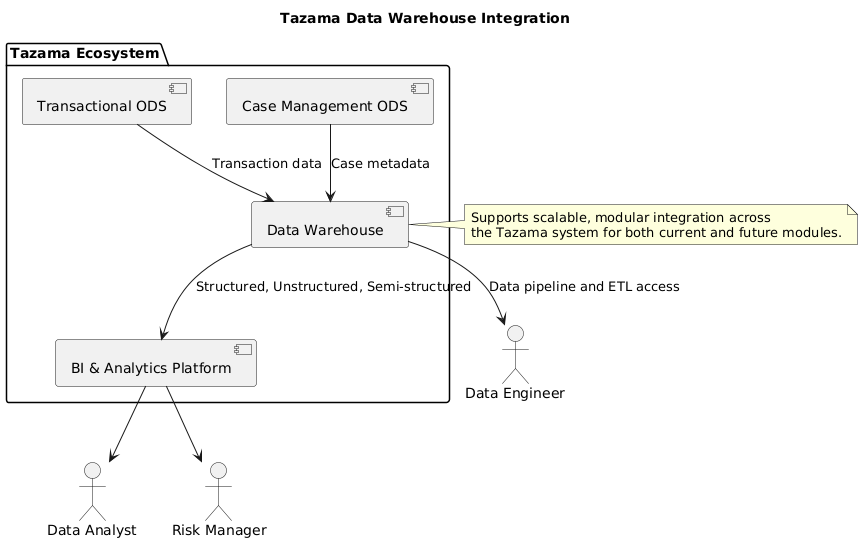
**BUSINESS REQUIREMENT SPECIFICATIONS DOCUMENT**

**DATAWARE HOUSE INTEGRATION WITH TAZAMA**

| **#** | **Requirement** |
| --- | --- |
| 1. **General Requirements** | |
| A.1 | The data warehouse platform should be capable of serving as a centralized platform to consolidate both structured, semi-structured and unstructured financial and non-financial data. |
| A.2 | The data warehouse platform must be capable of integrating with various modules within the Tazama landscape. For this project, the scope of integration will be limited to the Transactional ODS and the Case Management ODS and Jupyter Labswith the flexibility for future expansions as new modules are introduced *(please refer to the diagram below for detailed explanation of this requirement)* |
| A.3 | The data warehouse stream will provide query-able historical data for investigators to enhance case triaging and evidence collection. |
| A.4 | The data warehouse component of the project shall include a well-defined integration with the Jupyter Notebook environment, enabling authorized users to directly access curated and historical datasets for analytical purposes. |
| 1. **Evaluation Framework for Choosing the Correct Data Warehouse Platform** | |
| **B.1** | ***Native Capabilities and Features of Data Warehouse Platform*** |
|  | ***B.1.1. Scalability and Performance:***  *B.1.1.1.* **Concurrency Handling:**  This refers to the system's ability to efficiently manage and process multiple simultaneous queries or transactions without a degradation in performance.  *B.1.1.2.* **OLAP Performance:**  This refers to the system's ability to execute analytical queries, which are typically read-heavy, with high efficiency and minimal latency, ensuring optimal performance during data analysis operations.  *B.1.1.3.* **Real-time Ingestion:**  This refers to the system's capability to efficiently handle and process data as it is ingested (loaded) in real-time, ensuring that new data is immediately available for querying and analysis.  ***B1.2. Cloud Readiness and Useability:***  *B.1.2.1.* **Cloud/Hybrid Friendly:**  This refers to the system's capability to operate efficiently in cloud-based or hybrid environments, which combines both on-premises and cloud infrastructure, ensuring seamless integration and scalability across different deployment models.  *B.1.2.2.* **Setup Time:**  This refers to the amount of time required to deploy and configure the system, ensuring it is fully operational and ready for use.  *B.1.2.3.* **Ease of Use:**  This refers to the system's user-friendliness for developers and analysts, focusing on its accessibility, intuitive design, and ease of interaction for both technical and non-technical users.  *B.1.2.4.* **Operational Complexity:**  This refers to the level of effort required to manage and maintain the system on a day-to-day basis, including tasks related to monitoring, troubleshooting, and ensuring continuous system performance.  *B.1.2.5.* **Jupyter Notebook Support:**  This refers to the system’s ability to integrate seamlessly with Jupyter Notebooks, a popular tool for interactive data analysis and visualizations. This integration enhances usability by enabling real-time data exploration, visualization, and model experimentation within the system, fostering collaboration and knowledge sharing.  ***B.1.3. Data Integration and Flexibility:***  *B.1.3.1.* **SQL Support**  This refers to the extent and type of SQL support provided by the system, including compatibility with standard SQL queries, custom extensions, and the ability to execute complex analytical operations using SQL-based languages.  *B.1.3.2.* **Integration with BI Tools**  This refers to the system's capability to seamlessly connect and integrate with Business Intelligence (BI) tools such as Tableau, Power BI, and Superset, enabling effective data visualization, reporting, and analysis.  *B.1.3.3.* **Federated Queries**  This refers to the system's ability to perform queries across multiple, heterogeneous data sources simultaneously e.g. PostgreSQL, allowing for seamless data integration and analysis without the need for data consolidation into a single repository.  *B.1.3.4.* **Storage Format**  This refers to the method in which data is physically stored within the system, including the use of specific data formats (e.g., columnar, row-based) and storage structures designed to optimize data retrieval, storage efficiency, and query performance.  *B.1.3.5.* **NATS Support:**  This refers to the system's ability to integrate with NATS, a high-performance messaging system.  *B.1.3.6.* **No SQL Support:**  This feature indicates the system's capability to integrate with and support NoSQL databases ArangoDB  *B.1.3.7.* **Compatibility with Arango DB:**  This refers to the system’s ability to interface with ArangoDB, a multi-model database that supports graph, document, and key-value data models.  *B.1.3.8.* **Vector Mapping:**  This refers to the system's preferred capability to store data in a columnar format, using vectors to enhance the speed and efficiency of query execution.  ***B1.4. Data Management and Governance:***  *B.1.4.1.* **ACID Compliance:**  This refers to the system's adherence to the ACID properties—Atomicity, Consistency, Isolation, and Durability—ensuring reliable transaction processing and data integrity even in the event of system failures or concurrent access.  *B.1.4.2.* **Schema Evolution:**  This refers to the system's ability to accommodate changes in the data structure, such as adding or removing columns, without disrupting ongoing operations, ensuring flexibility in managing evolving data models over time.  *B.1.4.3.* **Security Features:**  This refers to the system's implementation of security measures such as authentication, encryption, and role-based access control (RBAC), designed to protect sensitive data and ensure secure access to system resources based on user roles and permissions.  *B.1.4.4.* **Data Governance Tools:**  This refers to the system's support for data governance functionalities, including metadata tracking, data lineage, and access control, often facilitated by tools such as Apache Atlas or Ranger, to ensure compliance, transparency, and secure data management across the system.  ***B.1.5. Monitoring, Metrics and Architecture:***  *B.1.5.1.* **Monitoring and Alerting:**  This refers to the system's capabilities for tracking cluster health and performance, including real-time monitoring, alert generation for potential issues, and proactive notifications to ensure optimal system operation and prompt issue resolution.  *B.1.5.2.* **Lakehouse Ready:**  This refers to the system's capability to support the Lakehouse architecture, which unifies the features of data lakes and data warehouses, enabling seamless integration of structured and unstructured data for analytics and real-time processing. |
| **B.2** | ***Alignment with Tazama Design Principles*** |
|  | ***B.2.1. Open Source First:***  This principle emphasizes the preference for using open-source software to build Tazama, ensuring flexibility, transparency, and cost-effectiveness.  ***B.2.2. Do not Reinvent a Serviceable Wheel:***  Tazama should focus on leveraging existing, proven solutions rather than developing custom software for known problems.  ***B.2.3. High Performance at Low Total Cost of Ownership:***  Tazama shall strive to deliver exceptional performance while minimizing costs across the entire system lifecycle, including setup, maintenance, and operational costs.  ***B.2.4. Keep it Simple and Keep it Inclusive:***  The system should be easy to adopt and use, regardless of the user's technical expertise.  ***B.2.5. Design for Failure:***  Tazama shall be designed to be resilient in the face of failures.  ***B.2.6. Appropriate Best Practices:***  Tazama shall follow industry best practices for software development, system architecture, and operational management. |
| **(C ) Implementation Requirements for the Selected Data Warehouse Tool** | |
| C.1 | The Data Warehouse must support native or library-based integration with JupyterLab for Python-based querying, model prototyping, and visualization. |
| C.2 | The Data Warehouse must allow querying across heterogeneous sources like ArangoDB without requiring ETL. |
| C.3 | The Data Warehouse platform shall support integration with NATS. |
| C.4 | The Data Warehouse shall provide integration support for common NoSQL databases |
| C.5 | The selected platform shall be capable of interfacing with ArangoDB, a multi-model database supporting graph, document, and key-value data. |
| C.6 | The Data Warehouse implementation shall enforce Role-Based Access Control (RBAC) to manage access to datasets, views, pipelines, and administrative functions. |

A.2 - Data Warehouse Integration:

This diagram illustrates the integration architecture of the Data Warehouse within the Tazama Ecosystem, showcasing how it interacts with operational data sources, downstream analytics platforms, and key system users.

### Key Components and Flows

#### 1. Transactional ODS

* Provides transaction-level data captured from the Tazama Transaction Monitoring System.

2. Case Management ODS

* Supplies case metadata from the Case Investigation Management System (CIMS),

3. Data Warehouse

* Serves as the central data repository, consolidating both transactional and case-level information.
* Capable of storing and processing structured, unstructured, and semi-structured data.
* Acts as the integration hub for downstream services and modules.

#### 4. BI & Analytics Platform

* Directly interfaces with the Data Warehouse to provide:
  + Visual dashboards
  + Reporting tools
  + Advanced analytics interfaces (e.g., JupyterLab)
* Consumes curated data to support data analysts and risk managers in decision-making.

#### 5. Data Engineer Access

* Engineers interact with the Data Warehouse via ETL pipelines and data access tools.
* Responsible for designing and managing ingestion flows, data modeling, and integration with additional systems.