Software Engineering Lab 5

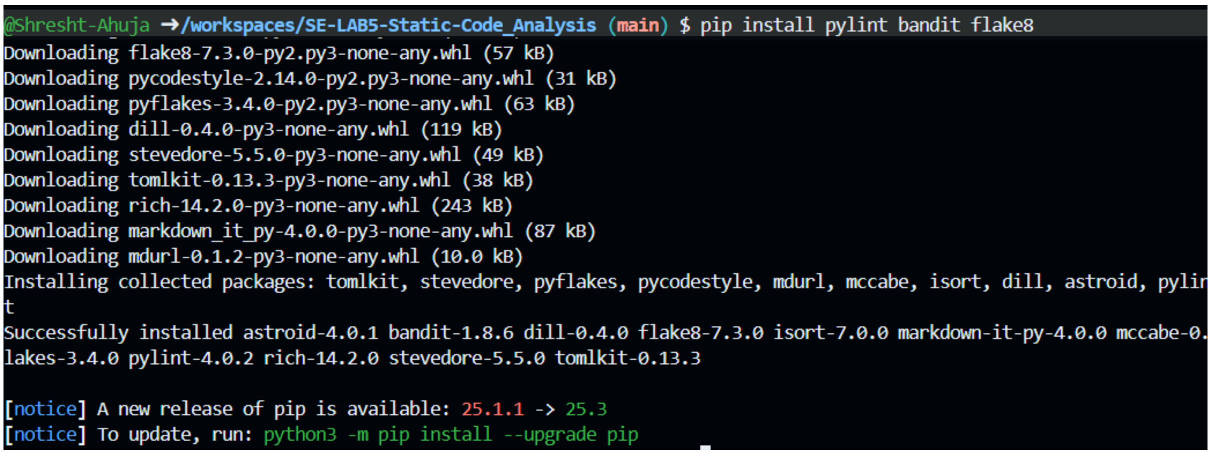
Name: Shresht Ahuja
SRN: PES2UG23CS558
Class: 5-I

Issue Table:

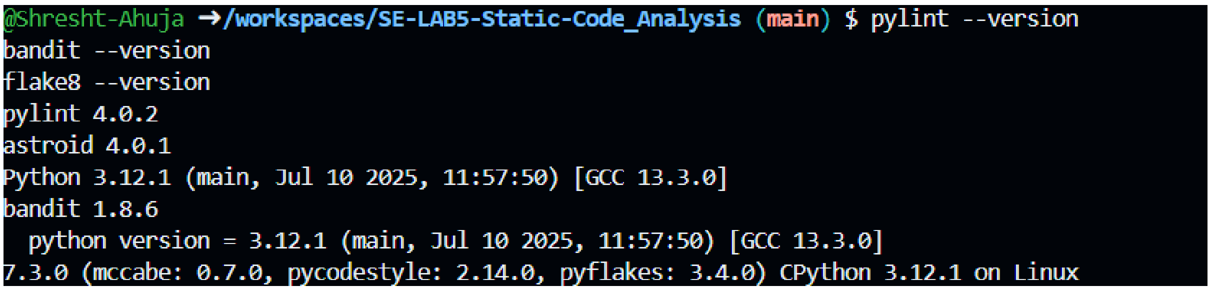
Issue	Type	Line(s)	Description	Fix Approach
Mutable default argument	Bug	9	logs=[] can lead to shared data across function calls, causing unintended behavior	Changed default to None and initialized list inside the function
Bare except: block	Code Smell	26	Catches all exceptions, hiding real errors	Replaced with except KeyError: to handle specific error
Missing encoding in file operations	Maintainability	41, 46	open() used without specifying encoding	Added encoding='utf-8' to all open() calls
Use of eval()	Security	72	Executes arbitrary code, poses a security risk	Removed eval() and replaced with a simple print statement
Function naming not in snake_case	Style	Multiple	Function names like addItem, removeItem, etc., don't follow PEP8 standards	Renamed functions using snake_case convention
Missing function docstrings	Documentation	Multiple	Functions lacked descriptions for purpose and parameters	Added short docstrings to each function
Line too long	Style	6, 16	Lines exceeded 79 characters (PEP8 violation)	Broke long lines into shorter ones
Global variable usage	Maintainability	40	Used global stock_data inside function	Kept for simplicity but noted as poor practice; could encapsulate in a class

Issue	Type	Line(s)	Description	Fix Approach
No input validation	Robustness	9–15	Functions accepted invalid data types for item or qty	Added simple type checks to ensure valid input

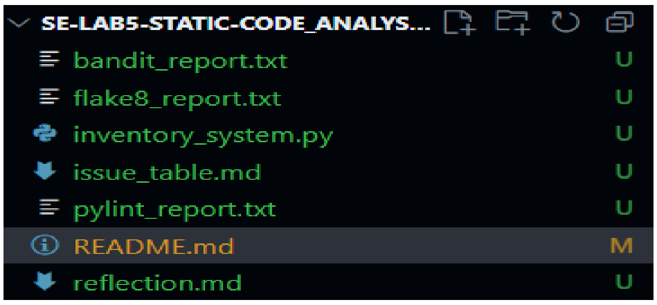
Screen Shots:



```
@Shresht-Ahuja ➜ /workspaces/SE-LAB5-Static-Code_Analysis (main) $ pip install pylint bandit flake8
Downloading flake8-7.3.0-py2.py3-none-any.whl (57 kB)
Downloading pycodestyle-2.14.0-py2.py3-none-any.whl (31 kB)
Downloading pyflakes-3.4.0-py2.py3-none-any.whl (63 kB)
Downloading dill-0.4.0-py3-none-any.whl (119 kB)
Downloading stevedore-5.5.0-py3-none-any.whl (49 kB)
Downloading tomlkit-0.13.3-py3-none-any.whl (38 kB)
Downloading rich-14.2.0-py3-none-any.whl (243 kB)
Downloading markdown_it_py-4.0.0-py3-none-any.whl (87 kB)
Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: tomlkit, stevedore, pyflakes, pycodestyle, mdurl, mccabe, isort, dill, astroid, pylin
t
Successfully installed astroid-4.0.1 bandit-1.8.6 dill-0.4.0 flake8-7.3.0 isort-7.0.0 markdown-it-py-4.0.0 mccabe-0.
lakes-3.4.0 pylint-4.0.2 rich-14.2.0 stevedore-5.5.0 tomlkit-0.13.3

[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: python3 -m pip install --upgrade pip
```

