F-19 - Prompt Files in GitHub Copilot

SUMMARY

In this lesson, we explored the **Prompt Files** feature in GitHub Copilot. Prompt Files allow us to create and manage reusable prompts that can be shared across projects, enhancing productivity by reducing repetitive task specifications. These prompts are simply markdown files and can be easily transferred between projects.

To activate the feature:

1. Ensure the feature is enabled (as it may be a preview feature). Open **VS Code settings** by pressing `Ctrl + Shift + P`, type in "workspace settings," and open `settings.json`.
2. Add `"chat.promptFiles": true` to ensure the functionality is active.
3. Create a `.github/prompts` folder in your project, if not auto-created.
4. To create a new prompt, use `Ctrl + Shift + P`, type "prompt," and select "chat create prompt," then specify your prompt details.

By adding prompts, we can streamline processes like server creation, code reviews, unit tests, and more. For instance, we can create a prompt to automatically set up a Python server using Flask with specific features like logging and authentication.

WHAT WE LEARNED

- Enabling and configuring prompt files in VS Code.
- Creating and managing project-specific prompts.
- Utilizing prompts for consistent code practices and task automation.
- Sharing prompt files across projects for team efficiency.

HOW WE CAN APPLY IT

- **Server Setup**: Automate the creation of backend server structures with predefined requirements.
- **Code Review**: Standardize code reviews with prompts enforcing code conventions.
- **Unit Testing**: Generate structured unit tests covering essential cases automatically.
- **Security Checks**: Automate security checks and prompt reviews based on compliance requirements.

TIPS AND TRICKS

- **Organization**: Keep your prompts well-organized with descriptive names for easy retrieval across projects.
- **Sharing**: Share prompt files with your team for consistency in coding practices.
- **Customization**: Tailor prompts to handle edge cases that are frequent in your specific development tasks.
- **Review Prompts**: Regularly review and update prompts to keep them aligned with evolving code standards.

EXAMPLES

Example 1: Creating a Simple Server Prompt

```
# Simple Server
Create a simple Python server using the Flask library. The server should:
- Contain a "Hello World" GET request
- Contain a POST request
- Include logging
- Implement user authentication with an authorization bearer token
```

Example 2: Code Review Prompt

```
# Follow Conventions
Review all Python files and ensure they meet the following conventions:
- All variables must be capitalized.
- All functions must start with "func".
- Use Django for servers, not Flask.
```

Example 3: Unit Test Prompt

```
# Unit Tests
Create unit tests for each function. Ensure:
- Tests cover edge cases like null inputs.
- Use PyTest module.
- The test names start with "test_case".
```

By employing these strategies, we can significantly streamline our development workflow, ensuring consistent and high-quality outcomes across all projects.