AI

- design languange
- Models
- Google email setup
- Google Woekspace app setup
- setup
- automations
- common issues found

design languange

● Design language summary

Vibe. Dark, minimal, slightly luminous. Information-dense but airy. Color is reserved — accent appears on actions and active states; semantic colors only for status. Most surfaces are low-contrast neutral; the eye is drawn by the one
accent.

Palette. Near-black background #0b0d12, two elevation tiers above it (#14171d panel, #1a1e26 panel-2). Borders are translucent white (7% default, 12% strong). Text is warm-white #e6eef6, muted #8a93a4, super-muted #5e6675. Accent is a
blue-violet #4f8cff with #6ea0ff for hover and a 15% soft variant for active backgrounds. A violet #8b5cf6 shows up only inside accent gradients (logos). Status: green #34d399, amber #fbbf24, red #f87171.

Type. Inter, system fallback. Base 14.5px / 1.5. Headings get tight tracking (-0.01em to -0.02em). Small labels are uppercase, ~0.78rem, 0.06–0.08em letter-spacing, in the muted tone. Numbers always use tabular figures. Antialiased.

Spacing & shape. Spacing scale of 4/8/12/16/24/32. Radii: 12px panels, 8px controls/inputs, 6px chips, 999px for bars. Two shadow tiers — subtle 0 2px 8px /35% for cards, lifted 0 8px 24px /45% for floating things. Active accents may
add a tinted glow (rgba(79,140,255,0.35)).

Layout. Sticky 240px sidebar + fluid main. On mobile the sidebar collapses to a horizontal top nav. Each page opens with a page-header (title + small meta on the right, divider underneath). Content lives inside card-panel blocks — never
floating directly on the bg.

Components.
- Stat card: tiny uppercase label with icon → big tabular number → muted sub-line. Border lightens on hover.
- Sidebar nav-item: icon + label, transparent base, accent-soft background when active. Destructive items shift to red on hover.
- Segmented control: pill group inside a panel-2 shell; active pill is solid accent with glow.
- Table: transparent rows, uppercase muted column headers, tabular nums right-aligned, 2% white row-hover.
- Bar (disk): 6px gradient fill, threshold-based color shift at 75% / 90%.
- Chip/folder pill: panel-2 with border, small inline icon, 6px radius.

Iconography. Bootstrap Icons (bi-*), inline with labels at slightly reduced opacity.

Motion. 0.15s ease for hover/active transitions; nothing splashy. Async sections fade to 50% opacity (is-loading); a single accent spinner for spinners.

Charts. Plotly with transparent bg, Inter font, 5% white gridlines, legend below the plot, custom dark hover label. Series colors come from a fixed 10-stop palette (blue → violet → green → amber → coral …) — never raw Plotly defaults.

Don'ts. No hard borders. No heavy drop shadows. No mixing accent colors for non-action UI. No raw Bootstrap-default buttons or tables. No solid color blocks for headers — keep them transparent over the panel.

Models

removes people and other dynamic subjects from a source video

https://huggingface.co/Lightricks/LTX-2.3-22b-IC-LoRA-Clean-Plate

video generation on a *reference sheet* — a single composite image inventorying the characters, props, and location of a scene

https://huggingface.co/Lightricks/LTX-2.3-22b-IC-LoRA-Ingredients

voice

piper - lightweight and good quality for a vps

Google email setup

Basic email access (app password)

1 - Sign into the google account

2 - Go here: https://myaccount.google.com/apppasswords

3 - Create the below in the .env

```
HIMALAYA_GMAIL_PASSWORD="app password here"
HIMALAYA_CONFIG=~/.hermes/profiles/CHANGE-PROFILE-NAME-HERE/himalaya.toml
```

