To help you finalize your project documentation, here are the **User Stories** and **JIRA Ticket Descriptions** for the first two critical sprints. These are designed to satisfy the core requirements of **metadata-driven ingestion** and **identity stitching**.

Sprint 1: Ingestion & Staging (The Foundation)

User Story 1.1: Metadata-Driven Mapping

As a System Administrator,

I want to configure source-to-target mapping via JSON/YAML files,

So that I can ingest new insurer CSV formats without changing the backend code.

- **JIRA Description:** Implement a MetadataEngine class that parses a configuration layer. This engine must map headers like "Insurer_ID" or "Policy_Type" from a raw CSV to the internal database schema.
- **Acceptance Criteria:** Successfully ingest a sample CSV where headers differ from the DB schema by only updating the JSON config.

User Story 1.2: Raw Data Landing Zone

As a Developer,

I want to store raw data in a temporary staging database,

So that we can recover from transformation failures without re-requesting files from insurers.

- **JIRA Description:** Create a Staging_DB schema to house raw, uncleaned data. This serves as the "source of truth" for the **Audit Trail**.
- **Acceptance Criteria:** Data is visible in the staging table immediately after the **File Watcher** triggers.

Sprint 2: Identity Resolution & Security (The Intelligence)

User Story 2.1: Rule-Based Identity Stitching

As a Customer,

I want to see all my policies from different insurers in one place,

So that I don't have to log into multiple portals.

- **JIRA Description:** Develop the **Identity Resolution Engine**. Since insurer files lack a common Customer ID , create a matching algorithm that uses a combination of **Mobile Number, Email ID, PAN, and DOB** to link records to a central Customer Master file.

- **Acceptance Criteria:** Two separate CSV records with the same PAN and Mobile number are successfully "stitched" to the same internal User ID.

User Story 2.2: PII Data Protection

As a Compliance Officer,

I want to ensure all Personally Identifiable Information (PII) is encrypted,

So that customer data is secure at rest and in transit.

- **JIRA Description:** Implement an **Encryption Service** using AES-256 for data at rest (database level) and TLS 1.2+ for data in transit. This must specifically cover fields like PAN, Phone, and Email.
- **Acceptance Criteria:** Database queries for the "Customer Master" table return encrypted strings for PII fields unless accessed via the authorized decryption key.

Summary Table for Sprint Planning

Sprint	Priority	Core Deliverable	Requirement Link
1	High	Metadata Engine & Massaging	Section 3.A
2	High	Identity Resolution & Encryption	Section 3.A
3	Medium	BFF Layer & Unified Dashboard	Section 3.B
4	Medium	Coverage Insights & Advisory	Section 3.B

Sprint 3: Core API & Auth Layer (The Bridge)

User Story 3.1: Secure Authentication & Session Management

As a Customer,

I want to log in securely using my credentials,

So that my personal policy and PII data remain protected.

- **JIRA Description:** Implement the **Auth Service** to handle user login. It must generate a secure **JWT (JSON Web Token)** upon successful authentication to be used for all subsequent API calls.
- **Acceptance Criteria:** A user can successfully log in and receive a token; unauthorized access to policy APIs returns a 401 Unauthorized error.

User Story 3.2: BFF Orchestration & Data Aggregation

As a Developer,

I want to implement a **Backend for Frontend (BFF)** layer,

So that I can aggregate data from multiple microservices into a single response for the UI.

- **JIRA Description:** Build the **BFF Service** to act as the primary entry point for the Web Portal. It will handle token validation with the **Auth Service** and fetch consolidated data from the **Core Policy APIs**.
- **Acceptance Criteria:** The BFF successfully merges data from the **Central Policy DB** and the **Analytics DB** into a single JSON response for the frontend.

Sprint 4: Unified Frontend Development (The Experience)

User Story 4.1: Unified Portfolio Dashboard

As a Customer,

I want to see all my insurance policies (Life, Health, Term) in a single view,

So that I can manage my entire portfolio without switching platforms.

- **JIRA Description:** Develop the **Unified Portfolio View** component in the frontend. This dashboard must render data fetched from the BFF, displaying key details like policy name, insurer, and status.
- **Acceptance Credits:** The dashboard correctly displays a "Life" policy and a "Health" policy side-by-side for a single "stitched" customer identity.

User Story 4.2: Coverage Parameter Visualization

As a Customer,

I want to see high-level limits like "Sum Insured" and "Waiting Periods" for each policy,

So that I know exactly what my coverage entails.

- **JIRA Description:** Create UI cards that display specific coverage_parameter data. This data must be pulled from the ingestion engine's output to ensure it matches the actual policy documents.

- **Acceptance Criteria:** Users can click on a policy to see a detailed breakdown of limits, waiting periods, and exclusions.

Technical Alignment Table

Sprint	Goal	Key Tech/Requirement	Success Metric
3	Secure Bridge	JWT Auth & BFF Layer	Single secure entry point for the UI.
4	Portfolio View	React/Angular Dashboard	Unified view of all policies (Life, Health, Term).

