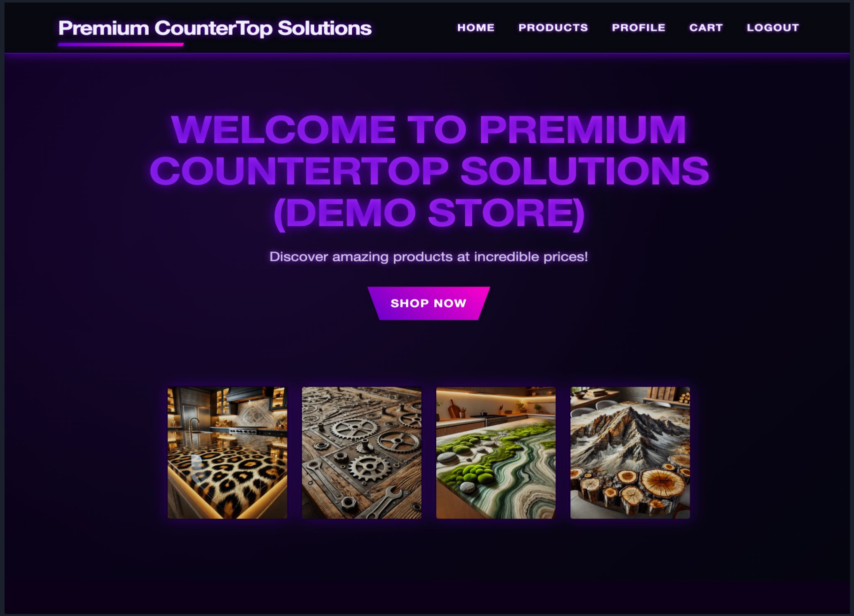
Kitchen Counter-Inventory Tracking Software

Group 5:
Nicole Cosmany
Stephen Tynan
Cole Akers
Patrick Ridley
Evan Embry

Vision and Description



Premium CounterTop Solutions

HOME PRODUCTS PROFILE CART LOGOUT

WELCOME TO PREMIUM COUNTERTOP SOLUTIONS (DEMO STORE)

Discover amazing products at incredible prices!

SHOP NOW

Interactive inventory tracking software for both customers and admin

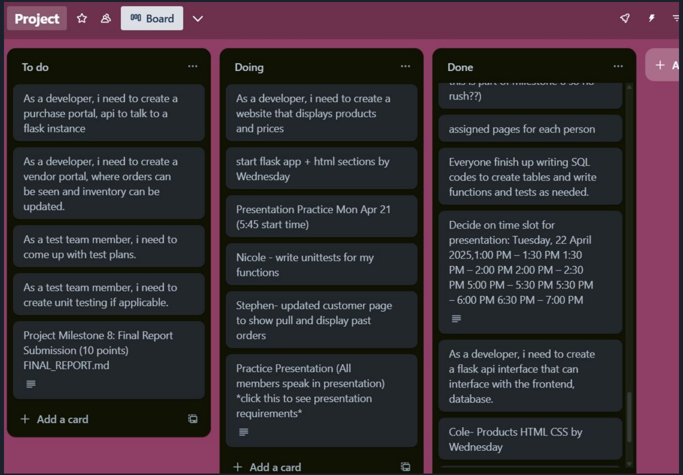
Description:

- Customers can browse products, login/logout, add items to cart, view their profile page, and checkout
- Admins can manage product inventory and view orders

Tools Overview

Tool	Purpose	Evaluation
Git + GitHub	Version control & collaboration	Effective, easy to use, can be scary!
Trello	Project tracking and task assignments	Great design easy to read and use
SQLite	Database	Effective, with some collaboration.
Flask	Backend web framework	Great tool for checking the site between changes
HTML/CSS	Frontend	Great tool to easily style and create web pages
VSCode/ CSEL	Code editing	VSCode good for deploying flask without worrying about special urls
Unittest	Testing	Effective way to test functions at once
Render	Deployment	Great tool to host site easy to integrate with git

Project Management: GitHub, Trello



Project · Board

To do

- As a developer, i need to create a purchase portal, api to talk to a flask instance
- As a developer, i need to create a vendor portal, where orders can be seen and inventory can be updated.
- As a test team member, i need to come up with test plans.
- As a test team member, i need to create unit testing if applicable.
- Project Milestone 8: Final Report Submission (10 points) FINAL_REPORT.md

