**Models.py**

from django.db import models

from django.urls import reverse

Branch=[

('IT','IT'),

('CSE','CSE'),

('MECH','MECH'),

('PROD','PROD'),

('CHEM','CHEM'),

('CIVIL','CIVIL'),

('TEXT','TEXT'),

('EXTC','EXTC'),

('ELECT','ELECT'),

('INSTRU','INSTRU')

]

Language=[

('English','English'),

('Marathi','Marathi'),

('Hindi','Hindi')

]

Available=[

('Yes','Yes'),

('No','No')

]

class Book(models.Model):

name = models.CharField(max\_length = 60)

author = models.CharField(max\_length = 40, default = 'Random')

isbn = models.CharField(max\_length = 12)

discription = models.TextField()

copies = models.IntegerField(default = 50)

available = models.CharField(max\_length = 3, choices = Available, default = 'Yes')

language = models.CharField(max\_length = 7, choices = Language, default = 'English')

image = models.ImageField()

def \_\_str\_\_(self):

return self.name

def get\_absolute\_url(self):

return reverse(book\_list)

class Student(models.Model):

student\_name = models.CharField(max\_length = 30)

book\_assign = models.ForeignKey(Book, on\_delete = models.CASCADE,blank=True, null=True)

reg\_no = models.CharField(max\_length = 15, default = '\*\*\*')

branch = models.CharField(max\_length = 6, choices = Branch, default = 'IT')

assign\_date = models.DateField('book assign date')

due\_date = models.DateField('Due on')

due\_fine = models.DecimalField(max\_digits = 5, decimal\_places = 2)

def \_\_str\_\_(self):

return self.student\_name

def total\_fine(self):

return self.due\_fine

class fine(models.Model):

student\_name = models.CharField(max\_length = 40, default = 'Robot')

reg\_no = models.CharField(max\_length = 15)

branch = models.CharField(max\_length = 6, choices = Branch, default = 'IT')

**views.py**

from django.contrib.auth.decorators import login\_required

from django.contrib.auth.mixins import LoginRequiredMixin

from django.views.generic import (TemplateView,ListView,DetailView,CreateView,UpdateView,DeleteView)

from .models import Book,Student

from django.contrib.auth.models import User

from django.urls import reverse\_lazy

from django.contrib.auth import login, authenticate, logout

from django.shortcuts import render, redirect,get\_object\_or\_404

from .forms import SignUpForm,FineForm

from django.urls import reverse

class about(TemplateView):

template\_name = 'book/about.html'

class Book\_list(ListView):

model = Book

class Student\_list(ListView):

model = Student

class Book\_create(LoginRequiredMixin, CreateView):

fields = ('name','author','isbn','discription','copies','available','language','image')

model = Book

success\_url = reverse\_lazy('book\_list')

class Student\_create(LoginRequiredMixin, CreateView):

fields = ('student\_name','book\_assign','reg\_no','branch','assign\_date','due\_date','due\_fine')

model = Student

success\_url = reverse\_lazy('student\_list')

class Book\_update(LoginRequiredMixin, UpdateView):

login\_url = '/login/'

redirect\_field\_name = 'book/book\_list.html'

fields = ('name','author','isbn','discription','copies','available','language','image')

model = Book

success\_url = reverse\_lazy('book\_list')

class Student\_update(LoginRequiredMixin, UpdateView):

login\_url = '/login/'

redirect\_field\_name = 'book/student\_list.html'

fields = ('student\_name','book\_assign','reg\_no','branch','assign\_date','due\_date','due\_fine')

model = Student

success\_url = reverse\_lazy('student\_list')

class Book\_delete(LoginRequiredMixin, DeleteView):

model = Book

success\_url = reverse\_lazy('book\_list')

class Student\_delete(LoginRequiredMixin, DeleteView):

model = Student

success\_url = reverse\_lazy('student\_list')

def signup(request):

if request.method == 'POST':

form = SignUpForm(request.POST)

if form.is\_valid():

form.save()

username = form.cleaned\_data.get('username')

raw\_password = form.cleaned\_data.get('password1')

user = authenticate(username=username, password=raw\_password)

login(request, user)

return redirect('book\_list')

else:

form = SignUpForm()

return render(request, 'registration/signup.html', {'form': form})

def Fine\_confirmation(request,pk):

form = FineForm()

if request.method == 'POST':

form = FineForm(request.POST)

if form.is\_valid():

student\_name = form.cleaned\_data['student\_name']

reg\_no = form.cleaned\_data['reg\_no']

