HIGHTOWER.DIGITAL® · CREATE EXCELLENCE!!™

Voice of the Customer — Use Case Report

IT REQUIREMENTS USE CASE · VOC1 · VERSION R1.A

Author: LW Atkinson (CEO/CTO), HighTower Digital Inc.

Date: 06/25/2026 · **Classification:** HDI Confidential

Methods: Agile Scrum + Lean + XP · **Product:** IT Requirements As-A-Service (MSP SLA #1–5)

File: VOC1_UCR1a

2 Voice of the Customer (Development Spec Document)

Voice of the Customer (VOC) translates validated discovery into a **Development Spec**. Using BABOK elicitation, the **Kano model** (must-be, performance, delighter), and **QFD** (Quality Function Deployment) to weight customer attributes against engineering characteristics, VOC produces the story backlog and the layered feature cards developers build from. Stories follow the **INVEST** criteria (Independent, Negotiable, Valuable, Estimable, Small, Testable) with Gherkin acceptance criteria.

2.1 Validated Design Document

2.1.1 STORY INDEX

Story	Title	Layer	Kano	Priority
ST-01	Submit a requirement	Frontend	Must-be	MUST
ST-02	See live SLA countdown	Frontend	Performance	SHOULD
ST-03	Route intake to architect	Middleware	Must-be	MUST
ST-04	Generate spec from intake	Backend	Performance	MUST
ST-05	Suggest acceptance criteria (AI)	Backend	Delighter	COULD

2.1.2 STORY BACKLOG

The backlog is ordered by **WSJF** (Cost of Delay ÷ Job Size). ST-03 and ST-04 carry the highest Cost of Delay because the spec pipeline is blocked without them; ST-05 is a delighter deferred to a later sprint.

2.1.3 STORY LIST (FEATURES / FUNCTIONS)

Each story maps to a Feature (§1.3) and Functions (§1.4). Example — **ST-01** realizes **F-01** via **FN-01**:

> *As a* client analyst, *I want* to submit a requirement with attachments and a stakeholder tag, *so that* it enters the SLA pipeline immediately. **AC:** Given a completed intake form, when I submit, then an item is created, the SLA timer starts, and I receive a ticket ID within 2 seconds.

2.2 Frontend Feature List

Stack: **React 18 + TypeScript**. Representative worked Feature Card below (ST-01); remaining frontend features (live SLA countdown, RTM viewer) follow the identical card template.

2.2.1 FEATURE CARD — REQUIREMENT INTAKE FORM

FRONTEND · FC-FE-01

Feature: Requirement Intake Form (realizes F-01 / ST-01)

(1) Header code

```
// IntakeForm.tsx — header & types
import { useState } from "react";
import { submitRequirement } from "../api/intake";

interface IntakePayload {
  title: string;
  description: string;
  stakeholder: string;
  attachments: File[];
}
```

(2) Function code

```
export function IntakeForm() {
  const [payload, setPayload] = useState<IntakePayload>({
    title: "", description: "", stakeholder: "", attachments: [],
  });
  const [ticketId, setTicketId] = useState<string | null>(null);

  async function handleSubmit(e: React.FormEvent) {
    e.preventDefault();
    if (!payload.title || !payload.description) return; // FN-01 guard
    const res = await submitRequirement(payload);       // starts SLA timer
    setTicketId(res.ticketId);                           // AC: <2s ticket id
  }
  return (
    <form onSubmit={handleSubmit} aria-label="Requirement intake">
      <input value={payload.title}
             onChange={e => setPayload({ ...payload, title: e.target.value })}
             placeholder="Requirement title" required />
      <textarea value={payload.description}
             onChange={e => setPayload({ ...payload, description: e.target.value })}
             placeholder="Describe the business need" required />
      <button type="submit">Submit to SLA pipeline</button>
      {ticketId && <p role="status">Ticket {ticketId} created.</p>}
    </form>
  );
}
```

(3) Footer code

```
// index.ts — barrel export
export { IntakeForm } from "./IntakeForm";
export type { IntakePayload } from "./IntakeForm";
```

(4) Use Case Example — A client analyst opens the portal, fills title + description, tags the VP of Product as stakeholder, and submits; a ticket ID returns instantly and the SLA timer begins.

(5) Step-by-Step Example 1. Navigate to `/intake`. 2. Enter title and description (client-side validation enforces non-empty — FN-01 guard). 3. Tag stakeholder; attach optional files. 4. Click **Submit** → `submitRequirement()` POSTs to middleware. 5. Receive ticket ID `< 2s`; status region announces it for screen readers (usability NFR).

