Ecommerce Vendor Management

Introduction

An Ecommerce vendor management system supports various features such as storing the product data for different vendors.

Here each vendor can save various products and fetch that information whenever required. The focus of the project is to design and implement various functionalities based on that.

Each vendor will have its user_name and password, based on that they can login and retrieve the saved product information.

This product will leverage the core OOPs concepts covered such as class, object, methods, inheritance, abstraction etc.

Housekeeping points

- This is a minimal example and may not follow some standard practices.
- We focus on the main flow, and not much error handling.
- To avoid handling command-line arguments, config file paths are hard-coded in the source files - not a general practice.

Program Organization

The simple program is structured in various layers.

1. **Abstractions:**
    a. Products.py: This file will have the implementation abstract class and methods that are going to be implemented in the Implementation package for product related tasks. The abstract method in Product.py will be there to do *add_product()*, *search_product(), all_products().*
    b. Vendor.py: This file will have the implementation abstract class and methods that are going to be implemented in the Implementation package for vendor related operations. The abstract method in Vendor.py will be there to do *login(), logout().*

2. **Implementation:**
    a. ProductsImplementation.py: In this file learners should implement the actual functionalities that are mentioned in driver.py for product implementation. The methods in the created class will be the extension of the methods created as abstract methods in the abstraction package.

b. VendorImplementation.py: In this file learners should implement the actual functionalities that are mentioned in driver.py for vendor implementation. The methods in the created class will be the extension of the methods created as abstract methods in the abstraction package.

3. **Models:**
    a. ProductModel.py: Implementation of the class and methods that are performing Create and Read operation on the product database.
    b. VendorModel.py: Implementation of class and methods that implements the functionality of loading existing vendors. In this model we are also checking if the correct vendor is passed for VendorSessionModel.
    c. VendorSessionModel.py: Before performing several operations such as adding the product information, reading the product data, functionality such as login, logout of the vendors are implemented in this file.

4. **driver.py:** This file imports the implementation package for product and vendor that has all the methods already implemented and available for use. Take a close look at the method names that are being used to perform the functionality.

Problem Statement

There are two main tasks at hand in this project. These involve writing codes to implement the Models package and Implementation package.

These packages will have implementation of functionalities for Products and Vendors both. In this project, you will write the functionality to add a vendor, login using the credentials of an existing vendor, add products in the catalog and query the products by name as well as the entire list, and logout.

You are provided with a project which contains the boilerplate code which includes the Driver.py. You have to write the Models and Implementations of the project.

1. (**Mandatory**) Add code for implementation package
    a. Implement the login method available under the VendorImplementation package. Implementation of this method has two operations, it should check if the passed `username` vendor is available and it is the credentials of a legitimate vendor. If it is true then it should perform the login method available under `vendorsessionmodel`.
    Valid Vendor Output: `*User xyz logged in successfully!*`
    Invalid Vendor Output: `*Invalid username or password.*`
    b. Implement the logout method available under the VendorImplementation package.
    Valid Vendor Output: `*User xyz logged out successfully!*`

c. Implement `a*dd_product*` method that will call the relevant method available in ProductModel.py after checking if the username is valid or not for the given operation.
d. Implement `search_product_by_name` method that will call the relevant method available in ProductModel.py after checking if the username is valid or not for the given operation. If the given product is not available print the message that the asked product is not available.
e. Implement `get_all_products` method that will call the relevant method available in ProductModel.py after checking if the username is valid or not for the given operation. If there are no products then it should print an error message, else print the product details.

2. (**Mandatory**) Add code for Models package
   a. Implement `search_product` method. If the product_name passed as parameter is available in the dictionary then return the product details else return `None`.
   b. Implement `all_products` method. Return all the data saved in the product dictionary.

Evaluation Rubric

Total Project Points: **100**

- Basic compilation without errors (**10%**) : **10 Points**
- Correctness:
  Correctness of implementation
  - Problem statement - point 1.a (**15%**) : **15 Points**
  - Problem statement - point 1.b (**15%**) : **15 Points**
  - Problem statement - point 1.c (**15%**) : **15 Points**
  - Problem statement - point 1.d (**15%**) : **15 Points**
  - Problem statement - point 1.e (**10%**) : **10 Points**
  - Problem statement - point 2.a (**10%**) : **10 Points**
  - Problem statement - point 2.b (**10%**) : **10 Points**

Program Instructions

1. Download the zipped folder named **M01-W01-02-Ecommerce vendor management.zip**, unzip it on your local machine, and save it. Go into the directory named **M01-W01-02-Ecommerce vendor management.**

2. Make sure you have Python 3.6 or higher installed. At your command prompt, run:

```
$ python --version
Python 3.7.3
```

If not installed, install the latest available version of Python 3.

3. Open the unzipped folder, and make sure that you have some code available in the Models package and driver.py. Implementation package and Abstraction Package are currently empty. Once you have added the code based on the given task, you can run the Driver.py to verify if the added code is working successfully or not.

   ```
   $ python3 Driver.py (On many Linux platforms)
                        OR
   $ python Driver.py      (On Windows platforms)
   ```
   In any case, one of these two commands should work.

4. After running **Driver.py,** you can examine the result. This file will currently run various simple calls that will communicate with different classes and methods in those classes. As you solve the problems, you'll be frequently modifying and running this file. You can comment or modify the initial code as needed.

5. Alternatively, you could install a popular Python **IDE**, such as **PyCharm** or **Visual Studio Code**, and select a command to build the project from there.A helper file/document that has necessary information about how to use PyCharm for the first time will also be available.

6. There will be another file that will help you to understand the import mechanism in python will also be available.Please take help of these files accordingly.

7. Once the program is ready to submit, zip the parent folder **M01-W01-02-Ecommerce vendor management**, and upload the new zip file as **M01-W01-02-Ecommerce vendor management.zip**. It is now ready for evaluation.