branch = form.cleaned\_data['branch']

student = get\_object\_or\_404(Student, pk=pk)

student.reg\_no = reg\_no

student.branch = branch

student.save()

return redirect('return\_fine', pk = student.id)

return render(request, 'book/fine\_confirmation.html', {'form' : form})

def return\_fine(request, pk):

if request.user.is\_authenticated:

fine = get\_object\_or\_404(Student, pk = pk)

return render(request,'book/fine\_payment.html', {'fine':fine})

def logoutuser(request):

logout(request)

return redirect('login')

def show\_book(request, pk):

if request.user.is\_authenticated:

info = get\_object\_or\_404(Book, pk = pk)

return render(request,'book/show\_book.html', {'info':info})

**urls.py**

from django.urls import path,include

from . import views

from django.contrib.auth import views as auth\_views #new

urlpatterns = [

path('',views.about.as\_view(), name = 'about'),

path('book\_list/', views.Book\_list.as\_view(), name = 'book\_list'),

path('student\_list/', views.Student\_list.as\_view(), name = 'student\_list'),

path('book\_create/', views.Book\_create.as\_view(), name = 'Create-Book'),

path('student\_create/', views.Student\_create.as\_view(), name = 'Create-Student'),

path('book\_update/<int:pk>/', views.Book\_update.as\_view(), name = 'book\_update'),

path('student\_update/<int:pk>/', views.Student\_update.as\_view(), name = 'student\_update'),

path('book\_delete/<int:pk>/', views.Book\_delete.as\_view(), name = 'book\_delete'),

path('student\_delete/<int:pk>/', views.Student\_delete.as\_view(), name = 'student\_delete'),

path('fine\_confirmation/<int:pk>/',views.Fine\_confirmation, name = 'confirm'),

path('return\_fine/<int:pk>/',views.return\_fine, name = 'return\_fine'),

path('show\_book/<int:pk>/',views.show\_book, name = 'show\_book'),

path('logout/',views.logoutuser,name='logout'),

#new

path('reset\_password/',auth\_views.PasswordResetView.as\_view(template\_name = "registration/password\_reset.html"), name = 'reset\_password'),

path('reset\_password\_sent/',auth\_views.PasswordResetDoneView.as\_view(), name = 'password\_reset\_done'),

path('reset/<uidb64>/<token>',auth\_views.PasswordResetConfirmView.as\_view(), name = 'password\_reset\_confirm'),

path('reset\_password\_complete/',auth\_views.PasswordResetCompleteView.as\_view(), name = 'password\_reset\_complete'),

]

**Forms.py**

from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

from .models import fine,Book,Student

from django.forms import ModelForm

class SignUpForm(UserCreationForm):

first\_name = forms.CharField(max\_length=30, required=False, help\_text='Optional.')

last\_name = forms.CharField(max\_length=30, required=False, help\_text='Optional.')

email = forms.EmailField(max\_length=254, help\_text='Required. Inform a valid email address.')

class Meta:

model = User

fields = ('username', 'first\_name', 'last\_name', 'email', 'password')

class FineForm(ModelForm):

class Meta():

model = fine

fields = '\_\_all\_\_'

#class BookCreate(forms.ModelForm):

# class Meta:

# model = Book

# fields = '\_\_all\_\_'

#class StudentCreate(forms.ModelForm):

# class Meta:

# model = Student

# fields = '\_\_all\_\_'

**API PROGRAMS: -**

**Views.py**

from rest\_framework import generics

from book.models import Book,Student

from .serializers import BookSerializer,StudentSerializer

class BookAPIView(generics.ListAPIView):

queryset = Book.objects.all()

serializer\_class = BookSerializer

class StudentAPIView(generics.ListAPIView):

queryset = Student.objects.all()

serializer\_class = StudentSerializer

**urls.py**

from django.urls import path

from .views import BookAPIView,StudentAPIView

urlpatterns = [

path('book/', BookAPIView.as\_view()),

path('student/', StudentAPIView.as\_view()),

]

**Serializers.py**

from rest\_framework import serializers

from book.models import Book,Student

class BookSerializer(serializers.ModelSerializer):

class Meta:

model = Book

fields = ('id','name', 'author', 'isbn', 'discription', 'copies', 'available', 'language', 'image')

class StudentSerializer(serializers.ModelSerializer):

class Meta:

model = Student

fields = ('id','student\_name','book\_assign','reg\_no','branch','assign\_date','due\_date','due\_fine')

**SNAPSHOTS: -**



















