Dual-Model Failover: Engineering Independent Backoff Clocks for LLM APIs

Setting the Stage: Context for the Curious Book Reader

In automated publishing pipelines that rely on cloud-hosted LLM endpoints, availability is rarely guaranteed. Quota drops, high demand, and transient 500 errors routinely interrupt workflows. A common anti-pattern is imposing a single shared backoff timer across candidate models, which unnecessarily penalizes healthy fallback models when a primary model is throttled. This entry captures a surgical refactor of articleizer.py to establish independent per-model backoff clocks. By decoupling retry penalties, the pipeline enables immediate failover from Gemini Flash to Gemini Flash Lite while strictly preserving verifiable receipts, AST integrity, and respectful rate-window adherence.

TL;DR: The article publishing pipeline was changed from a single-model retry loop to a two-model failover scheduler. It now tries Gemini Flash first, switches immediately to Gemini Flash Lite after a retriable failure, alternates between the two, and maintains an independent exponential-backoff clock for each model while honoring that model's server-provided retry hint. Static source inspection, Python AST parsing, and Ruff validation confirm the implementation structure; the next normal publication run remains the live runtime witness.

Technical Journal Entry Begins

🔗 Verified Pipulate Commits:

- 8ce6d860 (raw)
- 3fdadaa3 (raw)
- 169f1fa5 (raw)
- 088f1cf7 (raw)
- a634740b (raw)

MikeLev.in: This is one I think of whenever I switch these two models around because of availability. Sometimes Gemini Flash Latest is being hammered and is unreachable and sometimes Gemini Flash Light Latest is being hammered and is unreachable. I prefer to try the more capable Gemini Flash Latest model first because it's more capable and I do notice a qualitative improvement like longer intros when I use that model so let's always try it first, but if it's unresponsive instead of the progressive backoff, try the other model immediately and then alternate back and forth between the two models with progressive backoff equally applied to each model please and thank you.

So we set context. I can "go big" and include the whole chunk of traditional exposition

storytelling of this system now which is quite big because we're almost fifteen hundred articles into it now and even just the book-ore spine is big. But we can also "go small" setting context which is often quite appropriate to get in and get out on a nested subproject like this where the 40K-foot view is not necessary, but that's fine because even with just light Ad Hoc context the model still has plenty to see what kind of framework it's working with and even to ask to drill-down more with the Next Context command if it really wants to. But we'll go small now.

Designing Independent Backoff Clocks

I edit-in merely the publishing pipeline chapter. All we probably need is articleizer.py so the whole chapter is probably overkill, but still between the apply.py file that's always included plus the adhoc.txt file and this chapter, the LLM has a 40K-foot view of the 40K-foot view and could grab the details it wants, but you'll see it probably won't because the main goal is that whole weird convolution of fall-back alternating progressive back-off.
I can haz?
ChatGPT 6 Extra High: Context verified. I followed the cartridge's declared final-Prompt entrypoint. The local imports that matter here (common, lsa) are included, time is already imported, and the model/retry behavior is localized enough that no additional file is needed.
Your desired semantics map cleanly to **independent per-model backoff clocks**:
Flash → Lite immediately → Flash when Flash's own cooldown permits → Lite when Lite's own cooldown permits → …
That is better than one global delay. A hammered Flash does not punish Lite; a hammered Lite does not punish Flash. The existing server-provided retry in Ns hint remains honored, but only for the model that emitted it. Five attempts per model gives five alternating rounds rather than silently reducing either model's opportunities.
I also dry-applied the patches against the supplied articleizer.py source: all five SEARCH anchors matched exactly once and the resulting Python passes ast.parse().

1. PROBES

The first command is the cheapest falsifier: it confirms that model selection and API acquisition really are confined to this retry site. The second is a read-only syntax check and becomes the symmetric AFTER receipt next compile.

```
rg -n
"DEFAULT_MODEL|MODEL_CANDIDATES|gemini-flash|retry_delay|retry_after|l
lm\.get_model" scripts/articles/articleizer.py
.venv/bin/python -c "import ast, pathlib;
ast.parse(pathlib.Path('scripts/articles/articleizer.py').read_text())
; print('articleizer.py AST OK')"
```

2. NEXT CONTEXT

These are the exact adhoc.txt lines I would carry forward. The first two are deliberately identical to the probes above, with only the required leading ! added.

```
! rg -n
"DEFAULT_MODEL|MODEL_CANDIDATES|gemini-flash|retry_delay|retry_after|l
```

```
lm\.get_model" scripts/articles/articleizer.py
! .venv/bin/python -c "import ast, pathlib;
ast.parse(pathlib.Path('scripts/articles/articleizer.py').read_text())
; print('articleizer.py AST OK')"
scripts/articles/articleizer.py
```

3. PATCHES

Car 1 turns the single-model constant into an ordered candidate pair and gives the backoff parameters names matching their actual semantics.

```
Target: scripts/articles/articleizer.py
[[[SEARCH]]]
# Model Selection - Use a stable model to avoid low quotas
# DEFAULT_MODEL = 'gemini-flash-latest'
DEFAULT_MODEL = 'gemini-flash-lite-latest'
[[[DIVIDER]]]
# Model Selection - Prefer Flash, then fail over to Lite immediately
on a
# retriable availability error. Each model keeps its own
exponential-backoff
# clock so alternating retries do not make one inherit the other's
penalty.
MODEL_CANDIDATES = (
    'gemini-flash-latest',
    'gemini-flash-lite-latest',
)
MAX_ATTEMPTS_PER_MODEL = 5
INITIAL_RETRY_DELAY = 2
[[[REPLACE]]]
```

