Prepared By:

Anuj Saraswat (Product Leader, Innefu labs)

Version 0.1

Date: May 29, 2025

Software Design Document (SSD)

Intelligence Fusion Center (IFC)

# 1. Introduction

## 1.1 Purpose

The purpose of this Software Design Document (SSD) is to provide a comprehensive architectural and detailed design for the Intelligence Fusion Center (IFC), Release V1.0.0. This document serves as a blueprint for development, guiding the implementation process and ensuring that the system is built to meet all specified functional and non-functional requirements. It translates the "Software Requirements Analysis Process" output (RCD) into a detailed design, aligning with ISO/IEC 12207's "Software Architectural Design Process" and "Software Detailed Design Process."

## 1.2 Scope

This SSD covers the architectural design, high-level component design, interface design, data design, and specific non-functional design considerations for the IFC System, Release V1.0.0. It defines the major modules, their interactions, and the underlying technologies. It outlines the design for all functionalities detailed in the Requirements Capture Document (RCD-IFC-V1.0). This document does not include detailed code-level implementation specifics for every minor function, but rather the overall structure and design patterns.

## 1.3 Definitions and Acronyms

* **API:** Application Programming Interface
* **BDR:** Big Data Repository
* **DB:** Database
* **DIS:** Data Ingestion Service
* **FR:** Functional Requirement
* **GIS:** Geographical Information System
* **GUI:** Graphical User Interface
* **IFC:** Intelligence Fusion Center
* **IMINT:** Imagery Intelligence
* **ISO/IEC 12207:** International standard for software life cycle processes.
* **LLM:** Large Language Model
* **NFR:** Non-Functional Requirement
* **OCR:** Optical Character Recognition
* **RCD:** Requirements Capture Document
* **RTM:** Requirements Traceability Matrix
* **SAS:** Search & Analytics Service
* **SSD:** Software Design Document
* **UI:** User Interface
* **V&V:** Verification and Validation

## 1.4 References

* Requirements Capture Document (RCD-IFC-V1.0)
* Test Plan Document (TP-IFC-V1.0)
* Test Result Report (TR-IFC-V1.0-20250603)
* ISO/IEC 12207: Systems and software engineering — Software life cycle processes

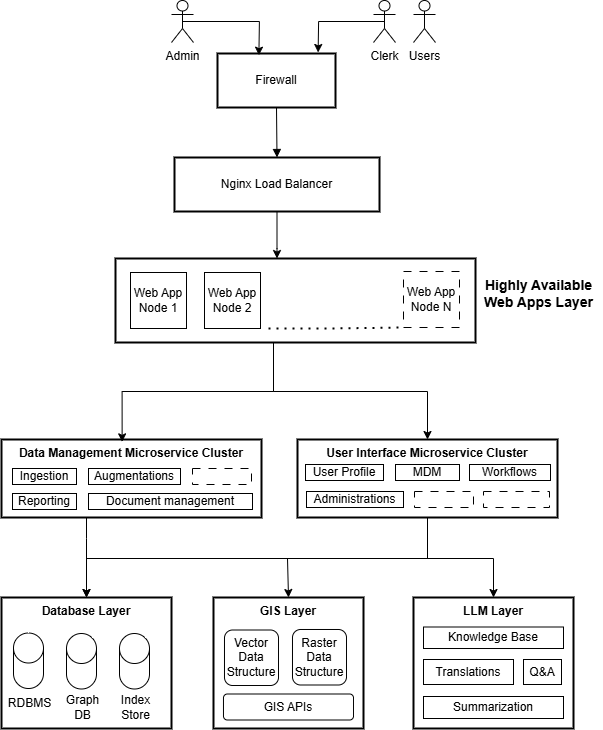
# 2. System Architecture

## 2.1 Architectural Overview

The IFC System employs a modul ar, microservices-oriented architecture designed for scalability, maintainability, and high availability. It follows a multi-tier approach:

* **Presentation Layer (UI/UX):** Web-based application accessible via standard browsers. Developed using modern front-end frameworks.
* **Application/Services Layer:** Core business logic, APIs, and microservices for various functionalities (e.g., Ingestion, Search, Workflow, Analytics). These services communicate via RESTful APIs.
* **Data Layer:** Persistent storage for structured, semi-structured, and unstructured data, including relational databases, NoSQL databases, and a search index.
* **Integrations:** Interfaces with other systems (e.g., LLM module, OCR engines, GIS).

