MYSORE UNIVERSITY SCHOOL OF ENGINEERING

Manasagangotri campus, Mysuru-570006 (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:
PALLAVI.M
Reg No : 21SECD30

Introduction:

Blogging platforms allow users to create, publish, and manage articles online. Django, a powerful Python web framework, simplifies web development by providing reusable components. This guide walks you through building a multi-page blogging system where users can write and view blog posts.

Multi-Page Blogging System with the following features:

 Users can write blog posts containing a title, author, content, and published date.

 Django’s generic CreateView, ListView, and DetailView to manage blogs.

Project overview:

```
BLOG_PROJECT1/
|── blog/
│   ├── __pycache__/
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   ├── views.py
│   ├── templates/
│   │   └── blog/
│   │       ├── blog_form.html
│   │       ├── blog_detail.html
│   │       ├── blog_list.html
│
|── blog_project/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── asgi.py
│   ├── wsgi.py
│
│── static/
│   ├── images
│   │       ├── blogimage.jpg
│
│── staticfiles/
│   ├── admin/
│   │   ├── css/
│   │   │   ├── autocomplete.css
│   │   │   ├── base.css
│   │   │   ├── changelists.css
│   │   │   ├── dark_mode.css
│   │   │   ├── dashboard.css
│   │   │   ├── forms.css
│   │   │   ├── login.css
│   │   │   ├── nav_sidebar.css
│   │   │   ├── responsive.css
│   │   │   ├── responsive_rtl.css
│   │   │   ├── rtl.css
│   │   │   ├── unusable_password_field.css
│   │   │   ├── widgets.css
│   │   │   └── vendor/
│   │   │       └── (vendor CSS files)
│   │   │
│   │   ├── img/
│   │   │   ├── gis/
│   │   │   ├── calendar-icons.svg
│   │   │   ├── icon-addlink.svg
│   │   │   ├── icon-alert.svg
│   │   │   ├── icon-calendar.svg
│   │   │   ├── icon-changelink.svg
│   │   │   ├── icon-clock.svg
│   │   │   ├── icon-deletelink.svg
│   │   │   ├── icon-hidelink.svg
│   │   │   ├── icon-no.svg
│   │   │   ├── icon-unknown-alt.svg
│   │   │   ├── icon-unknown.svg
│   │   │   ├── icon-viewlink.svg
│   │   │   ├── icon-yes.svg
│   │   │   ├── inline-delete.svg
│   │   │   ├── LICENSE
│   │   │   ├── README.txt
│   │   │   ├── search.svg
│   │   │   ├── selector-icons.svg
│   │   │   ├── sorting-icons.svg
│   │   │   ├── tooltag-add.svg
│   │   │   └── tooltag-arrowright.svg
│   │   │
│   │   ├── js/
│   │   │   ├── admin/
│   │   │   │   ├── DateTimeShortcuts.js
│   │   │   │   ├── RelatedObjectLookups.js
│   │   │   │
│   │   │   ├── vendor/
│   │   │   │   ├── jquery/
│   │   │   │   ├── select2/
│   │   │   │   ├── xregexp/
│   │   │   │
│   │   │   ├── actions.js
│   │   │   ├── autocomplete.js
│   │   │   ├── calendar.js
│   │   │   ├── cancel.js
│   │   │   ├── change_form.js
│   │   │   ├── core.js
│   │   │   ├── filters.js
│   │   │   ├── inlines.js
│   │   │   ├── jquery.init.js
│   │   │   ├── nav_sidebar.js
│   │   │   ├── popup_response.js
│   │   │   ├── prepopulate.js
│   │   │   ├── prepopulate_init.js
│   │   │   ├── SelectBox.js
│   │   │   ├── SelectFilter2.js
│   │   │   ├── theme.js
│   │   │   ├── unusable_password_field.js
│   │   │   └── urlify.js
│
│── manage.py
│── db.sqlite3
```

Detailed steps Implementation:

Step 1: Install Django and Create a Virtual Environment

# Create a virtual environment

python -m venv venv

# Activate the virtual environment

# On Windows:

venv\Scripts\activate

# On macOS/Linux:

source venv/bin/activate

# Install Django

pip install Django

Step 2: Create a Django Project

Run the following command to create a Django project:

django-admin startproject BLOG_PROJECT1

cd BLOG_PROJECT1

Step 3: Create a Django App

python manage.py startapp blog

Step 4: Configure settings.py

Open *BLOG_PROJECT1/settings.py* and add *'blog'* to *INSTALLED_APPS*

Step 5: Create the blogModel:

Run migrations to apply the model:

python manage.py makemigrations

python manage.py migrate

Step 6: Register the Model in Django Admin:

In *blog/admin.py:*

Step 7: Create Views for blog Management:

In *blog/views.py*

Step 8: Configure URLs:

Create *blog/urls.py*
Link the *blog* app to the project's main *urls.py* in *BLOG_PROJECT1/urls.py*

Step 9: Create HTML Templates:

1. *blog_form.html*

2. blog_detail.html

3. blog_list.html

Step 10: Create a Superuser for Admin Panel:

python manage.py createsuperuser

Step 11: Run the Django Development Server

python manage.py runserver

Conclusion

You have successfully created a **Django-based Multi-Page Blogging System** with:

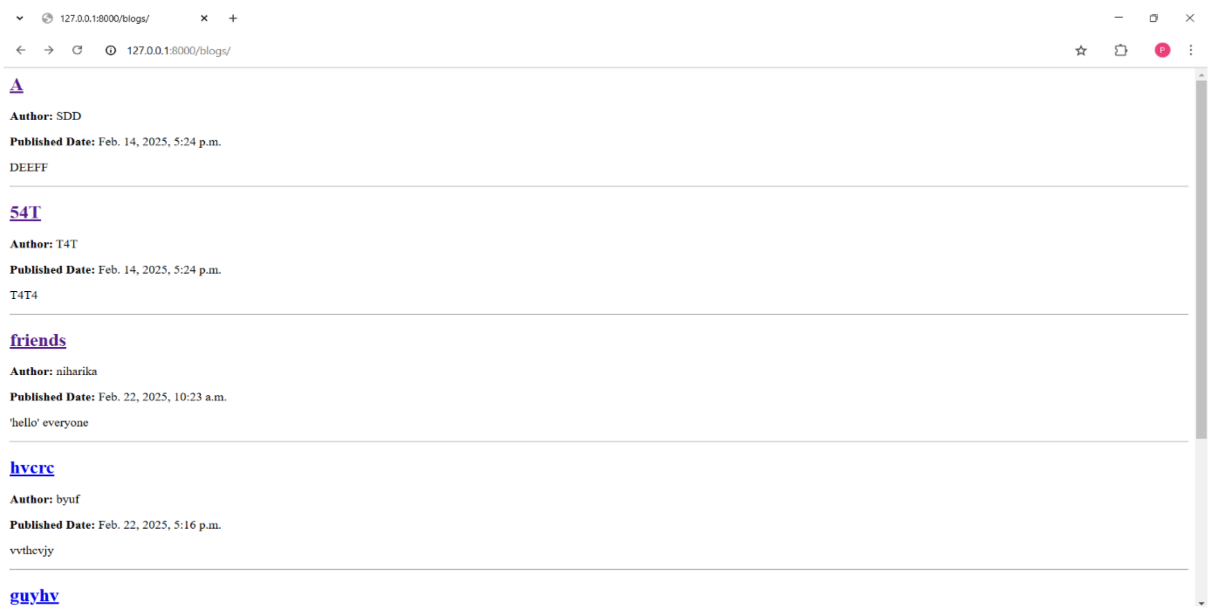
 Blog posts containing a title, author, content, and published date.

 Create, List and Detail views using Django's generic CreateView, ListView & DetailView .

 Users can create and post their blogs

 reverse_lazy() to redirect users to the blog list after successfully posting an article.

Output:



A

Author: SDD

Published Date: Feb. 14, 2025, 5:24 p.m.

DEEFF

54T

Author: T4T

Published Date: Feb. 14, 2025, 5:24 p.m.

T4T4

friends

Author: niharika

Published Date: Feb. 22, 2025, 10:23 a.m.

