

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 Template for frontend
- Virtual 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
3 from rest_framework import serializers
4 from library_component.models import Library_data
5
6 class BookLibrarySerializer(serializers.ModelSerializer):
7
8     class Meta:
9         model = Library_data
10         fields = ('id', 'Book id', "description", "create date", "Book title", "Book author", "Status",
```

Book library serializer.py

```
src > library_component > views.py > ...
1 from django.shortcuts import render, reverse, redirect
2 from .models import Library_data
3 from .serializers import BookLibrarySerializer
4 from rest_framework.response import Response
5 from rest_framework import status, generics
6
7
8
9
10
11
12
13 class BookView(generics.CreateAPIView):
14     serializer_class = BookLibrarySerializer
15     queryset = Library_data.objects.all()
16
17     def post(self, request, format=None):
18         serializer = BookLibrarySerializer(data=request.data)
19         if serializer.is_valid():
20             serializer.save()
```

Book list view api

```
src > library_component > views.py > ...
31 class BookRetrieveView(generics.CreateAPIView):
32     serializer_class = BookLibrarySerializer
33     queryset = Library_data.objects.all()
34     def create(self, request, *args, **kwargs):
35         queryset = Library_data.objects.filter(pk=kwargs['pk']).first()
36         if queryset:
37             serializer = self.serializer_class(queryset)
38             return Response({"status": True, "data": serializer.data}, status=status.HTTP_200_OK)
39         return Response({"status": False, "data": "Data not found"}, status=status.HTTP_404_NOT_FOUND)
40
41
```

Book detail view api

```
src > library_component > views.py > ...
55
56 class BookDetailDeleteView(generics.DestroyAPIView):
57     serializer_class = BookLibrarySerializer
58     def delete(self, request, *args, **kwargs):
59         queryset = Library_data.objects.filter(pk=kwargs['pk']).first()
60         if queryset:
61             serializer = self.serializer_class(queryset, data=request.data, partial=True)
62             if serializer.is_valid():
63                 queryset.delete()
64                 return Response({"status": True, "data": "Data deleted successfully"}, status=status.HTTP_200_OK)
65             return Response({"status": False, "data": serializer.errors}, status=status.HTTP_400_BAD_REQUEST)
66         return Response({"status": False, "data": "Data not found"}, status=status.HTTP_404_NOT_FOUND)
67
```

Book delete view api

```

40
41
42 class BookDetailUpdateView(generics.UpdateAPIView):
43     serializer_class = BookLibrarySerializer
44     def update(self, request, *args, **kwargs):
45         queryset = Library_data.objects.filter(pk=kwargs['pk']).first()
46         if queryset:
47             serializer = self.serializer_class(queryset, data=request.data, partial=True)
48             if serializer.is_valid():
49                 serializer.save()
50                 return Response({"status": True, "data": "Data updated successfully"}, status=status
51                 return Response({"status": False, "data": serializer.errors}, status=status.HTTP_400_BAD
52                 return Response({"status": False, "data": "Data not found"}, status=status.HTTP_404_NOT_FOUNI
53
54

```

update api

Help

models.py | urls.py .../library_component X | form.py | book_list.html | book_list_student.html | urls.py .../f

src > library_component > urls.py > ...

```

1 from django.urls import path
2 from .views import BookView, BookRetrieveView, BookDetailUpdateView, BookDetailDeleteView
3
4 urlpatterns = [
5     path('', BookView.as_view(), name="add_book"),
6     path('book/get-detail/<int:pk>', BookRetrieveView.as_view(), name="get_detail"),
7     path('book/update-detail/<int:pk>', BookDetailUpdateView.as_view(), name="update_detail"),
8     path('book/delete-detail/<int:pk>', BookDetailDeleteView.as_view(), name="delete"),
9
10

```

Help

models.py X | form.py | book_list.html | book_list_student.html

src > library_component > models.py > User

```

1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3 from django.db.models.signals import post_save
4
5
6
7
8 class User(AbstractUser):
9     is_organisor = models.BooleanField(default=True)
10
11
12 class UserProfile(models.Model):
13     user = models.OneToOneField(User, on_delete=models.CASCADE)
14
15
16 class Library_data(models.Model):
17     Book_id=models.CharField(max_length=50)
18     description = models.CharField(max_length=150)
19     create_date=models.DateTimeField(null=True, blank=True)
20     Book_title=models.CharField(max_length=100)
21     Book_author=models.CharField(max_length=150)
22     Status=models.CharField(max_length=100)
23     Issued_to=models.CharField(max_length=100)
24     class Meta:
25         ordering = ['create_date',]
26
27
28
29 def __str__(self):
30     return f"{self.Book_title} {self.Book_author}"
31
32

```

Ln 8, Col

Model.py

- 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

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

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.

Password confirmation*

Enter the same password as before, for verification.

Signup

© 2022 Copyright:Book Library

- admin password reset using email using django console email backend

for create new admin

Book Library Home Login					
select			Search		Q
BOOK ID	BOOK TITLE	DESCRIPTION	BOOK AUTHOR	STATUS	ISSUED TO
1234	ganga sagar	this book is nice	vidyadhar kumar	avail	sikander

© 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

Book Library

Home

Book List

Add User

Add Book

Welcome :kunal

Logout

+

select

Search

BOOK ID	BOOK TITLE	DESCRIPTION	BOOK AUTHOR	STATUS	ISSUED TO	EDIT
1234	ganga sagar	this book is nice	vidyadhar kumar	avail	sikander	<div><div>+</div><div></div><div></div><div></div></div>

© 2022 Copyright:Book Library

detail, delete book by id

Book Library

Home

Book List

Add User

Add Book

Welcome :kunal

Logout

Go back to Book List

Add a new Book

Book id*

Description*

Create date

Book title*

Book author*

Status*

Issued to*

Submit

add new book

book search filter

Book Library

Home

Login

select

select

1234

Search

	BOOK AUTHOR	STATUS	ISSUED TO		
1234	ganga sagar	this book is nice	vidyadhar kumar	avail	sikander

© 2022 Copyright:Book Library

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

