NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Datbase Management System

Mini Project on

"Blog Application System"

Submitted to

Ms Vaishali Bangera
Assistant Professor Gd-1
Department of CSE
NMAM Institute Of Technology Nitte

Submitted by:
Prathiksha Kini 4NM20CS139
Palguni Samaga 4NM20CS127

Table of Contents

Chapter-1 Introduction

Overview:

The purpose of the blog application system is to facilitate a technological communication platform. The application system enables users to create, update, and delete their blog posts, create and manage profiles. The blog application enables users to channelize their thoughts or experience with the rest of the world.

This project report will introduce how to build a blog application system using the Django framework. Django is an open-source web application framework written in python. This blog application system built using Django has four major components with different functionalities that will be introduced later. We will introduce various features of the Django framework and SQLite3 RDBMS in the later section. In the end, snapshots are attached to demonstrate UI.

• The project aims to create a desirable blog application for users with a suitable UX design and proper backend management.
• The project uses the Django web framework (python framework) to implement the blog application.
• The project uses SQLite3 software library for a relational database management system in the backend to store, retrieve, and perform necessary backend operations.
• HTML, Bootstrap CSS, and Django Template Language implement the front-end to customize the user interface.

Features and Significance:

As mentioned above, this application system has four vital components/features: User registration/authentication, User blog posts, add category, and profile modification.

• **User registration/Authentication:** Any blog application will include this primary feature to register users in their application. To access all the other features, the user must register into the application. We collect users' fundamental data such as email, username, first name, last name, password, and store it in the database. Moreover the users can change their password for security purposes.

• **Profile modification:** This is an extended feature of user registration. Here, users can create and modify their profiles. Users can change their profile picture, email address, and their usernames. The altered data reflects in the database system and the front-end of the application.

• **User blog posts:** Once the user has registered and set up a profile, they can post blogs and modify them accordingly. Users can comment on the blog posts as well as view the profile of the author.

• **Add category:** The users can create a category of their own and assign their blog to a particular category. The blogs belonging to a particular category can be filtered out.

Developing a complex website like the blog application must incorporate security, scalability, and simple architecture for development and database management. Django framework with SQLite3 RDBMS is significant to develop such a complex application.

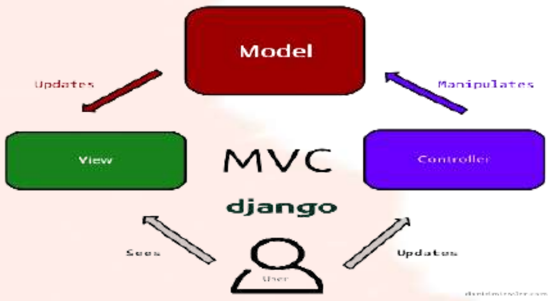
Django Framework:

Django is an open-source web application framework written in python. It is a high-level framework that encourages rapid development and clean, pragmatic design. Django consists of three major parts: Model, View, and Template.

Model: Model is a single, definitive data source that contains the essential field and behavior of the data. Python classes implement models. Usually, one model is one table in the database. Each python object in the model represents a field of a table in the database. Django provides a set of automatically generated database application programming interfaces (APIs) for the convenience of users.

View: A view is a short form of the view file. It is a file containing a Python function that takes web requests and returns web responses. A response can be HTML content or XML documents or a “404 error” and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function with a particular URL, we need to use a structure called URLconf that maps URLs to view functions.

Template: Django’s template is a simple text file that can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase. Django templates allow the developers to implement the front-end logic of the application.



Model View Controller(MVC) in Django

Template Inheritance:

The most powerful – and thus the most complex – part of Django's template engine is template inheritance. Template inheritance allows you to build a base "skeleton" template that contains all the common elements of your site and defines blocks that child templates can override.

- Create a base.html template that holds the main look-and-feel of your site.
- Create individual templates for each type of page, such as a news article or blog entry. These templates extend the appropriate section template.
- If you use {% extends %} in a template, it must be the first template tag in that template. Template inheritance won't work, otherwise.

```
{% block content %}
        ...
 {% endblock %}
```

- To display static images use the tag

```
{% load static %}
```

Chapter-2 Hardware and Software Requirements

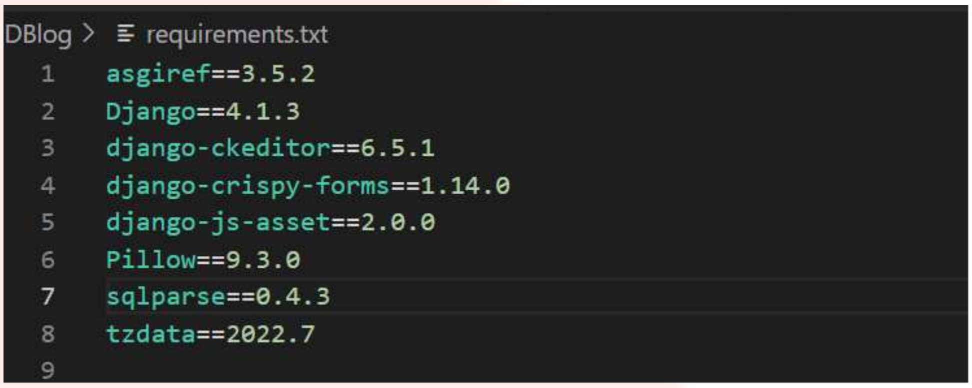
Hardware Requirements:

Django framework doesn't have many system requirements. Since it's primarily written in python, the system requirements must satisfy to install Python language. Similarly, SQLite3 can also run on any modern system. Hardware requirements for this project are primarily concerned with the system hardware requirements (PC). We recommend Intel Core i3 processor or above for the development phase with a minimum 4GB PC Ram and HDD or SSD hard disk. Server end hardware requirements aren't significant for this project because we can host this project on the cloud with various services.

Software Requirements:

This blog application uses python language for development with SQLite database management system. We have used the Django framework version 4.1.3 and hence we have to use python version 3.6 or above and SQLite version 3.8 or above. The Integrated Development Environment used for this project is Visual Studio Code.

requirements.txt


DBlog > requirements.txt

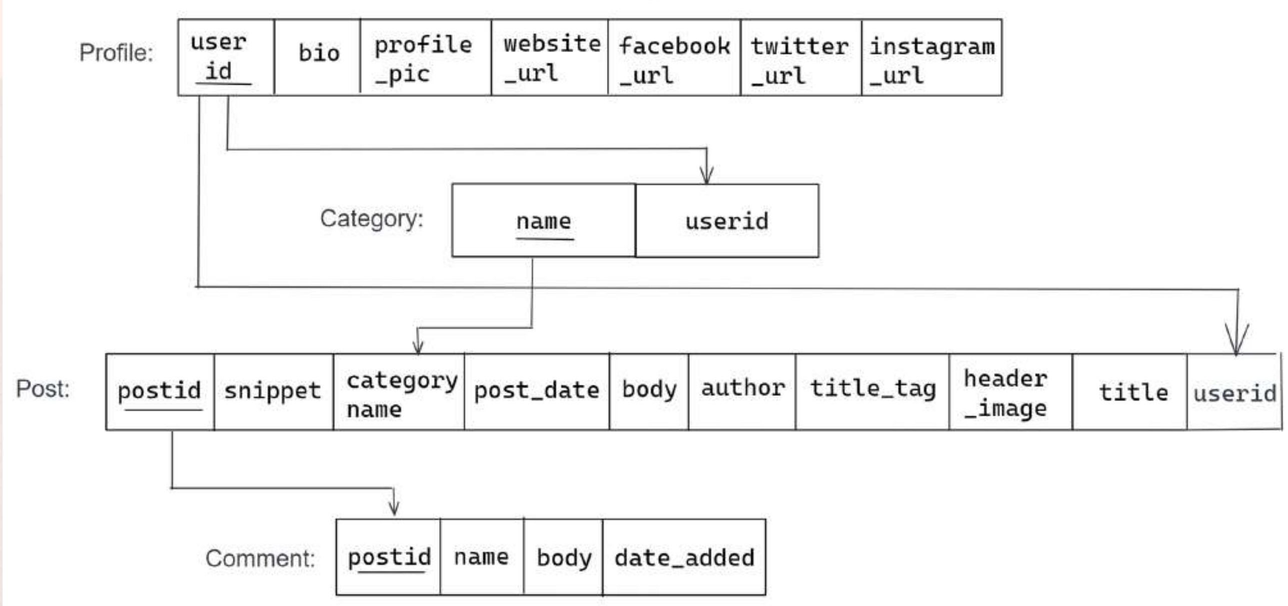
```
asgiref==3.5.2
Django==4.1.3
django-ckeditor==6.5.1
django-crispy-forms==1.14.0
django-js-asset==2.0.0
Pillow==9.3.0
sqlparse==0.4.3
tzdata==2022.7
```

These are the minimum requirements for the project. Server end hardware and software requirements are not significant for this project. If the blog application attracts massive data, the server end hardware requirements may vary accordingly and expand as big as an individual data center with multiple servers.

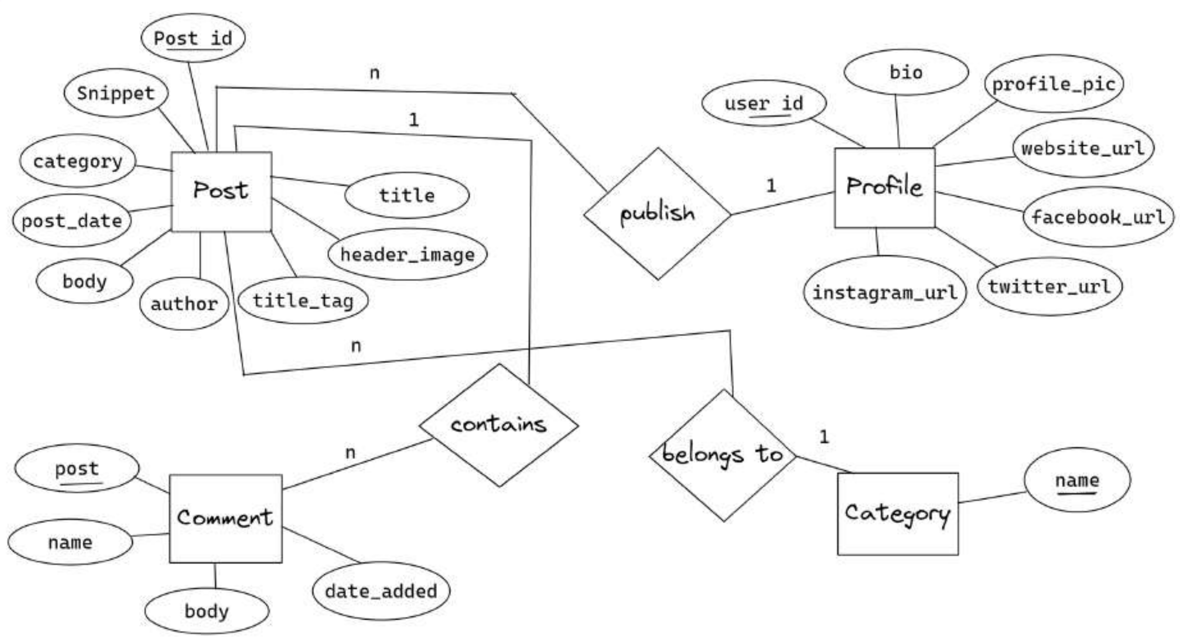
- Make sure you have Python installed in your system. If not you can quickly install Python from https://www.python.org/downloads/
- Once Python is installed in your system set up the environment variables
- Clone the project from https://github.com/pkini2002/DBlog-Application
- Follow the steps as mentioned in README.md for installing the dependencies and for running the project in your local machine!
- Ensure proper internet connection and run the project in any browser.

Chapter-3 Design and Implementation

Schema diagram



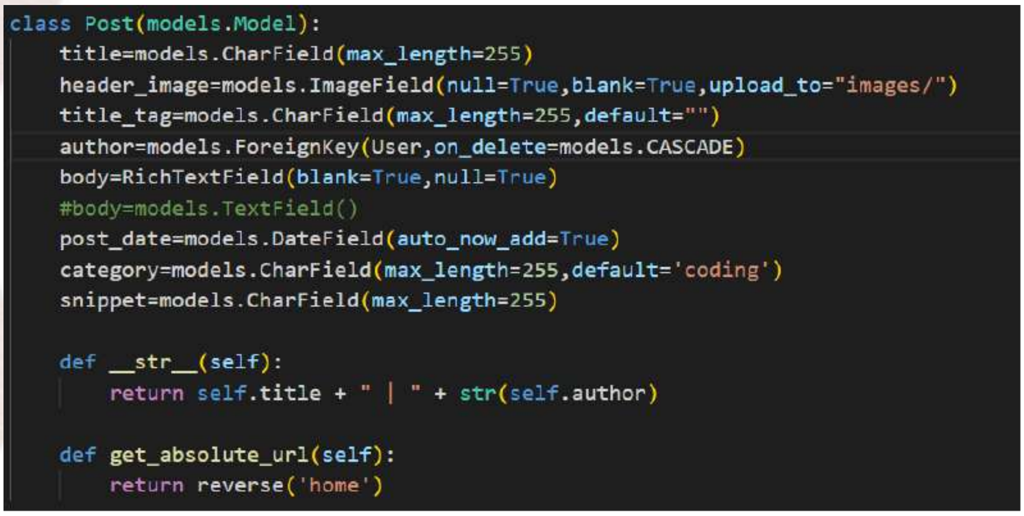
ER Diagram



Backend Development with Django and Sqlite:

This section covers the backend development and overall logic used to develop this project.

Creating tables: To get started, we need to create a backend of the system that is the database. All the tables in the database for this blog application system include information about users, blog posts, categories, and so forth. These tables are created initially when the blog application is deployed. Most information will be inserted or updated in the database dynamically (For example, registering a user). Every time we want to post a blog, we will insert a tuple into the blog_post table to make the database consistent with the real world. To have a blog_post table in the database, we first need to choose a database. Django supports almost all popular databases such as MySQL, sqlite3, and oracle database. The one we used for this blog application system is sqlite3. We only need to write one sentence to set up the database: **DATABASE_ENGINE = 'sqlite3'** Next, we create the blog_post table in the database. Django uses a class called model to represent the database table schema. To create a table, we just need to write a new class derived from model class. Here is an example:



```
class Post(models.Model):
    title=models.CharField(max_length=255)
    header_image=models.ImageField(null=True,blank=True,upload_to="images/")
    title_tag=models.CharField(max_length=255,default="")
    author=models.ForeignKey(User,on_delete=models.CASCADE)
    body=RichTextField(blank=True,null=True)
    #body=models.TextField()
    post_date=models.DateField(auto_now_add=True)
    category=models.CharField(max_length=255,default='coding')
    snippet=models.CharField(max_length=255)

    def __str__(self):
        return self.title + " | " + str(self.author)

    def get_absolute_url(self):
        return reverse('home')
```

Python query sets and SQL:

This section covers most of the Django query sets and raw SQL that have been used in the blog application. Query sets are the list of objects that have been created using the models. We can perform operations such as add, delete, retrieval, and many more. It's implemented in python and can interact with database tables.

• **Filter method:** filter() method is used when we need to perform queries with certain conditions. We have used the filter method to filter the blogs based on category.

• **All method:** The simplest way to retrieve an object (tuple) from the database table is to call all() method when using query sets.

Generic Views: Writing web applications can be monotonous, because we repeat certain patterns again and again. Django tries to take away some of that monotony at the model and template layers, but web developers also experience this boredom at the view level.

Views used:

- **ListView** : List View refers to a view (logic) to display multiple instances of a table in the database.

- **DetailView:** Detail View refers to a view (logic) to display one instances of a table in the database.

- **CreateView** : Create View refers to a view (logic) to create an instance of a table in the database.

- **PasswordChangeView:** Django provides authentication and authorization. For Changing Password you need to get authenticated first. In your urls.py, we need to import PasswordChangeView from Django Auth. By default, Django PasswordChangeView will render template registration/change_password.

- **UpdateView:** Update View refers to a view (logic) to update a particular instance of a table from the database with some extra details.

