**Sprint Review & Retrospective – SNHU Travel  
(ChadaTech Scrum Pilot)**

This retrospective summarizes the outcomes of the SNHU Travel application, which served as ChadaTech’s Scrum-Agile pilot. Over the sprint(s), I rotated through Product Owner, Scrum Master, Developer, and Tester roles. The focus of this paper is to evaluate how Scrum roles, practices, and events contributed to delivering working software, how we handled interruptions, what tools supported the work, and whether Scrum was the optimal approach for this project.

**Applying Roles**

• Product Owner (PO): I maintained the Product Goal—‘Enable travelers to discover, book, and manage trips quickly with transparent pricing and alerts.’ I curated and prioritized the Product Backlog, ensuring each user story had clear acceptance criteria. Example: the ‘Price Alerts’ epic was split into stories for subscribing, receiving notifications, and managing thresholds; we prioritized ‘subscribe’ first to prove end‑to‑end value.

• Scrum Master (SM): I facilitated Sprint Planning, Daily Scrums, and the Sprint Review/Retrospective, removed blockers (e.g., environment access), and protected WIP. I also coached the team to adopt a clear Definition of Done (DoD): code reviewed, unit tests >80% coverage, CI build green, feature flags set, and demo ready.

• Developer(s): I implemented the ‘Search & Filter Destinations’ and ‘Price Alerts – subscribe’ stories. Pair programming shortened onboarding time and reduced defects. Trunk‑based development with small PRs accelerated integration, while feature flags let us merge incomplete work safely.

• Tester/QA: I wrote Gherkin‑style scenarios for acceptance criteria (Given/When/Then) and automated smoke tests in CI. We added API contract tests for the booking provider to detect upstream changes early.

**Completing User Stories via Scrum-Agile SDLC**

Scrum events and artifacts are mapped cleanly to SDLC stages in an iterative flow: requirements discovery (backlog refinement), design (whiteboard spikes, ADRs), implementation (sprints), verification (automated tests, review), and deployment (CI/CD). For example, the ‘Search & Filter Destinations’ story went from rough persona needs to a Definition of Ready checklist, a small design spike and wireframe, then implementation behind a feature flag. Demos in the Sprint Review captured stakeholder feedback that we used to adjust the filter facets in the very next sprint.

**Handling Interruptions & Change**

Mid‑sprint, SNHU Travel updated its target market to include international clients, requiring multi‑currency support. Using Scrum, we negotiated scope: we kept the current sprint goal focused on delivering the Price Alert subscription end‑to‑end, and queued ‘Display multi‑currency amounts’ as the top item for the next sprint. Because the architecture used currency‑agnostic amounts internally and we worked in thin slices, the change was absorbed with minimal churn.

Another interruption involved a partner API deprecating an endpoint. Contract tests failed in CI, triggering an immediate conversation with stakeholders. We added a spike story to evaluate migration, time‑boxed it to one day, and updated our integration wrapper while maintaining service via a feature flag.

**Communication & Collaboration**

Samples of effective communication include:

"@PO We can deliver subscription opt‑in and confirmation this sprint, but push notification templates to next sprint to protect the Goal. Can we accept email‑only notifications for now to validate value?" – Daily Scrum thread

"Test finding: filtering by ‘beach’ + ‘family‑friendly’ returns expected sets, but performance degrades beyond 10,000 results. Recommend indexing ‘themes’ and paging at 50 results. Created ticket TRV‑128 and linked to performance dashboard." – QA note in board comment

These examples worked because they were concise, decision‑oriented, and visible in shared tools (board comments, chat threads). They turned communication into collaboration by making trade‑offs explicit and inviting quick alignment.

**Organizational Tools & Scrum Principles**

Tools: Kanban‑style boards for transparency; Jira queries and dashboards for flow metrics (cycle time, WIP, throughput); a shared Working Agreement; and CI/CD with automated tests. Scrum events amplified effectiveness: Sprint Planning focused on a single Sprint Goal; Daily Scrum inspected plan vs. Goal; Backlog Refinement kept work thin and testable; the Sprint Review harvested feedback; the Retrospective improved ways of working.

Most impactful practices were: a crisp Definition of Done, feature flags, pairing on complex changes, and keeping PRs under 300 lines. Cycle time dropped from ~6 days to ~3.5 days by the final sprint, while defect escape rates decreased.

**Evaluating the Scrum-Agile Approach**

Pros observed during SNHU Travel:

• Continuous stakeholder feedback via Sprint Reviews improved fit (e.g., filter facets, alert thresholds).

• Early risk discovery through spikes and contract tests reduced integration surprises.

• Team cohesion improved through pairing, swarming, and visible WIP limits.

• Ability to pivot to multi‑currency with limited rework due to small, vertical slices and flags.

Cons/challenges encountered:

• Upfront forecasting for external dependencies remained difficult without historical flow data.

• Stakeholders initially expected fixed scope per sprint; education was needed around Sprint Goals vs. scope flexibility.

• Protecting focus time required discipline to avoid mid‑sprint scope creep.

Was Scrum the best approach? For a novel product with uncertain requirements and the need for rapid feedback, yes. Scrum’s empiricism and time‑boxed cadence fit the SNHU Travel context better than a linear, stage‑gated model. If requirements were stable and integrations minimal (e.g., compliance‑driven, fixed‑spec work), a Waterfall approach might have sufficed. Still, for this project, the Scrum pilot accelerated learning and delivery.

**Conclusion**

The Scrum pilot delivered working increments, shortened cycle time, and strengthened team culture at ChadaTech. Based on these results, I recommend scaling Scrum with guardrails: invest in Product Ownership, coaching for new Scrum Masters, and platform‑level CI/CD to support more teams. Continue measuring outcomes (lead time, deployment frequency, user satisfaction) to guide improvements.