Any time when we execute/run python code then {% %} when showing to user {{ }}
https://data-flair.training/blogs/django-static-files-handling/

https://simpleisbetterthancomplex.com

https://www.youtube.com/channel/UCASZ7zW_Egu0T4KG3YEdGfw/videos all tutorial Live
https://ultimatedjango.com/learn-django/lessons/install-setup-django-2/ django Linux

https://automationpanda.com/2017/09/14/django-projects-in-pycharm-community-edition/ setup
https://learndjango.com/tutorials/template-structure django structure

https://microservicesdocumentation.readthedocs.io/en/latest/contributing/python.html microservices
https://simpleisbetterthancomplex.com/tutorial/2017/08/01/how-to-setup-amazon-s3-in-a-django-project.html uploading user uploaded static file into S3 directly.
https://docs.celeryproject.org/en/stable/django/first-steps-with-django.html Django with celery
https://simpleisbetterthancomplex.com/tutorial/2017/08/20/how-to-use-celery-with-django.html celery

https://django-bootstrap4.readthedocs.io/en/latest/index.html for Django- bootstrap tutorial
https://django-extensions.readthedocs.io/en/latest/ for Django extensions
http://ccbv.co.uk/ Django CRUD operation using class base view.
https://www.linuxhelp.com/create-web-application-using-django on Linux
https://www.linuxhelp.com/how-to-create-basic-web-applications-using-django-part-2
https://simpleisbetterthancomplex.com/tutorial/2017/02/21/how-to-add-recaptcha-to-django-site.html reCaptcha using
https://realpython.com/location-based-app-with-geodjango-tutorial/ location base web apps

user request

Project urls

App urls

Views.py --> Model--> data show through templete.

views.py then its goes in side Home.py class then its render for front page
We create template then redder it inside views.py then we can pass data as dictionary key back to template like home.html using {{ abc }} **double curly use while you want to show something** {% %} **single use while using if ,for, any command** certified

To integrate Django and reacts JS

https://scotch.io/tutorials/build-a-to-do-application-using-django-and-react **nice tutorial**
You can use axios to get Django rest api data into react page.
In Django app : pip install django-cors-headers https://github.com/adamchainz/django-cors-headers **put setting in settings.py**
In react app : npm install axios -- save https://www.npmjs.com/package/axios **use http request to API and fetch data**

pip install django-cors-headers :
A Django App that adds Cross-Origin Resource Sharing (CORS) headers to responses.
This allows in-browser requests to your Django application from other origins. **Always put in first in installed app/ middleware**

% Csrf_token % its work for authentication

To show data to User Page (create **.html file i**nside template folder then make **noticeListViews** class in views.py which inherit From **(List View)** then Create **URL inside Project url then create APP URL**

Create registration page

Search on google then turn on

```
C:\Users\com>pip install django-registration-redux
```

install it then inside setting.py put 'registration' under INSTALL APP top of 'college' then put it inside setting.py
Then go terminal and migrate **only. We do make** makemigaraion **only at the time of create or changse table (modules)**

```
ACCOUNT_ACTIVATION_DAYS=3

EMAIL_HOST= 'smtp.gmail.com'
EMAIL_PORT= 587
EMAIL_USE_TLS= True

EMAIL_HOST_USER= 'ec.smtp.test2@gmail.com'
EMAIL_HOST_PASSWORD= 'xxxx'
```

your email

put it Inside project url go into views.py put decorator where you want to show only after login then put this code inside base.html page

```
{% if user.is_authenticated %}
<a href="/accounts/logout">Logout</a>
{% else %}
<a href="/accounts/login">Login</a>
{% endif %}
```

If "Less secure app access" is off for your account

If "Less secure app access" is turned off for your account, you can turn it back on. We recommend

```
LOGIN_REDIRECT_URL = "/"
```

For direct login

```
path('accounts/', include('registration.backends.default.urls'))
```

If you want to make any file you have to work for out place 1, view 2, App url 3 form.html 4 base.html

show all list for super user (admin) and only show the list for particular user

```
@method_decorator(login_required, name="dispatch")
class NoticeListView(ListView):
    model = Notice
    def get_queryset(self):
        if self.request.user.is_superuser:
            return Notice.objects.order_by("-id")
        else:
            return Notice.objects.filter(branch=self.request.user.profile.branch).order
```

Keep it inside views.py

```
from django.utils.decorators import method_decorator
```

If you want to run uit once only when application load the put this inside **__init__.py then goto apps.py**

```
default_app_config = 'college.apps.CollegeConfig'
```

Put ready() method this method register @receiver inside mysignal.py

```
def ready(self):
    import college.mysignal
```

Two way pass data by form
GET : we can pass only text , its less secure
POST : we can pass tex, file, image all , more secure
Inside view.py We can only upload image or file through POST method

```
def get_queryset(self):
    si = self.request.GET.get("si")
    if si == None:
        si = ""
```

```
.filter(Q(subject__icontains = si) | Q(msg__icontains = si)).order_by("-id
```

```
<input value="{{ request.GET.si }}" type="text" name="si" />
```

You can see value inside search fields

If you want to use image in you app then you have to install Pillow the go to settings and put Media Root and URL

```
PROJECT_ROOT = os.path.realpath(os.path.dirname(__file__))
MEDIA_ROOT = PROJECT_ROOT + '/static/'
MEDIA_URL = '/media/'
```

Then go to project url put this url

```
+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

If you uploding some file or image then this required In your form

```
<form enctype="multipart/form-data" method="post">
```

Make form good look use it then go setting and put **crisppy_forms** under install apps

```
pip install django-crispy-forms
```

Inside forms.html

```
{% load crispy_forms_tags %}
{{form | crispy}}
```

Inside **Templates** for create and update we use **form** like **modulesname_form**.html

If you want to pass something internally at the time of forms submit without user know then

```
class QuestionCreate(CreateView):
    model = Question
    fields = ["subject", "msg"]
    def form_valid(self, form):
        self.object = form.save()
        self.object.user = self.request.user
        self.object.save()
        return HttpResponseRedirect(self.get_success_url())
```

Our project

To working with image install pillow
step 1, create models. Imagefields inside Model.py 2nd put Media root and media url then 3rd create Url of your main project like

```
+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

https://www.youtube.com/channel/UCTZRcDjjkVajGL6wd76UnGg/playlists django project tutorial

Showing two table data

```
@method_decorator(login_required, name="dispatch")
class MyList(TemplateView):
    template_name = "college/mylist.html"
    def get_context_data(self, **kwargs):
        context = TemplateView.get_context_data(self, **kwargs)
        context["notices"] = Notice.objects.order_by("-id")[:3]
        context["questions"] = Question.objects.order_by("-id")[:3]
        return context;
```

2.d we create function of forms and object of form inside Veiws.py then passed as dictionary

```
from .forms import EmployeeForm

# Create your views here.

def employee_list(request):
    return render(request,"employee_register/employee_list.html")

def employee_form(request):
    form = EmployeeForm()
    return render(request,"employee_register/employee_form.html",{'form':form})
```

4 flows in django
view - class ka method or class banta tha
Url .py meai uska url banta tha
templateke under HTML file banta tha
base .html se link banta tha

To get data input from frontend side using forms.

