**CONTENTS**

**LIST OF CONTENTS PAGE NO**

**1**[**.**](https://docs.google.com/document/d/1kJZktaO0pPRdo00rmaDJq1UpTCWYdKsT6pmtnlOFwHk/edit#heading=h.fzahbmza80ps) **SOURCE CODE 1**

1[.1.](https://docs.google.com/document/d/1kJZktaO0pPRdo00rmaDJq1UpTCWYdKsT6pmtnlOFwHk/edit#heading=h.lh1bjxf81rog) Views.py 1

1.2. designer.html 3

**2**[**.**](https://docs.google.com/document/d/1kJZktaO0pPRdo00rmaDJq1UpTCWYdKsT6pmtnlOFwHk/edit#heading=h.2pv3oupaurdx) **CODE SNIPPETS 4**

2[.1.](https://docs.google.com/document/d/1kJZktaO0pPRdo00rmaDJq1UpTCWYdKsT6pmtnlOFwHk/edit#heading=h.4vjmtlf43ja7) Notification view 4

2[.2.](https://docs.google.com/document/d/1kJZktaO0pPRdo00rmaDJq1UpTCWYdKsT6pmtnlOFwHk/edit#heading=h.4vjmtlf43ja7) ConnectionUpdate view 4

2[.3.](https://docs.google.com/document/d/1kJZktaO0pPRdo00rmaDJq1UpTCWYdKsT6pmtnlOFwHk/edit#heading=h.4vjmtlf43ja7) Connection view 5

2[.4.](https://docs.google.com/document/d/1kJZktaO0pPRdo00rmaDJq1UpTCWYdKsT6pmtnlOFwHk/edit#heading=h.4vjmtlf43ja7) Chat view 5

1. **SOURCE CODE:**

**1.1. views.py**

class NotificationView(LoginRequiredMixin, View):

template\_name = "main/notification.html"

def get\_list(self, user):

return CustomNotification.objects.select\_related("from\_user", "to\_user").filter(

to\_user=user

)

def get(self, request, \*args, \*\*kwargs):

print(request.user)

print(self.get\_list(request.user))

return render(

request, self.template\_name, {"notifications": self.get\_list(request.user)}

)

def post(self, request):

try:

data = request.POST.dict()

if data["type"] == "connection":

data["from\_user"] = request.user

form = CustomNotificationForm(data=data)

if form.is\_valid():

form.save()

connec\_obj = Connection.objects.create(

from\_user=data["from\_user"], to\_user\_id=data["to\_user"]

)

channel\_layer = get\_channel\_layer()

channel = f"notification\_{str(request.user.id)}"

async\_to\_sync(channel\_layer.group\_send)(

channel,

{

"type": "notify",

"command": "new\_notification",

"from\_user": connec\_obj.from\_user.email,

"to\_user": connec\_obj.to\_user.first\_name,

"message": data["message"],

},

)

return JsonResponse({"message": ["Success"]})

return JsonResponse({"error": [form.errors]}, status=400)

return JsonResponse({"error": ["command not found"]}, status=400)

except Exception as e:

return JsonResponse({"error": [e.\_\_str\_\_()]}, status=400)

class ConnectionUpdateView(LoginRequiredMixin, View):

def get(self, request):

data = Connection.objects.filter(

Q(from\_user=request.user, status="Accepted")

| Q(to\_user=request.user, status="Accepted")

).values(

["from\_user\_email", "from\_user\_\_image", "to\_user\_\_image", "to\_user\_\_email"]

)

return JsonResponse({"data": list(data)})

def post(self, request, id, \*args, \*\*kwargs):

try:

form = ConnectionForm(

request.POST,

instance=Connection.objects.get(from\_user\_id=id, to\_user=request.user),

)

if form.is\_valid():

form.save()

CustomNotification.objects.filter(

from\_user\_id=id, to\_user=request.user

).update(seen=True)

return redirect("notification")

return JsonResponse({"error": [form.errors]}, status=400)

except Exception as e:

return JsonResponse({"error": [e.\_\_str\_\_()]}, status=400)

class ConnectionView(LoginRequiredMixin, View):

def get(self, request):

data = Connection.objects.filter(

Q(from\_user=request.user, status="Accepted")

| Q(to\_user=request.user, status="Accepted")

).values(

"from\_user\_\_email", "from\_user\_\_image", "to\_user\_\_image", "to\_user\_\_email"

)

return JsonResponse({"data": list(data)})

class ChatView(LoginRequiredMixin, View):

template\_name = "main/chat.html"

def get(self, request, \*args, \*\*kwargs):

form = PlanForm()

return render(request, self.template\_name)

**Description:** Views are Python functions or classes that receive a web request and deliver a web response in the Django framework, as shown in the following source code from Django views.py files. The possible responses can be a plain HTTP response, an HTML template response, or an HTTP redirect response that sends the user to another page.

**1.2. Designer.html**

{% extends 'base\_layout/index.html' %}

{% load static %}

{% load main\_tags %}

{% block body %}

<div class='designer-outer-body mt-5'>

<div class='container-fluid row mx-auto'>

{% for designer in designers %}

<div class="col col-md-6 col-xl-4 mb-4 mb-lg-0">

<div class="card mb-3" style="border-radius: .5rem;">

<div class="row g-0">

<div class="col-md-4 gradient-custom text-center text-white"

style="border-top-left-radius: .5rem; border-bottom-left-radius: .5rem;">

<img src={% if designer.imgae %}{{designer.image.url}}{% else %}"static/img/default.png"{% endif %} class="profile-img-circular mt-5" />