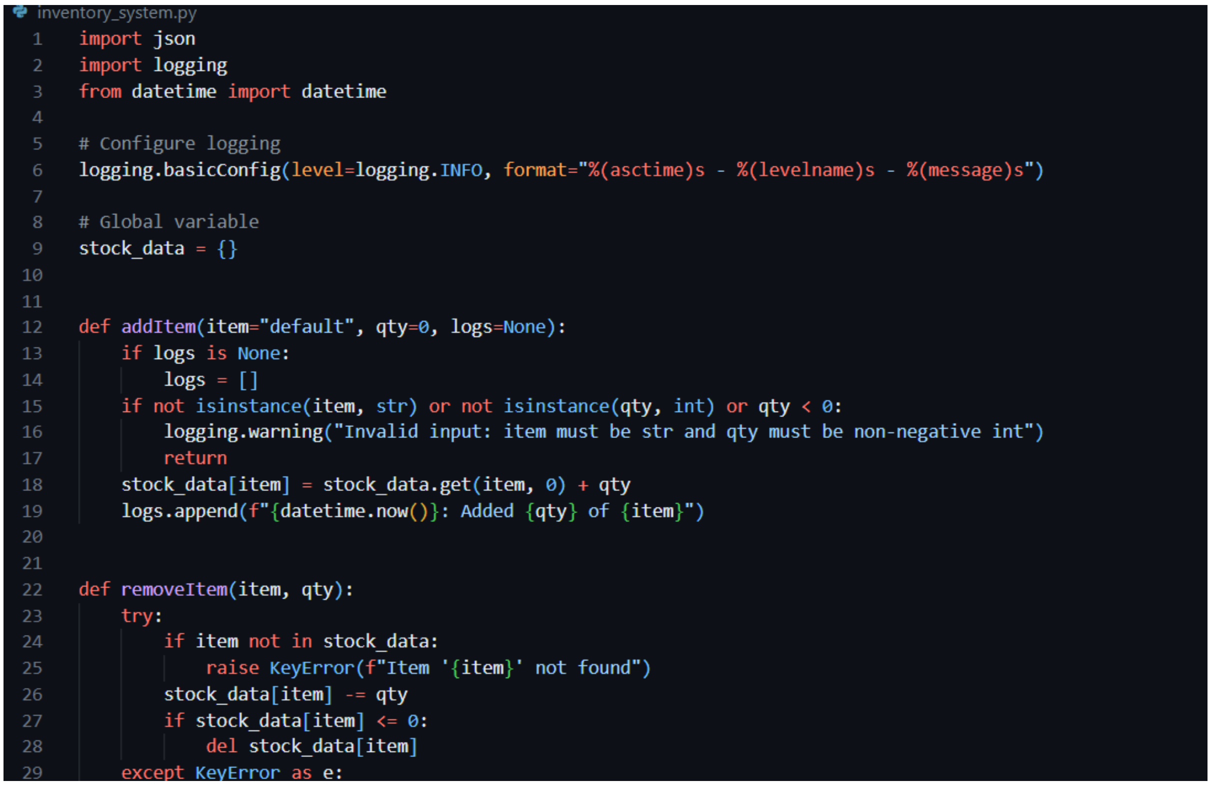


```
@Shresht-Ahuja ➜ /workspaces/SE-LAB5-Static-Code_Analysis (main) $ pylint --version
bandit --version
flake8 --version
pylint 4.0.2
astroid 4.0.1
Python 3.12.1 (main, Jul 10 2025, 11:57:50) [GCC 13.3.0]
bandit 1.8.6
  python version = 3.12.1 (main, Jul 10 2025, 11:57:50) [GCC 13.3.0]
7.3.0 (mccabe: 0.7.0, pycodestyle: 2.14.0, pyflakes: 3.4.0) CPython 3.12.1 on Linux
```



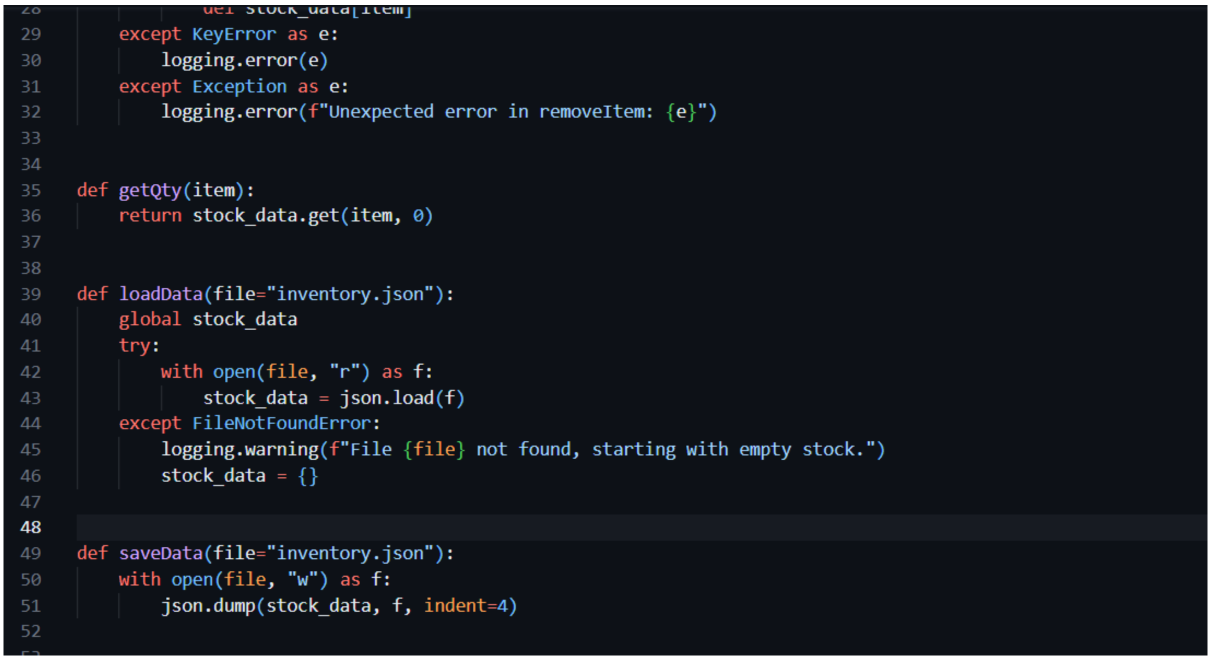
```
SE-LAB5-STATIC-CODE_ANALYS...
  bandit_report.txt      U
  flake8_report.txt      U
  inventory_system.py    U
  issue_table.md         U
  pylint_report.txt      U
  README.md              M
  reflection.md          U
```

