LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Title

CS6P05 Final Year Project MAD

Food Share - Android App

Artifact – web application development

Student Details

Name: Sita Ram Thing

London Met Id: 22015892

College Id: NP01MA4S220003

Islington College, Kathmandu

24 April 2024

Contents

List of figure

1. Introduction

1.1 Django Rest Framework

Figure 1: Django rest framework

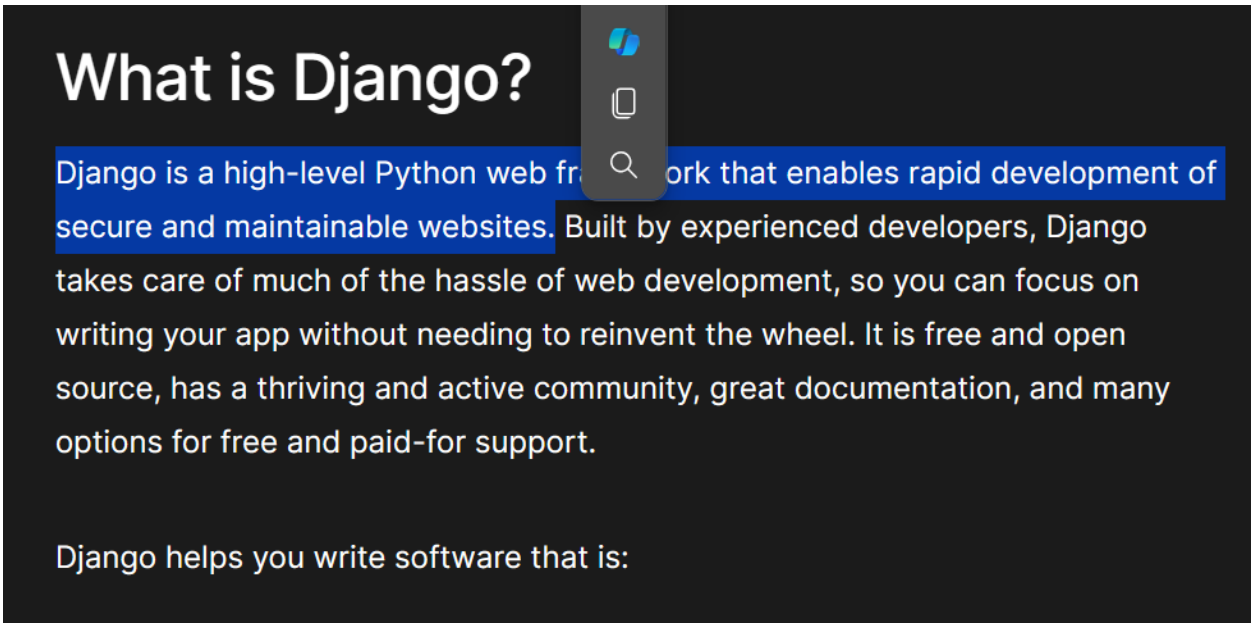


Figure 2: Introduction about the Django

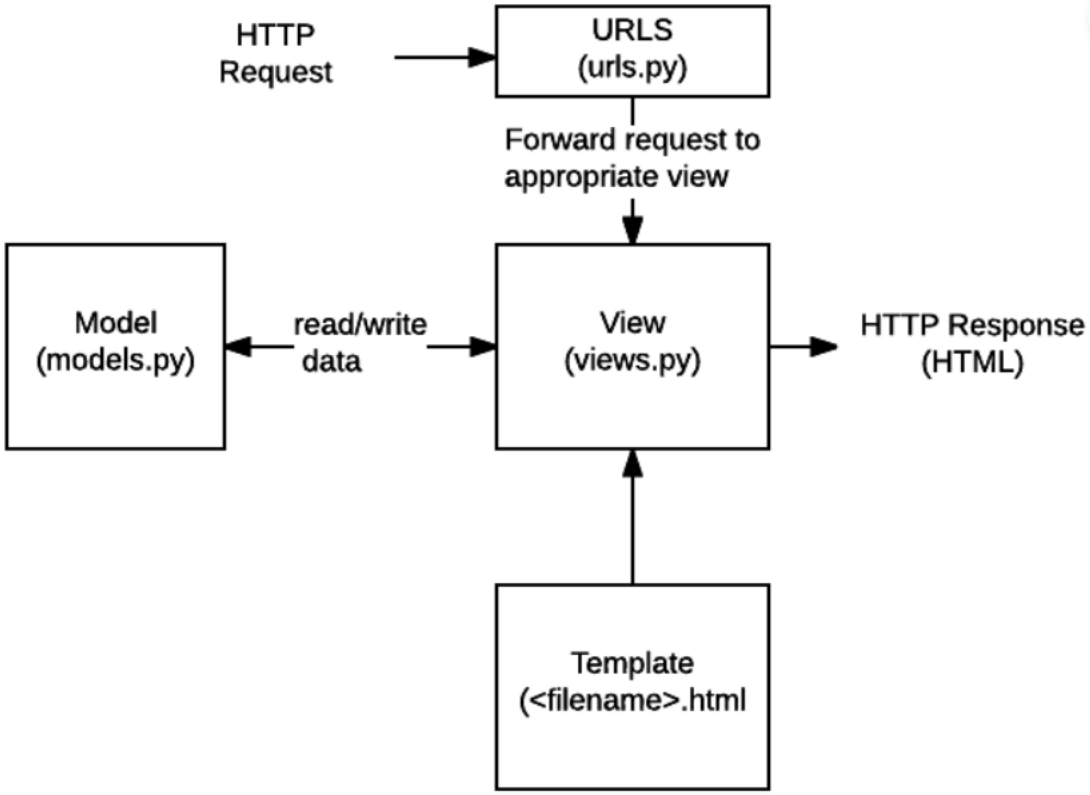


Figure 3: Http connection

Install using `pip`, including any optional packages you want...

```
pip install djangorestframework
pip install markdown       # Markdown support for the browsable API.
pip install django-filter  # Filtering support
```

...or clone the project from github.

```
git clone https://github.com/encode/django-rest-framework
```

Add `'rest_framework'` to your `INSTALLED_APPS` setting.

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

Figure 4: Install the Django

If you're intending to use the browsable API you'll probably also want to add REST framework's login and logout views. Add the following to your root `urls.py` file.

```
urlpatterns = [
    ...
    path('api-auth/', include('rest_framework.urls'))
]
```

Note that the URL path can be whatever you want.

Figure 5: Api URL generate.

```python
from django.urls import path, include
from django.contrib.auth.models import User
from rest_framework import routers, serializers, viewsets

# Serializers define the API representation.
class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ['url', 'username', 'email', 'is_staff']

# ViewSets define the view behavior.
class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer

# Routers provide an easy way of automatically determining the URL conf.
router = routers.DefaultRouter()
router.register(r'users', UserViewSet)

