

Web Scrapping Project Documentation

Overview:

The project consists of 4 core elements: Crawler, Scheduler, Apis & Tests.

Crawler: It is a program written in python which crawls a website (<https://books.toscrape.com>) using the popular scraping library beautifulsoup.

Scheduler: It is a python program that runs continuously and runs the crawler after a certain time to find any new or modified data in the website. If it finds any the database (DB) is updated accordingly, and json and csv reports are generated after the scheduler has finished crawling the entire website.

Apis: They are codes written in python using FastApi to query and retrieve book data stored in the DB.

Tests: They are used to test the apis and the crawler.

- Incode documentation is also provided for better understanding.

Primary Technologies and Libraries used:

Crawler: beautifulsoup

Scheduler: apscheduler

Apis: FastAPI

Tests: pytest

Database: MongoDB (MongoDB Compass)

Detailed Sections of The Project

Crawler:

This module contains an asynchronous web crawler designed to scrape **books.toscrape.com** (or any similar site) and store book metadata, snapshots, and change logs into a MongoDB database.

It handles:

- Fetching all paginated book links

- Scraping book details
- Detecting content changes via hashing
- Storing snapshots only when content changes
- Parallel crawling with concurrency limits
- Returning a list of all changes detected on a run

Environment Variables

- **BASE_URL** – Base website to crawl
- **CRAWL_CONCURRENCY** – Number of simultaneous requests
- **CRAWL_RETRIES** – Number of retry attempts for failed requests

Defaults:

- BASE_URL = "https://books.toscrape.com"
- CONCURRENCY = 10
- RETRIES = 3

Main Class: Crawler

`__init__(base_url, concurrency)`

Initializes the crawler with:

- Base URL
- Async HTTP client
- Semaphore to limit concurrency

`async fetch(url)`

Fetches HTML from a URL with:

- Concurrency limit
- Automatic retries
- Error logging

Returns: **HTML string**

`async get_all_book_links()`

Scrapes *every page* of the catalog to collect all book detail URLs.

Workflow:

1. Start at page 1
2. Extract <a> links for all books
3. Follow the **Next** button until pages end
4. Deduplicate links while keeping order

Returns: **List of book detail URLs**

`parse_rating(soup)`

Extracts star rating based on CSS class:

"One" → 1, "Two" → 2, ...

Returns: **Integer or None**

`parse_book(html, url)`

Parses a single book page and extracts:

- Title
- Description
- Category
- Prices (incl/excl tax)
- Availability
- Number of reviews
- Image URL
- Rating
- Unique book ID (derived from URL path)
- Timestamp
- Content hash

Returns:

(book_data_dict, raw_html)

`async process_book(url)`

Processes a single book.

Behavior:

If book exists in DB

- Compares stored hash vs. scraped hash
- If changed:
 - Create snapshot
 - Update book record
 - Insert change log
 - Return change info
- If unchanged:
 - No snapshot created
 - Returns None

If book is new

- Create snapshot
- Insert book
- Insert change log
- Return change info

Returns: **Change record or None**

`async run(resume=False)`

Main crawling workflow:

1. Get all book links
2. Process each book concurrently
3. Collect change records

Returns: **List of changes detected this run**

Helper Functions

`async crawl_all_books()`

Utility for schedulers:

1. Runs the crawler
2. Fetches all books from DB (without raw HTML)
3. Returns them as a list

`async main()`

Simple script entry point:

```
asyncio.run(c.run())
```

Summary of Data Saved

Books Collection

- Metadata fields (title, category, price, etc.)
- content_hash
- raw_snapshot_id (link to snapshots collection)

Snapshots Collection

- Full HTML
- Timestamp
- Source URL

Change Log Collection

- book_id
- change_type (new / updated)
- timestamps
- old vs new hash (for updates)

Book Model Using Pydantic

```
class Book(BaseModel):
    id: str = Field(..., description="Unique id")
    title: str
    description: Optional[str]
    category: Optional[str]
    price_including_tax: Optional[float]
    price_excluding_tax: Optional[float]
    availability: Optional[str]
    num_reviews: Optional[int]
    image_url: Optional[HttpUrl]
    rating: Optional[int] # 1-5
    source_url: HttpUrl
    crawl_timestamp: datetime
    content_hash: str
    raw_snapshot_id: Optional[str]
```

Scheduler:

This module manages **automated crawling** of the book website using APScheduler. It triggers the crawler periodically, detects changes, and generates a daily report from those changes.

Logging Setup

A logger named "scheduler" is configured to print timestamped logs to the console to help track job execution and errors.

Function: `scheduled_crawl()`

An asynchronous task executed by the APScheduler.

Workflow:

1. Logs start of scheduled crawl
2. Creates a Crawler instance
3. Runs the crawler using `c.run()`
 - Returns a list of **new or updated books** detected during this run
4. Logs how many changes were detected
5. Calls `generate_daily_report(changes)`
 - Generates a report *only* from this run's changes
6. Closes the crawler's HTTP client

Returns:

Nothing, but generates a report and updates the DB.

Function: `async_main()`

Sets up and starts the APScheduler.

Behavior:

- Creates an AsyncIOScheduler for async tasks

Adds a job:

```
scheduler.add_job(scheduled_crawl, "interval", minutes=15)
```

- → Runs `scheduled_crawl()` **every 15 minutes**
- Starts the scheduler
- Keeps the program alive indefinitely using `await asyncio.Event().wait()`

Script Entry Point

If the file is executed directly:

```
asyncio.run(async_main())
```

This launches the scheduler, which will then execute the crawl job every 15 minutes.

Summary

- `scheduled_crawl()`
→ Runs crawler → gets changes → generates report
- `async_main()`
→ Starts APScheduler → schedules the job → keeps program running
- APScheduler interval: **15 minutes**

Alert & Report:

These modules are responsible for creating and sending reports respectively, after the scheduler crawls the entire website at a defined time interval.

Alert

The `alert.py` file is responsible for sending an email with the generated reports after each scheduler crawl. For testing purposes, an email is sent regardless of whether there are any changes or not.

Report

The `report.py` file generates csv & json reports every time the scheduler crawls the website. The files contain any change in data of the website or no change message.

APIs:

This module defines the **FastAPI backend** for the Book Scraper system. It exposes endpoints to retrieve books, book details, and change logs. It also enforces API key authentication, rate limiting, pagination, and optional filtering.

App Initialization

- Loads environment variables (e.g., API_PORT)
- Creates a FastAPI app with metadata
- Registers global rate limiting via SlowAPI
- Enables CORS for all origins (useful for frontend access)

API Key Authentication

All endpoints include:

```
dependencies=[Depends(get_api_key)]
```

Only clients providing a valid API key can access the API.

