

TRAINING

Hands-on AI-Assisted Development

IT track Day 3

Automation and certification



Do not redistribute these materials

Redistribution of these materials is prohibited.

These materials are copyrighted by Givery, Inc. Use them only for your own learning and work reference. Do not share them outside your company, on social media or with third parties.

During the course, do not take screenshots or recordings that show these materials.

Today's goals

Make your Day 2 tests run end to end and reusable next month

Run import through reload as an E2E test

Pick 4 cases from your Day 2 test case table, run them in the browser, and decide where to fix the failure.

Save your steps and reviewer in files

Turn test design and regression checks into skills, and your security review into a custom agent.

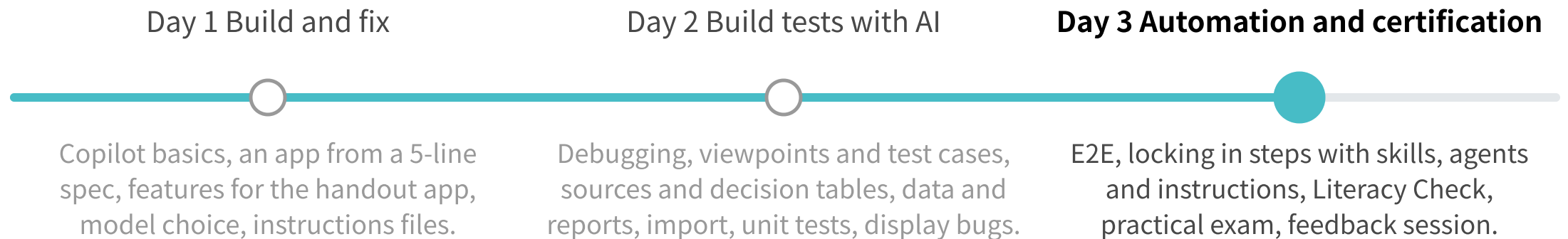
Import data in a new format on your own

In the exam, import September data using only what you built in 3 days; submit decisions with reasons.

If you follow the guide's schedule, the afternoon exam fits in the time

Where today fits in the 3 days

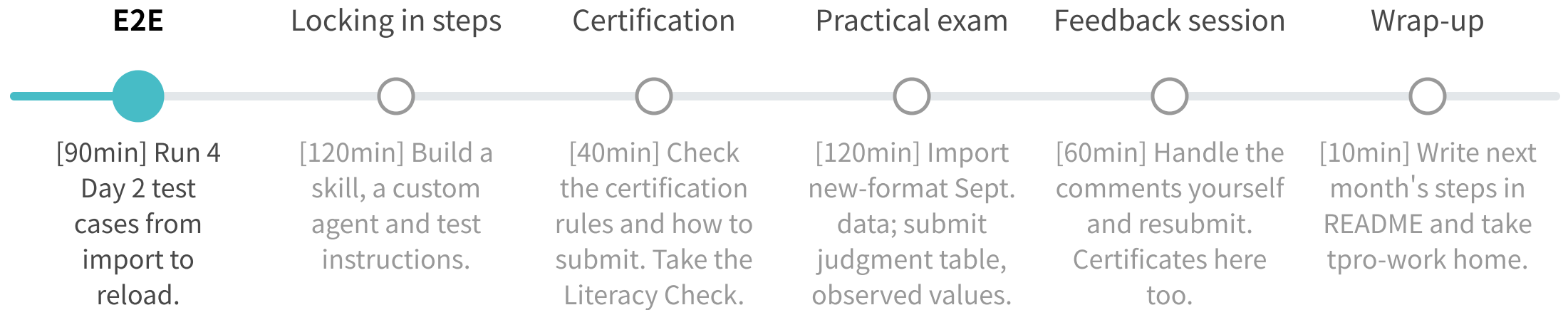
Day 3 makes the Day 1 build, checked on Day 2, **repeatable**



Today you reuse your files from Day 2 more than you learn new screens

Today's schedule

The morning's harness is **used as is** in the afternoon exam



Check your files first [10min]. Lunch break comes after Locking in steps



01

Day 2 review and today's setup

First, check that you have all your files

Exercise 1: Check your files

Check that your Day 2 files open, in the first 3 minutes

Scenario

Today's E2E, skills and practical exam all use your own files through Day 2. If any are missing, the exercises stop mid-morning, so check first.

Steps

- 1 Extract the handout ZIP into the tpro-work folder and check that a day3 folder appears in tpro-work.
- 2 Open tests.html in the browser and check that the SUMMARY line appears.
- 3 Check that testcases.md, decision-log.md and security-notes.md are directly in tpro-work. In the chat, write "all 3 OK" or the missing file names.

Result

The chat has your line: "all 3 OK" or the missing file names.



02

E2E that runs in the browser only

Run import, export and reload automatically inside your own index.html

From unit tests to end-to-end tests

Some bugs **fail end to end** even when each function **PASSes**

What we have done so far

In tests.html you tested 6 functions and toCsv one by one and fixed the export bug. Each test passes values and checks the result.



What we do next

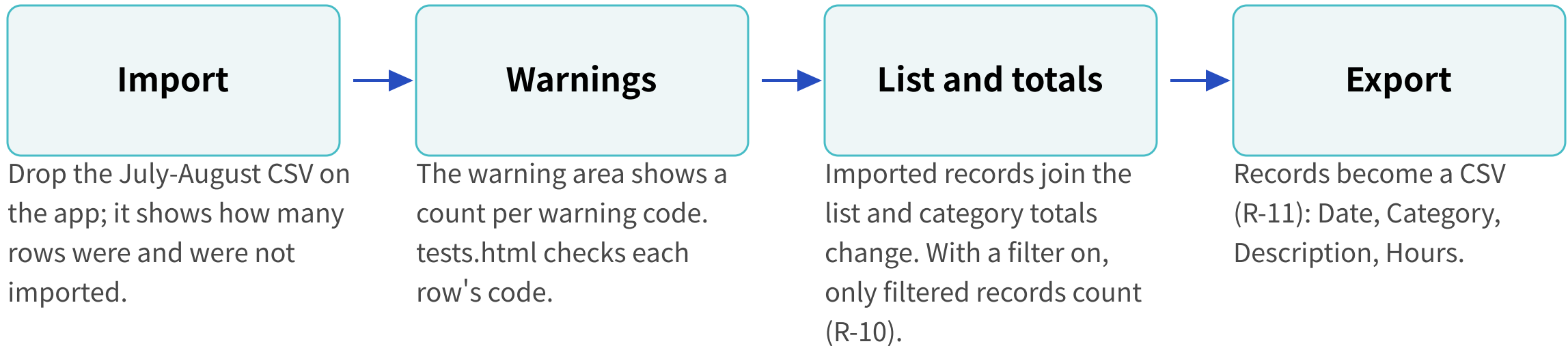
Import a CSV twice, reload an export. Order and state bugs don't show up in one-function tests.