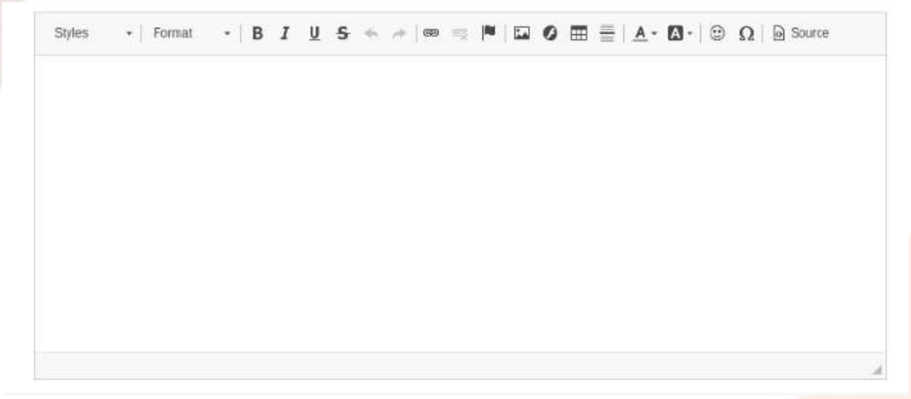
- **DeleteView:** Delete View refers to a view (logic) to delete a particular instance of a table from the database.

RichText Editor: RichTextField is generally used for storing paragraphs that can store any type of data. Rich text is the text that is formatted with common formatting options, such as bold, italics, images, URLs that are unavailable with plain text.

Syntax: field_name=RichTextField()

Now install django-ckeditor package by entering the following command in your terminal or command prompt.

pip install django-ckeditor



User Interface and View Functions:

So far, we have our backend database and the frontend web page user interface. What we need now is the logic in between to deal with the user requests and maintain the database. Django view component provides a set of application programming interfaces to fulfill our need and help us implement the logic. The Django view file is where we write our function or define classes to achieve the above two goals. First, it is used to pass parameters to the template and call the right template for the user. Every time we input a URL in the address bar or click a hyperlink in the system, Django will call the right view function based on that URL. Then the function will return a template as well as the corresponding parameters. Thus, we can see the actual web page displaying the information we need. Second, if we submit something such as blog post, the function will have an http request as its input parameter. Based on that parameter the database is updated or the user is provided the required information. Although we have created the blog post table and various other tables, we can update them using the Django model component and SQL queries, we should not allow the user to manipulate the database directly. Otherwise, it would be a disaster for both the users and the technical maintenance team. Instead, we create a user interface to let users interact with the data indirectly. Django provides the template component to create the user interface for users. A template is just a simple html file with some Django syntax mixed in. Every template corresponds to a web page which the users will use to interact with the system. A template can also have styling references such as CSS files and JavaScript files. In this project, we have used Bootstrap CSS styling to make the User Interface Development simpler.

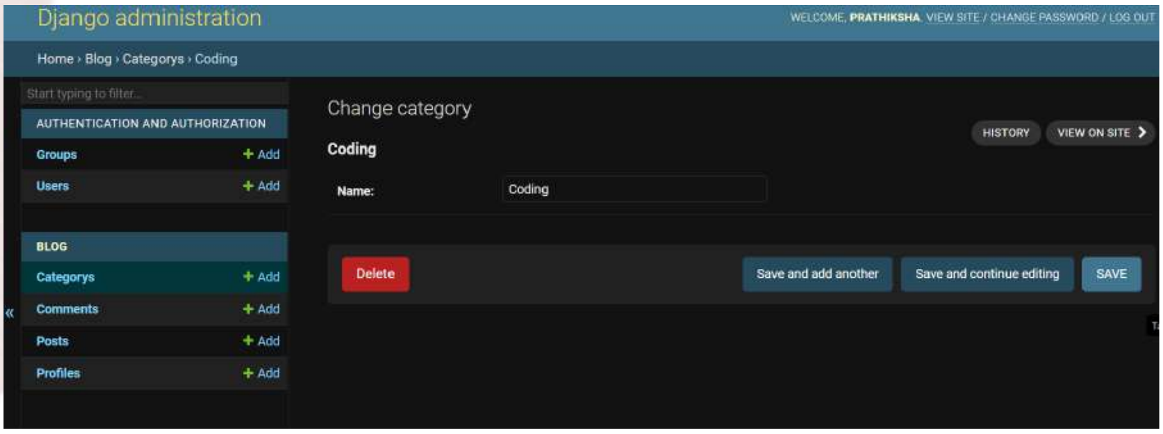
Chapter-4 Results and Conclusion

Observations and Learning:

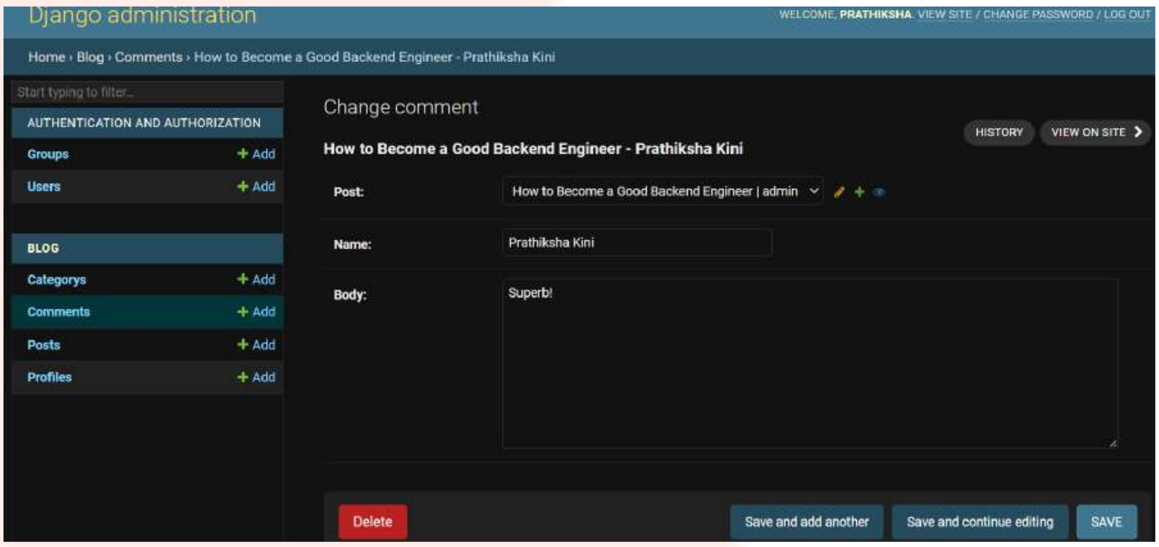
- Working on the backend can be time-consuming and yet can be simplified when working with the Django framework.
- SQLite3 provides a robust and flexible Database Management System that is very suited for the scalability of products.
- Normalization of Database is important to keep the database clean and organized.
- Working with Django and SQLite3 in the backend is faster and, the query results load quickly.

