Online Herb Shopping

System Design

Version 1.0

10.12.2017

Orhan Aykut Kardeş

Berke Kağan Yıldız

Batuhan Kurt

Mert Günay

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc433996772)

[1.1. Purpose of the System 1](#_Toc433996773)

[1.2. Design Goals 1](#_Toc433996774)

[1.3. Definitions, Acronyms, and Abbreviations 2](#_Toc433996775)

[1.4. References 3](#_Toc433996776)

[2. Current Software Architecture 3](#_Toc433996777)

[3. Proposed Software Architecture 3](#_Toc433996778)

[3.1. Overview 3](#_Toc433996779)

[3.2. System Decomposition 3](#_Toc433996780)

[3.3. Hardware Software Mapping 5](#_Toc433996781)

[3.4. Persistent Data Management 6](#_Toc433996782)

[3.5. Access Control and Security 6](#_Toc433996783)

[3.6. Global Software Control 6](#_Toc433996784)

[3.7. Boundary Conditions 7](#_Toc433996785)

[4. Subsystem Services 8](#_Toc433996786)

[5. References 9](#_Toc433996787)

SYSTEM DESIGN DOCUMENT

# Introduction

Our Herbco system design is quite simple but very functional. This web-site provides to buy the herbal products or medicines by the visitors. Customers can easily find what they want with our functional system design. After the design stage, we can decrease the time required the implementation.

## Purpose of the System

As we said in RAD, this project is a shopping site where users can access herbs and herbal products online in order to reach a healthier life or to get rid of their disease. The main aim of this project is to help the user to find herbs and herbal products that will be good for the health of the user, depending on the health condition or disease. The Herbco website, which contains several quality brands, contains products that the user is looking for. Distributors can add herbal products and herbs to the system and its information. This system also allows the user to place order which will add the items to the user’s cart and make payment for the same. As a result, this system helps the user to achieve a healthier life and cure his or her illness.

## Design Goals

The purpose of our project is to fulfill the requirements of software engineering completely. It is to specify all functional and nonfunctional functions together. With the definition of the functions, we have defined all the requirements for our Herbco project and have prepared an infrastructure for future versions or new projects. In our Herbco system design, we provide to our users or our visitors with easy access to our website to search for herbal products with product names or categories, and read product descriptions, registration and shopping. The features our system evaluates based on non-functional functions are as follows:

* **Dependability**

One of the most significant non-functional requirement is system security. The user security is on the front board in our Herbco system. In addition to security and safety, we paid attention to robustness, reliability, availability and fault tolerance criteria to make a complex system.

* **Maintenance**

Maintenance of your Herbco system is periodically performed by the administrator. Of course, when we are creating to our system, shortcomings such as extensibility, modifiability, adaptability, portability and readability were taken into consideration.

* **End User Criteria**

On our Herbco website, users and visitors can search, sign up, advance search, view their own information, and buy what they want by looking at their products.Our system efficiently stores and retrieves user data in a dynamic manner. In addition , we noticed that utility and usability factors are important for us. The Herbco system supports Microsoft and MacOS operating systems.

* **Performance**

Our Herbco system is responsive and it can accomplish a maximum number of tasks easily. The memory space of our system is available for speed optimizations. As we mentioned the response time, through put and memory criteria are significant for our system.

* **Cost**

We try to accomplish optimal level for cost of our system when we develop it. Also this cost not only for design considerations but managerial , as well. The maintaining backward compatibility with a previous system can add to the development cost while reducing the transition cost. By the way, we handle the development cost , deployment cost , upgrade costs , maintenance cost and administration cost.

## Definitions, Acronyms, and Abbreviations

The abbreviations and definitions contained in the document are given below:

* Herbco: Herb Shopping Company
* Admin: Herbco system admin
* Distributor: Company and brand CEO who sell their products on the site
* Manager: The manager and owner of this system
* Model: A schematic description of a system that accounts for its known or inferred properties.
* System: Any interacts by the application are considered to be done by the system.
* Efficiency: The properties of an algorithm, which is the amount of computational resources used by the algorithm.
* Service: Service is a keyword. Purpose of the service is to provide the customer with a secure payment system.
* OOP: Object Oriented Programming
* POP: Procedural or Produce Oriented Programming
* SQLite: Structed Query Language Lite
* HTML: Hypertext Markup Language
* CSS: Cascading Style Sheets
* MIT License: Massachusetts Institute of Technology License
* API (Django): Application Programming Interface
* SDD: System Design Document.

## References

Requirements Analysis Document (17.10.2017)

<https://www.djangoproject.com>

<https://developer.mozilla.org/en-US/.../Django/Introduction>

<https://github.com/django>

# Current Software Architecture

Our Herbco system is a smart system and, we designed our system on the web. Our system differs from other sites, other sites sell their brands through the site, but in the Herbco system, distributors sell their own products on Herbco, and Herbco does not sell their own brands.

Other difference of our system is that if the products added by the distributors are not liked by the users, as soon as they are removed by the manager of the Herbco system. The communication between the system administrators, distributors and users works in a synchronous manner. The platform is running synchronously, the host and system administrator are very quick to inform. As a result, we used the MVC (Model, View, Controlled) architecture style for our Herbco system. Because, MVC is well suited for interactive systems, especially when multiple views of the same model are needed.

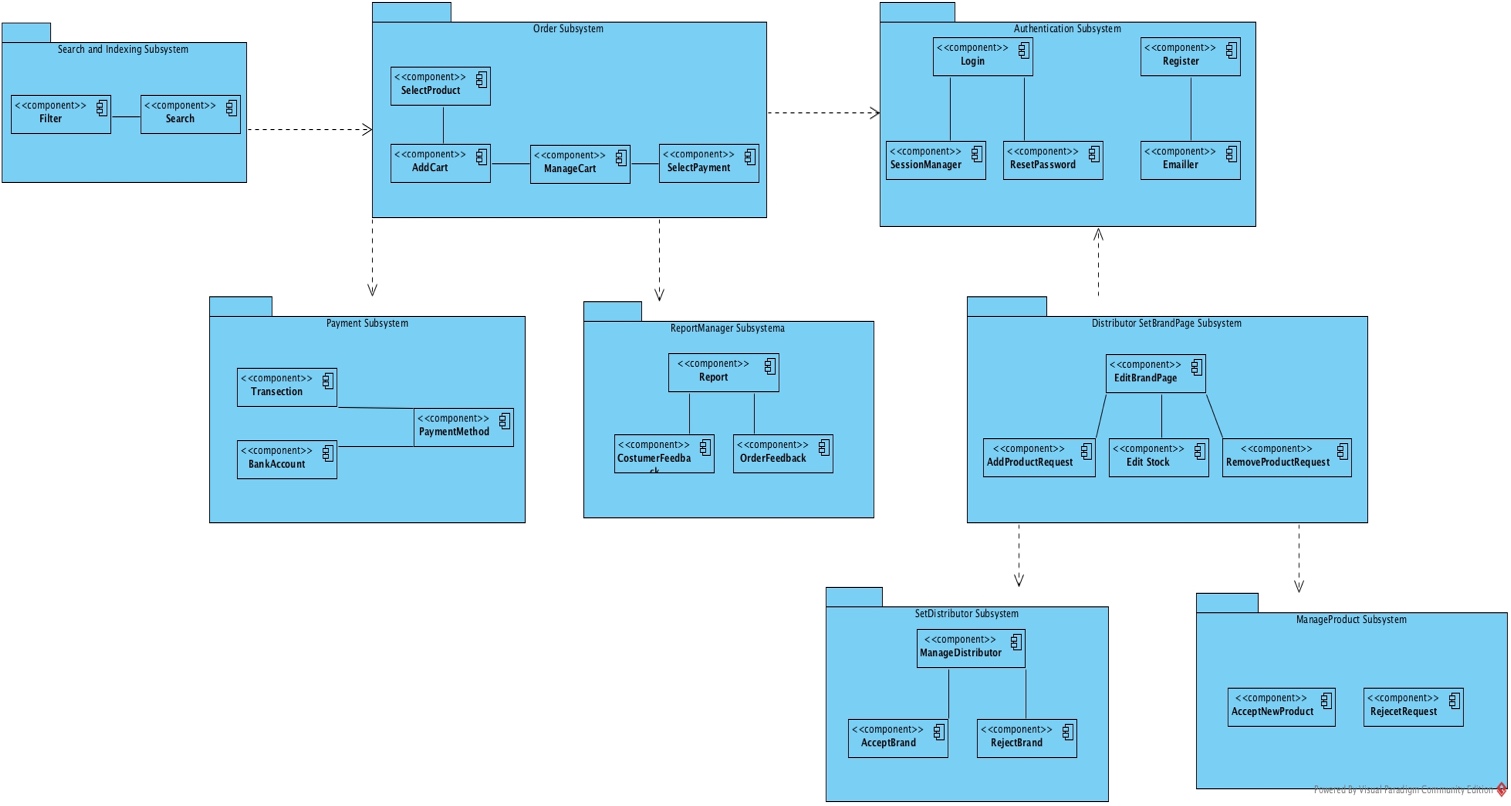
# Proposed Software Architecture

Documents the requirements elicitation and the analysis model of the new system. The Herbco system is web based. Our project will be very useful for patients looking for natural, herbal products. Our system has different features than other similar systems. For example, other systems work with many companies and sell their products, but the Herbco system passes its products to a certain quality standard and presents them to its users under its own protection. Our Herbco system also serves not only for members, but also for users who visit our system without a membership. Those who visit us can search for products, read descriptions, get information, and only have to sign up for shopping. In addition, our system contains all the features that exist in similar systems, for example advance search, easy shopping, simple membership and a clear interface.

