```
import os
from typing import Optional


import requests
from pydantic import BaseModel, Field



class SubmitPullRequestSchema(BaseModel):
    # repo_owner: str = Field(
    #     "kyegomez",
    #     example="kyegomez",
    #     description="The owner of the GitHub repository.",
    # )
    # repo_name: str = Field(
    #     "swarms",
    #     example="swarms",
    #     description="The name of the GitHub repository.",
    # )
    file_path: str = Field(
        ...,
        example="README.md",
        description="The path to the file within the repository.",
    )
    new_content: str = Field(
        ...,
        description="The new content to be written to the file.",
    example="New content for the file.",
)
commit_message: str = Field(
    ...,
    description="The commit message for the change.",
    example="Updated README.md content",
)
pr_title: str = Field(
    ...,
    description="The title of the pull request.",
    example="Update README.md",
)
pr_body: Optional[str] = Field(
    None,
    description="The body of the pull request.",
    example="This PR improves the README.md content.",
)

class Config:
    schema_extra = {
        "example": {
            # "repo_owner": "kyegomez",
            # "repo_name": "swarms",
            "file_path": "README.md",
            "new_content": "New content for the file.",
            "commit_message": "Updated README.md content",
                "pr_title": "Update README.md",
                "pr_body": "This PR improves the README.md content.",
            }
        }


class GetFileContentSchema(BaseModel):
    repo_owner: str = Field(
        ...,
        example="kyegomez",
        description="The owner of the GitHub repository.",
    )
    repo_name: str = Field(
        ...,
        example="swarms",
        description="The name of the GitHub repository.",
    )
    file_path: str = Field(
        ...,
        example="README.md",
        description="The path to the file within the repository.",
    )
    branch: str = Field(
        default="main",
        example="main",
        description="The branch name to fetch the file from.",
    )


    class Config:
        schema_extra = {
            "example": {
                "repo_owner": "kyegomez",
                "repo_name": "swarms",
                "file_path": "README.md",
                "branch": "main",
            }
        }



def get_github_file_content(
    file_path: str,
    repo_owner: str = "kyegomez",
    repo_name: str = "swarms",
    branch: str = "main",
) -> str:
    """
    Fetches the content of a file from a GitHub repository.


    Args:
        repo_owner (str): The owner of the repository (e.g., 'kyegomez').
        repo_name (str): The name of the repository (e.g., 'swarms').
        file_path (str): The path to the file within the repository.
        branch (str): The branch name (default is 'main').


    Returns:

        str: The content of the file as a string.


    Raises:

        requests.exceptions.RequestException: If there is an error with the request.

        ValueError: If the file content cannot be decoded.

    """

    url = f"https://raw.githubusercontent.com/{repo_owner}/{repo_name}/{branch}/{file_path}"

    try:

        response = requests.get(url)

        response.raise_for_status()

        return response.text

    except requests.exceptions.RequestException as e:

        print(f"Error: {e}")

        raise

    except ValueError as e:

        print(f"Error decoding file content: {e}")

        raise



# out = get_github_file_content("README.md")

# print(out)

def submit_pull_request(

    file_path: str,
    new_content: str,
    commit_message: str,
    pr_title: str,
    pr_body: Optional[str] = None,
    repo_owner: str = "kyegomez",
    repo_name: str = "swarms",
) -> None:
    """
    Submits a pull request to a GitHub repository by modifying a specified file.

    Args:
        token (str): GitHub personal access token.
        repo_owner (str): The owner of the repository (e.g., 'kyegomez').
        repo_name (str): The name of the repository (e.g., 'swarms').
        file_path (str): The path to the file within the repository.
        new_content (str): The new content to write to the file.
        commit_message (str): The commit message for the change.
        pr_title (str): The title of the pull request.
        pr_body (Optional[str]): The body of the pull request (default is None).

    Raises:
        Exception: If any error occurs during the process.
    """
    try:
        from github import Github
```

```
token = os.getenv("GITHUB_TOKEN")
g = Github(token)
repo = g.get_repo(f"{repo_owner}/{repo_name}")

# Get the file
contents = repo.get_contents(file_path)
current_branch = repo.get_branch("main")

# Create a new branch
new_branch = "modify_" + file_path.replace("/", "_").replace(
    ".", "_"
)
repo.create_git_ref(
    ref=f"refs/heads/{new_branch}",
    sha=current_branch.commit.sha,
)

# Update the file
repo.update_file(
    contents.path,
    commit_message,
    new_content,
    contents.sha,
    branch=new_branch,
)
```

```
    # Create a pull request
    repo.create_pull(
        title=pr_title, body=pr_body, head=new_branch, base="main"
    )
    print("Pull request created successfully.")
except Exception as e:
    print(f"Error: {e}")
    raise
```