Snapshots of the Project:

Django admin:


Django administration
WELCOME, PRATHIKSHA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Blog › Categorys › Coding
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
BLOG
Categorys + Add
Comments + Add
Posts + Add
Profiles + Add
Change category
Coding
HISTORY
VIEW ON SITE
Delete
Save and add another
Save and continue editing
SAVE

Name: Coding


Django administration

WELCOME, PRATHIKSHA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Blog › Comments › How to Become a Good Backend Engineer - Prathiksha Kini

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

BLOG

Categorys + Add

Comments + Add

Posts + Add

Profiles + Add

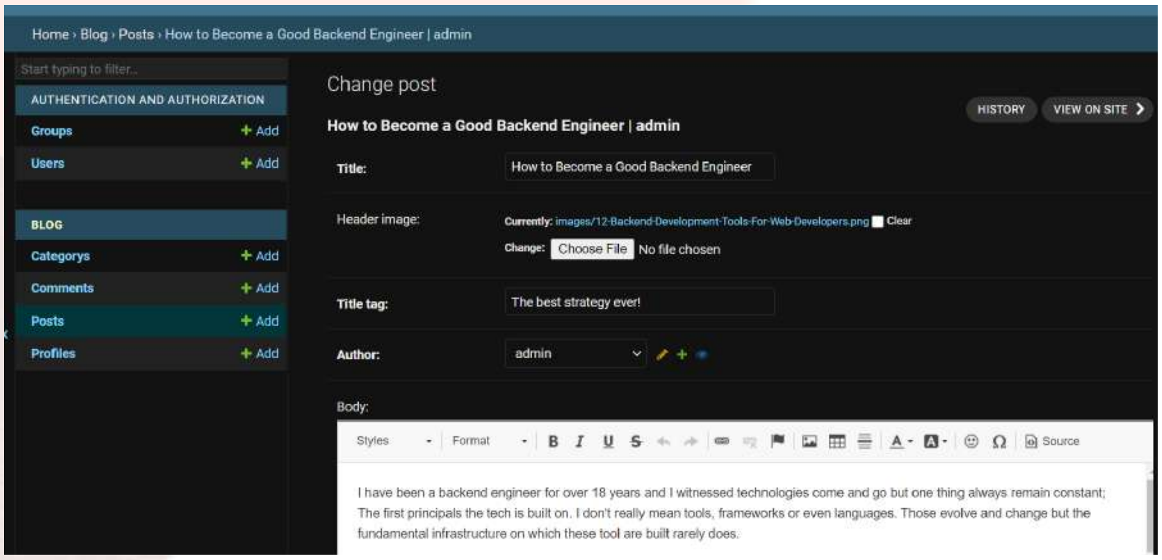
Change comment

HISTORY VIEW ON SITE ›

How to Become a Good Backend Engineer - Prathiksha Kini

Post:	How to Become a Good Backend Engineer | admin
Name:	Prathiksha Kini
Body:	Superb!

Delete Save and add another Save and continue editing SAVE


Home › Blog › Posts › How to Become a Good Backend Engineer | admin

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

BLOG

Categorys + Add

Comments + Add

Posts + Add

Profiles + Add

Change post

HISTORY VIEW ON SITE ›

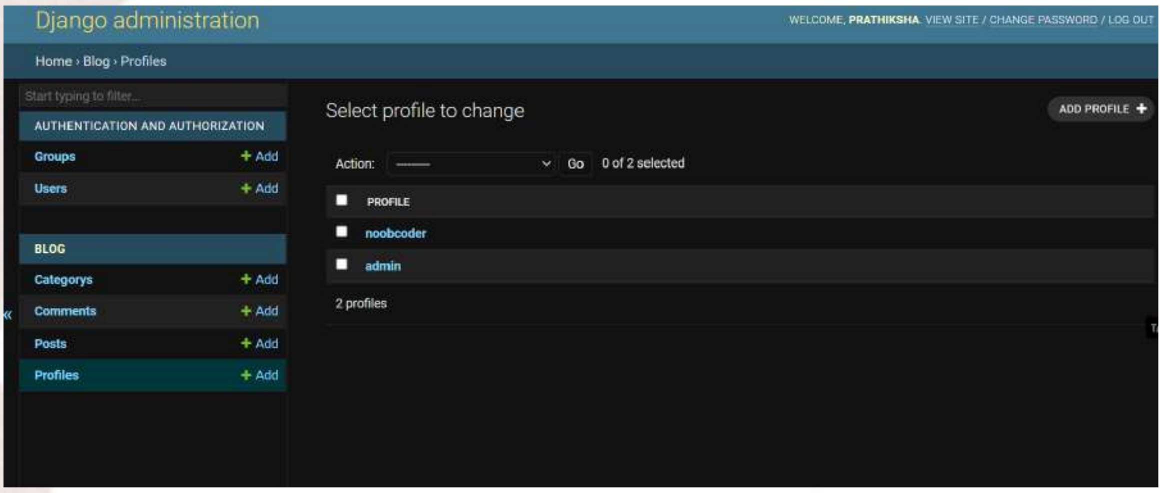
How to Become a Good Backend Engineer | admin

Title:	How to Become a Good Backend Engineer
Header image:	**Currently:** images/12-Backend-Development-Tools-For-Web-Developers.png Clear **Change:** Choose File No file chosen
Title tag:	The best strategy ever!
Author:	admin

Body:

Styles Format B I U S Source

I have been a backend engineer for over 18 years and I witnessed technologies come and go but one thing always remain constant; The first principals the tech is built on. I don't really mean tools, frameworks or even languages. Those evolve and change but the fundamental infrastructure on which these tool are built rarely does.


Django administration

WELCOME, PRATHIKSHA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Blog › Profiles

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

BLOG

Categorys + Add

Comments + Add

Posts + Add

Profiles + Add

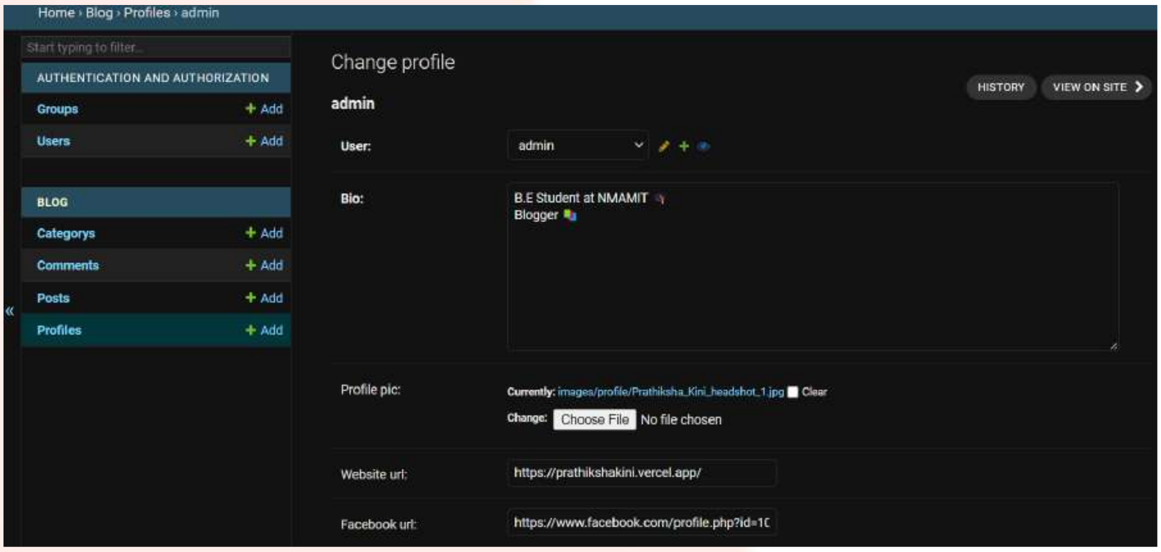
Select profile to change

ADD PROFILE +

Action: --------- Go 0 of 2 selected

PROFILE
noobcoder
admin

2 profiles


Home › Blog › Profiles › admin

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

BLOG
- Categorys + Add
- Comments + Add
- Posts + Add
- Profiles + Add

Change profile

HISTORY | VIEW ON SITE

admin

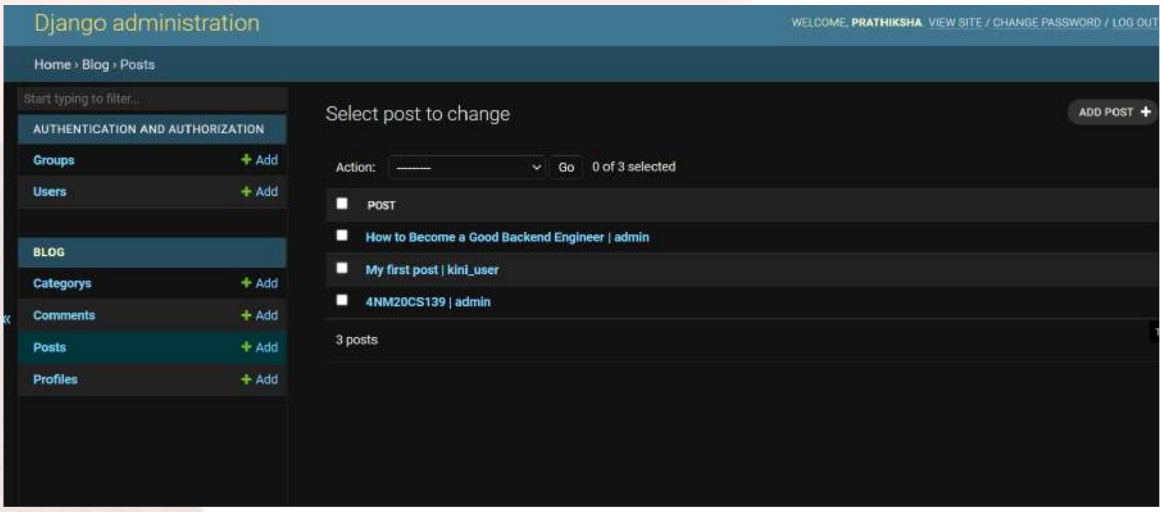
User: admin

Bio:
B.E Student at NMAMIT
Blogger

Profile pic:
Currently: images/profile/Prathiksha_Kini_headshot_1.jpg Clear
Change: Choose File No file chosen

Website url: https://prathikshakini.vercel.app/

Facebook url: https://www.facebook.com/profile.php?id=1C


Django administration

WELCOME, PRATHIKSHA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Blog › Posts

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

BLOG
- Categorys + Add
- Comments + Add
- Posts + Add
- Profiles + Add

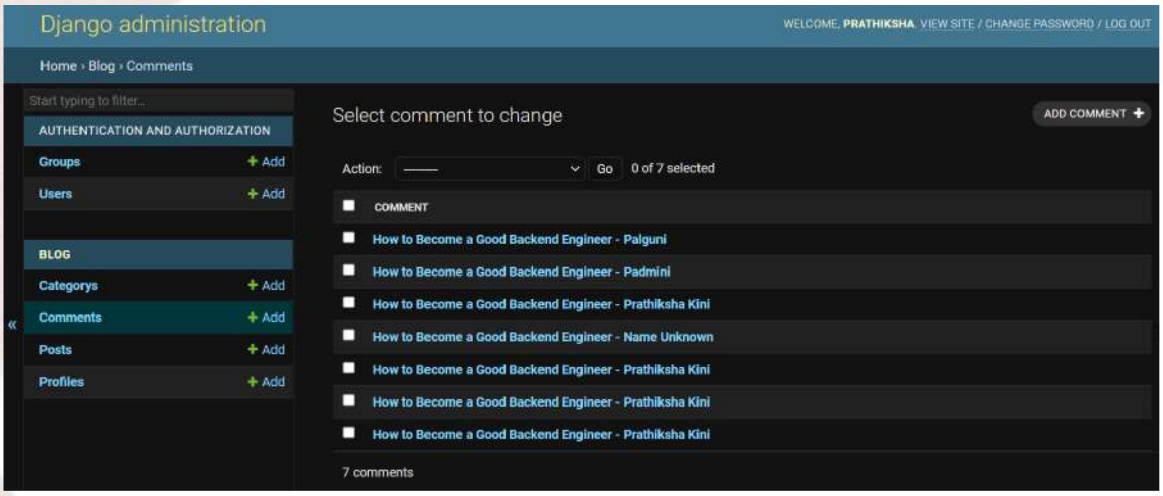
Select post to change

ADD POST +

Action: --------- Go 0 of 3 selected

POST
How to Become a Good Backend Engineer | admin
My first post | kini_user
4NM20CS139 | admin

3 posts


Django administration

WELCOME, PRATHIKSHA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Blog › Comments

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

BLOG
- Categorys + Add
- Comments + Add
- Posts + Add
- Profiles + Add

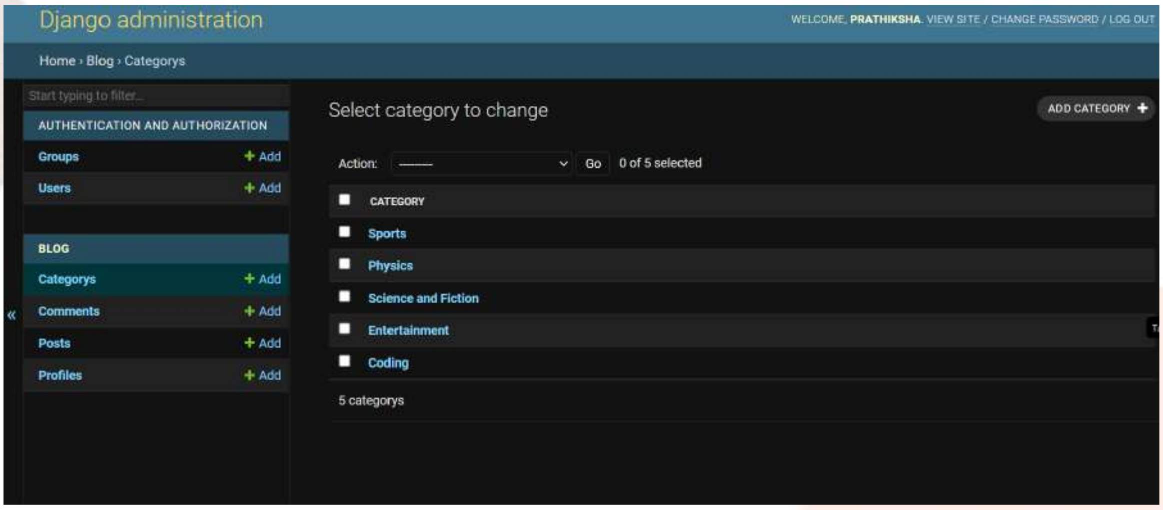
Select comment to change

ADD COMMENT +

Action: --------- Go 0 of 7 selected

COMMENT
How to Become a Good Backend Engineer - Palguni
How to Become a Good Backend Engineer - Padmini
How to Become a Good Backend Engineer - Prathiksha Kini
How to Become a Good Backend Engineer - Name Unknown
How to Become a Good Backend Engineer - Prathiksha Kini
How to Become a Good Backend Engineer - Prathiksha Kini
How to Become a Good Backend Engineer - Prathiksha Kini

7 comments


Django administration

WELCOME, PRATHIKSHA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Blog › Categorys

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

BLOG
- Categorys + Add
- Comments + Add
- Posts + Add
- Profiles + Add

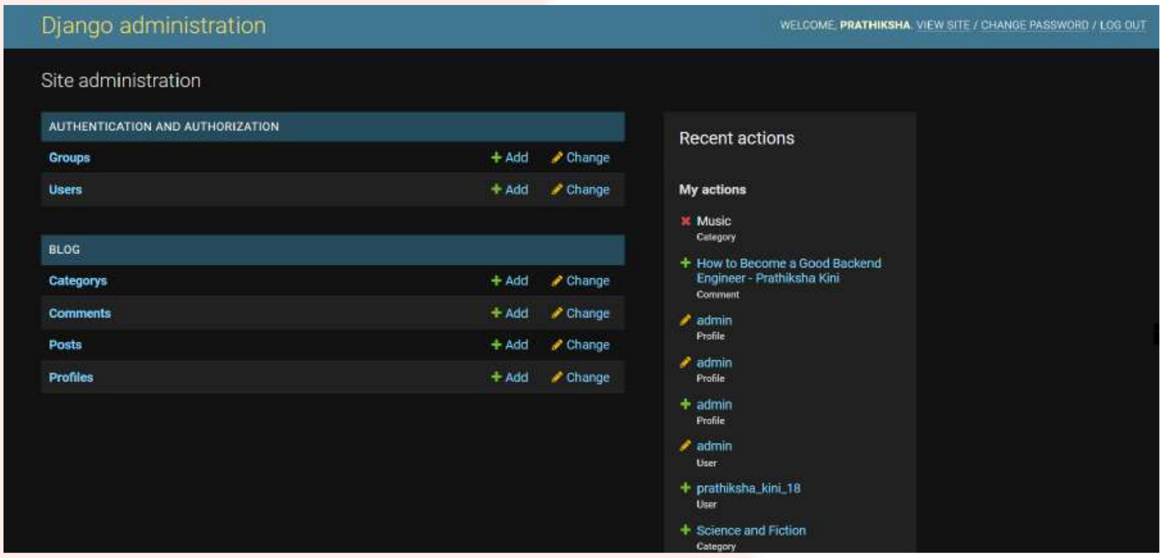
Select category to change

ADD CATEGORY +

Action: --------- Go 0 of 5 selected

CATEGORY
Sports
Physics
Science and Fiction
Entertainment
Coding

5 categorys


Django administration

WELCOME, PRATHIKSHA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change
Users	+ Add	Change