## Overview

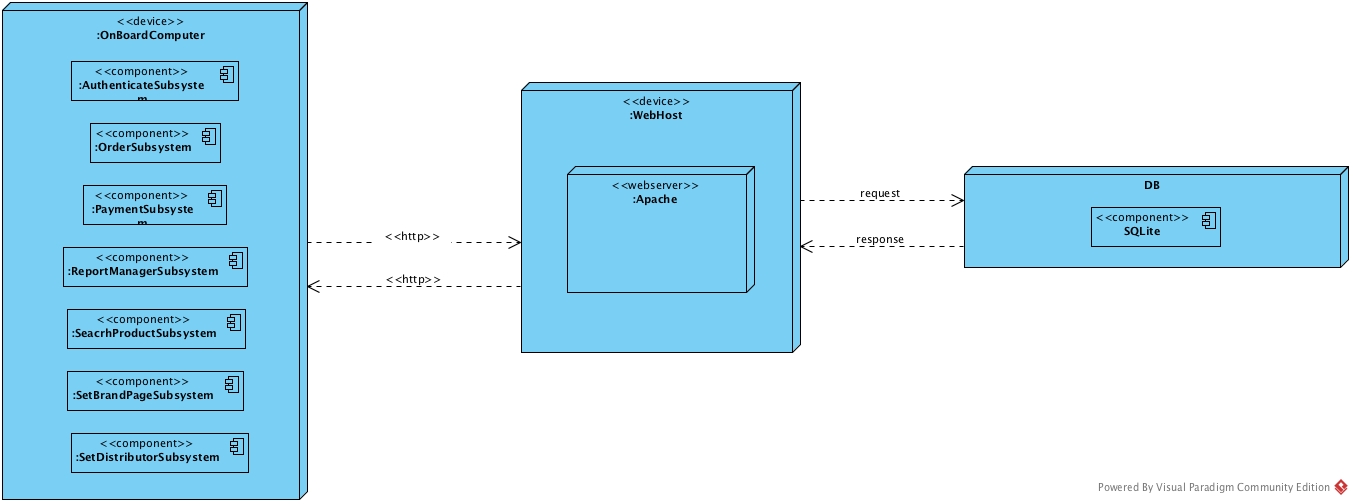
In our Herbco system, we designed subsystems based on software engineering requirements, working more efficiently and working together. This provides us a coherence. We divided to our system some subsystems that are ; add item interface , Admin interface , the basket interface , category interface , approve item interface , distributor accept interface , distributor interface , items of distributor interface , distributor register interface , the web-site main page , the main-page without sign in home page , product detail interface and the manager home page interface.

## System Decomposition



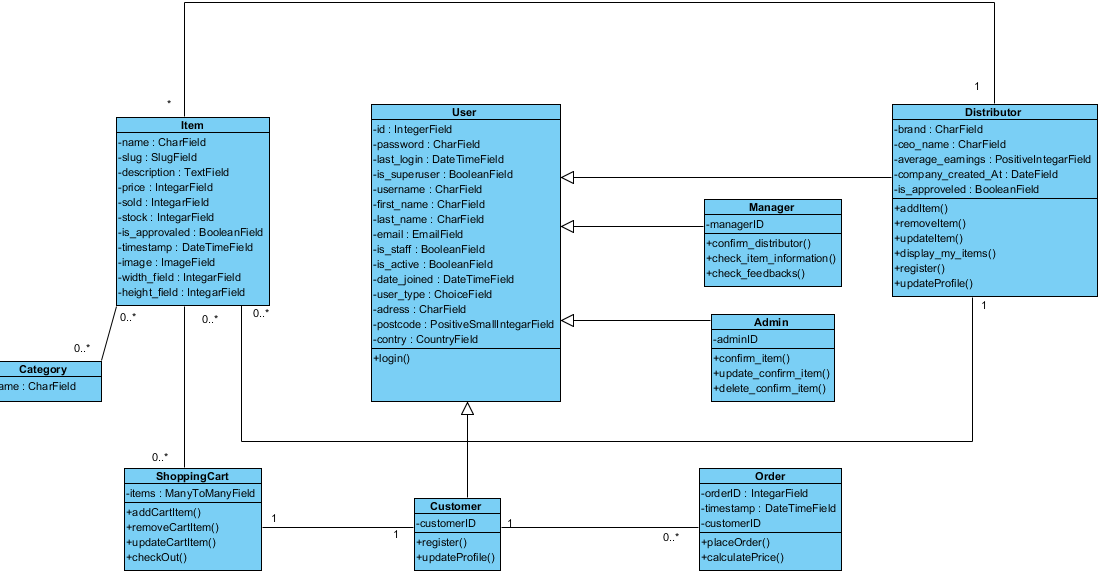
* *Order Subsystem:* This subsystem is responsible for ordering and creating new order-requesting units. This subsystem is also the subscriber to the payment subsystem, making order based on products when any payment is made.
* *Authentication Subsystem:* This subsystem is fully responsible for user login and register. Authorized persons can access the system through this subsystem. Customers and vendors can register with the system through this system.
* *Payment Subsystem:* This subsystem is responsible for managing payment transactions and credit card, bank account balance management. This subsystem is in communication with the order subsystem.
* *ReportManager Subsystem:* This sub-system is responsible for creating problems and handling them. In addition, the feedback generated by this Subsystem User is responsible for managing the complaints. It provides related problems and functions.
* *SearchAndIndexing Subsystem:* This subsystem is responsible for the detailed product search and listing of the user. The user can search for the product as he / she wants. This subsystem is associated with ordering.
* *SetBrandPage Subsystem:* This subsystem is related to managing vendor's own systems. The seller wants to sell the products he wants to sell through this system. The condition is that there is no manager approval and registration in order to do this.
* *ManagerProduct Subsystem:* This subsystem is a system that allows the admin to add and remove products. Admin can add and remove products through this system. it is associated with the subsystem belonging to the distributor.
* *SetDistributor Subsystem*: This subsystem is a system that allows vendor approval. the manager can add or remove the vendors it approves. it is associated with the subsystem belonging to the distributor.

## Hardware Software Mapping



Herbco is web-based online herb shopping system. Herbco is connecting to web server when user tries to visit and login in the system. Herbco system connects to web host with http protocol. Multiple users can access web server simultaneously. Web host have web server. Web server is written with Java. The system connects to database by using Javax libraries. Users that customer, restaurant owner, Admin and Manager which just make an order, use the Herbco system. These actors communicate with Herbco system with http-https protocol web browser in their personal computers. Furthermore, Herbco database server using SQLite for communicating. SQLite is a relational database management system found in a programming library. SQLite is a popular choice as embedded database software for local / client storage applications such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages. The browsers Google Chrome, Opera, Safari, and the Android Browser all allow for storage information, and a SQLite database within the browser, using the Web SQL database technology, although this is rapidly becoming deprecated

## Persistent Data Management

Our system will use the django framework. Django framework creates sqlite databse. This will allow the database to be easily integrated with and accessed by the rest of the system. The database will retain user information for functions such as login for user, show herbs that are created by distributor etc. Our database structure is seen below with entity field’s relations etc.

## Access Control and Security

The application is a multiuser application so it consists of 4 types of users which are customer, manager, distributor and administrator. Because of this, the application will provide different interfaces for each user type.

First, the administrator will connect to the system with the membership interface, and will do the administrator's duties, such as adding, updating, editing, deleting etc. By the way, registration is not necessary for the administrator through the system website, information will be entered manually into the database at the beginning of the system and the administrator will be the authority that will access the database directly. As a summary, the administrator does not have to register because it is initially registered in the database and the system with Django framework. The administrator registers the manager. Each registration process included the manager registration made by the administrator, will be done with the user interfaces of the system. The system will store all the information in the database and in the login processes again the system will use them by collecting data from the database. The information in the database will use both the confirmation and the use of the system for users. All types of users must log in to the system with their username and password.

