Django Internship Assignment Report

Submitted By:
R.AKHILDEV REDDY
Applied for: Django Developer Intern
Date: 23 April 2025

Agenda:
The project requires designing a backend system to handle users, clients, and projects with the correct relationships and validation rules and operational capabilities.

Tech Stack
Framework: Django, Django REST Framework (DRF)
Language: Python
Database: MySQL/PostgreSQL
Tools: Postman for API Testing, Django Admin for user management

Project Setup:
- Established a new project through Django along with an associated app.
- The default database in settings.py used MySQL as its configuration.
- The Django REST Framework was installed before its setup as a system.
- Django User definition and custom models for Client and Project have been implemented.
- Used Django's default User model.

Models structure:

User
Default Django auth.User

Client
id: Auto-generated
client_name: CharField
created_by: ForeignKey (User)
created_at: DateTimeField
updated_at: DateTimeField

Project
id: Auto-generated
project_name: CharField
client: ForeignKey (Client)
users: ManyToManyField (User)

created_by: ForeignKey (User)
created_at: DateTimeField

API Endpoints

Clients
GET /clients/: List all clients.
POST /clients/: Create a new client.
GET /clients/<id>/: Retrieve client info with assigned projects.
PUT/PATCH /clients/<id>/: Update client info.
DELETE /clients/<id>/: Delete client.

Projects
POST /projects/: Create a new project with client ID and user assignments.
GET /projects/: Get all projects assigned to the logged-in user.
DELETE /projects/<id>/: Delete a project.

Code:

1. **Requirements.txt:**

```
Django==4.2.7
djangorestframework==3.14.0
mysqlclient==2.2.0
```

2. **settings.py:**

```python
from pathlib import Path
BASE_DIR = Path(__file__).resolve().parent.parent
SECRET_KEY = 'not to disclose'
DEBUG = True
ALLOWED_HOSTS = []
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'rest_framework.authtoken',
    'api',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'client_project_manager.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'client_project_manager.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'client_project_db',
        'USER': 'root',
        'PASSWORD': 'Akhil@456',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        },
    }
}
```

```python
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
        'rest_framework.authentication.SessionAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
}
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
STATIC_URL = 'static/'
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

3. client_project_manager/urls.py:

```python
from django.contrib import admin
from django.urls import path, include
from rest_framework.authtoken import views as token_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls')),
    path('api-token-auth/', token_views.obtain_auth_token),
    path('api-auth/', include('rest_framework.urls')),
]
```

4. api/Models.py:

```python
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone


class Client(models.Model):
    client_name = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    created_by = models.ForeignKey(User, on_delete=models.CASCADE, related_name='created_clients')

    def __str__(self):
        return self.client_name


class Project(models.Model):
    project_name = models.CharField(max_length=100)
    client = models.ForeignKey(Client, on_delete=models.CASCADE, related_name='projects')
    users = models.ManyToManyField(User, related_name='assigned_projects')
    created_at = models.DateTimeField(auto_now_add=True)
    created_by = models.ForeignKey(User, on_delete=models.CASCADE, related_name='created_projects')

    def __str__(self):
        return f"{self.project_name} - {self.client.client_name}"
```

5. api/serializers.py:

```python
from rest_framework import serializers
from django.contrib.auth.models import User
from .models import Client, Project

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username']

class ProjectListSerializer(serializers.ModelSerializer):
    client_name = serializers.CharField(source='client.client_name', read_only=True)
    created_by = serializers.CharField(source='created_by.username', read_only=True)

    class Meta:
        model = Project
        fields = ['id', 'project_name', 'client_name', 'created_at', 'created_by']

class ProjectSerializer(serializers.ModelSerializer):
    id = serializers.IntegerField(read_only=True)
    project_name = serializers.CharField(required=True)
    client = serializers.PrimaryKeyRelatedField(read_only=True)
    client_id = serializers.IntegerField(write_only=True)
    users = UserSerializer(many=True, read_only=True)
    client_name = serializers.SerializerMethodField()
    created_by = serializers.SerializerMethodField()

    class Meta:
        model = Project
        fields = ['id', 'project_name', 'client', 'client_id', 'client_name', 'users', 'created_at', 'created_by']
    def get_client_name(self, obj):
        return obj.client.client_name
    def get_created_by(self, obj):
        return obj.created_by.username
    def create(self, validated_data):
        client_id = validated_data.pop('client_id')
        users_data = self.context['request'].data.get('users', [])
```

```python
try:
            client = Client.objects.get(id=client_id)
            project = Project.objects.create(
                project_name=validated_data['project_name'],
                client=client,
                created_by=self.context['request'].user
            )
            if isinstance(users_data, list):
                for user_id in users_data:
                    try:
                        user = User.objects.get(id=user_id)
                        project.users.add(user)
                    except User.DoesNotExist:
                        pass
            return project
        except Client.DoesNotExist:
            raise serializers.ValidationError("Client with this ID does not exist")

class ProjectDetailSerializer(serializers.ModelSerializer):
    id = serializers.IntegerField(read_only=True)
    name = serializers.CharField(source='project_name')
    class Meta:
        model = Project
        fields = ['id', 'name']

class ClientSerializer(serializers.ModelSerializer):
    created_by = serializers.ReadOnlyField(source='created_by.username')
    class Meta:
        model = Client
        fields = ['id', 'client_name', 'created_at', 'created_by']
        read_only_fields = ['id', 'created_at', 'created_by']

class ClientDetailSerializer(serializers.ModelSerializer):
    projects = ProjectDetailSerializer(many=True, read_only=True)
    created_by = serializers.ReadOnlyField(source='created_by.username')
    class Meta:
        model = Client
        fields = ['id', 'client_name', 'projects', 'created_at', 'created_by', 'updated_at']

```

6. api/views.py:

```python
from rest_framework import viewsets, status
from rest_framework.response import Response
from rest_framework.permissions import IsAuthenticated
from django.contrib.auth.models import User
from .models import Client, Project
from .serializers import (
    ClientSerializer,
    ClientDetailSerializer,
    ProjectSerializer,
    ProjectListSerializer
)

class ClientViewSet(viewsets.ModelViewSet):
    queryset = Client.objects.all()
    permission_classes = [IsAuthenticated]
    def get_serializer_class(self):
        if self.action == 'retrieve':
            return ClientDetailSerializer
        return ClientSerializer
    def perform_create(self, serializer):
        serializer.save(created_by=self.request.user)

class ProjectViewSet(viewsets.ModelViewSet):
    permission_classes = [IsAuthenticated]
    def get_serializer_class(self):
        if self.action == 'list':
            return ProjectListSerializer
        return ProjectSerializer
    def get_queryset(self):
        return Project.objects.filter(users=self.request.user)
    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        print("Users data:", request.data.get('users'))
        project = serializer.save()
        response_data = {
            'id': project.id,
            'project_name': project.project_name,
            'client': project.client.client_name,
            'users': [{'id': user.id, 'name': user.username} for user in project.users.all()],
            'created_at': project.created_at,
            'created_by': project.created_by.username
            }
        return Response(response_data, status=status.HTTP_201_CREATED)
```

7. **api/urls.py:**

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import ClientViewSet, ProjectViewSet

router = DefaultRouter()
router.register(r'clients', ClientViewSet)
router.register(r'projects', ProjectViewSet, basename='project')

urlpatterns = [
    path('', include(router.urls)),
]
```

8. **api/admin.py:**

```python
from django.contrib import admin
from .models import Client, Project

admin.site.register(Client)
admin.site.register(Project)
```

Database:

MySQL Workbench:

Navigator

SCHEMAS

Filter objects

- client_project_db
  - Tables
    - api_client
    - api_project
    - api_project_users
    - auth_group
    - auth_group_permissions
    - auth_permission
    - auth_user
    - auth_user_groups
    - auth_user_user_permissions
    - authtoken_token
    - django_admin_log
    - django_content_type
    - django_migrations
    - django_session
  - Views

Administration Schemas

auth_user | api_project | **api_client** | api_project_users

Limit to 1000 rows

```
1 • SELECT * FROM client_project_db.api_client;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	client_name	created_at	updated_at	created_by_id
1	Nimap	2025-04-23 06:43:48.816350	2025-04-23 06:43:48.816350	1
2	Infotech	2025-04-23 06:43:57.105015	2025-04-23 06:43:57.105015	1
NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor

Navigator

SCHEMAS

Filter objects

- client_project_db
  - Tables
    - api_client
    - api_project
    - api_project_users
    - auth_group
    - auth_group_permissions
    - auth_permission
    - auth_user
    - auth_user_groups
    - auth_user_user_permissions
    - authtoken_token
    - django_admin_log
    - django_content_type
    - django_migrations
    - django_session
  - Views

Administration Schemas

auth_user | **api_project** | api_client | api_project_users

Limit to 1000 rows

```
1 • SELECT * FROM client_project_db.api_project;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	project_name	created_at	client_id	created_by_id
1	Project A	2025-04-23 06:44:25.381948	1	1
2	Project B	2025-04-23 06:44:40.069337	2	1
NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor

Navigator

SCHEMAS

Filter objects

- client_project_db
  - Tables
    - api_client
    - api_project
    - api_project_users
    - auth_group
    - auth_group_permissions
    - auth_permission
    - auth_user
    - auth_user_groups
    - auth_user_user_permissions
    - authtoken_token
    - django_admin_log
    - django_content_type
    - django_migrations
    - django_session
  - Views

Administration Schemas

auth_user | api_project | api_client | **api_project_users**

Limit to 1000 rows

```
1 • SELECT * FROM client_project_db.api_project_users;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	project_id	user_id
1	1	1
2	1	2
3	2	2
4	2	3
NULL	NULL	NULL

Result Grid | Form Editor

Navigator

SCHEMAS

Filter objects

- client_project_db
  - Tables
    - api_client
    - api_project
    - api_project_users
    - auth_group
    - auth_group_permissions
    - auth_permission
    - auth_user
    - auth_user_groups
    - auth_user_user_permissions
    - authtoken_token
    - django_admin_log
    - django_content_type
    - django_migrations
    - django_session
  - Views

Administration Schemas

auth_user | api_project | api_client | api_project_users

Limit to 1000 rows

```
1 • SELECT * FROM client_project_db.auth_user;
```

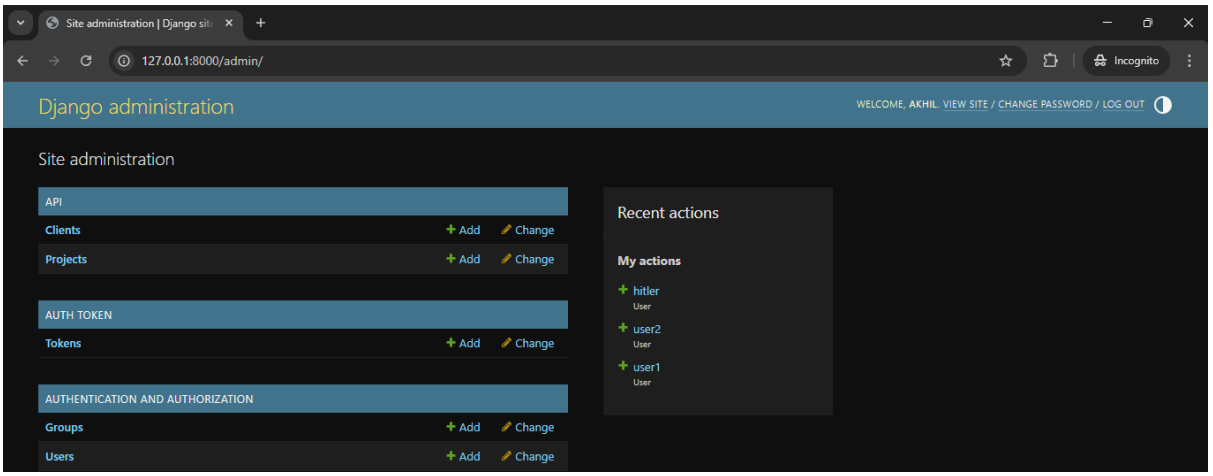
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	password	last_login	is_superuser	username	first_name
1	pbkdf2_sha256$1000000$WckDavnNb5fxKQFC...	2025-04-23 06:41:41.896298	1	akhil	
2	pbkdf2_sha256$1000000$XuXPDFs5eQ3LmL04...	2025-04-23 06:50:25.532884	0	user1	
3	pbkdf2_sha256$1000000$njLL6arhxSoc2KwXE5...	2025-04-23 06:52:01.702324	0	user2	
4	pbkdf2_sha256$1000000$yfCwHj5N9xMRmmgf...	NULL	0	hitler	
NULL	NULL	NULL	NULL	NULL	NULL

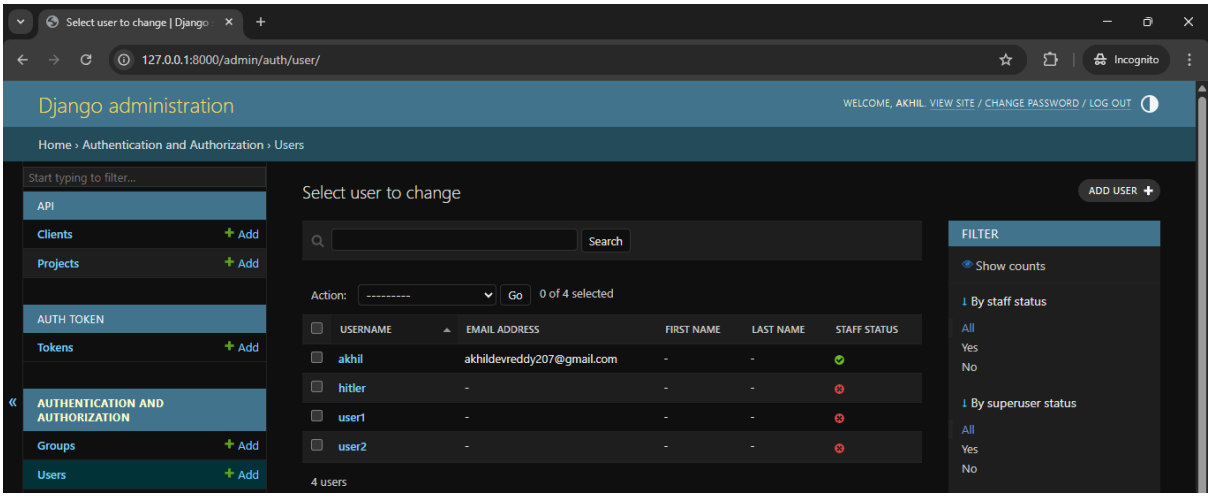
Result Grid | Form Editor

Outputs:

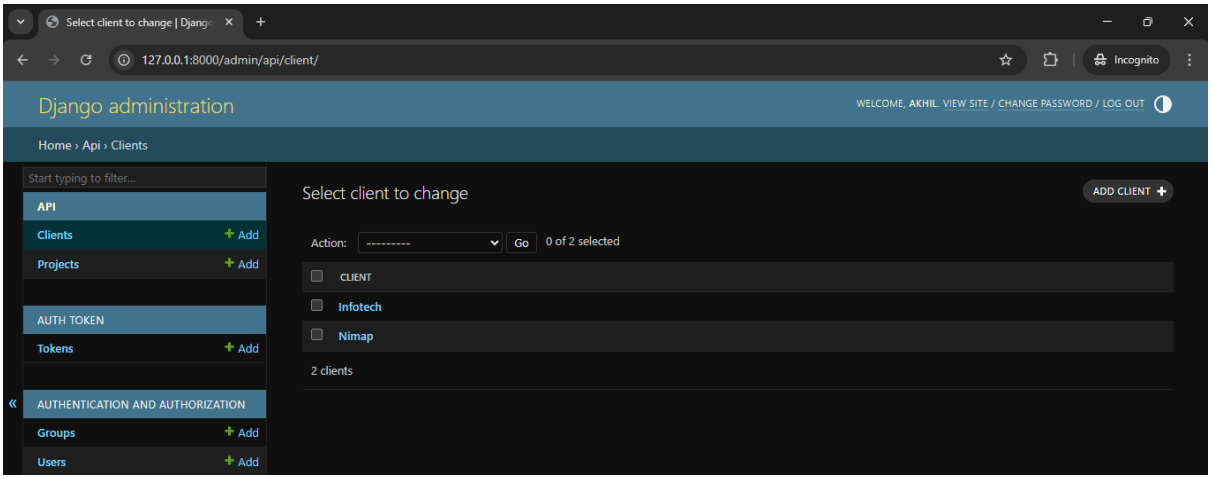
Admin-panel:



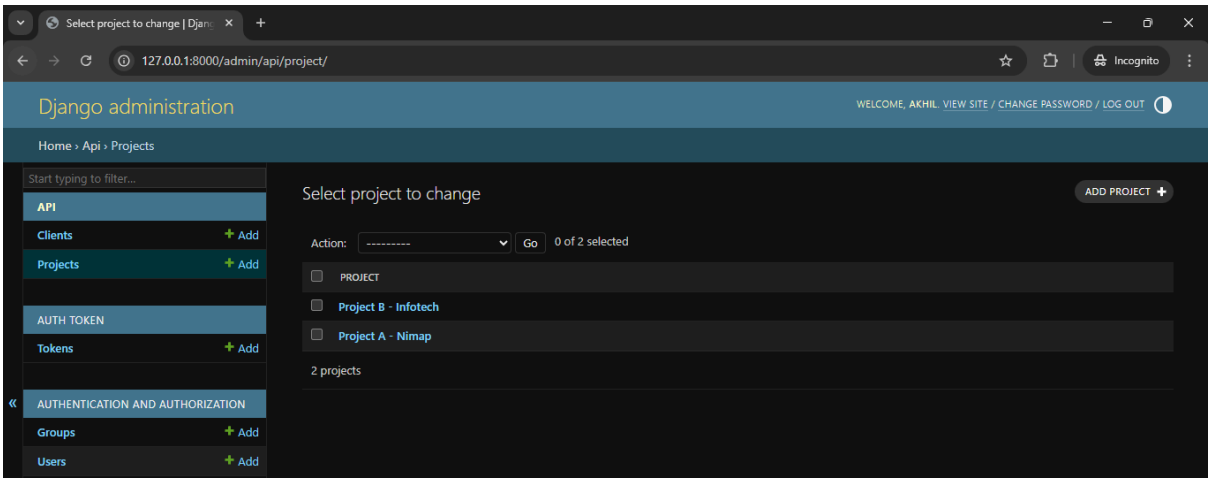
User:



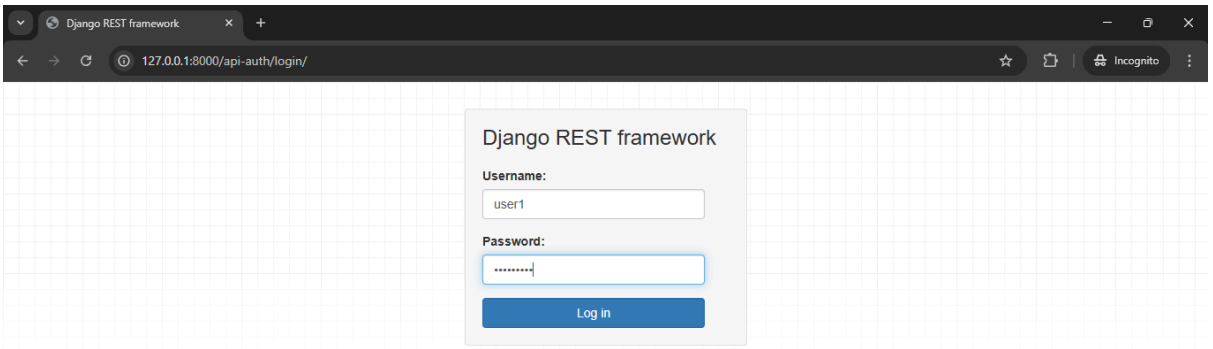
Clients:



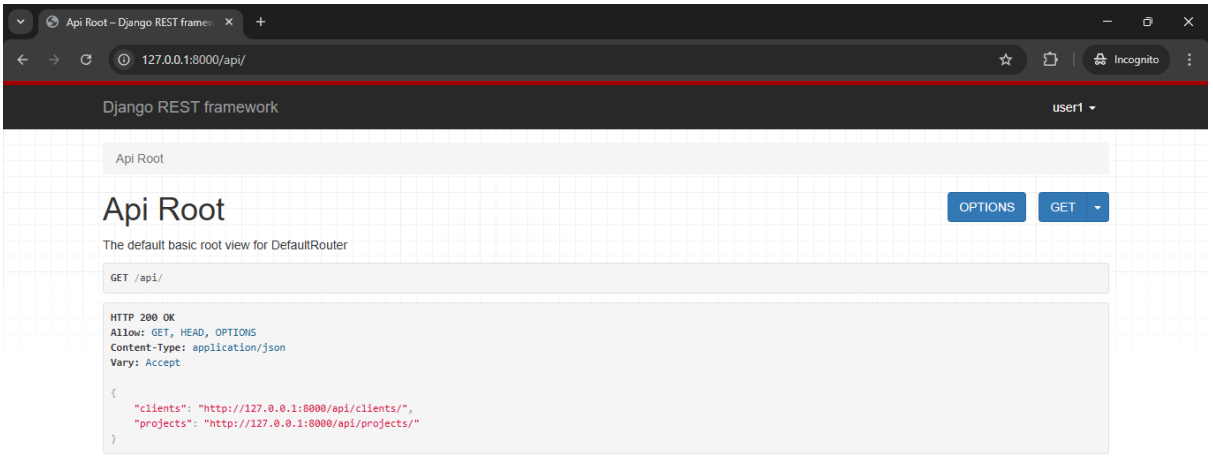
Projects:



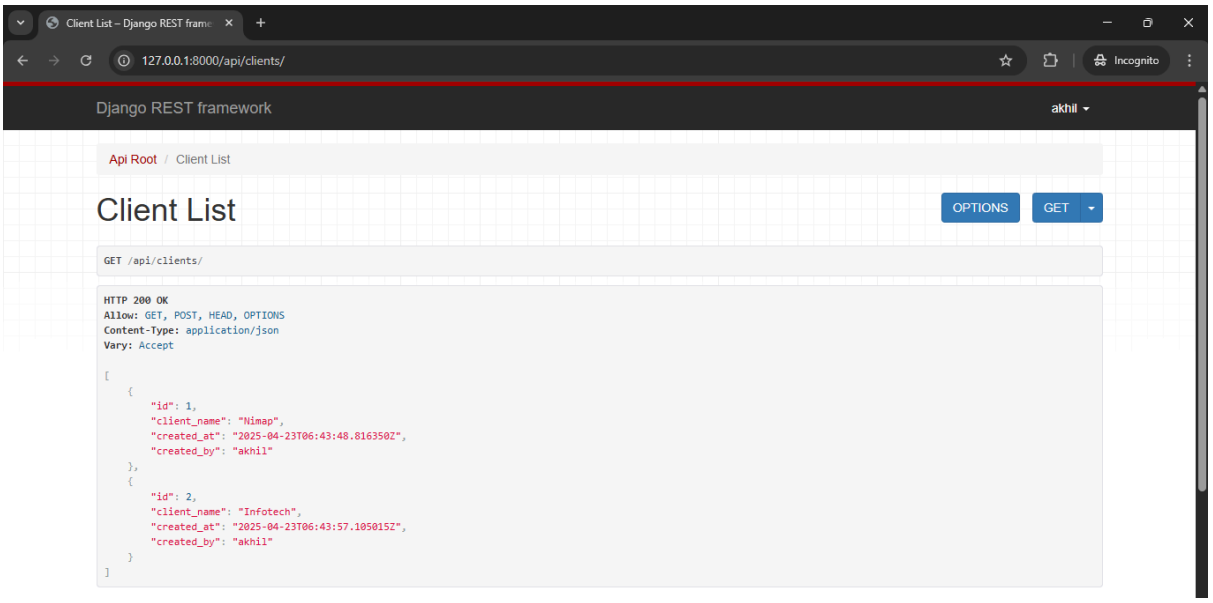
Login:



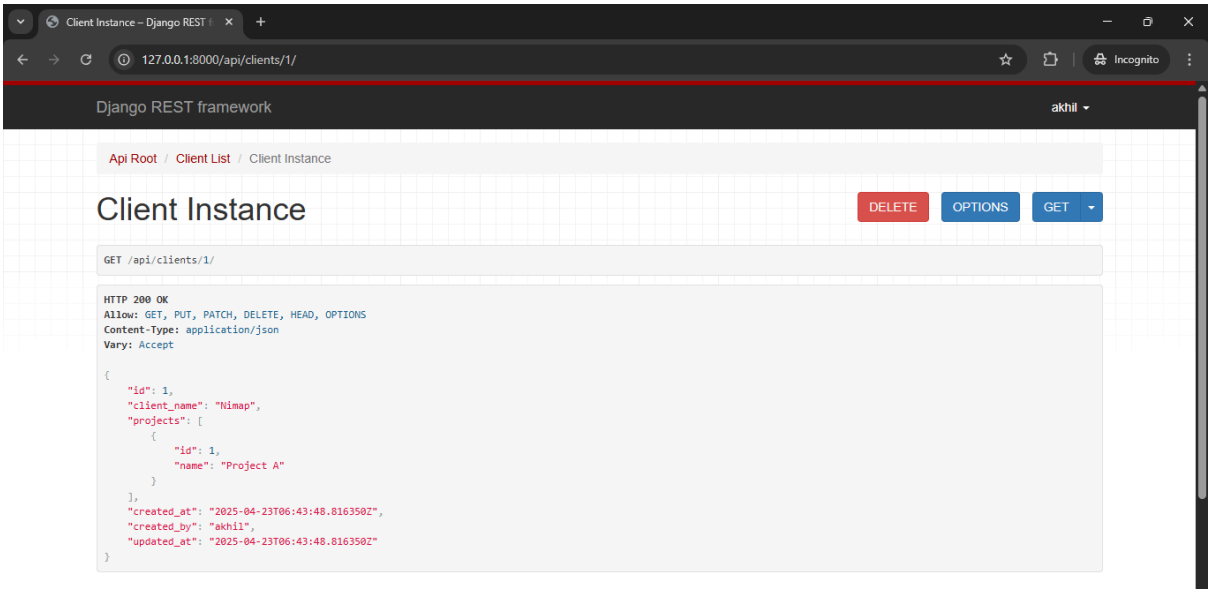
Api root:



Client list: (assosiated to that particular user)



Client instance:



Project list: (projects associated to that particular user)