The architecture is designed to allow for horizontal scaling of individual services and data stores.



## 2.2 Key Architectural Decisions

* **Microservices Pattern:** Enables independent development, deployment, and scaling of modules.
* **RESTful APIs:** Standardized communication protocol between services.
* **Event-Driven Architecture:** Utilized for asynchronous operations (e.g., data ingestion, long-running processing tasks) to ensure responsiveness.
* **Technology Stack Diversity:** Selection of best-fit technologies for specific tasks (e.g., Python for data processing and backend, Angular for frontend).

## 2.3 Design Principles

* **Modularity:** Separation of concerns, clear boundaries between modules.
* **Scalability:** Design for horizontal scaling of services and data stores.
* **Security by Design:** Security considerations integrated from the outset, not as an afterthought (e.g., RBAC, encryption).
* **Maintainability:** Clear code structure, consistent design patterns, comprehensive documentation.
* **Extensibility:** Ability to easily add new data sources, analytical modules, or reporting formats.
* **Fault Tolerance:** Mechanisms to handle failures gracefully and recover quickly.

# 3. Data Design

## 3.1 Data Model (Conceptual & Logical)

The IFC system's data model is designed to support diverse data types (structured, semi-structured, unstructured) and their complex interrelationships.

* **Conceptual Data Model:** High-level entities include Documents, Events, Orbat, Locations, Units, IMINT Records etc. Relationships exist between these entities (e.g., an Event can involve People at a Location and be described in Documents).
* **Logical Data Model:** This expands the conceptual model by defining attributes for each entity and specifying relationships with cardinality. Key attributes include metadata (source, ingestion date, classification), content (raw text, extracted entities), and temporal/geospatial properties.

## 3.2 Database Schema

The system employs a polyglot persistence strategy using different database technologies optimized for specific data types and access patterns.

#### **3.2.1 Raw Data Schema**

This schema represents the data as it is ingested from source systems, retaining its original structure as much as possible.

**Example (Conceptual):**

* **Excel Data:**
* filename: String (e.g., "unit2\_intell\_report.xlsx")
* sheet\_name: String (e.g., "Sheet1")
* row\_number: Integer
* column\_headers: Array of Strings
* row\_data: JSON Object (key-value pairs representing cell data)
* ingestion\_timestamp: Timestamp
* **Access DB / SQL DB Data:**
* database\_name: String
* table\_name: String
* record\_id: String/Integer (Primary Key from source)
* field\_data: JSON Object (key-value pairs representing column data)
* ingestion\_timestamp: Timestamp
* **Scanned/Digital Document Data:**
* document\_id: String (Unique identifier)
* document\_type: String (e.g., "PDF", "Word Document")
* original\_filename: String
* file\_path\_or\_url: String (Location of the original file)
* ingestion\_timestamp: Timestamp
* metadata: JSON Object (e.g., {'author': 'Ramesh', 'creation\_date': '23-01-15'})

#### **3.2.2 Processed Data Schema**

This schema represents data after cleansing, transformation, OCR, and initial text analytics. It is optimized for storage in the unified data warehouse and for search.

**Example (Conceptual):**

* **Unified Record (Event entity):**
* HashId: String (System-generated unique ID)
* EventDate: String (e.g., '2024-05-10')
* SourceCountry: String (e.g., 'China’)
* TargetCountry: String (e.g., 'Nepal’)
* EventType: String (e.g., 'Movement’)
* EMFLAG: Long (e.g., 56783568)
* Latitude: String (e.g., '123.456’)
* Longitude: String (e.g., '321.654’)
* Unit: String (e.g., 'Unit1a’)
* Addresses: String (e.g., 'PO box 11, Pitampura, new delhi’)
* EventId: String (e.g., '123456789’)

#### **3.2.3 Intelligence Output Schema**

This schema represents the actionable insights, summaries, and reports generated by the system, often derived from LLM processing or advanced analytics.

