

Programming Assignment

For :

Advanced Programming Techniques

Under the theme of:

Creation of a blog project: Tricalidee

Réalisée par :

- Aishwarya Passi **096040**
- Kanza Nassabi **776063**
- Manvith Penklagar **775906**
- Omkar Ralhan **775963**

Submitted to:

Prof. Andreas Fischer :

SUMMARY

INTRODUCTION	2
General Presentation	3
1.1. Choice of the programming language:	3
1.2. Choice of the Web Framework:	3
Installation Process	5
2.1. Installation of Softwares needed :	5
2.2. Creation of the blog « Tricalidee »:	6
2.3. Types of Users:	8
2.3.1. Creating an ADMIN:	10
2.3.2. Creating a New User:	11
2.4. Features used in the blog Tricalidee:	13
2.4.1. Add/Delete/Update Post:	13
2.4.2. Add Comment :	13
2.4.3. Tags:	15
2.4.4. Sort:	16
2.4.5. Pagination:	16
Deviation from Plan	16
3.1. Forgot password	16
3.2. Search Button:	17
3.3. Questions and Answers 'Forum':	17
3.4. Checking the Approved or Changing role in database:	18
Code Documentation	18
4.1. Settings:	19
4.2. Models:	19
4.2.1. Models.py:	19
4.2.2. Class Diagram:	20
4.3. Forms:	21
4.4. Views:	22
4.5. Templates	24
4.6. Tests :	26
Usage Documentation	27
Conclusion	28

INTRODUCTION

This project is an electrical blog, for all those Innovative minds out there who wish to display their ideas or seek help with some problems they are facing, when working on something. Our focus is on making the user experience simple, along with providing armature bloggers a sophisticated platform to display their ideas and thoughts. Registering to this blog is free of cost, So that users can enjoy writing and reading blog posts.



An open forum like Tricalidee is a step towards bringing about a revolution in the field of electrical blogging and innovation. The blog has been constructed using Django framework since it is an open-source python web framework used for rapid development, pragmatic, maintainable, clean design, and secures websites.

Chapitre 1:

General Presentation

1.1. Choice of the programming language:

We used Python to write our software application, The blog “Tricalidee” for so many reasons:

- It's a Readable and Maintainable Code, using Python helped us to emphasize on the readability, we use English words instead of punctuations, which was easy for us to handle and learn fast. It allowed us to express concepts without writing additional code, therefore it helped to maintain and update the software easily.
- It has Multiple Programming Paradigms, Python supports object oriented and structured programming, and it features a dynamic type system and automatic memory management.
- It is an Interpreted Programming Language. Which allowed us to run the same code on multiple platforms without recompilation, which was time saving because it was not required to recompile the code after making a change and we could see the impact of the changes we made immediately.
- It has a Robust Standard Library, and huge that it allowed us to choose from a wide range of modules according to our precise needs, and each module enabled us to add other functions without writing some additional code.
- It has many Open Source Frameworks and Tools, we had so many choices to pick from, and after making the right decision according to our needs, it speeded up the web application development.

1.2. Choice of the Web Framework:

Python has several open source frameworks like Django, Flask, Pyramid, Bottle and Cherryypy. However, for our project, we choose Django because of the following:

- Django has a clean pythonic structure. It's a Model–View–Controller (MVC), The MVC architecture allowed us to change the visual part of an app and the business logic part separately, without their affecting one another. Also the three layers (Model, View, and Template) are responsible for different things and can be used independently.