+ Add a card

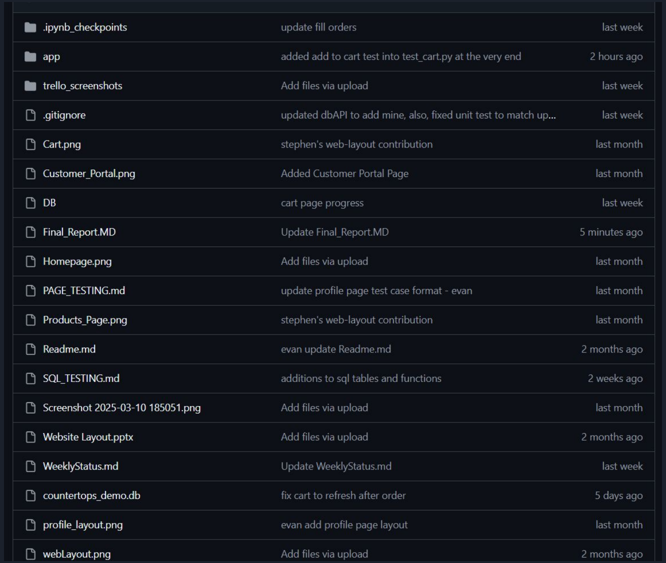
Doing

- As a developer, i need to create a website that displays products and prices
- start flask app + html sections by Wednesday
- Presentation Practice Mon Apr 21 (5:45 start time)
- Nicole - write unittests for my functions
- Stephen- updated customer page to show pull and display past orders
- Practice Presentation (All members speak in presentation) *click this to see presentation requirements*

+ Add a card

Done

- rush??)
- assigned pages for each person
- Everyone finish up writing SQL codes to create tables and write functions and tests as needed.
- Decide on time slot for presentation: Tuesday, 22 April 2025,1:00 PM – 1:30 PM 1:30 PM – 2:00 PM 2:00 PM – 2:30 PM 5:00 PM – 5:30 PM 5:30 PM – 6:00 PM 6:30 PM – 7:00 PM
- As a developer, i need to create a flask api interface that can interface with the frontend, database.
- Cole- Products HTML CSS by Wednesday



.ipynb_checkpoints	update fill orders	last week
app	added add to cart test into test_cart.py at the very end	2 hours ago
trello_screenshots	Add files via upload	last week
.gitignore	updated dbAPI to add mine, also, fixed unit test to match up...	last week
Cart.png	stephen's web-layout contribution	last month
Customer_Portal.png	Added Customer Portal Page	last month
DB	cart page progress	last week
Final_Report.MD	Update Final_Report.MD	5 minutes ago
Homepage.png	Add files via upload	last month
PAGE_TESTING.md	update profile page test case format - evan	last month
Products_Page.png	stephen's web-layout contribution	last month
Readme.md	evan update Readme.md	2 months ago
SQL_TESTING.md	additions to sql tables and functions	2 weeks ago
Screenshot 2025-03-10 185051.png	Add files via upload	last month
Website Layout.pptx	Add files via upload	2 months ago
WeeklyStatus.md	Update WeeklyStatus.md	last week
countertops_demo.db	fix cart to refresh after order	5 days ago
profile_layout.png	evan add profile page layout	last month
webLayout.png	Add files via upload	2 months ago

Trello (project tracker):

- used trello to track weekly tasks for individuals and group tasks

Github (version control):