Also use font asome link and icon https://fontawesome.com/

1st Create forms.py we need to put all the model properties inside forms.py for that we have to import model name like (Employee)
All the properties of model become the field of form page where you can input your data

```
employee_register > forms.py > EmployeeForm > Meta
from django import forms
from .models import Employee


class EmployeeForm(forms.ModelForm):

    class Meta:
        model = Employee
        fields = '__all__'
```

3rd go inside Employee_forms.html and put {{ form}} inside block content
Then you will be able to see all the field of your model into front page

```
{% extends "employee_register/base.html" %}
{% load crispy_forms_tags %}


{% block content %}
<form action="" method="post" autocomplete="off">
    {% csrf_token %}
    {{form|crispy}}
</form>
{% endblock content %}
```

Using crispy_forms we can make form look good

```
ip install django-crispy-forms
```

Inside settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'employee_register',
    'crispy_forms'
]

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

You can remove * mean required validation of the fields

```
self.fields['emp_code'].required = False
```

Recived data throough get and post and also showing data into services page

```
def employee_form(request):
    if request.method == "GET":
        form =  EmployeeForm()
        return render(request,"employee_register/employee_form.html",{'form':form})
    else:
        form =  EmployeeForm(request.POST)
        if form.is_valid():
            form.save()
        return redirect('/employee/list')
```

You can update models fields in to forms as you liike inside forms.py

```
class EmployeeForm(forms.ModelForm):

    class Meta:
        model = Employee
        fields = ('fullname','mobile','emp_code','position')
        labels = {
            'fullname':'Full Name',
            'emp_code':'EMP'
        }
```

update Fields name

We can modify
Like this with
Bootstrap in forms.html
We can put 2 field in
one single line

```
{% load crispy_forms_tags %}


{% block content %}
<form action="" method="post" autocomplete="off">
    {% csrf_token %}
    {{form.fullname|as_crispy_field}}
    {{form.mobile|as_crispy_field}}
    <div class="row">
        <div class="col-md-4">
            {{form.emp_code|as_crispy_field}}
        </div>
        <div class="col-md-8">
            {{form.position|as_crispy_field}}
        </div>
    </div>
</form>
{% endblock content %}
```

It show select option inside the positon fields.

```
def __init__(self, *args, **kwargs):
    super(EmployeeForm,self).__init__(*args, **kwargs)
    self.fields['position'].empty_label = "Select"
```

```
<button type="submit" class="btn btn-success"><i class="fas fa-database"></i> Submit</button>
```

Create button in foms.html

Displaying data from database to the main frontend pages MVT workflow

1. create models
2. Register model in admin.py and do migrations
3. Create views inside views.py also link template .html file inside it
4. Create url of views created inside apps ,urls.py
5. Then goto project url.py and put the apps url link inside it
6. Then goto template folder and create .html file and link with base.html with it.

We can put models name using forloop inside {{ item }} inside block content. Now you can see in page.

importing data in database using forms front page
Instead create model inside models.py we create form into foms.py all other are sampe process

1. create forms.py file then inside form create modelname+form class like (EmployeeForm) then put your model name in meta class.
2. create a formview inside views.py like Def employee_form and also link .html into form views and render it to .html and pass the form object as dictionary
3. create template .html file like employee_form.html and put form object inside using {{ form|crispy }} under {% block content %} then we can see all the form models fields into service page

Simpleisbetterthancomplex.com
Django REST with react by valentinog.com for API Web service

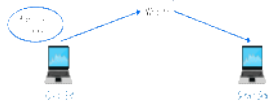
Mention the differences between Django, Pyramid and Flask.

Ans:

- Flask is a "micro framework" primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.
- Pyramid is built for larger applications. It provides flexibility and lets the developer use the right tools for their project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.
- Django can also be used for larger applications just like Pyramid. It includes an ORM.

Explain the use of session in Django framework?

Ans: Django provides a session that lets you store and retrieve data on a per-site-visitor basis. Django abstracts the process of sending and receiving cookies, by placing a session ID cookie on the client si de, and storing all the related data on the server side.

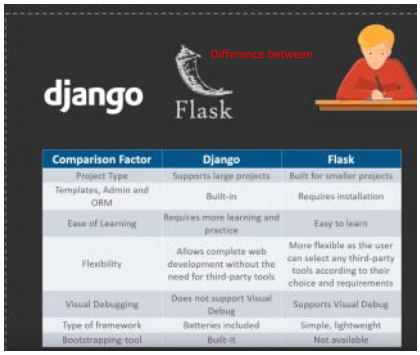


ORM : in model object we put data inside then its store in database when you read data you will get model object not a SQL that called ORM

In project

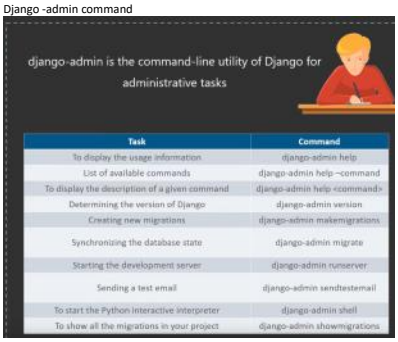
```
returnNotice.objects.filter(branch=self.request.user.profile.branch)
 branch wala column match with login user branch
```

Loginuser ka profile hai branch mai profile -user hoga request mai request hoga -self main
Self ko vita request user ko vitra profile profile ko vitra branch hunchh



Comparison Factor	Django	Flask
Project Type	Supports large projects	Built for smaller projects
Templates, Admin and ORM	Built-in	Requires installation
Ease of Learning	Requires more learning and practice	Easy to learn
Flexibility	Allows complete web development without the need for third-party tools	More flexible as the user can select any third-party tools according to their choice and requirements
Visual Debugging	Does not support Visual Debug	Supports Visual Debug
Type of framework	Batteries included	Simple, lightweight
Bootstrapping tool	Built-it	Not available

Django -admin command



Task	Command
To display the usage information	django-admin help
List of available commands	django-admin help --command
To display the description of a given command	django-admin help <command>
Determining the version of Django	django-admin version
Creating new migrations	django-admin makemigrations
Synchronizing the database state	django-admin migrate
Starting the development server	django-admin runserver
Sending a test email	django-admin sendtestemail
To start the Python interactive interpreter	django-admin shell
To show all the migrations in your project	django-admin showmigrations

