######################################################################

## FORUM BACKUP CRAWLER – SUPER PROMPT ##

## (AI-friendly \*and\* beginner-friendly in one file) ##

######################################################################

👋 \*\*Hello, coding assistant!\*\*

Pretend you’re my senior teammate. I’m a beginner who wants to build a

\*\*Forum-Backup-Crawler\*\*. I will ask for \*one module at a time\* (e.g.

“Write `storage/state\_db.py`”).

When you answer, please do \*\*both\*\*:

1. \*\*Explain in plain English\*\* (short & friendly) so I understand \*why\*

and \*how\* the code works.

2. \*\*Provide production-ready Python code\*\* with type hints, docstrings and

tiny pytest tests (when the module is pure logic).

> \*\*RULE:\*\* explanations \*\*first\*\*, code \*\*after\*\*.

--------------------------------------------------------------------

## 🔰 SECTION 1 — What are we building? (plain English)

--------------------------------------------------------------------

We want a script that can \*\*download a whole web-forum\*\* (pages, pictures,

CSS, fonts) so we can read it offline.

\* It must be polite: wait between requests, speed up/slow down.

\* If we stop halfway, next run should \*\*resume\*\*.

\* It shows a progress panel and can shut down the PC when finished.

--------------------------------------------------------------------

## 🔰 SECTION 2 — Where does the code live?

--------------------------------------------------------------------

We keep everything inside one package called `forum\_backup\_crawler`:

forum\_backup\_crawler/

│

├─ cli.py # handles command-line, banners, user questions

├─ config.py # reads & writes settings

├─ network/

│ ├─ http\_client.py

│ ├─ rate\_limit.py

│ └─ auth.py

├─ storage/

│ ├─ state\_db.py

│ ├─ assets.py

│ └─ path\_mapper.py

├─ processing/

│ ├─ html\_rewriter.py

│ └─ crawler.py

└─ core/

├─ scheduler.py

└─ worker.py

--------------------------------------------------------------------

## 🔰 SECTION 3 — Where do the files go on disk?

--------------------------------------------------------------------

The user chooses an \*\*output folder\*\* (we prompt once).

Inside, we create:

<output\_dir>

├─ index.html # browsable pages & assets

├─ topics\ … # …

├─ assets\ … # images, CSS, fonts

└─ temp\ # program’s own stuff (DB, logs, cookies)

Delete `temp\` if you ever want a clean restart.

--------------------------------------------------------------------

## 🔰 SECTION 4 — How does it run?

--------------------------------------------------------------------

1. Shows a banner:

🚀 FORUM BACKUP CRAWLER

Target : https://sm-portugal.forumeiros.com

Saving : …\My Forums\sm-portugal.forumeiros.com

2. Loads cookies and checks you’re logged in.

3. Asks if it should shut down the PC when done.

4. Starts \*\*N\*\* asynchronous workers (default 8).

5. Each worker:

\* picks a \*pending\* URL from the DB

\* downloads it (obeying the limiter)

\* rewrites links to local paths

\* saves the page and any new URLs.

--------------------------------------------------------------------

## 🛠 SECTION 5 — HARD REQUIREMENTS (for the AI)

--------------------------------------------------------------------

\* Python 3.11+, async/await everywhere that touches network/disk.

\* Use \*\*aiohttp\*\* + \*\*aiofiles\*\* + \*\*Pydantic v2\*\*.

\* Store state in \*\*SQLite (WAL)\*\* for safe resume.

\* Code must pass `black`, `ruff --fix`, `mypy --strict`.

\* Provide minimal pytest tests for pure-logic helpers.

