WHITE PAPER · MEDICAL DEVICE SOFTWARE (SAMD) USE CASE

From specification to traceable backlog. Why regulated teams still write tickets by hand, and what changes when they don't.

How Sprint-Ready turns medical device design documents into source-cited, deduplicated, audit-recorded work items in Azure DevOps and Jira, and how that approach compares with ALM suites, AI story generators and general-purpose AI assistants.

Version 1.0 Published 24 September 2026 Sprint Ready LTD, Company No. 17375471

Approx. 25-minute read

1 Executive summary

Medical device software teams already write detailed specifications. The regulations they work under (IEC 62304, ISO 14971, ISO 13485 as incorporated into the FDA's Quality Management System Regulation, and EU MDR Annex II) then expect every requirement to be traceable through design, testing and risk control. The step in between, turning the specification into developer work items, is still done by hand in most teams, and it is the step where the traceability chain usually breaks.

The tools on the market mostly sit either side of that step. ALM and eQMS platforms manage, trace and audit requirements once someone has entered them. General-purpose AI assistants will draft tickets on request, but they tend to fill gaps with plausible guesses, which regulated work cannot accept. **Sprint-Ready is built for the step in between:** it reads the team's own design documents and writes granular tickets straight into the Azure DevOps or Jira project the engineers already use, under five engineering constraints:

1. **Source Lock.** Every ticket cites the document section it came from. Where the source is silent, the engine raises an explicit `DATA_GAP` instead of inventing a value.
2. **Dedup Guard.** Every ticket is scored against the team's *live* backlog with two independent similarity signals before anything is created. A merge needs three gates to agree.

3. **An embedded audit trail.** The routing decision, both scores, both thresholds and the confidence tier are written onto the work item itself.
4. **A deterministic preliminary risk band.** Severity × probability is computed in code, never by the model. It fails safe to "unclassified" and is labelled preliminary, for confirmation under the organisation's own ISO 14971 process.
5. **A permanent traceability record.** It is independent of the PM tool, and the full matrix downloads in one click.

770
live backlog items each ticket was checked against in a logged production run (7 Sept 2026)

0
requirements invented where the source was silent in that run

£1.22
billed for 3 traced, deduplicated tickets (2.44 credits at the Self-Serve rate)

24
real work items created in logged September 2026 runs: 11 in Azure DevOps, 13 in Jira

Figures are from Sprint-Ready's own logged production executions against internal test projects. They are listed with execution IDs and work item numbers on the Trust & Compliance page. They are measurements of specific runs, not guarantees for every document.

This paper sets out the regulatory context, the evidence that requirements work is where regulated software projects are most exposed, why a general AI assistant does not solve the problem, and exactly how Sprint-Ready's method works. It then gives a fair comparison with the alternatives, the security model, the economics, guidance for adopting the tool inside a quality management system, and a plain list of limitations.

2 The use case: a SaMD team between specification and sprint

> **Illustrative composite.** The team below is not a Sprint-Ready customer. It is a composite built to make the workflow concrete, and its numbers mirror the defaults in the public calculator on sprintready.dev, which any reader can change.

Consider a 14-person company building a clinical decision-support app classified as software as a medical device. Its software is developed under IEC 62304, with a quality system aligned to ISO 13485, risk management under ISO 14971, and plans for both a US premarket submission and EU MDR technical documentation. Engineering runs Scrum in Azure DevOps; the regulatory lead keeps the design history file and the traceability matrix.

Each month the product and systems engineers produce about eight specification documents: new features, change requests, and updates prompted by risk analysis or usability findings. Each decomposes into about twelve developer tickets. A senior business analyst or product owner turns each document into work items by hand:

- They read the specification section by section and split it into tickets small enough for a sprint.
- They write acceptance criteria that are specific and testable.
- They search the backlog for items that already cover the same ground, from memory and keyword search.
- They record where each ticket came from, often as a document name in the description, sometimes not at all.
- They note the safety relevance for the risk file, if there is time.

At twenty-five minutes a ticket, that is about forty hours a month of senior time, or £2,400 at a £450 day rate, before a line of code is written. The more serious cost appears later. When the team prepares its submission or faces an audit, the regulatory lead reconstructs the chain from requirement to ticket to test to risk control by hand, weeks or months after the tickets were written, from a backlog where some items have been merged, renamed or duplicated and nobody remembers why.

That reconstruction step is what this paper is about. The goal is not simply faster tickets. It is tickets that carry their own traceability from the moment they are created, so the chain never has to be rebuilt.

3 What regulators expect

None of the frameworks below requires a particular tool, and none dictates how tickets are written. All of them expect a traceable, documented path from inputs to implementation to verification, with risk controls visible along it. The summaries below paraphrase the requirements; teams should always work from their own licensed copies of the standards.