2.3 Middleware Features List

Stack: **Node.js + Express/TypeScript**. The middleware authenticates, starts the SLA timer, and routes intake to an available architect.

2.3.1 FEATURE CARD — INTAKE ROUTER & SLA TIMER

MIDDLEWARE · FC-MW-01

Feature: Intake Router + SLA Timer (realizes F-02 / ST-03)

(1) Header code

```
// intakeRouter.ts — header
import { Router, Request, Response } from "express";
import { startSlaTimer } from "../sla/timer";
import { enqueueForArchitect } from "../queue/architectQueue";
export const intakeRouter = Router();
```

(2) Function code

```
intakeRouter.post("/intake", async (req: Request, res: Response) => {
  const { title, description, stakeholder } = req.body;
  if (!title || !description) {
    return res.status(422).json({ error: "title and description required" });
  }
  const ticketId = `HD-${Date.now().toString(36).toUpperCase()}`;
  await startSlaTimer(ticketId, req.user.slaTier);     // FN-02: 15-min clock
  await enqueueForArchitect(ticketId, { title, description, stakeholder });
  return res.status(201).json({ ticketId });           // AC: <2s
});
```

(3) Footer code

```
// app.ts — mount
import { intakeRouter } from "./routes/intakeRouter";
app.use("/api", intakeRouter);
export default app;
```

(4) Use Case Example — The POST from FC-FE-01 hits `/api/intake`; the router validates, mints a ticket ID, starts the tier-aware SLA timer, and enqueues the item for the next available architect.

(5) Step-by-Step Example 1. Receive POST `/api/intake`. 2. Validate payload (mirror of frontend guard — defense in depth). 3. Generate ticket ID; persist intake. 4. `startSlaTimer(ticketId, tier)` — 15 min (Emergency), 1 h (Partner), 4 h (Advisory). 5. `enqueueForArchitect()` and return `201` with ticket ID.

2.4 Backend Feature List

Stack: **Python (FastAPI)**. The backend turns triaged intake into INVEST-compliant stories with Gherkin acceptance criteria and persists the RTM.

2.4.1 FEATURE CARD — SPEC GENERATOR

BACKEND · FC-BE-01

Feature: Spec Generator (realizes F-03 / ST-04)

(1) Header code

```python
# spec_generator.py — header
from dataclasses import dataclass
from typing import List

@dataclass
class UserStory:
    story_id: str
    role: str
    want: str
    benefit: str
    acceptance: List[str]   # Gherkin lines
```

(2) Function code

```python
def generate_story(intake: dict) -> UserStory:
    """Convert triaged intake into an INVEST-compliant story (FN-03/FN-04)."""
    role = intake.get("stakeholder_role", "user")
    want = intake["description"].strip()
    benefit = intake.get("outcome", "achieve the stated business outcome")
    acceptance = [
        "Given a valid request",
        f"When the {role} performs the action",
        "Then the system fulfills it within the NFR latency budget",
    ]
    return UserStory(
        story_id=f"ST-{abs(hash(want)) % 10000:04d}",
        role=role, want=want, benefit=benefit, acceptance=acceptance,
    )
```

(3) Footer code

```python
# api.py — FastAPI route
from fastapi import FastAPI
from spec_generator import generate_story
app = FastAPI()

@app.post("/spec")
def create_spec(intake: dict):
    story = generate_story(intake)
    return story.__dict__
```

(4) Use Case Example — A queued intake item is posted to `/spec`; the generator returns a structured story with role, want, benefit, and Gherkin acceptance criteria, ready for backlog insertion.

(5) Step-by-Step Example 1. Architect triages intake from the queue (FC-MW-01). 2. POST the triaged object to `/spec`. 3. `generate_story()` builds INVEST story + Gherkin AC. 4. Story persisted; RTM row added linking story → stakeholder → test. 5. Story pushed to Jira (FN-06) — now developer-ready.

2.5 Technical Requirements

2.5.1 BUSINESS CASES

BC-02: *"Each developer-ready story eliminates an average 1.7 clarification cycles, saving ~40 minutes of cross-team coordination per story."*

2.5.2 EDGE CASES

- Intake in a non-English locale → generator routes to localization NFR path.
- Story fails INVEST "Small" check (estimate > 8 points) → auto-flagged for splitting.
- Duplicate intake (same hash) → merged with a traceability note.