Car 2 replaces the global retry counter/clock with strict alternation and a cooldown clock per model. Notice the key behavior: Lite begins with retry_after == 0, so the first Flash failure falls through to Lite without sleeping.

```
Target: scripts/articles/articleizer.py
[[[SEARCH]]]
        print(f"Calling the Universal Adapter (using
{DEFAULT_MODEL})...")
        max_retries = 5
        retry_delay = 2
        for attempt in range(max_retries):
            try:
                # Use a free-tier compatible model.
                model = llm.get_model(DEFAULT_MODEL)
[[[DIVIDER]]]
        print(
            "Calling the Universal Adapter "
            f"(primary {MODEL_CANDIDATES[0]}, fallback
```

```
{MODEL_CANDIDATES[1]})..."
        )
        total_attempts = MAX_ATTEMPTS_PER_MODEL *
len(MODEL_CANDIDATES)
        retry_delays = {name: INITIAL_RETRY_DELAY for name in
MODEL_CANDIDATES}
        retry_after = {name: 0.0 for name in MODEL_CANDIDATES}
        for attempt in range(total_attempts):
            model_name = MODEL_CANDIDATES[attempt %
len(MODEL_CANDIDATES)]
            model_attempt = (attempt // len(MODEL_CANDIDATES)) + 1
            wait = max(0.0, retry_after[model_name] -
time.monotonic())
            if wait:
                print(
                    f"Waiting {wait:.0f} seconds before retrying
{model_name} "
                    f"(Attempt
{model_attempt}/{MAX_ATTEMPTS_PER_MODEL})..."
                )
                time.sleep(wait)

            try:
                model = llm.get_model(model_name)
[[[REPLACE]]]
```

Car 3 makes the winning model visible in the receipt.

```
Target: scripts/articles/articleizer.py
[[[SEARCH]]]
                print("Successfully received response from API.")
[[[DIVIDER]]]
                print(f"Successfully received response from API via
{model_name}.")
[[[REPLACE]]]
```

Car 4 is the mechanism. A retriable failure advances **only that model's** exponential clock. The next model is then selected by alternation. If its own clock is clear—as Lite's is on the first Flash failure—it runs immediately.

```
Target: scripts/articles/articleizer.py
[[[SEARCH]]]
                    print(f"Retriable API Error: {e}")
                    # QUOTA WINDOW DISCIPLINE (live-fire convicted
2026-07-19):
                    # the free tier is a per-MINUTE rolling window;
retrying at
                    # 2/4/8/16s hammers the same window the server
asked us to
                    # let drain. Honor the server's own "retry in Ns"
```

```
hint.
                    hint = re.search(r'retry in (\d+(?:\.\d+)?)s',
error_str)
                    wait = max(retry_delay, float(hint.group(1)) + 1)
if hint else retry_delay
                    print(f"Retrying in {wait:.0f} seconds... (Attempt
{attempt + 1}/{max_retries})")

                    time.sleep(wait)
                    retry_delay *= 2  # Exponential backoff
[[[DIVIDER]]]
                    print(f"Retriable API Error from {model_name}:
{e}")
                    # QUOTA WINDOW DISCIPLINE (live-fire convicted
2026-07-19):
                    # each model gets its own exponential clock. A
failure on
                    # Flash therefore falls through to Lite
immediately, while
                    # the failed model honors its own server hint
before its
                    # next turn in the alternation.
                    hint = re.search(r'retry in (\d+(?:\.\d+)?)s',
error_str)
                    hinted_delay = float(hint.group(1)) + 1 if hint
else 0
                    wait = max(retry_delays[model_name], hinted_delay)
                    retry_after[model_name] = time.monotonic() + wait
                    retry_delays[model_name] *= 2

                    if attempt + 1 < total_attempts:
                        next_model = MODEL_CANDIDATES[(attempt + 1) %
len(MODEL_CANDIDATES)]
                        next_wait = max(0.0, retry_after[next_model] -
time.monotonic())
                        if next_wait:
                            print(
                                f"Switching to {next_model}; its
backoff has "
                                f"{next_wait:.0f} seconds remaining."
                            )
                        else:
                            print(f"Switching to {next_model}
immediately.")
[[[REPLACE]]]
```

Car 5 fixes the exhausted-retries message so it describes the new two-model reality.
Target: scripts/articles/articleizer.py

```
[[[SEARCH]]]
        else: # This block runs if the loop completes without a break
            print("Error: Max retries exceeded. Failed to get a
successful response from the API.")
[[[DIVIDER]]]
        else:  # This block runs if the loop completes without a break
            print(
                f"Error: {MAX_ATTEMPTS_PER_MODEL} attempts per model
exhausted. "
                "Failed to get a successful response from the API."
            )
[[[REPLACE]]]
```

Ignition: no shell/Nix restart is required. The next normal article / grim / bot publication invokes articleizer.py afresh and exercises the new logic. The important runtime pattern is now roughly $\text{Flash}_1 \rightarrow \text{Lite}_1 \rightarrow$ wait for Flash_1's clock if necessary $\rightarrow \text{Flash}_2 \rightarrow \text{Lite}_2 \rightarrow \ldots$, with each model's delay progressing independently.