IEC 62304 (medical device software lifecycle)

The software development plan must address traceability between system requirements, software requirements, software system testing and the risk control measures implemented in software (clause 5.1.1, as summarised in [4]). The standard in force is IEC 62304:2006 with Amendment 1:2015. A second edition is in committee draft. As reported by the Johner Institute, the draft replaces software safety classes with "process rigor levels" and broadens the standard's scope, and publication is currently forecast no earlier than 2028 [5]. That is a draft, not a requirement, but it signals where scrutiny of AI-assisted development is heading.

ISO 14971 (risk management)

The risk management file must allow each identified hazard to be traced to the implementation and verification of its risk control measures (clause 4.5, as summarised in [4]). Each organisation defines its

own severity and probability scales in its risk management plan. That is why any tool-assigned risk band can only ever be a *preliminary* input.

ISO 13485 and the FDA Quality Management System Regulation

ISO 13485 clause 7.3.2 requires documented methods to ensure design outputs trace to design inputs [4]. Since **2 February 2026**, the FDA's Quality Management System Regulation has incorporated ISO 13485:2016 by reference and now specifically requires risk management [3]. For US-market teams, ISO 13485's design traceability requirements now apply directly through the QMSR.

FDA premarket software documentation (June 2023 guidance)

The FDA's guidance on premarket submissions for device software functions expects a software requirements specification at both the Basic and Enhanced documentation levels. It states that *"each requirement identified in the software requirements specification should be traceable"*, and describes traceability as linking user needs, system and software requirements, design, testing and implemented risk controls [1]. Its older General Principles of Software Validation guidance goes further: *"a software requirements traceability analysis should be conducted to trace software requirements to (and from) system requirements and to risk analysis results"* [2].

EU MDR Annex II (technical documentation)

Annex II requires the technical documentation to identify precisely the controlled documents that evidence conformity with each general safety and performance requirement. It also requires software verification and validation evidence describing the design and development process [6]. The practical effect is the same: an auditor must be able to follow the thread.

> **What this means for tickets.** None of these frameworks asks for tickets directly. But in an Agile team the work item *is* where a software requirement becomes a unit of implementation and test. If the ticket does not carry its source, its risk relevance and a record of how it relates to existing work, the traceability chain has a gap at exactly the point where most of the work happens.

4 Why the problem persists

Requirements errors are the most expensive errors

A widely cited NASA study found that a requirements error that costs one unit to fix during the requirements phase costs 3 to 8 units during design, 21 to 78 during integration and test, and from 29 to more than 1,500 once the system is in operation [8]. PMI research reported that "47 percent of unsuccessful projects fail to meet goals due to inaccurate requirements management" [9].

Software design is a leading cause of device recalls

IEEE Spectrum's analysis of the FDA's database of about 56,000 recall records since 2002 lists software among the biggest recall categories, with "software design far and away the biggest problem" within it [10]. The FDA's own early analysis found that 7.7% of recalls between 1992 and 1998 were software-related, and that 79% of those were caused by defects introduced by changes after initial release [2]. That is exactly where an untraced backlog does most damage.

Traceability failures show up in inspections

Design controls are a persistent source of FDA inspection observations. In the FDA's FY2014 Form 483 data, 515 of 3,740 quality-system observations (about 14%) concerned design controls. The most frequent were design validation, which includes software validation, followed by design changes [11]. Recent warning letters make the pattern concrete:

- **May 2025.** The FDA noted that a requirements document contained obsolete requirements. For a software requirement to warn users of low-quality data, it wrote: "you did not have any testing to verify" it [12].
- **October 2025.** Another letter stated: "Test reports do not confirm that design outputs meet design inputs" [13].

The market mostly manages tickets after someone has written them

Requirements management platforms, traceability tools and ALM suites are built on an unstated assumption: a person has already read the specification, broken it into work items, written the acceptance criteria and entered them. The tools then manage and audit that work, and several now add AI to check quality, suggest links or draft tests (see section 9). The authoring step itself, from a real design document to a backlog of traced tickets in the tool the engineers use, is still largely manual. That is the step Sprint-Ready automates.

5 Why a general AI assistant isn't the answer

The obvious shortcut is to paste the specification into a chat assistant and ask for user stories. Research on large language models suggests the drafts would often be useful: one study found that GPT-4-drafted requirements specifications could match the output quality of an entry-level software engineer [18]. Two problems make this unsafe as a process for regulated work.

