HiCounselor

Build An E-Commerce Website In Django, Python and Javascript From Scratch - Module 1

About the Project:

This project you will work on is similar to amazon; it is an E-Commerce website for vendors to sell their products with one click and get customer traction easily. It helps customers deliver the item to their doorstep at a minimal price and within a short time. This website is specified for some products; In this project, you'll learn Add To Cart and Payment Integration methods.

Project Description:

This project will help you to understand how an E-Commerce website is built in Django. In this project, we are going to work on three aspects.

1. Admin Panel
2. Customer Panel
3. Retailer Panel

Admin Panel: Admin can add the retailer to change his status, approve or reject. He can see how many retailers are there, how many are approved and how many are pending with status. An admin can monitor total deliveries, pending payout, retailer's personal information, bank details, and the retailer has selected subscription. In certain conditions, the admin can remove the retailer as well. Admin has the authority to do everything; he has permission to see all deliveries with small things like payments, history, customer information, and retailer information. He can download all the data in a CSV file. The main functionality is adding items in the Item table, generating a promo code for retailers, and adding the sub-category for retailers. Similarly, he can update or delete the functionality.

__Customer Panel:__ Customers can access all of the products added by the admin. He can create his account. He does not require permission from the admin or retailer to sign up. He can add the product to the cart and buy the products. There is functionality for online payment which we did through third-party API. At one time, customers could only buy products from the same retailer and not from others. If the customer declines the previous cart or successfully buys products, he is only allowed to buy from other retailers.

__Retailer Panel__: The retailer has the choice of selling products. The retailer can add business hours for customers when the product is available. They can update their bank account and see the payment history of items sold. The data can be downloaded in CSV format. They can manage the delivery of items requested by customers, see all the information on orders, take action on it, and see the total revenue generated through delivery. On the dashboard page, they can see customer count and repeated customers and have a filter to check monthly details.

Prerequisites for the module (mandatory)

1. **__Languages that you have to use:__**

Sr no	**Language**	**Version**
1	Python	version- 3.10.2
2	Javascript	version- ES6
3	Jquery	version- 3.3.1
4	HTML5	

5	CSS3	
6	Mysql for database	
7	Bootstrap5	

Framework you can use:

1) Django, version- 4.0.3

MODULE 1 (Week 1):

Customer Panel: Customers can access all of the products which the admin adds by creating their own account. He does not require any permission from the admin or retailer. He works independently through all the processes. He can add the product to the cart and buy the product. There is functionality for online payment which is done by a third-party API. At the time, customers could only buy products from the same retailer, not from others. But when he removes them or buys the product from the cart then only he is allowed to buy from other retailers.

Tasks Need To Be Completed In This Module:

Task 1: Explanation to sign in the process

In this task, you have to develop a sign-in process with the database module and the coding.

- You must use HTML, CSS, JavaScript, JQuery and Django views functions.
- With HTML syntax, you only show the fields as UI.
- In JavaScript, you can get the value according to the id of each field.
- jQuery is useful for calling the ajax() function from where we send our parameter as we took in Javascript with some endpoint as URL.
- URL calls the views to function where we write business logic for our function. Here you can filter the respective data if it is available in the database, if not then we save it to the database.

Task 2: Explanation of the Sign-up process:

You need to develop a signup page in this task.

- you will have to find the interlink of pages, learn to maintain big data and save all the data to create a new entry in the database.
- It includes HTML, CSS, JavaScript, jQuery, and Django views functions.
- By the HTML syntax, we only show the fields as UI.
- In JavaScript, we get the value according to the id of each field. Here jQuery is useful for calling the ajax() function, from where we send our parameter as we took in JavaScript along with some endpoint as URL.
- URL calls the views to function where we write business logic for our function.
- Here, you can filter the respective data if it is available in the database; if not, then save it to the database. This is how the Sign Up functionality works.

Task 3: Explanation of Forgot Password Process:

This task explains the "Forgot Password" process with the coding. Assume you have lots of accounts here and there, but your email id or mobile number is similar or limited. You decided on your password accordingly, and you forgot the password; then how will you retrieve it if you don’t know any option to get it and your important data was there.

- Here is the solution for you - this task consists of HTML, CSS, JavaScript, jQuery, and Djangoviews functions.
- By the HTML syntax, we only show the fields as UI in JavaScript we get the value according to the id of each field.
- Here jQuery is useful for calling the ajax() function from where we send our parameter as we took in JavaScript, with some endpoint as URL.
- URL calls the views to function where we write business logic for our function.
- Here, we filter the respective data if it is available in the database. If not, then save it to the database. This is how the Forgot Password functionality works.

Task 4: Explanation to Django Admin.

