Project Structure:

Django API:

api/models.py: Define the data model.

api/views.py: Implement API views.

api/serializers.py: Create serializers for data.

api/urls.py: Define API endpoints.

api/views.py: Implement the logic for retrieving, storing, editing, and deleting data.

api/authentication.py: Implement authentication and authorization.

manage.py: Django management script.

Step-by-Step Guide:

1. Create Django Project and App:

```
$ django-admin startproject myapi
$ cd myapi
$ python manage.py startapp api
```

2. Define Data Model (api/models.py):

```
from django.db import models
class UserData(models.Model):
    name = models.CharField(max_length=255, unique=True)
```

3. Create Serializers (api/serializers.py):

```
from rest_framework import serializers
from.models import UserData
class UserDataSerializer(serializers.ModelSerializer):
class Meta:
    model = UserData
    fields = ('id', 'name')
```

4. Implement API Views (api/views.py):

```
from rest_framework import generics
from rest_framework.response import Response
```

```
from rest_framework.decorators import authentication_classes, permission_classes
from rest_framework.permissions import IsAuthenticated
from rest_framework.authentication import TokenAuthentication
from .models import UserData
from .serializers import UserDataSerializer

@authentication_classes([TokenAuthentication])
@permission_classes([IsAuthenticated])
class UserDataListCreateView(generics.ListCreateAPIView):
    queryset = UserData.objects.all()
    serializer_class = UserDataSerializer
    def perform_create(self, serializer):
     # Save the user making the request
        serializer.save(name=self.request.user.username)

@authentication_classes([TokenAuthentication])
@permission_classes([IsAuthenticated])
class UserDataDetailView(generics.RetrieveUpdateDestroyAPIView):
    queryset = UserData.objects.all()
    serializer_class = UserDataSerializer
```

5. Define API Endpoints (api/urls.py):

```
from django.urls import path
from .views import UserDataListCreateView, UserDataDetailView
urlpatterns = [
    path('data/', UserDataListCreateView.as_view(), name='data-list-create'),
    path('data/<int:pk>/', UserDataDetailView.as_view(), name='data-detail'),
    ]
```

6. Implement Authentication (api/authentication.py):

```
from rest_framework.authentication import TokenAuthentication
from rest_framework.permissions import IsAuthenticated


class CustomTokenAuthentication(TokenAuthentication):
    def authenticate_credentials(self, key):
        model = self.get_model()
        try:
            user, token = super().authenticate_credentials(key)
        except Exception:
            return None
        return user, token


class IsOwnerOrReadOnly(IsAuthenticated):
    def has_object_permission(self, request, view, obj):
        if request.method in ['GET', 'HEAD', 'OPTIONS']:
            return True
        return obj.name == request.user.username
```

7. Update Settings (myapi/settings.py):

Add the following configurations:

```
INSTALLED_APPS = [
    # ...
    'api',
    'rest_framework',
    'rest_framework.authtoken',
    ]
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'api.authentication.CustomTokenAuthentication',
        ],
'DEFAULT_PERMISSION_CLASSES': [
        'api.authentication.IsOwnerOrReadOnly',
        ],
    }
```

8. Apply Migrations and Runserver:

```
$ python manage.py makemigrations
$ python manage.py migrate
$ python manage.py createsuperuser  # Follow the prompts to create a superuser
$ python manage.py runserver
```