Type of framework	Batteries included	Simple, lightweight
Bootstrapping-tool	Built-it	Not available

To start the Python interactive interpreter	django-admin shell
To show all the migrations in your project	django-admin showmigrations

15 What are the different inheritance styles in Django?

Inheritance style	Description
Abstract base classes	Used when you want to use the parent class to hold information
Multi-table inheritance	Used when you have to subclass an existing model
Proxy models	Used if you only want to modify the Python level behaviour of a model

17 What are 'signals'?

Signal	Description
django.db.models.signals.pre_save django.db.models.signals.post_save	Sent before or after a model's save() method is called
django.db.models.signals.pre_delete django.db.models.signals.post_delete	Sent before or after a model's delete() method or queryset's delete() method is called
django.db.models.signals.m2m_changed	Sent when Django starts or finishes an HTTP request

21 What are 'sessions'?

- Allows you to store and retrieve arbitrary data based on the per-site-visitors
- This framework stores data on the server-side
- Takes care of sending and receiving cookies
- These cookies consist of a session ID but not the actual data itself

24 Explain the caching strategies of Django?

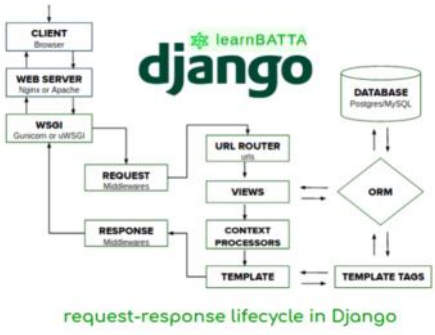
Strategy	Description
Memcached	Memory-based cache server
Filesystem caching	Cache values are stored as separate files
Local-memory caching	Default cache in case you have not specified any other
Database caching	Cache data will be stored in the database

5 Explain the use of Middlewares in Django.

- Middleware is a framework that is a light and low-level plugin system for altering Django's input and output globally
- It is a framework of hooks into the request/response processing
- Each component in middleware has some particular task
- For example, the AuthenticationMiddleware is used to associate users with requests using sessions
- Django provides many other middleware such as cache middleware, common middleware , GZip middleware, etc

29 Explain how a request is processed in Django?

- Django first determines which root URLconf module is to be used
- That particular Python module is loaded
- Then, Django looks for the variable urlpatterns
- URL patterns are run by Django, and it stops at the first match of the requested URL
- Once that is done, the Django then imports and calls the given view
- In case none of the URLs match the requested URL, Django invokes an error-handling view



request-response lifecycle in Django

Explain how to use file-based sessions?

In order to make use of file-based sessions, you will need to set the SESSION_ENGINE setting to "django.contrib.sessions.backends.file".

Give the exception classes present in Django.

Exception	Description
AppRegistryNotReady	Raised when you try to use your models before the app loading process
ObjectDoesNotExist	This is the base class for DoesNotExist exceptions
EmptyResultSet	This exception may be raised if a query won't return any result
FieldDoesNotExist	This exception is raised by a model's _meta.get_field() function in case the requested field does not exist
MultipleObjectsReturned	This is raised by a query if multiple objects are returned and only one object was expected

How can you limit admin access so that the objects can only be edited by those users who have created them?

Django's ModelAdmin class provides customization hooks using which, you can control the visibility and editability of objects in the admin. To do this, you can use the get_queryset() and has_change_permission().

What to do when you don't see all objects appearing on the admin site?

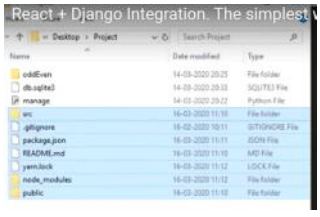
Inconsistent row counts are a result of missing Foreign Key values or if the Foreign Key field is set to null=False. If the ForeignKey points to a record that does not exist and if that foreign is present in the list_display method, the record will not be shown the admin changelist.

How to make a variable available to all the templates?

django

You can make use of the RequestContext in case all your templates require the same objects .This method takes an HttpRequest as its first parameter and it automatically populates s the context with a few variables

Just copy all the react project(app folder) file into Django project folder where manage.py for integration.



What is the project structure in Django?

```
|--manage.py
~..myproject
      |-- _ _ init__.py
      |-- settings.py
      |-- urls.py
      ~-- wsgi.py
```

To integrate Django and reacts JS

```
|-- settings.py
|-- urls.py
~-- wsgi.py
```

How request work :
http://example.com/cute-puppies. The browser does not care which part of the backend will handle the request, it just wants answers. (NOW.)
The requests arrives at the server, is passed to the web server (Nginx for example), and the web server hands the request over to the app server (Django handled by Gunicorn). The web server simply does what it's configured to do. Nothing magical here.

Anyway. Django gets the request, processes it, maybe looks up some data in a database and sends out a response.

there is many way to send the email
SMTP , other email services provider like sencris other console backend

Security Topics

- **Secret Keys**
- **Pickling**
- **SQL Injection**
- Cross-Site Request Forgery
- Cross-Site Scripting
- Session Hijacking
- Social Engineering
- Authentication
- Logs
- Admin

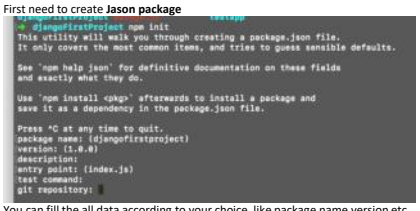
Webserver used in programing
java - tomcat web server
.net - IIS web server
php - apache web server
pythons & Django - Django development web server (comes with built in)

HTTP protocol also called as stateless protocol
Stateless: mean it cannot remember the value given in the web page.it doesn't maintain history
If you want to keep and see data after submitted then we need to do **state management**
Client side if you want to remember data in html5 then use **web storage** is there using this we can store web page value in client side browser.
In JavaScript : cookies are there to store web data in client side browser.

To integrate Django and reacts JS

https://scotch.io/tutorials/build-a-to-do-application-using-django-and-react **nice tutorial**
pip install django-cors-headers :
A Django App that adds Cross-Origin Resource Sharing (CORS) headers to responses. This allows in-browser requests to your Django application from other origins.

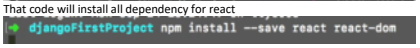
First need to create **Jason package**

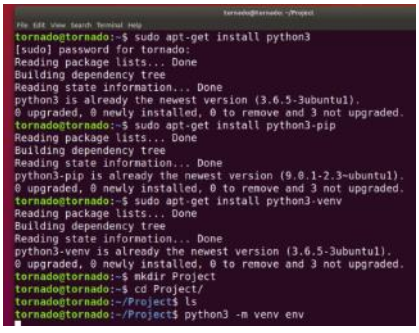


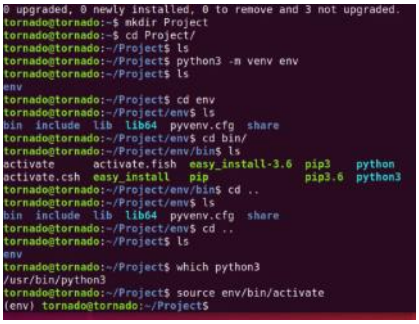
You can fill the all data according to your choice like package name version etc.

