Software Design Document

AI Agent Module

For Prowiz ERP System

Version 1.0

Document ID: SDD-AI-001

Nithin Vadekkapat
Director-AI

Dhvani Analytic Intelligence Pvt. Ltd.
AI Solutions Division

July 3, 2025

Document Revision History

Version	**Date**	**Author**	**Description**
1.0	July 3, 2025	Nithin Vadekkapat	Initial version

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of the AI Agent Module is to provide a set of specialized agents that can autonomously perform tasks related to process design and documentation. The module aims to streamline workflows, improve accuracy, and enhance collaboration across teams. Key functionalities include:

- **Preprocesses the image**: Enhances image quality, adjusts contrast, and optimizes resolution for analysis
- **Detects symbols and equipment**: Uses computer vision to identify pumps, valves, vessels, and instrumentation
- **Extracts text labels**: Applies OCR to extract equipment tags, specifications, and process parameters
- **Maps spatial relationships**: Analyzes the positioning and connections between identified components
- **Builds process graph**: Constructs a NetworkX graph representing the complete process topology
- **Generates structured outputs**: document presents the detailed software design for the AI Agent Module within the Prowiz ERP system. It describes the architecture, interfaces, and operational workflows for each specialized agent.

1.2 Scope

The scope of this document covers:

- Functional responsibilities of each AI agent (Compliance, Document Generation, Diagram Parsing, Email, ERP-Chat).

- Scope and boundaries for each agent (what each agent handles vs. out of scope).
- Example use cases demonstrating invocation and outputs for each agent.

Chapter 2

Overall Architecture

2.1 Module Overview

The AI Agent Module exposes a REST/GRPC API, orchestrates incoming requests via a central orchestrator, and dispatches tasks to individual agent services. It leverages a shared LLM runtime, embedding service with Vector DB, a project-scoped Knowledge Base, a message queue for resilience, and observability via Prometheus and Loki.

2.2 System Flow Architecture

The following diagram illustrates the complete system flow and architecture of the AI Agent Module:

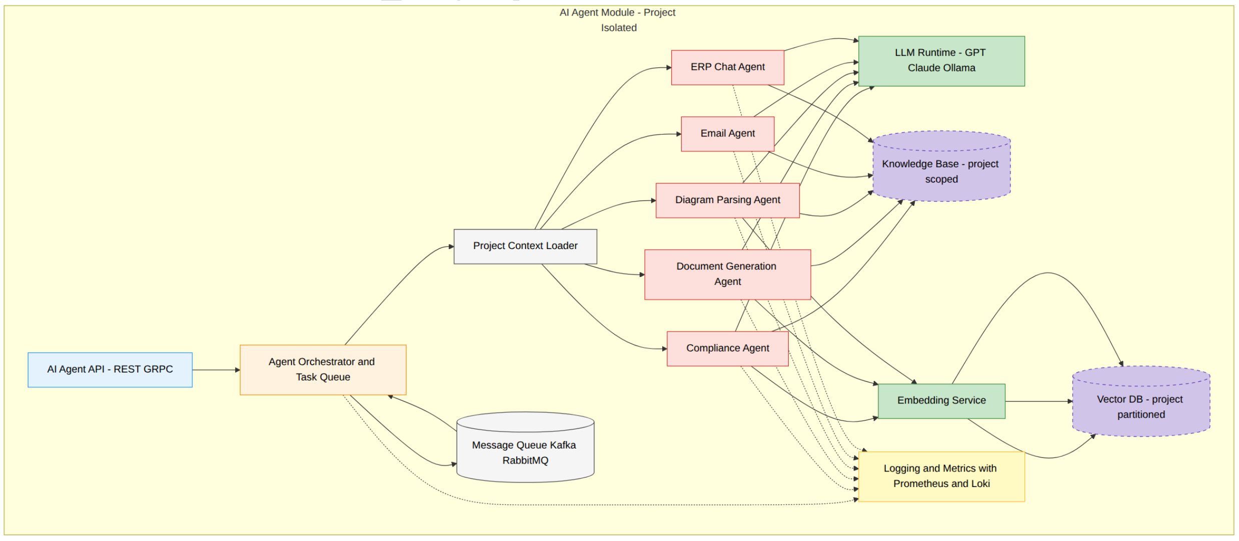
AI Agent Module - Project Isolated
ERP Chat Agent
LLM Runtime - GPT Claude Ollama
Email Agent
Knowledge Base - project scoped
Diagram Parsing Agent
Project Context Loader
Document Generation Agent
Compliance Agent
AI Agent API - REST GRPC
Agent Orchestrator and Task Queue
Embedding Service
Vector DB - project partitioned
Message Queue Kafka RabbitMQ
Logging and Metrics with Prometheus and Loki

Figure 2.1: AI Agent Module System Flow Architecture

2.2.1 System Components Overview

The AI Agent Module consists of the following key components and their functionalities:

API Gateway & Orchestrator

- Exposes REST/GRPC API endpoints for external system integration
- Routes incoming requests to appropriate specialized agents
- Manages request orchestration and workflow coordination

Specialized AI Agents

- **Compliance Agent**: Validates parameters against specifications and business rules
- **Document Generation Agent**: Creates templated documents with data merging and narratives
- **Diagram Parsing Agent**: Extracts structured data from P&ID and CAD diagrams
- **Email Agent**: Parses inbound emails for project instructions and attachments
- **ERP-Chat Agent**: Converts natural-language commands to ERP API calls

Shared Infrastructure Services

- **LLM Runtime**: GPU-hosted language model processing on Kubernetes
- **Embedding Service & Vector DB**: Semantic search using Faiss/Qdrant with project partitioning
- **Knowledge Base**: MongoDB collections with project-scoped data storage
- **Message Queue**: Kafka-based request handling and retry mechanisms
- **Observability**: Prometheus metrics and Loki logging for system monitoring

Chapter 3

Compliance Agent

3.1 Function

The Compliance Agent operates within the AI Layer of the system architecture and performs AI-powered validation of structured data received from upstream preprocessing layers. It conducts knowledge-based compliance checking and rule-based validation to produce detailed compliance reports.

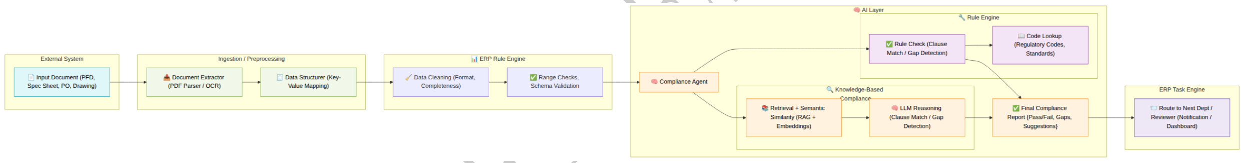
External System
Input Document (PFD, Spec Sheet, PO, Drawing)
Ingestion / Preprocessing
Document Extractor (PDF Parser / OCR)
Data Structurer (Key-Value Mapping)
ERP Rule Engine
Data Cleaning (Format, Completeness)
Range Checks, Schema Validation
AI Layer
Compliance Agent
Rule Engine
Rule Check (Clause Match / Gap Detection)
Code Lookup (Regulatory Codes, Standards)
Knowledge-Based Compliance
Retrieval + Semantic Similarity (RAG + Embeddings)
LLM Reasoning (Clause Match / Gap Detection)
Final Compliance Report {Pass/Fail, Gaps, Suggestions}
ERP Task Engine
Route to Next Dept / Reviewer (Notification / Dashboard)

Figure 3.1: Compliance Agent Workflow and Architecture

3.2 Scope and Boundaries

3.2.1 AI Layer Scope (In-Scope)

The Compliance Agent is responsible only for components within the **AI Layer** subgraph:

- **Compliance Agent Core**: Central orchestration of AI-powered compliance validation
- **Knowledge-Based Compliance**:
  - RAG (Retrieval + Semantic Similarity) with embeddings
  - LLM Reasoning for clause matching and gap detection
- **Rule Engine Integration**:
  - Rule-based compliance checks and clause matching

  - Code lookup against regulatory codes and standards databases
- **Structured Output Generation**: Final compliance report generation with pass/fail status, gaps, and suggestions

3.2.2 External Dependencies (Out-of-Scope)

The following components are **outside** the AI Layer and handled by other subsystems:

- **External System**: Input document handling (PFDs, spec sheets, purchase orders, drawings)
- **Ingestion/Preprocessing Layer**:
  - Document extraction (PDF parser/OCR)
  - Data structuring (key-value mapping)
- **ERP Rule Engine Layer**:
  - Data cleaning (format, completeness)
  - Data validation (range checks, schema validation)
- **ERP Task Engine**: Routing to next department/reviewer and notification systems

3.2.3 Integration Boundaries

- **Input**: Receives structured, validated data from the ERP Rule Engine layer
- **Output**: Delivers compliance reports to the ERP Task Engine for routing
- **Dependencies**: Accesses shared knowledge bases and regulatory code databases
- **Limitations**: Cannot modify upstream data processing or downstream workflow routing

3.3 Architecture Components

The Compliance Agent operates exclusively within the AI Layer and consists of the following components:

3.3.1 AI Layer Components (Compliance Agent Scope)

- **Compliance Agent Core**: Central coordinator that receives structured data from upstream layers and orchestrates AI-powered validation processes
- **Knowledge-Based Compliance Module**:
  - **RAG System**: Retrieval and semantic similarity matching using embeddings

  - **LLM Reasoning Engine**: Advanced reasoning for clause matching and gap detection
- **Rule Engine Interface**:
  - **Rule Check Processor**: Executes deterministic compliance checks
  - **Code Lookup Service**: Interfaces with regulatory codes and standards databases
- **Report Generator**: Produces structured compliance reports with pass/fail status, identified gaps, and improvement suggestions

3.3.2 External Layer Dependencies (Outside AI Agent Scope)

These components are managed by other subsystems but provide inputs/outputs to the AI Layer:

- **Upstream Dependencies**:
  - Document Extractor (Ingestion Layer)
  - Data Structurer (Ingestion Layer)
  - Data Cleaning (ERP Rule Engine Layer)
  - Data Validation (ERP Rule Engine Layer)
- **Downstream Dependencies**:
  - ERP Task Engine routing and notification system

3.4 Error Handling and Fallbacks

The Compliance Agent must gracefully handle common error scenarios:

- **Missing Spec Ranges**: If a parameter range is absent, logs a *MissingRange* event and falls back to default thresholds.
- **API Timeouts**: Retries up to three times with exponential backoff before returning *DegradedCompliance* status.
- **Invalid Payloads**: Returns *400 Bad Request* with detailed errors.
- **Service Unavailability**: Uses cached rules and issues an *AgentFallback* alert.

