

## Lab 10 Part 11

### Voucher Codes

Many online shops give out voucher codes to customers that can be redeemed for discounts on their purchases. A voucher code is usually valid for a specific time frame. The code can be redeemed one or multiple times.

We are going to create a voucher code system for our shop. Our voucher codes will be valid for clients that enter the voucher in a specific time frame. The voucher codes will not have any limitations in terms of the number of times they can be redeemed, and they will be applied to the total value of the shopping cart. For this functionality, we will need to create a model to store the voucher code, a valid time frame, and the discount to apply.

#### Step 1: Database Model

Create a new app called **vouchers** inside the project using the following command:

---

```
python manage.py startapp vouchers
```

---

Edit the **settings.py** file of shop\_project and add the **vouchers** application to the **INSTALLED\_APPS** setting.

Let's start by creating the **Voucher** model. Edit the **models.py** file of the vouchers application and copy-paste the following code:

```
from django.core.validators import MinValueValidator, MaxValueValidator

class Voucher(models.Model):
    code = models.CharField(max_length=50, unique=True)
    valid_from = models.DateTimeField()
    valid_to = models.DateTimeField()
    discount = models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(100)])
    active = models.BooleanField()

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

This is the model that we are going to use to store voucher codes. The Voucher model contains the following fields:

- code: The code that users must enter to apply the voucher code to their purchase.
- valid\_from: The datetime value that indicates when the voucher code becomes valid.
- valid\_to: The datetime value that indicates when the voucher code becomes invalid.
- discount: The discount rate to apply (this is a percentage, so it takes values from 0 to 100). We use validators for this field to limit the minimum and maximum accepted values.
- active: A Boolean that indicates whether the voucher code is active.

Run the following command to generate the initial migration for the vouchers application:

---

```
python manage.py makemigrations vouchers
```

---

The output should include the following lines:

```
Migrations for 'vouchers':
  vouchers\migrations\0001_initial.py
  - Create model Voucher
```

Then, we execute the next command to apply migrations:

---

```
python manage.py migrate
```

---

You should see an output that includes the following line:

```
Applying vouchers.0001_initial... OK
```

The migrations are now applied in the database.

## Step 2 Django Admin

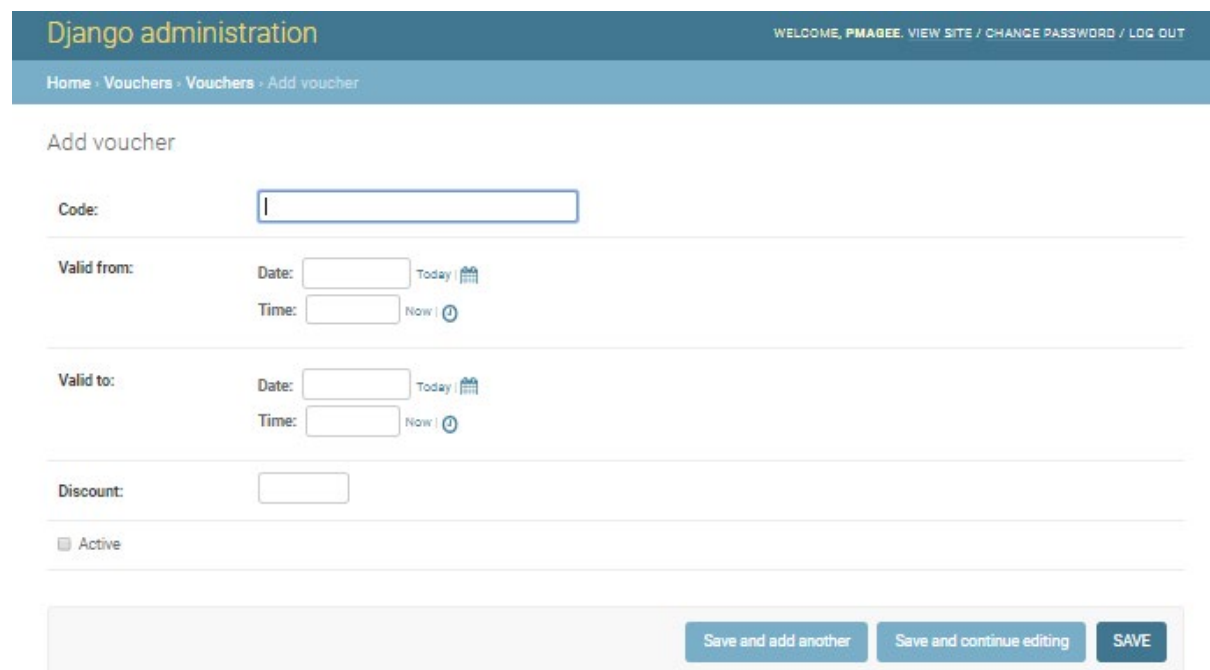
Let's add the Voucher model to the administration site. Edit the **admin.py** file of the **vouchers** application and copy-paste the following code:

```
from .models import Voucher

class VoucherAdmin(admin.ModelAdmin):
    list_display = ['code', 'valid_from', 'valid_to', 'discount', 'active']
    list_filter = ['active', 'valid_from', 'valid_to']
    search_fields = ['code']

admin.site.register(Voucher, VoucherAdmin)
```

The Voucher model is now registered in the administration site. Run your local server and log into Django Admin. You should now see the Vouchers model listed, click on it, and then click “Add Voucher” and you should see the following form:



The screenshot shows the Django Admin interface for adding a new voucher. The header includes the Django Admin logo and user information. The breadcrumb trail is 'Home > Vouchers > Vouchers > Add voucher'. The form is titled 'Add voucher' and contains the following fields:

- Code:** A text input field.
- Valid from:** A date and time selection section. The 'Date' field has a 'Today' button and a calendar icon. The 'Time' field has a 'Now' button and a clock icon.
- Valid to:** A date and time selection section. The 'Date' field has a 'Today' button and a calendar icon. The 'Time' field has a 'Now' button and a clock icon.
- Discount:** A text input field.
- Active:** A checkbox labeled 'Active'.

At the bottom right of the form are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

Fill in the form to create a new coupon that is valid for the current date and make sure that you check the Active checkbox and click the SAVE button.

### Step 3: Applying a voucher to the shopping cart

We can store new vouchers and make queries to retrieve existing vouchers. Now we need a way for customers to apply voucher codes to their purchases. The functionality to apply a voucher code would be as follows:

1. The user adds products to the shopping cart.
2. The user can enter a voucher code in a form displayed in the shopping cart detail page.
3. When a user enters a voucher code and submits the form, we look for an existing voucher with the given code that is currently valid. We must check that the voucher code matches the one entered by the user and that the active attribute is True, and that the current datetime is between the `valid_from` and `valid_to` values.
4. If a voucher is found, we save it in the user's session and display the cart, including the discount applied to it and the updated total amount.
5. When the user places an order, we save the voucher to the given order.

Create a new file inside the **vouchers** application directory and name it **forms.py**.

Copy-paste the following code to it:

```
from django import forms

class VoucherApplyForm(forms.Form):
    code = forms.CharField()
```

This is the form that we are going to use for the user to enter a voucher code. Edit the **views.py** file inside the **vouchers** application, delete the top line of code and copy-paste the following code to it:

```
from django.shortcuts import redirect
from django.utils import timezone
from django.views.decorators.http import require_POST
from .models import Voucher
from .forms import VoucherApplyForm

def voucher_apply(request):
    now = timezone.now()
    form = VoucherApplyForm(request.POST)
    if form.is_valid():
        code = form.cleaned_data['code']
        try:
            voucher = Voucher.objects.get(code__iexact=code,
valid_from__lte=now, valid_to__gte=now, active=True)
            request.session['voucher_id'] = voucher.id
        except Voucher.DoesNotExist:
            request.session['voucher_id'] = None
```

```
return redirect('cart:cart_detail')
```

The **voucher\_apply** view validates the voucher and stores it in the user's session. In the view, we perform the following tasks:

1. We instantiate the VoucherApplyForm form using the posted data and we check that the form is valid.
2. If the form is valid, we get the code entered by the user from the form's `cleaned_data` dictionary. We try to retrieve the Voucher object with the given code. We use the `iexact` field lookup to perform a case-insensitive exact match. The coupon must be currently active (`active=True`) and valid for the current datetime. We use Django's `timezone.now()` function to get the current datetime and we compare it with the `valid_from` and `valid_to` fields performing `lte` (less than or equal to) and `gte` (greater than or equal to) field lookups, respectively.
3. We store the voucher ID in the user's session.
4. We redirect the user to the `cart_detail` URL to display the cart with the voucher applied.

#### Step 4: URLs

We need a URL pattern for the **voucher\_apply** view. Create a new file inside the **vouchers** application directory and name it **urls.py**.

Copy-paste the following code to it:

```
from django.urls import path
from . import views

app_name = 'vouchers'

urlpatterns = [
    path('apply/', views.voucher_apply, name='apply'),
]
```

Then, edit the main `urls.py` of **onlineshop** and include the vouchers URL patterns as follows:

```
13 | path('vouchers/', include('vouchers.urls')),
```

## Step 5: Applying vouchers to orders

We are going to store the voucher that may be applied to each order.

First, we need to modify the **Order** model to store the related **Voucher** object, if there is any.

Edit the **models.py** file of the **orders** application and copy-paste the following imports to it:

```
from django.core.validators import MinValueValidator, MaxValueValidator
from vouchers.models import Voucher
```

Then, copy-paste the following fields to the **Order** model just after the **shippingCountry** field:

```
    voucher = models.ForeignKey(Voucher,
                               related_name='orders',
                               null=True,
                               blank=True,
                               on_delete=models.SET_NULL)
    discount = models.IntegerField(default = 0,
                                   validators=[MinValueValidator(0),
                                               MaxValueValidator(100)])
```

These fields allow us to store an optional voucher for the order and the discount percentage applied with the voucher. The discount is stored in the related Voucher object, but we include it in the Order model to preserve it if the voucher is modified or deleted.

We set `on_delete` to `models.SET_NULL` so that if the voucher gets deleted, the voucher field is set to Null.

## Step 6: Migrations

We need to create a migration to include the new fields of the Order model. Run the following command from the command line:

```
python manage.py makemigrations order
```

Apply the new migration with the following command:

```
python manage.py migrate
```

## Step 7: Edit the Cart View

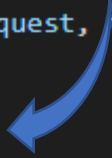
Now, edit the **cart/views.py** file

Copy-paste the following imports:

```
from vouchers.models import Voucher
from vouchers.forms import VoucherApplyForm
from decimal import Decimal
```

Copy-paste the following code to the start of the **cart\_detail** view function see location below:

```
discount = 0
voucher_id = 0
new_total = 0
voucher = None
```




```
42 def cart_detail(request, total=0, counter=0, cart_items = None):
43     discount =0
44     voucher_id=0
45     new_total=0
46     voucher=None
47
48     try:
```

Copy-paste the following code to the location shown below:

```
voucher_apply_form = VoucherApplyForm()

try:
    voucher_id = request.session.get('voucher_id')
    voucher = Voucher.objects.get(id=voucher_id)
    if Voucher != None:
        discount = (total*(voucher.discount/Decimal('100')))
        new_total = (total - discount)
        stripe_total = int(new_total * 100)
except:
    ObjectDoesNotExist
    pass
```



```
59     data_key = settings.STRIPE_PUBLISHABLE_KEY
60     voucher_apply_form = VoucherApplyForm()
61
62     try:
63         voucher_id = request.session.get('voucher_id')
64         voucher = Voucher.objects.get(id=voucher_id)
65         if voucher !=None:
66             discount = (total*(voucher.discount/Decimal('100')))
67             new_total = (total - discount)
68             stripe_total = int(new_total * 100)
69     except:
70         ObjectDoesNotExist
71         pass
```

Line 60: We add the voucher form so that we can pass it into the template as part of the context data

Line 63: We try to get the **voucher\_id** session key from the current session and store its value in the **voucher\_id** variable

Line 64: Using the id we retrieve the voucher object from the database if it exists


Line 65-68: If the voucher exists then we calculate the discount. The standard division symbol '/' rounds to zero so we get the decimal result by using the Decimal module. We calculate the new total by subtracting the discount from the total and we also assign this new total to the stripe\_total variable.

Lines 69-71: If the voucher doesn't exist then the exception ObjectDoesNotExist is thrown.



Add the following code to the **cart\_detail** view function at the location shown:

```
if voucher != None:
    order_details.total = new_total
    order_details.voucher = voucher
    order_details.discount = discount
    order_details.save()
```