'hello' everyone

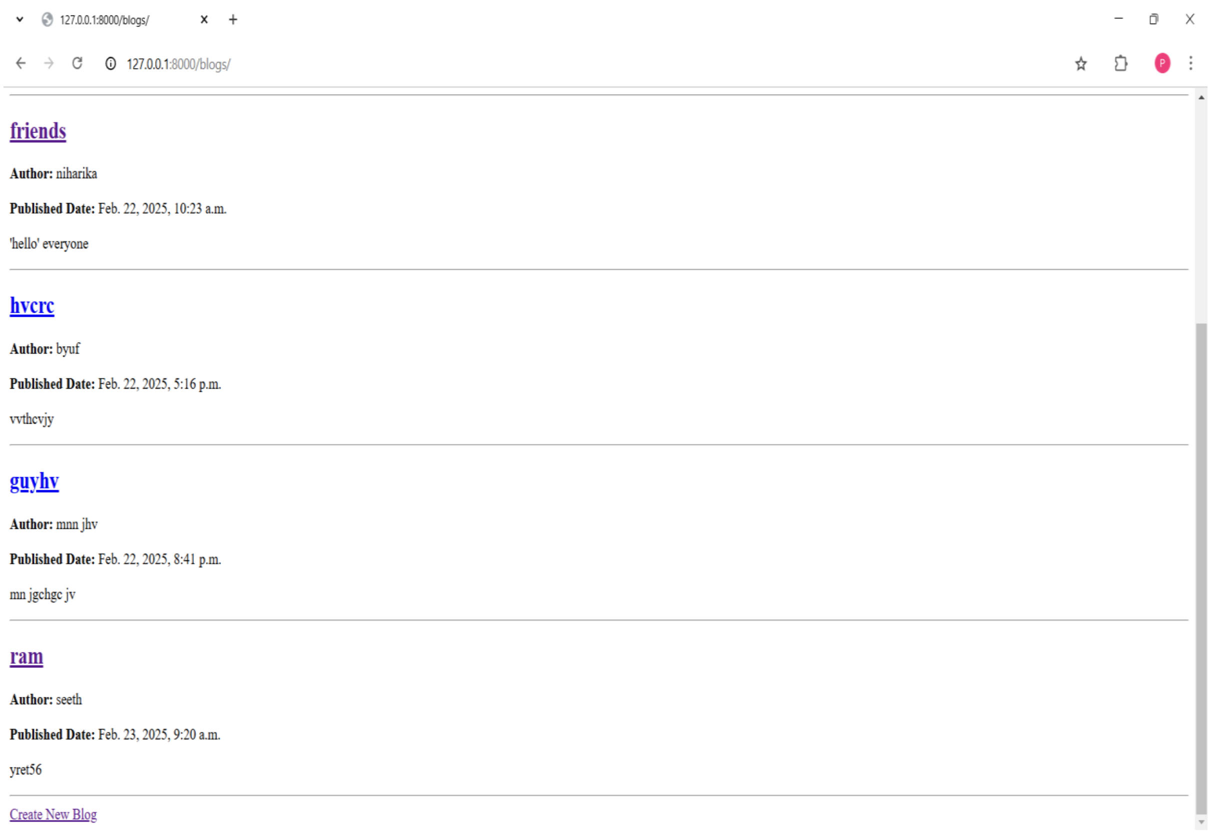
hvcrc

Author: byuf

Published Date: Feb. 22, 2025, 5:16 p.m.

vvthcvjy

guyhv



friends

Author: niharika

Published Date: Feb. 22, 2025, 10:23 a.m.

'hello' everyone

hvcrc

Author: byuf

Published Date: Feb. 22, 2025, 5:16 p.m.

vvthcvjy

guyhv

Author: mnn jhv

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

mn jgchgc jv

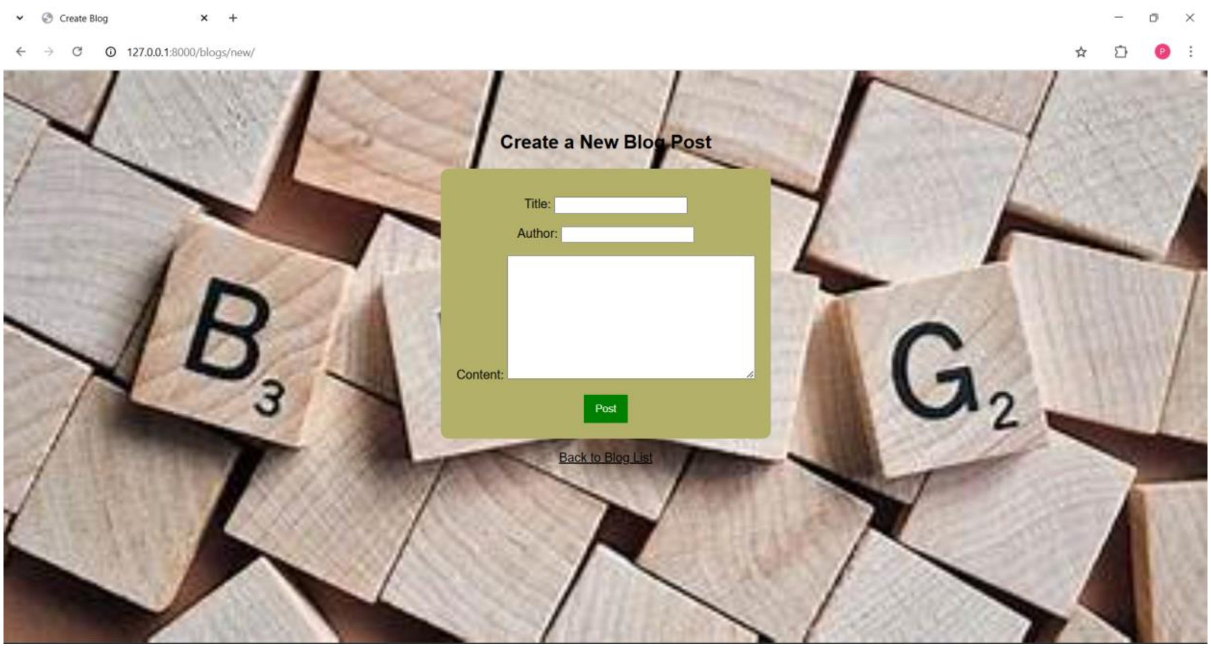
ram

Author: seeth

Published Date: Feb. 23, 2025, 9:20 a.m.

yret56

Create New Blog



Create a New Blog Post

Title:

Author:

Content: