Crawl Pipeline - Summary of Changes

crawl.py (pipeline layer)

Modified the crawl logic to properly follow links and recurse into sub-pages.

```
# crawl_pipeline/pipeline/crawl.py

# inside crawl()
try:
    links = link_extractor.extract(url, html)
    print(f"■ Extracted {len(links)} links from {url}")
except Exception as e:
    print(f"■■ Failed to extract links from {url}: {e}")
    links = []

for link in links:
    full_url = urljoin(url, link)   # normalize
    await crawl(
        full_url,
        depth + 1,
        max_depth,
        saved,
        limits,
        html_fetcher,
        fetch_and_save_html,
        link_extractor,
        req,
        visited,
    )
```

Reason: Why: Initially, only the root page was being saved. We added recursive crawling and logging of extracted links to make sure the crawler traverses through all reachable pages up to max_depth.

link_extractor.py (ports layer)

Improved link extraction logic to normalize and resolve relative & absolute links.

```
from html.parser import HTMLParser
from typing import List
from yarl import URL

class _HrefParser(HTMLParser):
    def __init__(self):
        super().__init__()
        self.links: list[str] = []
    def handle_starttag(self, tag, attrs):
        for (k,v) in attrs:
            if k in ("href","src") and v:
                self.links.append(v)

def extract(base_url: str, html: bytes) -> List[str]:
    p = _HrefParser()
    p.feed(html.decode("utf-8", errors="ignore"))
    base = URL(base_url)
    abs_links: list[str] = []
    for l in p.links:
        try:
            u = URL(l)
            if not u.is_absolute():
                u = base.join(u)   # relative → absolute
            abs_links.append(str(u))
        except Exception:
```

```
            continue
    return abs_links
```

Reason: Why: The old extractor returned 0 links because it wasn't normalizing relative URLs like /about. This fix ensures links are resolved against the base URL.

storage.py (ports layer)

Standardized how files are saved (save_bytes with explicit keyword args).

```
def save_bytes(*, content: bytes | str, url: str, canonical_type: str, root: str) -> str:
    if isinstance(content, str):
        content = content.encode("utf-8")
    sha = hashlib.sha256(content).hexdigest()
    path = _layout(root, url, sha, canonical_type)
    tmp = str(path) + ".tmp"
    with open(tmp, "wb") as f:
        f.write(content)
        f.flush()
        os.fsync(f.fileno())
    os.replace(tmp, path)
    return str(path.resolve())
```

Reason: Why: The crawler pipeline was crashing because we were passing save_bytes(path, html, "text/html") incorrectly. Now the interface is keyword-based, ensuring consistent file saving.

playwright_fetcher.py (new, ports layer)

New file added to handle JavaScript-heavy websites.

```
from playwright.async_api import async_playwright

class PlaywrightHtmlFetcher:
    async def fetch_html(self, url: str, timeouts: int = 30, retry: int = 2):
        for attempt in range(retry):
            try:
                async with async_playwright() as p:
                    browser = await p.chromium.launch(headless=True)
                    page = await browser.new_page()
                    await page.goto(url, timeout=timeouts * 1000)
                    html = await page.content()
                    await browser.close()
                    return "text/html", html.encode("utf-8")
            except Exception as e:
                if attempt == retry - 1:
                    raise e
```

Reason: Why: Some sites render content only after JavaScript executes (like SPAs). The default fetcher couldn't see links. This Playwright fetcher renders the full DOM before extraction.

cli.py (entrypoint layer)

Wired the new fetcher and storage methods into the pipeline.

```
from crawl_pipeline.ports.playwright_fetcher import PlaywrightHtmlFetcher
from crawl_pipeline.ports.storage import save_bytes

html_fetcher = PlaywrightHtmlFetcher()

async def fetch_and_save_html(url: str, html: bytes):
    return save_bytes(
        content=html,
        url=url,
        canonical_type="html",
        root="./downloads"
```

)

Reason: Why: The CLI was pointing at non-existent methods (storage.save_html). Now it calls save_bytes correctly and uses the new fetcher.