Name : Bhaurao Banshi Walve.

Project : Library Management System.

Start project using command **django-admin startproject libmanagesys**

In **libmanagesys** project create two app’s by using **python manage.py startapp appname**

1. account
2. Library

First add account and Library apps in settings.py file

**settings.py**

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'account',

    'Library',

]

To use mysql database we need to change default sqlite3 database as.

**settings.py**

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.mysql',

        'NAME': 'libmanagesystem',

        'USER': 'root',

        'PASSWORD': '1234',

        'HOST': 'localhost',

        'PORT': '3306',

    }

}

Include tamplate and midia directory in settings.py file .

**settings.py**

import os

TEMPLATES = [

    {

        #...,

        'DIRS': [os.path.join(BASE\_DIR,'templates')],

        'APP\_DIRS': True,

        #.....

    },

]

STATIC\_URL = '/static/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'library/media')

MEDIA\_URL = '/media/'

STATICFILES\_DIRS = [

    os.path.join(BASE\_DIR, "library/static")

]

Create customize User models to change default behavior of login with username and password to email is and password.

Crate account app in project and make backends.py file and write following code.

**account/backends.py**

from django.contrib.auth.backends import ModelBackend

from django.contrib.auth import get\_user\_model

from django.db.models import Q

UserModel = get\_user\_model()

class EmailBackend(ModelBackend):

    def authenticate(self, request, username=None, password=None, \*\*kwargs):

        try:

            user = UserModel.objects.get(Q(email\_\_iexact=username))

        except UserModel.DoesNotExist:

            UserModel().set\_password(password)

            return

        except UserModel.MultipleObjectsReturned:

            user = UserModel.objects.filter( Q(email\_\_iexact=username)).order\_by('id').first()

        if user.check\_password(password) and self.user\_can\_authenticate(user):

            return user

Create custom model for user .write code in account/models.py file

**account/models.py**

from django.db import models

from django.contrib.auth.models import AbstractUser

from django.utils.translation import gettext\_lazy as \_

class CustomUser(AbstractUser):

    email = models.EmailField(\_('email address'), unique=True)

Register CustomUser model in admin.py file.

**account/ admin.py**

from django.contrib import admin

from account.models import CustomUser

admin.site.register(CustomUser)

After that add this file class in settings.py file.

**settings.py**

AUTH\_USER\_MODEL = 'account.CustomUser'

AUTH\_USER\_MODEL = 'account.CustomUser'

AUTHENTICATION\_BACKENDS = ['account.backends.EmailBackend']

Now edit next app Library

Create database table by using models functions, To create table Student, Book, Librarian, IssuedBook by using model class. Write following code in Library/models.py

**Library/models.py**

from django.db import models

from datetime import datetime,timedelta

from account.models import CustomUser

from django.contrib.auth import get\_user\_model

User = get\_user\_model()

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(CustomUser, 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)+']'

class Librarian(models.Model):

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

    sr\_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.id)+']'

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)

Now register all this models in admin site .Write following code in Library/admin.py file

**Library/admin.py**

from django.contrib import admin

from .models import \*

admin.site.register(Book)

admin.site.register(Student)

admin.site.register(IssuedBook)

admin.site.register(Librarian)

Write urls for project level ,In Project level urls include app’s urls and also give media path

**urls.py**

from django.contrib import admin

from django.urls import path

from django.urls.conf import include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',include('Library.urls')),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Write urls for Library app ,In this urls include all urls of project as follows.

**Library/urls.py**

from django.urls import path

from . import views

urlpatterns = [

    path('books',views.base ,name="books"),

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

    path("logout/", views.logout\_view, name="logout"),

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

    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("update\_book/<int:id>/", views.update\_book, name="update\_book"),

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

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

]

To write views for each each activity of Library management system , first include all library of django which are need to write views.py file

**Library/views.py**

from django.contrib.auth.forms import  AuthenticationForm

from django.shortcuts import render , redirect ,HttpResponse

from django.contrib.auth.models import auth

from django.contrib import messages

from .models import \*

from . import forms, models

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

from datetime import date

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

# Create your views here.

def home(request):

    return render(request, "Library/home.html")

Write view for display all book’s in library using function base .

**Library/views.py**

def base(request):

    books = Book.objects.all()

    return render(request , "Library/index.html", {'books':books})

User add\_book() function to add books in library , The librarian and superuser can add books in library. Use decorator for this function.

**Library/views.py**

@login\_required(login\_url = '/admin\_login')

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, "Library/add\_book.html", {'alert':alert})

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

To retrieve or display all book’s in library to superuser or librarian. for this use function view\_books() .