During registration, field filling does not require access to the database, while completion of the process requires the data to be written to the database, which requires read and write access to the database. In that case, the required database fields will be blocked and simultaneous access of multiple users will be denied.

For some situation like updating or deleting information it is necessary to update one of the tables in the database in its phase of completion and therefore must be handled with more care since several users can be the cause of updating the table at the same time. This will also be avoided by blocking.

Finally, viewing the information or lists again requires read-only access to the database. Therefore, multi-user access does not impose problems and new restrictions.

As last words, the usernames and passwords of users will be stored in the user table. No one else accepts the administrator can have access to this information. Authentication interfaces are different for each type of user and will be directed to their own main pages after the login process.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Actors/  Classes | Admin | Manager | Customer | Distributor | Shopping Card | Order | DbConnection |
| Admin | confirm\_item()  update\_confirm\_item()  delete\_confirm\_item() |  |  |  |  |  | Login()  logout() |
| Manager |  | confirm\_distributor()  check\_item\_information()  check\_feedbacks() |  |  |  |  | Login()  logout() |
| Customer |  |  | Register()  updateProfile() |  | AddCartItem()  removeCartItem()  checkOut() | PlaceOrder()  calculatePrice() | Login()  logout()  Register() |
| Distributor |  |  |  | AddItem()  removeItem()  updateItem()  display\_my\_items()  register()  updateProfile() |  |  | Login()  logout()  Register() |

## Global Software Control

Our system has MVC (Model - View - Controller) software architecture. Herbco is thread safety but also multithreaded program either because our system must provide many users at the same time to order herbs online. And also, some functions of our system should be synchronized for providing less problems during debugging and testing but especially while website is working for real customers. But still threads became with many problems.

## Boundary Conditions

Startup: go to system URL and login

Shut Down: click log out and close browser

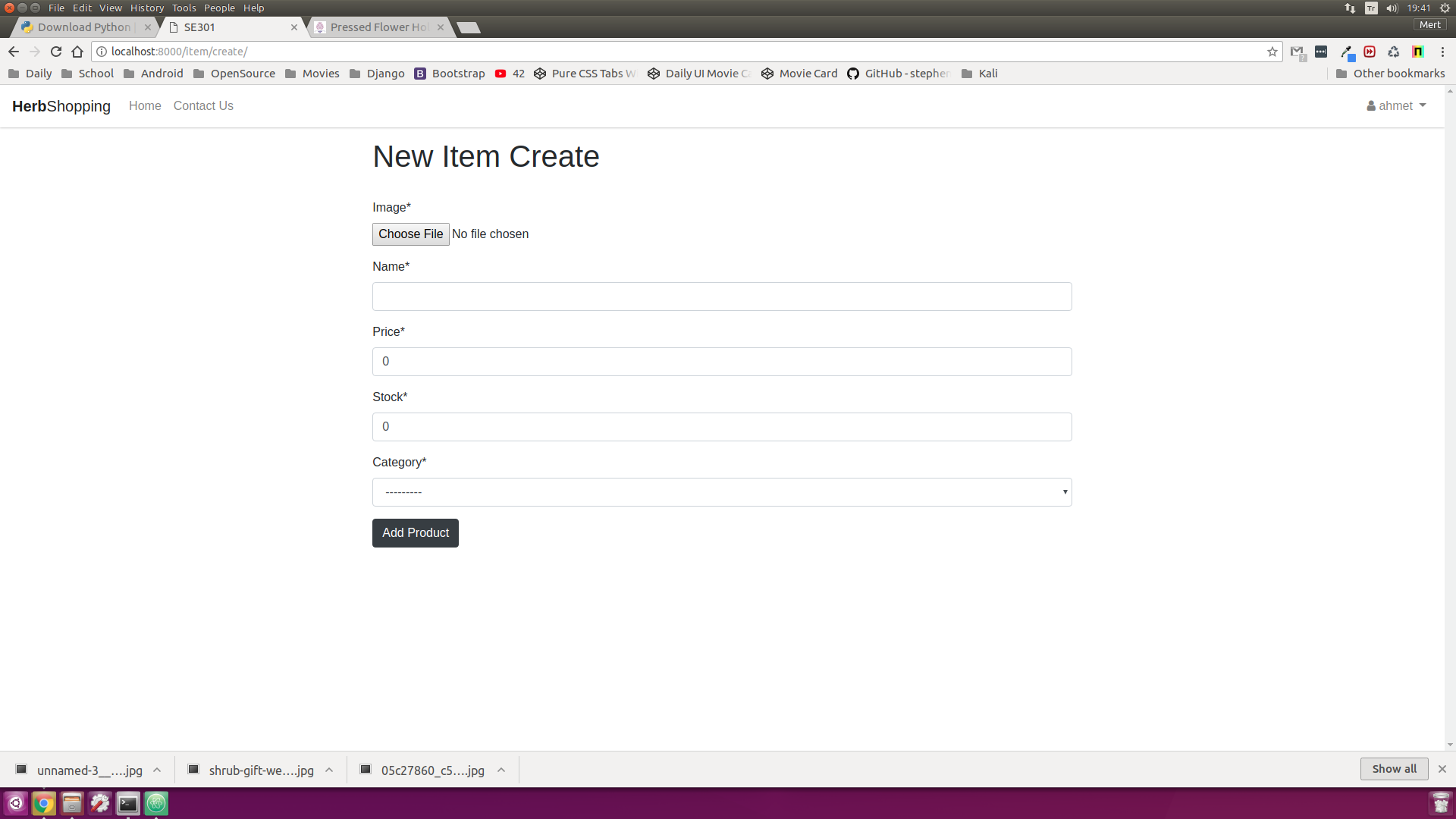
Error Conditions:

* Logging in:
  + Username or password field is blank.
  + Password is not 8 characters long or more.
  + Password and username don’t match.
  + Username is wrong or does not exist.
  + The welcome screen does not appear after logging in.
* User settings
  + User is unable to change certain settings or changes don’t reflect.
  + Between the time of editing and updating, the system crashes.
* Data Entry
  + The system fails when the dispatcher is entering information.
* Distributor Entry
* Distributor informations cant be not exist.
* Distributor informations cant be blank.
* Customer Entry
  + Customer informations cant be not exist.
  + Customer informations cant be blank
* Distributor Table
  + Display items can't be used if there is no item added.
* ShoppingCart table:
* Item quantity cant be zero.
* Item quantity cant exceed the stock amount of the Item.
* Logging out
  + Dispatcher unable to logout.

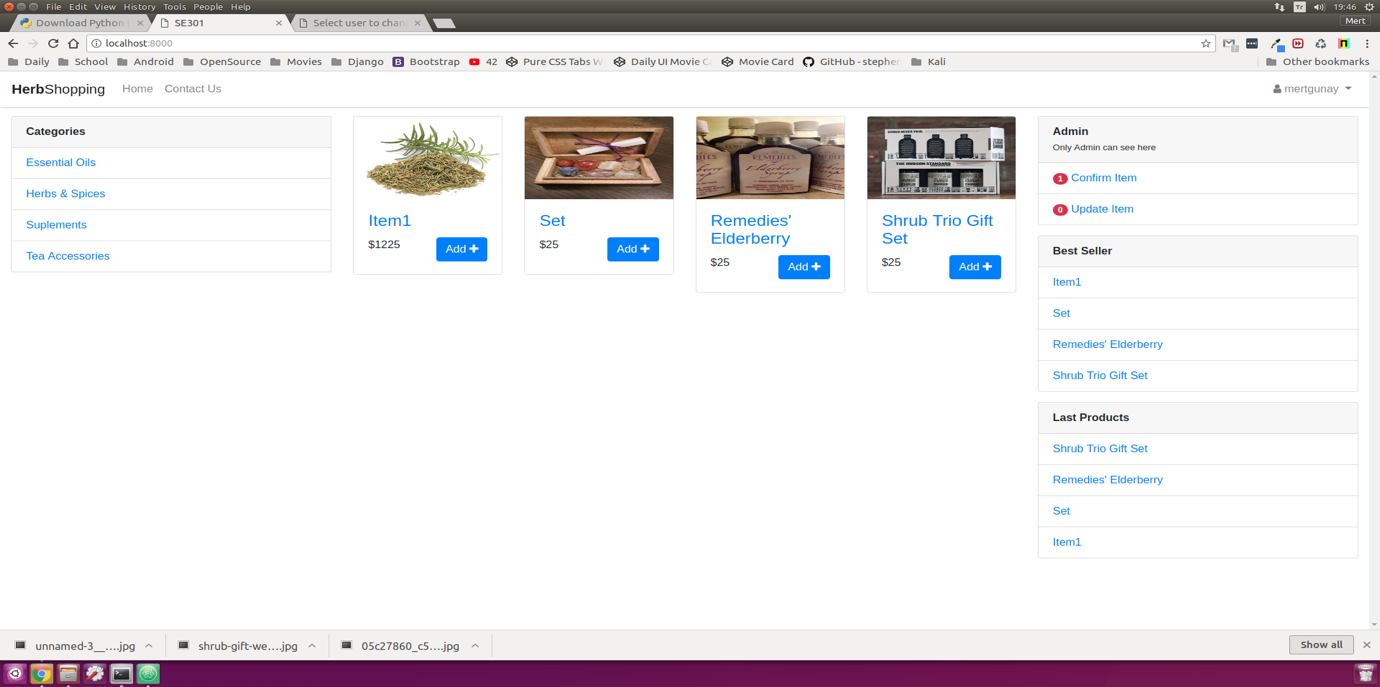
# Subsystem Services

We prefer to divide our system to subsystems to adapt each other themselves and, provide the whole system functionalities.