- Worked collaboratively on github repo

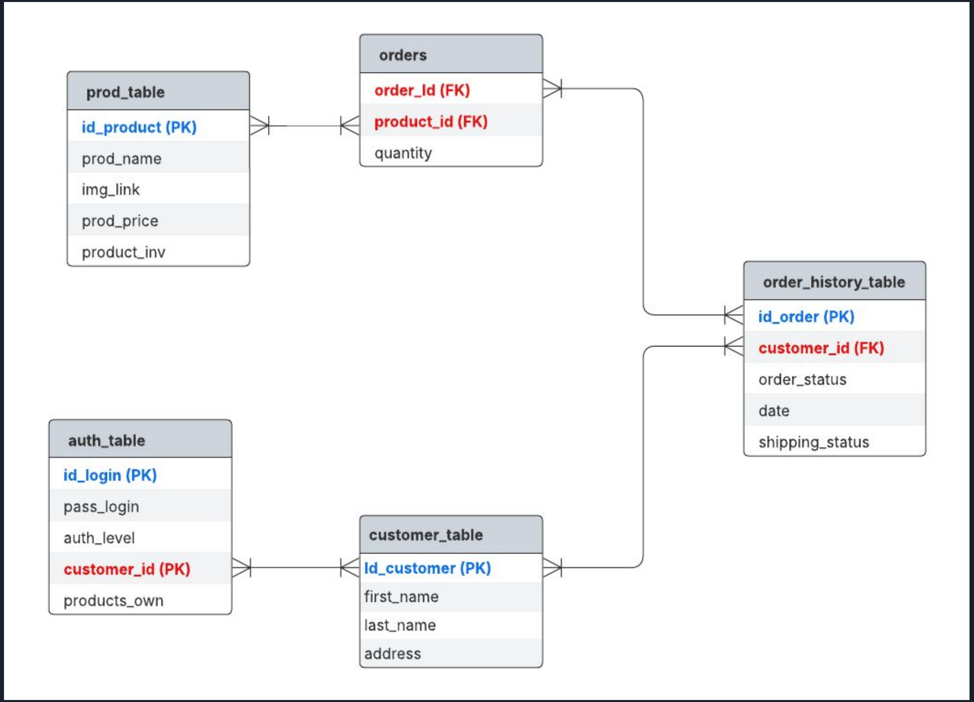
SQLite, Flask

Decision on SQLite

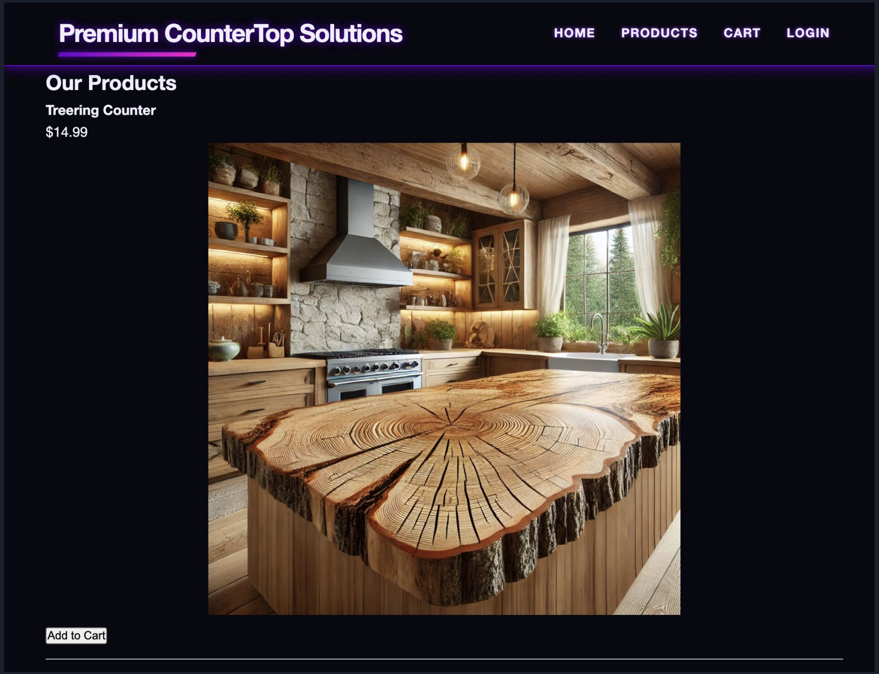
- Perfect for Demo
- Can preserved to share after class
- Easy to integrate into Flask

Flask Major Functions

- Database create (reset on Home)
- Fill demo data
- Authorization (user/login)
- Product Page
- Cart updates order history
  - Add purchases
  - Post purchase
- Profile filters on user auth level



