Visibility of System Status:

The linter does a great job of providing a lot of information to the user both before and after execution. Once installed it pops up and shows you when each step is completed. When you execute your program it provides all of the information that you would need in order to find and eliminate any lingering bugs. The error messages provided are comprehensive and while this is perfect for a team working in production, it is not the most beginner friendly linter. That being said, it is a great source of learning because you can see exactly what went wrong and where it went wrong and then use that information in order to improve not just your current project, but your knowledge base for future projects.

Match Between System and the Real World:

I mentioned this in the previous passage, a lot of the feedback that you receive once you run the linter is pretty dense. This is not something that could be used by inexperienced programmers. However, the target audience for this project is professional teams who need to coordinate in order to create cohesive software and for that this linter is perfect. By using complex language and providing highly detailed feedback, the teams using the linter have the best shot of creating software that will meld together seamlessly when the time has come for it to be pushed to production.

User Control and Freedom:

This linter lacks a lot of user control and freedom which is the point. The purpose of using this linter is to limit the amount of variation in code that will ultimately be combined into the production ready project. By cutting back on user freedom, it means that the code will be more similar and there is less of a chance of any merge conflicts. Additionally, the strict rules that are used by the linter mean that less time will be spent on code review because everybody is using the same strict standards.

Consistency and Standards:

The standards for this are very consistent with those used by modern day professionals. If you read through the feedback that is provided all of the details align with modern code standards. This is essential because the whole point of the project is to make sure that the product that is being produced is high quality. Consistency is one of the most important components of the project as well.

Error Prevention:

There are two ways to look at this heuristic when analyzing the project, the first way would be the error messages that are provided when you run your project and there is some kind of error. The messages provided are thorough and provide important feedback that can then be used by the user in order to improve their project. I do not think that these error messages are what is being addressed with the heuristic though, I would like to focus more on the types of error messages that arise when the linter itself is not working. An example being during installation of the linter, if there is an error then the project provides you with an error message that lets you know what the issue is and provides some information that can be used to fix it. I think the information provided is enough to get by but could do well with being beefed up a bit in order to be more user friendly.

Recognition Rather than Recall:

This category does not really apply to the project as there is never a need to memorize any of the information that it provides. When you run the linter and it provides you with feedback it does show error codes but it also gives you the reason that the error was raised and the ways in which it can be fixed. At no point would an person using the linter need to have those codes memorized.

Flexibility and Efficiency of Use:

The linter is extremely efficient to use and once it is installed there is almost no additional maintenance required other than updates. The linter is meant to be installed and then seamlessly integrated into whatever project you are working on. I am not sure about the levels of flexibility with this project, there does not seem to be a lot of functionality outside of its main purpose but I think this is a good thing as the linter is not meant to be a super complicated piece of software that requires a significant depth of knowledge in order to be used.

Aesthetic and Minimalist Design:

This is a difficult category to use to judge the linter and I think would be more apt for when you analyze something like a web application. That being said, this linter is extremely minimalistic in that the only time it provides you with any on screen feedback it's just text. The linter is extremely minimalistic as the emphasis for the project is to streamline the code review and development processes which would be hindered by the inclusion of unnecessary features.

Help User Recognize, Diagnose and Recover From Errors:

Once again there are two ways in which I interpret this heuristic. The first being the feedback that the user receives when they run the linter. The error messages that crop up are comprehensive in the details that they provide. The second interpretation would be any feedback provided if the linter fails to run or there is an error during the installation process. I did not have issues with either of these so I cannot really speak to whether or not the feedback provided is helpful or not.

Help and Documentation:

I reviewed the documentation and the conclusion that I came to was that while helpful enough to answer most basic questions that would arise when installing and using the linter, it is pretty difficult to find really comprehensive documentation.