**Worksheet 10**

**Authentication & Orders**

Download and extract the file “Worksheet 10 Skeleton Program” from Moodle.

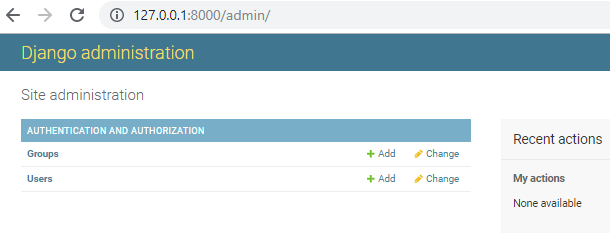
**Sign Up**

The first task in this exercise is to authenticate users on the front end website and create a sign up form

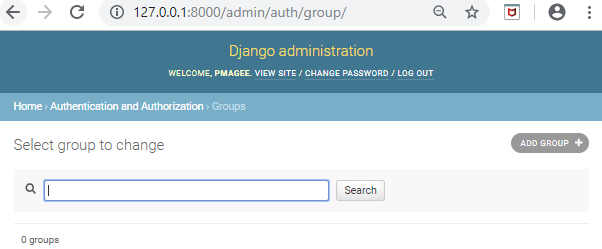
Access the admin backend and create a group that will be assigned to every customer that is created on the website

Go to <http://127.0.0.1:8000/admin/> and log in with your superuser account details

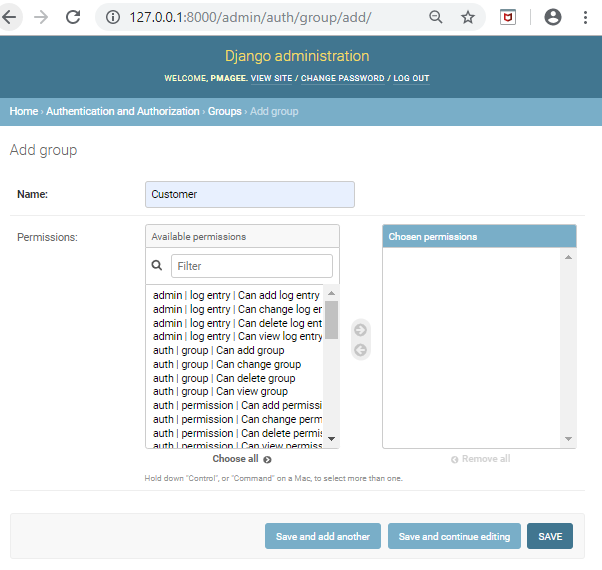
Click on the **Groups** link shown below:



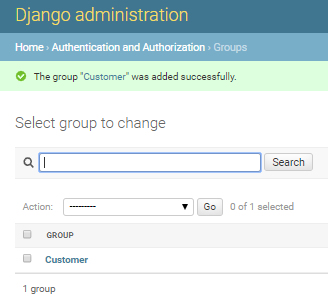
You will see an option on the top right-hand side called **ADD GROUP** as shown below. Click on this link in order to create a new group



In order to create a new group, you need to give it a name. Type the name Customer in the Name field as shown below & click the **Save** button



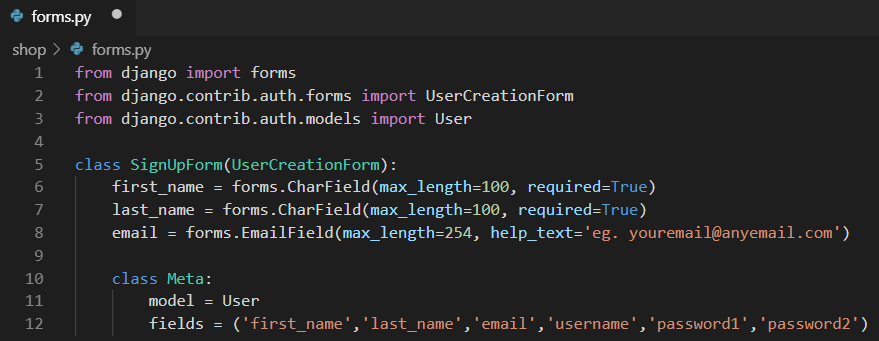
When you click the SAVE button you will see that the group has been created:

****

Log out of the admin website and proceed to VS Code

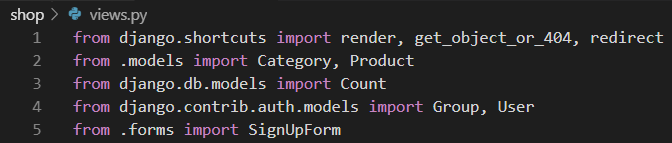
Right click on the **shop** app folder and create a new file called **forms.py**

Enter the following code in to the **forms.py** file

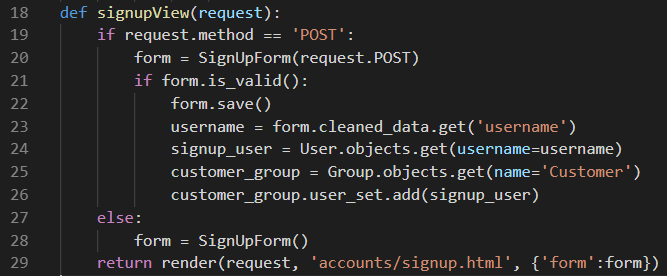


We map the model in order to save the data in the user table

Open **shop/views.py** and add the following imports at the top of the file:



Add the following function to the same file:

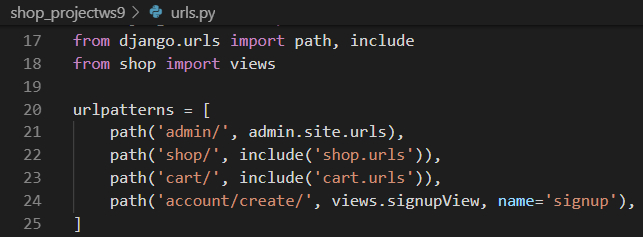


**Template**

Right click on the **shop/templates** folder and create a new folder called **accounts**. Go to your browser & access Moodle. Right click on the file called **signup.html** and choose the menu option “Save Link as”. Navigate to the accounts folder inside your project and click save.

**URL**

Open the **shop\_projectws9/urls.py** file and add the following code to map the route to the view:

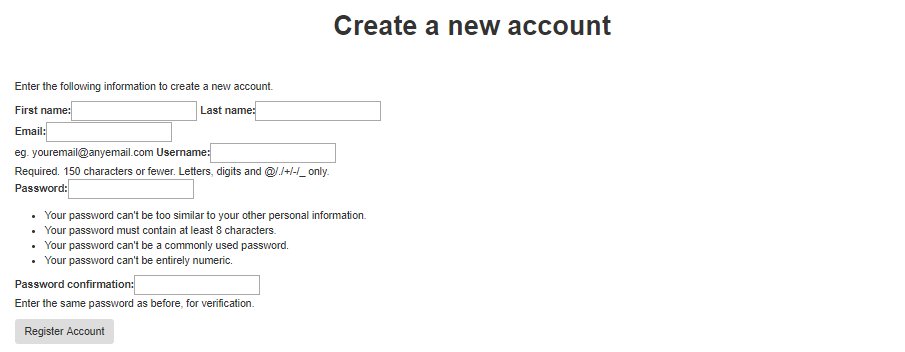


Save the file and open the terminal and restart the server

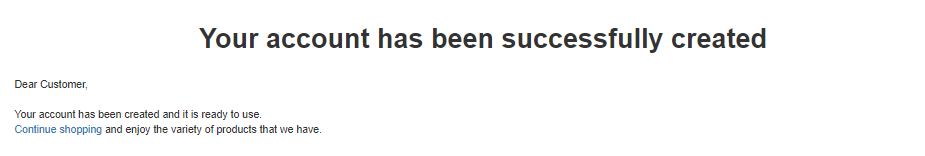
In the browser type the following url into the url bar URL bar:

<http://127.0.0.1:8000/account/create/>

The sign up page should be displayed as shown below:

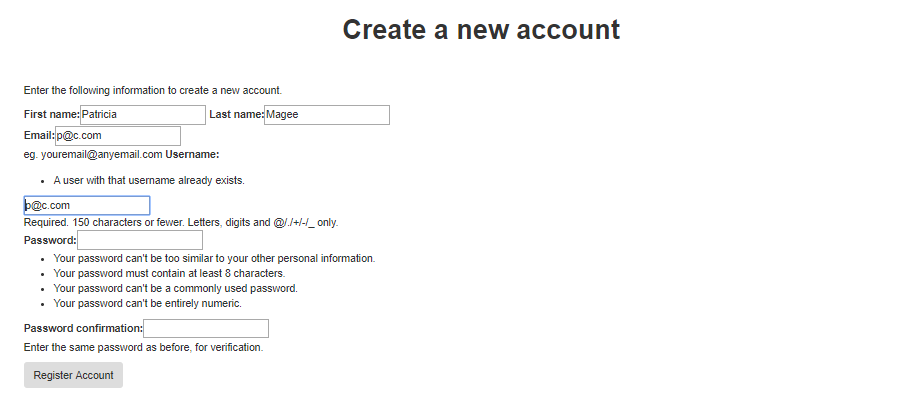


Fill in the information required on the form to register a new account. When you click on the **Register Account** button you should see the following message:



Access the <http://127.0.0.1:8000/account/create/> url again and try and create a new user using the same details as before.

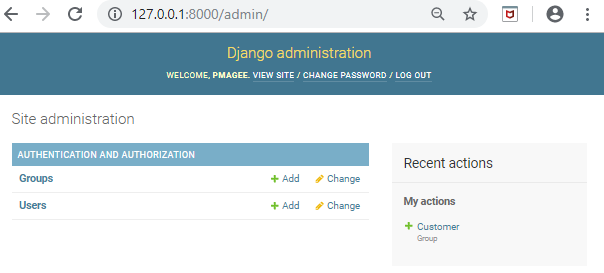
When you click the **Register Account** button you will see a message displayed as shown below. Django does not allow us to register the same user name again in the system.



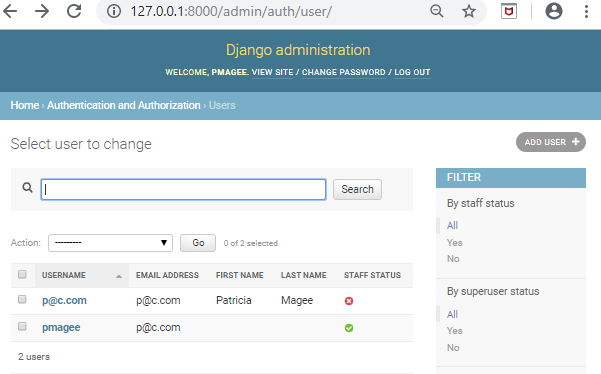
Next we will access the admin side of the website to view the new customer that we created.

