Library Management System

Specific Prerequisites and setup :

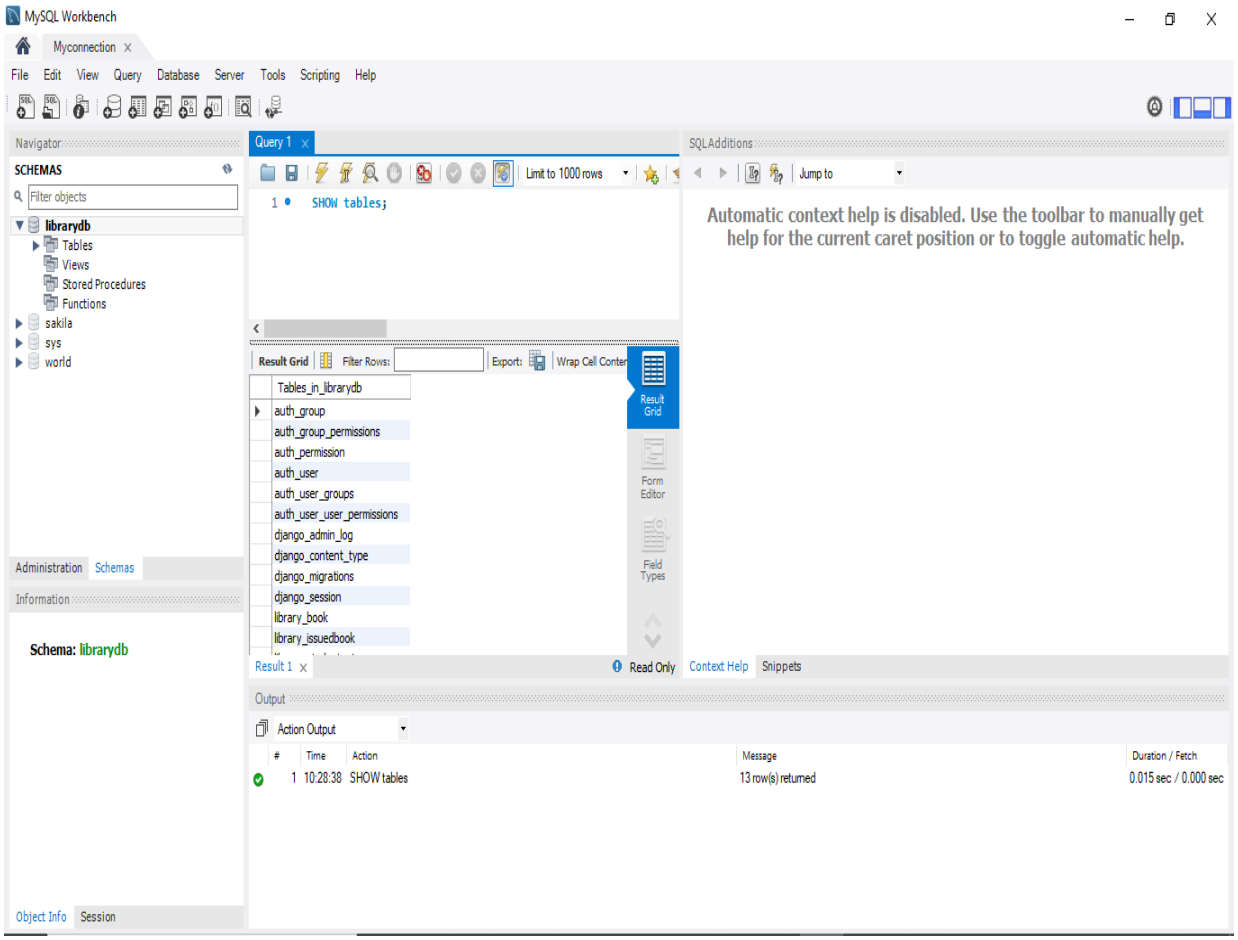
- Python 3.0 or greater
- Django 4.0
- VS Code
- MySQL 8.0
- MySQL server 8.0
- MySQL Workbench 8.0 CE
- Get the code:
- Type in terminal: pip install mysqlclient
- Type in terminal: py manage.py makemigrations
- Type in terminal: py manage.py migrate
- Start the application : in terminal run 'py manage.py runserver'
- Run : 127.0.0.1:8000 in browser
- Some style sheet from online source bootstrap

