Plog app.

V.01

—

Dragons

G3_Team.

Let’s take a tour...

Outline...

I. Introduction

- Overview of the Blog Application
- Purpose and target audience

II. Installation and Setup

- Prerequisites
- Installation instructions

III. Database Schema

- Models (BlogPost and Comment)
- CRUD operations for models
- Database migrations

IV. Authentication and Authorization

- Authentication levels (Admin, Writer, Viewer)
- Permissions for each level

V. Frontend Development

- Bootstrap integration
- Styling the application

VI. Views and Templates

- Explanation of views (create, publish, view, comment)
- Template structure

VII. Development and Testing

- Developing views and functionality
- Unit testing and functional testing

VIII. Admin Interface

- Managing user groups
- Content management using the admin interface

IX. Bug Fixes and Best Practices

- Documenting and addressing bugs
- Documentation best practices

X. Sprint Review Meetings

- Purpose and conduct of review meetings
- Feedback and decisions.

1. Introduction

Project Overview...

The Simple Blog Application is a web-based platform designed to allow users to create, publish, view, and comment on blog posts. This documentation provides comprehensive guidance on how to set up, use, and maintain the application. It also covers the implementation of authentication levels (viewer, writer, and admin).

Purpose...

"The primary purpose of this documentation is to guide students and learners through the understanding, creation, and utilization of the Simple Blog Application. It serves as a structured learning resource, assisting students in comprehending the software's key concepts, features, and implementation. This document is designed to facilitate a step-by-step journey, ensuring clarity and accessibility in every aspect of the project. It aligns with educational goals by encouraging hands-on learning, promoting good documentation practices, and fostering collaboration among learners. Through straightforward language and illustrative visuals."

Scope...

1. **Models and CRUD Operations:**
    a. An introduction to the two core models, BlogPost, and Comment.
    b. Step-by-step instructions for implementing Create, Read, Update, and Delete (CRUD) operations for both models, making it suitable for students

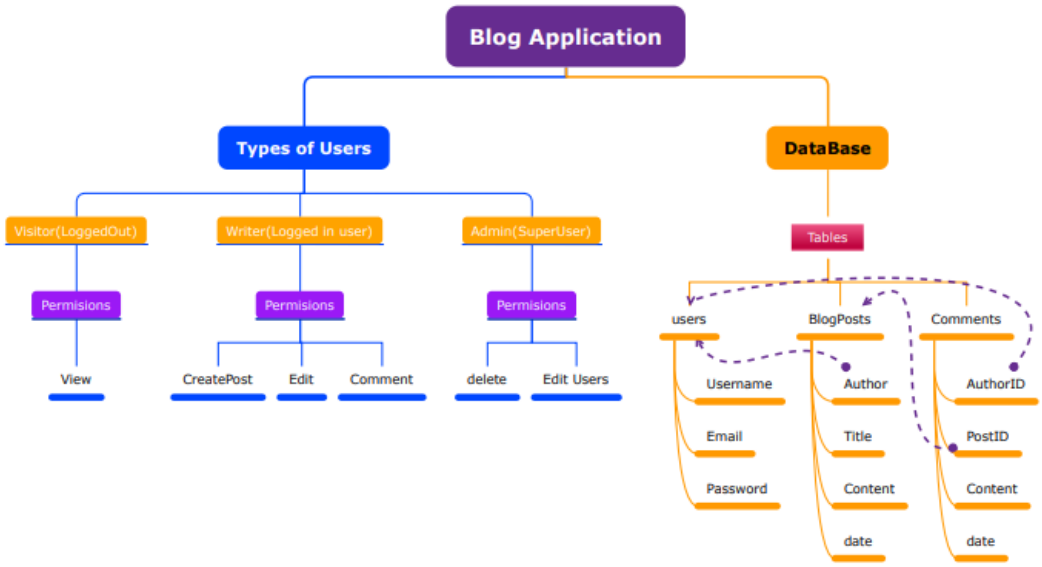
2. **Authentication and Authorization:**
    a. Explanation of the authentication system, including three authorization levels (Admin, Writer, Viewer), designed to teach students about access control.
    b. Permissions management, including:
        i. Viewer Level:
            1. Reading blog posts.
            2. Reading comments.
        ii. Writer Level:
            1. Creating, reading, updating, and deleting their own posts.
            2. Creating and reading comments on any post.
        iii. Admin Level:
            1. Accessing the Admin Panel.
            2. Viewing, modifying, adding blog posts or comments.
            3. Accessing user profiles and adding new Writers or Admins.

