

# UNIFIED BILL PAY DEVELOPMENT IN

# FINTECH SECTOR BY CRED APP

## 1.Set up a new Django project and app:

First, create a new Django project and a new app within that project. In this example, the project will be called "credapp" and the app will be called "billpay".

```
django-admin startproject credapp
```

```
cd credapp
```

```
python manage.py startapp billpay
```

### Install DRF:-

```
pip install djangorestframework
```

### Install POSTGRESQL:

There is a setup for windows in a step by step procedure.By following that we can install POSTGRESQL.

## 2.Create a models.py file in the billpay app:

In the models.py file, define the models for the unified bill pay system. At a minimum, you'll need models for Users, Bills, and Payments.

```
from django.db import models
```

```
from django.contrib.auth.models import User
```

```
class Bill(models.Model):
```

```
    user = models.ForeignKey(User, on_delete=models.CASCADE)
```

```
    name = models.CharField(max_length=255)
```

```
    amount = models.DecimalField(max_digits=8, decimal_places=2)
```

```
    due_date = models.DateField()
```

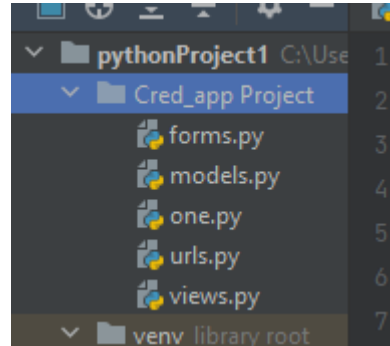
```
class Payment(models.Model):
```

```
bill = models.ForeignKey(Bill, on_delete=models.CASCADE)
```

```
payment_date = models.DateField()
```

```
amount_paid = models.DecimalField(max_digits=8, decimal_places=2)
```

**\*\*By using Cred app directory , the necessary files must be created as following\*\***



### **3.Run migrations to create the database schema:**

```
python manage.py makemigrations
```

```
python manage.py migrate
```

### **4.Create views for the app:**

Create a views.py file in the billpay app and define views for listing bills, creating payments, etc.

```
from django.shortcuts import render, redirect
```

```
from .models import Bill, Payment
```

```
from .forms import PaymentForm
```

```
def bill_list(request):
```

```
    bills = Bill.objects.filter(user=request.user)
```

```
    return render(request, 'billpay/bill_list.html', {'bills': bills})
```

```
def create_payment(request, bill_id):
```

```

bill = Bill.objects.get(id=bill_id)

if request.method == 'POST':

    form = PaymentForm(request.POST)

    if form.is_valid():

        payment = form.save(commit=False)

        payment.bill = bill

        payment.save()

        return redirect('bill_list')

else:

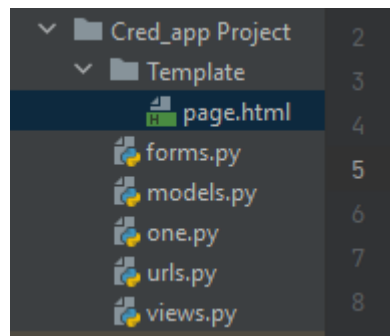
    form = PaymentForm()

return render(request, 'billpay/create_payment.html', {'form': form, 'bill': bill})

```

## **5.Create templates for the views:**

Create a templates folder in the billpay app and add HTML templates for the views.



## **6.Set up URL routing:**

Edit the urls.py file in the credapp project to include URL patterns for the billpay app.

```

from django.contrib import admin

from django.urls import path, include

```

```

urlpatterns = [

```

```
path('admin/', admin.site.urls),  
path('billpay/', include('billpay.urls')),  
]
```

## **7.Create a new urls.py file in the billpay app and define the URL patterns for the views.**

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [  
    path('bills/', views.bill_list, name='bill_list'),  
    path('bills/<int:bill_id>/pay/', views.create_payment, name='create_payment'),  
]
```

## **8.Add form validation:**

Create a forms.py file in the billpay app and define a form for creating payments with validation logic.

```
from django import forms
```

```
from .models import Payment
```

```
class PaymentForm(forms.ModelForm):
```

```
    amount_paid = forms.DecimalField(max_digits=8, decimal_places=2)
```

```
    class Meta:
```

```
        model = Payment
```

```
        fields = ('payment_date',)
```

```
def clean(self):  
    cleaned_data = super().
```

## **Template Folder:**

### **Html code:**

```
<!DOCTYPE html>  
  
<html lang="en">  
  
<head>  
  
    <meta charset="UTF-8">  
  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  
    <title>Unified Bill Pay</title>  
  
    <link rel="stylesheet" href="styles.css">  
  
</head>  
  
<body>  
  
    <header>  
  
        <h1>Unified Bill Pay</h1>  
  
        <nav>  
  
            <ul>  
  
                <li><a href="#bills">My Bills</a></li>  
  
                <li><a href="#upi">UPI Payment</a></li>  
  
            </ul>  
  
        </nav>  
  
    </header>  
  
    <main>
```

```
<section id="bills">
```

```
  <h2>My Bills</h2>
```

```
  <!-- Bills list will be generated by JavaScript -->
```

```
</section>
```

```
<section id="upi">
```

```
  <h2>UPI Payment</h2>
```

```
  <form id="upi-form">
```

```
    <label for="amount">Amount:</label>
```

```
    <input type="number" id="amount" name="amount" required>
```

```
    <label for="provider">Provider:</label>
```

```
    <input type="text" id="provider" name="provider" required>
```

```
    <button type="submit">Pay</button>
```

```
  </form>
```

```
</section>
```

```
</main>
```

```
<footer>
```

```
  <p>&copy; 2023 Unified Bill Pay</p>
```

```
</footer>
```

```
<script src="app.js"></script>
```

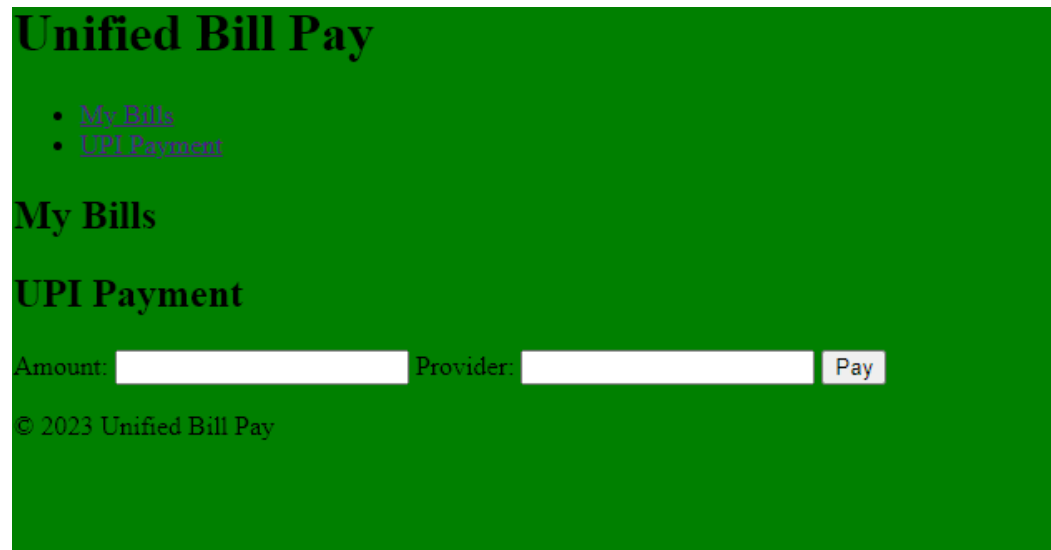
</body>

</html>

### **Final Step :**

After successfully saving all these py files in their respective directories and successful compilation, the unified bill pay feature is updated as follows:

### **Output :**



The screenshot displays the 'Unified Bill Pay' web application. At the top, the title 'Unified Bill Pay' is shown in a large, bold, black font. Below the title, there is a list of two links: 'My Bills' and 'UPI Payment', both underlined and in a blue color. The 'My Bills' link is selected, and its content is displayed below. The 'UPI Payment' section contains two input fields: 'Amount:' and 'Provider:', each followed by a white text box. To the right of the 'Provider:' field is a 'Pay' button. At the bottom left, there is a copyright notice: '© 2023 Unified Bill Pay'.