PDFy Documentation

Overview

This project is a web application built using Flask that allows users to upload DOCX, TXT, DOC, and PDF files, convert them to PDF format, and download the converted file. The application also displays metadata for the uploaded files and the converted PDF files.

Features

- File Upload: Users can upload files with supported formats (DOCX, TXT, DOC, and PDF).
- File Conversion: The uploaded files are converted to PDF using LibreOffice in headless mode.
- Metadata Display: After conversion, the application displays metadata for the uploaded file (name, size, upload date).
- File Download: Users can download the converted PDF files.
- Flash Messages: The app uses flash messages to inform the user about errors or success.
- Timeout: The file gets deleted after a timeout period to maintain privacy and avoid stacking of files.

Technologies Used

- Flask: A Python web framework for building the web application.
- LibreOffice: A free and open-source office suite used to perform the conversion from DOCX (and other formats) to PDF.
- Werkzeug: A library used by Flask to handle file security and utilities.
- Subprocess: A module used to call LibreOffice for file conversion in headless mode.
- HTML/CSS: Used for the basic layout and design of the application, with some custom styling.

Project Structure

```
file-conversion-app/
│
├── app.py              # Main Flask application file
├── requirements.txt    # Python dependencies for the project
├── Dockerfile          # Docker configuration for containerizing the app
├── uploads/            # Folder to store uploaded files
├── converted/          # Folder to store converted PDF files
├── templates/          # Folder for HTML templates
│   ├── upload.html     # File upload page template
```

```
│   └── result.html       # Conversion result page template
└── static/               # Folder for static files (CSS, JS, etc.)
    └── css/
        └── style.css     # Stylesheet for the application
```

Installation and Setup

1. Clone the Repository:
   Clone the project from your source (GitHub, etc.):
   ```bash
   git clone <repository-url>
   cd file-conversion-app
   ```

2. Install Dependencies:
   First, install the necessary Python packages. If you haven't already, create a virtual environment and activate it:
   ```bash
   python -m venv venv
   source venv/bin/activate  # On Windows, use 'venv\Scripts\activate'
   ```
   Then, install the dependencies from `requirements.txt`:
   ```bash
   pip install -r requirements.txt
   ```

3. Install LibreOffice:
   - Windows: Download and install LibreOffice from the official website.
   - Linux (for Docker): The Dockerfile installs LibreOffice using `apt-get`.

4. Run the Flask Application:
   To start the development server:
   ```bash
   python app.py
   ```
   The application will run at `http://127.0.0.1:5000`.

Docker Setup

To run the application in a Docker container:

1. Build the Docker Image:

From the project root, build the Docker image:
```bash
docker build -t flask-libreoffice .
```

2. Run the Docker Container:
After building the image, run the container:
```bash
docker run -p 5000:5000 flask-libreoffice
```

The app will be available at `http://localhost:5000`.

How It Works

1. **File Upload**:
- The user uploads a DOCX, DOC, TXT, or PDF file using the web form on the **home page**.
- The file is saved to the server in the `uploads` directory.

2. **File Conversion**:
- The file is passed to the `convert_to_pdf()` function, which uses LibreOffice in headless mode to convert the file into a PDF.
- The converted PDF is saved to the `converted` folder.

3. **Metadata Display**:
- After conversion, the metadata of the uploaded file is displayed (file name, size, upload date).
- A download link for the converted PDF is provided.

4. **Download the Converted File**:
- The user can download the converted PDF by clicking the **Download** button.
- The application serves the file using Flask's `send_file()` method.

Endpoints

- `/`: The home page where users can upload files.
  - Method: GET and POST
  - Description: Displays the upload form. Handles the file upload and starts the conversion process.

- `/result`: The result page showing the metadata and download link for the converted file.
  - Method: GET

- Description: Displays the metadata and provides the user with a link to download the converted PDF file.

- `/download/<filename>`: Endpoint for downloading the converted PDF file.
  - Method: GET
  - Description: Serves the converted PDF file for download.

Error Handling

- If the user uploads a file with an unsupported format, an error message is flashed, and the user is prompted to upload a valid file type.
- If the file conversion fails, an error message is displayed, and the user can try uploading a new file.