Then co back terminal then

That code will install all dependency for react

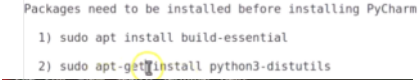


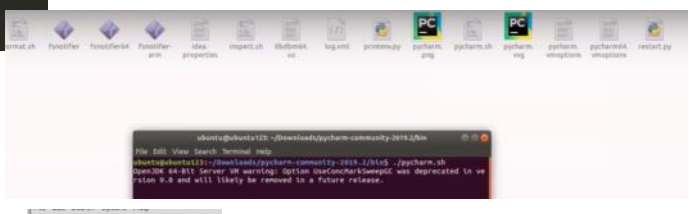




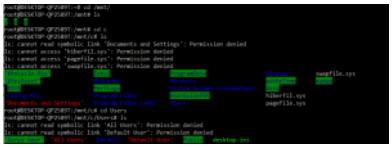
React Js Installatio in linux

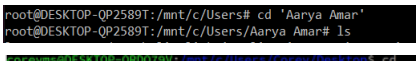
Pycharm installation

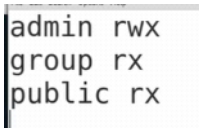




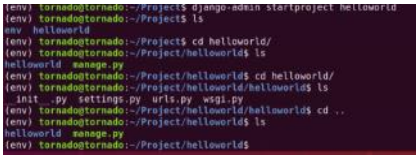
Ubantu access on windows 10

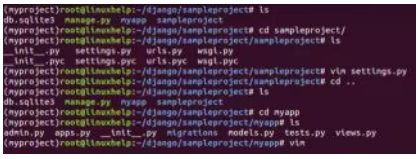


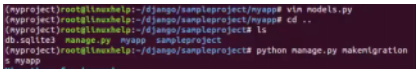


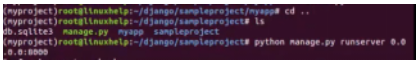


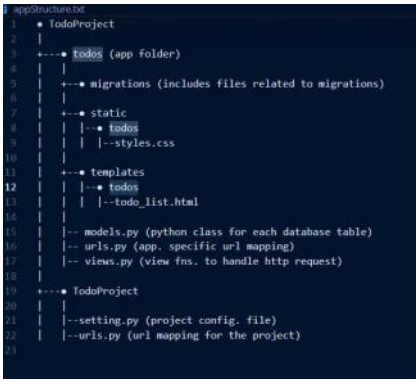
To create shortcut directory



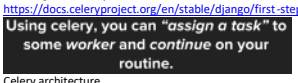




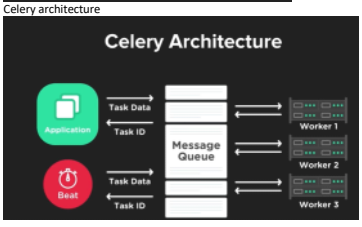




Working with celery:
https://docs.celeryproject.org/en/stable/django/first-steps-with-django.html



Celery architecture



Celery need database to store a task queue and perform one by one so we have rabitMQ message queue database & we can use redis as **broker database server**
But rabitmq is good for backed(broker database) processing than redis
Periodic task : use celerybeat tools (like supervior of worker , its keep your app and worker alive)

```
(celery-demo) [~/projects/mine/channel/co
hardik.p$ pip install django-celery-beat
```

In settings.py inder INSTALLED_APP

```
'django_celery_results',
'django_celery_beat',
```

Python manage.py inspectdb : show database table
need to install 2 model like
pip install celery it helps distributed task
pip install django-celery its help for integration
Got to your project settings.py THEN
Put djcelery package into INSTALLED_APPS

To create shortcut directory

Working on model

model. Object. All() = return all the model of the database.
Model .object. Filter() = return filter the items model of the database
Model .object. save() = insert or update new model into database
Model .object. Filter() = to get single model from database (like product model)

And store them into variable to display like
variablename = model. Object. All() = return all the model of the database.
Now you can create template to display this model into html page for user like
Inside template folder create index html
Instead sowing views index function use index.html by using render () method and put inside templete index.html

```
def index(request):
    products = Product.objects.all()
    return render(request, 'index.html')
```

And we can pass the model data through dictionary.

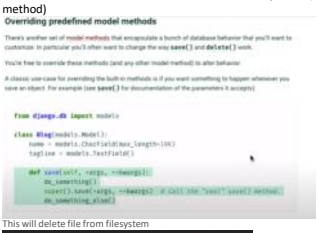
```
def index(request):
    products = Product.objects.all()
    return render(request, 'index.html',
                  {'products': products})
```

CURD OPERATIONS inside views.py

```
def emp(request):
    if request.method == "POST":
        form = EmployeeForm(request.POST)
        if form.is_valid():
            try:
                form.save()
                return redirect()
            except:
                pass
    else:
        form = EmployeeForm()
    return render(request, "index.html", {'form':form})

def show(request):
    employees = Employee.objects.all()
    return  render(request,"show.html",{'employees': employees})

def edit(request, id):
    employee = Employee.objects.get(id=id)
    return  render(request,"edit.html",{'employee' : employee})
```

```
def update(request, id):
    employee = Employee.objects.get(id=id)
    form = EmployeeForm(request.POST, instance= employee)
    if form.is_valid():
        form.save()
        return redirect('/show')
    return  render(request,"edit.html",{'employee': employee})
def delete(request, id):
    employee = EmployeeForm.objects.get(id=id)
    employee.delete()
    return redirect("/show")
```

We can override the models to delete file from file systems.(if not delete file even use delete post method)



This will delete file from filesystem

```
def delete(self, *args, **kwargs):
    self.pdf.delete()
    self.cover.delete()
    super().delete(*args, **kwargs)
```

From <https://www.kobrien.me/blog/post/building-simple-search-django/>
Search form

```
def search(request):
    query_string = ''
    found_entries = None
    if ('q' in request.GET) and request.GET['q'].strip():
        query_string = request.GET['q']
        entry_query = utils.get_query(query_string, ['title', 'bod
y',])
        posts = Post.objects.filter(entry_query).order_by('create
d')
        return render(request, 'search.html', { 'query_string': que
ry_string, 'posts': posts })
    else:
        return render(request, 'search.html', { 'query_string': 'Nu
ll', 'found_entries': 'Enter a search term' })
```

