https://chatgpt.com/share/68045a96-c894-8000-b957-0b997b0cf360

https://chatgpt.com/share/680458d3-cfdc-8000-a774-a4518a6d895a

https://chatgpt.com/share/68045727-c098-8000-8f53-c4aa9c46667f

Gemini:  
<https://gemini.google.com/app/3bbb033bb6de44dd>

Okay, I have processed the document "Summarize.docx". Here is the Software Requirements Specification (SRS) generated as exhaustively as possible based on the content provided in that document.

**Software Requirements Specification (SRS)**

Project: Reactive Microservices Framework for Semantic Integration (Name inferred)

Version: 1.0 (Based on document being a "first draft" )

Date: April 20, 2025

Table of Contents

1. Introduction
   * 1.1 Purpose
   * 1.2 Scope
   * 1.3 Definitions, Acronyms, and Abbreviations
   * 1.4 References
   * 1.5 Overview
2. Overall Description
   * 2.1 Product Perspective
   * 2.2 Product Functions1 (High-Level Summary)
   * 2.3 User Classes and Characteristics
   * 2.4 Operating Environment
   * 2.5 Design and Implementation Constraints
   * 2.6 Assumptions and Dependencies2
3. Specific Requirements
   * 3.1 Functional Requirements3
     + 3.1.1 Data Ingestion and Transformation (Datasources Service)
     + 3.1.2 Data Aggregation (Aggregation Service)
     + 3.1.3 Data Alignment (Alignment Service)
     + 3.1.4 Context/Interaction Activation (Activation Service)
     + 3.1.5 Orchestration (Augmentation Service)
     + 3.1.6 Unified Access (Consumer API Service)
     + 3.1.7 Core Model Management (Helper Services)
     + 3.1.8 Administration and Management
   * 3.2 External Interface Requirements
     + 3.2.1 User Interfaces
     + 3.2.2 Hardware Interfaces
     + 3.2.3 Software Interfaces
     + 3.2.4 Communications4 Interfaces
   * 3.3 Non-Functional Requirements
     + 3.3.1 Performance Requirements
     + 3.3.2 Security Requirements
     + 3.3.3 Reliability Requirements
     + 3.3.4 Availability Requirements
     + 3.3.5 Maintainability Requirements
     + 3.3.6 Portability Requirements
     + 3.3.7 Usability Requirements
     + 3.3.8 Compliance/Regulatory Requirements
   * 3.4 Data Requirements
     + 3.4.1 Data Representation and Formats
     + 3.4.2 Data Persistence and Storage
     + 3.4.3 Data Synchronization and Provenance
4. Other Requirements / Features to Explore
5. Appendices

