Library Management System

→ 1 Technology used :

- django.contrib messages for message pop up
- Django Framework
- javascript for global search in table
- Django REST Framework
- Crispy forms
- Bootstrap version 5.2 CDN for frontend
- Django Temaplate for frontend
- Vertual environment
- EMAIL_BACKEND = "django.core.mail.backends.console.EmailBackend"
- For mysql used Xampp php my admin mysql
- OS:Ubuntu
- corsheaders for connect react js to django server
- → Requirement for setup django with versions

```
asgiref == 3.5.2
```

certifi==2022.9.14

charset-normalizer==2.1.1

Django==4.1.1

django-cors-headers==3.13.0

django-crispy-forms==1.14.0

django-filter==22.1

djangorestframework==3.13.1

djangorestframework-simplejwt==5.2.0

idna==3.4

PyJWT==2.4.0

pytz = 2022.2.1

requests == 2.28.1

sqlparse==0.4.2

urllib3==1.26.12

- → Django REST Framework
 - 1. API Created
 - → bookCreate view
 - → Library books List view
 - → Books Retrieve view(Detail view)
 - → Book update view
 - → Book Delete view
 - → Used generic view for api
 - → url for api : http://127.0.0.1:8000/api

```
src > library_component > & serializers.py > % BookLibrarySerializer > % Meta

1 from datetime import datetime, date
                                                                  2
                rest_framework import serializers  library component.models import Library data
          class BookLibrarySerializer(serializers.ModelSerializer):
                    model = Library_data
fields = ('id','Book_id', "description", "create_date", "Book_title","Book_author","Status",
                             Book library serializer.py
book_list.html
                                                                                             book list student.html
src > library_component > 💠 views.py > ...
         from django.shortcuts import
                                             render,reverse,redirect
         from .models import Library_data
         from .serializers import BookLibrarySerialize
   3
         from rest_framework.response import Response
         from rest_framework import status, generics
   9
  10
  11
12
  13
14
         class BookView(generics.CreateAPIView):
             serializer_class = BookLibrarySerializer
              queryset = Library_data.objects.all()
  15
  16
                  post(self, request, format=None):
serializer = BookLibrarySerializer(data=request.data)
  18
                   if serializer.is_valid():
  19
                       serializer.save()
                     Book list view api
  src > library_component > 🏺 views.py > ...
          class BookRetrieveView(generics.CreateAPIVi(⋈):
    31
    32
               serializer class = BookLibrarySerializer
               queryset = Library data.objects.all()
    33
               def create(self, request, *args, **kwargs):
    queryset = Library_data.objects.filter(pk=kwargs['pk']).first()
    34
    35
                    if queryset:
    36
    37
                         serializer = self.serializer_class(queryset)
                    return Response({"status": True, "data": serializer.data}, status=status.HTTP_200_0K)
return Response({"status": False, "data": "Data not found"}, status=status.HTTP_404_NOT_FOUND
    38
    39
    40
            Book detail view api
  views.py .../library_component X
                                    models.py
                                                     form.py
                                                                       book_list.html
                                                                                            book_list_student.html
                                                                                                                         urls.py
  src > library_component > 💠 views.py > ...
    55
          class BookDetailDeleteView(generics.DestroyAPIView):
    56
               serializer_class = BookLibrarySerializer
    57
                                                                                                                  Ž
               def delete(self, request, *args, **kwargs):
    queryset = Library_data.objects.filter(pk=kwargs['pk']).first()
    58
    59
```

serializer = self.serializer_class(queryset, data=request.data, partial=True)

queryset.delete()
return Response({"status": True, "data": "Data deleted successfully"}, status=status

return Response({"status": False, "data": serializer.errors}, status=status.HTTP_400_BAD return Response({"status": False, "data": "Data not found"}, status=status.HTTP_404_NOT_FOUND

Book delete view api

if queryset:

if serializer.is valid():

60

61

62

63 64 65

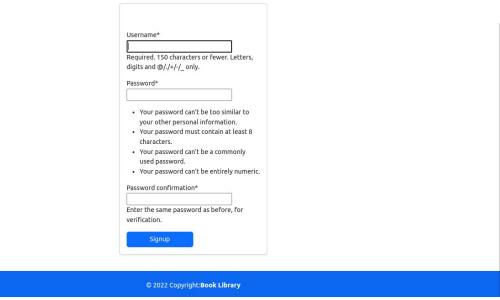
66 67

```
40
41
42
     class BookDetailUpdateView(generics.UpdateAPIView):
43
         serializer class = BookLibrarySerializer
44
         def update(self, request, *args, **kwargs):
             queryset = Library_data.objects.filter(pk=kwargs['pk']).first()
45
46
             if queryset:
47
                 serializer = self.serializer class(queryset, data=request.data, partial=True)
48
                 if serializer.is valid():
                     serializer.save()
49
                     return Response({"status": True, "data": "Data updated successfully"}, status=status
50
                 return Response({"status": False, "data": serializer.errors}, status=status.HTTP 400 BAD
51
             return Response({"status": False, "data": "Data not found"}, status=status.HTTP 404 NOT FOUNI
52
53
```

