Edit Coin and adding coin to cart

For placing an edit coin with valid URL in coin details page , first we need to update the **urls.py** in project folder :

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from rest_framework import routers
from coins.views import CoinViewSet, CoinSearchView, coins_table, coin_details, edit_coin

# Create a router for registering viewsets
router = routers.DefaultRouter()

# Register CoinViewSet with the router
router.register(r'coins', CoinViewSet)

# Define URL patterns
urlpatterns = [
   # Admin site URL
   path('admin/', admin.site.urls),
   # API endpoints for coins using the router
   path('api/', include(router.urls)),
   path('coins/search/<path:path_params>/', CoinSearchView.as_view(),
name='coin-search'),
   path('coins/', coins_table, name='coins-table'),  # URL for the coins table HTML page
   path('coin/<int:coin_id>/', coin_details, name='coin-details'),  # URL for the coin details
page
   path('edit-coin/<int:coin_id>/', edit_coin, name='edit_coin'),
   path("", include(("coins.urls", "coins"), "coins")),
]
# Serve media files in DEBUG mode
if settings.DEBUG:
   urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Followed by, you need to update the **views.py** with :

```
@login_required
def edit_coin(request, coin_id):
   coin = get_object_or_404(Coin, pk=coin_id)

   if request.user.id != coin.created_by_id:
       return redirect('coins:home')

   if request.method == 'POST':
        form = CoinForm(request.POST, request.FILES, instance=coin)
        if form.is_valid():
            form.save()
            messages.success(request, 'Coin updated successfully!')
            return redirect(reverse('coin-details', kwargs={'coin_id': coin_id}))
        else:
            messages.error(request, 'Please correct the errors below.')
    else:
        form = CoinForm(instance=coin)

    return render(request, 'edit_coin.html', {'form': form, 'coin': coin})
```

Next place the edit button in **coin_details..html** in respective place (only to be visible for the owner of coin creation) :

```
{% extends "base.html" %}
{% load bootstrap4 %}
{% block content %}
  <div class="container mt-5">
      <h1 class="text-center mb-4"><strong>COIN DETAILS</strong></h1>
      <div class="row justify-content-center">
          <div class="col-md-12">
              <div class="card border-0 shadow-lg" style="background-color: #c7e619;">
                  <div class="card-body">
                      <h3 class="card-title text-center mb-4">{{ coin.coin_name }}</h3>
                      <div class="row">
                          <div class="col-md-6">
                              <p class="card-text"><strong>Year:</strong> {{ coin.coin_year }}</p>
                              <p class="card-text"><strong>Country:</strong> {{ coin.coin_country
}}</p>
                              <p class="card-text"><strong>Material:</strong> {{
coin.coin_material }}</p>
                              <p class="card-text"><strong>Weight:</strong> {{ coin.coin_weight
}}</p>
                              <p class="card-text"><strong>Starting Bid:</strong> {{
coin.starting_bid }}</p>
                              <p class="card-text"><strong>Coin Status:</strong> {{
coin.coin_status }}</p>
                          </div>
                          <div class="col-md-6">
                              {% if coin.coin_image %}
                              <p class="text-center"><img src="{{ coin.coin_image.url }}" alt="{{
coin.coin_name }}" class="img-fluid rounded"></p>
                              {% else %}
                              <p class="text-center">No Image</p>
                              {% endif %}
                              <p class="card-text"><strong>Description:</strong> {{ coin.coin_desc
}}</p>
```

```
                    </div>
                 </div>
              </div>
           </div>
           <div>&nbsp;</div>
           <div class="text-center">
              <div class="btn-group">
                 <form action="{% url 'coins:home' %}">
                    <button type="submit" class="btn btn-secondary">Back to
Home</button>
                 </form><span>&nbsp;</span>
                 {% if request.user.id == coin.created_by_id %}
                 <form action="{% url 'edit_coin' coin_id=coin.coin_id %}">
                    <button type="submit" class="btn btn-warning">Edit Coin</button>
                 </form><span>&nbsp;</span>
                 {% endif %}
              </div>
           </div>
        </div>
     </div>
  </div>
{% endblock content %}
```

Now , next we are going to implement the "Buy Now" option for buying the coin from coin details page :

First create the models for cart items page according to the requirements :