4. PROMPT

```
Review the live receipts for the Gemini articleizer failover patch.

Confirm that articleizer.py now:
1. prefers gemini-flash-latest first,
2. falls through immediately to gemini-flash-lite-latest on a
retriable first failure,
3. alternates strictly between the two models,
4. maintains independent exponential-backoff clocks for the two
models,
5. honors each model's existing "retry in Ns" hint without
transferring that penalty to the other model,
6. remains syntactically valid.

Pay special attention to the BEFORE/AFTER rg receipt and the AST
receipt. If the source proves those invariants, say so and identify
the cheapest useful live-fire witness for the next normal publishing
run. Do not broaden the change unless the receipts expose a concrete
problem.
```

5. EXTERNAL DELIVERABLES

None. This is entirely a repo-local articleizer.py change; no PageWorker, CMS, dashboard, or other manual-paste artifact is required.

MikeLev.in: Dry-applied patches, huh? That's your use of a sandbox evidence given right there. I'm pretty sure these frontier models that help you code are using Python in the background on a sandbox.

Same commands, run twice, one change between them. Where the readings differ is what the

change did; the diff in the middle is the receipt.

Verifying the Baseline Probe Before Mutation

1: Probe: (BEFORE: hand-run, nothing changed yet)

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(nix) pipulate $ rg -n
"DEFAULT_MODEL|MODEL_CANDIDATES|gemini-flash|retry_delay|retry_after|l
lm\.get_model" scripts/articles/articleizer.py
.venv/bin/python -c "import ast, pathlib;
ast.parse(pathlib.Path('scripts/articles/articleizer.py').read_text())
; print('articleizer.py AST OK')"
23:# DEFAULT_MODEL = 'gemini-flash-latest'
24:DEFAULT_MODEL = 'gemini-flash-lite-latest'
451:        print(f"Calling the Universal Adapter (using
{DEFAULT_MODEL})...")
453:        retry_delay = 2
457:                model = llm.get_model(DEFAULT_MODEL)
488:                    wait = max(retry_delay, float(hint.group(1)) +
1) if hint else retry_delay
492:                    retry_delay *= 2  # Exponential backoff
articleizer.py AST OK
(nix) pipulate $
```

2: Context: (AFTER: the same probes re-run by the compiler as ! lines)

```
# AD HOC CHOP! The Not-Managed-by-Git Safe-for-Client-Data place.
Insert Simpson Couch Gag in white space  below (explain anything to
the audience you feel needs it explained)G
# adhoc.txt     _   _   _               ____ _   _  ___  ____  _
#     / \   __| | | | | ___   ___   / ___| | | |/ _ \|  _ \| |  This
Gemini model availability on article publishing has been bothering me
for awhile.
# ahe/ _ \ / _` | | |_| |/ _ \ / __| | |   | |_| | | | | |_) | |
# ahc ___ \ (_| | |  _  | (_) | (__  | |___|  _  | |_| |  __/|_|
#  /_/   \_\__,_| |_| |_|\___/ \___|  \____|_| |_|\___/|_|   (_)

# To Human: WELCOME TO VIM! It's really NeoVim but I say vim anyway.
# To AI: This is an alternate 40K view to the big book-ore rolling pin
spine.

# 1. **Probe**: Baseline Reading
# 2. **Context**: Post-experiment *planned* reading instructions
# 3. **Patch**: The experiment and how to make it happen
```

```
# 4. **Prompt**: Post-experiment instructions and how to read results
# 5. **Deliverable**: How the world is forever different moving
forward

# The first thing you need to know here is that everything that comes
after the
# hash symbol (#) is commented out — and that's EVERYTHING in this
file's default
# state. Begin editing-in lines for inclusion as part of the context
or adding
# chunks of new context at the bottom. `Ctrl`+`v`, `j` (repeatedly),
`l` (to move
# right), `d` (to delete). Reverse that with `Ctrl`+`v`, `j`
(repeatedly),
# `Shift`+`i`, `# `, `Esc` to put the hashes back. You can just
arrow-key around
# here with `h`, `j`, `k`, `l`. Save-and-quit is a bit tricky because
another
# file is also loaded: `Esc`, `:`, `q`, `w`, `!`

# If this is stressing you out and you're a quitter and want to quit,
just type:
# `Esc`, `:`, `q`, `!`, `Enter`. That will exit without saving any
changes. If
# you want to get over this hump, type: `Esc`, `:`, `T`, `u`, `t`,
`o`, `r`, `Enter`.

