Coverage for **invoices/views.py**: 48%

122 statements | 58 run | 64 missing | 0 excluded

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

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

```
101     # If customer id == 0 render blank form
102     if customer_id == '0':
103         form = InvoiceCustomerForm()
104     # Else render form with customer details
105     else:
106         this_customer = InvoiceCustomer.objects.get(id=customer_id)
107         form = InvoiceCustomerForm(instance=this_customer)
108 
109     template = 'invoices/customer_form.html'
110     context = {
111         'form': form,
112         'customer_id': customer_id,
113     }
114     return render(request, template, context)
115 
116 
117 @login_required()
118 def invoice(request, invoice_id):
119     """ View to return invoice form """
120     # Prevent non-owners from accessing material
121     if invoice_id != '0':
122         user_id = request.user.id
123         owner_id = Invoice.objects.get(id=invoice_id).user_id
124         if user_id != owner_id:
125             messages.info(
126                 request,
127                 'You cannot access this item.'
128             )
129             return redirect('/invoices/')
130 
131     subscribed = StripeCustomer.objects.filter(user=request.user).exists()
132     if subscribed is False:
133         messages.info(
134             request,
135             'You cannot add new invoices as you do not have a subscription'
136         )
137         return redirect('/invoices/')
138 
139     if request.method == 'POST':
140         # If invoice id == 0 save new instance
141         if invoice_id == '0':
142             form = InvoiceForm(request.POST, user=request.user)
143         # Otherwise update existing invoice
144         else:
145             this_invoice = Invoice.objects.get(id=invoice_id)
146             form = InvoiceForm(
147                 request.POST,
148                 instance=this_invoice,
149                 user=request.user
150             )
151 
152         if form.is_valid():
153             hold_form = form.save(commit=False)
```

```
154
155             # If VAT registered, calculate VAT and gross
156             subscriber = get_object_or_404(UserProfile, user=request.user)
157             if subscriber.vat_number:
158                 vat = hold_form.invoice_subtotal * 0.2
159                 gross = hold_form.invoice_subtotal + vat
160             # Otherwise set VAT to nil and gross to subtotal value
161             else:
162                 vat = 0.00
163                 gross = hold_form.invoice_subtotal
164
165             hold_form.invoice_vat = vat
166             hold_form.invoice_gross = gross
167             hold_form.user = request.user
168             hold_form.save()
169             messages.success(request, 'Invoice updated successfully!')
170
171     # If invoice id == 0 render blank form
172     if invoice_id == '0':
173         form = InvoiceForm(user=request.user)
174     # Else render form with invoice details
175     else:
176         this_invoice = Invoice.objects.get(id=invoice_id)
177         form = InvoiceForm(instance=this_invoice, user=request.user)
178
179     template = 'invoices/invoice_form.html'
180     context = {
181         'form': form,
182         'invoice_id': invoice_id,
183     }
184     return render(request, template, context)
185
186
187 @login_required()
188 def delete_invoice(request, invoice_id):
189     """ View to delete an invoice """
190     # Prevent non-owners from accessing material
191     user_id = request.user.id
192     owner_id = Invoice.objects.get(id=invoice_id).user_id
193     if user_id != owner_id:
194         messages.info(
195             request,
196             'You cannot access this item.'
197         )
198         return redirect('/invoices/')
199
200     subscribed = StripeCustomer.objects.filter(user=request.user).exists()
201     if subscribed is False:
202         messages.info(
203             request,
204             'You cannot delete invoices as you do not have a subscription'
205         )
206         return redirect('/invoices/')
```

```
207
208     Invoice.objects.filter(id=invoice_id).delete()
209     return redirect('/invoices/')
210
211
212 @login_required()
213 def delete_customer(request, customer_id):
214     """ View to delete a customer and all invoices """
215     # Prevent non-owners from accessing material
216     user_id = request.user.id
217     owner_id = InvoiceCustomer.objects.get(id=customer_id).user_id
218     if user_id != owner_id:
219         messages.info(
220             request,
221             'You cannot access this item.'
222         )
223         return redirect('/invoices/')
224
225     subscribed = StripeCustomer.objects.filter(user=request.user).exists()
226     if subscribed is False:
227         messages.info(
228             request,
229             'You cannot delete customers as you do not have a subscription'
230         )
231         return redirect('/invoices/')
232
233     InvoiceCustomer.objects.filter(id=customer_id).delete()
234     return redirect('/invoices/')
```