Agenda of this week: To setup the Django project to start developing all the routes, views, models etc.

Steps need to follow to setup the project:

>pip freeze	--To check which packages installed in PYTHON
>pip install django	-- If Django not installed.
>pip install mysqlclient	-- To work with MySQL database.
>django-admin startproject bookclub	--To create a new project in Django

To Run Django Project

>change directory to the project directory i.e.,C:\Users\user1\Desktop\bookclub\bookclub> --
>python manage.py runserver

To Stop Django Project

>CTRL + C

Start develop the models used for the web app. Here in django models are equal to tables in MySQL. Some models are given below for better understanding;

Models.py

```
class Books(models.Model):
    book_id = models.AutoField(primary_key=True)
    book_title = models.CharField(max_length=255)
    book_category = models.ForeignKey(BookCategories, models.DO_NOTHING)
    quantity = models.IntegerField()
    author = models.CharField(max_length=255)
    edition = models.CharField(max_length=255)
    published_date = models.DateField()
    publisher = models.CharField(max_length=255)
    isbn = models.CharField(db_column='ISBN', max_length=255)  # Field name
made lowercase.
    student = models.ForeignKey('Students', models.DO_NOTHING)
    image =
models.FileField(upload_to="books/",max_length=250,null=True,default=None)

    def __str__(self):
        return self.book_title

    class Meta:
        managed = False
        db_table = 'books'
        verbose_name_plural = "books"
        ordering = ['book_id']

class Classes(models.Model):
    class_id = models.AutoField(primary_key=True)
```

```
    class_field = models.CharField(db_column='class', max_length=255)  # Field
renamed because it was a Python reserved word.

    def __str__(self):
        return self.class_field

    class Meta:
        managed = False
        db_table = 'classes'
        verbose_name_plural = "classes"
        ordering = ['class_id']



class Majors(models.Model):
    major_id = models.AutoField(primary_key=True)
    major = models.CharField(max_length=255)

    def __str__(self):
        return self.major

    class Meta:
        managed = False
        db_table = 'majors'
        verbose_name = "major"
        ordering = ['major_id']
```

urls.py file:

```
from django.contrib import admin
from django.urls import path
from bookclub import views
from django.conf import settings
from django.conf.urls.static import static
from lms.views import registration_view


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
    path('about/', views.about),
    path('books/', views.books),
    path('bookdetails/<int:book_id>/', views.bookdetails),
    path('contact/', views.contact),
    path('register/', registration_view, name="register"),
    path('userlist/', views.userslist),
    path('userpanel/', views.userpanel),
    path('admin_nav/', views.admin_nav),
    path('addprofile/', views.gender_view),
```

```
    path('showform/', views.showform),
    path('getuserid/<user_id>', views.getuserid),
    path('getbooks/', views.getbooks, name="getbooks"),
    path('bookborrowsview/', views.bookborrowsview),

]

if settings.DEBUG:
    urlpatterns+=static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

views.py file:

```
rom django.http import HttpResponse, HttpResponsePermanentRedirect
from django.shortcuts import render, redirect
from lms.models import *
from lms.forms import FormStudents, FormBorrows, MyForm, BorrowForm
from django.contrib import messages
from django.contrib.auth import login, authenticate
from lms.forms import RegistrationForm
from django.contrib.auth.models import User, Group
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.conf import settings


def home(request):
    post = Books.objects.all()
    return render(request,"index.html",{'posts':post})

def about(request):
    return render(request,"about.html")

def books(request):
    post = Books.objects.all()
    return render(request,"books.html",{'posts':post})

def bookdetails(request, book_id):
    post = Books.objects.filter(book_id=book_id)
    return render(request,"book-details.html",{'posts':post})

def contact(request):
    return render(request,"contact.html")

def getuserid(request):
    user = AuthUser.objects.latest('id')
    user1= User.objects.get(username = user).pk
    return user1
```

admin.py file:

```
from django.contrib import admin
from .models import *
from django.utils.html import format_html
from django.db import models
from django.conf import settings
from . models import *
from django.contrib.auth.models import Group
from lms import views
from django.shortcuts import render, redirect
from lms.forms import MyForm, BorrowForm
from django.contrib import messages



admin.site.site_header = 'BookClub'


@admin.action(description='Average')
def Average(modeladmin, request, queryset):
    queryset.update(rating = 'a')

@admin.action(description='Bad')
def Bad(modeladmin, request, queryset):
    queryset.update(rating = 'b')

@admin.action(description='Excellent')
def Excellent(modeladmin, request, queryset):
    queryset.update(rating = 'e')

class BookBorrowAdmin(admin.ModelAdmin):
    list_display = ('book', 'borrow_date', 'rating')
    actions = [Average,Bad,Excellent]

    def get_queryset(self, request, *args, **kwargs):
        if request.user.is_authenticated:
            user_id = int(request.user.id)
            current = User.objects.get(id=user_id).pk
            return BookBorrows.objects.all().filter(student=current)

    add_form_template = 'admin/lms/bookborrows.html'

    def add_view(self, request, form_url="", extra_context=None):
        return BookBorrowAdmin.bookborrowsview(reque
```

After creating all models and other files. Projects needs to migrate the database to the database server with all its default tables as well as our database model.

TO MIGRATE DEFAULT DATABASE IN DJANGO:

>python manage.py migrate

TO CREATE SUPERUSER TO ACCESS ADMIN PANEL:

>python manage.py createsuperuser

>python manage.py runserver