4 - create himalaya.toml

```
[accounts.account-nickname]
email = "account@gmail.com"
display-name = "account-nickname"
default = true

backend.type = "imap"
backend.host = "imap.gmail.com"
backend.port = 993
backend.encryption.type = "tls"
backend.login = "account@gmail.com"
backend.auth.type = "password"
backend.auth.cmd = "echo $HIMALAYA_GMAIL_PASSWORD"

message.send.backend.type = "smtp"
message.send.backend.host = "smtp.gmail.com"
message.send.backend.port = 587
message.send.backend.encryption.type = "start-tls"
message.send.backend.login = "account@gmail.com"
message.send.backend.auth.type = "password"
message.send.backend.auth.cmd = "echo $HIMALAYA_GMAIL_PASSWORD"

folder.aliases.inbox = "INBOX"
```

```
folder.aliases.sent = "[Gmail]/Sent Mail"
folder.aliases.drafts = "[Gmail]/Drafts"
folder.aliases.trash = "[Gmail]/Trash"
```

Google workspace setup

1 - Create app

2 - Enable APIs

3 - Install app secret

```
HERMES_HOME=~/.hermes/profiles/PROFILE ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/PROFILE/skills/productivity/google-workspace/scripts/setup.py --client-secret
~/.hermes/shared/google-app-secret.json
```

- Get the auth code

```
# Get new auth URL
HERMES_HOME=~/.hermes/profiles/gabriel ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/gabriel/skills/productivity/google-workspace/scripts/setup.py --auth-url

# Exchange the new code
HERMES_HOME=~/.hermes/profiles/gabriel ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/gabriel/skills/productivity/google-workspace/scripts/setup.py --auth-code
"PASTE_REDIRECT_URL_HERE"

# Verify
HERMES_HOME=~/.hermes/profiles/gabriel ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/gabriel/skills/productivity/google-workspace/scripts/setup.py --check
```

revoke a token

```
HERMES_HOME=~/.hermes/profiles/helios-ai ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/helios-ai/skills/productivity/google-workspace/scripts/setup.py --revoke
```

Google Woekspace app setup

Setting Up Gmail Per Hermes Profile

Each Hermes profile authenticates to its own Google account via OAuth. This guide covers the one-time Google Cloud setup, then the per-profile authorization.

Prerequisites

- A Hermes profile for each Gmail account you want to connect
- Access to the Google Cloud Console

Step 1: Create Google OAuth Credentials (one-time)

Do this once in the Google Cloud Console. The same OAuth client can be used across all profiles — each profile just signs in with a different Google account during the auth flow.

1. Go to Google Cloud Console → Project Selector
2. Create or select a project
3. Enable the APIs you need from the API Library:
   - Gmail API
   - Google Calendar API
   - Google Drive API
   - Google Sheets API

- Google Docs API
- Google Slides API
- People API (contacts)

4. Go to Credentials
   - Create Credentials → OAuth 2.0 Client ID
   - Application type: **Desktop app**
   - Click Create
5. If the app is in Testing mode, add each Google account you want to connect as a test user:
   - Go to Audience
   - Test users → Add users → add each Gmail address
6. Download the JSON file

> The client secret is just your app's identity — it doesn't grant access to any account by itself. Access comes from the per-profile auth flow where you sign in with a specific Google account.

Step 2: Place the client secret file

Put the downloaded JSON file in a shared location:

```
cp ~/Downloads/client_secret_*.json ~/.hermes/shared/google-app-secret.json
```

Step 3: Authorize each profile

Run these commands for **each profile** you want to connect. Replace `<profile>` with the profile name (e.g. `james`, `gabriel`, `helios-ai`).

Important: Use the Hermes venv Python (`~/.hermes/hermes-agent/venv/bin/python`), not system `python3` — the Google libraries require Python 3.10+ and the system Python is 3.9.

3a. Install the client secret into the profile

```
HERMES_HOME=~/.hermes/profiles/<profile> ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/setup.py --client-secret
~/.hermes/shared/google-app-secret.json
```

3b. Get the authorization URL

```
HERMES_HOME=~/.hermes/profiles/<profile> ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/setup.py --auth-url
```

This prints an OAuth URL. Open it in your browser and sign in with the Gmail account you want this profile to use.

> **Note:** After approving, the browser will likely show an error on `http://localhost:1` — this is expected. Copy the ENTIRE URL from the browser address bar.

> **Note:** The `--services` and `--format json` flags are NOT supported in the current version. The setup script uses a fixed set of scopes covering all Google Workspace APIs.