update api

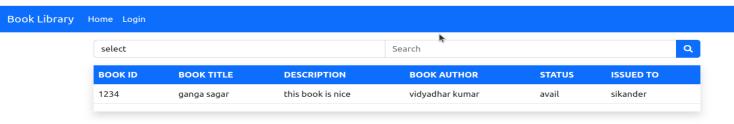
```
і неір
  models.py
                        urls.py .../library_component X
                                                                   form.py
                                                                                           book_list.html
                                                                                                                       book_list_student.html
                                                                                                                                                             urls.py
   src > library_component > 🥏 urls.py > ...
             from django.urls import path
              from .views import BookView,BookRetrieveView,BookDetailUpdateView,BookDetailDeleteView
      3
             urlpatterns = [
      4
      5
                    path('',BookView.as_view(),name="add_book"),
                   path(',book/lew.as_view(), name= add_book ),
path('book/get-detail/<int:pk>', BookRetrieveView.as_view(), name="get_detail"),
path('book/update-detail/<int:pk>', BookDetailUpdateView.as_view(), name="update_detail"),
path('book/delete-detail/<int:pk>', BookDetailDeleteView.as_view(), name="delete"),
       6
       7
      8
       9
```

- → Books library backend code written in app name: frontend
- → In view create,List,Detail,delete,update,login,signup Admin,logout,
- → for frontend django template ,bootstrap,crispy forms used User usecase
- → There is two user student and admin
- → student can only view books list all data
- → admin has to login after get option of add books, retrieve book detail, update book detail, delete book
- → url: http://127.0.0.1:8000



→ admin password reset using email using django console email backend

for create new admin



© 2022 Copyright:Book Library

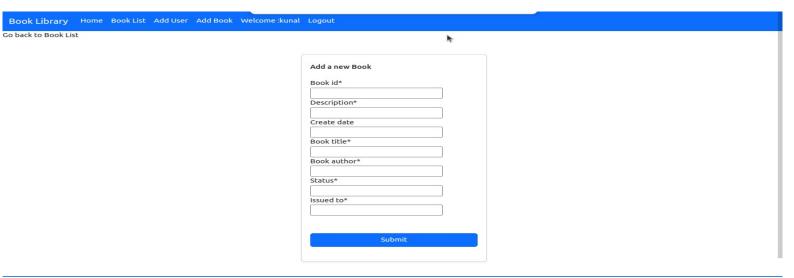
Home view for Students, search bar using select and search filter in javascript

after login admin get option of create new admin view all data add book ,edit book detail, view book



© 2022 Copyright:Book Library

detail, delete book by id



add new book

book search filter



Password reset email sent to customer mail in django console email backend

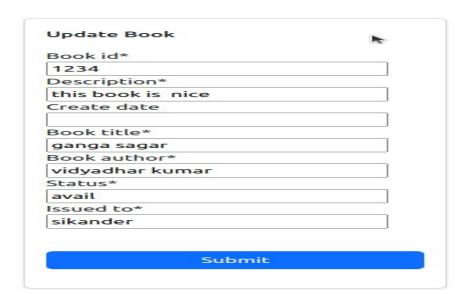
Content-Type: tex MIME-Version: 1.0 text/plain; charset="utf-8" Content-Transfer-Encoding: 7bit Subject: Password reset on 127.0.0.1:8000 To: kunal@gmail.com
Date: Fri, 16 Sep 2022 13:05:59 -0000
Message-ID: <166333355979.909319.18069651241729968260@kunal10x> You've requested to reset your password Please go to the following URL to enter your new password: http://127.0.0.1:8000/password-reset-confirm/MQ/bbuhpz-dd657fae2d3ca92c5541c8ba79d1e096/ Reset your password Email* Reset Password Already have an account? © 2022 Copyright:Book Library Book Library Home Login Enter your new password New password* · Your password can't be too similar to your other personal information. Your password must contain at least 8 characters. Your password can't be a commonly used password. Your password can't be entirely numeric. New password confirmation* © 2022 Copyright:Book Library