**Example (Conceptual):**

* **Intelligence Report:**
* report\_id: String (Unique ID)
* report\_title: String
* generation\_timestamp: Timestamp
* generated\_by\_user\_id: String (User who initiated generation or workflow)
* summary\_text: String (LLM-generated summary or analytical summary)
* key\_findings: Array of Strings
* associated\_record\_ids: Array of Strings (IDs of processed records used for report)
* report\_type: String (e.g., "Event Summary")
* status: String (e.g., "Draft", "Under Review", "Approved", "Published")
* output\_formats\_available: Array of Strings (e.g., "PDF", "Word", "HTML")
* geospatial\_context: JSON Object (Relevant locations for the intelligence)
* temporal\_context: JSON Object (Relevant timeframes for the intelligence)

# 4. Component Design

## 4.1 Module Breakdown

The IFC system is logically decomposed into the following major components/services:

* **Multilingual OCR:** Extracts text from ingested documents, supporting multiple languages.
* **Big Data Repository:** Stores and manages large-scale data for quick access and querying.
* **Search Tool:** Enables users to query and retrieve data from the repository.
* **Workflow Automation:** Orchestrates the overall process, automating sequences and dependencies between components.
* **Text Analytics:** Analyzes extracted text for insights, such as sentiment or keyword extraction.
* **Geospatial Analysis Module:** Processes location-based data for spatial insights.
* **Temporal Analysis Module:** Examines time-based patterns in data.
* **Graph Analysis Module:** Handles network or relationship-based data analysis.
* **Predictive Module:** Uses historical data to forecast trends.
* **IMINT Module:** Handles IMINT data for ingestion and visualization.
* **Widgets:** Display data visualizations or interactive dashboards.
* **Report Generation:** Compiles and formats outputs from various modules into reports.
* **AI/LLM Model:** Leverages artificial intelligence, such as large language models, for advanced processing (e.g., natural language understanding).

A diagram of a company

AI-generated content may be incorrect.

# 5. Interface Design

## 5.1 User Interface Design

The UI is a web-based application designed for intuitive interaction and responsiveness across desktop and tablet devices.

* **Layout:** Consistent header, navigation sidebar, and main content area.
* **Navigation:** Role-based menus providing access to relevant modules (Dashboard, Search, Ingestion, Workflows, Analysis, Reports, Admin).
* **Dashboards:** Customizable landing pages with widgets for quick overview.
* **Component Library:** Reusable UI components (buttons, forms, tables, charts, maps, timelines, graph visualizations) for consistency.
* **Error Handling:** Clear, user-friendly error messages and feedback.

*Please refer the IFC - Interface Design Document for more details*.

## 5.2 External System Interfaces

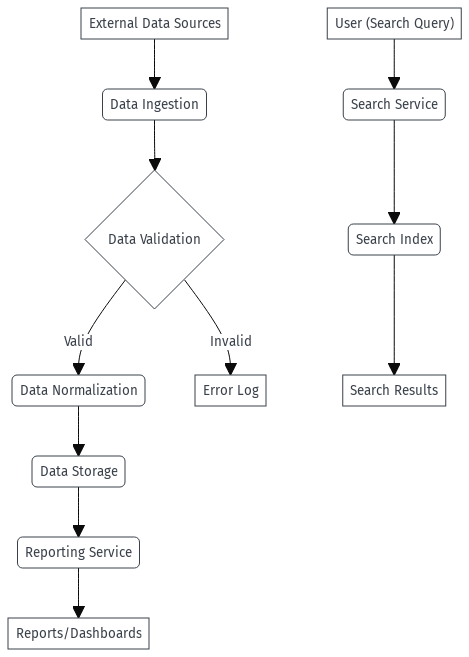
* **LLM API:** Flask API communication (HTTPS) with an internal self-hosted LLM for summarization and natural language search.
* **GIS Provider API:** Integration with an in-house mapping service APIs for base maps and advanced geospatial features.
* **OCR Engine API:** Interface with on-premises OCR solution for text extraction from images.
* **IFC Workstation:** Dedicated secure network connection (e.g., SFTP) for IMINT data transfer.

# 6. Diagrammatic Representations

This section outlines the purpose and scope of various diagrams that would be included in a completed Software Design Document. These diagrams visually represent the architectural and design decisions, providing clarity beyond textual descriptions.