This task will explain the Django admin which is complementary given by the Django framework to handle the database easily. In this task, you will have the database we include in our model file. Here we can add, update, get and delete an entry without permission. It allows us to do that but for this, you must be admin. Admin panel is a simple way to use for backend data management by which we can do the CRUD operations.
CRUD operations mainly consist of Add, Update, Get, and Delete functions. For the authentication process, we required a superuser, and for that we

created it by the Createspace command. It will allow you to log in and have the authority to change databases.

Task 5: <u>How to Home Page introduction</u>

In this task, you will be working on creating a Homepage for the Customer panel.

MODULE 2:

Admin Panel: In this module, you will be setting up an Admin Panel for the e-commerce website.

The Admin can add the retailer to the admin panel. They can change their status, approve or reject, check how many retailers are present and get an idea of useful retailer metrics, how many are approved and how many have a pending status. Admin can see the total deliveries and the pending payout, update retailer’s personal information, bank details, and retailer subscription. The main functionality is to add items in the Item table, generate a promo code for retailers and add sub-categories for retailers. Similarly, an admin can also update or delete the functionality.

Tasks To Be Completed In This Module:

Task #1: <u>Dashboard and Sign-In process creation:</u>

Here you will be creating the Sign - in and dashboard admin functionality. The HTML syntax contains the input fields to which we specified the id to get the value of each field.
It comprises HTML, CSS, JavaScript, jQuery, and Django views functions. By the HTML syntax, you will only show the fields as UI. In JavaScript, we get the value according to the id of each field. Here jQuery is useful for calling the ajax() function, from where you will send your parameter as we took in JavaScript, with some endpoint as URL. URL calls the views to function, where we write business logic for our function. Here we filter the respective data to determine if it is in the database or not. If not, then save it to the database. This is how you will be creating the Sign In functionality.

Task #2: How to Add a Retailer Process?

In this task, you will be working on setting up the retailer process that would show the information of all retailers added by the admin. Retailers cannot add themself. This permission is only for the admin. He is superior to both retailers and customers. In this task, you have to perform the CRUD operation related to the retailers.
It will comprise HTML, CSS, JavaScript, jQuery, and Django views functions. By the HTML syntax, You have to show the fields as UI. In JavaScript, you will get the value according to the id of each field. Here jQuery is useful for calling the ajax() function, from where we send our parameter as we took in JavaScript, with some endpoint as URL. URL calls the views to function, where you have to write business logic for the function. Here you will find out the data concerning the request. Here is where you will get the retailer details.

Task #3: How the Delivery process works:

This is all about the active and pending deliveries seen by the admin. In this task, you will filter from date to date, Download the CSV option to the admin user, and create access to see all items' deliveries and statuses. This part would comprise HTML, CSS, JavaScript, jQuery, and Django views functions. By the HTML syntax, we only show the fields as UI. In JavaScript, we get the value according to the id of each field. Here jQuery is useful for calling the ajax()function, from where you will be sending your parameter as we took in JavaScript, with some endpoint as URL. URL calls the views to function, where you will write business logic for the function. Here you will find out the data concerning the request. Here you will get the delivery details.

Task #4: How does the Payout Process Work?

You will be working on setting up the payment system for this particular task. Here, you will create filters by date, to date and addition with filters by status and by store name. In this task, the admin will make payments to the customer by removing the commission from the payment.
This would comprise HTML, CSS, JavaScript, jQuery, and Djangoviews functions. By the HTML syntax, you will only show the fields as UI. In JavaScript, you will get the value according to the id of each field. Here jQuery is useful for calling the ajax() function, from where we send our parameter as we took in JavaScript, with some endpoint as URL. URL calls the views to function, where we write business logic for our function. Here we find out the data concerning the request. Here you will receive the payout details. Here you have to call the third-party API for payment integration.

Task #5: Creation and Setting up of Log out process**.**

This task combines the Category, sub-category, and promo code sections. Also, you will be working on the creation of the functionality of logout. In this task, you will see how the admin can add, update, and delete the category, subcategory and promo code for the retailer. Once the user logs out from his own account, he will not access any of these functions.
This task comprises HTML, CSS, JavaScript, jQuery, and Django views functions. By the HTML syntax, you will only show the relevant fields as UI. In JavaScript, we get the value according to the id of each field. Here jQuery is useful for calling the ajax() function, from where we send our parameter as we took in JavaScript, with some endpoint as URL. URL calls the views to function, where we write business logic for our function. Here we find out the data concerning the request. Here you will create the pathway on how the Admin can add the category, sub-category, and promo code for referral use.

MODULE 3:

Retailer Panel: In this module, you will be working on setting up a Retailer Panel for the e-commerce website.

The retailer has the option of selling the products (anything and everything). They can customize customer business hours based on product availability, update bank account information, and view payment histories for sold items. To keep track of records, all of this data can be downloaded in CSV format. They can manage item delivery, review all orders placed, take action on them, and manage the total revenue generated by delivery. Any retailer can see his own customer count, repeat customers, and a filter to see monthly details on the dashboard page.

