mdexplore

Fast Markdown explorer for Ubuntu/Linux desktop: browse `.md` files in a directory tree and preview fully rendered output instantly.

Features

- Expandable left-pane directory tree rooted at a chosen folder.
- Shows Markdown files only (`*.md`), while still allowing folder navigation.
- Large right-pane rendered preview with scroll support.
- PlantUML diagram renders run in background workers so markdown preview stays responsive.
- Supports:
  - CommonMark + tables + strikethrough.
  - TeX/LaTeX math via MathJax.
  - Mermaid diagrams with improved dark-theme contrast.
  - Optional Rust Mermaid backend via `mmdr` (`--mermaid-backend rust`).
  - PlantUML diagrams (asynchronous local render with placeholders).
  - Markdown callouts (`> [!NOTE]` , `> [!TIP]` , `> [!IMPORTANT]` , `> [!WARNING]` , `> [!CAUTION]`).
- Top actions:
  - `^` moves root up one directory level.
  - `Refresh` rescans the current directory view to pick up new/deleted files.
  - `PDF` exports the current preview to `<filename>.pdf` with centered page numbering (`N of M`).
  - `Add View` creates another tabbed view of the same document at the current top visible line.
  - `Edit` opens the selected file in VS Code (`code` CLI).
- Window title shows the current effective root path.
- Preview cache keyed by file timestamp and size for fast re-open.
- Mermaid SVGs are cached in-memory per app run to avoid re-rendering when returning to previously viewed files.
- Mermaid keeps separate in-memory cache modes for GUI (`auto`) and PDF (`pdf`) rendering.
- Navigating back to a cached file still performs a fresh stat check; changed files re-render automatically.
- `F5` refresh shortcut for directory view rescan (same behavior as `Refresh` button).
- If the currently previewed markdown file changes on disk, preview auto-refreshes and shows a status bar message.
- Status bar reports active long-running work (preview load/render, PlantUML progress, PDF export) and returns to `Ready` instead of staying blank.
- Manual tree/preview pane resizing is preserved across `^` root navigation for the current app run.
- Right-click a Markdown file to assign a highlight color in the tree.
- Highlight colors persist per directory in `.mdexplore-colors.json` files.
- View-tab state persists per directory in `.mdexplore-views.json` files for documents that have more than one saved view or a custom tab label.
- Right-click menu includes `Clear All` to recursively remove all highlights from scope.
- Top-right color buttons copy matching highlighted files to clipboard.
- A pin button before the copy-color buttons copies the currently previewed markdown file.

- Search box includes an explicit `X` clear control that clears the query and removes bolded match markers.
- When search is active and a matched file is opened, preview matches are highlighted in yellow and the view scrolls to the first match.
- While dragging the preview scrollbar, mdexplore shows an approximate `current line / total lines` indicator beside the scrollbar handle.
- When search is active, preview scrollbar markers show the vertical positions of highlighted hits; clicking a marker jumps to the nearest hit in that cluster.
- Preview scroll position is remembered per markdown file for the current app session.
- Right-click selected text in the preview pane to use:
  - `Copy Rendered Text` for plain rendered selection text.
  - `Copy Source Markdown` for markdown source content. Copies matching source markdown using direct range mapping first, then selected-text/fuzzy line matching as fallback, and finally the full source file if no match is possible.
- Clipboard copy uses file URI MIME formats compatible with Nemo/Nautilus paste.
- Last effective root is persisted to `~/.mdexplore.cfg` on exit.
  - If no directory is selected at quit time, the most recently selected/expanded directory is used.

Requirements

- Ubuntu/Linux desktop with GUI.
- Python `3.10+`.
- `python3-venv` package available.
- Internet access for Mermaid only when no local Mermaid bundle is available.
- Internet access for MathJax only when no local MathJax bundle is available.
- Java runtime (`java` in `PATH`) for local PlantUML rendering.
- `plantuml.jar` available (vendored path by default, or set `PLANTUML_JAR`).
- Optional: VS Code `code` command in `PATH` for `Edit`.
- Optional: `mmdr` in `PATH` (or `MDEXPLORE_MERMAID_RS_BIN`) for Rust Mermaid backend.

Quick Start

From any directory:

```
/path/to/mdexplore/mdexplore.sh
```

When no `PATH` is supplied, the app opens:

1. the path stored in `~/.mdexplore.cfg` (if valid), otherwise
2. your home directory.

