Shoe Store Inventory Management System

Name

School

Professor

Course and Date

Contents

[Introduction 2](#_Toc89298289)

[Problem Statement 2](#_Toc89298290)

[Admin 3](#_Toc89298291)

[MVC Design Pattern 3](#_Toc89298292)

[View 3](#_Toc89298293)

[Model 4](#_Toc89298294)

# Introduction

The purpose of this documentation is to describe and explain the functionalities of the Shoe Store Inventory System. This inventory management system provides an interface where the owner of the shoe store can keep track of the inventory in a physical store. It is used by the owner of a physical store to keep track of the inventory in stock, inventory that needs restocking, the number of suppliers and the products from each to keep track of invoices. It also recalculates the stocks when they are sold and notifies the user when products need restocking.

# Problem Statement

Traditional physical stores are a bigger part of the day-to-day business around. This means that even with the mass transitions made by retailers and sellers to online platforms, physical stores are still a major thing. The problem with having a physical store is having to use traditional ledgers for bookkeeping and inventory management, which has the following disadvantages

1. It can be cumbersome – Traditional ledgers had so many pages and sources of information such as invoices, receipts, promissory notes and much other paperwork that made it such a tiresome part of a business.
2. Time-consuming – processing and analysing business transactions and bookkeeping using cash books and all the traditional inventory systems was time-consuming
3. Prone to errors – during keying in of information and keeping data such as data for future purposes is prone to errors due to forgetting and human errors, mishandling and loss of information which leads to profit loses.

This prompts the adoption of the computer-based inventory management system. This is because computer-based systems are easy to maintain and does not necessarily need the user to remember everything. The data being manipulated is only authorized to the right users and the errors are minimal.

# Admin

The main user of this system is the owner of the small-scale shoe store that wants to track all their inventory and be able to manage it for better results. The system allows them to the key is already available products in the store, update inventory when products are restocked, receive a visual notification when a product is lower than the threshold. The system is meant to make the owners work easier.

# MVC Design Pattern

The MVC design pattern was considered while making this web-based inventory management system. The application’s concerns are separated.

# View

The view displays directly from the model and it visualizes the data through different processes such as change. This package contains user interfaces which include:

* Manage view which displays inventory management for a particular product based on the row the button that was clicked resides.
* Add product view, this view displays a page that allows the administrator to key in information about a new product that does not exist in the database yet.
* Add supplier view which displays a page that allows the user to add a new supplier.
* Order view, this view opens the order page view to allow the owner of the store to make necessary changes to a product when it is sold.
* Index view, this is the view that displays all the details in the inventory database. It displays a dashboard of all the available products, the suppliers and a tally of the purchases made, orders received number of products available and number of suppliers that the store transacts with.

Since the system is built on a Python framework called python, it contains a set of python functions that correspond with all the view packages mentioned.

# Model

It contains the objects that hold the data, state and application. It provides an interface to retrieve the state and notify the observers. This project has these models:

* Orders, this holds the orders ever made in the store and holds them for future references.
* Purchases, this is a data store that holds data on all the purchases the owner of the Shoe Store has made.
* Suppliers. This model holds information about the suppliers that have ever been in contact or have had a transaction with the shoe store. It also displays the content through the index view and while adding inventory thought add product view.
* Products. This is the main model in this system because it holds the most crucial data needed by the owner of the shoe store. The products added into the system are stored in this model and the product management updates both the products model and purchases model.

# Controller

The controller takes the input from the user and interprets them, then either display them on a view or push them to a model. In this system, the controller manipulates the data it picks from the user through forms.