3. **Frontend Development with Bootstrap:**
4. **Development and Testing:**
5. **Project Review Meeting:**

This scope is designed to provide students with a hands-on educational journey, allowing them to learn essential concepts of web development, authentication, authorization, and frontend design while building a functional Simple Blog Application.

Getting Started...

First, determine the basic info about the model and users in this mind map...



Installation and Setup

Prerequisites

Before you begin, ensure you have the following installed:

- Python
- Django
- Virtual Environment

Installation

To set up the project, follow these steps:

Clone the project from the repository.

Create a virtual environment.

Install project dependencies.

Configure database settings.

Apply database migrations.

Project Structure

The project follows this directory structure:

```
blog_project/
├── blog_app/
│   ├── migrations/
│   ├── templates/
│   ├── views.py
│   ├── models.py
│   └── ...
├── blog_project/
│   ├── settings.py
│   ├── urls.py
│   └── ...
└── ...
```

Database Schema

Models

BlogPost Model

title: CharField - Title of the blog post.

content: TextField - Content of the blog post.

pub_date: DateTimeField - Publication date.

author: ForeignKey to User model - Author of the post.

Comment Model

text: TextField - Text of the comment.

post: ForeignKey to BlogPost model - The post to which the comment is associated.

author: ForeignKey to User model - Author of the comment.

Views and Templates

Views

Create View

The create view allows users to create new blog posts.

- URL: /create/
- Authentication: Writer or Admin

```python
from django.shortcuts import render,
redirect
from django.contrib.auth.forms import
UserCreationForm
from django.contrib.auth.models import
Group

# Create your views here.


def signUp(request):
    if request.method == 'POST':
        form =
UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            # Save the user account
            # Add the user to the
Writer group
            adminGroup =
Group.objects.get(name='Writer')  # Get
the Writer group from the database
            user.groups.add(adminGroup)
 # Add the user to the Writer group

            return redirect('login')
    else:
        form = UserCreationForm()

    return render(request,
'BlogApplication/Signup.html')
```

Publish View

The publish view allows authors to publish their draft posts.

- URL: /publish/<post_id>/
- Authentication: Writer or Admin

View Post View

The view post view displays an individual blog post and its comments.

- URL: /post/<post_id>/
- Authentication: Viewer, Writer, or Admin

Comment View

The comment view allows users to add comments to a blog post.

- URL: /comment/<post_id>/
- Authentication: Viewer, Writer, or Admin

Templates

- blog_post_list.html: Template for displaying a list of blog posts.
- blog_post_detail.html: Template for displaying an individual blog post and its comments.

Here is a the HTML for the home page.

```
{% extends "BlogApplication/Base/Base.html" %} {% block container %}
<div class="container my-3">
  <h2 class="text-center">Post List</h2>

  <div class="container">
    <div class="d-grid gap-2 w-75 flex-container justify-content-end mb-3">
      {% if request.user.is_authenticated %}
      <a href="{% url 'addPost' %}" class="btn btn-success mx-1">
        Add Post
      </a>
      {% endif %}
    </div>
    {% for post in posts %}
  <div class="container w-50 border rounded p-3 mb-3">
    <div class="row">
      <div class="col-md-12">
        <div class="flex-container d-flex mb-3">
          <h3 class="text-left d-inline-flex w-50">{{ post.title }}</h3>
          <div class="d-grid gap-2 w-50 d-flex justify-content-end">
            {% if request.user.is_authenticated and post.author == request.user %}
              <div>
                <a href="{% url 'edit' post.pk %}" class="btn btn-primary mx-1">
                  Edit
                </a>
                <a href="{% url 'delete' post.pk %}" class="btn btn-danger mx-1" onclick="return
confirm('Are you sure you want to delete this post?');">
                  Delete
                </a>
              </div>
            {% endif %}
          </div>
        </div>
        <p>
          {{ post.content|truncatewords:20 }}
          <a href="{% url 'details' post.pk %}" class="text-primary d-inline-block">
            Read More
          </a>
        </p>
        <p class="d-block text-muted">{{ post.pupDate }}</p>
      </div>
    </div>
  </div>
{% endfor %}

    {% comment %} {% for post in posts %}
    <div class="container w-50 border rounded p-3 mb-3">
      <!-- ^^^ Add border and padding classes here -->
      <div class="row">
        <div class="col-md-12">
          <div class="flex-container d-flex mb-3">
            <h3 class="text-left d-inline-flex w-50">{{ post.title }}</h3>
            <div class="d-grid gap-2 w-50 d-flex justify-content-end">
              {% if request.user.is_authenticated %}
              <div>
                <a href="{% url 'edit' post.pk %}" class="btn btn-primary mx-1">
                  Edit
                </a>
                <a
                  href="{% url 'delete' post.pk %}"
                  class="btn btn-danger mx-1"
                >
                  Delete
                </a>
              </div>
              {% endif %}
            </div>
          </div>
          <p>
            {{ post.content|truncatewords:20 }}
            <a href="{% url 'details' post.pk %}" class="text-primary">
              Read More
            </a>

          </p>
          <p>{{ post.pupDate }}</p>
        </div>
      </div>
    </div>
    {% endfor %} {% endcomment %}
  </div>
</div>
{% endblock container %}
```

