**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 Application Creation and Home page Setup in Django**

1. Create new application with name foodtaskerapp

Add foodtaskerapp in settings.py

Create home page

View

Create view

def home(request):

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

URL

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 print message hello python

**Basic Admin site Setup in Django**

1. Create basic migration and create Django dashboard and create super user

**Basic Bootstrap Setup in Django**

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

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’)

from django.contrib.auth import view

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

url(r’^restaurant/sign-out/$, views.Logout.as\_view(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 or specify the input take in form with hidden tag and redirect to home page

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 views and delete home.html outside the restaurant folder.

To secure home page that only authenticated user access,

Add decorator to redirect to login page

From Django.contrib.auth.decorator import login\_required

@login\_required(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’, ‘first\_name’, ‘last\_name’, ‘email’)

class RestaurantForm(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 Media 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 AND DEPLOYING DJANGO APPLICATION ON HEROKUP**

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-3.6.10 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==2.8.4 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

**Facebook Authentication in Django**

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 (2024968834489583) and App Secret code (0c548ff28af8368f229447dbf893ca10)

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

1. First install social-auth-app-django using command pip install social-auth-app-django

Then Install Django rest framework social oauth2 using pip install django-rest-framework-social-oauth2 and add to both app to requirements.txt

Or follow this Go to link <https://github.com/PhilipGarnero/dango-rest-framework-soical-oauth2>

Add these app to installed apps

'oauth2\_provider',

'social\_django',

'rest\_framework\_social\_oauth2',

Add url to urls.py file

(r'^api/soical/', include('rest\_framework\_social\_oauth2.urls')),

Add below line to middleware\_classes in settings.py

**'social\_django.middleware.SocialAuthExceptionMiddleware',**

**Add below lines to TEMPLATES in settings.py**

'social\_django.context\_processors.backends',

'social\_django.context\_processors.login\_redirect',

Add below REST\_FRAMEWORK at end of settings.py file

REST\_FRAMEWORK = {  
 'DEFAULT\_AUTHENTICATION\_CLASSES': (  
 'oauth2\_provider.contrib.rest\_framework.OAuth2Authentication',  
 'rest\_framework\_social\_oauth2.authentication.SocialAuthentication',  
 ),  
}

Add below line to Authentication Backend

AUTHENTICATION\_BACKENDS = (

'social\_core.backends.facebook.FacebookOAuth2',

'social\_core.backends.facebook.FacebookAppOAuth2',  
 'rest\_framework\_social\_oauth2.backends.DjangoOAuth2',  
 'django.contrib.auth.backends.ModelBackend',  
)

Add below lines to settings.py at the end

# Facebook configuration  
SOCIAL\_AUTH\_FACEBOOK\_KEY = '2024968834489583'  
SOCIAL\_AUTH\_FACEBOOK\_SECRET = '0c548ff28af8368f229447dbf893ca10'  
  
  
# Define SOCIAL\_AUTH\_FACEBOOK\_SCOPE to get extra permissions from facebook. Email is not sent by default, to get it,  
# you must request the email permission:  
SOCIAL\_AUTH\_FACEBOOK\_SCOPE = ['email']  
SOCIAL\_AUTH\_FACEBOOK\_PROFILE\_EXTRA\_PARAMS = {  
 'fields': 'id, name, email'  
}

// convert-token (sign-in / sign-up)

//revoke-token(sign-out)

Now open the admin site by 127.0.0.1:8000/admin and click application 🡪 click add 🡪 select user admin 🡪 Client type (confidential) 🡪 authorization type (resource owner password based) 🡪 name (any name)

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 by going to link

<https://developers.facebook.com/tools/accesstoken/>

if email address is not showing in user record 🡪 go to

<https://developers.facebook.com/tools/explorer>

then select app foodtaskerapp 🡪 click gettoken 🡪 select user token 🡪 select email 🡪

copy the access token and try again and check the user info

click on post, make sure method is posted in postman

Follow below process for sign out

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 this is the access token created again user login in dash

this will remove the token form access token in 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 below link

<https://python-social-auth.readthedocs.io/en/latest/pipeline.html>

copy below code and paste in settings.py file

SOCIAL\_AUTH\_PIPELINE = (  
 'social\_core.pipeline.social\_auth.social\_details',  
 'social\_core.pipeline.social\_auth.social\_uid',  
 'social\_core.pipeline.social\_auth.auth\_allowed',  
 'social\_core.pipeline.social\_auth.social\_user',  
 'social\_core.pipeline.user.get\_username',  
 'social\_core.pipeline.user.create\_user',

‘foodtaskerapp.social\_auth\_pipeline.create\_user\_by\_type,  
 'social\_core.pipeline.social\_auth.associate\_user',  
 'social\_core.pipeline.social\_auth.load\_extra\_data',  
 'social\_core.pipeline.user.user\_details',  
)

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)

Now Testing open postman and enter following information

It’s same like user user login, only add user\_type field with value either driver or customer

1. General Website Structure

Create few urls in urls.py file in foodtaskerapp

path(‘restaurant/account/’, views.restaurant\_account, name=’restaurant-account’),

path(‘restaurant/meal/’, views.restaurant\_meal, name=’restaurant-meal),

path(‘restaurant/order/’, views.restaurant\_order, name=’restaurant-order),

path(‘restaurant/report/’, views.restaurant\_report, name=’restaurant-report),

create restaurant\_account, restaurant\_meal, restaurant\_order & restaurant\_report views in views.py file in foodtaskerapp

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

def restaurant\_account(request):

return render(request, ‘restaurant/account.html’, {})

create other view same like above

create .html file for each views as well based on index.html page

1. Advance design for restaurant

Creating Base File

Create base.html file user templates directory

Copy home.html to base.html and replace title content with {% block title %}

{% endblock %}

And in body only enter {%block page %}{% endblock %}

Now open account.html page and remove everything and enter following code

{% extend ‘base.html’ %}

{%block title %} account page {%endblock%}

{%block sidebar %} some text goes {%endblock %}

{%block page %} {%endblock%}

Same code copy to meal, order and report page as well

Now rename the home.html file to name base.html in restaurant folder and add following code

{% extend ‘base.html’ %}

{%block title %} Restaurant {%endblock%}

{%block sidebar %}

Restaurant, Hi, {{request.user}}

<br>

<a href=”{% url ‘restaurant-account’ %}”>Account </a>

<a href=”{% url ‘restaurant-meal %}”>Meal </a>

<a href=”{% url ‘restaurant-order %}”>Order </a>

<a href=”{% url ‘restaurant-report %}”>Report </a>

{%endblock %}

Now open account.html file and change the base.html to restaurant/base.html

Repeate this process to all order, meal and report pages as well.

Check all pages

1. Advance design for signup and sign-in pages

Now create another base\_signup.html file in templates directory and copy the main base.html content to it and change block to heading and content

We create another base\_signup.html file because in these base there is no sidebar

Now refactor the sing\_in.html and sign\_up.html as per base\_signin.html

Open sing-in.html file and enter following code

Extend ‘base\_signup.html

{%block heading%} Restaurant – Sign In {%endblock %}

{%block content%} paste form code here {%endblock}%

Repeat same process with sign-up.html as well

1. Adding Bootstrap to Project

Open style.css file and add following code

.bg-blue {

Background-color: #3F3F63;

}

.btn-pink {

Color: white;

Background-color: #EE5462;

}

.btn-pink:hover,

.btn-pink:focus {

Color: white;

Backgoundcolder: #DA4F5D;

}

Open base\_sign and following code in body area

<body class=”bg-blue”>

<div class=”container”>

<div class=”row”>

<div class=”col-lg-4 col-lg-offset-4”>

<div class=”panel panel-body”>

<div class=”panel-body”>

<h3 class=”txt-center uppercate”>{%block heading%}</h1>

<br>

{%block content%}{%endblock%}

</div>

</div>

</div>

</div>

</div>

1. Adding Bootstrap3 Form to Django project

Install bootstrap using pip install Django-bootstrap3==7.0.1

Add the bootsrap3 in settings.py

{%load bootstrap3 %}

Now change the {{form}} with {%boostrap\_form form%} in sign\_in.html and sign\_up.html file

1. Bootstrap for Dashboard

27.

28. Adding Account Page

29. Adding Meal Page

Create a new model for Meal with following field

Restaurant , name , short\_description, image (meal\_images), price

Make migration and migrate and add to dashboard as well

Create url for url(‘restaurant/meal/add/$’, view.restaurant\_add\_meal, name=’restaturant-add-meal”),

Create view restaurant\_add\_meal(request),

Return render(request, restaurant/meal\_add\_meal.html,{})

Create meal\_add\_meal.html file and create a link from meal.html file

Now create form for meal model and add to meal\_add.html page

Create functionality to add meal information to database

31. Creating list of all meals for a restaurant

Enter following code in meal view

Meals = Meal.objects.filter(restaurant=request.user.restaurant).order\_by(“-id”)

return render(request, template, context)

32. Adding meal functionality

Add a url in urls.py file

path('restaurant/meal/edit/', views.restaurant\_edit\_meal, name='restaurant-edit-meal'),

create view and .html file accordingly

34. now create model for order and order detail with following information

Status\_choic = ((COOKING, ‘cooking’),(),)

Customer, restaurant, driver , address, total, status, created\_at , picked\_at

Order\_etals model

Order, meal, sub\_total, quantity

Enter one dummy record using dashboard

35. display all the order on order page and all create show detail of order as well

Make is same like meal page

37. Working with Django Rest Framework

Install Django rest framework using pip install djangoframwork==3.4.3

Create serializers.py and convert Restaurant to serializer

Field (id,name,phone.address, logo)

Create api.py file

Create customer\_get\_restaiurants() function to return all restauents

Create a url

Api/customer/restaurant/, apis.custmer\_get\_restaturants()