3.5 Performance and SLA Targets

The Compliance Agent targets:

- 95% of checks within 200 ms for 1,000 concurrent requests.
- 3,000 parameter validations per second throughput.

- 99.9% uptime SLA.
- <0.1% fallback rate.

3.6 Compliance Agent API Specification

Inputs

- JSON payload containing:
  - `projectId`
  - `parameters` (name, value)
  - `specRanges` (min, max)

Outputs

- JSON report with:
  - `status` (pass / fail)
  - list of violations

Trigger Events

- HTTP POST `/agents/compliance`
- Task dispatch from the Orchestrator

External Dependencies / APIs:

- Knowledge Base (MongoDB) for spec ranges
- External Rule Engine API
- Prometheus metrics endpoint
- Fallback cache of rules for offline operation

3.7 Use Case

An engineer submits a Process Flow Diagram (PFD) for compliance review. The Compliance Agent:

1. Extracts structured data from the PDF using OCR and parsing
2. Validates data completeness and format consistency
3. Performs rule-based checks against engineering specifications
4. Conducts knowledge-based validation using RAG and LLM reasoning
5. Generates a comprehensive compliance report highlighting any gaps or violations

6. Routes the report to the appropriate reviewer with notification

Chapter 4

Document Generation Agent

4.1 Function

The Document Generation Agent operates within the AI Layer of the system architecture and performs intelligent content generation for structured documents. It fetches structured data from ERP APIs and uses LLM-based content generation to produce narrative sections and templated content in JSON format for downstream processing.

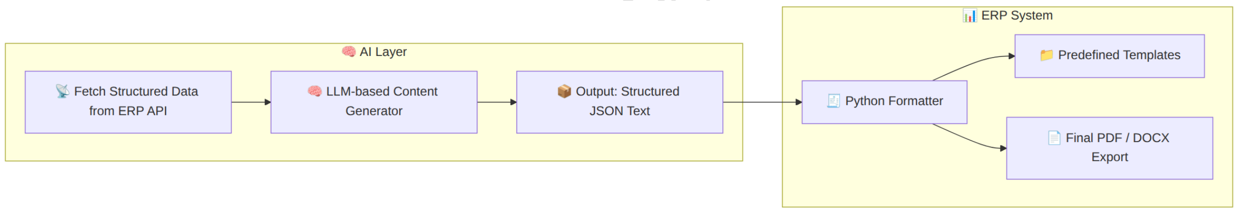
🧠 AI Layer
📡 Fetch Structured Data from ERP API
🧠 LLM-based Content Generator
📦 Output: Structured JSON Text
📊 ERP System
🧾 Python Formatter
📁 Predefined Templates
📄 Final PDF / DOCX Export

Figure 4.1: Document Generation Agent Workflow and AI Layer Integration

4.2 Scope and Boundaries

4.2.1 AI Layer Scope (In-Scope)

The Document Generation Agent is responsible only for components within the **AI Layer** subgraph:

- **Data Fetching**: Retrieval of structured data from ERP API endpoints
- **LLM-based Content Generation**:
  - Intelligent narrative text generation
  - Template content population with contextual information
  - Dynamic content adaptation based on document type and requirements

- **Structured Output Generation**: Production of organized JSON output containing generated text and structured content

4.2.2 External Dependencies (Out-of-Scope)

The following components are **outside** the AI Layer and handled by the ERP System:

- **ERP Layer Components**:
  - Python-based document formatting and layout processing
  - Predefined template management and selection
  - Final document export (PDF/DOCX generation)
  - Document storage and versioning systems

4.2.3 Integration Boundaries

- **Input**: Accesses structured data through ERP API interfaces
- **Output**: Delivers structured JSON content to ERP Layer formatting systems
- **Dependencies**: Requires access to project data, templates metadata, and content guidelines
- **Limitations**: Cannot directly control document formatting, template design, or final export processes

4.3 Architecture Components

The Document Generation Agent operates exclusively within the AI Layer and consists of the following components:

4.3.1 AI Layer Components (Document Generation Agent Scope)

- **ERP API Interface**: Establishes connections and fetches structured data from various ERP endpoints
- **LLM-based Content Generator**:
  - **Narrative Engine**: Generates contextual descriptions and explanations
  - **Template Populator**: Fills template fields with appropriate content
  - **Content Adapter**: Adjusts tone, style, and technical level based on document type
- **JSON Formatter**: Structures generated content into organized JSON output for downstream processing

4.3.2 External Layer Dependencies (Outside AI Agent Scope)

These components are managed by the ERP System but interface with the AI Layer:

- **Downstream Dependencies**:
  - Python Formatter (ERP Layer)
  - Predefined Templates (ERP Layer)
  - Document Export System (PDF/DOCX generation)

4.4 Template Versioning

To ensure consistent document formats and support rollback:

- All templates are stored with semantic version tags in the Knowledge Base (e.g., v1.2.0).
- Template updates require a changelog entry and approval before deployment.
- The agent selects the template version based on the project's configuration.

4.5 Sample Template Snippet

An example JSON-based template definition:

```json
{
  "templateId": "datasheet_equip_v1",
  "version": "1.2.0",
  "fields": [
    {"name": "equipmentName", "type": "text"},
    {"name": "capacity", "type": "number", "format": "%0.2f"},
    {"name": "operatingTemp", "type": "number", "unit": "\"C\""}
  ],
  "narrativeSections": [
    {"id": "introduction", "prompt": "Generate intro for {{
       equipmentName}} with capacity {{capacity}}."}
  ],
  "render": {
    "engine": "wkhtmltopdf",
    "options": {"margin-top": "15mm", "margin-bottom": "15mm"}
  }
}
```

4.6 Document Generation Agent

Inputs

- JSON containing:
  - `templateId`
  - context data fields

Outputs

- Generated document (PDF or Word) as URL or binary payload

Trigger Events

- HTTP POST `/agents/docgen`
- User or workflow-initiated request

External Dependencies / APIs:

- Knowledge Base for template retrieval
- LLM runtime for narrative sections
- Jinja2 (or equivalent) template engine
- Rendering engine (e.g. `wkhtmltopdf`)
- File Management API for storing the output document

4.7 Use Case

A manager requests a compliance certificate for a completed project. The Document Generation Agent:

1. Fetches structured project data, compliance results, and specifications from ERP APIs
2. Uses LLM-based content generation to create narrative sections describing compliance status
3. Generates contextual explanations for technical parameters and regulatory requirements
4. Outputs structured JSON containing generated content and data mappings
5. Passes JSON output to ERP Layer formatting system for final PDF certificate generation

Chapter 5

Diagram Parsing Agent

5.1 Function

The Diagram Parsing Agent operates as a specialized AI component that transforms raw P&ID (Process and Instrumentation Diagram) and CAD files into structured, machine-readable graph representations. It utilizes computer vision, OCR technology, and spatial analysis to identify equipment, extract connectivity information, and build comprehensive network models of industrial processes.