Rate Limiting

Each route uses:

```
@limiter.limit("X/hour")
```

Examples:

- /books: **100 requests/hour**
- /books/{book_id}: **5 requests/hour**
- /changes: **100 requests/hour**

This protects the API from abuse.

book_doc_to_resp(doc)

A helper function that extracts only the allowed fields from a MongoDB document before returning it to the client.

It prevents leaking internal fields like raw_snapshot_id or content_hash.

Endpoint: GET /books

Returns a **paginated list of books**, with optional filters and sorting.

Supported Query Parameters:

- category — filter by book category
- min_price, max_price — price range filter
- rating — filter by rating (1–5)
- sort_by — one of:
 - "rating" (desc)
 - "price" (asc)
 - "reviews" (desc)
- page, page_size — pagination controls

Response Body:

```
{
  "page": 1,
  "page_size": 20,
  "total": 1000,
  "results": [ ...list of books... ]
}
```

The query is translated into a MongoDB filter, sorted if needed, then paginated manually using skip() and limit().

Endpoint: GET /books/{book_id}

Returns **one book** by its ID.

- Looks up _id in MongoDB
- If not found → returns 404 Book not found
- Otherwise → returns sanitized book document

Endpoint: GET /changes

Returns recent changes (new or updated books) from the change_log collection.

Details:

- Sorted by newest first

- Limited by query parameter limit (default 100)
- Converts `_id` and `book_id` to strings for JSON compatibility

Response:

```
{
  "results": [
    {
      "_id": "log123",
      "book_id": "bk021",
      "change_type": "updated",
      "timestamp": "...",
      "old_hash": "...",
      "new_hash": "..."
    }
  ]
}
```

Running the API

Use the following command:

```
uvicorn api.main:app --host 0.0.0.0 --port <API_PORT> --reload
```

Rate Limiter:

This module sets up **rate limiting** for the FastAPI application using **SlowAPI**.

limiter

```
limiter = Limiter(key_func=get_remote_address)
```

- Creates a global Limiter instance
- `key_func=get_remote_address` → limits requests per client IP address
- Can be used as a decorator on FastAPI routes to enforce rate limits, e.g.:

```
@limiter.limit("100/hour")
```

register_rate_limit(app: FastAPI)

Registers the Limiter with the FastAPI app.

Actions:

1. Attaches the limiter to `app.state.limiter`
2. Registers a **429 Too Many Requests** exception handler to return a proper response when a client exceeds the limit.

Usage:

```
from fastapi import FastAPI
from api.rate_limit import register_rate_limit
```

```
app = FastAPI()
register_rate_limit(app)
```

After this, any route decorated with `@limiter.limit()` is enforced, and exceeding clients get a 429 response.

Summary

- Provides a **global rate limiter** for the API
- Limits are **per client IP**
- Handles **429 errors gracefully**
- Works seamlessly with FastAPI decorators

Tests:

conftest.py

This module provides **fixtures and fake database classes** for testing the FastAPI Books Scraper API without requiring a real MongoDB instance. It enables **unit and integration tests** of API endpoints in isolation.

Key Components

1. Fake MongoDB Classes

- **FakeCursor**
Simulates a Motor cursor with support for `sort()`, `skip()`, `limit()`, and `to_list()`.
- **FakeCollection**
Simulates a MongoDB collection with basic methods:

- `find_one()`, `find()`
 - `insert_one()`, `update_one()`
 - `count_documents()`
- Supports simple filtering, range queries (e.g., `$gte`, `$lte`), and sorting.
- **FakeDB**
Contains two collections:
 - `books` → stores book documents
 - `change_log` → stores book change records

These classes allow testing database-dependent API logic without a real database.

2. Pytest Fixtures

- **sample_books** → Returns a list of sample book documents for testing.
- **fake_db** → Returns a FakeDB instance preloaded with sample books and change logs.
- **client** → Provides an AsyncClient for testing the FastAPI app with:
 - `get_db` patched to return `fake_db`
 - API key dependency overridden to allow "testapikey"
 - Uses ASGITransport to make requests to the app in-memory

Purpose

This setup allows writing **async API tests** that:

- Query books with filters, sorting, and pagination
- Retrieve individual book details
- Fetch change logs
- Test authentication and rate-limiting logic

All without touching a real database or network.

test_api.py

This module contains **asynchronous pytest tests** for the FastAPI Books Scraper API.

It uses the client fixture (with a fake database and overridden API key) to test endpoints **without a real database or network**.

Tested Endpoints

1. **GET /books** – List books with optional filters, sorting, and pagination
2. **GET /books/{book_id}** – Retrieve a single book by ID
3. **GET /changes** – Retrieve recent changes (new or updated books)
4. **Authentication and rate limiting**

Tests Overview

Test	Purpose
test_list_books_no_filters	Returns all books when no filters are applied
test_list_books_category_filter	Returns only books in a specific category
test_list_books_price_range	Returns books within a specified price range
test_list_books_sorting_and_pagination	Validates sorting by rating and pagination
test_get_book_by_id_found	Returns correct book when ID exists
test_get_book_by_id_not_found	Returns 404 when book ID does not exist
test_changes_endpoint_default_limit	Returns all changes with default limit
test_changes_endpoint_with_limit	Returns only the specified number of change entries
test_unauthorized_access	Verifies 401 Unauthorized when API key is missing
test_validation_error_for_bad_query_param	Verifies 422 response for invalid query parameters
test_rate_limit_hit	Simulates rate limiting, expecting 429 if limit is exceeded

Key Features

- Uses **async tests** (@pytest.mark.asyncio) with AsyncClient
- Validates **response status codes**, payload structure, and content
- Covers **filtering, sorting, pagination, and authentication**
- Tests **rate limiting behavior** using SlowAPI configuration
- Fully isolated with **fake database and API key override**

Summary

These tests ensure that:

1. API endpoints behave correctly with various query parameters.
2. Authentication and rate limiting are enforced.
3. Edge cases (missing book, invalid input, query limits) are handled gracefully.

The test suite can be run with:
`pytest tests/test_api.py`

test_crawler.py

This module contains **asynchronous tests** for the Crawler class in the Books Scraper system.

Tested Functionality

`get_all_book_links()`

- Ensures the crawler correctly **collects all book detail URLs** from paginated catalogue pages.
- Uses **monkeypatching** to simulate HTTP responses, avoiding actual network requests.

Test Overview

Test	Purpose
<code>test_get_all_book_links_monkeypatched</code>	Simulates two catalogue pages with next links and multiple books. Verifies that <code>Crawler.get_all_book_links()</code> : <ul style="list-style-type: none">• Follows pagination• Extracts all book URLs• Preserves correct order

Key Features

- **monkeypatch** replaces `Crawler.fetch` with `fake_fetch` returning predefined HTML.
- Tests **HTML parsing, pagination, and URL joining** without external dependencies.
- Verifies that the crawler correctly returns all expected links.

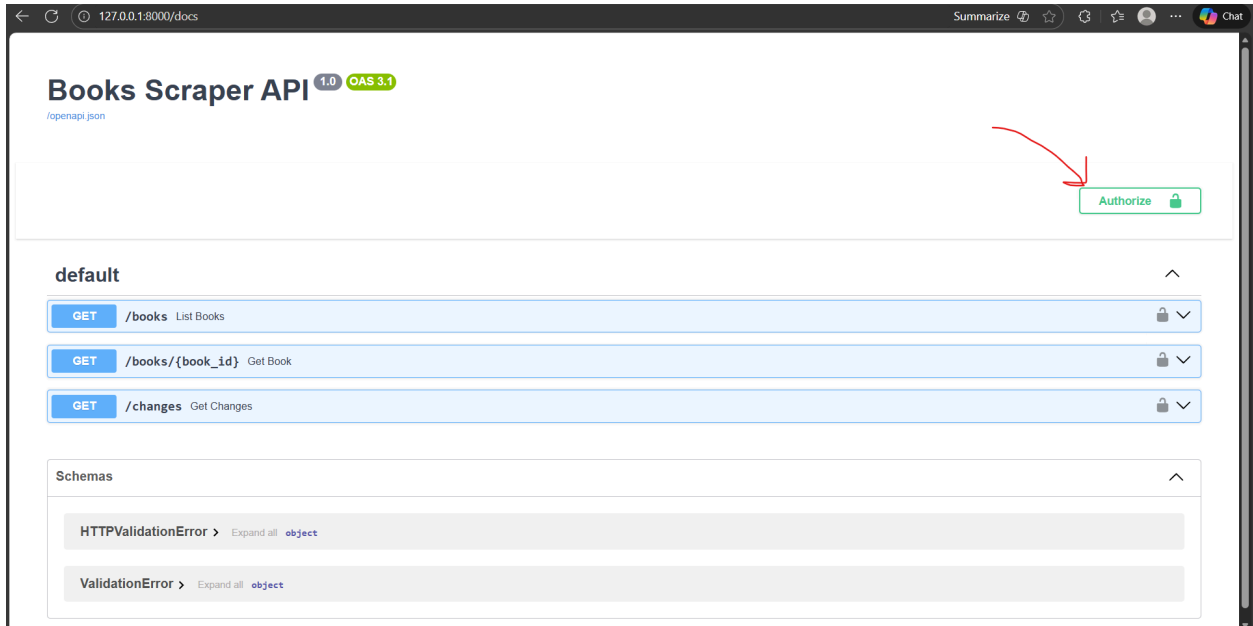
Summary

This test ensures the crawler:

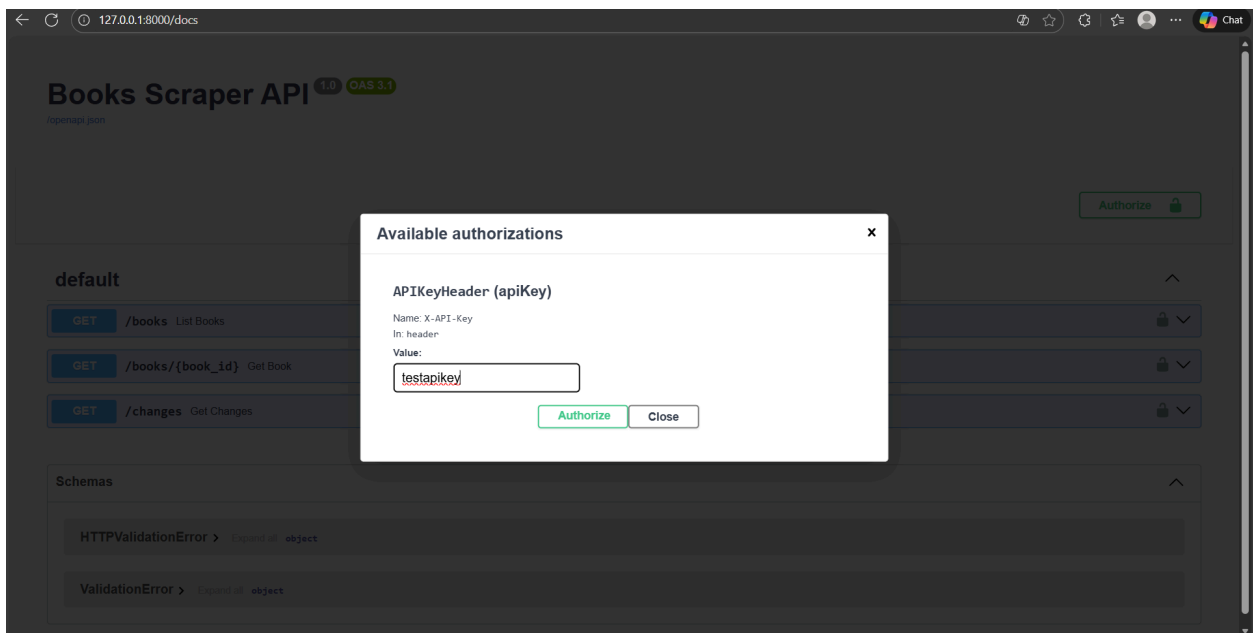
1. Handles multi-page catalogues.
2. Extracts book URLs from the HTML correctly.
3. Returns a deduplicated, ordered list of links.

Swagger UI for API testing:

- To use the UI for testing go to the following url (after turning on the server):
`http://127.0.0.1:8000/docs`
- After going to the UI click on the authorize button to enter the authorization header.