To open a specific root directory:

```
/path/to/mdexplore/mdexplore.sh /path/to/notes
```

What the launcher does:

- Creates `.venv` inside the project if missing.
- Uses `.venv/bin/python` directly (does not alter your current shell session).
- Installs dependencies when `requirements.txt` changes.
- Runs the app.

Usage

Wrapper script

```
mdexplore.sh [--mermaid-backend js|rust] [PATH]
```

- `PATH` is optional.
- `--mermaid-backend` is optional (`js` default, `rust` requires `mmdr`).
- Supports plain paths and `file://` URIs (for `.desktop` `%u` launches).
- If a file path is passed, mdexplore opens its parent directory.
- If omitted, `~/.mdexplore.cfg` is used (falling back to home directory).
- `--help` prints usage.

Direct Python run

```
python3 -m pip install -r /path/to/mdexplore/requirements.txt
python3 /path/to/mdexplore/mdexplore.py [--mermaid-backend js|rust] [PATH]
```

If `PATH` is omitted for direct run, the same config/home default rule applies.

File Highlights

- In the file tree, right-click any `.md` file.
- Choose a highlight color (`Highlight Yellow`, then `... <Color>`), or clear it.
- Colors are persisted in `.mdexplore-colors.json` in each affected directory.
- If a directory is not writable, color persistence fails quietly by design.
- Use the top-right "Copy to clipboard:" color buttons to copy files with a given highlight color.
- Use the pin button (first button after `Copy to clipboard:`) to copy the currently previewed markdown file.
- Right-click a directory to access recursive `Clear All` for that subtree.
- Copy/Clear operations are recursive and use scope in this order: selected directory, else most recently selected/expanded directory, else root.

PDF Export

- Click `PDF` (between `Refresh` and `Add View`) to export the currently previewed file.
- Output path is the previewed file path with `.pdf` extension in the same directory.
- Export is based on the active document content and print-prepared preview state (markdown + math + Mermaid + PlantUML).
- Export waits briefly for math/diagram/font readiness to reduce rendering artifacts.
- Mermaid PDF behavior is backend-specific:
  - JS backend: Mermaid is rendered through a print-safe monochrome/grayscale path.
  - Rust backend: Mermaid starts from a dedicated PDF SVG set rendered by `mmdr` (default theming), then applies print-safe grayscale normalization (multi-shade, with readable dark text).
- Export auto-scales page content into a print-style layout with top/side margins and an uncluttered footer band.
- Footer number font size is matched to the document's dominant scaled body text size.
- Pages are stamped with centered footer numbering as `1 of N`, `2 of N`, etc.

Known TODOs

- Diagram interaction state restore is not yet reliable across document switches:
  - Mermaid zoom/pan state may not restore consistently when returning to a file.
  - PlantUML zoom/pan state may not restore consistently when returning to a file.
- This is tracked as an explicit TODO for future maintenance; core markdown rendering, search, highlighting, and PDF export remain functional.

Multiple Views

- Click `Add View` to create another tab for the same currently previewed document.
- New tabs inherit the current view's top visible line/scroll position.
- By default, tab labels show the current top-most visible source line number for that tab.
- Tabs show a small left-side position bargraph indicating where that view sits within the document.
- Tabs use a fixed soft-pastel color sequence based on the order each view was opened.
- Tabs can be dragged to reorder without changing each tab's assigned color.
- Right-click a tab to assign a custom label (including spaces) up to 24 characters.
- Entering a blank custom label restores the default dynamic line-number label for that tab.
- When a tab receives a custom label, mdexplore stores that tab's current scroll position and top visible source line as the tab's saved beginning.
- Right-click a custom-labeled tab to use `Return to beginning`, which jumps that tab back to the stored label-time location.
- Relabeling a custom-labeled tab with different text resets the saved beginning to the scroll position at the time of relabeling.
- When a new view is added and the palette wraps, mdexplore skips any color already used by open tabs.
- If you switch to another markdown file and later return in the same app run, that file's tabs restore with their prior order and selected tab.
- View-tab state also persists across app restarts in `.mdexplore-views.json` beside the document directory, keyed by markdown filename.
- For custom-labeled tabs, `.mdexplore-views.json` also persists the stored label-time beginning location used by `Return to beginning`.
- If custom labels make the tab strip too wide for the window, tab scrolling is enabled through the tab bar's built-in scroll buttons.
- Only documents with more than one saved view, or with a custom tab label, are written to `.mdexplore-views.json`; untouched single-view documents continue to use the default one-tab state.
- The tab strip is hidden when there is only one unlabeled default view.
- If only one view remains and it has a custom label, its tab stays visible so the custom label and `Return to beginning` action remain available.
- Closing that sole remaining custom-labeled tab clears the custom label and bookmark, then returns the document to the hidden default single-view state.
- Tabs are closeable with `X`; at least one tab is always kept open.
- Maximum views per document: `8`.