1. Models fill gaps with plausible inventions

NIST's generative AI risk profile names this risk "confabulation": confidently stated but erroneous or false content [14]. The size of the risk depends on the task and the model, but even when a model is explicitly told to use only the text it is given, it does not reliably do so:

- On Vectara's public summarisation leaderboard (as of May 2026), where models are told to use only the passage provided, measured hallucination rates ranged from about 1.8% to 24% [15].

- Stanford researchers found that retrieval-grounded legal AI tools still hallucinated on 17% to 33% of queries [16].
- Research published by OpenAI in 2025 argues that models hallucinate partly because training and evaluation reward guessing over admitting uncertainty [17].

In a ticket, a confabulation looks like this. The specification says "lock the account after repeated failed attempts"; the assistant writes "lock after 5 failed attempts". The number is reasonable, and it is not in the source. Once it is in the backlog it gets implemented, tested and shipped as if someone had decided it. **In regulated software, a missing requirement is a question; an invented one is a finding.**

2. Ad hoc use has no traceability, no dedup and uncertain data handling

A chat window does not know the live backlog, does not record where each ticket came from, and leaves no audit record of what was generated or merged. Data handling also depends on the product tier:

- **OpenAI:** content in consumer ChatGPT may be used for training unless the user opts out [19].
- **Anthropic:** since August 2025, consumer Claude users choose whether their chats are used for training. The API and commercial products, including access through Amazon Bedrock, are excluded [20].

Enterprise and API tiers generally exclude customer content from training. But an engineer pasting a design specification into a personal account may not be on one.

So the answer is not "no AI". It is AI constrained by engineering that is designed for the failure modes that matter in regulated work.

6 The Sprint-Ready method

Sprint-Ready is an automated pipeline with five stages. It lands its output where the engineers already work, so there is no new system to adopt and nothing to migrate if a team stops using it.

1. **Ingest.** Word and Excel specifications are uploaded through the control panel. Text is extracted, cleaned of extraction noise and merged into one context. (PDF and flowchart-image paths are built but not yet counted as verified; see section 13.)

2. **Generate.** Anthropic's Claude model, running on Amazon Bedrock, drafts granular tickets under a strict system prompt. Each ticket has a title, a description, acceptance criteria, a list of data gaps, a section-level source reference and a preliminary hazard indication. Optional knowledge-base retrieval adds background on the client's architecture, but source citations only ever point to the document being processed.

3. **Check for duplicates.** Every candidate is scored against the client's live Azure DevOps or Jira backlog, fetched at run time, and against the other tickets in the same batch.

4. **Route.** The candidate is created as a new work item, merged into an existing one with a marker in the title, or skipped as a same-batch duplicate. Low-confidence tickets are tagged `NEEDS-CLARIFICATION`.

5. **Record.** The decision is written onto the work item, and a permanent traceability record is written to Sprint-Ready's own database.

The value is not in any single stage; tools exist that do some of them. It is in five specific engineering constraints applied together.

6.1 Source Lock: if the document doesn't say it, the ticket doesn't either

The generation prompt forbids the model from adding requirements, values or behaviours that are not in the source. Where a ticket needs information the document does not contain, the model has to put it in an explicit **Data Gaps & Blockers** list on the ticket, for example "retry limit not specified in source", rather than choosing a value. Each ticket also carries a section-level source reference (document, section, sentence or diagram node), repeated in a traceability footer structured to support GAMP 5 documentation.

This reverses the incentive described in section 5. A general assistant is rewarded for a complete-looking answer. Sprint-Ready is designed to reward an honest one. A ticket with three data gaps is not a failure: it tells the team exactly which questions to settle with the specification owner before sprint planning. In the logged 7 September production run, no requirement was invented where the source was silent.

> **What Source Lock does not do.** It is a constraint enforced through the prompt and output structure, not a mathematical guarantee. Every generated ticket still needs the same review a team gives any externally drafted requirement. Sprint-Ready removes the blank page, not the engineer's judgement.

6.2 Dedup Guard: checked against the live backlog before anything is created

Many duplicate-detection tools for Jira work *after* an issue exists, or while a person is creating one, and flag possible duplicates for that person to handle. Sprint-Ready runs its check automatically *before* bulk creation, against the backlog as it stands at run time:

- **Signal 1: symmetric similarity.** A character-bigram comparison of the whole candidate against each backlog item (title plus description, with markup stripped).
- **Signal 2: directional containment.** How much of the new ticket's vocabulary already exists in a candidate item, after a light stemmer normalises word forms such as detect/detection/detected.
- **Gates.** A near-identical item (symmetric similarity ≥ 0.95) routes to a merge. Otherwise a containment-based merge needs three conditions at once: containment ≥ 0.85, *and* symmetric similarity of at least 0.40, *and* at least six meaningful words compared.

