_savings",
            "value": "13000.00"
        },
        {
            "name": "min_out_of_pocket",
            "value": "0.00"
        }
    ],
    "forms": [
        {
            "name": "Enrollment Form",
            "link": "https://www.dupixent.com/support-savings/copay-card"
        }
    ],
    "funding": {
        "evergreen": "true",
```

```
        "current_funding_level": "Data Not Available"
    },
    "details": [
        {
            "eligibility": "Patient must have commercial insurance and be a
legal resident of the US",
            "program": "Patients may pay as little as $0 for every month of
Dupixent",
            "renewal": "Automatically re-enrolled every January 1st if used
within 18 months",
            "income": "Not required"
        }
    ]
}
```