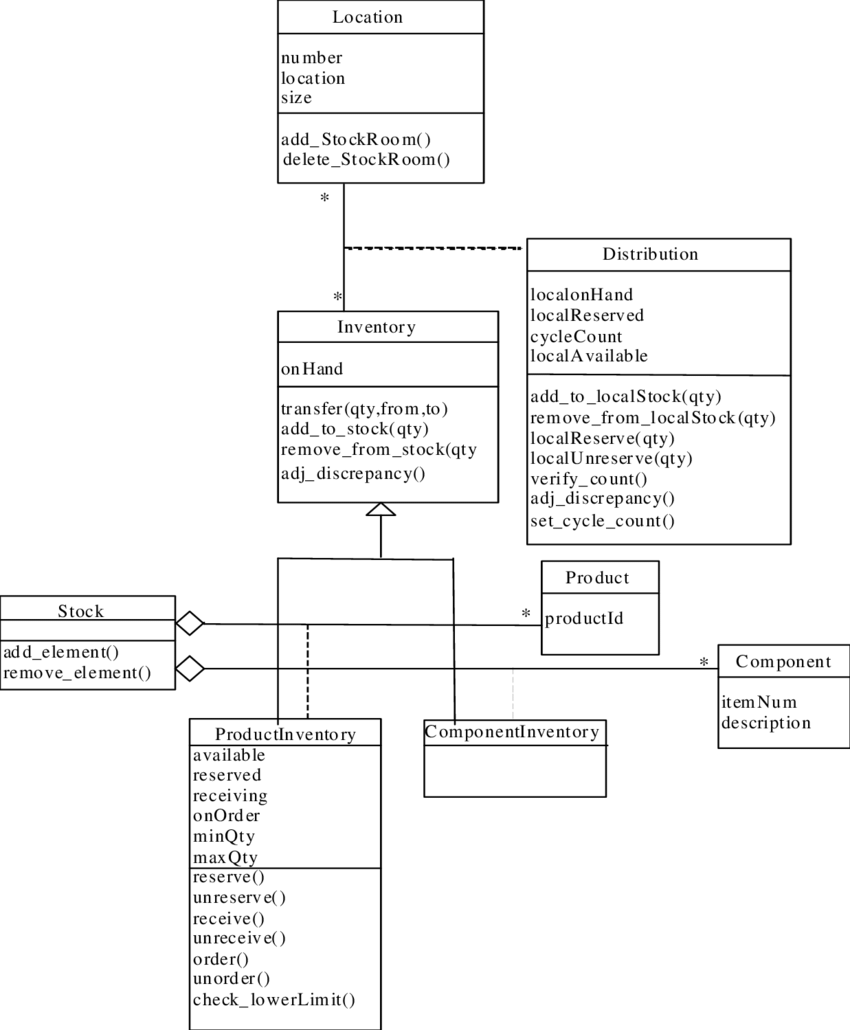
The available controller in the shoe store inventory system include:

* Manage product controller, which manages data from the user through the management view.
* Purchase product. This controller handles information about a new purchase done by the user. It then updates both the product model and the purchase models for an updated index view.
* Supplier. While adding a new supplier, the supplier controller grabs the data input by the user and adds them to the supplier model which feeds the database with new information and redirects the user to the index view to access the new and updated dashboard.
* Make order controller, this controller allows the user to place an order or rather to capture information about new sales.

# Strategy Design Pattern

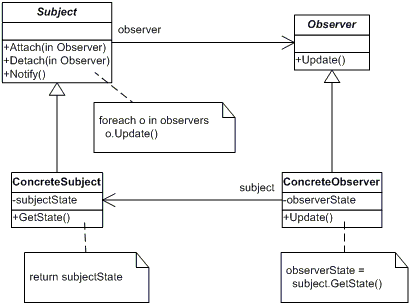
Strategy pattern talks about the aspects that change in the system. Separating what varies and encapsulating them will help in extending or altering the code without affecting the rest of the code.

All the items and types of Items are separated from each other.



# Observer Pattern

An observer pattern is implemented where there are one too many relationships between the objects. When the state of an object is changed, its dependents are notified, and the status is updated.



# Decorator pattern

The Decorator Pattern is implemented such that additional responsibilities can be added to the object dynamically. Decorators provide alternative to subclasses for extending the functionalities.