API Functions:

The following section lead you through the primary functions of the sample application. The following shows you how to program with the various features and services connecting with MySQL database, home page consist of Admin and Student, Next functions of Admin page and Student page.

Connect with Database:

Django default provides sqlite but we need here MySQL database therefore some changes need in our main projests 'settings.py' before do that we must make a new database for our project I made a database named as 'librarydb' and make root user and set password given below image



Now change the database in settings.py as follows

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'librarydb',
        'USER': 'root',
        'PASSWORD': 'badsa1',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

Home page:

Our home page consists of two main functions Admin, Student and another About us. The function defines our home page is inside of 'library/views.py' given below

```
def home_view(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('afterlogin')
    return render(request,'library/index.html')
```

Admin page:

From home page after clicking the Admin option you will get two function Login and Signup

```
def adminclick_view(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('afterlogin')
    return render(request,'library/adminclick.html')
```

Here if you choose the Signup option the below function work to take the new Admin in the database

```
def adminsignup_view(request):
    form=forms.AdminSigupForm()
    if request.method=='POST':
        form=forms.AdminSigupForm(request.POST)
        if form.is_valid():
            user=form.save()
            user.set_password(user.password)
            user.save()
```

```
            my_admin_group =
Group.objects.get_or_create(name='ADMIN')
            my_admin_group[0].user_set.add(user)

            return HttpResponseRedirect('adminlogin')
    return
render(request,'library/adminsignup.html',{'form':form})
```

If you choose the Login function then the below code works

```
def is_admin(user):
    return user.groups.filter(name='ADMIN').exists()

def afterlogin_view(request):
    if is_admin(request.user):
        return
render(request,'library/adminafterlogin.html')
```

<u>Inside the Admin page Admin can these works:</u>

-Create Admin account and Login.

- Can Add, View, Book

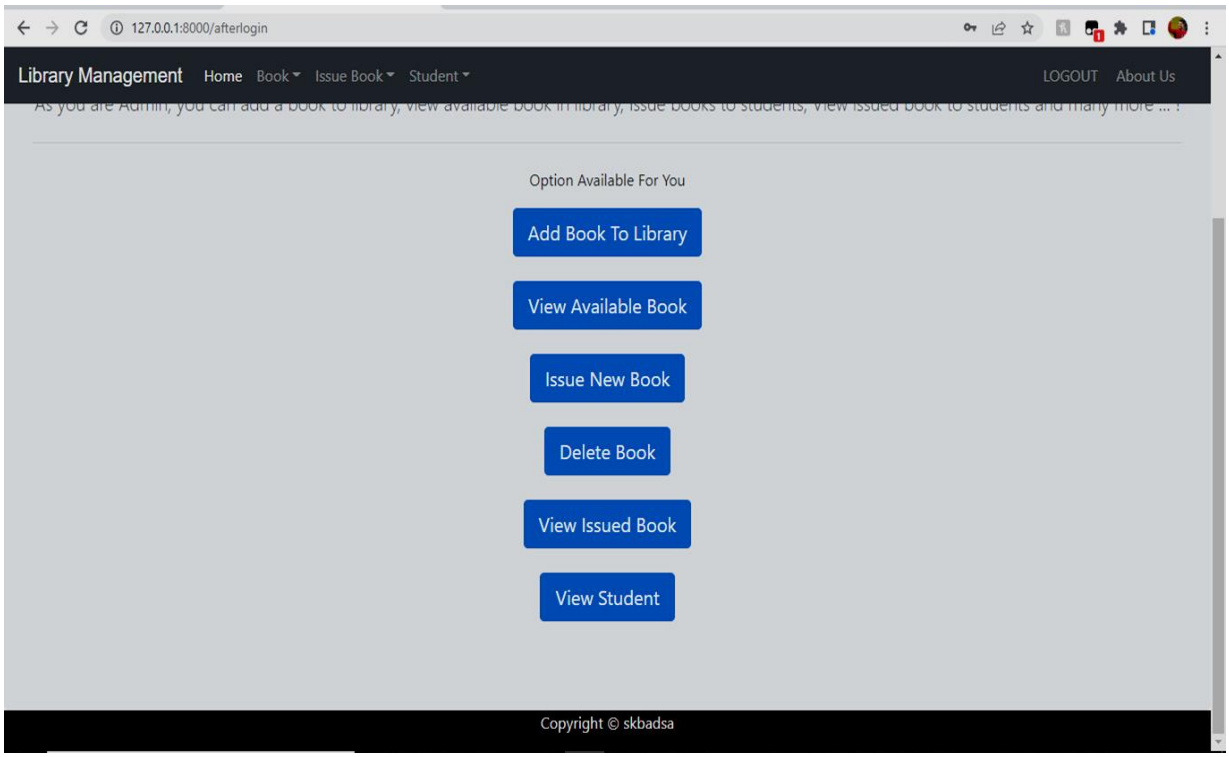
- Can Issue Book (added by Admin) to registered student.