**1. Introduction**

* **1.1 Purpose**
  + The primary purpose of this software is to serve as a reactive microservices framework that integrates diverse existing applications, services, and backends.
  + It aims to unify the use cases and interactions possible within and between these integrated systems by leveraging Semantic Web technologies, inference, and reactive functional programming streams processing.
  + The goal is to parse source data from integrated applications and, through inference and learning, provide a unified interface to browse and perform possible use cases (Contexts) and their interactions (Transactions).
  + This framework facilitates Business Integration (BI) and Enterprise Application Integration (EAI) through Semantic Web techniques.
  + It seeks to integrate the domains and functionality of various applications into a unified API or interface (e.g., a unified front end).
  + Ultimately, it acts as a "generator" of unified interfaces for integrating current or legacy applications/data sources, exposing them cohesively via web frontends, chatbots, APIs, etc..
* **1.2 Scope**
  + The framework encompasses the integration of diverse applications, services, backends, or data sources (DBs, APIs, documents, etc.).
  + It includes parsing/extracting data from integrated sources and transforming it into a unified representation (RDF SPO Triples).
  + The system performs inference over the transformed data through layered services (Aggregation, Alignment, Activation).
  + Inference includes determining entity types, states, relationships, equivalences, order, and potential use cases (Contexts) and their execution steps (Interactions) involving Actors playing Roles.
  + The framework generates and exposes a unified API (e.g., REST) and potentially a generic frontend service for browsing and invoking the inferred Contexts and Interactions.
  + It manages the execution of interactions (transactions), potentially involving multiple integrated applications.
  + It synchronizes transaction data back to the original applications.
  + The scope includes providing administration/management interfaces for workflow steps.
* **1.3 Definitions, Acronyms, and Abbreviations**
  + **Activation:** The process/service layer responsible for inferring and managing Use Cases (Contexts), Roles, Actors, and Interactions (Transactions) based on aligned data. Uses DCI pattern.
  + **Actor:** Resources (data entities) that play Roles in Context Interactions.
  + **Aggregation:** The process/service layer responsible for inferring Types, States, and Order from raw SPO triples.
  + **Alignment:** The process/service layer responsible for inferring relationships, performing ontology matching, aligning data to upper ontologies (Domains, Order), and inferring links/attributes.
  + **API:** Application Programming Interface. Used for exposing integrated functionality.
  + **Augmentation Service:** The orchestrating service that manages the workflow between Datasources, Aggregation, Alignment, Activation, Consumer API, and Helper services.
  + **BI:** Business Integration.
  + **Consumer API Service:** The service layer exposing the unified interface (e.g., REST API, generic frontend) to end-users/clients.
  + **Context:** Represents a use case involving resources (Actors) playing Roles.
  + **Core Model Classes:** The fundamental data structures (URI, URIOccurrence, Resource, Type, State, Kind, Statement) used for data representation and streaming between services.
  + **Datasources Service:** The service layer responsible for ETL, synchronization, and providing source data as streams to the Augmentation Service.
  + **DCI:** Data, Context, Interaction design pattern, used by the Activation service.
  + **DIDs:** Decentralized Identifiers (W3C standard considered).
  + **EAI:** Enterprise Applications Integration.
  + **ETL:** Extract, Transform, Load. Process used by Datasources Service.
  + **FCA:** Formal Concept Analysis. Technique considered for use in Naming Service.
  + **HAL:** Hypertext Application Language. Potential standard for REST API.