BLOG

Categorys	+ Add	Change
Comments	+ Add	Change
Posts	+ Add	Change
Profiles	+ Add	Change

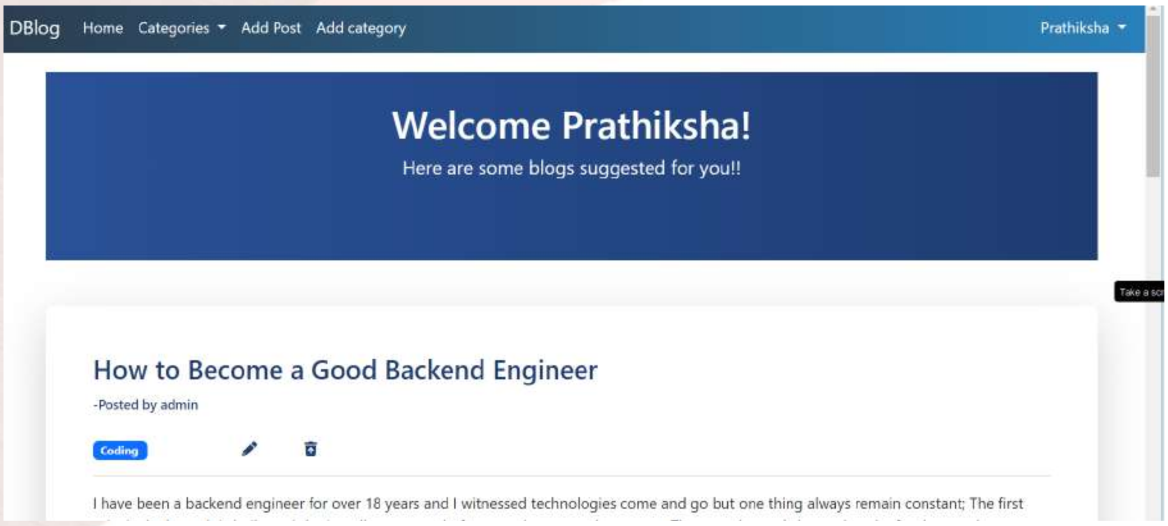
Recent actions

My actions

- Music
  Category
- How to Become a Good Backend Engineer - Prathiksha Kini
  Comment
- admin
  Profile
- admin
  Profile
- admin
  Profile
- admin
  User
- prathiksha_kini_18
  User
- Science and Fiction
  Category

UI Design:

Home Page


DBlog Home Categories Add Post Add category Prathiksha

Welcome Prathiksha!

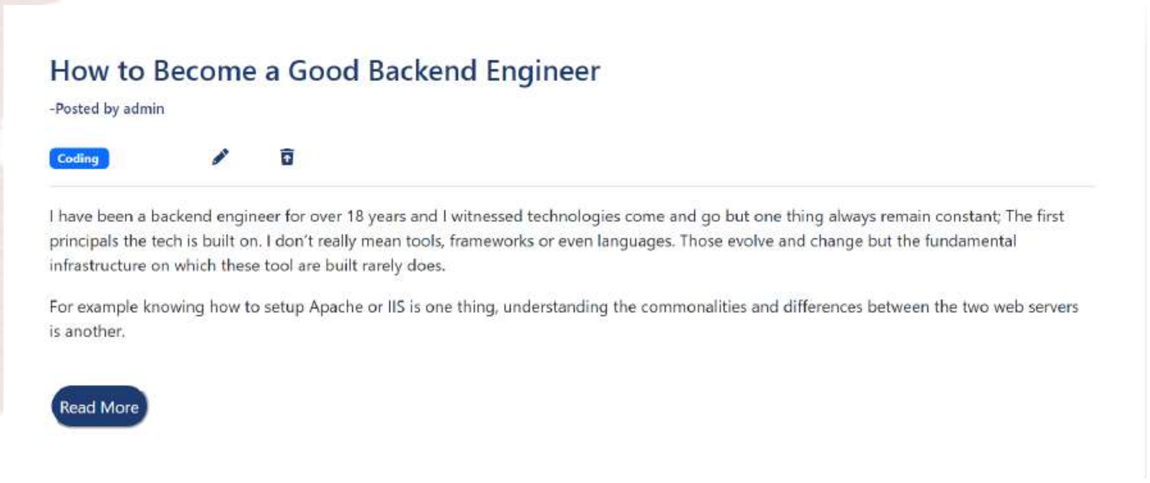
Here are some blogs suggested for you!!

How to Become a Good Backend Engineer

-Posted by admin

Coding

I have been a backend engineer for over 18 years and I witnessed technologies come and go but one thing always remain constant; The first


How to Become a Good Backend Engineer

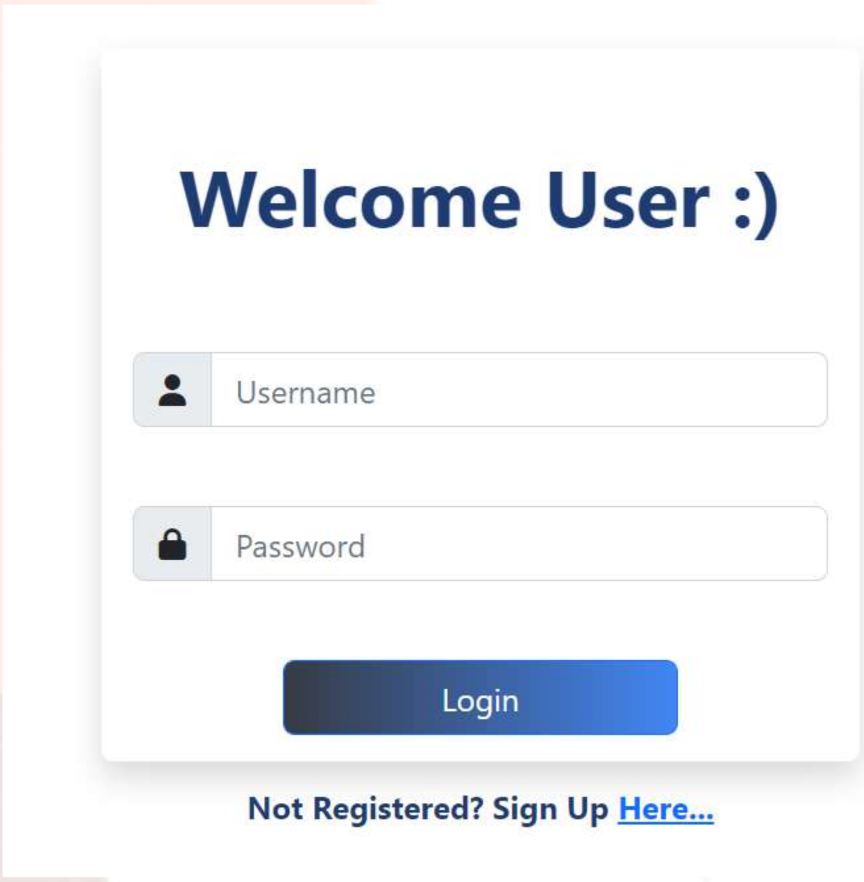
-Posted by admin

Coding

I have been a backend engineer for over 18 years and I witnessed technologies come and go but one thing always remain constant; The first principals the tech is built on. I don't really mean tools, frameworks or even languages. Those evolve and change but the fundamental infrastructure on which these tool are built rarely does.

For example knowing how to setup Apache or IIS is one thing, understanding the commonalities and differences between the two web servers is another.

