Author Quick Start — Simple-CMS Prototype

One-page guide for authors to create, validate, collaborate, and publish lessons.

1) Open repository in VS Code - File → Open Folder → select repository root. - Accept recommended extensions (YAML, Markdown, markdownlint, Live Share).

2) Scaffold a lesson (recommended)

```
# scaffold and open in VS Code
python .\scripts\create_lesson.py "My Lesson Title" --course Microso
```

Or copy the template manually:

```
mkdir -Force Content\Microsoft-Security\Module-01
Copy-Item Templates\lesson-template.md Content\Microsoft-Security\Mo
code Content\Microsoft-Security\Module-01\my-lesson-001.md
```

3) Edit frontmatter and content - Update YAML frontmatter fields: `id`, `title`, `course`, `module`, `status` (Draft | Review | Approved | Published), `authors`. - Save frequently. Use Live Share for pair editing.

4) Validate metadata and sync index

```
python .\scripts\check_status_consistency.py --fix
```

5) When ready for review - Update frontmatter to `status: "Review"` and push a branch or invite reviewers via Live Share.

6) Approve and publish

When a reviewer decides the lesson is final, run:

```
python .\scripts\workflow_transition.py <article-id> Approved
```

This will move the file into `Content/Published/` and update `content_status.json` to `Published` automatically.

7) Commit & open PR

```
git add Content\<course>\<module>\*.md content_status.json
git commit -m "Add/update <article-id>"
git push --set-upstream origin demo/<your-branch>
# Open PR via GitHub web UI or use `gh pr create` (if you have GitHu
```

Notes - Use branches for parallel work or Live Share for real-time collaboration. - If the checker reports parse errors, install `pyyaml` in the environment used by `python`:

```
python -m pip install pyyaml
```

If you want, I can create and open the PR for you (I need `gh` auth or a GitHub token), or you can click the PR creation link that I've opened in your browser.

Simple-CMS Demo Cheat Sheet

One-page checklist and copyable commands to run the Guided Hands-On demo.

Quick checklist

- Open the repository root in VS Code and accept recommended extensions.

- Scaffold a lesson with `create_lesson.py` or copy the template into `Content`.

- Edit YAML frontmatter: `id`, `title`, `course`, `module`, `status`, `authors`.

- Run the consistency checker to sync `content_status.json`.

- Commit, push a branch, and open a PR to run CI checks.

Commands (PowerShell) — guided demo (copy-paste):

```
# 1) create module folder (if needed)
mkdir -Force Content\Microsoft-Security\Module-01

# 2) scaffold a lesson (creates file and optionally runs the checker
python .\scripts\create_lesson.py "Intro to Microsoft Security" --co

# 3) (or) copy template manually
Copy-Item Templates\lesson-template.md Content\Microsoft-Security\Mo
code Content\Microsoft-Security\Module-01\ms-security-001.md

# 4) after editing, validate and auto-fix JSON
python .\scripts\check_status_consistency.py --fix

# 5) commit & push
git add Content\Microsoft-Security\Module-01\*.md content_status.jso
git commit -m "Add ms-security-001 (Draft)"
git push --set-upstream origin demo/your-branch
```

Speaker notes (short)

- 0:00 — Open folder; accept recommended extensions.
- 0:20 — Scaffold or copy the template and open file.
- 0:45 — Edit frontmatter and save.
- 1:10 — Run `check_status_consistency.py --fix` to show automatic sync.

- 1:40 — Start Live Share to collaborate and iterate live.

- 2:30 — Commit and open PR; show CI checks running.

If you want a printable one-page PDF, print this markdown from VS Code or convert with `pandoc`.

Scripts Reference

This file provides a one-sentence explanation, an example command, and a brief usage note for every script in the `scripts/` folder.

- `add_article.py`

- Explanation: Creates a new article file from a template and adds an entry to the content index.

- Example: `python scripts/add_article.py "My New Article" --module "Module-01" --course "Microsoft-Security"`

  Usage: Run to scaffold a new markdown article with frontmatter in the correct course/module path.

  `check_status_consistency.py`

- Explanation: Validates and optionally synchronizes YAML frontmatter `status` fields with `content_status.json`, normalizing synonyms.

- Example: `python scripts/check_status_consistency.py --fix --from-frontmatter`

  Usage: Use `--fix` to apply corrections to the JSON index and `--from-frontmatter` to rebuild the index from files.

  `convert_frontmatter_to_json.py`

- Explanation: Scans content files and converts their YAML frontmatter into a single `content_status.json` index.

- Example: `python scripts/convert_frontmatter_to_json.py --output content_status.json`

  Usage: Run when you want to regenerate the status index from the repository's markdown files.

  `create_lesson.py`

- Explanation: CLI helper to scaffold a lesson (slugify title, write frontmatter, and create folder structure).

- Example: `python scripts/create_lesson.py "Intro to X" --course "Microsoft-Security" --module "Module-01"`

`--open`

Usage: Use to create new lessons; `--open` can launch your editor and `--run-check` triggers consistency checks.

`install_precommit_hook.ps1`

- Explanation: PowerShell script that installs a Git pre-commit hook for local linting and validation.

- Example: `. emplates\scripts\install_precommit_hook.ps1` (run from repo root)

Usage: Run once per developer environment to add local pre-commit checks.

`markdown_to_pdf.py`

- Explanation: Converts configured Markdown files into PDFs by rendering Markdown → HTML → PDF using `markdown` and `xhtml2pdf`.

- Example: `python scripts/markdown_to_pdf.py`

Usage: Generates `Author-QuickStart.pdf` and `Demo-cheatsheet.pdf`; edit the `FILES` list to convert other docs.

`move_to_published.py`

- Explanation: Moves a content file into the `Content/Published/` folder and updates the index metadata.

- Example: `python scripts/move_to_published.py Content/Microsoft-Security/Module-01/foo.md`

Usage: Use for explicit, single-file publish operations that relocate the article to the Published pool.

`move_to_stage.py`

- Explanation: Moves a content file to an arbitrary stage folder (e.g., Draft, Review, Published) and updates `content_status.json`.

- Example: `python scripts/move_to_stage.py Content/X/Module-01/foo.md Published`

Usage: Run to manually change an article's folder stage and sync its path/status in the index.

`publish_article.py`

- Explanation: High-level wrapper that marks an article as published and triggers configured publish-time updates (e.g., metadata changes).

- Example: `python scripts/publish_article.py intro-to-microsoft-security`

  Usage: Use to mark a slug as published; it may call the mover or workflow script under the hood.

  `sync_status.py`

- Explanation: Synchronizes the `status` field between a single markdown file's frontmatter and the `content_status.json` entry.

- Example: `python scripts/sync_status.py Content/Microsoft-Security/Module-01/foo.md`

  Usage: Handy for correcting or pushing a single file's status into the index or updating the file from the index.

  `update_status.py`

- Explanation: CLI utility to update an article's status in `content_status.json` (and optionally its frontmatter).

- Example: `python scripts/update_status.py intro-to-microsoft-security Approved --frontmatter`

  Usage: Use when you need to change the status value programmatically; add `--frontmatter` to write it back into the file.

  `workflow_transition.py`

- Explanation: Implements workflow rules (normalizes stage names) and triggers publish actions (moves file to Published on Approved and updates JSON/frontmatter).

- Example: `python scripts/workflow_transition.py intro-to-microsoft-security Approved`

- Usage: Preferred entrypoint for scripted status transitions that may need to move files and update audit fields.

If you want the exact CLI flags and full help text for any specific script, I can append the script's `--help` or print its argument parser output into this document.