Table Of Major Errors Fixed
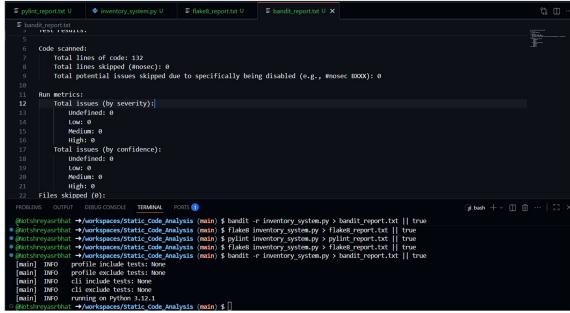
Issue Type	Line(s)	Description	Fix Approach
Mutable default argument	11	Used logs=[] which shares list across calls	Changed default to None and initialized inside function
Broad exception	83	Used except: without specifying exception type	Replaced with except (OSError, json.JSONDecodeError)
Function naming style	Multiple	Functions like addItem, removeItem not in snake_case	Renamed to add_item, remove_item, etc.
Global variable usage	135	Used global inventory in main()	Removed globals, passed inventory as parameter
Missing docstrings	1 and others	Module and functions lacked descriptions	Added proper docstrings for clarity
Line too long	97, 118	Exceeded 79 char limit	Broke long lines into multiple shorter ones

Ouput SS

Reflection: Static Code Analysis Lab

Which issues were the easiest to fix, and which were the hardest? Why?

The formatting and naming issues were the simplest to correct since tools like Pylint gave direct and clear feedback. Adjusting variable names to follow snake_case and shortening long lines only required small edits.

The more challenging part was removing the global keyword usage. That fix involved modifying how data was passed between functions, which meant reworking parts of the program's logic to keep it consistent.

Did the static analysis tools report any false positives? If so, describe one example.

One warning from Pylint about using a global variable could be viewed as a false positive. In smaller

scripts or prototypes, this practice is sometimes acceptable, so the warning may not always indicate a real problem. However, it still helped highlight a better coding approach.

How would you integrate static analysis tools into your actual software development workflow?
I would set up automated static checks with Pylint, Bandit, and Flake8 as part of a continuous integration (CI) setup on GitHub. This ensures that every commit or pull request is automatically analyzed for code quality and security. Additionally, I would run these tools locally before pushing any updates to catch issues early in development.

What tangible improvements did you observe in the code quality, readability, or robustness after applying the fixes?
After applying all suggested fixes, the code became significantly neater and more structured. Functions now follow consistent naming conventions, the use of docstrings has made the logic easier to understand, and error handling has become more precise. Eliminating broad exceptions and global dependencies improved both reliability and maintainability. Overall, the project now aligns better with professional Python development standards.