

# ONLINE E-COMMERCE-FLASK PL2

IIHT

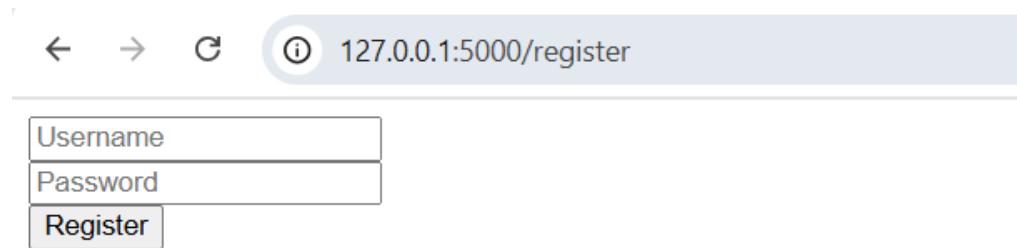
Time To Complete: 3 hours

# 1 PROBLEM STATEMENT

---

**Ecommerce** is a Flask web application which allows users to manage a list of shopping items and their associated price . The application demonstrates different HTTP methods and data input types including route variables, query strings, and JSON payloads and creating Database .

### Register.html



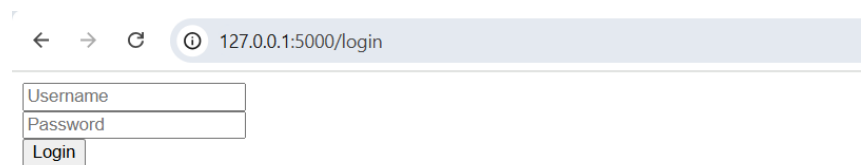
← → ↻ ⓘ 127.0.0.1:5000/register

Username

Password

Register

### Login .html



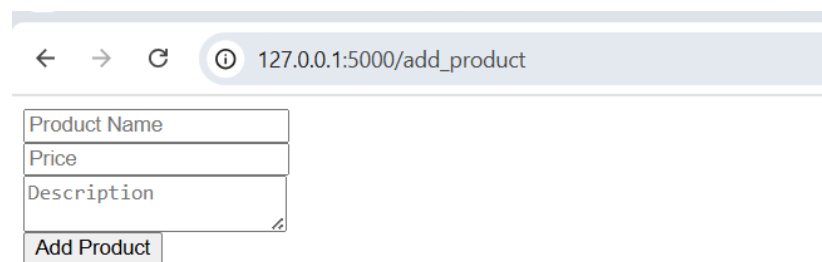
← → ↻ ⓘ 127.0.0.1:5000/login

Username

Password

Login

### Add\_product.html



← → ↻ ⓘ 127.0.0.1:5000/add\_product

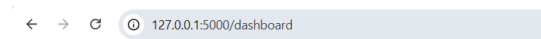
Product Name

Price

Description

Add Product

## Dashboard.html



### Welcome to Your Dashboard

#### Product List

Name	Price (₹)	Quantity	Description	Actions
Eggs	50.0	40	Basket of eggs	<a href="#">Edit</a>   <a href="#">Delete</a>
Pen	50.0	20	Box of pens	<a href="#">Edit</a>   <a href="#">Delete</a>
salt	100	60	Pack of salt	<a href="#">Edit</a>   <a href="#">Delete</a>
Mint	10.0	20	Pack of mint	<a href="#">Edit</a>   <a href="#">Delete</a>
Lays	10.0	10	Pack of lays	<a href="#">Edit</a>   <a href="#">Delete</a>

Total Cart Value: ₹220.0

[Add New Product](#) | [Logout](#)

User Story	User Story Name	User Story
Us_01	Ecommerce app	As a user, you should be add the products edit the product update the product and create require data in the database .

### The Database to be created using SQL lite

The upload the data you need to use the command init\_db.py the samples data will be loaded in to the database .

#### PRODUCT LIST

Name	Price (₹)	Quantity	Description	Actions
Eggs	50.0	40	Basket of eggs	<a href="#">Edit</a>   <a href="#">Delete</a>
Pen	50.0	20	Box of pens	<a href="#">Edit</a>   <a href="#">Delete</a>
salt	100	60	Pack of salt	<a href="#">Edit</a>   <a href="#">Delete</a>
Mint	10.0	20	Pack of mint	<a href="#">Edit</a>   <a href="#">Delete</a>
Lays	10.0	10	Pack of lays	<a href="#">Edit</a>   <a href="#">Delete</a>

Total Cart Value: ₹220.0

Functionality	Route	Method(s)	Details
Display homepage	/	GET	Renders <code>index.html</code> .
User registration	/register	GET, POST	GET renders <code>register.html</code> ; POST saves new user to <code>users</code> table.
User login	/login	GET, POST	GET renders <code>login.html</code> ; POST authenticates and starts user session.
User dashboard	/dashboard	GET	Shows product list and total value if logged in.
Add product	/add_product	GET, POST	GET renders <code>add_product.html</code> ; POST inserts new product.
Edit product	/edit_product/<int:id>	GET, POST	GET shows product in form; POST updates the product by ID.
Delete product	/delete_product/<int:id>	GET	Deletes product by ID and redirects to dashboard.
Get products via API	/api/products	GET	Returns all products in JSON format.
Add product via API (JSON)	/api/product	POST	Adds a product from JSON input; returns JSON success response.
User logout	/logout	GET	Logs out user and redirects to login page.

