BA TOOLKIT · FREE TEMPLATE

Meeting Notes to User Stories with AI: the Receipts Method

A prompt, three buckets, a receipts table and a check script so every AI-written story proves where it came from.

By **Prachit Parikh**, business analyst · github.com/prachitparikh/ba-toolkit

By Prachit Parikh. Companion to the video "Meeting Notes to User Stories with AI" on my YouTube channel.

AI tools still add things to summaries that were never in the source. On Vectara's hallucination leaderboard (updated 22 Oct 2026), the best model added unsupported content to 1.8% of document summaries, and plenty of well-known models sit between 5% and 10%. When the output is a backlog, one invented story is real work nobody asked for. This kit makes every AI-written story prove where it came from.

Rule 1: feed it the transcript, not the summary

A summary is already one AI's guess about what mattered. Stories built from it are a copy of a copy. Microsoft Teams can now produce an AI recap without saving the transcript (Microsoft Learn, "Intelligent recap without saving transcripts", updated 26 Oct 2026). That's useful for compliance, but for a requirements meeting ask the organizer to keep the transcript, or there's nothing to check against.

Rule 2: no receipt, no requirement

Every story carries a receipt: speaker, timestamp, and the exact words copied from the transcript. Not reworded.

The prompt (paste and edit)

```
You are helping a business analyst turn a meeting transcript into user stories.

Rules:
1. For every requirement, quote the exact sentence from the transcript that supports it,
   with the speaker and the timestamp. Copy the words exactly. Do not reword the quote.
2. If you can't find a supporting sentence, don't write the requirement.
   Put it in a list called Questions instead, with what you'd need to know.
3. Put every item in one bucket:
   - Decided: someone who owns the decision said yes.
   - Wished: someone said "we should probably", "it'd be nice", "maybe".
   - Assumed: nobody said it, but a story would need it.
4. Write user stories only for Decided items.

Output a table with columns: ID | Story | Bucket | Speaker | Timestamp | Exact quote
Then the Questions list. Then the Assumed list.

Transcript:
<paste the transcript here>
```

Rule 3: decided, wished, assumed

Bucket	What it sounds like	Where it goes
Decided	"Yes, that's the rule." from the person who owns it	Backlog, as a user story
Wished	"It'd be nice if...", "we should probably...", "maybe"	A question to the product owner: yes or no?
Assumed	Nobody said it, the story needs it	Assumptions log, with an owner and a date

The check (two parts)

1. **Does the quote exist?** Check it mechanically. Don't ask the AI whether its own quotes are real. Ctrl+F each quote in the transcript, or run `check_receipts.py` below. No match, no story.
2. **Does the quote support the story?** Read the quote next to the story. About ten seconds each. Watch for words the story added: a format, a frequency, a "must".

Receipts table (copy into your tool)

ID	Story	Bucket	Speaker	Timestamp	Exact quote	Quote found?	Supports story?
A	Refunds over $500 need manager approval	Decided	Dana (Finance)	12:04	"Refunds over $500 need a manager's approval."	Yes	Yes

ID	Story	Bucket	Speaker	Timestamp	Exact quote	Quote found?	Supports story?
B	Customers can see their refund status	Wished	Leo (Support)	18:40	"It'd be nice if customers could see their refund status."	Yes	Yes, but it's a wish: ask the PO
C	The refund list must export to PDF and Excel every morning	Decided	Priya (Ops)	31:15	"Maybe we export the refund list to Excel at some point."	Yes	**No**: "maybe", Excel only, no schedule
D	Refunds older than 90 days are rejected	Decided	Dana (Finance)	22:10	"Anything older than 90 days, we reject."	Yes	Yes

Northstar Systems is a made-up client. All example data is fictional.

check_receipts.py

```
"""Receipts check: flag any story whose quote is not an exact match in the transcript.
Usage: python3 check_receipts.py transcript.txt stories.csv
stories.csv columns: id,story,bucket,speaker,timestamp,quote
"""
import csv, re, sys

def norm(s):
    s = s.replace('’', "'").replace('‘', "'").replace('“', '"').replace('”', '"')
    return re.sub(r'\s+', ' ', s).strip().strip('"').lower()

transcript = norm(open(sys.argv[1], encoding='utf-8').read())
bad = 0
with open(sys.argv[2], newline='', encoding='utf-8') as f:
    for row in csv.DictReader(f):
        q = norm(row['quote'])
        if q and q in transcript:
            print(f"OK    {row['id']}  quote found  -> now read it: does it support \"
{row['story']}\"?")
        else:
            bad += 1
            print(f"NO MATCH  {row['id']}  \"{row['quote']}\"  -> move to Questions, not the
backlog")
print(f"\n{bad} story(ies) without an exact receipt")
sys.exit(1 if bad else 0)
```

Tested on the fictional example: story C passes the script (the quote exists) and fails the read. That's why the check has two parts.

Bonus: run it in reverse

Give the AI your final stories plus the transcript and ask: "Which sentences in the transcript contradict any of these stories? Quote them with timestamps." It catches the moment someone changed their mind at minute 35.

Sources

- Vectara, Hallucination Leaderboard (HHEM), last updated 22 Oct 2026: https://github.com/vectara/hallucination-leaderboard
- Microsoft Learn, Intelligent recap without saving transcripts, updated 26 Oct 2026: https://learn.microsoft.com/en-us/microsoftteams/intelligent-recap-without-saving-transcripts
- Microsoft Support, Recap in Microsoft Teams ("Some AI-generated content might be inaccurate, incomplete, or inappropriate."): https://support.microsoft.com/office/meeting-recap-in-microsoft-teams-c2e3a0fe-504f-4b2c-bf85-504938f110ef

ABOUT THE AUTHOR

Prachit Parikh is a business analyst in Apex, NC. He turns what stakeholders ask for into requirements a team can test, checks the data with SQL, and sees changes through UAT. More free templates: github.com/prachitparikh/ba-toolkit. Videos: youtube.com/@PrachitParikh1. Website and portfolio: prachitparikh.github.io.