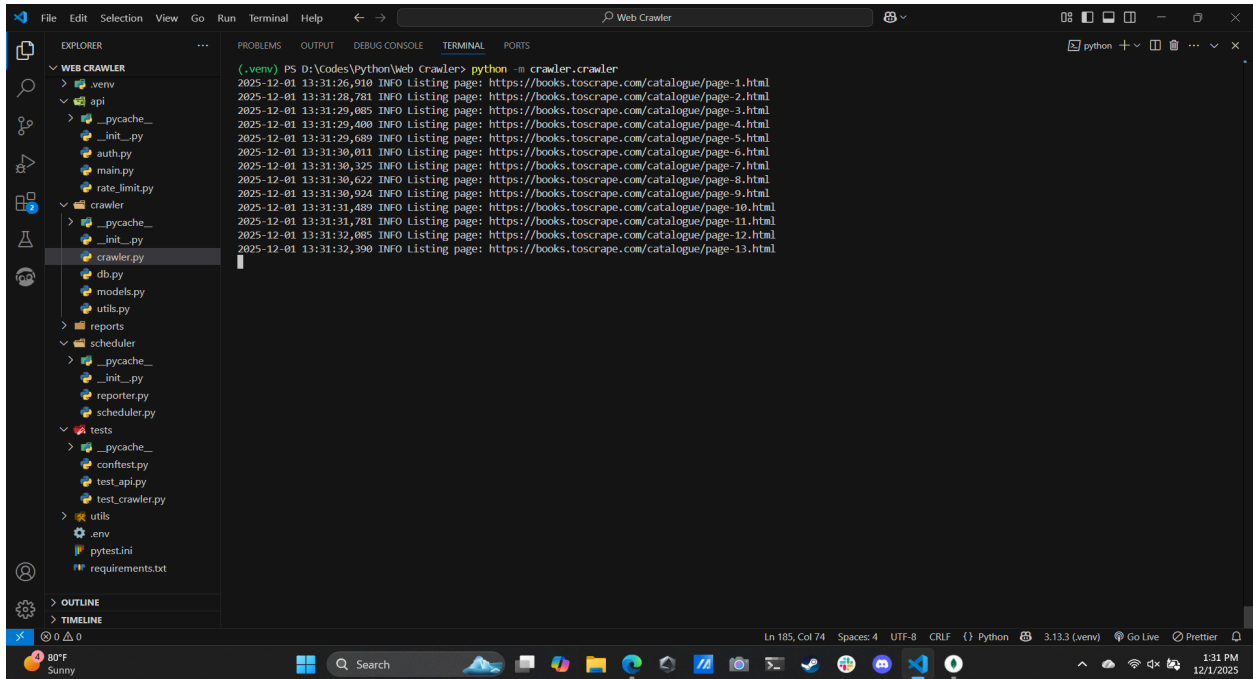
- Then input the value “testapikey” as the authentication header value like in the picture below:



- Now you can test the APIs.

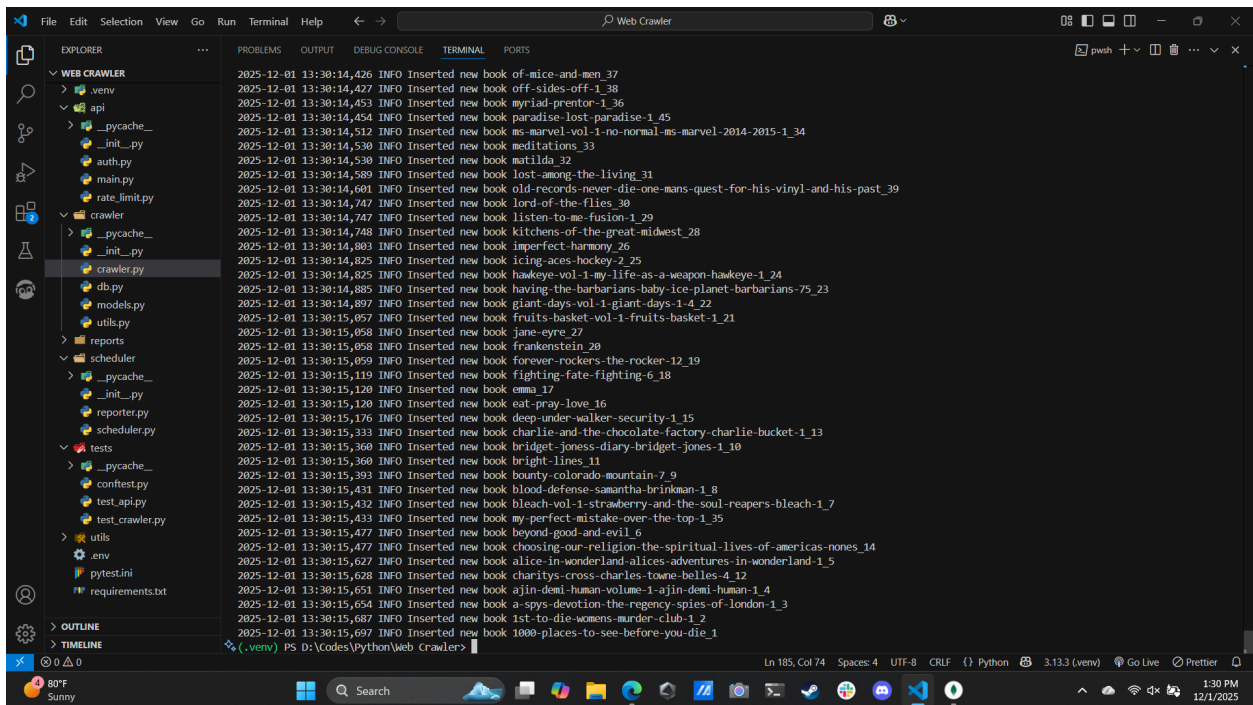
Important Screenshots:

1) Crawler starting to crawl the website.



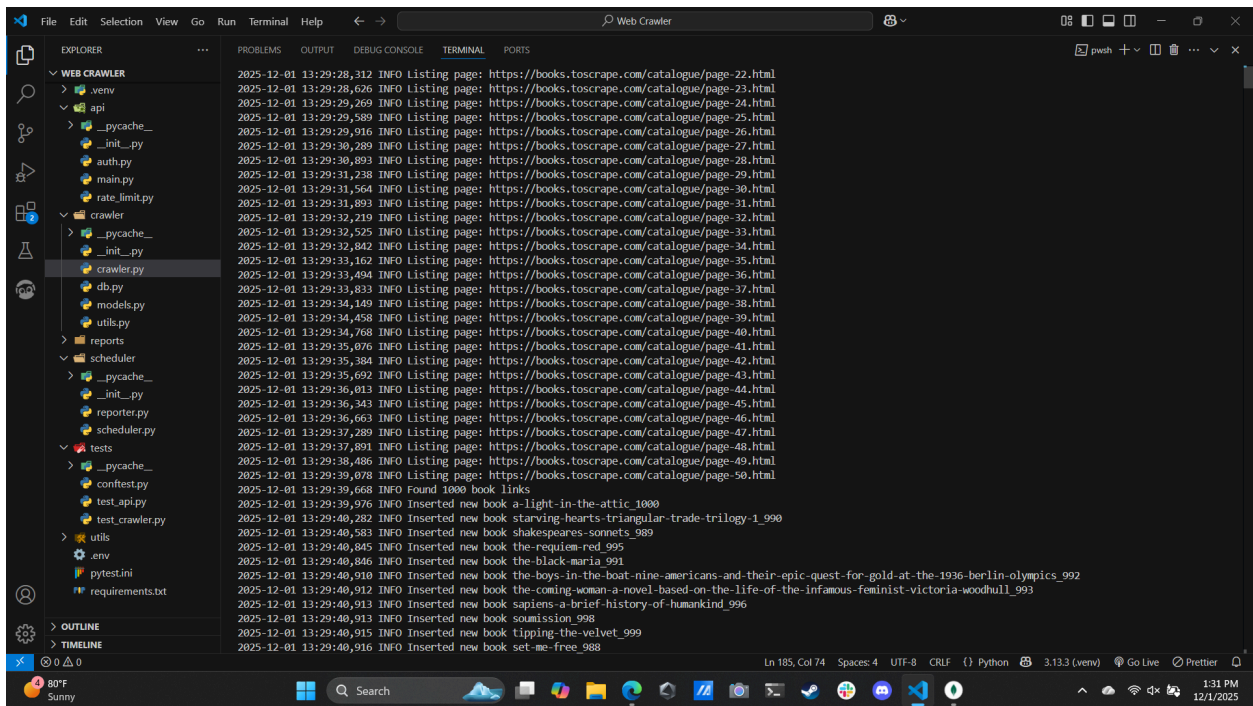
The screenshot shows the Visual Studio Code interface with the 'Web Crawler' project open. The Explorer pane on the left shows the project structure, including folders like 'api', 'crawler', 'scheduler', and 'tests'. The 'crawler' folder is expanded, showing files like 'crawler.py', 'db.py', 'models.py', 'utils.py', 'reports', 'reporter.py', and 'scheduler.py'. The 'crawler.py' file is selected. The Output pane on the right shows the terminal output of the crawler, displaying a list of URLs being crawled, such as 'https://books.toscrape.com/catalogue/page-1.html' through 'https://books.toscrape.com/catalogue/page-13.html'. The status bar at the bottom indicates the file is at line 185, column 74, in the 'python' language, with a file encoding of 'UTF-8' and a line ending of 'CRLF'.

