PDF to Image Conversion Application Documentation

Table of Contents

1. System Overview

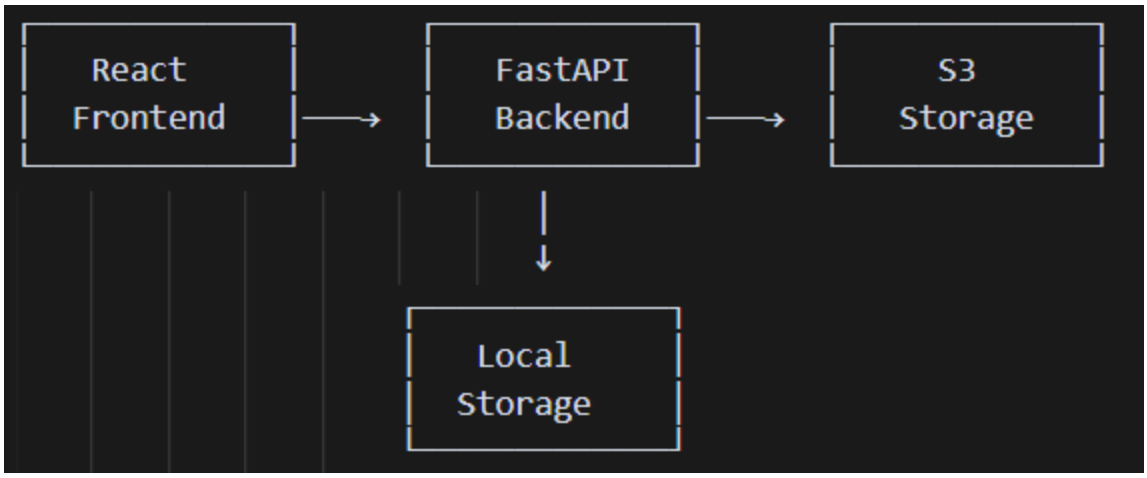
This application provides a seamless workflow for converting PDF files to images while maintaining folder structures. The system includes:

- A React-based web interface for uploading PDF files
- A FastAPI backend for processing and converting PDFs to images
- Automatic S3 cloud storage integration
- Health monitoring and status updates
- Background processing for handling large batches

Key features include:
- Support for folder uploads with nested directory structures
- Real-time conversion status updates
- Automatic uploading to S3 with organized date-based paths
- Cleanup of local files after successful processing

2. Architecture

The application follows a client-server architecture:



- Frontend: React application providing the user interface
- Backend: FastAPI server handling file processing, PDF conversion, and S3 uploads
- Local Storage: Temporary storage for uploaded PDFs and converted images
- S3 Storage: Permanent cloud storage with an organized directory structure

3. Installation and Setup

Prerequisites
- Python 3.7+
- Node.js and npm
- AWS account with S3 bucket
- Poppler (for PDF conversion)

Backend Setup

1. Install Python dependencies:

```
pip install -r requirements.txt
```

2. Install Poppler:
- Windows: Download from http://blog.alivate.com.au/poppler-windows/ and add to PATH
- macOS: `brew install poppler`
- Linux: `apt-get install poppler-utils`

3. Create a `.env` file in the API directory:

```
Upload_folder_path=./uploads
Images_folder_path=./images
AWS_ACCESS_KEY=your_access_key
AWS_SECRET_KEY=your_secret_key
S3_BUCKET_NAME=your_bucket_name
```

4. Start the FastAPI server:

```
terminal@terminal-temple ~ $ cd API
uvicorn main:app --reload
```

Frontend Setup

1. Install Node.js dependencies:

```
terminal@terminal-temple ~ $ cd frontend_app
npm install
```

2. Start the React development server:

```
terminal@terminal-temple ~ $ npm start
```

4. Frontend Components

The frontend is a React application with the following key components:

- **Server Status Indicator:** Shows the backend server status (online/offline)
- **File Upload Section:** Allows selection of files or folders
- **File List Display:** Shows selected files with paths and sizes
- **Progress Indicators:**
  - Upload progress bar
  - Conversion status updates
  - S3 upload status (internal only)
- **Error/Success Messages:** User feedback for operations

The frontend communicates with the backend through API calls using Axios.

