Lab 7 — Sprint Retrospective & Process Improvement

Software Project Management Labs

Learning Outcomes

By the end of this lab, you will be able to:

- Explain the purpose of a **Sprint Retrospective** in Agile projects.
- Derive simple **process metrics** from your Sprint 1 data (planned vs. completed, completion pattern over time).
- Identify **bottlenecks** and improvement opportunities in your current way of working.
- Create a small **Improvement Backlog** and define concrete actions for the next Sprint.

1 Sprint Retrospective: What & Why

So far, your labs have covered:

- Lab 4: Product Backlog (user stories, MoSCoW, Fibonacci estimation, dependencies).
- Lab 5: Sprint 1 Planning & Tracking (Sprint Backlog, Burndown Chart).
- Lab 6: Release Planning with velocity and multi-Sprint plan.

A **Sprint Retrospective** focuses on how the team worked, not only what they delivered. It answers:

- What **went well** during the Sprint?
- What **did not go well** (or felt painful / confusing)?
- What will we **change or experiment with** in the next Sprint?

In this lab, you will:

1. Use your Sprint 1 data (from Lab 5) to compute some simple Sprint metrics.
2. Reflect on your process using a **Keep / Improve / Try** retrospective.
3. Build a small **Improvement Backlog** for Sprint 2.

2 Simple Sprint 1 Metrics

We will use your Sprint 1 data from Lab 5 (Sprint Backlog and Burndown Table).

Step 1: Reconstruct Sprint 1 Story Data

From your Lab 5 report, collect for each Sprint 1 story:

- Story ID (e.g., US-01).
- Short title.
- Story points.
- Status at end of Sprint: Done / Not Done.
- **Completion day** (for Done stories), based on your Burndown Table, e.g. Day 2, Day 5, Day 9.

In your report, include a table like:

ID	Story (short)	Pts	Status	Completion Day
US-xx	...	...	Done / Not Done	Day ...
US-yy	...	...	Done / Not Done	Day ...

If a story is Not Done, you can write "–" for Completion Day.

Step 2: Compute Basic Metrics

For Sprint 1, compute:

- **Planned points**: sum of points of all Sprint 1 stories (this should match Lab 5).
- **Completed points**: sum of points of stories with Status = Done (this is your Sprint 1 velocity).
- **Completion distribution**:
  - Choose the *first half* of the Sprint (e.g. Days 1–5) and the *second half* (e.g. Days 6–10).
  - Compute how many points were completed in the first half vs. second half (based on Completion Day).
- **Average completion day** (only for Done stories):

$$\text{AvgCompletionDay} = \frac{\sum \_\text{story } i \text{ DoneDay}(i)}{\#\text{ of Done stories}}.$$

Include both the formulas *and* the final numeric values in your report.

Average Completion Day (what and why?)

Completion Day. Consider a Sprint with days **Day 1, Day 2, . . . , Day** N. For each user story, we record the day on which it reaches the *Done* column. We call this the **Completion Day** of the story.

Example (10–day Sprint):

Story	Done on	Completion Day
US-01	Day 2	2
US-02	Day 3	3
US-03	Day 8	8

Note: Completion Day is *not* the total number of days it took to implement the story; it only tells us *when* in the Sprint the story was finished.

How to compute the average. Take all stories that are **Done** in the Sprint and list their Completion Days. Then:

$$\text{AvgCompletionDay} = \frac{\text{sum of Completion Days for all Done stories}}{\text{number of Done stories}}.$$

Example (10–day Sprint, 5 Done stories with completion days $2, 3, 5, 7, 9$):

$$\text{AvgCompletionDay} = \frac{2+3+5+7+9}{5} = \frac{26}{5} = 5.2.$$

This means a "typical" story was finished around **Day 5**.

How to interpret the result. For a 10–day Sprint:

- If AvgCompletionDay ≈ 3: most stories finish early in the Sprint; work flows smoothly and there is buffer for bugs and changes.
- If AvgCompletionDay ≈ 8 or 9: most stories finish very late; work is "bunched up" at the end of the Sprint, which is risky (many items in progress, testing and review rushed).
- If AvgCompletionDay is around 5–6: stories finish in the middle; we then compare first–half vs second–half completed points to see if we still finish most work late.

