**FOOD and VEGTABLE SOCIAL SITE USING DJANGO**

**PBL REPORT**

**In**

**Internet and Web Programming**

**Submitted as a Course Project**

***by***

**Marla Akhil Reddy (13BCE1078)**

**Ramakanth (13BCE1)**

**Under the Guidance**

**Prof.Sandhya P**

**Associate Professor**

**School of Computing Science and Engineering**



**VIT University**

**Chennai- 600 127 – INDIA**

**May 2016**

|  |
| --- |
| **Contents**  **ABSTRACT 2**  **Description 3** |
| **Coding 4**  **Screen Shots 37** |
| **Referances 38** |

**Abstract:**

Now a days,Due to more and more people using websites for both showcasing their products, Adversicing and Socialising Themselves.There is Wide increse in number of Social Sites like LinkedIn,Instragram,PinIntrest etc..Which give Custom page and Adminstrative Options to their Users.One of best FrameWork to make this possible is Django Framework in Python.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://www.wikiwand.com/en/Reusability) and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://www.wikiwand.com/en/Don%27t_repeat_yourself). Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative[create, read, update and delete](https://www.wikiwand.com/en/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://www.wikiwand.com/en/Introspection_(computer_science)) and configured via admin models.

Some well-known sites that use Django include [Pinterest](https://www.wikiwand.com/en/Pinterest), [Instagram](https://www.wikiwand.com/en/Instagram), [Mozilla](https://www.wikiwand.com/en/Mozilla_Foundation), [The Washington Times](https://www.wikiwand.com/en/The_Washington_Times), [Disqus](https://www.wikiwand.com/en/Disqus), the[Public Broadcasting Service](https://www.wikiwand.com/en/Public_Broadcasting_Service), and [Bitbucket](https://www.wikiwand.com/en/Bitbucket).

**Description**

Admin.py File is used to Manage User from Django Administratin.

Forms.py File is used to handle Forms and Registration Items

Models.py is used to Handle Models or Templetes of Files.

Views.py is importants to handle model Views

Urls.py is useful to Configure Different Urls of Links

Templetes folder has all the fixed templetes of Html Front end Templetes which are handled by Django

We use sqlite3 as default Database to handle website.

**SCRIPT: (Home Page)**

<!DOCTYPE html>

<html>

<head>

<title>Vegetables - A Ecommerce Category Resposive Website | Home </title>

<link href="{% static 'css/bootstrap.css' %} " rel="stylesheet" type="text/css" media="all" />

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->

<script src="{% static 'js/jquery.min.js' %} "></script>

<link href="{% static 'css/style.css' %} " rel="stylesheet" type="text/css" media="all" />

<!--Ttheme-style-->

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="keywords" content="Vegetables web site, Vegtables ,Vegetables e commerce Website" /> <!--Js 2 -->

<script type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); } </script>

<!--fonts-->

<link href='http://fonts.googleapis.com/css?family=Exo:100,200,300,400,500,600,700,800,900' rel='stylesheet' type='text/css'>

<!--//fonts-->

<script src="{% static 'js/jquery.easydropdown.js' %}"></script> <!--Js 3 -->

</head>

<body>

<!--header-->

<div class="header"> <!--Ok1-->

<div class="container"> <!--Ok2-->

<div class="header-top"> <!--Ok3-->

<div class="logo"> <!--Ok4-->

<a href="{% static 'index.html' %}"><img src="{% static 'images/logo.png' %}" alt=" " ></a>

<!--4Ok--> </div>

<div class="search-in"> <!--Ok5-->

<div class="header-grid"> <!--Ok6-->

<ul>

<li class="in-up" ><a href="{% static 'contact.html' %}" class="scroll">Contact</a> <label>|</label></li>

<li class="in-up" ><a href="{% static 'signin.html' %}" class="scroll"> Sign in </a> <label>|</label></li>

<li ><a href="{% static 'signup.html' %}" class="scroll">Sign up</a> <label>|</label></li>

<li >

<select tabindex="4" class="dropdown">

<option value="" class="label" value="">Currency</option>

<option value="1">Rupees</option>

<option value="2">Dollar</option>

<option value="3">Others</option>

</select>

</li>

</ul>

<!--6Ok--></div>

<div class="search-top"> <!--Ok7-->

<div class="search"> <!--Ok8-->

<form>

<input type="text" value="Search" onfocus="this.value = '';" onblur="if (this.value == '') {this.value = '';}" >

<input type="submit" value="">

</form>

<!--8Ok--> </div>

<div class="cart"> <!--Ok9-->

<a href="#" class="cart-in"> </a>

<span></span>

<!--9Ok--> </div>

<div class="clearfix"> </div>

<!--7Ok--> </div>

<!--5Ok--> </div>

<div class="clearfix"> </div>

<!--4Ok--></div>

<div class="header-bottom-bottom">

<div class="top-nav">

<span class="menu"> </span>

<ul>

<li ><a href="about.html">About Us</a></li>

<li><a href="product.html" > Products</a></li>

<li><a href="services.html" > Services </a></li>

<li><a href="404.html" >Daily Market</a></li>

<li><a href="blog.html" > Blog </a></li>

<li><a href="contact.html" > Contact </a></li>

</ul>

<script>

$("span.menu").click(function(){

$(".top-nav ul").slideToggle(500, function(){

});

});

</script>

</div>

<div class="clearfix"> </div>

</div>

</div>

</div>

<div class="banner">

<!--slider-script-->

<script src="{% static 'js/responsiveslides.min.js' %}"></script> <!--Js 3 -->

<script>

$(function () {

$("#slider").responsiveSlides({

auto: true,

speed: 500,

namespace: "callbacks",

pager: true,

});

});

</script>

<!--//slider-script-->

<!-- Slideshow 4 -->

<div id="top" class="callbacks\_container">

<ul class="rslides" id="slider">

<li>

<img src="{% static 'images/banner.jpg' %}" alt="" />

<div class="banner-matter">

<div class="price">

<h2>We Got Every Vegetable in Your price</h2>

</div>

<div class="banner-para">

<p>You are lokking for Vegitables</p>

</div>

</div>

</li>

<li>

<img src="{% static 'images/banner1.jpg' %}" alt="" />

<div class="banner-matter">

<div class="price">

<h2>We Got Every Vegetable in Your price</h2>

</div>

<div class="banner-para">

<p>You are lokking for Vegitables</p>

</div>

</div>

</li>

<li>

<img src="{% static 'images/banner2.jpg' %}" alt="" />

<div class="banner-matter">

<div class="price">

<h2>We Got Every Vegetable in Your price</h2>

</div>

<div class="banner-para">

<p>You are lokking for Vegitables</p>

</div>

</div>

</li>

</ul>

</div>

</div>

<!-- //slider-->

<!--content-->

<div class="content">

<div class="container">

<div class="top-content">

<div class="content-top">

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi1.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi2.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi3.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="clearfix"> </div>

</div>

<div class="content-top">

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi4.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi5.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi6.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="col-md-3 look">

<h4><a href="blog.html">Check for Item in Blog </a></h4>

<a href="single.html"><img class="img-responsive" src="{% static 'images/pi7.jpg' %}" alt=" " ></a>

<p>This is a Vegitable and It's really good for Health </p>

<a href="blog.html" class="btn btn-1c">Learn More</a>

</div>

<div class="clearfix"> </div>

</div>

</div>

</div>

<!---->

<!--wmu Slider -->

<div class="content-bottom-top">

<div class="wmuSlider example1">

<div class="wmuSliderWrapper">

<article style="position: absolute; width: 100%; opacity: 0;">

<div class="content-bottom">

<div class="container">

<span class="corn"> </span>

<h3>Who we Are ?</h3>

<p>We are Vegitables Suppliers</p>

</div>

</div>

</article>

<article style="position: absolute; width: 100%; opacity: 0;">

<div class="content-bottom">

<div class="container">

<span class="corn corn-in"> </span>

<h3>Who we Are ?</h3>

<p>We are Vegitables Exporters </p>

</div>

</div>

</article>

<article style="position: absolute; width: 100%; opacity: 0;">

<div class="content-bottom">

<div class="container">

<span class="corn"> </span>

<h3>Who we Are ?</h3>

<p>We Export to Gulf and USA</p>

</div>

</div>

</article>

</div>

<script src="{% static 'js/jquery.wmuSlider.js' %}"></script>

<script>

$('.example1').wmuSlider();

</script>

</div>

</div>

<div class="map-top">

<div class="map">

<!--embedded map--

<iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d12947831.742778081!2d-95.665!3d37.599999999999994!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x54eab584e432360b%3A0x1c3bb99243deb742!2sUnited+States!5e0!3m2!1sen!2sin!4v1422444552833" ></iframe>

!-->

<iframe src="https://www.google.co.in/maps/@13.0475227,80.0689252,11z" ></iframe>

</div>

<div class="address">

<h5>Address</h5>

<p>Near Katpadi Rd, Vellore, Tamil Nadu 632014</p>

<a href="mailto:info@akhil.com" class="company">info@akhil.com</a>

</div>

</div>

</div>

<!--footer-->

<div class="footer">

<div class="container">

<p class="footer-grid">Copyright &copy; 2016 vegetables.com <a href="#" target="\_blank">Vegitables</a> </p>

</div>

</div></body></html>

**Coding**

**Models.py:**

from django.db import models

from django.contrib.auth.models import User

class Category(models.Model):

name = models.CharField(max\_length=128, unique=True)

views = models.IntegerField(default=0)

likes = models.IntegerField(default=0)

def \_\_unicode\_\_(self):

return self.name

class Page(models.Model):

category = models.ForeignKey(Category)

title = models.CharField(max\_length=128)

url = models.URLField()

views = models.IntegerField(default=0)

def \_\_unicode\_\_(self):

return self.title

class UserProfile(models.Model):

# A required line - links a UserProfile to User.

user = models.OneToOneField(User)

# The additional attributes we wish to include.

website = models.URLField(blank=True)

picture = models.ImageField(upload\_to='profile\_images', blank=True)

def \_\_unicode\_\_(self):

return self.user.username

**Admin.py:**

from django.contrib import admin

from rango.models import Category, Page, UserProfile

class PageAdmin(admin.ModelAdmin):

list\_display = ('title', 'category', 'url')

admin.site.register(Category)

admin.site.register(Page, PageAdmin)

admin.site.register(UserProfile)

Forms.py:

from django import forms

from django.contrib.auth.models import User

from rango.models import Page, Category, UserProfile

class CategoryForm(forms.ModelForm):

name = forms.CharField(max\_length=128, help\_text="Please enter the category name.")

views = forms.IntegerField(widget=forms.HiddenInput(), initial=0)

likes = forms.IntegerField(widget=forms.HiddenInput(), initial=0)

# An inline class to provide additional information on the form.

class Meta:

# Provide an association between the ModelForm and a model

model = Category

class PageForm(forms.ModelForm):

title = forms.CharField(max\_length=128, help\_text="Please enter the title of the page.")

url = forms.URLField(max\_length=200, help\_text="Please enter the URL of the page.")

views = forms.IntegerField(widget=forms.HiddenInput(), initial=0)

def clean(self):

cleaned\_data = self.cleaned\_data

url = cleaned\_data.get('url')

# If url is not empty and doesn't start with 'http://' add 'http://' to the beginning

if url and not url.startswith('http://'):

url = 'http://' + url

cleaned\_data['url'] = url

return cleaned\_data

class Meta:

# Provide an association between the ModelForm and a model

model = Page

# What fields do we want to include in our form?

# This way we don't need every field in the model present.

# Some fields may allow NULL values, so we may not want to include them...

# Here, we are hiding the foreign keys

fields = ('title', 'url','views')

class UserForm(forms.ModelForm):

username = forms.CharField(help\_text="Please enter a username.")

email = forms.CharField(help\_text="Please enter your email.")

password = forms.CharField(widget=forms.PasswordInput(), help\_text="Please enter a password.")

class Meta:

model = User

fields = ('username', 'email', 'password')

class UserProfileForm(forms.ModelForm):

website = forms.URLField(help\_text="Please enter your website.", required=False)

picture = forms.ImageField(help\_text="Select a profile image to upload.", required=False)

class Meta:

model = UserProfile

fields = ('website', 'picture')

**Links.py:**

from django.conf.urls import patterns, url

from rango import views

urlpatterns = patterns('',

url(r'^$', views.index, name='index'),

url(r'^index1', views.index, name='index1'),

url(r'^about/$', views.about, name='about'),

url(r'^about1', views.about, name='about1'),

url(r'^category/(?P<category\_name\_url>\w+)/$', views.category, name='category'),

url(r'^add\_category/$', views.add\_category, name='add\_category'),

url(r'^category/(?P<category\_name\_url>\w+)/add\_page/$', views.add\_page, name='add\_page'),

url(r'^register/$', views.register, name='register'),

url(r'^login/$', views.user\_login, name='login'),

url(r'^restricted/$', views.restricted, name='restricted'),

url(r'^logout/$', views.user\_logout, name='logout'),

#url(r'^search/$', views.search, name='search'),

url(r'^profile/$', views.profile, name='profile'),

url(r'^goto/$', views.track\_url, name='track\_url'),

url(r'^like\_category/$', views.like\_category, name='like\_category'),

url(r'^suggest\_category/$', views.suggest\_category, name='suggest\_category'),

url(r'^auto\_add\_page/$', views.auto\_add\_page, name='auto\_add\_page'),

)

**Views.py**

from django.template import RequestContext

from django.shortcuts import render\_to\_response

from django.http import HttpResponse

from rango.models import Category

from rango.models import Page

from rango.models import UserProfile

from rango.forms import UserForm, UserProfileForm

from rango.forms import CategoryForm, PageForm

from django.contrib.auth import authenticate, login, logout

from django.http import HttpResponseRedirect

from django.contrib.auth.decorators import login\_required

from datetime import datetime

from rango.bing\_search import run\_query

from django.contrib.auth.models import User

from django.shortcuts import redirect

def encode\_url(str):

return str.replace(' ', '\_')

def decode\_url(str):

return str.replace('\_', ' ')

def get\_category\_list(max\_results=0, starts\_with=''):

cat\_list = []

if starts\_with:

cat\_list = Category.objects.filter(name\_\_startswith=starts\_with)

else:

cat\_list = Category.objects.all()

if max\_results > 0:

if (len(cat\_list) > max\_results):

cat\_list = cat\_list[:max\_results]

for cat in cat\_list:

cat.url = encode\_url(cat.name)

return cat\_list

def index(request):

context = RequestContext(request)

top\_category\_list = Category.objects.order\_by('-likes')[:5]

for category in top\_category\_list:

category.url = encode\_url(category.name)

context\_dict = {'categories': top\_category\_list}

cat\_list = get\_category\_list()

context\_dict['cat\_list'] = cat\_list

page\_list = Page.objects.order\_by('-views')[:5]

context\_dict['pages'] = page\_list

if request.session.get('last\_visit'):

# The session has a value for the last visit

last\_visit\_time = request.session.get('last\_visit')

visits = request.session.get('visits', 0)

if (datetime.now() - datetime.strptime(last\_visit\_time[:-7], "%Y-%m-%d %H:%M:%S")).days > 0:

request.session['visits'] = visits + 1

else:

# The get returns None, and the session does not have a value for the last visit.

request.session['last\_visit'] = str(datetime.now())

request.session['visits'] = 1

# Render and return the rendered response back to the user.

return render\_to\_response('rango/index.html', context\_dict, context)

def about(request):

# Request the context.

context = RequestContext(request)

context\_dict = {}

cat\_list = get\_category\_list()

context\_dict['cat\_list'] = cat\_list

# If the visits session varible exists, take it and use it.

# If it doesn't, we haven't visited the site so set the count to zero.

count = request.session.get('visits',0)

context\_dict['visit\_count'] = count

# Return and render the response, ensuring the count is passed to the template engine.

return render\_to\_response('rango/index1.html', context\_dict , context)

def category(request, category\_name\_url):

# Request our context

context = RequestContext(request)

# Change underscores in the category name to spaces.

# URL's don't handle spaces well, so we encode them as underscores.

category\_name = decode\_url(category\_name\_url)

# Build up the dictionary we will use as out template context dictionary.

context\_dict = {'category\_name': category\_name, 'category\_name\_url': category\_name\_url}

cat\_list = get\_category\_list()

context\_dict['cat\_list'] = cat\_list

try:

# Find the category with the given name.

# Raises an exception if the category doesn't exist.

# We also do a case insensitive match.

category = Category.objects.get(name\_\_iexact=category\_name)

context\_dict['category'] = category

# Retrieve all the associated pages.

# Note that filter returns >= 1 model instance.

pages = Page.objects.filter(category=category).order\_by('-views')

# Adds our results list to the template context under name pages.

context\_dict['pages'] = pages

except Category.DoesNotExist:

# We get here if the category does not exist.

# Will trigger the template to display the 'no category' message.

pass

if request.method == 'POST':

query = request.POST.get('query')

if query:

query = query.strip()

result\_list = run\_query(query)

context\_dict['result\_list'] = result\_list

# Go render the response and return it to the client.

return render\_to\_response('rango/category.html', context\_dict, context)

@login\_required

def add\_category(request):

