VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI - 590018

**A MINI-PROJECT REPORT ON**

# "BLOG APPLICATION SYSTEM"

**Submitted in partial fulfillment of the requirements of the award of degree of**

**BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE & ENGINEERING**

**Submitted By:**

| | |
|---|---|
| **Nimesh M** | **[4VV18CS093]** |
| **Mohammad Hussam Khatib** | **[4VV18CS079]** |

**UNDER THE GUIDANCE OF**

| | |
|---|---|
| **Dr. Aditya C R** | **Ms. Harshitha K** |
| **Assoc. Prof** | **Asst. Prof** |
| **Dept of CSE** | **Dept of CSE** |
| **VVCE, Mysuru** | **VVCE, Mysuru** |

**2020-2021**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIDYAVARDHAKA COLLEGE OF ENGINEERING
MYSURU-570002**

Vidyavardhaka College of Engineering
Gokulam 3rd Stage,
Mysuru-570002
**Department of Computer Science and Engineering,**

## CERTIFICATE

This is to certify that the mini-project report entitled **"TITLE OF THE MINI-PROJECT"** is a bona fide work carried out by **NIMESH M** (4VV18CS093), **MOHAMMED HUSSAM KHATIB** (4VV18CS079) students of 5th semester Computer Science and Engineering, Vidyavardhaka **College of Engineering, Mysuru** in partial fulfillment for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi**, during the academic year **2020-2021**. It is certified that all the suggestions and corrections indicated for the internal assessment have been incorporated in the report deposited in the department library. The report has been approved as it satisfies the requirements in respect of mini-project work prescribed for the said degree.

| | | |
|---|---|---|
| **Signature of the Guide** | **Signature of the Guide** | **Signature of the HOD** |
| ____________________ | ____________________ | ____________________ |
| **(Dr. Aditya C R)** | **(Ms. Harshitha K )** | **(Dr. Ravi Kumar V)** |

| | |
|---|---|
| **Name of the Examiners** | **Signature with Date** |
| **1)** | |
| **2)** | |

## ACKNOWLEDGEMENT

The Mini-project would not have been possible without the guidance, assistance and suggestions of many individuals. I would like to express our deep sense of gratitude and indebtedness to each and every one who has helped me to make this project a success.

We heartily thank our beloved Principal, **Dr. B Sadashive Gowda** for his whole hearted support and for his kind permission to undergo the mini-project.

We wish to express our deepest gratitude to **Dr. Ravi Kumar V**, Head of Department, Computer Science and Engineering, VVCE, for his constant encouragement and inspiration in taking up this mini-project.

We gracefully thank our mini-project guides, **Dr. Aditya C R**, Associate Professor and **Ms. Harshitha K**, Assistant Professor, Dept. of Computer Science and Engineering for their encouragement and advice throughout the course of the mini-project work.

In the end, we are anxious to offer our sincere thanks to our family members and friends for their valuable suggestions and encouragement.

**Nimesh M**
(4VV18CS093)
**Mohammed Hussam Khatib**
(4VV18CS079)

# ABSTRACT

A blog is an informational social networking website deployed on the world wide web that consists of posts or text entries. It plays a vital role in communicating thoughts and sharing experiences with people across the globe. It provides users with various features and services such as creating, updating and deleting posts to make their experience of communicating through posts significant. A typical blog combines text, digital information, and links to other blogs.

This report introduces the process of creating a blog application system, which is a data-driven website used by administrators and end-users. This website has four major components: User registration/authentication, User blog posts, polling, and profile creation. Django framework, an open-source web application framework for complex data-driven website development, is used in this project. The vital part of this report will introduce the implementation of the blog application with the Django framework and SQLite3 database management system and the inner logic to perform extensive backend operations.

# TABLE OF CONTENTS

# LIST OF TABLES AND FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 OVERVIEW AND OBJECTIVES

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, and raise polls accordingly. 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 section 3.3 in detail. In the end, snapshots demonstrate front-end development.

- 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.

## 1.2 FEATURES AND SIGNIFICANCE

As mentioned above, this application system has four vital components/features: User registration/authentication, User blog posts, polling, 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, password, and store it in the database (Schema Diagram in Figure 3.2 shows the list of attributes/data collected from a user).
- **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.
- **Polling:** This is an extended feature for the blog application. Any authenticated user can raise polls and receive a quantitative opinion from other users. There are three categories of polling questions just for demonstration purposes.

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.

## 1.3 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.
Figure 1.3 illustrates the model-view-template structure of Django.

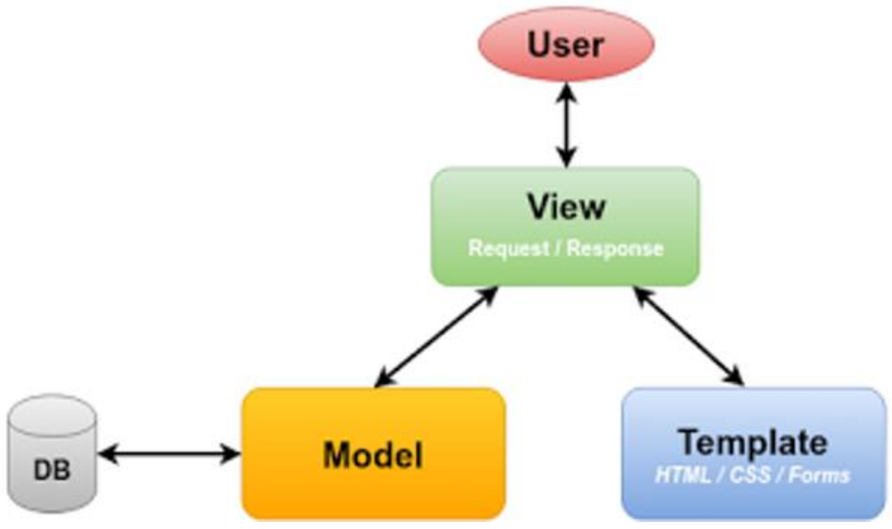


Figure 1.3 Model-View-Template Architecture

## 1.4 REPORT ORGANISATION

In the next chapter, which is Chapter 2, we give the hardware and software requirements for developing this project. Chapter 3 provides information about schema and ER diagrams and the design and implementation of the project. Chapter four concludes the entire project and provides snapshots of the front-end of the website.