The design is deliberately conservative, because in a regulated backlog **a wrong merge costs more than a duplicate**. A duplicate is visible and recoverable at grooming. A false merge can quietly absorb a requirement into another item and lose its separate scope, data gaps and acceptance criteria. Merges are non-destructive: they add a `[MERGED]` marker (counted, e.g. `[MERGED x2]`, on repeats) rather than overwriting history. If the live backlog cannot be fetched, the guard logs the failure and falls back to checking within the batch rather than halting the run. The failure is visible in the run log, not hidden.

6.3 The audit trail travels with the ticket

Every routing decision is written onto the work item itself, in the history and discussion of Azure DevOps or the issue record in Jira. The record states what the candidate was compared against, both similarity scores and both thresholds, the matched item if any, and the confidence tier with its reason. A reviewer opens the ticket and sees why it exists in its current form, without needing access to another system and without having to export and cross-reference logs.

6.4 A deterministic, fail-safe preliminary risk band

The model rates each ticket's severity and probability on a fixed scale and writes a hazard rationale. The **risk index and band are then computed in Sprint-Ready's own code**: severity × probability, banded (for example LOW, MEDIUM, HIGH). They are never taken from the model's own arithmetic. An unrecognised or malformed rating fails safe to **UNCLASSIFIED** rather than being forced into a plausible band.

The indication is labelled on the ticket as preliminary and non-certified, for the organisation's own risk management plan to confirm. ISO 14971 leaves each manufacturer to define its own scales, so the value of a tool here is a sensible, consistent starting point on every ticket instead of a blank field. It does not replace the risk team.

6.5 A permanent traceability record and one-click matrix

Each create or merge action writes a record to Sprint-Ready's database containing:

- client and seat;
- pipeline execution ID;
- source filename and section reference;
- ticket title and action taken;
- resulting work item ID and URL;
- matched item and similarity score for merges;
- the token-usage event;
- timestamp.

The record is independent of the PM tool. If a work item is later edited or deleted in Azure DevOps or Jira, the record of what was generated, from which source and when, stays intact. Client and seat

users can filter these records by document, file type, date and seat, and export the whole set as a traceability matrix on demand. That is the document an auditor asks for.

7 Worked example: one specification section, one ticket

The work item below was generated in a logged production run on 7 September 2026, against Sprint-Ready's own Azure DevOps test project. It is reproduced from the real item (Task #1533) and abridged for length.

Source: a test specification section on password reset tokens. **Output:**

```
TASK 1533 Implement password reset token expiration at 30 minutes

Acceptance criteria (8), including:
 · The system rejects any token where elapsed time exceeds exactly 30 minutes
(1800 seconds)
 · When a token is rejected due to expiration, the system returns an explicit
error code or message containing "INVALID_TOKEN"
 · Expired tokens cannot be refreshed or extended; a new password reset
request must be initiated

Data gaps & blockers (4):
 · Token generation mechanism and format not specified (JWT, random string,
signed token)
 · Storage location for token and timestamp not defined
 · Clock synchronisation requirements across distributed systems not specified
 · Behaviour at exact 30:00 boundary not specified (inclusive or exclusive of
the 1800th second)

Dedup Guard audit trail (written onto the work item):
 Decision: POST_CREATE (match basis: none)
 Symmetric similarity: 0.4444 (threshold 0.95)
 Containment score: 0.5185 (threshold 0.85)
 Confidence tier: MEDIUM: 4 data gaps flagged against 8 acceptance criteria,
reviewable but not blocking
 Generated: 2026-09-07T11:18:01Z
```

Three things to note:

1. **The 30-minute value is in the ticket because it is in the source.** The boundary behaviour at exactly 30:00 is not in the source, so it appears as a question rather than an assumption.
2. **The guard compared this ticket against 770 live backlog items**, found nothing close enough, and says so with numbers.
3. **The confidence tier explains itself**, so the reviewer knows what to look at first.

A Jira issue generated on 14 September (SCRUM-17, role-based access to an audit-log export) shows the risk side. It carries a severity (SERIOUS) and probability (OCCASIONAL) with a hazard rationale, a code-computed *indicative* band of HIGH (score 12), labelled as preliminary for the organisation's own classification under ISO 14971, and a traceability footer naming the source document and the user story it came from.

8 Production evidence

Sprint-Ready describes a capability as live only once it has cleared a real execution against a real external system. The runs below were against Sprint-Ready's own test projects, not customer accounts. Each is itemised with IDs on the Trust & Compliance page [38].