# Get the context from the request.

context = RequestContext(request)

cat\_list = get\_category\_list()

context\_dict = {}

context\_dict['cat\_list'] = cat\_list

# A HTTP POST?

if request.method == 'POST':

form = CategoryForm(request.POST)

# Have we been provided with a valid form?

if form.is\_valid():

# Save the new category to the database.

form.save(commit=True)

# Now call the index() view.

# The user will be shown the homepage.

return index(request)

else:

# The supplied form contained errors - just print them to the terminal.

print form.errors

else:

# If the request was not a POST, display the form to enter details.

form = CategoryForm()

# Bad form (or form details), no form supplied...

# Render the form with error messages (if any).

context\_dict['form'] = form

return render\_to\_response('rango/add\_category.html', context\_dict, context)

@login\_required

def add\_page(request, category\_name\_url):

context = RequestContext(request)

cat\_list = get\_category\_list()

context\_dict = {}

context\_dict['cat\_list'] = cat\_list

category\_name = decode\_url(category\_name\_url)

if request.method == 'POST':

form = PageForm(request.POST)

if form.is\_valid():

# This time we cannot commit straight away.

# Not all fields are automatically populated!

page = form.save(commit=False)

# Retrieve the associated Category object so we can add it.

try:

cat = Category.objects.get(name=category\_name)

page.category = cat

except Category.DoesNotExist:

return render\_to\_response( 'rango/add\_page.html',

context\_dict,

context)

# Also, create a default value for the number of views.

page.views = 0

# With this, we can then save our new model instance.

page.save()

# Now that the page is saved, display the category instead.

return category(request, category\_name\_url)

else:

print form.errors

else:

form = PageForm()

context\_dict['category\_name\_url']= category\_name\_url

context\_dict['category\_name'] = category\_name

context\_dict['form'] = form

return render\_to\_response( 'rango/add\_page.html',

context\_dict,

context)

def register(request):

# Request the context.

context = RequestContext(request)

cat\_list = get\_category\_list()

context\_dict = {}

context\_dict['cat\_list'] = cat\_list

# Boolean telling us whether registration was successful or not.

# Initially False; presume it was a failure until proven otherwise!

registered = False

# If HTTP POST, we wish to process form data and create an account.

if request.method == 'POST':

# Grab raw form data - making use of both FormModels.

user\_form = UserForm(data=request.POST)

profile\_form = UserProfileForm(data=request.POST)

# Two valid forms?

if user\_form.is\_valid() and profile\_form.is\_valid():

# Save the user's form data. That one is easy.

user = user\_form.save()

# Now a user account exists, we hash the password with the set\_password() method.

# Then we can update the account with .save().

user.set\_password(user.password)

user.save()

# Now we can sort out the UserProfile instance.

# We'll be setting values for the instance ourselves, so commit=False prevents Django from saving the instance automatically.

profile = profile\_form.save(commit=False)

profile.user = user

# Profile picture supplied? If so, we put it in the new UserProfile.

if 'picture' in request.FILES:

profile.picture = request.FILES['picture']

# Now we save the model instance!

profile.save()

# We can say registration was successful.

registered = True

# Invalid form(s) - just print errors to the terminal.

else:

print user\_form.errors, profile\_form.errors

# Not a HTTP POST, so we render the two ModelForms to allow a user to input their data.

else:

user\_form = UserForm()

profile\_form = UserProfileForm()

context\_dict['user\_form'] = user\_form

context\_dict['profile\_form']= profile\_form

context\_dict['registered'] = registered

# Render and return!

return render\_to\_response(

'rango/register.html',

context\_dict,

context)

def user\_login(request):

# Obtain our request's context.

context = RequestContext(request)

cat\_list = get\_category\_list()

context\_dict = {}

context\_dict['cat\_list'] = cat\_list

# If HTTP POST, pull out form data and process it.

if request.method == 'POST':

username = request.POST['username']

password = request.POST['password']

# Attempt to log the user in with the supplied credentials.

# A User object is returned if correct - None if not.

user = authenticate(username=username, password=password)

# A valid user logged in?

if user is not None:

# Check if the account is active (can be used).

# If so, log the user in and redirect them to the homepage.

if user.is\_active:

login(request, user)

return HttpResponseRedirect('/rango/')

# The account is inactive; tell by adding variable to the template context.

else:

context\_dict['disabled\_account'] = True

return render\_to\_response('rango/login.html', context\_dict, context)

# Invalid login details supplied!

else:

print "Invalid login details: {0}, {1}".format(username, password)

context\_dict['bad\_details'] = True

return render\_to\_response('rango/login.html', context\_dict, context)

# Not a HTTP POST - most likely a HTTP GET. In this case, we render the login form for the user.

else:

return render\_to\_response('rango/login.html', context\_dict, context)

@login\_required

def restricted(request):

context = RequestContext(request)

cat\_list = get\_category\_list()

context\_dict = {}

context\_dict['cat\_list'] = cat\_list

return render\_to\_response('rango/restricted.html', context\_dict, context)

# Only allow logged in users to logout - add the @login\_required decorator!

@login\_required

def user\_logout(request):

# As we can assume the user is logged in, we can just log them out.

logout(request)

# Take the user back to the homepage.

return HttpResponseRedirect('/rango/')

def search(request):

context = RequestContext(request)

cat\_list = get\_category\_list()

context\_dict = {}

context\_dict['cat\_list'] = cat\_list

result\_list = []

if request.method == 'POST':

query = request.POST['query'].strip()

if query:

# Run our Bing function to get the results list!

result\_list = run\_query(query)

context\_dict['result\_list'] = result\_list

return render\_to\_response('rango/search.html', context\_dict, context)

@login\_required

def profile(request):

context = RequestContext(request)

cat\_list = get\_category\_list()

context\_dict = {'cat\_list': cat\_list}

u = User.objects.get(username=request.user)

try:

up = UserProfile.objects.get(user=u)

except:

up = None

context\_dict['user'] = u

context\_dict['userprofile'] = up

return render\_to\_response('rango/profile.html', context\_dict, context)

def track\_url(request):

context = RequestContext(request)

page\_id = None

url = '/rango/'

if request.method == 'GET':

if 'page\_id' in request.GET:

page\_id = request.GET['page\_id']

try:

page = Page.objects.get(id=page\_id)

page.views = page.views + 1

page.save()

url = page.url

except:

pass

return redirect(url)

@login\_required

def like\_category(request):

context = RequestContext(request)

cat\_id = None

if request.method == 'GET':

cat\_id = request.GET['category\_id']

likes = 0

if cat\_id:

category = Category.objects.get(id=int(cat\_id))

if category:

likes = category.likes + 1

category.likes = likes

category.save()

return HttpResponse(likes)

def suggest\_category(request):

context = RequestContext(request)

cat\_list = []

starts\_with = ''

if request.method == 'GET':

starts\_with = request.GET['suggestion']

else:

starts\_with = request.POST['suggestion']

cat\_list = get\_category\_list(8, starts\_with)

return render\_to\_response('rango/category\_list.html', {'cat\_list': cat\_list }, context)

@login\_required

def auto\_add\_page(request):

context = RequestContext(request)

cat\_id = None

url = None

title = None

context\_dict = {}

if request.method == 'GET':

cat\_id = request.GET['category\_id']

url = request.GET['url']

title = request.GET['title']

if cat\_id:

category = Category.objects.get(id=int(cat\_id))

p = Page.objects.get\_or\_create(category=category, title=title, url=url)

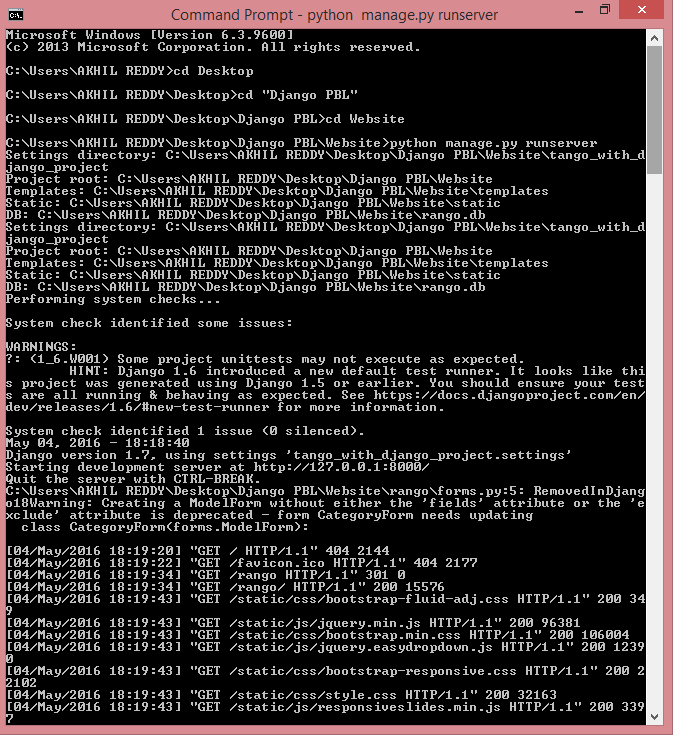
pages = Page.objects.filter(category=category).order\_by('-views')

# Adds our results list to the template context under name pages.

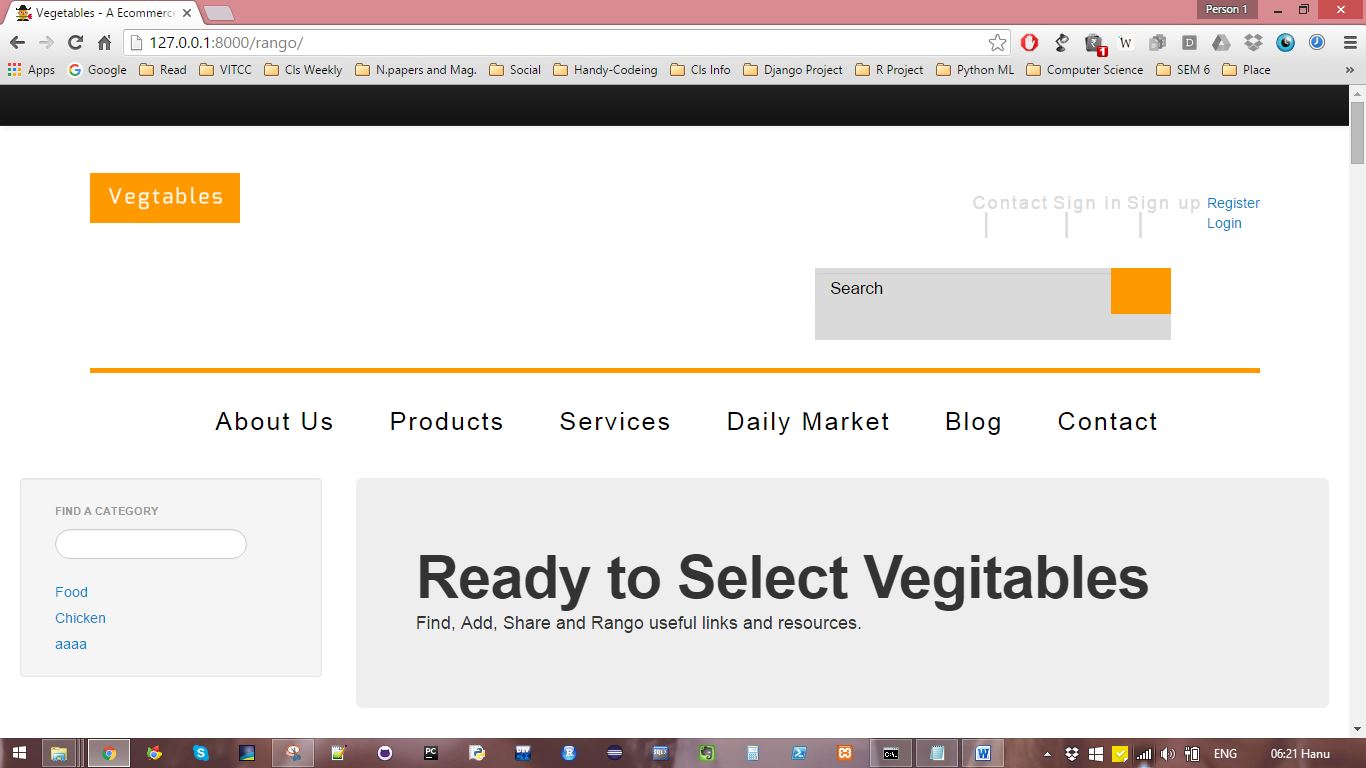
context\_dict['pages'] = pages return render\_to\_response('rango/page\_list.html', context\_dict, context)

