MiniTune BA Portfolio Case Study (Public)

Overview

MiniTune is a macOS menu-bar playback app. This case study covers how delivery moved from feature-only backlog management to a structured deployment and release reliability stream using Kanban.

Problem statement

Feature delivery was progressing, but release operations needed enterprise-grade structure:

- repeatable release readiness checks
- clear operational ownership
- auditable verification evidence
- clean separation between active release work and future feature backlog

Role and contribution

Digital BA + Tech BA (solo), covering:

- reverse-engineering of implemented architecture
- requirements and operating model definition
- Jira epic/story structure for deployment reliability
- UAT and release-governance documentation

Delivery approach

1) Baseline freeze

Created an implemented-vs-planned matrix to prevent scope drift and ensure documentation reflected real code paths only.

2) Kanban operating model

Standardised board operation:

- `Selected for Development` for backlog
- `In Progress` for active execution
- `Done` only with verification evidence

A dedicated deployment epic was introduced while leaving future integrations in backlog.

3) Functional and non-functional documentation

Produced:

- PRD-lite for deployment reliability
- FRD grouped by epic outcomes

- NFR coverage for reliability, security, privacy, and operational quality
- RAID and UAT artefacts

4) Release runbook

Documented end-to-end release flow:

- pre-release readiness checks
- packaging and publishing workflow
- endpoint verification checks
- rollback process

Architecture summary

```
flowchart LR
App["MiniTune macOS app"] --> API["Backend API"]
API --> RL["Rate limit and access controls"]
App --> Feed["Update feed domain"]
Feed --> Storage["Release artifact hosting"]
```

Key outcomes

- Deployment delivery became a first-class workstream in Jira.
- Release cards gained concrete Definition of Done evidence.
- Public and internal documentation variants were aligned and reusable.
- Backlog governance improved through explicit scope boundaries.

Lessons learned

- Solo delivery benefits strongly from enterprise-level structure when release complexity grows.
- Distinguishing implemented reality from planned roadmap prevents documentation debt.
- Kanban quality improves when status transitions are tied to operational evidence, not task completion alone.

Portfolio artefacts produced

- Implemented-vs-planned baseline
- Jira operating model and deployment epic backlog
- Internal Confluence pack
- Public portfolio pack
- PDF deliverables for internal handoff and external sharing