AI Coding Tools: A Best Practices Guide

This presentation will provide best practices for using AI tools effectively. We'll cover mindsets, capabilities, limitations, prompting, validation, and ethical usage. Learn how to integrate AI tools into your development workflow.

- Learn the strengths of your coding assistant (e.g., Cursor AI, Windsurf, Qodo, GitHub Copilot, etc.).
- Recognize that AI-generated code may contain errors and should always be reviewed.
- Use AI suggestions as a supplement, not a replacement, for fundamental coding knowledge.
- Adopt a slow transition from Manual to AI

Mindset Before Approaching AI

Curiosity

Approach AI with a curious mindset. Be open to new possibilities and experimentation.

Realism

Understand AI's capabilities and limitations. Don't expect it to solve every problem.

Ethics

Consider the ethical implications of using AI. Ensure responsible and fair usage.

Think of It as a Helper, Not a Replacement

AI can assist with coding, debugging, and explaining concepts, but it's not perfect. Always verify its output.

Break Down Your Problem Clearly

The better you describe your issue, the better response you'll get. Provide enough details like the programming language, error messages, or expected output.

Expect Multiple Iterations

You may not get the perfect answer on the first try. Ask follow-up questions and refine your prompts. Need patients If the first response isn't useful, rephrase your request, add more context, or ask in smaller steps.

Use It to Learn, Not Just to Copy-Paste

Try to understand the logic behind the code AI provides, so you can adapt it for different situations.

Review & Verify Everything

AI-generated code might contain errors or security risks. Always test before using it in production.

Set a Context About Your Need

Try to set detailed context about your requirement using Ai, GPT, Gemini, other tools and set it as context for the AI tool

General AI Practices

1 **Define Goals**

Clearly define what you want to achieve with AI.

2 **Data Quality**

Ensure your data is accurate and reliable.

3 **Understand the AI's Limitations**

AI tools are helpful but not foolproof; always review their output

4 **Keep Learning**

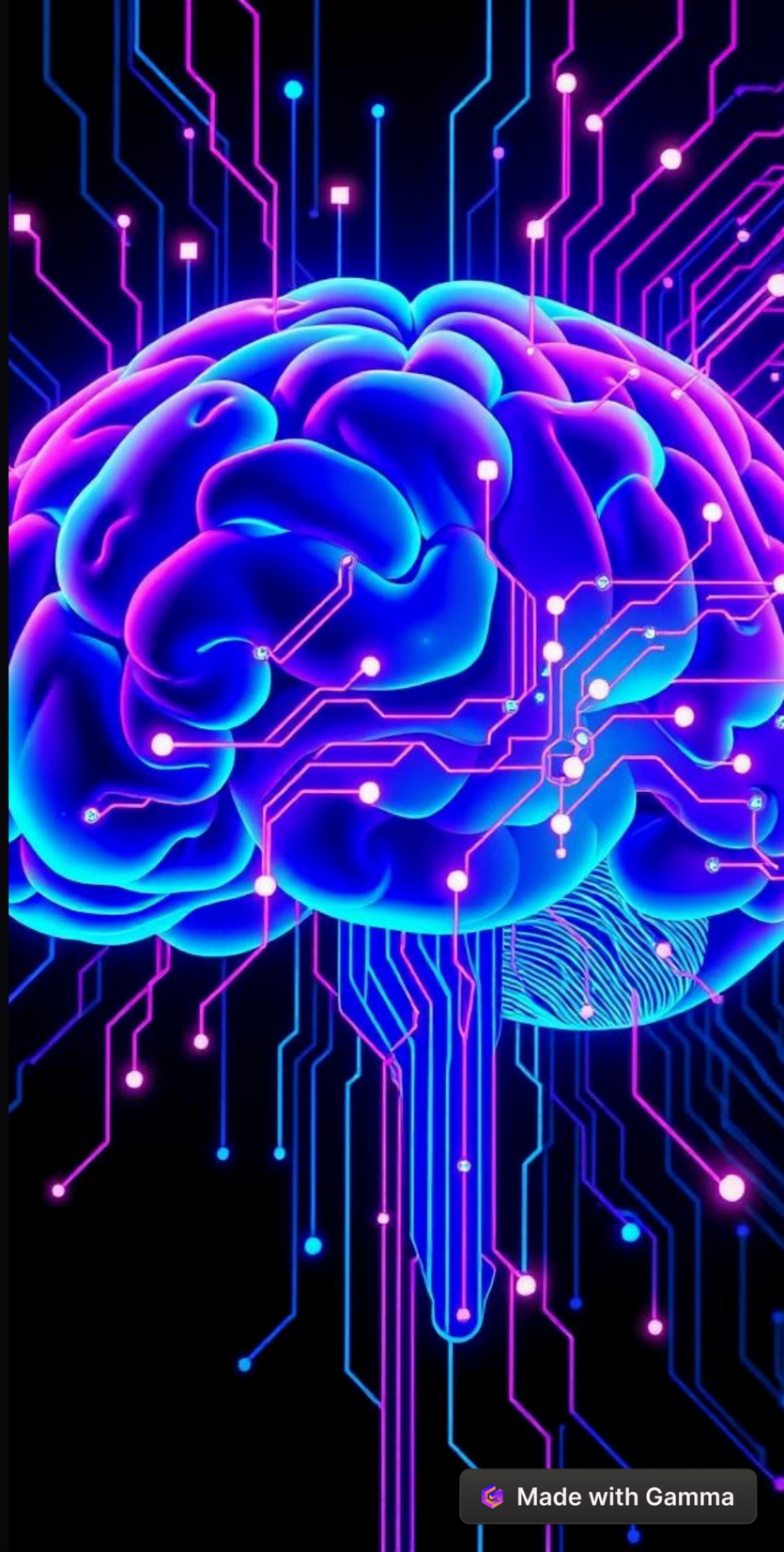
Use AI assistance as a learning tool rather than a crutch.