  + **HATEOAS:** Hypermedia as the Engine of Application State. Principle for REST API design.
  + **Helper Services:** Shared orthogonal services (Registry, Naming, Index) provided via Augmentation Service.
  + **Index Service:** Helper service; repository of aligned resources for Activation, enabling similarity resolution and managing conversational state.
  + **Interaction:** An instance or execution of a Context (use case), representing a transaction.
  + **Kind:** Core Model Class representing the role (Type/State) of a URIOccurrence in a context.
  + **Naming Service:** Helper service; manages upper ontology matching/alignment (Domains, Order, Activation) and set representations.
  + **Ontology Matching:** Process of finding equivalences between entities/relationships in different source domains.
  + **Order:** Inferred sequence or dimensional relationship between entities based on Type/State hierarchies or measures.
  + **OWL:** Web Ontology Language. Mentioned in context of reification.
  + **RDF:** Resource Description Framework. Used for representing data as triples.
  + **RDFS:** RDF Schema. Mentioned in context of reification.
  + **Registry Service:** Helper service; core graph model repository, handles provenance, embeddings, potentially using Topic Maps model.
  + **REST:** Representational State Transfer. Architectural style for the Consumer API Service.
  + **Resource:** Core Model Class representing a URIOccurrence with content.
  + **Role:** The function or type assigned to an Actor within a specific Context Interaction.
  + **Semantic Web:** The collection of technologies (RDF, SPARQL, Ontologies) used by the framework.
  + **SPARQL:** SPARQL Protocol and RDF Query Language. Potential endpoint for Registry and Naming services.
  + **SPO Triples:** Subject-Predicate-Object triples, the fundamental data structure in RDF used for representing raw and processed data.
  + **State:** Inferred classification based on common attribute values.
  + **Statement:** Core Model Class representing an SPO triple relationship, treated as a URIOccurrence itself (reification).
  + **TMDM:** Topic Maps Data Model. Mentioned as potential underlying representation.
  + **TMRM:** Topic Maps Reference Model. Mentioned for Registry service and potential underlying representation.
  + **Transaction:** An execution instance of a use case interaction.
  + **Type:** Inferred classification based on common attributes.
  + **Upper Ontology:** Abstract ontologies (Domains, Order, Activation) used for alignment and inference.
  + **URI:** Uniform Resource Identifier. Used as the basis for identifying all entities (nodes and arcs) in the graph model.
  + **URIOccurrence:** Core Model Class representing a specific occurrence of a URI within a statement, having a Kind.
  + **Use Case:** A specific scenario or function within or between applications, represented as a Context.
  + **XML:** Extensible Markup Language. Mentioned in relation to HyTime and potential transforms.
  + **XSLT:** Extensible Stylesheet Language Transformations. Mentioned for potential dynamic transforms.
* **1.4 References**
  + DCI: Data, Context, Interaction design pattern.
  + ISO Topic Maps Reference Model (TMRM).
  + ISO Topic Maps Data Model (TMDM).
  + ISO 15926.
  + W3C Resource Description Framework (RDF).
  + W3C RDF Schema (RDFS).
  + W3C Web Ontology Language (OWL).
  + W3C Decentralized Identifiers (DIDs).
  + HyTime / XML.
  + HATEOAS / HAL.
  + Machine Learning techniques (Classification, Clustering, Regression).
  + Formal Concept Analysis (FCA).
* **1.5 Overview**
  + This document outlines the requirements for a reactive microservices framework utilizing Semantic Web technologies. Section 2 provides an overall description of the product, its context, and constraints. Section 3 details the specific functional, interface, non-functional, and data requirements for each component. Section 4 lists other requirements and features for exploration. Section 5 provides placeholders for appendices.