Markdown Callouts

- mdexplore supports GitHub/Obsidian-style callouts written as blockquotes.
- Supported types: `NOTE`, `TIP`, `IMPORTANT`, `WARNING`, `CAUTION` (case-insensitive).
- `INFO` is accepted and styled as a `NOTE` callout.
- Custom titles are supported: `> [!WARNING] Custom title`.
- `+` / `-` markers are accepted in syntax, but callouts are rendered as non-collapsible boxes.

Example:

```
> [!NOTE]
> This is a note callout with **markdown** content.
```

Search and Match Highlighting

- Use `Search and highlight:` for non-recursive matching in the current effective scope.
- Matching files are shown in bold in the tree.
- Press `Enter` in the search field to run search immediately (skip debounce).
- Clicking the `X` in the search field clears search text and removes bolding.
- If search is active and you navigate to a different directory scope, search reruns automatically for that directory.
- Opening a matched file while search is active highlights matching text in yellow in the preview and scrolls to the first highlighted match.
- Preview scrollbar markers show where highlighted hits occur within the document; clicking a marker jumps to the nearest hit in that marker cluster.
- For `CLOSE(...)` queries, preview highlighting is constrained to the matched CLOSE window (not every occurrence of those terms in the document).
- Non-quoted terms are case-insensitive.
- Quoted terms are case-sensitive.
- Function-style operators accept both no-space and spaced forms before `(`: `CLOSE(...)` / `CLOSE (...)`, `OR(...)` / `OR (...)`, `AND(...)` / `AND (...)`, and `NOT(...)` / `NOT (...)`.
- `AND(...)`, `OR(...)`, and `CLOSE(...)` accept comma-delimited, space-delimited, or mixed argument lists.
- `AND(...)` and `OR(...)` are variadic and can take 2+ arguments.
- `CLOSE(...)` requires 2+ terms and matches only when all terms appear within 50 words of each other in file content.

Search examples:

```
joe
```

Matches `joe`, `JOE`, `JoE`, etc. (filename or content).

```
"Anne Smith"
```

Case-sensitive exact phrase match.

```
OR (Joe, "Fred")
```

Matches files containing either `Joe` (any case) or exact-case `"Fred"`.

```
CLOSE(Fred "Anne Smith" Joe)
```

Matches only if all listed terms occur within 50 words of each other.

```
Joe "Anne Smith" NOT draft
```

Implicit `AND`: equivalent to `Joe AND "Anne Smith" AND NOT draft`.

```
AND(alpha beta gamma)
```

Requires all three terms to match (equivalent to `alpha AND beta AND gamma`).

PlantUML Local Configuration

The app renders PlantUML blocks locally with `plantuml.jar`.

- Default jar path: `/path/to/mdexplore/vendor/plantuml/plantuml.jar`
- Override jar path with `PLANTUML_JAR`:

```
PLANTUML_JAR=/path/to/plantuml.jar /path/to/mdexplore/mdexplore.sh
```

Behavior details:

- Markdown preview loads immediately with `PlantUML rendering...` placeholders.
- Each diagram is replaced automatically as soon as it completes.
- Diagram replacements are applied in-place so scroll position is preserved.
- Failed diagrams show `PlantUML render failed with error ...` plus detailed stderr context (including line number when PlantUML provides one).
- Diagram progress continues while you browse other files; returning shows completed progress.

MathJax Local-First Configuration

Math rendering is local-first:

- mdexplore first tries a local `tex-svg.js` bundle (best rendering quality).
- If no local SVG bundle is found, it falls back to local `tex-mml-chtml.js`.
- If no local bundle is found, it falls back to CDN.