We use this metric in the retrospective to connect:

$$\textbf{metrics} \rightarrow \textbf{problems} \rightarrow \textbf{improvements}.$$

Step 3: Interpret the Numbers

In 5–8 bullet points, interpret what the metrics tell you. For example:

- Did many stories finish near the **end** of the Sprint?
- Did you start too many items at once and finish them late?
- Was the Sprint overly packed or underloaded compared to your capacity estimate?
- If some stories were Not Done, why do you think that happened?

3 Flow on the Board (Optional but Recommended)

If your team used a board tool (Trello/Jira/Notion/GitHub Projects):

- Take a screenshot of your Sprint 1 board at (or close to) the end of the Sprint.
- Briefly describe:
  - How many items were in "In Progress" vs. "Done".
  - Any cards that stayed in the same column for many days.
  - Any obvious bottlenecks (e.g. testing column always full).

If you do not have real screenshots, you may **recreate** a simplified board just for this lab and use that as your reference (but state that it is reconstructed).

4 Retrospective — Keep / Improve / Try

Step 4: Brainstorm Retrospective Items

As a group, list ideas in three columns:

- **Keep**: things that worked and should continue.
- **Improve**: things that were OK but could be better.
- **Try**: new experiments or practices for the next Sprint.

In the report, include a table like:

Keep	**Improve**	**Try**
Item 1	Item 1	Item 1
Item 2	Item 2	Item 2
Item 3	Item 3	Item 3

Try to connect your items to real observations from Sprint 1:

- e.g. "Keep: clear acceptance criteria on Must stories",
- e.g. "Improve: avoid starting more than 3 stories at once",
- e.g. "Try: mid-Sprint check-in focused only on blockers".

Step 5: Select Top 3 Improvement Actions

From the "Improve" and "Try" columns, choose **three** items that:

- are concrete (you know roughly what to do),
- are small enough to implement in the next Sprint,
- are likely to really help your team.

For each chosen improvement, specify:

- A short description.
- Why it matters (which problem/metric it addresses).
- How you will know if it worked (simple success criteria).

5 Improvement Backlog for Sprint 2

Step 6: Create an Improvement Backlog

Treat improvement actions as special internal tasks and create a small **Improvement Backlog**. Use a table like:

ID	Improvement Item	Reason / Target	Owner
IM-01	...	...	...
IM-02	...	...	...
IM-03	...	...	...

Step 7: Integrate into Sprint 2 Plan

In a short paragraph, describe how you will integrate these improvements into **Sprint 2**. For example:

- Will you treat them as Sprint backlog items (visible on the board)?
- Will you limit Work-in-Progress (WIP) as one of the experiments?
- Will you adjust how you use your board, stand-ups, or reviews?

6 Group Task (Using Labs 4–6)

Work in the same group as previous labs.

Task 1: Sprint 1 Metrics

Re-open your Lab 5 report and:

- Build the Sprint 1 story table (ID, title, points, status, completion day).
- Compute the metrics from Section 2 (planned vs. completed, completion distribution, average completion day).
- Write 5–8 bullet points interpreting the results.

Task 2: (Optional) Board Snapshot

Capture or reconstruct a board snapshot and briefly describe the flow (items in In Progress vs. Done, long-lived cards, bottlenecks).

Task 3: Retrospective — Keep / Improve / Try

Fill the Keep / Improve / Try table with your team's ideas.

Task 4: Improvement Backlog

Choose your top 3 improvement actions, create the Improvement Backlog table, and explain how these actions will be integrated into Sprint 2.

Task 5: Reflection

In 4–6 bullet points, summarize what you learned about your team's process and how this retrospective connects back to your Release Plan (Lab 6).

