# **Sprint Review and Retrospective**

**Project:** SNHU Travel – trip search and booking MVP  
 **Role for this deliverable:** Scrum Master

## **Applying Roles**

Our Scrum-Agile team operated with tight role clarity that translated into delivery speed.

* **Product Owner (PO):** curated the Product Backlog, defined acceptance criteria, and prioritized outcomes over output. Example: the PO reduced scope on “Booking flow” from three payment methods to one, which unblocked development and kept Sprint 2 goal achievable.
* **Scrum Master (me):** coached the team on events, removed blockers, and protected focus. I facilitated daily standups limited to 15 minutes, enforced a clear Definition of Done, and shielded the team when stakeholders attempted mid-sprint scope injections.
* **Developers:** full stack, test, and UX worked as one cross-functional unit. Pair programming on the itinerary search reduced defects in Sprint 1. A tester joined backlog refinement and turned acceptance criteria into Gherkin checks.
* **Stakeholders and SME:** a travel-agent SME validated fare rules early, which prevented rework during Sprint 3.

**Result:** Sprint goals were consistently met. Cycle time for user stories dropped after we adopted pair reviews and a tighter Definition of Done.

## **Completing User Stories**

Scrum practices mapped cleanly onto SDLC stages.

* **Discovery to backlog:** user stories were written with INVEST qualities. Example story: “As a traveler, I can filter by price range so that I can book within budget.”
* **Design and build:** stories flowed from “Ready” to “In-Progress,” “Code Review,” “Testing,” then “Done.” We used short vertical slices, which allowed testable increments every few days.
* **Verification:** acceptance criteria were automated where possible. For the price filter story, we added boundary tests for min and max values and a UI smoke test.
* **Validation:** a stakeholder demo at each Sprint Review prompted quick changes while the context was fresh.

**Outcome:** 90 percent of stories that entered a sprint were accepted by the end of the sprint. Carryover aligned with predictable causes such as late external dependency.

## **Handling Interruptions**

Midway through Sprint 2, leadership asked us to pivot from “package bundles” to “loyalty program lookup.” I ran a quick **Sprint Planning 2.0** session:

1. surfaced the impact, 2) cancelled the unfinished bundle stories, 3) pulled a ready loyalty story whose dependencies were clear, 4) reset the Sprint Goal to “search plus loyalty ID capture.”  
    We captured the change in a Sprint change log and updated our forecast. The team still delivered a cohesive increment because the sprint goal remained single-threaded.

**Lesson:** agility handled scope change without thrash because we preserved a clear Sprint Goal and small vertical slices.

## **Communication**

I kept communication short, visible, and respectful.

* **Standup note (sample):** “Yesterday: merged price filter PR #42. Today: hook filter into query builder. Blocker: API rate limit on test tenant.”
* **Backlog comment (sample):** “AC clarified with PO: default sort is ‘lowest price.’ Added edge case when price ties.”
* **Stakeholder email (sample):** “We can demo search results at Thursday’s Review. Payment UI is behind a feature flag, not part of this increment.”

**Why effective:** each example states facts, next steps, and blockers. This reduced meetings and invited fast help.

## **Organizational Tools**

* **Issue tracker:** Jira board with swimlanes for “Goal stories” vs “Other stories” kept focus visible during standups.
* **Git and pull requests:** required two approvals and a green build. Defects dropped after we added a short PR template with “risk, tests, rollback.”
* **CI checks:** unit suite, linter, and UI smoke tests ran on every PR.
* **Scrum events:**
  + **Planning:** we selected only stories with a clear “Ready” state.
  + **Daily scrum:** impediments surfaced quickly and I handled them offline.
  + **Review:** we demoed working software only, which kept feedback sharp.
  + **Retrospective:** we ran “Start, Stop, Continue,” picked one improvement per sprint, and actually tracked it.

**Verdict:** the board, PR template, and CI checks directly supported transparency, inspection, and adaptation.

## **Evaluating the Agile Process**

**Pros:** early stakeholder feedback, faster discovery of requirements, smaller batches that reduced risk, visible progress through burn-down and board flow metrics, and continuous learning through retrospectives.

**Cons:** coordination overhead for ceremonies, need for strong PO availability, and occasional friction with external vendors that operated on a waterfall schedule.

**Was Scrum-Agile the best fit?** Yes. The SNHU Travel MVP had evolving requirements and a need to demonstrate value early. Scrum maximized learning per week and limited rework. For a highly regulated system with fixed scope and long lead times, a plan-driven model or a hybrid could be better. For this MVP, Scrum was the right call.

**Actionable improvements:** reduce WIP to one story per developer, expand automated test coverage on the booking flow, and formalize a “DoR” checklist that includes mock API data and test accounts.