E-commerce Lab

THE LABS WILL BUILD UPON EACH OTHER. SO, THIS IS NOT AN INDEPENDENT SERIES, BUT A CONTINUOS FLOW OF WORK.

Work independently as much as possible: code plagiarism will be checked frequently

Task 1: - Database

- Download and set up the database in your local database environment.
- Database name is "shoppn"

Task 2: - Folder structure

Since we are doing separation of code (MVC) and following a modular structure to allow reuse of code, follow the folder structure below

- Admin
  - Admin controls goes here. *E.g. manage brand, manage categories, manage users, etc.*
- Actions / Functions – *this folder will handle some form actions or views or API calls*
- Classes/modules
- Controllers
- CSS – *do your own styling (free to use any css template)*
- Error – *write your own error log*
- Images
  - Product – *product images*
  - Customer – *customer image*
- JS - *feel free to use any js framework*
- Login
- Settings
  - core.php
  - db_class.php
  - db_cred.php *(no password for root user)*
- View
- index.php

You are free to add additional folders in a way that works for you

Task 3: - Register customer

- Actions / Functions
- Classes
    - cutomer_class.php – *a class that extends database connection and contains customer methods: add customer, edit customer, delete customer, etc.*
- Controllers
    - customer_controller.php – *creates an instance of the customer class and runs the methods*
- JS
    - *validate registration form using js file*
    - *Regular expressions for form validation*
- Login
    - login.php – just a php login form view for now
    - register.php – *form action will point to registerprocess.php (below) you can use js ajax for this*
        - Full name
        - Email – *a unique field. Check if email is available before adding new customer*
        - Password – *please encrypt the password in php before adding to database*
        - Country
        - City
        - Contact Number
        - Image – *This is null by default, so it does not need to be added immediately on sign up.*
        - User role – *This can be set to a default value, e.g., 1 or 2. It is meant for access control in the DB.*

      All the above are required fields in the database. Check the field length in database and validate accordingly. Validation should check for types as well: e.g. email, phone number, etc.

      There is a user role field which is also required. Do this from the SQL level. 1 is administrator, and 2 is customer

      Feel free to add a css/js loading feature when register button is clicked.

    - registerprocess.php – *capture form data, call the customer controller and run the necessary methods. Output appropriate error or redirect to login page if successful.*
- View
- Index.php
    - Menu – register | login *(Not logged in: will add more menu later)*

Due Date: 3rd March

Task 4: - Login

- Actions / Functions
- Classes
  - cutomer_class.php – *a class that extends database connection and contains customer methods: add customer, edit customer, delete customer, login, etc.*
- Controllers
  - customer_controller.php – *creates an instance of the customer class and runs the methods*
- Js
  - *validate login form using js file*
- Login
  - login.php – *form action will point to loginprocess.php (below) you can use js ajax for this*
    - Email
    - Password

    Validation should check for types as well: e.g. email, etc.

    Feel free to add a css/js loading feature when register button is clicked.

  - loginprocess.php – *capture form data, call the customer controller and run the necessary methods. Compare email and hashed password. Output appropriate error or redirect to index page if successful. Before redirect: set session for id and role. Session is checked in the core.php settings*
- View
- Index.php
  - Menu – register | logout *(logged in: will add more menu later)*

Due Date: 7th March

Task 5: - Add Brand

- Admin
  - Brand.php
    - Check if user is logged in
    - Check if user is admin
    - Only brand name. id is autogenerated in database
- Actions / Functions

    - Add_brand.php – *this processes the add brand just like login/register process. Calls the product controller class.*
- Classes
    - product_class.php – *a class that extends database connection and contains product methods: add product brand, add product category, add product, etc.*
- Controllers
    - product_controller.php – *creates an instance of the product class and runs the methods*
- Js
    - *validate brand form using js file*
- View
- Index.php
    - Menu – register | logout | brand *(you only see brand if you are an administrator)*

Due Date:12 March

Task 6: - Edit Brand

- Admin
    - Brand.php *(you can use the same add form for edit or create a new edit page. Provide appropriate navigations)*
        - Check if user is logged in
        - Check if user is admin
        - Only brand name is editable, not id
- Actions / Functions
    - update_brand.php – *this processes the update brand. Calls the product controller class.*
- Classes
    - product_class.php – *a class that extends database connection and contains product methods: update product brand, add product category, add product, etc.*
- Controllers
    - product_controller.php – *creates an instance of the product class and runs the methods*
- Js
    - *validate brand update form using js file*
- View
- Index.php
    - Menu – register | logout | brand *(you only see brand if you are an administrator)*

Due Date:14 march

Task 7: - Add Category

- Admin
  - Category.php
    - Check if user is logged in
    - Check if user is admin
    - Only category name. id is autogenerated in database
- Actions / Functions
  - Add_category.php – *this processes the add category just like login/register process. Calls the product controller class.*
- Classes
  - product_class.php – *a class that extends database connection and contains product methods: add product brand, add product category, add product, etc.*
- Controllers
  - product_controller.php – *creates an instance of the product class and runs the methods*
- Js
  - *validate category form using js file*
- View
- Index.php
  - Menu – register | logout | brand | Category*(you only see category if you are an administrator)*

Task 8: - Edit Category

- Admin
  - Category.php *(you can use the same add form for edit or create a new edit page. Provide appropriate navigations)*
    - Check if user is logged in
    - Check if user is admin
    - Only Category name is editable, not id
- Actions / Functions
  - update_category.php – *this processes the update category. Calls the product controller class.*
- Classes
  - product_class.php – *a class that extends database connection and contains product methods: update product category, add product category, add product, etc.*
- Controllers
  - product_controller.php – *creates an instance of the product class and runs the methods*
- Js
  - *validate category update form using js file*
- View

- Index.php
    - Menu – register | logout | brand | Category *(you only see category if you are an administrator)*

Due Date:18 March

Task 9: - Product Management (add & edit)

- Admin
- Actions / Functions
    - add_product.php – *this processes the add product just like add category process. Calls the product controller class.*
- Classes
    - product_class.php – *a class that extends database connection and contains product methods: add product brand, add product, edit product, etc.*
- Controllers
    - product_controller.php – *creates an instance of the product class and runs the methods*
- Images
    - Product – *product images will be stored here. E.g image_nane.png. you may face folder permission issues. Make sure the folder is read/write*
- Js
    - *validate add product form using js file*
- View – *using your own innovations, add and edit product form view goes to this folder. You can use the same form for add and edit, or use different form for add and edit. After successful add, redirect to index.*
    - Add/edit product form page file
        - Product id – *autogenerated in database during add. On edit, use value from database*
        - Product category – *a dropdown list. hint: select value is category cat_name & name/id is cat_id*
        - Product brand – *a dropdown list. hint: select value is brand brand_name & name/id is brand_id*
        - Product title
        - Product price
        - Product description
        - Product image - *image upload function. Store in database as "../images/product/image_name.png"*
        - Product keyword
- Index.php
    - Menu – register | logout | brand | Category | Add Product *(all users can see add product. But you only see after ,login)*

Due Date:25 March

Task 10: - Product Display (display & search)

- Admin
- Actions / Functions
    - product_functions.php - *you can handle all product code or search result code from here. You don't have to do it this way, but remember, modular code will be checked..*
- Classes
    - product_class.php – *a class that extends database connection and contains product methods: view all products, search products, view one product, etc.*
- Controllers
    - product_controller.php – *creates an instance of the product class and runs the methods*
- Images
- Js
- View
    - all_product.php - *a list of all products on offer and a pagination if product list is more than 10 (pagination is optional). Using your own experience with e-commerce site to style the list of all the products on the page or some products with pagination. Hint (for each product display the followings. Not in any particular order)*
        - Product id – *this will be appended to each product url for a single product view*
        - Product title
        - Product price
        - Product image
        - Add to cart – *just ad empty button/link for now*
    - single_product.php – *to view a single product in full view. The following are not in any particular order, use your own design.*
        - Product id – *you can use this behind the scene for add to cart purposes*
        - Product category
        - Product brand
        - Product title
        - Product price
        - Product description
        - Product image
        - Product keyword

        - Add to cart – *just an empty button/link for now*

    - product_search_result.php - *a list of product on offer search results and a pagination if product list is more than 10 (pagination is optional). Using your own experience with e-commerce site to style the list of all the products on the page or some products with pagination. Hint (for each product display the followings)*
        - Product id – *this will be appended to each product url for a single product view*
        - Product title
        - Product price
        - Product image
        - Add to cart – *just an empty button/link for now*

- Index.php
    - Menu – Register | All Products (list of all products) | Search Box (search product by title/name)

