I can add, update, and remove Books from the system

from django.db import models

from django.contrib.auth.models import User

from datetime import datetime,timedelta

class Book(models.Model):

    name = models.CharField(max\_length=200)

    author = models.CharField(max\_length=200)

    isbn = models.PositiveIntegerField()

    category = models.CharField(max\_length=50)

    def \_\_str\_\_(self):

        return str(self.name) + " ["+str(self.isbn)+']'

class Student(models.Model):

    user = models.OneToOneField(User, on\_delete=models.CASCADE)

    classroom = models.CharField(max\_length=10)

    branch = models.CharField(max\_length=10)

    roll\_no = models.CharField(max\_length=3, blank=True)

    phone = models.CharField(max\_length=10, blank=True)

    image = models.ImageField(upload\_to="", blank=True)

    def \_\_str\_\_(self):

        return str(self.user) + " ["+str(self.branch)+']' + " ["+str(self.classroom)+']' + " ["+str(self.roll\_no)+']'

def expiry():

    return datetime.today() + timedelta(days=14)

class IssuedBook(models.Model):

    student\_id = models.CharField(max\_length=100, blank=True)

    isbn = models.CharField(max\_length=13)

    issued\_date = models.DateField(auto\_now=True)

    expiry\_date = models.DateField(default=expiry)

from django.urls import path

from . import views

urlpatterns = [

    path("", views.index, name="index"),

    path("add\_book/", views.add\_book, name="add\_book"),

    path("view\_books/", views.view\_books, name="view\_books"),

    path("view\_students/", views.view\_students, name="view\_students"),

    path("issue\_book/", views.issue\_book, name="issue\_book"),

    path("view\_issued\_book/", views.view\_issued\_book, name="view\_issued\_book"),

    path("student\_issued\_books/", views.student\_issued\_books, name="student\_issued\_books"),

    path("profile/", views.profile, name="profile"),

    path("edit\_profile/", views.edit\_profile, name="edit\_profile"),

    path("student\_registration/", views.student\_registration, name="student\_registration"),

    path("change\_password/", views.change\_password, name="change\_password"),

    path("student\_login/", views.student\_login, name="student\_login"),

    path("admin\_login/", views.admin\_login, name="admin\_login"),

    path("logout/", views.Logout, name="logout"),

    path("delete\_book/<int:myid>/", views.delete\_book, name="delete\_book"),

    path("delete\_student/<int:myid>/", views.delete\_student, name="delete\_student"),

]

<div class="p-4 bg-light">

    <div class="container-fluid py-5">

      <h1 class="display-5 fw-bold">Custom jumbotron</h1>

      <p class="col-md-8 fs-4">Using a series of utilities, you can create this jumbotron, just like the one in previous versions of Bootstrap. Check out the examples below for how you can remix and restyle it to your liking.</p>

      <button class="btn btn-primary btn-lg" type="button">Example button</button>

    </div>

    <div class="container">

  <div class="row">

    <div class="col-lg-6">

      <img src="{% static 'Admin-icon.png' %}" width="50%" height="50%" alt=""><br><br>

        <a class="btn btn-outline-primary btn-lg" style="text-align:center" href="/admin\_login/">Admin</a>

    </div>

    <div class="col-lg-6">

      <img src="{% static 'student.png' %}" width="50%" height="50%" alt=""><br><br>

      <a class="btn btn-outline-primary btn-lg" href="/student\_login/">Student</a>

    </div>

  </div>

  </div>

</div>

I can add, update, view, and remove Member from the system

<div class="container">

    <form method="POST"> {% csrf\_token %}

    <div class="row mt-4">

        <div class="form-group col-md-12">

            <label><i style="font-weight: bold;">Book Name</i></label>

            <input type="text" class="form-control mt-2" name="name" placeholder="Enter name of the Book" required>

        </div>

    </div>

    <div class="row mt-4">

        <div class="form-group col-md-12">

            <label><i style="font-weight: bold;">Author Name</i></label>

            <input type="text" class="form-control mt-2" name="author" placeholder="Enter name of the Author" required>

        </div>

    </div>

    <div class="row mt-4">

        <div class="form-group col-md-12">

            <label><i style="font-weight: bold;">ISBN Number</i></label>

            <input type="number" class="form-control mt-2" name="isbn" placeholder="Enter ISBN number of the book" required>

        </div>

    </div>

    <div class="row mt-4">

        <div class="form-group col-md-12">

            <label><i style="font-weight: bold;">Category</i></label>

            <input type="text" class="form-control mt-2" name="category" placeholder="Enter Category of the book" required>

        </div>

    </div>

    <button type="submit" class="btn btn-outline-primary mt-4">Add Book</button>

</div>

</form>

def add\_book(request):

    if request.method == "POST":

        name = request.POST['name']

        author = request.POST['author']

        isbn = request.POST['isbn']

        category = request.POST['category']

        books = Book.objects.create(name=name, author=author, isbn=isbn, category=category)

        books.save()

        alert = True

        return render(request, "add\_book.html", {'alert':alert})

    return render(request, "add\_book.html")

I can view, borrow, and return available Books

<div class="container mt-4">

    <h1 class="text-center"><u>All Books List</u></h1>

    <table class="table table-hover" id="example">

        <thead>

            <tr>

                <th>Sr.No</th>

                <th>Book Name</th>

                <th>Author</th>

                <th>ISBN Number</th>

                <th>Category</th>

                <th>Delete</th>

            </tr>

        </thead>

        <tbody>

            {% for book in books %}

            <tr>

                <td>{{forloop.counter}}.</td>

                <td>{{book.name}}</td>

                <td>{{book.author}}</td>

                <td>{{book.isbn}}</td>

                <td>{{book.category}}</td>

                <td><a href="/delete\_book/{{book.id}}/" class="btn btn-danger" onclick="return confirm('Are you sure you want to delete this book?')">Delete</a></td>

            </tr>

            {% endfor %}

        </tbody>

    </table>

    </div>

Once a book is borrowed, its status will change to BORROWED

def borrow\_book(request, pk):  
 book\_instance = get\_object\_or\_404(BookInstance, pk=pk)  
 if request.method == 'POST':  
 if request.user.is\_authenticated:  
 book\_instance.borrower = request.user  
 book\_instance.due\_back = datetime.date.today() + datetime.timedelta(weeks=3)  
 book\_instance.status = STATUS\_ON\_LOAN  
 book\_instance.save()  
 return HttpResponseRedirect(reverse('dashboard\_customer'))  
  
 context = {  
 'book\_instance': book\_instance,  
 }  
  
return render(request, 'catalog/book\_detail.html', context)

Once a book is returned, its status will change to AVAILABLE

def create\_themen(request):  
 new\_topic = Topic(topic=request.POST['topic'])  
 new\_topic.save()  
 return render(request, 'topic.html', {'topic': topic.objects.all()})  
  
  
def create\_learning\_objective(request):  
 new\_learning\_objective = LearningObjective(learning\_objective=request.POST['learning\_objective'])  
 new\_learning\_objective.save()  
 new\_learning\_objective\_topic = Topic.objects.get(topic=request.POST['topic'])  
 new\_learning\_objective\_topic.new\_learning\_objective\_topic.add(new\_learning\_objective)  
 return render(request, 'learning\_objective.html', {  
 'topic': Topic.objects.all(),  
 'todo': TodoList.objects.all(),  
 'learning\_objective': LearningObjective.objects.all()  
 })

I can delete my own account

# delete my own account  
from django.contrib.auth import logout as auth\_logout, get\_user\_model  
from django.contrib.auth.decorators import login\_required  
from django.views.decorators.http import require\_http\_methods  
  
@login\_required  
@require\_http\_method(['POST'])  
def remove\_account(request):  
 user\_pk = request.user.pk  
 auth\_logout(request)  
 User = get\_user\_model()  
 User.objects.filter(pk=user\_pk).update(is\_active=False)