2.5.3 USE CASES

UC-02 Generate Development Spec: triaged intake → INVEST story + Gherkin AC → RTM update → Jira push.

2.5.4 'TECHNICAL REQUIREMENTS' CARD

TECH REQ CARD · TR-VOC-01

Title: INVEST compliance gate **Type:** Functional + Process Quality **Statement:** *Every generated story shall satisfy all six INVEST criteria before backlog insertion; stories estimated above 8 points shall be flagged for splitting.* **Acceptance (Gherkin):**

```
Feature: INVEST gate
  Scenario: Oversized story is flagged
    Given a generated story estimated at 13 points
    When it is submitted to the backlog
    Then the system flags it "Too large — split before commit"
```

Priority: MUST · **Verification:** Unit test + PO review

2.5.5 TECH REQUIREMENTS INDEX + PRIORITY

ID	Requirement	Priority	Method
TR-VOC-01	INVEST compliance gate	MUST	XP test-first
TR-VOC-02	Gherkin AC auto-generation	MUST	Scrum/BDD
TR-VOC-03	RTM persistence per story	SHOULD	BABOK trace
TR-VOC-04	Localization routing	COULD	Lean option
TR-VOC-05	AI acceptance-criteria suggestion	WON'T (R1)	Lean experiment

Appendix D — Glossary (Ubiquitous Language)

Term	Definition
Trouble-Ticket	Canonical term for a submitted requirement/incident entering the SLA pipeline.

Term	Definition
Developer-Ready Spec	A specification with API schema, data model, acceptance criteria, and edge cases — zero open interpretation.
SLA Tier	Advisory (4 h), Partner (1 h), Emergency Owner (15 min) response thresholds.
RTM	Requirements Traceability Matrix linking need → story → test.
NFR	Non-Functional Requirement (performance, scalability, reliability, security, etc.).
ADR	Architecture Decision Record — context, decision, consequences.
Bounded Context	DDD boundary within which a domain model and its terms are consistent.
INVEST	Independent, Negotiable, Valuable, Estimable, Small, Testable.
WSJF	Weighted Shortest Job First — Cost of Delay ÷ Job Size.

Appendix E — References

1. ISO/IEC/IEEE 29148:2018 — *Systems and software engineering — Life cycle processes — Requirements engineering.* IEEE SA. https://standards.ieee.org/standard/29148-2018.html
2. ReqView — *ISO/IEC/IEEE 29148 Requirements Specification Templates (StRS/SyRS/SRS/BRS/OpsCon).* https://www.reqview.com/doc/iso-iec-ieee-29148-templates
3. Schwaber, K. & Sutherland, J. — *The 2020 Scrum Guide.* https://scrumguides.org/scrum-guide.html
4. IIBA — *A Guide to the Business Analysis Body of Knowledge (BABOK® Guide v3).* https://www.iiba.org/
5. Blank, S. — *Jobs-To-Be-Done as the Front End of Customer Discovery.* https://steveblank.com/2021/11/04/market-definition-its-the-front-end-of-customer-discovery/
6. Ulwick, T. — *Jobs to Be Done (JTBD): The Original Framework.* Strategyn. https://strategyn.com/jobs-to-be-done/
7. Fitzpatrick, R. — *The Mom Test* (customer discovery interview method).
8. AltexSoft — *Nonfunctional Requirements: Examples, Types and Best Practices.* https://www.altexsoft.com/blog/non-functional-requirements/
9. TestQuality — *Gherkin User Stories & Acceptance Criteria Guide (2026).* https://testquality.com/gherkin-user-stories-acceptance-criteria-guide
10. Boost / PlatinumEdge — *INVEST Criteria for Agile User Stories.* https://www.boost.co.nz/blog/2021/10/invest-criteria
11. Architecture Decision Records. https://adr.github.io/

12. KanbanZone — *Agile Frameworks Compared: Scrum, Kanban, Lean, XP (2024).* https://kanbanzone.com/2024/agile-frameworks-compared/
13. PPM Express — *From RICE to WSJF: Prioritization Techniques.* https://www.ppm.express/blog/13-prioritization-techniques
14. GetProductPeople — *Prioritization Techniques: RICE, MoSCoW, ICE & Kano.* https://www.getproductpeople.com/blog/prioritization-techniques-rice-moscow-ice-kano
15. HighTower.Digital — *MSP SLA #1+5 — IT Requirements As-A-Service.* https://www.hightower.digital/msp-sla-15

Document control. HDI Confidential · Author: LW Atkinson (CEO/CTO) · Version R1.A · 06/25/2026.