1. First think through the problem, read the codebase for relevant files, and write a plan to tasks/todo.md. Always ask questions if anything is unclear, and NEVER ASSUME.
2. The plan should have a list of todo items that you can check off as you complete them
3. Before you begin working, check in with me and I will verify the plan.
4. Then, begin working on the todo items, marking them as complete as you go.
5. Please every step of the way just give me a high level explanation of what changes you made
6. Make every task and code change you do as simple as possible. We want to avoid making any massive or complex changes. Every change should impact as little code as possible. Everything is about simplicity.
7. Make sure security is tight and always production ready, no matter the cirumstance.
8. Also before coding, always have this perspective of "what would Mark Zuckerburg do in this situation"

9. After execution, please check through all the code you just wrote and make sure it follows security best practices. make sure there are no sensitive information in the front and and there are no vulnerabilities that can be exploited, and no crucial files like .env, and also before pushing to github check as well. And also please explain the functionality and code you just built out in detail. Walk me through what you changed and how it works. Act like you're a senior engineer teaching a 16 year old how to code.
8. Finally, add a review section to the todo.md file with a summary of the changes you made and any other relevant information.
9. And also, always check for syntax errors after code completion.
10. If you need me to clarify anything, or have questions, please feel free to ask, always ensure 100% crystal clarity before execution. Further, ensure that everything I tell you to do you know how to do it, if not please state. Please do not attempt to do something you do not have information about and assume. Ask me to do research if needed.

11. Upon finishing execution, update dev.md to include any information of functions/code that we need to remove before production.
12. Check if there is any legacy code, overlapping code, overlapping functions that could cause the error
13. Small, but crucial and important changes in steps.md so in case your memory was wiped, you can always reference it