**Server information**

Address :27.255.81.126

User name : user

Password : cap2!@#

**Django version 3.2.3**

**Database**

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2',

'NAME': 'masterdb3',

'USER': 'postgres',

'PASSWORD':'1111',

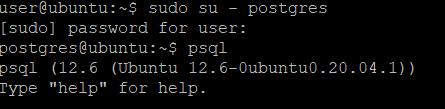
'HOST': '127.0.0.1',

'PORT': '5432',

}

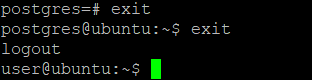
}

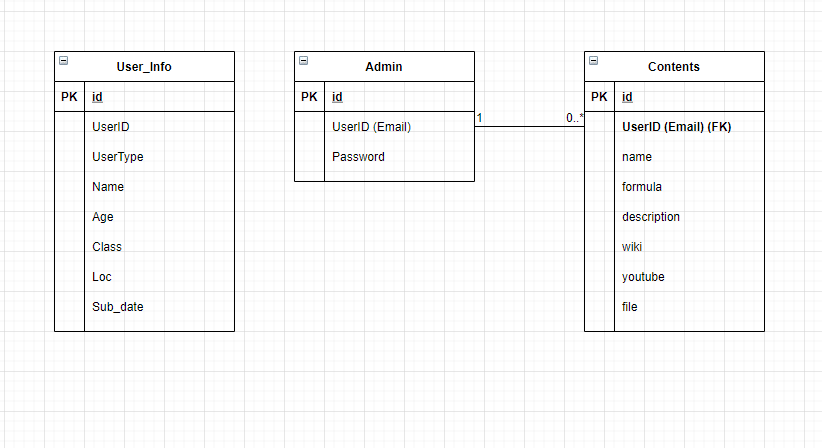
We use postgresql database here. The name of the database we are using is ‘masterdb3’.

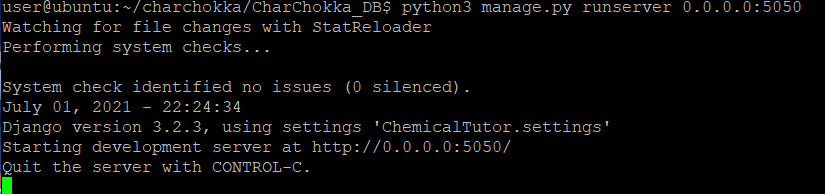


We can get access to the database like so. The password for sudo user is ‘cap2!@#’ .

By command ‘ exit ‘ exit from the database.

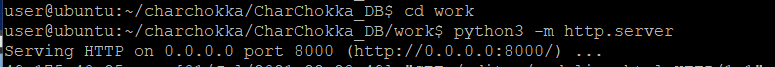




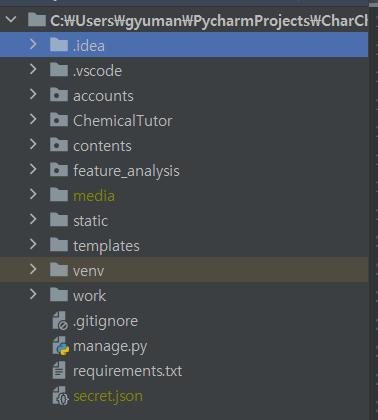


Run the django application on port 5050 by comand ‘ python3 manage.py runserver 0.0.0.0:5050 ‘.

This command should be done on the directory where manage.py file is in.



Modeling web application must be started in charchokka/CharChokka\_DB/work directory. This is done by command ‘ python3 –m http.server ‘. You can get access to the application on <http://27.255.81.126:8000/editor/modeling.html> in your browser.



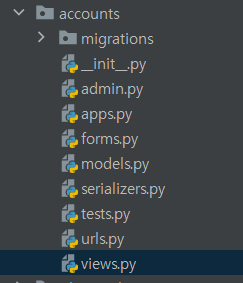
Basically we have 4 applications in our project.

**ChemicalTutor :**

This is our main application which links other 3 applications via url.py . Settings for this web project are also managed in settings.py .

**accounts** :

This is the application which manages the admin registration function, login function, creating and listing users.



**account/models.py**

**from** **django.db** **import** models

*# Create your models here.*

**class** **displayusername**(models.Model):

username=models.CharField(max\_length=100)

**class** **User\_Info**(models.Model):

UserID = models.CharField(max\_length=100)

UserType = models.CharField(max\_length=100)

Name = models.CharField(max\_length=100)

Age = models.IntegerField()

Class = models.CharField(max\_length=50)

Loc = models.CharField(max\_length=50)

Sub\_Date = models.DateField(auto\_now=**True**)

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

**return** self.UserID

we define our user model here. \*this is not the admin user\*

**account/views.py**

**from** **django.shortcuts** **import** render, redirect

**from** **django.conf** **import** settings

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

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

**from** **django.contrib.auth.models** **import** User

**from** **django.contrib.auth.forms** **import** UserCreationForm

**from** **django.contrib** **import** messages

**from** **.models** **import** \*

**from** **.forms** **import** UserForm

*# Create your views here.*

**def** home(request):

user\_info = User\_Info.objects.all()

context = {'users':user\_info}

**return** render(request, 'admin2.html', context)

we render our user information page by the function ‘home’

**def** showusername(request):

displaynames=User.objects.all()

**return** render(request, 'admin5.html', {"displayusername":displaynames})

**def** register(request):

form = UserCreationForm()

**if** request.method=='POST':

form=UserCreationForm(request.POST)

**if** form.is\_valid():

form.save()

messages.success(request, 'Admin User has been registered.')