**2. Overall Description**

* **2.1 Product Perspective**
  + The product is a framework designed to integrate existing, diverse applications, services, or backends.
  + It acts as an overlay on top of existing systems, not necessarily replacing them.
  + It provides a unified interface (API and/or frontend) to interact with the combined functionality and data of the integrated systems.
  + It aims to enable interaction between different instances or deployments of the framework across organizations.
* **2.2 Product Functions (High-Level Summary)**
  + **Data Integration:** Extracts/transforms data from various sources (databases, APIs, documents) into a unified RDF triple format.
  + **Semantic Inference:** Infers entity types, states, relationships, equivalences, order, and potential use cases (Contexts/Interactions) from the integrated data using layered processing (Aggregation, Alignment, Activation).
  + **Unified Access Generation:** Creates a discoverable, unified API (e.g., REST) and/or a generic frontend (e.g., web wizards) exposing the inferred use cases and interactions.
  + **Use Case Management:** Allows users to browse past interactions, invoke new use cases (Contexts), and execute transactions (Interactions) involving actors playing roles.
  + **Data Synchronization:** Synchronizes data changes resulting from transactions back to the original source applications, managing provenance.
* **2.3 User Classes and Characteristics**
  + **Integrators/Administrators:** Users responsible for configuring the framework, adding datasources, managing the integration workflow (viewing inferred types, alignments, contexts), and potentially designing interactions. Technical expertise in the integrated systems and potentially Semantic Web concepts is implied.
  + **End-Users:** Users who interact with the generated unified interface (API or frontend) to browse available use cases, invoke interactions, provide necessary input (e.g., via forms/wizards), and achieve desired outcomes. Their characteristics are diverse, depending on the integrated applications.
* **2.4 Operating Environment**
  + The framework is architected as a set of microservices, implying a distributed deployment environment.
  + Requires connectivity to the backend datasources (databases, APIs, file systems) of the applications being integrated.
  + The Consumer API Service requires infrastructure to host a REST API and potentially a web server for the generic frontend.
  + Helper services require infrastructure for data storage (graph database, key/value store) and potentially SPARQL endpoints.
  + Potential use of Java APIs for DIDs is mentioned.
* **2.5 Design and Implementation Constraints**
  + **Architecture:** Must follow a reactive microservices architecture with specific layers: Datasources, Augmentation (orchestrator), Aggregation, Alignment, Activation, Consumer API, and shared Helper Services (Registry, Naming, Index).
  + **Data Model:** Must use Semantic Web technologies. The core data representation must be RDF SPO (Subject-Predicate-Object) Triples, based on the defined Core Model Classes. URIs are fundamental identifiers.
  + **Processing:** Must utilize reactive functional programming streams processing for communication and data manipulation between services.
  + **Inference:** Must implement inference logic for Type, State, Order (Aggregation); Relationships, Ontology Matching, Upper Ontology Alignment (Alignment); and Contexts, Roles, Interactions, Actors (Activation). Machine Learning techniques (Classification, Clustering, Regression) should be leveraged for these tasks.
  + **Design Patterns:** The Activation service/ontology must follow the DCI (Data, Context, Interaction) design pattern.
  + **API:** The Consumer API Service must expose a REST API, potentially adhering to HATEOAS/HAL principles.
  + **Potential Technologies:** Consideration should be given to SPARQL, XML/XSLT, W3C DIDs, FCA, Sets Representation, and Topic Maps models.
* **2.6 Assumptions and Dependencies**
  + **Data Accessibility:** Assumes the data sources of the applications to be integrated are accessible to the Datasources Service.
  + **Data Translatability:** Assumes that diverse source data formats (tabular, APIs, documents) can be meaningfully translated into RDF SPO triples. A basic translation for tabular data is suggested (Row PK, Column Name, Column Value).
  + **Inference Viability:** Assumes that the inference algorithms employed in Aggregation, Alignment, and Activation can successfully derive meaningful types, states, relationships, alignments, and use case descriptions from the data.
  + **Synchronization Capability:** Assumes that data resulting from transactions within the framework can be synchronized back to the original application backends where necessary.
  + **Semantic Coherence:** Assumes that semantic relationships and equivalences exist between the data of the applications to be integrated and that these can be discovered and modelled.