So, we are gonna update the **models.py** like this :

```
class Coin(models.Model):
  # Define choices for coin status
  STATUS_CHOICES = (
     ('Select', 'Select'),  # Placeholder option
     ('available', 'Available'),
     ('sold', 'Sold'),
     ('pending', 'Pending'),
  )

  coin_id = models.AutoField(primary_key=True)  # Auto-incrementing primary key
  coin_image = models.ImageField(upload_to='coin_images/', null=True, blank=True)   #
Image field to store coin image
  coin_name = models.CharField(max_length=100)  # Char field for coin name
  coin_desc = models.TextField()  # Text field for coin description
  coin_year = models.IntegerField()  # Integer field for coin year
```

```
    coin_country = models.CharField(max_length=50)  # Char field for coin country
    coin_material = models.CharField(max_length=50)  # Char field for coin material
    coin_weight = models.FloatField()  # Float field for coin weight
    starting_bid = models.FloatField()
    rate = models.FloatField()  # Float field for starting bid
    coin_status = models.CharField(max_length=50, choices=STATUS_CHOICES)  # Char
field with choices for coin status
    created_by_id = models.IntegerField(null=True, blank=True)

    def __str__(self):
        return self.coin_name  # Return the coin name as its string representation

class CartItem(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    coin = models.ForeignKey(Coin, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)
    price = models.FloatField()
    created_at = models.DateTimeField(auto_now_add=True)

    def save(self, *args, **kwargs):
        # Set the price to the rate of the associated coin multiplied by the quantity
        self.price = self.coin.rate * self.quantity
        super().save(*args, **kwargs)
```

Then as usual we need to make sure we are migrating after the model creation via terminal :

```
python manage.py makemigrations
python manage.py migrate
```

Next update your **urls.py** to make the valid button links :

(This is for multiple purposes : Buying, cart page, saving changes, removing items)

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from rest_framework import routers
from coins.views import CoinViewSet, CoinSearchView, coins_table, coin_details, edit_coin,
add_to_cart, cart, remove_item, save_changes

# Create a router for registering viewsets
router = routers.DefaultRouter()
```

```
# Register CoinViewSet with the router
router.register(r'coins', CoinViewSet)

# Define URL patterns
urlpatterns = [
  # Admin site URL
  path('admin/', admin.site.urls),
  # API endpoints for coins using the router
  path('api/', include(router.urls)),
  path('coins/search/<path:path_params>/', CoinSearchView.as_view(),
name='coin-search'),
  path('coins/', coins_table, name='coins-table'),  # URL for the coins table HTML page
  path('coin/<int:coin_id>/', coin_details, name='coin-details'),  # URL for the coin details
page
  path('edit-coin/<int:coin_id>/', edit_coin, name='edit_coin'),
  path('add-to-cart/<int:coin_id>/', add_to_cart, name='add_to_cart'),
  path('cart/', cart, name='cart'),
  path('remove_item/<int:item_id>/', remove_item, name='remove_item'),
  path('save_changes/', save_changes, name='save_changes'),
  path("", include(("coins.urls", "coins"), "coins")),
]
# Serve media files in DEBUG mode
if settings.DEBUG:
  urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Next make sure the **views.py** is updated :

```
from .models import Coin, Profile, SearchHistory, CartItem
@login_required
def save_changes(request):
   if request.method == 'POST':
      for key, value in request.POST.items():
         if key.startswith('quantity_'):
            item_id = int(key.split('_')[1])
            quantity = int(value)
            cart_item = CartItem.objects.get(pk=item_id)
            cart_item.quantity = quantity
            cart_item.price = cart_item.coin.rate * quantity
            cart_item.save()
        messages.success(request, 'Changes saved successfully!')
        return redirect('cart')  # Assuming 'cart' is the name of the URL pattern for the cart page
    else:
        # Handle GET request (if needed)
        pass

@login_required
def remove_item(request, item_id):
    item = get_object_or_404(CartItem, id=item_id)
    item.delete()
    return redirect('cart')

@login_required
def cart(request):
    user = request.user
    cart_items = CartItem.objects.filter(user=user)

    # Calculate total price
    total_price = sum(item.quantity * item.coin.rate for item in cart_items)

    for item in cart_items:
        item.price = item.quantity * item.coin.rate

    return render(request, 'cart.html', {'cart_items': cart_items, 'total_price': total_price})


@login_required
def add_to_cart(request, coin_id):
    if request.method == 'POST':
        # Get the selected coin
        coin = Coin.objects.get(pk=coin_id)
        # Get the current user
        user = request.user
        # Create a CartItem object
        cart_item = CartItem.objects.create(user=user, coin=coin)
        # Redirect to the cart page
        return redirect('cart')
    else:
        # Handle GET request (if needed)
        Pass
```