Book Library Home Login

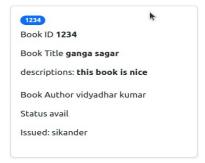
Password reset complete

You have successfully reset your password. Click <u>here</u> to login

© 2022 Copyright:Book Library



Book Library Home Book List Add User Add Book Welcome :kunal Logout



© 2022 Copyright:Book Library

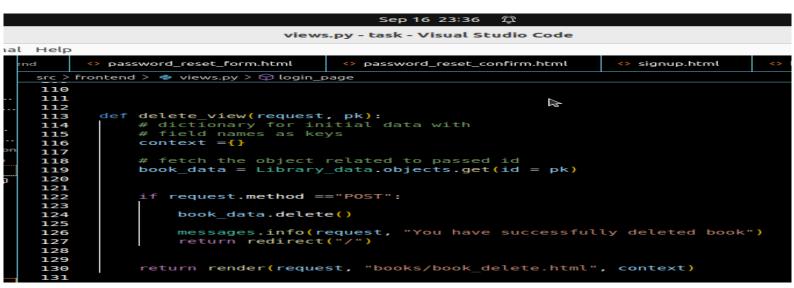
book detail view

frontend app MVT code

```
signup.html
       ♦ password_reset_form.html
♦ password_reset_confirm.html

    views.py .../frontend × ▷ ∨ □ ···

                                                                                                                  login.html
\operatorname{src} > \operatorname{frontend} > 	 \operatorname{\rellow} = \operatorname{views.py} > \operatorname{\rellow} \operatorname{login\_page}
  25
                return reverse("login")
login_page(request):
 26
27
28
30
33
33
34
35
37
38
39
40
41
44
45
46
47
50
51
52
53
54
55
                                                                         S
                form_class=LoginForm
                message = ''
                form=LoginForm
                if request.method == 'POST':
                      form = form_class(request.POST)
                      if form.is_valid():
                            user = authenticate(
                                 username=form.cleaned_data['username'],
password=form.cleaned_data['password'],
                            if user is not None:
    login(request, user)
    message = 'Hello {user.username}! You have been logged in'
    return redirect("/home")
                                  message = 'Login failed!'
                 return render(request, 'registration/login.html', context={ 'message': message,'form':form})
          @login_required
          def index(request):
                response=requests.get('http://127.0.0.1:8000/api').json()
print("Response:",response)
                print("Response : ", response)
return render(request, 'books/book_list.html',{'response':response})
          def home (request).
                                                                                                 Ln 43, Col 39 Spaces: 4 UTF-8 LF Python 3.10.4 ('task_env': venv) 👨 🕻
```



```
Sep 16 23:35 🖺
                                                                                                                                                40 🗐
                                   views.py - task - Visual Studio Code
                                                                                                                                                nal Help
          → password_reset_form.html
                                                                             signup.html
                                                                                                ♦ login.html
views.py .../frontend ×
> < □ ···</p>
   25 26 27 28 29 30 31 32 33 34 35 36 41 42 44 45 44 45 51 52 53 45 55
                                                              1
                 form class=LoginForm
                 message = ''
                 message =
form=LoginForm
if request.method == 'POST':
    form = form_class(request.POST)
                     if form.is_valid():
    user = authenticate(
        username=form.cleaned_data['username'],
        password=form.cleaned_data['password'],
                           )
if user is not None:
    login(request, user)
    message = 'Hello {user.username}! You have been logged in'
    return redirect("/home")
                 etse:
| message = 'Login failed!'
return render(request, 'registration/login.html', context={ 'message': message,'form':form})
           @login_required def index(reques
                 index(request):
                 response=requests.get('http://127.0.0.1:8000/api').json()
print("Response :".response)
                 print("Response :",response)
return render(request,'books/book_list.html',{'response':response})
                                                                                  Ln 43, Col 39 Spaces: 4 UTF-8 LF Python 3.10.4 ('task_env': venv) 👂 🕻
                    context['form']= form
return render(request, "books/book_create.html", context)
     81
82
83
                                                                Sep 16 23:34 💆
                                              settings.py - task - Visual Studio Code
   Help
   (I) READM
 ook create.html
                         <> navbar.html
                                                        base.html
                                                                                 scripts.html

≡ requirements.txt

               AUTH_USER_MODEL = 'library_component.User'
                                                                                               S
               EMAIL_BACKEND = "django.core.mail.backends.console.EmailBackend"
LOGIN_REDIRECT_URL="/books"
LOGOUT_REDIRECT_URL="/"
               CORS_ORIGIN_WHITELIST = [
'http://localhost:3000',
               # Default primary key field type
# <u>https://docs.djangoproject.com/en/4.l/ref/settings/#default-auto-field</u>
               DEFAULT AUTO FIELD = 'django.db.models.BigAutoField'
```

Ln 14, Col 29 Spaces: 4

Sep 16 23:34 🛭 🗘

settings.py - task - Visual Studio Code