```
99         order_details.save()
100         if voucher != None:
101             order_details.total = new_total
102             order_details.voucher = voucher
103             order_details.discount = discount
104             order_details.save()
105     for order_item in cart_items:
```

Lines 100-104: If there is a voucher code then we need to apply this to the order by assigning the new discounted total to the order total variable, and then setting the two new Order model fields, voucher, and discount. We then call the save() function again to save the Order object.

Add the following code to the **cart\_detail** view function at the location shown:

```
if voucher != None:
    discount =
(oι.price*(voucher.discount/Decimal('100'))))
    oi.price = (oi.price - discount)
else:
    oi.price = oi.price*oi.quantity
```



```
110         order = order details)
111         if voucher != None:
112             discount = (oi.price*(voucher.discount/Decimal('100'))))
113             oi.price = (oi.price - discount)
114         else:
115             oi.price = oi.price*oi.quantity
116         oi.save
117         '''Reduce stock when order is placed or saved'''
```

Lines 111-115: We do something similar for the Order Item so that the discounted price, if there is a voucher, is set.

Add the following data to the render function.

```
return render(request, 'cart.html', {'cart_items':cart_items, 'total':total, 'counter':counter,
                                     'data_key': data_key, 'stripe_total':stripe_total,
                                     'description': description, 'voucher_apply_form': voucher_apply_form,
                                     'new_total': new_total, 'voucher':voucher, 'discount':discount
                                     })
```

## Step 8: Edit the Cart Template

We now need to modify our **cart.html** template so that it displays the voucher form and then also displays the discount and new total for the order.

Open **cart.html** and delete the following block of code:

```
90         <td>
91             Please review your shopping cart items before proceeding with the order payment.
92         </td>
93     </tr>
94     <tr>
95         <td class="text-left">
96             Your total is: <strong>€{{ total }}</strong>
97         </td>
98     </tr>
99 </tbody>
```

Copy-paste the following code into the **cart.html** file at the location where you just deleted the above block of code.

```
{% if voucher %}

    <tr class="subtotal">
        <td>Total</td>
        <td colspan="4"></td>
        <td class="num">€{{ total|floatformat:"2" }}</td>
    </tr>
    <tr>
        <td>
            "{{ voucher.code }}" voucher
            ({{ voucher.discount }}% off)
        </td>
        <td colspan="4"></td>
        <td class="num neg">
            - €{{ discount|floatformat:"2" }}
        </td>
    </tr>
    <tr class="total">
        <td>Total</td>
        <td colspan="4"></td>
        <td class="num">
            €{{ new_total|floatformat:"2" }}
        </td>
    </tr>
{% else %}
    <tr class="total">
        <td>Total</td>
        <td colspan="4"></td>
        <td class="num">
            €{{ total|floatformat:"2" }}
        </td>
    </tr>
</tbody>
```

```

        {% endif %}
    </tbody>
</table>
<p>Apply a voucher:</p>
<form action="{% url 'vouchers:apply' %}" method="post">
    {{ voucher_apply_form }}
    <input type="submit" value="Apply">
    {% csrf_token %}
</form>

```

The if else block of code that you just copied into **cart.html** will display the voucher details, discount, and new total if a voucher exists as shown below:

CHECKOUT	
Please review your shopping cart items before proceeding with the order payment.	
Total	€50.00
"NOV2022" voucher (10% off)	- €5.00
Total	€45.00
Apply a voucher:	
Code:	<input type="text"/> <input type="button" value="Apply"/>

The code below is a section of the code that you just copied in to **cart.html**. This will display the form to enter a voucher code and apply it to the current cart.

```

73     </table>
74     <p>Apply a voucher:</p>
75     <form action="{% url 'vouchers:apply' %}" method="post">
76         {{ voucher_apply_form }}
77         <input type="submit" value="Apply">
78         {% csrf_token %}
79     </form>

```

This will display the form to enter a voucher code and apply it to the current cart.

Apply a voucher:	
Code:	<input type="text"/> <input type="button" value="Apply"/>

**Step 9: Try it out**

Open <http://127.0.0.1:8080/> in your browser, add some products to the cart, and apply the voucher you created by entering its code in the form.

**Step 10:**

Use the following command to generate a Python requirements.txt file:

**pip freeze > requirements.txt**

**Step 11:**

Push your code to the Lab 10 repo