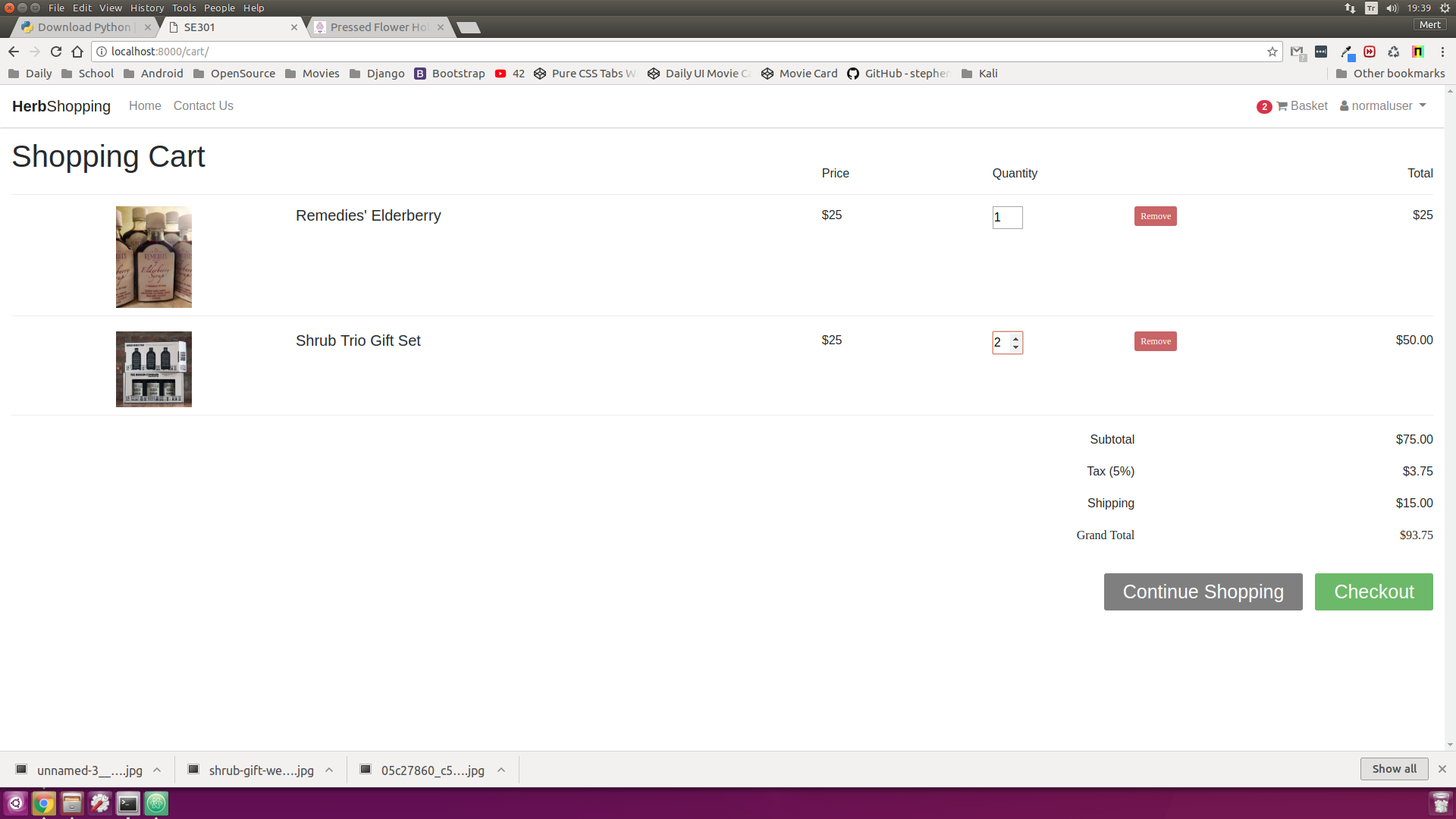
* **Add Item Interface;**



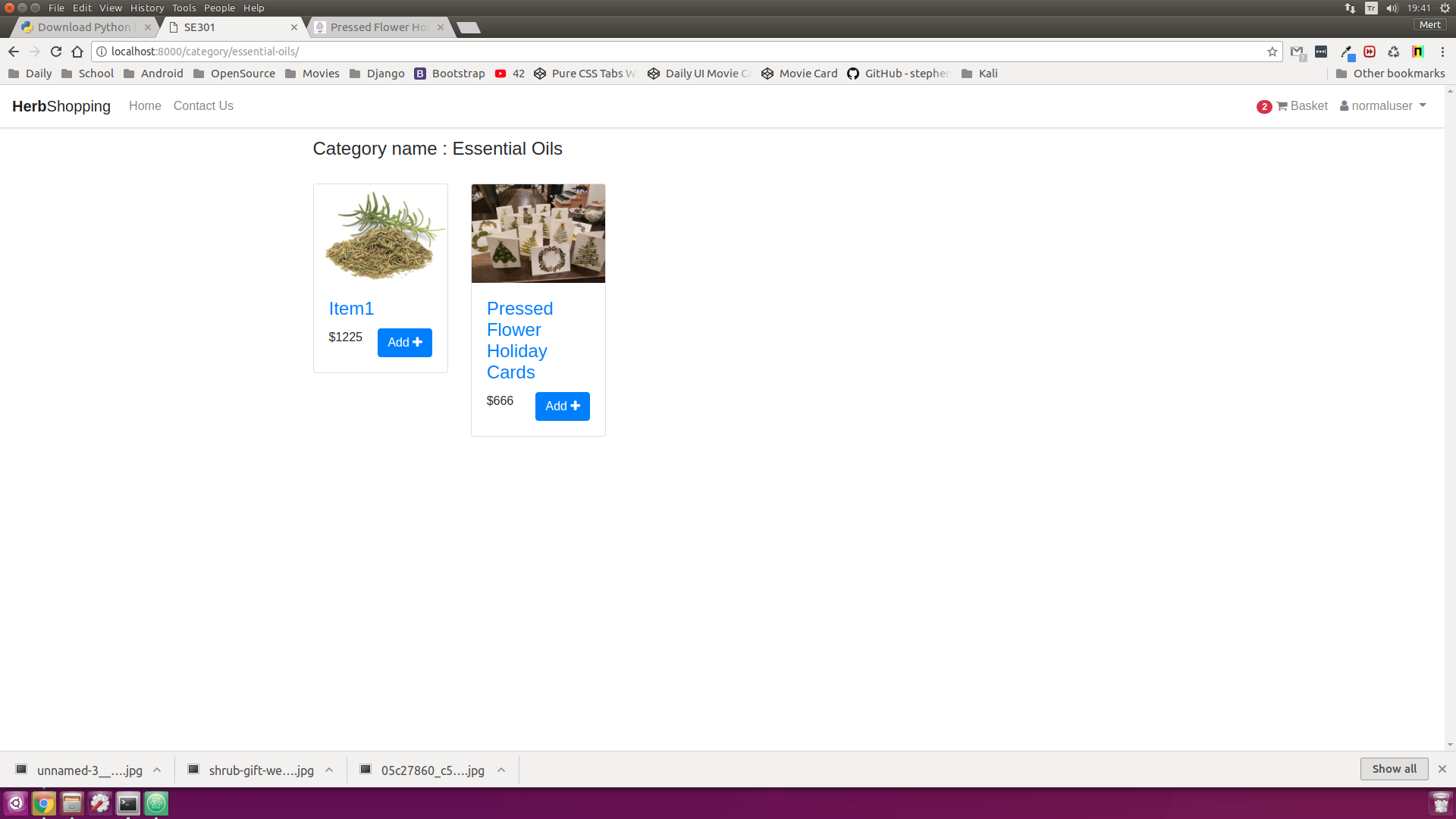
* **Admin Interface**



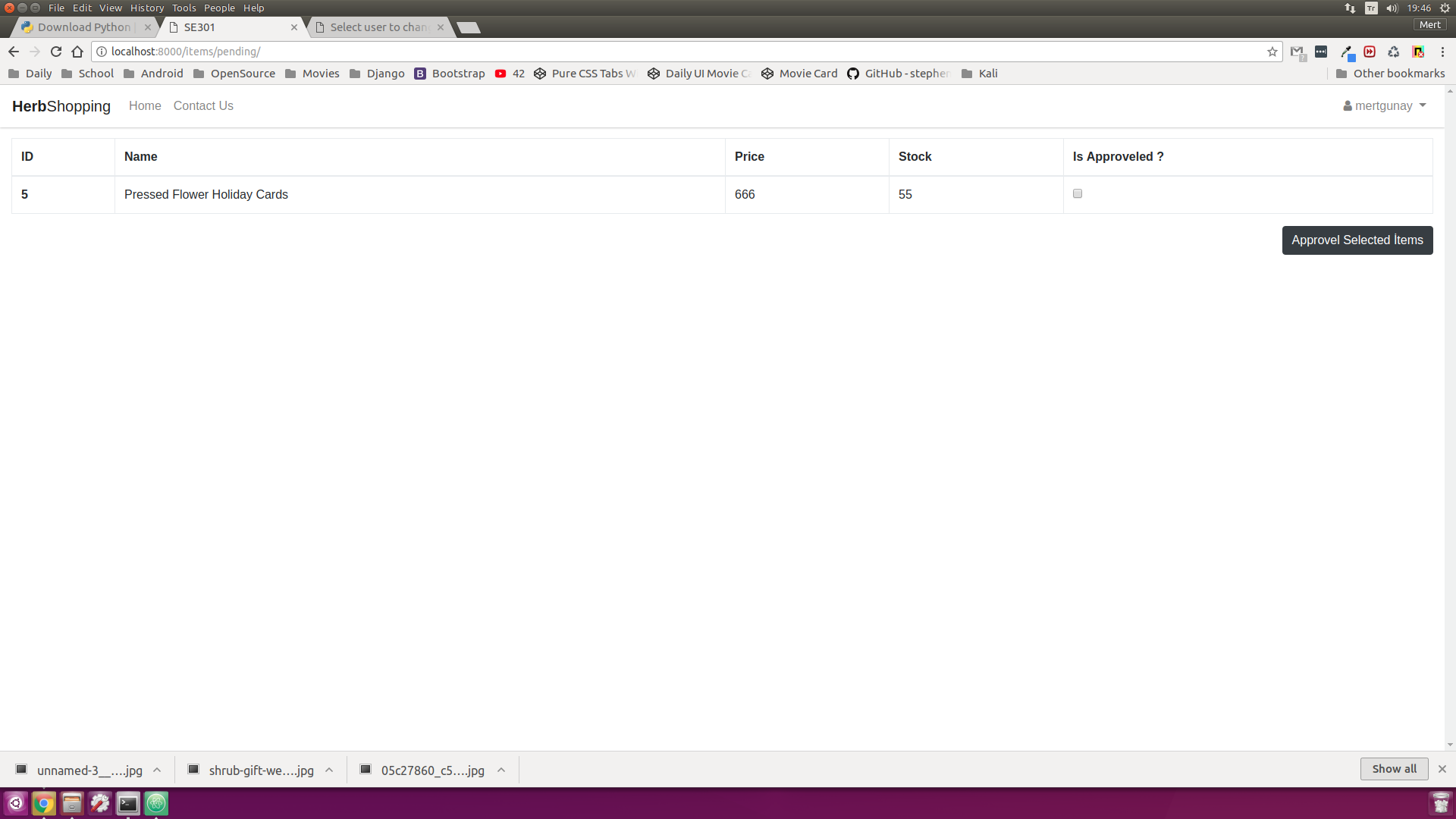
* **Basket Interface**