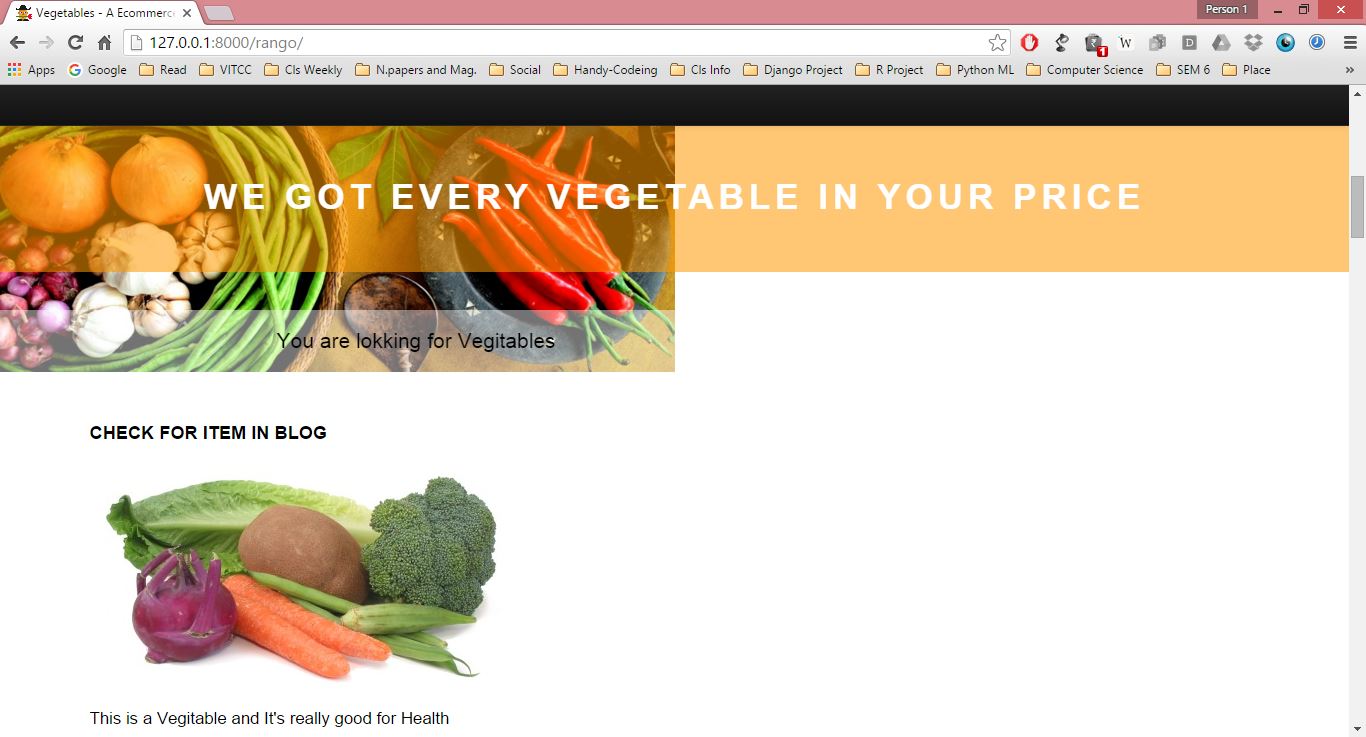
**SCREEN SHOTS of OUTPUT:**

**Running Server:**

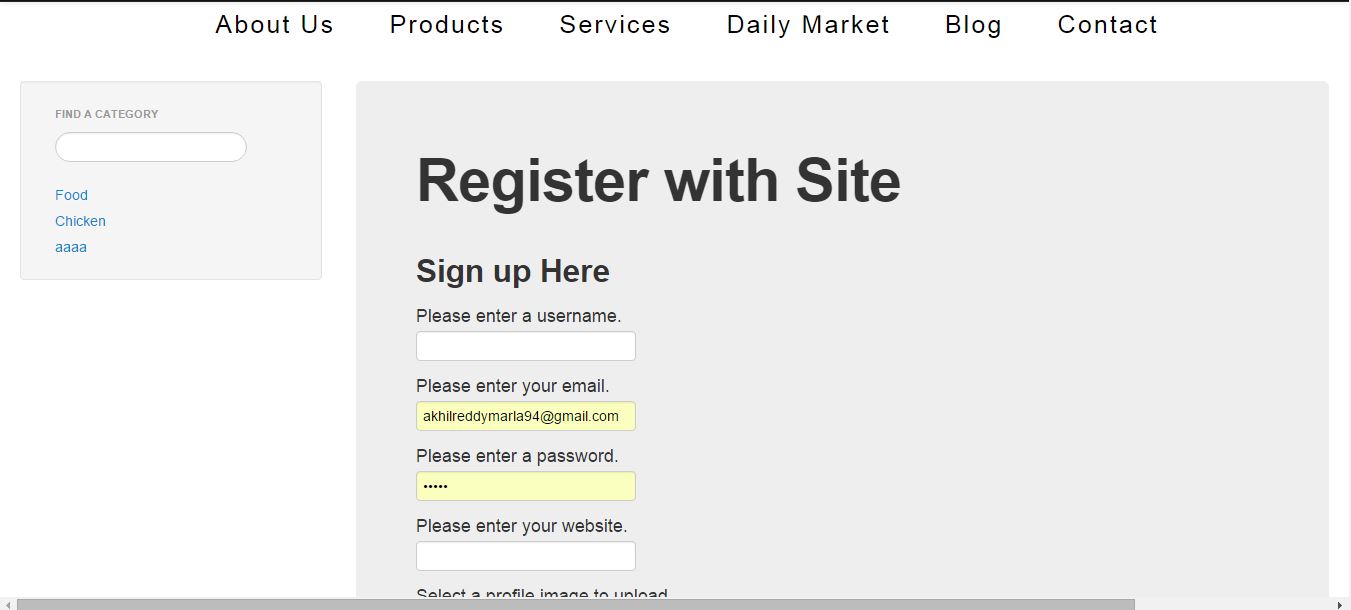


**Website Home Page:**

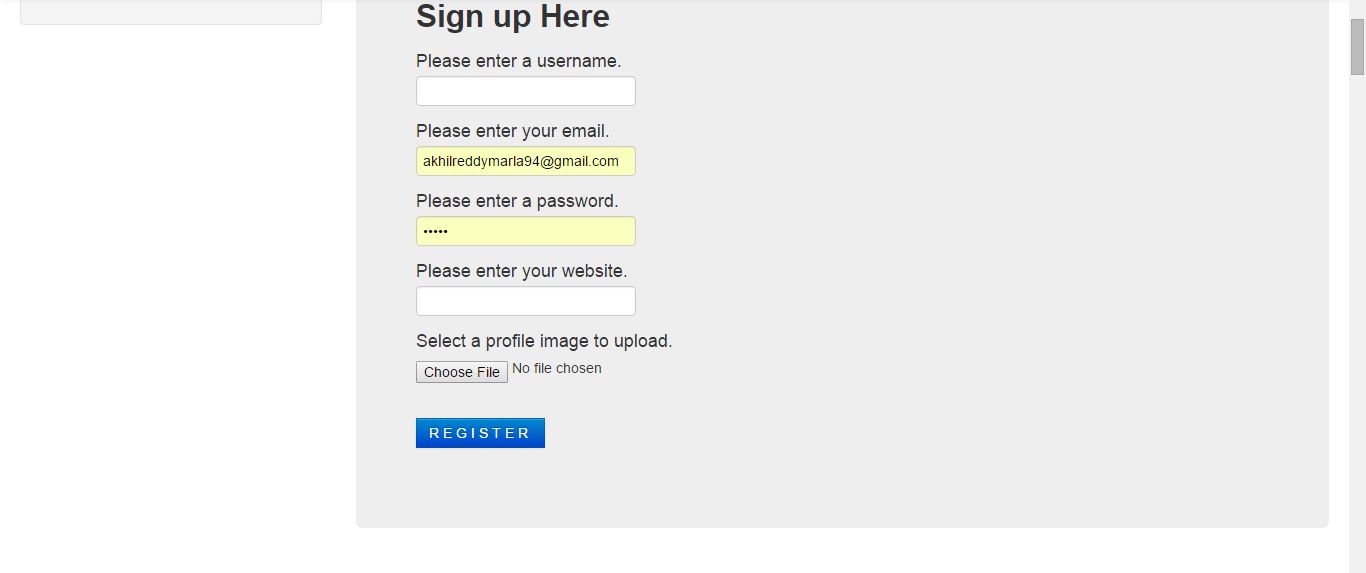




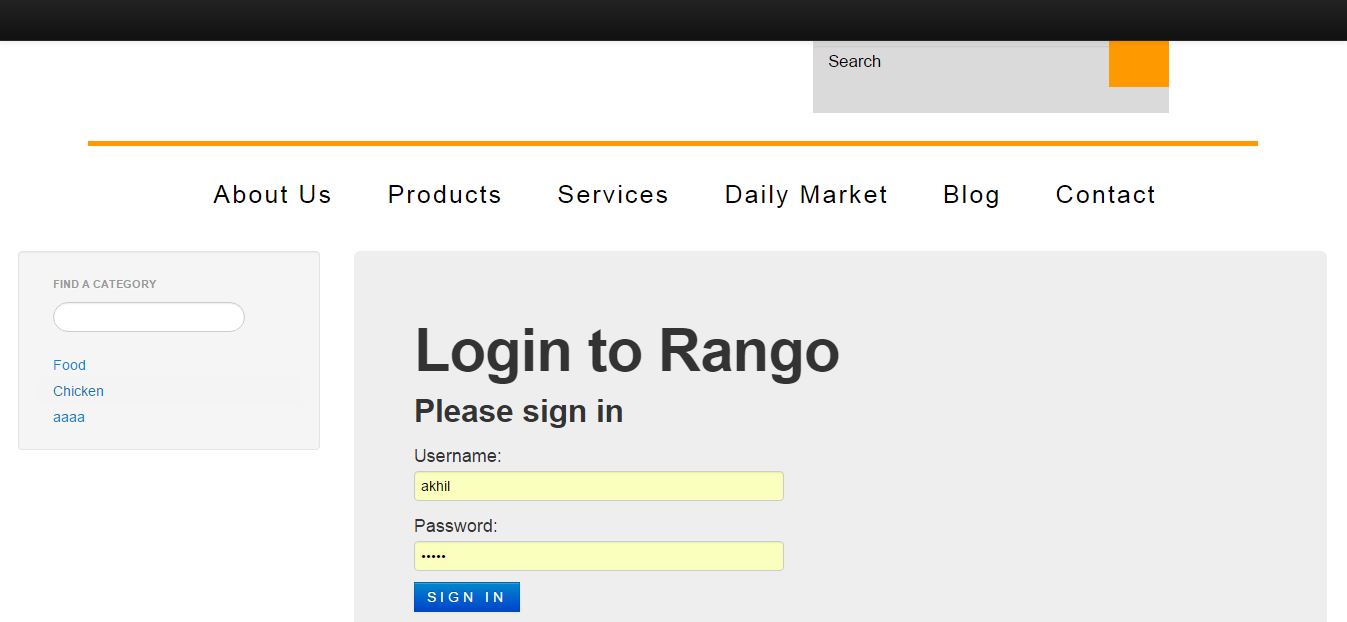
**Register to Website:**

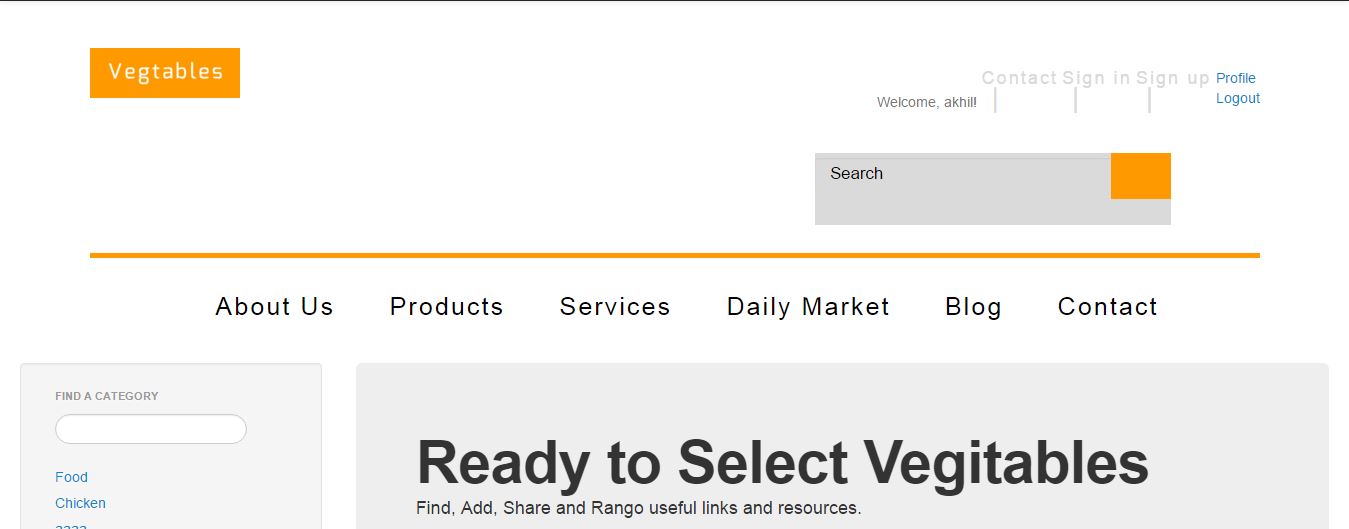


Adding Details like Image:

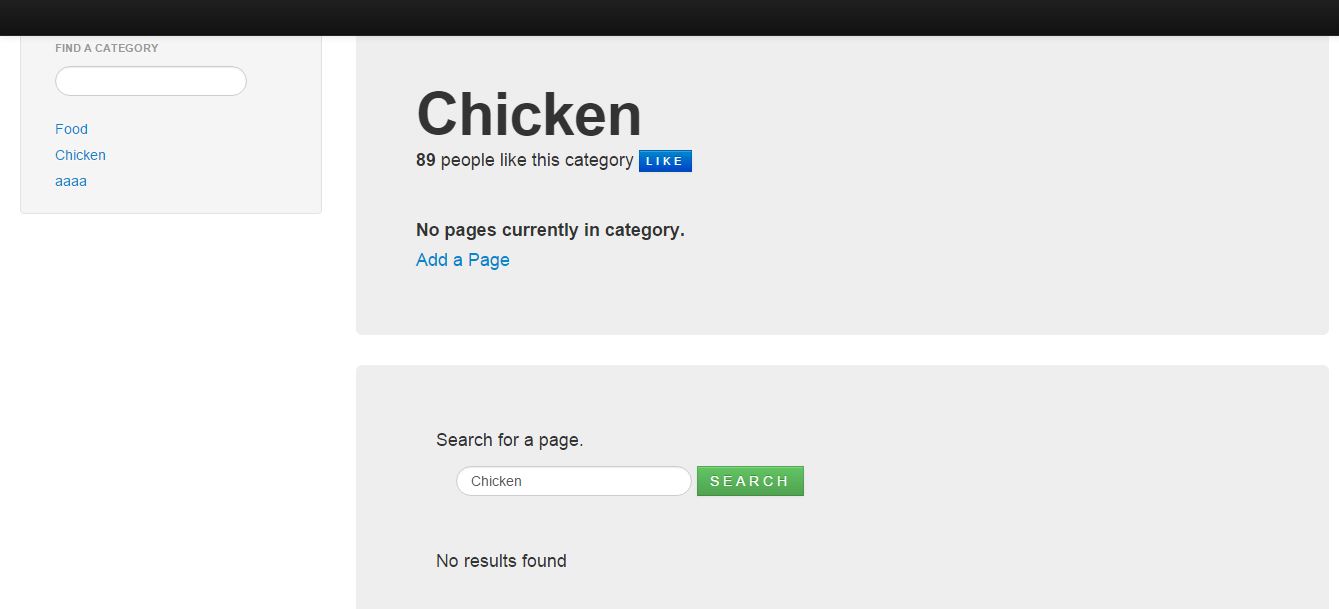
****

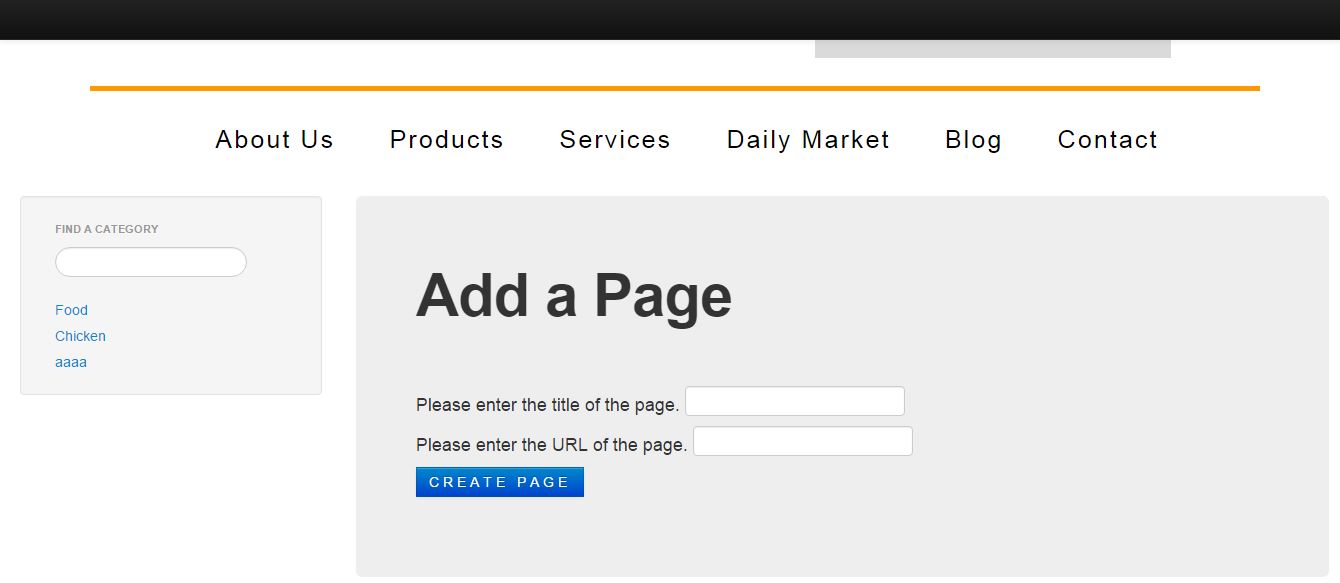
**Login:**

****

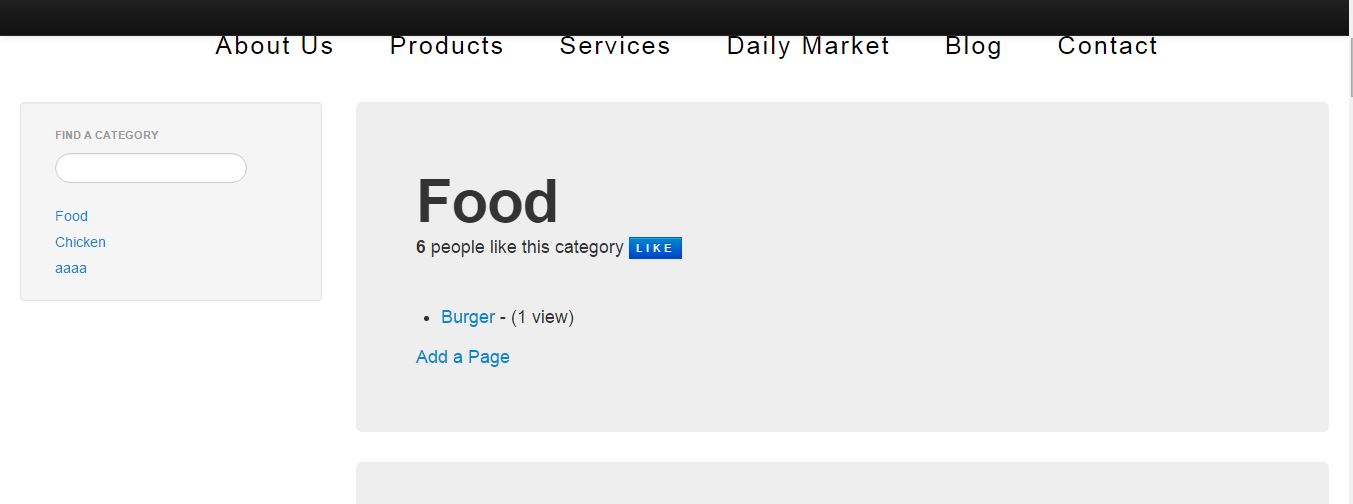
****

**Add Page Links of Good Websites or Resturents**

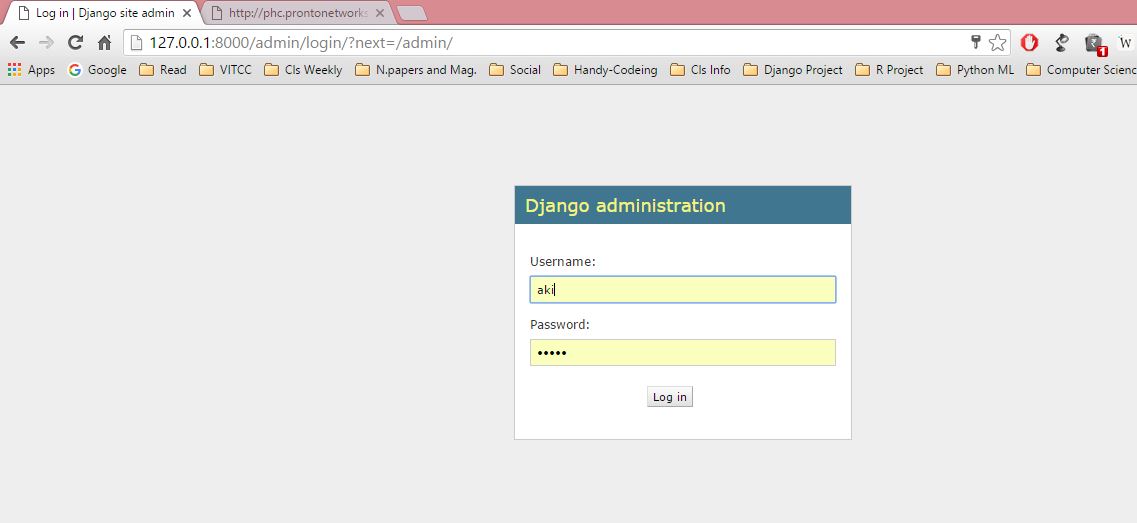


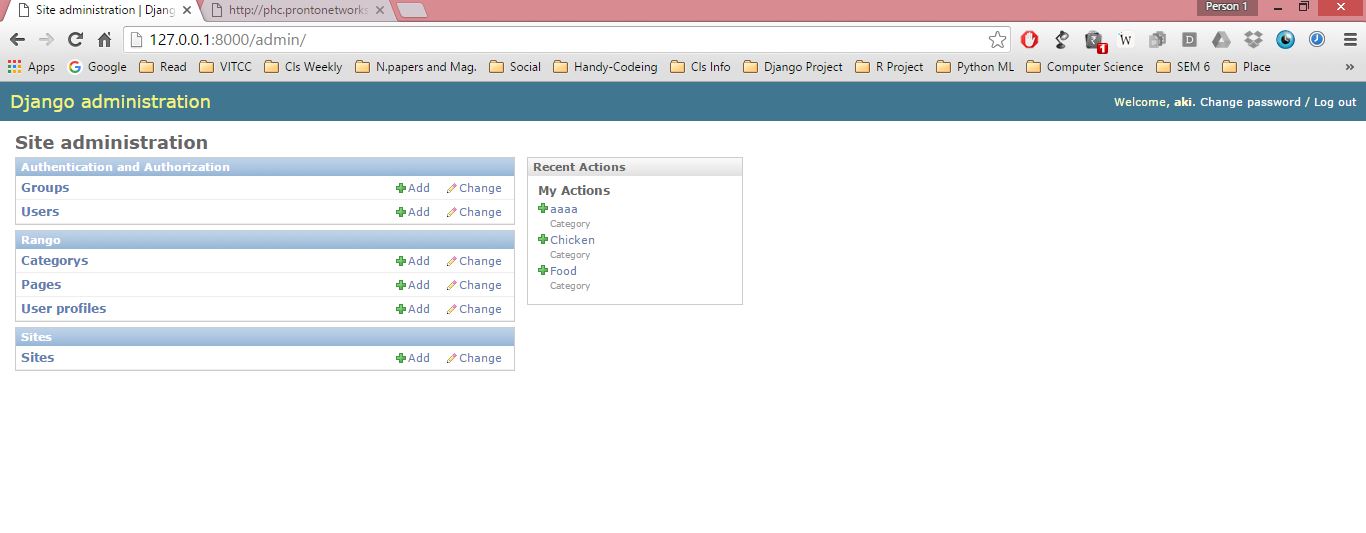


**Views of Page:**

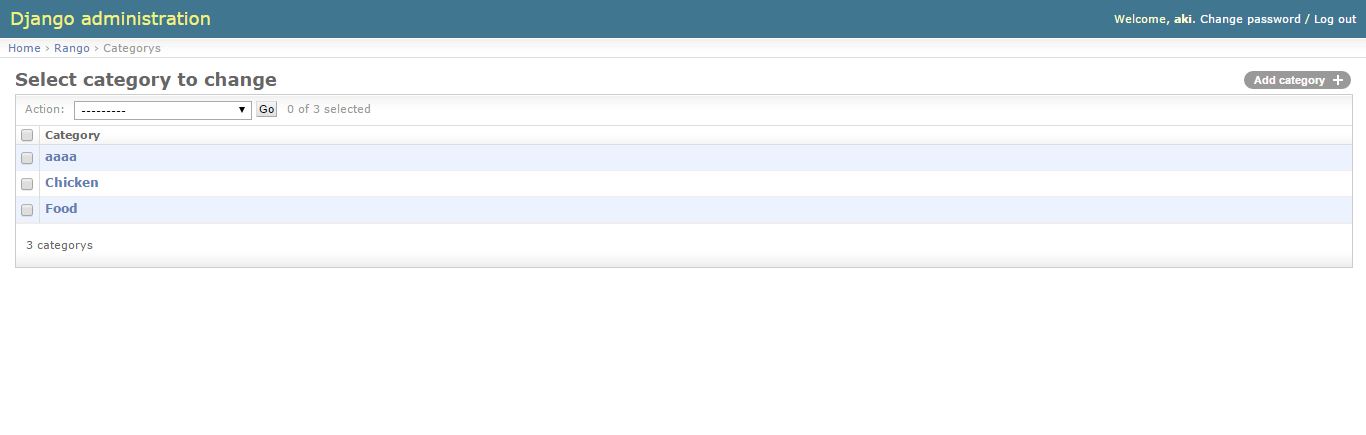


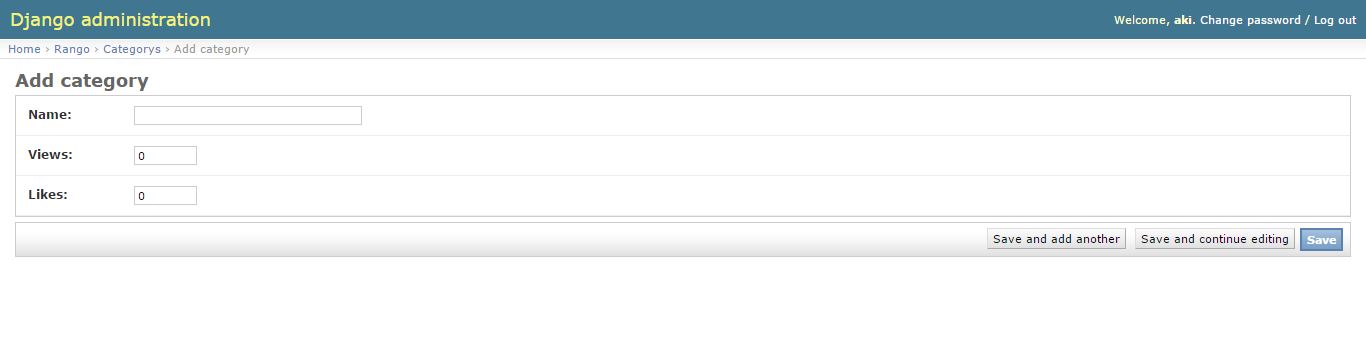
**Admin Side:**

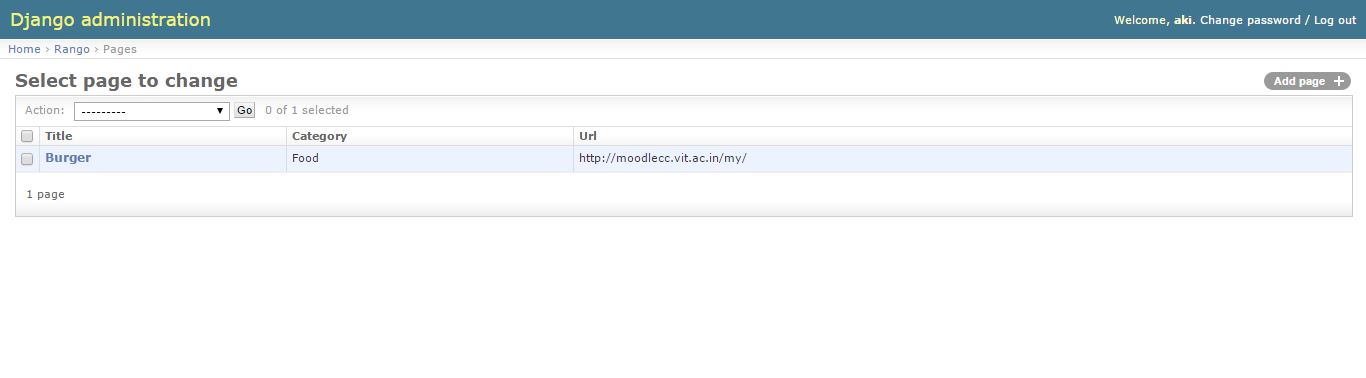


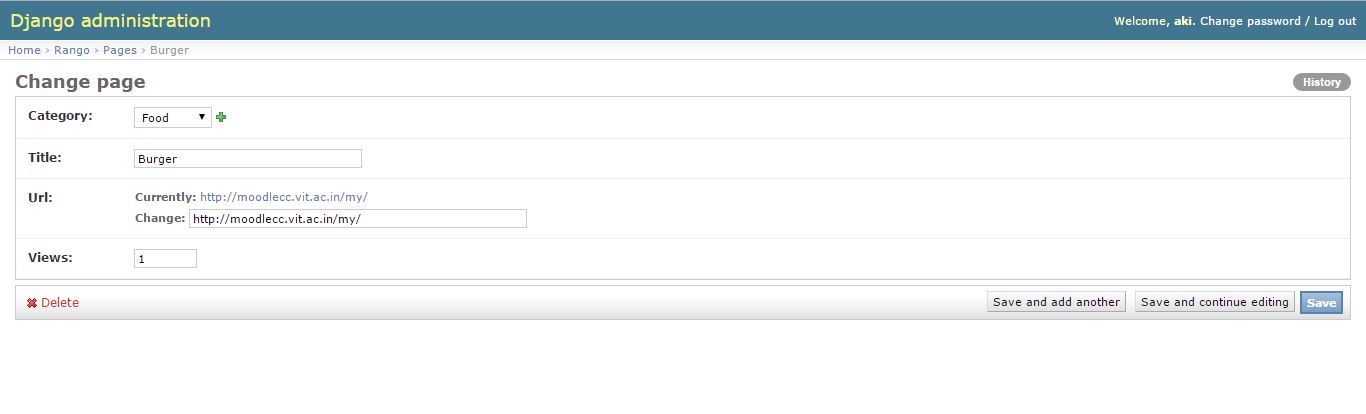


**Create Categories:**

****

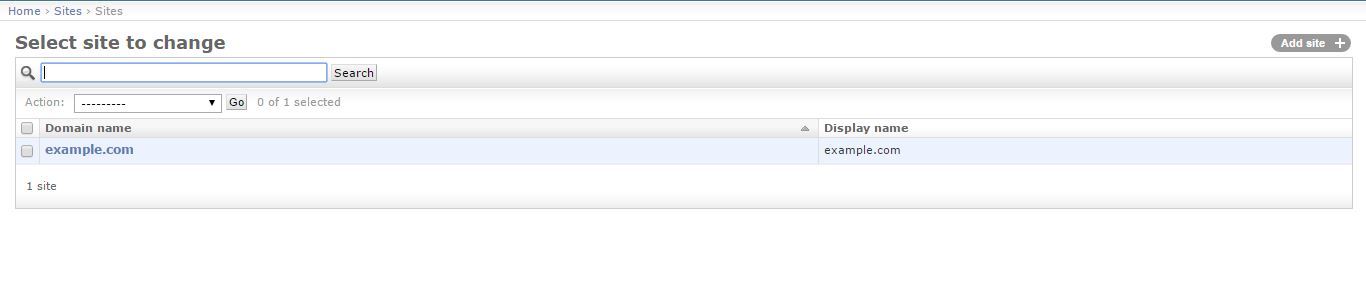
**Handle Pages:**



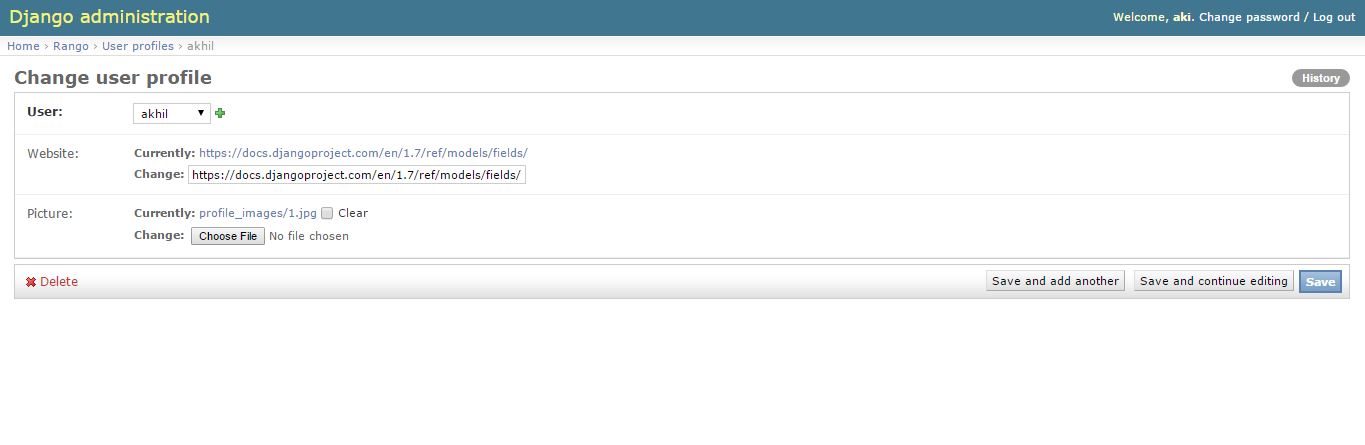


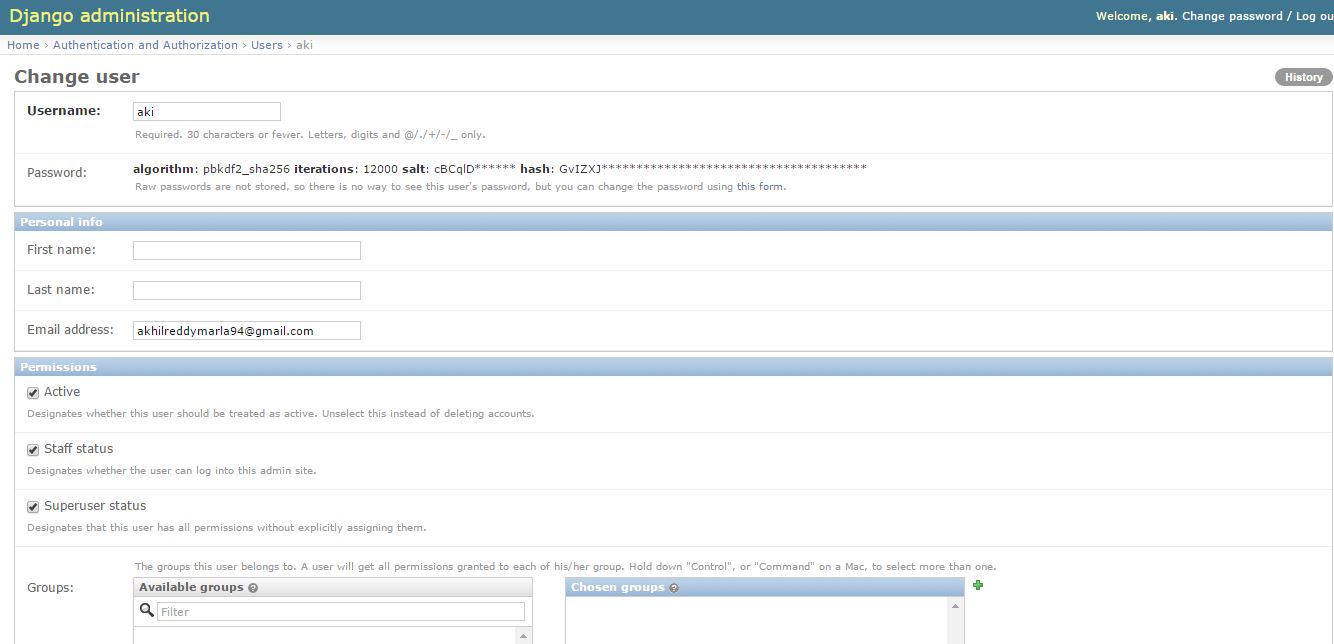
**Handle Sites:**



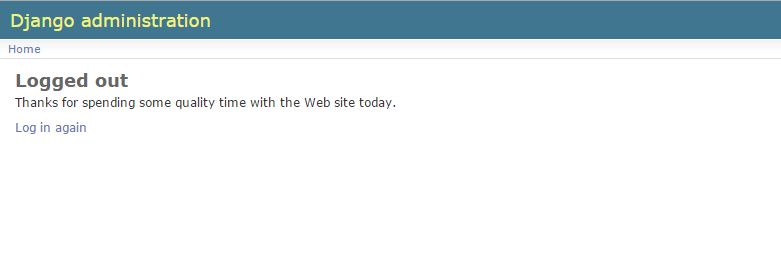
****

**Handle Users:**





**Logout:**



**Referances:**

<https://www.wikiwand.com/>

<https://www.djangoproject.com/start/>

<https://code.djangoproject.com/wiki/DjangoResources#Djangoapplicationcomponents>

**Tangowithdjango e book**

Google.com for Modules.