- Can view Issued book with issued date and expiry date.

- Can delete a book.

- Can view Fine .

- Can View Students that are registered into system.



The below code here given for Admin page work functions

```
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def addbook_view(request):
    #now it is empty book form for sending to html
    form=forms.BookForm()
    if request.method=='POST':
        #now this form have data from html
        form=forms.BookForm(request.POST)
        if form.is_valid():
            user=form.save()
            return render(request,'library/bookadded.html')
    return render(request,'library/addbook.html',{'form':form})
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def viewbook_view(request):
    books=models.Book.objects.all()
    return render(request,'library/viewbook.html',{'books':books})
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def issuebook_view(request):
```

```
    form=forms.IssuedBookForm()
    if request.method=='POST':
        #now this form have data from html
        form=forms.IssuedBookForm(request.POST)
        if form.is_valid():
            obj=models.IssuedBook()
            obj.enrollment=request.POST.get('enrollment2')
            obj.isbn=request.POST.get('isbn2')
            obj.save()
            return render(request,'library/bookissued.html')
    return render(request,'library/issuebook.html',{'form':form})


@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def viewissuedbook_view(request):
    issuedbooks=models.IssuedBook.objects.all()
    li=[]
    for ib in issuedbooks:
        issdate=str(ib.issuedate.day)+'-'+str(ib.issuedate.month)+'-'+str(ib.issuedate.year)
        expdate=str(ib.expirydate.day)+'-'+str(ib.expirydate.month)+'-'+str(ib.expirydate.year)
        #fine calculation
        days=(date.today()-ib.issuedate)
        print(date.today())
        d=days.days
        fine=0
        if d>15:
            day=d-15
            fine=day*10
        books=list(models.Book.objects.filter(isbn=ib.isbn))
        students=list(models.StudentExtra.objects.filter(enrollment=ib.enrollment))
        i=0
        for l in books:
        t=(students[i].get_name,students[i].enrollment,books[i].name,books[i].author,issdate,expdate,fine)
            i=i+1
            li.append(t)
    return render(request,'library/viewissuedbook.html',{'li':li})
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def viewstudent_view(request):
    students=models.StudentExtra.objects.all()
    return render(request,'library/viewstudent.html',{'students':students})
```

Student Page:

- Create account and Login.

- Can view their issued book only with expiry date.