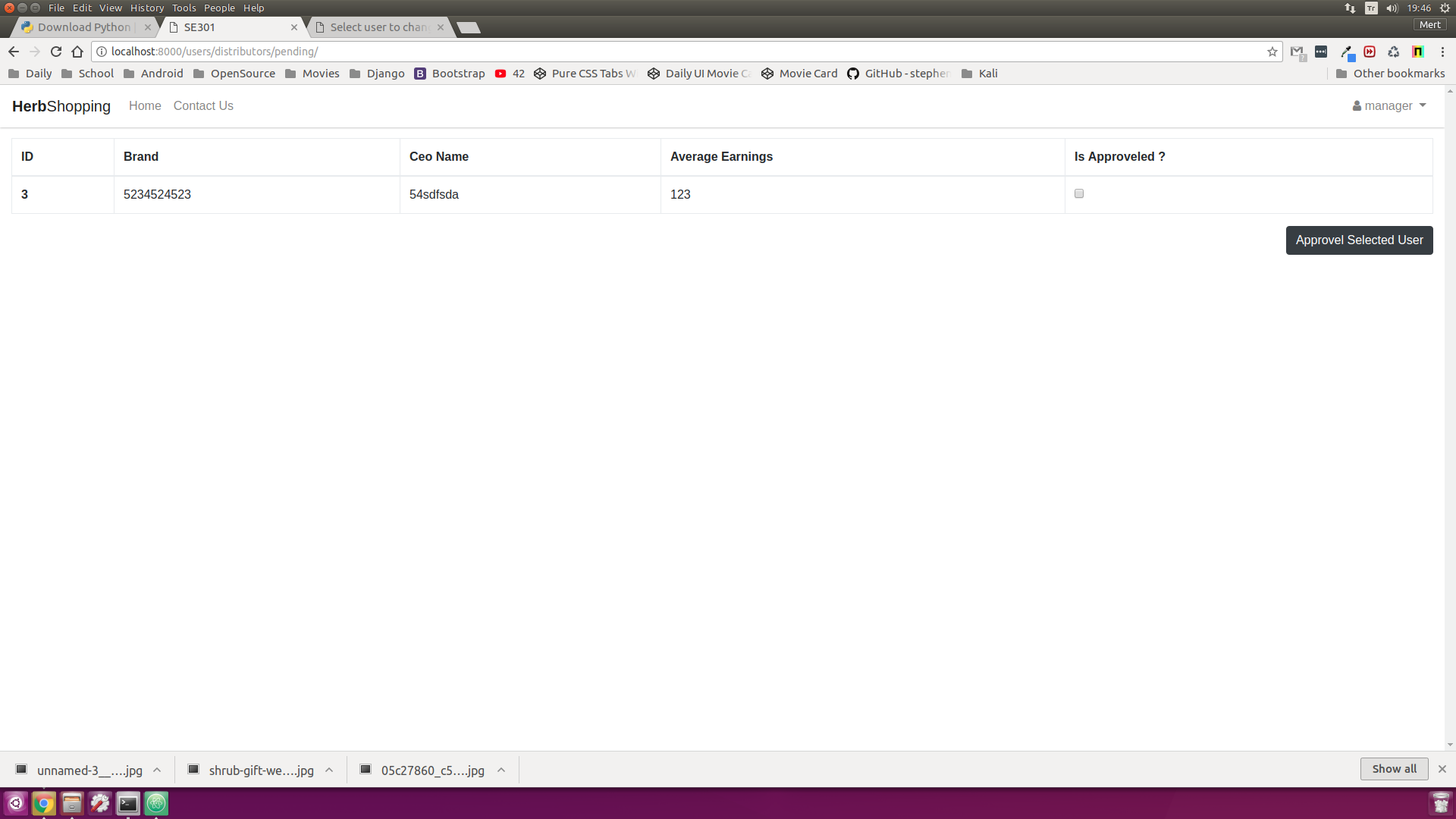
* **Category Interface**



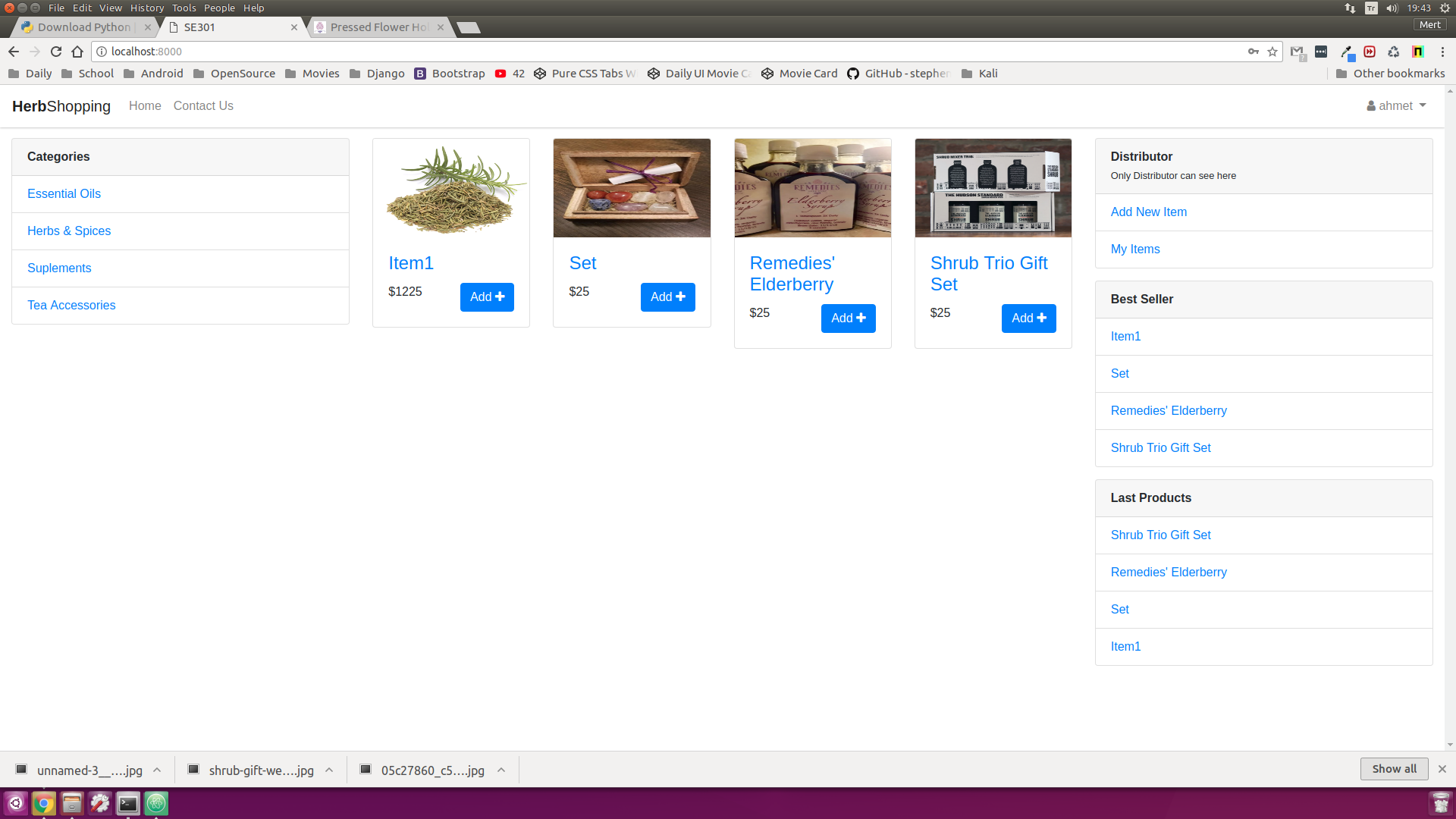
* **Approve Item Interface**



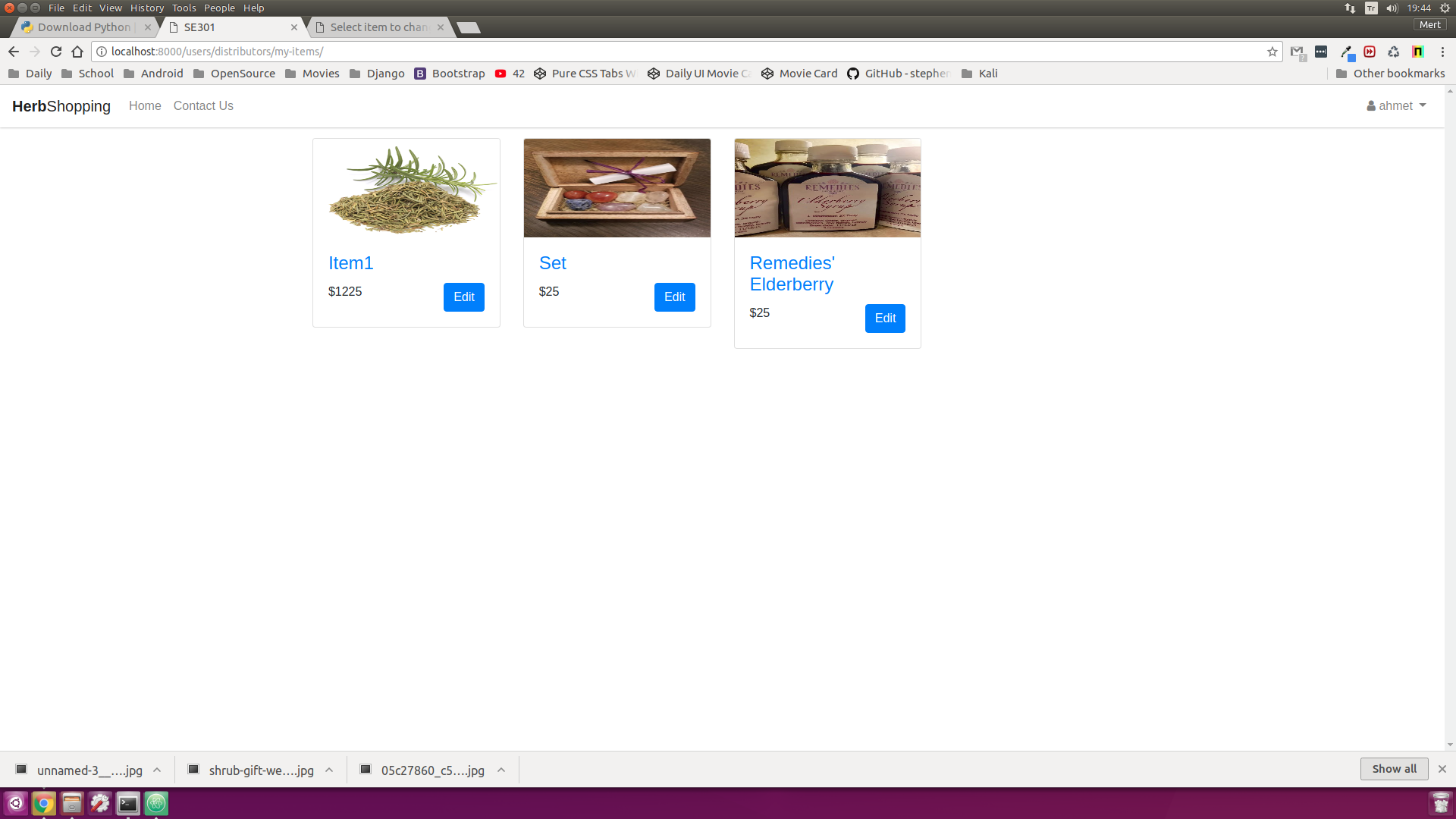
* **Distributor Accept Interface**