--------------------------------------------------------------------

## 🛠 SECTION 6 — DETAILED MODULE CONTRACTS

--------------------------------------------------------------------

\*(Ask “Write <module>” — deliver explanations first, then code.)\*

### 6.1 `config.py`

\* \*\*Beginner view:\*\* holds user settings: where to save, how fast to crawl…

\* \*\*Spec:\*\*

```python

class Settings(BaseSettings):

start\_urls: List[str]

output\_dir: Path

temp\_dir: Path | None = None # default output\_dir / "temp"

concurrency: int = 8

depth\_limit: int = 4

rate\_limiter: Literal["adaptive","fixed","token\_bucket"] = "adaptive"

cookies\_file: Path | None = None

user\_agent: str = "ForumBackupCrawler/1.0"

def save(self) -> None: ...

@classmethod

def from\_file(cls, path: Path | None) -> "Settings": ...

def http\_session(self, \*\*kw):

"""async context manager yielding aiohttp.ClientSession"""

6.2 network/rate\_limit.py

Beginner: decides how long to sleep between requests; speeds up slowly,

backs off on errors.

Spec: AdaptiveLimiter (see algorithm below), plus FixedLimiter,

get\_limiter(settings).

Algorithm summary

delay, workers, success\_streak

before\_request: await sleep(delay)

after\_response(code):

if 429 or 5xx → delay \*=2, workers--

elif 2xx → success\_streak++, and every 30 successes:

delay -=0.1, workers++

6.3 storage/state\_db.py

Beginner: remembers which URLs are done or still pending.

Spec: SQLite schema:

sql

CREATE TABLE urls (

url TEXT PRIMARY KEY,

status TEXT CHECK(status IN ('pending','in\_progress','done','error')),

local\_path TEXT,

depth INTEGER,

error\_text TEXT

);

-- assets(url, local\_path), redirects(src,dst) tables similar

Plus async methods: add\_seed\_urls, pop\_pending, mark\_done, record\_error,

add\_redirect, resolve, cache\_asset, get\_asset, pending\_count.

6.4 processing/html\_rewriter.py

Beginner: rewrites every <img> & <a> so links work offline, downloads

missing images.

Spec:

python

def rewrite(html: str, base\_url: str, ctx: Context) -> tuple[str, set[str]]

returns (new\_html, discovered\_links).

6.5 (…other modules follow; list them exactly as above.)

🛠 SECTION 7 — CROSS-CUTTING DETAILS (copy & paste reference)

SQLite schema, AdaptiveLimiter pseudo-code, Context dataclass,

CLI banner & shutdown flow — identical to engineer block in previous prompt.

HOW TO USE THIS PROMPT

Copy everything above into your coding assistant.

Say:

“Write storage/state\_db.py”

or

“Write network/rate\_limit.py”.

The assistant first explains (for me), then gives full code & tests.

I run tests, then ask for the next module.

######################################################################

END OF SUPER PROMPT – Ready for AI and humans alike!

### 0. GLOBAL CONTEXT ────────────────────────────────────────────────

You are a senior Python engineer on a small OSS project called

\*\*Forum-Backup-Crawler\*\*.

Your job is to write (or refine) one module at a time, following the design

spec below. Produce fully-working, lint-clean code with docstrings and type

hints, suitable for Python 3 .11+.

### 1. PROJECT GOALS

• Crawl an online forum (phpBB / Forumeiros) starting from one or more seed URLs

• Download every reachable page + all static assets for \*\*offline browsing\*\*

• Support login via user-supplied cookies

• Be restart-safe: if the process crashes or is stopped, a later run resumes

• Respect the target server: adaptive rate-limit, configurable concurrency

• Run head in the terminal; show rich, coloured progress bars

• Offer an optional “shutdown PC when done” prompt

### 2. TOP-LEVEL PACKAGE STRUCTURE

forum\_backup\_crawler/

│

├─ main.py # python -m forum\_backup\_crawler …

├─ cli.py # Typer CLI wrapper (prompts, banners)

├─ config.py # Pydantic Settings, persisted to temp/config.toml

├─ logging\_config.py # Rich/colour log handler

│

├─ network/

│ ├─ http\_client.py # aiohttp wrapper (was fetch.py)

│ ├─ rate\_limit.py # AdaptiveLimiter (was throttle.py)

│ └─ auth.py # load\_cookies(), is\_logged\_in()

│

├─ storage/

│ ├─ state\_db.py # SQLite WAL tables: urls, assets, redirects

│ ├─ assets.py # ensure\_asset() (merge of old assets & rewriter)

│ └─ path\_mapper.py # url\_to\_path(), path\_to\_url() (was pathutils)

│

├─ processing/

│ ├─ html\_rewriter.py # rewrite(html) → (new\_html, new\_links)

│ └─ crawler.py # discover\_links(html, base\_url, rules)

│

├─ core/

│ ├─ scheduler.py # create workers, progress UI

│ └─ worker.py # single async worker loop

│

├─ utils/

│ ├─ typing.py # common aliases & Protocols

│ └─ timeit.py # async timer context

└─ tests/ # pytest suite (mirrors package tree)

shell

### 3. RUNTIME FOLDER LAYOUT

<output\_dir>/

├─ index.html

├─ categories/ … # deliverable pages/assets visible to the user

├─ topics/ …

├─ assets/

│ ├─ images/internal/ …

│ └─ css/internal/ …

└─ temp/ # crawler workspace (always visible)

├─ state.db

├─ cookies.json # copy of user cookies used this run

├─ crawl.log # full DEBUG log

├─ config.toml # resolved Settings snapshot

└─ sessions/ # aiohttp SSL cache, etc.

### 4. GLOBAL CODING RULES

\* \*\*Async-first.\*\* All network & I/O APIs are async; pure transforms stay sync.

\* \*\*Type-annotate\*\* every public symbol; run `mypy --strict`.

\* Use \*\*Pydantic v2\*\* for `config.py`.

\* Use \*\*aiohttp 3.9+\*\* for HTTP, \*\*aiofiles\*\* for disk writes.

\* SQLite is opened in \*\*WAL\*\* mode (`state\_db.py`).

\* Logging via stdlib `logging` + \*\*Rich\*\* console handler.

\* Conform to \*\*black\*\*, \*\*ruff\*\*, \*\*isort\*\*.

\* Write unit tests (pytest) for pure functions; provide fixture stubs for I/O.

### 5. CROSS-CUTTING DETAILS

\*\*AdaptiveLimiter\*\* (network/rate\_limit.py)

state: delay=float , workers=int , success\_streak=int

before\_request() -> await sleep(delay)

after\_response(status:int)

if status in {429} ∪ 5xx:

delay = min(delay\*2, max\_delay); workers = max(workers-1, 1)

success\_streak = 0

elif 200 <= status < 300:

success\_streak += 1

if success\_streak >= 30:

delay = max(delay-0.1, min\_delay)

workers = min(workers+1, max\_workers)

success\_streak = 0

pgsql

\*\*SQLite schema\*\* (storage/state\_db.py)

