Django Cheat Sheet

Creating a Django Project

1. Open terminal and create a new project folder. Navigate to new folder location
2. virtualenv --python=python3.6 venv # to create new virtual environment
3. source venv/bin/activate
4. pip3 install django
5. pip3 install pillow # if you are using images in your project
6. django-admin startproject [project name]
7. navigate to new project folder
8. python3 manage.py runserver # to check install worked
9. CTRL+C, then python3 manage.py migrate # to apply migrations and remove error messages
10. python3 manage.py createsuperuser [username] # create username and super long password
11. python3 manage.py runserver # to test installation
12. Open project urls.py and edit admin url:
    url(r'^convergio/', admin.site.urls),
13. Remember this url to access the admin interface. Don't use default admin url to make your admin view more secure

Connecting PostgreSQL to a Django Project

1. Create a new database in PostgreSQL, and assign a DB user with admin privileges
2. Go to project settings.py and scroll down to DATABASES section. Replace with:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': '[databasename]',
        'USER': '[username]',
        'PASSWORD': '[password]',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

3. Scroll down to TIMEZONE and replace code. e.g. TIME_ZONE = 'Australia/Perth'

Creating a Django App

1. Open terminal and navigate to django project folder (where manage.py exists)
2. django-admin startapp [app name] # use pluralised names, e.g. posts
3. Navigate to new app folder.
4. Create new subfolder called “templates”
5. Navigate to new templates folder.
6. Create new subfolder called same name as your app (e.g. posts)
7. Go back to project folder
8. Open project urls.py and add “from [appname] import views”. Then add url path for new view. e.g. url(r’^$’, views.home, name=‘home’)
9. Open views.py and add:

```
def home(request):
        return render(request, ‘posts/home.html’)
```

10. Navigate to projectfolder\appfolder\templates\appfolder
11. Create new file ‘home.html’
12. Go to project folder and open settings.py. Scroll down to INSTALLED_APPs and add ‘appname’, to the end of the list of apps.

Creating Models for Django

1. Go to app folder and open models.py
2. Create a new model by defining a class. Define data elements of the model by defining field objects. Also capitalise class name. e.g.

```
class Post(models.Model):
        title = models.CharField(max_length=200)
        pub_date = models.DateTimeField()
        image = models.ImageField(upload_to=‘media/’)     # have to create media folder in app
        body = models.TextField()

        def __str__(self):
                return (self.title)
```

3. Go to terminal and stop server (if running)
4. python3 manage.py makemigrations
5. python3 manage.py migrate
6. Go to app folder and open admin.py. Add “from .models import Modelname”.
7. Net add “admin.site.register(Modelname)”
8. Run django server and go to admin page
9. Model should be visible in admin interface

Creating Static Images to Django Project

1. Go to app folder and create subfolder called “static”
2. Navigate to static folder and create anew subfolder with same name as app
3. In html file access static image using:

```
{% load static %}
<img src=“{% static ‘[appname]/[imagefilename]’ %}” />
```

Forcing POST Requests

1. In your app\views.py function definitions, add an extra IF statement:

```
def upvote(request, pk):
        if request.method == ‘POST’:                    # new if statement
                post.votes_total += 1
                post.save()
                return redirect(‘home’)
```

Extending Templates in a Django Project

1. At project root folder, create a new templates folder
2. Inside new folder, create a new html file “base.html”
3. Add common html/boostrap/JS code in base.html file
4. Open project\settings.py and navigate to TEMPLATES section. In ‘DIRS’ field modify to:
   ‘DIRS’: [‘templates’],
5. Within base.html insert code:

```
<header stuff>

{% block content %}
{% endblock %}

<footer stuff>
```

6. Open any app\template html file and add:

```
{% extends ‘base.html’ %}

<app template html code>

{% endblock %}
```

User Authentication in a Django Project

1. In app\urls.py add “url(r’^signup/, views.signup, name=‘signup’),” and “url(r’^login/, views.loginview, name=‘login’),” and “url(r’^logout/, views.logoutview, name=‘logout’),”
2. In app\views.py, add “from django.contrib.auth.models import User” and “from django.contrib.auth import authenticate, login, logout”
3. Add these functions:

```
# sign up function
def signup(request):
    if request.method == 'POST':
        if request.POST['password1'] == request.POST['password2']:
            try:
                user = User.objects.get(username=request.POST['username'])
                return render(request, 'accounts/signup.html', {'error':'Username not available!'})
            except User.DoesNotExist:
                user = User.objects.create_user(request.POST['username'],
password=request.POST['password1'])
                login(request, user)
                return render(request, 'accounts/signup.html')
        else:
            return render(request, 'accounts/signup.html', {'error':'Passwords didn\'t match'})
    else:
        return render(request, 'accounts/signup.html')

# login function
def loginview(request):
    if request.method == 'POST':
        user = authenticate(username=request.POST['username'],
password=request.POST['password'])
        if user is not None:
            login(request, user)
            if 'next' in request.POST:
                return redirect(request.POST['next'])
            return redirect('home')
        else:
            return render(request, 'accounts/login.html', {'error':'Invalid username or password.'})
    else:
        return render(request, 'accounts/login.html')

#logout function
def logoutview(request):
    if request.method == 'POST':
        logout(request)
        return redirect('home')
```

4. In html add this code (menu example):

```
<li><a href="#" onClick="document.getElementById('logout').submit()">Logout</a></li>
<form id="logout" method="POST" action="{% url 'accounts:logout' %}">
        {% csrf_token %}
        <input type="hidden">
</form>
```

Checking if User Authenticated in a Django Project

1. Example of code in a navbar:

```
<div class="collapse navbar-collapse" id="myNav">
        <ul class="nav navbar-bar navbar-right">
        {% if user.is_authenticated %}
                <li><a href="[path1]">Option 1</a></li>
                <li><a href="[path2]">Option 2</a></li>
                <li><a href="[logoutpath]">Logout</a></li>
        {% else %}
                <li><a href="{% url 'accounts:signup' %}">Signup</a></li>
                <li><a href="{% url 'accounts:login' %}">Login</a></li>
        {% endif %}
        </ul>
</div>
```

Using Static Files in a Django Project

1. In project\urls.py add:

```
from django.conf.urls.static import static
from django.conf import settings
```

then add this suffix to url definitions:

```
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

2. Go to project\settings and add to bottom under STATIC_URL:

```
STATIC_URL = '/static/'
STATIC_ROOT = BASE_DIR
```

3. create folder [projectname]\[appname]\static\[appname]
4. save static files in this new folder location