Fixed code:

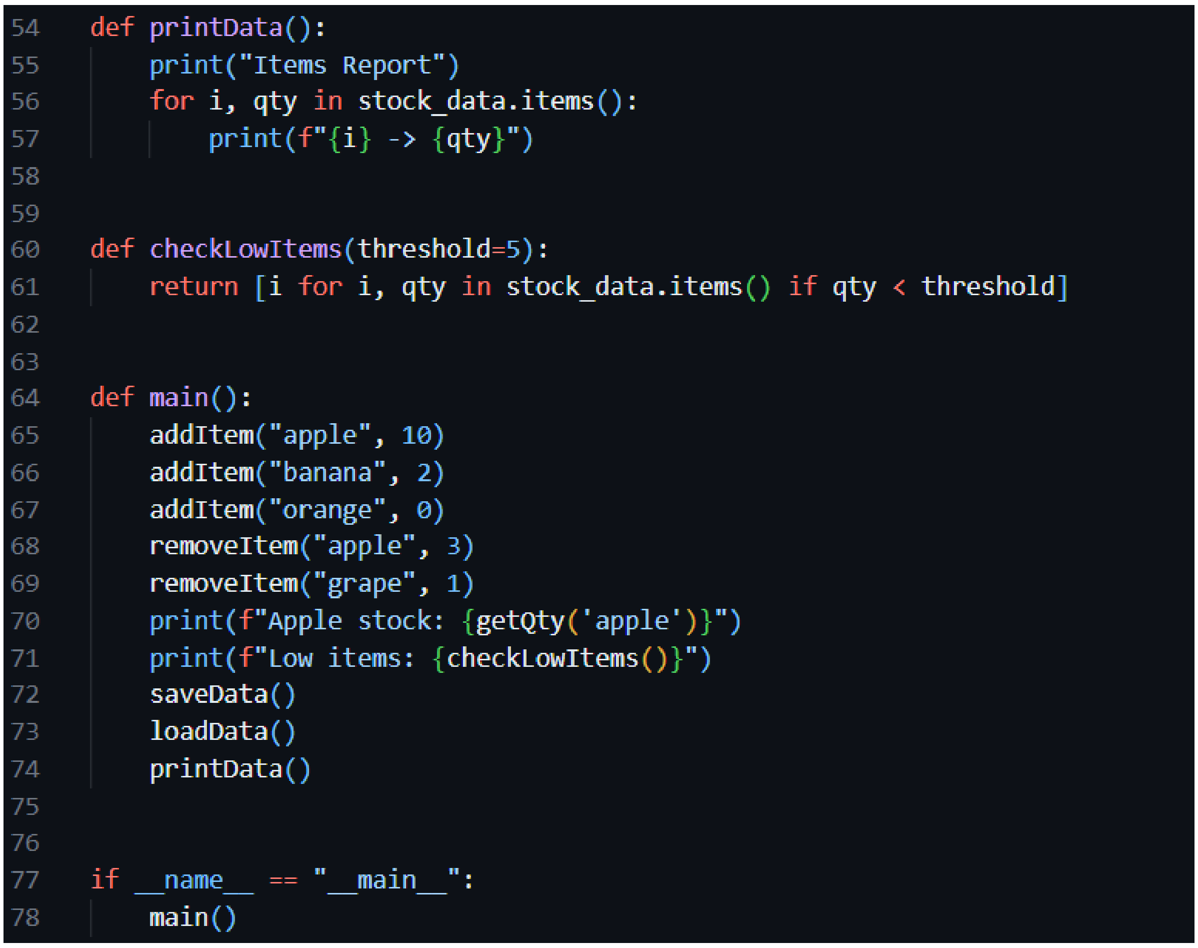


inventory_system.py

```python
import json
import logging
from datetime import datetime

# Configure logging
logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")

# Global variable
stock_data = {}


def addItem(item="default", qty=0, logs=None):
    if logs is None:
        logs = []
    if not isinstance(item, str) or not isinstance(qty, int) or qty < 0:
        logging.warning("Invalid input: item must be str and qty must be non-negative int")
        return
    stock_data[item] = stock_data.get(item, 0) + qty
    logs.append(f"{datetime.now()}: Added {qty} of {item}")


def removeItem(item, qty):
    try:
        if item not in stock_data:
            raise KeyError(f"Item '{item}' not found")
