1. Load javascript files at the end of the body of the webpage so that the page gets loaded even before javascript files are processed otherwise user will have to wait till JS is loaded.
2. Responsive design has 2 approaches

Traditional -> Large screen first

Mobile first design (More popular – First design for small screen)

1. In order to overwrite bootstrap files, simply create your own CSS files and include them in the webpage **after** the bootstrap files. If bootstrap files are included after our CSS files then bootstrap files will overwrite out CSS.
2. DIRS key in settings.py tells Django to look into the listed destinations for templates. The base.html is stored using this functionality in templates folder.
3. Images should be in jpg format. Photoshop has a “save for web” option which compresses image to lowest file size possible maintaining quality. Use alt tag in img for SEO. Good page titles which are different for all pages, meta descriptions, dynamic xml sitemap is created and submitted to search engines.

<meta name="description" content="By Paul Boag Choosing a content management system can be tricky.ook" />

1. W3c linkchecker
2. Implementing post with Ajax in Django needs csrf to be implemented in a special way. Add the below lines of jquery in the js files.

$.ajaxSetup({

beforeSend: function(xhr, settings) {

function getCookie(name) {

var cookieValue = null;

if (document.cookie && document.cookie != '') {

var cookies = document.cookie.split(';');

for (var i = 0; i < cookies.length; i++) {

var cookie = jQuery.trim(cookies[i]);

// Does this cookie string begin with the name we want?

if (cookie.substring(0, name.length + 1) == (name + '=')) {

cookieValue = decodeURIComponent(cookie.substring(name.length + 1));

break;

}

}

}

return cookieValue;

}

if (!(/^http:.\*/.test(settings.url) || /^https:.\*/.test(settings.url))) {

// Only send the token to relative URLs i.e. locally.

xhr.setRequestHeader("X-CSRFToken", getCookie('csrftoken'));

}

}

});

1. For email system we use celery because it will manage tasks asynchronously so that user does not have to wait TILL THE MAIL is sent by the system. This will help to keep website load faster even when it has to send a mail.

For celery we need a worker, we are using rabbitmq which does not store the results permanently. For permanent result storage we could use redis. For rabbitmq, we need erlang. So, erlang -> rabbitmq . Don’t forget to set add C:\Program Files\RabbitMQ Server\rabbitmq\_server-3.6.10\sbin to the Environment variable PATH. To install Django celery, use pip install Django-celery in cmd. Then follow instructions of <https://www.youtube.com/watch?v=lZQtMPMeT8c>

Our app will generate tasks and push them into message queue (Rabbitmq), Celery listens to the queue and executes tasks.

1. Celery 4.0 is not compatible for windows. It gives a windows error 87 parameter is incorrect.
2. Celery 4.0 supports Django 1.8 and newer versions. Please use Celery 3.1 for versions older than Django 1.8.
3. But my Django version is 1.11.1 and celery 4.0 is not supported with windows !! Still, using celery version 3.1.11 with Django 1.11.1 works fine. Documentation of celery 3.x and 4.x are very different. So, follow doc at <http://www.pythondoc.com/celery-3.1.11/getting-started/next-steps.html>
4. To run celery command used is -> celery worker –app=cashback –l info

In another terminal open python manage.py runserver

1. Cannot import name \_uuid\_generate\_random error in celery, this is solved by upgrading to kombu version 3.0.30
2. AppRegistryNotReady Models/Apps aren’t loaded yet error. This happens when importing an application configuration or a models module triggers code that depends on the app registry. Solution is to import Django in celery.py and execute Django.setup().
3. Set up Email variables in settings.py and in accounts security settings enable access to less secure apps of the senders mail to use the send\_mail() in Django.

**Proj/proj/Init.py**

from \_\_future\_\_ import absolute\_import, unicode\_literals

# This will make sure the app is always imported when

# Django starts so that shared\_task will use this app.

from .celery\_tasks import app as celery\_app

\_\_all\_\_ = ['celery\_app']

**Proj/proj/Celery\_tasks.py**

from \_\_future\_\_ import absolute\_import

from celery import Celery

import os

import django

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE','cashback.settings')

django.setup()

from django.conf import settings

app = Celery('cashback',

broker='amqp://localhost//',

include=['cashback.tasks'])

app.config\_from\_object('django.conf:settings')

# Optional configuration, see the application user guide.

app.conf.update(

CELERY\_TASK\_RESULT\_EXPIRES=3600,

)

if \_\_name\_\_ == '\_\_main\_\_':

app.start()

**proj/proj/settings.py**

CELERY\_BROKER\_URL = 'amqp://localhost//'

#Broker settings

BROKER\_URL = "localhost"

BROKER\_USER = ""

BROKER\_PASSWORD = ""

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_HOST\_USER = 'madhuragarwal499@gmail.com'

EMAIL\_HOST\_PASSWORD = 'password'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

**Proj/proj/Tasks.py**

from \_\_future\_\_ import absolute\_import

from django.core.mail import send\_mail

from cashback.celery\_tasks import app

from django.template.loader import render\_to\_string

from django.utils.html import strip\_tags

from django.core.mail import EmailMultiAlternatives

from django.conf import settings

@app.task

def add(x, y):

return x + y

@app.task

def SignupTask( email):

html\_content = render\_to\_string('base.html')

text\_content = strip\_tags(html\_content)

send\_mail('Welcome', text\_content, settings.EMAIL\_HOST\_USER,[settings.EMAIL\_HOST\_USER, email], fail\_silently=False)

**user\_login/views.py**

(using the tasks in views)

from cashback.tasks import SignupTask,mul

SignupTask.delay(user.email)

mul.delay(2,5)

1. The requirements.txt for this celery working module on 28 may :

amqp==1.4.5

anyjson==0.3.3

billiard==3.3.0.17

celery==3.1.13

dj-database-url==0.4.1

Django==1.11.1

django-celery==3.2.1

gunicorn==19.6.0

kombu==3.0.30

librabbitmq==1.6.1

olefile==0.44

Pillow==4.1.1

psycopg2==2.6.2

pytz==2017.2

requests==2.9.1

vine==1.1.3

virtualenv==15.1.0

whitenoise==2.0.6