Date (2026)	What was proven	Evidence
7 Sept	End-to-end generation on AWS Bedrock with live-backlog dedup	Execution #10914: 3 tickets; each checked against 770 live Azure DevOps items with zero fetch errors; billed 2.44 credits (£1.22 at the Self-Serve rate); work items #1532–#1534
9 Sept	Word specification ingestion at batch scale	One .docx produced 8 Azure DevOps work items (#1535–#1542) in a single run
9 Sept	Hazard and risk classification	All 8 tickets in the smoke test carried populated hazard fields; code-computed bands
9 Sept	Traceability matrix export	Pre-deployment self-testing caught and fixed three defects before release; live route verified
14 Sept	Jira create and traceability routing	13 Jira issues across two runs (SCRUM-5 to SCRUM-17), including a Jira-specific traceability defect found and fixed the same day
24 Sept	Failure alerting	A deliberate failure drill confirmed a failed production run automatically emails the operator

Sprint-Ready records defects found and fixed during verification, not only passes. We consider that part of the evidence.

9 How it compares

A fair comparison starts with the fact that most of these products are capable tools built for a different job. They are not alternatives to one another in every respect, and several now use AI in ways that partly overlap with Sprint-Ready. The summary below is based on each vendor's own public pages as checked on **24 September 2026**. Products change quickly, and a marketing page is not proof of behaviour either way. "Not described" means we could not find the capability on their public pages, not that it cannot exist.

9.1 The landscape by category

Category	Examples	What they do well (per their public pages)	Where Sprint-Ready differs	Pricing model
Regulated eQMS / ALM for medtech	Ketryx, Matrix Req, Greenlight Guru	• System of record for design controls, with IEC 62304 and ISO 14971 traceability. • Ketryx describes AI that drafts requirements, tests and specifications, automatically detects conflicting or duplicate requirements, and integrates with Jira and Azure DevOps [25]. • Matrix Req offers AI-assisted creation and two-way Jira/Azure DevOps integration [31]. • Greenlight Guru offers multi-level traceability with Jira integration [32].	Sprint-Ready authors granular developer tickets from your existing design documents in bulk. Every ticket is source-locked and deduplicated against the live engineering backlog before creation, and the decision is recorded on the item. It complements a system of record rather than replacing one.	Ketryx: free for pre-market companies under $2M raised, otherwise contact sales [26]. Matrix: bespoke. Greenlight: demo-first.
Enterprise ALM suites	Jama Connect, Siemens Polarion, PTC Codebeamer, Perforce ALM	• Mature requirements, test and risk management at scale. • Jama Connect Advisor scores requirement quality, suggests missing trace links and parses PDFs into structured requirements. Further capabilities, including requirement generation, are described as on the way [27]. • Polarion X offers a paid AI add-on that includes similarity analysis of work items [29]. • Codebeamer AI checks requirement quality and generates test cases [30].	No change of platform. Engineers stay in Azure DevOps or Jira, with no rip-and-replace. Output is ready-to-work sprint tickets rather than requirement objects inside a separate suite.	Quote-based or contact sales [28] [39]
AI inside Azure DevOps	Copilot4DevOps (Modern Requirements); GitHub Copilot for Azure Boards	• Copilot4DevOps elicits, analyses and converts requirements and generates documents inside Azure DevOps, with published per-user pricing [33]. • GitHub Copilot turns an existing work item into a draft pull request [34].	Neither describes pre-creation dedup against the live backlog, section-level source locking, or a decision record written onto each item. GitHub Copilot starts from a work item; Sprint-Ready produces the work items.	Copilot4DevOps: $30–$40 per user per month (annual), Enterprise custom

AI inside Jira	Atlassian Rovo; marketplace duplicate apps (e.g. Duplicate AI); AI story generators (e.g. StoriesOnBoard)	• Rovo captures work from Confluence and other sources and can break work down into tasks, and is included in paid cloud plans [35]. • Duplicate apps find similar issues, including while an issue is being created [36]. • Story generators draft stories and acceptance criteria from a project description [37].	Regulatory traceability, source-locked gap flagging and a recorded dedup decision are not described. Sprint-Ready also works the same way in Azure DevOps.	Mostly published per-user pricing
General AI assistants	ChatGPT, Claude, Microsoft 365 Copilot (used ad hoc)	Fast, flexible drafting. Enterprise and API tiers generally exclude customer content from training.	No backlog awareness, no source lock, no traceability record. Consumer tiers may train on content unless opted out [19][20].	Per user
Manual authoring	A senior BA or product owner	Judgement, context and accountability.	Sprint-Ready keeps the judgement (review) and removes the authoring time and the after-the-fact reconstruction.	About £60 an hour of senior time

9.2 The combination that matters

Each of Sprint-Ready's individual mechanisms has a relative somewhere in the market:

- AI drafting exists (Ketryx, Copilot4DevOps, Rovo, story generators).
- Duplicate detection exists (Ketryx for requirements, Polarion similarity analysis, Jira marketplace apps).
- Traceability exists in every eQMS.

