**E-Commerce Application Testing**

**Objective**

Project Details: E-Commerce Application

It is an e-commerce application to regulate product, customer and order management system created within the Python environment. It delivers fundamental functions such as, stock control, customer profiling, order fulfilment, and transparency.

**Key Features**

**Product Management:**

Buy products that need to be restocked and apply different percent of discounts.

Determine products which are no longer available in stores and estimate their value.

**Customer Management:**

Provide basic operational customer database and keep records of their purchasing standards.

There is nothing as organization’s customers; instead, customers can be categorized according to their spending levels as follows:

Detailed customer information and customer expenditure.

**Order Management:**

Order and return products, fix stock records.

The orders can be retrieved by date range or by the customer number.

Produce bills in details and probable delivery time of products.

**Reporting and Analytics:**

Present the company’s sales report and determine the company’s best-selling products.

Regular customer expenses ought to be tracked and the buying patterns should be assessed.

Create detailed customer orders account.

**Tested Application Details**

* **Language**: Python
* **Purpose**: An e-commerce system for managing products, customers, and orders, with features like order placement, customer analytics, and inventory management.

**Key Features**

1. **Product Management**
   * Adding, restocking, applying discounts, and listing out-of-stock products.
2. **Customer Management**
   * Adding customers, updating details, and tracking loyalty levels.
3. **Order Management**
   * Placing, canceling, and retrieving orders by date or customer.
4. **Reports and Analytics**
   * Generating sales reports, identifying high-spending customers, and calculating inventory value.

**Tools Used**

1. **Unit Testing**:
   * Used Python's unittest framework to create and execute test cases.

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

1. **Coverage Analysis**:
   * Measured code coverage using coverage.py.
   * Command:

coverage run -m unittest discover

coverage report

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

I have **83%** code coverage for the project that I created.

1. **Mutation Testing**:
   * Ensured the robustness of tests using mutmut to simulate bugs and evaluate if the tests could detect them.
   * Command:

mutmut run

mutmut results

A screenshot of a computer

Description automatically generated

**Testing Results**

**1. Key Findings**

* The application passed all test cases.
* Edge cases, such as invalid inputs and non-existent IDs, were handled correctly.
* Reports and analytics methods (e.g., find\_top\_selling\_product) produced accurate results.

**Challenges**

* Handling edge cases for empty datasets (e.g., no customers or no products).
* Ensuring mutation coverage for complex methods such as find\_most\_purchased\_product.

A screenshot of a computer

Description automatically generated