PromptER (Prompt Studio) — User Manual & Quickstart

A practical guide to every control in the app and a minimal quickstart tailored for Gen-AI-native users.

---

1) What this app does

PromptER helps you draft, expand, and evaluate prompts using a **Skeleton-of-Thought → Tree-of-Thought → Optimized Prompt** workflow. It also includes a web scraper to discover prompts and a quality checker to analyze and validate prompts.

---

2) Workspace layout

A. Sidebar

📂Project Explorer

A collapsible tree of your working directory so you can browse files/folders quickly.

Project Folders (expander)

- **New folder name**: Type a folder name to create (safe-sanitized).
- **Create Folder**: Creates the folder under the current working directory.
- **Target folder for saves**: Choose where app outputs will be written.
- 💾**Save Skeleton**: Saves the current skeleton (JSON).
- 💾**Save ToT**: Saves the latest Tree-of-Thought (TXT).
- 💾**Save Optimized**: Saves the final optimized prompt (TXT).

Why care? You must select a *Target folder* before any save buttons can write files.

🔦Git Sync (expander)

End-to-end Git workflow for the chosen directory. - **Target directory**: Pick working path (current folder or your selected save folder). - **Remote repository** - **GitHub repo URL**: Remote URL (e.g., `https://github.com/user/repo.git`). - **Branch**: Branch to push/pull (default `main`). - 📥**Clone**: Clone the repo into the target directory (creates a subfolder if non-empty). - 🔧**Init Repo**: Initialize an empty Git repo in the target directory. - **Remotes**: View remotes; optionally **Set origin to repo URL above**. - **User identity**: Set **user.name** and **user.email** for commits (local override). - **Commit** - **Add All**: Stage all changes. - 🔗**Commit**: Commit with message. - 🧹**Clean Untracked**: Remove untracked files (destructive). - **Sync** - 📡**Push**: Push to remote branch. - 📥**Pull**: Pull (rebase) from remote branch.

B. Main area (Tabs)

🧰Prompt Studio

Panel 1: User Entries - **Goal**: One-line objective for the prompt. - **Context**: Key background/constraints. - **Role**: The assistant/agent role or persona. - **Output Format**: Desired structure/format of the response. - **Next → Skeleton**: Proceeds to the skeleton panel.

Panel 2: Skeleton & ToT - **Controls** - 🗜️**Load Model**: Starts the LLM backend used in the app. - **Number of steps** *(slider)*: How many steps in the skeleton. - 📜 **Generate Skeleton**: Builds a numbered Skeleton-of-Thought for your Goal/Context/Role/Format. - **Choose step** *(dropdown)*: Select a skeleton step to work on. - **Text Curvature** (expander) - **n-token (curvature length)** *(slider)*: Window length used for the similarity curve. - **Visible steps** *(multi-select)*: Choose which step curves to show. - **Chart**: Shows token-wise semantic curvature per step. - **Tree of Thought** (expander) - 🗜️ **Generate Tree of Thought**: Produces multiple ways to elaborate the selected step. - **Tree of Thought** *(textarea)*: Read-only output capture.

Panel 3: Prompt Build - **Select a step to add**: Pick which skeleton steps to include in the final prompt. - **Tree of Thought for {step}** *(textarea)*: Paste/adjust ToT expansions per selected step. - 🧮 **Generate Optimized Prompt**: Produces the combined, optimized prompt. - **Optimized Prompt** *(textarea)*: Final prompt text (editable). - **PromptLID vs Steps** *(bar chart)*: Similarity of final prompt vs each skeleton step (with **PromptLID (steps avg)** badge). - **Curvature: Goal vs Optimized (overlay)** (expander) - **Visible steps** *(multi-select)* and overlay chart comparing semantic curvature of Goal vs Optimized per step.

📃Web Scraper

Discover prompts from supported sites, rank them, and optionally add them to the Prompt Studio corpus. - **Site preset**: Quick set of known sources. - **Search query**: Text to rank scraped prompts against. - **Top K** *(slider)*: Number of top results to display. - **Base URL**: Start URL for crawling (auto-filled from preset). - **Similarity method**: `auto`, `tfidf`, or `fuzzy`. - **Max pages** *(slider)*: Crawl budget. - **HTTP timeout (s)**, **Rate limit (s)**: Networking controls. - **Disable concurrency** *(checkbox)*: Force single-threaded scraping. - **Add scraped prompts to Prompt Studio corpus** *(checkbox)*: Append best prompts to your corpus (deduped, capped). - 🔎**Scrape & Search**: Runs discovery, scraping, ranking. - **Results table**: Shows title/score/author/tags/url/description; 💾**Download JSON** exports. - **Per-result expander**: View description, raw text, and source URL.