# Wire up our API using automatic URL routing.
# Additionally, we include login URLs for the browsable API.
urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework'))
]
```

Figure 6: Model serializer

Let's take a look at a quick example of using REST framework to build a simple model-backed API.

We'll create a read-write API for accessing information on the users of our project.

Any global settings for a REST framework API are kept in a single configuration dictionary named `REST_FRAMEWORK`. Start off by adding the following to your `settings.py` module:

```
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly'
    ]
}
```

Figure 7: Set the authenticate on permission

1.1 Rest API Development

1.2.1 Models

```python
# create custom User table AbstractBaseUser
class Users(AbstractBaseUser):
    email = models.EmailField(verbose_name='Email', max_length=255, unique=True)
    username = models.CharField(max_length=100)
    role = models.CharField(max_length=10, null=True)

    address = models.CharField(max_length=100, null=True)
    contact_number = models.CharField(max_length=16, unique=True, null=True)
    gender = models.CharField(max_length=10, null=True)
    date_of_birth = models.DateField(default=datetime.now, null=True)
    abouts_user = models.TextField(max_length=500, null=True)
    photo_url = models.ImageField(upload_to='user_images/', null=True, max_length=500)

    is_admin = models.BooleanField(default=False)
    is_active = models.BooleanField(default=False)
    ngo = models.ForeignKey(Ngo, on_delete=models.CASCADE, null=True) # FK (donor id)
    created_by = models.CharField(max_length=100, null=True, default='Self')
    created_date = models.DateField(auto_now_add=True)
    modify_by = models.CharField(max_length=50, null=True)
    modify_date = models.DateField(null=True)
    is_delete = models.BooleanField(default=False)
```

