MYSORE UNIVERSITY SCHOOL OF ENGINEERING

Manasagangotri campus, Mysuru570006
(Approved by AICTE, New Delhi)

UNIVERSITY OF MYSORE

Full Stack Development(21CD71) Assessment Report On:

"Multi-Page Blogging System"

Under the guidance :

Mr. Karthik M N
Assistant Professor,

Department of Computer Science
& Design, MUSE.

Submitted by

Niharika.B
Reg No : 21SECD26

Introduction: The Multi-Page Blogging System is a Django-based web application designed to allow customer to write blog posts containing a title, author, content, and published date efficiently. This report details the modifications made to improve the overall user experience (UX) while maintaining system functionality

Project overview:

blog_project/

├── blog

│ ├── __init__.py
│ ├── admin.py
│ ├── apps.py
│ ├── models.py
│ ├── views.py
│ ├── urls.py
│ ├── tests.py

│

├── templates/
│ ├── blog/
│ │ ├── blog_detail.html
│ │ ├── blog_form.html
│ │ ├── blog_list.html

│

```
├── blog_project/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   └── asgi.py
├── static/
│   ├── images/
│   │   └── pineapple.jpg
│
├── staticfiles
│   ├── CSS
│   ├── img
│   └── JS
├── db.sqlite3
├── manage.py
└── README.md
```

Detailed steps Implementation:

Step 1: Set Up Django Project and Application

• Create a new Django project using django-admin startproject complaints_project.
• Navigate into the project directory and create an app using django-admin startapp complaints.

Step 2: Install Dependencies

• Ensure Django is installed using pip install django.

Step 3: Configure Project Settings

• Add the complaints app to INSTALLED_APPS in settings.py.
• Configure templates and static files for styling.

Step 4: Create a Form for a Multi-Page Blogging System

• Define a Blogging in forms.py to handle user input

Step 5:Create Templates (`templates/blog/`)

- `blog_list.html`: Display all blog posts
- `blog_detail.html`: Show a single blog post
- `blog_form.html`: Allow users to create a new blog post

Step 6:Configure Static Files & Media Storage (`settings.py`)

- Set up `STATIC_URL` and `MEDIA_URL`
- Ensure images and CSS files are properly loaded

Step 7: Test & Run Server

- Start the server: `python manage.py runserver`
- Verify all views and functionalities are working as expected

Step 8: Deployment & Final Testing

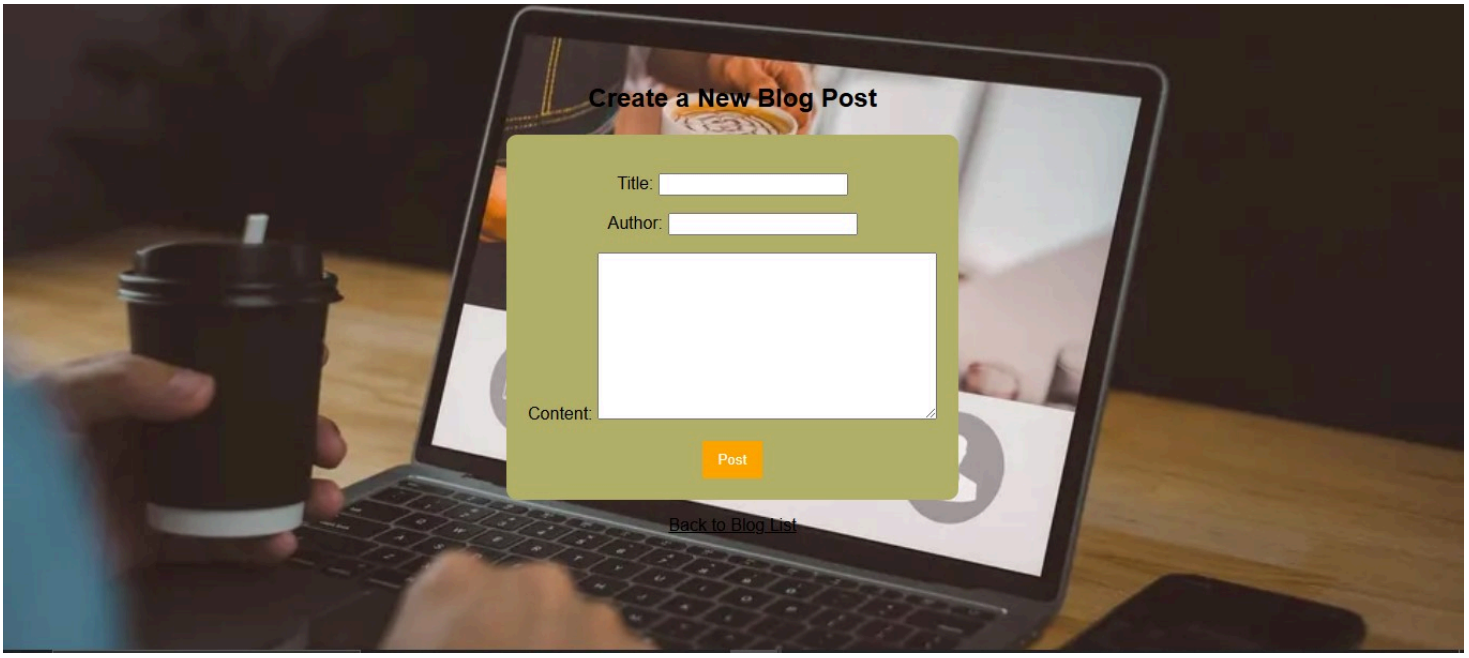
- Use `collectstatic`: `python manage.py collectstatic`
- Ensure all static and media files are accessible
- Deploy on a hosting platform (e.g., Heroku, PythonAnywhere)