**return** redirect('/adminlist/')

**return** render(request, 'admin6.html', {'form':form})

We render admin information page through ‘**showusername()**’. When ‘add admin’ button is clicked, **register()** function renders ‘admin6.html’ which has the input form for registering administrator. ‘UserCreationForm()’ method above let us handle input fields corresponding to our model. The parameter ‘user’ here is not the user defined in our models.py but imported from django library below.

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

**from** **django.contrib.auth.models** **import** User

**from** **django.contrib.auth.forms** **import** UserCreationForm

**def** createUser(request):

form = UserForm()

**if** request.method == 'POST':

form = UserForm(request.POST)

**if** form.is\_valid():

form.save()

**return** redirect('home')

context = {'form':form}

**return** render(request, 'user\_form.html', context)

**def** updateUser(request, pk):

user = User\_Info.objects.get(id=pk)

form = UserForm(instance=user)

**if** request.method == 'POST':

form = UserForm(request.POST, instance=user)

**if** form.is\_valid():

form.save()

**return** redirect('home')

context = {'form': form}

**return** render(request, 'user\_form.html', context)

**def** deleteUser(request, pk):

user = User\_Info.objects.get(id=pk)

**if** request.method=='POST':

user.delete()

**return** redirect('home')

context = {'item':user}

**return** render(request, 'delete.html', context)

we define the creation, updating, deletion of our user. User here is the user we defined in our models.py . This is different from the admin user.

**def** logoutUser(request):

logout(request)

**return** redirect('/login')

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

**def** index(request):

**return** render(request, 'admin1.html')

**def** loginPage(request):

**if** request.user.is\_authenticated:

**return** redirect('index')

**else**:

**if** request.method == 'POST':

username = request.POST.get('username')

password = request.POST.get('password')

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

**if** user **is** **not** **None**:

login(request, user)

**return** redirect('index')

**else**:

messages.info(request, 'Username OR password is incorrect')

context = {}

**return** render(request, 'admin\_login.html', context)

**logoutUser()** : logout and redirects to the login page.

**index():** renders our main page : which has four icons for each application.

**loginPage():** renders the login page and authenticate admin user. We use authenticate method by importing django library as below.

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

**accounts/urls.py**

urlpatterns = [

path('', index, name = 'index'),

path('login/', loginPage, name='login'),

path('adminlist/', showusername, name='adminlist'),

path('register/', register, name='register'),

path('logout/', logoutUser, name='logout'),

path('home/', home, name='home'),

path('create\_user/', createUser, name='create\_user'),

path('update\_user/<str:pk>/', updateUser, name='update\_user'),

path('delete\_user/<str:pk>/', deleteUser, name='delete\_user'),

]

We link the functions defined above with the url and avails our application to call the function by its name.

**contents**:

**contents/models.py**

**from** **django.db** **import** models

**from** **django.contrib.auth.models** **import** User

*# Create your models here.*

**class** **Contents**(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE, null=**True**)

name = models.CharField(max\_length=100,null=**True**)

formula = models.CharField(max\_length=20,null=**True**)

description = models.TextField(null=**True**, max\_length=100)

wiki = models.CharField(null=**True**, max\_length=100)

youtube = models.CharField(null=**True**, max\_length=100)

file = models.FileField(null=**True**)

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

**return** self.name

like accounts/models.py above, we make our Model for contents. We need refer User (admin user) in our contents so that the User and content can be in one to many relationship.

**contents/serializers.py**

**from** **rest\_framework** **import** serializers

**from** **.models** **import** Contents

**class** **ContentsSerializer**(serializers.ModelSerializer):

**class** **Meta**:

model = Contents

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

we used serializer to use Django Restframework APIs. This is used for communication with the mobile application.

**contents/views.py**

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

**from** **django.core.paginator** **import** Paginator

**from** **rest\_framework** **import** generics, viewsets

**from** **.serializers** **import** ContentsSerializer

**from** **.models** **import** Contents

**from** **.forms** **import** AddContent

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

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

**def** add\_show(request):

**if** request.method == 'POST':

fm = AddContent(request.POST, request.FILES)

**if** fm.is\_valid():

fs=fm.save(commit=**False**)

fs.user=request.user

fs.save()

**else**:

fm = AddContent()

cont = Contents.objects.all()

paginator = Paginator(cont,5)

page\_number=request.GET.get('page')

cont\_obj=paginator.get\_page(page\_number)

**return** render(request, 'admin3.html', {'form':fm, 'contents': cont\_obj})

*# Create your views here.*

*#목록*

**class** **ContentsListCreateAPIView**(generics.ListCreateAPIView):

queryset = Contents.objects.all()

serializer\_class = ContentsSerializer

*# 수정/삭제하기*

**class** **ContentsDetailGenericAPIView**(generics.RetrieveUpdateDestroyAPIView):

queryset = Contents.objects.all()

serializer\_class = ContentsSerializer

add\_show(request) : renders the form for our content model, process the POST request and save the POST data into our database. request.FILES processes the upload of our 3d model file.

**ContentsListCreateAPIView :** listing the content data by using django’s restframework api.

**ContentsDetailGenericAPIView :** APIView from Django Restframework. We use this for communication with our mobile application. The serializer in contents/serializers.py parses our data into json format and ContentsDetailGenericAPIView renders the json data. It makes it possible for the application to get the data from the database.

**contents/urls.py**

**from** **django.conf.urls** **import** url

**from** **django.urls** **import** path, include

**from** **.** **import** views

**from** **.views** **import** ContentsDetailGenericAPIView, ContentsListCreateAPIView

urlpatterns = [

path('', views.add\_show, name="contents"),

path('list/', ContentsListCreateAPIView.as\_view(), name='contents\_list\_create'),

path('update/<int:pk>', ContentsDetailGenericAPIView.as\_view(), name='contents\_update'),

]

Linking the views with url.

Additional work needed in user activity status, feature analysis and filtering in contents.