Figure 8: Users model.

```python
# crate the Food model
class Food(models.Model):
    FOOD_TYPE_CHOICES = (
        ('Others', 'Others'),
        ('Cake', 'Cake'),
        ('Green vegetables', 'Green vegetables'),
        ('Biscuits & Chocolates', 'Biscuits & Chocolates'),
        ('Sweet Snack', 'Sweet Snack'),
        ('Stable Food', 'Stable Food'),
        ('Fruits', 'Fruits'),
        ('Meets', 'Meets'),
        ('Water & Cold Drinks', 'Water & Cold Drinks'),
    )
    STATUS_CHOICES = (('New', 'New'), ('Pending', 'Pending'), ('Completed', 'Completed'),)

    food_name = models.CharField(max_length=100)
    food_types = models.CharField(max_length=50, choices=FOOD_TYPE_CHOICES, default='Others')
    quantity = models.IntegerField(null=True)
    expire_time = models.CharField(max_length=10, null=True)
    pick_up_location = models.CharField(max_length=100)
    latitude = models.DecimalField(max_digits=22, decimal_places=16, default=0.0)
    longitude = models.DecimalField(max_digits=22, decimal_places=16, default=0.0)
    descriptions = models.TextField(null=True)
    stream_url = models.ImageField(upload_to='food_images/', null=True, max_length=500)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='New')
    created_by = models.CharField(max_length=100, null=True)
    created_date = models.DateField(auto_now_add=True)
    modify_by = models.CharField(max_length=50, null=True)
    modify_date = models.DateField(null=True)
    donor = models.ForeignKey(Users, on_delete=models.CASCADE, null=True) # FK (donor id)
    is_delete = models.BooleanField(default=False)
```

Figure 9: Food Model

```python
# Reports
class Report(models.Model):
    COMPLAINT_CHOICES = (
        ('donor', 'Complaint to Donor'),
        ('volunteer', 'Complaint to Volunteer'),
        ('farmer', 'Complaint to Farmer'),
    )
    complaint_to  = models.CharField(max_length=30, choices=COMPLAINT_CHOICES, null=True)
    descriptions = models.TextField(null=True)
    is_verify = models.BooleanField(default=False)
    created_by = models.CharField(max_length=50, null=True)
    created_date = models.DateField(auto_now_add=True)
    modify_by = models.CharField(max_length=50, null=True)
    modify_date = models.DateField(null=True)
    is_delete = models.BooleanField(default=False)
    food = models.ForeignKey(Food, on_delete=models.CASCADE) #FK
```

Figure 10: Report Model

```python
# create the History table
class History(models.Model):
    STATUS_CHOICES = (('Pending', 'Pending'), ('Completed', 'Completed'),)

    descriptions = models.TextField(max_length=300, null=True)
    distributed_location = models.CharField(max_length=100, null=True)
    rating_point = models.IntegerField(validators=[MinValueValidator(0), MaxValueValidator(5)], default=0)
    distributed_date = models.DateField(null=True)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='Pending')
    created_by = models.CharField(max_length=50, null=True)
    created_date = models.DateField(auto_now_add=True)
    modify_by = models.CharField(max_length=50, null=True)
    modify_date = models.DateField(null=True)
    is_delete = models.BooleanField(default=False)
    volunteer = models.ForeignKey(Users, on_delete=models.CASCADE, null=True) # FK (volunteer id)
    food = models.ForeignKey(Food, on_delete=models.CASCADE, null=True) # FK (food id)

    # create the notification table
class Notification(models.Model):
    title = models.TextField(null=True)
    descriptions = models.TextField(null=True)
    created_by = models.CharField(max_length=100, null=True)
    created_date = models.DateField(auto_now_add=True)
    is_delete = models.BooleanField(default=False)
    food = models.ForeignKey(Food, on_delete=models.CASCADE) # FK

class Device(models.Model):
    token = models.TextField(null=True)
    created_by = models.CharField(max_length=100, null=True)
    created_date = models.DateField(auto_now_add=True)
    is_delete = models.BooleanField(default=False)
    user = models.ForeignKey(Users, on_delete=models.CASCADE) # FK
```

