### Updated Version of Orgo v2 Blueprint Section 1 - System Architecture

#### Section 1: System Architecture (Detailed and Specific)

This section defines the structure of Orgo’s communication platform, focusing on its components, their interactions, and the workflow logic. It provides a precise guide for implementing each module, ensuring modularity, offline capability, and role-based routing. The document has been updated to incorporate a generalized task handling system that replaces domain-specific task modules with a unified, metadata-driven approach.

### 1.1 Purpose of System Architecture

Objective: To create a structured, secure, and efficient platform for email-based communication workflows.

Outcome:

* A system capable of routing tasks dynamically based on metadata attributes.
* Generalized task management ensuring scalability, modularity, and efficient workflows.
* A unified task handler to simplify task logic across domains and eliminate redundancies.

### 1.2 High-Level Architecture

The updated architecture replaces domain-specific task handlers with a centralized Task Management Module. The Task Management Module dynamically adapts to all task types based on database-driven metadata. The flow remains modular and scalable while simplifying logic.

Core Components:

1. Email Reception:
   * Emails are received, parsed, and analyzed for actionable data.
2. Unified Task Handler:
   * Located at /core\_services/task\_handler.py.
   * Handles all task types using attributes like type, priority, and metadata.
3. Database:
   * Stores tasks, workflows, logs, and configurations.
   * Dynamically supports new task types by updating metadata without code changes.
4. Offline Sync Module:
   * Provides task handling offline using SQLite, syncing to PostgreSQL when online.

### 1.3 Component Breakdown

1. Email Server:

* Role: Handles incoming and outgoing emails.
* Protocols:
  + SMTP for sending.
  + IMAP/POP3 for receiving.
* Integration:
  + Connects to existing infrastructure securely using TLS.

2. Email Parser:

* Role: Extracts actionable data from emails.
* Functions:
  + Extracts metadata (e.g., sender, subject, keywords).
  + Detects and stores attachments.
* Technology:
  + Python libraries: imaplib, smtplib, email.

3. Unified Task Handler:

* Role: Processes all tasks dynamically based on metadata.
* Functions:
  + Determines task behavior based on type and metadata.
  + Routes tasks dynamically and logs their progress.
* Example:
  + Maintenance Task: metadata.subtype = plumbing.
  + HR Task: metadata.subtype = conflict\_resolution.
* Technology:
  + Python with dynamic dispatch logic.

4. Database:

* Role: Centralized data storage for all workflows and tasks.
* Types:
  + PostgreSQL for scalable deployments.
  + SQLite for offline operations.
* Data Stored:
  + Tasks with type, metadata, and status.
  + Workflow rules and logs.

5. Action/Response System:

* Role: Automates responses and updates workflows.
* Functions:
  + Sends notifications or replies.
  + Updates task statuses dynamically.
* Technology:
  + Python libraries: smtplib, json.

6. Offline Sync Module:

* Role: Ensures uninterrupted task handling during connectivity issues.
* Functions:
  + Processes tasks locally with SQLite.
  + Syncs with PostgreSQL upon connectivity restoration.
* Technology:
  + py-outlook for local file handling.

### 1.4 Communication Flow

Step-by-Step Process:

1. Email Reception:
   * An email is received and forwarded to the system.
2. Parsing:
   * Metadata is extracted, and keywords are identified.
3. Task Creation:
   * The Task Handler creates a task entry in the database with relevant attributes:
     + Example: type=maintenance, metadata={"subtype": "plumbing", "priority": "high"}.
4. Routing:
   * The task is routed dynamically based on its attributes.
5. Execution and Updates:
   * Assigned personnel complete the task, updating the status in the system.

### 1.5 Features and Benefits

1. Generalized Task Handling:

* Handles tasks across domains dynamically, eliminating the need for domain-specific handlers.
* Supports scalability by relying on metadata.

2. Modularity:

* Independent components allow seamless upgrades and integrations.

3. Offline Capability:

* Ensures task handling continuity with SQLite for offline operations.

4. Scalability:

* Dynamically supports millions of tasks with PostgreSQL.
* Handles high-volume workflows using Redis or RabbitMQ.

5. Security:

* TLS encryption for data transmission.
* Role-Based Access Control (RBAC) for secure workflows.

### 1.6 Deliverables

1. Architecture Description:
   * Updated to reflect the centralized Task Management Module and dynamic workflows.
2. Component Details:
   * In-depth descriptions of the email parser, unified task handler, and database interactions.
3. Workflow Examples:
   * Dynamic task routing and handling across domains like maintenance, HR, and IT.

### Summary

This updated architecture consolidates task management into a single, centralized Task Management Module. It replaces domain-specific modules with a metadata-driven approach, improving scalability, flexibility, and maintainability. The design ensures that workflows remain robust, modular, and adaptable to changing organizational needs.