**Library/views.py**

@login\_required(login\_url = '/admin\_login')

def view\_books(request):

    books = Book.objects.all()

    return render(request, "Library/view\_books.html", {'books':books})

The superuser and librarian(admin) can view all students , for this use function in views.py as view\_student() .

**Library/views.py**

@login\_required(login\_url = '/admin\_login')

def view\_students(request):

    students = Student.objects.all()

    return render(request, "Library/view\_students.html", {'students':students})

The superuser and librarian(admin) can issue a book to particular student by using following issue\_book() function in views.py file

**Library/views.py**

@login\_required(login\_url = '/admin\_login')

def issue\_book(request):

    form = forms.IssueBookForm()

    if request.method == "POST":

        form = forms.IssueBookForm(request.POST)

        if form.is\_valid():

            obj = models.IssuedBook()

            obj.student\_id = request.POST['name2']

            obj.isbn = request.POST['isbn2']

            obj.save()

            alert = True

            return render(request, "Library/issue\_book.html", {'obj':obj, 'alert':alert})

    return render(request, "Library/issue\_book.html", {'form':form})

The forms for issue\_book() function is to create forms.py file in directory of Library and create class IssueBookForm

**Library/forms.py**

from django import forms

from django.contrib.auth.models import User

from . import models

class IssueBookForm(forms.Form):

    isbn2 = forms.ModelChoiceField(queryset=models.Book.objects.all(), empty\_label="Book Name [ISBN]", to\_field\_name="isbn", label="Book (Name and ISBN)")

    name2 = forms.ModelChoiceField(queryset=models.Student.objects.all(), empty\_label="Name [Branch] [Class] [Roll No]", to\_field\_name="user", label="Student Details")

    isbn2.widget.attrs.update({'class': 'form-control'})

    name2.widget.attrs.update({'class':'form-control'})

The superuser and librarian(admin) can retrieve or view all issue books to student by using function view\_issued\_book()

**Library/views.py**

@login\_required(login\_url = '/admin\_login')

def view\_issued\_book(request):

    issuedBooks = IssuedBook.objects.all()

    details = []

    for i in issuedBooks:

        days = (date.today()-i.issued\_date)

        d=days.days

        fine=0

        if d>5:

            day=d-5

            fine=day\*5

        books = list(models.Book.objects.filter(isbn=i.isbn))

        students = list(models.Student.objects.filter(user=i.student\_id))

        i=0

        for l in books:

            t=(students[i].user,students[i].user\_id,books[i].name,books[i].isbn,issuedBooks[0].issued\_date,issuedBooks[0].expiry\_date,fine)

            i=i+1

            details.append(t)

    return render(request, "Library/view\_issued\_book.html", {'issuedBooks':issuedBooks, 'details':details})

The student can see which book he/she taken from library and how much fine he/she want to pay in library.

**Library/views.py**

@login\_required(login\_url = '/student\_login')

def student\_issued\_books(request):

    student = Student.objects.filter(user\_id=request.user.id)

    issuedBooks = IssuedBook.objects.filter(student\_id=student[0].user\_id)

    li1 = []

    li2 = []

    for i in issuedBooks:

        books = Book.objects.filter(isbn=i.isbn)

        for book in books:

            t=(request.user.id, request.user.get\_full\_name, book.name,book.author)

            li1.append(t)

        days=(date.today()-i.issued\_date)

        d=days.days

        fine=0

        if d>6:

            day=d-5

            fine=day\*5

        t=(issuedBooks[0].issued\_date, issuedBooks[0].expiry\_date, fine)

        li2.append(t)

    return render(request,'Library/student\_issued\_books.html',{'li1':li1, 'li2':li2})

The student can view its own profile by using following function

**Library/views.py**

@login\_required(login\_url = '/student\_login')

def profile(request):

    return render(request, "Library/profile.html")

The student can be edit its own profile by using following function

**Library/views.py**

@login\_required(login\_url = '/student\_login')

def edit\_profile(request):

    student = Student.objects.get(user=request.user)

    if request.method == "POST":

        email = request.POST['email']

        phone = request.POST['phone']

        branch = request.POST['branch']

        classroom = request.POST['classroom']

        roll\_no = request.POST['roll\_no']

        student.user.email = email

        student.phone = phone

        student.branch = branch

        student.classroom = classroom

        student.roll\_no = roll\_no

        student.user.save()

        student.save()

        alert = True

        return render(request, "Library/edit\_profile.html", {'alert':alert})

    return render(request, "Library/edit\_profile.html")

The superuser and librarian(admin) can delete particular book record for the use function delete\_book() and delete particular Student record for the use function delete\_Student()

**Library/views.py**

def delete\_book(request, myid):

    books = Book.objects.filter(id=myid)

    books.delete()

    return redirect("/view\_books")

def delete\_student(request, myid):

    students = Student.objects.filter(id=myid)

    students.delete()

    return redirect("/view\_students")

The student can be modify its own account password using following function

**Library/views.py**

def change\_password(request):

    if request.method == "POST":

        current\_password = request.POST['current\_password']

        new\_password = request.POST['new\_password']

        try:

            u = User.objects.get(id=request.user.id)

            if u.check\_password(current\_password):

                u.set\_password(new\_password)

                u.save()

                alert = True

                return render(request, "Library/change\_password.html", {'alert':alert})

            else:

                currpasswrong = True

                return render(request, "Library/change\_password.html", {'currpasswrong':currpasswrong})

        except:

            pass

    return render(request, "Library/change\_password.html")

If student want to see how much fine he/she want to pay , how many books he/she taken from library and how many books available in library for that student need to register it self or create own account using following function

**Library/views.py**

def student\_registration(request):

    if request.method == "POST":

        username = request.POST['username']

        first\_name = request.POST['first\_name']

        last\_name = request.POST['last\_name']

        email = request.POST['email']

        phone = request.POST['phone']

        branch = request.POST['branch']

        classroom = request.POST['classroom']

        roll\_no = request.POST['roll\_no']

        image = request.FILES['image']

        password = request.POST['password']

        confirm\_password = request.POST['confirm\_password']

        if password != confirm\_password:

            passnotmatch = True

            return render(request, "Library/student\_registration.html", {'passnotmatch':passnotmatch})

        user = User.objects.create\_user(username=username, email=email, password=password,first\_name=first\_name, last\_name=last\_name)

        student = Student.objects.create(user=user, phone=phone, branch=branch, classroom=classroom,roll\_no=roll\_no, image=image)

        user.save()

        student.save()

        alert = True

        return render(request, "Library/student\_registration.html", {'alert':alert})

    return render(request, "Library/student\_registration.html")

Login view for student , Using this view student can login with this website or portal

**Library/views.py**

def student\_login(request):

    if request.method == "POST":

        username = request.POST['username']

        password = request.POST['password']

        user = authenticate(username=username, password=password)

        if user is not None:

            login(request, user)

            if request.user.is\_superuser or request.user.is\_staff :

                return HttpResponse("You are not a student!!")

            else:

                return redirect("/profile")

        else:

            alert = True

            return render(request, "Library/student\_login.html", {'alert':alert})

    return render(request, "Library/student\_login.html")

Login view for librarian(admin) and superuser , Using this view student can login with this website or portal

**Library/views.py**

def admin\_login(request):

    if request.method == "POST":

        username = request.POST['username']

        password = request.POST['password']

        user = authenticate(username=username, password=password)

        if user is not None:

            login(request, user)

            if request.user.is\_superuser or request.user.is\_staff :

                return redirect("/add\_book")

            else:

                return HttpResponse("You are not an admin.")

        else:

            alert = True

            return render(request, "Library/admin\_login.html", {'alert':alert})

    return render(request, "Library/admin\_login.html")

The only superuser can create librarian(admin) using following view

**Library/views.py**

def librarian\_registration(request):

    if request.method == "POST":

        username = request.POST['username']

        first\_name = request.POST['first\_name']

        last\_name = request.POST['last\_name']

        email = request.POST['email']

        phone = request.POST['phone']

        sr\_no = request.POST['sr\_no']

        image = request.FILES['image']

        password = request.POST['password']

        confirm\_password = request.POST['confirm\_password']

        if password != confirm\_password:

            passnotmatch = True

            return render(request, "Library/add\_librarian.html", {'passnotmatch':passnotmatch})

        user = User.objects.create\_user(username=username, email=email, password=password,first\_name=first\_name, last\_name=last\_name)

        librarian = Librarian.objects.create(user=user, phone=phone,sr\_no=sr\_no, image=image)

        user.is\_staff =True

        user.save()

        librarian.save()

        alert = True

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

    return render(request, "Library/add\_librarian.html")

The superuser and librarian(admin) can edit or update particular book record by using update\_book() view.

**Library/views.py**

@login\_required(login\_url = '/admin\_login')

def update\_book(request,id):

    book = Book.objects.get(id = id)

    if request.method == "POST":

        name = request.POST['name']

        author = request.POST['author']

        isbn = request.POST['isbn']

        category = request.POST['category']

        book.name= name

        book.author=author

        book.isbn=isbn

        book.category=category

        book.save()

        alert = True

        return render(request, "Library/update\_book.html", {'book':book ,'alert':alert})

    return render(request, "Library/update\_book.html",{'book':book} )