# Description

My project is to create a web application that allows users to access gutenberg books, their authors, their publishers, and rate the books once they have made an account.

# ER Diagram

Diagram

Description automatically generated

# Database Schema

Authors: (name: str, yearOfDeath:int)

Publishers: (name: str, stillAround:boolean)

Readers: (username: str, password, favoriteBookTitle:str, favoriteBookAuthor:str, favoriteAuthor:str)

Books: (authorName: str, title:str, link: str)

Rate: (username:str, authorName:str, title: str, rating:float)

Publishes: (authorName:str, title: str, publisherName:str, yearOfPublication:int, location:str)

*Note: Rate was not made into a supporting relationship because it is not a many-to-one relationship as books can be rated by multiple readers and readers and rate multiple books. Publishes is not a supporting relationship because it does not meet the condition of “for every E-entity, there must be exactly one existing F-identity related to it by R” (154 of our textbook)” because it is many-to-many since publishers can (and often do) publish multiple books and books can have multiple publishers – often one in the US and one in the UK.*

# Closure Sets and Functional Dependencies

## Authors

There are only two attributes, so this relation is in BCNF.

## Publishers

Similar to the Authors relation, there are only two attributes in the relation, so this is also in BCNF.

## Readers

The only nontrivial FD is username -> password, favoriteBookTitle, favoriteBookAuthor, favoriteAuthor. Since the left side of this FD (username) is a superkey, this relation is in BCNF.

## Books

The only nontrivial FD is authorName, title -> link. Since the left side of the FD is a superkey, this relation is in BCNF.

## Rate

The only non trivial FD in this relation is title, authorName, title, username -> rating and since authorName, title, username is the key for this relation, this relation is in BCNF.

## Publishes

The only non-trivial FD in this relation is title, authorName, title, publisherName-> location, yearOfPublication, and since authorName, title, publisherName is the key for this relation, it is in BCNF.

# Sample Data Input SQL File

username,password,favoriteBookTitle,favoriteBookAuthor,favoriteAuthor

K,password123,The Adventures of Ferdinand Count Fathom Complete,Null,Null

P,0c6U8M&1TO#l,Null,Null,Null

person1,D8p13K7mEGcsBXdRzc5u,Null,Null,T. Smollett

person2,aJUeLxrmJf7WRYx4kteV,The Enchanted April,Null,Null

testUser,password123,Null,Null,Null

someName,jBksGtBvFv,Little Women,Null,William Shakespeare

bookluver23,fbQNP7yPxt,Null,Null,Null

whaleChaser,password123,Moby Dick,Null,Herman Melville

vampireFan,fHsdmuVLvL,Null,Null,Bram Stoker

person3,password123,Middlemarch ,Null,Null

authorName,title,publisherName,yearOfPublication,location

Bram Stoker,Dracula,Archibald Constable and Company,1897,United Kingdom

E. M. Forster,A Room with a View,Edward Arnold,1908,United Kingdom

George Eliot,Middlemarch,William Blackwood and Sons,1871,Null

Louisa May Alcott,Little Women,Roberts Brothers,1868,United States

Elizabeth von Arnim,The Enchanted April,Macmillan & Co. Ltd.,1922,United Kingdom

Herman Melville,Moby Dick,Richard Bentley,1851,United States

Herman Melville,Moby Dick,Harper & Brothers,1851,United Kingdom

T. Smollett,The Adventures of Ferdinand Count Fathom — Complete,Null,1753,United Kingdom

T. Smollett,The Expedition of Humphry Clinker,W. Johnson and B. Collins,1771,United Kingdom

Henry Fielding,"History of Tom Jones, a Foundling",Andrew Millar,1749,United Kingdom

L. M. Montgomery,The Blue Castle: a novel,McClelland and Stewart ,1926,Canada

L. M. Montgomery,The Blue Castle: a novel,Frederick A. Stokes,1926,United States

William Shakespeare,The Complete Works of William Shakespeare,Wordsworth Editions Ltd,1997,Null

publisherName,stillAround

Archibald Constable and Company,"No"

Edward Arnold,"No"

William Blackwood and Sons,"No"

Roberts Brothers,NULL

Macmillan & Co. Ltd.,"Yes"

Richard Bentley,"No"

Harper & Brothers,"Yes"

Null,Null

W. Johnson and B. Collins,"No"

Andrew Millar,"No"

McClelland and Stewart,"Yes"

Frederick A. Stokes,"No"

Wordsworth Editions Ltd,"Yes"

authorName,title,averageRating,link

Bram Stoker,Dracula,NULL,https://www.gutenberg.org/ebooks/345.html.images

E. M. Forster,A Room with a View,NULL,https://www.gutenberg.org/ebooks/2641.html.images

George Eliot,Middlemarch,NULL,https://www.gutenberg.org/ebooks/145.html.images

Louisa May Alcott,Little Women,NULL,https://www.gutenberg.org/ebooks/37106.html.images

Elizabeth von Arnim,The Enchanted April,NULL,https://www.gutenberg.org/ebooks/16389.html.images

Herman Melville,Moby Dick,NULL,https://www.gutenberg.org/ebooks/2701.html.images

T. Smollett,The Adventures of Ferdinand Count Fathom — Complete,NULL,https://www.gutenberg.org/ebooks/6761.html.images

T. Smollett,The Expedition of Humphry Clinker,NULL,https://www.gutenberg.org/ebooks/2160.html.images

Henry Fielding,"History of Tom Jones, a Foundling",NULL,https://www.gutenberg.org/ebooks/6593.html.images

L. M. Montgomery,The Blue Castle: a novel,NULL,https://www.gutenberg.org/ebooks/67979.html.images

William Shakespeare,The Complete Works of William Shakespeare,NULL,https://www.gutenberg.org/ebooks/100.html.images

name,yearOfDeath

E. M. Forster,1970

George Eliot,1880

Louisa May Alcott,1923

Elizabeth Von Arnim,1941

L. M. Montgomery,1942

William Shakespeare,1616

Herman Melville,1891

T. Smollett,1771

Henry Fielding,1754

Mary Wollstonecraft Shelley,1851

Bram Stoker,1912

L. M. Montgomery,1942

### SQL

CREATE TABLE Authors

(

name VARCHAR(250) PRIMARY KEY,

yearOfDeath INTEGER

);

CREATE TABLE Publishers

(

name VARCHAR(250) PRIMARY KEY,

stillAround VARCHAR(4)

);

CREATE TABLE Books

(

authorName VARCHAR(250) NOT NULL,

title VARCHAR(250) PRIMARY KEY,

averageRating FLOAT DEFAULT NULL,

link VARCHAR(500) UNIQUE NOT NULL,

FOREIGN KEY (authorName) REFERENCES Authors (name)

);

CREATE TABLE Readers

(

username VARCHAR(250) PRIMARY KEY,

password VARCHAR(250) NOT NULL,

favoriteBookTitle VARCHAR(250) DEFAULT NULL,

favoriteBookAuthor VARCHAR(250) DEFAULT NULL,

favoriteAuthor VARCHAR(250) DEFAULT NULL,

FOREIGN KEY (favoriteAuthor) REFERENCES Authors (name),

FOREIGN KEY (favoriteBookTitle) REFERENCES Books(title),

FOREIGN KEY (favoriteBookAuthor) REFERENCES Authors(name)

);

CREATE TABLE Reading

(

username VARCHAR(250),

authorName VARCHAR(250),

title VARCHAR(250),

FOREIGN KEY (username) REFERENCES Readers(username)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (authorName) REFERENCES Authors(name),

FOREIGN KEY (title) REFERENCES Books(title)

);

DROP TABLE Reading;

CREATE TABLE Rate

(

username VARCHAR(250),

authorName VARCHAR(250),

title VARCHAR(250),

rating FLOAT,

FOREIGN KEY (username) REFERENCES Readers(username)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (authorName) REFERENCES Authors(name),

FOREIGN KEY (title) REFERENCES Books(title)

);

ALTER TABLE rate DROP CONSTRAINT rate\_ibfk\_1;

ALTER TABLE rate ADD CONSTRAINT rate\_ibfk\_1

FOREIGN KEY (username) REFERENCES Readers(username)

ON DELETE CASCADE

ON UPDATE CASCADE;

CREATE TABLE Publishes

(

authorName VARCHAR(250),

title VARCHAR(250),

publisherName VARCHAR(250),

yearOfPublication INTEGER,

location VARCHAR(200),

FOREIGN KEY (authorName) REFERENCES Authors(name),

FOREIGN KEY (title) REFERENCES Books(title),

FOREIGN KEY (publisherName) REFERENCES Publishers(name)

);

delimiter //

CREATE TRIGGER nullVals1

BEFORE UPDATE on readers

FOR EACH ROW

BEGIN

IF NEW.favoriteAuthor= "None" THEN

SET NEW.favoriteAuthor= NULL;