Read More



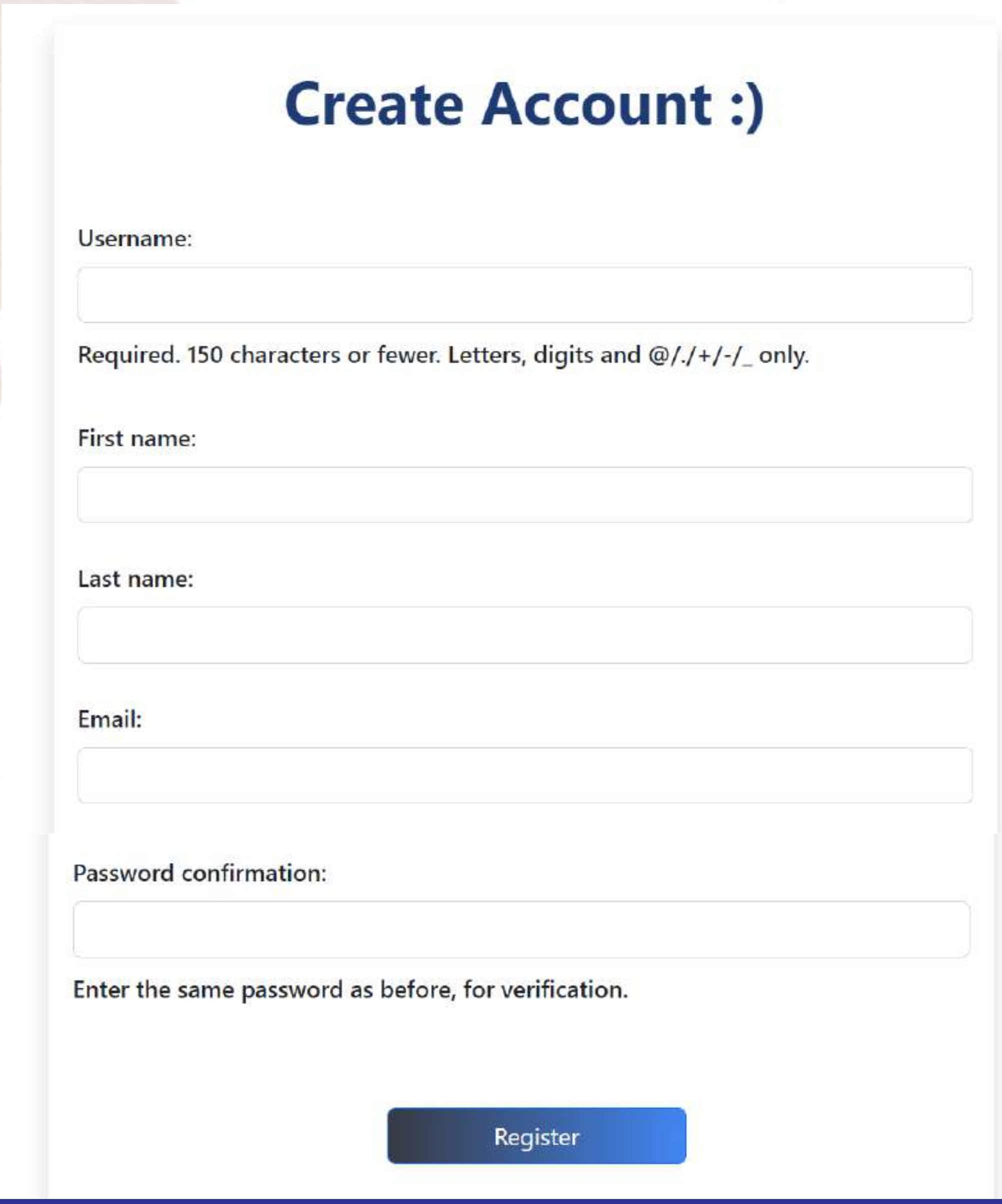
Welcome User :)

Username

Password

Login

Not Registered? Sign Up Here...



Create Account :)

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

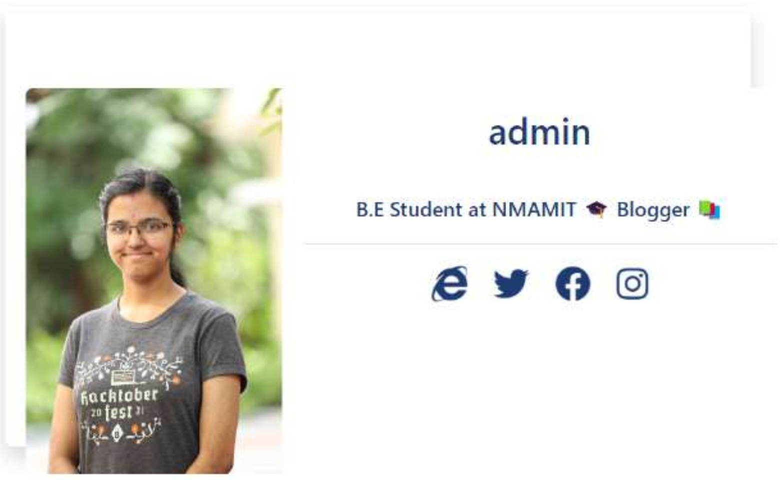
Last name:

Email:

Password confirmation:

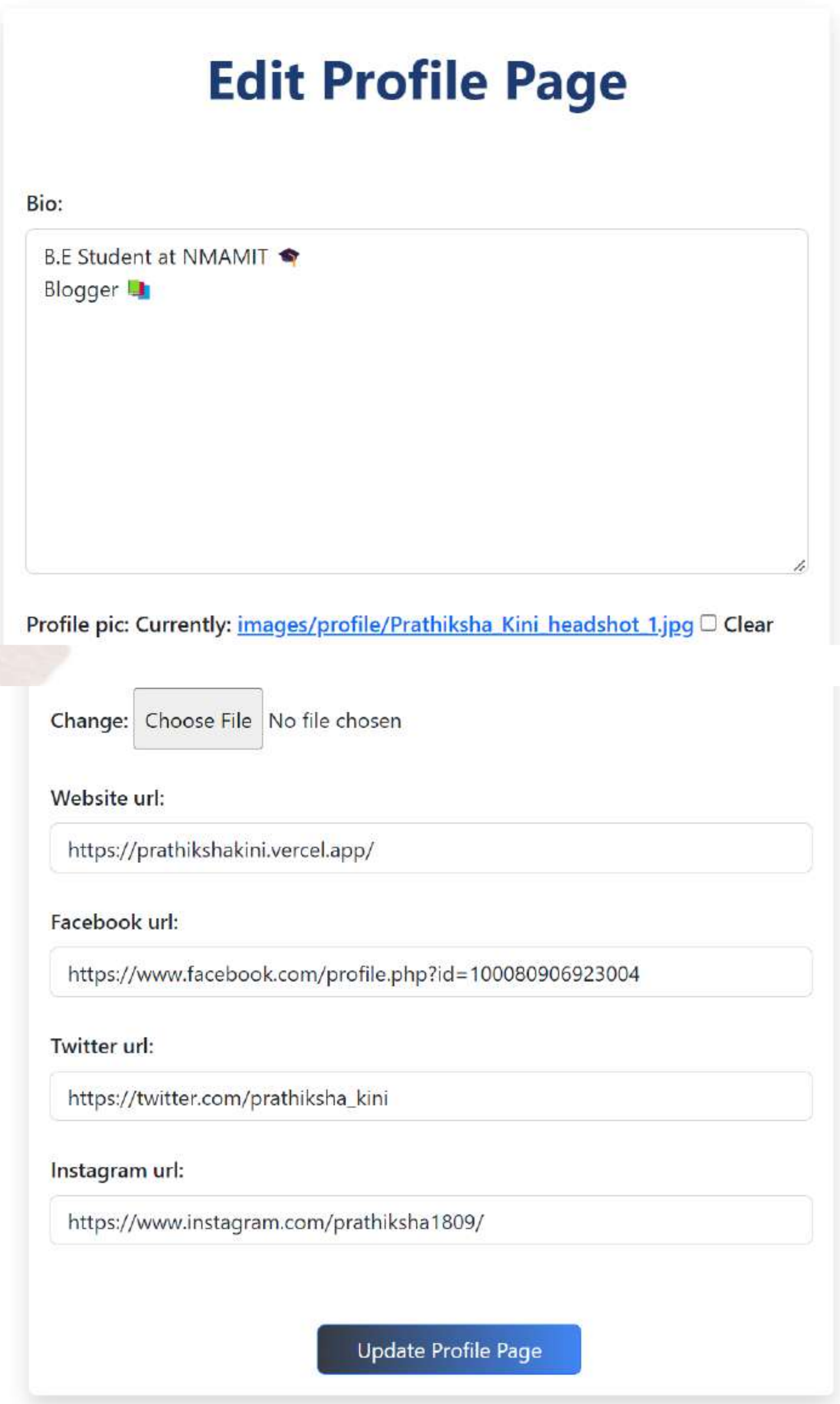
Enter the same password as before, for verification.