Why it matters later

The exam adds 1 scenario for the new format. Next month, just open index.html with ?e2e to rerun.

E2E does **not recheck** each row's result



e2e.js cannot test **encoding detection**

Stages e2e.js covers

Loaded only when you open `index.html?e2e`. It keeps CSV rows as strings, passes them to `WorkLogApp.importText(text, fileName)`, and counts warnings and list rows. For export, it replaces `URL.createObjectURL` to catch the Blob and sends `Blob.text()` back to `importText`. Each scenario resets records to the initial 8.

Stages e2e.js skips

Real drag and drop, saving downloads, encoding detection. e2e.js holds decoded strings, so it skips every step before a file becomes a string. A person checks these, or Playwright, shown next on the instructor's screen.

Before AI writes the scenarios, **you pick** the 4 cases

- 1 Open testcases.md**
Look at rows where Layer = E2E. Day 2 rows you wrote as order or state viewpoints are candidates too.
- 2 Pick 4**
Pick ones that span 2 or more of the 5 stages 2 pages back. Drop rows that check what unit tests already check. Always include one with a reload.
- 3 Write one reason each**
Create e2e-scenarios.md in tpro-work; for each, write who would misread what if it failed
- 4 Order them**
Put first the one that would hurt most if it failed

Fewer than 4 E2E rows? Check the Source column and move order or state rows to E2E

Exercise 2: Then for the 4 scenarios

Write Then **as counts** and e2e.js can check it directly

Scenario

The Worklog app saves nothing and resets to the initial 8 records on reload. The July-August CSV has rows identical to the initial 8 in all 4 columns, so the first import on Day 2 already doubled them in the list. Re-importing an export adds them again. The Import Specification does not cover duplicates of records already in the list.

Steps

- 1 Send the 4 to Ask and get them as Given, When, Then, with data given as row numbers in worklog_2026-07-08.csv [2min]
- 2 Rewrite Then for 2 of the 4 yourself, in 3 lines: count in the warning area, in the list, and rows in the exported file [3min]
- 3 Save to e2e-scenarios.md [1min]

Result

tpro-work/e2e-scenarios.md (4 scenarios, 2 with Then written by you)

Name **which import stage to start from**

A prompt pattern for Agent. Fill in the <> parts from the e2e.js design 3 pages back.

```
Create e2e.js. Load it only when index.html is opened with <how to open>, and run the
  <scenarios> in #e2e-scenarios.md in order.
- Keep CSV rows as strings in e2e.js and pass them to <function for the later import stage>
- Skip <stages not covered>
- Use assert to check the counts in the warning area and the list, per <unit to count>
- For export, catch the Blob with <function to replace>, and pass it back to the import
- Reset to <state to restore> for each scenario
- Do not use <things not to use>. Show results at the top of the page in tests.html format
```

In the diff, check that the new loading line in index.html is skipped without ?e2e

Exercise 2: Run with ?e2e and compare

If one does not match, that is the point of this exercise

Scenario

e2e.js runs the 4 scenarios in order on your index.html. Not all may PASS. You decide later what to fix for a failed one. First, compare each on-screen result with its Then.

Steps

- 1 Open tpro-work/index.html in the browser, add ?e2e to the end of the address and reopen it. Also check that no results appear without ?e2e [1min]
- 2 Compare each of the 4 results with its Then in e2e-scenarios.md [4min]

Result

PASS or FAIL for the 4 scenarios (results at the top of the page)

Map **which Then lines** the AI-written E2E test checks

Scenario

Agent wrote e2e.js. Even if all 4 PASS, a scenario may check none of the counts in its Then. Have Ask make a table first; then read the code.

Steps

- 1 Give Ask #e2e.js and #e2e-scenarios.md, and get a table of the assert line numbers that check each Then line per scenario. Unchecked lines get None
- 2 Pick one None row and search e2e.js yourself to confirm it is really unchecked
- 3 If it is unchecked, have Agent add an assert for that Then only, and reopen with ?e2e

Result

decision-log.md gets one line: the None Then line and whether you added an assert.

Until the client answers, **decide the handling yourself now**

- 1 Order them**
List the warnings after the first import and after reload in 2 columns, by warning code
- 2 Why codes remained**
For warning codes that also appear after reload, write one line in decision-log.md on why they remained
- 3 Why codes disappeared**
For warning codes gone after reload, write one line on why. In one of these 2 lines, also say how to handle rows identical to records already in the list until the client answers

Check against R-06 in the Import Specification and open-questions.md. If the spec does not decide, say so

Exercise 2: Isolate the failed scenario

Decide which file to fix, then let Agent fix it

Scenario

The cause can be in 3 places: how Then is written, the app code, or how e2e.js is built. If you ask for a fix without choosing, the AI tends to change Then or e2e.js to match the app. Then behavior that differs from the spec is locked in as PASS.

Steps

- 1 Give Ask the Then and the actual values of the failed scenario, and have it list possible causes in those 3 places. Do not ask for a fix yet [2min]
- 2 Decide the target yourself and have Agent fix only that file. Read the diff and Keep [2min]

Result

One fixed file: e2e-scenarios.md, the app code in index.html, or e2e.js

Exercise 4: Break it and see what fails

Break the code to see what the tests protect

Scenario

When all tests PASS, you cannot tell whether the code is protected or the tests check nothing. Disable the 15-minute step check in one place and compare how tests.html and ?e2e fail.