Next place the buy now button in **coin details page** like this :

```
            <div class="text-center">
                <div class="btn-group">
                    <form action="{% url 'coins:home' %}">
            <button type="submit" class="btn btn-secondary">Back to
Home</button>
          </form><span>&nbsp;</span>
          {% if request.user.id == coin.created_by_id %}
          <form action="{% url 'edit_coin' coin_id=coin.coin_id %}">
            <button type="submit" class="btn btn-warning">Edit Coin</button>
          </form><span>&nbsp;</span>
          <form method="post" action="{% url 'add_to_cart' coin.coin_id %}">
            {% csrf_token %}
            <button type="submit" class="btn btn-success">Buy Now</button>
          </form>
          {% endif %}
        </div>
      </div>
```

Next you can proceed with the cart page view and create the **cart.html** in templates/ folder :

```
{% extends "base.html" %}
{% load bootstrap4 %}
{% block content %}
<div class="container mt-5">
  <h1 class="mb-4 font-weight-bold">Shopping Cart <a href="{% url "coins:home" %}"
class="btn btn-warning btn-sm">Return</a></h1>
  <div class="row">
    <div>&nbsp;</div>
    <div class="col-md-8">
      {% bootstrap_messages %}
      {% if cart_items %}
      <form action="{% url 'save_changes' %}" method="post">
        {% csrf_token %}
        <table class="table">
          <thead>
            <tr class="text-center">
              <th scope="col">#</th>
              <th scope="col">Coin Name</th>
              <th scope="col">Quantity</th>
              <th scope="col">Rate</th>
              <th scope="col">Price</th>
              <th scope="col">Actions</th>
            </tr>
          </thead>
```