END IF;

END//

CREATE TRIGGER nullVals2

BEFORE UPDATE on readers

FOR EACH ROW

BEGIN

IF NEW.favoriteBookAuthor= "None" THEN

SET NEW.favoriteBookAuthor= NULL;

END IF;

END//

CREATE TRIGGER nullVals3

BEFORE UPDATE on readers

FOR EACH ROW

BEGIN

IF NEW.favoriteBookTitle= "None" THEN

SET NEW.favoriteBookTitle= NULL;

END IF;

END//

delimiter ;

/\* CREATE TRIGGER rating1

AFTER INSERT on rate

FOR EACH ROW

BEGIN

INSERT INTO books(averageRating)

(SELECT val

FROM

(SELECT books.authorName, books.title, AVG(rating) as val

FROM books

INNER JOIN Rate on rate.title= books.title AND rate.authorName= books.authorName

GROUP BY books.authorName, books.title) AS temp, NEW

WHERE NEW.authorName= temp.authorName AND NEW.title= temp.title);

UPDATE books SET averageRating =

SELECT temp.val

FROM

(SELECT books.authorName, books.title, AVG(rating) as val

FROM books

INNER JOIN Rate on rate.title= books.title AND rate.authorName= books.authorName

GROUP BY books.authorName, books.title) AS temp

WHERE NEW.authorName= temp.authorName AND NEW.title= temp.title;

END; \*/

ALTER TABLE Books

DROP COLUMN averageRating;

delimiter //

CREATE TRIGGER nullVals4

BEFORE INSERT on readers

FOR EACH ROW

BEGIN

IF NEW.favoriteAuthor= "None" THEN

SET NEW.favoriteAuthor= NULL;

END IF;

END//

CREATE TRIGGER nullVals5

BEFORE INSERT on readers

FOR EACH ROW

BEGIN

IF NEW.favoriteBookAuthor= "None" THEN

SET NEW.favoriteBookAuthor= NULL;

END IF;

END//

CREATE TRIGGER nullVals6

BEFORE INSERT on readers

FOR EACH ROW

BEGIN

IF NEW.favoriteBookTitle= "None" THEN

SET NEW.favoriteBookTitle= NULL;

END IF;

END//

delimiter ;

ALTER TABLE authors ADD COLUMN yearOfBirth INTEGER;

INSERT INTO authors VALUES("Charles Dickens", 1812, 1870);

ALTER TABLE Publishers MODIFY COLUMN stillAround VARCHAR(4);

# Source Code

#LL: I'm using the code found here [https://www.geeksforgeeks.org/profile-application-using-python-flask-and-mysql/] as a foundation for my project

#http://localhost:5000/

# Store this code in 'app.py' file

from flask import Flask, render\_template, request, redirect, url\_for, session

from flask\_mysqldb import MySQL

from MySQLdb import \_mysql

import MySQLdb.cursors

import re

from mysql.connector import (connection)

import webbrowser

app = Flask(\_\_name\_\_)

app.secret\_key = 'your secret key'

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = "DBHW2\_2"

app.config['MYSQL\_PASSWORD'] = "0Ct18Rs3n&q&"

app.config['MYSQL\_DB'] = 'Books'

password= "0Ct18Rs3n&q&"

user= "DBHW2\_2"

mysql = MySQL(app)

@app.route('/')

@app.route('/login', methods =['GET', 'POST'])

def login():

msg = ''

if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

username = request.form['username']

password = request.form['password']

cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

cursor.execute('SELECT \* FROM readers WHERE username = % s AND password = % s', (username, password, ))

account = cursor.fetchone()

if account:

session['loggedin'] = True

session['username'] = account['username']

msg = 'Logged in successfully !'

return render\_template('index.html', msg = msg)

else:

msg = 'Incorrect username / password !'

return render\_template('login.html', msg = msg)

@app.route('/logout')

def logout():

session.pop('loggedin', None)

session.pop('username', None)

return redirect(url\_for('login'))

@app.route('/register', methods =['GET', 'POST'])

def register():

msg = ''

if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

#print(request.form)

username = request.form['username'].strip()

password = request.form['password'].strip()

fBT= request.form["favoriteBookTitle"].strip()

fBA= request.form["favoriteBookAuthor"].strip()

fA= request.form["favoriteAuthor"].strip()

cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

cursor.execute('SELECT \* FROM readers WHERE username = % s', (username, ))

#cursor.close()

account = cursor.fetchone()

if account:

msg = 'Account already exists !'

# elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

# msg = 'Invalid email address !'

# elif not re.match(r'[A-Za-z0-9]+', username):

# msg = 'name must contain only characters and numbers !'

else:

if fA == "":

fA= "None"

if fBT =="":

fBT= "None"

if fBA =="":

fBA= "None"

if "password" =="":

msg= "You need to enter a value for your password"

#TODO

cursor.execute("SELECT DISTINCT name FROM Authors")

val= cursor.fetchall()

authorList= []

for i in val:

authorList.append(list(i.values())[0])

cursor.execute("SELECT DISTINCT authorName FROM books")

fAuthorList= []

val= cursor.fetchall()

# print("val", val)

# print()

for i in val:

fAuthorList.append(list(i.values())[0])

cursor.execute("SELECT DISTINCT title FROM books")

fBookList= list()

val= cursor.fetchall()

for i in val:

fBookList.append(list(i.values())[0])

#cursor.execute('SELECT authorName FROM Books WHERE title= %s', (fBT,))

#cursor.execute('SELECT authorName FROM Books WHERE title= %s', [fBT])

#('SELECT \* FROM readers WHERE username = % s', (username, ))

# print([username, password, fBT, fBA, fA])

if (fA != "None") and (fA not in list(authorList)):

msg= "The author you entered is not in our database."

return render\_template('register.html', msg = msg)

if (fBA != "None") and (fBT != "None"):

cursor.execute('SELECT authorName FROM Books WHERE title = % s', (fBT, ))

rAN= list(cursor.fetchall())

# print(rAN)

if (fBA != "None") and (fBA not in fAuthorList):

msg= "The author you entered has not written a book in our database."

return render\_template('register.html', msg = msg)

if (fBT != "None") and (fBT not in fBookList):

msg= "The book you entered is not in our database"

return render\_template('register.html', msg = msg)

if len(rAN) > 0:

if fBA != rAN[0]:

msg= "The author of your favorite book and the title of your favorite book do not match."

return render\_template('register.html', msg = msg)

else:

# print([username, password, fBT, fBA, fA])

#cursor.execute("UPDATE readers SET password= %s, favoriteBookTitle= %s, favoriteBookAuthor= %s, favoriteAuthor= %s WHERE username= %s"), (password, fBT, fBA, fA, username)

cursor.execute('INSERT INTO readers VALUES (% s, % s, %s, % s, % s)', (username, password, fBT, fBA, fA))

mysql.connection.commit()

msg = 'You have successfully registered!'

elif request.method == 'POST':

msg = 'Please fill out the form !'

#return render\_template('register.html', msg = msg)

return render\_template('register.html', msg = msg)

@app.route("/index")

def index():

if 'loggedin' in session:

return render\_template("index.html")

return redirect(url\_for('login'))

@app.route("/display")

def display():

if 'loggedin' in session:

cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

cursor.execute('SELECT \* FROM readers WHERE username = % s', (session['username'], ))

account = cursor.fetchone()

return render\_template("display.html", account = account)

return redirect(url\_for('login'))

@app.route("/displayBooks", methods =['GET', 'POST'])

def displayBooks(): #Ll: taken from here [https://dev.mysql.com/doc/connector-python/en/connector-python-example-cursor-select.html]

#LL: also from [https://www.quora.com/How-can-I-display-a-table-from-a-database-in-a-web-application-using-Flask]

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute("SELECT \* FROM Books")

data = cursor.fetchall() #data from database

return render\_template("displayBooks.html", value=data)

@app.route("/viewAuthors", methods =['GET', 'POST'])

def viewAuthors():

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute("SELECT \* FROM Authors")

data = cursor.fetchall() #data from database

return render\_template("viewAuthors.html", value=data)

@app.route("/viewPublishers", methods =['GET', 'POST'])

def viewPublishers():

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute("SELECT \* FROM Publishers")

data = cursor.fetchall() #data from database

return render\_template("viewPublishers.html", value=data)

@app.route("/viewPublishes", methods =['GET', 'POST'])

def viewPublishes():

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute("SELECT \* FROM Publishes")

data = cursor.fetchall() #data from database

return render\_template("viewPublishes.html", value=data)

@app.route("/deleteAccount", methods =['GET', 'POST'])

def deleteAccount():

while "loggedin" in session:

cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

cursor.execute('DELETE FROM readers WHERE username = % s', (session['username'], ))

mysql.connection.commit()

msg = 'You have successfully deleted your account!'

