CRÉDIT AGRICOLE
CORPORATE & INVESTMENT BANK

Good practices using GitHub Copilot

MARCH 2025

Coding good practices with GitHub Copilot

Development processes should be coupled with coding best practices to maximize the benefits from GitHub Copilot, and mitigate associated risks.

CUSTOMIZE SUGGESTIONS

- Provide GitHub Copilot with context **keeping important files (ex: documentation, playbooks) opened in the IDE** to add context to the prompt for accurate suggestions
- Define **custom instruction files** (template of guidelines) to provide more business context and guide GitHub Copilot's suggestions .These instructions should be defined by the Software Factory in link with the tech lead on each application and shared with the team. This functionality is available in Vs Code, Visual Studio, and Neovim
- **Make the code as clean as possible**, since the quality of Copilot outputs heavily depend on the quality of existing code
    - Use precise and domain-specific variable names, define unambiguous method names, use naming conventions
    - Use methodologies like Test Driven-Development (TDD) & Domain-Driven Design (DDD) to provide GitHub Copilot with a stronger business context, improving its ability to interpret code, suggest unit tests, and propose enhancements
- Use **prompt engineering** by giving clear and precise directives on the action to be performed or the code to be generated and providing example of inputs and outputs to orient GitHub Copilot solution
- Write **clear comments and document the code** (it is not always self-standing), especially for classes/functions/ methods. Classes/function headers should be mandatory and include input & output parameters as GitHub Copilot will rely on these code comments to be more efficient.
- Adhere to the **Principles of the Software Craftsmanship Manifesto**

UNDERSTAND, REVIEW & TEST THE GENERATED CODE

- Analyze suggestions can expose you to new methods, functions, and development best practices. **Take time to understand, review and verify** that **GitHub Copilots' suggestions** are aligned with your teams' coding standards and conventions. Whenever you accept a suggestion from GitHub Copilot, you will hold the final responsibility for the result
- Always **test the generated code** by performing unit and functional tests to ensure reliability and performance

MAINTAIN SECURITY

- Adopt a **security-first mindset** by reviewing Copilot's code carefully to maintain security and detect potential vulnerabilities introduced by the generated code & comments
- **Check the licenses** of the suggested libraries
- Always **apply best security practices** (input validation, error handling, etc.).
- The use of GitHub Copilot is authorized until C3 source code only *(this authorization has been subjected to a NSU validation)*. **Do not use GitHub Copilot on C4 Code** (ex: strategic code, algo-trading, information related to the bank information system) to protect sensitive codebases
- For **complex or high-risk tasks**, prefer writing the code manually

<u>The developer is responsible for the acceptation of the suggestions from GitHub Copilot.</u>

Do not hesitate to share these good practices using GitHub Copilot with your teammates. Collaboration and exchanging experiences can improve its use across the team!