# CHAPTER 2

# HARDWARE AND SOFTWARE REQUIREMENTS

## 2.1 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. Below is a table that depicts the systems' hardware and software requirements.

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.

## 2.2 SOFTWARE REQUIREMENTS

This blog application uses python language for development with SQLite database management system. We have used the Django framework version 3.1 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 PyCharm 2020 version.

| **Hardware Requirements** | **Software Requirements** |
|---|---|
| Development end:<br>Processor: Intel Core i3 or above<br>PC RAM Size: 4GB | Development end:<br>Python version: 3.6 or above<br>SQLite version: 3.8 or above |

Table 2 Hardware and Software Requirements

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.

# CHAPTER 3

# DESIGN AND IMPLEMENTATION

## 3.1 SCHEMA DIAGRAM

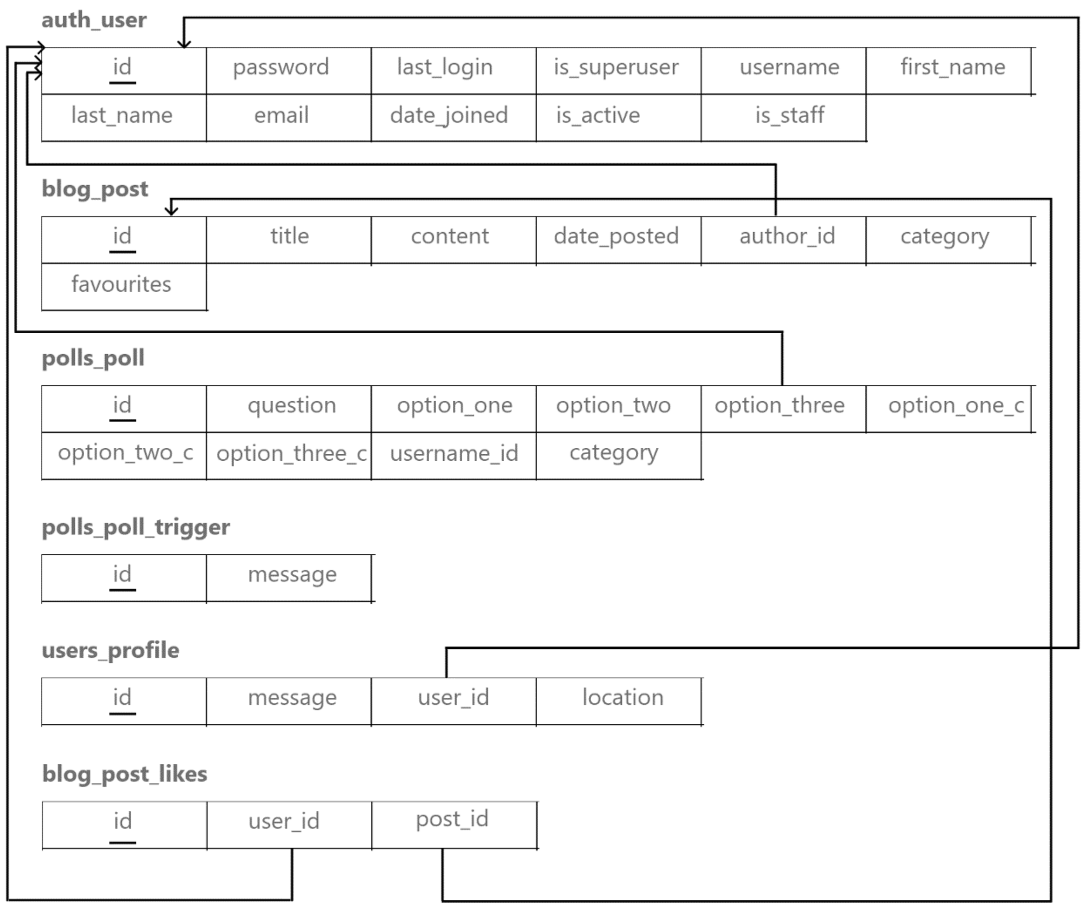


Figure 3.1 Schema Diagram

The above figure (Figure 3.1) represents the schema diagram for the database used in the project. We have created six tables with multiple attributes and relations. The entity-relationship diagram is displayed in figure 3.2.