```
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: Password reset on 127.0.0.1:8000
From: webmaster@localhost
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/>

[Book Library](#) [Home](#) [Login](#)

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*

Confirm new password

© 2022 Copyright: **Book Library**

Password reset complete

You have successfully reset your password. Click [here](#) to login

Update Book

Book id*

1234

Description*

this book is nice

Create date

Book title*

ganga sagar

Book author*

vidyadhar kumar

Status*

avail

Issued to*

sikander

Submit

1234

Book ID 1234

Book Title **ganga sagar**

descriptions: **this book is nice**

Book Author vidyadhar kumar

Status avail

Issued: sikander

book detail view

frontend app MVT code

```
src > frontend > views.py > login_page
25     return reverse("login")
26 def login_page(request):
27
28     form_class=LoginForm
29
30     message = ''
31     form=LoginForm
32     if request.method == 'POST':
33         form = form_class(request.POST)
34
35         if form.is_valid():
36             user = authenticate(
37                 username=form.cleaned_data['username'],
38                 password=form.cleaned_data['password'],
39             )
40             if user is not None:
41                 login(request, user)
42                 message = 'Hello {user.username}! You have been logged in'
43                 return redirect("/home")
44             else:
45                 message = 'Login failed!'
46     return render(request, 'registration/login.html', context={ 'message': message, 'form':form})
47
48 @login_required
49 def index(request):
50     response=requests.get('http://127.0.0.1:8000/api').json()
51     print("Response :",response)
52     return render(request,'books/book_list.html',{'response':response})
53
54
55
56 def home(request):
```

```
Sep 16 23:36
views.py - task - Visual Studio Code
src > frontend > views.py > login_page
110
111
112
113 def delete_view(request, pk):
114     # dictionary for initial data with
115     # field names as keys
116     context ={}
117
118     # fetch the object related to passed id
119     book_data = Library_data.objects.get(id = pk)
120
121
122     if request.method == "POST":
123
124         book_data.delete()
125
126         messages.info(request, "You have successfully deleted book")
127         return redirect("/")
128
129
130     return render(request, "books/book_delete.html", context)
131
```


Sep 16 23:35

views.py - task - Visual Studio Code

```
src > frontend > views.py > login_page
25 |         return reverse("login")
26 | def login_page(request):
27 |
28 |     form_class=LoginForm
29 |
30 |     message = ''
31 |     form=LoginForm
32 |     if request.method == 'POST':
33 |         form = form_class(request.POST)
34 |
35 |         if form.is_valid():
36 |             user = authenticate(
37 |                 username=form.cleaned_data['username'],
38 |                 password=form.cleaned_data['password'],
39 |             )
40 |             if user is not None:
41 |                 login(request, user)
42 |                 message = 'Hello {user.username}! You have been logged in'
43 |                 return redirect("/home")
44 |             else:
45 |                 message = 'Login failed!'
46 |         return render(request, 'registration/login.html', context={ 'message': message, 'form': form})
47 |
48 | @login_required
49 | def index(request):
50 |     response=requests.get('http://127.0.0.1:8000/api').json()
51 |     print("Response :",response)
52 |     return render(request,'books/book_list.html',{'response':response})
53 |
54 |
55 |
56 | def home(request):
57 |
58 |
59 |
60 |
61 |     context['form']= form
62 |     return render(request, "books/book_create.html", context)
63 |
64 |
```

Ln 43, Col 39 Spaces: 4 UTF-8 LF Python 3.10.4 ('task_env': venv)

Sep 16 23:34

settings.py - task - Visual Studio Code

```
src > library_management > settings.py > ...
129 |
130 |
131 | AUTH_USER_MODEL = 'library_component.User'
132 |
133 |
134 | STATIC_URL = 'static/'
135 |
136 | STATICFILES_DIRS = [
137 |     BASE_DIR / "static"
138 | ]
139 |
140 |
141 | EMAIL_BACKEND = "django.core.mail.backends.console.EmailBackend"
142 | LOGIN_REDIRECT_URL="/books"
143 | LOGIN_URL = "/login"
144 | LOGOUT_REDIRECT_URL="/"
145 |
146 |
147 |
148 | CORS_ORIGIN_WHITELIST = [
149 |     'http://localhost:3000',
150 | ]
151 |
152 |
153 |
154 |
155 |
156 | # Default primary key field type
157 | # https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
158 |
159 | DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
160 |
```

Ln 14, Col 29 Spaces: 4 UTF-8

```
76 }
77
78 WSGI_APPLICATION = 'library_management.wsgi.application'
79
80 # Database
81 # https://docs.djangoproject.com/en/4.1/ref/settings/#databases
82
83
84 DATABASES = {
85     'default': {
86         'ENGINE': 'django.db.backends.mysql',
87         'NAME': 'book_library_db',
88         'HOST' : '127.0.0.1',
89         'PORT' : '3306',
90         'USER': 'root',
91         'PASSWORD': '',
92     }
93 }
94
95
```