Steps

- 1 Copy worklog-core.js and keep the copy as worklog-core_ok.js
- 2 In Ask, find the if line that rejects values not in 15-minute steps. Type false && right after if (on that line and save
- 3 Reopen tests.html and index.html?e2e. In decision-log.md, write one line each: the failed unit test case IDs and the E2E FAIL lines
- 4 Copy worklog-core_ok.js back over worklog-core.js, and check that tests.html shows the original SUMMARY line again

Result

decision-log.md has 2 lines on unit and E2E failures, and worklog-core.js is restored.

Done when

Done when

- `index.html?e2e` shows 4 results, and without `?e2e`, `e2e.js` is not loaded
- `e2e.js` runs from `WorkLogApp.importText` and does not create a File
- The 2 Thens you rewrote are written as counts
- After the `e2e` run, opening without `?e2e` shows the initial 8 records again
- You can say what to fix for the failed scenario, and why
- `decision-log.md` has 2 lines on warning code differences between first import and reload

If you're stuck

- `WorkLogApp.importText` not found → The second import step is not separated. Add a line to your prompt to split out the step that takes a string
- Export starts a download and stops → Put the `URL.createObjectURL` replacement before the export
- Every row is a warning → `e2e.js` calls the earlier file-reading stage. Start from `importText`
- Reload count differs from Then → Decide where to handle it: scenario, app or open questions

If time allows

- Add a fifth scenario. Things you decide a person checks, such as how it looks in Excel, get a Manual mark in `e2e-scenarios.md` and are not automated

The tool for real operations is **only on the instructor's PC**

1

From import to warning count

Drop the July-August CSV with a real drag and drop, and check the warning count

2

From export to reload

Click the export button, drop the saved file in again, and check the count

3

Steps covered

Receiving the file, detecting the encoding, and saving the download

Playwright runs on Node. It cannot be installed on your PC today

Same counts, but **different code paths**

e2e.js

Passes a decoded string to importText, skipping file receiving and encoding detection. Export captures the Blob as a string. It runs in the browser, so run it after every fix

Playwright

Drops a real file and saves a real download, with encoding detection. It needs Node and admin rights, so today only the instructor's PC runs it

Run e2e.js daily; the real thing **once before release**

What to check	Where	When
Counts for import, warnings, list, totals, export and reload	e2e.js	After every fix, ?e2e
Drag and drop, saving downloads	Playwright (a PC with Node)	Once before release
Detecting encodings like Shift_JIS	Playwright or a person	When the import format changes
How it looks in Excel	A person	When export changes

In your e2e-scenarios.md, mark items that only real operations can check as Manual

Exercise 5: Add run results to the test report

Fill the Day 2 test report **with today's results**

Scenario

On Day 2 you submitted the report with all cases Not run. Today you ran tests.html and E2E, so add the results and resubmit. For manual items, write whether you checked them.

Steps

- 1 Paste the tests.html SUMMARY line and the 4 ?e2e results into report.md
- 2 Give Ask #report.md and have it split the Not run section into run results and manual items. Decide the pass/fail column yourself
- 3 Rebuild report.html and report.pdf with the same 4 steps as Day 2

Result

report.md and report.pdf in tpro-work are replaced with versions that include run results.

Completion criteria for Exercises 3-5

Read and break the E2E, then **fill in the report** to finish

Done when

- decision-log.md has the None Then line and lines on unit and E2E failures
- worklog-core.js is restored and the SUMMARY line matches Exercise 1
- report.pdf has run results, with FAIL and HOLD listed separately

If you're stuck

- report.pdf is still old. Reopen report.html, then save it as PDF
- Cannot find the line you broke. In Ask, send "Which line checks the 15-minute steps?"

If time allows

- Open data.csv in Excel and add how dates and hours look to the report's manual items

Break

After the break, **move today's steps into files**

After the break

Keep tpro-work open in VS Code.

Next section

You will create a skill, a custom agent and an instructions file for tests.

We resume in 10 minutes.

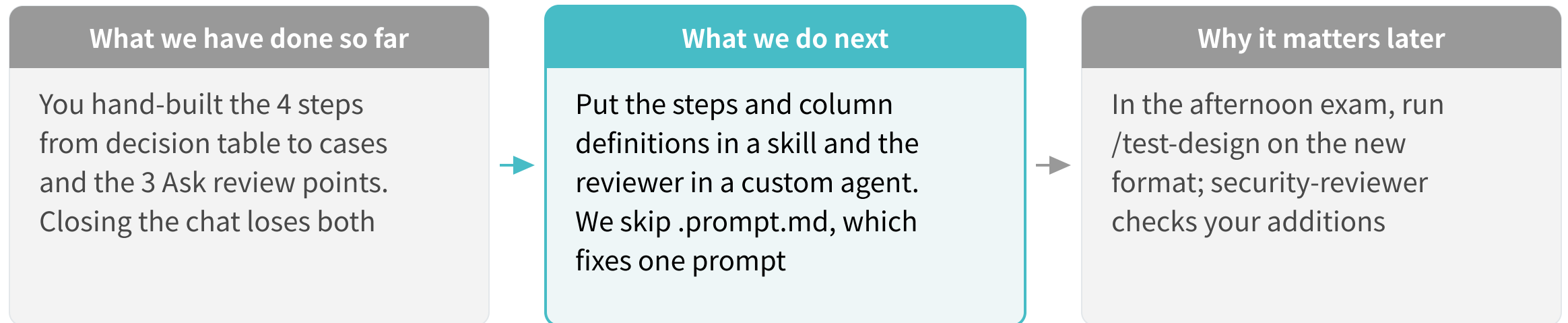
03

Saving steps and the reviewer

Make Day 2 test case design a skill, and the 3 review points sent before export a read-only reviewer

What is lost when you close the chat

Save **only prompts you will reuse next month** in files



3 places to store them

They load at different times, so write each thing once

Place	Files	When loaded	Made today
Instructions	instructions	Every chat, automatically; or for files matching applyTo	Instructions for tests
Skill	skills	When judged relevant, or from /	test-design, regress
Agent	agents	When the parent calls it as a subagent	security-reviewer

All 3 go under .github. The Day 1 copilot-instructions.md is also an instructions file

Only **agents** return results in a separate block

Agent Skills

A folder with `.github/skills/<name>/SKILL.md`. name and description are required; name matches the folder name, in lowercase and hyphens. Copilot loads it when the description looks relevant; you can also call it from `/`. Put column definitions and samples in references and refer to them. Results appear in the parent's response

Custom agent

A file at `.github/agents/<name>.agent.md`. Write description, tools and model in the frontmatter. Read-only tools keep the reviewer from changing files. When the parent Agent needs it, it runs as a subagent in its own context, and only its result returns in a separate block

How a subagent's model is chosen

A child does not run on a model **above the parent's**

Priority	Where it is set	How we use it today
1	A model name in the prompt that calls it	Not used today
2	The model field in .agent.md	Set it here today. Copy the picker label exactly, character for character
3	The model selected in the parent chat	If 2 is not set, the child runs on the parent's model

If you set a model above the parent's, the child does not run and you get a list of available models. It only works one way: a stronger parent and a lighter child

The AI does not write **assumptions you fixed midway** into steps

1

Recall your Day 2 test case design

Open testcases.md and open-questions.md, and find where you corrected the AI output

2

Write 3 lines

Write 3 assumptions in decision-log.md so the next user of this skill does not get stuck where you did

3

Do not send yet

Next, create it with /create-skill and compare the resulting SKILL.md with these 3 lines

Your 3 lines can differ from others'. Write them from what you actually fixed on Day 2

First decide just 2 things: a name and when to use it

1

Choose the parent model

In the model picker, pick a model with others above and below it, to try one above the parent later.

2

Type /create-skill and send

Type /create-skill in the input box, describe the skill you want in 1 or 2 English sentences, and send.

3

Answer the questions

When asked for the name, when to use it and files to reference, answer one at a time.

4

Read the diff and Keep

Read the body of SKILL.md, check that the steps match your assumptions, then Keep.

5

Check that / lists it

Type / in the input box and check that the skill name appears in the list.

Exercise 6: Extract the test-design skill

Read the generated steps and add missing assumptions

Scenario

On Day 2, 4 steps built the import section of testcases.md (you write conditions, decision table, add to the test case table, check requirement IDs). Make them reusable next month. Set the parent model below the top of the list, to try a higher one later.

Steps

- 1 In the model picker, set the parent model to one with models above and below it.
Create test-design with /create-skill [2min]
- 2 Check SKILL.md against the 3 lines in decision-log.md, add any missing, Keep [1min]
- 3 Create references/columns.md with the 8 column headings of testcases.md and how to fill each, and refer to it from the body [1min]

Result

tpro-work/.github/skills/test-design/SKILL.md and references/columns.md

Check the output on **a requirement not covered on Day 2**

Scenario

Apply your skill to R-08 (holidays), a requirement with no cases from Day 2. Check that the Source column has requirement IDs and undecided rows stay PENDING.

Steps

- 1 Run /test-design with import-spec_v1.md attached and get R-08 cases [2min]
- 2 Check that the table has the 8 columns of columns.md and fills in Source [1min]

Result

R-08 test case table (chat reply; not added to testcases.md yet)

Exercise 6: Create security-reviewer

Give the reviewer **no tools to edit**

Scenario

Turn the 3 review points you sent to Ask by hand before the export work into a read-only reviewer. The points are already in security-notes.md. Rewrite them as instructions. Give it a lighter model than the parent.

Steps

- 1 In the Command Palette, run "Chat: New Custom Agent" to create .github/agents/security-reviewer.agent.md [1min]
- 2 Write a one-line description of when it should be called [1min]
- 3 In the body, write the 3 points from security-notes.md as instructions [3min]
- 4 Limit tools to read-only ones, copy a model name below the parent's from the picker into model, and save [1min]

Result

tpro-work/.github/agents/security-reviewer.agent.md

Check on screen which, skill or agent, **gets its own section**

A template to send to the parent Agent. Fill in < > with your own files and requirements

```
Run the <agent name> agent as a subagent and have it review <files and places to check>.  
Then use the <skill name> skill to list missing cases for <spec and requirement ID>.  
Do not edit the 2 results; show them side by side. <What not to rewrite>.
```

Hover over the section and write the AI credits shown as one line in model-memo.txt

Exercise 6: Set a model above the parent's

Make it fail **once on purpose** and see the model list returned

Scenario

With a wrong model setting, the subagent does not run. See it once now, not during the practical exam. Change only the model line in security-reviewer.agent.md.

Steps

- 1 Change model to a model above the parent's, save, and send the same request as on the previous page [2min]
- 2 It does not run and returns usable models. Restore model and save [1min]

Result

security-reviewer.agent.md with the original model restored

Exercise 6: Turn one finding into a test

Expected values come only from **the new test case row**

Scenario

Pick one security-reviewer finding or R-07 gap from test-design. Add it to the test case table and tests. Expected values taken from the code lock in wrong behavior as PASS.