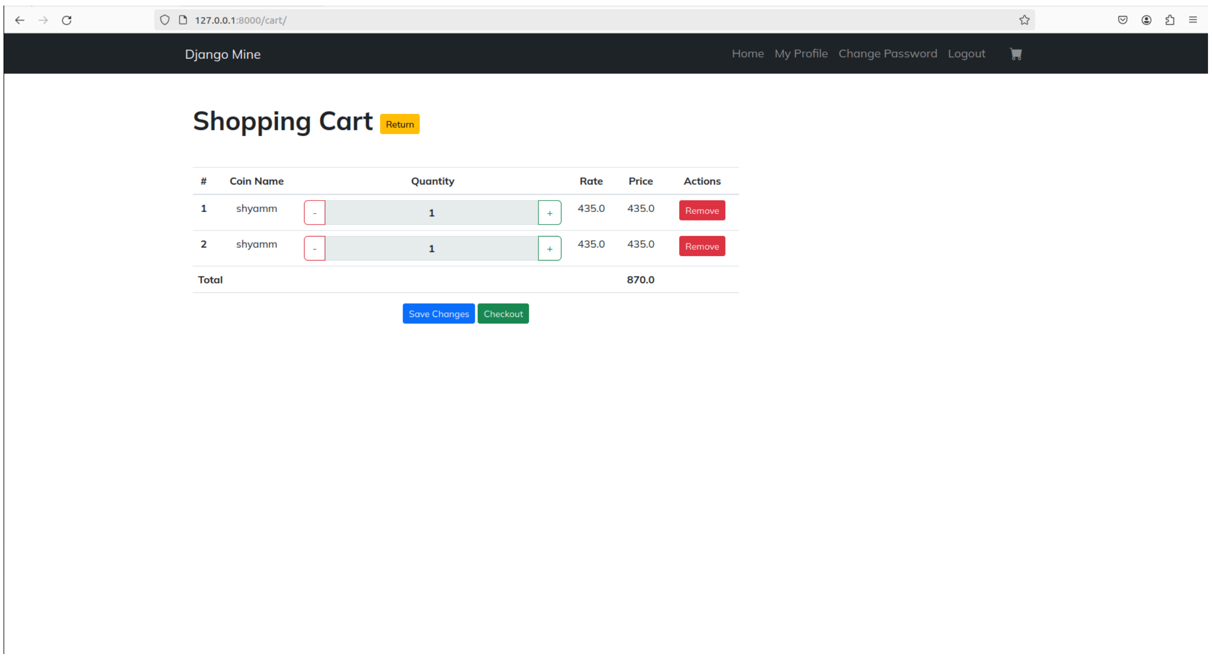
```
                <tbody>
                  {% for item in cart_items %}
                  <tr class="text-center">
                    <th scope="row">{{ forloop.counter }}</th>
                    <td>{{ item.coin.coin_name }}</td>
                    <td>
                      <div class="input-group">
                        <div class="input-group-prepend">
                          <button class="btn btn-outline-danger" type="button"
onclick="decrementQuantity({{ item.id }})">-</button>
                        </div>
                        <input type="text" class="form-control text-center font-weight-bold"
name="quantity_{{ item.id }}" id="quantity_{{ item.id }}" value="{{ item.quantity }}"
readonly>
                        <div class="input-group-append">
                          <button class="btn btn-outline-success" type="button"
onclick="incrementQuantity({{ item.id }})">+</button>
                        </div>
                      </div>
                    </td>
                    <td>{{ item.coin.rate }}</td>
                    <td class="price">{{ item.price }}</td>
                    <td>
                      <a href="{% url 'remove_item' item.id %}" class="btn btn-danger
btn-sm">Remove</a>
                    </td>
                  </tr>
                  {% endfor %}
                  <tr>
                    <td colspan="4" class="font-weight-bold">Total</td>
                    <td class="font-weight-bold total-price text-center">{{ total_price }}</td>
                    <td></td>
                  </tr>
                </tbody>
              </table>
              <div class="text-center">
                <button type="submit" class="btn btn-primary btn-sm">Save
Changes</button>
                <a href="#" class="btn btn-success btn-sm">Checkout</a>
              </div>
            </form>
            {% else %}
            <div class="alert alert-info" role="alert">
              There are no items in your shopping cart.
            </div>
            {% endif %}
          </div>
      </div>
</div>
```

```
<script>
  function updatePrice(itemId) {
      var quantityInput = document.getElementById('quantity_' + itemId);
      var priceCell =
document.querySelector(`#quantity_${itemId}`).closest('tr').querySelector('.price');
      var rate = parseFloat(priceCell.previousElementSibling.textContent);
      var quantity = parseInt(quantityInput.value);
      var price = rate * quantity;
      price = price.toFixed(1);
      priceCell.textContent = price;
      updateTotalPrice();
  }

  function updateTotalPrice() {
      var totalPrice = 0;
      var priceCells = document.querySelectorAll('.price');
      priceCells.forEach(function(cell) {
          totalPrice += parseFloat(cell.textContent);
      });
      document.querySelector('.total-price').textContent = totalPrice.toFixed(1);
  }
  function incrementQuantity(itemId) {
      var inputField = document.getElementById('quantity_' + itemId);
      var currentValue = parseInt(inputField.value);
      if (!isNaN(currentValue)) {
          inputField.value = currentValue + 1;
          updatePrice(itemId);
      }
  }

  function decrementQuantity(itemId) {
      var inputField = document.getElementById('quantity_' + itemId);
      var currentValue = parseInt(inputField.value);
      if (!isNaN(currentValue) && currentValue > 1) {
          inputField.value = currentValue - 1;
          updatePrice(itemId);
      }
  }
</script>
{% endblock content %}
```

Here, you can do the following actions :

1. By default , one quantity will be added when you buy the coin, it will redirect to the cart page after your purchase.
2. You can increase and decrease the number of coins you wish to purchase.
3. You can remove the purchase at any time as you wish which product should be removed accurately.

4. You can modify the number of coins at flexible timings, as you can access the cart page from home itself from navbar contents.
5. For every action you done, you can able to see the price and total amount updating itself dynamically.
6. So, once you finalised your purchase with needed coins and you can add multiple coin products to the cart page as you needed, you can proceed to make the changes to the cart page.



Shopping Cart

#	Coin Name	Quantity	Rate	Price	Actions
1	shyamm	- 1 +	435.0	435.0	Remove
2	shyamm	- 1 +	435.0	435.0	Remove
Total				870.0	

==