next8n

SOP: TECHNICAL LEAD

Workflow Automation Delivery Framework

ENTERPRISE EDITION

Version: 2.0

Date: December 28, 2025

Author: Mirza Iqbal

Contact: mirza.iqbal@next8n.com

Table of Contents

Quality Assurance

Code Review Checklist

QA Testing Oversight

Technical Support for Developers

Daily Sync

Unblocking Issues

Client Technical Interactions

Technical Calls

Explaining Technical Concepts

Production Oversight

Monitoring Alerts

Production Changes

Knowledge Management

Technical Best Practices

Team Learning

Metrics

Technical Metrics

Best Practices

SOP: Technical Lead

Standard Operating Procedure for Architecture, Oversight & Quality

Role Overview

Title: Technical Lead / Solutions Architect
Reports To: Agency Owner / Technical Director
Receives From: Project Manager
Coordinates With: Developer(s)
Interfaces With: Client (for technical discussions)

Primary Objective:
Design technical solutions, oversee development quality, and ensure successful delivery.

Daily Workflow

Morning Routine (30 min)

```
Review developer progress
Check for blockers/questions
Review QA queue
Check production alerts
Prioritize technical decisions
```

Core Activities

```
1. ARCHITECTURE & DESIGN (2-3 hours)
    Design new solutions
    Technical discovery
    Integration research
    Proof of concepts

2. OVERSIGHT & REVIEW (2-3 hours)
    Code/workflow review
    Developer support
    Problem solving
    Quality assurance

3. CLIENT TECHNICAL (1 hour)
    Technical calls
    Complex explanations
    Credential guidance

4. DOCUMENTATION & PLANNING (1 hour)
    Technical documentation
    Architecture records
    Best practices
```

Technical Discovery

Pre-Build Analysis

```
FOR EACH NEW PROJECT:

1. REQUIREMENTS REVIEW
    Read scope of work thoroughly
    Identify all integrations needed
    List data flows
    Note complexity factors
    Identify risks

2. INTEGRATION RESEARCH
   For each integration:
    API documentation reviewed
    Authentication method understood
    Rate limits noted
    Capabilities/limitations documented
    n8n node exists/needs custom

3. ARCHITECTURE DESIGN
    Workflow structure mapped
    Data flow diagrammed
    Error handling planned
    Logging strategy defined
    AI components designed

4. EFFORT ESTIMATION
    Hours by component
    Risk factors added
    Buffer included
    Timeline realistic
```

Architecture Documentation

```
# [Project Name] - Technical Architecture

## Overview
[Brief description of the solution]

## Workflow Structure

### Workflow 1: [Name]
```

Trigger Processing AI Output

```
Components:
- Trigger: [Type, source]
- Processing: [Logic description]
- AI: [Model, purpose]
- Output: [Actions taken]

### Workflow 2: [Name]
[Same structure]

## Integrations

| Service | Purpose | Node | Auth Type |
|---------|---------|------|-----------|
| [Service] | [Purpose] | [Node name] | [OAuth/API Key] |

## Data Flow
```

[Source] [Processing] [Destination]
↓
[Logging/Monitoring]

```
## Error Handling Strategy
- Retry logic: [Details]
- Fallback behavior: [Details]
- Notification: [Method]
- Logging: [Where]

## AI Components

### [AI Component Name]
- Model: [GPT-4, Claude, etc.]
- Purpose: [What it does]
- Prompt strategy: [Brief description]
- Expected output: [Format]
- Guardrails: [Safety measures]

## Security Considerations
- [Security item 1]
- [Security item 2]

## Known Limitations
- [Limitation 1]
- [Limitation 2]

## Risks & Mitigations
| Risk | Likelihood | Impact | Mitigation |
|------|------------|--------|------------|
| [Risk] | [H/M/L] | [H/M/L] | [Action] |
```

Developer Assignment

Task Breakdown

```
PROJECT TASK BREAKDOWN:

Project: [Name]
Developer: [Assigned]
Start: [Date]
Target: [Date]

TASKS:

1. Environment Setup
   - n8n access configured
   - Credentials connected
   - Test data available
   Est: [X hours]

2. Workflow 1: [Name]
   - Build trigger
   - Core logic
   - Output actions
   - Error handling
   Est: [X hours]

3. Workflow 2: [Name]
   [Same breakdown]
   Est: [X hours]

4. AI Components
   - Prompt development
   - Testing/tuning
   - Guardrails
   Est: [X hours]

5. Integration & Testing
   - End-to-end testing
   - Edge case testing
   - Documentation
   Est: [X hours]

TOTAL ESTIMATE: [X hours]
BUFFER (20%): [X hours]
```

Developer Briefing

```
BRIEFING CHECKLIST:

 Share architecture document
 Walk through requirements
 Explain key decisions
 Highlight risks/complexities
 Answer questions
 Confirm understanding
 Set first checkpoint
```

Quality Assurance

Code Review Checklist

```
WORKFLOW REVIEW:

STRUCTURE:
 Workflow well-organized
 Logical node ordering
 No orphan nodes
 Subworkflows used appropriately

NAMING:
 All nodes clearly named
 Naming convention followed
 Sticky notes explain logic

ERROR HANDLING:
 Try/catch patterns used
 Fallback logic present
 Graceful degradation
 Error notifications configured

SECURITY:
 No hardcoded credentials
 Credentials by reference only
 Input validation present
 Output sanitization (if needed)

PERFORMANCE:
 No unnecessary loops
 Efficient data handling
 Rate limits respected
 Timeouts configured

AI (If Applicable):
 Prompts well-structured
 Guardrails in place
 Output parsing robust
 Fallback for failures

DOCUMENTATION:
 Workflow description complete
 Complex logic explained
 Edge cases noted
```

QA Testing Oversight

```
QA SIGN-OFF REQUIREMENTS:

 Developer testing complete
 Test log reviewed
 Edge cases covered
 Error handling verified
 AI quality acceptable
 Performance acceptable
 Security review passed
 Documentation complete

APPROVAL:
 Approved for client QA
or
 Revisions required: [List]
```

Technical Support for Developers

Daily Sync

```
QUICK DAILY SYNC (15 min):

- What did you complete?
- What are you working on?
- Any blockers?
- Need any decisions?
```

Unblocking Issues

```
WHEN DEVELOPER IS STUCK:

1. UNDERSTAND
   "Walk me through what you've tried..."

2. DIAGNOSE
   Review the issue together
   Check logs/executions
   Identify root cause

3. GUIDE
   Explain solution approach
   Point to resources
   Don't just do it for them (usually)

4. VERIFY
   Have them implement
   Review the fix
   Confirm understanding

5. DOCUMENT
   Note the issue/solution
   Add to knowledge base if recurring
```

Client Technical Interactions

Technical Calls

```
WHEN TO JOIN CLIENT CALLS:

 Complex technical discussions
 Architecture explanations
 Troubleshooting sessions
 Handover technical portion
 Security-related topics
 When PM needs support
```

Explaining Technical Concepts

```
COMMUNICATION GUIDELINES:

1. KNOW YOUR AUDIENCE
   - Technical vs non-technical
   - Adjust language accordingly

2. USE ANALOGIES
   - "It's like a digital assistant..."
   - "Think of it as an automated worker..."

3. FOCUS ON OUTCOMES
   - What it does for them
   - Not how it works internally

4. VISUAL AIDS
   - Diagrams
   - Flow charts
   - Screen shares

5. CHECK UNDERSTANDING
   "Does that make sense?"
   "Any questions on that part?"
```

Production Oversight

Monitoring Alerts

```
ALERT RESPONSE:

 CRITICAL (Error rate spike, system down):
1. Acknowledge alert
2. Investigate immediately
3. Disable if causing issues
4. Notify PM and client
5. Fix and restore
6. Post-mortem

 HIGH (Significant errors):
1. Review within 2 hours
2. Assess impact
3. Plan fix
4. Implement
5. Verify

 MEDIUM (Elevated errors):
1. Review within 24 hours
2. Add to task list
3. Schedule fix

 LOW (Minor issues):
1. Log for review
2. Batch with other work
```

Production Changes

```
CHANGE MANAGEMENT:

BEFORE PRODUCTION CHANGES:
 Test in staging/backup first
 Document the change
 Have rollback plan
 Schedule appropriate time
 Notify relevant parties

AFTER CHANGES:
 Monitor closely
 Verify expected behavior
 Document completion
```

Knowledge Management

Technical Best Practices

```
MAINTAIN:

 Best practices document
 Common patterns library
 Troubleshooting guide
 Integration cheat sheets
 Prompt template library
 Security guidelines
```

Team Learning

```
KNOWLEDGE SHARING:

- Weekly tech sync
- Problem-solving sessions
- New tool/technique demos
- Post-mortem reviews
- Documentation updates
```

Metrics

Technical Metrics

```
TRACK MONTHLY:

Quality:
- Bugs found in production
- Rework rate
- QA rejection rate

Performance:
- Average estimation accuracy
- Delivery on-time rate
- Client satisfaction

Team:
- Developer utilization
- Skill development
- Knowledge sharing
```

Best Practices

```
DO:
 Document architecture decisions
 Review all work before client delivery
 Stay current on n8n updates
 Build reusable components
 Share knowledge with team
 Consider security in every design
 Plan for failure/edge cases
 Mentor developers

DON'T:
 Skip code review
 Approve without testing
 Let technical debt accumulate
 Make production changes without testing
 Ignore security concerns
 Overcomplicate solutions
 Skip documentation
 Hoard knowledge
```

Next: See `06-sop-developer.md` for Developer procedures.

This document is confidential and intended for authorized use only.