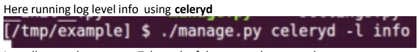
To store stattic file in to AWS S3 we need to install 2 things
1. Boto3
2. Django-storages

django-storages is an open-source library to manage storage backends like Amazon S3, Dropbox, OneDrive .
<https://simpleisbetterthancomplex.com/tutorial/2017/08/01/how-to-setup-amazon-s3-in-a-django-project.html>
Go to amazon s3, setup and add user setup AIM policy and group give full permission like built-in policy **AmazonS3FullAccess then get User**, **Access key ID** and the **Secret access key**.
Then
Inside settings.py

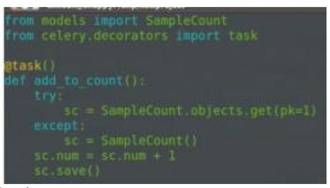
Got to your project settings.py THEN
Put djcelery package into INSTALLED_APPS

BROKER_URL : path for django ORM database table where store task

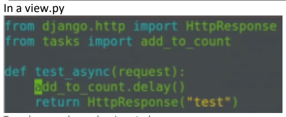
Here running log level info using **celeryd**



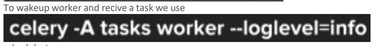
Its pull up works process . Take task of the que and execute them
Let see process of create a job : using decorator @shared_task



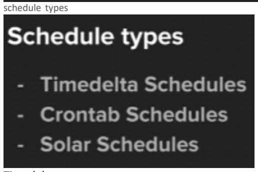
In a view.py



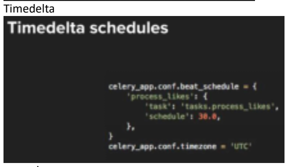
To wakeup worker and recive a task we use



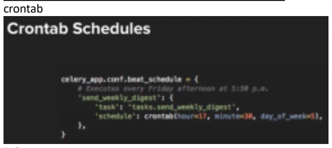
schedule types



Timedelta

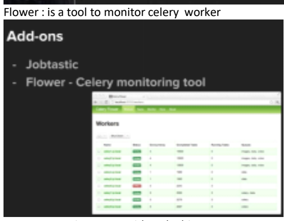


crontab

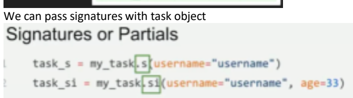


Solar

Flower : is a tool to monitor celery worker



We can pass signatures with task object



We can pass immutable signature so no extra parameter allow
Here callback error

OneDrive".
<https://simpleisbetterthancomplex.com/tutorial/2017/08/01/how-to-setup-amazon-s3-in-a-django-project.html>
Go to amazon s3, setup and add user setup AIM policy and group give full permission like
built-in policy **AmazonS3FullAccess then get User**, **Access key ID** and the **Secret access key**.
Then
Inside settings .py
Put storage in INSTALLED_APP[ ]

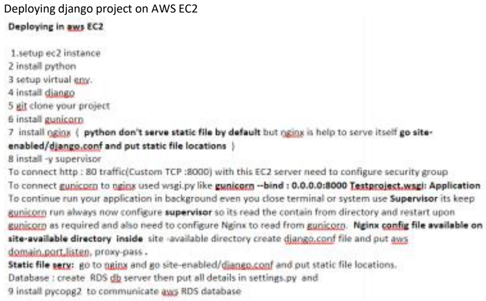
we will add the following to our /settings.py file:

```
AWS_ACCESS_KEY_ID = 'aws-access-key-id'
AWS_SECRET_ACCESS_KEY = 'secret-access-key'
AWS_STORAGE_BUCKET_NAME = 'name-of-the-bucket'
AWS_S3_REGION_NAME = 'name-of-the-region'
AWS_S3_ENDPOINT_URL = 'https://s3.amazonaws.com'
S3DIRECT_DESTINATIONS = {
    'primary_destination': {
        'key': 'uploads/',
        'allowed': ['image/jpg', 'image/jpeg', 'image/png', 'video/mp4'],
    },
}
```

Then in home.py

Then run this command
python manage.py **collectstatic** more info https://testdriven.io/blog/storing-django-static-and-media-files-on-amazon-s3/

Deploying django project on AWS EC2



Deploying in aws EC2

1.setup ec2 instance
2 install python
3 setup virtual env.
4 install django
5 git clone your project
6 install gunicorn
7 install nginx (**python don't serve static file by default** but nginx is help to serve itself **go site-enabled/django.conf and put static file locations**)
8 install -y supervisor
To connect http : 80 traffic(Custom TCP :8000) with this EC2 server need to configure security group
To connect gunicorn to nginx used wsgi.py like **gunicorn --bind : 0.0.0.0:8000 Testproject.wsgi: Application**
To continue run your application in background even you close terminal or system use **Supervisor** its keep gunicorn run always now configure **supervisor** so its read the contain from directory and restart upon gunicorn as required and also need to configure Nginx to read from gunicorn. **Nginx config file available on site-available directory inside** site -available directory create django.conf file and put aws domain.port.listen, proxy-pass .
Static file serv: go to nginx and go site-enabled/django.conf and put static file locations.
Database : create RDS db server then put all details in settings.py and
9 install pycopg2 to communicate aws RDS database

Callbacks

```
from project.tasks import (
    my_task,
    success_handler,
    error_handler,
    general_handler
)

my_task.apply_async(
    link=success_handler.si(),
    link_error=error_handler.s()
)
task_s = my_task.s()
task_s.link(general_handler.s())
task_s.link_error(general_handler.s())
task_s.apply_async()
```

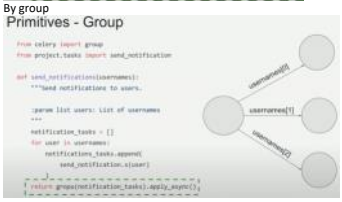
Its saying run another task if first task is finished

Primitives - Chain

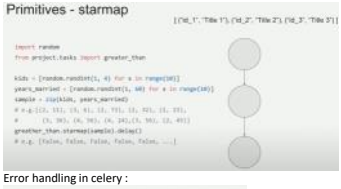
```
from celery import chain
from project.tasks import (
    save_object,
    send_notification
)

def save_and_notify(username, data):
    """Save user data and notify.

    :param str username: Username.
    :param dict data: Data to save.
    """
    return chain(
        save_object.s(username, data),
        send_notification.s()
    ).apply_async()
```

By group

Primitives - Group



Primitives - starmap



Error handling in celery :

Handling Errors

- Retry
- **Use an error Callback**
  - **link_error**
  - **on_error**

```
@shared_task(bind=True)
def notify_user(self, username, data_id):
    """Notify user.

    :param str username: Username.
    :param int data_id: Data Id.
    """
    data = user_login_data(data_id)
    try:
        notifications.notify(username, data)
    except NotificationError as exc:
        LOGGER.error(exc, exc_info=True)
        raise self.retry(exc=exc)
```