## 3.2 ENTITY-RELATIONSHIP DIAGRAM

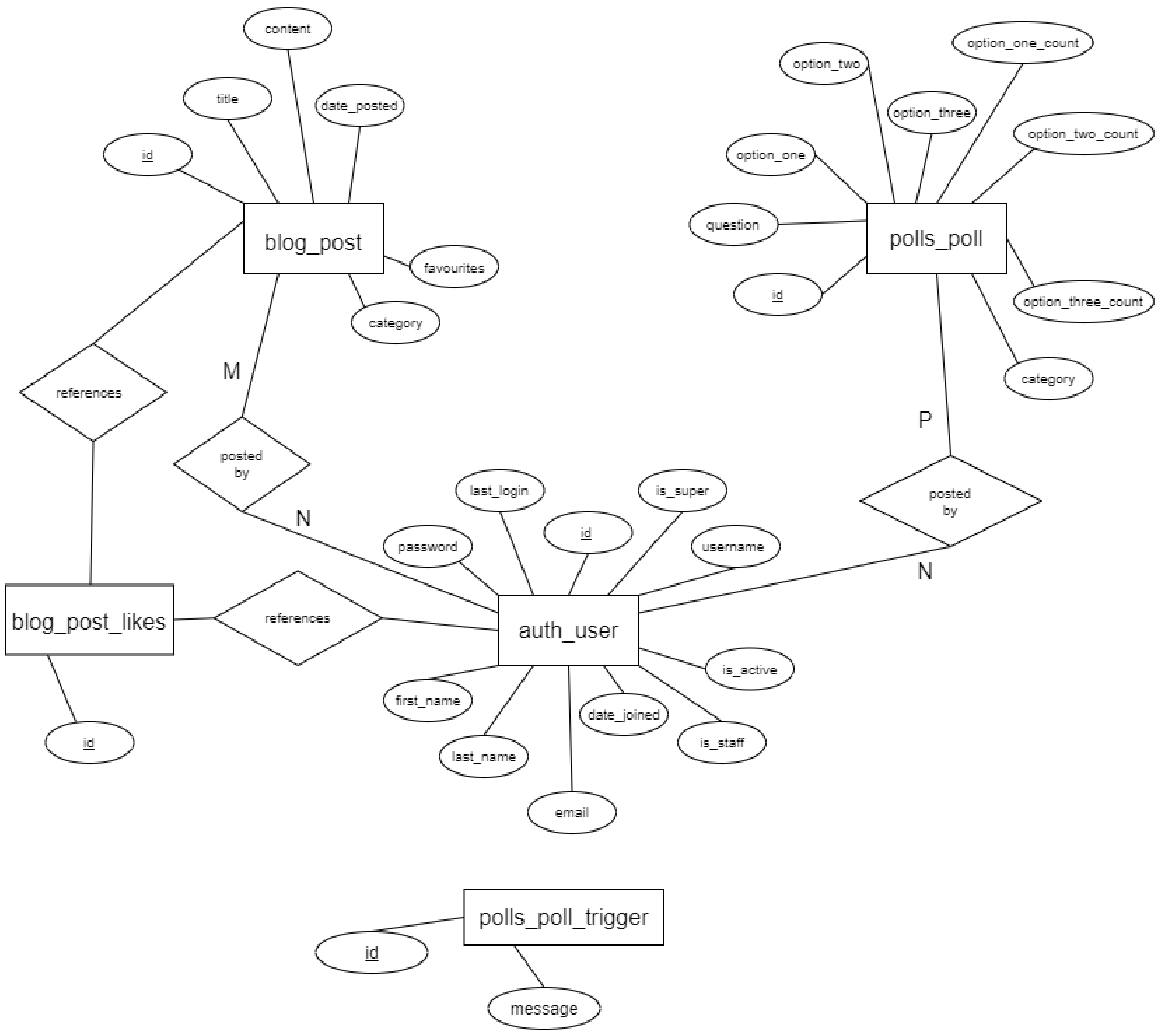


Figure 3.2 Entity Relationship Diagram

The above diagram represents entity-relationship. The polls_poll_trigger table is independent and does not relate with any other table. The table is used to insert data when a trigger is generated.

## 3.3 BACKEND DEVELOPMENT WITH DJANGO AND SQLITE

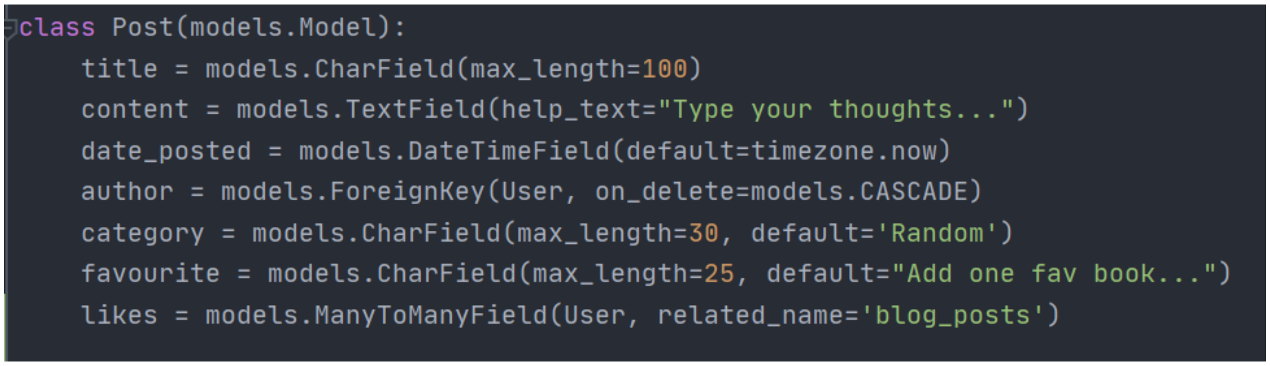
This section covers the backend development and overall logic used to develop this project. In section 3.3.1, we will describe the creation of tables. In section 3.3.2, we will describe python query sets and RAW SQL such as triggers, joins, etc. used in Django. And finally, in section 3.3.3 we will describe the gist of Django signals used for profile creation.

