```
import inspect

import os

import sys

import threading


from dotenv import load_dotenv


from scripts.auto_tests_docs.docs import DOCUMENTATION_WRITER_SOP

from swarm_models import OpenAIChat


load_dotenv()


api_key = os.getenv("OPENAI_API_KEY")


model = OpenAIChat(

    model_name="gpt-4",

    openai_api_key=api_key,

    max_tokens=4000,

)



def process_documentation(item):

    """

     Process the documentation for a given function using OpenAI model and save it in a Markdown
file.

    """
    doc = inspect.getdoc(item)
    source = inspect.getsource(item)
    input_content = (
        f"Name: {item.__name__}\n\nDocumentation:\n{doc}\n\nSource"
        f" Code:\n{source}"
    )
    print(input_content)

    # Process with OpenAI model
    processed_content = model(
        DOCUMENTATION_WRITER_SOP(input_content, "swarms.utils")
    )

    doc_content = f"# {item.__name__}\n\n{processed_content}\n"

    # Create the directory if it doesn't exist
    dir_path = "docs/swarms/utils"
    os.makedirs(dir_path, exist_ok=True)

    # Write the processed documentation to a Markdown file
    file_path = os.path.join(dir_path, f"{item.__name__.lower()}.md")
    with open(file_path, "w") as file:
        file.write(doc_content)


def main():
    # Gathering all functions from the swarms.utils module
    functions = [
        obj
        for name, obj in inspect.getmembers(
            sys.modules["swarms.utils"]
        )
        if inspect.isfunction(obj)
    ]

    threads = []
    for func in functions:
        thread = threading.Thread(
            target=process_documentation, args=(func,)
        )
        threads.append(thread)
        thread.start()

    # Wait for all threads to complete
    for thread in threads:
        thread.join()

    print("Documentation generated in 'docs/swarms/utils' directory.")


if __name__ == "__main__":
    main()
```