Handling Results

```
from celery.result import AsyncResult
from project.celery_app import app

result = AsyncResult(
    "5d19117a-4068-450c-b571-b9a2b68c5880",
    backend=app.backend,
    app=app
)
```

Other way

Handling Results

```
from project.tasks import check_task

result = check_task.AsyncResult("5d19117a-4068-450c-b571-b9a2b68c5880")
```

Delete all the task from queue. Using puge commands.

```
cliosoft$ celery -A celery_demo purge
```

If you want to delete specific task then use it

```
cliosoft$ celery -A celery_demo purge -Q test
```

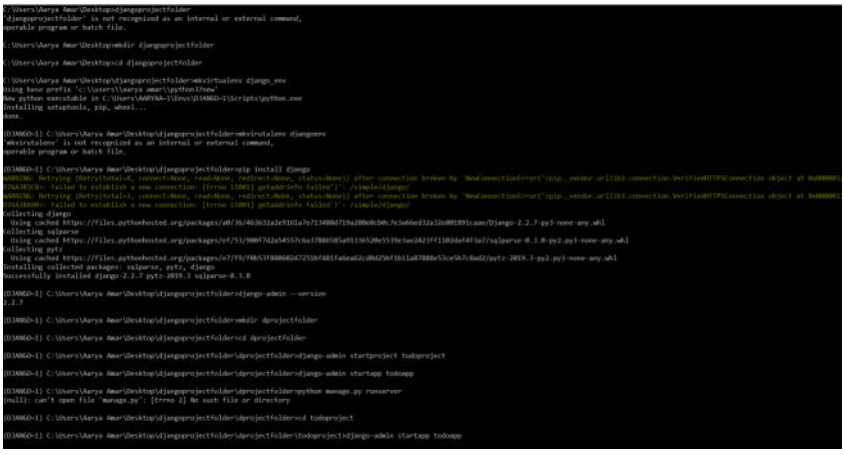
For monitor task use flower

```
cliosoft$ pip install flower
```

Start flower

```
cliosoft$ celery -A celery_demo flower
```

Django project setup
https://github.com/sumitkumar1503/schoolmanagement project




```
(DJANGO-1) C:\Users\Aarya Amar\Desktop\djangoprojectfolder\dprojectfolder\todoproject>python manage.py runserver
```

Write click on Django app folder and make templates folder . Make it inside app folder

```
class PassengerInline(admin.StackedInline):
    model = Passenger.flights.through
    extra = 1

class FlightAdmin(admin.ModelAdmin):
    inlines = [PassengerInline]

class PassengerAdmin(admin.ModelAdmin):
    filter_horizontal = ("flights",)

admin.site.register(Airport)
admin.site.register(Flight, )
admin.site.register(Passenger, PassengerAdmin)
```

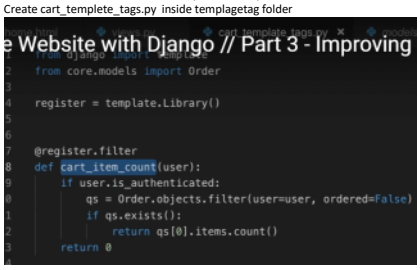
Project Setup for Ecommerce :
For authentication : **pip install django-allauth**
Or install djang-registration-redux
Then we will get automatic login page.
Import settings inside models.py for link an order with user:

```
class Order(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
                             on_delete=models.CASCADE)
```

To get free bootstarp template goto https://mdbootstrap.com/
And you can also go to Mdbootstarp/jQuery /alert for message alart
To display message install django-message framworkhttps://docs.djangoproject.com/en/3.0/ref/contrib/messages/
For modify login page goto in github then search this and clone or Download template folder and copy file to old template folder then Edit login.html also use crispy forms

pennersr / django-allauth

To add and remove item from cart top icon on top nav-bar then Create cart_templete_tags.py inside templagetag folder



For country selection we need to install packages : **pip install django-countries**

pip install nepali-phone-number
https://pypi.org/project/python-dotenv/ for environmental variable to managing key don't forget to keep .env file inside gitignore file, don't publish .env file on server
https://www.youtube.com/watch?v=XbHmiKvreLM&list=PLIrxD0HtieHhS8VzuMCfQD4uJ9yne1mE6&index=41

Displaying data from database to the main frontend pages MVT

1. create models
2. Register model in admin.py and do migrations
3. Create views inside views.py also link template .html file inside it
4. Create url of views created inside apps ,urls.py
5. Then goto project url.py and put the apps url link inside it
6. Then goto template folder and create .html file and link with base.html with it.
   We can put models name using forloop inside {{ item }} inside block content. Now you can see in page.

importing data in database using forms front page
Instead create model inside models.py we create form into foms.py all other are sampe process

1. create forms.py file then inside form create modelname+form class like (EmployeeForm) then put your model name in meta class.
2. create a formview inside views.py like Def employee_form and also link .html into form views and render it to .html and pass the form object as dictionary
3. create template .html file like employee_form.html and put form object inside using {{ form|crispy }} under {% block content %} then we can see all the form models fields into service page

Django and react app creations : https://www.youtube.com/watch?v=uZgRbnlsgrA

```
Matthews-MacBook-Pro:tutorials matthew$ ls
Matthews-MacBook-Pro:tutorials matthew$ mkdir djreact
Matthews-MacBook-Pro:tutorials matthew$ cd djreact/
Matthews-MacBook-Pro:djreact matthew$ ls
Matthews-MacBook-Pro:djreact matthew$ mkdir frontend
Matthews-MacBook-Pro:djreact matthew$ mkdir backend
Matthews-MacBook-Pro:djreact matthew$ ls
backend         frontend
Matthews-MacBook-Pro:djreact matthew$
```

Then go to backend folder and install all or use requirment.txt file

```
Matthews-MacBook-Pro:djreact matthew$ cd backend/
Matthews-MacBook-Pro:backend matthew$ ls
Matthews-MacBook-Pro:backend matthew$ virtualenv env
Using base prefix '/Library/Frameworks/Python.framework/Versions
New python executable in /Users/matthew/OneDrive/tutorials/djrea
in/python3.6
Also creating executable in /Users/matthew/OneDrive/tutorials/dj
v/bin/python
Installing setuptools, pip, wheel...done.
Matthews-MacBook-Pro:backend matthew$ source env/bin/activate
(env) Matthews-MacBook-Pro:backend matthew$ pip install django
Collecting django
  Using cached https://files.pythonhosted.org/packages/ab/15/cf
99c68b696fbb4df59e82bbccc686770f4e44c9094/Django-2.0.7-py3-none-
Collecting pytz (from django)
  Using cached https://files.pythonhosted.org/packages/30/4e/27
177a0842ed90dc789ce5d1ed740887653b898779a/pytz-2018.5-py2.py3-n
Installing collected packages: pytz, django
Successfully installed django-2.0.7 pytz-2018.5
(env) Matthews-MacBook-Pro:backend matthew$
```