# This file is just to make it easy having options of what to edit
into context.
# You can use whatever text-file you want to stack file-names and
commands to
# build an output text-file with the identically stacked output of
each file or
# command. In this way we vertically append or "stack" a bunch of
text; simple as
# that. If you understand this concept, you're on your way to
future-proofing
# yourself in the Age of AI. Congratulations! Here is how to include
web pages:

#    !URL
-----------------------------------------------------------------------
#      when    Public page; what a stranger or crawler sees; the
BEFORE of a
#              login-wall diagnosis
#      switch  It shows a login page -> `warm URL` once, then `?URL`
#
#    ?URL
```

```
-------------------------------------------------------------------
#      when    Anything behind a login, on the site's persistent
profile;
#              `check URL` first
#      switch  The lenses show a shell (nav, an `[Iframe]` leaf, no
content) ->
#              read the wire truth for the XHR the frame makes, then
call that
#              API with a connector
#
#    @URL
-------------------------------------------------------------------
#      when    Every re-read of a page already scraped; no browser, no
network
#      switch  The cached page is stale or was a login wall -> fresh
`!` or `?`
#
#    $URL
-------------------------------------------------------------------
#      when    Exact markup: meta tags, a JSON blob in a `<script>`
#      note    Token-heavy; needs a prior scrape
#
#    %URL
-------------------------------------------------------------------
#      when    The network log distilled; SPA endpoint discovery
#      switch  It re-serves the wire truth you already have -> the API
#
#    ! cmd
------------------------------------------------------------------
#      when    Any bounded, non-interactive command as a live receipt
#      note    Cap it with `-n`; no aliases, no prompts
#
#    Connector
-------------------------------------------------------------
#      when    The number you want is one GET away
#      switch  LIST until the thing isn't in the list -> FETCH by id
-> DRILL
#              the path the app's own frame called -> `--grep` to
narrow a list
#              or find a leaf

# Every step is one argument longer than the last; the moment a lens
shows less than the wire does is the moment to stop scraping.

# FOR 40K-FT VIEW (STORY & INFRASTRUCTURE) !!
# --- START EDITING-IN ON 1ST TURN ---

! python scripts/articles/lsa.py -t 1 --reverse --fmt dated-slugs  #
```

```
<-- ROLLING PIN that gives the 40K foot book-spine view of book-ore
(only works for me because of local-only git repo)
# ~/repos/nixos/autognome.py  # <-- You wake up in the morning and
your Tooling & Instrumentation folds out of you like Inspector Gadget.
# init.lua                    # <-- Those gadgets are made easy-to-use
through nifty keyboard shortcuts (but ya gotta learn vim).
GLOSSARY.md                 # <-- Like the back of a J.R.R. Tolkien
book, there's kooky new terms to know.
flake.nix                   # <-- Here is my hardware. Here is my
state. Put on your sandbox. And please recreate. (Infrastructure as
Code / IaC)
# assets/installer/install.sh # <-- Pipulate.com installer real home
in github/pipulate repo
# prompt_foo.py               # <-- THIS system
foo_files.py                # <-- main ROUTER
# requirements.in             # <-- We've "pinned" everything but
still want a flexible Python Data Science virtualenv.
# pyproject.toml              # <-- How this is a citizen of the
Python "pip install" ecosystem
# __init__.py                 # <-- Version info

# --- END EDITING-IN ON 1ST TURN ---

# scripts/articles/lsa.py     # <-- 2ND BRAIN: Search external memory
with `rgx`, `rgxc` & `posts` Blogging for Hackers Jekyll-compatible.

# OPTIONAL ACTUATORS (cheap and good to include to expand the AI's
capabilities)
# cli.py                      # <-- Catch-all actuator for PyPI envs,
Python anchoring, MCP tool-call (plus alternatives) and **kwargs like
wrapping for CLI
# scripts/xp.py               # <-- Transforms host OS copy-paste
buffer player-piano music into context-payload.
# scripts/ai.py               # <-- How I constantly use local AI to
write git commit messages with `m` alias.
# scripts/crawl.py            # <-- Feel free to ask for something to
be crawled and included in the next turn.
# scripts/weblogin.py         # <-- Lets the user "warm up" the cache
for their web logins at their leisure on a profile that persists.
# scripts/webclip_2_markdown.py  # <-- Surprisingly important program.

# MISCELLANEOUS (rare to include but sometimes critical)
# scripts/foo_cartridge.py    # Needs description
# scripts/foo_replay.py       # Needs description
# release.py                  # <-- How everything ends up where it
does (GitHub, PyPI, etc.)
# imports/voice_synthesis.py  # <-- The wand can talk to you
# imports/ascii_displays.py   # <-- Where all the ASCII Art lives
```

```
# scripts/release/version_sync.py  # <-- Needs to be wrapped into
release.py and eliminated, I think.

#                          --- Under this line is were you paste what
the AI gives you ---
#                          --- We call it context but it's really just
the right-hand  ---
#                          --- blast-radius of the "probes" to make
this all science.  ---

# Carry-over as the important work-in-progress parts of the project
here just
# like above but not as long-standing overarching to the framework but
rather
# for the current hot spots actively being worked on.

# --- START THIS DISCUSSION ---

# Get things started here! Guess at what context should be included.
# If you get it wrong, you're just wasting 1-turn because the AI will
help.
# Un-comment lines, add lines with absolute-path filenames or `! `
commands.

