This term my team started a Scrum Agile approach for the SNHU Travel application. Previously, Chada Tech had approached creating software using a waterfall model, however, my team is responsible for introducing the agile methodologies. This is a plain account of what we did, why it worked, and where the approach showed opportunities for improvement.

First, about roles and why they mattered. When I took on the Product Owner responsibilities, I focused on engaging stakeholders directly so that priorities were based on what customers actually needed. In one conversation a stakeholder said they wanted a top 5-10 list of travel destination they might like built into the application. I took the users want and made it into a simple to understand user story. I essentially turned a vague ask into a feature we could ship. As a result, we wrote a clear story “As an End User, I want to choose whether the list shows the Top 5 or Top 10 destinations so that I can control how many options I see at a glance”. When I served as Scrum Master, my attention shifted to flow. I watched for scope creep, nudged the team to keep stories small and vertical, reinforced our Definition of Done, and cleared roadblocks quickly. When the top five idea started to balloon into personalization and ranking logic, I pushed for a quick decision to ship manual toggles first and defer ranking for a later Sprint. In the Developer role, I implemented the toggle UI, connected it to a simple endpoint, added unit tests, and verified acceptance criteria end to end. The three perspectives fit together well.

Next, on how user stories reached completion. Scrum helped because we kept every story truly vertical. The top five feature shipped as a complete slice that the user could click, not as a half-finished backend or a disconnected mock. We kept stories clear by distilling stakeholder input into specific acceptance criteria. For example, on the Destinations page, clicking a toggle on a card adds that destination to a comparison tray. If a traveler tries to add a sixth item, the interface shows a friendly limit message. We enforced a simple Definition of Done that required peer review, passing tests, and a demo path. Done meant ready to show, not almost working on a single laptop. Backlog refinement sessions kept us honest. If a story felt like an epic, we split it. If a requirement was fuzzy, we clarified it before Sprint Planning so estimation and commitments were credible.

When plans changed, we adapted without losing momentum. We used Planning Poker for estimates so everyone shared the mental model of complexity. That small habit paid off when priorities changed toward detox and wellness travel. We reordered the backlog to put wellness features first, re-estimated the affected stories in minutes, adjusted the Sprint Goal rather than the entire plan, and kept the timebox. A smaller, clearer increment emerged that still lined up with the new business need. Because estimation was collaborative and the backlog was a living thing, the team understood both the why and the what of the pivot. The change did not stall delivery because we never tried to refactor the entire roadmap in one sitting.

Communication stayed short, specific, and focused on unblocking. In our weekly standups we kept to a simple pattern. Yesterday I finished the top five toggle UI and unit tests. Today I will wire the toggle state to the backend and add an acceptance test for the five item limit. My blocker is the final list of wellness tags from the Product Owner. That style worked because it was concrete and surfaced the only dependency that mattered. For asynchronous alignment, I posted brief decision notes. Decision: add a quick DoD checklist to each story that includes peer review, tests passing, and a demo path. Why: reduce surprises in Sprint Review. Impact: about fifteen minutes per story now that will save rework later. Clear messages like these built trust and encouraged collaboration, since everyone could see the tradeoffs and the rationale.

Regarding the tools and Scrum events that helped most. A Kanban board in Jira made work visible and kept work in progress at a reasonable level. Backlog refinement and Planning Poker turned rough ideas into buildable slices with shared estimates. A lightweight burndown trend acted like a smoke alarm in the second half of the Sprint, useful for prompting a scope cut before the end. The events themselves tied the system together. Sprint Planning aligned capacity with a realistic Sprint Goal. Standups surfaced blockers quickly. Sprint Review gave stakeholders working software to react to, including the top five toggles and the wellness filter. Retrospective produced small experiments like the DoD checklist that actually stuck.

Now to the heart of the evaluation for this project. The Agile approach brought several pros. The feedback loop was fast. Stakeholders saw real software early, which limited the risk of building the wrong thing. Change was manageable. Reordering the backlog and re-estimating with Planning Poker made the wellness pivot a matter of hours rather than a derailing event. Quality improved through cadence. The DoD, peer review, and mini demos reduced late surprises. Morale and clarity were better than a plan driven approach in this context because the team had a clear Sprint Goal and visible flow.

There were also cons and tradeoffs. Stakeholder time became a critical dependency. When feedback slipped, ambiguity grew and rework risk rose. Estimation required discipline. Planning Poker is fast only when stories are refined, and rushed refinement produced uneven estimates. Tooling added a small startup cost. Setting up boards, checklists, and basic automation took some time before it began paying back.

Taking both sides into account, Scrum Agile was the right fit for SNHU Travel. The work called for evolving requirements, early user validation, and frequent content and UX adjustments. Short iterations and a living backlog matched those needs better than a predictive plan with long phases. If the project had been compliance heavy with fixed requirements and tight change control, a predictive or hybrid model might have performed better. Here, where themes and user experience benefited from rapid feedback, Scrum allowed us to learn and deliver at the same time.

In closing, this pilot showed that a light, disciplined Scrum practice can deliver steady increments while still welcoming change. Direct stakeholder conversations led to clear user stories like the top five toggle. Planning Poker and small slices kept re-planning painless. Simple ceremonies and a visible board kept the team aligned and reduced uncertainty. If ChadaTech wants faster learning cycles and better alignment between what is built and what users value, expanding Scrum Agile beyond this pilot is a sound move, supported by coaching, lightweight tooling, and the habit of making small, real improvements every Sprint.