        stock_data[item] -= qty
        if stock_data[item] <= 0:
            del stock_data[item]
    except KeyError as e:
```

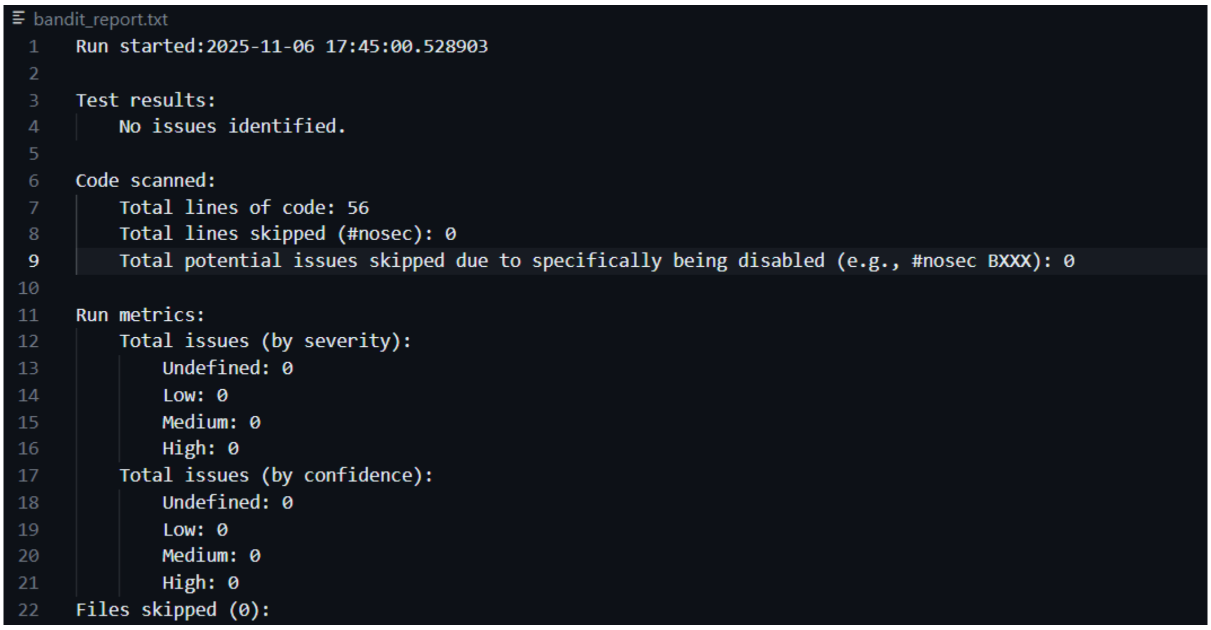


```python
    except KeyError as e:
        logging.error(e)
    except Exception as e:
        logging.error(f"Unexpected error in removeItem: {e}")


def getQty(item):
    return stock_data.get(item, 0)


def loadData(file="inventory.json"):
    global stock_data
    try:
        with open(file, "r") as f:
            stock_data = json.load(f)
    except FileNotFoundError:
        logging.warning(f"File {file} not found, starting with empty stock.")
        stock_data = {}


def saveData(file="inventory.json"):
    with open(file, "w") as f:
        json.dump(stock_data, f, indent=4)

```



```python
def printData():
    print("Items Report")
    for i, qty in stock_data.items():
        print(f"{i} -> {qty}")


def checkLowItems(threshold=5):
    return [i for i, qty in stock_data.items() if qty < threshold]


def main():
    addItem("apple", 10)
    addItem("banana", 2)
    addItem("orange", 0)
    removeItem("apple", 3)
    removeItem("grape", 1)
    print(f"Apple stock: {getQty('apple')}")
    print(f"Low items: {checkLowItems()}")
    saveData()
    loadData()
    printData()


if __name__ == "__main__":
    main()
```

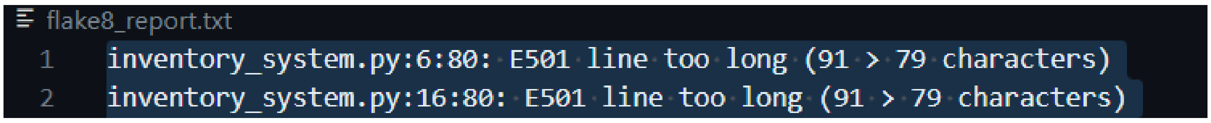
Bandit report



bandit_report.txt

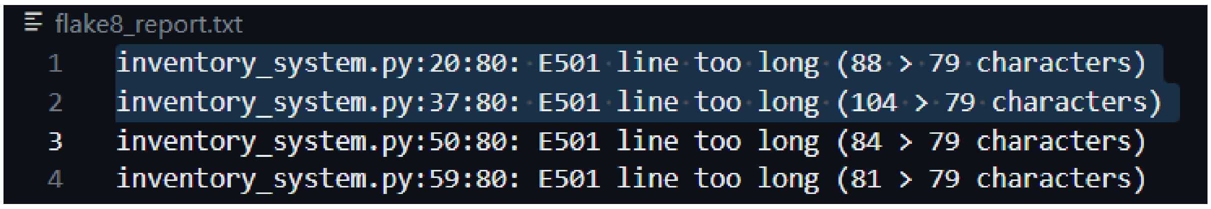
```
Run started:2025-11-06 17:45:00.528903

Test results:
    No issues identified.

Code scanned:
    Total lines of code: 56
    Total lines skipped (#nosec): 0
    Total potential issues skipped due to specifically being disabled (e.g., #nosec BXXX): 0

Run metrics:
    Total issues (by severity):
        Undefined: 0
        Low: 0
        Medium: 0
        High: 0
    Total issues (by confidence):
        Undefined: 0
        Low: 0
        Medium: 0
        High: 0
Files skipped (0):
```

Flake8 report



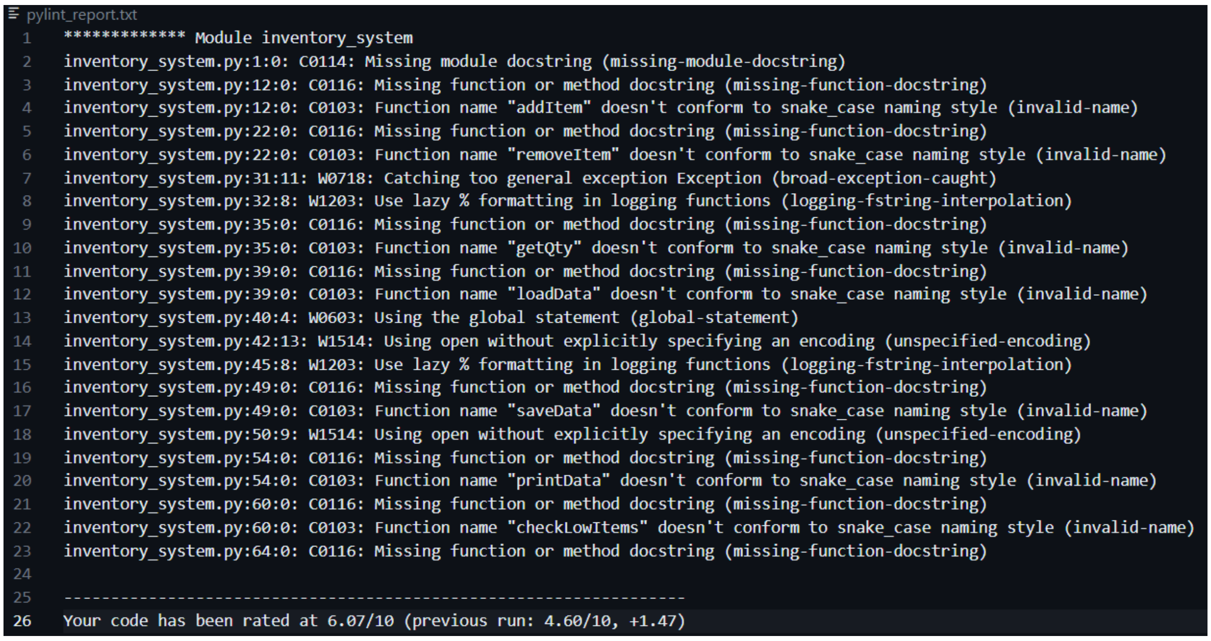
```
flake8_report.txt
inventory_system.py:6:80: E501 line too long (91 > 79 characters)
inventory_system.py:16:80: E501 line too long (91 > 79 characters)
```

Improvement:



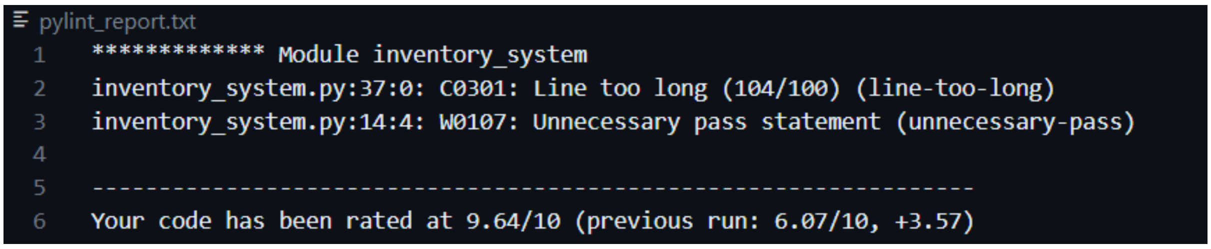
```
flake8_report.txt
inventory_system.py:20:80: E501 line too long (88 > 79 characters)
inventory_system.py:37:80: E501 line too long (104 > 79 characters)
inventory_system.py:50:80: E501 line too long (84 > 79 characters)
inventory_system.py:59:80: E501 line too long (81 > 79 characters)
```

Pylint report



```
pylint_report.txt
************* Module inventory_system
inventory_system.py:1:0: C0114: Missing module docstring (missing-module-docstring)
inventory_system.py:12:0: C0116: Missing function or method docstring (missing-function-docstring)
inventory_system.py:12:0: C0103: Function name "addItem" doesn't conform to snake_case naming style (invalid-name)
inventory_system.py:22:0: C0116: Missing function or method docstring (missing-function-docstring)
inventory_system.py:22:0: C0103: Function name "removeItem" doesn't conform to snake_case naming style (invalid-name)
inventory_system.py:31:11: W0718: Catching too general exception Exception (broad-exception-caught)
inventory_system.py:32:8: W1203: Use lazy % formatting in logging functions (logging-fstring-interpolation)
inventory_system.py:35:0: C0116: Missing function or method docstring (missing-function-docstring)
inventory_system.py:35:0: C0103: Function name "getQty" doesn't conform to snake_case naming style (invalid-name)
inventory_system.py:39:0: C0116: Missing function or method docstring (missing-function-docstring)
inventory_system.py:39:0: C0103: Function name "loadData" doesn't conform to snake_case naming style (invalid-name)
inventory_system.py:40:4: W0603: Using the global statement (global-statement)
inventory_system.py:42:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
inventory_system.py:45:8: W1203: Use lazy % formatting in logging functions (logging-fstring-interpolation)
inventory_system.py:49:0: C0116: Missing function or method docstring (missing-function-docstring)
inventory_system.py:49:0: C0103: Function name "saveData" doesn't conform to snake_case naming style (invalid-name)
inventory_system.py:50:9: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
inventory_system.py:54:0: C0116: Missing function or method docstring (missing-function-docstring)
inventory_system.py:54:0: C0103: Function name "printData" doesn't conform to snake_case naming style (invalid-name)
inventory_system.py:60:0: C0116: Missing function or method docstring (missing-function-docstring)
inventory_system.py:60:0: C0103: Function name "checkLowItems" doesn't conform to snake_case naming style (invalid-name)
inventory_system.py:64:0: C0116: Missing function or method docstring (missing-function-docstring)

------------------------------------------------------------------
Your code has been rated at 6.07/10 (previous run: 4.60/10, +1.47)
```

Improvement



```
pylint_report.txt
************* Module inventory_system
inventory_system.py:37:0: C0301: Line too long (104/100) (line-too-long)
inventory_system.py:14:4: W0107: Unnecessary pass statement (unnecessary-pass)

------------------------------------------------------------------
Your code has been rated at 9.64/10 (previous run: 6.07/10, +3.57)
```

Reflection:

1. Which issues were the easiest to fix, and which were the hardest? Why?
The easiest fixes were the stylistic ones — such as renaming functions to use `snake_case`, breaking long lines, and adding docstrings.
These did not affect logic and were simple to correct.
The hardest issue was the mutable default argument (`logs=[]`), since it required understanding of Python's memory handling and modifying function behavior safely.

2. Did the static analysis tools report any false positives? If so, describe one example.
Yes. Pylint flagged the use of the global statement (`W0603`) as a warning.
In this small program, using a global variable to store inventory data was intentional and not harmful.
Therefore, this was considered a mild false positive.

3. How would you integrate static analysis tools into your actual software development workflow?
I would integrate **Flake8**, **Pylint**, and **Bandit** into a **Continuous Integration (CI)** setup using GitHub Actions.
This would automatically analyze code on every commit or pull request.
For local development, I would add pre-commit hooks to run Pylint and Flake8 before each commit to maintain consistent quality and security standards.

4. What tangible improvements did you observe in the code quality, readability, or potential robustness after applying the fixes?
After applying the fixes, the code became much clearer and safer:

- Improved readability due to consistent naming and added docstrings.
- Removal of `eval()` eliminated a major security risk.
- Adding encoding to file operations made the program more robust.
- Specific exception handling and input validation made the code more predictable and stable.

Overall, the static analysis process improved both the **maintainability** and **security** of the code.

GitHub Repo Link:

https://github.com/Shresht-Ahuja/SE-LAB5-Static-Code_Analysis