CSCI 466 Group Project (Spring 2025)

Web-based Store (300 pts)

Introduction

This is the group project for 466, in which you will design and develop and application using the tools covered in the class. You must work together in the groups chosen by you (before the deadline), or in groups assigned to you if you did not elect to choose your own before the deadline. There will be no individual submissions, just the one for a group.

The project will involve using most of the elements learned in the class, from designing the database schema with an ER diagram, to converting it to relations, to translating those relations into SQL DDL, and all the way to implementing the web application with PHP/PDO.

Application

The application you will be designing and implementing will be a web-based, database-driven tool that handles an online store. You will be responsible for designing and implementing both the user-facing portion of the web store and the employee-facing side.

Inventory

Your application must allow the owner of a store to look at his inventory. This will include, at minimum, a list of all of the products and how many of each of them are left.

It must also be able to showcase the products that are in stock. This includes a listing that indicates what items are available, as well as another view that allows a user of the webpage to see details on an individual product. You should be able to get to the project details page for a given project easily from the product list view.

Each of the product detail pages should include a way of adding some number of the product in question to the shopping cart.

Shopping Cart

Each user will have a shopping cart. It starts off empty and grows as users add more items to it. Many products can be in a shopping cart, and a user may intend to order than one of a given product. There must be a page where a user can view the current contents of their shopping cart.

In this page, the user should have an option to remove things from the cart, or to change the quantity that they will be ordering.

In order to do this, you will need to store some kind of state information. There are many ways to do this, including writing info to the database, hidden form elements, and session variables. I am not requiring any specific approach, but it does need to work.

Once the order has been submitted, the items ordered should be associated with the order, so they can be looked up later. The shopping cart should reset to empty once the order is complete.

The page showing the cart should have a link to another page that allows them to check out.

Checking out

The checkout page should allow the user to enter a shipping address, and billing information (see the note below). It will show them, at a minimum, the total value of the items in their cart, and allow them to finish submitting an order.

Note: AT NO POINT SHOULD ANY ACTUAL CREDIT CARD INFORMATION BE ENTERED. MAKE UP FAKE DATA AND KEEP YOUR BILLING INFO SAFE.

Orders

When an order is placed, the info on the order needs to be placed into the database. When an order goes into the system, it will be marked as "Processing". This will be changed as the store workers fulfill orders. Your application will need it to generate three different views:

1. Order tracking for the user. The user should be able to visit a URL within your site and see information on their order and its current status. This would include things like whether the order has been processed, shipped, etc., as well as info on tracking numbers for shipping. The amount paid for each order should be a part of this view, as well as the total amount paid for all orders.
2. A list of all of the outstanding orders, which the employees of the store can use as a guide to know which items still need to ship.
3. An order fulfillment page that allows store employees to see details on individual orders, and mark them as shipped, add notes, contact the user, etc.

Allowed tools

Obviously, any of the tools that have been covered in class are acceptable. This mostly means using our MariaDB server (an absolute requirement), and PHP/HTML forms/PDO.

Each semester, I have a lot of people asking me if they can use other tools, such as JavaScript/CSS/etc. The answer I give is that I do not have a problem with the use of such tools, as long as they're not a shortcut that allows you to skip the work of actually doing the project.

This means that any of the code implementing requirements detailed in this write-up must be designed and written by your group. Obviously this would not be the case if you just downloaded and included a library that already did the entire project.

However, outside of the main functionality, it is acceptable to use libraries. Things like Bootstrap, React.js, etc. are acceptable, as long as they're supplementing your work, and not replacing it.

As an example, if your group wants to implement more dynamic functionality with JavaScript and the DOM, any code that interfaces with the database must be written by members of the group.

You can use CSS and JQuery all you want to change the appearance of the webpage, but since these tools were not discussed in class, they will obviously not be requirements.

Taking it further

Although none of the items here are required, they are useful features that you might choose to add to the project if you wanted to push things further than the minimum.

- Make a web interface for store employees to add/edit products.
- Add functionality that allows for customization of various properties of products upon ordering. Some examples of this would be variable colors/sizes.
- Add functionality that allows for a store employee to add promo codes that can be used when ordering to affect the final price.
- Add functionality for store employees to set up temporary sales during some time period. For example, a Black Friday sale that is 20% off on some item for one day only.
- Design a system of loyalty reward points for ordering that can be redeemed later.
- Design a wishlist system where users can keep track of items they'd like to order at a later date.

Some notes

- This is a group project, and you need to be able to coordinate with the others in your group. This is the reason that NIU provides Teams, so I would encourage you to coordinate through it, though this is not a requirement. I know that some of you prefer to use Discord/Telegram/etc. and this is not a problem, but remember that each group is responsible for implementing their own application: code should not be shared between groups, and any discussions should be held in a setting not viewable by people outside of the group.
- This project is designed to give everyone a chance to apply their knowledge, and to learn where the gaps in that knowledge may be. To that end, please make sure that all of your group members are involved in each of the steps. It might be easy to carry/be carried, but it is better for everyone if it's actually a team effort. I will be adding an individual submission portion at the end for a survey where you can rate the contribution level of each of your group members. For the most part, the group should succeed or fail as a whole, but there have been students that disappeared and never contributed in prior semesters, and I do not wish to allow those students to be rewarded for their "stealth".
- IMPORTANT: Every group member is expected to contribute. If an individual does not contribute, their score will be made to reflect it, even if the group does succeed in spite of their lack of contribution.
- I would recommend setting up some sort of version control repository (`git` is one that is commonly used) to coordinate the coding between members of your group. If you move the database login information into a separate file that is included, you can minimize the amount of code that needs to change between users as they independently test/develop. Students in the past have set up GitHub repos for their group to great effect. Make sure, however, that only your group members have access to the repo.

What to turn in?

Submit, via Blackboard (one submission per group), the following:

1. The ER diagram (in PDF format) that you designed for the database that is used for the application. This should be your first step, and the other steps should be based upon this ER diagram. It would be best to draw this with some sort of software, but if no one in your group can make that happen, then hand-drawn and scanned diagrams will be accepted, but legibility will be critical and points will be lost for anything a grader cannot read. All entities and relationships must be drawn, along with any identifiers or intersection data. Other attributes do not need to be drawn, but must appear in…

2. … a description of all of the entities, relationships, and attributes that are a part of your ER diagram. This will obviously include their names, their purpose, and any additional data that is important to know. This can be a part of your ER diagram PDF, or a separate PDF file, but it must be present and easy to find.
3. In a PDF, include the relational schema of the database, converted from the ER diagram. Include information on which attributes are primary/foreign keys, and make sure to identify what the home relation is for the foreign keys used. This can be in the same PDF, or a separate one, but it must be present and in an obvious location. It should be based on the ER diagram from before. We will check to make sure it matches, so make sure that any changes made are applied to the ER diagram and the schema here.
4. An SQL script, suitable to run with MariaDB, containing the DML code to create the database designed and detailed in the previous portions.
5. Another SQL script, suitable to run with MariaDB, that inserts the sample data needed to make your application run properly. This should include at least 20 different products (of your choice), 5 customers, and at least one order per customer.
6. A tarball or zip file containing the PHP code and any additional web-facing files that implement the application.
7. A web link to your group's application running on the student web server. This can be served from any of the group members' `public_html` on `turing/hopper`, but it does need to be working there to facilitate grading. Make sure not to delete the implementation until after grading has been completed.

DO NOT submit Word documents. It is acceptable to write them in Word, but make sure to print/export them to PDF before submission.