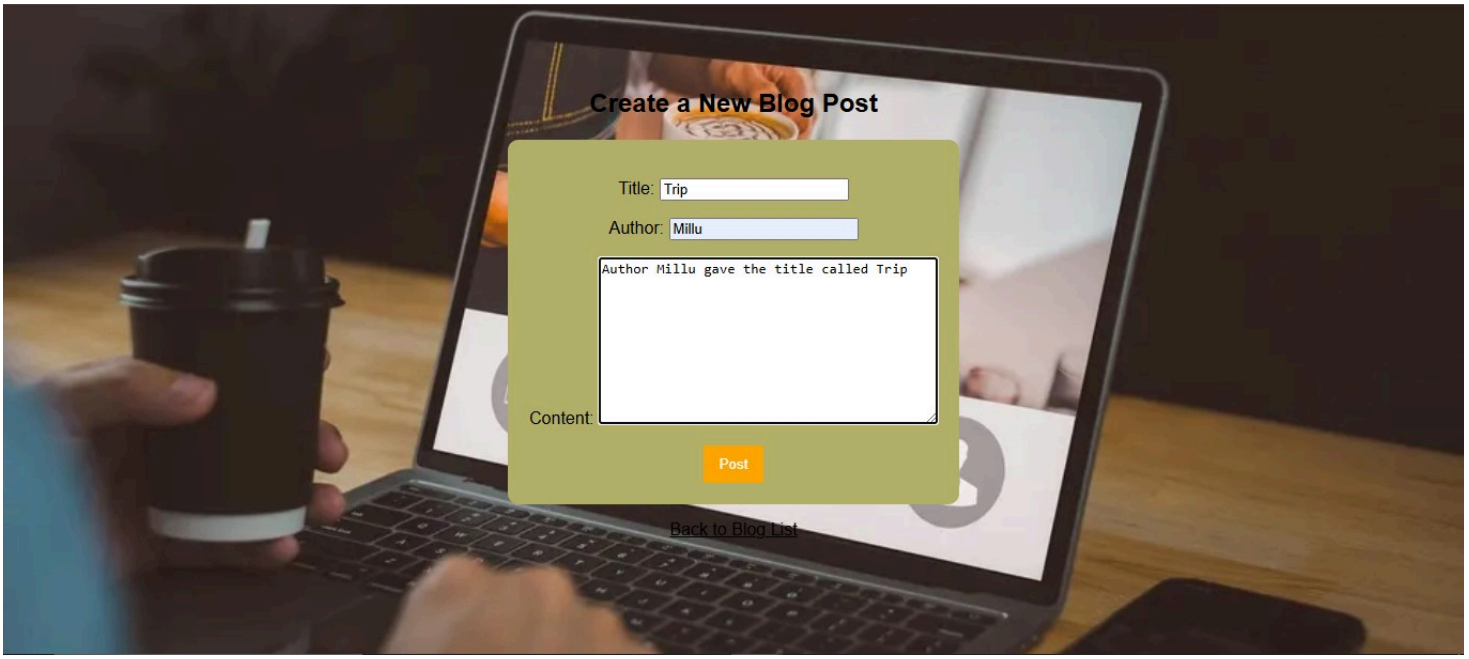
Conclusion

In this project, we successfully developed a **multi-page blogging system** using Django. The system allows users to create, view, and list blog posts efficiently with **Django's generic views (ListView, DetailView, CreateView)**. By implementing structured **URL configurations**.

Additionally, we utilized **reverse_lazy()** for smooth redirections and properly configured **static and media files** for handling images and styling. The use of Django's built-in features helped streamline development while ensuring a scalable and maintainable application.

Output:


Create a New Blog Post
Title:
Author:
Content:
Post
Back to Blog List


Create a New Blog Post
Title: Trip
Author: Millu
Author Millu gave the title called Trip
Content:
Post
Back to Blog List

guyhv

Author: mnn jhv

Published Date: Feb. 22, 2025, 8:41 p.m.

mn jgchgc jv

---

vycxrv

Author: vvtycr6tyvy

Published Date: Feb. 23, 2025, 8:33 a.m.

.mnugcxextcvbvxcfvgh

Trip

Author: Millu

Published Date: Feb. 23, 2025, 10:42 a.m.

the title is Trip and Author is Millu

Create New Blog