Register



admin

B.E Student at NMAMIT 🎓 Blogger 📚



Edit Profile Page

Bio:

B.E Student at NMAMIT 🎓
Blogger 📚

Profile pic: Currently: images/profile/Prathiksha_Kini_headshot_1.jpg ☐ Clear

Change: Choose File No file chosen

Website url:

https://prathikshakini.vercel.app/

Facebook url:

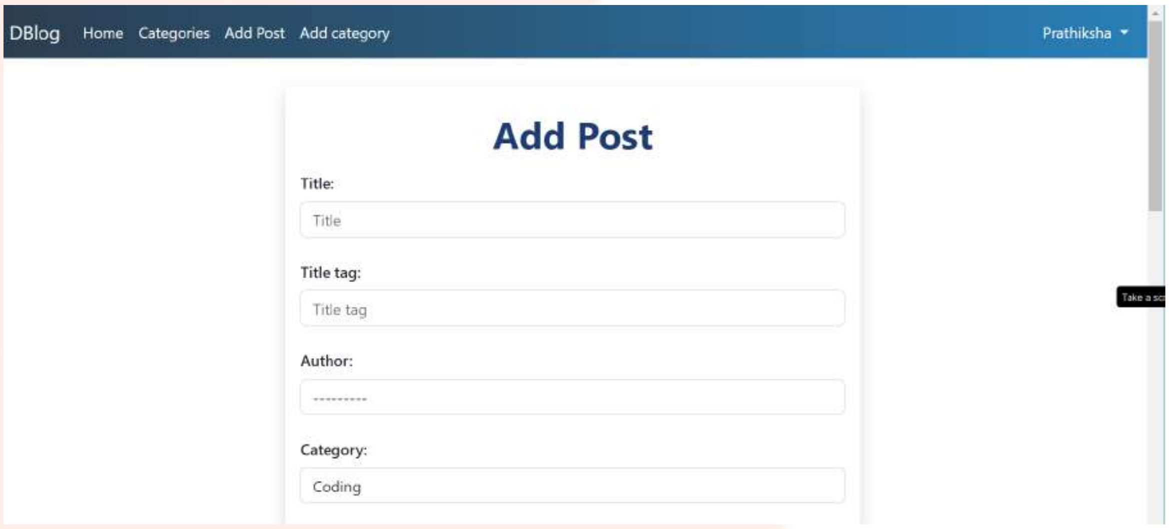
https://www.facebook.com/profile.php?id=100080906923004

Twitter url:

https://twitter.com/prathiksha_kini

Instagram url:

https://www.instagram.com/prathiksha1809/


DBlog Home Categories Add Post Add category Prathiksha

Add Post

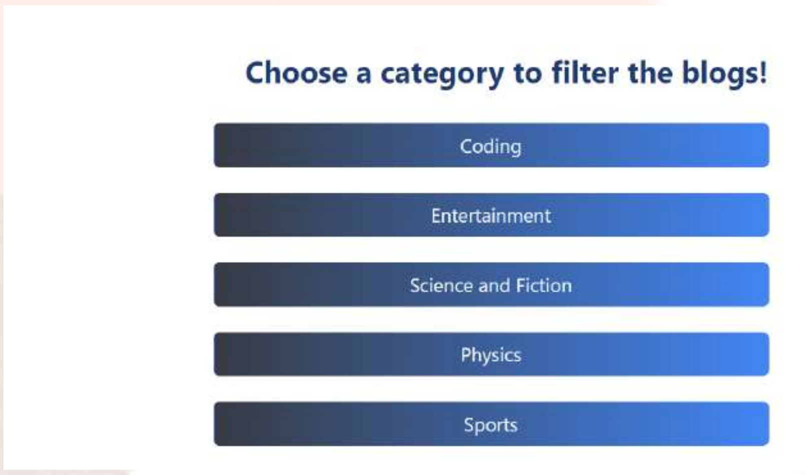
Title: Title

Title tag: Title tag

Author: ---------

Category: Coding

Take a sc


Choose a category to filter the blogs!

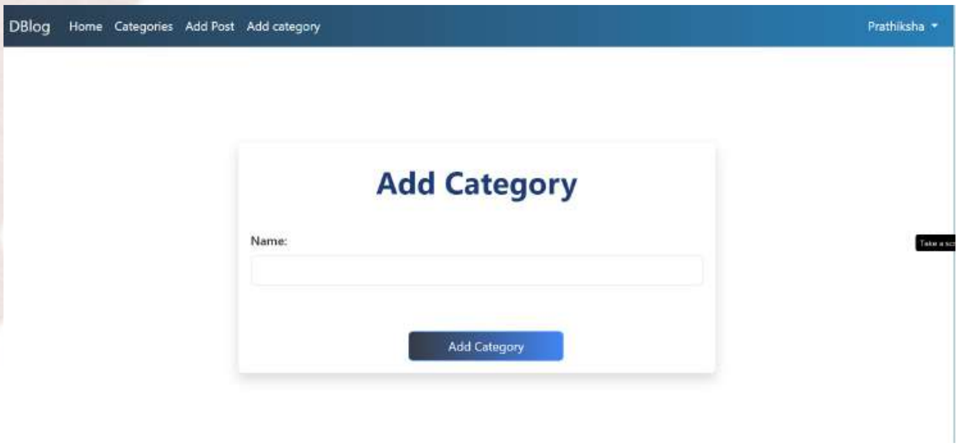
Coding

Entertainment

Science and Fiction

Physics

Sports

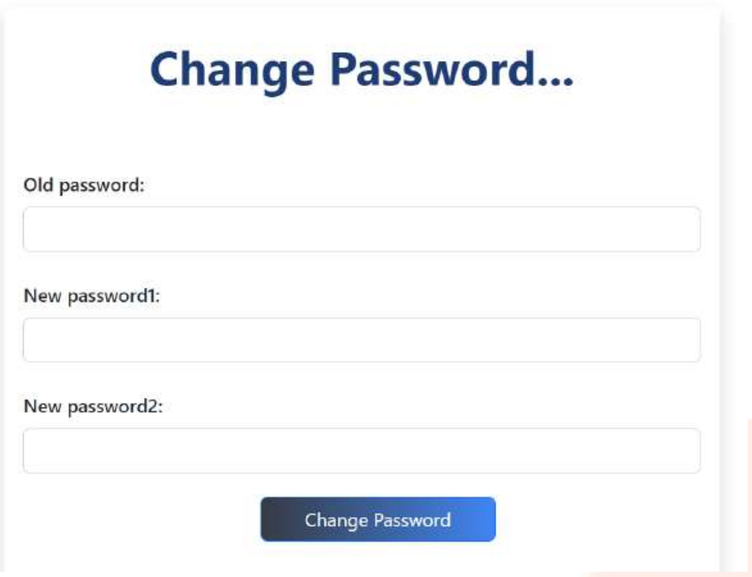

DBlog Home Categories Add Post Add category Prathiksha

Add Category

Name:

Add Category

Take a sc


Change Password...

Old password:

New password1:

New password2:

Change Password

Conclusion:

The Django framework gives us a simple and reliable way to create a blog application system. It provides powerful functionalities and concise syntax to help programmers deal with the database, the web page, and the inner logic. The experience of developing the database tables with SQLite in the system also helped me learning a lot of website development with Django. SQLite tables with Django framework are suitable to store massive datasets and yet, perform the queries with minimal retrieval time. These tables are logical and normalized.

Reference:

- Django documentation: https://www.djangoproject.com
- Python documentation: https://docs.python.org
- Stack Overflow Solutions
- Sqlite3 documentaion https://www.sqlite.org/docs.html