Coverage for **subscriptions/views.py**: 32%

92 statements | 29 run | 63 missing | 0 excluded

```
1  """ View information for subscription pages """
2  # From https://testdriven.io/blog/django-stripe-subscriptions/
3
4  import datetime
5  import stripe
6  from django.conf import settings
7  from django.contrib import messages
8  from django.contrib.auth.decorators import login_required
9  from django.contrib.auth.models import User
10 from django.http.response import JsonResponse, HttpResponse
11 from django.shortcuts import render, redirect
12 from django.views.decorators.csrf import csrf_exempt
13 from profiles.models import UserProfile
14 from .models import StripeCustomer
15
16 # Create your views here.
17
18
19 @login_required
20 def subscribe(request):
21     """ View to return checkout page """
22
23     def make_date(date_value):
24         """ Convert Stripe value to user friendly date """
25         nice_date = datetime.datetime.fromtimestamp(date_value).strftime(
26             '%d-%m-%Y')
27         return nice_date
28
29     try:
30         # Retrieve the subscription & product
31         stripe_customer = StripeCustomer.objects.get(user=request.user)
32         stripe.api_key = settings.STRIPE_SECRET_KEY
33         subscription = stripe.Subscription.retrieve(
34             stripe_customer.stripeSubscriptionId)
35         subscription_start = make_date(subscription.current_period_start)
36         subscription_end = make_date(subscription.current_period_end)
37         product = stripe.Product.retrieve(subscription.plan.product)
38         cancelled = subscription.cancel_at_period_end
39
40         # https://stripe.com/docs/api/subscriptions/object
41         # https://stripe.com/docs/api/products/object
42
43         context = {
44             'subscription': subscription,
45             'product': product,
46             'subscription_start': subscription_start,
47             'subscription_end': subscription_end,
48             'cancelled': cancelled,
49         }
50
51         return render(request, 'subscriptions/subscribe.html', context)
52
53     # Show checkout page if not already subscribed
54     except StripeCustomer.DoesNotExist:
55
```

```
56          plan_name = request.session.__getitem__('plan_name')
57
58          context = {
59              'plan_name': plan_name,
60          }
61
62          return render(request, 'subscriptions/subscribe.html', context)
63
64
65  @csrf_exempt
66  def stripe_config(request):
67      """ Handle Stripe AJAX request """
68      if request.method == 'GET':
69          stripe_configs = {'publicKey': settings.STRIPE_PUBLIC_KEY}
70          return JsonResponse(stripe_configs, safe=False)
71
72
73  @csrf_exempt
74  def create_checkout_session(request):
75      """ Create a checkout session """
76      if request.method == 'GET':
77          domain_url = settings.DOMAIN_URL
78          stripe.api_key = settings.STRIPE_SECRET_KEY
79          plan_stripe_id = request.session.__getitem__('plan_stripe_id')
80          customer = UserProfile.objects.get(user=request.user)
81          try:
82              checkout_session = stripe.checkout.Session.create(
83                  client_reference_id=request.user.id if request.user.is_authenticated else None,
84                  success_url=domain_url + 'subscriptions/success?session_id={CHECKOUT_SESSION_ID}'
85                  cancel_url=domain_url + 'subscriptions/abort/',
86                  payment_method_types=['card'],
87                  mode='subscription',
88                  customer_email=customer.email,
89                  line_items=[
90                      {
91                          'price': plan_stripe_id,
92                          'quantity': 1,
93                      }
94                  ]
95              )
96              return JsonResponse({'sessionId': checkout_session['id']})
97          except Exception as e:
98              return JsonResponse({'error': str(e)})
99
100
101 @login_required
102 def success(request):
103     """ Return page for successful subscription """
104     return render(request, 'subscriptions/success.html')
105
106
107 @login_required
108 def abort(request):
109     """ Return page for aborted subscription """
110     return render(request, 'subscriptions/abort.html')
111
112
113 @csrf_exempt
114 def stripe_webhook(request):
115     """ Create new StripeCustomer on subscription """
```

```
116     stripe.api_key = settings.STRIPE_SECRET_KEY
117     endpoint_secret = settings.STRIPE_ENDPOINT_SECRET
118     payload = request.body
119     sig_header = request.META['HTTP_STRIPE_SIGNATURE']
120     event = None
121 
122     try:
123         event = stripe.Webhook.construct_event(
124             payload, sig_header, endpoint_secret
125         )
126     except ValueError as e:
127         # Invalid payload
128         return HttpResponse(status=400)
129     except stripe.error.SignatureVerificationError as e:
130         # Invalid signature
131         return HttpResponse(status=400)
132 
133     # Handle the checkout.session.completed event
134     if event['type'] == 'checkout.session.completed':
135         session = event['data']['object']
136 
137         # Fetch all the required data from session
138         client_reference_id = session.get('client_reference_id')
139         stripe_customer_id = session.get('customer')
140         stripe_subscription_id = session.get('subscription')
141 
142         # Get the user and create a new StripeCustomer
143         user = User.objects.get(id=client_reference_id)
144         StripeCustomer.objects.create(
145             user=user,
146             stripeCustomerId=stripe_customer_id,
147             stripeSubscriptionId=stripe_subscription_id,
148         )
149         print(user.username + ' just subscribed.')
150 
151     return HttpResponse(status=200)
152 
153 
154 @login_required()
155 def cancel(request):
156     """ Cancel subscription effective from end of current period """
157     stripe_customer = StripeCustomer.objects.get(user=request.user)
158     stripe.api_key = settings.STRIPE_SECRET_KEY
159     subscription = stripe_customer.stripeSubscriptionId
160 
161     if request.method == 'POST':
162         stripe.Subscription.modify(
163             subscription,
164             cancel_at_period_end=True
165         )
166 
167         return redirect('/invoices/')
168 
169     return render(request, 'subscriptions/cancel.html')
170 
171 
172 @login_required()
173 def reactivate(request):
174     """ Reactivate subscription if still valid """
175     stripe_customer = StripeCustomer.objects.get(user=request.user)
```

```
176     stripe.api_key = settings.STRIPE_SECRET_KEY
177     subscription = stripe_customer.stripeSubscriptionId
178 
179     stripe.Subscription.modify(
180         subscription,
181         cancel_at_period_end=False
182     )
183 
184     messages.success(request, 'Subscription reactivated successfully!')
185 
186     return redirect('/invoices/')
```

« index coverage.py v6.3.3, created at 2022-05-21 11:31 +0000