CS6400
Naila Fatima
nfatima3@gatech.edu
ID: I4

Interim Project Report

Databases for Book Management

Application Description

I will be creating an application which allows users to keep track of books that they have read and order books from bookstores. The application will also recommend books to the user (based on the books that they have read or on the genre in which they are interested in) as well as find bookstores which carry a particular book. The application will also allow users to rate books that they have read and this feature will enable it to recommend books. Each book can be rated on a scale of 1-5 where

1 : Did not like book at all
2 : Found the book uninteresting
3 : Found the book average
4 : Found the book interesting
5 : Liked the book a lot

The users of this application consist of individuals who want to keep track of the books that they are interested in as well as find stores which sell these books. It is known that reading is an extremely beneficial activity- it stimulates the mind while improving concentration. As there has been a steady decline in leisure reading over the years, the importance of reading has increased in the present day. For readers, choosing a book from the thousands of available choices is often a daunting task. It is likely that the availability of a book recommendation system will make it easier for readers to select a book by providing them with a list of books which satisfy specific conditions. Readers can search for books which are written by a certain author or those which belong to a certain genre. A rating system is incorporated in order to allow readers to provide feedback about the book. This rating system is used to recommend books to users- books which are highly rated will be recommended to users.

After a book has been selected, the application allows the users to find bookstores which carry the book. An order can be made for a certain book from the bookstores which carry it. The application keeps track of all the orders made by the users. This feature allows the application to recommend best-selling books to the users.

Learning Objectives

By doing this project, I want to become proficient in SQL. Since this project makes use of MySQL (a RDBMS), I want to learn how to work with databases. So far, I have only learnt theoretical details about databases but have not been able to practically make use of them. I believe that this project will allow me to make practical use of databases and hence improve my concepts.

In order to implement this system, I have made use of the *mysql.connector* library (which is written in Python) in order to automatically generate data. Certain relations in my applications have hundreds of entries- it would be a tedious task to manually enter the queries which is why I have used the library to create queries using Python. The creation of databases has allowed me to understand how to quickly add entries to pre-existing table.

The user is able to select certain conditions which a book must satisfy. For example, the user may be interested in a certain genre or in the works of a certain author. The different choices available to users have allowed me to learn how to construct correct queries.

The results retrieved from the database are then displayed on an HTML webpage. I have made use of the Flask library to establish a connection between MySQL and HTML by using a Python script.

Functionality of the System

The application will be able to perform the following functions:

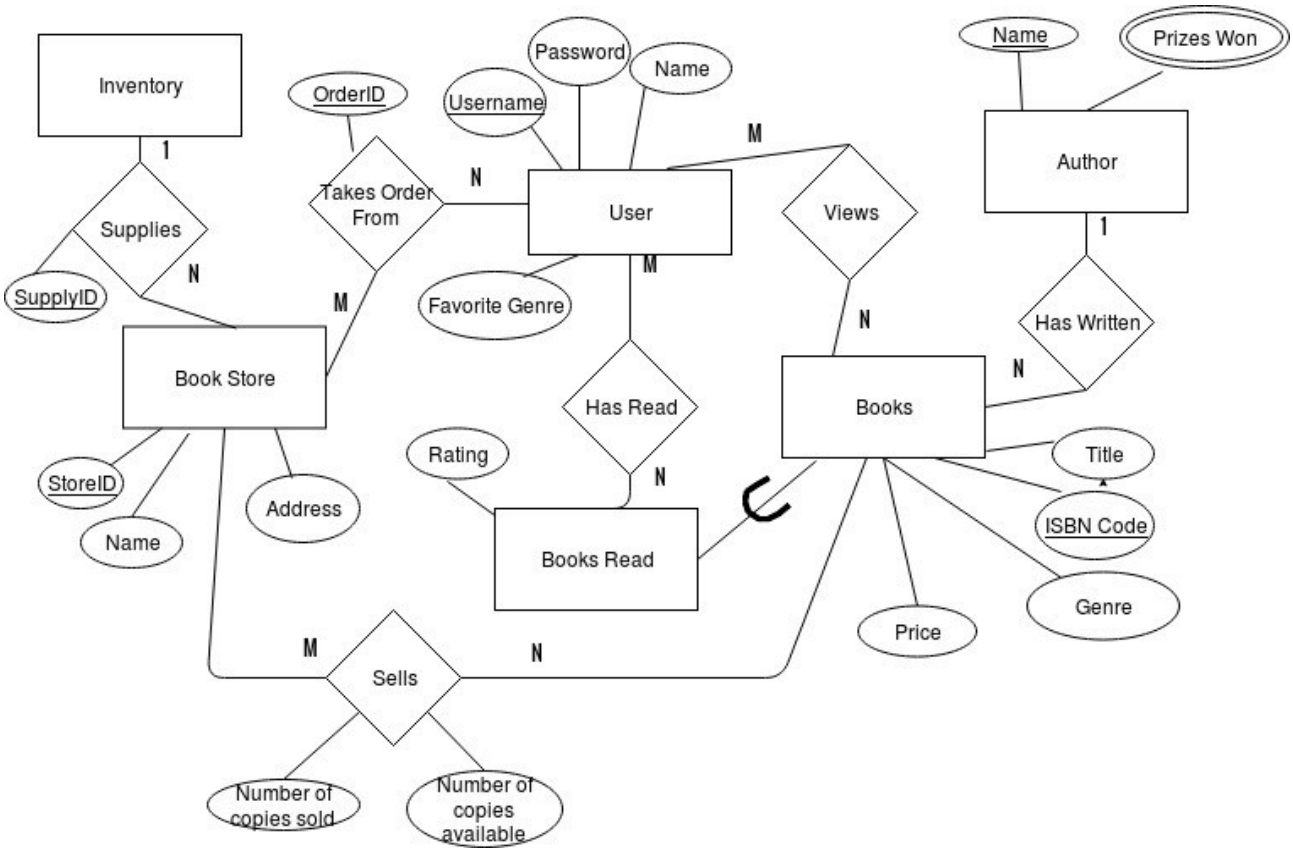
1. Allow users to login: Users will have to use a username and password in order to login to their accounts. The username should be unique for each user.
2. Find books by a particular author: Given an author, all books by that author should be retrieved.

3. Find books of a particular genre: Given a particular genre, all books belonging to that genre should be retrieved.
4. Allow users to rate a book that they have read: Each user will be allowed to rate a book that they have read. They can give a particular book only one rating- either 1,2,3,4 or 5. These ratings will help the application recommend books to users.
5. Find the highest rated book of an author: Find a book written by a particular author which has the highest average rating. This can be considered to be the most-liked book of the author.
6. Recommend a book to the user based on the books that they have read and liked or if they are a new reader, recommend a book of their favorite genre: Each user will be asked to specify a favorite genre (only one). If they are a new reader (if they have not added any books that they have read), they will be recommended a book belonging to their favorite genre. If a user has books which they have read, they will be recommended the highest rated book belonging to the genre which the user has read most.
7. Find a bookstore which carries a particular book: Given a particular book title, retrieve information about the bookstore which carries the book (if in stock) as well as its price.
8. Find the most “critically-acclaimed” book: For all the books in the database, find the book which has the most criticial acclaim. The score for each book is calculated as

   score = Number of copies sold + Number of 5-ratings

   The book with the highest score will be assumed to be the most critically acclaimed.
9. Find all books ordered by a particular user.
10. Find books by authors who have won a certain prize.
11. Find the book with the maximum orders.
12. Automatic ordering of books from the inventory when the number of available copies goes below a certian number.

Enhanced Entity Relation (EER) Diagram



The entity types in my application include *user, books, books read, author, book store* and *inventory*. It should be noted that the entity *books read* is a subclass of the entity *books*. The relationship *views* exists between the entity types *user* and *books* showing that a user is able to view the books in the database. The relationship is M:N as a user can view all books and a book can be viewed by all users. The relationship *has read* exists between the entities *user* and *books read.* This relationship is M:N since a user can read N books (no restrictions on the number of books read by a user) and a book can be read by M users. The relationship *has written* exists between the entities author and books. This relationship is 1:N as an author can write N books (no restrictions on the number of books written by an author) but a book can be written only by one author. The relationship *takes orders from* relates the entity bookstore with the entity user. This relationship stores the details of the orders made to the bookstores by users. Since a bookstore can take orders from N users and a user can order from M bookstores, this relationship is M:N. The *sells* relationship exists between the bookstore and books entity. This relationship is M:N as a bookstore can sell N books and a book can be sold by M bookstores.

I have created the following tables in order to enable this functionality. The schema of the relations is as shown below.

User

Username	Password	Name	Favorite Genre

Books

ISBN Code	Title	Genre	Price	Author

Note: ISBN code is a unique identifier for a book.

Books Read

Username	ISBN Code	Rating	Author

Note: The rating is on a scale of 1-5. This table relates a user entity with the books that they have read. Since each user can read multiple books, there can be multiple ISBN codes for each username and since each book can be read by multiple users, there can be multiple usernames for each ISBN code. Due to this, neither username or ISBN code is a primary key.

Author

Name	Hugo Award	Man Booker Prize	Pulitzer Prize

Note: If an author has received a certain award, it will be indicated with a “Yes” for the corresponding column. We are only considering 3 awards in this application: the Hugo awards, the Man Booker prize and the Pulitzer prize.

Bookstore

StoreID	Name	Address

Orders

OrderID	Username	StoreID	ISBN Code

Book Selling

StoreID	ISBN Code	Number of copies available	Number of copies sold

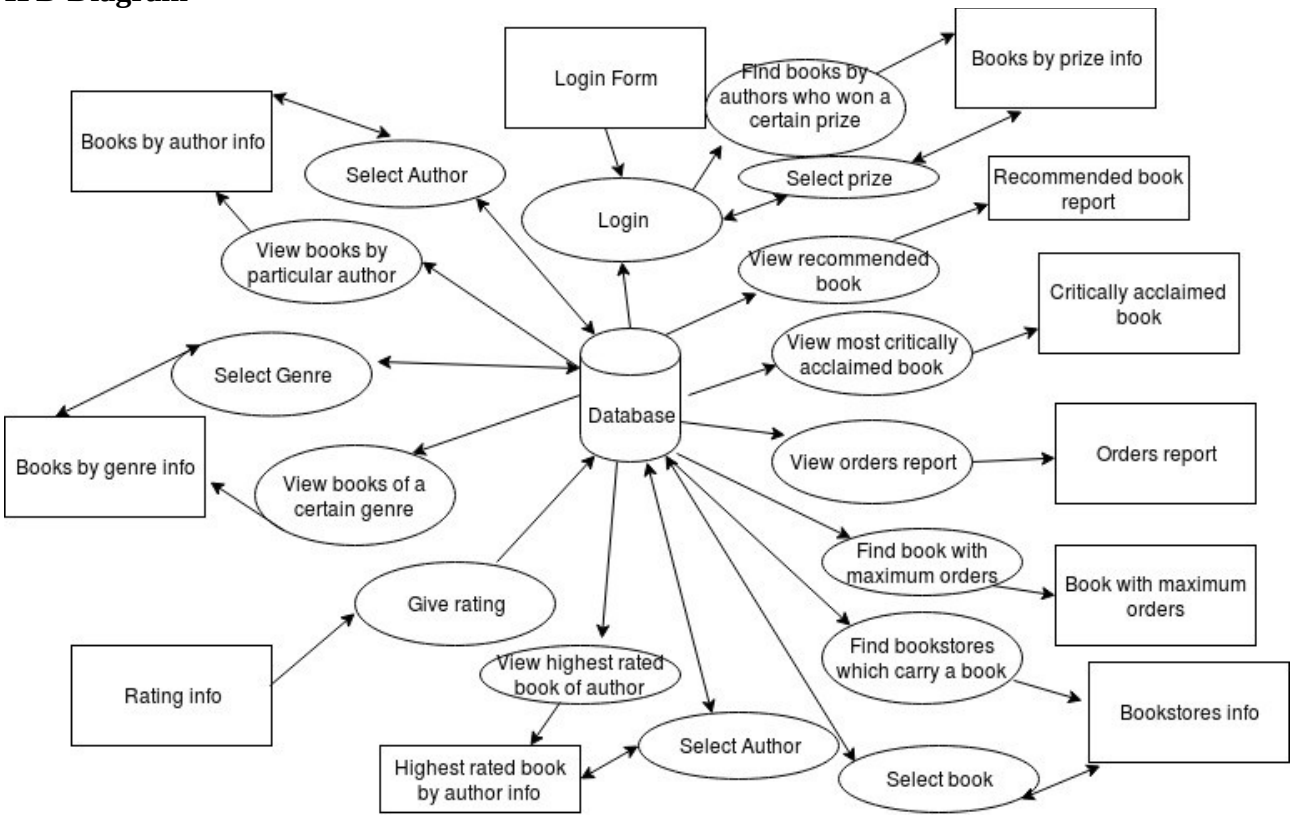
Note: This table relates the books with the bookstores which sell them.

Inventory Supply Orders

SupplyID	StoreID	ISBN Code

Note: We assume that the inventory always has a supply of the books.

IFD Diagram



Data Used

The data was automatically generated and inserted in the tables. Attributes such as username and bookstore address were taken from random generators on the internet. The usernames were taken from the Jimpix Username Generator whereas the bookstore addresses were taken from the Street Name Generator. A list of names was generated by using the Name Generator. A list of books were selected from Goodreads and appropriate values were given to their attributes (title, author and genre).

In order to generate entries for the User relation, each username (which was created by an online generator) was allotted a password, a name and a favorite genre. Passwords are randomly generated strings whereas names have been selected from a list generated by an online Name Generator. The genres considered by this application include Mystery, Fantasy, Drama, Romance and Non-fiction. A random genre was chosen from this list and assigned as the favorite genre for a particular username. Currently, information about 31 users is included in this relation.

In order to generate entries for the Books relation, a list of books was selected from the book-keeping website Goodreads. Information about the title, genre and author of these selected books was stored. For each book, a random string of 5 characters was used as the ISBN code and a random number was assigned to the price. As of now, 51 books are included in this relation.

In order to create the Books_Read table, a random username was chosen from the User relation and a random book was selected from the Books relation. An entry was added to the Books_Read table showing that the selected username had read the selected book. Information about the ISBN code and author name was taken from the Books relation. A random number between 1 and 5 was used as the rating for that particular entry. There are 601 entries present in this relation.

In order to create the Author relation, the prizes awarded to each author were noted. This application only considers 3 awards: the Hugo award, the Man Booker Prize and the Pulitzer Prize. Information about the awards granted to a particular author was taken from Google. Each tuple in the Author relation contains the author's name as well as information about the awards won by the author. If a particular award has been granted to a certain author, the attribute for the award has the value "Yes". Currently, 32 entries are present in this relation.

In order to create the Bookstore relation, random store names were used and a list of addresses were generated by the Street Name Generator. For each store in the list, a random address was allotted with a unique integer store ID. There are 19 entries in this relation.

The Book_Selling relation was created by selecting one random tuple from the Bookstore and Books relations. The store ID of the selected bookstore is entered along with the ISBN code of the selected book. Random numbers have been used to represent the number of available copies and the number of copies sold. There are 800 entries in this relation.

The Orders relation was created by selecting random bookstores from the Bookstore relation, random books from the Books relation and random usernames from the User relation. The store ID has been taken from the Bookstore relation, the ISBN code has been taken from the Books relation and the username has been taken from the User relation. A unique order ID has been allotted to each tuple which contains the selected username, book title and store ID. This relation has 400 entries currently.

The Inventory relation was created by selecting random bookstores from the Bookstore relation and random books from the Books relation. The store ID of the selected bookstores is taken and the ISBN code of the selected book is used. A unique supply ID is assigned to each tuple which contains the selected store ID and ISBN code. Currently, this relation has 400 entries.

The *random* and *string* libraries of Python have been used to generate several of the entries. The *random* library has often been used to create relations by referencing random tuples from other relations.

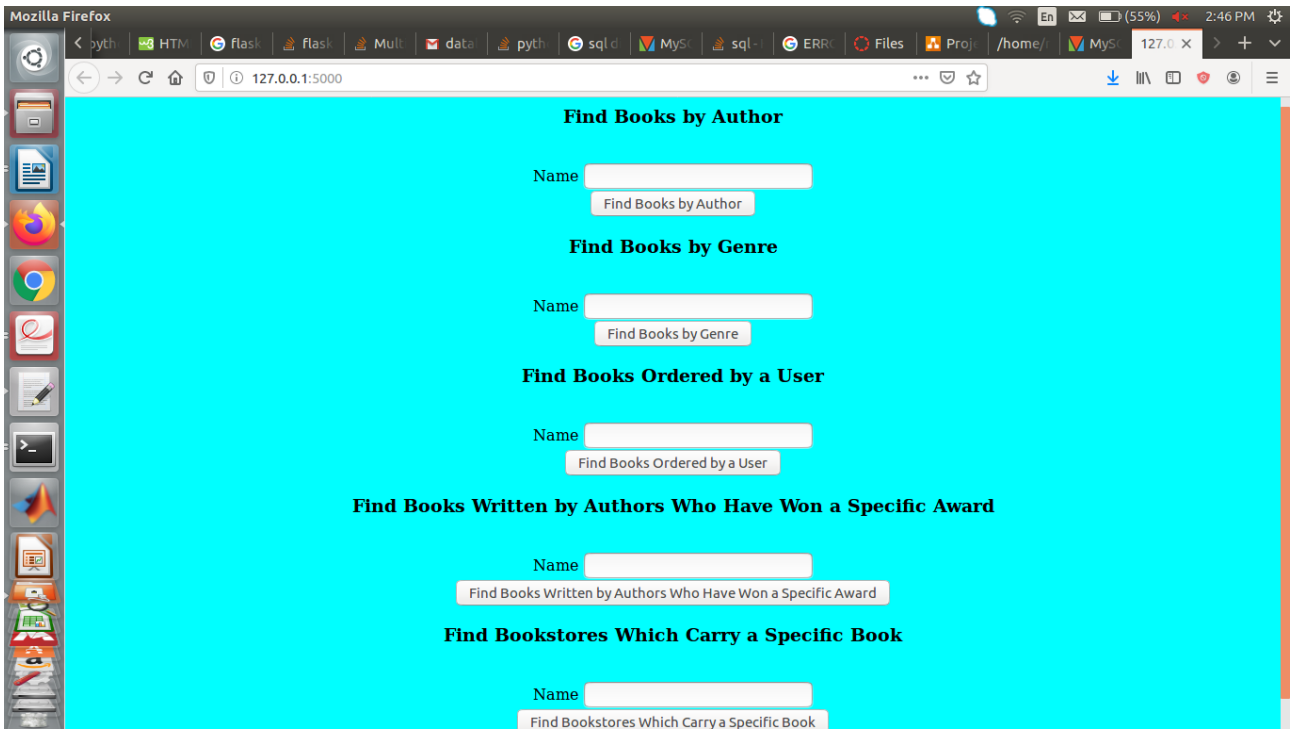
The *mysql.connector* library was used to establish a connection between the Python code and the MySQL database. In order to insert values into an existing table, the entry values were stored in a list and the syntax of the insert query was specified. The *executemany*() function was used to add several entries simultaneously to a specified relation and a *commit*() function ensured that the changes reflected in the database.

Interface Details

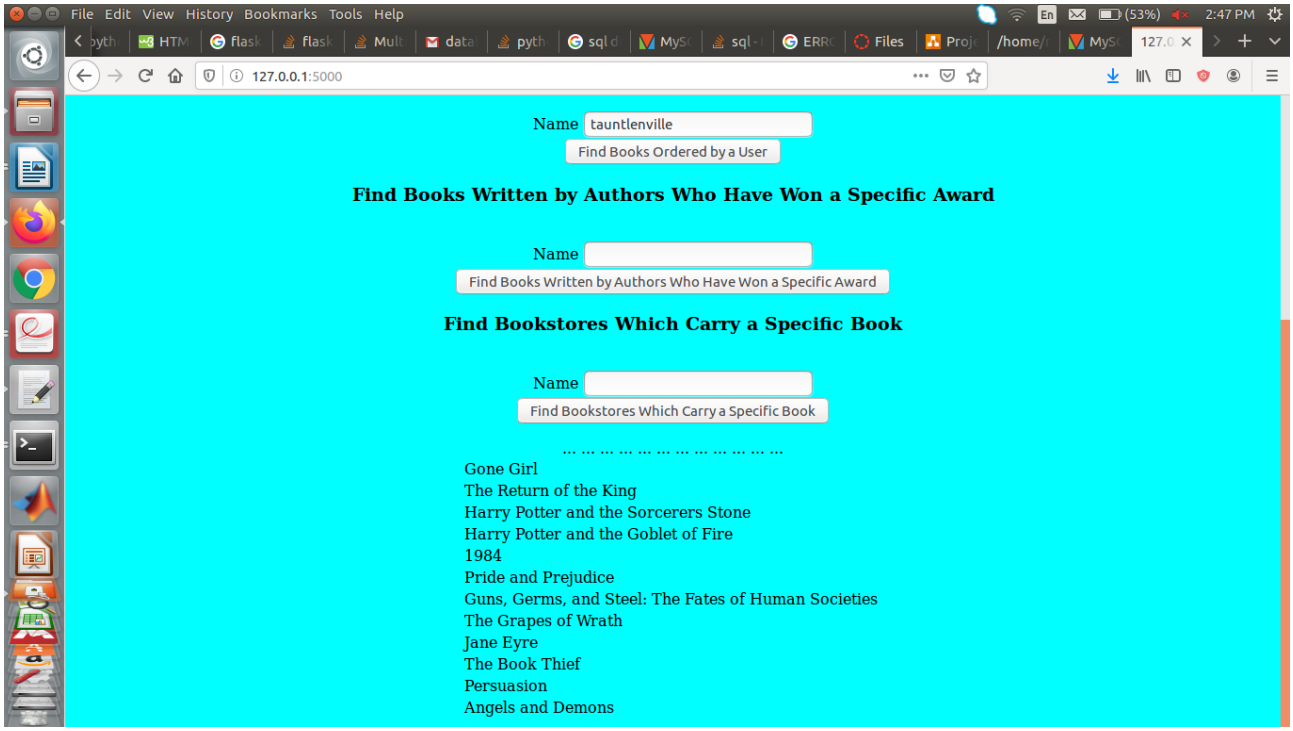
An HTML web page has been used to provide input to the application and view the retrieved results. The Flask library has been used to provide an interface between the Python script and the HTML document. The flask_mysqldb library has been used to establish an interface between the Python script and the MySQL database.