5 **Review & Verify Thoroughly**

Run unit tests and integration tests to ensure AI-generated code works as expected.

6 **Do not push a multiple requests in one step**

Keep as question as part by part set the context, so the accuracy would high. Do not apply large change in single step



General AI Practices

1 Don't apply for large suggestion in optimization

Do not apply large code optimization suggestion in single iteration, it might arise error.

2 Balance AI and Manual Coding

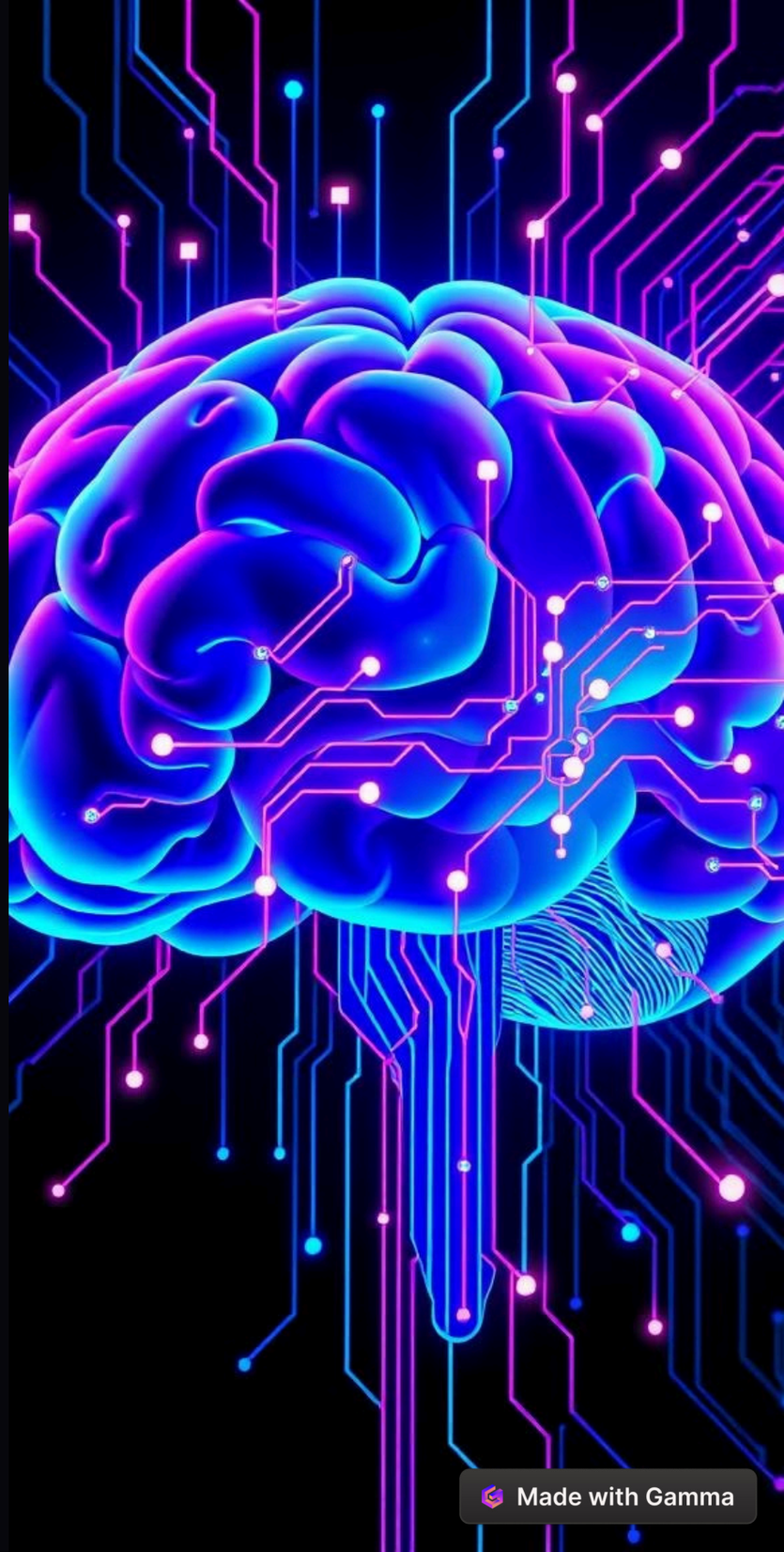
Use AI to assist but continue developing problem-solving skills manually.

3 Ensuring Ethical and Secure AI Usage

Avoid inputting sensitive or proprietary data into AI tools. Be mindful of AI's training data sources and potential biases. Respect licensing and open-source policies when using AI-generated code.

4 Continuous Improvement and Adaptation

Stay updated with new features and improvements in AI coding assistants.



Capabilities of AI as Code Assistant

Code Completion

AI can auto-complete functions, methods, and syntax to speed up coding.

Bug Detection

AI can identify syntax errors, runtime errors, and potential security risks.

Code Optimization

AI suggests performance improvements, better algorithms, and efficient structures.

Automated Documentation

AI can generate comments, docstrings, and documentation from code.

Code Refactoring

AI suggests restructuring and optimizing existing code for better readability and efficiency.

Multi-Language Support

AI assists in coding across multiple programming languages.

Debugging Assistance

AI helps in identifying, understanding, and fixing bugs in the code.

Learning and Guidance

AI provides explanations, examples, and alternative approaches to solve coding challenges.

Writing Unit Test

AI will help us to write unit test to improve the coverage of different scenarios and reduce the bug scenarios.

Challenges in Using AI Coding Tools

Accuracy Issues

AI-generated code may contain logical errors or security vulnerabilities. To improve accuracy

Context Awareness

AI may struggle to understand project-specific in terms of clarity in context given

Over-Reliance

Developers might become overly dependent on AI, reducing their problem-solving skills.

Ethical Concerns & Integration Challenges

Potential copyright and security issues when using AI-generated code. AI tools may not always align with team workflows or coding standards.

The Importance of Prompting

1

Clear Instructions

Be clear and specific with prompts to get the best suggestions.

2

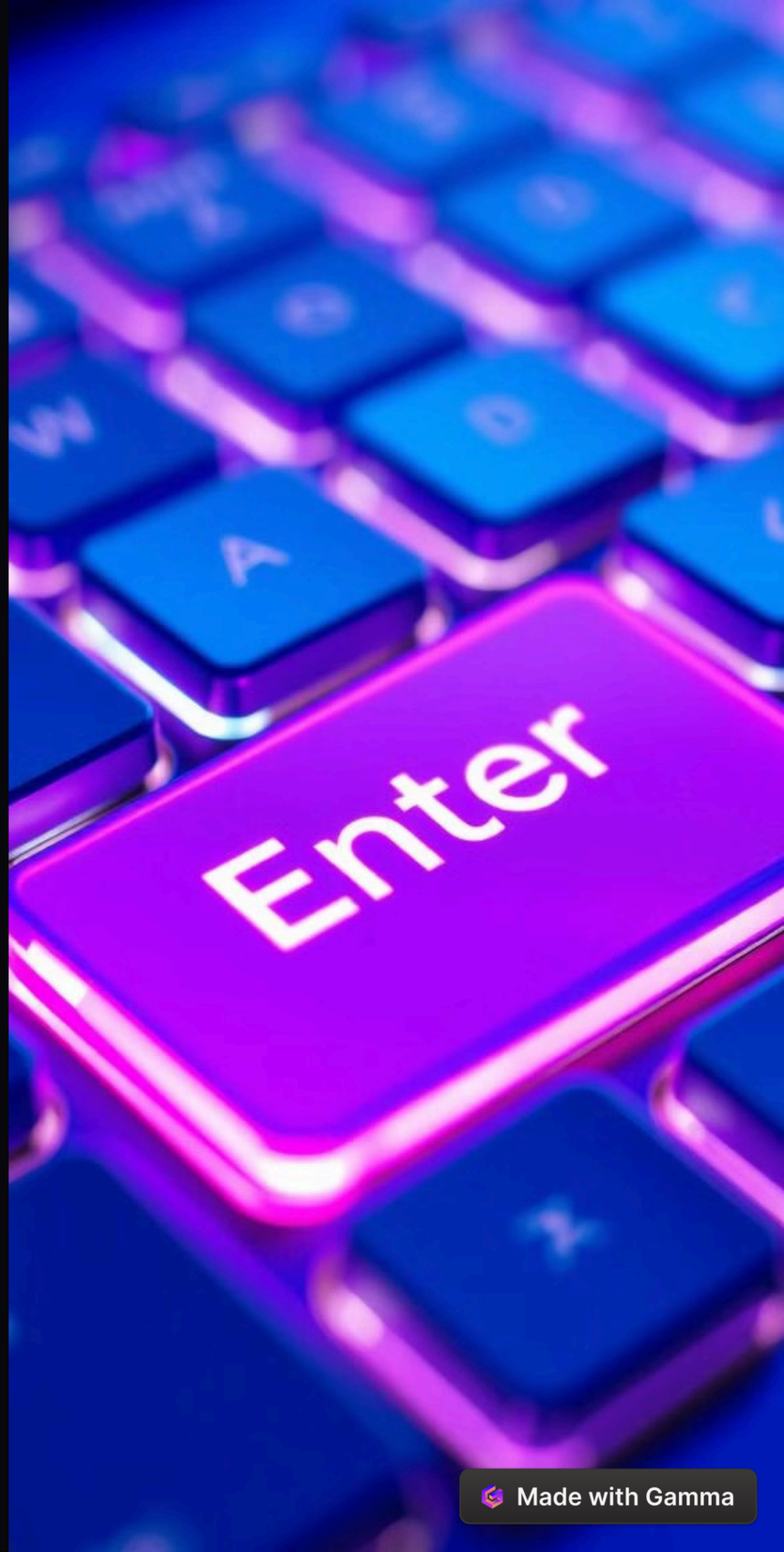
Iterate Prompts

Refine prompts based on AI responses. Use step-by-step queries when working on complex problems

3

Context

Provide sufficient context, such as code snippets, expected behavior, and constraints.



Ethical and Secure AI Usage

Data Privacy

Protect sensitive data used by AI. Avoid inputting sensitive or proprietary data into AI tools.