> **Notes** • Respects `robots.txt` and uses sitemap-first discovery. • Adding to corpus is available when a Goal is set. Corpus is deduped and capped to avoid bloat.

📜Prompt Quality Check

Analyze saved prompts and run ad-hoc optimization/validation.

Open Previous Runs & loaders - **Folder**: Choose a directory (recursive picker). - **Optimized prompt files**: Pick a previously saved `optimized_prompt_*.txt`. - **Optimized Prompt (loaded)** / **Tree of Thought (loaded)**: Display fields.

Ad-hoc Prompt Optimization & Relevance Scoring - **Enter a prompt to optimize** *(textarea)*: Input any prompt to analyze. - **Token Relevance Spider Chart** - **Normalize radar values** *(checkbox)*: Per-prompt min–max normalization. - **Use cosine distance** *(checkbox)*: Switch distance metric. - **Chart**: Per-token relevance profile. - **Relevance scoring iterations** *(slider)*: Iterations to run. - **Use llama3 to generate outputs each iteration** *(checkbox)*: Toggle model-based sampling. - **Sampling temperature** *(slider)*: Sampling randomness. - **Run Optimization**: Plots score evolution per token across iterations.

Category Validation - **Prompt category**: `Query-response`, `Instruction`, `Generative Prompt`, or `Co-thinking / Collaborative Prompting`. - **Validate prompt against category — {category}**: Generates an audit JSON with dimension scores and a short summary. - **Validation report (read-only)**: Output view. - 💾 **Download report** and on-disk save. - 🧹 **Clear Embedding Cache**: Clear cached embeddings if memory grows.

---

3) Under the hood (for power users)

- Uses **Ollama** (default model id `llama3`) to generate skeletons, ToT, and validations.
- Uses **Hugging Face** `meta-llama/Llama-3.1-8B` for token-level entropy stats.
- Embedding model: **SentenceTransformer** `all-MiniLM-L6-v2` to compute PromptLID/curvature and relevance metrics.

---

4) Getting Started (for Gen-AI-native users)

A minimal, fast path through the app.

1) **Define it** → In *Prompt Studio / Panel 1*, fill **Goal**, **Context**, **Role**, **Output Format**, then **Next → Skeleton**. 2) **Spin up LLM** → In *Panel 2 / Controls*, click 🗜️ **Load Model**, set **Number of steps**, 📜 **Generate Skeleton**, then **Choose step**. 3) **Explore** → In *Tree of Thought*, click 🗜️ **Generate Tree of Thought** to fan out options; skim the textarea. 4) **Assemble** → In *Panel 3*, **Select a step to add**, paste/trim its ToT, then 🧮 **Generate Optimized Prompt**. 5) **Check** → Review **Optimized Prompt**, glance at **PromptLID vs Steps** and the **Curvature overlay**; iterate if needed. 6) **Save/Version** → In *Project Folders*, use 💾 **Save Optimized** (and optionally **Save Skeleton/ToT**). In *Git Sync*, **Add/Commit/Push** to your repo. 7) **Enrich** *(optional)* → Use *Web Scraper* to pull Top-K prompts matching your query; add to corpus and re-optimize. 8) **Audit** *(optional)* → In *Prompt Quality Check*, try the **Spider Chart**, run **Run Optimization**, and **Validate** against a category.

---

5) Troubleshooting & tips

- **Nothing saves?** Pick a *Target folder for saves* in **Project Folders** first.
- **Clone/Push fails?** Ensure Git is installed and credentials are configured. Use **Init Repo** if the folder isn't a repo yet.
- **Scraper finds nothing?** Increase **Max pages**, check **Base URL**, or try another **Site preset**. Some sites block crawling; honor errors about `robots.txt`.

- **Charts empty?** Generate a skeleton and select steps; add an **Optimized Prompt** before comparing curvature overlays.
- **Validation errors?** Ensure the LLM has been loaded (**Load Model**) if you enabled model-based features.

End of manual.