Document

Document Generator

LangGraph-based document generator for converting multiple input formats (PDF, Markdown, TXT, web articles) into PDF and PPTX outputs using 100% Python implementation.

Table of Contents

Features

■ **Multiple Input Formats**:

- PDF documents (with OCR support via Docling)
- Markdown files (.md) with frontmatter support
- Plain text files (.txt)
- DOCX, PPTX, XLSX documents
- Web articles (URLs)
- Images (PNG, JPG, TIFF)

■ **Multiple Output Formats**:

- PDF (ReportLab with custom styling)
- PPTX (python-pptx for PowerPoint)

■ **Advanced Features**:

- Advanced PDF parsing with IBM's Docling (OCR, table extraction, layout analysis)
- Web content extraction with Microsoft's MarkItDown
- LangGraph workflow orchestration
- Automatic retry on generation errors (max 3 attempts)
- Comprehensive error handling and logging
- Docker containerization for portability

■ **Pure Python**:
- No Node.js dependencies
- Runs on Python 3.11+
- Fully containerized with Docker

Architecture

Hybrid Clean Architecture combining:
- **Domain Layer**: Pure business logic (models, enums, exceptions, interfaces)
- **Application Layer**: Use case orchestration (parsers, generators, LangGraph nodes)
- **Infrastructure Layer**: External integrations (Docling, MarkItDown, file I/O)

LangGraph Workflow:

```
detect_format → parse_content → transform_content → generate_output → validate_output
                                                                              ↓
                                                                    (retry on error, max 3x)
```

Tech Stack

Component	Technology	Version	Purpose
Document Parsing	Docling	2.66.0	Advanced PDF/DOCX/PPTX parsing with OCR
Document Conversion	MarkItDown	0.0.1a2	HTML/web articles to Markdown
PDF Generation	ReportLab	4.2.5	Professional PDF creation
PPTX Generation	python-pptx	1.0.2	PowerPoint presentations
Workflow Orchestration	LangGraph	0.2.55	State machine workflow
Validation	Pydantic	2.10.5	Data validation
Logging	Loguru	0.7.3	Structured logging

Package Manager	uv	latest	Fast Python package installation

Installation

Local Development

1. **Prerequisites**:

- Python 3.11+

- uv package manager ([install uv](https://github.com/astral-sh/uv))

2. **Install dependencies**:

```
    make setup-docgen
```

Or manually:

```
    uv pip install -e ".[dev]"
```

Docker (Recommended for Production)

1. **Build Docker image**:

```
    make docker-build
```

Or manually:

```
    docker build -t doc-generator:latest .
```

Usage

Command Line (Local)

Basic usage:

```
  python scripts/run_generator.py <input> --output <format>
```

Examples:

```
# Markdown to PDF
python scripts/run_generator.py src/data/article.md --output pdf

# Web article to PPTX
python scripts/run_generator.py https://example.com/article --output pptx

# PDF to PPTX (extract and convert)
python scripts/run_generator.py src/data/document.pdf --output pptx

# With verbose logging
python scripts/run_generator.py input.md --output pdf --verbose

# With log file
python scripts/run_generator.py input.md --output pdf --log-file output.log
```

Using Makefile:

```
# Convert markdown to PDF
make run-docgen INPUT=src/data/article.md OUTPUT=pdf

# Convert URL to PPTX
make run-docgen INPUT=https://example.com/article OUTPUT=pptx
```

Docker Usage

Direct Docker run:

```
# Markdown to PDF
docker run --rm \
  -v $(pwd)/src/data:/app/src/data \
  -v $(pwd)/src/output:/app/src/output \
  doc-generator:latest src/data/article.md --output pdf

# Web article to PPTX (no input mount needed)
docker run --rm \
  -v $(pwd)/src/output:/app/src/output \
  doc-generator:latest https://example.com/article --output pptx
```

Using Makefile:

```
make docker-run INPUT=src/data/article.md OUTPUT=pdf
```

Using Docker Compose:

1. Edit `docker-compose.yaml` to set the command:

```
    command: ["src/data/sample.md", "--output", "pdf"]
```

2. Run:

```
   make docker-compose-up
   # or
   docker-compose up
```

Python API

```
from doc_generator.application.graph_workflow import run_workflow

# Run workflow
result = run_workflow(
    input_path="src/data/article.md",
    output_format="pdf"
)

# Check results
if result["errors"]:
    print(f"Errors: {result['errors']}")
else:
    print(f"Generated: {result['output_path']}")
```

Docker Deployment

Building for Production

```
# Build image
docker build -t doc-generator:latest .

# Tag for registry
docker tag doc-generator:latest your-registry/doc-generator:v1.0.0

# Push to registry
docker push your-registry/doc-generator:v1.0.0
```

