Final Project

CS-333 (Database Management Systems)

Shen Gao

**INTRODUCTION**

In today’s world, databases exist everywhere in our daily lives, especially in businesses where large amounts of information are stored in order to ensure smooth operation. Databases exist to allow records organization, search for a specific record, and inventory control. An example of a place that databases are used is a library. At a library, records-keeping contain information about the books it has, information of patrons, lend dates and due dates of books, to name a few.

My database is based on this need. It is a series of tables that is created to ensure the operation of a fictitious home library. Based on this database, I am able to, hypothetically, lend books to my friends and family. The purpose of this database is to track the names of the books and their authors, their dates of check out and return, and customer information in order to ensure the return of books. Examples of functionality include searching a book by its title, searching by author, searching whether a book has been lent or returned, adding name to customer list, changing customer name, and donating a book.

**DATA MODEL**

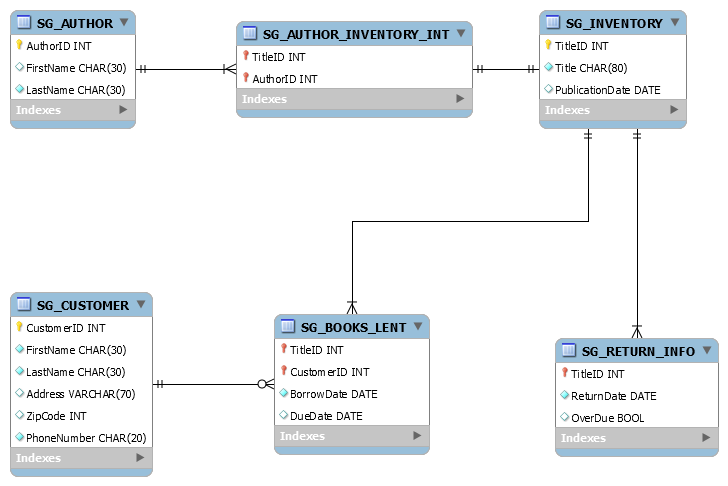


Figure 1

**Relationships**

SG\_AUTHOR & SG\_AUTHOR\_INVENTORY\_INT & SG\_INVENTORY:

From SG\_AUTHOR to SG\_AUTHOR\_INVENTORY\_INT, the relationship is one to many. This is because in the intersection table, one author can correspond to many TitleIDs, meaning an author can write one or many books. From SG\_INVENTORY to SG\_AUTHOR\_INVENTORY\_INT, it is a one-to-one relationship. This is because each unique TitleID is linked only to one author. In other words, a book is only written by one author. From the intersection table to either the SG\_AUTHOR table or the SG\_INVENTORY table, the relationship is one-to-one. This is because each unique combination of foreign keys in the intersection table only corresponds to one row of value in the AUTHOR table as well as one row of value in the INVENTORY table.

SG\_INVENTORY & SG\_RETURN\_INFO:

From INVENTORY to RETURN INFO, the relationship is one to many. This is because one book can be borrowed, and therefore returned many times. The other way around, however, is a one-to-one relationship. One row of record in the RETURN\_INFO table corresponds to only one book in the INVENTORY table.

SG\_INVENTORY & SG\_BOOKS\_LENT:

From SG\_INVENTORY to SG\_BOOKS\_LENT table, the relationship is one to many. This is because the BOOKS\_LENT table records every book that has been borrowed. The one-to-many relationship means that one book can be borrowed many times, to many different customers. The other way around, going from BOOKS\_LENT to INVENTORY, the relationship is one-to-one. This is because one record from the BOOKS\_LENT table can only be linked to one book in the INVENTORY table.

SG\_BOOKS\_LENT & SG\_CUSTOMER:

From SG\_BOOKS\_LENT table to SG\_CUSTOMER table, the relationship is one-to-one. This is because each record in the BOOKS\_LENT table is a unique combination of CustomerID and TitleID, so that one record here can only belong to one customer in the SG\_CUSTOMER table. If going from SG\_CUSTOMER to SG\_BOOKS\_LENT table, the relationship is optional-to-many. My database assumes that a customer can exist in the SG\_CUSTOMER table without having borrowed a book, they can be potential customers. In addition, one customer can borrow many books, and so can have many records in the BOOKS\_LENT table.

