IT / Support Ticketing System
Project Documentation

Single Organization • Multiple Teams

Version: 1.1
Date: December 22, 2025
Owner: Creative Solutions in Healthcare

1. Executive Summary

This document defines the end-to-end design and delivery plan for an internal IT / Support Ticketing System for a single organization with multiple operational teams (e.g., IT, Facilities, HR, Security, Applications). The system supports multi-channel intake (starting with portal and email), structured triage and routing, SLA tracking, auditable workflows, and reporting. The architecture is designed to scale in ticket volume and team count without major rewrites.

2. Objectives and Success Metrics

2.1 Objectives

Provide a single place for employees to request help and track progress.

Ensure tickets route to the correct team quickly and consistently.

Meet SLA targets for first response and resolution with automatic escalation.

Improve operational visibility via dashboards, backlog aging, and trend reporting.

Enable reliable notifications and a complete audit trail for accountability.

2.2 Success Metrics (examples)

Median time-to-triage < 15 minutes during business hours.

P1 first response SLA compliance >= 95%.

Reduction in duplicate requests (deflection + clear comms) by 10–20% within 90 days.

p95 ticket list load time <= 400 ms under normal load.

Notification delivery (event to send) <= 60 seconds for 99% of events.

3. Scope

3.1 In Scope (Phase 1)

Requester portal: create ticket, view status, add replies, attachments.

Email intake: create or append to tickets from inbound emails.

Agent console: triage, assign, collaborate (internal notes), resolve/close.

Team-based routing and assignment (rules + optional round-robin).

SLA: first response + resolution timers with pause rules.

Notifications: email + (optional) Teams/Slack in later phase.

Reporting: team backlog, SLA compliance, basic agent/team metrics.

Audit trail: immutable ticket events and access logs.

3.2 Out of Scope (initially)

Full CMDB / asset management (can integrate later).

Advanced omnichannel (voice, SMS) beyond portal/email (future).

Customer-facing external tenant model (multi-tenant SaaS).

Complex ITIL change/problem modules (future).

4. Stakeholders and Roles

4.1 Primary Roles

Requesters (Employees): submit requests and communicate updates.

Agents: triage and work tickets; may belong to multiple teams.

Team Leads / Managers: manage queues, escalations, reporting.

Admins: configure routing rules, SLAs, teams, permissions, integrations.

System Owners: product owner and technical owner responsible for outcomes.

4.2 RACI (simplified)

Workstream	R	A	C	I
Requirements & scope	Product Owner	System Owner	Team Leads	All teams
Architecture & security	Tech Lead	System Owner	Security/Compliance	Team Leads
Implementation	Engineering	Tech Lead	Product Owner	All teams
UAT & acceptance	Team Leads	Product Owner	Engineering	All users
Go-live & ops	Tech Lead	System Owner	Engineering	All teams

5. Assumptions and Constraints

Single organization deployment; teams are internal organizational units.

Phase 1 channels: Web Portal + Email.

Identity via existing SSO (e.g., Azure AD / OIDC).

Attachments stored in object storage with malware scanning.

Business hours and holidays may differ per team (Facilities vs IT vs Security).

Data retention and audit requirements will be defined during discovery.

6. Functional Requirements

6.1 Ticket Lifecycle and Statuses

Recommended statuses (configurable):

New (untriaged)

Triaged (categorized and owned by a team)

Assigned (optional explicit state)

In Progress

Waiting on Requester

Waiting on Vendor

Resolved

Closed

Reopened

6.2 Ticket Fields (minimum)

Subject, description, category/subcategory

Priority (P1–P4) and optional severity

Assigned team, assignee, watchers/followers

Channel (portal/email), requester department/location

Tags and custom fields (phase 2 if needed)

6.3 Intake Channels

6.3.1 Web Portal

Create ticket with validation and category guidance.

View ticket list and details; add replies; upload attachments.

Requester sees only their tickets unless explicitly shared.

6.3.2 Email

Inbound email creates a ticket or appends to an existing ticket (tokenized reply-to).

Support for attachments and email threading.

Auto-reopen logic when requester replies to a resolved ticket (configurable).

6.4 Routing and Assignment

Rule-based team routing (category, keyword, department, location, VIP).

Assignment strategies per team: queue-only, round-robin, skill-based, on-call.

Transfer ticket to another team with audit event and notifications.

Share ticket with another team for collaboration without transferring ownership (optional).

6.5 Collaboration

Public replies visible to requester; internal notes visible to agents only.

@mentions notify users and add them as watchers.

Parent/child ticket linking (optional in phase 2).

6.6 SLA and Escalation

First response SLA and resolution SLA tracked per ticket.

Pause SLA when status is Waiting on Requester/Vendor (configurable).

SLA breach triggers: notify team lead, page on-call, or elevate priority (automation).

6.7 Notifications

Email notifications for requester and assignee/team on key events.

Digest options (daily backlog) for team leads (phase 2).

Optional Teams/Slack integration (phase 2).

6.8 Attachments

Secure uploads, virus scanning, and permission-checked downloads.

Audit downloads (who/when) for sensitive attachments (optional).

6.9 Reporting

Team backlog (open by priority, aging buckets).

SLA compliance (first response + resolution).

Volume trends by category and channel.

Agent workload distribution.

6.10 Administration

Manage teams, membership, schedules, and on-call policies.

Configure categories, routing rules, SLAs, notification templates.

Role management (agent/lead/admin) and team scoping.

7. Non-functional Requirements

7.1 Performance

Ticket list endpoints: p95 <= 400 ms with pagination and indexed filters.

Ticket detail: p95 <= 300 ms excluding large attachments.

Email ingestion to ticket update: <= 2 minutes for 99% of messages.

7.2 Availability and Reliability

No lost notifications: use transactional outbox + retries + dead-letter queue.

Background workers are horizontally scalable.

Graceful degradation: if search index is delayed, system still functions via DB reads.

7.3 Security

SSO (OIDC/SAML), MFA enforced by IdP.

RBAC and team-scoped access checks on every request.

Encrypt data in transit (TLS) and at rest (DB + object storage).

Secrets managed via vault/managed secrets service.

Full audit log for ticket changes and admin configuration changes.

7.4 Compliance / Data Governance

PII minimization: limit free-form sensitive fields; apply redaction guidelines.

Retention policy for tickets, messages, attachments, and audit logs.

Export capability for legal/HR holds (future).

7.5 Observability

Centralized logs, metrics, and tracing (OpenTelemetry).

Dashboards: API latency, error rate, queue lag, outbox backlog, SLA worker lag.

Alerting on failed email ingestion, rising DLQ, SLA breach processing delay.

8. Architecture

8.1 High-Level Components

Clients
- Requester Portal (Web)
- Agent Console (Web)
- Email (Inbound/Outbound)

API Layer
- API Gateway / BFF (auth, rate limits, validation)

Core Application (Modular Monolith in Phase 1)
- Ticket Module (tickets, workflow, assignment)
- Messaging Module (public replies, internal notes, mentions)
- Routing Module (rules, scoring, assignment strategies)
- SLA Module (timers, breach actions)
- Notification Module (email templates, send pipeline)
- Admin Module (teams, roles, config)

Async Backbone
- Queue / Event Bus (jobs: notifications, SLA checks, indexing)
- Outbox Publisher (reliable event publish)

Data Stores
- Primary DB (Postgres recommended)
- Cache (Redis)

- Object Storage (attachments)
- Optional Search (OpenSearch/Elastic in Phase 2/3)

8.2 Reliability Pattern: Transactional Outbox

All state-changing operations write (1) the business data and (2) an outbox row in a single DB transaction. A publisher process reads outbox rows and publishes events to the queue. This ensures notifications, SLA updates, and indexing are not lost if the app crashes between DB commit and event publish.

8.3 Deployment Topology (reference)

App service (API + UI) behind load balancer.

Worker service(s) for queue consumption (scale out horizontally).

Managed Postgres, Redis, and object storage.

Email provider (inbound webhook + outbound SMTP/API).

9. Data Model (Logical)

9.1 Core Entities

Ticket: authoritative record of status, priority, assigned team/assignee.

TicketMessage: conversation entries (public/internal).

TicketEvent: immutable audit/event stream for every important change.

Team and TeamMember: membership, roles, scheduling metadata.

SLA Policy + SLA Instance: per-ticket timers and next_due_at scheduling.

Outbox: reliable event publishing.

9.2 Indexing Guidelines (examples)

tickets(status, assigned_team_id, updated_at DESC)

tickets(assignee_id, status, updated_at DESC)

tickets(requester_id, created_at DESC)

ticket_messages(ticket_id, created_at ASC)

sla_instances(next_due_at)

10. API Specification (High-Level)

10.1 Requester APIs

POST /tickets (idempotent)

GET /tickets?mine=true&status= (paginated)

GET /tickets/{id}

POST /tickets/{id}/messages (public reply)

POST /tickets/{id}/attachments

10.2 Agent APIs

GET /tickets?team_id=&assignee_id=&status=&priority=&q=

PATCH /tickets/{id} (safe fields only; use transitions for status)

POST /tickets/{id}/transition (explicit workflow action)

POST /tickets/{id}/notes (internal note)

POST /tickets/{id}/transfer (change owning team)

POST /tickets/{id}/share (collaboration share)

10.3 Admin APIs

POST /teams, PATCH /teams/{id}

POST /routing-rules, PATCH /routing-rules/{id}

POST /sla-policies, PATCH /sla-policies/{id}

POST /roles, PATCH /roles/{id}

GET /reports/*

11. Key Workflows (End-to-End)

11.1 Create Ticket via Portal

1) Requester submits POST /tickets (idempotency-key)
2) Ticket Service creates Ticket + TicketEvent + Outbox in one transaction
3) Outbox Publisher emits TICKET_CREATED
4) Routing worker assigns team/assignee and emits TICKET_ROUTED / TICKET_ASSIGNED
5) Notification worker sends emails

6) SLA worker initializes SLA instance and schedules next_due_at
7) UI shows ticket number and status

11.2 Reply via Email

1) Email provider posts inbound webhook
2) Inbound handler validates org mailbox, parses ticket token
3) Append TicketMessage (public) + TicketEvent + Outbox
4) Notifications to assignee/watchers
5) SLA updates: first response timer stops when agent responds; resolution pauses/unpauses by status

11.3 SLA Breach

1) SLA worker queries sla_instances where next_due_at <= now()
2) Marks breach flags, writes TicketEvent SLA_BREACHED
3) Enqueues escalation actions (notify lead, page on-call, bump priority)
4) Records action results for auditability

11.4 Transfer or Share Between Teams

Transfer:
- Owning team changes; assignment may reset; SLA policy may change.
Share:
- Adds collaborating team; does not change owning team; permission is read or write.
Both operations emit events and notify affected teams.

12. UX Outline

12.1 Requester Portal

Create ticket with guided categories and templates per category.

My Tickets view with filters: Open/Closed, priority, updated time.

Ticket detail with message thread, attachments, and status timeline.

12.2 Agent Console

Team Backlog view with saved filters and quick assignment.

My Work view: assigned tickets, due soon, SLA at risk.

Ticket detail: internal + public tabs, actions panel, assignment controls.

Bulk actions: assign, set priority, add tag (optional).

12.3 Admin Console

Team management, routing rules editor, SLA policy editor.

Notification templates and channel configuration.

Audit log viewer for admin changes.

13. Implementation Plan and Timeline

Reference timeline assumes 12 weeks total with 2-week sprints. Adjust based on team size and parallelization.

13.1 Milestones (Summary)

Milestone	Weeks	Key Deliverables	Exit Criteria
M0: Discovery & Design	1–2	Requirements, workflow, data model, architecture decisions	Signed scope + backlog + architecture
M1: Core Ticketing MVP	3–6	Portal, agent console, ticket lifecycle, team routing, email outbound	Create/assign/reply/resolve works end-to-end
M2: Email Inbound + SLA	7–8	Inbound email threading, SLA timers, breach actions	SLA reports match expected; breach alerts fire
M3: Reporting + Hardening	9–10	Dashboards, audit improvements, performance tuning	p95 targets met; UAT sign-off

M4: Go-Live + Stabilization	11–12	Rollout, training, runbooks, monitoring	Production live; support processes stable

13.2 Sprint Plan (Detailed)

Sprint 1 (Weeks 1–2): Discovery and Foundations

Finalize requirements, categories, priority definitions, and SLAs per team.