5. Backend API

The FastAPI backend provides several endpoints:

Health Check
- GET `/health`: Verifies the server is running

File Upload
- POST `/upload-files/`: Accepts PDF files and saves them to the local uploads directory

PDF Conversion
- POST `/convert-pdfs/`: Initiates PDF to image conversion for uploaded files
  - Takes an optional `folder_path` parameter to specify a subdirectory
  - Returns a `task_id` for status tracking

Status Endpoints
- GET `/conversion-status/{task_id}`: Reports conversion progress
- GET `/s3-upload-status/{task_id}`: Reports S3 upload progress

Background Processing
The backend uses FastAPI's `BackgroundTasks` to handle processing without blocking:
1. PDF file conversion using pdf2image
2. S3 upload of converted images
3. Local directory cleanup

6. Application Workflow

1. PDF Upload:
- User selects PDF files or folders via the frontend
- Files are uploaded to the backend and stored in the uploads directory
- Backend preserves the original folder structure

2. Automatic Conversion:
- Backend initiates PDF to image conversion immediately after upload
- Each PDF is processed to extract pages as JPG images
- Each file gets a unique GUID to prevent naming conflicts
- Images are stored in a structured format: `**images/GUID/filename/page_X.jpg**`

3. **S3 Upload**:
- Converted images are automatically uploaded to S3
- Files are organized by date: `**year/month/YYMMDD/GUID/filename/page_X.jpg**`
- Upload progress is tracked internally

4. **Cleanup**:
- After successful S3 upload, local files are removed
- Both uploads and images directories are cleaned

7. S3 Storage Format

Files are stored in S3 using the following path structure

test/YYYY/MMM/YYMMDD/GUID/filename/page_X.jpg

Where:
- `test/`: A constant prefix for all uploads
- `YYYY`: 4-digit year (e.g., 2023)
- `MMM`: 3-letter month abbreviation (e.g., Jan, Feb)
- `YYMMDD`: 2-digit year, month, day (e.g., 230523)
- `GUID`: Unique identifier for each PDF file
- `filename`: Original PDF filename (without extension)
- `page_X.jpg`: Individual page images (starting from page_1.jpg

This structure ensures organized storage with date-based prefixes and unique identifiers.

8. Configuration Options

The application can be configured through environment variables:
Backend Variables
- `Upload_folder_path`: Directory for PDF file storage (default: "./uploads")
- `Images_folder_path`: Directory for converted images (default: "./images")
- `AWS_ACCESS_KEY`: AWS access key for S3 authentication
- `AWS_SECRET_KEY`: AWS secret key for S3 authentication
- `S3_BUCKET_NAME`: Target S3 bucket name

PDF Conversion Options
- Currently set to 300 DPI for high-quality image conversion
- JPEG format for all converted images

9. Troubleshooting

Common Issues

Upload Failures
- Verify the server is running (`/health` endpoint)
- Check file size limits (browser or server restrictions)
- Ensure proper folder permissions for uploads directory

Conversion Errors
- Check Poppler installation and accessibility
- Verify PDF files are valid and not corrupted
- Check server logs for specific error messages

S3 Upload Issues
- Verify AWS credentials are correct and have S3 permissions
- Check S3 bucket exists and is accessible
- Confirm network connectivity to AWS services

Logging

The application uses Python's logging module with INFO level:
- All operations are logged with timestamps
- File paths and conversion details are recorded
- S3 upload results are tracked
- Errors include detailed exception information

Server logs provide the best source of troubleshooting information in case of issues.

This documentation provides an overview of the PDF-to-image conversion application. For further assistance or feature requests, please contact the development team.