Due Date:1 April

Task 11: - Add to Cart

- Admin
- Actions / Functions
    - add_to_cart.php – *to process the add to cart button/link click event. Takes product id, quantity (default is 1 if not specified), client IP address, customer id (if the customer is logged in). Check for duplicate before adding to cart. If the product is already in cart, advise the user to go to cart and increase quantity.*

- Classes
    - cart_class.php – *a class that extends database connection and contains cart methods: view all products in cart, add to cart, remove from cart, etc.*
- Controllers
    - cart_controller.php – *creates an instance of the cart class and runs the methods*
- Images
- Js

- View
    - all_product.php - *a list of all products on offer and a pagination if product list is more than 10 (pagination is optional). Using your own experience with e-commerce site to style the list of all the products on the page or some products with pagination. Hint (for each product display the followings. Not in any particular order)*
        - Product id – *this will be appended to each product url for a single product view*
        - Product title
        - Product price
        - Product image
        - Add to cart – *calls add_to_cart.php and supplies the necessary data*
    - single_product.php – *to view a single product in full view. The following are not in any particular order, use your own design.*
        - Product id – *you can use this behind the scene for add to cart purposes*
        - Product category
        - Product brand
        - Product title
        - Product price
        - Product description
        - Product image
        - Product keyword
        - Add to cart – *calls add_to_cart.php and supplies the necessary data*
    - product_search_result.php - *a list of product on offer search results and a pagination if product list is more than 10 (pagination is optional). Using your own experience with e-commerce site to style the list of all the products on the page or some products with pagination. Hint (for each product display the followings)*
        - Product id – *this will be appended to each product url for a single product view*
        - Product title
        - Product price
        - Product image
        - Add to cart – *calls add_to_cart.php and supplies the necessary data*
    - cart.php - *a list of all products in cart. Using your own experience with e-commerce site to style the list of all the products in cart on the page. Provide options to continue shopping or proceed to payment. Hint (for each product display the followings. Not in any particular order)*
        - *manage quantity*
        - *remove from cart*

- Index.php
    - Menu – Register | All Products | Cart (*total items in cart*) (this page will display the list of items in the cart)

Due Date: 4 April

Task 12: - Cart Management

- Admin
- Actions / Functions
    - remove_from_cart.php – *to process the remove from cart button/link click event. Takes product id, client IP address or customer id (if the customer is logged in)*
    - manage_quantity_cart.php – *to increase/decrease quantity from cart. Takes product id, new quantity, client IP address or customer id (if the customer is logged in)*

- Classes
    - cart_class.php – *a class that extends database connection and contains cart methods: view all products in cart, update quantity, remove from cart, etc.*
- Controllers
    - cart_controller.php – *creates an instance of the cart class and runs the methods*
- Images
- Js
- View
    - cart.php - *a list of all products in cart. Using your own experience with e-commerce site to style the list of all the products in cart on the page. Provide options to continue shopping or proceed to payment. Hint (for each product display the followings. Not in any particular order)*
        - *manage quantity*
        - *remove from cart*

- Index.php

- Menu – Register | All Products | Cart (*total items in cart*) (this page displays the list of items in the cart)

Due Date:18 April

ask 12: - Payment – this will be hosted live

PayPal links:

- HTML form payment
- Code sample
- Payment Variables

Paystack : https://paystack.com/

- Admin
- Actions / Functions
    - process_payment.php – *to process paystack payment (failed or success). Based on parameters, ~~Send order details as email to customer~~, save transaction in orders table, for invoice_no use php_mt_rand(), use appropriate status message (success, inprogress, etc), save payment details in payment table, move cart to order_details table, delete customer transaction from cart and finally redirect to payment_success.php else payment_failed.php*
- Classes
    - cart_class.php – *a class that extends database connection and contains cart methods: process payment, record transaction, etc.*
- Controllers
    - cart_controller.php – *creates an instance of the cart class and runs the methods*
- Images
- Js
- View
    - cart.php - *a list of all products in cart. Using your own experience with e-commerce site to style the list of all the products in cart on the page. Provide options to continue shopping or proceed to payment. Hint (for each product display the followings. Not in any particular order)*

    - payment.php – *summary of cart (product, quantity, price & total price) for payment. Use PayStack test payment – that means the customer email must be part of the request)*
    - payment_success.php – *display success message to customer and show invoice (order details)*
    - payment_failed.php – *display payment failure to customer. This page is optional as failed payment can be handled directly from payment.php*

- Index.php

Due Date: 30 April