Figure 11: History, Notification, Device model.

1.2.2 Serialize

```python
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = Users
        exclude = ('password',)  # Exclude the password field

    def validate_image(self, value):
        try:
            get_image_dimensions(value)
        except AttributeError:
            raise serializers.ValidationError("The uploaded file is not recognized as a valid image.")
        except ValidationError:
            raise serializers.ValidationError("The uploaded image is either not an image or a corrupted image.
        return value

    def to_representation(self, instance):
        representation = super().to_representation(instance)
        modify_date = representation.get('modify_date')

        if modify_date:
            if isinstance(modify_date, str):  # Check if modify_date is a string
                modify_date = datetime.strptime(modify_date, '%Y-%m-%d')

            representation['modify_date'] = modify_date.date()

        return representation
```

Figure 12: User serializer.

```python
class NgoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Ngo
        fields = "__all__"

class FoodSerializer(serializers.ModelSerializer):
    class Meta:
        model = Food
        fields = '__all__'

class HistorySerializer(serializers.ModelSerializer):
    class Meta:
        model = History
        fields = '__all__'

class ReportSerializer(serializers.ModelSerializer):
    class Meta:
        model = Report
        fields = '__all__'

class NotificationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Notification
        fields = '__all__'

class DeviceSerializer(serializers.ModelSerializer):
    class Meta:
        models = Device
        fields = '_all_'

```

Figure 13: Serializer class.

1.2.3 URL

```python
from django.urls import path, include
from django.contrib import admin
from rest_framework.routers import DefaultRouter
from foodshare.views import * # import all from views
(variable) urlpatterns: list views import *

urlpatterns = [
    # by default api
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls')),
    path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('api/token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
    path('api/token/verify//', TokenVerifyView.as_view(), name='token_verify'),

    # apis for mobile
    path('api/authenticate/token', LoginUser.as_view(), name='get_authentication_token'),
    path('api/register/user', RegisterUser.as_view(), name='get_register_user'),
    path('api/logout/user', LogoutView.as_view(), name='get_logout_user'),
    # path('api/is/token/expired', TokenExpired.as_view(), name='is_token_expired'),

    # Profile
    path('api/user/profile', UserProfile.as_view(), name='get_user_profile'),
    path('api/update/profile', UpdateProfile.as_view(), name='set_update_profile'),
    path('api/update/profile/image', UpdateProfilePicture.as_view(), name='set_update_profile'),
    path('api/fcm/device/token/save', DeviceTokenView.as_view(), name='set_device_details'),

    # NGOs api
    path('api/ngo/profile', NgoProfile.as_view(), name='get_ngo_profile'),
    path('api/register/ngo', AddNgoView.as_view(), name="add_ngo_profile"),
    # users
    path('api/all/types/user', GetAllUsersView.as_view(), name='get_users_by_search'),
    path('api/account/activate', UserAccountActivateView.as_view(), name='set_account_activate'),

    # password
    path('api/email/verify', EmailVerify.as_view(), name='get_email_verify'),
    path('api/update/password', UpdatePassword.as_view(), name='set_update_password'),

    # manage account
    path('api/account/delete', DeleteAccount.as_view(), name='set_delete_account'),
```

Figure 14: Api URL development.