Now install rest api

```
(env) Matthews-MacBook-Pro:backend matthew$ pip install djangorestframework
Collecting djangorestframework
```

Then create project

```
(env) Matthews-MacBook-Pro:backend matthew$ django-admin startproject djreact
```

https://automationpanda.com/2018/02/08/django-projects-in-visual-studio-code/ to check how we work in vs code with django
https://automationpanda.com/2017/09/14/django-projects-in-pycharm-community-edition/to check how we work in pycharm with django
Go and install all the setting from that automation panda link create vscode folder inside backend folder and inside create settings.json file then copy all the setting file
Then create django app create model did all the migration and all run server.
Go to Django restframwork and put all the setting in url.py and settings.py
https://www.django-rest-framework.org/#installation
To use default api for our frontend applications
Goto your project app and create folder api inside that create serializer.py and __init__.py now create serializer of creeated model using **genericview, APIView** like createapiview. displayapiview

Now deactivate your backend and move to frontend and create react app

```
(env) Matthews-MacBook-Pro:backend matthew$ deactivate
Matthews-MacBook-Pro:backend matthew$ ls
env     src
Matthews-MacBook-Pro:backend matthew$ cd ..
Matthews-MacBook-Pro:djreact matthew$ ls
backend         frontend
Matthews-MacBook-Pro:djreact matthew$ cd frontend/
Matthews-MacBook-Pro:frontend matthew$ ls
Matthews-MacBook-Pro:frontend matthew$ create-react-app
```

After create rest api and react app we will display rest api data into reaact app
Install axios package (to fetch data from django api)

```
^CMatthews-MacBook-Pro:gui matthew$ npm install axios --save
```

First go to frontend and inside react app folder remove the logo of react
Goto react app and ls now install bootstrap for styling
npm install react-bootstrap bootstraphttps://react-bootstrap.github.io/getting-started/introduction/ and put all the setting

Django project setup in gitbash

```
Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ python --version
Python 3.7.0

Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ python -m venv hospitalvenv

Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ ls
```

For login and logout registration step

a. Go to command prompt and type following command
   pip install django-registration-redux
b. Go to settings.py inside your app and update INSTALLED_APP
   INSTALLED_APPS = [
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
   'registration',
   'college',
   ]
c. In settings.py add following code at the end:
   ACCOUNT_ACTIVATION_DAYS=3

   EMAIL_HOST= 'smtp.gmail.com'

Inside setting.py put it in last

```
ACCOUNT_ACTIVATION_DAYS=3

EMAIL_HOST='smtp.gmail.com'
EMAIL_HOST_USER='thakurayaamar@gmail.com'
EMAIL_HOST_PASSWORD='gmail12#us'
EMAIL_PORT=587
EMAIL_USE_TLS=True
```

Inside project url.py (

```
path('accounts/',include('registration.backends.default.urls')),
```

30) Migrate
31) Update views.py add method decorator for views where you want login required:

```
from django.utils.decorators import method_decorator
from django.contrib.auth.decorators import login_required
@method_decorator(login_required, name="dispatch")
class NoticeDetailView(DetailView):
    model = Notice
```

32) Update your htmls and put condition for links based on login sta
a. For example you might need to show login link only oif user is nc logged in, and logout only if he/she has logged in
b. Go to base.html and update the code:

```
<!DOCTYPE html>
<html>
```

INSIDE BASE.HTML

```
<h1>Menu Etc</h1>
{% if user.is_authenticated %}
<a href="/accounts/logout">Logout</a>
{% else %}
<a href="/accounts/login">Login</a>
{% endif %}
{% block content %}
{% endblock %}
```

33) Go to settings.py add the following line at the end of the file:

```
LOGIN_REDIRECT_URL = "/"
```

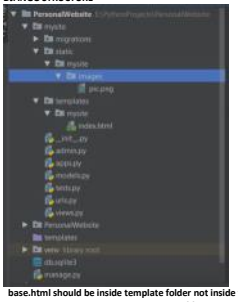
34) Run as Django and enjoy

Create signal receiver

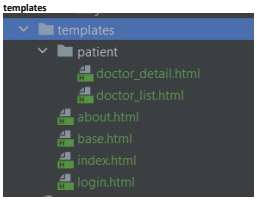
```
from django.dispatch.dispatcher import receiver
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from college.models import Profile

@receiver(post_save, sender=User)
def save_profile(sender, instance, created, **kw)
    if created:
```

```
Python 3.7.0

Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ python -m venv hospitalvenv

Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ ls
hospitalvenv/  ram.txt

Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ source hospitalvenv/scripts/activate
(hospitalvenv)
Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ pip install django
Collecting django
  Downloading https://files.pythonhosted.org/packages/87/dc/3f96196718beab06849f
94f54ff20ffec78af8424c866c61cdced30fb1ce/Django-3.0.9-py3-none-any.whl (7.5MB)
Collecting pytz (from django)
  Downloading https://files.pythonhosted.org/packages/4f/a4/879454d49688e2fad93e
59d7d4efda580b783c745fd2ec2a3adf87b0808d/pytz-2020.1-py2.py3-none-any.whl (510kB
)
Collecting sqlparse>=0.2.2 (from django)
  Downloading https://files.pythonhosted.org/packages/85/ee/6e821932f413a5c4b76b
e9c5936e313e4fc626b33f16e027866e1d60f588/sqlparse-0.3.1-py2.py3-none-any.whl (40
kB)
Collecting asgiref~=3.2 (from django)
  Downloading https://files.pythonhosted.org/packages/d5/eb/64725b25f99101030 7fd
18a9e0c1f0e6dff2f03b22fc4bcbcdb2244f60db/asgiref-3.2.10-py3-none-any.whl
Installing collected packages: pytz, sqlparse, asgiref, django
Successfully installed asgiref-3.2.10 django-3.0.9 pytz-2020.1 sqlparse-0.3.1
You are using pip version 10.0.1, however version 20.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' comm
and.
(hospitalvenv)
Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$  django-admin startproject patientengagement .
(hospitalvenv)
Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ python manage.py runserver
Watching for file changes with StatReloader
[03/Aug/2020 23:30:50] "GET / HTTP/1.1" 200 16351
[03/Aug/2020 23:30:50] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[03/Aug/2020 23:30:50] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.
1" 200 86184
[03/Aug/2020 23:30:50] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP
/1.1" 200 85876
[03/Aug/2020 23:30:50] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1
.1" 200 85692
Not Found: /favicon.ico
[03/Aug/2020 23:30:50] "GET /favicon.ico HTTP/1.1" 404 1983
```

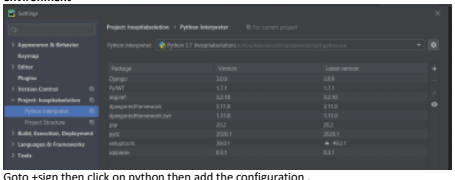
DJANGO STRUCTURE



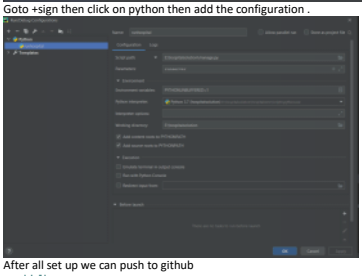
base.html should be inside template folder not inside the app name folder
doctor _list or doctor _detail.html kind of file should in inside app name folder under templates



Go Project setting click on **setting sign** then click on **add** and put in **existing environment**



Goto +sign then click on python then add the configuration .



After all set up we can push to github

```
# add files
(venv)$ git add .
# commit files
(venv)$ git commit -m "configured DB settings"
```

<https://ultimatedjango.com/learn-django/lessons/configure-the-database/?trim=yes>

After close gitbash start project again

```
Aarya Amar@DESKTOP-QP2S89T MINGW64 ~ (master)
$ cd E:\hospitalsolution

Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ source/hospitalvenv/scripts/activate
bash: source/hospitalvenv/scripts/activate: No such file or directory

Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ source hospitalvenv/scripts/activate
(hospitalvenv)
Aarya Amar@DESKTOP-QP2S89T MINGW64 /e/hospitalsolution (master)
$ dir
db.sqlite3  hospitalvenv  manage.py  patient  patientengagement
```

```python
from college.models import Profile

@receiver(post_save, sender=User)
def save_profile(sender, instance, created, **kw)
    if created:
        Profile.objects.create(user=instance)
```

If you want run code only one time after server start then goto init.py

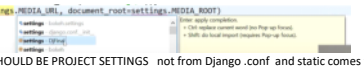
```
default_app_config = 'college.apps.CollegeConfig'
```

Then go to apps.py and if you right any code inside ready() method it will run only once

```
from django.apps import AppConfig


class CollegeConfig(AppConfig):
    name = 'college'
    def ready(self):
        import college.mysignal
```

IMAGE KO

```
+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```



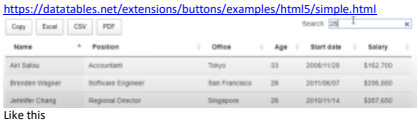
SETTINGS SHOULD BE PROJECT SETTINGS not from Django .conf and static comes from Django while importing

To show google maps on your given location then type in every address word use +

```
href="http://maps.google.com/maps?q=Indrapuri+Bhopal">Indrapuri Bhopal</a></p>
```

If you want to show page and serach data in table then use it and put all link before </head> (use script tag in all link) inside base.html so it work for all the page

https://datatables.net/extensions/buttons/examples/html5/simple.html



Like this

```
<!--datatable cdn start-->
<script src="https://code.jquery.com/jquery-3.3.1.js"></script>
<script src="https://cdn.datatables.net/1.10.20/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/buttons/1.6.1/js/dataTables.buttons.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jszip/3.1.3/jszip.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.53/pdfmake.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.53/vfs_fonts.js"></script>
<script src="https://cdn.datatables.net/buttons/ html5.min.js"></script>
<!--datatable cdn end-->
</head>
```

https://www.youtube.com/watch?v=aFC9TmLCbSs

Need to put <link in CSS code

```
<link rel="stylesheet" href="https://cdn.datatables.net/1.10.20/css/jquery.dataTables.min.css">
<link rel="stylesheet" href="https://cdn.datatables.net/buttons/1.6.1/css/buttons.dataTables.min
```

```
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD ..." to unstage)

        renamed:    petient/__init__.py -> patient/__init__.py
        renamed:    petient/admin.py -> patient/admin.py
        new file:   patient/apps.py
        renamed:    petient/migrations/__init__.py -> patient/migrations/__init_
_.py
        renamed:    petient/models.py -> patient/models.py
        renamed:    petient/tests.py -> patient/tests.py
        renamed:    petient/views.py -> patient/views.py
        deleted:    petient/apps.py

Changes not staged for commit:
  (use "git add ..." to update what will be committed)
  (use "git checkout -- ..." to discard changes in working directory)

        modified:   .idea/misc.xml
        modified:   patientengagement/__pycache__/settings.cpython-37.pyc
        modified:   patientengagement/settings.py

Untracked files:
  (use "git add ..." to include in what will be committed)

        patient/__pycache__/
        patient/migrations/__pycache__/

(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git init
Reinitialized existing Git repository in E:/hospitalsolution/.git/
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD ..." to unstage)

        renamed:    petient/__init__.py -> patient/__init__.py
        renamed:    petient/admin.py -> patient/admin.py
        new file:   patient/apps.py
        renamed:    petient/migrations/__init__.py -> patient/migrations/__init_
_.py
        renamed:    petient/models.py -> patient/models.py
        renamed:    petient/tests.py -> patient/tests.py
        renamed:    petient/views.py -> patient/views.py
        deleted:    petient/apps.py

Changes not staged for commit:
  (use "git add ..." to update what will be committed)
  (use "git checkout -- ..." to discard changes in working directory)

        modified:   .idea/misc.xml
        modified:   patientengagement/__pycache__/settings.cpython-37.pyc
        modified:   patientengagement/settings.py

Untracked files:
  (use "git add ..." to include in what will be committed)

        patient/__pycache__/
        patient/migrations/__pycache__/

(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git add .
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git commit -m "install django and rest "
$ git remote -v

Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git push -u origin master
Everything up-to-date
Branch 'master' set up to track remote branch 'master' from 'origin'.
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
Now you can check all the changes will appear in git hub  repository
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git pull origin master
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/thakuramar/hospitalsolution
 * branch            master     -> FETCH_HEAD
   a196ced..4684c11  master     -> origin/master
Updating a196ced..4684c11
Fast-forward
 patient/views.py | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
(hospitalvenv)
After pull project now change in your pycharm then push it  to git hub
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add ..." to update what will be committed)
  (use "git checkout -- ..." to discard changes in working directory)

        modified:   patient/views.py

no changes added to commit (use "git add" and/or "git commit -a")
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git add .
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git commit -m "check"
[master 4773737] check
 1 file changed, 1 insertion(+), 1 deletion(-)
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git push origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 368 bytes | 368.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/thakuramar/hospitalsolution.git
   4684c11..4773737  master -> master
(hospitalvenv)
Aarya Amar@DESKTOP-QP2589T MINGW64 /e/hospitalsolution (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
(hospitalvenv)
Aarya Amar@D
```