**Document Manager Application Documentation**

**1. Project Overview**

**1.1 Introduction**

The Document Manager application is a simple web-based system designed to allow users to upload, store, and view documents securely from any device. It supports document upload, storage, and retrieval through an intuitive web interface built using Python Flask. The project includes containerization with Docker to enable easy deployment and portability.

**1.2 Technologies Used**

* **Python:** Core programming language.
* **Flask:** Lightweight web framework for building the application.
* **HTML/CSS:** For rendering the web pages.
* **Docker:** For containerizing the application to simplify deployment.
* **PyInstaller:** To convert the Flask app into a standalone executable.
* **OS:** To handle file paths and directories.
* **Datetime:** (optional) for timestamping or future enhancements.

**2. Features**

**2.1 Document Upload**

* Users can upload documents via a simple web form.
* Uploaded files are stored in a designated folder on the server.

**2.2 Document Viewing**

* Users can view and download uploaded documents from the web interface.
* The app lists all uploaded files dynamically.

**2.3 Cross-Device Accessibility**

* The app can be accessed from any device with a web browser.
* Docker containerization allows easy deployment across platforms.
* Standalone executable (.exe) can be created for offline usage on Windows.

**3. File Structure**

DocumentManager/

├── app.py # Main Flask application script

├── Dockerfile # Docker configuration file

├── requirements.txt # Python dependencies

├── templates/

│ └── index.html # HTML template for the main page

├── uploads/ # Folder where uploaded documents are stored

├── dist/ # Folder created after PyInstaller build (for .exe)

└── README.md # Project documentation

**4. How to Run the Application**

**4.1 Requirements**

* Python 3.7+
* Required Python packages:
  + Flask

Install packages via pip:

pip install Flask

**4.2 Running Locally with Python**

1. Clone or download the project folder.
2. Navigate to the project directory.
3. Run the Flask app:

python app.py

1. Open your browser and visit http://localhost:5000.

**4.3 Running with Docker**

1. Ensure Docker is installed and running.
2. Build the Docker image:

docker build -t docmanager .

1. Run the container:

docker run -d -p 5000:5000 -v /absolute/path/to/uploads:/app/uploads --name docmanager\_container docmanager

1. Access the app via http://localhost:5000 from your browser.

**4.4 Creating Standalone Executable (.exe)**

1. Install PyInstaller:

pip install pyinstaller

1. Build executable:

pyinstaller --onefile --add-data "templates;templates" app.py

1. Run the .exe file in the dist/ folder.
2. Open your browser and go to http://localhost:5000 manually.

**5. Modules and Functionality**

**5.1 app.py**

* Flask app managing routes for uploading and serving documents.
* Stores uploaded files in /uploads folder.
* Renders the list of uploaded files on the home page.

**5.2 Dockerfile**

* Defines the container environment using a Python slim image.
* Copies source code and installs dependencies.
* Sets up working directory and exposes port 5000.
* Runs the Flask app inside the container.

**5.3 Templates (index.html)**

* Provides the web UI with a form for file upload.
* Displays a list of uploaded documents with download links.

**6. Error Handling**

**6.1 File Upload Validation**

* Checks if a file is selected before uploading.
* Handles file saving errors gracefully.

**6.2 Docker Errors**

* Provides guidance for port conflicts or volume mount issues.

**6.3 Executable Issues**

* Explains the necessity to manually open the browser after running the .exe.

**7. Docker Containerization**

**7.1 Why Docker?**

* Encapsulates the app environment for consistency across platforms.
* Simplifies deployment without worrying about local dependencies.
* Enables easy scaling and portability.

**7.2 Dockerfile Summary**

* Base image: python:3.11-slim
* Copies project files, installs dependencies, sets working directory.
* Exposes port 5000 and runs the Flask app command.

**7.3 Build and Run Commands**

docker build -t docmanager .

docker run -d -p 5000:5000 -v /path/to/uploads:/app/uploads --name docmanager\_container docmanager

**8. Future Enhancements**

* Add user authentication and document privacy controls.
* Support for multiple document formats with preview.
* Mobile-friendly UI design.
* Automated backup and synchronization of uploaded files.

**9. Contributing**

* Contributions are welcome via pull requests or issue reports.
* Follow PEP 8 style guidelines for Python code.