Coverage for **invoices/views.py**: 53%

95 statements | 50 run | 45 missing | 0 excluded

```
1  """ View information for invoicing pages """
2
3  import datetime
4  import stripe
5  from django.conf import settings
6  from django.shortcuts import render, redirect, get_object_or_404
7  from django.contrib import messages
8  from django.contrib.auth.decorators import login_required
9  from profiles.models import UserProfile
10 from subscriptions.models import StripeCustomer
11 from .models import InvoiceCustomer, Invoice
12 from .forms import InvoiceCustomerForm, InvoiceForm
13
14 # Create your views here.
15
16
17 @login_required()
18 def dashboard(request):
19     """ View to return dashboard page """
20
21     # Get user and create invoices and customers, check subscription
22     user = request.user
23     invoices = Invoice.objects.filter(
24         user_id__exact=user).select_related('customer_code')
25     customers = InvoiceCustomer.objects.filter(user_id__exact=user)
26     subscribed = StripeCustomer.objects.filter(user=user).exists()
27
28     # Function to make Stripe date values into human date
29     def make_date(date_value):
30         """ Convert Stripe value to user friendly date """
31         nice_date = datetime.datetime.fromtimestamp(
32             date_value).strftime('%d-%m-%Y')
33         return nice_date
34
35     try:
36         # Retrieve the subscription & product for Dashboard
37         stripe_customer = StripeCustomer.objects.get(user=request.user)
38         stripe.api_key = settings.STRIPE_SECRET_KEY
39         subscription = stripe.Subscription.retrieve(
40             stripe_customer.stripeSubscriptionId)
41         subscription_start = make_date(subscription.current_period_start)
42         subscription_end = make_date(subscription.current_period_end)
43         product = stripe.Product.retrieve(subscription.plan.product)
44         cancelled = subscription.cancel_at_period_end
45
46         context = {
47             'subscription': subscription,
```

```
48              'product': product,
49              'subscription_start': subscription_start,
50              'subscription_end': subscription_end,
51              'user': user,
52              'invoices': invoices,
53              'customers': customers,
54              'cancelled': cancelled,
55              'subscribed': subscribed,
56          }
57
58      # Set the subscription for Dashboard where no subs in place
59      except StripeCustomer.DoesNotExist:
60          subscription = 'No current subscription'
61          context = {
62              'subscription': subscription,
63              'user': user,
64              'invoices': invoices,
65              'customers': customers,
66              'subscribed': subscribed,
67          }
68
69      template = 'invoices/dashboard.html'
70
71      return render(request, template, context)
72
73
74  @login_required()
75  def customer(request, customer_id):
76      """ View to return customer form """
77
78      subscribed = StripeCustomer.objects.filter(user=request.user).exists()
79      if subscribed is False:
80          messages.info(
81              request,
82              'You cannot add new customers as you do not have a subscription'
83          )
84          return redirect('/invoices/')
85
86      if request.method == 'POST':
87          # If customer id == 0 save new instance
88          if customer_id == '0':
89              form = InvoiceCustomerForm(request.POST)
90          # Otherwise update existing customer
91          else:
92              this_customer = InvoiceCustomer.objects.get(id=customer_id)
93              form = InvoiceCustomerForm(request.POST, instance=this_customer)
94          if form.is_valid():
95              hold_form = form.save(commit=False)
96              hold_form.user = request.user
97              hold_form.save()
98              messages.success(request, 'Customer updated successfully!')
99
100     # If customer id == 0 render blank form
```

```
101     if customer_id == '0':
102         form = InvoiceCustomerForm()
103     # Else render form with customer details
104     else:
105         this_customer = InvoiceCustomer.objects.get(id=customer_id)
106         form = InvoiceCustomerForm(instance=this_customer)
107 
108     template = 'invoices/customer_form.html'
109     context = {
110         'form': form,
111         'customer_id': customer_id,
112     }
113     return render(request, template, context)
114 
115 
116 @login_required()
117 def invoice(request, invoice_id):
118     """ View to return invoice form """
119 
120     subscribed = StripeCustomer.objects.filter(user=request.user).exists()
121     if subscribed is False:
122         messages.info(
123             request,
124             'You cannot add new invoices as you do not have a subscription'
125         )
126         return redirect('/invoices/')
127 
128     if request.method == 'POST':
129         # If invoice id == 0 save new instance
130         if invoice_id == '0':
131             form = InvoiceForm(request.POST, user=request.user)
132         # Otherwise update existing invoice
133         else:
134             this_invoice = Invoice.objects.get(id=invoice_id)
135             form = InvoiceForm(
136                 request.POST,
137                 instance=this_invoice,
138                 user=request.user
139             )
140 
141         if form.is_valid():
142             hold_form = form.save(commit=False)
143 
144             # If VAT registered, calculate VAT and gross
145             subscriber = get_object_or_404(UserProfile, user=request.user)
146             if subscriber.vat_number:
147                 vat = hold_form.invoice_subtotal * 0.2
148                 gross = hold_form.invoice_subtotal + vat
149             # Otherwise set VAT to nil and gross to subtotal value
150             else:
151                 vat = 0.00
152                 gross = hold_form.invoice_subtotal
153 
```

```
154             hold_form.invoice_vat = vat
155             hold_form.invoice_gross = gross
156             hold_form.user = request.user
157             hold_form.save()
158             messages.success(request, 'Invoice updated successfully!')
159 
160     # If invoice id == 0 render blank form
161     if invoice_id == '0':
162         form = InvoiceForm(user=request.user)
163     # Else render form with invoice details
164     else:
165         this_invoice = Invoice.objects.get(id=invoice_id)
166         form = InvoiceForm(instance=this_invoice, user=request.user)
167 
168     template = 'invoices/invoice_form.html'
169     context = {
170         'form': form,
171         'invoice_id': invoice_id,
172     }
173     return render(request, template, context)
174 
175 
176 @login_required()
177 def delete_invoice(request, invoice_id):
178     """ View to delete an invoice """
179     Invoice.objects.filter(id=invoice_id).delete()
180     return redirect('/invoices/')
181 
182 
183 @login_required()
184 def delete_customer(request, customer_id):
185     """ View to delete a customer and all invoices """
186     InvoiceCustomer.objects.filter(id=customer_id).delete()
187     return redirect('/invoices/')
```