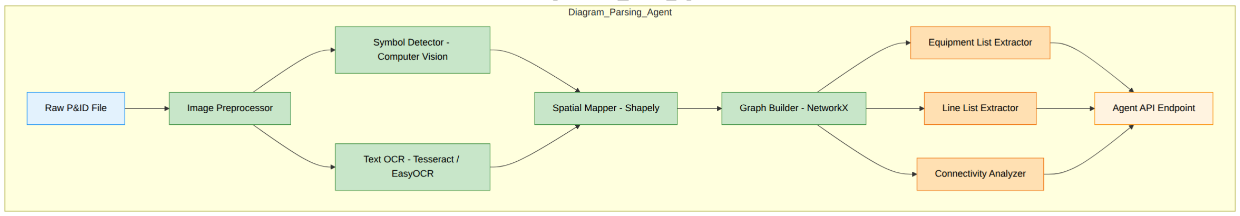

Figure 5.1: Diagram Parsing Agent Processing Pipeline and Component Architecture

5.2 Scope and Boundaries

5.2.1 Diagram Parsing Agent Scope (In-Scope)

The Diagram Parsing Agent is responsible for the complete processing pipeline within its component boundary:

- **Image Preprocessing**:
  - Raw P&ID file ingestion and format normalization
  - Image enhancement and noise reduction
  - Resolution optimization for analysis

- **Computer Vision Processing**:
  - Symbol detection using advanced computer vision algorithms
  - Equipment identification and classification
  - Shape and pattern recognition for industrial symbols
- **Text Recognition and Extraction**:
  - OCR using Tesseract/EasyOCR for text extraction
  - Label identification and association
  - Tag number and specification text recognition
- **Spatial Analysis and Mapping**:
  - Spatial relationship analysis using Shapely
  - Coordinate mapping and positioning
  - Connection line detection and tracing
- **Graph Construction**:
  - Network graph building using NetworkX
  - Node and edge relationship establishment
  - Topology validation and verification
- **Structured Output Generation**:
  - Equipment List Extraction with specifications
  - Line List Extraction with connectivity details
  - Connectivity Analysis with relationship mapping
- **API Service Integration**:
  - RESTful API endpoint provision
  - JSON output formatting and delivery
  - Error handling and status reporting

5.2.2 External Dependencies (Out-of-Scope)

The following components are outside the Diagram Parsing Agent's direct control:

- **CAD Authoring and Modification**: Cannot create or edit original CAD files
- **File Storage and Management**: Does not handle long-term storage of input or output files

- **Diagram Validation**: Does not verify engineering accuracy or compliance of source diagrams
- **Real-time Collaboration**: Does not support multi-user editing or version control
- **External System Integration**: Direct integration with CAD software or ERP systems beyond API endpoints

5.2.3 Technical Limitations

- **Input Format Constraints**: Limited to supported image formats (PDF, PNG, JPEG, TIFF)
- **Diagram Quality Dependencies**: Performance depends on input diagram clarity and resolution
- **Symbol Library Scope**: Recognition limited to trained symbol sets and standard industrial notation
- **Complex Topology Handling**: May require manual validation for highly complex or non-standard layouts

5.3 Architecture Components

The Diagram Parsing Agent consists of a comprehensive processing pipeline with specialized components:

5.3.1 Core Processing Components

- **Image Preprocessor**:
  - **Format Handler**: Supports multiple input formats (PDF, PNG, JPEG, TIFF)
  - **Enhancement Engine**: Noise reduction, contrast adjustment, resolution optimization
  - **Segmentation Module**: Region of interest identification and extraction
- **Symbol Detector (Computer Vision)**:
  - **Pattern Matching**: Template-based symbol recognition
  - **Machine Learning Models**: Trained classifiers for equipment identification
  - **Confidence Scoring**: Reliability assessment for detected symbols
- **Text OCR Engine**:
  - **Tesseract Integration**: Primary OCR engine for text extraction
  - **EasyOCR Support**: Alternative OCR for improved accuracy

  - **Text Processing**: Label cleaning, formatting, and association
- **Spatial Mapper**:
  - **Shapely Integration**: Geometric analysis and spatial relationships
  - **Coordinate System**: Consistent positioning and measurement
  - **Connection Tracing**: Line following and intersection detection
- **Graph Builder**:
  - **NetworkX Integration**: Graph data structure construction
  - **Node Management**: Equipment and junction point representation
  - **Edge Definition**: Connection and flow relationship establishment

5.3.2 Output Generation Components

- **Equipment List Extractor**:
  - **Equipment Catalog**: Comprehensive equipment identification and specification
  - **Metadata Association**: Equipment properties, ratings, and characteristics
  - **Standardization**: Consistent naming and classification schemes
- **Line List Extractor**:
  - **Piping Identification**: Line routing and specification details
  - **Flow Direction**: Process flow analysis and direction determination
  - **Material Specifications**: Pipe sizing, materials, and ratings
- **Connectivity Analyzer**:
  - **Topology Mapping**: Complete process flow connectivity
  - **Relationship Analysis**: Equipment interdependencies and process relationships
  - **Validation Checks**: Connectivity consistency and completeness verification
- **API Endpoint**:
  - **RESTful Interface**: Standard HTTP API for external integration
  - **JSON Formatting**: Structured output in JSON format
  - **Error Handling**: Comprehensive error reporting and status codes

5.4 Diagram Parsing Agent

Inputs

- P&ID file (PDF, SVG, PNG, JPEG)

Outputs

- JSON graph including:
  - equipment list
  - line list
  - connectivity data

Trigger Events

- HTTP POST `/agents/diagram-parse`
- File-upload event from File Management

External Dependencies / APIs:

- OpenCV for symbol detection
- Tesseract or EasyOCR for text extraction
- Shapely for spatial mapping
- NetworkX for graph construction
- Prometheus for performance metrics

5.5 Use Case

An engineering team uploads a Process Flow Diagram (P&ID) for a chemical processing unit. The Diagram Parsing Agent:

1. **Receives the P&ID file**: Ingests the PDF/image file and validates format compatibility
2. **Preprocesses the image**: Enhances image quality, adjusts contrast, and optimizes resolution for analysis
3. **Detects symbols and equipment**: Uses computer vision to identify pumps, valves, vessels, and instrumentation
4. **Extracts text labels**: Applies OCR to extract equipment tags, specifications, and process parameters
5. **Maps spatial relationships**: Analyzes the positioning and connections between identified components
6. **Builds process graph**: Constructs a NetworkX graph representing the complete process topology
7. **Generates structured outputs**:

- Equipment List: JSON array of all identified equipment with tags and specifications
- Line List: Comprehensive piping and connection details
- Connectivity Analysis: Process flow relationships and interdependencies

8. **Delivers via API**: Returns structured JSON data through RESTful API endpoint for integration with downstream systems

Chapter 6

Email Agent

6.1 Function

The Email Agent operates within the AI Layer of the system architecture and performs intelligent processing of inbound emails to extract structured information, classify content types, and route processed data to appropriate ERP modules. It utilizes NLP parsing, intent classification, and RAG-based analysis to understand email content and context.

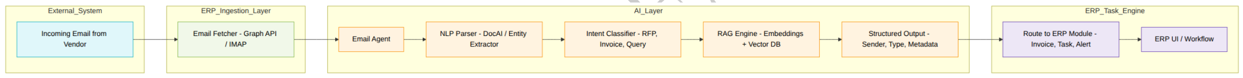
External_System
Incoming Email from Vendor
ERP_Ingestion_Layer
Email Fetcher - Graph API / IMAP
AI_Layer
Email Agent
NLP Parser - DocAI / Entity Extractor
Intent Classifier - RFP, Invoice, Query
RAG Engine - Embeddings + Vector DB
Structured Output - Sender, Type, Metadata
ERP_Task_Engine
Route to ERP Module - Invoice, Task, Alert
ERP UI / Workflow

Figure 6.1: Email Agent Processing Workflow and Layer Integration

6.2 Scope and Boundaries

6.2.1 AI Layer Scope (In-Scope)

The Email Agent is responsible only for components within the **AI Layer** subgraph:

- **Email Agent Core**: Central orchestration of email processing workflow within AI layer
- **NLP Parsing**:
  - DocAI integration for document analysis
  - Entity extraction from email content and attachments
  - Natural language understanding of email body text
- **Intent Classification**:
  - Email type identification (RFP, Invoice, Query, Instructions)

  - Content categorization and priority assessment
  - Context-aware classification using ML models
- **RAG Engine Integration**:
  - Embeddings generation for semantic analysis
  - Vector database queries for context matching
  - Knowledge retrieval for enhanced understanding
- **Structured Output Generation**:
  - Sender identification and metadata extraction
  - Email type classification results
  - Structured metadata and parameter extraction

6.2.2 External Dependencies (Out-of-Scope)

The following components are **outside** the AI Layer and handled by other systems:

- **External System**: Incoming emails from vendors, clients, and external sources
- **ERP Ingestion Layer**:
  - Email fetching services (Graph API/IMAP connectivity)
  - Email server authentication and connection management
  - Raw email retrieval and initial processing
- **ERP Task Engine**:
  - Routing to appropriate ERP modules (Invoice, Task, Alert systems)
  - ERP UI and workflow integration
  - Email delivery and notification systems
  - Task creation and assignment workflows

6.2.3 Integration Boundaries

- **Input**: Receives structured email data from ERP Ingestion Layer
- **Output**: Delivers processed email metadata and classification to ERP Task Engine
- **Dependencies**: Accesses shared knowledge bases and vector databases for context
- **Limitations**: Cannot fetch emails directly, deliver responses, or create ERP tasks

6.3 Architecture Components

The Email Agent operates exclusively within the AI Layer and consists of the following components:

6.3.1 AI Layer Components (Email Agent Scope)

- **Email Agent Core**:
  - **Workflow Orchestrator**: Manages the email processing pipeline
  - **Context Manager**: Maintains email processing state and history
- **NLP Parser**:
  - **DocAI Integration**: Processes attachments and document content
  - **Entity Extractor**: Identifies key entities, parameters, and values
  - **Text Analyzer**: Processes email body content and metadata
- **Intent Classifier**:
  - **Content Categorizer**: Classifies emails as RFP, Invoice, Query, etc.
  - **Priority Assessor**: Determines urgency and importance levels
  - **Context Analyzer**: Understands business context and relationships
- **RAG Engine**:
  - **Embedding Generator**: Creates semantic representations of email content
  - **Vector Search**: Queries knowledge base for relevant context
  - **Knowledge Retriever**: Accesses historical and reference information
- **Output Formatter**:
  - **Metadata Extractor**: Structures sender, recipient, and timing information
  - **Classification Reporter**: Provides content type and confidence scores
  - **Parameter Organizer**: Formats extracted parameters and instructions

6.3.2 External Layer Dependencies (Outside AI Agent Scope)

These components are managed by other systems but interface with the AI Layer:

- **Upstream Dependencies**:
  - Email Fetcher services (ERP Ingestion Layer)
  - External email systems and vendors
- **Downstream Dependencies**:

- ERP Module Router (ERP Task Engine)
- ERP UI and workflow systems

6.4 Email Agent

Inputs

- Raw MIME email (headers, body, attachments)

Outputs

- JSON containing:
  - extracted instructions
  - parameter values
  - attachment metadata

Trigger Events

- HTTP POST `/agents/email-parse`
- Incoming email webhook

External Dependencies / APIs

- MIME parsing library (e.g. Python `email`, MimeKit)
- LLM runtime for intent detection
- Virus-scan service for attachments
- Prometheus metrics endpoint

6.5 Use Case

A vendor sends an email with subject “Run simulation at 6 bar pressure for Project Alpha” with P&ID attachments. The Email Agent:

1. Receives structured email data from the ERP Ingestion Layer email fetcher
2. Uses NLP parser with DocAI to extract text from email body and analyze P&ID attachments
3. Identifies entities: “simulation”, “6 bar pressure”, “Project Alpha”
4. Classifies the email intent as “Instruction/Task Request” using the intent classifier
5. Applies RAG engine to match against similar historical requests and project context
6. Generates structured output with sender metadata, classification (Instruction), and extracted parameters

7. Delivers structured output to ERP Task Engine for routing to appropriate project management module

Chapter 7

ERP-Chat Agent

7.1 Function

The ERP-Chat Agent operates within the AI Layer of the system architecture and provides intelligent natural language interface for querying ERP systems. It processes user queries through natural language understanding, semantic analysis, and API routing to deliver structured responses about equipment, project data, and system information.

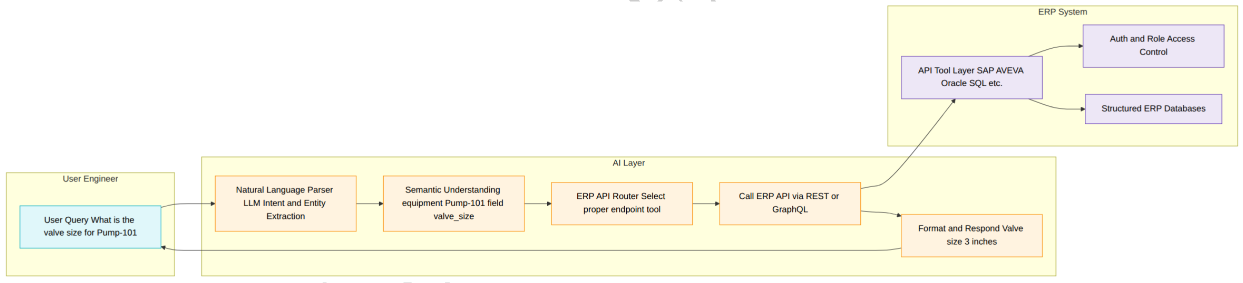
ERP System
Auth and Role Access Control
API Tool Layer SAP AVEVA Oracle SQL etc.
Structured ERP Databases
AI Layer
User Engineer
User Query What is the valve size for Pump-101
Natural Language Parser LLM Intent and Entity Extraction
Semantic Understanding equipment Pump-101 field valve_size
ERP API Router Select proper endpoint tool
Call ERP API via REST or GraphQL
Format and Respond Valve size 3 inches

Figure 7.1: ERP-Chat Agent Natural Language Processing and API Integration

7.2 Scope and Boundaries

7.2.1 AI Layer Scope (In-Scope)

The ERP-Chat Agent is responsible only for components within the **AI Layer** subgraph:

- **Natural Language Processing**:
  - LLM-based intent extraction from user queries
  - Entity recognition and parameter identification
  - Context understanding and query disambiguation

- **Semantic Understanding**:
  - Equipment identification and mapping (e.g., Pump-101 to valve_size field)
  - Relationship analysis between entities and attributes
  - Domain-specific terminology interpretation
- **ERP API Routing**:
  - Selection of appropriate API endpoints and tools
  - Query translation to API-compatible formats
  - REST/GraphQL call preparation
- **Response Formatting**:
  - Natural language response generation
  - Context-aware answer formatting
  - User-friendly presentation of technical data

7.2.2 External Dependencies (Out-of-Scope)

The following components are **outside** the AI Layer and handled by other systems:

- **User Interface**: External user interaction systems and chat interfaces
- **ERP System Components**:
  - API Tool Layer (SAP, AVEVA, Oracle, SQL databases)
  - Authentication and Role-based Access Control
  - Structured ERP databases and data storage
  - Task orchestration and workflow management logic

7.2.3 Integration Boundaries

- **Input**: Receives natural language queries from external user interfaces
- **Output**: Delivers formatted natural language responses to users
- **Dependencies**: Accesses ERP APIs through secure authentication layers
- **Limitations**: Cannot modify ERP data, create tasks, or perform administrative actions beyond read queries

7.3 Architecture Components

The ERP-Chat Agent operates exclusively within the AI Layer and consists of the following components:

7.3.1 AI Layer Components (ERP-Chat Agent Scope)

- **Natural Language Parser**:
  - **Intent Extraction**: Identifies user query objectives and goals
  - **Entity Recognition**: Extracts equipment names, parameters, and values
  - **Context Manager**: Maintains conversation context and query history
- **Semantic Understanding Engine**:
  - **Equipment Mapper**: Links natural language references to ERP entities
  - **Field Resolver**: Maps requested information to database fields
  - **Relationship Analyzer**: Understands equipment hierarchies and dependencies
- **ERP API Router**:
  - **Endpoint Selector**: Chooses appropriate API tools and endpoints
  - **Query Builder**: Constructs REST/GraphQL queries
  - **Response Handler**: Processes API responses and error handling
- **Response Formatter**:
  - **Natural Language Generator**: Creates human-readable responses
  - **Data Presenter**: Formats technical information appropriately
  - **Context Integrator**: Includes relevant context in responses

7.3.2 External Layer Dependencies (Outside AI Agent Scope)

These components are managed by other systems but interface with the AI Layer:

- **Upstream Dependencies**:
  - User interface systems and chat platforms
- **Downstream Dependencies**:
  - ERP API Tool Layer (SAP, AVEVA, Oracle, SQL)
  - Authentication and Access Control systems
  - ERP databases and data repositories

7.4 ERP Action Agent

Inputs

- Natural-language command
- Project context
- JWT authentication token

Outputs

- Confirmation JSON containing:
  - `taskId`
  - status message

Trigger Events

- HTTP POST `/agents/erp-action`
- Chatbot or CLI command

External Dependencies / APIs

- ERP Task Engine REST API
- JSON Schema validator
- LLM runtime for NL-to-API mapping
- Prometheus & Loki for logging

7.5 Use Case

A user asks: "What is the valve size for Pump-101?" The ERP-Chat Agent:

1. Parses the natural language query using LLM for intent and entity extraction
2. Identifies "Pump-101" as equipment entity and "valve size" as the requested field
3. Uses semantic understanding to map the query to appropriate ERP database fields
4. Routes the query to the correct ERP API endpoint with proper authentication
5. Calls the ERP API via REST/GraphQL to retrieve the valve size information
6. Formats the response into natural language: "Valve size: 3 inches"
7. Delivers the formatted response back to the user

Chapter 8

Shared Services and Infrastructure

8.1 Infrastructure Architecture Overview

The AI Agent Module is built on a robust, scalable infrastructure architecture that leverages microservices patterns, containerization, and Infrastructure as Code (IaC) principles. This chapter details the comprehensive infrastructure design, deployment strategy, and architectural decisions that enable high availability, scalability, and maintainability.

8.2 Kubernetes Architecture

The system leverages Kubernetes orchestration to provide container management, service discovery, and automated scaling capabilities.

8.2.1 Network and Ingress Layer

- **DNS Resolution**: External DNS routing to load balancer
- **Load Balancer**: High-availability traffic distribution
- **Ingress Controller**: Kubernetes-native traffic routing and SSL termination
- **Service Mesh**: Inter-service communication, security, and observability

8.2.2 Control Plane Configuration

- **High Availability**: 3-node control plane with etcd clustering
- **API Server**: RESTful API for cluster management
- **Scheduler**: Intelligent workload placement across nodes
- **Controller Manager**: Automated cluster state management

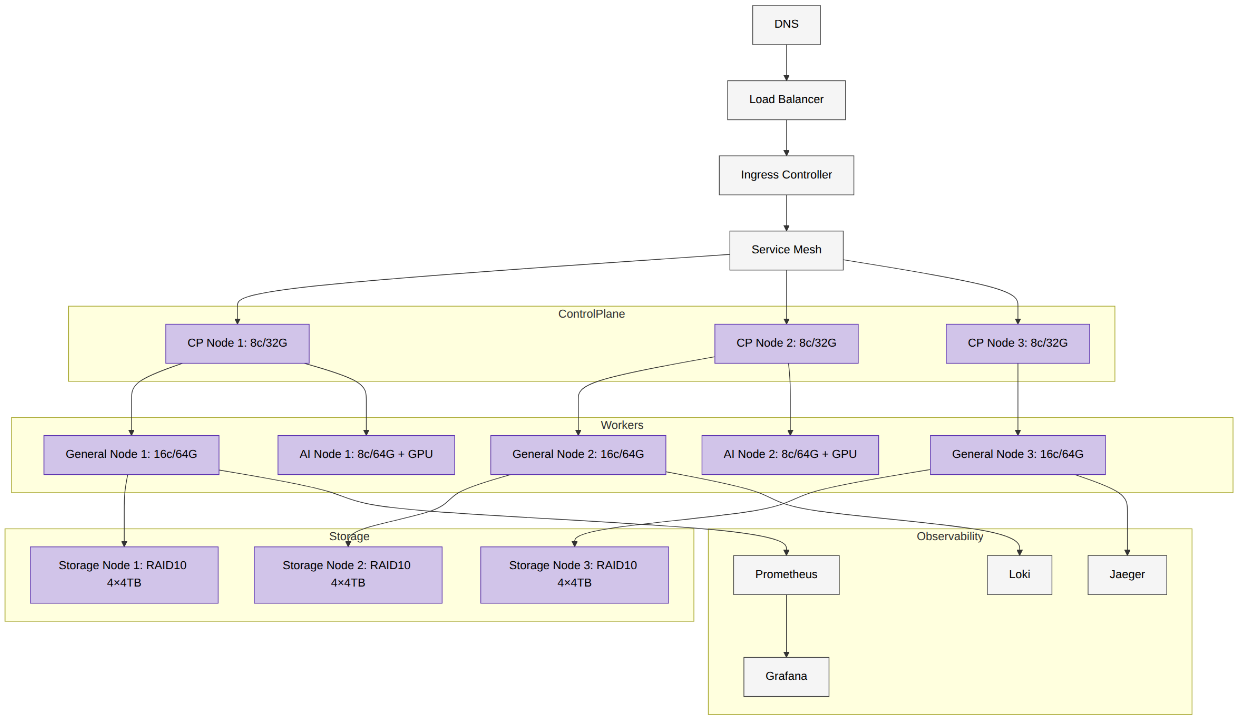

Figure 8.1: Kubernetes Cluster Architecture and Service Topology

8.2.3 Worker Node Specialization

- **General Workers**: Application services, API gateways, business logic
- **AI Workers**: GPU-accelerated nodes for machine learning workloads
- **Storage Workers**: Persistent volume hosting and data services

8.3 Microservices Architecture Benefits

8.3.1 Scalability and Performance

- **Independent Scaling**: Each AI agent can scale based on demand
- **Resource Optimization**: GPU resources allocated only to AI workloads
- **Load Distribution**: Workloads distributed across specialized nodes
- **Horizontal Scaling**: Add more instances of high-demand services

8.3.2 Development and Deployment Agility

- **Independent Development**: Teams can develop agents independently
- **Technology Diversity**: Different agents can use optimal technology stacks

- **Rapid Deployment**: Individual service updates without system-wide impact
- **Rolling Updates**: Zero-downtime deployments with gradual rollouts

8.3.3 Fault Isolation and Resilience

- **Failure Containment**: Agent failures don't cascade to other services
- **Circuit Breakers**: Automatic failure detection and recovery
- **Redundancy**: Multiple instances provide fault tolerance
- **Health Monitoring**: Proactive issue detection and resolution

8.3.4 Maintainability and Testing

- **Modular Architecture**: Clear separation of concerns and responsibilities
- **Independent Testing**: Unit and integration testing per service
- **Version Management**: Independent versioning and compatibility
- **Code Ownership**: Clear responsibility boundaries for teams

8.4 Infrastructure as Code (IaC) Implementation

8.4.1 IaC Advantages and Benefits

- **Reproducibility**: Consistent environment creation across development, staging, and production
- **Version Control**: Infrastructure changes tracked in Git with full audit trail
- **Automated Deployment**: Eliminate manual configuration errors and reduce deployment time
- **Disaster Recovery**: Rapid environment reconstruction from code definitions
- **Cost Management**: Resource optimization through programmatic resource allocation
- **Compliance**: Automated compliance checks and policy enforcement

8.4.2 IaC Technology Stack

- **Terraform**: Infrastructure provisioning and resource management
- **Ansible**: Configuration management and application deployment
- **Helm Charts**: Kubernetes application packaging and templating
- **GitOps**: Git-based deployment workflows with ArgoCD/Flux

8.4.3 IaC Implementation Strategy

- **Modular Design**: Reusable infrastructure components and templates
- **Environment Separation**: Distinct configurations for dev/staging/production
- **State Management**: Centralized state storage with locking mechanisms
- **Change Validation**: Automated testing and validation pipelines

8.5 Core Infrastructure Services

8.5.1 LLM Runtime

- **Architecture**: Hosted on dedicated GPU nodes via Kubernetes
- **Resource Allocation**: Tesla T4 GPUs with 128GB RAM per node
- **Model Management**: Containerized model serving with version control
- **Load Balancing**: Automatic request distribution across GPU nodes
- **Scaling Strategy**: Horizontal Pod Autoscaler based on GPU utilization

8.5.2 Embedding Service and Vector Database

- **Technology Stack**: Faiss/Qdrant for semantic retrieval operations
- **Data Partitioning**: Project-scoped collections for data isolation
- **Performance Optimization**: In-memory indexing with persistent storage
- **Backup Strategy**: Automated vector index backup and restoration
- **Query Optimization**: Efficient similarity search with caching layers

8.5.3 Knowledge Base

- **Database Technology**: MongoDB collections scoped by project
- **Data Architecture**: Document-oriented storage for flexible schemas
- **Replication**: 3-node replica sets for high availability
- **Indexing Strategy**: Optimized indexes for query performance
- **Data Governance**: Role-based access control and audit logging

8.5.4 Message Queue

- **Technology**: Apache Kafka topics for agent requests and retries
- **Partitioning Strategy**: Topic partitioning for parallel processing

- **Durability**: Persistent message storage with configurable retention
- **Consumer Groups**: Load distribution across multiple agent instances
- **Dead Letter Queues**: Failed message handling and retry mechanisms

8.5.5 Observability and Monitoring

- **Metrics Collection**: Prometheus counters and custom metrics
- **Log Aggregation**: Loki log ingestion with structured logging
- **Distributed Tracing**: Jaeger for request flow analysis
- **Visualization**: Grafana dashboards for operational insights
- **Alerting**: PrometheusAlert manager for incident response

8.6 Security and Compliance

8.6.1 Container Security

- **Image Scanning**: Automated vulnerability assessment
- **Runtime Security**: Pod security policies and admission controllers
- **Network Policies**: Micro-segmentation and traffic isolation
- **Secret Management**: Kubernetes secrets with external secret operators

8.6.2 Authentication and Authorization

- **Identity Provider**: Integrated OIDC/SAML authentication
- **RBAC**: Kubernetes role-based access control
- **Service Authentication**: mTLS for inter-service communication
- **API Security**: OAuth 2.0/JWT token validation

8.7 Operational Excellence

8.7.1 Deployment Strategy

- **CI/CD Pipelines**: Automated testing and deployment workflows
- **Blue-Green Deployments**: Zero-downtime production updates
- **Canary Releases**: Gradual rollouts with automated rollback
- **Feature Flags**: Runtime feature control and A/B testing

8.7.2 Backup and Disaster Recovery

- **Data Backup**: Automated database and persistent volume backups
- **Cross-Region Replication**: Geographic redundancy for critical data
- **Recovery Testing**: Regular disaster recovery drills
- **RTO/RPO Goals**: 15-minute Recovery Time/Point Objectives

Chapter 9

Deployment and Operations

9.1 Deployment Strategy

The deployment strategy follows a robust CI/CD pipeline leveraging GitOps principles to ensure consistent, repeatable deployments across environments. The pipeline includes automated testing, security scanning, and approval gates to maintain high quality and compliance standards.

9.1.1 CI/CD Pipeline Stages

- **Source Control**: Git repository for version control and collaboration
- **Build Stage**: Automated builds with Docker images for each service
- **Test Stage**: Unit, integration, and end-to-end tests
- **Security Scanning**: Automated vulnerability scanning of container images
- **Staging Deployment**: Deploy to staging environment for validation
- **Approval Gates**: Manual review and approval for production deployment
- **Production Deployment**: Automated deployment to production cluster
- **Monitoring and Rollback**: Post-deployment monitoring with automated rollback

9.2 Operational Monitoring

Operational monitoring is critical for maintaining system health and performance. The observability stack includes:

- **Metrics Collection**: Prometheus for real-time metrics and alerting
- **Log Aggregation**: Loki for structured log storage and querying
- **Distributed Tracing**: Jaeger for end-to-end request tracing

- **Dashboards**: Grafana for visualizing metrics, logs, and traces
- **Alerting**: Prometheus Alertmanager for incident response

9.3 Incident Management

Incident management processes are established to ensure rapid response and resolution of operational issues. The process includes:

- **Incident Detection**: Automated alerts from monitoring systems
- **Incident Triage**: Initial assessment and categorization of incidents
- **Incident Response Team**: Designated team for incident resolution
- **Root Cause Analysis**: Post-incident review to identify underlying issues
- **Incident Reporting**: Documentation of incidents and resolutions for future reference

9.4 Backup and Disaster Recovery

Backup and disaster recovery strategies are highly relevant for on-premise deployments, as they ensure data integrity and service availability in the event of hardware failures, site outages, or other disruptions. For on-premise environments, it is critical to implement:

- **Automated Backups**: Schedule regular backups of all databases and persistent volumes to local and, if possible, offsite storage.
- **Cross-Site Replication**: If multiple physical locations are available, replicate critical data to a secondary site for geographic redundancy.
- **Disaster Recovery Drills**: Regularly test recovery procedures to validate backup integrity and team readiness.
- **Recovery Time Objectives (RTO)**: Define and monitor RTOs (e.g., 15 minutes for critical services) to ensure rapid restoration.
- **Recovery Point Objectives (RPO)**: Set RPOs (e.g., 15 minutes for critical data) to minimize data loss in case of failure.
- **Offsite Backup Storage**: Use secure, encrypted offsite or cloud storage for backup copies if regulatory and security policies permit.