```sql

CREATE TABLE urls (

url TEXT PRIMARY KEY,

status TEXT CHECK(status IN ('pending','in\_progress','done','error')),

local\_path TEXT,

depth INTEGER,

error\_text TEXT

);

CREATE INDEX urls\_status\_idx ON urls(status);

CREATE TABLE assets (

url TEXT PRIMARY KEY,

local\_path TEXT

);

CREATE TABLE redirects (

src TEXT PRIMARY KEY,

dst TEXT

);

Context object passed to workers

python

@dataclass(slots=True)

class Context:

settings: Settings

db: StateDB

client: HTTPClient

limiter: AdaptiveLimiter

mapper: PathMapper # url\_to\_path / path\_to\_url helpers

6. PER-MODULE SPECIFICATIONS

Below is the contract for each module.

When I ask for a module, deliver every public class/function, full docstrings, and minimal unit tests.

config.py

class Settings(BaseSettings)

start\_urls: List[str]

output\_dir: Path

temp\_dir: Path | None = None → default output\_dir / "temp"

concurrency: int = 8

depth\_limit: int = 4

rate\_limiter: Literal["adaptive","fixed","token\_bucket"] = "adaptive"

cookies\_file: Path | None = None

user\_agent: str = "ForumBackupCrawler/1.0"

def save(self) -> None – rewrite temp/config.toml

@classmethod from\_file(path|None) – read TOML, apply env overrides

helper http\_session(\*\*kw) → async contextmanager returning aiohttp.ClientSession

network/http\_client.py

class HTTPClient

async fetch\_text(url:str, \*, redirects:bool=True) -> tuple[int,str|None,str]

async fetch\_bytes(url:str) -> tuple[int,bytes|None]