Transparency

Be transparent about AI's role in decision-making.

Accountability

Take responsibility for AI's actions and outcomes.

Respect Policy & License

Respect licensing and open-source policies when using AI-generated code.

Best Practices for Cursor AI

- Set the context before starting any activities through Cursor IDE - **https://www.youtube.com/watch?v=WZ8g6deOyAk**
- For better and more accurate codebase answers, you can index your codebase. **https://docs.cursor.com/context/codebase-indexing**
- Use `.cursorignore` to let each developer configure which folders and paths they work on in the monorepo
- Add `.cursorignore` to your global `.gitignore`
- Use Symbols to communicate with LLM in better way to refer the files, contents etc. **https://docs.cursor.com/context/@-symbols/overview**
- Do not initiate optimization or bulk bug files in one stretch, it will make complexities, so try to do one by one or one context at a time
- Use it to auto-complete complex functions and detect potential errors early.
- Integrate it with your IDE to streamline workflow and improve efficiency.
- Cross-check Cursor AI-generated suggestions with official documentation and industry best practices.
- Always use the latest stable Cursor AI model
- Do not use Cursors prompt chaining or continuous prompting unnecessarily.
- Split code into clean, modular functions for optimal suggestions.
- Use file-aware context only when major changes are needed.
- Disable auto-refactor unless reviewed manually.
- Use precise, minimal prompts (e.g., refactor this function to use async/await").
- Leverage Cursor Notepad for adding reference and setting context. Notepads are powerful context-sharing tools in Cursor that bridge the gap between composers and chat interactions. **https://docs.cursor.com/beta/notepads**

Best Practices for WindSurf

- Recommednding to Pin relevant contaxt before starting it **https://docs.codeium.com/context-awareness/overview#context-pinning**
- Pinning class/struct definition files that are inside your repo but in a module separate from your currently active file.
- Do not initiate optimization or bulk bug files in one stretch, it will make complexities, so try to do one by one or one context at a time.
- If you highlight a block of code before invoking Command, it will edit the selection. Otherwise, it will do a pure generation.
- For better and more accurate codebase answers, you can index your codebase. Indexing already there as default - **https://docs.codeium.com/context-awareness/local-indexing**
- Use `.codeiumignore` to remove unnecessary refernces and file from code base - **https://docs.codeium.com/context-awareness/local-indexing#codeiumignore**
- Take advantage of WindSurfs contextual understanding to improve web development workflows.
- Set Windsurf rules to explicitly define your own rules for cascade to follow. Cascade will be aware of your rules at all times and it will act like what you defined in the rule file.

Keeping AI as a Companion

To successfully integrate AI, consider how the tools are developed and maintained. AI should be a partner to improve your work and to make your life easier. AI must work to support and supplement your decision making, not replace it.

To effectively utilize AI coding tools, consider these points:

- **Continuous Learning:** Stay updated with AI advancements and coding practices.
- **Balanced Approach:** Combine AI assistance with human expertise and problem-solving.
- **Critical Thinking:** Evaluate AI-generated suggestions and code to ensure accuracy and alignment with project goals.
- **Community Engagement:** Share experiences, insights, and best practices with other developers using AI tools.
- Leverage new tool and features coming into the market and start thinking in GenAI way.

References

- https://www.youtube.com/watch?v=WZ8g6deOyAk
- Vibe Coding - https://www.youtube.com/watch?v=YWwS911iLhg
- Cursor Tips - https://dev.to/heymarkkop/cursor-tips-10f8
- https://www.youtube.com/watch?v=Rgz6mX93C4Y
- https://docs.cursor.com/get-started/welcome
- Forum - https://forum.cursor.com/t/best-practices-for-medium-large-projects/21206
- https://docs.codeium.com/
- https://www.youtube.com/watch?v=Bk6WL8E5Hr8 - Windsurf Top Tips
- Best Practices for Building a SaaS with Windsurf and Makerkit - https://makerkit.dev/blog/tutorials/build-saas-windsurf

We are looking into more tool like
Stay tune...