Problem PACK: My Policy

Assessment Brief for Students

1. Background & Context

Insurance aggregators and banks often struggle with "Data Silos" where a single customer's policies are scattered across different insurers in varying formats. While institutions possess large volumes of data, it is rarely "stitched together" to provide a meaningful, unified view. Currently, there is no modular, configurable framework that can ingest these heterogeneous files (CSV/Excel) and map them to a central customer identity.

2. Objective of This Task

You are required to design and demonstrate an **End-to-End Policy Management System** that:

- **Ingests & Validates:** Reads multi-source data from various insurers, performing automated data validation and massaging.
- **Maps Records:** Uses a metadata-driven engine to link disparate policy records to a central Customer Master file.
- **Visualizes:** Provides a **Front-End Interface** where customers can log in to view their unified insurance portfolio and identify protection gaps.

3. Requirements & Deliverables

A) Data Ingestion & Engineering (Backend)

- **Metadata Engine:** Build a configuration layer (JSON/YAML) to map source headers (e.g., "Insurer_ID") to your database schema without hard-coding logic.
- **Data Massaging:** Standardize formats (e.g., currency, date formats, and status codes) across different files.

- **Mapping Logic:** Policy information available from various insurers don’t have customer id information so to map the policy with individual customer, we need to map them using mobile number, email id’s, PAN number, along with combination of DOB etc. Create this rule as a generic layer to map source information with individual customers.
- **Data encryption:** implement data encryption for PII of customer (data at rest, data at transit)

B) Customer Portal (Front-End)

- **Secure Login:** A customer-facing entry point to access personal records.
- **Unified Portfolio View:** A dashboard showing all policies (Life, Health, Term) in one place.
- **Coverage Insights:** A visual summary highlighting "Protection Gaps"—showing the user where they might be underinsured.
- **Human-Readable Advisory:** Explanations for each policy and why specific details matter to the user's financial health.

C) Documentation & Diagrams

- **HLD/LLD:** Architecture explaining the flow from raw data ingestion to UI rendering.
- **Data Flow Sequence:** A diagram showing how metadata is read and how validations are triggered before reaching the UI.
- **Agile Plan:** A sprint breakdown including roles and ticket assignments.

---

4. Technical Specifications for Metadata

Your engine must interpret the following fields during ingestion to drive the UI:

Header	Expected Output & UI Impact
customer_details	Used for login authentication and profile mapping.
validation_rules	Ensures the ingested CSV/Excel data meets quality standards before database entry.
existing_policies	Ingested data used to populate the "My Policy" dashboard.
coverage_parameter	Defines policy limits to be displayed on the UI (e.g., Sum Insured, Waiting Periods).

---

5. Expected Outcome (Success Metrics)

- **Single Unified Journey:** The customer experiences a seamless flow from login to policy viewing.
- **Config-Driven Logic:** Adding a new insurer's CSV format requires zero code changes—only a metadata update.
- **Advisory-Led UX:** The interface focuses on helping the customer understand their coverage rather than just listing data.