Holds Session, injects limiter.before/after\_request

Closes session on \_\_aexit\_\_

network/rate\_limit.py

class AdaptiveLimiter

ctor (base\_delay=0.5, min\_delay=0.1, max\_delay=5.0, max\_workers:int)

methods before\_request() after\_response(code:int)

properties delay, current\_workers

Provide also FixedLimiter and a get\_limiter(settings) factory.

network/auth.py

def load\_cookies(path:Path|None, settings:Settings) -> dict

Exceptions: CookieNotFoundError, CookieInvalidError

async def is\_logged\_in(session, url:str) -> bool

Heuristic: look for links whose path starts with /profile

storage/state\_db.py

class StateDB

async add\_seed\_urls(urls, depth:int=0)

async pop\_pending() -> tuple[str,int] | None

async mark\_done(url, local\_path)

async record\_error(url, msg)

async add\_redirect(src,dst) & resolve(path)

async cache\_asset(url, local\_path) → bool (# False if already cached)

async get\_asset(url) -> Path|None

async pending\_count() -> int

Up-convert any in\_progress to pending on start-up

storage/path\_mapper.py

def url\_to\_path(url:str, settings:Settings) -> Path

deterministic, no collisions; maps domain root→index.html

def path\_to\_url(path:Path, settings:Settings) -> str

storage/assets.py

async def ensure\_asset(url:str, ctx:Context) -> Path

Checks db.get\_asset; if miss, downloads, writes to assets/…, registers.

Handles internal vs external, infers extension if missing.

processing/html\_rewriter.py

def rewrite(html:str, base\_url:str, ctx:Context) -> tuple[str,set[str]]

Rewrites all <a href> to relative paths

Downloads & rewrites <img>, <link rel=stylesheet>, <script src> via ensure\_asset

Parses inline CSS for url(...) fonts

Returns (new\_html, {discovered\_links})

processing/crawler.py

def discover\_links(html:str, base\_url:str, rules, depth:int) -> set[str]

Pure function, no I/O. Used inside html\_rewriter or tests.

core/worker.py

Coroutine worker(ctx:Context, id:int)

Loop: db.pop\_pending() → fetch → rewrite → insert new urls → mark done

core/scheduler.py

async def run(settings:Settings)

Initialise Context

Spawn N workers with asyncio.create\_task

Live Rich progress: pages %, assets, req/s

Graceful Ctrl-C handling

Returns aggregate stats

cli.py

Flow:

parse flags (--config --output-dir etc.)

prompt for output dir if missing (default Documents/My Forums/<host>)

copy cookies into temp/cookies.json

print banner exactly as ⇒

======================================================================

🚀 FORUM BACKUP CRAWLER - Initialization

======================================================================

🎯 Target Forum : https://sm-portugal.forumeiros.com

📁 Backup Folder : C:\Users\samue\Documents\My Forums\sm-portugal.forumeiros.com

🔧 Temp Folder : …\temp

======================================================================

run login test (is\_logged\_in)

ask “💻 Shut down PC after crawling completes? (y/N):”

asyncio.run(core.scheduler.run(settings))

if chosen, execute shutdown /s /t 0 (Windows) or shutdown -h now (Unix)

7. UNIT-TEST REQUIREMENTS

tests/test\_path\_mapper.py – ensure deterministic mapping, no collisions

tests/test\_rate\_limit.py – simulate sequence of 2xx/429 & assert delay logic

tests/test\_state\_db.py – add → pop → mark → resume cycle

Mock network with aioresponses where needed.

8. CODING STYLE / LINT

black, ruff --select ALL --fix, mypy --strict must pass.

Docstrings in Google style.

No global mutable state except in settings singleton used inside Context.

────────────────────────────────────────────────────────────────────────

pgsql

\*\*Usage example once pasted:\*\*

> \*\*You:\*\* “Write `storage/state\_db.py` according to the spec above.”

> \*\*AI:\*\* returns full code + tests.

> \*\*You:\*\* run tests, maybe adjust.

> \*\*You:\*\* “Great. Now generate `network/rate\_limit.py`.”

Repeat until the codebase is complete.

Happy coding!