**3. Specific Requirements**

* **3.1 Functional Requirements**
  + **3.1.1 Data Ingestion and Transformation (Datasources Service)**
    - FR-DS-01: The Datasources Service shall extract data from the backend datasources (databases, APIs, documents, etc.) of the integrated applications.
    - FR-DS-02: The Datasources Service shall transform the extracted source data into RDF SPO Triples based on the Core Model Classes.
    - FR-DS-03: The Datasources Service shall provide the transformed data as graph model streams to the Augmentation Service.
    - FR-DS-04: The Datasources Service shall handle synchronization of data updates between the framework and the integrated application backends.
    - FR-DS-05: The Datasources Service shall manage data provenance information related to synchronized data.
    - FR-DS-06: The Datasources Service shall consume updated information from the Augmentation Service to update backend datasources.
    - FR-DS-07: The Datasources Service input/output shall be Core Model Classes Statements / SPO Triples.
  + **3.1.2 Data Aggregation (Aggregation Service)**
    - FR-AG-01: The Aggregation Service shall receive raw SPO triple streams from the Augmentation Service (originating from Datasources Service).
    - FR-AG-02: The Aggregation Service shall perform Type inference by identifying subjects with common attributes.
    - FR-AG-03: The Aggregation Service shall perform State inference by identifying subjects of the same type with common attribute values.
    - FR-AG-04: The Aggregation Service shall infer Type hierarchies based on superset/subset relationships of common attributes (more specific types inherit from/are 'after' more generic types).
    - FR-AG-05: The Aggregation Service shall infer State hierarchies based on superset/subset relationships of attribute values or defined sequences.
    - FR-AG-06: The Aggregation Service shall resolve cycles in type hierarchies using state information.
    - FR-AG-07: The Aggregation Service shall provide inferred Type/State hierarchies for use in Alignment Service's Order upper ontology.
    - FR-AG-08: The Aggregation Service shall leverage Machine Learning Classification techniques.
    - FR-AG-09: The Aggregation Service input/output shall be Core Model Classes Statements.
  + **3.1.3 Data Alignment (Alignment Service)**
    - FR-AL-01: The Alignment Service shall receive Aggregation statements (Types, States, Hierarchies) via the Augmentation Service.
    - FR-AL-02: The Alignment Service shall align inferred concepts and relationships from integrated application domains into a shared "Domains" upper ontology.
    - FR-AL-03: The Alignment Service shall materialize mappings between integrated domain concepts/relationships and the Domains upper ontology.
    - FR-AL-04: The Alignment Service shall perform ontology matching to find and map equivalent entities and relationships across source domains.
    - FR-AL-05: The Alignment Service shall align measures (attribute values) into dimensional units based on Type/State hierarchies from Aggregation.
    - FR-AL-06: The Alignment Service shall establish order relationships (e.g., before, greater than, contains) between measures in an "Order" (dimensional) upper ontology.
    - FR-AL-07: The Alignment Service shall materialize measure relationships and map dimensional unit occurrences into the Order upper ontology.
    - FR-AL-08: The Alignment Service shall infer possible links/relationships between resources based on the aligned model (Domains upper ontology).
    - FR-AL-09: The Alignment Service shall infer possible attributes and their values for resources based on the aligned model.
    - FR-AL-10: The Alignment Service shall leverage Machine Learning Clustering techniques.
    - FR-AL-11: The Alignment Service input/output shall be Core Model Classes Statements.
  + **3.1.4 Context/Interaction Activation (Activation Service)**
    - FR-AC-01: The Activation Service shall receive aligned data streams via the Augmentation Service.
    - FR-AC-02: The Activation Service shall discover available Use Cases (Contexts) based on resource types, states, and order relationships.
    - FR-AC-03: The Activation Service shall determine which Roles are played by which resource types (Actors) within specific Contexts, considering their state and order.
    - FR-AC-04: The Activation Service shall allow instantiation of Contexts into Transactions (Interactions).
    - FR-AC-05: The Activation Service shall assign specific Actor Resources to play Roles within an Interaction instance.
    - FR-AC-06: The Activation Service shall infer the business logic (data flow, steps) for Transactions based on the Alignment Service's Domains and Order upper ontologies.
