Step for Django Projects

1. Create Virtual Environment and Activate it
   1. sudo easy\_install pip
   2. sudo pip install virtualenv
   3. source virtualenv/bin/activate
2. Install Django in Virtual Environment
   1. Pip install Django==version
3. Create Django project outside the Virtual Environment
   1. Django-admin startproject projectname
4. Create Database for Project
   1. Python manage.py migrate
5. Create superuser for project
   1. Python manage.py createsupersuer
6. Run the Local Server and Check the Project using browser
   1. Python manage.py runserver
7. Create Application in Django Project and activating in project
   1. Python manage.py startapp app\_name
8. Create Model / Table for Application
   1. Create class for each model in model.py file
9. Registering model in database
   1. Run makemigration and migrate command
10. Registering the model in Admin interface
    1. Enter admin.site.register(model\_name)
11. Add some demo data in Models
12. Working with Django Shell
    1. Python manage.py shell
       1. To get all values from table (topic = Topic.objects.all())
       2. To get a specific record from table (t = Topic.objects.get(id=1)
       3. To get related record in child table for record ( t.entry\_set.all())
13. Creating URL , VIEW AND HTML Home page,
    1. Create a URL in application urls.py file
    2. Create a view in application views.py file
    3. Create a .html file in template folder of application
14. Creating base template and Inherit in other pages, apply to index.html
15. Creating Additional Pages
16. Creating List page for all topics
17. Creating detail page for individual topic and show related entries
18. Creating to add topic
19. Create a page to show all the entries related to specific topic
20. Creating pages to allow users to enter data in site
21. Creating ModelForm in application for each model
22. Create page to add topic
23. Create page to add entry against individual topic
24. Create page to allow editing entry
25. Create user login and logoff functionality using Django built-in functionality
26. Create user Registration page
    1. Create url for registration
    2. Create view for registration
    3. Create table for registration
27. Associate topic to specific user and not allow any change without login
28. Not edit allow to access direct link
29. Using Bootstrap to style the Website

Updating pip manually

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

python get-pip.py

EPL Programming language

ZPL BOOKSOLVE NOT SUPPORTING

Basics

1. Create virtual environment with name myvirtaulenv/foodtasker & activate
2. Install Django 1.10 in virtual environment
3. Create Django project with name foodtasker
4. Add project to git and code editor like pycharm
5. Create .gitignore file as well
6. Create application with name foodtaskerapp in project
7. Create Home page for application
8. Create Django dashboard and super user
9. Add bootstrap 3.3.7 to project

Authentication (Sign-in / Sign-out / Registration)

**Foodtasker Project step by step**

**Development Environment Setup**

1. Install Python 3.5.2
2. Create Heroku account ([rizwan.wakil2020@gmail.com](mailto:rizwan.wakil2020@gmail.com) & M@g1t0112)

Download Heroku command line and install on system 🡪 go to <https://devcenter.heroku.com/articles/heroku-command-line> --> download and install

1. Choose any code editor, I am using PyCharm
2. To test restful API, install POSTMAN chrome extension
3. Install XCode version 8 on Mac for developing IOS App

**Basic Project Setup in Django**

Create virtual environment with name myvirtualenv/foodtasker on desktop

Install Django==1.10 in virtual environment

Create foodtasker project inside the virtual environment

Run the server to make sure, project is working

Initialize git

Add project to code editor (PyCharm)

Create .gitignore ( \_\_pyachhe \_\_/ , db.sqlite3, \*.pyc )

Add all file to git and commit “init git”

**Basic App and Home page Setup in Django**

1. Create new application with name foodtaskerapp

Add foodtaskerapp in settings.py

Create home page

Create view

def home(request):

return render(request, ‘home.html’, {})

Create URL in urls.py file in foodtasker

url(r’^$, home,name=’home’)

Create Template & home.html file

create templates folder in foodtaskerapp and create home.html file in it print message hello python

**Basic Admin site Setup in Django**

1. create djnago daskborad and create super user

**Basic Bootstrap Setup in Django**

1. adding bootstarp to project, download bootstrap 3.3.7 and create a new folder static foodtaskerapp and create css, js , font and img folder and copy file from bootstrap to project

add following code in home.html

at top of page add below line

{%load static%}

Add link for css and js file and our one css file

**Basic Login / Logoff Functionality in Django**

1. creating authentication function for restaurant

create new url in urls.py in foodtasker

url(r’^restaurant/sign-in/$, views.login, {‘template\_name’, ‘restaurant/sign\_in.html’ ,name=’restaurant-sign-in’)

url(r’^restaurant/sign-out/$, views.logout, {‘next\_page’, ‘/’ ,name=’restaurant-sign-out’)

create restaurant folder in templates and then create sign\_in.html file and include the sign-in form

add LOGIN\_REDIRECT\_URL in settings.py

create another file with name home.html inside restaurant folder and add link for logout functionality

Add another url url(r’^restaurant/$, restaurant\_home,name=’restaurant-home’)

Create new function restaurant\_home and point to restaurant\_home.html and redirect the previous home view.

