Sprint 1 Retrospective

Sprint: Sprint 1
Date: February 19, 2026

What Went Well

1. **Clear Planning Paid Off**
   - Having detailed user stories with acceptance criteria made implementation straightforward
   - Story point estimates were accurate — completed exactly 7 points as planned
2. **CI/CD Setup Was Smooth**
   - GitHub Actions configured quickly with matrix testing (Node 18.x and 20.x)
   - Pipeline passed on first run after push
3. **Test-Driven Approach**
   - Writing comprehensive tests helped catch edge cases early
   - Achieved 91%+ code coverage
   - Tests serve as documentation for API behavior
4. **Incremental Commits**
   - Made 4 focused commits showing clear progression
   - Each commit represented a logical unit of work
   - Easy to track what changed and when

What Didn't Go Well

1. **Git Authentication Issues**
   - Encountered credential conflicts when pushing to GitHub
   - Had to use GitHub Desktop as workaround
   - Lost time troubleshooting authentication
2. **Implemented More Than Planned**
   - Built Update (US-4) and Delete (US-5) endpoints in Sprint 1
   - Should have stuck strictly to Sprint 1 scope
   - Made it harder to demonstrate incremental delivery across sprints

3. **No Logging from the Start**
    - Didn't include console logging for API operations
    - Will need to add this in Sprint 2
    - Should have been part of initial setup

Action Items for Sprint 2

Improvement 1: Add Structured Logging

What: Implement console logging for all API operations (create, read, update, delete)
Why: Need observability into API behavior; required for Sprint 2 monitoring deliverable
How: Add `console.log` statements in controller methods with operation type, task ID, and timestamp

Improvement 2: Commit More Frequently

What: Make smaller, more frequent commits during implementation
Why: Current commits bundle multiple changes; harder to demonstrate truly incremental work
How: Commit after each logical change (e.g., separate commits for each endpoint, tests, validation)

Improvement 3: Test Edge Cases First

What: Write failing tests before implementing features
Why: Caught some edge cases late in Sprint 1 (e.g., empty string validation)
How: Follow strict TDD: write test → see it fail → implement → see it pass

Key Learnings

1. **Agile Planning Works**
    - Breaking work into user stories with clear acceptance criteria made it easy to know when "done" was done
    - Sprint planning helped focus effort on high-priority items
2. **CI/CD Catches Issues Early**
    - Pipeline ran tests automatically on push

    - Would catch regressions before they reach production
3. **DevOps is About Automation**
    - Automated tests + automated pipeline = confidence in deployments
    - Manual testing doesn't scale

Sprint 2 Focus

- Complete US-4 (Update Task) and US-5 (Delete Task) with proper commit history
- Add logging/monitoring as planned
- Apply the 3 improvements identified above
- Maintain 90%+ test coverage