Go to <http://127.0.0.1:8000/admin/> and log in with your superuser account details

Click on the **Users** link as shown below



In the list of users you should see the new user that you just created during registration



Note that the staff status has a red x which means this user has no admin privileges on the back end of the website

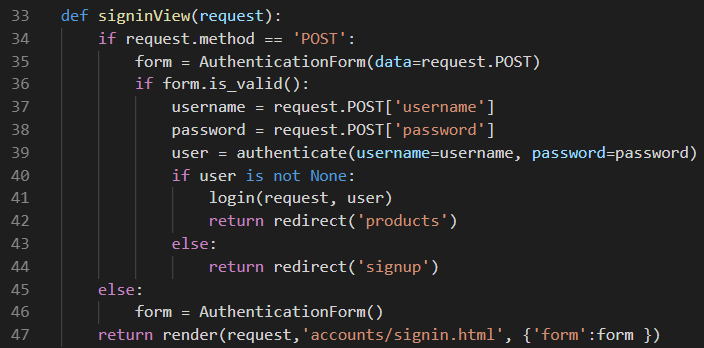
**Sign In**

The second task in this exercise is to provide sign in functionality on the front end website and create a sign in form

In VS Code, open **shop/views.py** and add the following **imports** at the top of the file:



Add the following function to the same file:

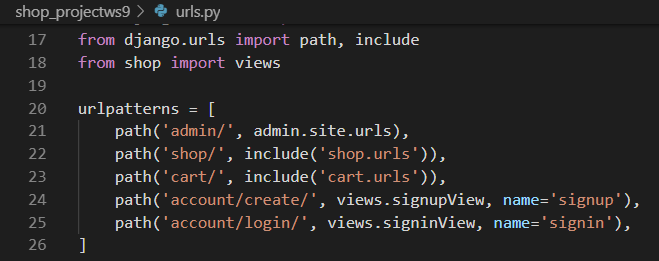


**Template**

Go to your browser & access Moodle. Right click on the file called **signin.html** and choose the menu option “Save Link as”. Navigate to the **accounts** folder inside your project and click save.

**URL**

Open the **shop\_projectws9/urls.py** file and add the following code to map the route to the view:

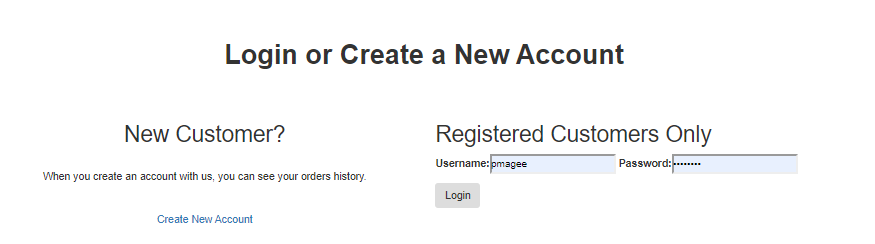


Save the file and open the terminal and restart the server

In the browser type the following url into the url bar:

<http://127.0.0.1:8000/account/login/>

The login page should be displayed as shown below:



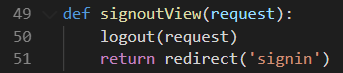
**Sign Out**

The third task in this exercise is to provide sign out functionality on the front end website.

In VS Code, open **shop/views.py** and add the following **import** at the top of the file:

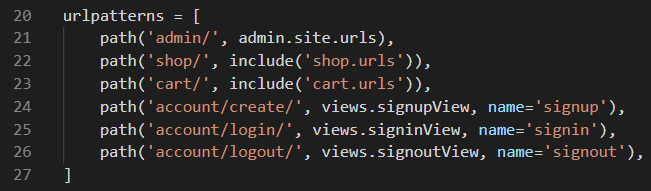


Add the following function to the same file:



**URL**

Open the **shop\_projectws9/urls.py** file and add the following code to map the route to the view:

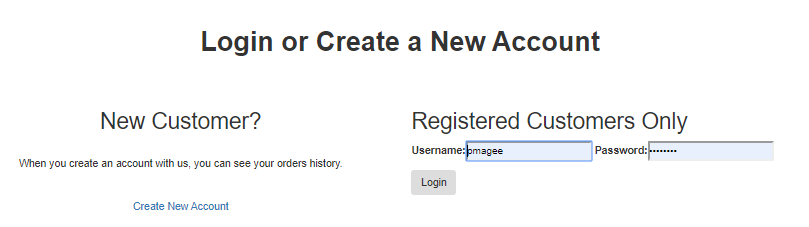


Save the file and open the terminal and restart the server

In the browser type the following url into the url bar:

<http://127.0.0.1:8000/account/logout/>

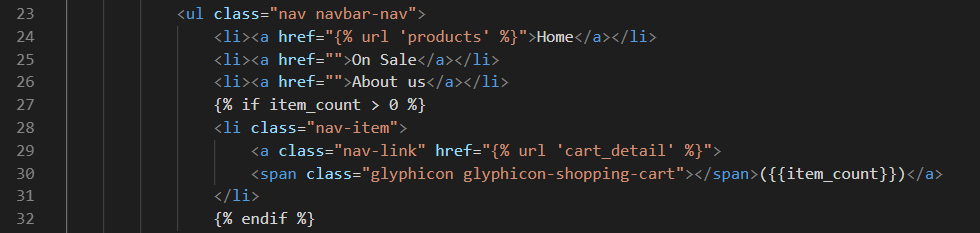
When you access the url above, you are logged out and redirected to the following page:

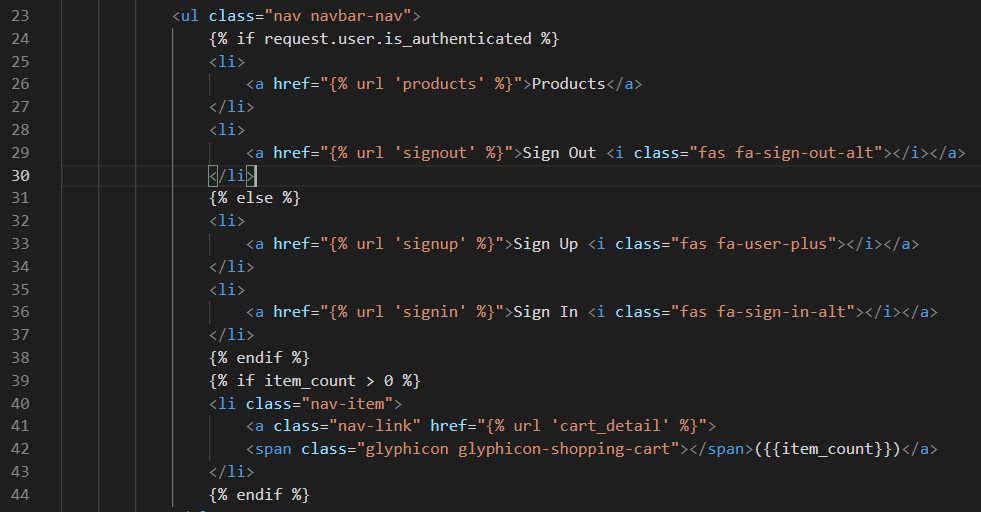


**Add links to navbar**

We will now add links to our nav bar for log in/ log out etc.

Open the base.html file and replace the 3 lines highlighted below with the block of code provided





In the browser type the following url into the url bar:

<http://127.0.0.1:8000/shop/>

If you are already logged in, you should see the following options in the navbar:

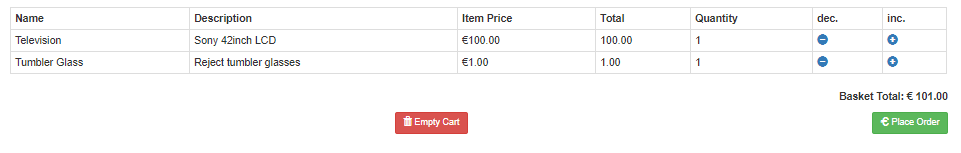


If you click the **Sign Out** menu option, you should see the following:



**Place an Order**

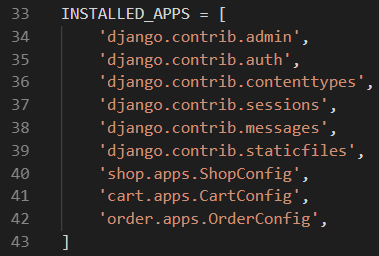
When the user adds items to the shopping cart, the cart is displayed as shown below. We now need to add functionality to the website for the **Place Order** button



We will place the functionality for orders in a new app called **order**. Type the following command at he command line in order to create a new **app** called **order**

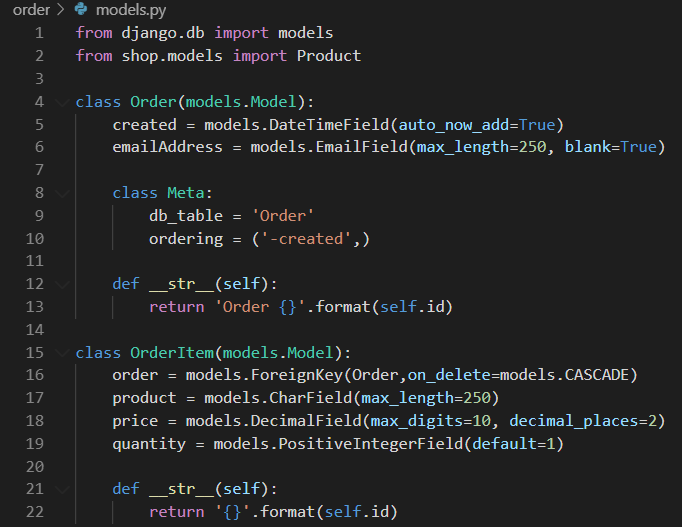


Edit settings.py file to let Django know about the new app:



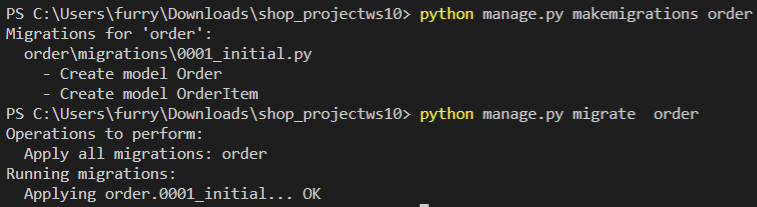
**Database Model**

Open the **order/models.py** file and add the following code to create the database model:

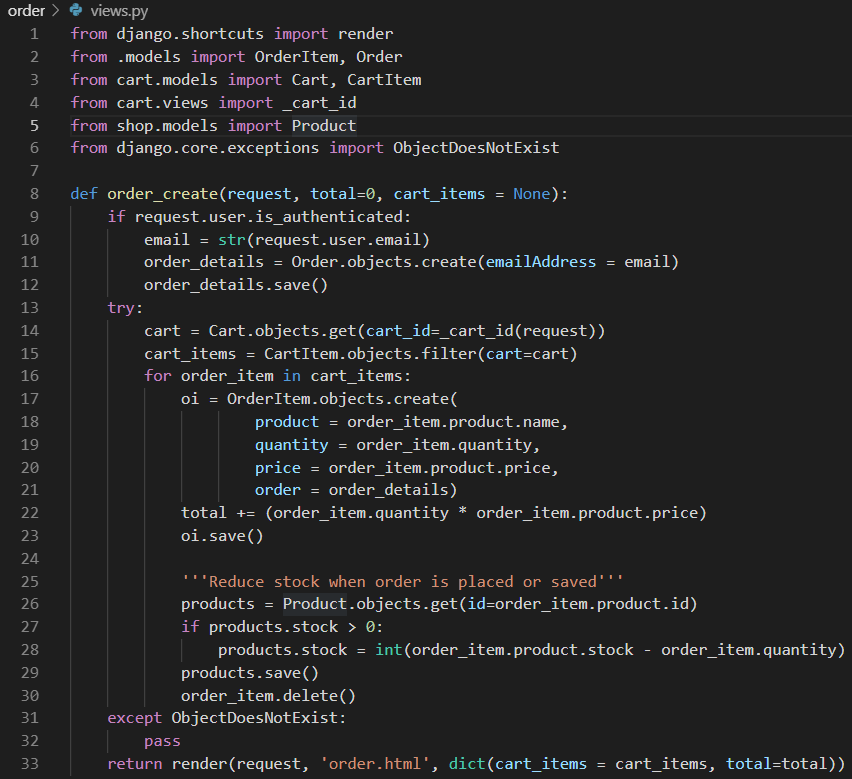


In order to activate the new database model type in the following commands:

**Command Line**

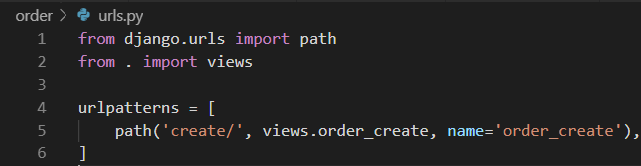


Open the **order/views.py** file and add the following code to create:

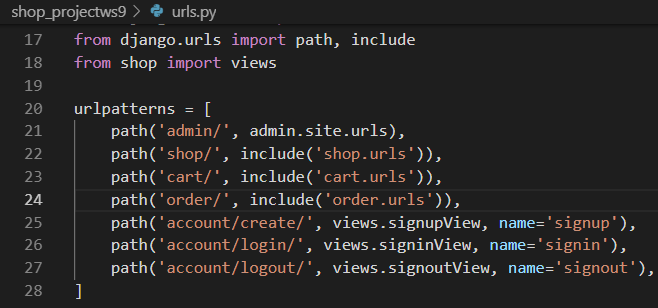


**URLs**

Create a new file **urls.py** inside the **order** app and use the following code to set the paths

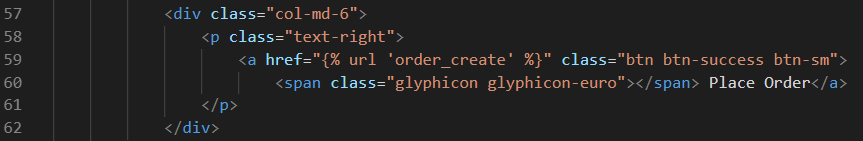


Open the **shop\_projectws9/urls.py** file and add in line 24:



**Templates – cart**

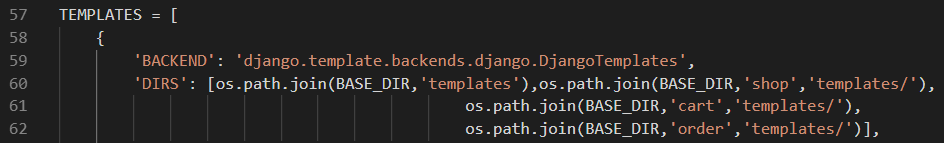
Open **cart.html** and add the following link



**Templates – order**

Create a **templates** folder inside the **order** app

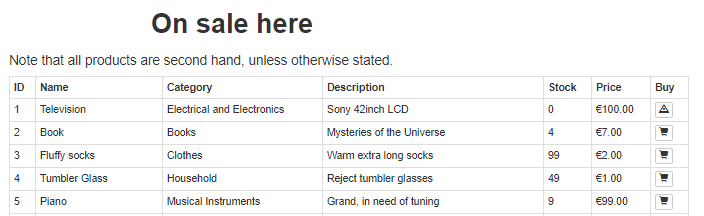
Edit the **settings.py** file to include this **templates** folder as shown below:



Restart the server and add some items to the cart and press the **Place Order** button. You should see a screen like the following:



**Out of Stock**

If an item is out of stock, the user should not be able to click on the trolley icon to add it to the shopping cart. Instead we will display a different image in the table as shown below: 

Open the file **products.html** and change the code as shown below:

