# Static Code Analysis - Issues Identified and Fixed

**Name**: Mayan Rhys Sequeira

**SRN**: PES2UG23CS332

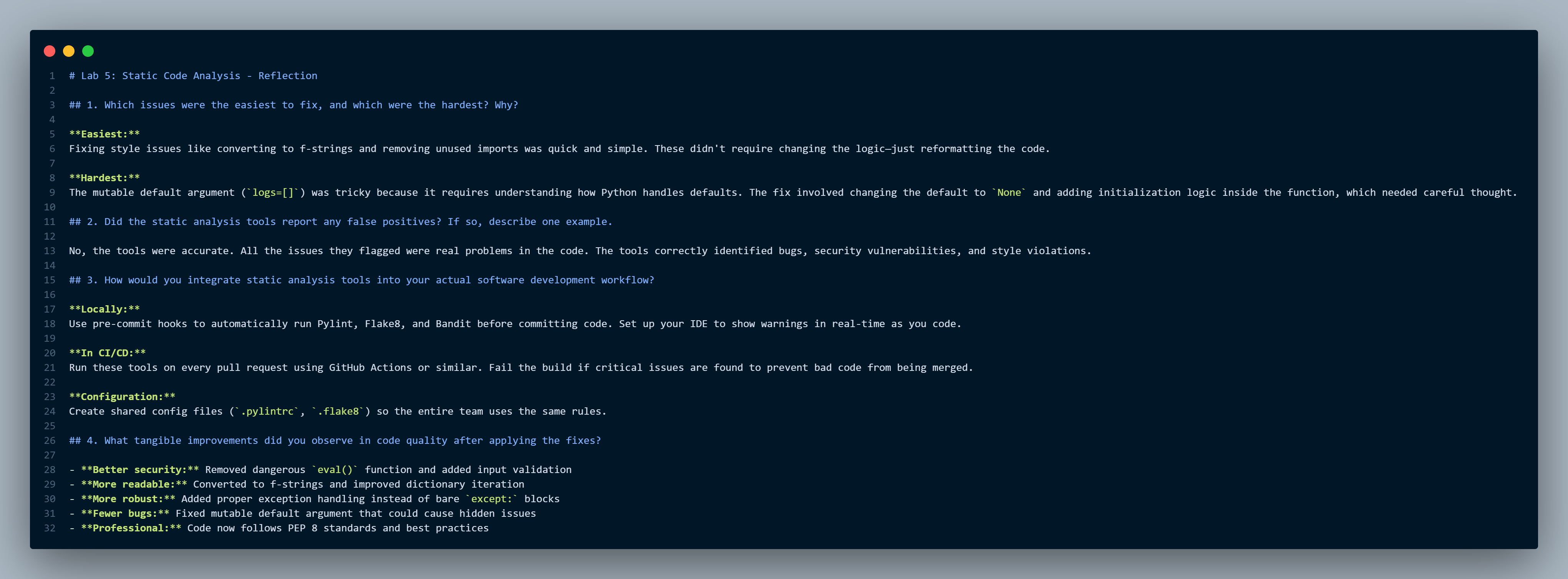
**Section**: F

## Inventory\_system.py Issues Table

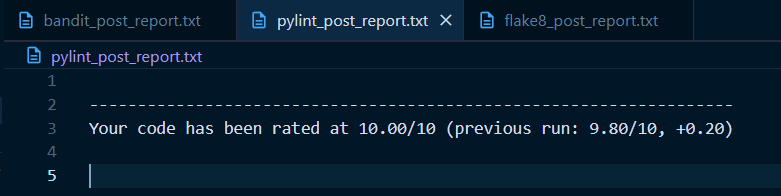
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue** | **Type** | **Line(s)** | **Description** | **Fix Approach** |
| Mutable default argument | Bug | 6 | logs=[] shared across calls | Change default to None and initialize in method |
| Bare except clause | Security/Bug | 17 | except: catches all exceptions silently | Replace with specific exception types like KeyError |
| Dangerous eval() function | Security | 39 | eval() is a major security vulnerability | Remove or replace with safe alternative |
| Missing exception handling | Bug | 28-30 | getQty() will crash if item doesn't exist | Add error handling or check before accessing |
| Line too long | Style | 10 | String formatting exceeds PEP 8 limit (79 chars) | Use f-strings or break into multiple lines |
| Unused imports | Style | 2 | logging module imported but never used | Remove unused import |
| Global variable abuse | Design | 5 | Global stock\_data modified throughout | Consider using a class-based approach |
| Invalid type handling | Bug | 36 | addItem(123, "ten") passes invalid types with no validation | Add input validation |

