***Open Lab Project Report***

**Name: Book Corner**

**Description:** Online bookstore where you can browse books of various genres and place orders.

**Done by:**

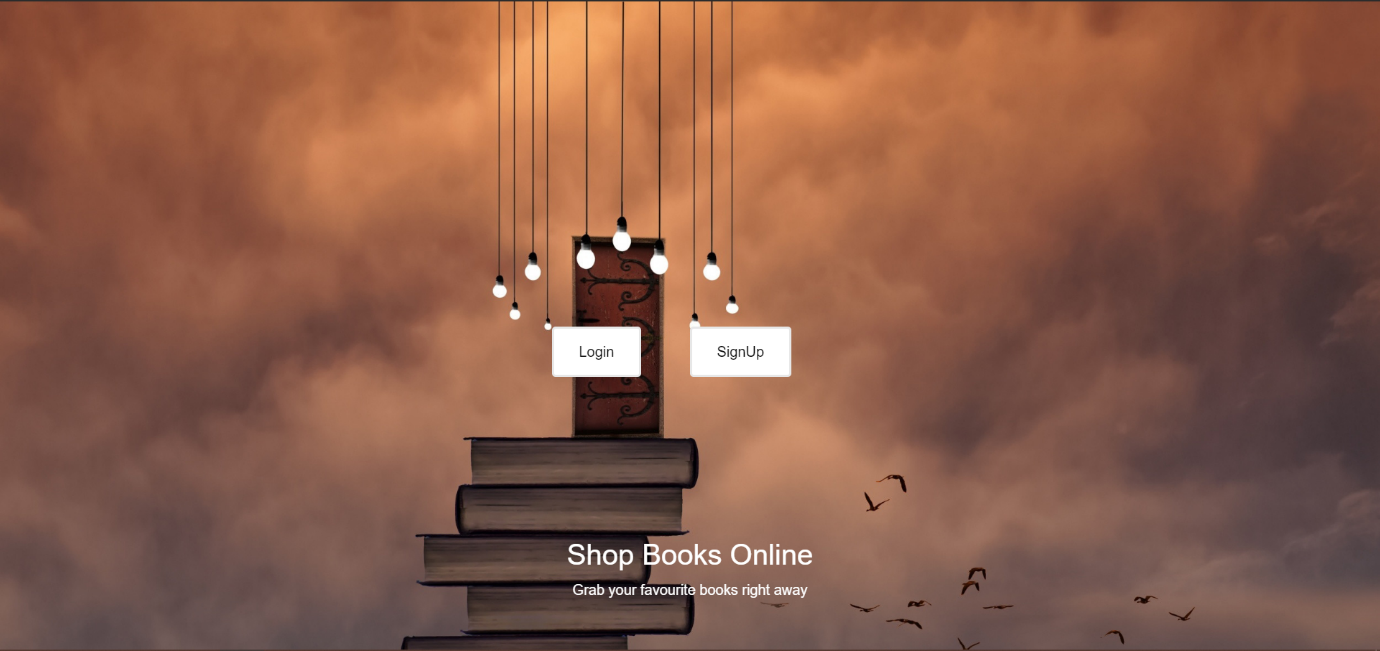
Niveth Saran V J (CB.EN.U4CSE17337)

Website: <https://tinyurl.com/BookCornerNive>

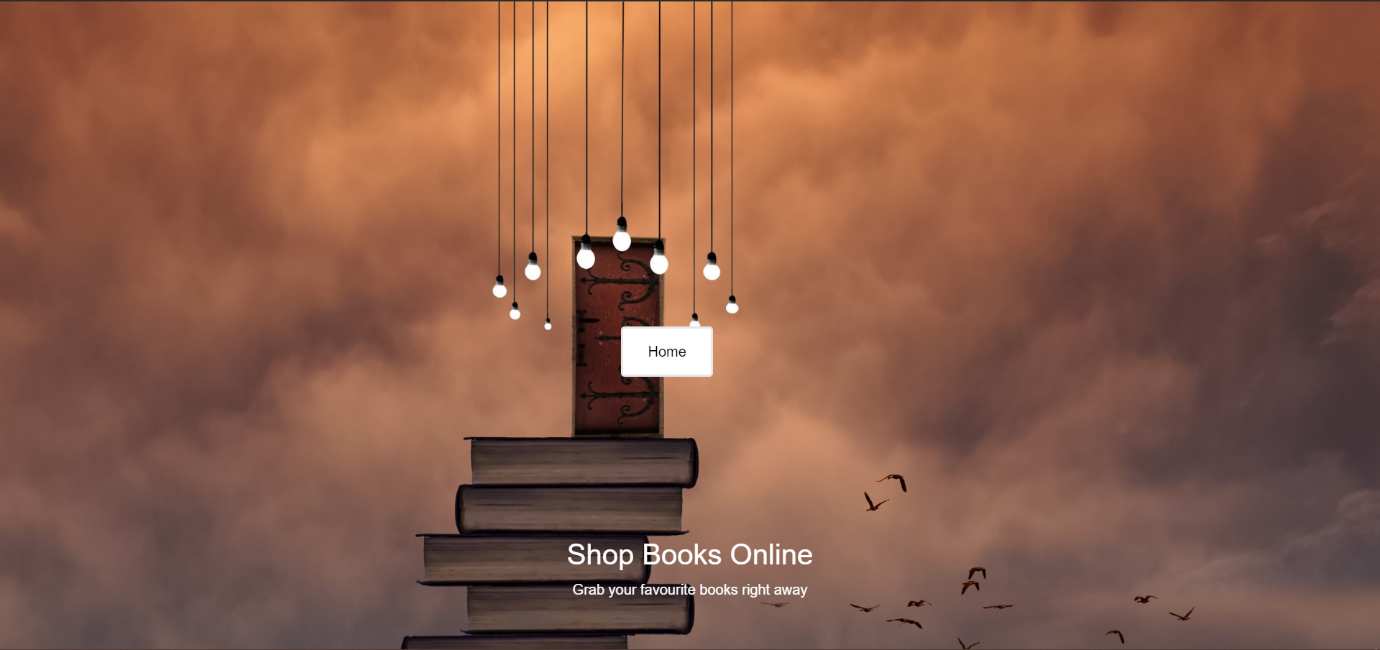
Codebase: [https://tinyurl.com/BookCornerCodeBase](https://tinyurl.com/bookcornercodebase)