    - FR-AC-07: The Activation Service shall align inferred Contexts, Roles, Interactions, and Actors to an "Activation" upper ontology.
    - FR-AC-08: The Activation upper ontology shall enable the Consumer API Service to expose available Contexts and their states (Interactions).
    - FR-AC-09: The Activation upper ontology shall support querying by the Consumer API Service to build Interaction scenarios based on desired outcomes.
    - FR-AC-10: The Activation Service shall populate possible Actors for Roles based on queries for desired outcomes.
    - FR-AC-11: The Activation Service shall follow the DCI (Data, Context, Interactions) design pattern.
    - FR-AC-12: The Activation Service shall materialize the ordered steps/invocations data flow of transactions declaratively into the model.
    - FR-AC-13: The Activation Service shall leverage Machine Learning Regression techniques.
    - FR-AC-14: The Activation Service input/output shall be Core Model Classes Statements.
  + **3.1.5 Orchestration (Augmentation Service)**
    - FR-AU-01: The Augmentation Service shall orchestrate the workflow between Datasources, Aggregation, Alignment, and Activation services.
    - FR-AU-02: The Augmentation Service shall feed graph model streams from the Datasources Service to the Aggregation Service.
    - FR-AU-03: The Augmentation Service shall manage data flow and control signals between the orchestrated services (Aggregation, Alignment, Activation).
    - FR-AU-04: The Augmentation Service shall provide access to shared Helper Services (Registry, Naming, Index) for the orchestrated services.
    - FR-AU-05: The Augmentation Service shall manage communication (messages, events) with the Datasources Service for data retrieval and synchronization, using Helper Services.
    - FR-AU-06: The Augmentation Service shall manage communication (messages, events) with the Consumer API Service for exposing functionality and receiving requests, using Helper Services.
    - FR-AU-07: The Augmentation Service shall provide the Consumer API Service with Activation stream facilities to instantiate Contexts and perform Interactions.
    - FR-AU-08: The Augmentation Service shall provide updated information (e.g., conversational state) to the Consumer API Service.
  + **3.1.6 Unified Access (Consumer API Service)**
    - FR-CA-01: The Consumer API Service shall expose the inferred Contexts (Use Cases) from the Activation Service via a REST API.
    - FR-CA-02: The Consumer API Service shall allow clients to create new Interactions (instantiate Contexts).
    - FR-CA-03: The Consumer API Service shall allow clients to browse existing/past Interactions.
    - FR-CA-04: The Consumer API Service shall allow clients to update or continue existing Interactions.
    - FR-CA-05: The Consumer API Service shall support querying for possible Interaction scenarios based on a desired outcome.
    - FR-CA-06: The Consumer API Service shall present possible Actors in Roles that fulfill a desired outcome, based on Activation Service results.
    - FR-CA-07: The Consumer API Service shall potentially provide a generic forms-based frontend for rendering Interaction instances and flows (wizards).
    - FR-CA-08: The Consumer API Service shall interact with the Augmentation Service to send requests and receive data/metadata.
    - FR-CA-09: The Consumer API Service shall potentially expose available actions ("gestures", "verbs") based on resource content types, aligning with Domain Driven Design principles.
    - FR-CA-10: The Consumer API Service shall potentially leverage HATEOAS/HAL principles in its REST API.
    - FR-CA-11: The Consumer API Service input/output shall involve Core Model Classes Statements (internally or via Augmentation).
    - FR-CA-12: The Consumer API Service shall leverage Machine Learning Regression techniques (likely related to outcome prediction or scenario suggestion).
  + **3.1.7 Core Model Management (Helper Services)**
    - FR-HS-RG-01: The Registry Service shall act as the central repository for the core graph model.
    - FR-HS-RG-02: The Registry Service shall store, retrieve, and share the results of stream processing from Aggregation, Alignment, and Activation services.
    - FR-HS-RG-03: The Registry Service shall provide storage, possibly as a hierarchical key/value store or based on Topic Maps.
    - FR-HS-RG-04: The Registry Service shall manage provenance information for datasource synchronization.
    - FR-HS-RG-05: The Registry Service shall potentially store embeddings.
    - FR-HS-RG-06: The Registry Service shall provide a SPARQL endpoint for querying the graph model.