Authentication and Authorization

User Groups.

Viewer:

Permissions: Read-only access to blog posts and comments.

Writer:

Permissions: Create, edit, and publish blog posts. Comment on blog posts.

Admin:

Permissions: Full control over blog posts, comments, and user management.

User Profiles

The user profile model includes a field for the user's auth group

(viewer, writer, or admin).

Access Control

Authorization checks are implemented in views using decorators and middleware.

Development and Testing

Developing Views

- Views are developed following Django's class-based views pattern.
- Authentication and authorization checks are integrated into views using built-in decorators.

Unit Testing

- Unit tests are written using Django's testing framework (TestCase class).
- Example unit test cases for views, models, and forms.

Functional Testing

- Functional tests ensure the application works as expected.
- Test cases for user authentication, authorization, and sorting of blog posts.

Admin Interface

Managing Groups and Users

- Create and manage user groups (viewer, writer, admin).
- Assign groups to user accounts.

Managing Blog Content

- Use the Django admin interface to manage blog posts and comments.
- Create, edit, and delete blog posts and comments.
- Assign authors and publication dates.

Bug Fixes

All details in GitHub repo...

Please check it...

https://github.com/DragonsEG/Blog_App_G3_Team

Documentation Best Practices

Effective documentation is crucial for understanding and maintaining the Simple Blog Application. Here are some best practices to ensure that the documentation is clear, useful, and well-structured:

Use Clear and Concise Language:

Write in a straightforward manner, avoiding unnecessary technical jargon.

Include Visuals:

Enhance the documentation with visuals such as screenshots and diagrams.

Edit:

Carefully review the documentation to eliminate errors, typographical mistakes, and inconsistencies. Ensure that the content flows logically and is easy to follow.

Review Meeting

Sprint review meetings play a vital role in assessing the project's progress and ensuring alignment with goals.

Feedback and Decisions

The review meeting also serves as an opportunity to collect feedback From team members. It is a collaborative space where discussions may lead to decisions such as adjusting project priorities.

Conclusion

In conclusion, this documentation has provided a comprehensive overview of the Simple Blog Application, covering various aspects of its development and usage. Key points covered include:

- Our start and setup instructions.
- Database schema and CRUD operations.
- Authentication and authorization levels.
- Frontend development using Bootstrap.
- Views, templates, and testing procedures.
- Admin interface usage.
- Bug fixes and documentation best practices.
- The purpose and content of review meetings.

Appendix

The appendix section includes additional resources and links to further enhance understanding and support the use of the Simple Blog Application.

Links:

Auth

https://youtube.com/playlist?list=PLLxk3TkuAYnryu1lEcFaBr358IynT7l7H&si=tpVHHFx9FNeE3zaY

https://youtu.be/yqWaa5yegmI?si=FXhXtOqgwaNrTpLe

https://www.youtube.com/watch?v=5ZoGy46kHMw&ab_channel=ProgrammingSecrets-Tips%26Tricks

Django unit testing

https://www.youtube.com/playlist?list=PLbpAWbHbi5rMF2j5n6imm0enrSD9eQUaM

Agile

https://youtube.com/playlist?list=PLcdCk5ljWQ-p3t9E_vPcGzZMGLoAeRUO5&si=Ro2rZm5NHOS-L35G

Permissions

https://youtu.be/eBsc65jTKvw?si=vwA83C7ZpgM-Tue1

Customize the Django admin (Watch till the end of the playlist)

https://www.youtube.com/watch?v=3P78wNcEvII&list=PLknwEmKsW8OtK_n48UOuYGxJPbSFrICxm&index=35&ab_channel=AbdelrahmanGamal