CSCI UA.0060 Spring 2025

Assignment 7 – Bookstore Part 2

Deliverables

Upload your PyCharm project to GitHub.

Overall Requirements

In this assignment, you will update the bookstore website that you created in the last assignment to retrieve data from the database instead of your hard coded lists and then add Search functionality to the site.

Specific Requirements

1. Create a database called “bookstore” (please make sure you only use lowercase letters in the name and have “bookstore” as one word) and implement the following schema:

book(**bookId**, *categoryId*, title, author, ISBN, price, image, readNow)

category(**categoryId**, categoryName)

1. Populate the database with records that reflect the 4 categories and 16 books (four in each of the categories that you have already created). You can do this in several different ways – you can re-enter the data manually or write your own little program to insert the records from the lists that you produced in the last assignment.
2. Clone the bookstore repository for this assignment into PyCharm. The project is almost identical to the one you started with the last assignment. Therefore, you need to copy the following from the previous assignment repository to this one:

* Index.html – the welcome page
* Category.html – the page that displays books for a category
* Base.html - used for the header and footer for the other pages
* images folder with its subfolders for books, categories and misc.
* app.py file

There may be other files too, if you did something beyond the basic assignment instructions, such as updated css files or additional html pages, etc.

1. In order to update the site to access the database, you should not have to change any of the HTML pages. All the changes happen in app.py:
   1. Insert the mysql.connector import statement
   2. Remove the category and book lists. (Yes, I know you spent a bunch of time putting in that data, but hopefully you used the lists to populate the database!).
   3. Insert the code to create a database object and make a connection to the bookstore database.
   4. Before any of the function definitions, insert some code to retrieve all the categories from the database and create a categories list using cursor.fetchall(). Doing this before the function defintions means that you only retrieve the category data from the database once.
   5. Update the category function by removing the code that creates the selected book list. Replace it with code that retrieves books from the database using the selected category in the WHERE clause of the query. Create a books list using cursor.fetchall() and pass that list to the category.html page. You should be able to do that without changing the return statement if you use the same variable name as you used for the selected book list.
   6. At this point, your site should look exactly as it did before, except that now all the data is coming from the database. You can prove that this is the case by adding or updating data in the database and seeing the updates reflected on your site.
2. Now we are going to implement the Search feature for the site:
   1. Copy the category.html page to a new page and call it search.html
   2. Update the new search.html file to remove the lefthand <section>. This will approximately be from line 5 to line 20.
   3. Update app.py by replacing the placeholder code for the search function (probably starts somewhere around line 40 . The following replacement code should get you started:

@app.route('/search', methods=['POST'])  
def search():  
 search\_info = request.form['search']  
 *# Retrieve the books from the database that include the search string in the book’s title (the search\_info variable)*

*# Fetch all the rows in a list of tuples.* selected\_books = cursor.fetchall()  
 return render\_template("search.html", categories = categories, books = selected\_books)

You need to insert the code to instantiate the database cursor, specify the query and execute it. The query is a little tricky because you must use the parameter %s (to prevent SQL Injection attacks) and that needs to be wrapped with a % wild card on either side. I suggest you research the CONCAT function for this. If you get stuck, post something on the forum and I will help.

1. If you want to make the search feature a little more sophisticated, you can change the query to include authors as well as books. This is not a requirement, but you’re welcome to try it.
2. When you have completed the assignment, commit and push the PyCharm project folder to GitHub.

Grading Rubric

See Brightspace for Grading Rubric