Steps

- 1 Pick one. Have it added to the Import section of testcases.md with a new ID, and a test with the same expected result added to tests/core.tests.js [2min]

Result

One row in testcases.md and one test in tests/core.tests.js

Done when

Done when

- SKILL.md has 3 lines of premises you added and refers to references/columns.md
- /test-design output has 8 columns and fills in Source
- security-reviewer.agent.md has read-only tools and model exactly as in the picker
- One security-reviewer section appears, and the test-design result is in the parent's reply. You can say in one sentence why a skill gets no section
- The parent's model is not the top one in the list
- You saw that a model above the parent's does not run and returns usable models

If you're stuck

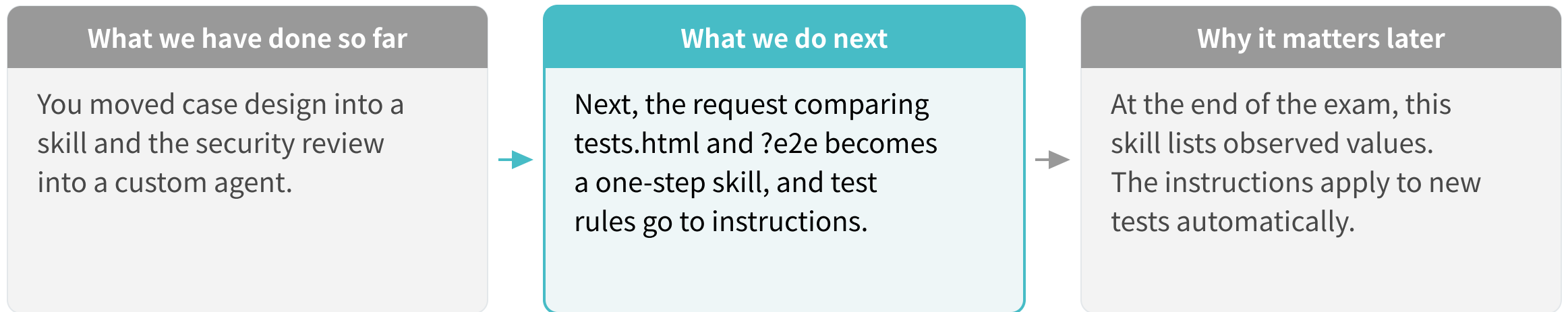
- /create-skill does not appear → If your version meets the minimum, use New Skill File... and have Agent write the body
- Subagent not called → Check that the mode is Agent and agent is on in the tool picker
- Name in model has no effect → It differs by even 1 character. Copy it again from the picker
- security-reviewer tries to fix code → Edit tools are still in tools

If time allows

- Add a third agent that first builds a column map when a CSV in another format arrives. You decide whether to use it in the practical exam

From skills and agents to small skills and instructions

If you type **the same request** each time, put it in a file too



Exercise 7: Make the regression check a small skill

Turn the request you typed per fix into **one skill called with /**

Scenario

Each time you fixed the import, you reopened tests.html and ?e2e and copied the SUMMARY line and the 4 results. Typing the same request every time is a sign to move it into a file. Make it a one-step skill with no reference files.

Steps

- 1 Type /create-skill and send "Make a skill named regress: compare the pasted SUMMARY line and ?e2e results with the last run in report.md; report only changes"
- 2 Open the new .github/skills/regress/SKILL.md, check that it states what to compare against and the report format, add missing lines and save
- 3 Paste the SUMMARY line, send /regress and compare with the Exercise 5 report counts

Result

tpro-work/.github/skills/regress/SKILL.md exists, and /regress returns a side-by-side comparison of results.

Exercise 8: Instructions for writing tests

Turn your usual test reminder into instructions **for tests only**

Scenario

Since Day 2 you have said "Take expected values from testcases.md, not the code" for every test. Instructions for the tests folder apply even when you forget.

Steps

- 1 Type /create-instructions and send "Create instructions that apply only to files in the tests folder. Take expected values from testcases.md; never copy them from worklog-core.js"
- 2 Check that applyTo at the top of the new file is tests/**. If not, fix it and save
- 3 Send Agent "Add one R-08 case to tests/core.tests.js" and check that the new file appears in the referenced files above the reply

Result

Instructions for tests exist under tpro-work/.github/instructions and are read when you have tests written.

Exercise 9: Improve a skill

When it misses, add one line to SKILL.md on the spot

Scenario

A skill is at its roughest on the day you create it. Add the points you fixed while using it, and next month you and your colleagues will not get stuck in the same place.

Steps

- 1 Run /test-design on R-09 (duplicates) and find one thing you want to fix in the returned test case table
- 2 Turn it into a one-line rule in the SKILL.md steps so it stays fixed, and save
- 3 Send the same request again and check that the new line takes effect

Result

SKILL.md has your added line, and the miss is gone in the second output.

Completion criteria for Exercises 6 to 9

You are done when **your own files** are in all 3 places

Done when

- Called test-design and security-reviewer, and can say only the agent got a section
- `.github/skills/regress/SKILL.md` exists and `/regress` returns a comparison
- The tests instructions use `applyTo tests/**` and are referenced when writing tests
- `SKILL.md` has a line you added after using it

If you're stuck

- `/create-instructions` does not appear: your version is old. Create a new file in `.github/instructions` by hand and have Agent write the body

If time allows

- Add one line to security-reviewer to check what did not fail in Exercise 4

04

Certification and Literacy Check

In the Literacy Check and the practical exam, show what you learned in 3 days with your own results

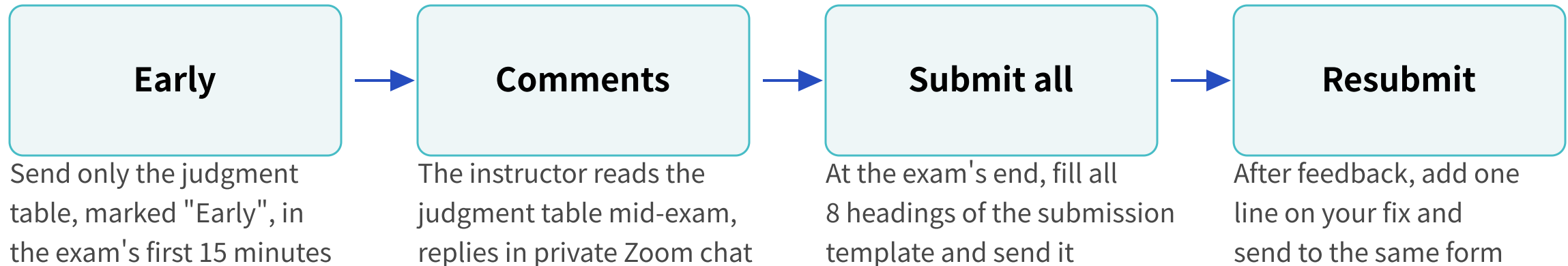
3 parts of the certification

None of the 3 is **meant to fail you**

Part	What is checked	What you submit
Literacy Check	Knowledge from the 3 days. You may use materials; just do not ask Copilot.	Text of your 20 answers
Practical exam	Judgment table handling and evidence match observed counts	Early judgment table, then submission template
Feedback session	Whether you can handle the returned comments yourself	Resubmit with a 1-line fix

Code tidiness and total hours are not checked in any part.

Submit only via **one text field** in the submission form



The 8 headings of the submission template

Spell warning codes **exactly as in the template**

Heading	What to write
1. Judgment table	Handling of J1-J9, Y/N for import and warning, and line numbers as evidence
2. Tests	The tests.html SUMMARY line, new-format prefix, and case IDs you added
3. First import counts	Rows imported, rows not imported, and counts by warning code
4. Explanation after reload	How the counts changed when you dropped the file again, and why
5. Subagent finding	One security-reviewer finding and how you handled it
6. Harness used	Names of files you opened. If you used the instructor's skeleton, its name too
7. Open issues	What you could not decide in time
8. 4 confidentiality rules	Check that you wrote no client names, real data, code or totals

How to take the Literacy Check

Hold off on Copilot **for this check only**

1

See in browser

Right-click day3/02_literacy-check.html and open it in the browser

2

Answer 20 questions

Allow about 1 minute per question. You may look at materials and your files from the 3 days

3

Skip questions you are unsure of

The unanswered count shows at the bottom. Go back to them at the end

If you finish early, reread today's decision-log.md.

The instructor counts the responses to confirm receipt

1

Review unanswered questions

Check that the unanswered count at the bottom is 0

2

Compile answers

Click the button at the bottom to show your answers as text

3

Paste into the form and send

In the submission form, write your name in "Name", paste the answers into "Text to submit" and send

Q7 and Q9 What tests can tell you

Miss these and you may trust **AI expected values** in the exam

Question	Correct understanding	Where it matters in the exam
Q7 Passing vs meeting the spec	Even if all your tests pass, gaps you wrote no tests for are unchecked	Judgment table J9, HOLD rows
Q9 Who sets acceptance criteria	A person decides what counts as done. Let AI suggest drafts	Writing expected values for new-format tests yourself

Output changes from September

From September, hours come from **the new attendance system**

Item	July and August	September
Columns	4: Date, Category, Description, Hours	6, with English headers and no hours column
Hours format	Hours as a number in one column	From start, end and break minutes
Category	4 names or an alias in the category map	Category code, mapped via the code table
Encoding	UTF-8	Shift_JIS

Where the excerpt is silent, **decide yourself** and write why

Import Specification v1

Applies in the exam too. 15-minute steps, categories not in the map, and holidays work as on Day 2

Judgment table

J1-J8 are scored. In J9, write one gap you found that the excerpt does not cover

Warning column

Show codes from the template's warning code table in the app's warning column

Instructor's skeleton

If stuck in the exam, we can give it to you. If used, list it under "Harness used" in the submission template

Extract into **day3** inside tpro-work

1**Get the ZIP**

The instructor posts 2 ZIPs, for Windows and Mac, in Zoom chat. Save the one for your OS

2**Extract into day3**

Windows: use Extract All and choose day3. Mac: move the new jitsugi folder into day3

3**Check the 5 files**

In the VS Code Explorer, check that jitsugi has 1 file in data and 4 in docs

If an opened CSV shows garbled text, leave it as is for now

Break

After the break: **the practical exam**

After the break

Check that day3/jitsugi has 5 files.

Next section

Read the timetable in the exam guide before you start.

We resume in 10 minutes.



05

Practical exam September attendance data

Import the new-format September data using only the instructions, tests and agents you built

Exam timetable

Each stage ends with **a good-enough line**

Stage	Part	Good enough when
Early judgment table	[15min]	Filled in and sent J1-J9
New-format import	[22min]	Matched one row with a hand calculation
New-format tests	[22min]	Added 10 or more
One E2E test	[14min]	One ran with ?e2e
Delegate to reviewer	[8min]	Wrote one line of findings
Observed values	[9min]	Copied the counts
Finish and submit	[18min]	Sent to the form

9 rows to decide in the judgment table

Write handling **in your own words**; do not pick from options

Row	What to decide
J1 Rows past midnight	Whether to count it on the start day or the end day
J2 Blank break	How to handle rows with an empty break field
J3 Break longer than work time	How to handle rows whose hours go negative after the break
J4 Break written as 1:00	How to read a break not written in whole minutes
J5 Hours not in 15-minute steps	How to handle start-end hours not in 15-minute steps

For each row, write handling, import Y/N, warning Y/N and evidence line no. The template has no options

9 rows to decide in the judgment table (cont.)

Row	What to decide
J6 MT and TR not in the code table	How to handle rows whose category code has no mapping
J7 Rows with overlapping times	How to handle 2 rows on the same day whose times overlap
J8 Rows matching records already in the list	How to settle the Day 2 HOLD items for September
J9 A gap you found yourself	Find and write one thing the excerpt does not cover

Send only the judgment table **first**, within 15 minutes

Scenario

From September, hours come from the new attendance system in Shift_JIS, with start, end and break times and category codes. The Day 2 import expects the old 4 columns.

Steps

- 1 Open day3/jitsugi/data/attendance_2026-09.csv in VS Code and write the mapping to the old 4 columns yourself. If garbled, use Reopen with Encoding [2min]
- 2 Pass the excerpt and column map to /test-design to get new-format viewpoints [2min]
- 3 Fill judgment table J1-J9 in submit_<surname>.md: handling, Y/Ns, evidence [7min]
- 4 Send only the judgment table, marked "Early", to the submission form [1min]

Result

Judgment table in tpro-work/submit_<surname>.md (sent early to the submission form)

Hand it to Agent **after deciding** to extend or split

Scenario

Import has 2 stages: file to string, then `WorkLogApp.importText`. Add the new encoding to the first and column mapping to the second. Show the template's warning codes.