    - FR-HS-RG-07: The Registry Service shall support URI-based retrieval and streams/events-based interfaces.
    - FR-HS-NM-01: The Naming Service shall manage the matching and alignment of data to the upper ontologies (Domains, Order, Activation).
    - FR-HS-NM-02: The Naming Service shall potentially use a Sets-based internal representation for the model.
    - FR-HS-NM-03: The Naming Service shall provide an API and functional operations for set processing related to matching and alignment.
    - FR-HS-NM-04: The Naming Service shall potentially use FCA contexts representation and concept lattices for alignment and attribute inference.
    - FR-HS-NM-05: The Naming Service shall resolve links/relationships in contexts.
    - FR-HS-NM-06: The Naming Service shall materialize inferred order relationships.
    - FR-HS-NM-07: The Naming Service shall provide a SPARQL endpoint.
    - FR-HS-NM-08: The Naming Service shall support URI-based retrieval and streams/events-based interfaces.
    - FR-HS-IX-01: The Index Service shall serve as a repository of aligned resources ready for activation.
    - FR-HS-IX-02: The Index Service shall support retrieval of resources via similarity resolution.
    - FR-HS-IX-03: The Index Service shall provide a dialog/conversational state-based interface.
    - FR-HS-IX-04: The Index Service shall resolve possible or actual Contexts/Interactions given resource representations.
    - FR-HS-IX-05: The Index Service shall resolve possible or populated context templates (Actor-in-Role placeholders) for Interactions.
    - FR-HS-IX-06: The Index Service shall, given a resource representation, context, and verb, retrieve the next resource representation in the Activation flow (e.g., a form with placeholders) 1 .
    - FR-HS-IX-07: The Index Service shall support streams/events-based interfaces.
  + **3.1.8 Administration and Management**
    - FR-AM-01: Each service (Datasources, Aggregation, Alignment, Activation, Augmentation, Consumer API, Helpers) shall have an administration/management interface.
    - FR-AM-02: The admin interfaces shall allow users to manage the workflow steps relevant to each service.
    - FR-AM-03: Admin functions shall include: adding datasources, viewing inferred types and their instances, viewing aligned upper ontologies, viewing current contexts/interactions, and browsing available API endpoint definitions.
* **3.2 External Interface Requirements**
  + **3.2.1 User Interfaces**
    - EI-UI-01: A generic forms-based frontend shall be provided for rendering Context Interaction instances.
    - EI-UI-02: The generic frontend shall dynamically generate interfaces based on the metadata of each Context provided by the Activation service via the Consumer API Service.
    - EI-UI-03: The frontend shall support wizard-like interaction flows for multi-step transactions.
    - EI-UI-04: Separate administration/management interfaces shall be provided for each service layer.
    - EI-UI-05: The admin interfaces shall allow visualization of inferred data (types, instances, alignments, contexts) and configuration management.
  + **3.2.2 Hardware Interfaces**
    - None specified in the document.
  + **3.2.3 Software Interfaces**
    - EI-SW-01: The Datasources Service must interface with various backend systems of integrated applications, including databases, existing APIs, and document repositories.
    - EI-SW-02: The framework may utilize Java APIs for interacting with W3C DID functionalities.
    - EI-SW-03: Internal services interface with Helper Services (Registry, Naming, Index) via the Augmentation Service.
  + **3.2.4 Communications Interfaces**
    - EI-CI-01: Internal communication between microservices (Datasources, Augmentation, Aggregation, Alignment, Activation, Consumer API, Helpers) shall occur via serialized Core Model Class Statements, transmitted as messages and events over streams.
    - EI-CI-02: The Consumer API Service shall expose a RESTful web service API.
    - EI-CI-03: The REST API should potentially follow HATEOAS/HAL principles.
    - EI-CI-04: The Registry Service shall provide a SPARQL endpoint.
    - EI-CI-05: The Naming Service shall provide a SPARQL endpoint.
    - EI-CI-06: Helper services shall support streams/events-based interfaces.
    - EI-CI-07: The Index Service shall support a conversational state transfer interface.
* **3.3 Non-Functional Requirements**
  + **3.3.1 Performance Requirements**
    - NFR-PERF-01: The framework must efficiently process reactive streams of data between services. (Specific metrics not provided).
  + **3.3.2 Security Requirements**
