InkLink Current Architecture

This document outlines the main components of the InkLink project and how they interact to provide content-sharing capabilities for reMarkable tablets.

---

1. CLI and Entry Points

Location: `src/inklink/main.py`

Responsibilities:

- Defines the `inklink` CLI group with `server` and `auth` commands.
- `server` command starts the HTTP server for URL sharing.
- `auth` command launches the FastAPI-based reMarkable Cloud authentication UI.
- Central entry point for application setup and dispatch.

---

2. Server Implementations

Location: `src/inklink/server.py`

Key Classes / Functions:

- `URLHandler` (extends `BaseHTTPRequestHandler`):
  - Handles `/share` POST requests.
  - Validates and extracts URLs.
  - Delegates processing to PDF or Web scraping pipelines.
- `run_server()`:
  - Initializes and runs `HTTPServer` with `URLHandler`.

Interactions:

- Instantiates service layer classes using `CONFIG`.
- Routes requests through `QRCodeService`, `PDFService`, `WebScraperService`, `DocumentService`, and `RemarkableService`.

---

3. Service Layer

All services reside under `src/inklink/services/`.

3.1 QR Code Generation

`QRCodeService` (`qr_service.py`):

- Generates QR code images for input URLs.

3.2 PDF Handling

`PDFService` (`pdf_service.py`):

- Detects PDF URLs.
- Downloads and extracts metadata (title, path).

3.3 Web Scraping

`WebScraperService` (`web_scraper_service.py`):

- Fetches and parses web pages.
- Extracts structured content (headings, paragraphs, lists, images).

3.4 Google Docs Integration

`GoogleDocsService` (`google_docs_service.py`):

- Authenticates via OAuth2.
- Exports Docs as HTML.
- Parses structure and images.

3.5 Document Conversion

`DocumentService` (`document_service.py`):

- Creates HCL scripts (`drawj2d` input) from structured content or PDFs.
- Invokes `drawj2d` to generate `.rm` files.
- Manages temporary files and layout parameters.

3.6 reMarkable Upload

`RemarkableService` (`remarkable_service.py`):

- Uses `rmapi` executable to upload `.rm` files.
- Handles retries, renaming, and fallback methods.

---

4. Configuration

Location: `src/inklink/config.py`

- Central `CONFIG` dictionary (host, ports, paths, fonts, dimensions, retry settings).
- Directory setup (`TEMP_DIR`, `OUTPUT_DIR`).

- Logging setup via `setup_logging()`.

---

5. Dependencies

Location: `pyproject.toml`

- Core: Python 3.10, `requests`, `beautifulsoup4`, `PyPDF2`, `markdown`, `qrcode`, `Pillow`
- Optional: Google API clients, `readability-lxml`, `rmscene`, `fastapi`, `uvicorn`
- Dev: `pytest`, `black`, `flake8`

---

6. Testing

Location: `tests/`

- **Authentication:** `test_auth.py`
- **Server URL handling:** `test_server.py`
- **Service Layer:**
  - `test_qr_service.py`
  - `test_pdf_service.py`
  - `test_web_scraper_service.py`
  - `test_google_docs_service.py`
  - `test_document_service.py`
  - `test_remarkable_service.py`

Tests validate initialization, core methods, error paths, and end-to-end conversion workflows.

Component Interaction Flow

1. **User** invokes `inklink server` or browser extension →
2. **Server** receives URL via `/share` →
3. **QRCodeService** generates QR image →
4. **PDFService** or **WebScraperService** fetches content →
5. **DocumentService** builds HCL and calls `drawj2d` →
6. **RemarkableService** uploads `.rm` via `rmapi` →
7. **Server** returns JSON success/failure response.