# Technologies used

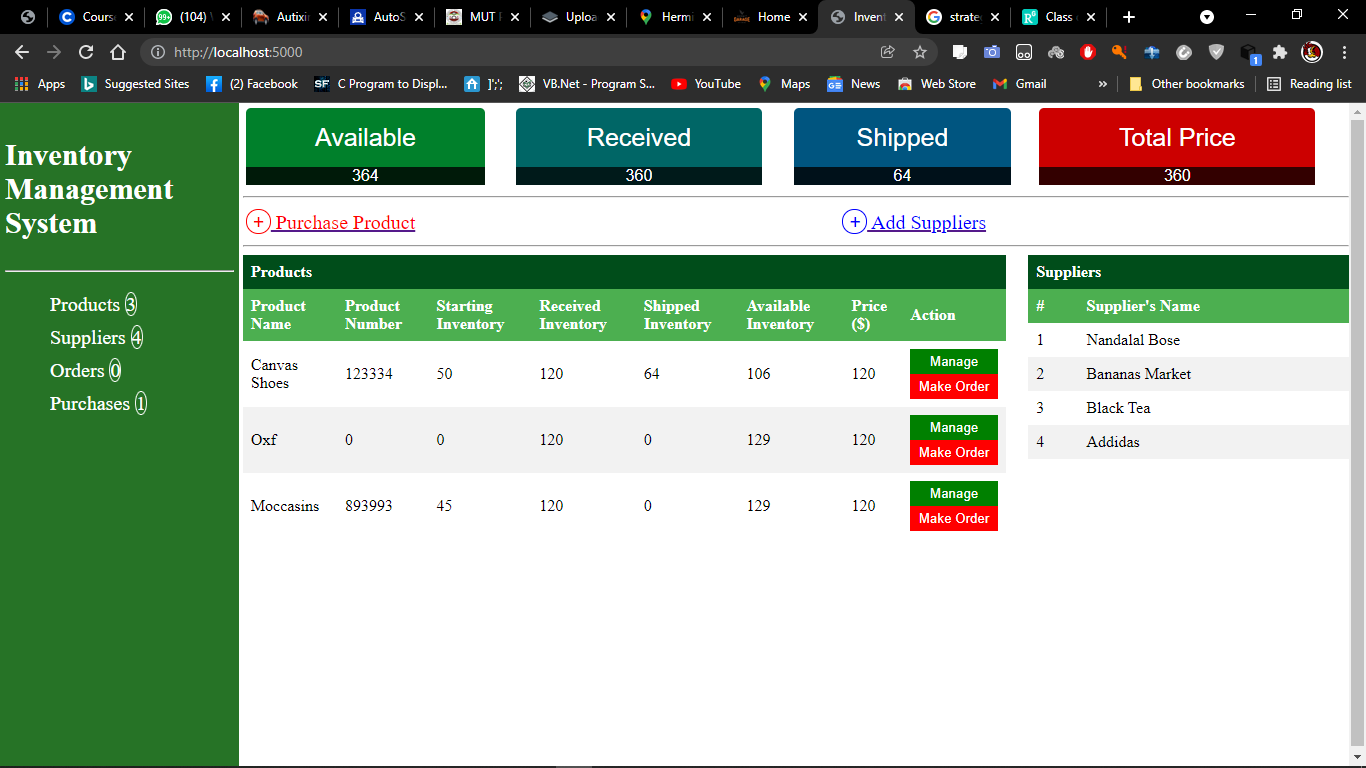
Operating System – Windows 10

Programming language – Python 3.8 through Flask 1.1.4