9.5 Security and Compliance

Security and compliance are paramount in the AI Agent Module. As the AI Agent Module is a subsystem within the larger ERP system, core responsibilities such as authentication, authorization, and RBAC are managed by the ERP platform. The AI subsystem enforces the following measures within its scope:

- **Container Security**: Image scanning, runtime security policies, and network segmentation
- **Service Communication Security**: mTLS for secure inter-service communication within the AI subsystem
- **Data Encryption**: Encryption at rest and in transit for sensitive data handled by the AI module
- **Compliance Audits**: Regular security audits and vulnerability assessments for AI-specific components
- **Incident Response Plan**: Established procedures for handling security incidents and breaches within the AI subsystem

Note: All user authentication, authorization, and RBAC enforcement are handled by the ERP system. The AI Agent Module relies on the ERP for identity and access management.

Chapter 10

Hardware and Software Requirements

10.1 Hardware Requirements

The AI Agent Module requires a robust hardware infrastructure to support its computational and storage needs. The infrastructure is designed to run on a single physical host with virtualized components to provide isolation, resource management, and scalability.

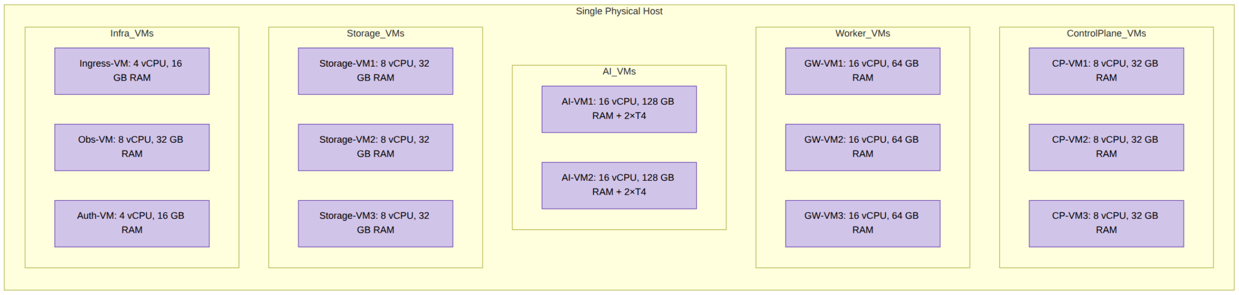
Single Physical Host
Infra_VMs
Ingress-VM: 4 vCPU, 16 GB RAM
Obs-VM: 8 vCPU, 32 GB RAM
Auth-VM: 4 vCPU, 16 GB RAM
Storage_VMs
Storage-VM1: 8 vCPU, 32 GB RAM
Storage-VM2: 8 vCPU, 32 GB RAM
Storage-VM3: 8 vCPU, 32 GB RAM
AI_VMs
AI-VM1: 16 vCPU, 128 GB RAM + 2×T4
AI-VM2: 16 vCPU, 128 GB RAM + 2×T4
Worker_VMs
GW-VM1: 16 vCPU, 64 GB RAM
GW-VM2: 16 vCPU, 64 GB RAM
GW-VM3: 16 vCPU, 64 GB RAM
ControlPlane_VMs
CP-VM1: 8 vCPU, 32 GB RAM
CP-VM2: 8 vCPU, 32 GB RAM
CP-VM3: 8 vCPU, 32 GB RAM

Figure 10.1: Physical Host VM Distribution and Resource Allocation

10.1.1 VM Distribution Strategy

The physical host is divided into specialized virtual machine pools, each optimized for specific workload types:

Control Plane VMs

- **Configuration**: 3 VMs × (8 vCPU, 32 GB RAM)
- **Purpose**: Kubernetes control plane management, API server, etcd, scheduler
- **High Availability**: Triple redundancy ensures cluster availability during node failures
- **Resource Optimization**: Moderate CPU/RAM allocation focused on management overhead

General Worker VMs

- **Configuration**: 3 VMs × (16 vCPU, 64 GB RAM)
- **Purpose**: General application workloads, API gateways, business logic services
- **Scalability**: Higher CPU/RAM allocation for compute-intensive tasks
- **Load Distribution**: Distributed scheduling across multiple nodes

AI-Specialized VMs

- **Configuration**: 2 VMs × (16 vCPU, 128 GB RAM + 2×T4 GPUs)
- **Purpose**: LLM runtime, embedding generation, AI model inference
- **GPU Acceleration**: Dedicated Tesla T4 GPUs for AI workloads
- **Memory Optimization**: High RAM allocation for large model loading

Storage VMs

- **Configuration**: 3 VMs × (8 vCPU, 32 GB RAM)
- **Purpose**: Distributed storage, database hosting, persistent volumes
- **Data Redundancy**: Triple replication for data durability
- **I/O Optimization**: Optimized for storage operations and data access

Infrastructure Support VMs

- **Ingress VM**: 4 vCPU, 16 GB RAM – Load balancing and traffic routing
- **Observability VM**: 8 vCPU, 32 GB RAM – Monitoring, logging, and metrics
- **Authentication VM**: 4 vCPU, 16 GB RAM – Identity and access management

10.2 Server Specifications and Expected Pricing

Component	Qty	Unit Cost (INR)	Total (INR)	Notes
Intel Xeon Gold 6348 CPU	2	573,750	1,147,500	RCP 3,072 × 83/USD
32 GB DDR4 ECC RDIMM	16	45,000	720,000	20 K per 32 GB module
NVIDIA Tesla T4 16 GB GPU	4	325,506	1,302,021	ServerBasket list price
1 TB NVMe SSD (enterprise)	4	27,000	108,000	12 K/TB
8 TB HDD (enterprise, RAID-10)	12	45,000	540,000	20 K each
Dual-port 25 GbE NIC	2	56,250	112,500	SFP28 cards
Dual-port 10 GbE NIC	2	27,000	54,000	SFP+ cards
Chassis, PSUs, Board, Fans, etc.	—	225,000	225,000	Supermicro or similar
Scaled Total			**4,209,021**	42.1 lakhs (4.2 M)

Table 10.1: Server Bill of Materials

10.3 Software Requirements

Component	Version	License	Cost (INR)	Notes
Core Infrastructure				
Kubernetes	1.26.0	Open Source	Free	kubeadm managed
Docker Engine	20.10.12	Open Source	Free	Container runtime
Helm	3.9.0	Open Source	Free	K8s package manager
Prometheus	2.36.0	Open Source	Free	Metrics/alerting
Grafana	9.2.0	Open Source	Free	Dashboards
Loki	2.6.0	Open Source	Free	Log aggregation
Jaeger	1.42.0	Open Source	Free	Tracing
ArgoCD	2.5.0	Open Source	Free	GitOps delivery
Terraform	1.4.0	Open Source	Free	IaC provisioning
Ansible	2.12.0	Open Source	Free	Config mgmt
Databases and Messaging				
MongoDB	6.0.0	Open Source	Free	Document DB
Qdrant	0.9.0	Open Source	Free	Vector DB
Apache Kafka	3.4.0	Open Source	Free	Message queue
Redis	7.0.0	Open Source	Free	Cache/broker
PostgreSQL	15.0	Open Source	Free	Relational DB (opt)
AI/ML and LLM Stack				
Python	3.10	Open Source	Free	Backend/AI dev
FastAPI	0.95.0	Open Source	Free	API framework
Uvicorn	0.22.0	Open Source	Free	ASGI server
Gunicorn	20.1.0	Open Source	Free	WSGI server (opt)
LangChain	0.1.0	Open Source	Free	LLM orchestration
LangGraph	0.0.20	Open Source	Free	LLM workflow
LangSmith	0.0.40	Open Source	Free	LLM tracing
vLLM	0.3.0	Open Source	Free	LLM inference
HF Transformers	4.35.0	Open Source	Free	Model hub
Mistral	7B/8x7B	Open Source	Free	Open LLMs
DeepSeek	7B/67B	Open Source	Free	Open LLMs
Llama	2/3, 7B/13B	Open Source	Free*	Meta LLMs
TinyLlama/Nano LLMs	1.1B/3B	Open Source	Free	Small LLMs
SentenceTransformers	2.2.2	Open Source	Free	Embeddings
Tesseract OCR	5.3.0	Open Source	Free	OCR engine
EasyOCR	1.6.2	Open Source	Free	OCR library
NetworkX	3.1	Open Source	Free	Graphs
Shapely	2.0.1	Open Source	Free	Spatial analysis
NVIDIA GPU Stack				
NVIDIA Drivers	525.60.11	Proprietary	Free*	GPU support
NVIDIA CUDA Toolkit	11.8.0	Proprietary	Free*	GPU programming
NVIDIA cuDNN	8.6.0	Proprietary	Free*	Deep learning
Total	—	—	**Free**	All open/free use

Chapter 11

Implementation Plan

11.1 Project Overview and Timeline

The AI Agent ERP System implementation follows a structured 15-month development plan starting July 15, 2025, and concluding by October 2026. The project is organized into parallel development streams with strategic dependencies and integration points to ensure efficient resource utilization and risk mitigation.

11.2 Development Phases

11.2.1 Phase 1: Infrastructure Foundation (July September 2025)

Infrastructure Design & Node Setup (60 days)

- **Duration**: July 15 - September 13, 2025
- **Deliverables**:
  - Complete hardware procurement and physical server setup
  - VM distribution strategy implementation with specialized pools
  - Kubernetes cluster initialization with 3-node control plane
  - Network configuration with ingress controllers and load balancers
- **Key Milestones**:
  - Hardware delivery and data center setup
  - Hypervisor installation and VM provisioning
  - Kubernetes control plane deployment
  - Basic networking and storage configuration

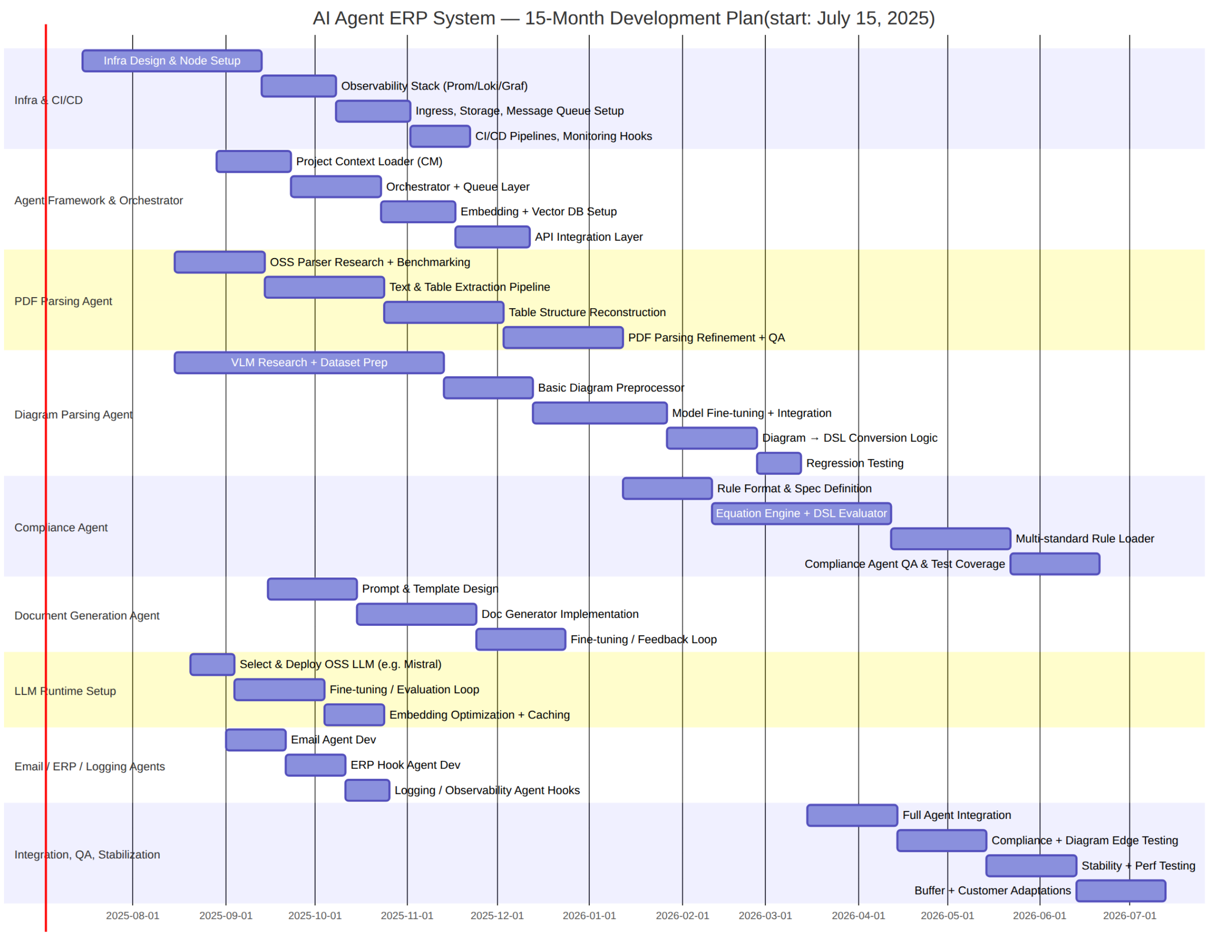

Figure 11.1: AI Agent ERP System 15-Month Development Timeline

Observability Stack Deployment (25 days)

- **Duration**: September 14–October 8, 2025
- **Dependencies**: Infrastructure Design completion
- **Deliverables**:
  - Prometheus metrics collection and alerting setup
  - Loki log aggregation deployment
  - Grafana dashboards and visualization configuration
  - Jaeger distributed tracing implementation
- **Success Criteria**:
  - Full cluster visibility with real-time metrics
  - Automated alerting for critical system events
  - Centralized log aggregation across all services

Core Services Setup (25 days)

- **Duration**: October 9–November 2, 2025
- **Dependencies**: Observability stack completion
- **Deliverables**:
  - Ingress controllers and external access configuration
  - Persistent storage setup with distributed storage layer
  - Apache Kafka message queue deployment
  - Redis caching layer implementation

CI/CD Pipeline Implementation (20 days)

- **Duration**: November 3–November 22, 2025
- **Dependencies**: Core services completion
- **Deliverables**:
  - GitOps workflow with ArgoCD deployment
  - Automated testing pipelines for all components
  - Container registry and security scanning integration
  - Deployment monitoring and rollback mechanisms

11.2.2 Phase 2: Agent Framework Development (August–November 2025)

Project Context Loader & Configuration Management (25 days)

- **Duration**: August 29–September 22, 2025
- **Parallel Track**: Can start while infrastructure is being finalized
- **Deliverables**:
  - Project-scoped knowledge base implementation
  - Configuration management system for agent parameters
  - Dynamic context loading mechanisms
  - Role-based access control for project data

Orchestrator & Queue Layer (30 days)

- **Duration**: September 23–October 22, 2025
- **Dependencies**: Context loader completion
- **Deliverables**:
  - Central orchestrator service with workflow management
  - Message queue integration for agent coordination
  - Request routing and load balancing logic
  - Dead letter queue handling for failed requests

Embedding & Vector Database Setup (25 days)

- **Duration**: October 23–November 16, 2025
- **Dependencies**: Orchestrator completion
- **Deliverables**:
  - Qdrant vector database deployment and configuration
  - Embedding generation pipeline with SentenceTransformers
  - Semantic search capabilities implementation
  - Vector index optimization and caching layer

API Integration Layer (25 days)

- **Duration**: November 17–December 11, 2025
- **Dependencies**: Vector database completion

- **Deliverables**:
  - REST/GRPC API gateway implementation
  - Authentication and authorization middleware
  - API rate limiting and security controls
  - External system integration endpoints

11.2.3 Phase 3: Specialized Agent Development (August 2025–March 2026)

PDF Parsing Agent Development

- **Research Phase** (30 days): August 15–September 13, 2025
  - Open-source parser evaluation and benchmarking
  - Performance testing with representative document samples
  - Technology stack selection and architecture design
- **Text & Table Extraction Pipeline** (40 days): September 14–October 23, 2025
  - OCR integration with Tesseract and EasyOCR
  - Table detection and structure recognition
  - Text extraction with layout preservation
- **Table Structure Reconstruction** (40 days): October 24–December 2, 2025
  - Advanced table parsing algorithms
  - Cell merging and header detection
  - Data validation and quality assurance
- **Refinement & Quality Assurance** (40 days): December 3–January 11, 2026
  - Accuracy optimization and error handling
  - Performance tuning and scalability testing
  - Integration testing with downstream systems

Diagram Parsing Agent Development

- **VLM Research & Dataset Preparation** (90 days): August 15–November 13, 2025
  - Vision Language Model evaluation and selection
  - P&ID and CAD diagram dataset collection and annotation
  - Training infrastructure setup and optimization

- **Basic Diagram Preprocessor** (30 days): November 14–December 13, 2025
  - Image preprocessing and enhancement algorithms
  - Format normalization and quality optimization
  - Symbol detection pipeline implementation
- **Model Fine-tuning & Integration** (45 days): December 14, 2025–January 27, 2026
  - Computer vision model training and validation
  - Spatial analysis integration with Shapely
  - OCR integration for text and label extraction
- **Diagram to DSL Conversion** (30 days): January 28–February 26, 2026
  - NetworkX graph construction algorithms
  - Equipment list and connectivity analysis
  - Structured output generation and validation
- **Regression Testing** (15 days): February 27–March 13, 2026
  - Comprehensive test suite development
  - Performance benchmarking and optimization
  - Edge case validation and error handling

Compliance Agent Development

- **Rule Format & Specification Definition** (30 days): January 12–February 10, 2026
  - Compliance rule definition language design
  - Regulatory standard mapping and categorization
  - Validation framework architecture
- **Equation Engine & DSL Evaluator** (60 days): February 11–April 11, 2026
  - Mathematical equation parsing and evaluation
  - Domain-specific language interpreter
  - Rule execution engine with audit logging
- **Multi-standard Rule Loader** (40 days): April 12–May 21, 2026
  - Dynamic rule loading from multiple regulatory sources
  - Version control and rule update mechanisms
  - Conflict resolution and priority management

