Audion Docs AI

Русский · User Guide

Contents

Auditing documents with a language model and applying what it finds: Word and PowerPoint in, an error report and a corrected copy out.

Why It Exists

Proofreading a long document is work a model does well. *Editing* a document is work a model does dangerously: it will rewrite the whole paragraph, change things nobody asked about, and hand back a file where you cannot tell what moved.

So the two jobs are separated here.

```
parse → audit by the model → report → correction
```

The model finds. The code fixes.

The audit reads the document and records what it found: where the error is, what the text was, what it should become, and how confident the model is. The correction step **never calls the model again** — it reads the existing record and applies only exact single replacements marked with high confidence.

Everything else — the uncertain, the arguable, anything needing a rewrite — is never applied. It stays in the report as a separate table of unresolved items.

Principles

Correction does not improvise. A replacement is applied only when both the exact original text and the exact new text are known. No "rewrite this paragraph better".

The cache is signed by the document. A model's answer is bound to the file's hash and the run parameters. Edit the file but keep the name, and the old answer will not be reused.

The model is never picked at random. With no model specified, the last successfully probed one is used, then the first favourite — but never an arbitrary string from the list the provider returned. A separate button runs a small probe of the chosen model and remembers the result with its date.

PDF is read-only. Text is taken from it for reports, but it is never edited: a PDF is an output, not a source.

The depth of the model's work is chosen explicitly. Each provider names it differently, and the program passes that provider's own parameter rather than a rough common one.

Providers

Four to choose from, switched in the window: OpenAI, Gemini, xAI, Claude. The key and the model sit together, the audit rules below, and there is a single action.

The application's standard models are `gpt-5.6-luna`, `gemini-3.6-flash`, `grok-4.3`, and `claude-sonnet-5`; whatever is chosen in the window or set by hand takes precedence.

What Comes Out

file	what is inside
XLSX report	findings row by row: where, what, why
DOCX report	the same as a readable document, with the unresolved table

file	what is inside
corrected copy	the document with the exact replacements applied
annotated copy	the original with anchors at places needing attention

The source document is never modified — edits go into a copy.

Next

- User Guide — step by step.
- `tools\GUI_WORKFLOW_RU.md` — the window and the order of work.
- `tools\DOCUMENT_NORMALIZATION_RU.md` — how correction works.
- `tools\COMMA_LOWERCASE_RESTORE_RU.md` — restoring commas and lower case.
- `tools\TASK_PDF_EXTRACTION_RU.md` — extraction from PDF into reports.

All four are Russian.

Technical Reference

Running

```
launcher_gui.cmd          windowed
launcher_project.cmd      command line, reports in English
launcher_project_ru.cmd   the same in Russian
builder_main.cmd          environment build, checks, release
```

Settings

`config\llm_settings.yaml` — chunk sizes, retries, timeouts, names of the rule files. There are no separate launchers per provider pair any more: everything is chosen in the window.

Advanced Parameters

Hidden in a collapsible block — chunk size and overlap, minimum chunk count, retries and timeouts, manual model override, resuming from where a run broke, render-map strictness. The main path is not cluttered with them.

Python runs unbuffered, so the configuration and the progress of each chunk appear in the log immediately rather than after completion.

Workbench Naming

One shared vocabulary across all Audion projects: **Source**, **Add file…**, **Target**, **Reset**, **Delete**, **List**. In Russian: **Источник**, **Добавить файл…**, **Назначение**, **Сбросить**, **Удалить**, **Список**.