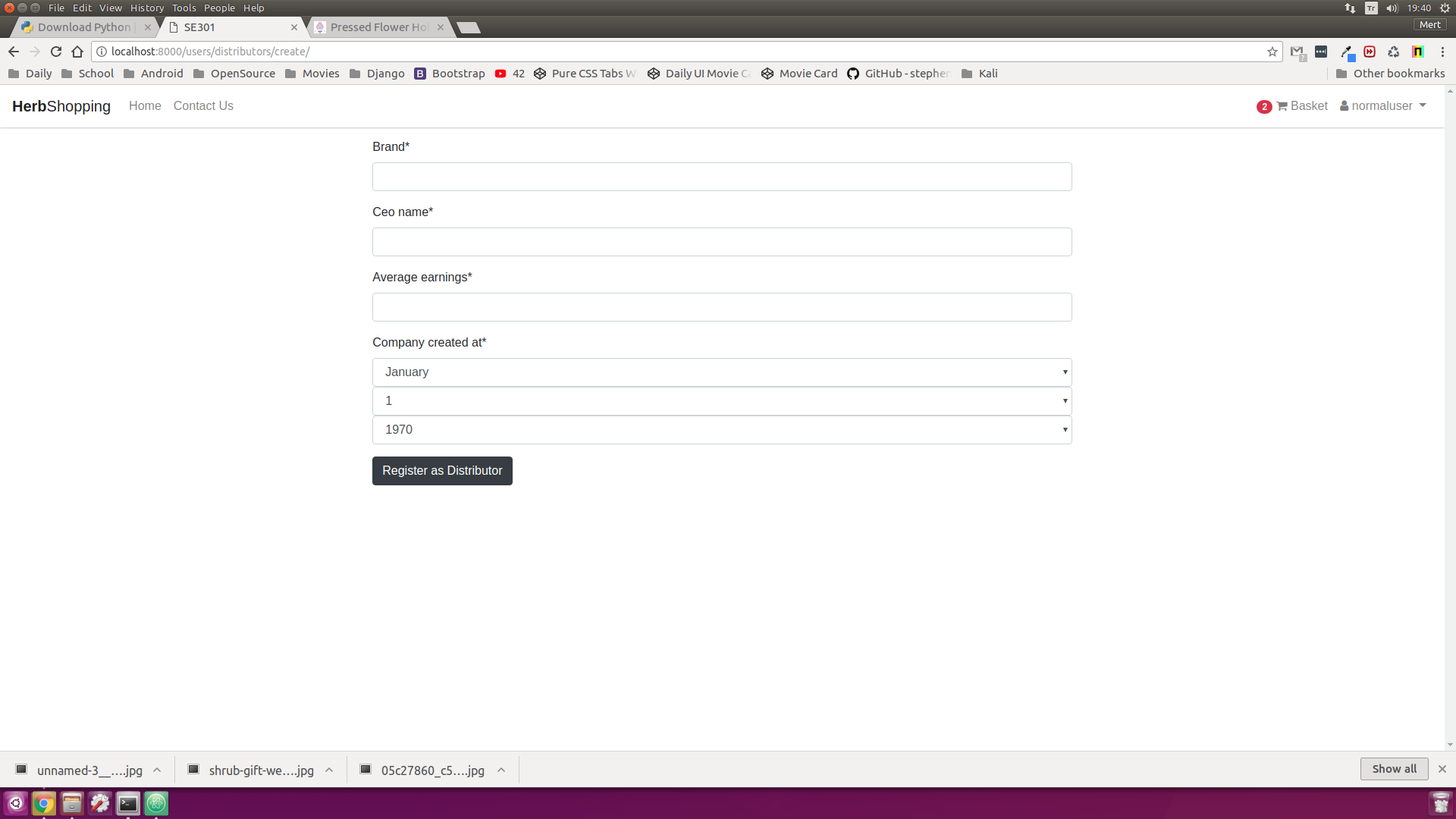
* **Distributor Interface**



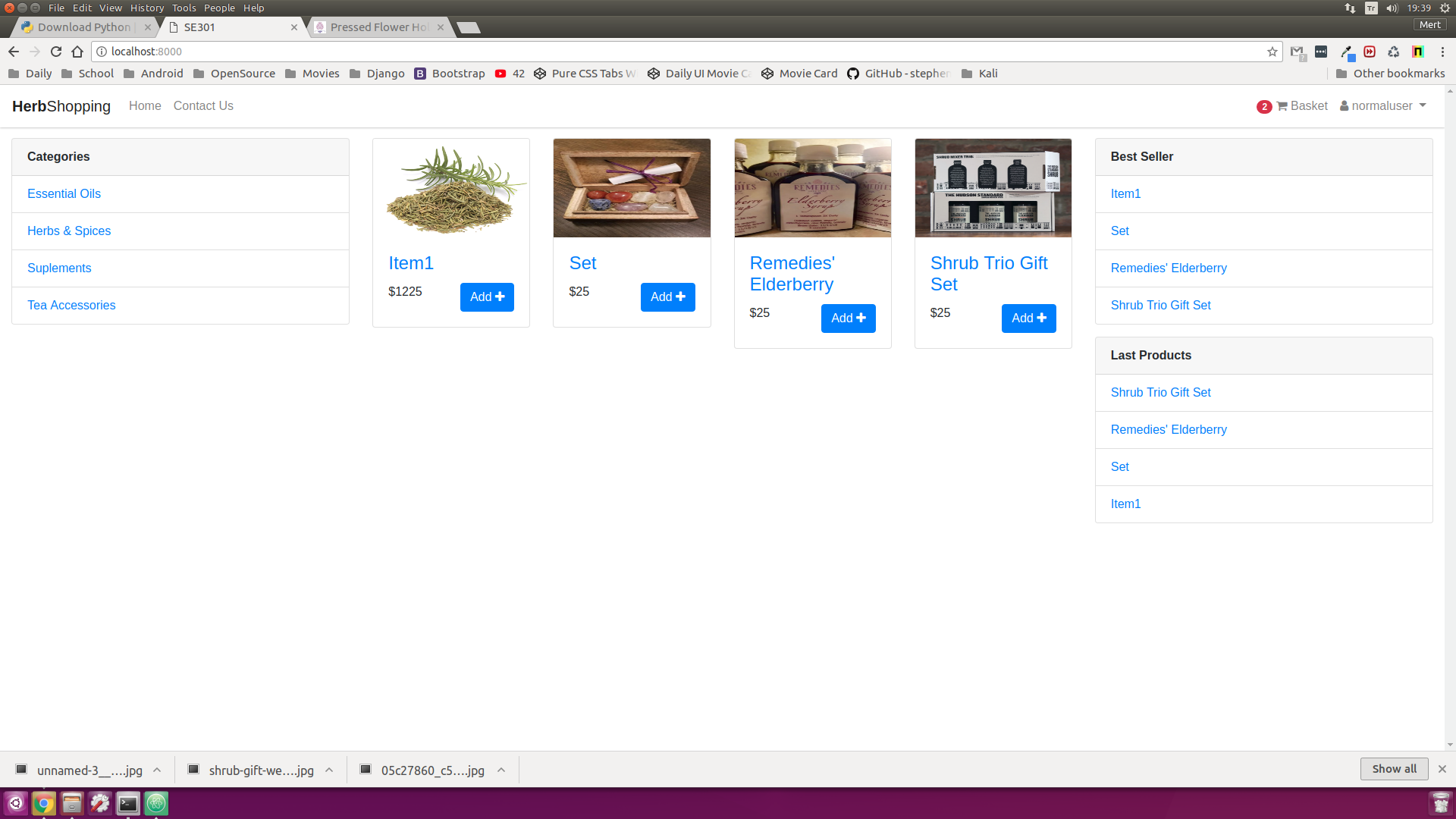
* **Items of Distributor Interface**



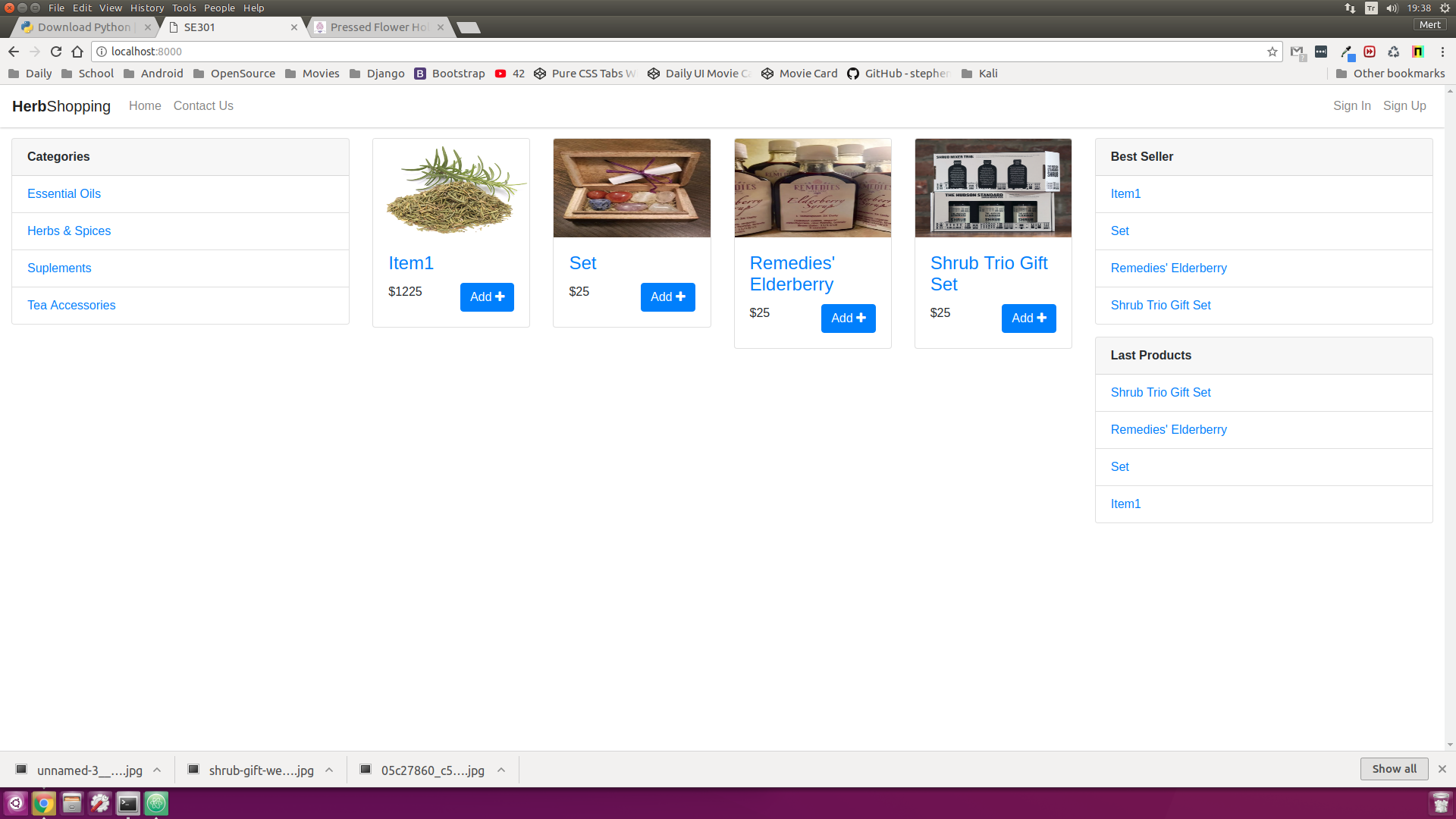