Running in Production

```
# Run with volume mounts
docker run -d \
  --name doc-generator \
  -v /path/to/data:/app/src/data \
  -v /path/to/output:/app/src/output \
  -e LOG_LEVEL=INFO \
  doc-generator:latest src/data/input.md --output pdf
```

Development

Setup Development Environment

```
# Install dependencies with dev extras
make setup-docgen

# Or manually
uv pip install -e ".[dev]"
```

Running Tests

```
# Run all tests with coverage
make test-docgen

# Or manually
pytest tests/ -v --cov=src/doc_generator --cov-report=term-missing
```

Linting and Type Checking

```
# Lint and type check
make lint-docgen

# Or manually
ruff check src/doc_generator
mypy src/doc_generator
```

Cleaning Generated Files

```
# Clean output and cache files
make clean-docgen
```

Project Structure

```
src/doc_generator/
■■■ domain/                          # Core business logic (zero dependencies)
■   ■■■ models.py                    # Pydantic models (WorkflowState, Config)
■   ■■■ content_types.py             # Enums (ContentFormat, OutputFormat)
■   ■■■ exceptions.py                # Custom exceptions
■   ■■■ interfaces.py                # Protocols (ContentParser, OutputGenerator)
■
■■■ application/                     # Use case orchestration
■   ■■■ graph_workflow.py            # LangGraph state machine
■   ■■■ parsers/                     # Input parsers
■   ■   ■■■ unified_parser.py        # Docling-based parser (PDF, DOCX, PPTX)
■   ■   ■■■ markdown_parser.py       # Markdown with frontmatter support
■   ■   ■■■ web_parser.py            # MarkItDown-based web parser
■   ■■■ generators/
■   ■   ■■■ pdf_generator.py         # ReportLab PDF generation
■   ■   ■■■ pptx_generator.py        # python-pptx PPTX generation
■   ■■■ nodes/                       # LangGraph nodes
■       ■■■ detect_format.py         # Format detection
■       ■■■ parse_content.py         # Content parsing
■       ■■■ transform_content.py     # Content transformation
■       ■■■ generate_output.py       # Document generation
■       ■■■ validate_output.py       # Output validation
■
■■■ infrastructure/                  # External integrations
■   ■■■ docling_adapter.py           # Docling wrapper
■   ■■■ markitdown_adapter.py        # MarkItDown wrapper
■   ■■■ file_system.py               # File I/O operations
■   ■■■ pdf_utils.py                 # ReportLab utilities
■   ■■■ pptx_utils.py                # python-pptx utilities
■   ■■■ logging_config.py            # Loguru configuration
■
■■■ utils/                           # Shared utilities

scripts/
■■■ run_generator.py                 # CLI entry point

tests/                               # Test suite
■■■ test_parsers.py
■■■ test_generators.py
■■■ test_workflow.py

config/
■■■ settings.yaml                    # Configuration

Dockerfile                           # Docker image definition
docker-compose.yaml                  # Docker Compose configuration
pyproject.toml                       # Python dependencies
Makefile                             # Automation tasks
```

Testing

Unit Tests

Test individual components:

```
pytest tests/test_parsers.py -v
pytest tests/test_generators.py -v
```

Integration Tests

Test end-to-end workflows:

```
pytest tests/test_workflow.py -v
```

Manual Testing

```
# Test markdown to PDF
make run-docgen INPUT=README.md OUTPUT=pdf

# Check output
ls -lh src/output/*.pdf
```

Configuration

Configuration is managed through `config/settings.yaml`:

```
generator:
  input_dir: "src/data"
  output_dir: "src/output"
  default_output_format: "pdf"
  max_retries: 3

logging:
  level: "INFO"

pdf:
  page_size: "letter"
  margin:
    top: 72
    bottom: 18
    left: 72
    right: 72

pptx:
  layout: "LAYOUT_16x9"
  slide_width: 960
  slide_height: 540
```

Troubleshooting

Common Issues

ImportError: Docling not available:

```
# Install Docling explicitly
uv pip install docling==2.66.0
```

ImportError: MarkItDown not available:

```
# Install MarkItDown with all extras
uv pip install "markitdown[all]==0.0.1a2"
```

Docker build fails:

```
# Rebuild without cache
docker build --no-cache -t doc-generator:latest .
```

Permission denied on output directory:

```
# Fix permissions
chmod 755 src/output
```

Contributing

1. Follow the clean architecture pattern
2. Add type hints to all functions
3. Write comprehensive docstrings
4. Add unit tests for new features
5. Update README with new capabilities

License

MIT License - See LICENSE file for details

Acknowledgments

- **Docling** by IBM Research - Advanced document parsing
- **MarkItDown** by Microsoft - Document-to-markdown conversion
- **ReportLab** - Professional PDF generation
- **python-pptx** - PowerPoint presentations
- **LangGraph** - Workflow orchestration