Coverage for **subscriptions/views.py**: 33%

100 statements | 33 run | 67 missing | 0 excluded

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

```
48          plan_name = request.session.__getitem__('plan_name')
49          user = request.user.id
50
51          context = {
52              'plan_name': plan_name,
53              'user': user,
54          }
55
56          return render(request, 'subscriptions/subscribe.html', context)
57
58
59  # https://testdriven.io/blog/django-stripe-subscriptions/
60  @csrf_exempt
61  def stripe_config(request):
62      """ Handle Stripe AJAX request """
63      if request.method == 'GET':
64          stripe_configs = {'publicKey': settings.STRIPE_PUBLIC_KEY}
65          return JsonResponse(stripe_configs, safe=False)
66
67
68  # https://testdriven.io/blog/django-stripe-subscriptions/
69  @csrf_exempt
70  def create_checkout_session(request):
71      """ Create a checkout session """
72      if request.method == 'GET':
73          domain_url = settings.DOMAIN_URL
74          stripe.api_key = settings.STRIPE_SECRET_KEY
75          plan_stripe_id = request.session.__getitem__('plan_stripe_id')
76          customer = UserProfile.objects.get(user=request.user)
77          try:
78              checkout_session = stripe.checkout.Session.create(
79                  client_reference_id=(request.user.id if
80                                       request.user.is_authenticated else None),
81                  success_url=domain_url + 'subscriptions/success'
82                                           '?session_id={CHECKOUT_SESSION_ID}',
83                  cancel_url=domain_url + 'subscriptions/abort/',
84                  payment_method_types=['card'],
85                  mode='subscription',
86                  customer_email=customer.email,
87                  line_items=[
88                      {
89                          'price': plan_stripe_id,
90                          'quantity': 1,
91                      }
92                  ]
93              )
94              return JsonResponse({'sessionId': checkout_session['id']})
95          except Exception as e:
96              return JsonResponse({'error': str(e)})
97
98
99  # https://testdriven.io/blog/django-stripe-subscriptions/
100 @login_required
```

```
101 def success(request):
102     """ Return page for successful subscription """
103     if not (subscribed := StripeCustomer.objects.filter(
104             user=request.user).exists()):
105         messages.info(
106             request,
107             'You cannot view this page as you do not have a subscription'
108         )
109         return redirect('/invoices/')
110     return render(request, 'subscriptions/success.html')
111 
112 
113 # https://testdriven.io/blog/django-stripe-subscriptions/
114 @login_required
115 def abort(request):
116     """ Return page for aborted subscription """
117     return render(request, 'subscriptions/abort.html')
118 
119 
120 # https://testdriven.io/blog/django-stripe-subscriptions/
121 @csrf_exempt
122 def stripe_webhook(request):
123     """ Create new StripeCustomer on subscription """
124     stripe.api_key = settings.STRIPE_SECRET_KEY
125     endpoint_secret = settings.STRIPE_ENDPOINT_SECRET
126     payload = request.body
127     sig_header = request.META['HTTP_STRIPE_SIGNATURE']
128     event = None
129 
130     try:
131         event = stripe.Webhook.construct_event(
132             payload, sig_header, endpoint_secret
133         )
134     except ValueError as e:
135         # Invalid payload
136         return HttpResponse(status=400)
137     except stripe.error.SignatureVerificationError as e:
138         # Invalid signature
139         return HttpResponse(status=400)
140 
141     # Handle the checkout.session.completed event
142     if event['type'] == 'checkout.session.completed':
143         session = event['data']['object']
144 
145         # Fetch all the required data from session
146         client_reference_id = session.get('client_reference_id')
147         stripe_customer_id = session.get('customer')
148         stripe_subscription_id = session.get('subscription')
149 
150         # Get the user and create a new StripeCustomer
151         user = User.objects.get(id=client_reference_id)
152         StripeCustomer.objects.create(
153             user=user,
```

```
154             stripeCustomerId=stripe_customer_id,
155             stripeSubscriptionId=stripe_subscription_id,
156         )
157 
158     return HttpResponse(status=200)
159 
160 
161 @login_required()
162 def cancel(request):
163     """ Cancel subscription effective from end of current period """
164     if not (subscribed := StripeCustomer.objects.filter(
165             user=request.user).exists()):
166         messages.info(
167             request,
168             'You cannot view this page as you do not have a subscription'
169         )
170         return redirect('/invoices/')
171 
172     stripe_customer = StripeCustomer.objects.get(user=request.user)
173     stripe.api_key = settings.STRIPE_SECRET_KEY
174     subscription = stripe_customer.stripeSubscriptionId
175 
176     if request.method == 'POST':
177         stripe.Subscription.modify(
178             subscription,
179             cancel_at_period_end=True
180         )
181 
182         return redirect('/invoices/')
183 
184     return render(request, 'subscriptions/cancel.html')
185 
186 
187 @login_required()
188 def reactivate(request):
189     """ Reactivate subscription if still valid """
190     stripe_customer = StripeCustomer.objects.get(user=request.user)
191     stripe.api_key = settings.STRIPE_SECRET_KEY
192     subscription = stripe_customer.stripeSubscriptionId
193 
194     stripe.Subscription.modify(
195         subscription,
196         cancel_at_period_end=False
197     )
198 
199     messages.success(request, 'Subscription reactivated successfully!')
200 
201     return redirect('/invoices/')
202 
203 
204 @login_required()
205 def check_subs(request):
206     """ Check for existing subs and send back to JS """
```

```
207     subscribed = StripeCustomer.objects.filter(user=request.user).exists()
208     results = {'subscribed': subscribed}
209 
210     return JsonResponse(results)
```

« index coverage.py v6.3.3, created at 2022-05-31 11:29 +0000