checklist for Core Services without file size or naming considerations, focusing only on functionality, validation, and consistency:

---

Core Services Checklist

1. Email Handling

[ ] Email Parsing:

Extract subject, sender, recipient, and body from incoming emails.

Handle attachments securely (e.g., sanitize filenames and paths).

[ ] Validation:

Check that required fields (e.g., subject, sender) are present.

Reject invalid or malformed emails gracefully.

[ ] Error Handling:

Retry failed email fetches or sends with exponential backoff.

Log detailed error messages for failed operations.

[ ] Security:

Sanitize all email content to prevent injection attacks.

Use secure authentication for IMAP/SMTP connections (TLS/SSL).

---

2. Workflow Management

[ ] Rule-Based Routing:

Load routing and escalation rules from YAML files.

Validate all rules on load (e.g., required fields, correct data types).

[ ] Workflow Execution:

Log every workflow step, including task creation, assignment, and escalation.

Ensure workflows handle both success and failure scenarios.

[ ] Validation:

Ensure workflows match predefined schemas before execution.

Handle unexpected inputs gracefully (e.g., invalid task data).

[ ] Escalations:

Automatically escalate overdue tasks to higher authorities.

Notify relevant stakeholders during escalations.

---

3. Task Management

[ ] Task Lifecycle:

Define task states (e.g., pending, in-progress, completed) and transitions.

Automatically update task states based on workflow outcomes.

[ ] Notifications:

Send notifications for task updates (e.g., creation, completion, escalation).

Allow for multiple notification channels (e.g., email, SMS).

[ ] Error Handling:

Log and retry failed task operations (e.g., notifications not sent).

Provide actionable error messages for debugging.

---

4. Database Operations

[ ] Connection Management:

Establish connections to both PostgreSQL (online mode) and SQLite (offline mode).

Support reconnections on failure with retries.

[ ] CRUD Operations:

Implement reusable functions for Create, Read, Update, and Delete.

Ensure all queries are parameterized to prevent SQL injection.

[ ] Validation:

Validate all database inputs (e.g., required fields, correct data types).

Handle empty or missing query results gracefully.

---

5. Logging

[ ] Activity Logs:

Record user and system actions (e.g., task creation, workflow execution).

Include timestamps and unique identifiers for all log entries.

[ ] Error Logs:

Capture errors with severity levels (INFO, WARNING, ERROR).

Ensure logs provide actionable details (e.g., source of the error, stack trace).

[ ] Security Logs:

Log access attempts (e.g., successful and failed logins).

Track sensitive actions (e.g., data anonymization, escalations).

[ ] Retention:

Enforce log retention policies (e.g., delete logs older than 6 months).

---

6. Validation

[ ] Configuration Validation:

Validate all loaded configurations (e.g., email, database, workflows) for required keys and correct data types.

Provide clear error messages for missing or invalid keys.

[ ] Input Validation:

Sanitize and validate all inputs to workflows, tasks, and database operations.

[ ] Output Validation:

Ensure outputs (e.g., parsed email data, workflow results) conform to expected schemas.

---

7. Security

[ ] Access Control:

Ensure only authorized modules or users can access core services.

Validate API tokens or credentials for all external requests.

[ ] Data Encryption:

Encrypt sensitive data (e.g., passwords, API keys) both in transit and at rest.

[ ] Sanitization:

Sanitize all external inputs (e.g., email content, API payloads) to prevent injection attacks.

[ ] Audit Trails:

Maintain detailed logs of all sensitive actions for auditing purposes.

---

8. Testing

[ ] Unit Tests:

Write tests for all core functions (e.g., email parsing, task lifecycle).

[ ] Integration Tests:

Test interactions between email, workflows, tasks, and databases.

[ ] Mocking:

Use mock services for external dependencies (e.g., email servers, databases).

[ ] Edge Cases:

Test edge scenarios (e.g., missing email fields, invalid task data).

---

Optimized Workflow

1. Begin with Core Functionality:

Focus on implementing core logic for each module (e.g., email handling, workflows).

2. Iterate with Tests:

Write tests alongside each feature to catch errors early.

3. Integrate Modules Gradually:

Test each interaction (e.g., email → workflow → database) incrementally.

4. Enforce Validation:

Validate inputs, configurations, and outputs rigorously at every stage.

---

This checklist ensures that core services are robust, secure, and scalable while maintaining a clear focus on validation and error handling. Would you like help applying this checklist to a specific module (e.g., email handling or workflows)?