KHULNA UNIVERSITY

Computer Science and Engineering Discipline

Course No. : CSE-3106
Course Name : Software Development Laboratory

Code Review of The Software Project:

Inventory Management System

Project by:
- **Name:** Arnob Chakroborty
  Student ID: 210205
- **Name:** Samia Khanom Asha
  Student ID: 210222

Reviewed by:
- **Name:** Gayotry Gope Toma
  Student ID: 210212
- **Name:** Jannatul Ferdous Shova
  Student ID: 210228

Submitted to:
Dr. Amit Kumar Mondal
Associate Professor
Computer Science and Engineering
Khulna University, Khulna

Name of the Project:

Inventory Management System.

Introduction:

The provided code consists of multiple classes and functions for an Inventory Management System implemented using Tkinter. Below are some observations and recommendations for improvement:

Code Smells:

- ***Code Duplication:***

There's some duplication of code across different classes, such as defining GUI elements like labels, entries, and buttons. Consider consolidating common GUI elements into a separate method or class to avoid redundancy.

- ***Responsibility Overload:***

Classes like InventoryManagementSystem, InventoryGUI, InventoryTrackerGUI, and InventoryManagementGUI handle both GUI construction and application logic. This violates the Single Responsibility Principle (SRP). Separation of concerns should be applied to make the codebase more modular and maintainable.

- ***Readability:***

The code lacks proper documentation (docstrings) explaining the purpose of classes and methods. Adding docstrings will improve code readability and maintainability. Variable names could be more

descriptive, especially in classes like InventoryGUI, InventoryTrackerGUI, and InventoryManagementGUI

- ***User Feedback:***

User feedback messages displayed using messagebox are hard-coded strings. It's recommended to externalize these strings to improve internationalization and ease of maintenance.

- ***Error Handling:***

Error handling is minimal. It's recommended to handle potential exceptions more robustly, providing informative messages to the user. Consider using try-except blocks where file I/O operations are performed to handle potential errors gracefully.

Specific Recommendations:

- ***Refactor Classes:***

Divide the functionality of the existing classes into smaller, more focused classes, adhering to the Single Responsibility Principle (SRP). Consider creating separate classes for GUI construction, user authentication, data manipulation, and inventory management.

- ***Modularize GUI Construction:***

Extract common GUI elements (labels, entries, buttons) into separate methods or classes to reduce duplication and improve maintainability.

- ***Improve Error Handling:***

Implement robust error handling mechanisms, including try-except blocks, to handle exceptions gracefully and provide informative error messages to users.

- ***Enhance Readability:***

Add docstrings to classes and methods explaining their purpose, inputs, and outputs.
Use meaningful variable names that convey the purpose of the variables.

- ***Separate Application Logic from GUI:***

Decouple application logic from GUI construction to improve code modularity and facilitate testing and maintenance.

- ***Consider Externalizing Constants:***

Externalize hard-coded strings (e.g., success and error messages) into constants or configuration files for easy modification and internationalization.

- ***Optimize Code Size:***

Refactor methods and classes exceeding the recommended size limits to improve code readability and maintainability.

- ***Conclusion:***

In conclusion, the provided code exhibits several code smells, including code duplication, responsibility overload, and readability issues. By refactoring the code following the recommended practices, such as

modularization, error handling improvement, and separating concerns, the codebase can become more maintainable, readable, and scalable.