

AUTOMATE **Advanced** 45 MIN · UPDATED 2026-09-30

Build an automated QA testing insight agent.

Build a Copilot Studio agent that reads test failure logs and produces diagnostic insight so engineers spend less time on manual triage.

MICROSOFT 365

SharePoint

Teams

WORKS WITH

Copilot Studio

Business Outcome

Faster triage of test failures, so engineering time goes to fixing bugs instead of reading logs to find them.

Workflow Steps

Step 1: Categorize your common failure types

SharePoint

Before building the agent, list the usual buckets your failures fall into – flaky test/environment issue, actual regression, test data problem, dependency/infra failure – so the agent classifies against a known taxonomy instead of inventing categories per run.

Step 2: Build the log-analysis agent

Copilot Studio

In Copilot Studio, have the agent read the failure log for a given test run and return the failure category, the specific error/stack trace it based that on, and whether the same failure has appeared in recent runs (suggesting flakiness vs. a new regression).

PROMPT

Here is the failure log from last night's test run for the checkout-flow suite. Categorize each failure as flaky/regression/data/infra, cite the specific error line for each, and flag any failure that also appeared in the prior 3 runs.

Step 3: Post the triage summary where the team works

Teams

SharePoint

Have the flow post the categorized summary as a formatted message (or an Adaptive Card, if you want engineers to react/claim items inline) into the team's existing triage channel before the daily standup, with failures grouped by category so the flaky-test pile is visually separate from likely regressions.

A sorted channel post gets acted on; a link to a raw log file usually doesn't.

Step 4: Let engineers confirm, then close the loop

Teams

Add a simple reply convention (a 📄 /📄 reaction, or a short reply) engineers use after they investigate to confirm or correct the agent's category, and review that feedback weekly.

A pattern of regressions getting mislabeled "flaky" specifically is the one you want to catch early, since it's the failure mode that lets a real bug ship unnoticed.

✓ Check the work

- Spot-check "flaky" classifications against actual re-run history — mislabeling a real regression as flaky lets a bug ship.
- Confirm cited error lines match the actual log, not a paraphrase.
- Track classification accuracy over a few weeks using engineer feedback and retune the category definitions if one bucket is consistently wrong.

Expected Outcome

Each test failure paired with a likely cause category and the specific log lines supporting it, posted where the team already triages.

Want help rolling this out across your team? **Go to Liminals.Tech** 📄