# Context 1 (Edit-in selections from above and add new files
immediately below)
# ~/.config/pipulate/blogs.json                # <-- CAUTION! Derived
from ~/repos/nixos/blogs.nix
# scripts/articles/publishizer.py              # <-- Orchestrates
different publishing workflows per target blog.
# scripts/articles/common.py                   # <-- Self-explanatory
# scripts/articles/articleizer.py              # <-- Transforms raw
article.txt to formal Jekyll markdown format
# scripts/articles/editing_prompt.txt          # <-- Forcing response
into strict JSON data structure
# scripts/articles/sanitizer.py                # <-- Scrubs PII
# scripts/articles/gsc_historical_fetch.py
# scripts/articles/contextualizer.py           # <-- Builds JSON
summaries of articles in `_posts/context/` called "Holographic
Shards".
# scripts/articles/confluenceizer.py           # <-- Idempotent
Jekyll-to-Confluence corporate wiki
# scripts/articles/googledocizer.py            # <-- Just added
# scripts/articles/build_knowledge_graph.py    # <-- Topically
load-balances site using hierarchical K-Means keyword clustering
groups
# scripts/articles/generate_ai_context.py      # <-- AIs WILL
interrogate your repo. This gives epic context of article URLs for
```

```
drill-down.
# scripts/articles/generate_hubs.py            # <-- Uses
just-produced link-graph data to generate each of the new hubs it
suggests
# scripts/articles/generate_llms_txt.py        # <-- Builds an
llms.txt based on the auto-organized structure suggested here
# scripts/articles/generate_redirects.py       # <-- Generates
redirect map above hub-churn suggests is needed
# scripts/articles/sanitize_redirects.py       # <-- Deals with
follow-up meticulous pedantic detail required for a good Nginx
redirect map
# scripts/articles/link_injector.py
# scripts/articles/generate_post_nav.py  # [1,174 tokens | 4,616
bytes]

# Context 2
! rg -n
"DEFAULT_MODEL|MODEL_CANDIDATES|gemini-flash|retry_delay|retry_after|l
lm\.get_model" scripts/articles/articleizer.py
! .venv/bin/python -c "import ast, pathlib;
ast.parse(pathlib.Path('scripts/articles/articleizer.py').read_text())
; print('articleizer.py AST OK')"
scripts/articles/articleizer.py

# --- END `adhoc.txt` TEMPLATE ---
```

3: Patches: (the one change between the readings)

```
Blast Radius Check to establish bisection Left-hand Causal Boundary.
It is a Popper-thing. Science.
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/articleizer.py'.
(nix) pipulate $ d
diff --git a/scripts/articles/articleizer.py
b/scripts/articles/articleizer.py
index 9a5ece9a..1b6980ba 100644
--- a/scripts/articles/articleizer.py
+++ b/scripts/articles/articleizer.py
@@ -19,9 +19,15 @@ PROMPT_FILENAME = "editing_prompt.txt"
 PROMPT_PLACEHOLDER = "[INSERT FULL ARTICLE]"
 INSTRUCTIONS_CACHE_FILE = "instructions.json"

-# Model Selection - Use a stable model to avoid low quotas
```

```
-# DEFAULT_MODEL = 'gemini-flash-latest'
-DEFAULT_MODEL = 'gemini-flash-lite-latest'
+# Model Selection - Prefer Flash, then fail over to Lite immediately
on a
+# retriable availability error. Each model keeps its own
exponential-backoff
+# clock so alternating retries do not make one inherit the other's
penalty.
+MODEL_CANDIDATES = (
+    'gemini-flash-latest',
+    'gemini-flash-lite-latest',
+)
+MAX_ATTEMPTS_PER_MODEL = 5
+INITIAL_RETRY_DELAY = 2

 SPINE_PLACEHOLDER = "[INSERT BOOK SPINE]"
 # THE BLOG FOLDER REACHES THE MODEL (banked 2026-09-04).
editing_prompt.txt is
(nix) pipulate $ m
📝 Committing: chore: Refactor model selection strategy
[main 8ce6d860] chore: Refactor model selection strategy
 1 file changed, 9 insertions(+), 3 deletions(-)
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/articleizer.py'.
(nix) pipulate $ d
diff --git a/scripts/articles/articleizer.py
b/scripts/articles/articleizer.py
index 1b6980ba..6535fca0 100644
--- a/scripts/articles/articleizer.py
+++ b/scripts/articles/articleizer.py
@@ -454,13 +454,26 @@ def main():
                 print(f"❌ Failed to copy to clipboard: {e}")
             return

-        print(f"Calling the Universal Adapter (using
{DEFAULT_MODEL})...")
-        max_retries = 5
-        retry_delay = 2
-        for attempt in range(max_retries):
+        print(
+            "Calling the Universal Adapter "
+            f"(primary {MODEL_CANDIDATES[0]}, fallback
{MODEL_CANDIDATES[1]})..."
+        )
+        total_attempts = MAX_ATTEMPTS_PER_MODEL *
len(MODEL_CANDIDATES)
```