Add decorator to redirect to login page

@login\_reguired(login\_url=’/restaurant/sign-in/’)

1. registering restaurant owner and restaurant

owner will be used in Django User table but for restaurant we will create model

create model with name Restaurant and with fields (user, name, address, phone, logo)

install pillow 3.3.0

make migration

create new url for sign up

url(r’^restaurant/sign-up/$, views.restaurant\_sign\_up, ,name=’restaurant-sign-up’)

create restaurant\_sign\_up function

add following code in restaurant\_sign\_up view

create sign-up.html file in template

1. create forms.py file in foodtaskerapp and add following code

class Userform(forms.modelform):

email = forms.Charfiled (max\_length=100, required=True)

password = froms.charfield(widget=forms.input())

model = User

fields = (‘username’, ‘password’, ‘fist\_name’, ‘last\_name’, ‘email’)

class Restaurentform(forms.modelform):

model = Restaurant

fields = (‘name’, ‘address’, ‘phone’, ‘,logo’)

add following code in restaurant\_sign\_up view

show blank user and restaurant form and if submit data then register the user

**Basic Static file setup in Django**

add MEDIA\_ROOT = os.path.join(BASE\_DIR, ‘media’)

AND MEDIA\_URL =’/media/’

to settings.py file

add following line to urls.py file in foodtasker

+ static(setting.MEDIA\_URL, docment\_root=settings.MEDIA\_ROOT)

Check in Django dashboard

add restaurant model to dashboard

**HEROKU SETUP**

1. Setting for Heroku

From virtualenv type 🡪 Heroku login

Type email address and password

Type 🡪 heroku create

Create new file in project with name runtime.txt at manage.py level and type python-3.5.2 inside it

Install gunicorn using pip install gunicorn==19.6.0

Create requirements.txt and add install apps

Django==1.10

Gunicorn==19.6.0

Pillow==3.3.0

Create new file Procfile without extension 🡪 add following code

web: gunicorn foodtasker.wsgi --log-file –

1. Static resource on Heroku setup

Install whitenoise using pip install whitenoise==3.2.1 and add to requirements.txt

Add STATIC\_ROOT to settings.py (os.path.join(BASE\_DIR, ‘staticfiles’)

Open wsgi.py file in foodtasker and add following lines

From whitenoise.django import DjangoWhiteNoice

Application = DjangoWhiteNoise(application)

1. Configuring database on Heroku

Install using pip install dj-database-url==0.4.1 and copy into requirements.txt

Add psycopg2==2.6.2 in requirements.txt

Add following lines to settings.py file

Import dj\_database\_url

db\_from\_env = dj\_database\_url.config()

DATABASES[‘default’].update(db\_from\_env)

Add all change in git and commit and

Pushing project to Heroku from local system

git push heroku master

Heroku run python manage.py migrate command to create table

Heroku run python manage.py createsuperuser

Heroku open

1. Authentication function with facebook

Open developers.facebook.com and login then click new app

Select website 🡪 give name Foodtasker and create app id 🡪

Copy App ID and App Secret code

Go to setting in facebook and click add platform 🡪 <http://localhost:8000> and save

1. Install Django rest framework social oauth2 using pip install Django-rest-framework-social-oauth2==1.0.4 and add to requirements.txt

Go to link <https://github.com/PhilipGarnero/dango-rest-framework-soical-oauth2> and follow the procedure

Add url wih name api/social/

Make migration and login to dashboard and create application

1. Testing and create new user from facebook

Follow 🡪 user signup from mobile and will be validated from facebook after validation facebook will send token to our mobile app 🡪 mobile app will send token to restframework 🡪 rest framework will send back access token to mobile app

We will use postman in place of mobile app

Open postman and enter the following url for sigin and signup

http://localhost:8000/api/social/convert-token

Then click on params 🡪

Key value

grant\_type convert\_token

client\_id copy from dashboard

client\_secert copy from dashboard

backend facebook

token get from github link to facebook account

Open postman and enter the following url for sign-out

http://localhost:8000/api/social/revoke-token

Then click on params 🡪

Key value

client\_id copy from dashboard

client\_secert copy from dashboard

token copy from dashboard

1. Create model for customer and driver same like restaurant

Customer / Driver same field name for both models

user, avatar phone, address

add model to db

add PIPELINE from Django-restframework

create new file with name social\_auth\_pipeline and add following code

from foodtaskerapp.models import driver , customer

def create\_user\_by\_type(backend,user,request, response,\*args, \*\*kwargs):

if backend.name == ‘facebook’:

avatar = ‘https://graph.facebook.com/%s/picture?type=large’ % response[‘id’]

if request[‘user\_type’] == ‘driver’ and not Driver.objects.filter(user\_id=user.id):

Driver.objects.create(user\_id=user.id, avatar=avatar)

Elif not Customer.objects.filter(user\_id=user.id)

Customer.objects.create(user\_id=user.id, avatar=avatar)

mention this function In pipeline like below

foodtaskerapplsocial\_auth\_pipeline.create\_user\_by\_type

now test using api

1. Creattion of Structure for Dashboard
2. Create following URLs
3. url(r’^restaurant/account/$’, views.restaurant\_account, name=’restaurant-account’)
4. url(r’^restaurant/meal/$’, views.restaurant\_meal, name=’restaurant-meal)
5. url(r’^restaurant/order/$’, views.restaurant\_order, name=’restaurant-order)
6. url(r’^restaurant/report/$’, views.restaurant\_report, name=’restaurant-report)