7 What to Submit (One File per Group)

File name: `Lab7_RetrospectiveReport.pdf`

Your report must contain:

1. **Sprint 1 Metrics**
   - Table of Sprint 1 story data (ID, title, points, status, completion day).
   - Computed metrics (planned vs. completed, completion distribution, AvgCompletionDay).
   - Short interpretation (5–8 bullet points).
2. **Retrospective Table**
   - Keep / Improve / Try table with your team's ideas.
3. **Improvement Backlog**
   - At least 3 improvement items with reason/target and owner.
   - Brief explanation of how they will be integrated into Sprint 2.
4. **Reflection**
   - 4–6 bullet points on what you learned about your team's process.
5. **Team members**

Checklist before submitting

- Sprint 1 metrics are computed from your *own* Lab 5 data (not the example).
- Keep / Improve / Try items are specific and realistic (not generic sentences).
- Improvement Backlog has at least 3 concrete items with owners.
- Reflection connects *metrics* → *problems* → *improvements*.

Appendix C — Example: Retrospective for IT Community App

This appendix illustrates how Lab 7 might look for the IT Community App example. Your group must use your own Sprint 1 data instead.

C.1 Sprint 1 Story Data (Example)

Recall Sprint 1 stories (from Labs 5–6):

ID	Story (short)	Pts	Status	Completion Day
US-01	Sign up with email/password	3	Done	Day 2
US-02	Login + remember me	2	Done	Day 3
US-03	Create text post	5	Done	Day 5
US-05	Comment on a post	3	Done	Day 6
US-06	Like/unlike a post	2	Done	Day 8
US-08	Search posts by keyword/tag	5	Done	Day 8
US-10	Report/hide toxic content	5	Done	Day 9

All Sprint 1 stories are Done, total = 25 points (velocity from the Lab 5 example).

C.2 Example Metrics

- Planned points: 25.
- Completed points: 25.
- First half of Sprint (Days 1–5):
  - Stories completed: US-01 (Day 2), US-02 (Day 3), US-03 (Day 5).
  - First-half completed points: $3 + 2 + 5 = 10$.
- Second half of Sprint (Days 6–10):
  - Stories completed: US-05 (3), US-06 (2), US-08 (5), US-10 (5).
  - Second-half completed points: $3 + 2 + 5 + 5 = 15$.
- Average completion day:

$$\text{AvgCompletionDay} = \frac{2 + 3 + 5 + 6 + 8 + 8 + 9}{7} = \frac{41}{7} \approx 5.86.$$

Interpretation (example):

- More points were completed in the **second half** of the Sprint (15 vs. 10).
- The average completion day is around Day 6, meaning many stories finished fairly late.
- The team might have started several stories early but only finished them near the end.

C.3 Example Keep / Improve / Try

Keep	Improve	Try
Daily stand-up meeting on MS Teams.	Break big stories (like search + moderation) into smaller tasks earlier.	Introduce a WIP limit of 2 stories per developer.
Using Trello labels for Epics (Auth, Posting, Social, ...).	Avoid starting too many stories at the same time.	Use a short mid-Sprint check-in focused only on blocked items.
Clear acceptance criteria on key stories (US-01, US-03).	Finish testing earlier instead of last 2 days.	Timebox code reviews to avoid long waiting times.

C.4 Example Improvement Backlog

ID	Improvement Item	Reason / Target	Owner
IM-01	Apply WIP limit of 2 active stories per developer.	Reduce late clustering of completed work; encourage finishing before starting new.	Alice
IM-02	Split large stories into smaller tasks before Sprint starts.	Make progress more incremental; reduce risk of "big bang" at the end.	Bob
IM-03	Add mid-Sprint "blocking issues" check-in (15 minutes).	Detect blocked tasks earlier; improve flow through the board.	Carol

Integration into Sprint 2 (example):

- IM-01, IM-02, IM-03 will be added as visible tasks in the Sprint 2 board.
- During Sprint 2 planning, the team will explicitly agree on the WIP limit and how to enforce it.
- At the end of Sprint 2, the team will check whether fewer stories finish very late in the Sprint (compare AvgCompletionDay and burndown shape).

Students can use this example as a pattern, but must fill Lab 7 with data and reflections from their own project.