**ASSUMPTIONS**

It is assumed that there can be multiple copies of the same book, and that each book has only one author (no co-authors or the such). Customers in the SG\_CUSTOMER table does not have to necessarily have a record of borrowing/returning books. In other words, some are potential customers. In addition, it is assumed that the due date is 3 weeks from the borrow date. In my database, book titles and Authors may reference real-life persons, but are entirely made up. They have not written any books in real life corresponding to their name in the database. In addition, SG\_RETURN\_INFO only contains information about books that have been returned. I had intended for the SG\_AUTHOR table to have unique entries in the sense that an author’s name (if it’s the same person) cannot appear twice – a composite key should have been in place in order to implement this. However, during the completion of my project, it is nearing the end when I realized that my SG\_AUTHOR table does not have a composite primary key. This results in the user being able to submit (donate a book) author names that may already exist and create duplicate entries.

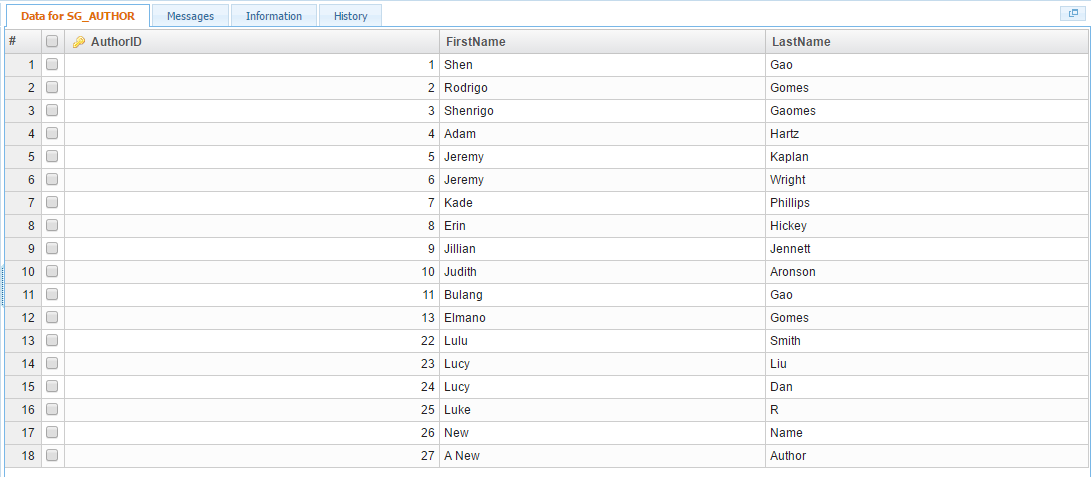
**SCRIPTS (tables + stored procedures)**