Sprint 5: Insights, Advisory & Reporting

User Story 5.1: Protection Gap Identification

As a Customer,

I want to see a clear visual summary of my "Protection Gaps,"

So that I can identify where I am underinsured compared to industry recommendations.

- **JIRA Description**: Build a logic engine that compares coverage_parameter data (like Sum Insured) from the **Central Policy DB** against standard benchmarks for the user's demographic.
- **Acceptance Criteria**: The UI must highlight specific categories (Life, Health, or Term) where the user's coverage is 0 or below recommended levels.

User Story 5.2: Human-Readable Financial Advisory

As a Customer,

I want to read simple, non-technical explanations of my policies,

So that I understand why specific coverage details matter to my financial health.

- **JIRA Description**: Implement an **Advisory Engine** that maps technical policy status codes and waiting periods to human-readable strings.
- **Acceptance Criteria**: Each policy in the **Unified Portfolio View** should have a corresponding "Why this matters" section that is easy for a layperson to understand.

User Story 5.3: Reporting & Compliance Tracking

As an Internal Auditor,

I want to access a reporting dashboard,

So that I can track the success rate of data ingestion and identity resolution across different insurers.

- **JIRA Description**: Create a **Reporting Service** that pulls metadata from the **Staging DB** and **Audit Trail**. This dashboard should track total records ingested, failed validation rules, and "stitched" versus "unmapped" records.
- **Acceptance Criteria**: Generate a weekly report showing the volume of data processed per insurer and the accuracy of the **Metadata Engine**.

Final Deliverable Checklist

To ensure you meet all the success metrics for your **End-to-End Policy Management System**:

Metric	Required Feature	Project Outcome
Unified Journey	Secure Login -> Dashboard	Customer experiences a seamless flow from auth to viewing.
Config-Driven	Metadata Engine	Adding a new insurer CSV requires zero code changes.
Stitched Identity	Identity Resolution Service	Policies map to a Central Master file using Mobile/PAN/Email/DOB.
Security	PII Encryption	Data is encrypted at rest and in transit.

o wrap up the **My Policy** project's Agile execution, here are the **User Stories** and **JIRA Ticket Descriptions** for **Sprint 6**. This final phase focuses on end-to-end validation, performance, and ensuring the success metrics—especially the "Zero-Code" mapping and "Stitched Identity"—are met.

Sprint 6: Testing & Success Validation

User Story 6.1: End-to-End "Single Unified Journey" Test

As a Quality Assurance Engineer,

I want to simulate a full customer journey from raw file ingestion to frontend rendering,

So that I can verify that the system works seamlessly as a single unit.

- **JIRA Description:** Execute an Integration Test where a raw CSV is placed in the **Ingestion Layer**, processed through the **Identity Resolution Engine**, and verified on the **Customer Web Portal**.
- **Acceptance Criteria**: A new policy ingested for "Nakshatra Bhat" appears on the dashboard within the defined latency period without manual intervention.

User Story 6.2: "Zero-Code" Metadata Validation

As a System Operator,

I want to add a new insurer's file format by only updating a JSON configuration file,

So that I can prove the system's modularity and reduce maintenance costs.

- **JIRA Description**: Perform a "New Insurer Onboarding" test. Create a JSON mapping for a previously unknown CSV structure and trigger the **Metadata Engine**.
- **Acceptance Criteria**: The system successfully ingests and maps the new file format without requiring any Java/Python code changes or service restarts.

User Story 6.3: Identity Stitching Accuracy Audit

As a Data Analyst,

I want to verify that the "Stitching" logic correctly groups policies based on Mobile, PAN, and Email,

So that users do not see other customers' data or experience fragmented portfolios.

- **JIRA Description**: Run a test suite containing edge cases (e.g., matching name but different PAN, or matching Phone but different DOB) to validate the **Identity Resolution Engine**.
- **Acceptance Criteria**: 100% accuracy in mapping policies to the correct **Central Customer Master** file based on the defined rule-set.

User Story 6.4: PII Security & Encryption Audit

As a Security Architect,

I want to perform a penetration test and database audit on encrypted fields,

So that I can confirm that PII is protected against unauthorized access.

- **JIRA Description**: Inspect the **Central Policy DB** and network traffic to ensure PII (PAN, Mobile, Email) is encrypted at rest and in transit using the **Encryption Service**.
- **Acceptance Criteria**: Direct database queries for PII return cipher-text, and API intercepts show only encrypted payloads or masked data.

Final Project Success Matrix

Goal	Feature	Verification
Data Silo Removal	Identity Resolution	Multiple CSV sources linked to one ID.
Scalability	Metadata Engine	New IC formats added via JSON only.
Advisory UX	Coverage Insights	UI highlights protection gaps & advisory.
Security	Encryption Service	100% PII encryption coverage