2) Crawler inserting books into MongoDB.



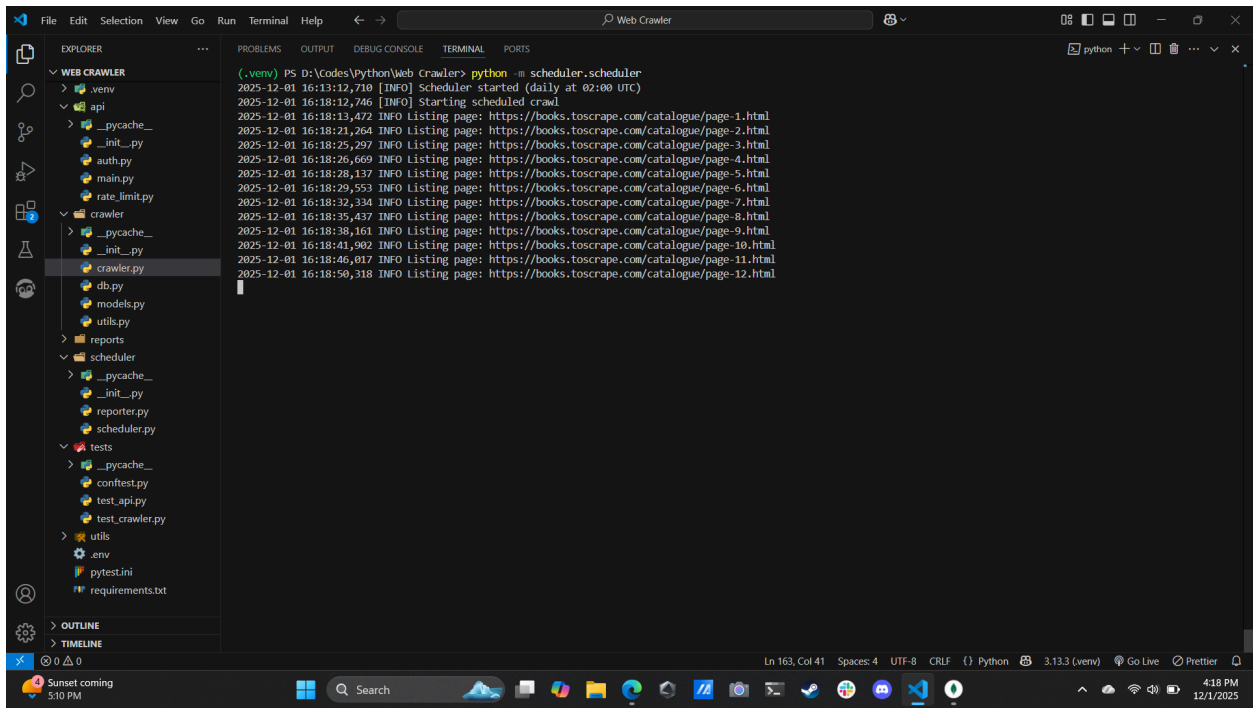
The screenshot shows the Visual Studio Code interface with the 'Web Crawler' project open. The Explorer pane on the left shows the project structure, including folders like 'api', 'crawler', 'scheduler', and 'tests'. The 'crawler' folder is expanded, showing files like 'crawler.py', 'db.py', 'models.py', 'utils.py', 'reports', 'reporter.py', and 'scheduler.py'. The 'crawler.py' file is selected. The Output pane on the right shows the terminal output of the crawler, displaying a list of books being inserted into MongoDB, such as 'of-mice-and-men_37', 'off-slides-off-1_38', 'myriad-pretor-1_36', 'paradise-lost-paradise-1_45', 'ms-marvel-vol-1-no-normal-ms-marvel-2014-2015-1_34', 'meditations_33', 'matilda_32', 'lost-among-the-living_31', 'old-records-never-die-one-mans-quest-for-his-vinyl-and-his-past_39', 'lord-of-the-flies_30', 'listen-to-me-fusion-1_29', 'kitchens-of-the-great-midwest_28', 'imperfect-harmony_26', 'icing-aces-hockey-2_25', 'hawkeye-vol-1-my-life-as-a-weapon-hawkeye-1_24', 'having-the-barbarians-baby-ice-planet-barbarians-75_23', 'giant-days-vol-1-giant-days-1-4_22', 'fruits-basket-vol-1-fruits-basket-1_21', 'jane-eyre_27', 'frankenstein_20', 'forever-rockers-the-rocker-12_19', 'fighting-fate-fighting-6_18', 'emma_17', 'eat-pray-love_16', 'deep-under-walker-security-1_15', 'charlie-and-the-chocolate-factory-charlie-bucket-1_13', 'bridget-jones-diary-bridget-jones-1_10', 'bright-lines_11', 'bounty-colorado-mountain-7_9', 'blood-defense-samantha-brinkman-1_8', 'bleach-vol-1-strawberry-and-the-soul-reapers-bleach-1_7', 'my-perfect-mistake-over-the-top-1_35', 'beyond-good-and-evil_6', 'choosing-our-religion-the-spiritual-lives-of-americas-nones_14', 'alice-in-wonderland-alices-adventures-in-wonderland-1_5', 'charitys-cross-charles-towne-belles-4_12', 'ajin-demi-human-volume-1-ajin-demi-human-1_4', 'a-spys-devotion-the-regency-spies-of-london-1_3', '1st-to-die-womens-murder-club-1_2', and '1000-places-to-see-before-you-die_1'. The status bar at the bottom indicates the file is at line 185, column 74, in the 'python' language, with a file encoding of 'UTF-8' and a line ending of 'CRLF'.

