Code Documentation

# Description

The provided code is a Streamlit application that allows users to upload their code files, configure various settings, and generate documentation using the MLflow API. It provides an intuitive user interface for users to interact with the application and customize the generated documentation.

# Table of Contents

1. Introduction  
2. Requirements  
3. Usage Instructions  
4. Code Comments  
5. Examples  
6. Configuration  
7. Troubleshoot  
8. Limitations  
9. Conclusions  
10. Usage Examples

## 1. Introduction

This application uses the Streamlit framework to build a user-friendly interface for uploading code files and configuring various settings. It then uses the MLflow API to generate documentation based on the user-provided settings and uploaded code. The generated documentation can be downloaded in various formats (docx, pdf, txt, html, md).

## 2. Requirements

To run this code, you'll need the following libraries:  
\* `streamlit` for building the web-based user interface  
\* `requests` for making HTTP requests to the MLflow API  
\* `base64` for encoding and decoding binary data  
\* `json` for working with JSON data  
\* `os` for environment variable management  
\* `datetime` for handling dates and timestamps  
\* `pathlib` for working with file paths  
\* `mlflow` for interacting with the MLflow API

## 3. Usage Instructions

To use this application, follow these steps:  
1. Upload a code file (.py,.java,.js,.jsx) to the application.  
2. Configure the desired generation options (theme, output format, repository name, target folder path, personal access token, branch name, and commit message).  
3. Click the "Generate Documentation" button to start the generation process.  
4. Once the generation is complete, you can view the logs and download the generated documentation in the desired format.

## 4. Code Comments

The following sections of the code are worth highlighting:  
\* The `uri\_from` function, which encodes image assets into base64 strings.  
\* The `payload` dictionary, which contains the necessary parameters for the MLflow API request.  
\* The `requests.post` call, which sends the encoded payload to the MLflow API.

## 5. Examples

Here's an example of how the application might be used:  
\* User uploads a code file (`my\_code.py`) containing some sample Python code.  
\* User configures the generation options (theme, output format, repository name, target folder path, personal access token, branch name, and commit message).  
\* User clicks the "Generate Documentation" button.  
\* Application generates documentation and displays the logs.

## 6. Configuration

The following settings can be configured:  
\* Theme (Technical, Non-Technical, Tester)  
\* Output format (docx, pdf, txt, html, md)  
\* Repository name (owner/repo)  
\* Target folder path (relative path inside the repository)  
\* Branch name (main or another branch)  
\* Commit message (default: "Add generated documentation")  
\* Personal access token (for authentication)

## 7. Troubleshoot

Common issues and solutions:  
\* API request timeouts: Reduce input file size or use the Jupyter notebook interface instead.  
\* Authentication errors: Verify the personal access token and ensure it has the required scope.

## 8. Limitations

This application has the following limitations:  
\* Supports up to 1000 characters of code per file due to API timeout.  
\* Does not support large code files; consider using the Jupyter notebook interface for larger inputs.

## 9. Conclusions

This Streamlit application provides a convenient and user-friendly interface for generating documentation using the MLflow API. While there are some limitations, it offers a practical solution for users who need to generate documentation for their code projects.

## 10. Usage Examples

Practical use cases:  
\* Automating documentation generation for large-scale software development projects.  
\* Creating documentation for machine learning models using the MLflow API.  
\* Generating documentation for existing codebases to facilitate knowledge sharing and collaboration.