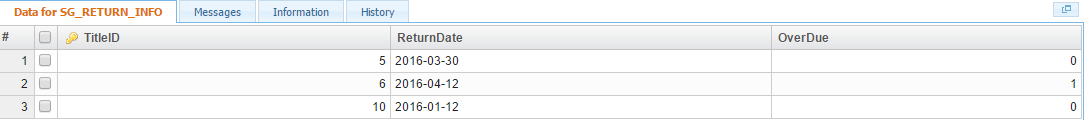
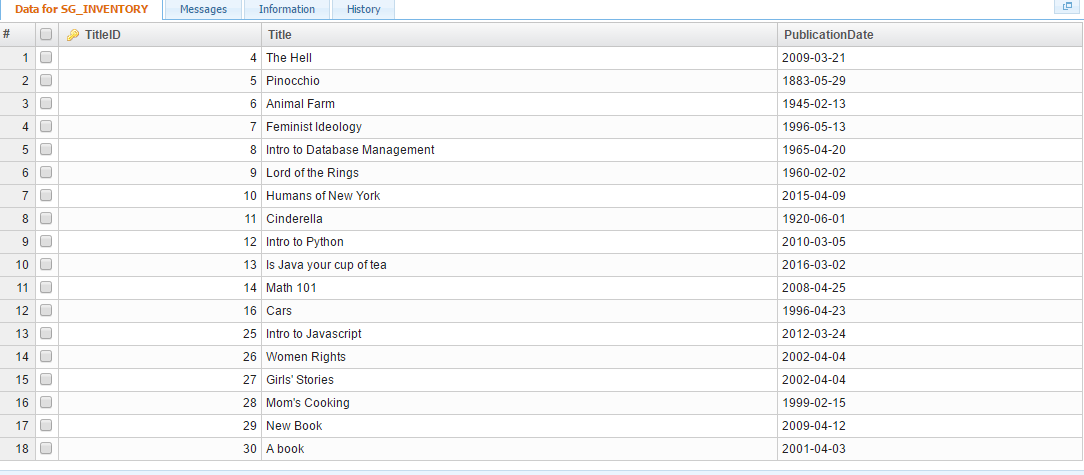
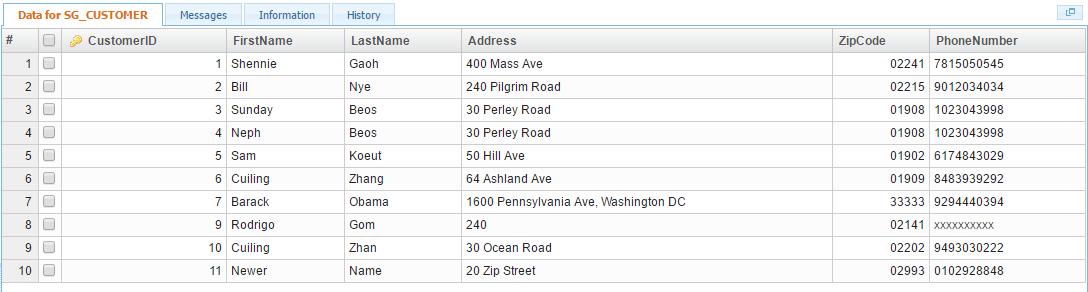
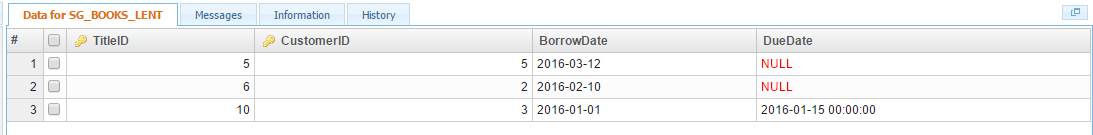
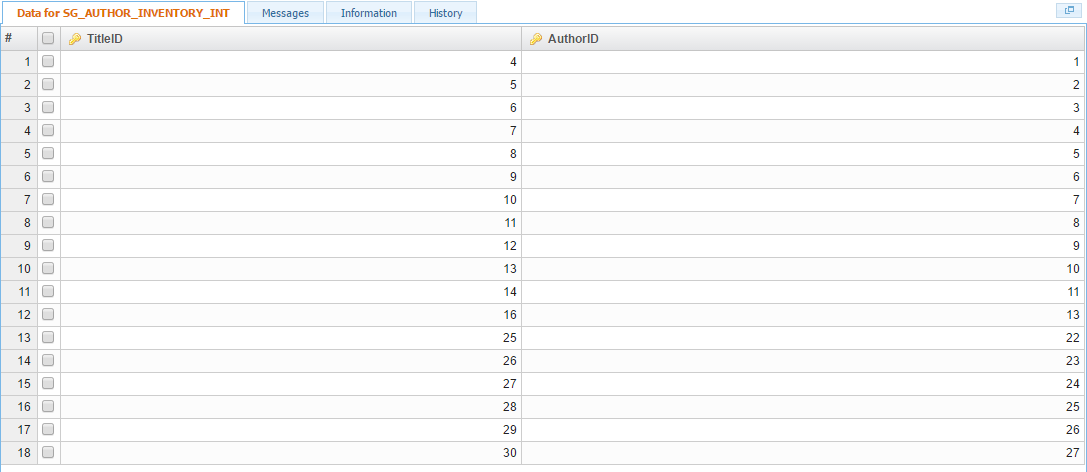
Tables – see attached file

Stored Procedures – see attached file

**SCREENSHOTS (tables + forms + results)**

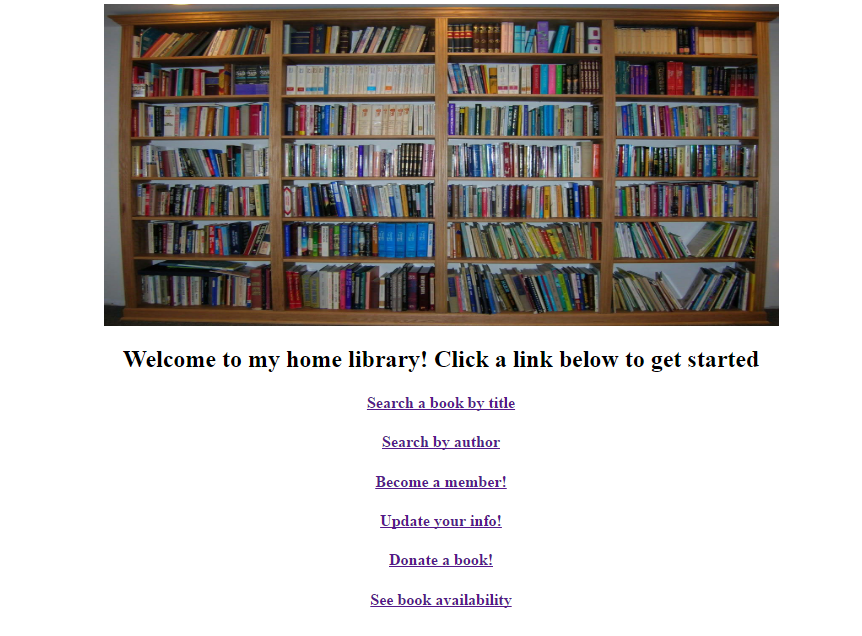
**Tables**



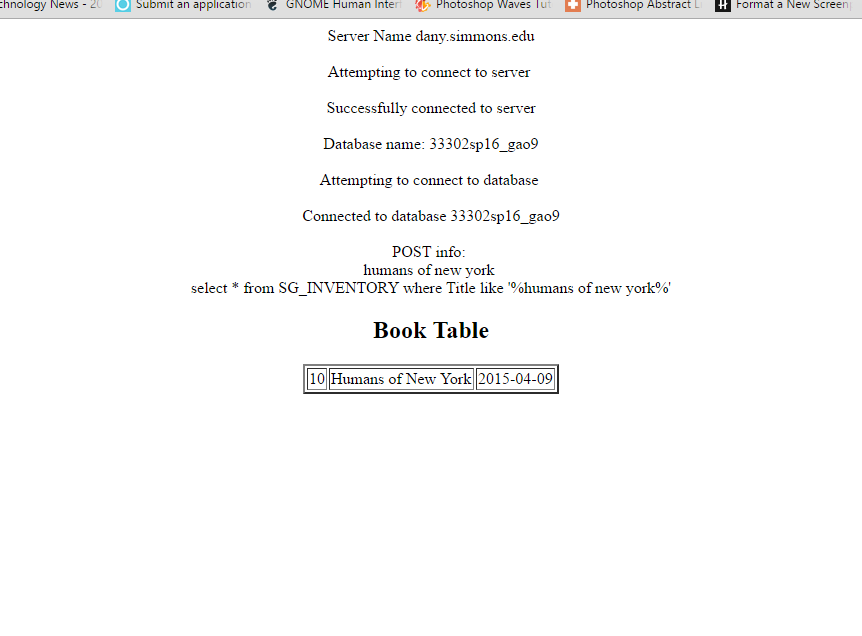
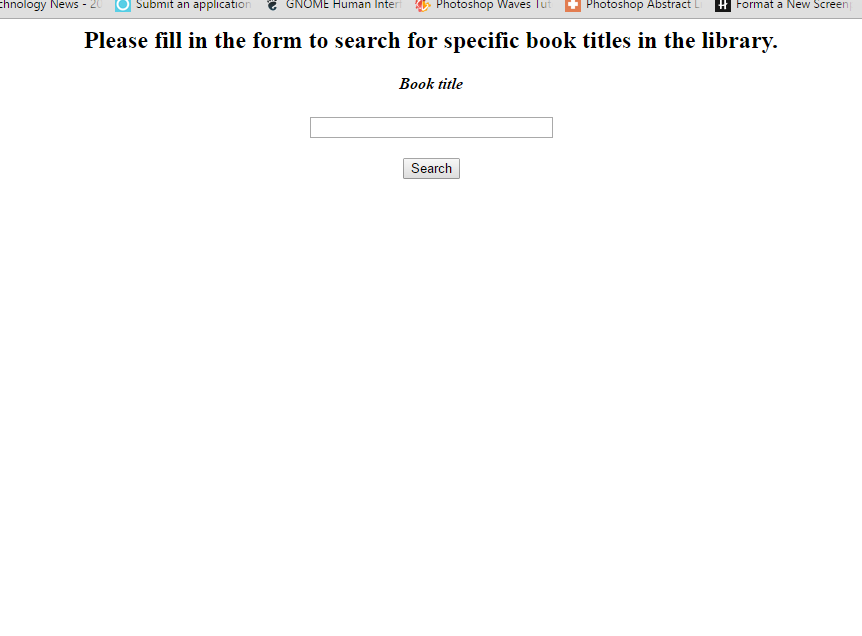