### 3.3.1 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, polls, 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. 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=100)
    content = models.TextField(help_text="Type your thoughts...")
    date_posted = models.DateTimeField(default=timezone.now)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    category = models.CharField(max_length=30, default='Random')
    favourite = models.CharField(max_length=25, default="Add one fav book...")
    likes = models.ManyToManyField(User, related_name='blog_posts')
```

Figure 3.3.1 Creating a table

This code snippet (Figure 3.3.1) creates a table blog_post and stores information when a blog is posted. This is one table representation. "likes" and "author" objects refers to the auth_user (User) table. The table name takes the convention 'folderName_className' and everything is in lowercase. Django structures every component of a website in the form of modular apps (folders). Every time a user posts a blog, the data is inserted into a tuple of the blog_post table.

### 3.3.2 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. This application uses SQL to perform complex queries that use various SQL clauses and operations such as creating triggers, etc.

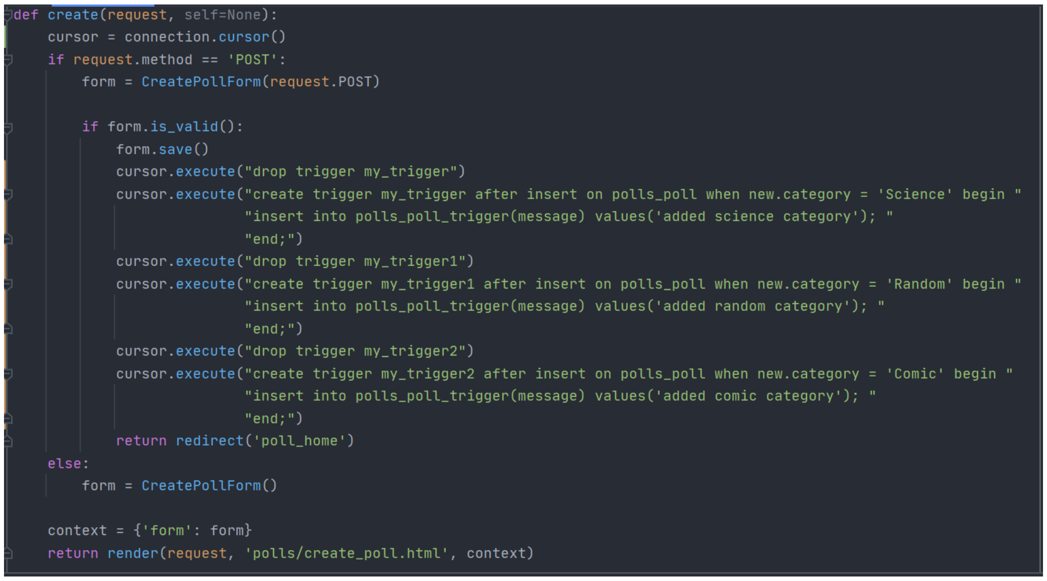
Python Query sets:

- Save method: To save an object/tuple in the database table, we use the save() method. If a model has an AutoField — an auto-incrementing primary key — then that auto-incremented value will be calculated and saved as an attribute on your object the first time you call save().
- Add method: We use add() method to add a record to the relation. Add method is generally called for ManyToManyField in Django.
- All method: The simplest way to retrieve an object (tuple) from the database table is to call all() method when using query sets.
- Filter method: filter() method is used when we need to perform queries with certain conditions.
- Get method: get() method is similar to filter() method except that, get() will raise an exception “DoesNotExist” if no results match the query.

SQL Implementation:

- Clauses: Many clauses have been used in this project. Few of which include order by, group by, inner joins, and unions, etc. These clauses have been used to display certain information to the user on the screen.
- Triggers: Two triggers are used on the polls poll table. The user can raise polls under either of three categories: comic, random, and science. When a user raises a poll, a trigger is generated based on the category received and inserts data into the polls_poll_trigger table. The data inserted is used to display the type of the recent poll.

Figure 3.3.2 is a code snippet of the use of triggers in the create function of polls poll table. The concept of views is explained in section 3.4.



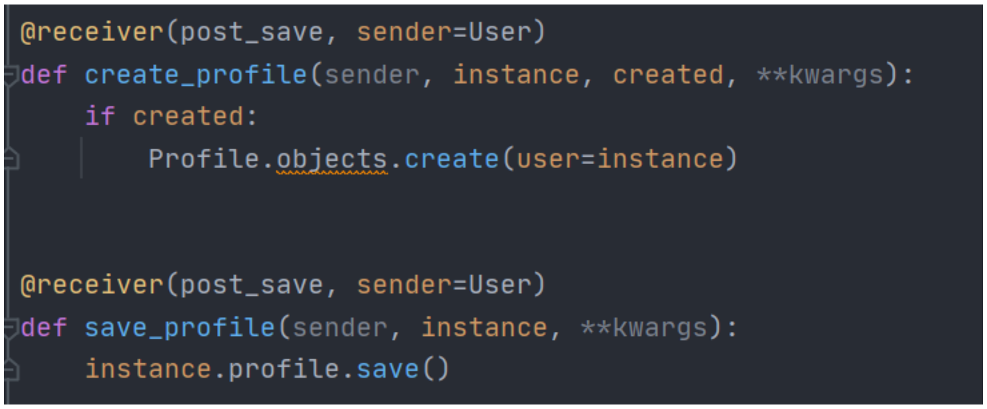
```python
def create(request, self=None):
    cursor = connection.cursor()
    if request.method == 'POST':
        form = CreatePollForm(request.POST)

        if form.is_valid():
            form.save()
            cursor.execute("drop trigger my_trigger")
            cursor.execute("create trigger my_trigger after insert on polls_poll when new.category = 'Science' begin "
                           "insert into polls_poll_trigger(message) values('added science category'); "
                           "end;")
            cursor.execute("drop trigger my_trigger1")
            cursor.execute("create trigger my_trigger1 after insert on polls_poll when new.category = 'Random' begin "
                           "insert into polls_poll_trigger(message) values('added random category'); "
                           "end;")
            cursor.execute("drop trigger my_trigger2")
            cursor.execute("create trigger my_trigger2 after insert on polls_poll when new.category = 'Comic' begin "
                           "insert into polls_poll_trigger(message) values('added comic category'); "
                           "end;")
            return redirect('poll_home')
    else:
        form = CreatePollForm()

    context = {'form': form}
    return render(request, 'polls/create_poll.html', context)
```

Figure 3.3.2 Triggers

### 3.3.3 DJANGO SIGNALS

The **signals** are utilities that allow us to associate **events** with **actions**. We can develop a function that will run when a **signal** calls it.

The most common example of using Django signals is when we want to create a user profile as soon as the user creates his account which can be done by creating a signals.py file in your app directory with the following code:



```python
@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)


@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()
```

Figure 3.3.3 Creating a signal

We have a sender "sender=User" and a signal of "post_save". The "post_save" signal sends a signal when a User is saved. The signal will be received by the receiver which here, is the create_profile function. The

create_profile function takes all of the arguments that our post_save signal passed to it, one of which is an instance. Then we check if that user was created and then create a profile object with the user equal to the instance of the user that was created.
In the second function, we are just saving the profile once the user saves his account details and such.

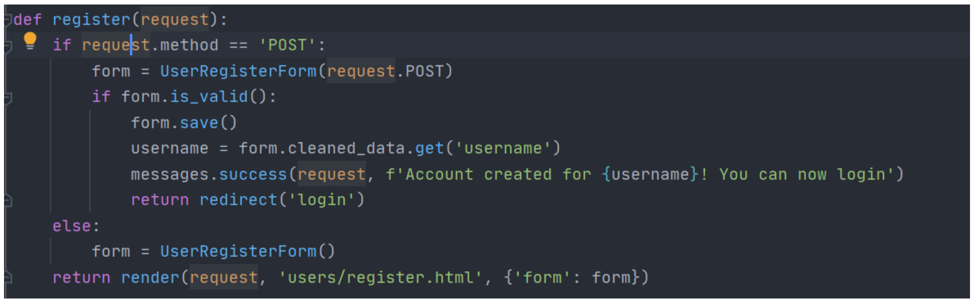
## 3.4 USER INTERFACE AND VIEW FUNCTIONS

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 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.