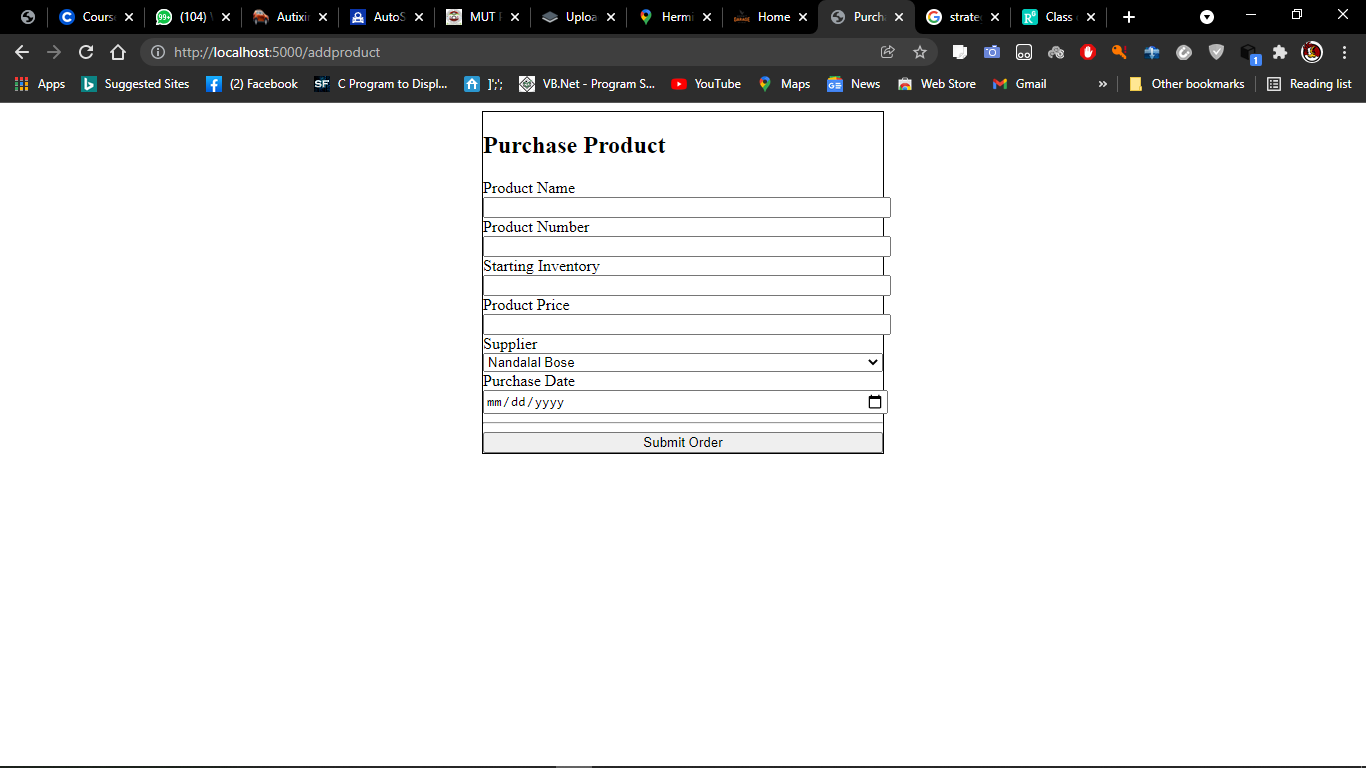
MySQL thought PyMySQL

IDE used Sublime Text

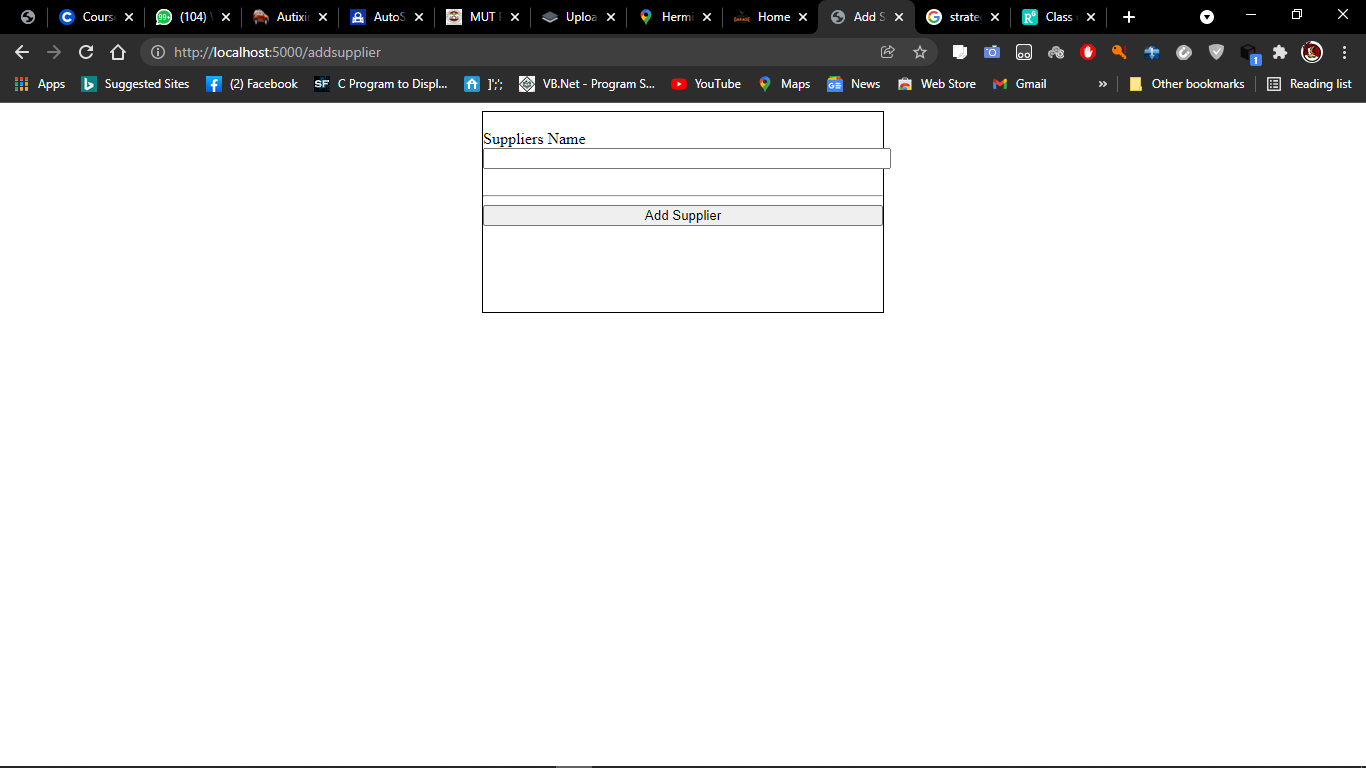
# Inventory management home page