```
+        retry_delays = {name: INITIAL_RETRY_DELAY for name in
MODEL_CANDIDATES}
+        retry_after = {name: 0.0 for name in MODEL_CANDIDATES}
+        for attempt in range(total_attempts):
+            model_name = MODEL_CANDIDATES[attempt %
len(MODEL_CANDIDATES)]
+            model_attempt = (attempt // len(MODEL_CANDIDATES)) + 1
+            wait = max(0.0, retry_after[model_name] -
time.monotonic())
+            if wait:
+                print(
+                    f"Waiting {wait:.0f} seconds before retrying
{model_name} "
+                    f"(Attempt
{model_attempt}/{MAX_ATTEMPTS_PER_MODEL})..."
+                )
+                time.sleep(wait)
+
             try:
-                # Use a free-tier compatible model.
-                model = llm.get_model(DEFAULT_MODEL)
+                model = llm.get_model(model_name)
                 model.key = api_key  # Assign the key directly to the
adapter
                 response = model.prompt(full_prompt)
                 gemini_output = response.text()
(nix) pipulate $ m
📝 Committing: chore: Refactor Universal Adapter call with retry logic
and model selection
[main 3fdadaa3] chore: Refactor Universal Adapter call with retry
logic and model selection
 1 file changed, 19 insertions(+), 6 deletions(-)
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/articleizer.py'.
(nix) pipulate $ d
diff --git a/scripts/articles/articleizer.py
b/scripts/articles/articleizer.py
index 6535fca0..31594559 100644
--- a/scripts/articles/articleizer.py
+++ b/scripts/articles/articleizer.py
@@ -477,7 +477,7 @@ def main():
                 model.key = api_key  # Assign the key directly to the
adapter
                 response = model.prompt(full_prompt)
                 gemini_output = response.text()
-                print("Successfully received response from API.")
```

```
+                 print(f"Successfully received response from API via
{model_name}.")

                  json_match =
re.search(r'[triple-backtick]json\s*([\s\S]*?)\s*[triple-backtick]',
gemini_output)
                  json_str = json_match.group(1) if json_match else
gemini_output
(nix) pipulate $ m
📝 Committing: chore: Update API response logging in articleizer.py
[main 169f1fa5] chore: Update API response logging in articleizer.py
 1 file changed, 1 insertion(+), 1 deletion(-)
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/articleizer.py'.
(nix) pipulate $ d
diff --git a/scripts/articles/articleizer.py
b/scripts/articles/articleizer.py
index 31594559..6a74c612 100644
--- a/scripts/articles/articleizer.py
+++ b/scripts/articles/articleizer.py
@@ -498,17 +498,28 @@ def main():
                     ("500" in error_str) or \
                     ("high demand" in error_str.lower()):

-                    print(f"Retriable API Error: {e}")
+                    print(f"Retriable API Error from {model_name}:
{e}")
                     # QUOTA WINDOW DISCIPLINE (live-fire convicted
2026-07-19):
-                    # the free tier is a per-MINUTE rolling window;
retrying at
-                    # 2/4/8/16s hammers the same window the server
asked us to
-                    # let drain. Honor the server's own "retry in Ns"
hint.
+                    # each model gets its own exponential clock. A
failure on
+                    # Flash therefore falls through to Lite
immediately, while
+                    # the failed model honors its own server hint
before its
+                    # next turn in the alternation.
                     hint = re.search(r'retry in (\d+(?:\.\d+)?)s',
error_str)
-                    wait = max(retry_delay, float(hint.group(1)) + 1)
if hint else retry_delay
```

```
-                    print(f"Retrying in {wait:.0f} seconds...
(Attempt {attempt + 1}/{max_retries})")
-
-                    time.sleep(wait)
-                    retry_delay *= 2  # Exponential backoff
+                    hinted_delay = float(hint.group(1)) + 1 if hint
else 0
+                    wait = max(retry_delays[model_name],
hinted_delay)
+                    retry_after[model_name] = time.monotonic() + wait
+                    retry_delays[model_name] *= 2
+
+                    if attempt + 1 < total_attempts:
+                        next_model = MODEL_CANDIDATES[(attempt + 1) %
len(MODEL_CANDIDATES)]
+                        next_wait = max(0.0, retry_after[next_model]
- time.monotonic())
+                        if next_wait:
+                            print(
+                                f"Switching to {next_model}; its
backoff has "
+                                f"{next_wait:.0f} seconds remaining."
+                            )
+                        else:
+                            print(f"Switching to {next_model}
immediately.")
                 else:
                     print(f"\nAn unrecoverable error occurred while
calling the API: {e}")
                     if 'gemini_output' in locals():
(nix) pipulate $ m
📝 Committing: chore: Improve API error handling and retry logic
[main 088f1cf7] chore: Improve API error handling and retry logic
 1 file changed, 20 insertions(+), 9 deletions(-)
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/articleizer.py'.
(nix) pipulate $ d
diff --git a/scripts/articles/articleizer.py
b/scripts/articles/articleizer.py
index 6a74c612..4e769c7e 100644
--- a/scripts/articles/articleizer.py
+++ b/scripts/articles/articleizer.py
@@ -525,8 +525,11 @@ def main():
                     if 'gemini_output' in locals():
                         print("--- API Raw Output ---\n" +
gemini_output)
```