Tasks Need To Be Completed In This Module:

Task #1: Process of signing in and creating a profile page

Here you will look at the Sign-In and Profile page functionality of Retailers in this task. We have thoroughly explained the functionality using logic and coding. The most important point here is that there are three conditions with the same function, and the response is different for minor changes. Retailers can add and update their business hours as needed.

It comprises HTML, CSS, JavaScript, jQuery, and Django view functions in this section. The HTML syntax only allows you to display the fields as UI. In JavaScript, we obtain the value based on the ID of each field. jQuery is useful for calling the ajax() function, from which we send our parameter from JavaScript, with some endpoint as URL. The URL calls the views function, where we write our function's business logic. If the data is available in the database, it is filtered here; otherwise, it is saved to the database. This is how the Sign In feature works.

Task #2: How does the retail delivery page work?

Here you will learn how information is displayed about all deliveries related to the retailer, such as the order requested/canceled by the customer. Retailers have the authority to manage data and download data in CV format for internal use.

It will comprise HTML, CSS, JavaScript, jQuery, and Django view functions in this task. The HTML syntax only allows you to display the fields as UI. In JavaScript, we retrieve the values based on the id of each field. In this case, jQuery is useful for calling the ajax() function, from which we send our parameter from the JavaScript with some endpoint as URL. The URL calls the views function, where we write our function's business logic. We gather information about the request and obtain delivery information.

Task #3: Retail Inventory Page Creation

This task will teach you how the retailer adds or removes inventory. The retailer does not have the ability to categorize the products, but he can select and remove items from the panel if they are not available.

This task will include HTML, CSS, JavaScript, jQuery, and Django view functions. The HTML syntax only allows us to display the fields as UI. In JavaScript, we obtain the value based on the ID of each field. In this case, jQuery is useful for calling the ajax() function, from which we send our parameter from the JavaScript with some endpoint as URL. The URL calls the views function, which is where we write our function's business logic.

Task #4: How to Create the Dashboard and Logout Page

This task will teach you how to use the dashboard, change your password, and log out. In this task, we will examine how the customer count is displayed on the retailer's dashboard. Any customer who joins today will be considered a new customer, and anyone who returns will be considered a returning customer.
This task includes HTML, CSS, JavaScript, jQuery, and Django view functions. The HTML syntax only allows us to display the fields as UI. In JavaScript, we obtain the value based on the ID of each field. In this case, jQuery is useful for calling the ajax() function, from which we send our parameter from the JavaScript with some endpoint as URL. The views function is called by the URL. The time span of all days from Monday to Sunday is updated here. Retailers can control delivery opening and closing times.

Live Project Submission Guidelines, Instructions & Prerequisites:

1. Go thoroughly through the project description and guidelines.
2. Start developing/ writing your coding on any **Integrated Development Environment(IDE)** or **Visual Studio and Notepad++** you are familiar with per the project prerequisites mentioned above. The prerequisites are mandatory for any project submission not following the guidelines and instructors will be disqualified.

Weekly Progress Submission Process:

3. Once you have completed the code, you must create two files separately:
    a. a ZIP file of the "readme version " of the code for module 1.
    b. A video of the final UI based on the code written for all tasks in module 1
    c. Share both files in the google form here and submit the form for evaluation.

Evaluation Criteria:

1. The project submission guidelines and prerequisites are mandatory and non-negotiable.
2. The submissions that are completed will be considered for evaluation.

Session Schedule & Dates:

Day	Date	Time	Session
Monday	**21st November**	**6:00 PM PST**	**Project Kickoff Call/ Introduction to Module 1**
Friday	**25th November**	**6:00 PM PST**	**Doubt Solving Session for Module 1**
Thursday	**1st December**	**6:00 PM PST**	**Module 1 Submission**
Thursday	**1st December**	**6:00 PM PST**	**Introduction to Module 2 and Module 1 Solution Analysis**
Thursday	**21st December**	**6:00 PM PST**	**Introduction to Module 3**

*Any changes in the dates and time of the sessions will be informed in advance.

Important Links related to the Live Project

1. **Live E-commerce Kick-Off Session by Project team:** Click here
2. **Module 1,2 & 3 Submission form**: click here once you have completed modules separately. You can use the same link to make your submissions.
3. **If you have any doubts or questions**, you can add them to the google form here. Our team will answer all your doubts and queries regarding the project.

4. **Module 2 Introduction session scheduled** for 1st Dec, Thursday, 6:00 PM: click here.
   Register on this link and join live for the session.
5. **Introduction to Module 2 Session and Doubt Clearing session**: Click here.
6. **Introduction to Module 3 Session** : Click here