Project Requirements Document: ORC AI - Autonomous Workflow Orchestration

I. Introduction

ORC AI is envisioned as a pioneering AI agent designed to autonomously orchestrate complex workflows within existing enterprise systems, specifically targeting Apache Airflow and Autosys. [User Initial Query] This project aims to transcend traditional, static workflow automation by embedding intelligent, dynamic decision-making directly into the orchestration layer. 1 ORC AI will function as a "conductor," coordinating diverse AI and non-AI services, leveraging Large Language Models (LLMs) and a robust tool-use framework to reason, plan, and execute tasks with minimal human intervention. 2

II. Project Goals and Objectives

The overarching goal of the ORC AI project is to transform enterprise operations by delivering unprecedented levels of efficiency, agility, and accuracy through autonomous workflow orchestration. 4

SMART Objectives:

- **Increase Operational Efficiency:** Automate complex, repetitive tasks and optimize resource utilization to significantly reduce manual effort and human errors. 4
- **Enhance Decision-Making:** Enable real-time data analysis, pattern identification, and insight generation, facilitating quicker and more informed responses to operational changes. 4
- **Improve Accuracy:** Reduce the likelihood of errors through consistent execution and autonomous identification of discrepancies, with mechanisms to trigger human review for critical issues. 4
- **Increase Agility:** Dynamically adapt workflows to real-time data and unforeseen conditions, allowing for rapid adjustments to new parameters or priorities. 3
- **Achieve Scalability:** Design the system to handle large volumes of work without compromising quality or efficiency, intelligently distributing tasks and optimizing resource allocation. 4
- **Reduce Costs:** Expand the scope of process automation, enable more efficient resource allocation, and minimize errors, leading to substantial productivity uplift and cost savings. 4

III. Scope

This section defines the boundaries of the ORC AI project, outlining what will be included and what will be deferred for future phases.

In-Scope:

- **Core AI Agent Development:** Development of a foundational AI agent architecture capable of reasoning, planning, and dynamic tool use. **3**
- **Apache Airflow Integration:** Seamless integration with Apache Airflow for workflow orchestration, including API-driven control (triggering DAGs, monitoring status, modifying configurations). **5**
- **Knowledge Graph Integration:** Implementation of a Knowledge Graph (Neo4j) to provide contextual understanding, manage workflow dependencies, and enable explainable decision-making. **10**
- **Real-time Monitoring & Alerting:** Development of comprehensive monitoring and observability capabilities using Prometheus, Grafana, and OpenTelemetry for real-time insights and anomaly detection. **12**
- **Fault Tolerance & Recovery:** Implementation of robust mechanisms for fault tolerance and stateful recovery for both ORC AI's internal state and the workflows it manages. **15**
- **Security & Access Control:** Establishment of a robust security framework, including authentication (Keycloak) and fine-grained authorization, with consideration for external policy engines. **17**
- **MLOps Best Practices:** Adherence to MLOps principles for continuous integration, continuous deployment, versioning, and automated testing of AI models and workflows. **19**
- **Synthetic Data Generation:** Capability to generate synthetic data for development, testing, and initial model training. **20**

Out-of-Scope (for initial prototype phase):

- **Full Autosys Integration:** While Autosys is a target, the initial prototype will focus exclusively on Apache Airflow integration.
- **Advanced User Interface for Workflow Creation:** The prototype's UI will primarily focus on real-time monitoring and operational insights (e.g., Streamlit dashboards), not dynamic workflow creation by end-users.
- **Complex Multi-Agent Collaboration:** Initial multi-agent capabilities will be limited to foundational interactions, with advanced collaborative patterns deferred.
- **Direct Real-time Data Processing within Airflow:** Airflow will orchestrate external streaming tools (e.g., Kafka, Flink) for real-time data processing rather than performing it directly. **22**

IV. Key Features and Capabilities

ORC AI will deliver a suite of intelligent features designed to automate and optimize enterprise workflows.

