Checklist for Domain Modules

Domain Modules represent specific workflows and tasks tailored to organizational functions, such as HR, maintenance, or education. This checklist ensures these modules remain focused, reusable, and secure, while aligning with the core system.

---

1. Structure

[ ] Dedicated Subdirectories:

Each domain must have the following subdirectories:

/tasks/: Contains task-related scripts (e.g., task creation, updates, completion).

/templates/: Stores templates for emails, reports, or notifications.

/rules/: YAML files defining workflow routing and escalation rules.

/logs/: Tracks domain-specific actions or errors.

[ ] Common Directory:

Reusable templates or rules shared across multiple domains are placed in /common/.

---

2. Task Management

[ ] Task Types:

Define specific task types relevant to the domain (e.g., maintenance, harassment\_report).

[ ] Lifecycle States:

Ensure all tasks follow a lifecycle (pending, in-progress, completed).

[ ] Validation:

Validate task inputs (e.g., required fields, correct data types) before processing.

[ ] Logging:

Log each task's lifecycle events (e.g., creation, assignment, escalation).

---

3. Workflow Rules

[ ] Rule Format:

Define rules in YAML files with the following structure:

- condition: "subject contains 'urgent'"

action:

route\_to: "supervisor@organization.com"

priority: "high"

escalate\_after: "2 hours"

[ ] Rule Validation:

Validate all rules for required fields and correct formats on load.

[ ] Customizability:

Allow domain-specific overrides for global rules.

---

4. Templates

[ ] Email Templates:

Include placeholders for dynamic content (e.g., <TASK\_ID>, <USER\_NAME>).

Validate that all placeholders are replaced before sending emails.

[ ] Report Templates:

Standardize report formats across domains.

Include metadata (e.g., date, author, task details) for traceability.

[ ] Notification Templates:

Separate notification content by medium (e.g., email, SMS).

---

5. Domain Logs

[ ] Activity Logs:

Track all domain-specific actions (e.g., task assignments, escalations).

[ ] Error Logs:

Record errors specific to the domain (e.g., rule mismatches, failed task updates).

[ ] Retention:

Follow global log retention policies (e.g., 6 months for activity logs).

---

6. Security

[ ] Access Control:

Restrict domain modules to authorized roles or users.

[ ] Data Protection:

Anonymize sensitive information (e.g., reporter names in harassment workflows).

[ ] Input Sanitization:

Sanitize all inputs from external sources (e.g., email payloads).

---

7. Reusability

[ ] Shared Components:

Extract common task logic (e.g., task validation) into reusable functions.

[ ] Common Rules:

Use shared YAML rules for workflows applicable across domains.

[ ] Reusable Templates:

Store shared templates (e.g., notification formats) in /common/templates/.

---

8. Integration

[ ] Core Services Integration:

Ensure tasks and workflows interact seamlessly with core services (e.g., email, logging).

[ ] Validation:

Verify that tasks created in domain modules are compatible with workflow rules.

[ ] Notifications:

Send notifications (e.g., task updates, escalations) to relevant users or groups.

---

9. Testing

[ ] Unit Tests:

Test each task script for correctness and edge cases.

[ ] Integration Tests:

Validate that domain workflows integrate correctly with core services.

[ ] End-to-End Tests:

Simulate real-world scenarios to ensure workflows execute as expected.

---

10. Scalability

[ ] Task Volume:

Ensure task scripts can handle increased volumes without performance degradation.

[ ] Adding New Domains:

New domain modules should follow the same directory structure and checklist.

---

Example Application for a Maintenance Domain

Structure:

/domain\_modules/maintenance/

├── tasks/

│ ├── plumbing\_tasks.py

│ ├── electrical\_tasks.py

│ └── generic\_tasks.py

├── templates/

│ ├── maintenance\_email.html

│ └── maintenance\_report.html

├── rules/

│ └── maintenance\_rules.yaml

└── logs/

└── maintenance\_activity.log

Checklist Applied:

[ ] Tasks (plumbing\_tasks.py, electrical\_tasks.py) validate inputs (e.g., location, urgency) before execution.

[ ] Email templates (maintenance\_email.html) include placeholders like <TASK\_ID> and <ASSIGNED\_USER> and validate replacements.

[ ] Workflow rules (maintenance\_rules.yaml) route tasks based on subject or priority.

[ ] Logs (maintenance\_activity.log) track task creation, assignment, and escalations.

---

Optimized Workflow

1. Begin with Shared Components:

Define reusable templates, rules, and task scripts.

2. Validate Configurations:

Ensure workflow rules and templates are validated before use.

3. Integrate with Core Services:

Test domain-specific workflows with email and logging services.

4. Write Tests for Domain Modules:

Simulate common workflows and edge cases.

---

This checklist ensures domain modules are modular, secure, and seamlessly integrated with Orgo’s core. Let me know if you'd like an example checklist for a specific domain (e.g., HR or education)!