Define workflows and permissions; draft UI wireframes.

Set up repo, CI/CD, environments, auth (SSO), database migrations.

Implement Ticket + Message core schema and basic API skeleton.

Sprint 2 (Weeks 3–4): Ticketing Core + Agent Console

Ticket creation (portal) and ticket list/detail views.

Agent console backlog, assignment, status transitions.

Internal notes vs public replies; watchers.

Outbox + queue wiring; email outbound notifications (basic).

Sprint 3 (Weeks 5–6): Routing + Attachments + Audit

Routing rules engine (team routing + assignment strategies).

Attachment upload/download with malware scanning integration.

TicketEvent audit stream for all state changes.

Basic admin: teams, membership, categories, routing rules CRUD.

Sprint 4 (Weeks 7–8): Email Inbound + SLA Engine

Inbound email parsing and ticket threading tokens.

SLA policy model + per-ticket SLA instance creation and updates.

Breach detection worker + escalation actions.

Edge cases: reopen on reply, vendor waiting, transfer effects.

Sprint 5 (Weeks 9–10): Reporting + Performance Hardening

Dashboards (team backlog aging, SLA compliance, volume trends).

DB index tuning and caching for ticket list endpoints.

Rate limiting, idempotency keys, concurrency control (version).

Security review: access checks, audit coverage, retention plan draft.

Sprint 6 (Weeks 11–12): UAT, Rollout, and Stabilization

User acceptance testing with each team; fix defects.

Training materials + admin handbook + support runbooks.

Production cutover plan, monitoring alerts, and post-go-live triage process.

30-day stabilization with weekly review of metrics and routing/SLA tuning.

14. Testing and QA

Unit tests: workflow transitions, routing rules, SLA calculations.

Integration tests: email ingestion, notifications, attachment scanning.

Load tests: ticket list filtering, message posting, burst notifications.

Security tests: authorization matrix validation, audit completeness, OWASP checks.

UAT: scenario-based scripts per team (top 20 request types).

15. Release and Rollout

Pilot with 1–2 teams first (e.g., IT Service Desk + Facilities).

Gradual onboarding of remaining teams after workflow/routing tuning.

Migrate or link existing tickets if required (optional).

Define support model: L1 triage, L2 platform support, escalation path.

16. Operations and Runbooks

16.1 Standard Runbooks

Email ingestion failure: retry, DLQ replay, provider status check.

Notification backlog: scale workers, inspect outbox backlog, rate limits.

SLA worker lag: verify next_due_at queries, worker health, time sync.

Attachment issues: storage permissions, AV scan failures, reprocess flow.

Permission issues: role/team membership verification and audit review.

16.2 Maintenance

Routine DB maintenance (vacuum/analyze), index health, partition rollover.

Retention jobs for old attachments and events (per policy).

Quarterly access review for admins and team leads.

17. Risks and Mitigations

Risk	Impact	Mitigation
Routing rules mis-route tickets	Slow resolution; dissatisfaction	Start simple; log rule decisions; weekly tuning with leads
SLA timers incorrect due to business hours/holidays	False breaches or missed breaches	Centralize calendar logic; test scenarios; UAT with each team
Email threading/tokenization failures	Duplicate tickets; lost context	Embed token in reply-to + body; fallback subject parsing; monitor ingestion errors
Reporting impacts production DB	Performance degradation	Use read replicas or aggregates tables; schedule heavy queries off-peak
Permission bugs expose internal notes	Security incident	Strict visibility flags; automated tests; code reviews; least privilege

18. Appendix

18.1 Priority Definitions (example)

P1: Critical outage / safety issue / widespread impact. Immediate response required.

P2: High impact to a department or critical workflow; timely response.

P3: Standard request or incident with workaround available.

P4: Low priority / informational / cosmetic.

18.2 Glossary (example)

SLA: Service Level Agreement; time-based targets for response/resolution.

Outbox: DB table for reliable event publishing after transactions commit.

DLQ: Dead-letter queue for messages that repeatedly fail processing.