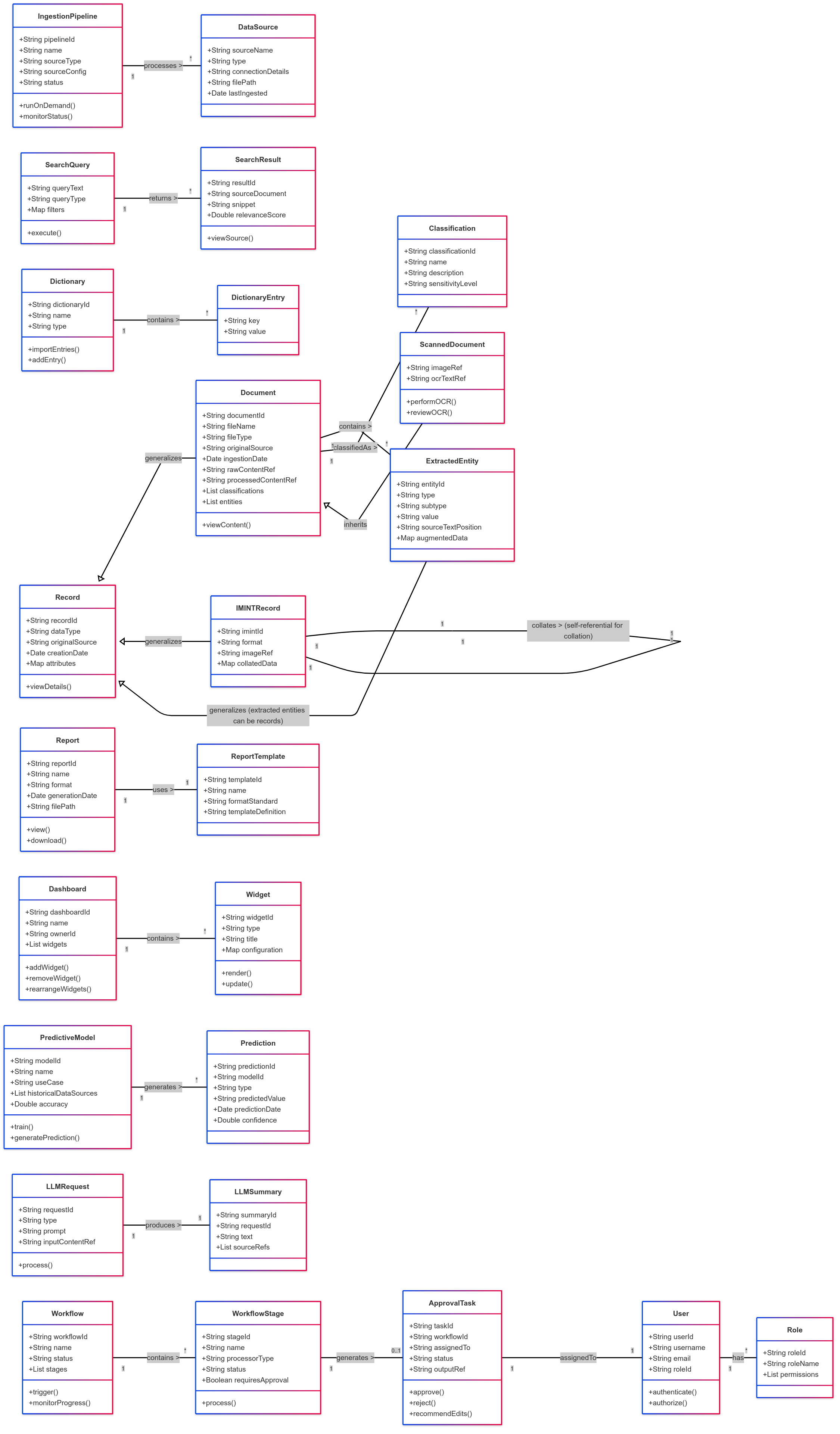
## 6.1 Data Flow Diagram (DFD)

To illustrate the flow of data through the IFC system, showing how information moves from external inputs, through various processes, and to data stores or external outputs. It helps understand data transformations and system boundaries.

****

## 6.2 Class Diagram

To illustrate the static structure of the IFC system, showing the system's classes, their attributes, methods, and the relationships between them. It is fundamental for object-oriented design.



## 6.3 Sequence Diagrams

To illustrate the dynamic behaviour of the IFC system by showing the sequence of messages exchanged between objects or components over time to perform a specific function or use case scenario.

**Workflow Automation (Example: Document Review & Approval Workflow)**

This diagram illustrates the flow for a document going through a review and approval workflow, initiated by the system and actioned by a user.

A screenshot of a computer

AI-generated content may be incorrect.

# 7. Approvals

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Name** | **Signature** | **Date** |
| Project Manager | [Manager's Name] |  | June 5, 2025 |
| Lead System Architect | [Architect's Name] |  | June 5, 2025 |
| Development Lead | [Dev Lead's Name] |  | June 5, 2025 |
| Product Owner | [PO's Name] |  | June 5, 2025 |