3c. Exchange the authorization code

Paste the redirect URL (or just the code) back:

```
HERMES_HOME=~/.hermes/profiles/<profile> ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/setup.py --auth-code
"PASTE_REDIRECT_URL_HERE"
```

3d. Verify it worked

```
HERMES_HOME=~/.hermes/profiles/<profile> ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/setup.py --check
```

Should print `AUTHENTICATED`.

Step 4: Test it

```
GAPI=~/.hermes/hermes-agent/venv/bin/python

# Search unread emails
HERMES_HOME=~/.hermes/profiles/<profile> $GAPI ~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/google_api.py gmail search "is:unread" --max 5

# List calendar events
HERMES_HOME=~/.hermes/profiles/<profile> $GAPI ~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/google_api.py calendar list
```

Step 5: Repeat for each profile

Run Steps 3a–3d again for each profile, signing in with a different Gmail account each time. Each profile stores its own token at:

```
~/.hermes/profiles/<profile>/google_token.json
```

File Locations Per Profile

File	Path	Purpose
Client secret	`~/.hermes/profiles/<profile>/google_client_secret.json`	OAuth client credentials (copied from shared file)
OAuth token	`~/.hermes/profiles/<profile>/google_token.json`	Authenticated session (unique per profile)

How It Works

Piece	Role
Client secret	Your app's identity — "I am Hermes app". Shared across all profiles.
Auth flow (browser sign-in)	Pick which Google account to connect. Done once per profile.
Token (`google_token.json`)	Proof that a specific Gmail account authorized the app. One per profile.

One client secret, many tokens — each token tied to whichever Google account signed in during that profile's auth flow.

Troubleshooting

Problem	Fix
`NOT_AUTHENTICATED`	Run Steps 3b–3d again
`REFRESH_FAILED`	Token revoked or expired — redo Steps 3b–3d
`HttpError 403: access_denied`	Add the Gmail address as a test user in Google Cloud Console → Audience
`HttpError 403: Insufficient Permission`	Missing API scope — revoke and redo
`ModuleNotFoundError` / `No matching distribution`	Use the Hermes venv Python: `~/.hermes/hermes-agent/venv/bin/python`
`zsh: no such file or directory: python`	Don't use the `GSETUP` variable pattern in zsh. Call the venv Python directly with `HERMES_HOME=...` prefix

Revoking Access

To disconnect a profile's Gmail and re-authorize with a different account:

```
# Revoke
HERMES_HOME=~/.hermes/profiles/<profile> ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/setup.py --revoke

# Re-authorize (Steps 3b–3d)
HERMES_HOME=~/.hermes/profiles/<profile> ~/.hermes/hermes-agent/venv/bin/python
~/.hermes/profiles/<profile>/skills/productivity/google-workspace/scripts/setup.py --auth-url
```

Quick Reference: All Commands For One Profile

```
PROFILE=james
HH="HERMES_HOME=~/.hermes/profiles/$PROFILE"
PY=~/.hermes/hermes-agent/venv/bin/python
SCRIPT=~/.hermes/profiles/$PROFILE/skills/productivity/google-workspace/scripts/setup.py

# 1. Install client secret
$HH $PY $SCRIPT --client-secret ~/.hermes/shared/google-app-secret.json

# 2. Get auth URL (sign in with the Gmail account for this profile)
$HH $PY $SCRIPT --auth-url

# 3. Exchange code (paste the redirect URL)
$HH $PY $SCRIPT --auth-code "http://localhost:1/?code=4/0A..."

# 4. Verify
$HH $PY $SCRIPT --check

# 5. Test Gmail
GAPI=~/.hermes/profiles/$PROFILE/skills/productivity/google-workspace/scripts/google_api.py
$HH $PY $GAPI gmail search "is:unread" --max 5
```

setup

choose llm provider (sliding scale with smartness to price)

choose chat channels

setup optimal config

delete unnecessary skills (gaming, inference-sh, media(spotify, songsee, heartmula, gif-search) redteaming, smarthome, social media, yuanbao.)

setup and connect camoufox

clone this into skills: https://github.com/hardikpandya/stop-slop.git

automations

weekly cost and time saver report

Look back over the last seven days of all sessions chat logs and give me a cost estimate of time saved based off. If Jamie was to do these tasks himself manually at $1000 an hour how much time and money he would've saved.

common issues found