Inventory Management System - Project Summary

# Objective

To build a real-world object-oriented Python application for managing different product types, handling stock operations, and persisting data using JSON — while practicing solid OOP principles.

# Core Features

**1. Product Abstraction**

- `Product` is an abstract base class (`abc.ABC`) with encapsulated attributes: `product\_id`, `name`, `price`, `quantity\_in\_stock`

- Abstract methods: `to\_dict()`, `from\_dict()`

- Concrete methods: `restock()`, `sell()`, `get\_total\_value()`, `\_\_str\_\_()`

**2. Product Subclasses**

- Electronics: Adds `brand` and `warranty\_years`

- Grocery: Adds `expiry\_date`, `is\_expired()`

- Clothing: Adds `size` and `material`

- All subclasses override `\_\_str\_\_()` and support JSON serialization.

**3. Inventory Manager**

- Manages a collection of products.

- Features: add/remove/search, sell/restock, filter, inventory value, remove expired groceries, save/load from JSON.

**4. CLI Menu**

- Interactive menu: add, sell, restock, search, save/load, remove expired, exit.

# Advanced Extras

- Encapsulation: Private attributes (`\_` prefix)

- Custom Exceptions:

- `DuplicateProductIDError`

- `ProductNotFoundError`

- `InsufficientStockError`

- `InvalidProductDataError`

- JSON Persistence with full subclass support

# Technologies Used

- Python 3.x

- `abc` module (abstract base classes)

- `datetime` for expiry checking

- `json` for file I/O

- OOP Concepts: Inheritance, Encapsulation, Polymorphism, Abstraction

# Learning Outcomes

- Strong grasp of OOP design in Python

- Practical use of abstract classes and inheritance

- Handling real-world product variation via polymorphism

- Managing object serialization/deserialization

- Building interactive CLI tools

- Exception handling and input validation