Project Report

1. Introduction

This project demonstrates two backend implementations — one using Django REST Framework and another using FastAPI — for handling basic CRUD operations on a form that stores Name, Phone, and Address. Both versions are connected to a PostgreSQL database and deployed online via Render.

2.Objective

To build and deploy a Django-based backend system with RESTful APIs, PostgreSQL integration, and a fully functional admin panel. The project is hosted on Render and tested using Postman.

3.Technology Stack

- Backend: Python, Django
- Database: PostgreSQL
- API Framework: Django REST Framework (DRF)
- Deployment: Render.com
- Version Control: Git & GitHub
- API Testing: Postman
- Admin Interface: Django Admin Panel

4.Features Implemented

- REST API with GET & POST endpoints
- Admin panel to manage records (FormData)
- PostgreSQL database integration
- API tested using Postman
- Deployment on Render with start.sh and render.yaml
- Environment variables securely configured

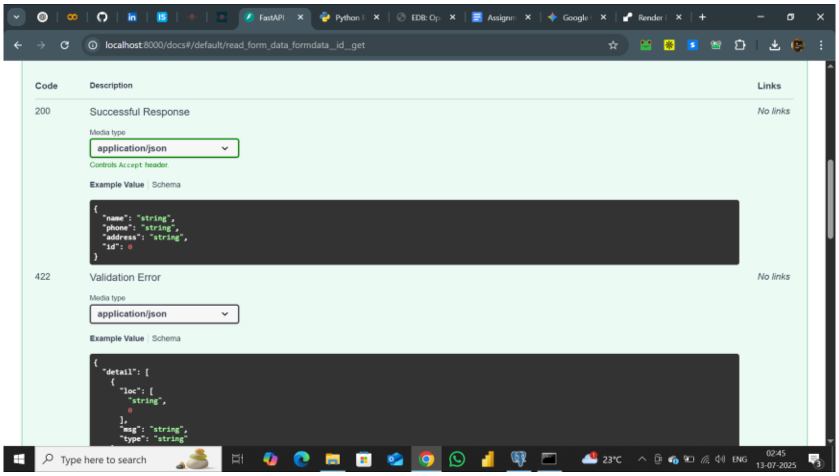
5.Screenshots

Include the following:
Below are the added screenshots demonstrating project functionality:

- Admin panel login
- FormData entries
- Postman API tests
- Live deployed site (homepage/API)

5.1. FastAPI Implementation

The FastAPI version of the project was developed and deployed successfully. Below is the Swagger UI screenshot.



localhost:8000/docs#/default/read_form_data_formdata__id__get

Code	Description	Links
200	Successful Response	No links
422	Validation Error	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "name": "string",
  "phone": "string",
  "address": "string",
  "id": 0
}
```

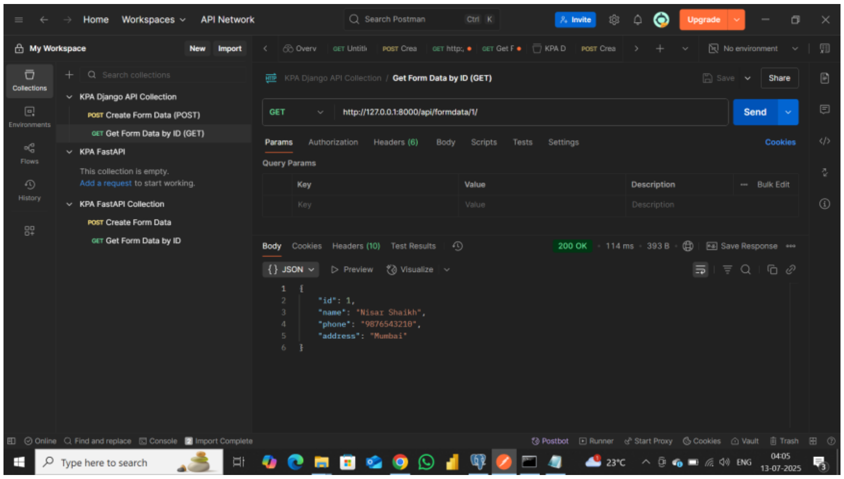
Media type: application/json

Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
```

5.2. Testing with Postman

This screenshot shows how the API was tested in Postman using GET and POST requests.