```python
def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Account created for {username}! You can now login')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})
```

Figure 3.4 View function

# CHAPTER 4

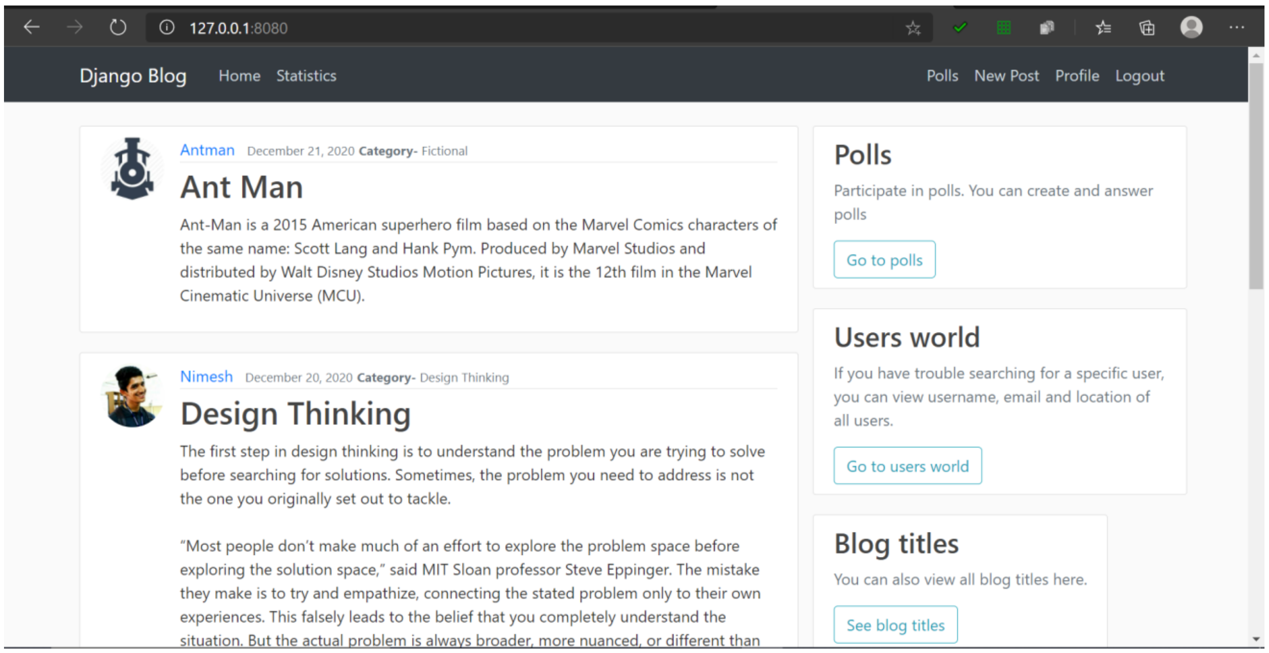
# RESULTS

## 4.1 OBSERVATIONS AND LEARNING

There were interesting observations and learnings made in the process of developing the blog application system.

- 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.
- Triggers and Signals are an essential part of DBMS operations to track certain changes made in the DBMS when a user interacts with the product.
- 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.

## 4.2 SNAPSHOTS OF THE PROJECT



Antman December 21, 2020 **Category-** Fictional

Ant Man

Ant-Man is a 2015 American superhero film based on the Marvel Comics characters of the same name: Scott Lang and Hank Pym. Produced by Marvel Studios and distributed by Walt Disney Studios Motion Pictures, it is the 12th film in the Marvel Cinematic Universe (MCU).

Nimesh December 20, 2020 **Category-** Design Thinking

Design Thinking

The first step in design thinking is to understand the problem you are trying to solve before searching for solutions. Sometimes, the problem you need to address is not the one you originally set out to tackle.