HTML/CSS,



Our Products

Treering Counter

$14.99

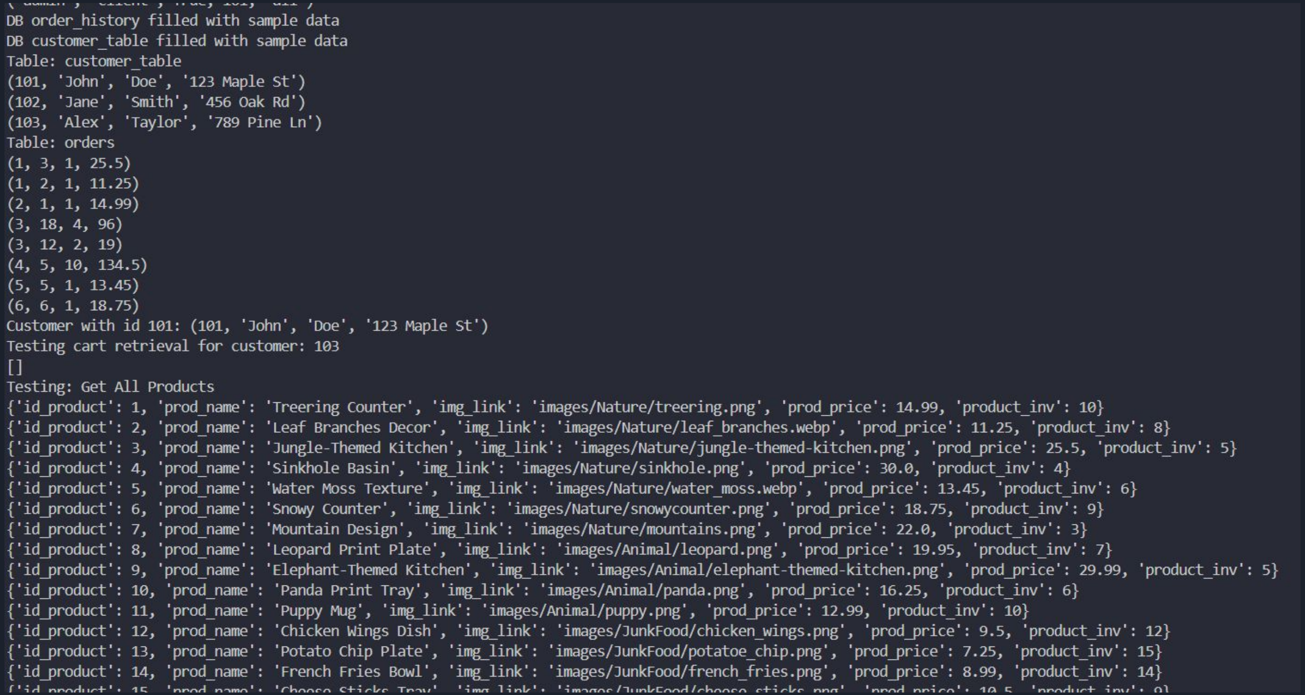
HTML:

- Structured site using elements like <header>,<main>,<section>
- Linked to Flask routes `href="{{ url_for(...) }}"`

CSS:

- Styled site with dark theme and accent purple and white colors
- Custom fonts and hover effects

