F-12 - AI-Assisted Code Reviews and Commit-Message Generation

SUMMARY

In this lesson, we explored how GitHub Copilot can assist us through a comprehensive code change workflow. This covers everything from reviewing code to generating commit messages. Here's a detailed step-by-step outline of what we covered:

1. **Set Up a GitHub Repository**: To get started, ensure you have a GitHub repository linked to your Visual Studio Code (VS Code). If you're unfamiliar with how to set up a GitHub repository, there are numerous guides available online.

2. **Create a Virtual Environment**: Use the command `python -m venv .env` in your terminal to create a virtual environment. Activate it with `source .env/bin/activate` (or `Scripts\activate` on Windows).

3. **Install Prerequisites**: Ensure all necessary dependencies are installed by running `pip install -r requirements.txt .`

4. **Run the Application**: Launch your Flask app using `python app.py` and visit it in your browser to verify functionality.

5. **AI-Assisted Code Review**: GitHub Copilot can perform code reviews by highlighting potential issues and offering comments, akin to a senior developer's feedback. Select the code and ask Copilot for a review.

6. **Code Modification with Copilot**: Use Copilot Chat to modify the code, such as changing the returned crypto prices from three to ten. This showcases how Copilot can directly edit code to improve functionality.

7. **Staging and Reviewing Changes**: After making changes, stage them using the source control feature in VS Code. Copilot can review only the staged changes and provide comments for corrections.

8. **Commit Message Generation**: Instead of writing commit messages manually, we can leverage Copilot to generate descriptive commit messages based on the changes made.

WHAT WE LEARNED

- How to set up a GitHub repository linked to VS Code.

- Creating and activating a Python virtual environment.
- Installing dependencies with `requirements.txt`.
- Running a Flask application.
- Performing AI-assisted code reviews using GitHub Copilot.
- Modifying code with Copilot Chat.
- Staging changes in VS Code.
- Generating commit messages using GitHub Copilot.

HOW WE CAN APPLY IT

- **Team Collaborations**: Streamline the code review process by offering AI-generated insights.
- **Productivity Enhancement**: Save time on routine tasks like writing commit messages or identifying code errors.
- **Code Consistency**: Maintain consistent code quality with AI-assistance, especially useful in large projects.
- **New Developers**: Lower the barrier for new team members to engage with code reviews and understand coding standards.

TIPS AND TRICKS

- **Customize Commit Messages**: You can set specific formatting for your commit messages to fit team standards.
- **Refine Copilot Suggestions**: Always review and refine Copilot's suggestions, as it's not infallible.
- **Use as a Learning Tool**: Treat Copilot's insights as guidance, especially for junior developers. Verify any suggestions it provides.
- **Continuous Feedback**: Always provide feedback on Copilot's suggestions to improve its performance.

EXAMPLES

Here's a basic Python example illustrating the usage of GitHub Copilot for generating a commit message:

```
# Original function fetching three prices
def fetch_crypto_prices():
    return api.get_crypto_prices(['BTC', 'ETH', 'LTC'])

# Modify to fetch top 10 prices using Copilot
def fetch_crypto_prices():
    return api.get_crypto_prices(['BTC', 'ETH', 'LTC', 'XRP', 'BCH', 'ADA', 'LINK', 'DOT', 'BNB',
```

```
# Review changes using Copilot
# Stage changes and generate commit message
```

This session demonstrates how GitHub Copilot becomes an invaluable assistant in our development workflow, automating repetitive tasks, and aiding in maintainable practices.