What we did not find on any public page we reviewed is **all of the following in one automated step, inside the team's existing Azure DevOps or Jira project:**

Capability	Manual	Regulated eQMS / ALM	Enterprise ALM	AI in ADO / Jira	General AI	Sprint-Ready
Writes developer tickets from your existing design documents, in bulk	By hand	Partly (AI drafting within the eQMS)	Partly (parsing, upcoming generation)	Partly (from descriptions or pages)	On request	Yes
Flags missing information instead of inventing it	If disciplined	Quality checks flag incomplete requirements, after authoring	Quality checks flag ambiguous or incomplete requirements, after authoring	Not described	Not by default; unreliable even when instructed	Yes: data gaps on every ticket
Section-level source citation on each ticket	If disciplined	Via manual linking	Via manual or suggested links	Not described	No	Yes: automatic
Checks the live backlog for duplicates *before* creating	Memory and search	Within requirements	Similarity analysis (add-on)	Apps flag similar issues	No	Yes: two signals, three gates
Decision and scores recorded on the work item itself	No	Not described	Not described	Not described	No	Yes
Deterministic, fail-safe preliminary risk band per ticket	Separately	Risk module (manual)	Risk module (manual)	Not described	No	Yes: computed in code
Engineers stay in Azure DevOps / Jira	Yes	Yes, via integration	Usually a separate suite	Yes	Copy and paste, or connectors	Yes
Full price published; try without a sales call	n/a	Free early tier (Ketryx); otherwise sales	Quote-based	Mostly	Yes	Yes: 100 free credits

Sources are cited in the table in 9.1. "Not described" reflects public pages on 24 Sept 2026 only. Sprint-Ready is not a replacement for an eQMS or design history file system. Because it writes into Azure DevOps or Jira, teams

whose eQMS or ALM already integrates with those tools keep that system of record unchanged.

10 Security and intellectual property

A medical device specification is core intellectual property. Sprint-Ready's handling of it, stated precisely:

- **Where it runs.** The application and database run on AWS in the EU (Stockholm, eu-north-1) region. Model inference runs on Amazon Bedrock through its **EU cross-region inference profile**. AWS documents that this profile routes requests only to AWS Regions within the EU geography, over the AWS network [23].
- **No training on your content.** AWS states that Bedrock inputs and outputs are not shared with any model provider, that neither AWS nor third-party model providers use them to train models, and that model providers have no access to customer prompts and completions [21][22]. Separately, Anthropic's own Commercial Terms bar it from training models on commercial customers' content [24]. We quote these commitments as published and do not claim anything stronger, such as instant deletion.
- **Tenant isolation.** Every request identifies the user from a cryptographically signed server-side session, never from an ID a browser could supply. Isolation is enforced in application code on every request. There is not yet a separate database-level isolation layer, which we state rather than imply otherwise.
- **Credentials.** Azure DevOps and Jira tokens are encrypted at rest and never displayed again after entry, including to Sprint-Ready staff.
- **Account security.** Failed-login lockout with audit logging, hardened session cookies, optional TOTP two-factor authentication with backup codes, and signature-verified inbound webhooks. Administrative and authentication events go to an append-only audit log.
- **Retention.** Tickets, citations and traceability records are kept indefinitely by design, because they are the audit trail. There is no published fixed deletion period yet for raw source files after processing. Where a procurement process or data processing agreement needs one, we agree it in writing.
- **Full isolation.** For organisations that need documents never to leave their own boundary, the Enterprise tier deploys the pipeline inside the customer's own cloud (VPC). It is currently delivered as a design partnership while that deployment path is completed.

11 The economics

The public calculator on sprintready.dev uses the composite team from section 2 as its default. Every input can be changed.

Input or output	Value	Basis
Tickets per month	96	8 specification documents × 12 tickets
Authoring time by hand	40 hours	25 minutes per ticket, covering decomposition, acceptance criteria and the source link
Cost by hand	£2,400 a month	£450 day rate, 7.5 productive hours (about £60 an hour)
Sprint-Ready processing	about £39 a month	Measured 0.81 credits per ticket (7 Sept run) at £0.50 per credit
Professional subscription	£825 a month	12-month minimum term; includes 5,000 credits a year
Net saving	**£1,575 a month**	Cost by hand minus the Professional subscription

At the measured rate, a ticket costs about **41p** to generate at Self-Serve prices, against about **£25** of senior time to write by hand. For smaller teams the subscription is not the right comparison. A team writing 40 tickets a month spends about £1,000 of senior time; the same volume costs about £16 in credits on the Self-Serve tier, or on the £0-a-month Pre-Seed Incubator tier for companies that have raised under £1M.

