```
import inspect
import os
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-4-1106-preview",
    openai_api_key=api_key,
    max_tokens=4000,
)


def process_documentation(
```

```
    item,
    module: str = "swarms.structs",
    docs_folder_path: str = "docs/swarms/structs",
):
    """
    Process the documentation for a given class or function using OpenAI model and save it in a Python file.
    """
    doc = inspect.getdoc(item)
    source = inspect.getsource(item)
    is_class = inspect.isclass(item)
    item_type = "Class Name" if is_class else "Name"
    input_content = (
        f"{item_type}:"
        f" {item.__name__}\n\nDocumentation:\n{doc}\n\nSource"
        f" Code:\n{source}"
    )

    # Process with OpenAI model
    processed_content = model(
        DOCUMENTATION_WRITER_SOP(input_content, module)
    )

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

    # Create the directory if it doesn't exist
```

```
    dir_path = docs_folder_path
    os.makedirs(dir_path, exist_ok=True)

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

    print(
        f"Processed documentation for {item.__name__}. at {file_path}"
    )


def main(module: str = "docs/swarms/structs"):
    items = []

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

    # Wait for all threads to complete
    for thread in threads:
```

```
        thread.join()

    print(f"Documentation generated in {module} directory.")


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