Here’s a double-checked, scalable structure and list of additional files required to fully support the organization and its roles, ensuring the system is future-proof while remaining functional for a small team of 12.

---

Scalable File Structure

The file structure is designed to accommodate the current 12-member team while leaving room for expansion as the organization grows or adds new features. Each file supports modularity, extensibility, and scalability.

/orgo/

├── infrastructure/

│ ├── config/

│ │ ├── config.yaml # Global configurations (database, email)

│ │ ├── rules.yaml # Workflow rules and routing logic

│ │ └── logging.yaml # Logging configurations

│ ├── utils/

│ │ ├── backup.py # Database and log backups

│ │ ├── sync.py # Offline synchronization

│ │ └── health\_check.py # System performance monitoring

│ └── setup.py # Initial setup script

├── core\_services/

│ ├── email/

│ │ ├── email\_parser.py # Handles email parsing

│ │ ├── email\_client.py # SMTP/IMAP client

│ │ └── email\_queue.py # Queue system for email workflows

│ ├── workflow/

│ │ ├── rule\_engine.py # Workflow routing and escalation logic

│ │ ├── workflow\_manager.py # Manages active workflows

│ │ └── escalation.py # Handles task escalations

│ ├── database/

│ │ ├── db\_connector.py # Connects to the database

│ │ └── db\_operations.py # CRUD operations

│ └── logging/

│ ├── activity\_logger.py # Logs workflow actions

│ └── security\_logger.py # Logs security events

├── domain\_modules/

│ ├── maintenance/

│ │ ├── maintenance\_workflow.py # Maintenance-specific workflow logic

│ │ ├── templates/

│ │ │ └── maintenance\_email.html

│ │ └── rules/

│ │ └── maintenance\_rules.yaml

│ ├── hr/

│ │ ├── harassment\_workflow.py # Sensitive workflow logic

│ │ ├── templates/

│ │ │ └── harassment\_report.html

│ │ └── rules/

│ │ └── hr\_rules.yaml

│ └── education/

│ ├── education\_workflow.py # Education-specific workflow logic

│ ├── templates/

│ │ └── parent\_teacher\_notification.html

│ └── rules/

│ └── education\_rules.yaml

├── interfaces/

│ ├── api/

│ │ ├── endpoints.py # REST API endpoints

│ │ ├── serializers.py # Validation for API requests

│ │ └── auth.py # Handles authentication and RBAC

│ ├── admin/

│ │ ├── dashboard.py # Admin dashboard logic

│ │ ├── templates/

│ │ │ └── dashboard.html # Admin dashboard interface

│ │ └── static/

│ │ └── styles.css

│ └── web/

│ ├── web\_client.py # Web-based client logic

│ └── templates/

│ └── user\_feedback.html

├── tests/

│ ├── unit/

│ │ ├── test\_email\_parser.py

│ │ ├── test\_rule\_engine.py

│ │ └── test\_db\_operations.py

│ ├── integration/

│ │ ├── test\_workflows.py

│ │ ├── test\_escalation.py

│ │ └── test\_offline\_sync.py

│ └── e2e/

│ ├── test\_maintenance\_workflow.py

│ └── test\_hr\_workflow.py

├── logs/

│ ├── activity.log # Tracks workflow activity

│ ├── error.log # Tracks errors for debugging

│ ├── security.log # Tracks security-related events

│ └── retention\_policy.txt # Log retention policy

├── docs/

│ ├── installation.md # Installation guide

│ ├── workflow\_examples.md # Examples of workflows

│ └── troubleshooting.md # Common issues and fixes

├── static/

│ ├── css/

│ │ └── styles.css # Global CSS for web components

│ └── images/

│ └── logo.png # System logo

├── templates/

│ ├── email/

│ │ ├── task\_assignment.html # Email template for task assignment

│ │ └── escalation\_notification.html

│ └── reports/

│ ├── task\_summary.html # Report template for tasks

│ └── financial\_report.html

├── requirements.txt # Python dependencies

├── main.py # Entry point for Orgo

└── README.md # Project overview

---

Additional Files for Scalability

1. Role-Specific Workflow Rules:

Location: /domain\_modules/<module\_name>/rules/

Ensures each domain module has its own scalable routing rules.

Example:

- condition: "subject contains 'emergency'"

action:

route\_to: "emergency@organization.com"

attach: ["emergency\_protocol.pdf"]

escalate\_after: "1 hour"

2. Advanced Templates:

Location: /templates/email/ and /templates/reports/

Examples:

Dynamic templates for reports (e.g., weekly\_summary.html).

Customizable escalation notifications.

3. Modular Database Schema:

Location: /infrastructure/scripts/schema.sql

Includes tables for logs, workflows, and role-based permissions.

CREATE TABLE user\_roles (

id SERIAL PRIMARY KEY,

role\_name VARCHAR(50) NOT NULL,

permissions JSONB NOT NULL

);

CREATE TABLE escalations (

id SERIAL PRIMARY KEY,

task\_id INT REFERENCES tasks(id),

escalated\_to INT REFERENCES users(id),

escalation\_reason TEXT

);

4. Scalable Logging Framework:

Location: /core\_services/logging/

Adds modular loggers for activity, error, and security.

def log\_activity(action, details):

log\_entry = {

"timestamp": datetime.now(),

"action": action,

"details": details

}

write\_to\_log("activity.log", log\_entry)

5. Task Management Dashboard:

Location: /interfaces/admin/templates/dashboard.html

Scales to display tasks, priorities, and user activities for all roles.

6. API Endpoint Expansion:

Location: /interfaces/api/

Adds new endpoints for each workflow (e.g., education\_endpoints.py).

---

Scalability Features in the Structure

1. Modularity:

New modules or workflows can be added without disrupting the core functionality.

2. Role-Based Expansion:

Roles like Rookies or Tech Geniuses can have separate templates, rules, and dashboards.

3. Multi-Team Support:

Supports multiple teams (e.g., HR, Maintenance) by modularizing workflows and routing rules.

4. Offline Compatibility:

Includes tools like sync.py to handle .pst or .mbox files for offline synchronization.

5. Monitoring and Alerts:

Logs and dashboards track performance, and alerts are generated for escalations or delays.

---

Next Steps

Develop Templates: Create email and report templates tailored for each role.

Build Rules: Write routing rules for workflows like task escalation and sensitive reports.

Test Scalability: Simulate real-world scenarios with small-scale data to ensure scalability.

Would you like help creating specific examples of any files or features? Let me know!