3) Crawler crawled the website and started to insert books into MongoDB.



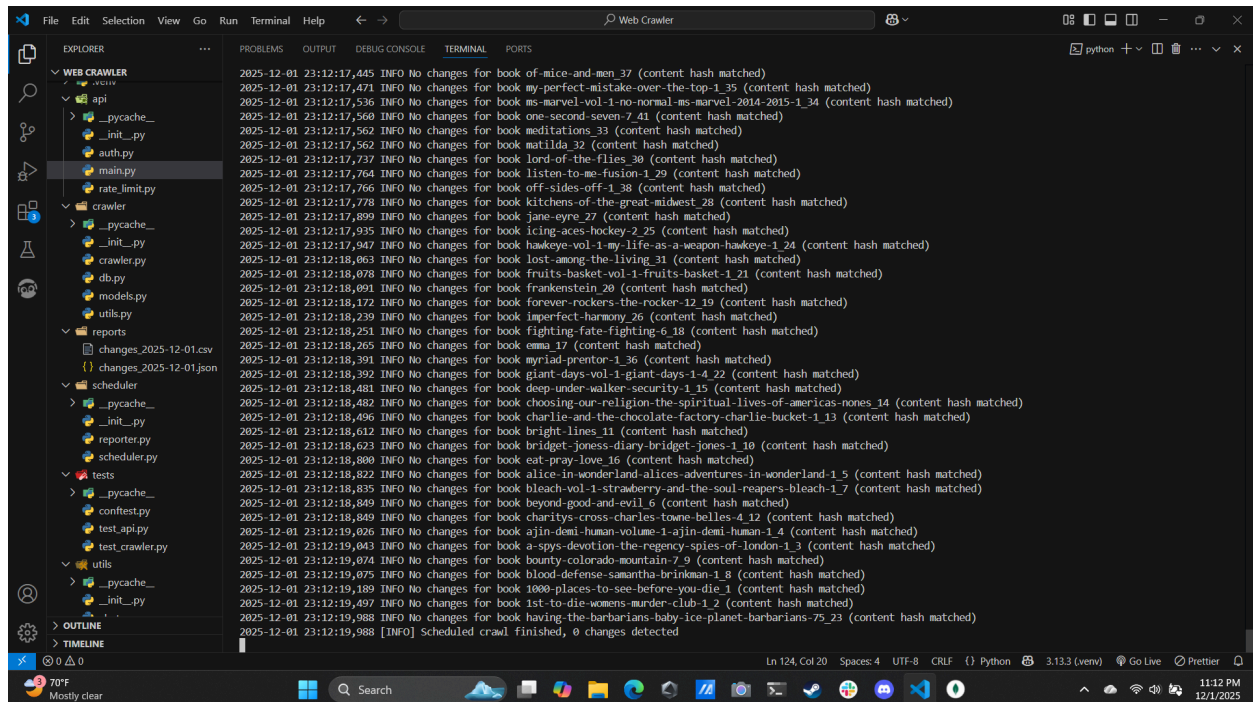
```
2025-12-01 13:29:28,312 INFO Listing page: https://books.toscrape.com/catalogue/page-22.html
2025-12-01 13:29:28,626 INFO Listing page: https://books.toscrape.com/catalogue/page-23.html
2025-12-01 13:29:29,269 INFO Listing page: https://books.toscrape.com/catalogue/page-24.html
2025-12-01 13:29:29,589 INFO Listing page: https://books.toscrape.com/catalogue/page-25.html
2025-12-01 13:29:29,916 INFO Listing page: https://books.toscrape.com/catalogue/page-26.html
2025-12-01 13:29:30,289 INFO Listing page: https://books.toscrape.com/catalogue/page-27.html
2025-12-01 13:29:30,893 INFO Listing page: https://books.toscrape.com/catalogue/page-28.html
2025-12-01 13:29:31,238 INFO Listing page: https://books.toscrape.com/catalogue/page-29.html
2025-12-01 13:29:31,564 INFO Listing page: https://books.toscrape.com/catalogue/page-30.html
2025-12-01 13:29:31,893 INFO Listing page: https://books.toscrape.com/catalogue/page-31.html
2025-12-01 13:29:32,219 INFO Listing page: https://books.toscrape.com/catalogue/page-32.html
2025-12-01 13:29:32,525 INFO Listing page: https://books.toscrape.com/catalogue/page-33.html
2025-12-01 13:29:32,842 INFO Listing page: https://books.toscrape.com/catalogue/page-34.html
2025-12-01 13:29:33,162 INFO Listing page: https://books.toscrape.com/catalogue/page-35.html
2025-12-01 13:29:33,494 INFO Listing page: https://books.toscrape.com/catalogue/page-36.html
2025-12-01 13:29:33,833 INFO Listing page: https://books.toscrape.com/catalogue/page-37.html
2025-12-01 13:29:34,149 INFO Listing page: https://books.toscrape.com/catalogue/page-38.html
2025-12-01 13:29:34,458 INFO Listing page: https://books.toscrape.com/catalogue/page-39.html
2025-12-01 13:29:34,768 INFO Listing page: https://books.toscrape.com/catalogue/page-40.html
2025-12-01 13:29:35,076 INFO Listing page: https://books.toscrape.com/catalogue/page-41.html
2025-12-01 13:29:35,384 INFO Listing page: https://books.toscrape.com/catalogue/page-42.html
2025-12-01 13:29:35,692 INFO Listing page: https://books.toscrape.com/catalogue/page-43.html
2025-12-01 13:29:36,013 INFO Listing page: https://books.toscrape.com/catalogue/page-44.html
2025-12-01 13:29:36,343 INFO Listing page: https://books.toscrape.com/catalogue/page-45.html
2025-12-01 13:29:36,663 INFO Listing page: https://books.toscrape.com/catalogue/page-46.html
2025-12-01 13:29:37,289 INFO Listing page: https://books.toscrape.com/catalogue/page-47.html
2025-12-01 13:29:37,891 INFO Listing page: https://books.toscrape.com/catalogue/page-48.html
2025-12-01 13:29:38,486 INFO Listing page: https://books.toscrape.com/catalogue/page-49.html
2025-12-01 13:29:39,078 INFO Listing page: https://books.toscrape.com/catalogue/page-50.html
2025-12-01 13:29:39,668 INFO Found 1000 book links
2025-12-01 13:29:39,976 INFO Inserted new book a-light-in-the-attic_1000
2025-12-01 13:29:40,282 INFO Inserted new book starving-hearts-triangular-trade-trilogy_1_990
2025-12-01 13:29:40,583 INFO Inserted new book Shakespeares-sonnets_989
2025-12-01 13:29:40,845 INFO Inserted new book the-requirem-red_995
2025-12-01 13:29:40,846 INFO Inserted new book the-black-maria_991
2025-12-01 13:29:40,910 INFO Inserted new book the-boys-in-the-boat-nine-americans-and-their-epic-quest-for-gold-at-the-1936-berlin-olympics_992
2025-12-01 13:29:40,912 INFO Inserted new book the-coming-woman-a-novel-based-on-the-life-of-the-infamous-feminist-victoria-woodhull_993
2025-12-01 13:29:40,913 INFO Inserted new book sapiens-a-brief-history-of-humankind_996
2025-12-01 13:29:40,913 INFO Inserted new book soumission_998
2025-12-01 13:29:40,915 INFO Inserted new book the-velvet_999
2025-12-01 13:29:40,916 INFO Inserted new book set-me-free_988
```

4) Scheduler starting to crawl.