KPA Django API Collection / Get Form Data by ID (GET)

GET http://127.0.0.1:8000/api/formdata/1/

Query Params

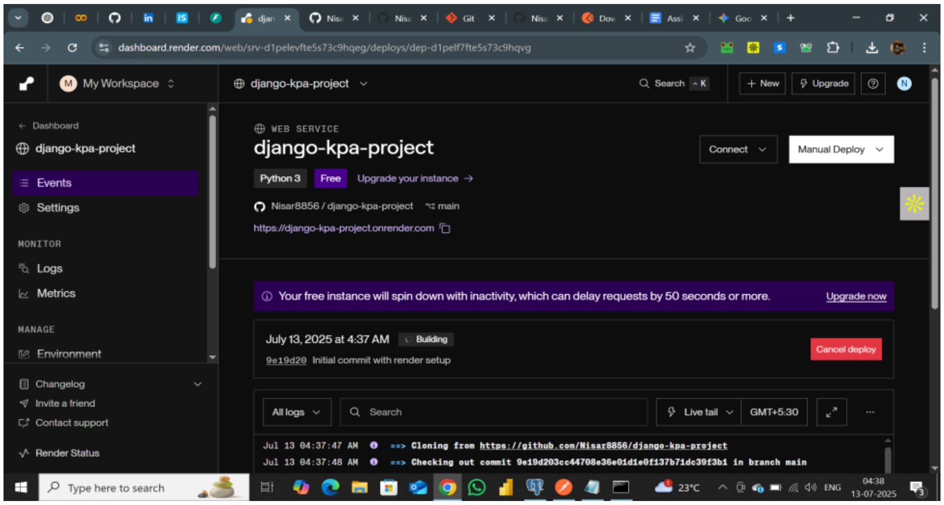
Key	Value	Description
Key	Value	Description

200 OK · 114 ms · 393 B

```
{
    "id": 1,
    "name": "Nisar Shaikh",
    "phone": "9876543210",
    "address": "Mumbai"
}
```

5.3. Render Deployment

The following screenshots highlight the deployment process of the FastAPI and Django projects on Render.



dashboard.render.com/web/srv-d1pelevfte5s73c9hqeg/deploys/dep-d1pelf7fte5s73c9hqvg

WEB SERVICE

django-kpa-project

Python 3 · Free · Upgrade your instance →

Nisar8856 / django-kpa-project · main

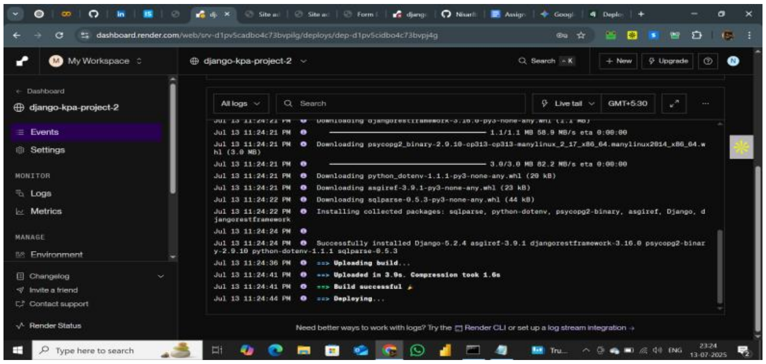
https://django-kpa-project.onrender.com

Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. Upgrade now

July 13, 2025 at 4:37 AM · Building

9e19d20 Initial commit with render setup

```
Jul 13 04:37:47 AM  ==> Cloning from https://github.com/Nisar8856/django-kpa-project
Jul 13 04:37:48 AM  ==> Checking out commit 9e19d203cc44708e36e01d1e0f137b71dc39f3b1 in branch main
```

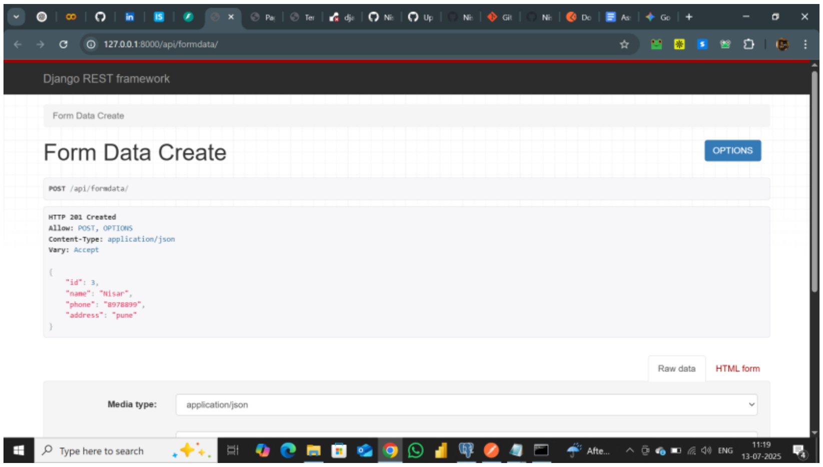


```
Jul 13 11:24:21 PM    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 58.9 MB/s eta 0:00:00
Jul 13 11:24:21 PM    Downloading psycopg2_binary-2.9.10-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.w
hl (3.0 MB)
Jul 13 11:24:21 PM    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 82.2 MB/s eta 0:00:00
Jul 13 11:24:21 PM    Downloading python_dotenv-1.1.1-py3-none-any.whl (20 kB)
Jul 13 11:24:21 PM    Downloading asgiref-3.9.1-py3-none-any.whl (23 kB)
Jul 13 11:24:22 PM    Downloading sqlparse-0.5.3-py3-none-any.whl (44 kB)
Jul 13 11:24:22 PM    Installing collected packages: sqlparse, python-dotenv, psycopg2-binary, asgiref, Django, d
jangorestframework
Jul 13 11:24:24 PM
Jul 13 11:24:24 PM    Successfully installed Django-5.2.4 asgiref-3.9.1 djangorestframework-3.16.0 psycopg2-binar
y-2.9.10 python-dotenv-1.1.1 sqlparse-0.5.3
Jul 13 11:24:36 PM    ==> Uploading build...
Jul 13 11:24:41 PM    ==> Uploaded in 3.9s. Compression took 1.6s
Jul 13 11:24:41 PM    ==> Build successful 🎉
Jul 13 11:24:44 PM    ==> Deploying...
```

Need better ways to work with logs? Try the Render CLI or set up a log stream integration →

5.4. Django REST Framework Implementation

The Django API interface is shown below:



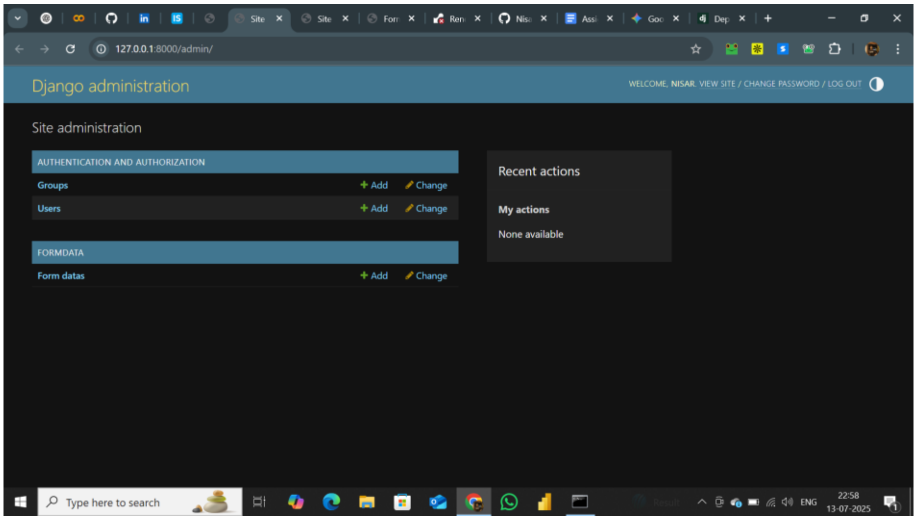
Form Data Create

```
POST /api/formdata/

HTTP 201 Created
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 3,
    "name": "Nisar",
    "phone": "8978899",
    "address": "pune"
}
```

Media type: application/json

5.5. Django Admin Panel

This screenshot confirms that FormData is registered and visible in the admin dashboard:



Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

Users

FORMDATA

Form datas

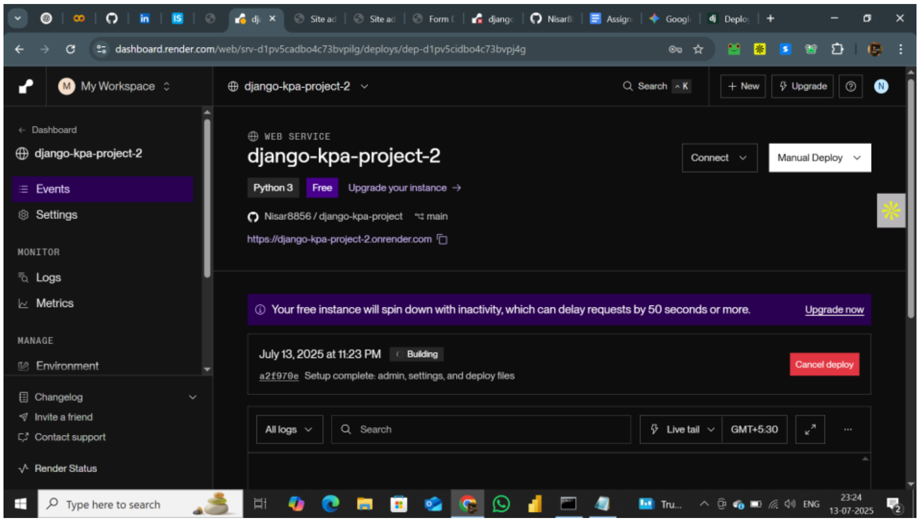
Recent actions

My actions

None available

5.6. Django Render Final Dashboard

Final successful deployment of the Django version on Render:



dashboard.render.com/web/srv-d1pv5cadbo4c73bvpilg/deploys/dep-d1pv5cidbo4c73bvpj4g

WEB SERVICE

django-kpa-project-2

Python 3 Free Upgrade your instance →

Nisar8856 / django-kpa-project main

https://django-kpa-project-2.onrender.com

Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. Upgrade now

July 13, 2025 at 11:23 PM Building

a2f970e Setup complete: admin, settings, and deploy files

Cancel deploy

API Endpoints

- GET /api/formdata/ → Fetch all records
- POST /api/formdata/ → Submit a new record
- Admin /admin/ → Admin login interface

Deployment

- Platform: Render.com
- Live URL: https://your-project-name.onrender.com
- GitHub Repo: https://github.com/Nisar8856/django-kpa-project

Learning Outcomes

- Hands-on experience with Django backend and ORM
- Setting up and using PostgreSQL with Django
- Cloud deployment using Render
- Managing environment variables for production
- Git-based version control and collaboration

Conclusion

This project provided a full-cycle backend development experience — from coding and testing to deployment and presentation. It simulates real-world software delivery and prepares me for production-ready backend roles.