Local lookup order:

1. `MDEXPLORE_MATHJAX_JS` (explicit file path)
2. `mathjax/es5/tex-svg.js` under the repo
3. `mathjax/tex-svg.js` under the repo
4. `assets/mathjax/es5/tex-svg.js` under the repo
5. `vendor/mathjax/es5/tex-svg.js` under the repo
6. System paths such as `/usr/share/javascript/mathjax/es5/tex-svg.js`
7. Local CHTML bundle paths (`tex-mml-chtml.js`) as fallback

Example override:

```
MDEXPLORE_MATHJAX_JS=/absolute/path/to/tex-svg.js
/path/to/mdexplore/mdexplore.sh
```

Mermaid Local-First Configuration

Mermaid rendering is local-first:

- mdexplore first tries a local `mermaid.min.js` bundle.
- If no local bundle is found, it falls back to CDN.
- If `--mermaid-backend rust` is used, mdexplore uses `mmdr` instead of Mermaid JS for diagram SVG generation.

Local lookup order:

1. `MDEXPLORE_MERMAID_JS` (explicit file path)
2. `mermaid/mermaid.min.js` under the repo
3. `mermaid/dist/mermaid.min.js` under the repo
4. `assets/mermaid/mermaid.min.js` under the repo
5. `assets/mermaid/dist/mermaid.min.js` under the repo
6. `vendor/mermaid/mermaid.min.js` under the repo
7. `vendor/mermaid/dist/mermaid.min.js` under the repo
8. System paths such as `/usr/share/javascript/mermaid/mermaid.min.js`

Example override:

```
MDEXPLORE_MERMAID_JS=/absolute/path/to/mermaid.min.js
/path/to/mdexplore/mdexplore.sh
```

Rust backend executable override:

```
MDEXPLORE_MERMAID_RS_BIN=/absolute/path/to/mmdr /path/to/mdexplore/mdexplore.sh
--mermaid-backend rust
```

Render Architecture Map

- See `RENDER-PATHS.md` for the detailed render/caching architecture:
  - GUI vs PDF render flow.
  - Mermaid JS vs Rust backend branches.
  - Cache ownership and data flow between Python and in-page JavaScript.
  - Restore path after PDF export and why separate Mermaid cache modes exist.

Project Structure

```
mdexplore.py       # Qt application, file tree, renderer integration
mdexplore.sh       # launcher (venv create/install/run)
mdexplore.desktop.sample # sample desktop launcher entry for user customization
mdexplor-icon.png  # primary app icon asset (preferred)
requirements.txt   # Python runtime dependencies
README.md          # project docs
RENDER-PATHS.md    # detailed render/caching path diagrams + maintenance
narrative
AGENTS.md          # coding-agent maintenance guide
DESCRIPTION.md     # repository description + suggested topic tags
LICENSE            # MIT license
```

Development

Basic local checks:

```
bash -n mdexplore.sh
python3 -m py_compile mdexplore.py
```

Troubleshooting

If you see `ModuleNotFoundError: No module named 'PySide6.QtWebEngineWidgets'`:

- Re-run `mdexplore.sh`; it now performs a runtime import check and auto-reinstalls dependencies when the venv is incomplete.
- If it still fails, run:
  - `rm -rf /path/to/mdexplore/.venv`
  - `/path/to/mdexplore/mdexplore.sh`

If `Edit` does nothing:

- Ensure VS Code is installed.
- Run `code --version` and confirm it is available in your `PATH`.

If the dock/menu still shows an old app icon:

- Ensure your launcher contains:
  - `Icon=/absolute/path/to/mdexplor-icon.png`
  - `StartupWMClass=mdexplore`
- Refresh desktop entries:
  - `update-desktop-database ~/.local/share/applications`
- Remove old pinned `mdexplore` favorites from the dock and pin it again.
- Log out/in if the shell still shows the previous cached icon.

If running the launcher appears to do nothing:

- Watch terminal output; the launcher now prints setup/launch status.
- First dependency install can take time because Qt packages are large.
- Ensure you are in a GUI session (`DISPLAY` or `WAYLAND_DISPLAY` must be set).
- For desktop/dock launches (`Terminal=false`), check launcher log:
  - `~/.cache/mdexplore/launcher.log`
  - log is auto-trimmed to the most recent 1000 lines

Security Notes

- Markdown HTML is enabled (`html=True`), so preview untrusted Markdown with care.
- Mermaid uses local-first loading with CDN fallback.
- MathJax uses local-first loading with CDN fallback.

Contributing

1. Keep changes focused and small.
2. Run syntax checks before submitting.
3. Update docs (`README.md` , `AGENTS.md`) when behavior changes.

License

This project is licensed under the MIT License. See `LICENSE` .