session.pop('loggedin', None)

session.pop('username', None)

return render\_template("deleteAccount.html")

return redirect(url\_for('login'))

@app.route("/searchBooks", methods =['GET', 'POST'])

def searchBooks():

msg = ''

if request.method == 'POST':

#print("hi")

bA = request.form['bA'].strip()

bAD = request.form['bAD']

bT= request.form["bT"].strip()

bTD= request.form["bTD"]

bL= request.form["bL"].strip()

bLD= request.form["bLD"]

print([bA, bAD, bL])

if bA== "" or bAD== "":

p1= ""

if bT== "" or bTD=="":

p2= ""

if bL== "" or bLD=="":

p4= ""

# cursor.execute("UPDATE readers SET password='"+ password+ "', favoriteBookTitle='"+ fBT+ "',favoriteBookAuthor='"+fBA+ "',favoriteAuthor='"+ fA + "' WHERE username='"+username+"'")

if bA !="" and bAD !="":

p1= "authorName"+ bAD+ "'"+bA+ "'"+","

if bT != "" and bTD != "":

p2= "title"+ bTD+ "'"+bT+"'"+","

if bL != "" and bLD != "":

p4= "link"+ bLD+ "'"+ bL+ "'"

if p1 == "" and p2 == "" and p4 == "":

msg= "You did not correctly enter enough information for a search"

return render\_template('searchBooks.html', msg = msg)

s= ("SELECT \* FROM books WHERE "+p1 + p2 +p4).strip(",")

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

#cursor.execute('SELECT \* FROM books WHERE username = s', (session['username'], ))

#cursor.execute("SELECT \* FROM books WHERE authorName %s %s, title %s %s, averageRating %s %s, link %s %s"), (bAD, bA, bTD, bT, bRD, bR, bLD, bL)

cursor.execute(s)

data = cursor.fetchall() #data from database

return render\_template("searchBooks.html", value=data)

#s= ("UPDATE Students2.records SET ssn='"+ ssnVal+ "', firstName='"+ fNameVal+ "',lastName='"+lNameVal+ "',address='"+ addrVal + "',state='"+ stateVal+ "', zipcode= '"+ zipcodeVal+ "' WHERE id='"+keyVal+"'")

return render\_template('searchBooks.html', msg = msg)

@app.route("/searchAuthors", methods =['GET', 'POST'])

def searchAuthors():

msg = ''

if request.method == 'POST':

aN = request.form['aN'].strip()

aND = request.form['aND']

aB= request.form["aB"].strip()

aBD= request.form["aBD"]

aD= request.form["aD"].strip()

aDD= request.form["aDD"]

#print([bA, bAD, bL])

if aN== "" or aND== "":

p1= ""

if aB== "" or aBD=="":

p2= ""

if aD== "" or aDD=="":

p3= ""

if aN !="" and aND !="":

p1= "name"+ aND+ "'"+aN+ "'"+","

if aB != "" and aBD != "":

p2= "yearOfBirth"+ aBD+ "'"+aB+"'"+","

if aD != "" and aDD != "":

p3= "yearOfDeath"+ aDD+ "'" + aD+ "'" +","

if p1 == "" and p2 == "" and p3 == "":

msg= "You did not correctly enter enough information for a search"

return render\_template('searchAuthors.html', msg = msg)

s= ("SELECT \* FROM authors WHERE "+p1+ p2+ p3).strip(",")

print(s)

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute(s)

data = cursor.fetchall() #data from database

return render\_template("searchAuthors.html", value=data)

return render\_template('searchAuthors.html', msg = msg)

@app.route("/searchPublishers", methods =['GET', 'POST'])

def searchPublishers():

msg = ''

if request.method == 'POST':

#print(request.method)

#print("hi")

pN = request.form['pN'].strip()

pND = request.form['pND']

pO= request.form["pO"].strip()

pOD= "="

#print([bA, bAD, bL])

if pN== "" or pND== "":

p1= ""

if pO== "" or pOD=="":

p2= ""

# cursor.execute("UPDATE readers SET password='"+ password+ "', favoriteBookTitle='"+ fBT+ "',favoriteBookAuthor='"+fBA+ "',favoriteAuthor='"+ fA + "' WHERE username='"+username+"'")

if pN !="" and pND !="":

p1= "name"+ pND+ "'"+pN+ "'"+","

if pO != "" and pOD != "":

p2= "stillAround = "+ "'"+pO+"'"

if p1 == "" and p2 == "":

msg= "You did not correctly enter enough information for a search"

return render\_template('searchPublishers.html', msg = msg)

s= ("SELECT \* FROM publishers WHERE "+p1+ p2).strip(",")

print(s)

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute(s)

data = cursor.fetchall() #data from database

return render\_template("searchPublishers.html", value=data)

return render\_template('searchPublishers.html', msg = msg)

@app.route("/searchPublishes", methods =['GET', 'POST'])

def searchPublishes():

msg = ''

if request.method == 'POST':

print(request.form)

#print("hi")

aN = request.form['aN'].strip()

aND = request.form['aND']

bT= request.form["bT"].strip()

bTD= request.form["bTD"]

pN= request.form["pN"].strip()

pND= request.form["pND"]

y= request.form["y"].strip()

yD= request.form["yD"]

l= request.form["l"].strip()

lD= request.form["lD"]

if aN== "" or aND== "":

p1= ""

if bT== "" or bTD=="":

p2= ""

if pN== "" or pND=="":

p3= ""

if y== "" or yD=="":

p4= ""

if l== "" or lD== "":

p5= ""

if aN !="" and aND !="":

p1= "authorName"+ aND+ "'"+aN+ "'"+","

if bT != "" and bTD != "":

p2= "title"+ bTD+ "'"+bT+"'"+","

if pN != "" and pND != "":

p3= "publisherName"+ pND+ "'" + pN+ "'" +","

if y != "" and yD != "":

p4= "yearOfPublication"+ yD+ "'"+ y+ "'" + ","

if l != "" and lD != "":

p5= "location"+ lD+ "'"+ l+ "'"

if p1 == "" and p2 == "" and p3 == "" and p4 == "" and p5 == "":

msg= "You did not correctly enter enough information for a search"

return render\_template('searchPublishes.html', msg = msg)

s= ("SELECT \* FROM publishes WHERE "+p1 + p2 + p3 + p4 + p5).strip(",")

print(s)

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute(s)

data = cursor.fetchall() #data from database

return render\_template("searchPublishes.html", value=data)

#s= ("UPDATE Students2.records SET ssn='"+ ssnVal+ "', firstName='"+ fNameVal+ "',lastName='"+lNameVal+ "',address='"+ addrVal + "',state='"+ stateVal+ "', zipcode= '"+ zipcodeVal+ "' WHERE id='"+keyVal+"'")

return render\_template('searchPublishes.html', msg = msg)

@app.route("/viewRatings", methods =['GET', 'POST'])

def viewRatings():

conn= MySQLdb.connect("localhost", user, password, "Books")

cursor = conn.cursor()

cursor.execute("SELECT authorName, title, AVG(rating) FROM rate GROUP BY authorName, title")

data = cursor.fetchall() #data from database

return render\_template("viewRatings.html", value=data)

@app.route("/rateBooks", methods =['GET', 'POST'])

def rateBooks():

msg = ''

print(request.form)

if request.method == 'POST' and 'authorName' in request.form and "title" in request.form and "rating" in request.form:

username = session['username']

aN= request.form["authorName"].strip()

t= request.form["title"].strip()

r= request.form["rating"].strip()

cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

cursor.execute("SELECT DISTINCT name FROM Authors")

val= cursor.fetchall()

authorList= []

for i in val:

authorList.append(list(i.values())[0])

cursor.execute("SELECT DISTINCT title FROM books")

fBookList= list()

val= cursor.fetchall()

for i in val:

fBookList.append(list(i.values())[0])

if (t == "") and (t not in list(fBookList)):

msg= "The book you entered is not in our database."

return render\_template('rateBooks.html', msg = msg)

if (aN == "") and (aN not in list(fBookList)):

msg= "The author you entered is not in our database."

return render\_template('rateBooks.html', msg = msg)

cursor.execute('SELECT authorName FROM Books WHERE title = % s', (t, ))

rAN= list(cursor.fetchall())

if aN != rAN:

msg= "The author you entered did not write the book you entered."

return render\_template('rateBooks.html', msg = msg)

cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

# cursor.execute("INSERT INTO readers(username, password) VALUES (%s, %s)", (username, password))

cursor.execute('INSERT INTO rate VALUES (%s, %s, %s, %s)', (username, aN, t, r))

mysql.connection.commit()

msg = 'You have successfully submited your rating!'

elif request.method == 'POST':

msg = 'Please fill out the form !'

return render\_template('rateBooks.html', msg = msg)

@app.route("/update", methods =['GET', 'POST'])

def update():

msg = ''

if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

# print("request form favebooktitle")

# print(request.form["favoriteBookTitle"])

# print("empty string??")

# print(request.form["favoriteBookTitle"]== "")

username = request.form['username'].strip()

password = request.form['password'].strip()

fBT= request.form["favoriteBookTitle"].strip()

fBA= request.form["favoriteBookAuthor"].strip()

fA= request.form["favoriteAuthor"].strip()

# email = request.form['email']

# organisation = request.form['organisation']

# address = request.form['address']

# city = request.form['city']

# state = request.form['state']

# country = request.form['country']

# postalcode = request.form['postalcode']

cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

cursor.execute('SELECT \* FROM readers WHERE username = % s', (username, ))

account = cursor.fetchone()

# if account:

# msg = 'Account already exists !'

# elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

# msg = 'Invalid email address !'

# elif not re.match(r'[A-Za-z0-9]+', username):

# msg = 'name must contain only characters and numbers !'

# else:

#s= "INSERT INTO readers VALUES (%s, %s)", (username, password)

if fA == "":

fA= "None"

if fBT =="":

fBT= "None"

if fBA =="":

fBA= "None"

if "password" =="":

msg= "You need to enter a value for your password"

return render\_template('update.html', msg = msg)

# s= ("UPDATE Students2.records SET ssn='"+ ssnVal+ "', firstName='"+ fNameVal+ "',lastName='"+lNameVal+ "',address='"+ addrVal + "',state='"+ stateVal+ "', zipcode= '"+ zipcodeVal+ "' WHERE id='"+keyVal+"'")

#cursor.execute('UPDATE accounts SET username =% s, password =% s, email =% s, organisation =% s, address =% s, city =% s, state =% s, country =% s, postalcode =% s WHERE id =% s', (username, password, email, organisation, address, city, state, country, postalcode, (session['id'], ), ))

print([username, password, fBT, fBA, fA])

#cursor.execute("UPDATE readers SET password= %s, favoriteBookTitle= %s, favoriteBookAuthor= %s, favoriteAuthor= %s WHERE username= %s"), (password, fBT, fBA, fA, username)

cursor.execute("UPDATE readers SET password='"+ password+ "', favoriteBookTitle='"+ fBT+ "',favoriteBookAuthor='"+fBA+ "',favoriteAuthor='"+ fA + "' WHERE username='"+username+"'")

mysql.connection.commit()

msg = 'You have successfully updated your profile!'

# =============================================================================

# if "favoriteAuthor" != "" and ("favoriteBookTitle" == "" and "favoriteBookAuthor" == ""):

# #cursor.execute('INSERT INTO readers(username, password, favoriteAuthor) VALUES (% s, % s, % s)', (username, password, fA))

# print("1")

# cursor.execute("UPDATE readers SET password= %s, favoriteAuthor= %s WHERE username= %s"), (password, fA, username)

# mysql.connection.commit()

# msg = 'You have successfully registered !'

#

# elif "favoriteAuthor" == "" and ("favoriteBookTitle" != "" and "favoriteBookAuthor" != ""):

# #cursor.execute('INSERT INTO readers(username, password, favoriteBookTitle, favoriteBookAuthor) VALUES (% s, % s, %s, % s)', (username, password, fBT, fBA))

# print("2")

# cursor.execute("UPDATE readers SET password= %s, favoriteBookTitle= %s, favoriteBookAuthor= %s WHERE username= %s"), (password, fBT, fBA, username)

# mysql.connection.commit()

# msg = 'You have successfully registered !'

#

# elif ("favoriteBookTitle" != "" and "favoriteBookAuthor" == ""):

# msg= "You have not fully entered your favorite book"

#

# elif ("favoriteBookTitle" == "" and "favoriteBookAuthor" != ""):

# msg= "You have not fully entered your favorite book"

#

# elif "favoriteAuthor" == "" and ("favoriteBookTitle" == "" and "favoriteBookAuthor" == ""):

# print("3")

# #cursor.execute("INSERT INTO readers(username, password) VALUES (%s, %s)", (username, password))

# cursor.execute("UPDATE readers SET password= %s WHERE username= %s"), (password, username)

# mysql.connection.commit()

# msg = 'You have successfully registered !'

#

#

# else:

# # s= ("UPDATE Students2.records SET ssn='"+ ssnVal+ "', firstName='"+ fNameVal+ "',lastName='"+lNameVal+ "',address='"+ addrVal + "',state='"+ stateVal+ "', zipcode= '"+ zipcodeVal+ "' WHERE id='"+keyVal+"'")

# #cursor.execute('INSERT INTO readers VALUES (% s, % s, %s, % s, % s)', (username, password, fBT, fBA, fA))

# print("4")

# cursor.execute("UPDATE readers SET password= '"+password+"',favoriteBookTitle= '"+ fBT+ "',favoriteBookAuthor= '"+ fBA+ "',favoriteAuthor= '"+ fA+ "' WHERE username= '"+ username+"'")

# mysql.connection.commit()

# msg = 'You have successfully registered !'

# =============================================================================

elif request.method == 'POST':

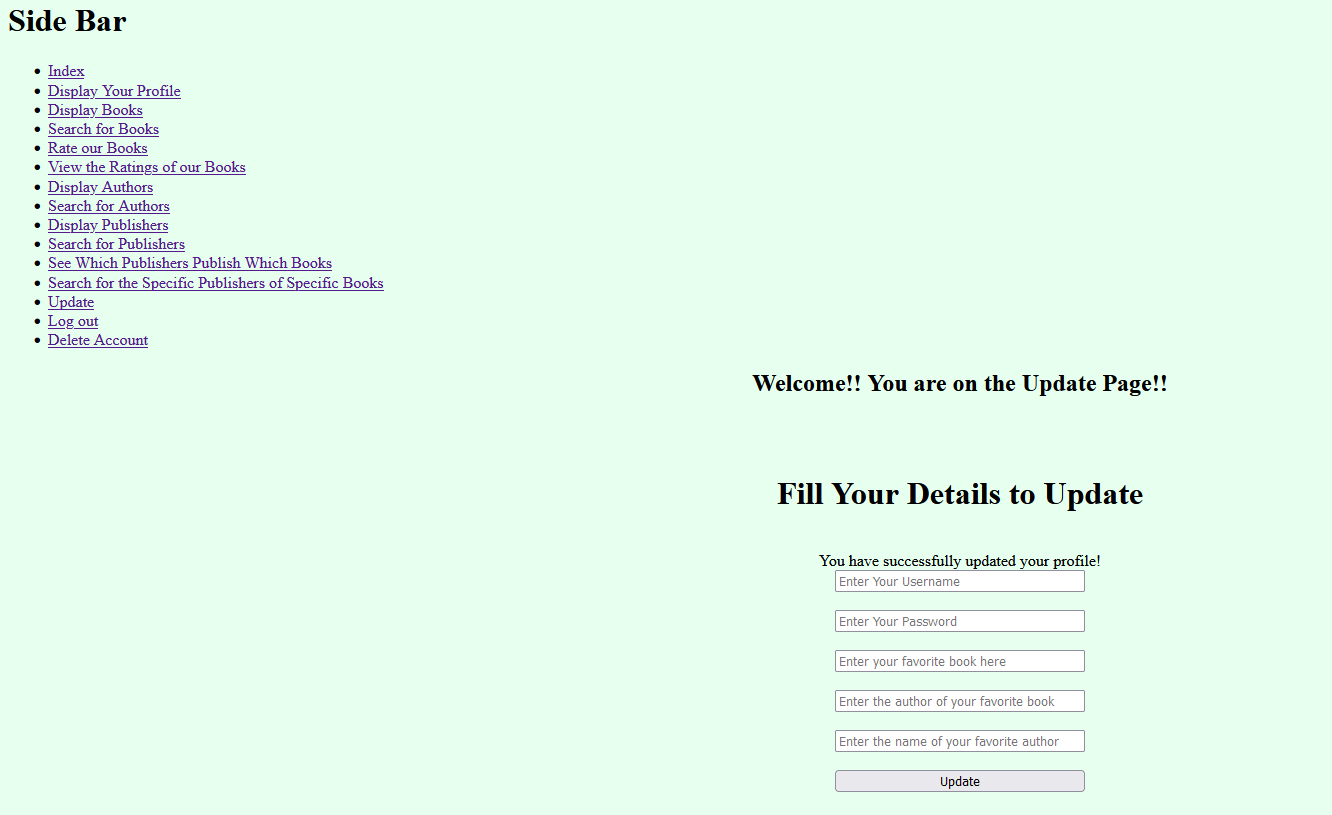
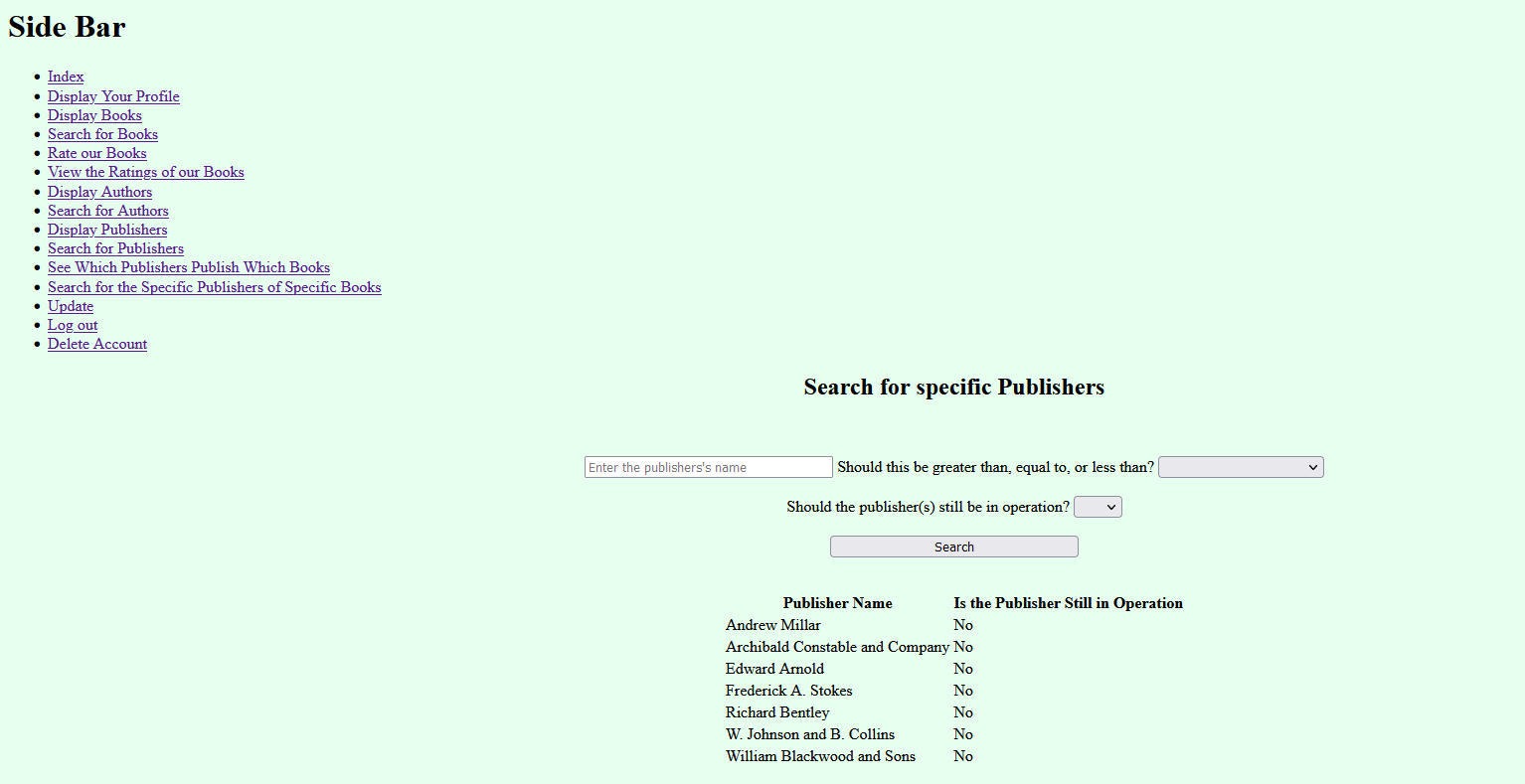
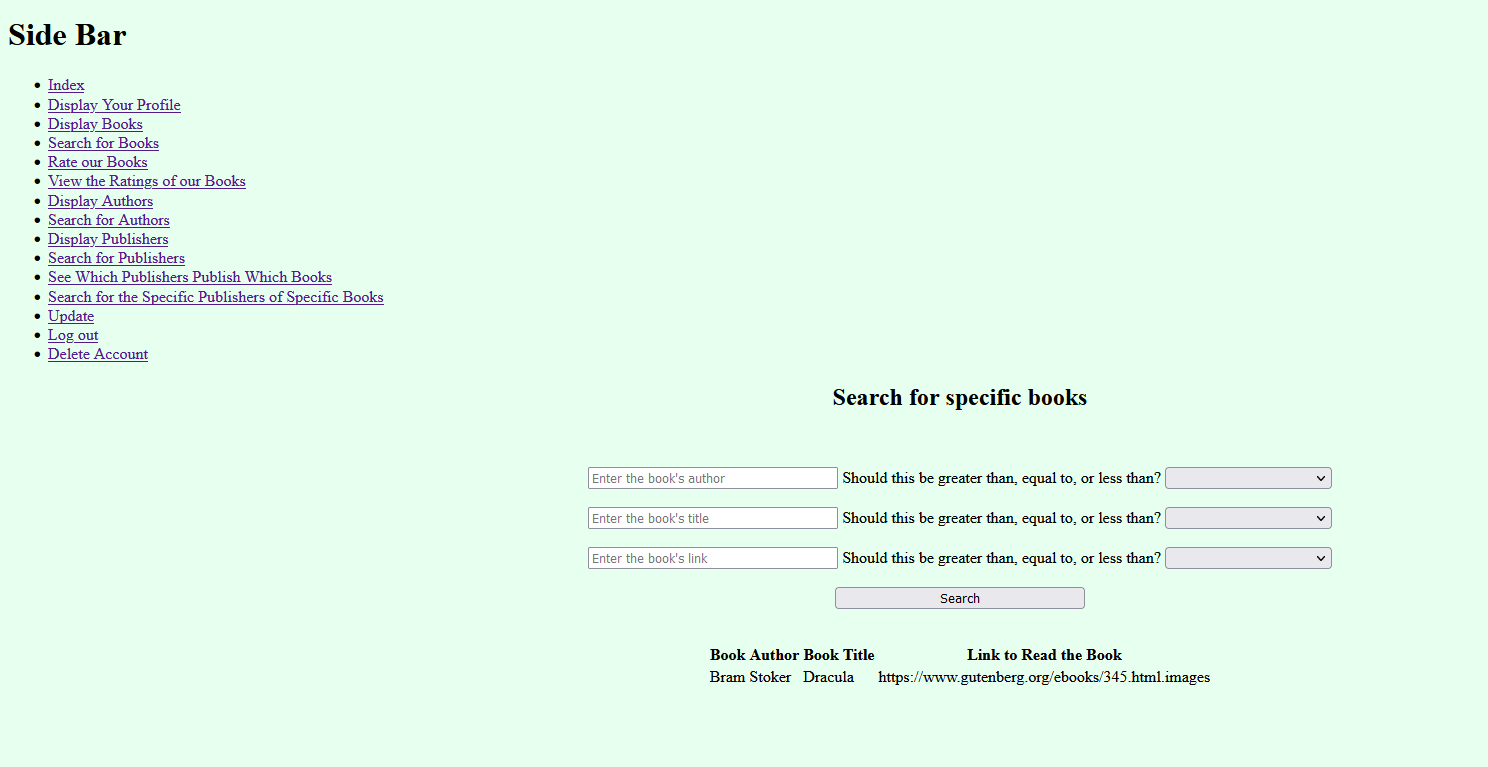
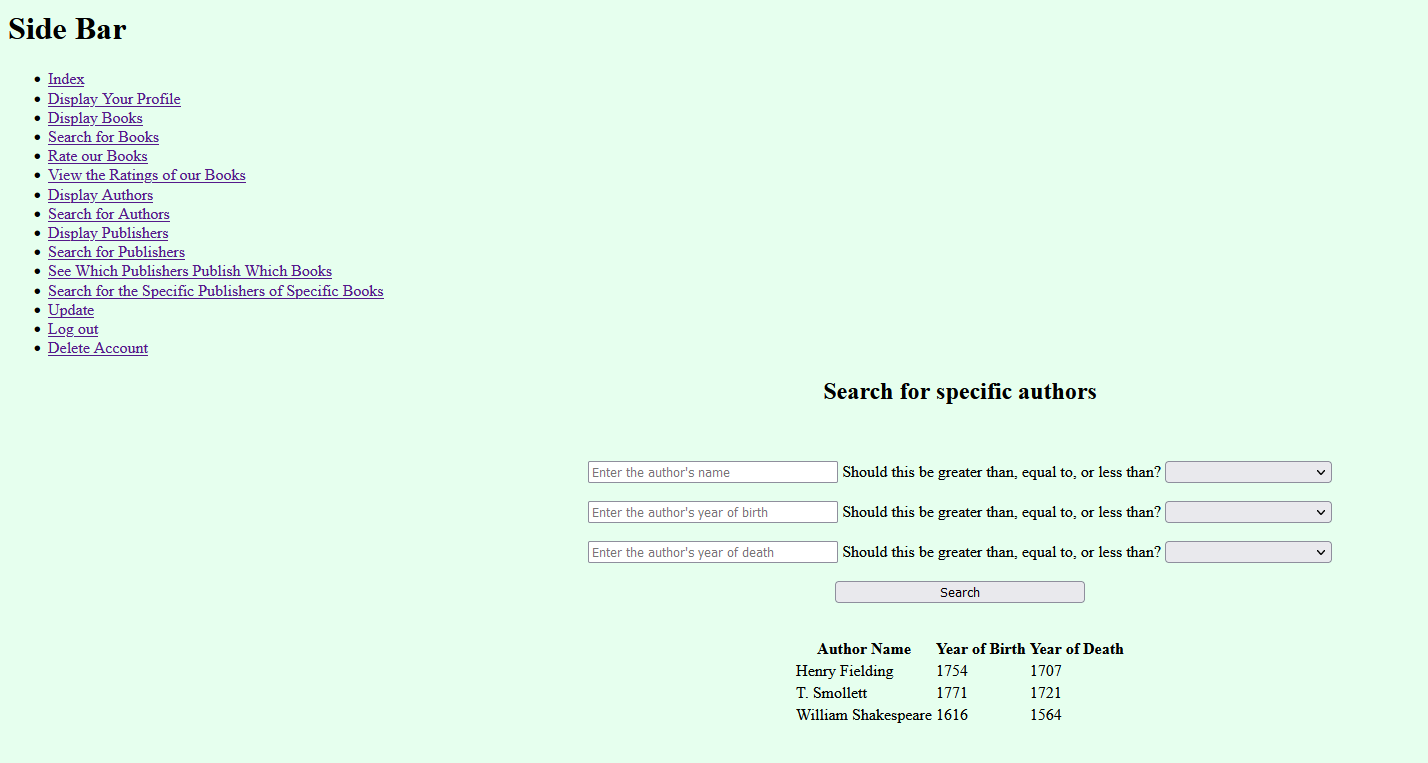
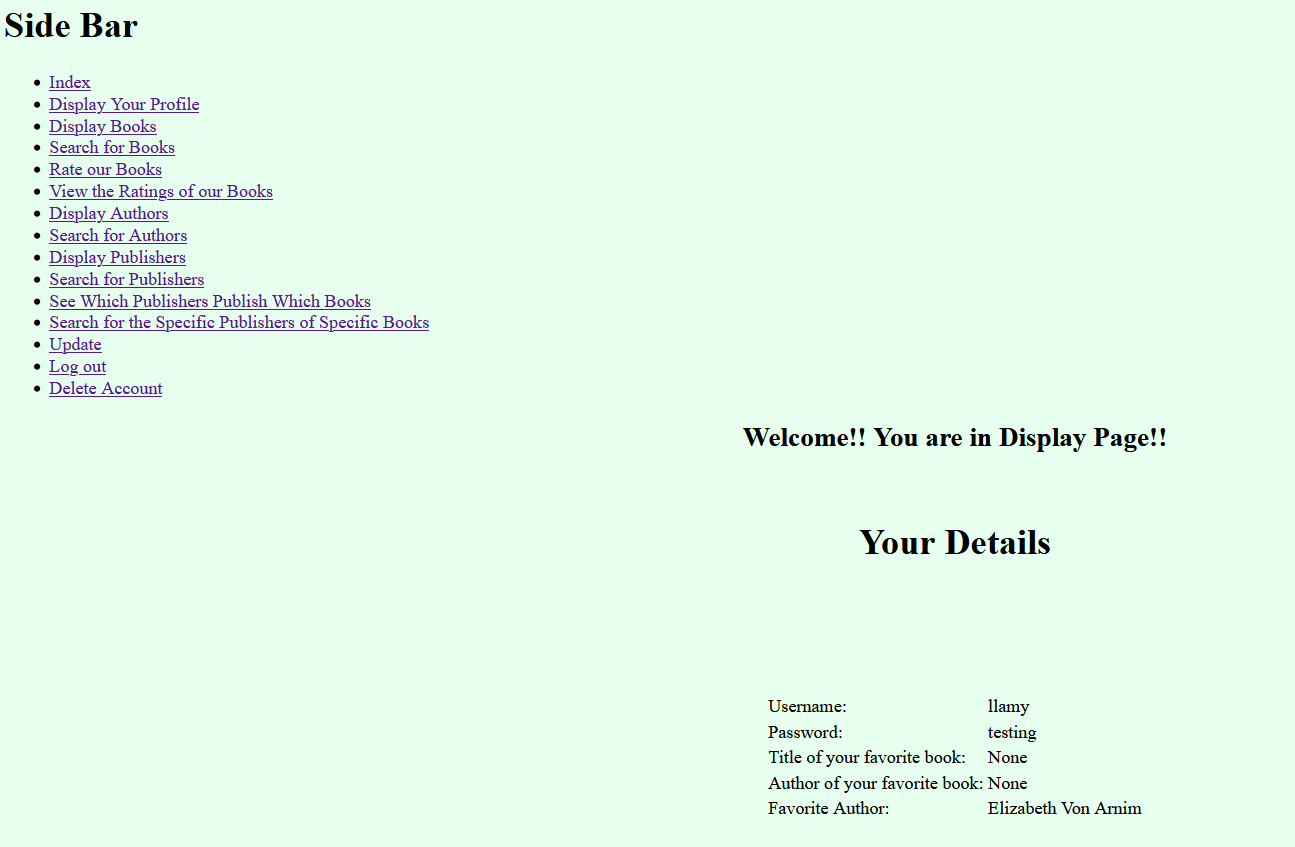
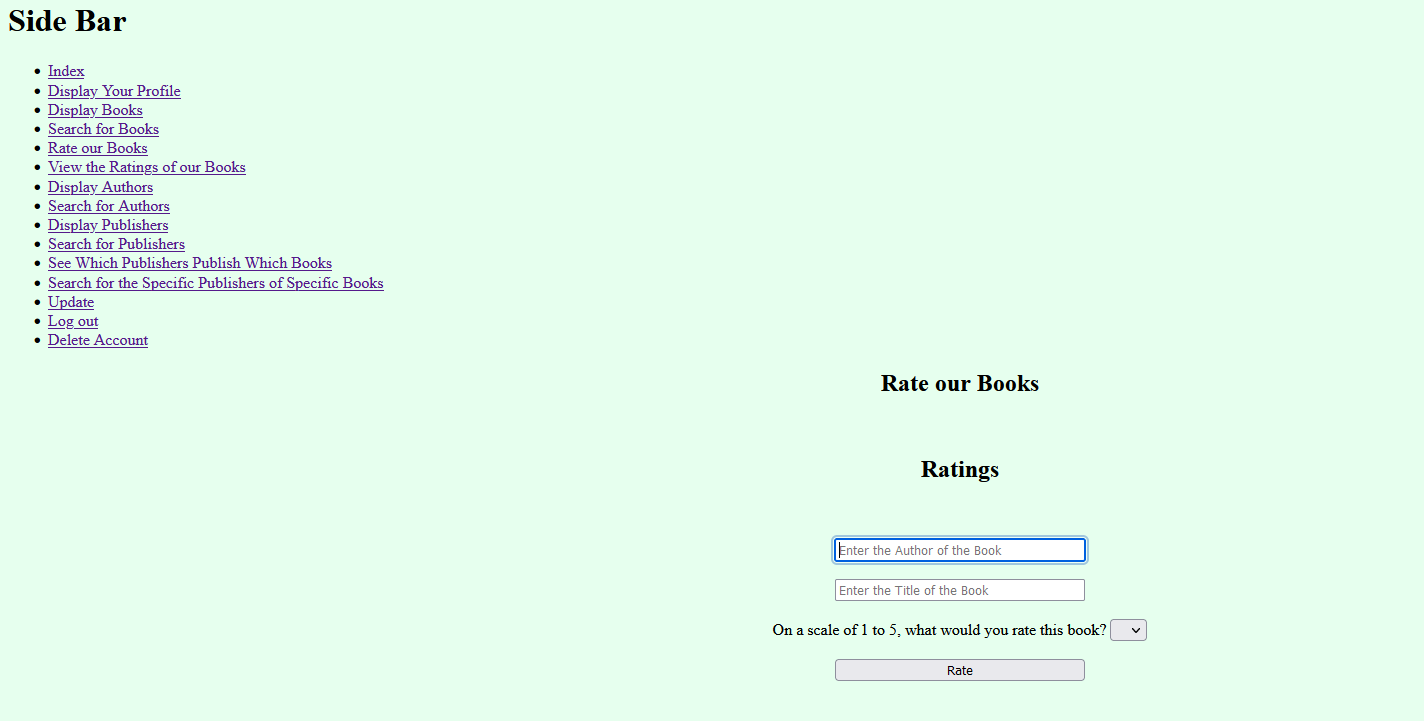