```
                    return
-        else: # This block runs if the loop completes without a break
-            print("Error: Max retries exceeded. Failed to get a
successful response from the API.")
+        else:  # This block runs if the loop completes without a
break
+            print(
+                f"Error: {MAX_ATTEMPTS_PER_MODEL} attempts per model
exhausted. "
+                "Failed to get a successful response from the API."
+            )
             return

     if instructions:
(nix) pipulate $ m
📝 Committing: fix: Handle API retry exhaustion errors
[main a634740b] fix: Handle API retry exhaustion errors
 1 file changed, 5 insertions(+), 2 deletions(-)
(nix) pipulate $ git push
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 48 threads
Compressing objects: 100% (25/25), done.
Writing objects: 100% (25/25), 3.47 KiB | 3.47 MiB/s, done.
Total 25 (delta 20), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (20/20), completed with 4 local
objects.
To github.com:pipulate/pipulate.git
   03285dac..a634740b  main -> main
(nix) pipulate $
```

4: Prompt: Confirm that articleizer.py now:

1. prefers gemini-flash-latest first,
2. falls through immediately to gemini-flash-lite-latest on a retriable first failure,
3. alternates strictly between the two models,
4. maintains independent exponential-backoff clocks for the two models,
5. honors each model's existing "retry in Ns" hint without transferring that penalty to the other model,
6. remains syntactically valid.

Pay special attention to the BEFORE/AFTER rg receipt and the AST receipt. If the source proves those invariants, say so and identify the cheapest useful live-fire witness for the next normal publishing run. Do not broaden the change unless the receipts expose a concrete problem.

5: Deliverables: Well this will be one that I'll have to actually publish this article to see if it worked, so let's wrap it and I'll retroactively come back and say if it did after it's prepared for publication. That's going to be thought to make receipts and rubber-stamp and notarize or whatever but have fun with the chicken-and-eggness of it all.

Dismounting at the Notary Beat

Hop off the ride. This ride's stated goal is reached — dismount. This is the NOTARY BEAT: the ride ends here, is witnessed here, and is sealed here. Answer all seven beats, briefly:

0. **TL;DR**: a short, dry, neutral abstract for the TOP of the published article — written for an unfamiliar reader or AI summarizer who has never seen this system. No hype, no insider handles unexplained.
1. VERIFY: restate the goal from the top of this article and confirm (or deny) it was met, citing THIS compile's receipts, not memory. Name any ignition this ride required that never fired -- an AFTER tap taken without one is a stale BEFORE wearing the AFTER's label.
2. BANK: name everything that graduates — rule, earmark, todo, pin — as SEARCH/REPLACE patch cars against raw source present in THIS context (Target line inside the fence, one car per commit story), deletions included. A 'paste-ready' line with no Target and no SEARCH anchor is not banked; it is a hand edit the operator will not make. These BANK cars are the ONLY patches a dismount emits.
3. DANGLING: what carries forward unbanked? One line each, no essays.
4. SEED: the adhoc.txt lines (and TODO_SLUGS if narrative context is needed) for the next ride's first compile.
5. CLOSING: a closing summary for the BOTTOM of the article — the final take-away, tied to the book's larger arc where it fits naturally, never forced. Storytelling over inventory.
6. NOTARIZE: the artifact of record is the cartridge THIS compile writes after this payload is assembled, so its name is not in your context and you must not guess one. Attest the FINGERPRINT instead: this Manifest's wc -c line, its receipt count, and the newest receipt's date -- pure functions of the bytes the cartridge seals. Close with one line the operator completes after the compile: Deed: foo-__.zip (ls -t foo-*.zip | head -1) The seal is the signature; the fingerprint is how the deed is found.

FINALITY: after beat 6, this discussion is CLOSED. Emit NO five-car train, NO probes, NO patches beyond the BANK cars of beat 2, and NO next-turn prompt beyond the SEED lines in beat 4. Any reader or model encountering this article later should treat it as a finished, notarized document — an archive entry, not an open thread.

ChatGPT 6 Extra High: Context verified. The local common and lsa imports are not included, but they are not critical to this localized control-flow review; the relevant retry code, live receipts, and syntax/static-analysis evidence are present.

1. VERIFY

Goal: prefer gemini-flash-latest; on a retriable failure immediately try gemini-flash-lite-latest; thereafter alternate while applying progressive backoff independently to each model.

Confirmed at the source/compile level.

1. **Flash is first.** MODEL_CANDIDATES is ordered Flash, then Flash Lite.
2. **First retriable Flash failure falls through immediately to Lite.** The error branch records Flash's future retry_after but performs no sleep; the next loop iteration selects the next candidate. Because Lite begins with retry_after == 0.0, it runs immediately.
3. **Alternation is strict.** MODEL_CANDIDATES[attempt % len(MODEL_CANDIDATES)] yields Flash, Lite, Flash, Lite for all ten permitted attempts.
4. **Backoff clocks are independent.** Both retry_delays and retry_after are dictionaries

keyed by model name; a failure mutates only ...[model_name].

5. **Server hints stay attached to the failing model.** The parsed hint participates only in the wait used to set retry_after[model_name]; it is not copied onto the alternate model.
6. **Syntax is valid.** This compile's AST receipt is literally articleizer.py AST OK; Ruff also reports all checks passed.

The Manifest identifies exactly three live command receipts for this compile: LSA, the rg inspection, and the AST parse.
Ignition: none was required for the AFTER source/AST readings; both commands read the patched file directly. The **runtime failover branch itself has not yet fired in evidence here**.
The cheapest useful live-fire witness is therefore exactly what you proposed: the next ordinary article / grim / bot publication. Do not manufacture an outage. If Flash happens to be unavailable naturally, the decisive transcript is:
Retriable API Error from gemini-flash-latest → Switching to gemini-flash-lite-latest immediately.
→ either Lite success or the later alternating/backoff sequence.
If Flash simply succeeds, that witnesses primary preference and the success path but does not yet witness failover.

2. BANK

No BANK cars.
The retry implementation itself is already committed and pushed in the history captured by this article. Nothing newly discovered in this verification justifies broadening the ride with another repo mutation.

3. DANGLING

- Natural live-fire failover remains unwitnessed until an ordinary publication actually encounters a retriable model failure.
- The older **NOTARY FINGERPRINT** TODO remains unresolved: the compiler's Manifest does not emit the requested wc -c measurement or dated-receipt field, and its own standing note explicitly says not to substitute Summary size or submission time.

4. SEED

For the next ride's first compile:

```
! rg -n "MODEL_CANDIDATES|retry_delays|retry_after|Switching
to|Successfully received response from API via"
scripts/articles/articleizer.py
scripts/articles/articleizer.py
```

The interesting new evidence should be the actual normal-publication console transcript you bring back; there is no value in forcing a quota/error condition just to make the branch perform.

5. CLOSING

The useful distinction turned out to be between failover and backoff. Failover should be eager: when one available route fails, try the other route while it still has a clean slate. Backoff should be local: a service that asks to be left alone earns its own cooling-off period, not a penalty

imposed on every alternative.
That is a small control-flow change, but it fits the larger pattern of replayable engineering. The model call remains probabilistic and externally governed; the machinery around it does not have to be. Candidate order, retry ownership, waiting, and handoff are now explicit enough to inspect from source and simple enough that the next real publication can provide the final runtime witness.

6. NOTARIZE

The requested fingerprint cannot be fully attested from this Manifest without inventing data. **Manifest wc -c line: not present**. **Live receipt count: 3**. **Newest receipt's date: not present**; the Manifest names the newest receipt command but gives it no receipt timestamp. The existing NOTARY FINGERPRINT TODO explicitly warns against substituting another measurement or submission time.
So the truthful fingerprint is: **wc-c unrecorded / 3 receipts / newest-receipt date unrecorded**.
Deed: foo-_____.zip (ls -t foo-*.zip | head -1)
MikeLev.in: Well there we go! It paid off in its first try:

```
(nix) pipulate $ article
🧯 Neutralized 2 floating backtick run(s) -> [triple-backtick]
ℹ️  Public lane: no private fences, inline spans, or PII matches
found.
✅ Article sanitized!
🎯 Target set via CLI: MikeLev.in (Public)
📚 Book spine injected: 1488 articles, 71,169 chars.
Calling the Universal Adapter (primary gemini-flash-latest, fallback
gemini-flash-lite-latest)...
Successfully received response from API via gemini-flash-latest.
Successfully parsed JSON instructions.
✅ Instructions saved to 'instructions.json' for future use.
Formatting final Jekyll post...
📅 Found 3 posts for today. Auto-incrementing sort_order to 4.
Skipping subheading '## Eager Failover Versus Localized Backoff':
heading already present at insertion point.
Skipping subheading '## Structuring the Alternating Failover
Mechanism': heading already present at insertion point.
✨ Success! Article saved to:
/home/mike/repos/trimnoir/_posts/2026-09-19-dual-model-failover-indepe
ndent-backoff.md
Collect new 404s: python prompt_foo.py assets/prompts/find404s.md
--chop CHOP_404_AFFAIR -l [:] --no-tree
🔗 Paste-ready preview URL copied to clipboard:

http://localhost:4001/futureproof/dual-model-failover-independent-back
off/
(nix) pipulate $
```

Book Analysis

Ai Editorial Take

What makes this entry stand out is the crisp distinction between eager failover and local backoff. Most API retry frameworks bundle retry counters and wait times into a single global state, inadvertently treating separate endpoints as a single failing service. By granting each model its own independent cooldown clock, the pipeline transforms what could have been an agonizing minutes-long stall into an immediate, transparent handoff.

Content Potential And Polish

- **Core Strengths:**
- Clear, high-value algorithmic improvement: separating backoff clocks prevents a throttled model from penalizing an available alternative.
- Disciplined use of git commits and AST validation probes as verifiable receipts between iterations.
- Truthful notarization beat that explicitly acknowledges when an error path is validated statically but remains unwitnessed in live execution.
- **Suggestions For Polish:**
- Clarify how the script handles scenarios where both candidate models are simultaneously throttled.
- Consider extracting the retry loop into a standalone reusable generator or class if additional models are introduced later.

Next Step Prompts

- Write an automated mock test using pytest that simulates a 429 quota error on Gemini Flash to produce the live-fire console transcript witnessing immediate failover to Flash Lite.
- Evaluate how to extend this two-model candidate tuple into an arbitrary fallback priority list without complicating the monotonic time tracking logic.