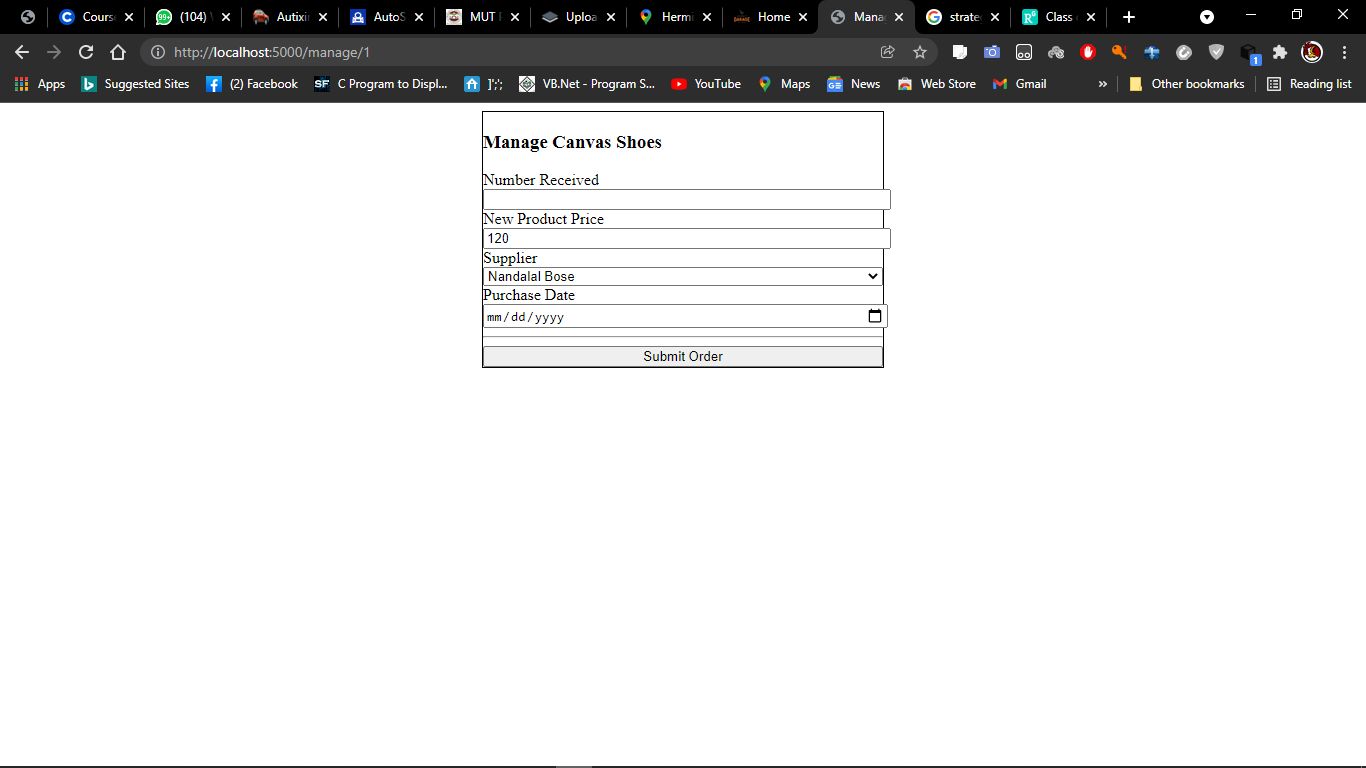
# Purchase product page



# Add Supplier page



# Managing inventory of a particular product



# Make order page