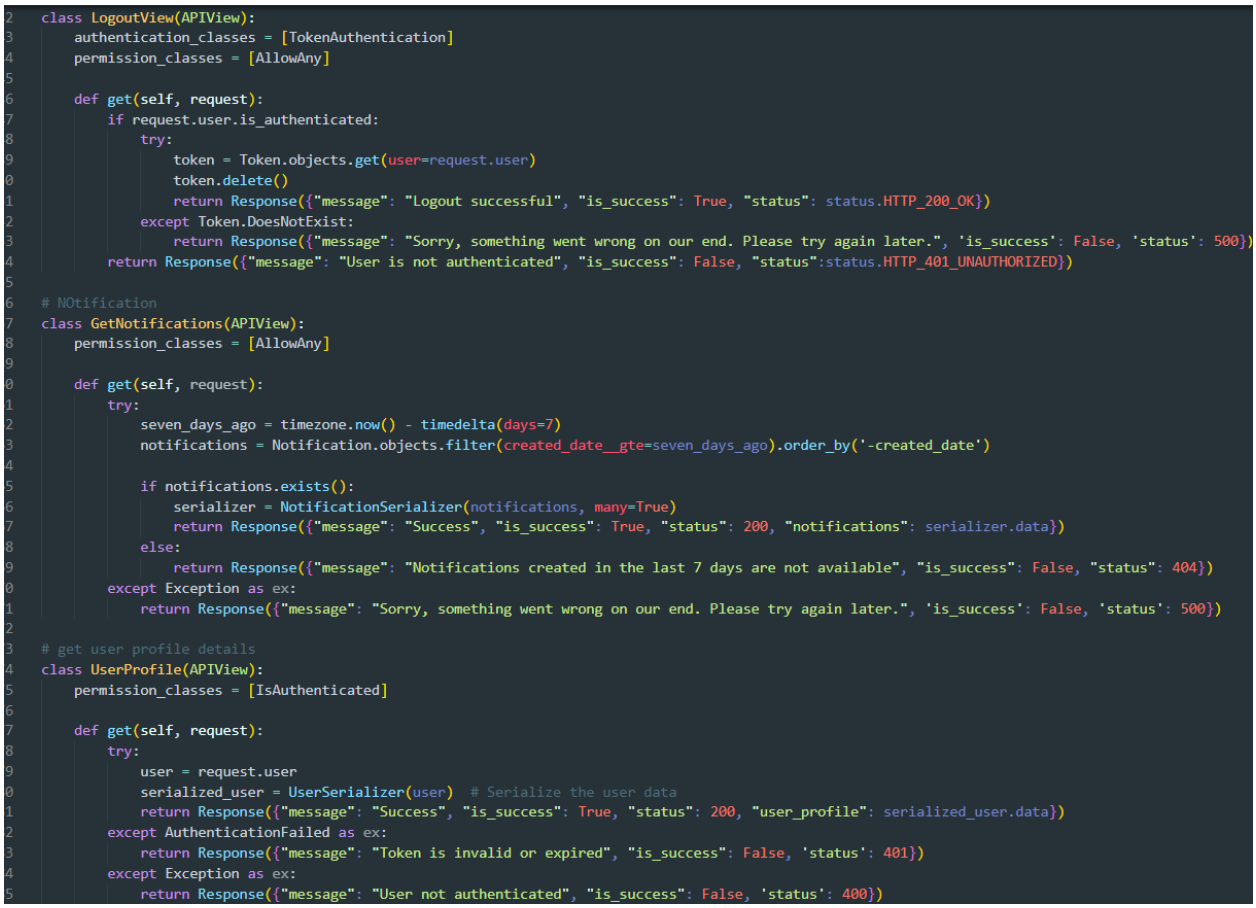
1.1.4 API Function

```python
# login authentication
class LoginUser(APIView):
    authentication_classes = [TokenAuthentication]
    permission_classes = [AllowAny]

    def post(self, request):
        try:
            email = request.data.get('email')
            password = request.data.get('password')

            if email and password:
                auth_user = authenticate(request, username=email, password=password)
                if auth_user:
                    user = Users.objects.filter(email=email).first()
                    if auth_user.is_active and not user.is_delete:  # Fix the condition
                        refresh = RefreshToken.for_user(auth_user)
                        access_token = str(refresh.access_token)

                        serializer = UserSerializer(user)
                        profile = user.photo_url.url if user.photo_url else None

                        response_auth = {
                            'id': serializer.data['id'],
                            'username': serializer.data['username'],
                            'email': serializer.data['email'],
                            'profile': profile,
                            'role': serializer.data['role'],
                            'access_token': access_token,
                        }
                        return Response({'message': 'Login successful', 'is_success': True, 'status': 200, "auth": response_auth})
                    else:
                        return Response({'message': 'Your account is not activate', 'is_success': False, 'status': 401})
                else:
                    return Response({'message': 'The account does not have authentication permission.', 'is_success': False, 'status': 401})
            else:
                return Response({'message': 'Please provide email and password', 'is_success': False, 'status': 400})
        except Exception as e:
            return Response({"message": "Sorry, something went wrong on our end. Please try again later.", 'is_success': False, 'status': 500})
```