"Most people don't make much of an effort to explore the problem space before exploring the solution space," said MIT Sloan professor Steve Eppinger. The mistake they make is to try and empathize, connecting the stated problem only to their own experiences. This falsely leads to the belief that you completely understand the situation. But the actual problem is always broader, more nuanced, or different than

Polls

Participate in polls. You can create and answer polls

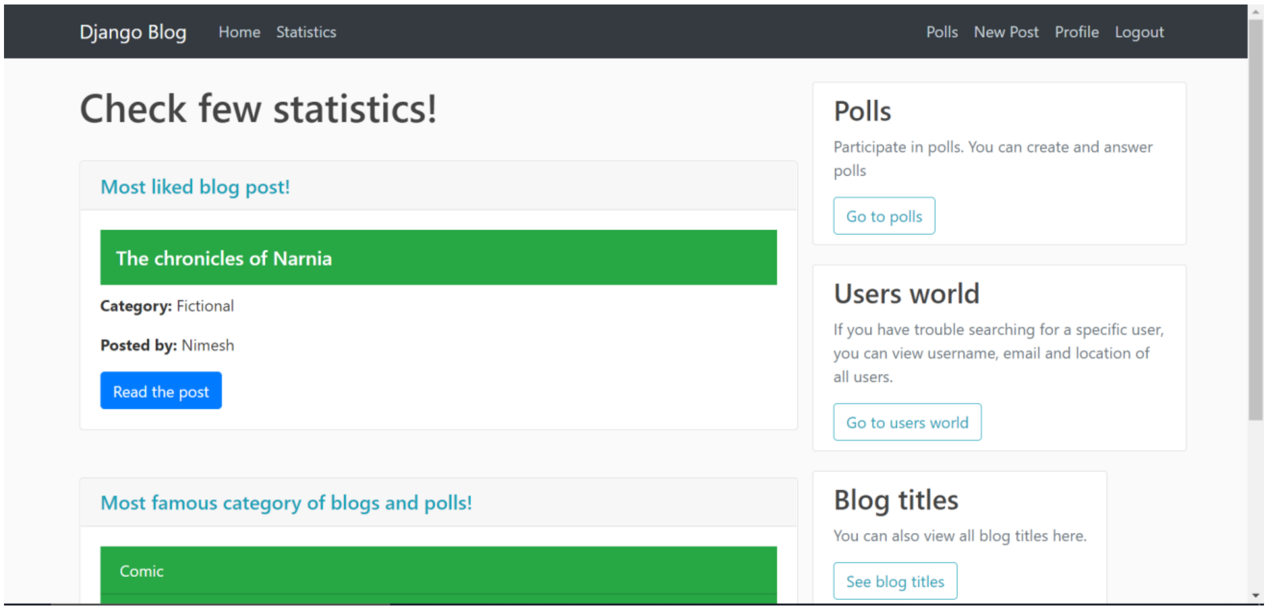
Users world

If you have trouble searching for a specific user, you can view username, email and location of all users.

Blog titles

You can also view all blog titles here.

Figure 4.2.1 Home Page



Check few statistics!

Most liked blog post!

The chronicles of Narnia

**Category:** Fictional

**Posted by:** Nimesh

Most famous category of blogs and polls!

Comic

Polls

Participate in polls. You can create and answer polls

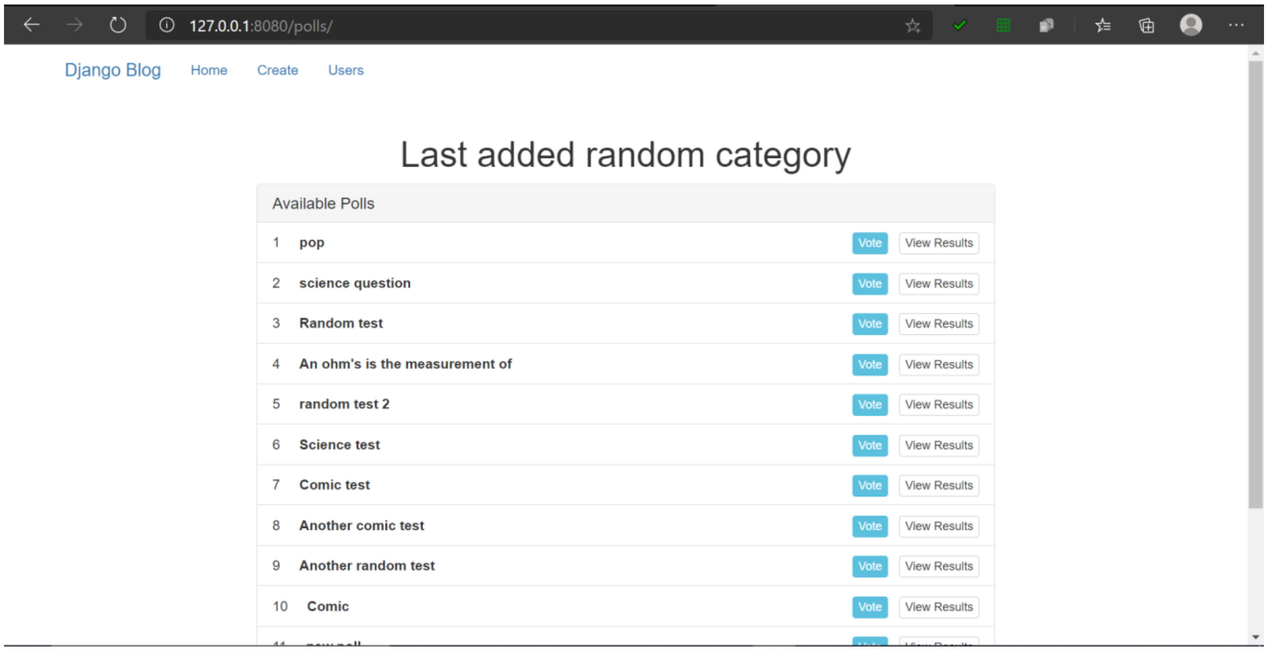
Users world

If you have trouble searching for a specific user, you can view username, email and location of all users.

Blog titles

You can also view all blog titles here.

Figure 4.2.2 Statistics Page

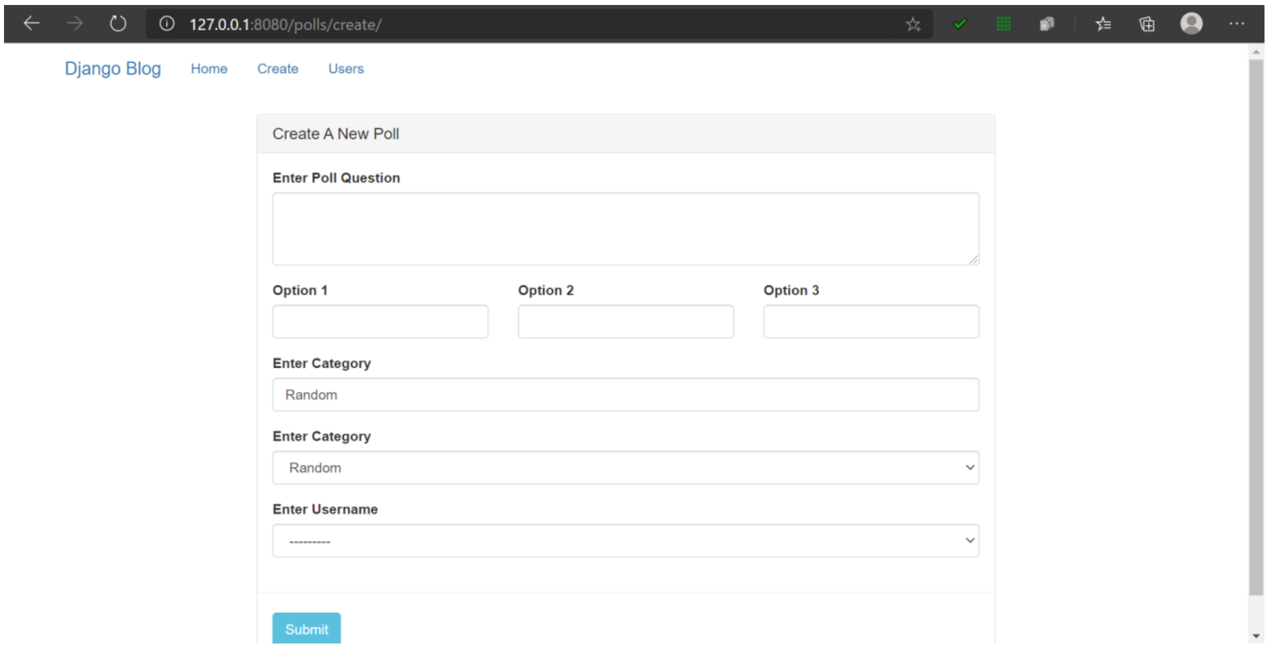


Last added random category

Available Polls

| | | | |
|---|---|---|---|
| 1 | pop | Vote | View Results |
| 2 | science question | Vote | View Results |
| 3 | Random test | Vote | View Results |
| 4 | An ohm's is the measurement of | Vote | View Results |
| 5 | random test 2 | Vote | View Results |
| 6 | Science test | Vote | View Results |
| 7 | Comic test | Vote | View Results |
| 8 | Another comic test | Vote | View Results |