**Screenshots:**

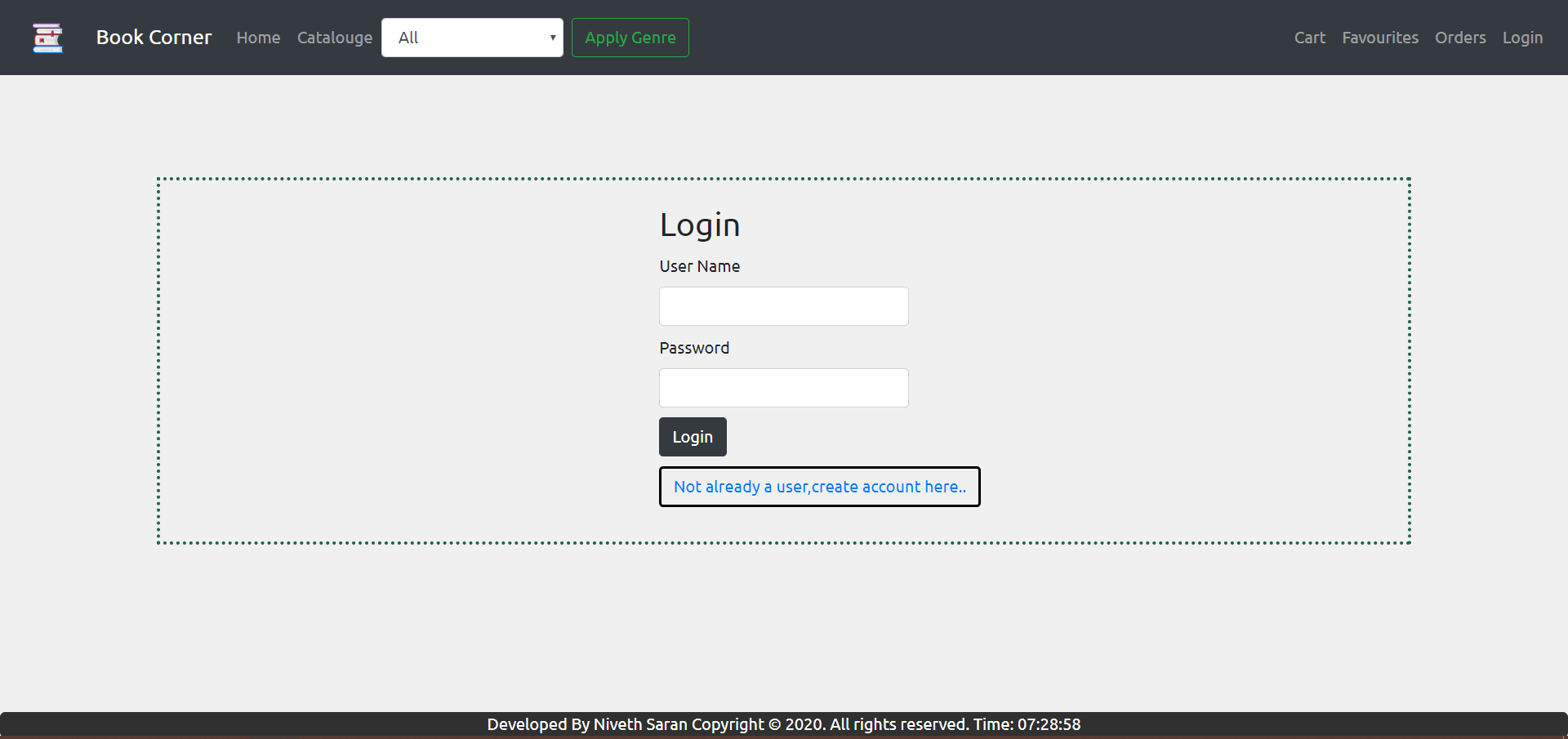
**Start-up page when logged out:**

****

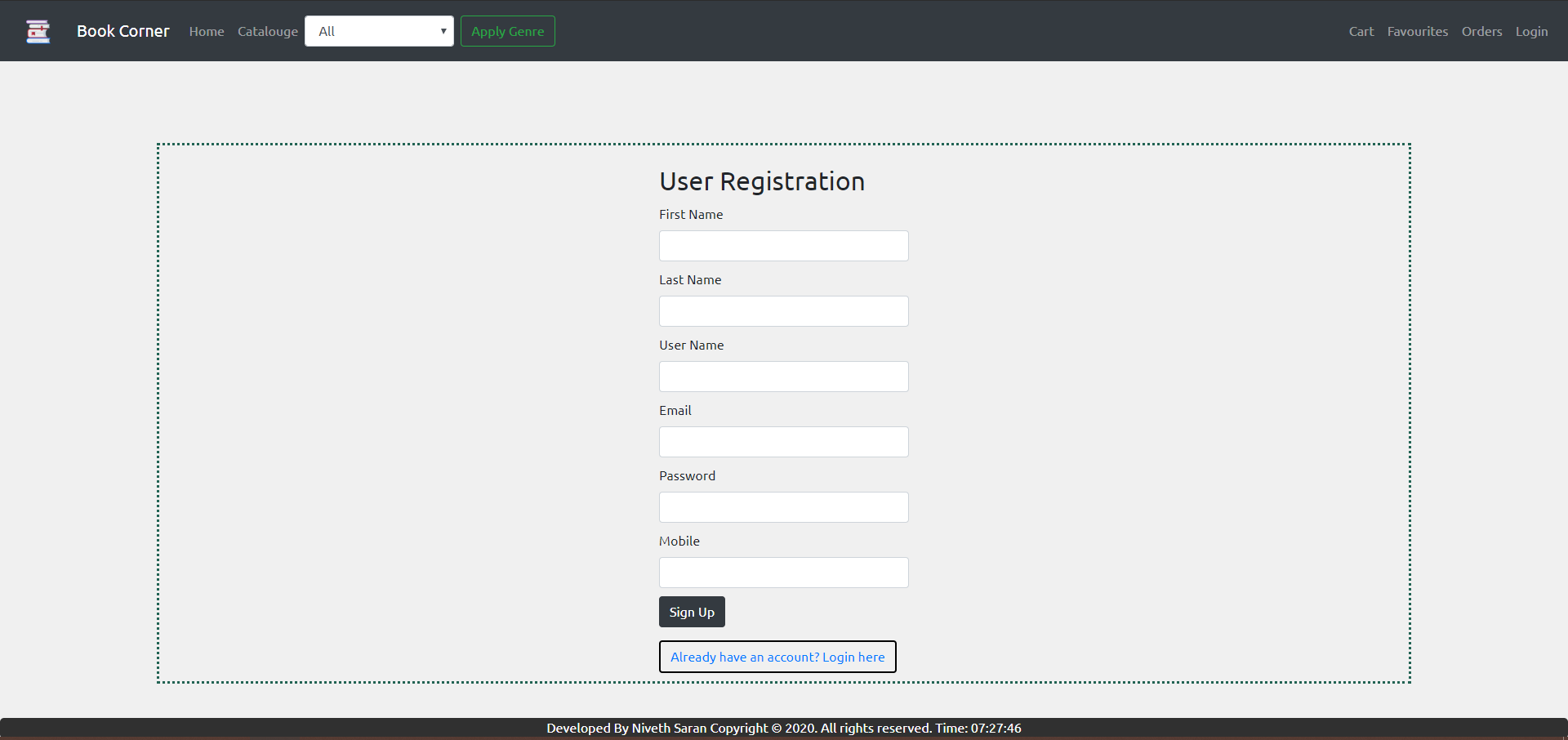
**Start-up page when logged in:**

****

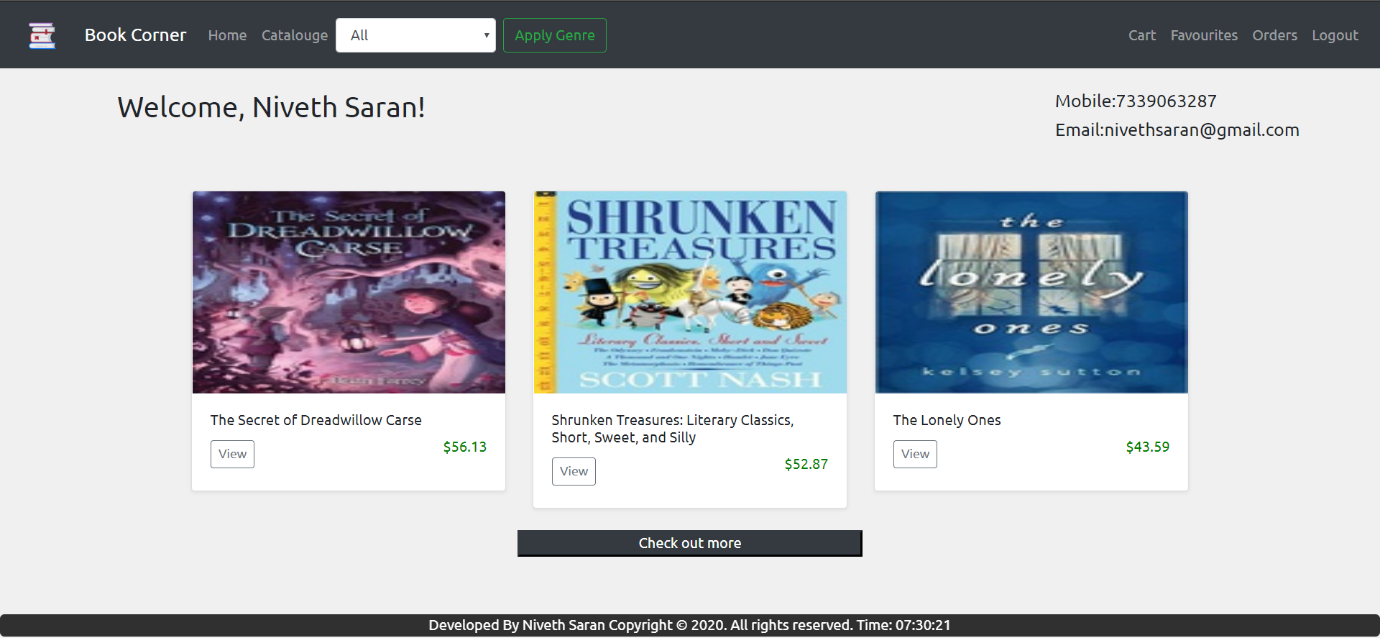
**Sign-In Page:**

****

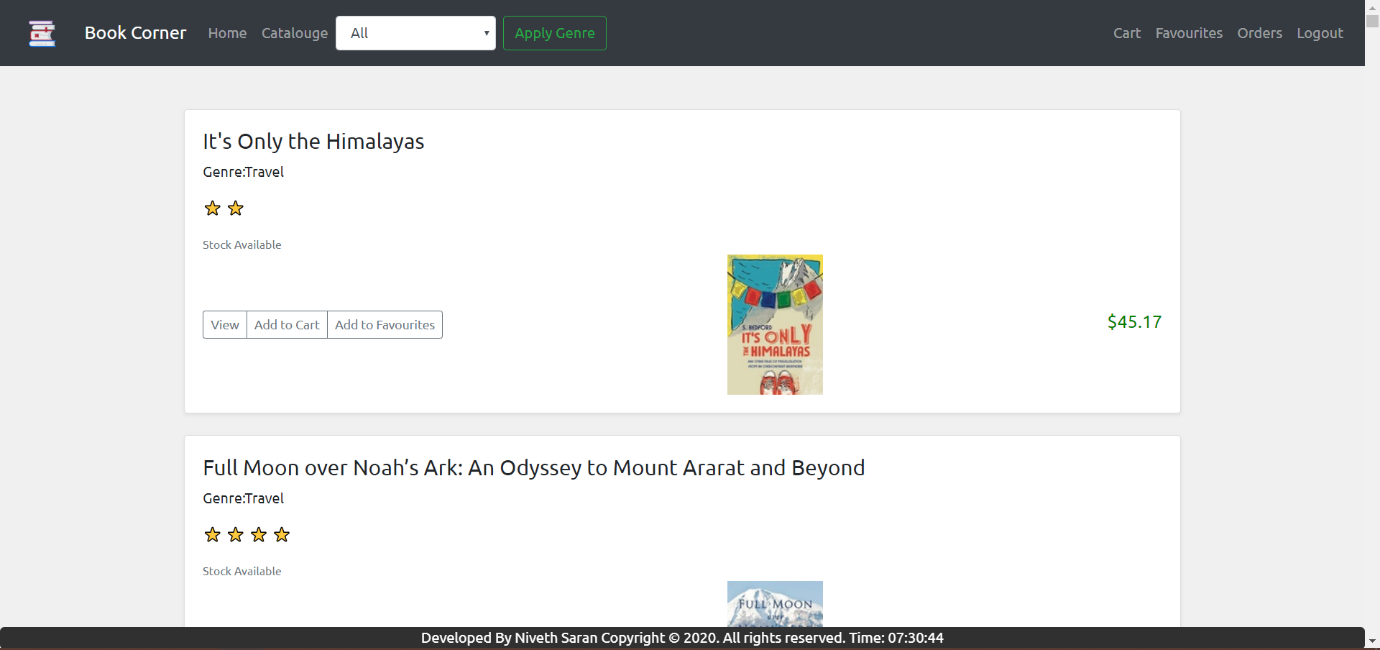
**Sign-Up Page:**

****

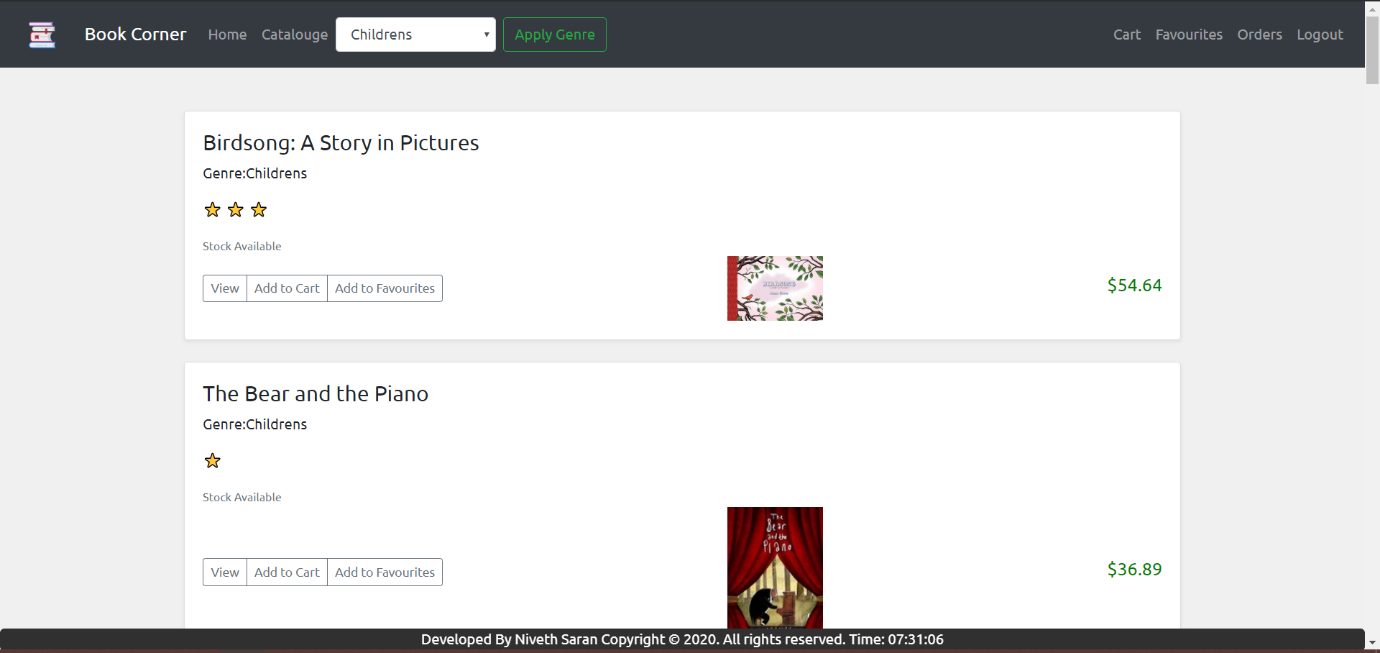
**Home Page:**

****

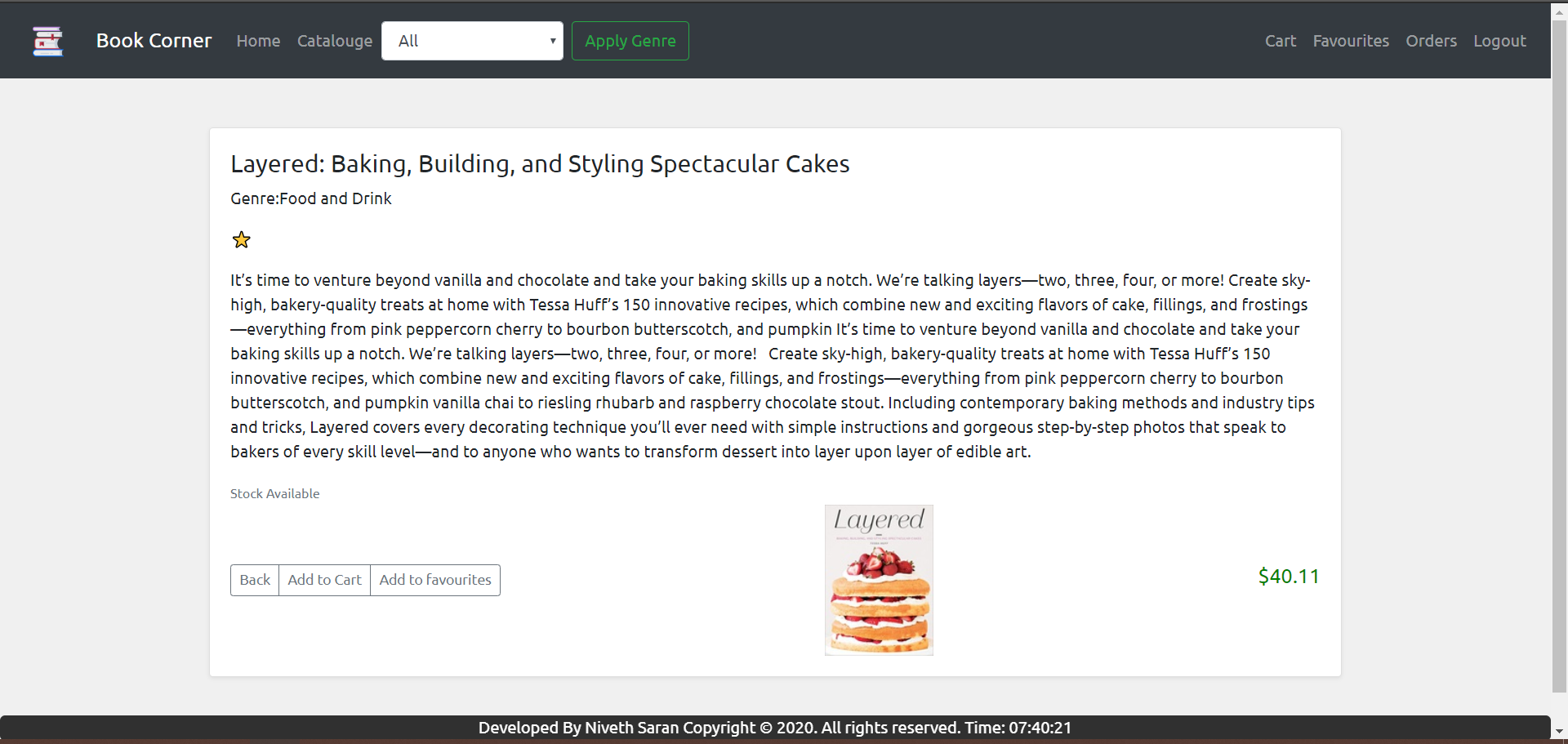
**Catalogue Page:**

****

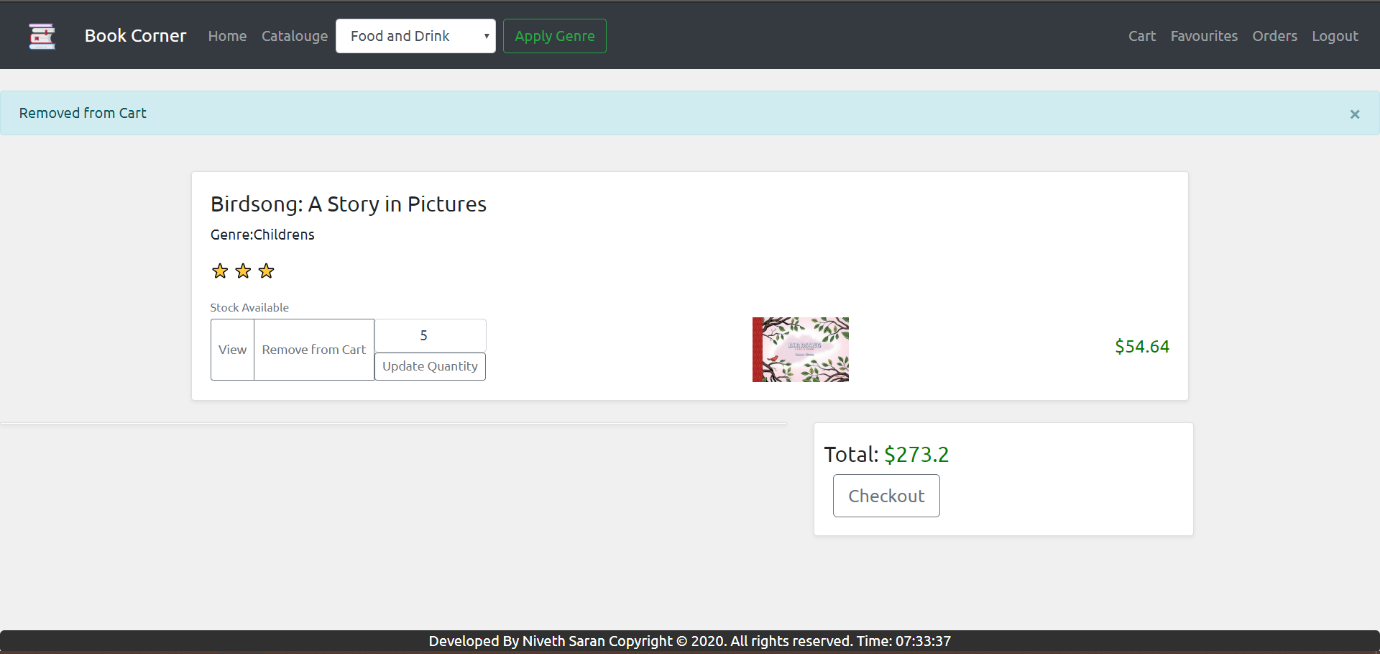
**Catalogue with Genre applied:**

****

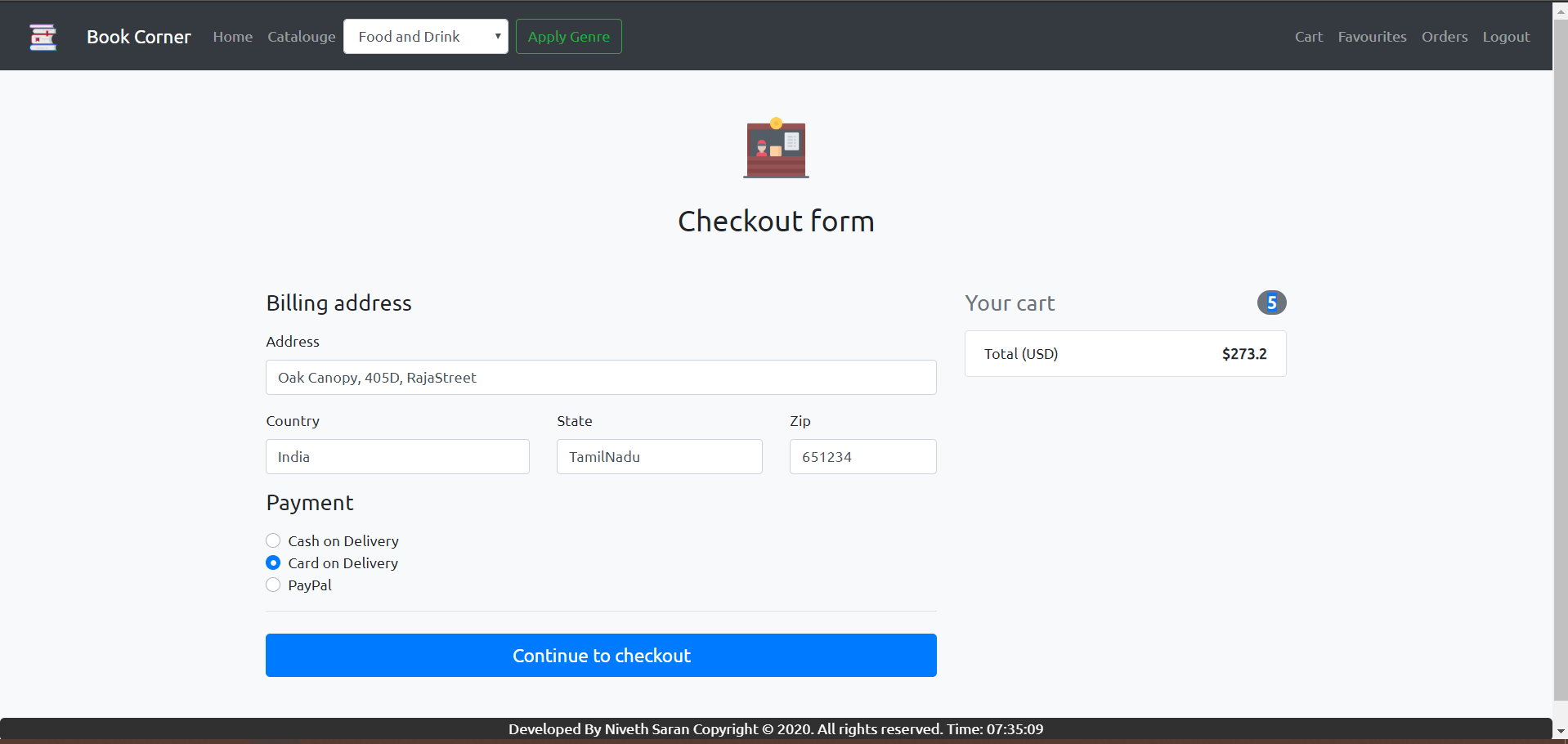
**View Book or Books Details Page:**

****

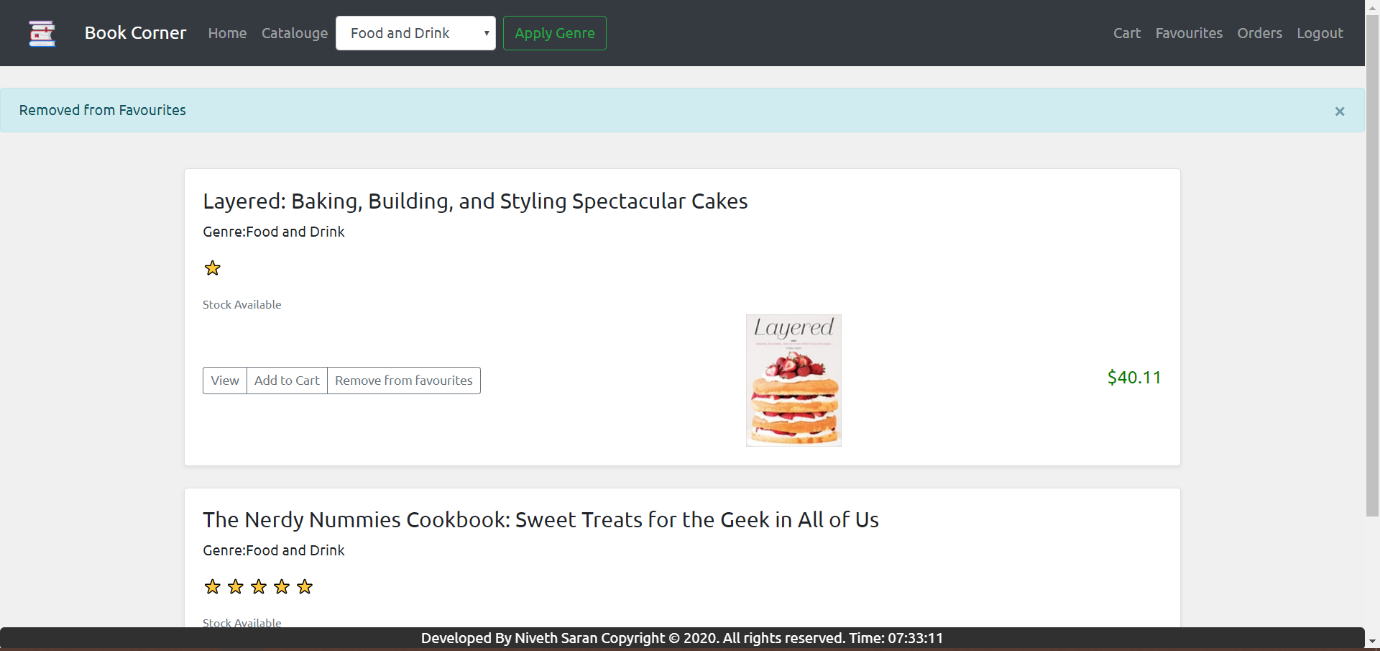
**Cart:**

****

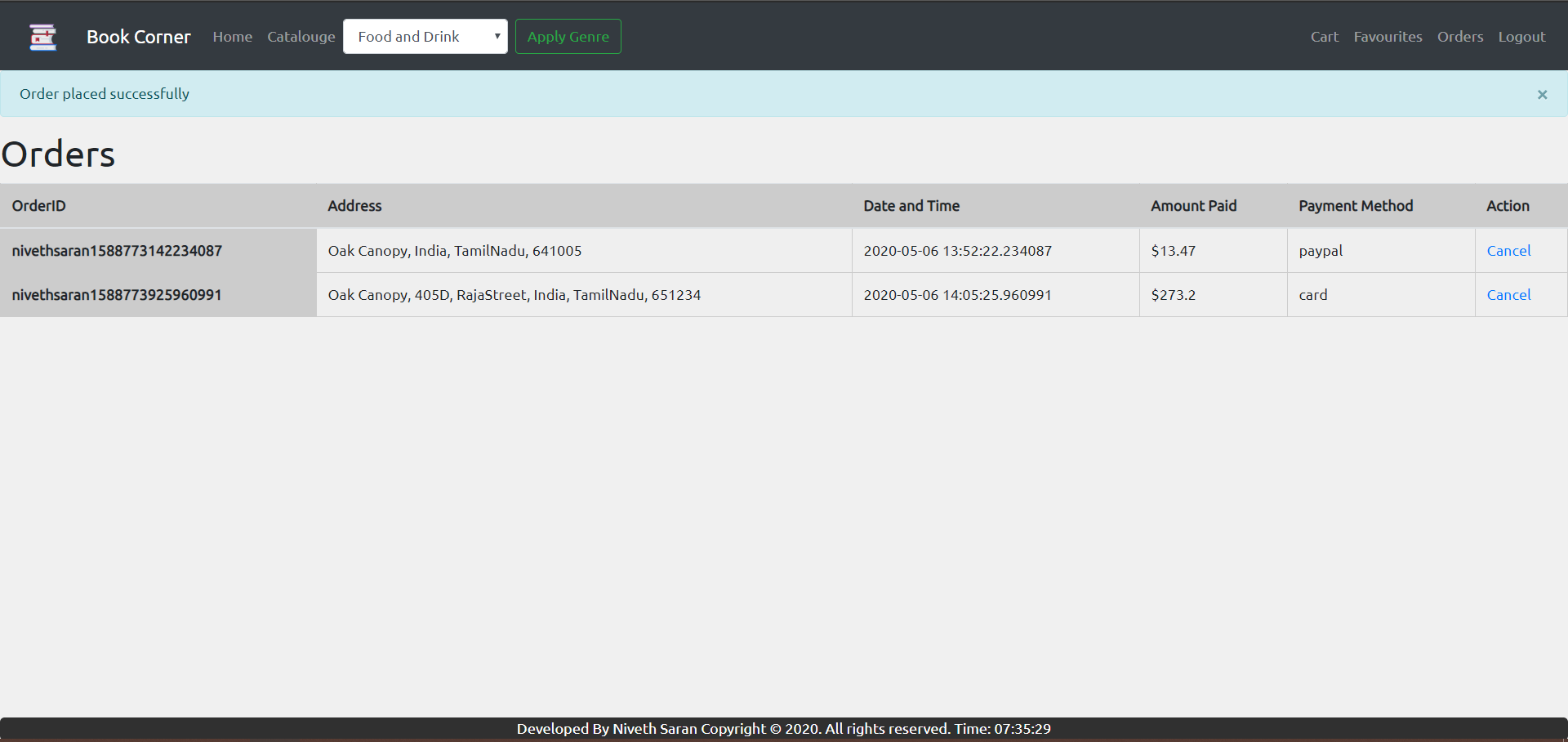
**Checkout Page:**

****

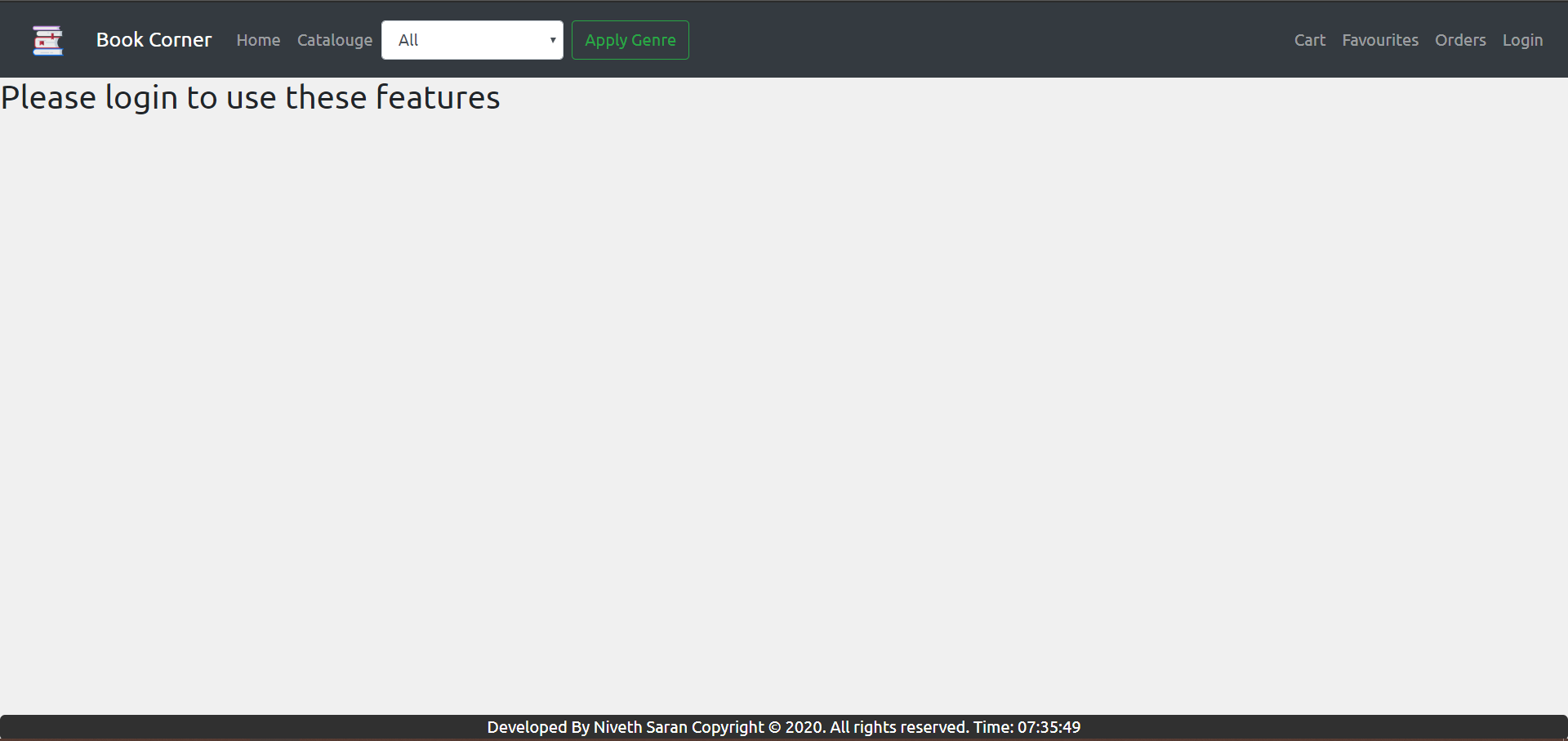
**Favourites:**

****

**Orders Page:**

****

**Logged Out Message:**

****

**Code:** (Available at <https://github.com/nivethsaran/BookStore> )

~The above GitHub repo is private~

Codebase(Zip): [https://tinyurl.com/BookCornerCodeBase](https://tinyurl.com/bookcornercodebase)

app.py

**import** re

**from** urllib3.exceptions **import** InsecureRequestWarning

**import** warnings

**from** datetime **import** datetime, timedelta, timezone, date

**import** datetime

**import** time

**import** urllib3

**import** requests

**from** bs4 **import** BeautifulSoup

**import** random

**import** os

**from** datautil **import** **\***

**from** flask **import** **\***

app **=** Flask(\_\_name\_\_)

**import** sqlite3,json

warnings.simplefilter('ignore', InsecureRequestWarning)

ROOT\_FOLDER**=** os.path.dirname(os.path.abspath(\_\_file\_\_))

today **=** date.today()

app.config['SECRET\_KEY'] **=** 'BookBasket'

@app.route('/')

**def** startup():

   message**=**''

   session['genre'] **=** 'All'

**if** 'username' **in** session:

      message**=**'Working'

**return** render\_template('startup.html',message**=**message)

#Login Route

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

**def** login():

      error**=**''

**if** request.method **==** 'POST':

**try**:

            username **=** request.form['username']

            password **=** request.form['password']

**if** len(username) **==** 0 **or** len(password) **==** 0:

               flash('Please fill both fields to login')

**else**:

               conn **=** **None**

**try**:

                  bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

                  conn **=** sqlite3.connect(bookstore)

                  dbURL **=** "SELECT username,password,fname,lname,email,mobile FROM user where username=?"

                  cursor **=** conn.cursor()

                  cursor.execute(dbURL,(username,))

                  rows**=**cursor.fetchall()

                  clen**=**int(len(rows))

**if** clen**==**0:

                     flash('Account does not exist')

**else**:

**for** row **in** rows:

**if** row[0]**==**username **and** row[1]**==**password:

                           fullname**=**row[2]**+**' '**+**row[3]

                           session['fullname']**=**fullname

                           session['username']**=**username

                           session['mobile']**=**row[5]

                           session['email']**=**row[4]

                           session['genre']**=**'All'

**return** (redirect(url\_for('home')))

**else**:

                           flash('Wrong passwords')

**except** Exception **as** e:

                    print(e)

                    flash('Some unexpected error occured, so please try again')

**finally**:

**if** conn:

                     conn.close()

**except** Exception **as** e:

             print(e)

             flash('Some unexpected error occured, so please try again')

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

#Function to check is mobile number is valid

**def** isValidNumber(number):

    Pattern **=** re.compile("(0/91)?[5-9][0-9]{9}")

**return** Pattern.match(number)

**def** isValidZip(number):

    Pattern **=** re.compile("[0-9]{6}")

**return** Pattern.match(number)

#Function to check if password is valid

**def** validPassword(password):

   flag **=** 0

**while** **True**:

**if** (len(password) **<** 8):

           flag **=** **-**1

**break**

**elif** **not** re.search("[a-z]", password):

           flag **=** **-**1

**break**

**elif** **not** re.search("[A-Z]", password):

           flag **=** **-**1

**break**

**elif** **not** re.search("[0-9]", password):

           flag **=** **-**1

**break**

**elif** **not** re.search("[\_@$#]", password):

           flag **=** **-**1

**break**

**elif** re.search("\s", password):

           flag **=** **-**1

**break**

**else**:

           flag **=** 0

**return** 'valid'

**if** flag **==** **-**1:

**return** 'invalid'

#Signup Route

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

**def** signup():

**if** request.method**==**'POST':

      fname**=**request.form['fname']

      lname**=**request.form['lname']

      username**=**request.form['username']

      password**=**request.form['password']

      email**=**request.form['email']

      mobile**=**request.form['mobile']

      regex **=** '^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$'

**if** fname **==** '' **or** lname **==** '' **or** username **==** '' **or** password **==** '' **or** email **==** '' **or** mobile **==** '':

         flash('All fields are mandatory')

**elif** **not** re.search(regex,email):

            flash('Enter Valid email')

**elif** validPassword(password)**==**'invalid':

         flash('Weak Password A strong paassword should contain min. 8 characters, a special char, a digit, a uppercase and a lowercase letter')

**elif** **not** isValidNumber(mobile):

         flash('Invalid Mobile Number')

**else**:

         conn **=** **None**

**try**:

            bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

            conn **=** sqlite3.connect(bookstore)

            dbURL **=** "INSERT INTO user values(?,?,?,?,?,?)"

            cursor **=** conn.cursor()

            cursor.execute(dbURL,(fname,lname,username,password,email,mobile))

            conn.commit()

            resp**=**make\_response(redirect(url\_for('login')))

**return** resp

**except** Exception **as** e:

            print(e)

            flash('Duplicate username')

**finally**:

**if** conn:

               conn.close()

**return** render\_template('signup.html',genrelist**=**genrelist)

#Home Route

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

**def** home():

   listbm **=** []

   recommendedGenre**=**request.cookies.get('genreCookie')

**if** 'username' **in** session:

      conn **=** **None**

**try**:

         bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

         conn **=** sqlite3.connect(bookstore)

**if**(recommendedGenre **is** **None**):

            dbURL **=** "SELECT \* FROM books where genre='Childrens' "

            cursor **=** conn.cursor()

            cursor.execute(dbURL)

**else**:

            dbURL **=** "SELECT \* FROM books where genre=?"

            cursor **=** conn.cursor()

            cursor.execute(dbURL,(recommendedGenre,))

**for** row **in** cursor:

            listbm.append(row)

**if**(len(listbm)**<**3):

            listbm.append(listbm[0])

            listbm.append(listbm[0])

         random.shuffle(listbm)

**except** Exception **as** e:

         print(e)

         print('hello')

**return** render\_template('home.html', listbm**=**listbm, genrelist**=**genrelist)

**else**:

**return** render\_template('home.html', genrelist**=**genrelist)

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

**def** cart():

   cartlist **=** []

   price**=**0;

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

   dbURL **=** "SELECT B.name,B.id,B.rating,B.price,B.image,B.details,B.genre,C.quantity from books as B,cart as C where C.id in (Select id from cart where username=?) and B.id==C.id"

   cursor **=** conn.cursor()

**if** 'username' **in** session:

      cursor.execute(dbURL, (session['username'],))

**for** row **in** cursor:

      cartlist.append(row)

      price**+=**row[7]**\***row[3]

   price**=**round(price,2)

**return** render\_template('cart.html', cartlist**=**cartlist, genrelist**=**genrelist,price**=**price)

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

**def** catalouge():

   booklist **=** []

   data**=**{'currently':**None**}

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**if** request.method**==**"POST":

      genre**=**request.form['genre']

      session['genre']**=**genre

**if**(genre**==**'All'):

         dbURL **=** "SELECT \* from books"

         cursor **=** conn.cursor()

         cursor.execute(dbURL)

**else**:

         dbURL **=** "SELECT \* from books where genre=?"

         cursor **=** conn.cursor()

         cursor.execute(dbURL,(genre,))

**for** row **in** cursor:

         booklist.append(row)

      resp **=** make\_response(render\_template('catalouge.html', booklist**=**booklist, genrelist**=**genrelist))

      resp.set\_cookie('genreCookie',genre)

**return** resp

**elif** request.method**==**"GET":

**if**('genre' **in** session):

**if**(session['genre'] **==** 'All'):

            dbURL **=** "SELECT \* from books"

            cursor **=** conn.cursor()

            cursor.execute(dbURL)

**else**:

            dbURL **=** "SELECT \* from books where genre=?"

            cursor **=** conn.cursor()

            cursor.execute(dbURL, (session['genre'],))

**else**:

         dbURL **=** "SELECT \* from books"

         cursor **=** conn.cursor()

         cursor.execute(dbURL)

**for** row **in** cursor:

         booklist.append(row)

**return** render\_template('catalouge.html', booklist**=**booklist, genrelist**=**genrelist)

**return** render\_template('catalouge.html', genrelist**=**genrelist)

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

**def** favourites():

   favlist **=** []

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

   dbURL **=** "SELECT \* from books where id in (Select id from fav where username=?)"

   cursor **=** conn.cursor()

**if** 'username' **in** session:

      cursor.execute(dbURL,(session['username'],))

**for** row **in** cursor:

      print(row)

      favlist.append(row)

**return** render\_template('favourites.html', favlist**=**favlist, genrelist**=**genrelist)

@app.route('/book/<int:bookno>', methods**=**["GET", "POST"])

**def** getbook(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

   dbURL **=** "select \* from books where id=?"

   cursor **=** conn.cursor()

   cursor.execute(dbURL,(int(bookno),))

   bookdata**=**[]

**for** row **in** cursor:

      bookdata**=**row

**break**

   desc**=**getProductDescription(bookdata[5])[:**-**7]

**return** render\_template('book.html', bookdata**=**bookdata,desc**=**desc ,genrelist**=**genrelist)

**def** getProductDescription(link):

   page **=** requests.get(link)

   soup **=** BeautifulSoup(page.content, 'html.parser')

   article **=** soup.find('div', id**=**'content\_inner').find('article', class\_**=**"product\_page")

   paras**=**article.findAll('p')

   desc**=**(paras[3].string)

**return** desc

**def** getBookDetails(booknum):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

   dbURL **=** "select \* from books where id=?"

   cursor **=** conn.cursor()

   cursor.execute(dbURL, (int(booknum),))

   bookdata **=** []

**for** row **in** cursor:

      bookdata **=** row

**return** row

@app.route('/addtocart/<int:bookno>', methods**=**["GET", "POST"])

**def** addtocart(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "insert into cart values(?,?,1)"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (session['username'], int(bookno),))

      conn.commit()

**except** Exception **as** e:

      print(e)

      flash('Already in cart')

**return** redirect(url\_for('catalouge'))

   flash("Added to Cart")

**return** redirect(url\_for('catalouge'))

@app.route('/addtocartfromfav/<int:bookno>', methods**=**["GET", "POST"])

**def** addtocartfromfav(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "insert into cart values(?,?,1)"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (session['username'], int(bookno),))

      conn.commit()

**except** Exception **as** e:

      print(e)

      flash('Already in cart')

**return** redirect(url\_for('favourites'))

   flash("Added to Cart")

**return** redirect(url\_for('favourites'))

@app.route('/addtocartfromview/<int:bookno>', methods**=**["GET", "POST"])

**def** addtocartfromview(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "insert into cart values(?,?,1)"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (session['username'], int(bookno),))

      conn.commit()

**except** Exception **as** e:

      print(e)

      flash('Already in cart')

**return** redirect(url\_for('getbook', bookno**=**bookno))

   flash("Added to Cart")

**return** redirect(url\_for('getbook', bookno**=**bookno))

@app.route('/updatecart/<int:bookno>', methods**=**["POST"])

**def** updateCart(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**if**(request.method**==**"POST"):

      quantity**=**request.form['quantity']

**if**(int(quantity)**<**1):

         flash('Quantity must be atleast 1')

**return** redirect(url\_for('cart'))

**try**:

         dbURL **=** "update cart set quantity=? where username=? and id=?"

         cursor **=** conn.cursor()

         cursor.execute(dbURL, (quantity,session['username'], int(bookno),))

         conn.commit()

**except** Exception **as** e:

         print(e)

         flash('Some error occured, please try again')

**return** redirect(url\_for('cart'))

   flash("Quantity Updated")

**return** redirect(url\_for('cart'))

@app.route('/removefromcart/<int:bookno>', methods**=**["GET", "POST"])

**def** removefromcart(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "delete from cart where username=? and id=?"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (session['username'], int(bookno),))

      conn.commit()

**except** Exception **as** e:

      flash('Some error occured, please try again')

**return** redirect(url\_for('cart'))

   flash("Removed from Cart")

**return** redirect(url\_for('cart'))

@app.route('/addtofav/<int:bookno>', methods**=**["GET", "POST"])

**def** addtofav(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "insert into fav values(?,?)"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (session['username'],int(bookno),))

      conn.commit()

**except** Exception **as** e:

      flash('Already in favourites')

**return** redirect(url\_for('catalouge'))

   flash("Added to Favourites")

**return** redirect(url\_for('catalouge'))

@app.route('/addtofavfromview/<int:bookno>', methods**=**["GET", "POST"])

**def** addtofavfromview(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "insert into fav values(?,?)"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (session['username'], int(bookno),))

      conn.commit()

**except** Exception **as** e:

      print(e)

      flash('Already in favourites')

**return** redirect(url\_for('getbook',bookno**=**bookno))

   flash("Added to Favourites")

**return** redirect(url\_for('getbook', bookno**=**bookno))

@app.route('/removefromfav/<int:bookno>', methods**=**["GET", "POST"])

**def** removefromfav(bookno):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "delete from fav where username=? and id=?"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (session['username'], int(bookno),))

      conn.commit()

**except** Exception **as** e:

      flash('Error')

**return** redirect(url\_for('favourites'))

   flash("Removed from Favourites")

**return** redirect(url\_for('favourites'))

@app.route('/orders')

**def** orders():

   orderlist **=** []

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

   dbURL **=** "SELECT \* from orders where username=?"

   cursor **=** conn.cursor()

**if** 'username' **in** session:

      cursor.execute(dbURL, (session['username'],))

**for** row **in** cursor:

      address**=**row[2]**+**', '**+**row[3]**+**', '**+**row[4]**+**', '**+**str(row[5])

      dat**=**row[8]

      orderid**=**row[1]

      amountpaid**=**row[7]

      paymentMethod**=**row[6]

      orderlist.append((orderid,address,dat,amountpaid,paymentMethod))

**return** render\_template('orders.html', genrelist**=**genrelist, orderlist**=**orderlist)

@app.route('/cancel/<string:orderid>')

**def** cancel(orderid):

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

**try**:

      dbURL **=** "delete from orders where orderid=?"

      cursor **=** conn.cursor()

      cursor.execute(dbURL, (orderid,))

      conn.commit()

**except** Exception **as** e:

      flash("Error")

   flash("Order Cancelled")

**return** redirect(url\_for('orders'))

@app.route('/checkout')

**def** checkout():

   price **=** 0

   quantity**=**0

   bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

   conn **=** sqlite3.connect(bookstore)

   dbURL **=** "SELECT B.name,B.id,B.rating,B.price,B.image,B.details,B.genre,C.quantity from books as B,cart as C where C.id in (Select id from cart where username=?) and B.id==C.id"

   cursor **=** conn.cursor()

**if** 'username' **in** session:

      cursor.execute(dbURL, (session['username'],))

**for** row **in** cursor:

      quantity**+=**row[7]

      price **+=** row[7]**\***row[3]

   price**=**round(price,2)

**if** price**==**0 **or** quantity**==**0:

      flash("Cant checkout with ZERO items in cart")

**return** redirect(url\_for('cart'))

**return** render\_template('checkout.html', quantity**=**quantity, genrelist**=**genrelist, price**=**price)

@app.route('/processCheckout', methods**=**['POST'])

**def** processCheckout():

   address **=** request.form['address']

   total **=** request.form['total']

   paymentMethod**=** request.form['paymentMethod']

   country **=** request.form['country']

   state **=** request.form['state']

   pincode **=** request.form['zip']

   quantity **=** request.form['quantity']

**if**(pincode**==**'' **or** country**==**'' **or** state**==**'' **or** total**==**0 **or** address**==**''):

      flash("All fields are mandatory")

**return** redirect(url\_for('checkout',total**=**total,quantity**=**quantity))

**elif** **not** isValidZip(pincode):

      flash("Invalid Zip")

**return** redirect(url\_for('checkout', total**=**total, quantity**=**quantity))

**else**:

      username**=**session['username']

      datop **=** datetime.datetime.now()

      orderid**=** session['username']**+**str(int(datetime.datetime.timestamp(datop)**\***(10**\*\***6)))

      print(orderid)

      bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

      conn **=** sqlite3.connect(bookstore)

**try**:

         dbURL **=** "insert into orders values(?,?,?,?,?,?,?,?,?)"

         cursor **=** conn.cursor()

         cursor.execute(dbURL, (username,orderid,address,country,state,pincode,paymentMethod,total,datop))

         conn.commit()

**except** Exception **as** e:

         print(e)

         flash("Some error occured while placing order. Check your connection or try again")

**return** redirect(url\_for('checkout',quantity**=**quantity,total**=**total))

      deleteCart(username)

      flash("Order placed successfully")

**return** redirect(url\_for('orders'))

**def** deleteCart(username):

**try**:

      bookstore **=** os.path.join(ROOT\_FOLDER, 'bookstore.db')

      conn **=** sqlite3.connect(bookstore)

      dbURL2 **=** "delete from cart where username=?"

      cursor2 **=** conn.cursor()

      cursor2.execute(dbURL2, (username,))

      conn.commit()

**except** Exception **as** e:

      print(e)

@app.route('/logout')

**def** logout():

   session.pop('username',**None**)

   session.pop('fullname',**None**)

   session.pop('mobile',**None**)

   session.pop('email',**None**)

   session.pop('genre',**None**)

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

**if** \_\_name\_\_**==**'\_\_main\_\_':

    app.run(debug**=True**)

**Documentation:**

Open cmd,

Then execute

git clone <https://github.com/nivethsaran/BookStore.git>

Then execute.\venv\Scripts\activateto activate virtual environment.

Then execute

pip install -r requirements.txt

Then executeflask runto deploy on local server.

Open[**http://localhost:5000**](http://localhost:5000)in browser to view the flask app.

**About Project:**

* Book Corner is an online book store which consists of a variety of genres of books.
* User can login and logout of the platform
* User can view books and filter books with respect to their genres.
* Users can add and remove books to and from a favourites book list respectively
* Users can add and remove books to the cart and check the total in the cart page.
* Users can update quantity in cart and also checkout from the cart page.
* Users can view the description and rating of a book.
* Users can place order and also cancel placed orders from the Orders page.
* The homepage displays three books which are recommended to the user according to his preferences using cookies.
* Login and Signup uses sessions to manage an users session with the app.

**Database:**



About Database:

* books table is used to store details about the books
* user table is used to save details about the user and used during login and signup
* cart and fav table are used to store favourites and cart items for every user.
* Orders table is used to store placed orders and also updated when a order needs to be cancelled.

Normalization:

* 2NF was applied to the table to avoid data redundancies.
* 3NF wasn’t required due to the absence of transitive dependencies.

**Technical Details:**

Tech Stack:

* Flask
* SQLLITE DB
* Bootstrap CSS
* Beautiful Soup (Scraping)

Python Libraries Used:

* re
* datetime
* urllib3
* time
* warnings
* sqlite3
* bs4
* random
* requests
* flask libraries like redirect, url\_for, session etc.

**How the tech stack was used to build the app?**

* Sessions are used throughout the app. Once the user logs in it is stored in sessions and is popped out when user logs out.
* Sign-Up and Log-In page contains validation which uses the “re” library to match regex expressions.
* Used cookies to save the last searched genre of books and display results based on cookies in the home page.
* Made use of Variable Rules of Flask routing to Add books to cart and favourites from both the catalogue page and the View book page.
* The view book page views the synopsis of a book. This description is scraped off from [toscrape.com](http://books.toscrape.com/) using the Beautiful Soup library
* The connectivity to the Sqlite database was made using the sqlite3 python library. This library is used several times in this project.
* flash() method was used to display popup messages on screen about login errors, network error or other details.
* CSS is used to make the web app look visually better and to enhance the user experience.
* JavaScript was used to handle button clicks and to display a live clock in the footer of the webpage.