Testing - Database Basics



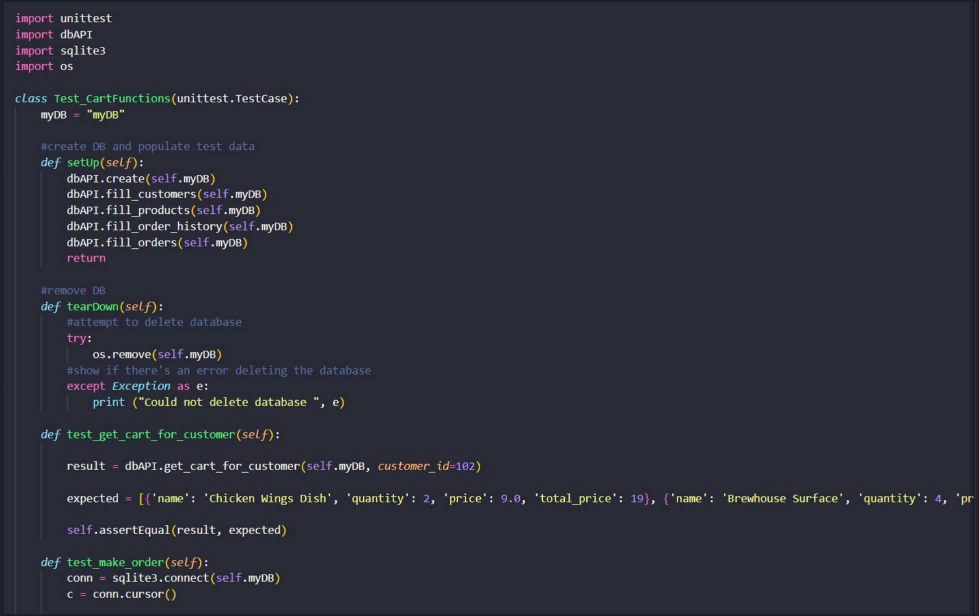
```
('admin', 'client', True, 101, 'all')
DB order_history filled with sample data
DB customer_table filled with sample data
Table: customer_table
(101, 'John', 'Doe', '123 Maple St')
(102, 'Jane', 'Smith', '456 Oak Rd')
(103, 'Alex', 'Taylor', '789 Pine Ln')
Table: orders
(1, 3, 1, 25.5)
(1, 2, 1, 11.25)
(2, 1, 1, 14.99)
(3, 18, 4, 96)
(3, 12, 2, 19)
(4, 5, 10, 134.5)
(5, 5, 1, 13.45)
(6, 6, 1, 18.75)
Customer with id 101: (101, 'John', 'Doe', '123 Maple St')
Testing cart retrieval for customer: 103
[]
Testing: Get All Products
{'id_product': 1, 'prod_name': 'Treering Counter', 'img_link': 'images/Nature/treering.png', 'prod_price': 14.99, 'product_inv': 10}
{'id_product': 2, 'prod_name': 'Leaf Branches Decor', 'img_link': 'images/Nature/leaf_branches.webp', 'prod_price': 11.25, 'product_inv': 8}
{'id_product': 3, 'prod_name': 'Jungle-Themed Kitchen', 'img_link': 'images/Nature/jungle-themed-kitchen.png', 'prod_price': 25.5, 'product_inv': 5}
{'id_product': 4, 'prod_name': 'Sinkhole Basin', 'img_link': 'images/Nature/sinkhole.png', 'prod_price': 30.0, 'product_inv': 4}
{'id_product': 5, 'prod_name': 'Water Moss Texture', 'img_link': 'images/Nature/water_moss.webp', 'prod_price': 13.45, 'product_inv': 6}
{'id_product': 6, 'prod_name': 'Snowy Counter', 'img_link': 'images/Nature/snowycounter.png', 'prod_price': 18.75, 'product_inv': 9}
{'id_product': 7, 'prod_name': 'Mountain Design', 'img_link': 'images/Nature/mountains.png', 'prod_price': 22.0, 'product_inv': 3}
{'id_product': 8, 'prod_name': 'Leopard Print Plate', 'img_link': 'images/Animal/leopard.png', 'prod_price': 19.95, 'product_inv': 7}
{'id_product': 9, 'prod_name': 'Elephant-Themed Kitchen', 'img_link': 'images/Animal/elephant-themed-kitchen.png', 'prod_price': 29.99, 'product_inv': 5}
{'id_product': 10, 'prod_name': 'Panda Print Tray', 'img_link': 'images/Animal/panda.png', 'prod_price': 16.25, 'product_inv': 6}
{'id_product': 11, 'prod_name': 'Puppy Mug', 'img_link': 'images/Animal/puppy.png', 'prod_price': 12.99, 'product_inv': 10}
{'id_product': 12, 'prod_name': 'Chicken Wings Dish', 'img_link': 'images/JunkFood/chicken_wings.png', 'prod_price': 9.5, 'product_inv': 12}
{'id_product': 13, 'prod_name': 'Potato Chip Plate', 'img_link': 'images/JunkFood/potatoe_chip.png', 'prod_price': 7.25, 'product_inv': 15}
{'id_product': 14, 'prod_name': 'French Fries Bowl', 'img_link': 'images/JunkFood/french_fries.png', 'prod_price': 8.99, 'product_inv': 14}
{'id_product': 15, 'prod_name': 'Cheese Sticks Tray', 'img_link': 'images/JunkFood/cheese_sticks.png', 'prod_price': 10.5, 'product_inv': 9}
```

Visually test that database tables are able to be created and populated.