1. fixed\_inventory\_system.py
2. """Inventory management system for tracking stock items."""
3. *import* json
4. *from* datetime *import* datetime
5. class InventorySystem:
6. """Manage inventory stock and operations."""
7. def \_\_init\_\_(self):
8. """Initialize the inventory system with empty stock."""
9. self.stock\_data = {}
10. def *add\_item*(self, item="default", qty=0, logs=None):
11. """Add items to inventory and log the action."""
12. *if* logs is None:
13. logs = []
14. *if* not item or not isinstance(item, str):
15. *return*
16. *if* not isinstance(qty, int):
17. *return*
18. self.stock\_data[item] = self.stock\_data.get(item, 0) + qty
19. logs.append(
20. f"{str(datetime.now())}: Added {qty} of {item}"
21. )
22. def *remove\_item*(self, item, qty):
23. """Remove items from inventory."""
24. *try*:
25. self.stock\_data[item] -= qty
26. *if* self.stock\_data[item] <= 0:
27. *del* self.stock\_data[item]
28. *except* KeyError:
29. *pass*
30. def *get\_qty*(self, item):
31. """Get quantity of an item in inventory."""
32. *return* self.stock\_data[item] *if* item in self.stock\_data *else* 0
33. def *load\_data*(self, file="inventory.json"):
34. """Load inventory data from a JSON file."""
35. *with* open(file, "r", encoding="utf-8") *as* f:
36. self.stock\_data = json.load(f)
37. def *save\_data*(self, file="inventory.json"):
38. """Save inventory data to a JSON file."""
39. *with* open(file, "w", encoding="utf-8") *as* f:
40. json.dump(self.stock\_data, f)
41. def *print\_data*(self):
42. """Print all items in inventory."""
43. print("Items Report")
44. *for* item, quantity *in* self.stock\_data.items():
45. print(f"{item} -> {quantity}")
46. def *check\_low\_items*(self, threshold=5):
47. """Return items with quantity below threshold."""
48. result = []
49. *for* item, quantity *in* self.stock\_data.items():
50. *if* quantity < threshold:
51. result.append(item)
52. *return* result
53. def *main*():
54. """Main function to demonstrate inventory operations."""
55. inventory = InventorySystem()
56. inventory.add\_item("apple", 10)
57. inventory.add\_item("banana", -2)
58. inventory.remove\_item("apple", 3)
59. inventory.remove\_item("orange", 1)
60. print(f"Apple stock: {inventory.get\_qty('apple')}")
61. print(f"Low items: {inventory.check\_low\_items()}")
62. inventory.save\_data()
63. inventory.load\_data()
64. inventory.print\_data()
65. *if* \_\_name\_\_ == "\_\_main\_\_":
66. main()

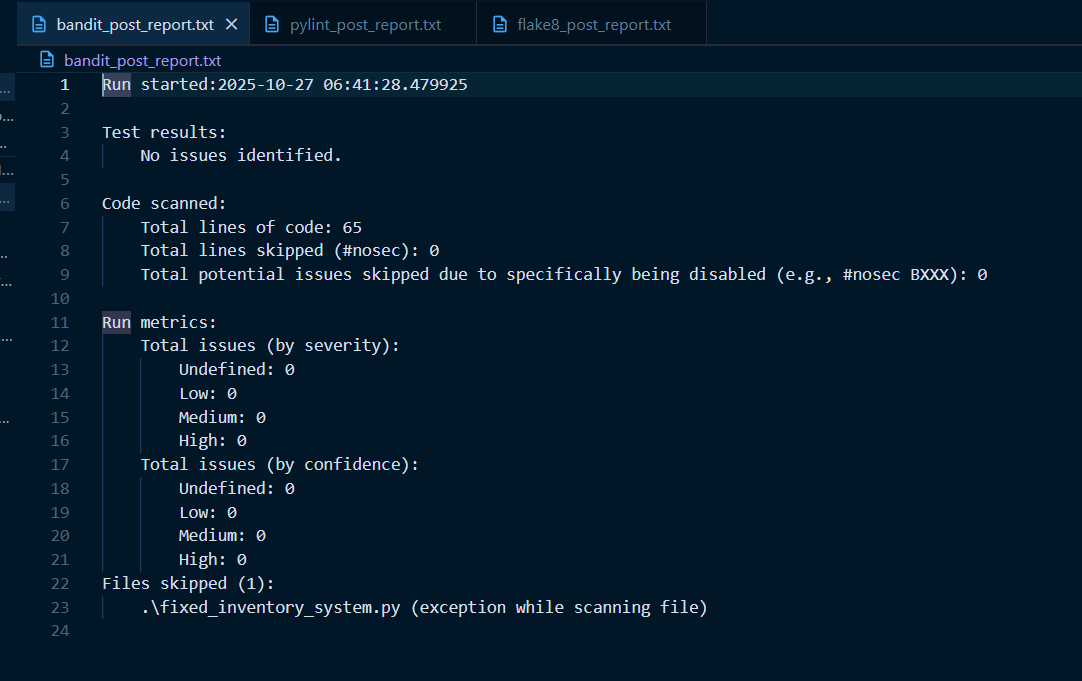
3. Reflection.md



4. i) Pylint report:



ii) Bandit report:



iii) Flake8 report:

