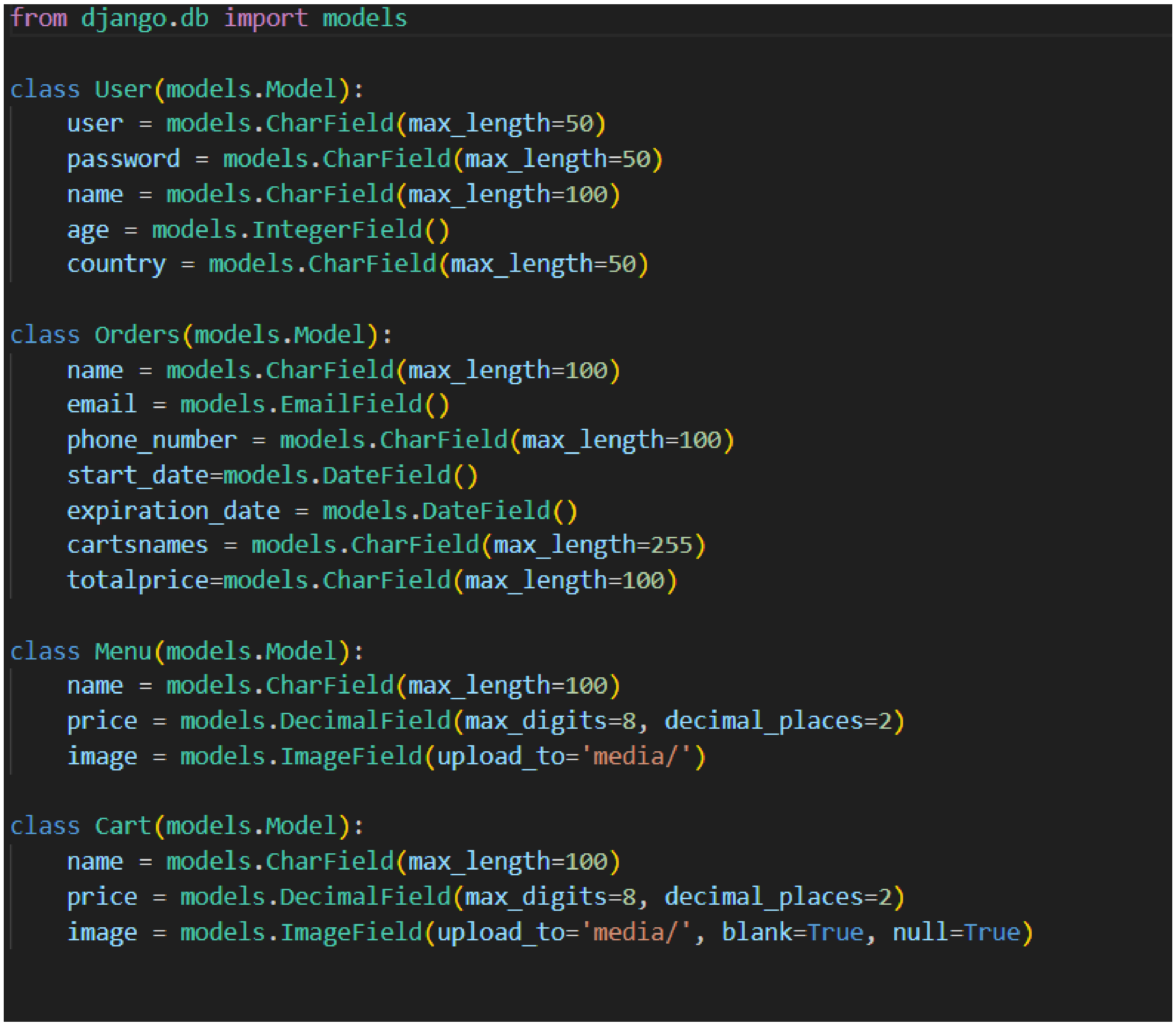
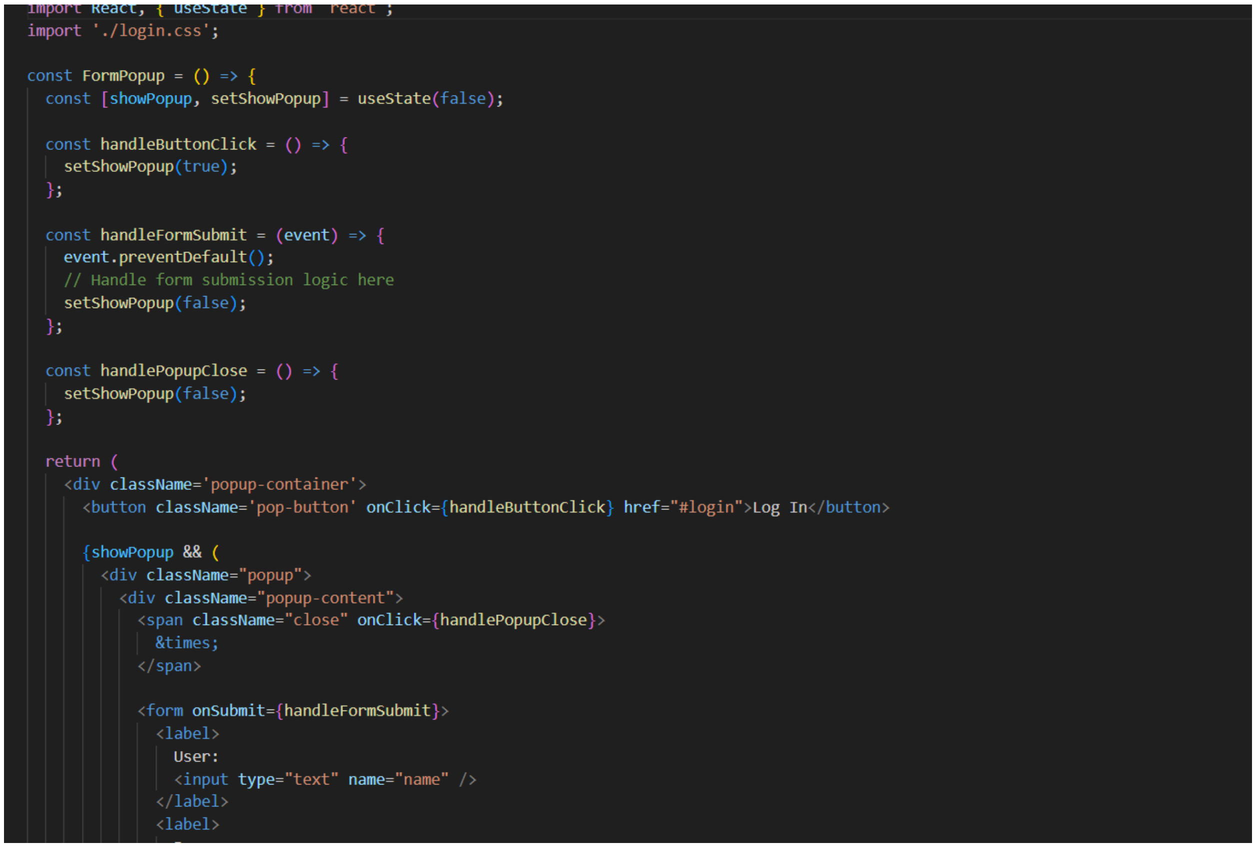


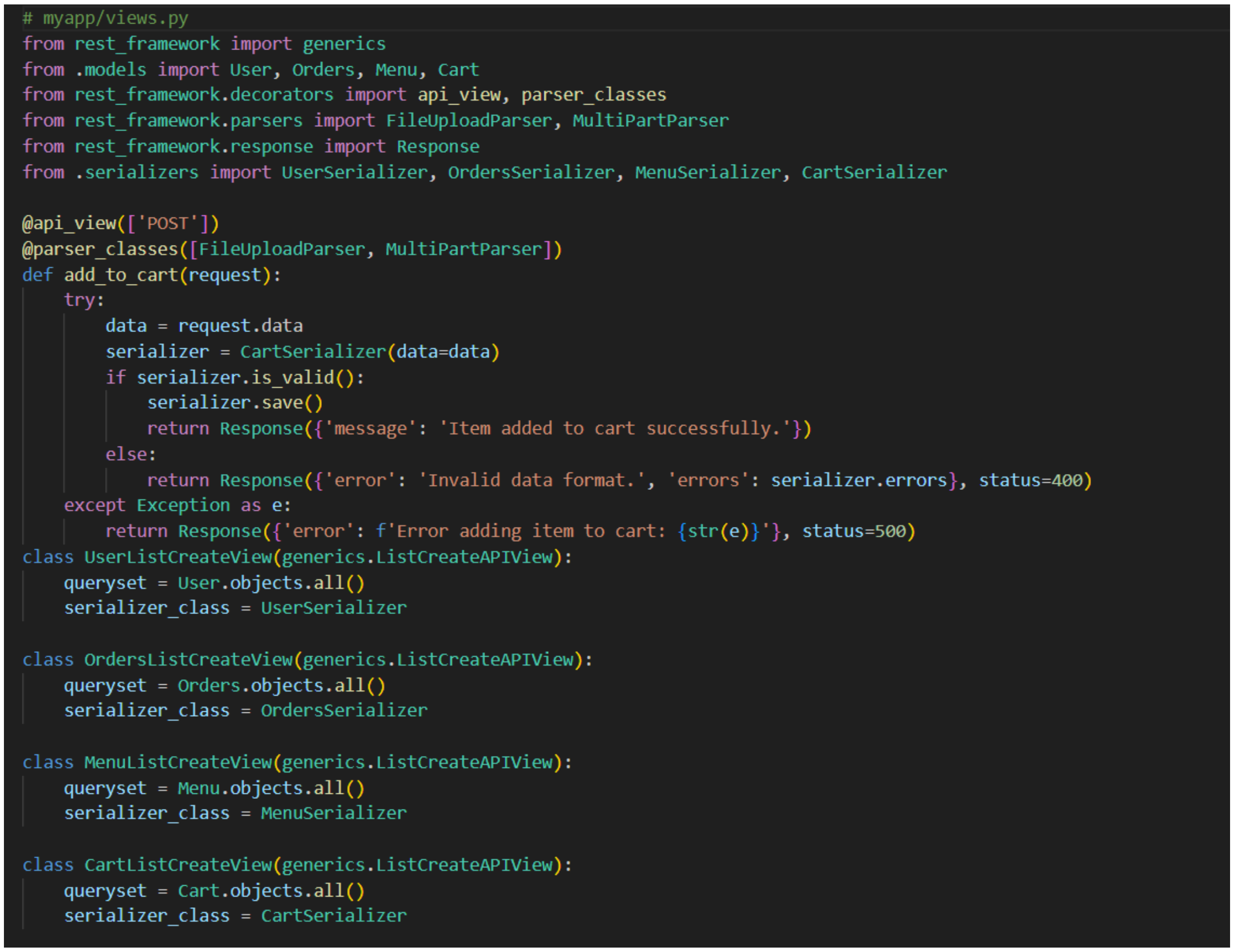
```python
from django.db import models

class User(models.Model):
    user = models.CharField(max_length=50)
    password = models.CharField(max_length=50)
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    country = models.CharField(max_length=50)

class Orders(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone_number = models.CharField(max_length=100)
    start_date=models.DateField()
    expiration_date = models.DateField()
    cartsnames = models.CharField(max_length=255)
    totalprice=models.CharField(max_length=100)

class Menu(models.Model):
    name = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=8, decimal_places=2)
    image = models.ImageField(upload_to='media/')

class Cart(models.Model):
    name = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=8, decimal_places=2)
    image = models.ImageField(upload_to='media/', blank=True, null=True)
```

This is an example of clean code in this picture it is proved that I have used the descriptive variable and function names



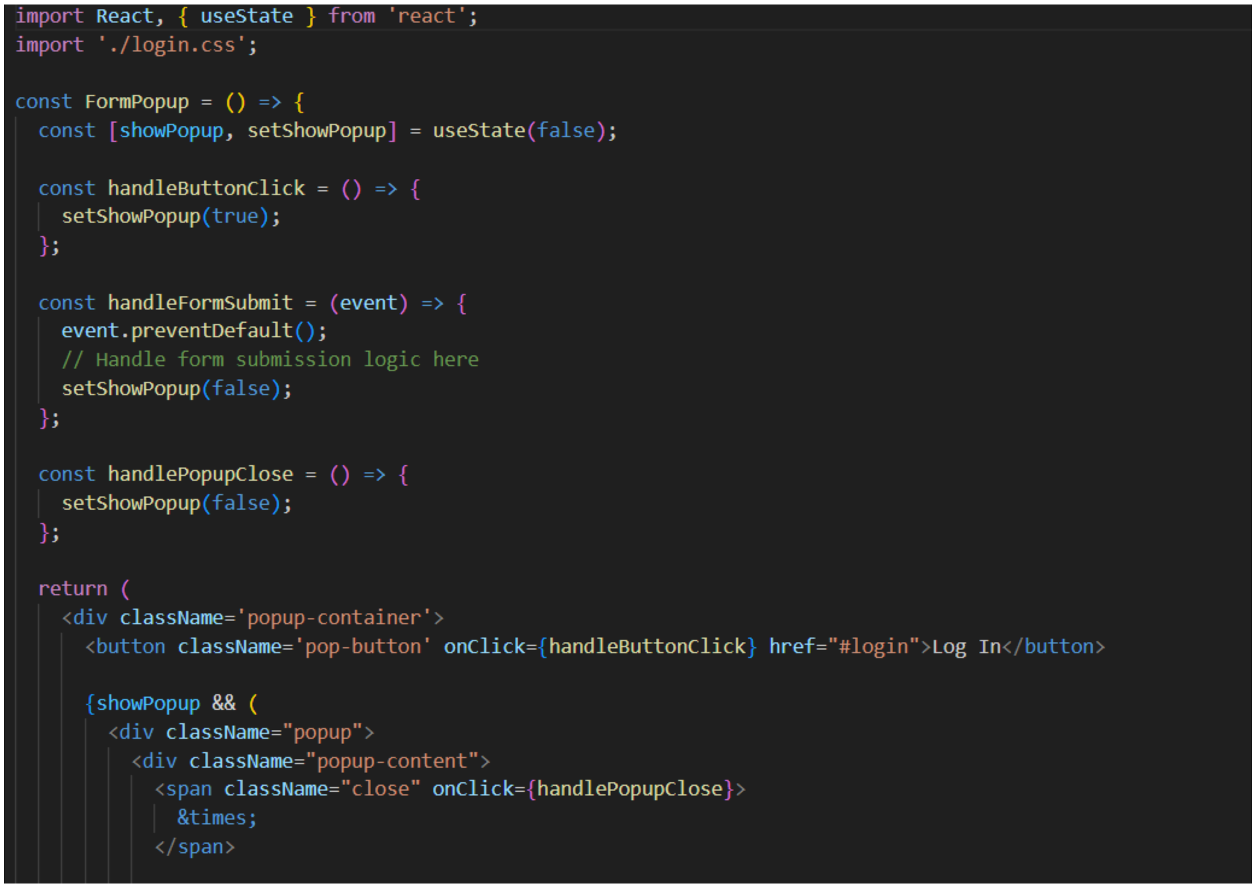
```
import React, { useState } from 'react';
import './login.css';

const FormPopup = () => {
  const [showPopup, setShowPopup] = useState(false);

  const handleButtonClick = () => {
    setShowPopup(true);
  };

  const handleFormSubmit = (event) => {
    event.preventDefault();
    // Handle form submission logic here
    setShowPopup(false);
  };

  const handlePopupClose = () => {
    setShowPopup(false);
  };

  return (
    <div className='popup-container'>
      <button className='pop-button' onClick={handleButtonClick} href="#login">Log In</button>

      {showPopup && (
        <div className="popup">
          <div className="popup-content">
            <span className="close" onClick={handlePopupClose}>
              &times;
            </span>

            <form onSubmit={handleFormSubmit}>
              <label>
                User:
                <input type="text" name="name" />
              </label>
              <label>
```

This is a second example as you can see it is easy to read and understand the code



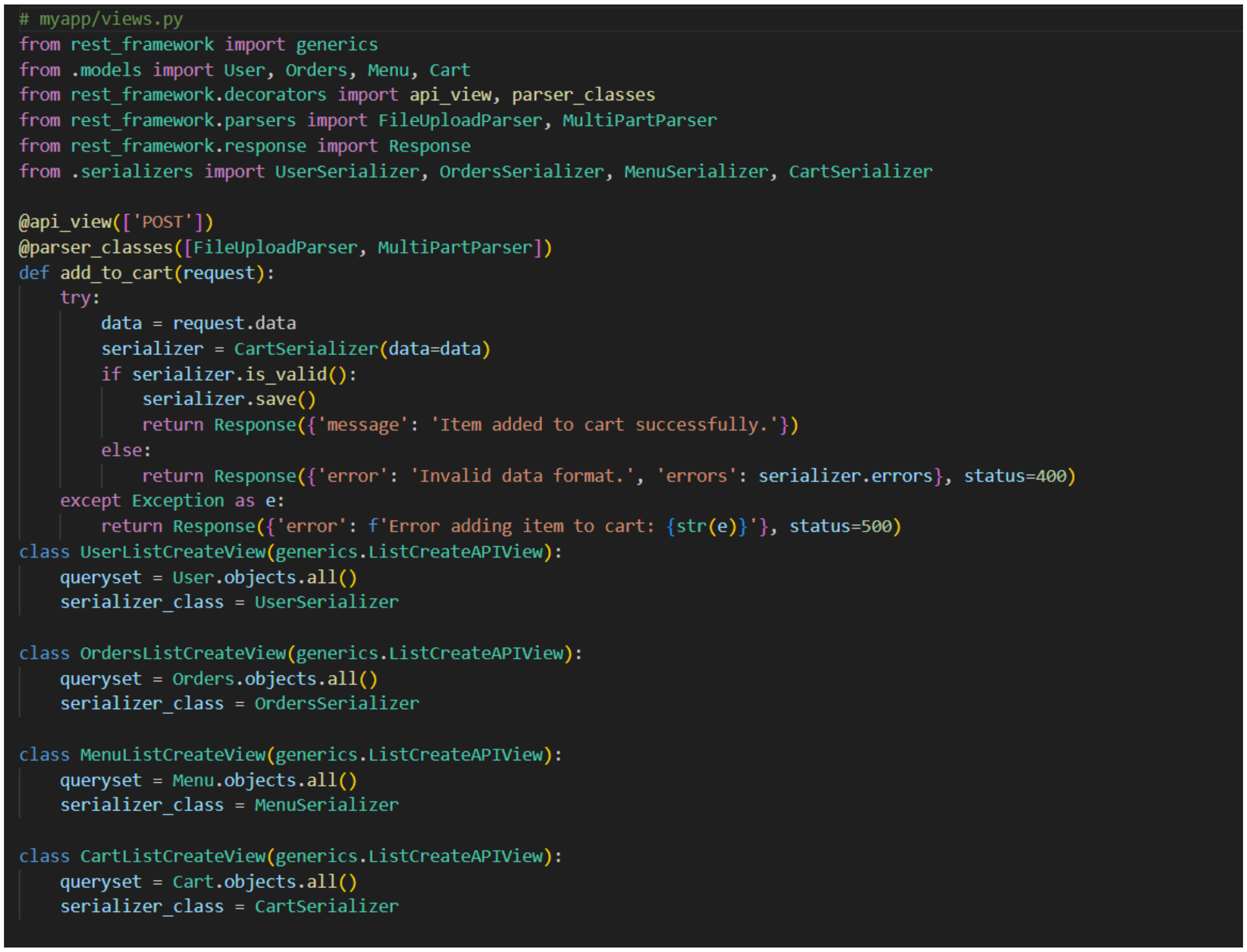
```python
# myapp/views.py
from rest_framework import generics
from .models import User, Orders, Menu, Cart
from rest_framework.decorators import api_view, parser_classes
from rest_framework.parsers import FileUploadParser, MultiPartParser
from rest_framework.response import Response
from .serializers import UserSerializer, OrdersSerializer, MenuSerializer, CartSerializer

@api_view(['POST'])
@parser_classes([FileUploadParser, MultiPartParser])
def add_to_cart(request):
    try:
        data = request.data
        serializer = CartSerializer(data=data)
        if serializer.is_valid():
            serializer.save()
            return Response({'message': 'Item added to cart successfully.'})
        else:
            return Response({'error': 'Invalid data format.', 'errors': serializer.errors}, status=400)
    except Exception as e:
        return Response({'error': f'Error adding item to cart: {str(e)}'}, status=500)
class UserListCreateView(generics.ListCreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer

class OrdersListCreateView(generics.ListCreateAPIView):
    queryset = Orders.objects.all()
    serializer_class = OrdersSerializer

class MenuListCreateView(generics.ListCreateAPIView):
    queryset = Menu.objects.all()
    serializer_class = MenuSerializer

class CartListCreateView(generics.ListCreateAPIView):
    queryset = Cart.objects.all()
    serializer_class = CartSerializer
```

Avoid magic numbers and strings so as you can see no magic numbers were used.



```
import React, { useState } from 'react';
import './login.css';

const FormPopup = () => {
  const [showPopup, setShowPopup] = useState(false);

  const handleButtonClick = () => {
    setShowPopup(true);
  };

  const handleFormSubmit = (event) => {
    event.preventDefault();
    // Handle form submission logic here
    setShowPopup(false);
  };

  const handlePopupClose = () => {
    setShowPopup(false);
  };

  return (
    <div className='popup-container'>
      <button className='pop-button' onClick={handleButtonClick} href="#login">Log In</button>

      {showPopup && (
        <div className="popup">
          <div className="popup-content">
            <span className="close" onClick={handlePopupClose}>
              &times;
            </span>
```

Comments to help understand what’s going on



```python
# myapp/views.py
from rest_framework import generics
from .models import User, Orders, Menu, Cart
from rest_framework.decorators import api_view, parser_classes
from rest_framework.parsers import FileUploadParser, MultiPartParser
from rest_framework.response import Response
from .serializers import UserSerializer, OrdersSerializer, MenuSerializer, CartSerializer

@api_view(['POST'])
@parser_classes([FileUploadParser, MultiPartParser])
def add_to_cart(request):
    try:
        data = request.data
        serializer = CartSerializer(data=data)
        if serializer.is_valid():
            serializer.save()
            return Response({'message': 'Item added to cart successfully.'})
        else:
            return Response({'error': 'Invalid data format.', 'errors': serializer.errors}, status=400)
    except Exception as e:
        return Response({'error': f'Error adding item to cart: {str(e)}'}, status=500)
class UserListCreateView(generics.ListCreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer

class OrdersListCreateView(generics.ListCreateAPIView):
    queryset = Orders.objects.all()
    serializer_class = OrdersSerializer

class MenuListCreateView(generics.ListCreateAPIView):
    queryset = Menu.objects.all()
    serializer_class = MenuSerializer

class CartListCreateView(generics.ListCreateAPIView):
    queryset = Cart.objects.all()
    serializer_class = CartSerializer
```

Error handling.