<h6 class='mt-2'>{{designer.first\_name}} {{designer.last\_name}}</h6>

<p>{{designer.user\_type}}</p>

{% if designer.is\_active %}

<button class='btn btn-danger designer-deactivate'

user-url={% url 'user-update' designer.id %}>Deactivate</button>

{% else %}

<button class='btn btn-success designer-activate'

user-url={% url 'user-update' designer.id %}>Activate</button>

{% endif %}

</div>

<div class="col-md-8">

<div class="card-body p-4">

<h6>Information</h6>

<hr class="mt-0 mb-2">

<div class="row pt-1">

<div class="col-4 mb-2">

<h6>Email</h6>

<p class="text-muted"><small>{{designer.email}}</small></p>

</div>

<div class="col-4 mb-2">

<h6>Date Joined</h6>

<p class="text-muted">{{designer.date\_joined|date:"Y-m-d"}} </p>

</div>

<div class="col-4 mb-2">

<h6>Date Joined</h6>

<p class="text-muted">{{designer.last\_login|date:"Y-m-d"}}</p>

</div>

</div>

<h6>Subscription Info</h6>

<hr class="mt-0">

<div class="row pt-1">

<div class="col-4 ">

<h6>Plan Name</h6>

<p class="text-muted "> <small>{{designer.subscription\_set.all|plan\_details}}

</small></p>

</div>

<div class="col-4 ">

<h6>Days Left</h6>

<p class="text-muted"><small>{{designer.subscription\_set.all|numRmDays}}</small></p>

</div>

<div class="col-4 ">

<h6>Plan Status</h6>

{% if designer.is\_active %}

<p class=" text-success">

Active

</p>

{% else %}

<p class="text-danger">

Deactive

</p>

{% endif %}

</div>

</div>

</div>

</div>

</div>

</div>

</div>

{% endfor %}

</div>

</div>

{% endblock body %}

{% block custom-script %}

<script src={% static 'js/designer.js' %}></script>

{% endblock custom-script %}

**Description:** The above code is an HTML template for our designer dashboard, which gives the designer access to four features: notifications, user views, and user chat, and also, check plans on the homepage.

**2. CODE SNIPPETS:**

**2.1. Notification view**

class NotificationView(LoginRequiredMixin, View):

template\_name = "main/notification.html"

def get\_list(self, user):

return CustomNotification.objects.select\_related("from\_user", "to\_user").filter(

to\_user=user

)

def get(self, request, \*args, \*\*kwargs):

print(request.user)

print(self.get\_list(request.user))

return render(

request, self.template\_name, {"notifications": self.get\_list(request.user)}

)

def post(self, request):

try:

data = request.POST.dict()

if data["type"] == "connection":

data["from\_user"] = request.user

form = CustomNotificationForm(data=data)

if form.is\_valid():

form.save()

connec\_obj = Connection.objects.create(

from\_user=data["from\_user"], to\_user\_id=data["to\_user"]

)

channel\_layer = get\_channel\_layer()

channel = f"notification\_{str(request.user.id)}"

async\_to\_sync(channel\_layer.group\_send)(

channel,

{

"type": "notify",

"command": "new\_notification",

"from\_user": connec\_obj.from\_user.email,

"to\_user": connec\_obj.to\_user.first\_name,

"message": data["message"],

},

)

return JsonResponse({"message": ["Success"]})

return JsonResponse({"error": [form.errors]}, status=400)

return JsonResponse({"error": ["command not found"]}, status=400)

except Exception as e:

return JsonResponse({"error": [e.\_\_str\_\_()]}, status=400)

**Description:** The Notification view class in the above code renders the application's notification page. Designers can check their notifications here, such as user connection requests or messages. As the user's id is displayed, the Designer may identify who the request is from and click to approve or reject it.

**2.2. ConnectionUpdate view**

**#user view**

class ConnectionUpdateView(LoginRequiredMixin, View):

def get(self, request):

data = Connection.objects.filter(

Q(from\_user=request.user, status="Accepted")

| Q(to\_user=request.user, status="Accepted")

).values(

["from\_user\_email", "from\_user\_\_image", "to\_user\_\_image", "to\_user\_\_email"]

)

return JsonResponse({"data": list(data)})

def post(self, request, id, \*args, \*\*kwargs):

try:

form = ConnectionForm(

request.POST,

instance=Connection.objects.get(from\_user\_id=id, to\_user=request.user),

)

if form.is\_valid():

form.save()

CustomNotification.objects.filter(

from\_user\_id=id, to\_user=request.user

).update(seen=True)

return redirect("notification")

return JsonResponse({"error": [form.errors]}, status=400)

except Exception as e:

return JsonResponse({"error": [e.\_\_str\_\_()]}, status=400)

**Description:**

In the above code, we have a ConnectionupdateView class. Which has a function that updates the status of the user's connection requests, such as whether they are accepted or rejected, and sends the status of the connection request to the user as a notification.

**2.3. Connection view**

class ConnectionView(LoginRequiredMixin, View):

def get(self, request):

data = Connection.objects.filter(

Q(from\_user=request.user, status="Accepted")

| Q(to\_user=request.user, status="Accepted")

).values(

"from\_user\_\_email", "from\_user\_\_image", "to\_user\_\_image", "to\_user\_\_email"

)

return JsonResponse({"data": list(data)})

**Description:**

We have a class called Connection View in the following code snippet that displays the details of the user whose connection request has been accepted, and they may now interact with that user.

**2.4. Chat view**

class ChatView(LoginRequiredMixin, View):

template\_name = "main/chat.html"

def get(self, request, \*args, \*\*kwargs):

form = PlanForm()

return render(request, self.template\_name)

**Description:** The following code sample uses the class chat view to render a chat.html page where the Designer and the user can connect and interact. It also shows all users who have had their connection requests accepted, so Designers can choose whom they want to talk with.