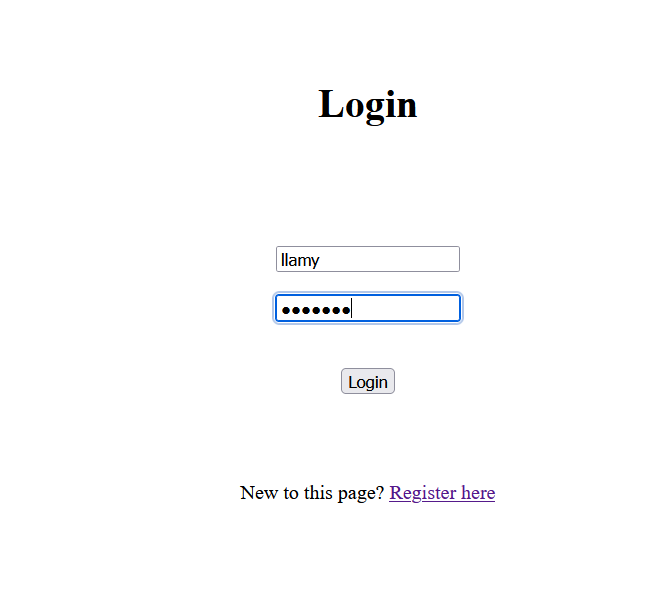
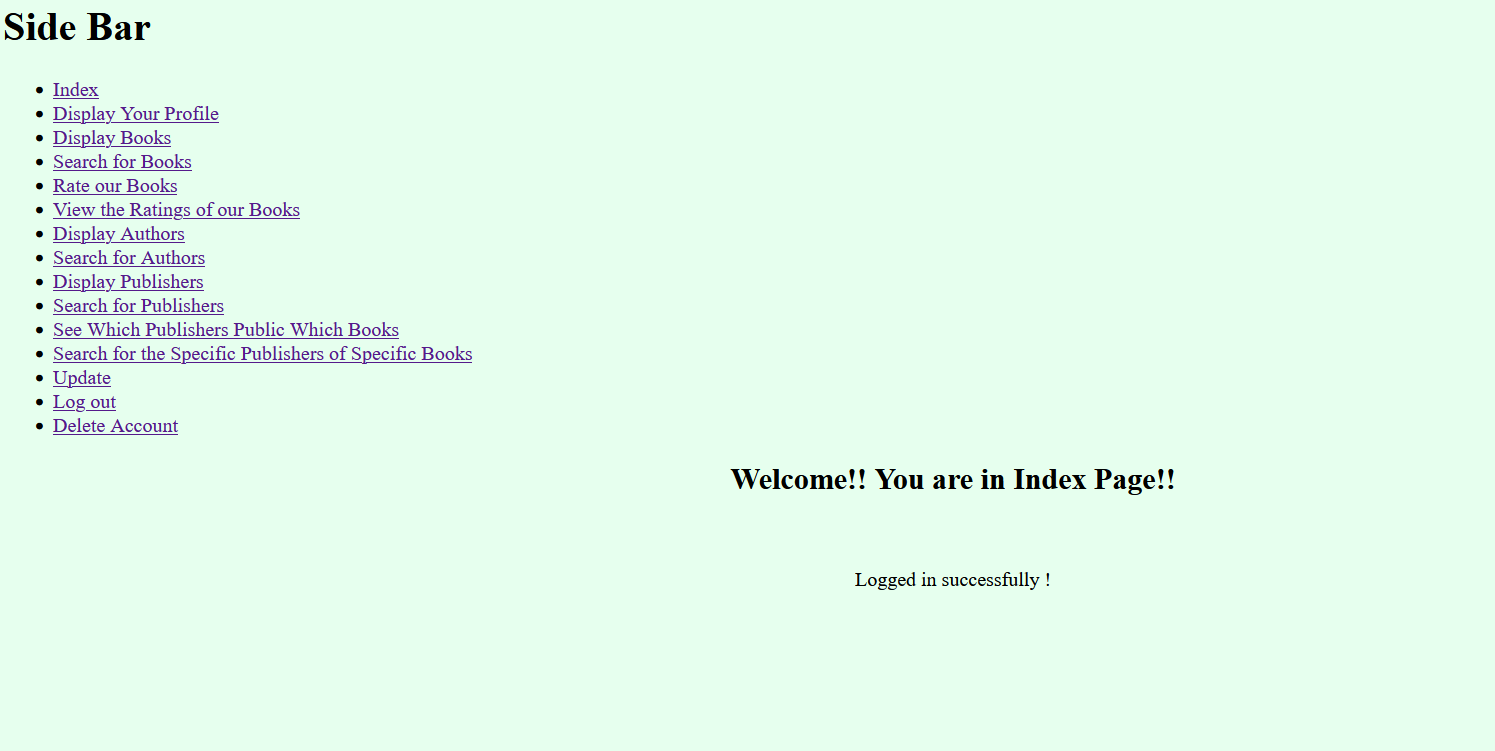
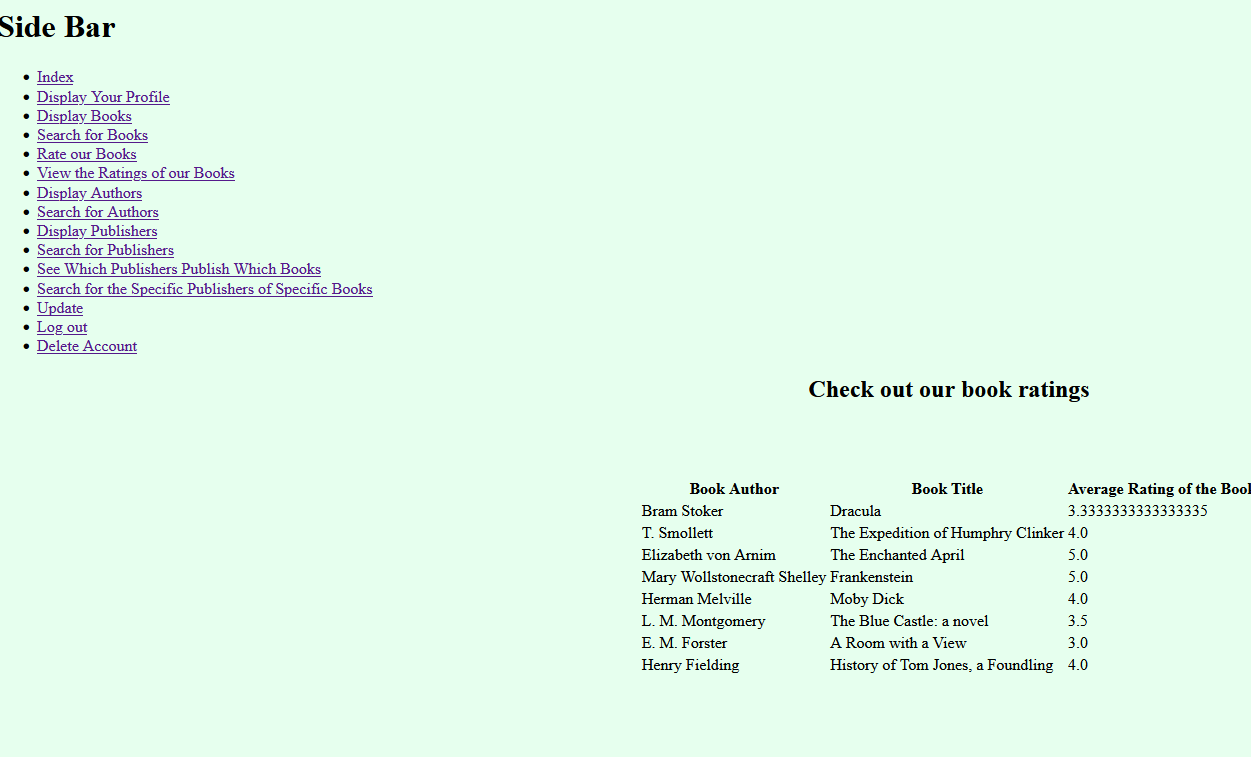
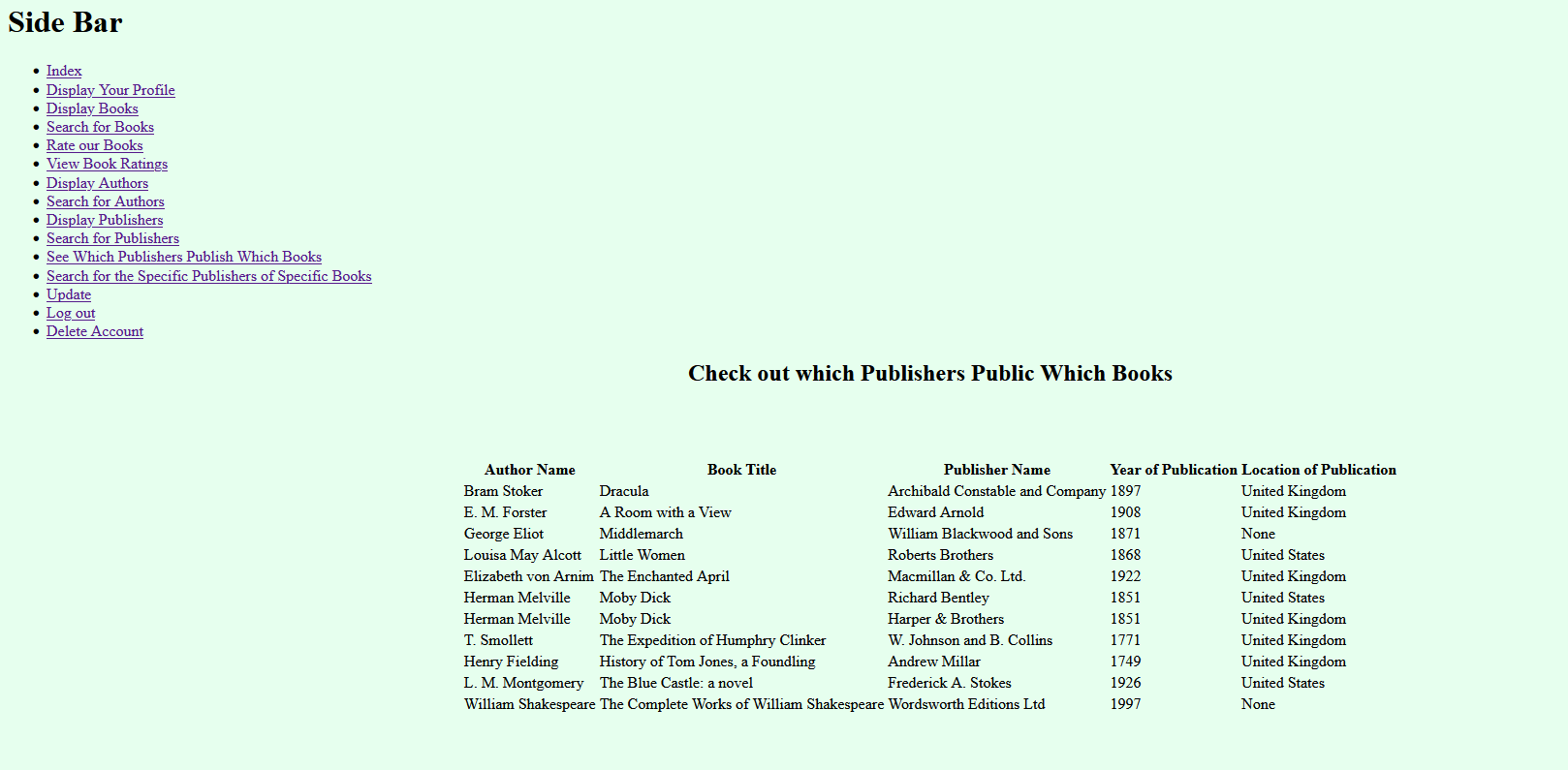
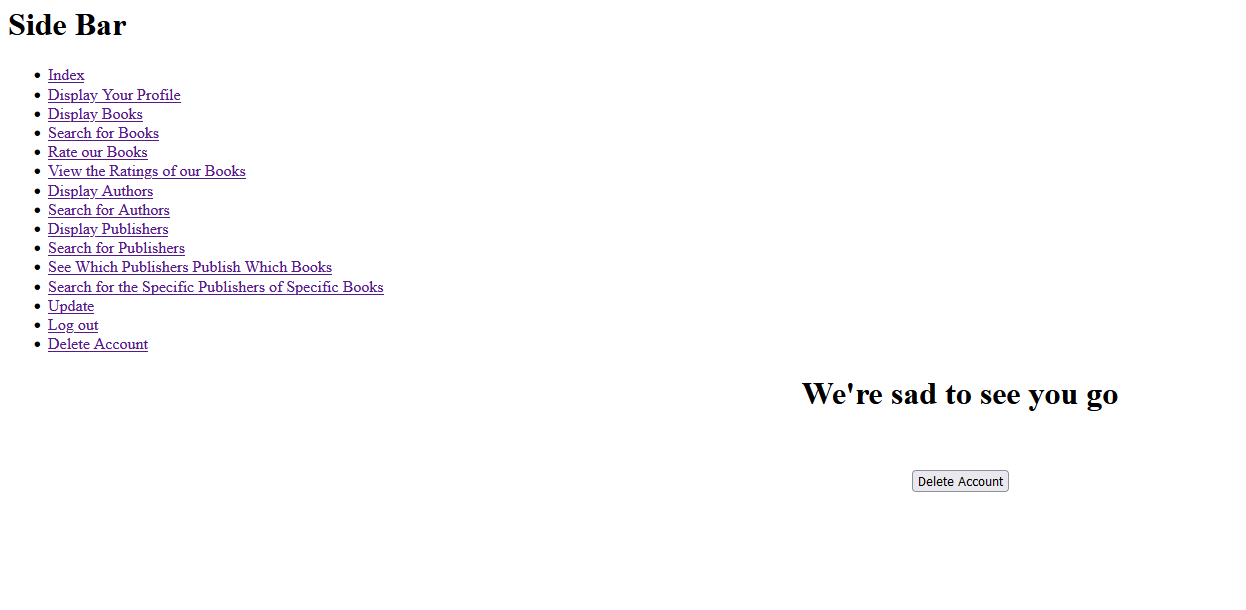
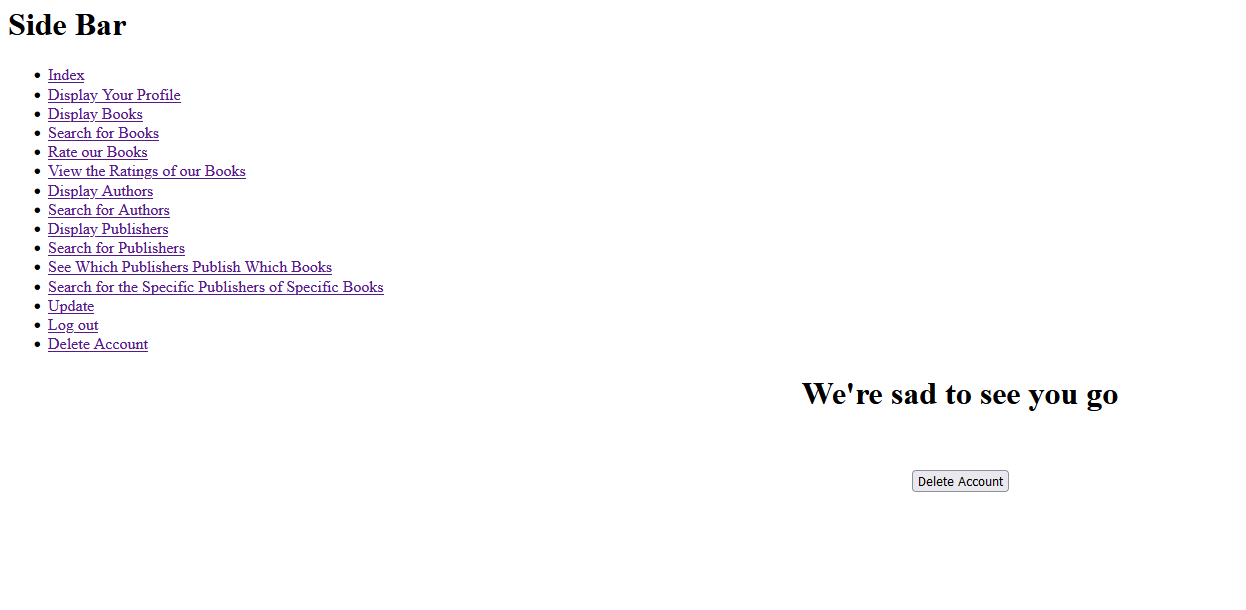
msg = 'Please fill out the form !'

return render\_template('update.html', msg = msg)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host ="localhost", port = int("5000"))

# Screenshots



# URL Link:

<https://github.com/llamb44/DB_finalProject.git>

Note, the folder that I’m using for this is so big that I am not able to add all the files to set up the environment. The csvs, sql schema, report, and the source code will be included but if you try to run it and it doesn’t run, then this is why.