- **Quality Assurance & Test Coverage** (30 days): May 22–June 20, 2026
  - Comprehensive test case development
  - Validation against known compliance scenarios
  - Performance optimization and error handling

Document Generation Agent Development

- **Prompt & Template Design** (30 days): September 15–October 14, 2025
  - LLM prompt engineering for document generation
  - Template system design and implementation
  - Content guidelines and style consistency
- **Document Generator Implementation** (40 days): October 15–November 23, 2025
  - ERP API integration for data fetching
  - LLM-based content generation pipeline
  - JSON output formatting and validation
- **Fine-tuning & Feedback Loop** (30 days): November 24–December 23, 2025
  - Content quality optimization and refinement
  - User feedback integration mechanisms
  - Performance tuning and scalability testing

LLM Runtime Setup

- **Model Selection & Deployment** (15 days): August 20–September 3, 2025
  - Open-source LLM evaluation (Mistral, DeepSeek, Llama)
  - GPU infrastructure optimization for model serving
  - vLLM deployment and configuration
- **Fine-tuning & Evaluation** (30 days): September 4–October 3, 2025
  - Domain-specific model fine-tuning for ERP context
  - Performance benchmarking and optimization
  - Model version management and deployment
- **Embedding Optimization & Caching** (20 days): October 4–October 23, 2025
  - Embedding model optimization for semantic search
  - Caching layer implementation for performance

- Vector database integration and tuning

Supporting Agents Development

- **Email Agent Development** (20 days): September 1–September 20, 2025
  - NLP parsing with DocAI integration
  - Intent classification and entity extraction
  - RAG engine integration for context understanding
- **ERP Hook Agent Development** (20 days): September 21–October 10, 2025
  - Natural language to ERP API translation
  - Semantic understanding for equipment queries
  - Response formatting and user interaction
- **Logging & Observability Agent Hooks** (15 days): October 11–October 25, 2025
  - Comprehensive logging integration
  - Performance metrics collection
  - Operational monitoring and alerting

11.2.4 Phase 4: Integration and Quality Assurance (March 2026–October 2026)

Full Agent Integration (30 days)

- **Duration**: March 15–April 13, 2026
- **Dependencies**: All individual agents completed
- **Deliverables**:
  - End-to-end workflow integration testing
  - Cross-agent communication validation
  - Performance optimization across the entire system
  - Integration with external ERP systems

Compliance & Diagram Edge Testing (30 days)

- **Duration**: April 14–May 13, 2026
- **Focus Areas**:
  - Complex diagram parsing scenarios with edge cases
  - Multi-standard compliance validation testing

- Performance testing with large document sets
- Accuracy validation against manual processes

Stability & Performance Testing (30 days)

- **Duration**: May 14–June 12, 2026
- **Testing Scope**:
  - Load testing with concurrent user scenarios
  - Memory and resource utilization optimization
  - Failure recovery and disaster recovery testing
  - Security penetration testing and vulnerability assessment

Buffer & Customer Adaptations (30 days)

- **Duration**: June 13–July 12, 2026
- **Activities**:
  - Customer-specific configuration and customization
  - Documentation finalization and user training materials
  - Deployment preparation and rollout planning
  - Final optimizations based on user acceptance testing

11.3 Risk Mitigation and Contingency Planning

11.3.1 Technical Risks

- **LLM Performance Issues**: Parallel evaluation of multiple open-source models with fallback options
- **Diagram Parsing Accuracy**: Extended research phase with multiple VLM approaches
- **Integration Complexity**: Modular development approach with well-defined interfaces
- **Hardware Delays**: Early procurement with backup supplier relationships

11.3.2 Schedule Risks

- **Parallel Development Streams**: Minimize critical path dependencies
- **Buffer Time Allocation**: 30-day buffer at project end for unforeseen issues

- **Milestone-based Validation**: Regular checkpoint reviews and scope adjustments
- **Resource Flexibility**: Cross-training team members across multiple components

11.4 Success Metrics and Deliverables

11.4.1 Phase Gate Criteria

- **Phase 1**: Infrastructure 99.9% uptime, full observability coverage
- **Phase 2**: API response times $<$ 200ms, successful load balancing
- **Phase 3**: Agent accuracy $>$ 95%, performance benchmarks met
- **Phase 4**: End-to-end workflows validated, customer acceptance achieved

11.4.2 Final Deliverables

- **Production-Ready System**: Fully deployed AI Agent Module with all specialized agents
- **Documentation Suite**: Complete technical documentation, user guides, and operational procedures
- **Training Materials**: Comprehensive training program for administrators and end users
- **Support Framework**: Incident response procedures, maintenance guidelines, and upgrade pathways

Chapter 12

Conclusion

The AI Agent Module for the Prowiz ERP System represents a comprehensive, modular, and scalable solution for automating process design, compliance, document generation, diagram parsing, email processing, and ERP interaction. This Software Design Document has detailed the architecture, component responsibilities, integration boundaries, infrastructure, deployment strategies, and implementation plan.

With a robust microservices-based architecture, Kubernetes orchestration, and Infrastructure as Code, the system is designed for high availability, security, and operational excellence. The phased implementation plan ensures risk mitigation, quality assurance, and alignment with business objectives.

The next steps involve executing the outlined development phases, rigorous testing, and seamless deployment, followed by continuous monitoring and iterative improvements. Successful implementation will streamline workflows, enhance accuracy, and deliver significant value to all stakeholders within the ERP ecosystem.

Appendix A

Glossary

AI Agent Module	A modular subsystem providing specialized AI-powered agents for process automation, compliance, document generation, diagram parsing, email processing, and ERP interaction within the Prowiz ERP System.
API	Application Programming Interface; a set of protocols for building and integrating application software.
API Gateway	A service that provides a single entry point for API requests, routing them to appropriate backend services.
ArgoCD	A declarative, GitOps continuous delivery tool for Kubernetes.
CI/CD	Continuous Integration/Continuous Deployment; automated processes for building, testing, and deploying code.
Compliance Agent	Specialized agent responsible for validating data against business rules and regulatory standards.
CPU	Central Processing Unit; the primary component of a computer that performs calculations.
CUDA	Compute Unified Device Architecture; a parallel computing platform by NVIDIA for GPU acceleration.
Dead Letter Queue	A message queue for storing messages that cannot be processed successfully.
DocAI	Document AI; tools or services for automated document analysis using AI.
Document Generation Agent	Agent that generates narrative and structured documents using LLMs and ERP data.
DSL	Domain-Specific Language; a computer language specialized to a particular application domain.

Embedding A vector representation of data (such as text) used for semantic search and similarity.

ERP Enterprise Resource Planning; integrated management of main business processes, often in real-time.

ERP-Chat Agent Agent that enables natural language querying and interaction with ERP systems.

Faiss Facebook AI Similarity Search; a library for efficient similarity search and clustering of dense vectors.

GPU Graphics Processing Unit; specialized hardware for parallel processing, often used in AI workloads.

Helm A package manager for Kubernetes applications.

HMBT Heat and Material Balance Table; a process engineering document.

IaC Infrastructure as Code; managing and provisioning computing infrastructure through machine-readable definition files.

Jaeger An open-source distributed tracing system.

JSON JavaScript Object Notation; a lightweight data-interchange format.

Kubernetes An open-source platform for automating deployment, scaling, and management of containerized applications.

LangChain An open-source framework for developing applications powered by language models.

LangGraph A workflow orchestration tool for LLMs.

LangSmith A tool for tracing and debugging LLM applications.

Llama A family of open-source large language models by Meta.

Loki A log aggregation system designed for efficiency and scalability.

LLM Large Language Model; a machine learning model trained on vast amounts of text data for natural language processing tasks.

LLM Runtime The infrastructure and environment for serving and running LLM inference workloads.

Message Queue A communication method for exchanging messages between services asynchronously.

Mistral An open-source large language model.

MongoDB A NoSQL, document-oriented database.

NLP Natural Language Processing; a field of AI focused on the interaction between computers and human language.

OCR Optical Character Recognition; technology for converting images of text into machine-encoded text.

PDF Portable Document Format; a file format for capturing and sending electronic documents.

PFD Process Flow Diagram; a type of flowchart that illustrates the relationships between major components in a process.

P&ID Piping and Instrumentation Diagram; a detailed diagram showing the piping and related components of a physical process flow.

Prometheus An open-source monitoring and alerting toolkit.

Qdrant An open-source vector database for storing and searching embeddings.

RAG Retrieval-Augmented Generation; an AI technique combining information retrieval with generative models.

RBAC Role-Based Access Control; a method of regulating access to computer or network resources.

Redis An in-memory data structure store, used as a database, cache, and message broker.

REST Representational State Transfer; an architectural style for designing networked applications.

SLA Service Level Agreement; a contract that defines the level of service expected from a service provider.

SQL Structured Query Language; a standard language for managing and manipulating databases.

Terraform An open-source IaC software tool.

Tesseract An open-source OCR engine.

Vector Database A database optimized for storing and searching vector embeddings.

vLLM An open-source fast and memory-efficient LLM inference engine.

VM Virtual Machine; an emulation of a computer system.

VLM Vision Language Model; a model that processes both visual and textual data.

Workflow Orchestrator A component that manages the execution order and dependencies of tasks or services.