- **Autonomous Workflow Orchestration:**

- **Dynamic Scheduling:** AI-driven adjustment of workflow timelines based on real-time conditions, leveraging predictive analytics to forecast demand patterns and optimize resource allocation.
- **Adaptive Control:** Ability to learn from past behavior, predict future states, and dynamically adapt to changing conditions, moving beyond static, pre-programmed logic.
- **Self-Healing Workflows:** Automatic detection and correction of issues, including rerouting data, restarting failed components, or adjusting configurations, often without human intervention.
- **Integration with Airflow/Autosys:**
- **API-driven Control:** Programmatic triggering of DAGs, monitoring their status, retrieving logs, and potentially modifying configurations via Airflow's REST API. **5**
- **Custom Operators/Plugins:** Development of custom Airflow operators or plugins to extend capabilities and encapsulate complex interactions with external systems. **23**
- **Event-driven Triggers:** Ability to react to changes within Airflow/Autosys (e.g., task failures, resource bottlenecks) via event-driven mechanisms, utilizing Apache Kafka as an event broker. **22**
- **Knowledge Graph Integration:**
- **Contextual Retrieval:** Ability to explore the graph structure to find precise nodes and relationships (e.g., identifying data pipelines feeding into a specific Airflow DAG). **10**
- **Explainable Reasoning:** Providing precise tracking of how an answer or decision is derived by linking it to specific source information, enhancing trust and auditability for autonomous actions. **10**
- **Dynamic Schema Inference:** Leveraging LLMs to dynamically infer and consolidate graph schema from input text, eliminating the need for rigid, predefined schemas and adapting to evolving datasets. **27**
- **AI Agent Core:**
- **LLM Integration:** Utilizing Large Language Models for natural language understanding, generation, complex reasoning, problem-solving, and selecting optimal courses of action. **3**
- **Tool Use:** Programmatic interaction with external systems (e.g., Airflow, Autosys, databases, APIs) via a robust set of "tools" based on the LLM's decisions. **3**
- **Reasoning Engine:** The core cognitive module responsible for processing information, making decisions, planning actions, and continuously improving strategies based on past experiences and feedback. **28**
- **Monitoring & Observability:**
- **Real-time Insights:** Comprehensive tracking of system performance, resource utilization, and workflow health using Prometheus, Grafana, and OpenTelemetry. **12**

- **Anomaly Detection:** Proactive identification of issues such as data drift, concept drift, or performance bottlenecks within workflows. **30**
- **Alerting:** Automated notification of critical events or decisions requiring human review via channels like Slack. **32**
- **Fault Tolerance & Recovery:**
- **Stateful Workflow Recovery:** Preservation of ORC AI's internal state and seamless recovery of complex, long-running processes, even during failures.
- **Automated Retries:** Automatic re-execution of failed operations without manual intervention for transient errors. **16**
- **Idempotent Pipelines:** Design of tasks and DAGs to be safely re-run multiple times without causing unintended side effects, crucial for recovery. **34**
- **Security & Access Control:**
- **Authentication:** Secure user authentication via Keycloak, supporting Single Sign-On (SSO) and Multi-Factor Authentication (MFA). **17**
- **Fine-Grained Authorization:** Granular control over resources and actions using Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and potentially Relationship-Based Access Control (ReBAC via external policy engines). **38**
- **Secure Communication:** All communication within the system and with integrated platforms will be secured using HTTPS with strong SSL/TLS certificates. **36**
- **Scalability & Performance Optimization:**
- **Dynamic Resource Allocation:** Intelligent allocation of computational resources (CPU, GPU) based on real-time demand and task priorities.
- **Containerization & Orchestration:** Deployment using Docker for consistent environments and Kubernetes for scalable workload orchestration. **30**
- **Cost Optimization:** Implementation of strategies like utilizing spot instances, improving AI model efficiency, and automating resource management to reduce operational costs. **41**

V. Technical Requirements / Technology Stack

The ORC AI system will be built upon a robust and modern technology stack, primarily leveraging open-source tools.

- **Workflow Orchestration:** Apache Airflow **5**
- **AI Agent Core:**
- **Large Language Models (LLMs):** Integration with leading LLM providers (e.g., OpenAI, Google, Anthropic, MistralAI) for cognitive capabilities. **43**
- **Custom Logic/Rule Engine:** Python-based rule engine (e.g., durable-rules) for defining and managing business rules and complex decision-making logic. **44**
- **Knowledge Graph:**