| 9 | Another random test | Vote | View Results |
| 10 | Comic | Vote | View Results |

Figure 4.2.3 Polls Page


127.0.0.1:8080/polls/create/

Django Blog Home Create Users

Create A New Poll

Enter Poll Question

Option 1 Option 2 Option 3

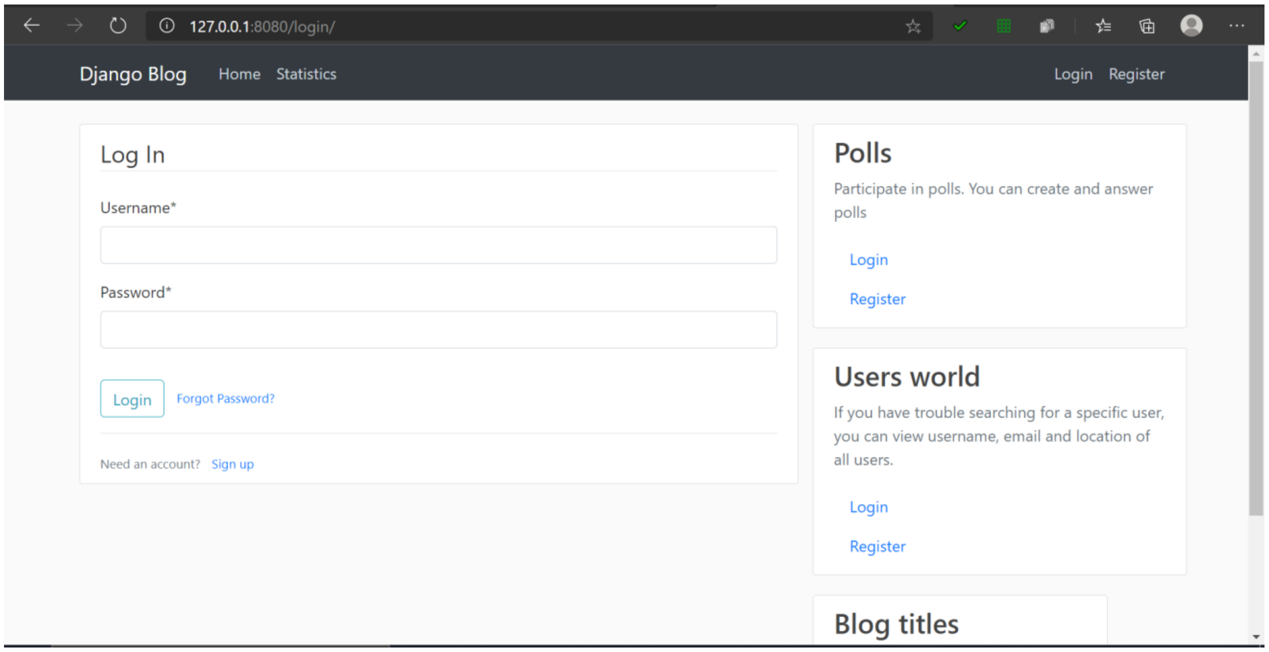
Enter Category: Random

Enter Category: Random

Enter Username: ---------

Submit

Figure 4.2.4 Poll Creation Page


127.0.0.1:8080/login/

Django Blog Home Statistics Login Register

Log In

Username*

Password*

Login Forgot Password?

Need an account? Sign up

Polls

Participate in polls. You can create and answer polls

Login

Register

Users world

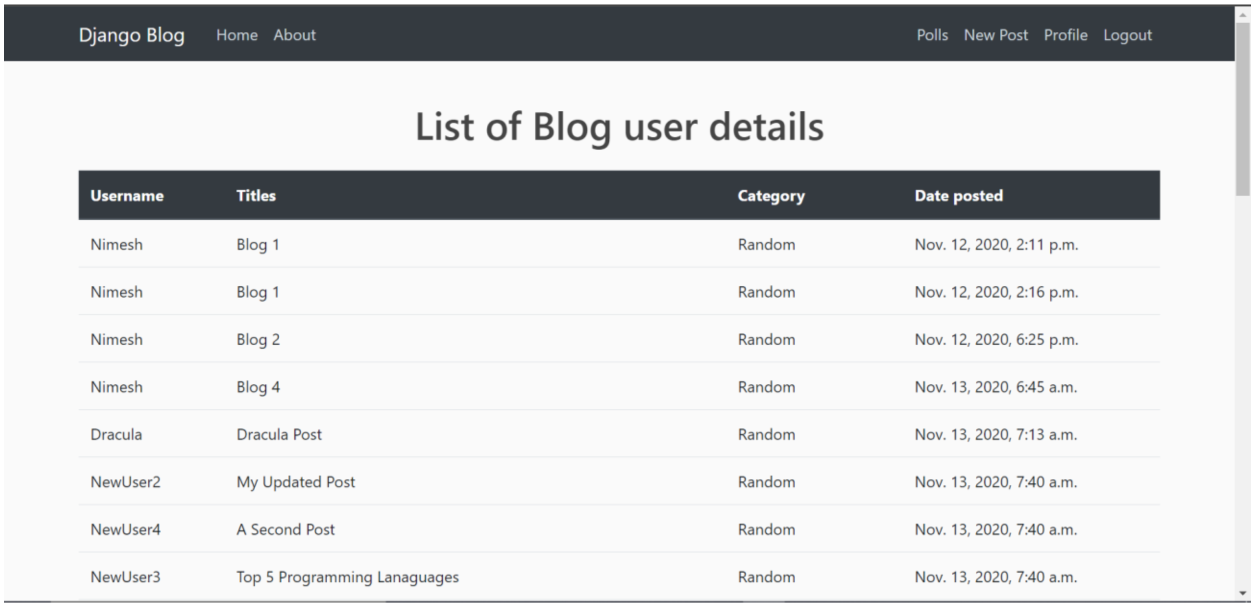
If you have trouble searching for a specific user, you can view username, email and location of all users.

Login

Register

Blog titles

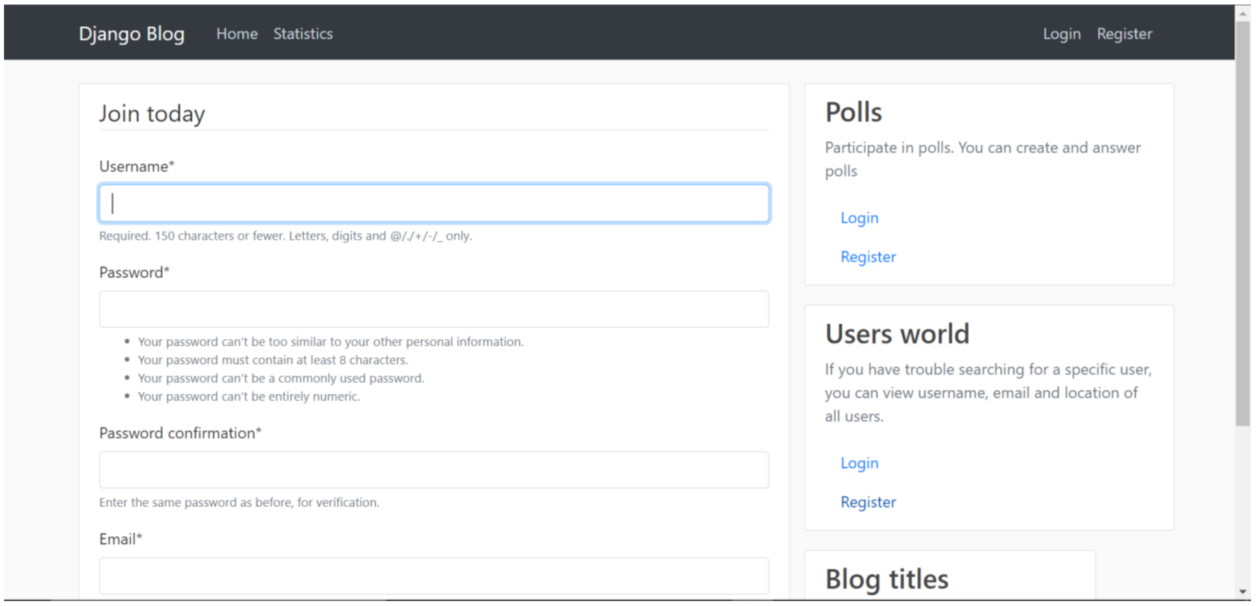
Figure 4.2.5 Login Page



List of Blog user details

| Username | Titles | Category | Date posted |
|---|---|---|---|
| Nimesh | Blog 1 | Random | Nov. 12, 2020, 2:11 p.m. |
| Nimesh | Blog 1 | Random | Nov. 12, 2020, 2:16 p.m. |
| Nimesh | Blog 2 | Random | Nov. 12, 2020, 6:25 p.m. |
| Nimesh | Blog 4 | Random | Nov. 13, 2020, 6:45 a.m. |
| Dracula | Dracula Post | Random | Nov. 13, 2020, 7:13 a.m. |
| NewUser2 | My Updated Post | Random | Nov. 13, 2020, 7:40 a.m. |
| NewUser4 | A Second Post | Random | Nov. 13, 2020, 7:40 a.m. |
| NewUser3 | Top 5 Programming Lanaguages | Random | Nov. 13, 2020, 7:40 a.m. |

Figure 4.2.6 Blog list



Join today

Username*

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

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Email*

Polls

Participate in polls. You can create and answer polls

Login

Register

Users world

If you have trouble searching for a specific user, you can view username, email and location of all users.

Login

Register

Blog titles

Figure 4.2.7 Registration Page

# 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. Within the Django framework, we have accomplished the requirements of the system. Once this system passes the testing phase, it can be further enhanced and made a suitable product for the users. It will make the work for instructors to manage the course much easier. The only limitation for this application system is that although the developers have been testing it with various use cases, it may still encounter problems during real-time use. However, even if that happens, the flexibility of Django and SQLite DBMS would provide a simple way to fix the problem, as well as add new features into the system.

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. Audio, video, and real-time chat integration will enhance the blog application system.

# REFERENCES

1. Django documentation. http://www.djangoproject.com/
2. Python documentation. Our Documentation | Python.org
3. SQLite documentation. https://sqlite.org/docs.html
4. Medium Articles
5. Stack Overflow [for error references]