Steps

- 1 Give Plan the contract, judgment table, excerpt, `worklog-core.js` and template [4min]
- 2 Extend the import or add a new function? Write one line in `decision-log.md` [2min]
- 3 Have Agent implement it in 2 stages; read the diff, then Keep [8min]
- 4 Hand-check 2 past-midnight rows, 1 blank-break row and 1 row off 15-min steps [6min]
- 5 Check warning column codes are spelled as in the template's warning code table [2min]

Result

New-format support in `worklog-core.js` and `worklog-config.js`

Group check 1

If import fails, **switch to the instructor's skeleton here**

Write in the chat

Write Yes or No twice: Did you send the judgment table? Did an imported row match your hand calculation?

If any answer is No

Get the instructor's skeleton for the import. Write the judgment table and test expected values yourself.

If both are Yes

Go on to the new-format tests.

Using the skeleton costs no points. Just note it under "Harness used"

Use only **expected values you wrote**

Scenario

tests.html counts PASS, FAIL and HOLD for the July and August cases. After adding new-format cases, show in the SUMMARY line that no July or August case turned to FAIL.

Steps

- 1 Choose a case ID prefix that marks the new format [1min]
- 2 Pick line numbers in attendance_2026-09.csv; write 10+ expected values yourself [11min]
- 3 Have Ask turn them into asserts, save to tests/ and reopen tests.html [6min]
- 4 Check FAIL_IDS in the SUMMARY line for IDs without your prefix, and decide where to fix new-format FAILs [4min]

Result

10+ new-format cases in tests/ and a new SUMMARY line

Give e2e.js **already decoded** strings

Scenario

e2e.js runs only with index.html?e2e and goes from import to export and reload. Encoding detection is in the stage before import, so e2e.js skips it.

Steps

- 1 Have Agent add one scenario to e2e.js that holds new-format rows as decoded strings [4min]
- 2 In e2e-scenarios.md, write its setup, steps and expected result in 3 lines [3min]
- 3 Open index.html?e2e and run from import through export and reload [3min]
- 4 Check that rows not in the code table and holiday rows show the warning codes from your judgment table [4min]

Result

One new-format scenario in e2e.js and 3 lines in e2e-scenarios.md

Group check 2

Use the rest of the time to **fill in the 8 headings**

Write in the chat

Write Yes or No twice: Do you have 10 or more tests? Did one scenario run with ?e2e?

If any answer is No

Stop there, write one line on how far you got under "Open issues", then move on.

Everyone

Open the 8 headings in submit and check which ones are still empty.

Writing unfinished work under "Open issues" is part of the submission, not a penalty

The reviewer **does not edit files**

Scenario

September notes also contain tags and rows starting with = + @. Have the read-only security-reviewer check your new functions for Day 2 holes coming back.

Steps

- 1 Set the mode to Agent and call security-reviewer as a subagent to review the functions you added [4min]
- 2 Pick one finding from its section, then open that file and line to check it [3min]
- 3 Write one line under "Subagent finding and fix" in submit [1min]

Result

One line under "Subagent finding and fix" in submit

Counts are checked **for the first import only**

Scenario

The instructor compares your counts with values computed from your judgment table's Y/N, and ignores totals. After reload, only your reason for the changed counts is read.

Steps

- 1 Reopen index.html to reset to the initial 8 records, and import September once [2min]
- 2 Copy rows imported, rows not imported and counts by warning code into submit [3min]
- 3 Reopen tests.html and copy the SUMMARY line as is into submit [2min]
- 4 Re-import the exported CSV and write in one sentence why the counts changed [2min]

Result

"Tests", "First import counts" and "Explanation after reload" in submit

Check the 4 rules one by one; paste in the submission form

Scenario

Submit text only. No code or real data: write decisions, evidence and observed values under the 8 headings of submit_template.md.

Steps

- 1 Write the case IDs you added, the harness file names you used and open issues [8min]
- 2 Reread the judgment table's evidence and post-reload notes. Fill missing headings [4min]
- 3 Match judgment table Y/N with first-run counts and the 4 confidentiality rules [4min]
- 4 Paste everything into the submission form and send [2min]

Result

All of submit_<surname>.md (to the submission form)

Done when: practical exam

Done when

- Judgment table J1-J9: every row has handling, both Y/Ns, evidence (line or requirement ID)
- You sent the judgment table once within 15 minutes of the start of the practical exam
- 10 or more new-format tests exist, and FAIL_IDS has no IDs without your prefix
- First import counts and counts by warning code are consistent with the judgment table's Y/N
- The submission text has no totals, real client names, real data or code

If you're stuck

- Every row gets a warning → You read with the old 4 headers. Go back to the column map
- Text is garbled → Handle the excerpt's encoding in the stage before import
- Hours go negative → Decide how to handle end-before-start rows, consistent with J1
- Out of time → Get the instructor's skeleton. If used, note it under "Harness used"

If time allows

- Move the new-format mapping into settings in worklog-config.js so one import switches between old and new formats. Check that old-format tests pass in both tests.html and ?e2e

Break

During the break, the instructor checks everyone's submission

Meanwhile

Results arrive by private Zoom chat: a heading name, or the word "Verified".

Next section

Handle the comments yourself and resubmit.

We resume in 10 minutes.