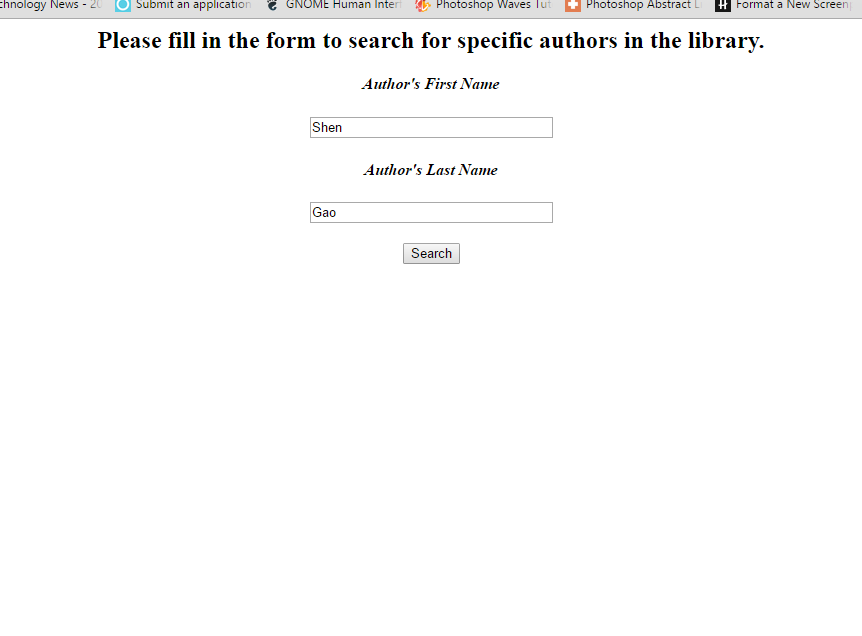
**Forms & Results**

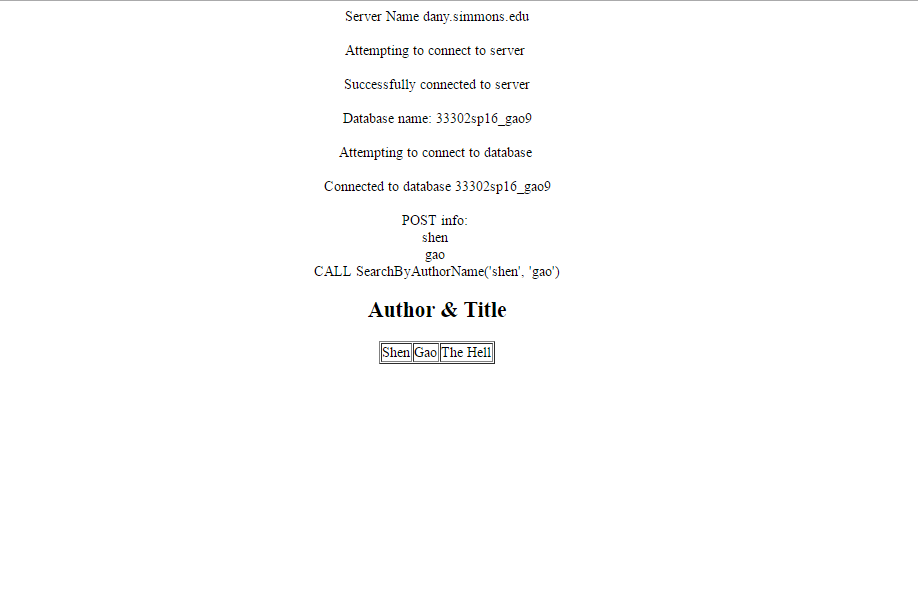
Home page:



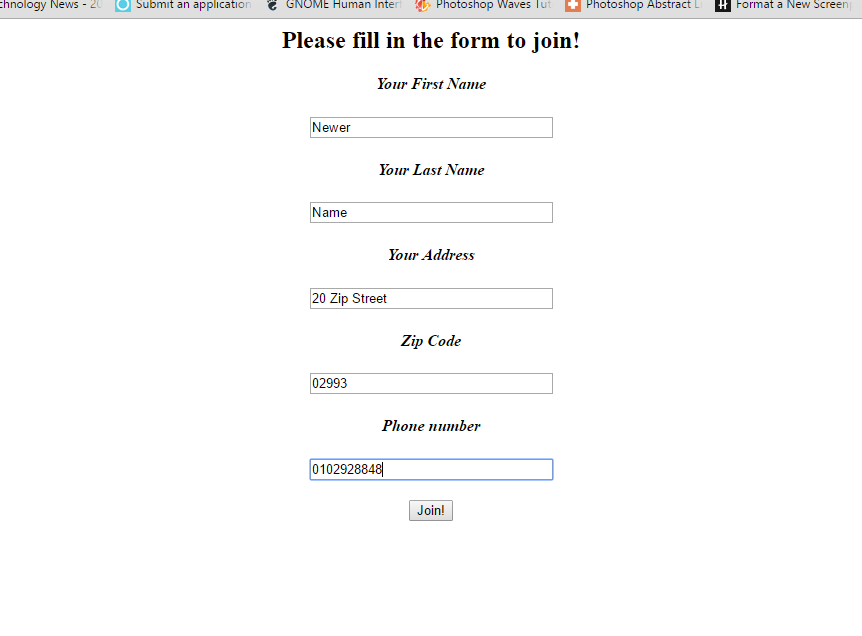
Search a book by title:

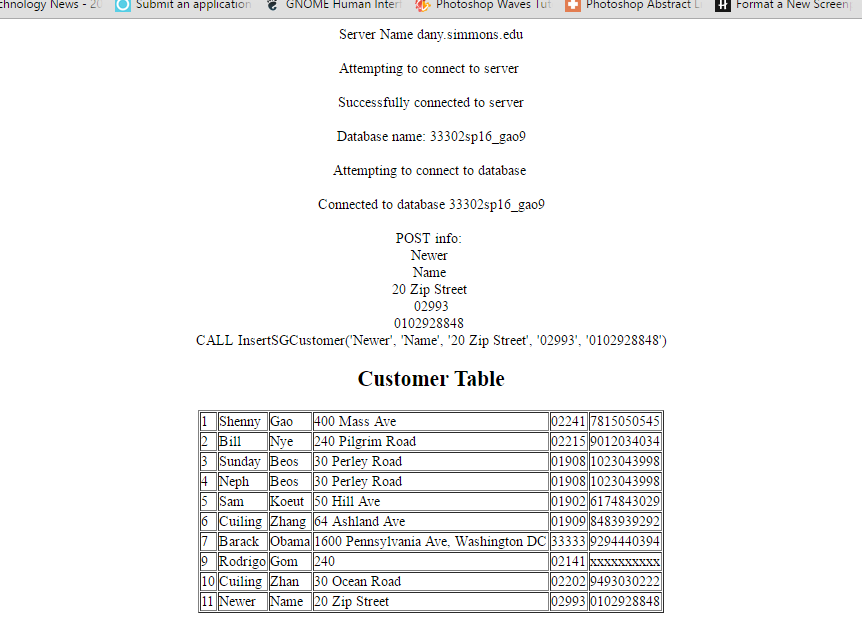


Search by author: 

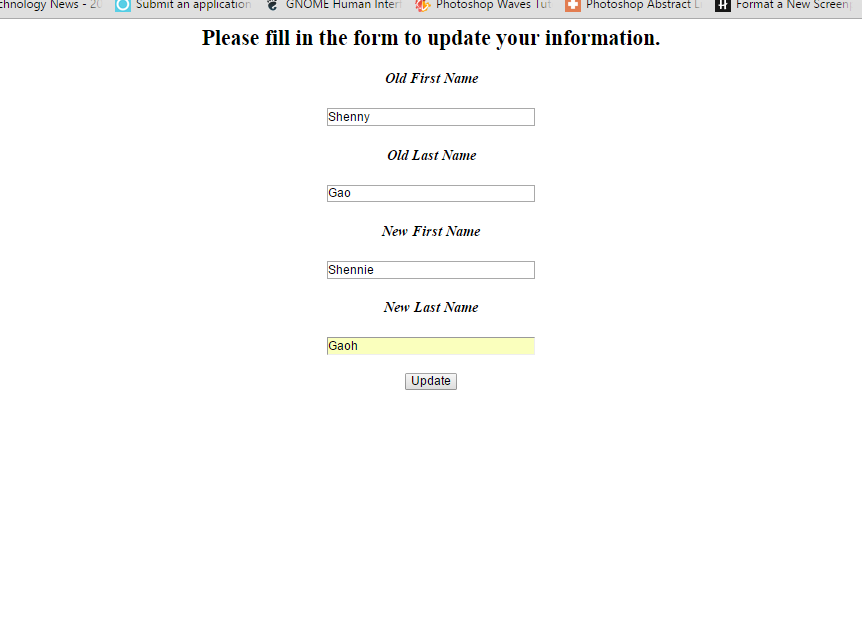


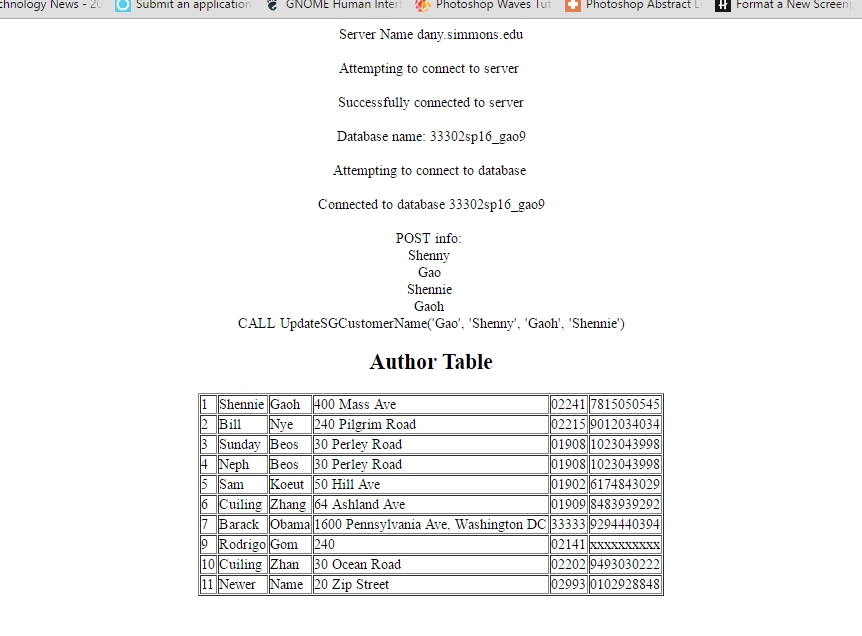
Become a member (insert customer):



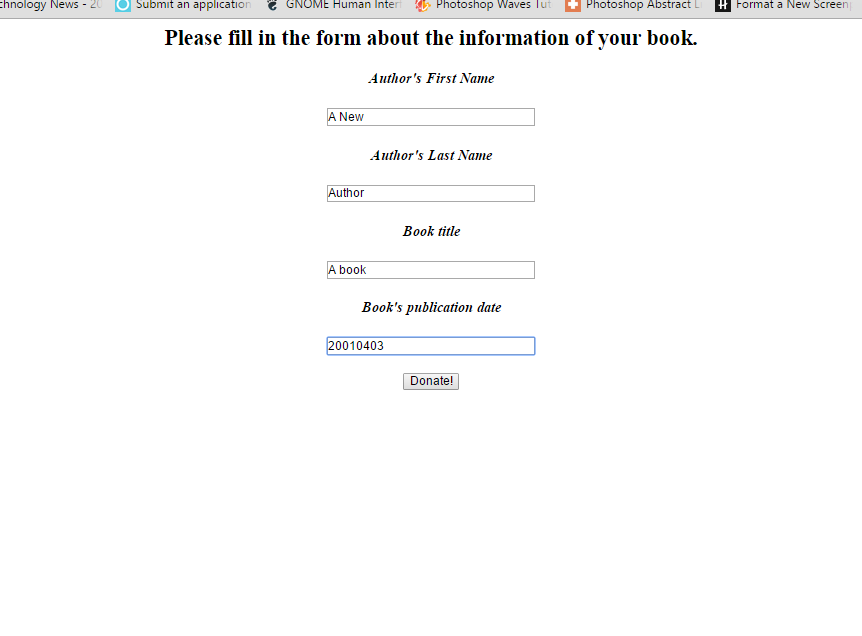


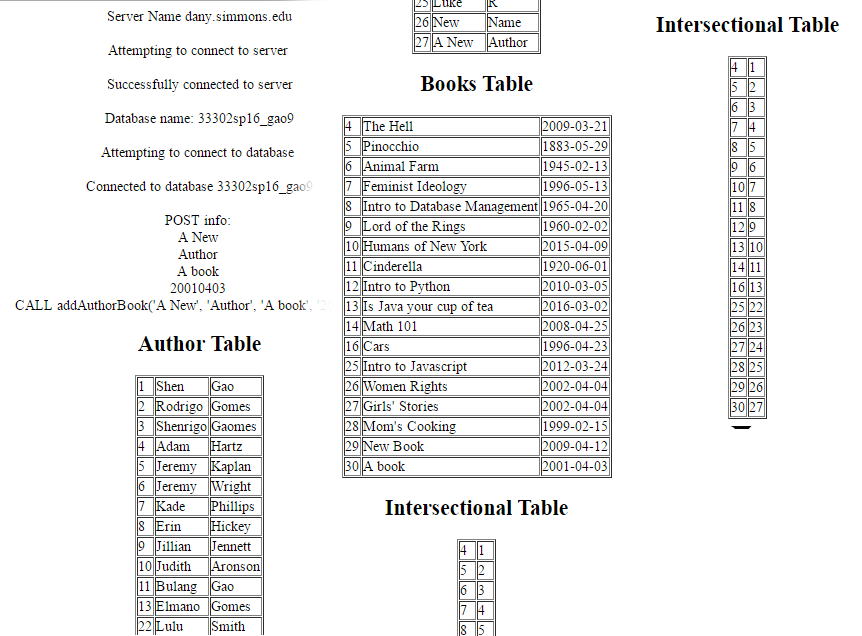
Update your info:



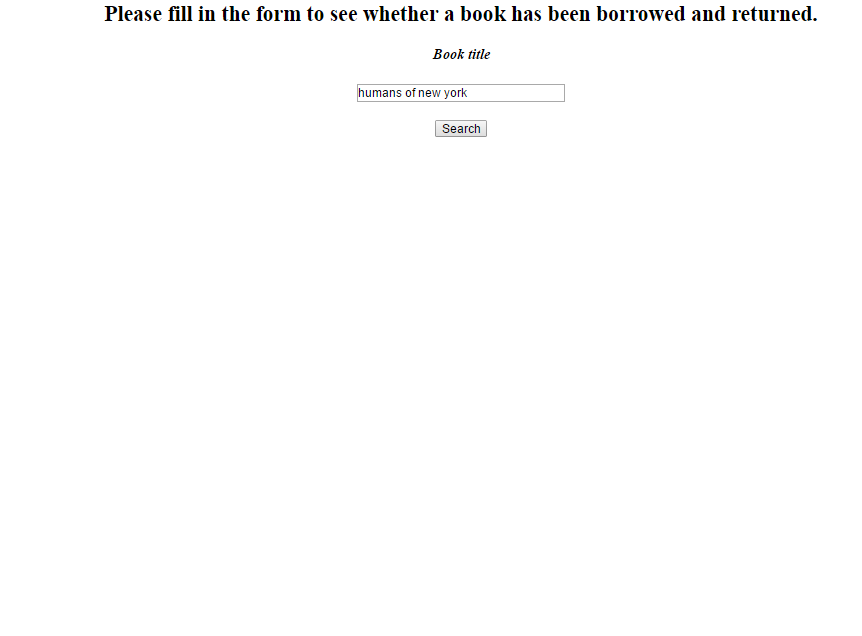


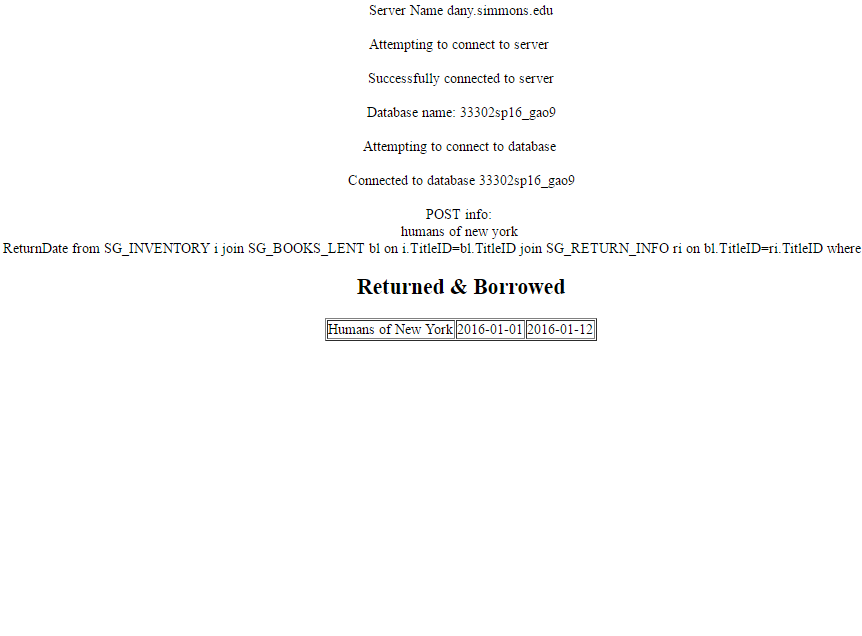
Donating a book:





See book availability (borrowed & returned):





(table above shows BorrowDate and ReturnDate respectively – 2016 01 01 and 2016 01 12)

**SOURCE CODE (user interface – HTML + PHP)**

See attached files