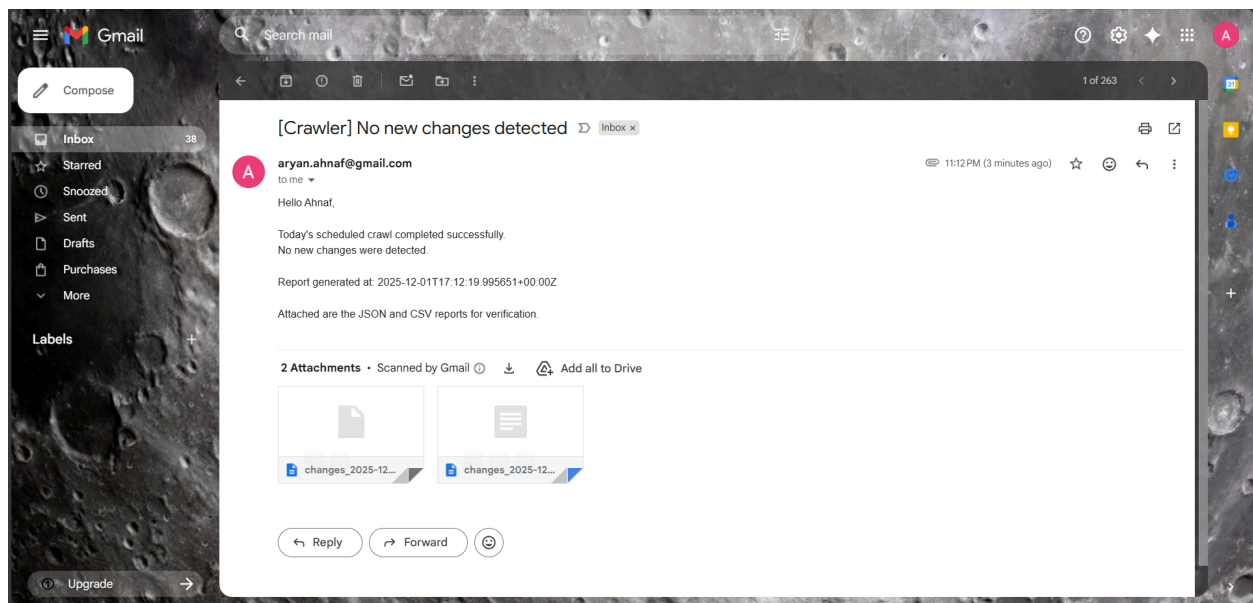


```
(.venv) PS D:\Codes\Python\Web Crawler> python -m scheduler.scheduler
2025-12-01 16:13:12,710 [INFO] Scheduler started (daily at 02:00 UTC)
2025-12-01 16:18:12,746 [INFO] Starting scheduled crawl
2025-12-01 16:18:13,472 INFO Listing page: https://books.toscrape.com/catalogue/page-1.html
2025-12-01 16:18:21,264 INFO Listing page: https://books.toscrape.com/catalogue/page-2.html
2025-12-01 16:18:25,297 INFO Listing page: https://books.toscrape.com/catalogue/page-3.html
2025-12-01 16:18:26,669 INFO Listing page: https://books.toscrape.com/catalogue/page-4.html
2025-12-01 16:18:29,137 INFO Listing page: https://books.toscrape.com/catalogue/page-5.html
2025-12-01 16:18:29,553 INFO Listing page: https://books.toscrape.com/catalogue/page-6.html
2025-12-01 16:18:32,334 INFO Listing page: https://books.toscrape.com/catalogue/page-7.html
2025-12-01 16:18:35,437 INFO Listing page: https://books.toscrape.com/catalogue/page-8.html
2025-12-01 16:18:38,161 INFO Listing page: https://books.toscrape.com/catalogue/page-9.html
2025-12-01 16:18:41,902 INFO Listing page: https://books.toscrape.com/catalogue/page-10.html
2025-12-01 16:18:46,017 INFO Listing page: https://books.toscrape.com/catalogue/page-11.html
2025-12-01 16:18:50,318 INFO Listing page: https://books.toscrape.com/catalogue/page-12.html
```

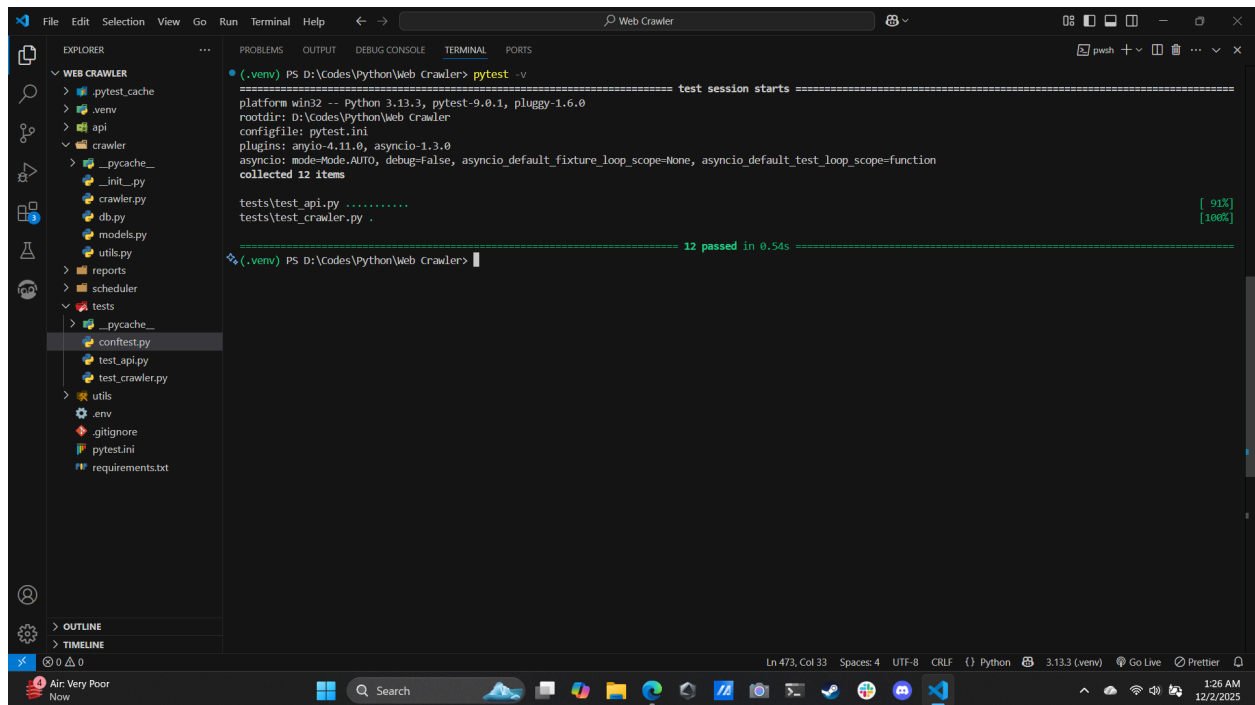
5) The scheduler finished crawling and comparing the hashes of each book to detect any changes and then generated reports and sent an email alert.



```
2025-12-01 23:12:17,445 INFO No changes for book of-mice-and-men 37 (content hash matched)
2025-12-01 23:12:17,471 INFO No changes for book my-perfect-mistake-over-the-top-1 35 (content hash matched)
2025-12-01 23:12:17,536 INFO No changes for book ms-marvel-vol-1-no-normal-ms-marvel-2014-2015-1 34 (content hash matched)
2025-12-01 23:12:17,560 INFO No changes for book one-second-seven-7 41 (content hash matched)
2025-12-01 23:12:17,562 INFO No changes for book meditations 33 (content hash matched)
2025-12-01 23:12:17,562 INFO No changes for book matilda 32 (content hash matched)
2025-12-01 23:12:17,737 INFO No changes for book lord-of-the-flies 30 (content hash matched)
2025-12-01 23:12:17,764 INFO No changes for book listen-to-me-fusion-1 29 (content hash matched)
2025-12-01 23:12:17,766 INFO No changes for book off-sides-off-1 38 (content hash matched)
2025-12-01 23:12:17,778 INFO No changes for book kitchens-of-the-great-midwest 28 (content hash matched)
2025-12-01 23:12:17,899 INFO No changes for book jane-eyre 27 (content hash matched)
2025-12-01 23:12:17,935 INFO No changes for book icing-aces-hockey-2 25 (content hash matched)
2025-12-01 23:12:17,947 INFO No changes for book hawkeye-vol-1-my-life-as-a-weapon-hawkeye-1 24 (content hash matched)
2025-12-01 23:12:18,063 INFO No changes for book lost-among-the-living 31 (content hash matched)
2025-12-01 23:12:18,078 INFO No changes for book fruits-basket-vol-1-fruits-basket-1 21 (content hash matched)
2025-12-01 23:12:18,091 INFO No changes for book frankenstein 29 (content hash matched)
2025-12-01 23:12:18,172 INFO No changes for book forever-rockers-the-rocker-12 19 (content hash matched)
2025-12-01 23:12:18,239 INFO No changes for book imperfect-harmony 26 (content hash matched)
2025-12-01 23:12:18,251 INFO No changes for book fighting-fate-fighting-6 18 (content hash matched)
2025-12-01 23:12:18,265 INFO No changes for book emma 17 (content hash matched)
2025-12-01 23:12:18,391 INFO No changes for book myriad-pretor-1 36 (content hash matched)
2025-12-01 23:12:18,392 INFO No changes for book giant-days-vol-1-giant-days-1-4 22 (content hash matched)
2025-12-01 23:12:18,481 INFO No changes for book deep-under-walker-security-1 15 (content hash matched)
2025-12-01 23:12:18,496 INFO No changes for book choosing-our-religion-the-spiritual-lives-of-americas-nones 14 (content hash matched)
2025-12-01 23:12:18,612 INFO No changes for book charlie-and-the-chocolate-factory-charlie-bucket-1 13 (content hash matched)
2025-12-01 23:12:18,623 INFO No changes for book bright-lines 11 (content hash matched)
2025-12-01 23:12:18,623 INFO No changes for book bridget-joneses-diary-bridget-jones-1 10 (content hash matched)
2025-12-01 23:12:18,800 INFO No changes for book eat-pray-love 16 (content hash matched)
2025-12-01 23:12:18,822 INFO No changes for book alice-in-wonderland-alices-adventures-in-wonderland-1 5 (content hash matched)
2025-12-01 23:12:18,835 INFO No changes for book bleach-vol-1-strawberry-and-the-soul-reapers-bleach-1 7 (content hash matched)
2025-12-01 23:12:18,840 INFO No changes for book beyond-good-and-evil 6 (content hash matched)
2025-12-01 23:12:18,840 INFO No changes for book charitys-cross-charles-tone-bellies-4 12 (content hash matched)
2025-12-01 23:12:19,026 INFO No changes for book ajin-demi-human-volume-1-ajin-demi-human-1 4 (content hash matched)
2025-12-01 23:12:19,043 INFO No changes for book a-spys-devotion-the-regency-spies-of-london-1 3 (content hash matched)
2025-12-01 23:12:19,074 INFO No changes for book bounty-colorado-mountain-7 9 (content hash matched)
2025-12-01 23:12:19,075 INFO No changes for book blood-defense-samantha-brinkman-1 8 (content hash matched)
2025-12-01 23:12:19,189 INFO No changes for book 1000-places-to-see-before-you-die 1 (content hash matched)
2025-12-01 23:12:19,497 INFO No changes for book 1st-to-die-womens-murder-club-1 2 (content hash matched)
2025-12-01 23:12:19,988 INFO No changes for book having-the-barbarians-baby-ice-planet-barbarians-75 23 (content hash matched)
2025-12-01 23:12:19,988 [INFO] Scheduled crawl finished, 0 changes detected
```



6) All 12 tests passed.



The screenshot shows a Visual Studio Code editor with a terminal window open. The terminal displays the output of a `pytest -v` command executed in a virtual environment. The output indicates that 12 tests were collected and all 12 passed successfully in 0.54 seconds. The test session details include the platform (win32), Python version (3.13.3), pytest version (9.0.1), and the root directory (D:\Codes\Python\Web Crawler). The tests passed are `tests\test_api.py` (91% coverage) and `tests\test_crawler.py` (100% coverage).

```
(.venv) PS D:\Codes\Python\Web Crawler> pytest -v
===== test session starts =====
platform win32 -- Python 3.13.3, pytest-9.0.1, pluggy-1.6.0
rootdir: D:\Codes\Python\Web Crawler
configfile: pytest.ini
plugins: anyio-4.11.0, asyncio-1.3.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 12 items

tests\test_api.py ..... [ 91%]
tests\test_crawler.py . [100%]

===== 12 passed in 0.54s =====
(.venv) PS D:\Codes\Python\Web Crawler>
```