The webpage contains textboxes to receive input information on the click of submit buttons. Appropriate labels have been provided so as to make the application user-friendly. A text area has been used to display the retrieved results on the webpage.

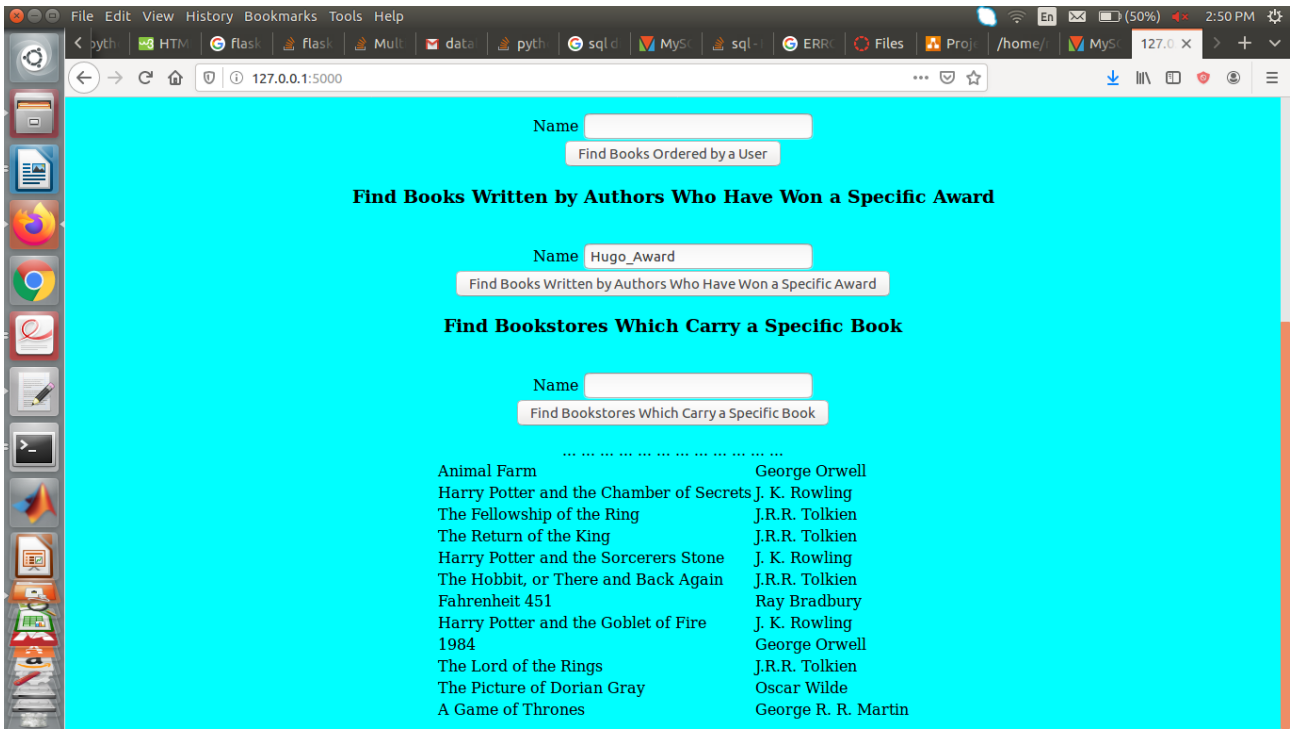
The interface of the application has been shown below:



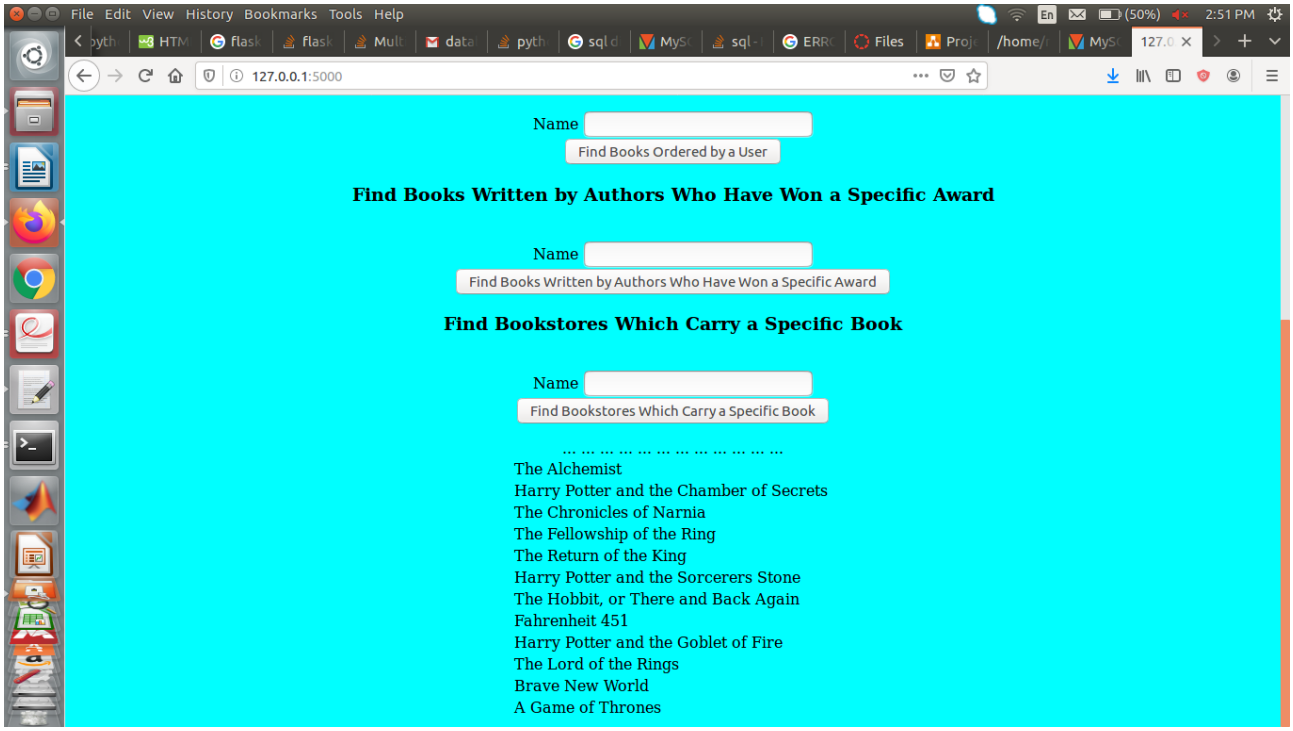
The webpage takes input through the text boxes and submit buttons. The retrieved results are presented in a text area as shown below.



The above image shows the books ordered by the user ‘tautlenville’.



The above image shows the books written by authors who have received a Hugo Award. The author's names are present next to the books they have written.



The above image shows all books belonging to the fantasy genre. Since all the fields are present in a vertical manner, one has to scroll up to input text. In the demo, I hope to have a more user-friendly webpage.

DBMS and libraries

DBMS used: MySQL

Libraries used: random, string, Flask, flask_mysqldb, mysql.connector

Future Work

Improve the interface and make it more user-friendly. Currently the webpage contains the input fields in a vertical manner. This makes the application a bit difficult to use since the user has to scroll up to input text and scroll down to view the retrieved results.

Add user login and registration functionality. Also, allow a user to add books to the database and rate books that they have read.

References

1. Jimpix Username Generator: https://jimpix.co.uk/words/random-username-generator.asp
2. Street Name Generator: https://www.name-generator.org.uk/street/
3. Goodreads: https://jimpix.co.uk/words/random-username-generator.asp
4. Name Generator: https://www.name-generator.org.uk/quick/