* **Distributor Register Interface**



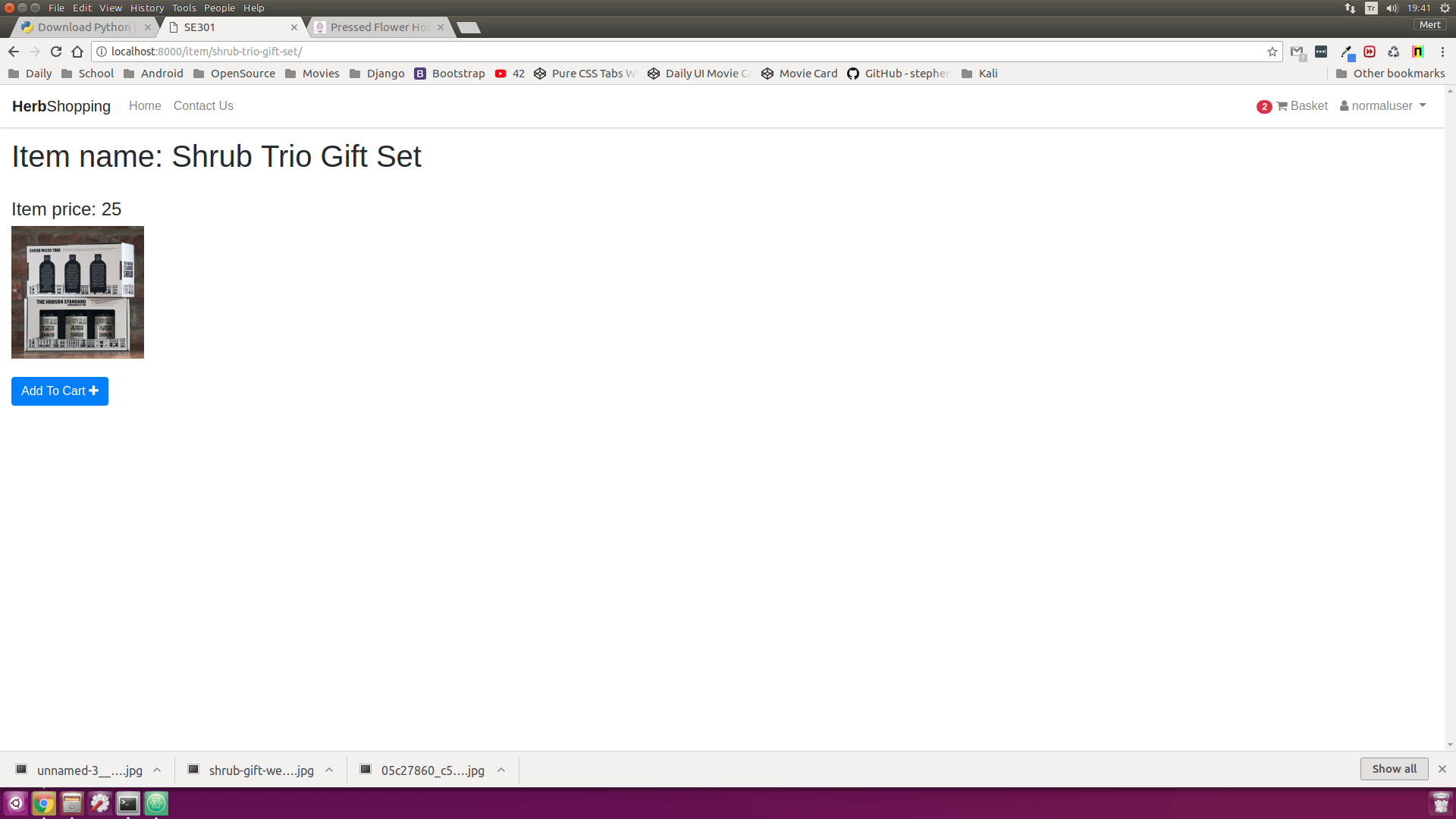
* **Web-site Main Page**



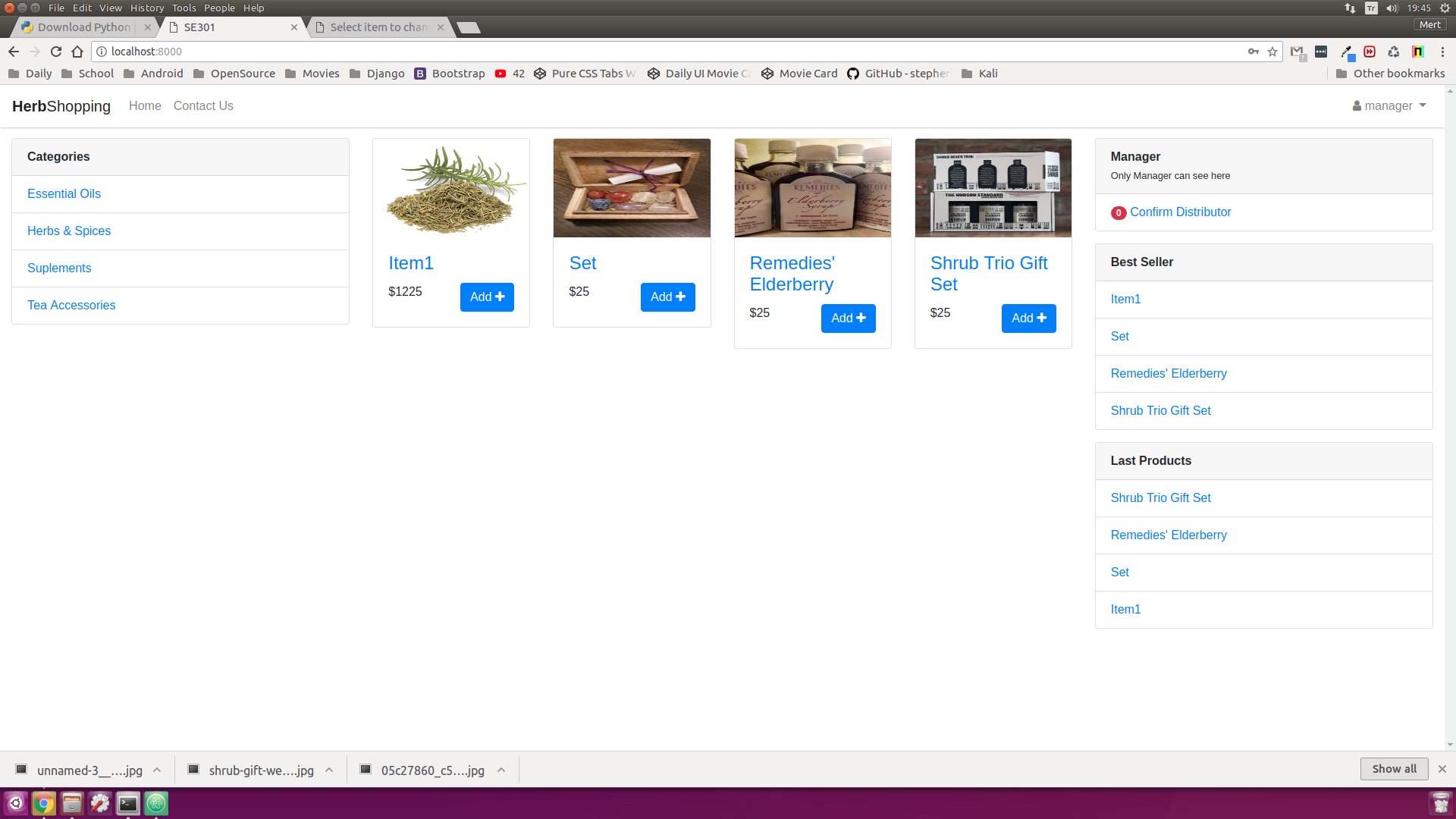
* **Main-page (Without Sign)**



* **Product Detail Interface**



* **Manager Home Page Interface**



# References

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. <https://docs.djangoproject.com/en/2.0/ref/databases/>
3. https://www.uml-diagrams.org/composite-structure-diagrams.html