Time doesn't matter to me at all — I'll have everything ready whenever you pick it up. I got the Sheet ID from the URL (`15G8kZ64p_61y5NmLm23uP0G2A6lGODHe2isL4OamS0A`) and I know the exact column structure from what we built, so I have everything I need.

Here's the complete Make.com Scenario A configuration, ready to build step by step.

Make.com Scenario A — Complete Build Guide

"Slack Text Log → Interactions Sheet"

Before You Start

You'll need these accounts ready:

- **Make.com** free account at make.com
- **Slack** — you'll create one dedicated channel: `#bprm-log`
- **Claude API key** — get one at console.anthropic.com (costs ~$0.01 per log entry)
- **Google account** connected to the sheet

Step 0 — Create the Slack Channel

In your Slack workspace, create a private channel called `#bprm-log`. This is instructor-only. You'll send every log entry here and receive One-Pager responses here. Keep it clean — no other conversation.

Step 1 — Create the Scenario in Make.com

1. Log into Make.com → click **Create a new scenario**
2. Name it: `BPRM – Slack Log to Sheet`
3. You'll build 6 modules in sequence

Module 1 — TRIGGER: Slack → Watch Messages

```
None
App: Slack
Module: Watch Messages
```

```
Settings:
  Channel: #bprm-log
  Trigger for: New messages only
  Watch: All messages

Connection: Connect your Slack workspace
  (Make.com will ask to authorize — allow it)
```

> This fires every time any message is posted to `#bprm-log`.

Module 2 — FILTER: Is this a Log entry (not a bot reply)?

After Module 1, click the **wrench icon** on the connecting line → Add a Filter.

```
None
Filter Name: Is a log command (not a bot response)

Condition:
  Field:     1. Text  (the message text from Module 1)
  Operator:  Does not start with
  Value:     🤖

AND

  Field:     1. User  (the user ID from Module 1)
  Operator:  Is not equal to
  Value:     [your Make.com bot's user ID — fill in after first test run]
```

> This prevents the scenario from looping when the bot posts its own confirmation message back to the channel.

Module 3 — Claude API: Parse the Slack Message

```
None
App: HTTP → Make an API Key Request

URL: https://api.anthropic.com/v1/messages
Method: POST
```

```
Headers:
  x-api-key:         [your Claude API key]
  anthropic-version: 2023-06-01
  content-type:      application/json

Body (Raw / JSON):
```

Paste this exactly, substituting `{{1.text}}` where shown (Make.com will auto-map the Slack message text):

JSON

```
{
  "model": "claude-sonnet-4-20250514",
  "max_tokens": 1000,
  "system": "You are a relationship intelligence assistant for Waukee APEX, a
work-based learning program. You help the instructor log business partner
interactions. Extract structured data from instructor notes and return ONLY
valid JSON with no explanation or markdown. Fields: partner_id (slug like
acme-corp), interaction_date (YYYY-MM-DD, use today if not specified),
interaction_type (one of: Meeting, Email, Phone, Site Visit, Event, BCC Log,
Screenshot), summary (2-3 sentence plain English), sentiment
(Positive/Neutral/Negative/Uncertain), follow_up_needed (true/false),
follow_up_notes (string or null), source. Today's date is {{formatDate(now;
\"YYYY-MM-DD\")}}.",
  "messages": [
    {
      "role": "user",
      "content": "{{1.text}}"
    }
  ]
}
```

> **Cost note:** This single Claude call is ~$0.003. At 5 logs/week that's ~$0.78/semester.

Module 4 — Parse JSON from Claude's Response

None

```
App: JSON → Parse JSON
```

```
JSON String:
  Use the "content" array from Module 3's response.

  In Make.com's field mapper, navigate to:
  Module 3 → content[] → [first item] → text

  That text field contains Claude's JSON string. Map it here.
```

This converts Claude's text response into a structured data object Make.com can use.

Module 5a — Google Sheets: Add Row to Interactions Tab

None

```
App: Google Sheets
Module: Add a Row

Spreadsheet ID: 15G8kZ64p_61y5NmLm23uP0G2A6lGODHe2isL4OamS0A
Sheet Name: 4_Interactions
```

Map each column **exactly** as follows (column order matters):

Column	Map to
A — interaction_id	`INT-{{formatDate(now; "YYYYMMDD")}}-{{random(100; 999)}}`
B — partner_id	`{{4.partner_id}}` (from JSON parse)
C — project_id	*(leave blank for now)*
D — date	`{{4.interaction_date}}`
E — type	`{{4.interaction_type}}`
F — logged_by	`Russ`
G — summary	`{{4.summary}}`
H — sentiment	`{{4.sentiment}}`
I — follow_up_needed	`{{4.follow_up_needed}}`
J — follow_up_notes	`{{4.follow_up_notes}}`

K — source `Slack`

Module 5b — Google Sheets: Update Partners Tab

```
None
App: Google Sheets
Module: Update a Row

Spreadsheet ID: 15G8kZ64p_61y5NmLm23uP0G2A6lGODHe2isL4OamS0A
Sheet Name: 1_Partners

Search Column: A (partner_id)
Search Value:  {{4.partner_id}}

Values to update:
  Column H — last_interaction_date: {{formatDate(now; "YYYY-MM-DD")}}
```

> This keeps the Partners tab current automatically every time you log.

Module 6 — Slack: Post Confirmation Back to Channel

```
None
App: Slack
Module: Create a Message

Channel: #bprm-log

Text:
🤖 Logged ✓
*{{4.partner_id}}* — {{4.interaction_type}} on {{4.interaction_date}}
Sentiment: {{4.sentiment}} | Follow-up: {{4.follow_up_needed}}
_{{4.summary}}_
```

> The 🤖 prefix is what the Module 2 filter looks for to prevent looping.

Testing It

Once built, send this test message to `#bprm-log`:

```
None
Update acme-corp: Met with Dana today. She approved the final logo
concepts and wants to discuss a potential fall project.
Very positive meeting.
```

You should see a new row appear in `4_Interactions` within ~15 seconds and a confirmation post back in Slack. If it works — the whole system is live.

Your Slack Command Conventions (save this somewhere)

```
None
# Log an interaction
Update [partner-id]: [your notes in plain English]
Log [partner-id]: [notes]

# Request a One-Pager (Scenario C — we'll build this next)
Summary [partner-id]
One-Pager [partner-id]

# Examples
Update metro-health: Check-in with James. Students presented
midpoint research. He wants final deck by May 9.

Update summit-eng: Tyler hasn't responded to the scoping template.
Following up next week.
```

Pick this up whenever you're ready. When you come back, if you run into any snags during the Make.com build, just describe what you're seeing and we'll troubleshoot it. Scenario C (the One-Pager) is a quick build once A is working — maybe 20 minutes.