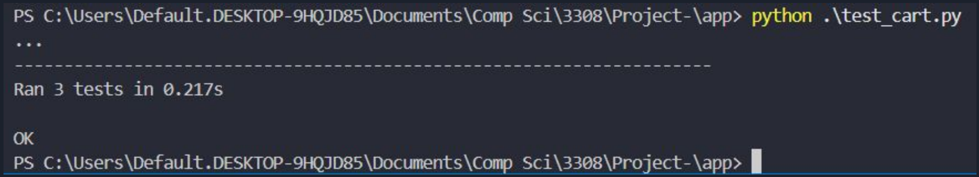
Testing - UnitTest

Unittest to test cart functions

1. Are pulling the customer's cart?
2. Can we take the cart and place an order?
3. Does the add to cart function work?



```python
import unittest
import dbAPI
import sqlite3
import os

class Test_CartFunctions(unittest.TestCase):
    myDB = "myDB"

    #create DB and populate test data
    def setUp(self):
        dbAPI.create(self.myDB)
        dbAPI.fill_customers(self.myDB)
        dbAPI.fill_products(self.myDB)
        dbAPI.fill_order_history(self.myDB)
        dbAPI.fill_orders(self.myDB)
        return

    #remove DB
    def tearDown(self):
        #attempt to delete database
        try:
            os.remove(self.myDB)
        #show if there's an error deleting the database
        except Exception as e:
            print ("Could not delete database ", e)

    def test_get_cart_for_customer(self):

        result = dbAPI.get_cart_for_customer(self.myDB, customer_id=102)

        expected = [{'name': 'Chicken Wings Dish', 'quantity': 2, 'price': 9.0, 'total_price': 19}, {'name': 'Brewhouse Surface', 'quantity': 4, 'pr

        self.assertEqual(result, expected)

    def test_make_order(self):
        conn = sqlite3.connect(self.myDB)
        c = conn.cursor()
```



```
PS C:\Users\Default.DESKTOP-9HQJD85\Documents\Comp Sci\3308\Project-\app> python .\test_cart.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.217s

OK
PS C:\Users\Default.DESKTOP-9HQJD85\Documents\Comp Sci\3308\Project-\app> 
```

Challenges and Solutions

Challenges:

- Collaborating across boundaries of responsibility
- Remote work
- Desire to split up task so we each can learn vs group sub teams into related task
    - Cause some timing issues and drift in indexes

Solutions:

- Tight communication and pre planning
- Coordinate specifics: variable names, function integration, page flow
- Discord channel help answer questions on timing and index real time.
    - At the end, unit testing uncovered any indexing issues

Reflections

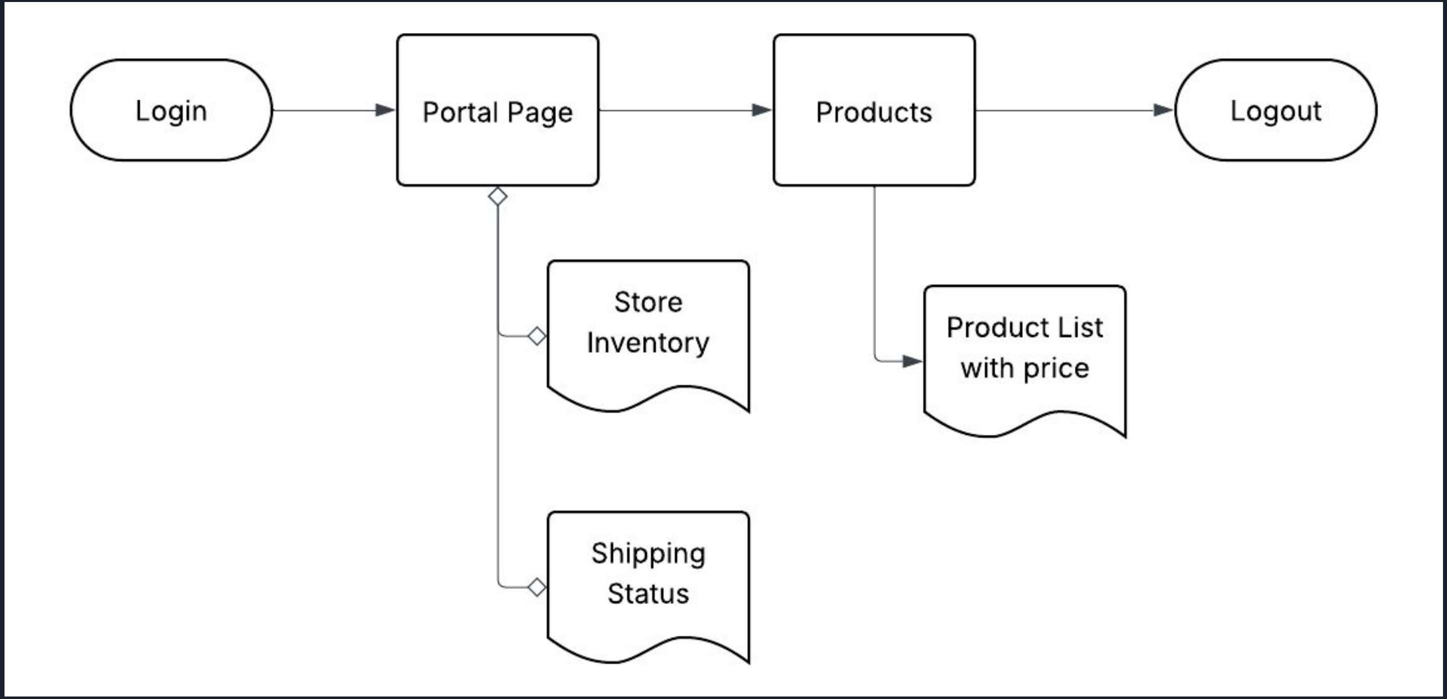
What we are proud of:

- A functional and stylish webapp
- Full customer flow-> login, view product, add to cart, place order, view past orders
- Multiple use cases - client, customer, admin

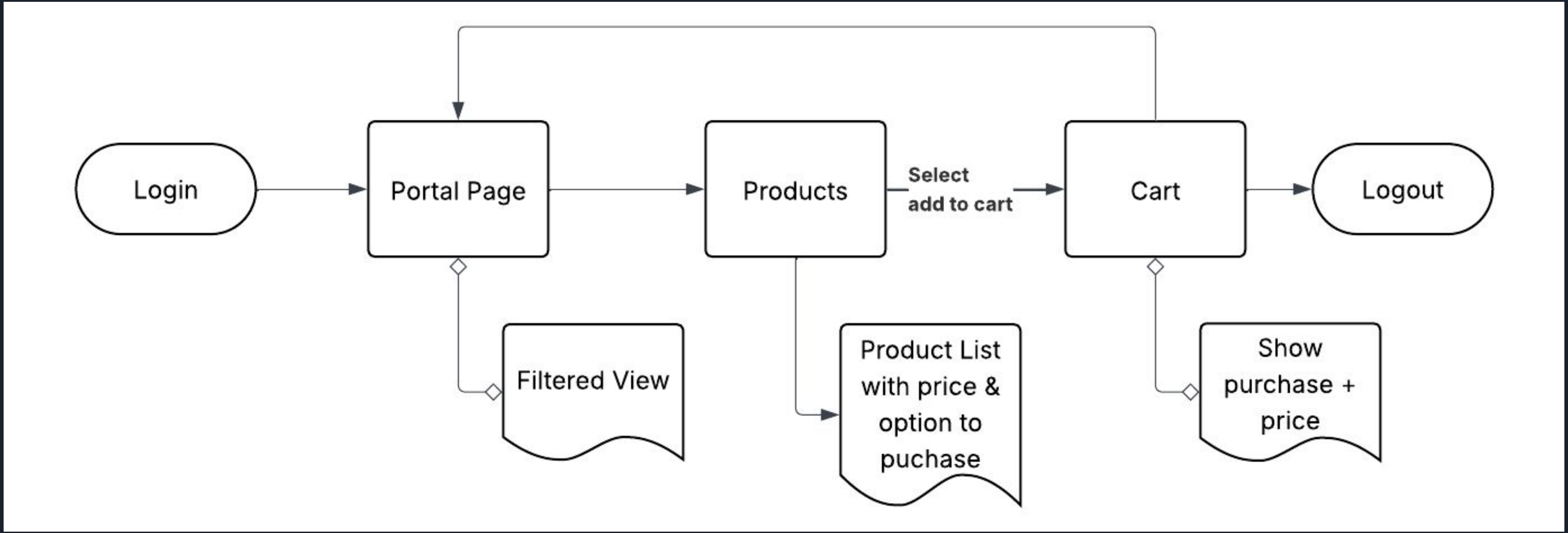
What we would do differently with more time:

- Continued styling and uniformity across pages
- Refine admin login - no cart, no product page, ability to update inventory/order status
- Create account function for new customers
- Abandoned cart feature - send email reminder

Today’s Demo (Client)



Demo on Customer



Live Demo

https://countertops.onrender.com

Questions?

Thanks for listening!

https://github.com/SoftwareGroup5/Project-

Countertop Co.