#### Resources views

Template File	Description	Status
<code>index.html</code>	The Home page for the eCommerce application	Already implemented
<code>register.html</code>	User registration page for creating a new account	Already implemented
<code>login.html</code>	Login page for registered users	Already implemented
<code>dashboard.html</code>	Admin dashboard displaying product listings and inventory value	Already implemented
<code>add_product.html</code>	Page to add a new product to the store	Already implemented

Template File	Description	Status
edit_product.html	Page to edit details of an existing product	Already implemented

In App.py

def index():	Already implemented
--------------	---------------------

#### Functions to be implemented

def get_db_connection():	Steps to be implemented <ul style="list-style-type: none"> <li>Implement database connection</li> <li>connect to 'store.db' the database name should be the same</li> </ul>	To be implemented
def register():	<ul style="list-style-type: none"> <li>The User need to create User registration</li> <li>For GET: Should render register.html template</li> </ul> For POST: Should get username and password from form The default username and password is 'admin' and 'admin123' this should be used as the same	To be implemented
def login():	<ul style="list-style-type: none"> <li>For GET: Should render login.html template</li> <li>For POST: Should get username and password from form</li> <li>Should validate credentials against users table</li> <li>Should set session['user_id'] on successful login</li> </ul> Should redirect to dashboard on success	To be implemented
def dashboard():	<ul style="list-style-type: none"> <li>Should check if user is logged in (session['user_id'])</li> <li>Should redirect to login if not authenticated</li> <li>Should fetch all products from database</li> <li>Should calculate total cart value (SUM of all product prices)</li> <li>Should render dashboard.html with products and total_value</li> </ul> Total value should be > 150 for sum of price of products	To be implemented
def add_product():	<ul style="list-style-type: none"> <li>Should check if user is logged in</li> <li>For GET: Should render add_product.html template</li> </ul>	To be implemented

	<ul style="list-style-type: none"> <li>For POST: Should get name, price, description from form</li> <li>Should insert new product into products table</li> </ul> <p>Should redirect to dashboard after successful addition</p>	
def edit_product(id):	<ul style="list-style-type: none"> <li>Should check if user is logged in</li> <li>For GET: Should fetch product by id and render edit_product.html</li> <li>For POST: Should update product with new name, price, description</li> <li>Should redirect to dashboard after successful update</li> </ul> <p>The user needs to update the price of salt to 100.0</p>	To be implemented
def delete_product(id):	<ul style="list-style-type: none"> <li>Should check if user is logged in</li> <li>Should delete product with given id from products table</li> <li>Should redirect to dashboard after successful deletion</li> <li>The product to be delete is mentioned in the below table</li> </ul>	To be implemented
def get_products():	<ul style="list-style-type: none"> <li>Should fetch all products from database</li> <li>Should return JSON array of products</li> <li>The URL use for GET function is /api/products</li> </ul>	To be implemented
def add_product_json():	<ul style="list-style-type: none"> <li>Should get JSON data from request</li> <li>Should validate required fields: name, price, quantity, description</li> <li>Should insert product into database</li> <li>The product to be inserted in mentioned below</li> <li>POST URL is /api/products</li> </ul>	To be implemented
def logout():	<ul style="list-style-type: none"> <li>Should remove current 'user_id' from session</li> <li>Should redirect to login page</li> </ul>	To be implemented

**Note delete\_product(id):**a product you need to delete only the mentioned details make the testcase run successfully

Name of the product	Price	Quantity	Description
Gum	10	10	Pack of gum

**You need to add an item (lays) to DB with Postman JSON data**

Name of the product	Price	Quantity	Description
Lays	10	20	Pack of gum

The USER IS required to use the POST MAN client to use the GET AND POST request

## 2 MANDATORY ASSESSMENT GUIDELINES

---

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. The editor Auto Saves the code.
4. If you want to exit(logout) and to continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.
7. To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.
8. Install 'flask' module before running the code. For this use the following command.  
`pip install flask`
9. Use the following command to  
run the server `python3 app.py`
10. Mandatory: Before final submission run the following commands to  
execute testcases `python3 -m unittest`
11. To test rest end points  
Click on 'Thunder Client' or use Ctrl+Shift+R ->Click on 'New Request' (at left side



of IDE)

12. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.