    - NFR-SEC-01: Secure communication and data handling must be ensured during interaction and synchronization with backend application datasources. (Specific mechanisms not provided).
    - NFR-SEC-02: If W3C DIDs are used, requirements related to decentralized identity management and secure interaction must be addressed.
  + **3.3.3 Reliability Requirements**
    - NFR-REL-01: Data synchronization between the framework and source applications must be reliable.
    - NFR-REL-02: Communication between distributed microservices must be reliable. (Specific metrics like MTBF not provided).
  + **3.3.4 Availability Requirements**
    - None specified in the document.
  + **3.3.5 Maintainability Requirements**
    - NFR-MAIN-01: The layered microservices architecture should promote modularity and ease of maintenance.
    - NFR-MAIN-02: Administration interfaces should facilitate management and monitoring of each service.
  + **3.3.6 Portability Requirements**
    - NFR-PORT-01: The potential use of W3C DIDs aims to facilitate interoperability between different framework deployments/instances across organizations.
  + **3.3.7 Usability Requirements**
    - NFR-USAB-01: The generated end-user interface (API/frontend) shall be unified, discoverable, and potentially conversational.
    - NFR-USAB-02: Wizard-like interfaces should simplify complex multi-step interactions for end-users.
  + **3.3.8 Compliance/Regulatory Requirements**
    - None specified in the document.
* **3.4 Data Requirements**
  + **3.4.1 Data Representation and Formats**
    - DR-FMT-01: The fundamental internal data representation shall be RDF SPO Triples.
    - DR-FMT-02: Data structures shall adhere to the defined Core Model Classes: URI, URIOccurrence, Resource, Type, State, Kind, Statement.
    - DR-FMT-03: Statements themselves shall be reified (treated as URIOccurrences) to capture context.
    - DR-FMT-04: URIs shall be used as identifiers for all graph nodes and arcs.
    - DR-FMT-05: Each URI shall have a unique prime number ID assigned at creation time (for potential FCA use).
    - DR-FMT-06: Core model classes must be serializable, potentially using JSON.
    - DR-FMT-07: Dimensional data shall be represented using Measures (Dimension, Unit, Value).
    - DR-FMT-08: Consideration should be given to representing the model using Sets for inference algorithms.
    - DR-FMT-09: Consideration should be given to FCA concept lattice representations.
    - DR-FMT-10: Consideration should be given to TMRM/TMDM-like representations and embeddings.
    - DR-FMT-11: Consideration should be given to XML representations and transformations.
  + **3.4.2 Data Persistence and Storage**
    - DR-STO-01: The Registry Service shall provide persistent storage for the core graph model, provenance data, and embeddings.
    - DR-STO-02: The Naming Service shall provide persistent storage for upper ontologies, alignment mappings, and potentially FCA contexts/lattices.
    - DR-STO-03: The Index Service shall provide persistent storage for aligned resources ready for activation.
  + **3.4.3 Data Synchronization and Provenance**
    - DR-SYNC-01: The framework must synchronize relevant data changes originating from Interaction executions back to the source application systems via the Datasources Service.
    - DR-SYNC-02: The framework must track the provenance of data ingested from and synchronized back to source systems, managed by the Datasources Service and stored in the Registry Service.

**4. Other Requirements / Features to Explore**

* OR-WEB3-01: Explore the use of Web3 technologies, specifically W3C Decentralized Identifiers (DIDs), for managing URIs and potentially smart contracts for interaction semantics, to facilitate inter-organizational deployment.
* OR-TECH-01: Investigate the use of related standards and technologies: HyTime/XML, ISO Topic Maps (TMRM/TMDM), ISO 15926.
* OR-TECH-02: Investigate dynamic XML/XSLT transformations for resource representations and views.
* OR-SEMIO-01: Incorporate a Semiotic Layer considering Objects/Signs/Concepts occurrences in Contexts.
* OR-ADDR-01: Implement semantically annotated hypermedia resource/object addressing and linking.
* OR-FUNC-01: Implement functional monadic parsers and traversals ("getters"/"setters") for manipulating core model representations.
* OR-ML-01: Explore association rule mining in addition to specified ML techniques.
* OR-ACTV-01: Define resource capabilities based on Content Type (e.g., Buy-able, Identify-able, Locatable) for Activation.

**5. Appendices**

* Appendix A: Core Model Class Diagram (Diagram not included in source text).
* Appendix B: Sets Representation Diagram (Diagram not included in source text).