06

Feedback and handling comments

Handle returned comments yourself. If text is not enough, fix it with the instructor on screen

Feedback uses **the same criteria** as in the certification briefing

What feedback checks

Whether judgment table handling and evidence line numbers match. How you treated FAIL and HOLD. Which harness from the 3 days you used to import

What feedback does not check

Correct totals, tidy code, screen design. Different decisions give different totals, so decisions and counts just need to be consistent

Feedback room and main room

We call just those needing **screen sharing** to the feedback room

- 1 Read out names**
The instructor reads out names for the feedback room: 2 or more mismatches, or missing template parts
- 2 Move to the feedback room**
If called, move to the feedback room via Zoom breakout rooms and share your screen one at a time
- 3 You do the typing**
The instructor does not type, even in the feedback room. They tell you how to fix it, and you type
- 4 Continue in the main room**
If not called, start handling the comments you received

You decide: fix it, or write why you won't

If you fix it

Go to the heading named in the comment and check whether the judgment table or the counts were wrong. Then add one line on what you fixed

If you don't fix it

If you see no need to change your decision, write one line on why, with evidence line numbers. The instructor counts that line as handling it too.

Start working **without waiting** for the feedback room

Scenario

During the break, check results arrived by private Zoom chat: the name of a mismatched heading, or the word "Verified".

Steps

- 1 Put the heading in the comment next to your judgment table and counts [4min]
- 2 Decide to fix it or explain why not, and add one line to submit [6min]
- 3 Pick one open issue; fix it or add one line on why not [15min]
- 4 If you fixed anything, reopen tests.html and ?e2e for a new SUMMARY line [5min]

Result

One line in submit_<surname>.md on how you handled the comment

Resubmit to the same form **in the same template**

1

Return to the main room

Those in the feedback room return now too

2

Add one line on handling

At the end of the submission text, add one line on how you handled the comment

3

Paste into the same form

Send everything again, separate from the early judgment table and the first submission

Done when: feedback session

Done when

- Your resubmitted text has one line on handling the comment
- If you fixed something, the SUMMARY line is from after the fix

If you're stuck

- Not sure what to fix → Start with the first open issue
- Disagree with a comment → Writing why not, with evidence line numbers, is handling

If time allows

- Try one fix from the feedback room that also applies to you

Common stumbling points

Share **just one** issue seen in 3 or more people with everyone

Where it diverged

The instructor shows on screen where they diverged: judgment table, first counts or SUMMARY line

Why it happened

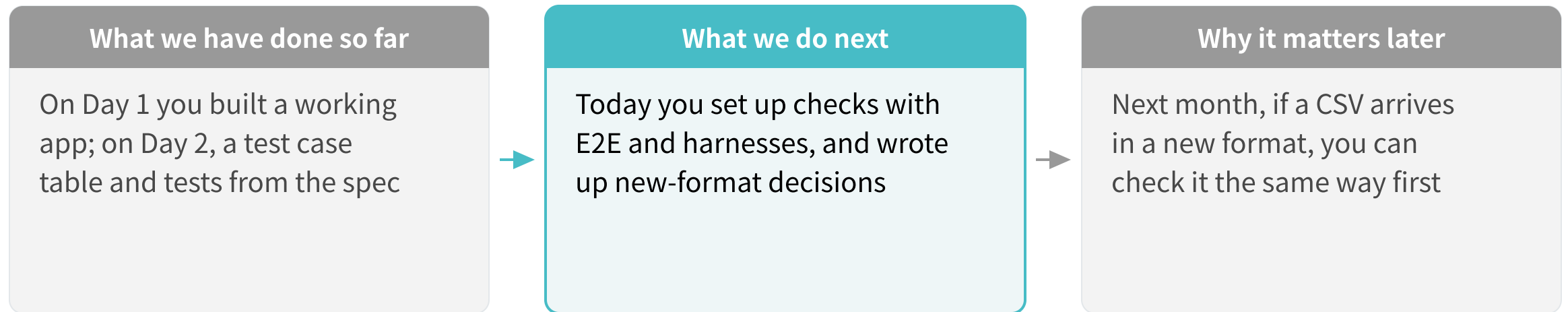
Tell apart a different decision, a copying error in observed values, or a wrong harness assumption

Preventing it next month

Show which file to add one line to so you don't get stuck there next month

What you achieved in 3 days

Today's certification: you can **check it yourself**, then submit



Getting your Certificate of Completion

The organizers send certificates **as PDFs**

Certificate of Completion

Here is a sample on screen. The issue date is October 31, 2026

How to receive it

The organizers send it as a PDF. Follow their notice for where and when

Pass or fail

There is no pass or fail by score

Checking the name list

Confirm how your name is written **in chat**

1

Read out the list

The instructor reads out the names on the certificate list

2

If you were not called

Send your name as it should appear to the instructor by private Zoom chat

3

Spelling differences

If the spelling or characters differ, also send it by private chat



07

Your 3-day results and next month's steps

Write down which files to open at the end of next month, and take it home

Deliverables from the 3 days

Beyond the app, you have **a repeatable way to check it**

Day	Location	Use next month
Day1	index.html and copilot-instructions.md	The Worklog app and always-on instructions
Day2	testcases.md and tests.html	A test case table with sources, and tests you recount just by opening
Day2	worklog-core.js and core-contract.md	Import logic and its contract. Fix these when the format changes
Day3	e2e.js and the regress skill	?e2e runs it all; /regress compares results
Day3	test-design and security-reviewer	Case design for new formats, and a watch on added code

The latest report.pdf is the one with run results

4 lines of steps for next month

Do not let AI write these 4 lines

Scenario

When the hours CSV arrives at the end of next month, you will start over from chat unless you have written down which files to open in which order.

Steps

- 1 Create tpro-work/README.md. On line 1, write what to drop onto which file [1min]
- 2 On lines 2 and 3, write what tests.html and ?e2e each check [2min]
- 3 On line 4, write what to change if the format changes, then save [2min]

Result

tpro-work/README.md (4 lines of steps)

If the README steps alone open both, you can take it home

1

Open tests.html

Open it as line 2 of the README says, and check that the SUMMARY line appears

2

Open ?e2e

Open index.html?e2e as line 3 of the README says, and check that results appear

3

Make a ZIP

Right-click tpro-work to ZIP the whole folder and save it. Check that .github is included

With .github in the ZIP, your 3 days of instructions, skills and agents open on next month's PC



Givery, Inc.

Teito Shibuya Bldg. 8F, 15-13 Nampeidai-cho, Shibuya-ku, Tokyo 150-0036, Japan