1. *Django Rest Framework :*

To install Django REST Framework, you can use pip, Python's package manager. Here's how you can install it:

```
pip install djangorestframework
```

If you're using a virtual environment (which is recommended), make sure it's activated before running the above command. After installation, you can start using Django REST Framework in your Django projects.

Remember to also add 'rest_framework' to the INSTALLED_APPS list in your Django project's settings.py file:

```
INSTALLED_APPS = [
    ...
    'Rest_framework',
    ...
]
```

Your expected directory must be like this to do the api rest framework, if any file missing add them in the respective:

```
BasicApp/
├── coins/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations/
│   │   └── __init__.py
│   ├── models.py
│   ├── serializers.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── BasicProject/
    ├── __init__.py
    ├── asgi.py
    ├── settings.py
```

```
    ├── urls.py
    └── wsgi.py
└── media/
    └── coin_images/
```

To implement CRUD APIs for managing coins in your Django project, you can follow these steps:

- Create Serializers: Create serializers to convert model instances to JSON format and vice versa. You can define serializers in the coins/serializers.py file.

```
from rest_framework import serializers  # Import serializers from Django REST Framework

from .models import Coin  # Import the Coin model

class CoinSerializer(serializers.ModelSerializer):  # Define a serializer for the Coin model

    # Define a HyperlinkedIdentityField for generating hyperlinks to individual coin details

    view_details = serializers.HyperlinkedIdentityField(view_name='coin-detail', lookup_field='pk')

    class Meta:

        model = Coin  # Specify the Coin model to serialize

        fields = ['coin_id', 'coin_image', 'coin_name', 'coin_desc', 'coin_year', 'coin_country', 'coin_material', 'coin_weight', 'starting_bid', 'coin_status', 'view_details']  # Define the fields to include in the serialised representation
```

- Define Views: Create views to handle CRUD operations for the Coin model. You can define viewsets using Django REST Framework's ModelViewSet class in the coins/views.py file.

```
from django.shortcuts import render  # Import render function from Django
```

```
from rest_framework import status  # Import status codes from Django REST Framework

from rest_framework.response import Response  # Import Response class from Django REST Framework

from rest_framework import viewsets  # Import viewsets from Django REST Framework

from .models import Coin  # Import the Coin model

from .serializers import CoinSerializer  # Import the CoinSerializer

class CoinViewSet(viewsets.ModelViewSet):  # Define a viewset for the Coin model

    queryset = Coin.objects.all()  # Define the queryset to fetch all coin objects

    serializer_class = CoinSerializer  # Specify the serializer class to use for the Coin model
```

- Configure URLs: Configure URLs to map the viewset to appropriate endpoints. You can define URL patterns in the BasicProject/urls.py file.

```
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

# 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)),
]
# Serve media files in DEBUG mode
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

- Run the Development Server: Start the Django development server to test your APIs.

  ```
  python manage.py runserver
  ```

  Your CRUD APIs for managing coins should now be accessible at /api/coins/ endpoint. You can perform CRUD operations (Create, Read, Update, Delete) using tools like Postman or by sending HTTP requests directly.

If you have implemented the CRUD APIs for managing coins as described earlier and have navigated to http://127.0.0.1:8000/api/coins/, you should see the following:

- **List of Coins**: If you've implemented the list view for your CoinViewSet, you should see a list of all coins available in your database. Each coin object will be represented in JSON format.
- **Create New Coin Form**: If you've implemented the create view for your CoinViewSet, you should see a form or a way to submit new coin data to create a new coin object.
- **Individual Coin Detail**: If you've implemented the retrieve view for your CoinViewSet, you can append the ID of a specific coin to the URL to view the details of that coin. For example, http://127.0.0.1:8000/api/coins/1/ would show the details of the coin with ID 1.
- **Update Coin Form**: If you've implemented the update view for your CoinViewSet, you should be able to edit the details of an existing coin by appending the ID of the coin to the URL and making a PUT or PATCH request with the updated data.

- **Delete Coin Endpoint**: If you've implemented the destroy view for your CoinViewSet, you should be able to delete a coin by appending the ID of the coin to the URL and making a DELETE request.

- To add a "view_details" link for showing individual details of each coin when displaying the list of all coins, you can modify the CoinSerializer to include a hyperlink field for the individual coin detail endpoint. Here's how you can do it:

```
# coins/serializers.py
from rest_framework import serializers
from .models import Coin
class CoinSerializer(serializers.ModelSerializer):
    view_details = serializers.HyperlinkedIdentityField(view_name='coin-detail', lookup_field='pk')
    class Meta:
        model = Coin
        fields = ['id', 'coin_name', 'coin_desc', 'coin_year', 'coin_country', 'view_details']
```

In this serializer:

- We've added a new field called view_details, which is a HyperlinkedIdentityField.
- We've specified the view_name parameter as 'coin-detail', which corresponds to the URL pattern for the individual coin detail endpoint.
- We've specified the lookup_field parameter as 'pk', which indicates that the primary key of the coin should be used to construct the URL.

Now, you'll be able to view the list in rest framework with individual links for CRUD.

==========