- **Database:** Neo4j (Property Graph Database) for storing and managing the knowledge graph. **11**
- **Python Package:** Neo4j GraphRAG Python package for knowledge graph construction, enrichment, and Retrieval Augmented Generation (GraphRAG). **43**
- **Data Streaming/Messaging:** Apache Kafka for real-time event streaming and decoupled communication between system components. **24**
- **Monitoring & Observability:**
- **Metrics Collection:** Prometheus for collecting and storing time-series metrics. **12**
- **Visualization:** Grafana for creating customizable dashboards and alerts. **12**
- **Telemetry Instrumentation:** OpenTelemetry for instrumenting applications to collect traces, metrics, and logs. **12**
- **Security:**
- **Identity & Access Management (IAM):** Keycloak for authentication, user management, and basic authorization. **17**
- **External Policy Engine (Optional for prototype, recommended for production):** Open Policy Agent (OPA) or Permit.io for advanced fine-grained authorization (ABAC, ReBAC). **18**
- **Containerization & Orchestration:**
- **Containerization:** Docker for packaging and ensuring consistent deployments of ORC AI components. **30**
- **Orchestration:** Kubernetes (for production-grade scalability and resource management). **30**
- **Synthetic Data Generation (for testing):** The Synthetic Data Vault (SDV) Python library or custom Python scripts for generating artificial datasets. **20**
- **Primary Programming Language:** Python **35**
- **UI/Dashboard (for monitoring):** Streamlit for real-time monitoring dashboards. **15**

VI. Non-Functional Requirements

These requirements define the quality attributes and constraints under which the ORC AI system must operate.

- **Performance:**
- **Task Duration:** Aim for a reduction in average task execution time by at least 20% within a quarter, targeting specific bottlenecks. **47**
- **API Response Time:** Critical API calls (e.g., for real-time agent decisions) should have a response time under 300ms. **49**
- **Event Processing Latency:** Minimum event processing latency of 50 milliseconds for critical real-time events. **16**
- **Event Broker Throughput:** The event broker (Kafka) should sustain throughputs of 5,000 to 15,000 messages per second. **16**

- **Business Rules Processing Rate:** The rule engine should process rules at a rate of 2,500-6,000 rules per second. **16**
- **Scalability:**
- **Dynamic Resource Scaling:** The system must dynamically scale compute resources (CPU, GPU) based on workload demand and priorities.
- **Airflow Worker Scaling:** Support horizontal scaling of Airflow workers using CeleryExecutor or KubernetesExecutor to handle increasing task loads. **48**
- **Data Volume Handling:** Efficiently process and manage large datasets (terabytes to petabytes) for knowledge graph construction and AI model training. **50**
- **Reliability & Availability:**
- **System Availability:** Target 99.95% availability for the ORC AI system and managed workflows, even during partial system outages. **16**
- **Mean Time to Repair (MTTR):** Reduce the average time taken to resolve incidents and restore full functionality.
- **Mean Time Between Failures (MTBF):** Maximize the average time between repairable failures to indicate high system reliability. **51**
- **Fault Tolerance:** The system must automatically recover from transient failures and system crashes without manual intervention. **15**
- **Security:**
- **Data Encryption:** Implement robust encryption for all data, both in transit and at rest, to protect AI models and sensitive data. **30**
- **Compliance:** Adhere to relevant data privacy regulations (e.g., GDPR, CCPA) and enterprise security policies. **30**
- **Auditability:** Provide comprehensive logging and auditing capabilities to track AI agent decisions and actions for compliance and debugging. **10**
- **Maintainability:**
- **Modular Design:** Employ a modular design for DAGs and AI components to enhance readability, reusability, and ease of maintenance. **35**
- **Documentation:** Maintain clear, consistent, and up-to-date documentation for code, configurations, and workflows. **19**
- **Automated Processes:** Utilize automated testing and deployment via CI/CD pipelines to ensure rapid and reliable updates. **19**
- **Usability:**
- **Monitoring Dashboards:** Provide intuitive monitoring dashboards for real-time insights into system performance and workflow status. **12**
- **Alerting:** Implement clear and actionable alerting mechanisms for critical events and anomalies. **32**
- **Human-in-the-Loop (HITL):** Define explicit human intervention points and feedback mechanisms for oversight, corrections, and continuous learning. **3**

- **Cost Efficiency:**
- **Cloud Cost Optimization:** Minimize cloud infrastructure costs through optimized resource allocation, rightsizing, and strategic use of cost-saving strategies (e.g., spot instances, model efficiency). **41**
- **FinOps Practices:** Implement proactive monitoring and optimization of AI spending through a Cloud FinOps framework. **41**

VII. Stakeholders

Key individuals and groups involved in the ORC AI project and their primary interests:

- **Product Owners/Business Analysts:** Responsible for defining business requirements, prioritizing features, and validating that the solution meets business needs. **53**
- **AI/ML Engineers:** Design, develop, train, and fine-tune the AI models and agents, ensuring their performance and accuracy. **19**
- **Data Engineers:** Design and implement data pipelines, manage data quality, and maintain the underlying data infrastructure for the knowledge graph and AI models. **53**
- **DevOps/Platform Engineers:** Manage the infrastructure, deployment pipelines, monitoring systems, and ensure the overall operational stability and scalability of ORC AI. **19**
- **Security Team:** Ensure the system adheres to enterprise security policies, data governance regulations, and protects sensitive information. **30**
- **End-Users/Operations Teams:** The primary beneficiaries of ORC AI; they will utilize the system for workflow orchestration and provide crucial feedback for continuous improvement. **3**
- **Executives:** Interested in high-level summaries of strategic goals, key performance indicators (KPIs), and the overall return on investment (ROI) of the project. **53**

VIII. Development Approach / Methodology

The ORC AI project will adopt an agile and iterative development methodology, deeply integrated with MLOps best practices to ensure rapid delivery, reliability, and continuous improvement.

- **Agile & Iterative Development:** The project will follow an iterative development process where objectives are clearly defined, tasks are broken into smaller, manageable segments, and outputs are thoroughly reviewed for accuracy, logic, and adherence to standards. This "Build, Review, Improve—Repeat" cycle is essential for adaptive AI systems. **9**
- **MLOps Best Practices:**

- **Automation:** Automation is central, transforming manual tasks into consistent, repeatable processes. This includes building CI/CD pipelines that automate model training, validation, testing, and deployment. **19**
- **Versioning:** Comprehensive version control will be applied not only to code but also to datasets, hyperparameters, model weights, and experiment results, ensuring reproducibility and traceability. **19**
- **Testing:** Rigorous testing will encompass validating code logic, data integrity, and model outputs, as well as regression testing, drift detection, and fairness audits. **19**
- **Monitoring:** Continuous monitoring of AI model performance and system health in production will be implemented to detect issues like data drift or concept drift. **19**
- **Reproducibility:** CI/CD pipelines will ensure that models can be rebuilt and retrained exactly the same way, codifying environments and configurations. **54**
- **Data-Driven Refinement:** The system will continuously monitor and analyze execution data to fine-tune resource allocation, adjust automation rules, and enhance performance, relying on metrics and KPIs to assess progress. **55**

IX. Success Metrics (Key Performance Indicators - KPIs)

The success of the ORC AI project will be measured against the following Key Performance Indicators:

- **Resource Utilization Improvement:** Quantifiable increase in the efficiency of computational resource allocation (e.g., CPU, GPU, memory).
- Target: +26% improvement in resource utilization with context-aware routing. **16**
- **Failure Resolution Time (MTTR):** Reduction in the average time taken to resolve incidents and restore full system functionality.
- Target: -40% reduction in failure resolution time. [User Initial Query]
- **Scheduling Accuracy:** How closely predicted labor needs (or resource requirements for tasks) match actual requirements.
- Target: 95% scheduling accuracy. [User Initial Query]
- **Process Agility Improvement:** Enhanced responsiveness of workflows to real-time changes and dynamic conditions.
- Target: +35% improvement in process agility with event-driven architecture. **16**
- **Time-to-Market Reduction:** Accelerated deployment of new business capabilities and AI solutions.
- Target: -40% reduction in time-to-market with event-driven architecture. **16**
- **System Availability:** Uptime of the ORC AI system and the workflows it orchestrates.
- Target: 99.95% system availability with event-driven architecture. **16**

X. Assumptions and Constraints

Assumptions:

- **API Access:** Necessary APIs for Airflow and Autosys (for future integration) will be available and provide sufficient programmatic control.
- **Cloud Infrastructure:** Access to a scalable cloud infrastructure and relevant cloud services (e.g., managed Kubernetes, object storage, managed databases) will be provided.
- **Data Availability:** Sufficient historical data will be available for training AI models, or realistic synthetic data can be generated effectively for prototyping.
- **Stakeholder Engagement:** Active involvement and commitment from all identified stakeholders for agile development and continuous feedback.

Constraints:

- **Initial Focus:** The initial prototype development will primarily focus on integration with Apache Airflow, with Autosys integration planned for later phases.
- **Budget & Resources:** Project development and operational costs will be subject to defined budget and resource availability.
- **Security & Compliance:** All development and deployment must adhere strictly to existing enterprise security policies and data governance regulations.
- **LLM Credit Usage:** LLM usage will be managed to be credit-conscious, optimizing prompts and model selection to minimize costs. **41**