> **What the model does not count.** Review time is kept, not saved: generated tickets still need review, and the calculator assumes they get it. The model also leaves out the larger, less predictable saving of not reconstructing traceability before an audit or submission, and the cost of the defects that section 4 shows are cheapest to catch at the requirements stage. The measured credit rate comes from one production run; longer or denser documents cost more.

12 Adopting it inside a quality management system

Sprint-Ready is **not a medical device** and does not become part of the product. It is a development tool that drafts candidate work items. Teams therefore account for it as they would any tool used in their software lifecycle: in the IEC 62304 development plan and tool list, and under their own procedures for software used in the quality system. The FDA's final Computer Software Assurance guidance (September 2025) applies a risk-based approach to software used as part of production or the quality system, scaling assurance activities to the risk of the tool's intended use [7]. A practical approach:

1. **Define the intended use.** "Drafts candidate work items from controlled specification documents. Every item is reviewed by a qualified team member before implementation." Sprint-Ready does not approve, release or verify anything.

2. **Identify the failure modes that matter:**
   - an invented requirement;

- a missed data gap;
- a wrong source reference;
- a missed duplicate;
- a false merge;
- a misleading preliminary risk band.

3. **Map the controls:**
   - Source Lock and data gaps;
   - the source reference on every ticket;
   - conservative dedup gates with a recorded decision;
   - the preliminary label and UNCLASSIFIED fallback on risk bands;
   - the `NEEDS-CLARIFICATION` tag;
   - above all, your own review before sprint commitment.

4. **Run your own assurance test.** Use a representative (or sanitised) specification that deliberately leaves out one value. Check that:
   - the gap is flagged, not filled;
   - citations point to the right sections;
   - re-uploading the same document produces merges or skips rather than duplicates;
   - the exported matrix matches the board.

   Keep the results as your objective evidence.

5. **Set the review rule in your SOP.** For example: every generated ticket is reviewed at backlog refinement, and tickets tagged `NEEDS-CLARIFICATION` are resolved with the specification owner before they are committed to a sprint.

The Trust & Compliance page provides the supplier-side information a quality team typically asks for: change control practice, testing evidence, data handling and the human-review model. Specific audit questions go to audit@sprintready.dev.

13 Limitations, and what we don't claim

Regulated buyers verify vendor claims, so here are ours stated plainly:

- **Not a validated GxP system, not an eQMS, not a medical device.** Sprint-Ready structures its output to support GAMP 5, IEC 62304 and ISO 14971 traceability downstream. It does not make a product compliant.
- **No enforced approval hold.** Tickets are created in the backlog once they clear the duplicate check. Review happens on the board, helped by the confidence tier and the `NEEDS-CLARIFICATION` tag. A hold-every-ticket gate was built and tested, then withdrawn because it added a manual step to every well-specified ticket.

- **Source Lock reduces invention; it cannot mathematically exclude it.** This is why review is part of the method.
- **Granularity mismatch in dedup.** When the model splits one source passage into several small tickets, each can be narrower than an existing consolidated backlog item and so not reach the merge gates. The design accepts a visible duplicate over a silent false merge; aggregating same-run tickets before comparison is on the roadmap.
- **PDF and flowchart-image ingestion** are built but have not yet had a dedicated, logged end-to-end verification on the current infrastructure. Word and Excel are verified.
- **Evidence so far is from internal test projects.** The Azure DevOps and Jira runs in section 8 used Sprint-Ready's own projects. We will say so on the Trust & Compliance page when a live customer account is behind the numbers.
- **Enterprise items in progress.** Dedicated VPC deployment and Entra ID single sign-on are delivered through a design partnership. Two-factor authentication is optional rather than enforced.

14 Conclusion

Medical device teams do not lack specifications or traceability tools. They lack a trustworthy way to get from one to the other: turning controlled design documents into the day-to-day work items engineers build and test against, without losing the thread that auditors, notified bodies and the FDA will follow later.

Sprint-Ready's position is narrow and deliberate. It automates that authoring step inside the tools teams already use, under constraints designed for regulated work:

- cite the source;
- flag the gap instead of filling it;
- check the live backlog before creating anything;
- record every decision on the item;
- compute risk bands in code;
- keep a permanent traceability record.

Every ticket cites its source. Every decision leaves a record.

Test it on your own document

100 free credits, no card and no sales call. That is enough for roughly 120 generated tickets at the measured rate. Run one specification through it and judge the output yourself.