Figure 15: Login API function.

```python
# Register user
class RegisterUser(APIView):
    authentication_classes = [TokenAuthentication]  # Ensure TokenAuthentication is included
    permission_classes = [AllowAny]

    def post(self, request):
        try:
            serializer = UserSerializer(data=request.data)
            if serializer.is_valid():
                # Extract validated data from the serializer
                email = serializer.validated_data['email']
                username = serializer.validated_data['username']
                role = serializer.validated_data['role']
                password = request.data.get('password')
                # Register user
                try:
                    Users.objects.create_user(email=email, username=username, role=role, password=password)
                    return Response({'message': 'User registered successfully', 'is_success': True, 'status': 200, 'system_token': settings.SECRET_KEY})
                except Exception as e:
                    return Response({'message': "Invali user", 'is_success': False, 'status': 500})
            else:
                return Response({'message': 'Enter a valid email address!', 'is_success': False, 'status': 400})
        except Exception as e:
            return Response({"message": "Sorry, something went wrong on our end. Please try again later.", 'is_success': False, 'status': 500})
```

Figure 16: Register User



```python
class LogoutView(APIView):
    authentication_classes = [TokenAuthentication]
    permission_classes = [AllowAny]

    def get(self, request):
        if request.user.is_authenticated:
            try:
                token = Token.objects.get(user=request.user)
                token.delete()
                return Response({"message": "Logout successful", "is_success": True, "status": status.HTTP_200_OK})
            except Token.DoesNotExist:
                return Response({"message": "Sorry, something went wrong on our end. Please try again later.", 'is_success': False, 'status': 500})
        return Response({"message": "User is not authenticated", "is_success": False, "status":status.HTTP_401_UNAUTHORIZED})

# NOtification
class GetNotifications(APIView):
    permission_classes = [AllowAny]

    def get(self, request):
        try:
            seven_days_ago = timezone.now() - timedelta(days=7)
            notifications = Notification.objects.filter(created_date__gte=seven_days_ago).order_by('-created_date')

            if notifications.exists():
                serializer = NotificationSerializer(notifications, many=True)
                return Response({"message": "Success", "is_success": True, "status": 200, "notifications": serializer.data})
            else:
                return Response({"message": "Notifications created in the last 7 days are not available", "is_success": False, "status": 404})
        except Exception as ex:
            return Response({"message": "Sorry, something went wrong on our end. Please try again later.", 'is_success': False, 'status': 500})

# get user profile details
class UserProfile(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request):
        try:
            user = request.user
            serialized_user = UserSerializer(user)  # Serialize the user data
            return Response({"message": "Success", "is_success": True, "status": 200, "user_profile": serialized_user.data})
        except AuthenticationFailed as ex:
            return Response({"message": "Token is invalid or expired", "is_success": False, 'status': 401})
        except Exception as ex:
            return Response({"message": "User not authenticated", "is_success": False, 'status': 400})
```

Figure 17: Logout, Get Notification and Update profile details file