

PROJECT REPORT:

RETAIL MANAGEMENT SYSTEM

OBJECTIVE:

To design and implement a basic retail management system using Object-Oriented Programming principles, with a focus on the use of accessor and mutator methods. The system will manage products, customers, and sales transactions, providing a practical application of fundamental OOP concepts.

PROJECT DESCRIPTION:

The retail management system will include functionalities for managing products, customers, and sales. Students will identify all entities within the system, create classes for each entity and implement methods to perform operations such as adding, updating, and deleting products, registering customers, processing sales, and generating reports. The project will emphasize the use of accessor and mutator methods to control access to class attributes.

INTRODUCTION

This report describes the functionality and implementation of an Inventory and Sales Management System using Python. The system consists of three primary classes: **Product**, **Customer**, and **Sale**, each with their respective methods and attributes. The system also includes a user interface that allows interaction with the system to manage products, customers, and sales transactions.

CLASS DESCRIPTIONS

Product Class

Attributes:

product_id:

name:

description:

price:

stock_quantity:

Methods:

Accessors (Getters):

Mutators (Setters):

`create_product()`: Class method to create and return a new product instance.

`update_product_details()`: Method to update the product details.

`check_stock_level()`: Method to check the stock level of the product.

`__str__()`: Method to provide a string representation of the product.

Customer Class

Attributes:

`customer_id`:

`name`:

`email`:

`purchase_history`:

Methods:

Accessors (Getters):

Mutators (Setters):

`register_customer()`: Class method to create and return a new customer instance.

`update_customer_info()`: Method to update the customer's information.

`add_purchase()`: Method to add a purchase to the customer's purchase history.

`__str__()`: Method to provide a string representation of the customer.

Sale Class

Attributes:

`sale_id`:

`customer`:

`product`:

`quantity`:

`total_price`: Total price for the sale, calculated as `price * quantity`.

Methods:

Accessors (Getters):

Mutators (Setters):

`process_sale()`: Method to process the sale, update stock levels, and add the sale to the customer's purchase history.

`__str__()`: Method to provide a string representation of the sale.

System Functionality:

The system provides the following functionality through a command-line interface:

- **Create a Product**

Users can create a new product by entering the product details. The product is then added to the product list.

- **Create a Customer**

Users can create a new customer by entering the customer details. The customer is then added to the customer list.

- **Update Customer Information**

Users can update the information of an existing customer by providing the customer ID and the new details.

- **Add a Purchase to Customer's History**

Users can manually add a purchase to a customer's purchase history by specifying the customer ID, product ID, and quantity.

- **Create and Process a Sale**

Users can create and process a new sale by providing the sale ID, customer ID, product ID, and quantity. The system checks stock levels, updates them accordingly, and adds the sale to the customer's purchase history.