Start free → | See the testing evidence

References

All links checked 24 September 2026. Where a standard is summarised through a secondary source, the reader should confirm the wording against a licensed copy.

1. US FDA. *Content of Premarket Submissions for Device Software Functions*, final guidance, June 2023. fda.gov/media/153781/download
2. US FDA. *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*, 2002. hhs.gov (PDF)
3. US FDA. *Quality Management System Regulation (QMSR)*. fda.gov
4. MedDevice (Substack). *V&V deep dive, part 2: traceability*. Secondary summary of IEC 62304 cl. 5.1.1, ISO 14971 cl. 4.5 and ISO 13485 cl. 7.3.2. meddevice.substack.com
5. Johner Institute. *IEC 62304 2nd edition: areas of application and changes*. blog.johner-institute.com
6. Med-di-dia. *Annex II EU MDR: technical documentation*. med-di-dia.com
7. US FDA. *Computer Software Assurance for Production and Quality System Software*, final guidance, September 2025. fda.gov
8. Stecklein, J. M. et al. *Error Cost Escalation Through the Project Life Cycle*, NASA, 2004. ntrs.nasa.gov
9. PMI. *Pulse of the Profession: Requirements Management, A Core Competency for Project and Program Success*, 2014. pmi.org
10. IEEE Spectrum. Analysis of FDA medical device recall records, November 2025. spectrum.ieee.org
11. US FDA. FY2014 Form 483 inspectional observations, medical devices. fda.gov/media/94574/download
12. US FDA. Warning letter, NeuroSync Inc., 9 May 2025. fda.gov
13. US FDA. Warning letter, Contec Medical Systems Co., Ltd., 2 October 2025. fda.gov
14. NIST. *AI 600-1: Artificial Intelligence Risk Management Framework: Generative AI Profile*, July 2024. nvlpubs.nist.gov
15. Vectara. *Hallucination Leaderboard* (updated May 2026). github.com/vectara/hallucination-leaderboard
16. Magesh, V. et al. *Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools*, Stanford RegLab. reglab.stanford.edu

17. Kalai, A. T. et al. *Why Language Models Hallucinate*, 2025. arxiv.org/abs/2509.04664
18. Krishna, M. et al. *Using LLMs in Software Requirements Specifications: An Empirical Evaluation*, IEEE RE 2024. arxiv.org/abs/2404.17842
19. OpenAI. *How your data is used to improve model performance*. help.openai.com
20. Anthropic. *Updates to Consumer Terms and Privacy Policy*, August 2025. anthropic.com
21. AWS. *Amazon Bedrock FAQs*. aws.amazon.com/bedrock/faqs
22. AWS. *Data protection in Amazon Bedrock*. docs.aws.amazon.com
23. AWS. *Increase throughput with cross-Region inference*. docs.aws.amazon.com
24. Anthropic. *Commercial Terms of Service*. anthropic.com/legal/commercial-terms
25. Ketryx. *Requirements management*. ketryx.com
26. Ketryx. *Pricing*. ketryx.com/pricing
27. Jama Software. *Artificial intelligence in Jama Connect*. jamasoftware.com
28. Jama Software. *Pricing*. jamasoftware.com
29. Siemens. *Polarion X 2512: what's new and noteworthy*, March 2026. blogs.sw.siemens.com
30. PTC. *PTC delivers new AI functionality with new ALM releases*, January 2026. ptc.com
31. Matrix One Health. *Matrix Req*. matrixone.health
32. Greenlight Guru. *Design control software*. greenlight.guru
33. Modern Requirements. *Copilot4DevOps pricing*. copilot4devops.com
34. Microsoft Learn. *Use GitHub Copilot with Azure Boards*. learn.microsoft.com
35. Atlassian. *AI in Jira*. atlassian.com
36. Atlassian Marketplace. *Duplicate AI: find and merge duplicate issues*. marketplace.atlassian.com
37. StoriesOnBoard. *AI user story generator*. storiesonboard.com
38. Sprint-Ready. *Trust & Compliance*. sprintready.dev/trust-and-compliance.html
39. Perforce. *Perforce ALM*. perforce.com

About this paper. Written by Sprint Ready LTD (Company No. 17375471, England & Wales). Version 1.0, 24 September 2026.

Sources. Descriptions of third-party products are based only on their own public pages on the date stated and may have changed. Product names are trademarks of their respective owners, used here for identification and comparison only. Sprint-Ready figures come from logged executions described on the Trust & Compliance page.

Not advice. This paper is not legal, regulatory or quality-affairs advice, and it is not a certification. Regulatory requirements should be confirmed against the current official texts and with qualified professionals.