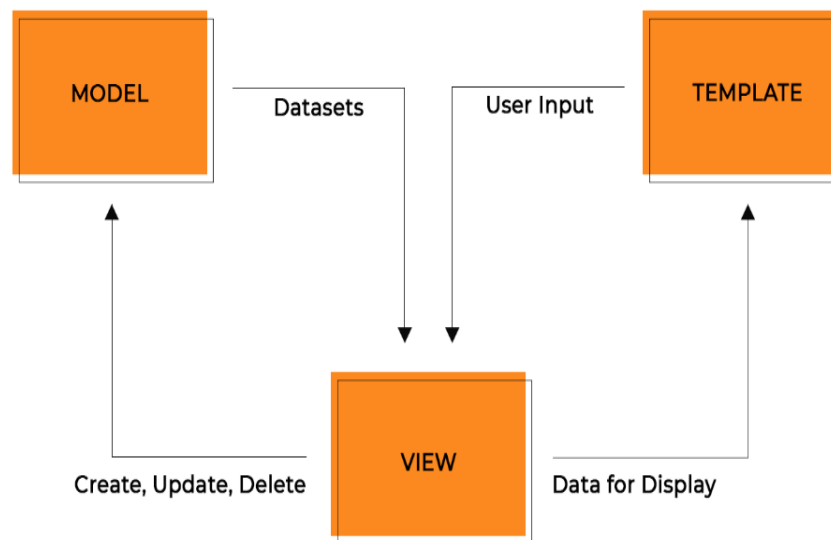


Figure 1: Django's Architecture MVT

- It has a Rich Ecosystem. Because many third-party applications can come with Django. These applications can be integrated depending on project requirements.
- Whenever we were trying to figure out how it works with Django, we could find the answers in Django Documentation, which was very helpful for us as Beginners.
- Having an Admin panel by default. Which helped us to manage our application easily. And there was a lot of room for customization thanks to having the third party applications, and it allowed us to modify the interface with using wrappers according to the touch we wanted to add to our blog.
- Its efficient object-relational mapper that helped to interact with databases. The library (ORM) was automatically transferring the data which is stored in databases such as PostgreSQL and MySQL into objects commonly used in the application code.

Chapitre 2:

Installation Process

2.1. Installation of Softwares needed :

We installed Python and Atom as a python editor, to make it easy for us to organize the code.

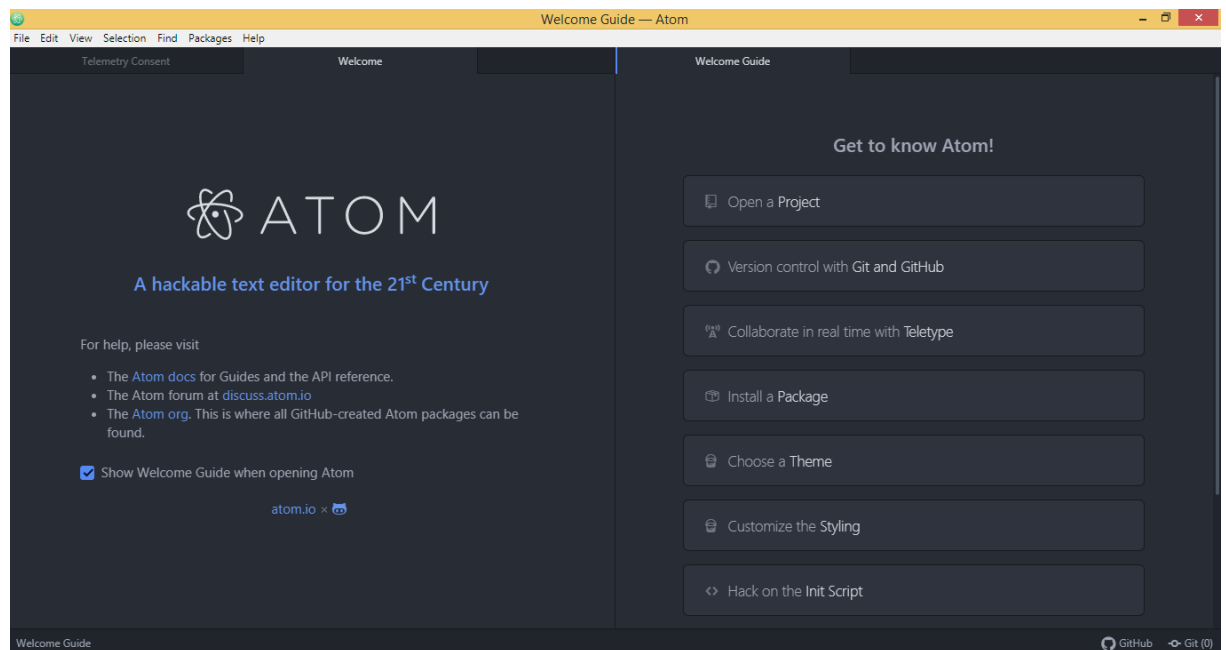


Figure 2; Python editor ATOM

We also installed TortoiseGit which is a free open-source client for the Git version control system. TortoiseGit manages files over time. Files are stored in a local repository.



Figure 3; TortoiseGit

The repository is much like an ordinary file server, except that it remembers every change ever made to our files and directories. This allowed us to recover older versions of our files and examine the history of how and when our data changed, and who changed it.

2.2. Creation of the blog « Tricalidee »:

You can manually do the installation process by following the steps defined below:

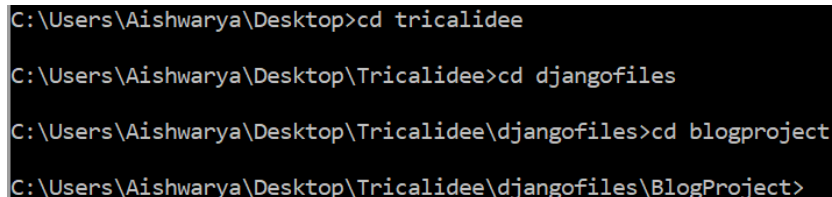
1. Clone the whole project into the local storage. The link of the repository for Tricalidee is:

<https://mygit.th-deg.de/mp26906/tricalidee>

Once the files are pulled to the local system, the blog needs to be made functional. In order to make it functional we need to run the website locally on a personal computer or laptop.

2. Navigate to the root project name 'BlogProject'. This can be done by the command '**cd<space>folder name**' in command prompt. To move out of a directory use '**cd..**'.

The Figure 4 depicts the navigation from one directory to another. ('Tricalidee' is the local folder name where the project has been cloned; you can use any name you want.)



```
C:\Users\Aishwarya\Desktop>cd tricalidee
C:\Users\Aishwarya\Desktop\Tricalidee>cd djangofiles
C:\Users\Aishwarya\Desktop\Tricalidee\djangofiles>cd blogproject
C:\Users\Aishwarya\Desktop\Tricalidee\djangofiles\BlogProject>
```

Figure 4: Command Window Navigation

3. Now if the python version which is used is 2 >=2.7.9 or Python 3 >=3.4. pip is already installed, we will need it to install Django and all the requirements we need. Before installing Django, the developer will need a virtual environment, either it has to be created or activated.
4. Once having the virtualenv, Django can be installed running the command: "pip install Django"
5. Using 'windows commander' as well, the developer needs to install the file requirements.txt, instead of installing all the list of software packages that are needed to for the blog to be functional, we set all of those in one file.

```

requirements.txt
1  atomicwrites==1.3.0
2  attrs==19.1.0
3  colorama==0.4.1
4  Django==2.2
5  django-clear-cache==0.3
6  django-crispy-forms==1.7.2
7  django-discover-runner==1.0
8  django-role-permissions==2.2.1
9  entrypoints==0.3
10 flake8==3.7.7
11 flake8-django==0.0.4
12 importlib-metadata==0.18
13 mccabe==0.6.1
14 more-itertools==7.0.0
15 packaging==19.0
16 Pillow==6.0.0
17 pluggy==0.12.0
18 py==1.8.0
19 pycodestyle==2.5.0
20 pyflakes==2.1.1
21 pyparsing==2.4.0
22 pytest==4.6.3
23 python-slugify==3.0.2
24 pytz==2019.1
25 six==1.12.0
26 sqlparse==0.3.0
27 text-unidecode==1.2
28 wcwidth==0.1.7
29 zipp==0.5.1

```

Figure 5:Package Requirements

6. The file requirements.txt can be simply installed by typing the command: **pip install -r requirements.txt**, as it shown below in Figure 6.

```

C:\Users\Aishwarya\Desktop\Tricalidee\django\files\BlogProject>pip install -r requirements.txt

```

Figure 6:Installing requirements.txt file

7. Once the requirements have been installed successfully, the developer will be able to run the server by typing the command “python manage.py runserver”. When the command is executed successfully a development server is created with IP address <http://127.0.0.1:8000/> as it shown in Figure 7. To quit the server, use the keys CTRL+C. This IP address will direct the user to the Tricalidee homepage.


```
CA. Command Prompt - python manage.py runserver
C:\Users\Aishwarya\Desktop\Tricalidee\django\files>cd blogproject

C:\Users\Aishwarya\Desktop\Tricalidee\django\files\BlogProject>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 27, 2019 - 15:59:11
Django version 2.2, using settings 'tricalidee.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 7:Development Server Creation

The homepage displays all the blog posts that have been posted by authors under various topics. And all users can have access to this page, it will be shown for Admins, Authors, Readers, Anonymous Users as well.

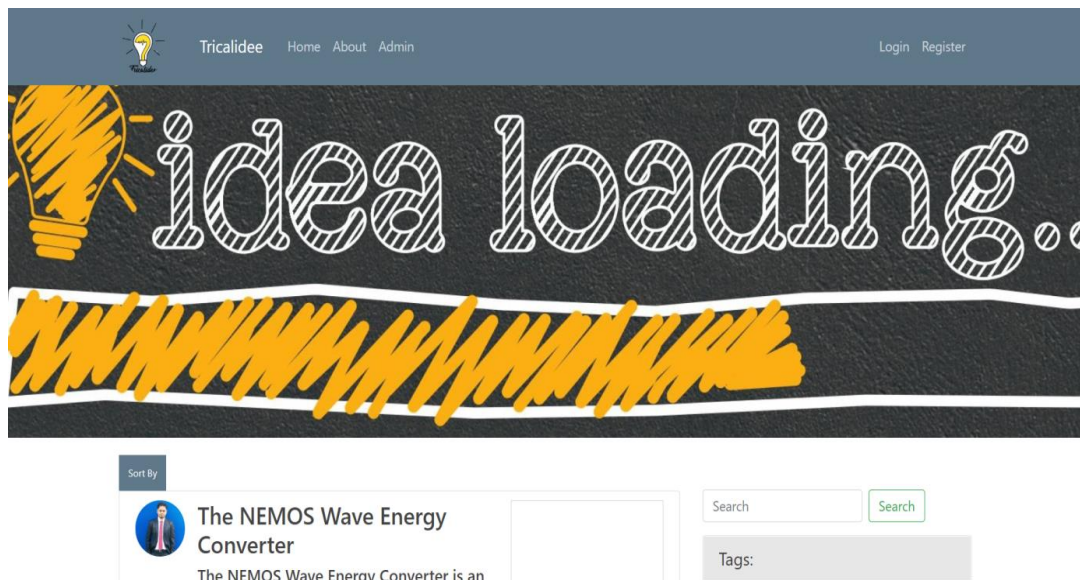


Figure 8:Blog Homepage

2.3. Types of Users:

We have four types of users, and each one of them has certain rights and permissions.

- **Admin:** Is the one who has control of the whole blog, the admin can give roles to users who wants to register in the blog, can change those roles according to the requests received from the users, can delete a post or modify it, approve a comment of an anonymous user or deny it, remove an account.

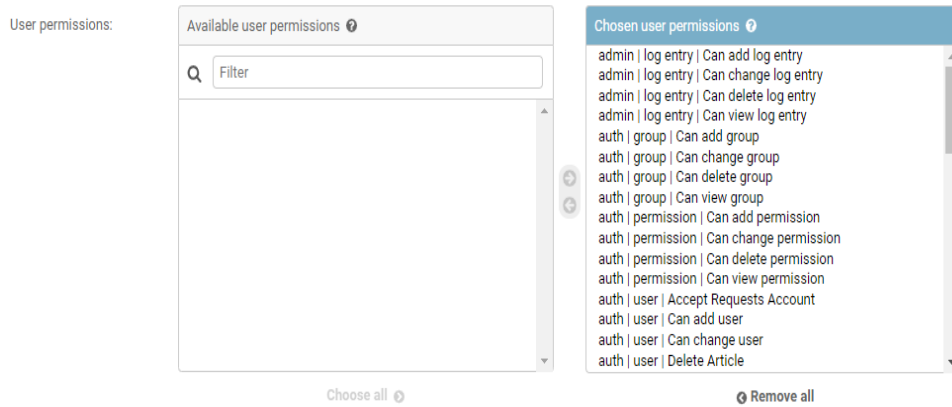


Figure 9:Admin rights

- Author: Can post articles, modify them or delete them, approve a comment from an anonymous user about an article the author posted, comment on other post without being inspected by the admin.

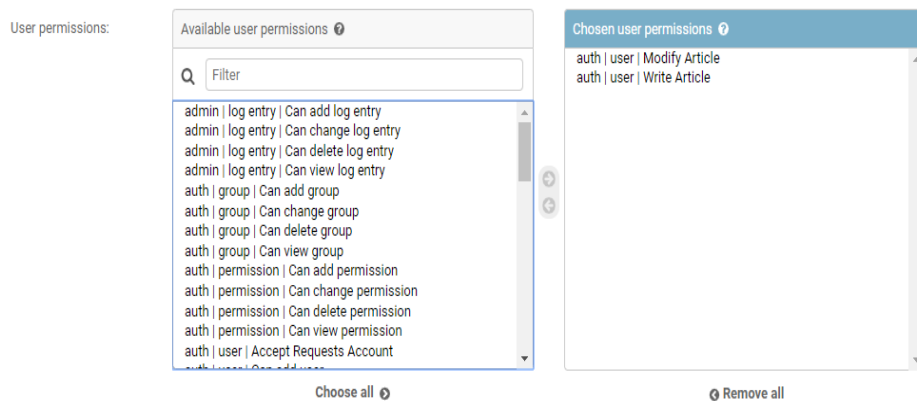


Figure 10:Author rights

- Reader or RegisteredUser: Is a user who had registered and can only read articles, can comment on posts as well without the comment being approved by the admin.

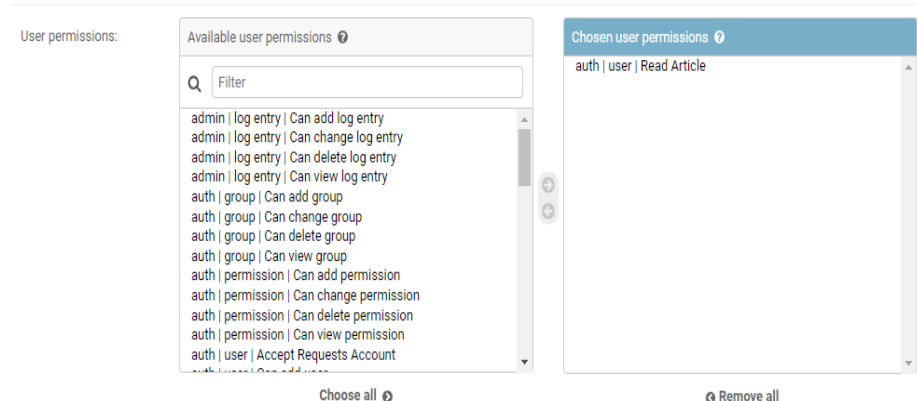


Figure 11:Reader rights

- **Anonymous User:** is a user who didn't register to the blog, however has access to see the posts, comment on them but it won't be displayed unless it's approved by the admin or the author of the post.

2.3.1. Creating an ADMIN:

Basically the admins are the owners of a blog that possess all rights of the blog. They can control what is being displayed in the blog and give permission to users to have a certain role. They can delete and modify the content posted by other user if the admin feels it is not in line with the ideology of the blog, and to approve or deny a comment from an anonymous user.

To be an admin ,these steps can be followed to make it happen:

1. In the command window go to the root directory where the project is present.
2. Then type the command “python manage.py createsuperuser”
3. Enter your desired user name and password. Once the admin is created, use the ‘python manage.py runserver’ command to develop the server connection.

```
C:\Users\Aishwarya\Desktop\Tricalidee\django\files\BlogProject>python manage.py createsuperuser
Username: tricalidee
Email address: tricalidee@gmail.com
Password:
Password (again):
Superuser created successfully.
C:\Users\Aishwarya\Desktop\Tricalidee\django\files\BlogProject>
```

Figure 12:Creating Admin Login

Then the developer can use the actual blog to be redirected to Django framework by clicking on Admin, so he gets access the database or all the control actions of the blog, To get access to all features of the blog such as approve user requests, changing their roles, the admin can click on login button in the navigation bar:



Figure 13:Navigation Bar

Once the admin logs in, on the right side they can find all requests pending for approval.

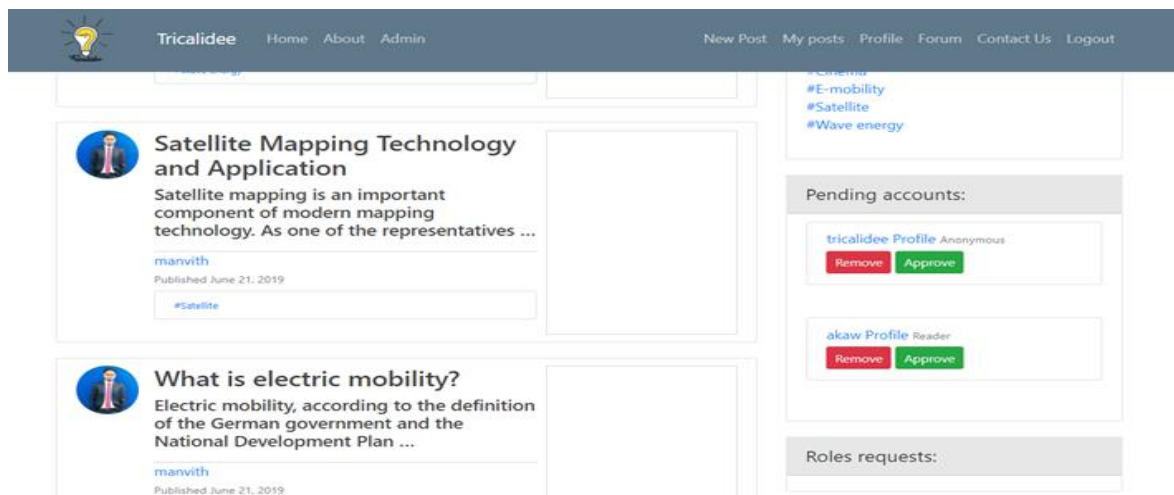


Figure 14:Admin view

The admin also needs to approve the comments posted by the Anonymous users by going to the concerned post and choose to approve or cancel, if it's denied the comment will not be displayed.

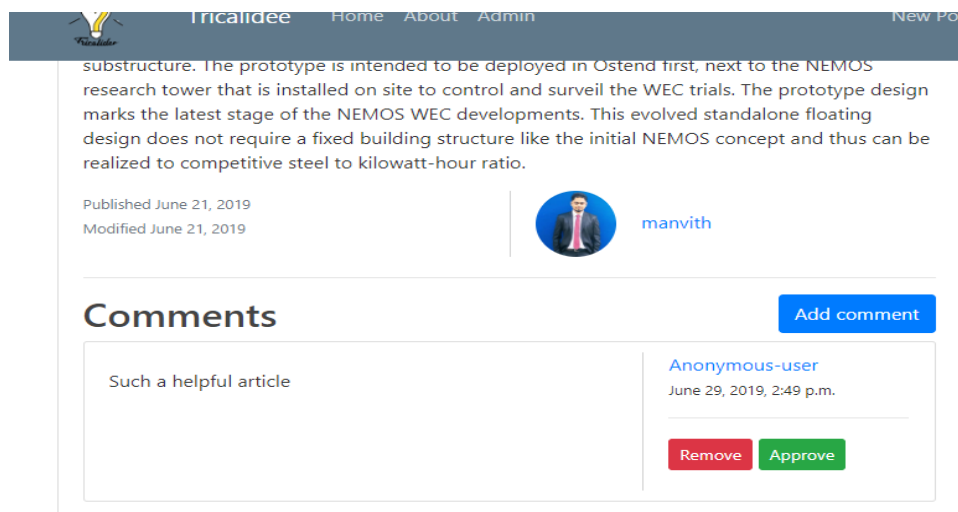
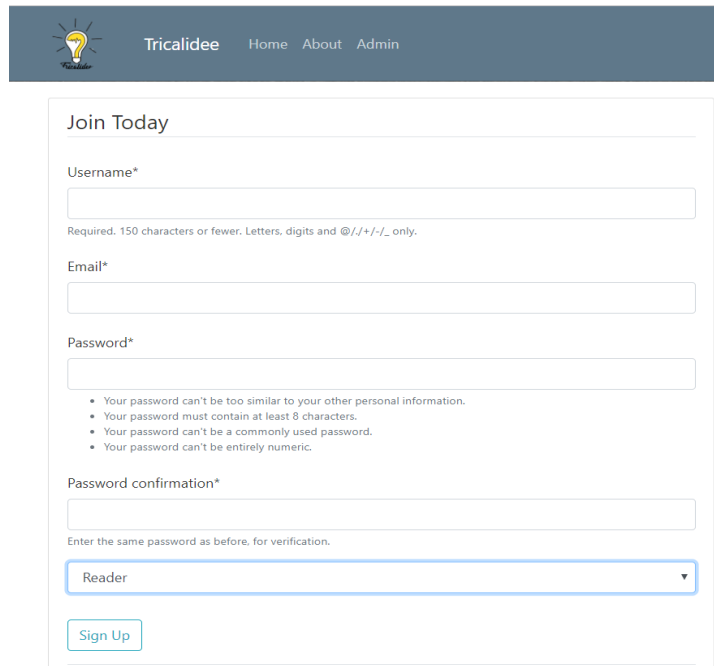


Figure 15:Admin view by Anonymous user

2.3.2. Creating a New User:

Tricalidee has two types of registered user. One is the Reader, other is the Author. When a new user registers to the Tricalidee blog, they need to select the type of account they wish to possess. If the new user went directly to sign-up, he/she will be considered as a reader by default. The sign-up button when clicked after entering all details, then redirects to the login page, when the registered user can enter their credentials and log into their account.



The registration form is titled "Join Today" and is located within a dark blue header bar that also contains the "Tricalidee" logo and navigation links for "Home", "About", and "Admin". The form itself is white with a light gray border. It contains the following fields and elements:

- Username***: A text input field with a placeholder. Below it, a small gray note states: "Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only."
- Email***: A text input field with a placeholder.
- Password***: A text input field with a placeholder. Below it, a list of password requirements is shown:
 - 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***: A text input field with a placeholder. Below it, a small gray note states: "Enter the same password as before, for verification."
- Reader**: A dropdown menu with a blue border and a downward arrow, currently showing "Reader".
- Sign Up**: A blue button with white text.

Figure 16:Registration form

2.3.2.1. Reader or RegisteredUser:

If a new user registered as a reader, then the functionality the user has in the blog is limited. The navigation bar as well is different for readers; it will be the same as an anonymous User (Figure 8) just with a permission of commenting on posts without being approved. Readers cannot add/ modify or delete posts.

2.3.2.2. Author:

If a new user registered as an author, they can login at the instant with limited features activated. After approval from the admin the author can create posts, Update and delete only their own posts. The navigation bar is different for an Author. An author has a profile created for them, the author can access all the posts put up by them. Also they can access the Tricalidee discussion forum page.

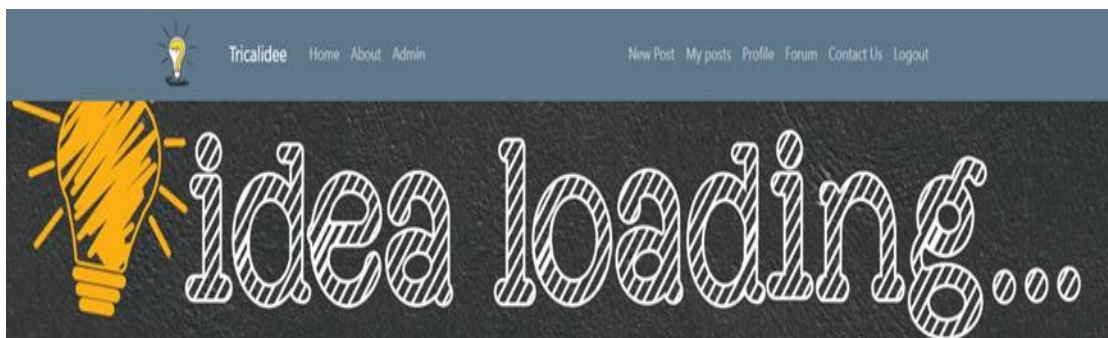
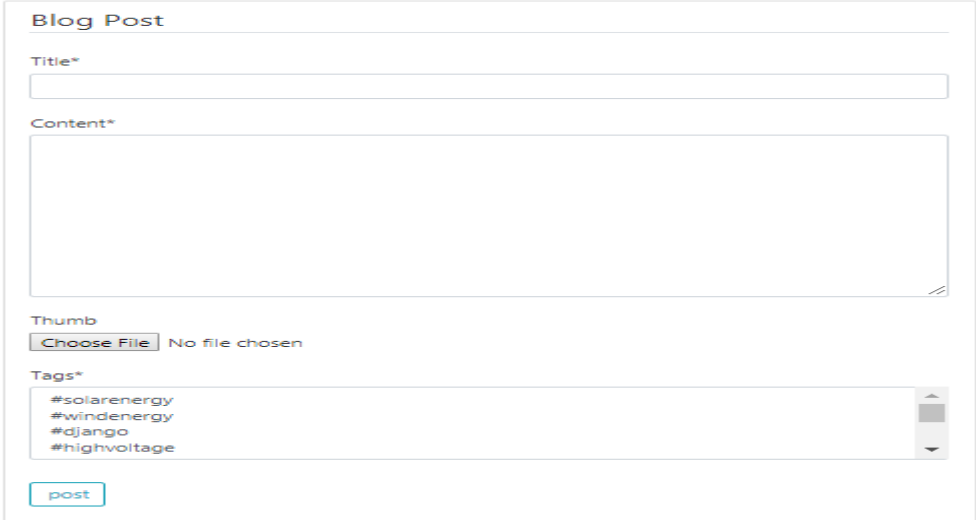


Figure 17:Author View

2.4. Features used in the blog Tricalidee:

2.4.1. Add/Delete/Update Post:

To post a new article, the Author or the Admin needs to click on new post, so it will get redirect to the following page, where Title, Content should be filled, and a Tag(s) should be chosen. If the Thumb, which is the Picture used for the Article post isn't in the right Format 'JPG,JPEG,PNG', the user will be redirect to the same page over again to put the right picture format.

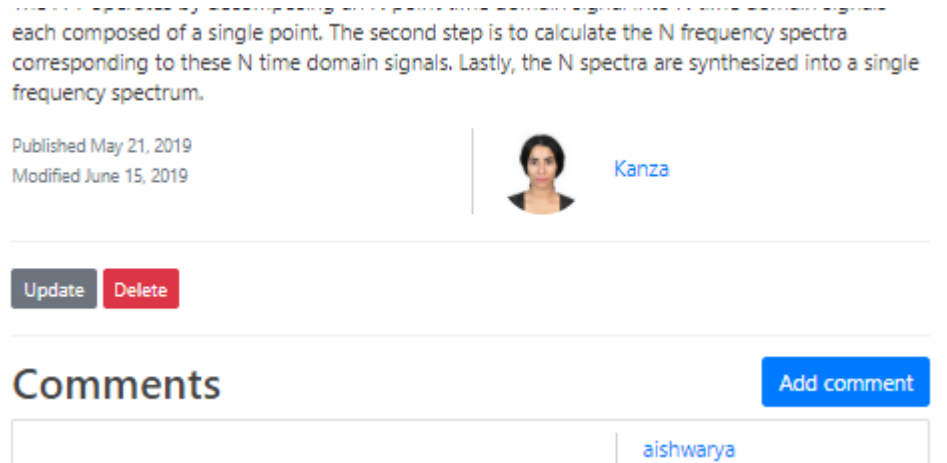


The screenshot shows a 'Blog Post' form with the following elements:

- Title***: A text input field.
- Content***: A large text area for the post content.
- Thumb**: A section for the post image, featuring a 'Choose File' button and the text 'No file chosen'.
- Tags***: A dropdown menu showing a list of tags: #solarenergy, #windenergy, #django, and #highvoltage.
- post**: A button to submit the new post.

Figure 18:New Post Form

The following figure, shows the page that displays when an admin or an author wants to delete or update a post.



The screenshot displays the 'Update-Delete Post page' with the following components:

- Post Snippet**: A preview of the post content, including text about frequency spectra.
- Metadata**: Publication date (May 21, 2019) and modification date (June 15, 2019).
- User Profile**: A profile picture and the name 'Kanza'.
- Actions**: 'Update' and 'Delete' buttons.
- Comments Section**: A section titled 'Comments' with an 'Add comment' button and a comment input field.

Figure 19:Update-Delete Post page

2.4.2. Add Comment :


As it's shown in the Figure 19, All users can comment on a post, however the comments made by an anonymous user won't be displayed unless it's approved by an Author or the

Admin. For example I will comment on a post as it is illustrated below (Figure 20),the Author block is shown only for anonymous users , because we need a name for anyone commenting .

ADD COMMENT

Author:

Such great information|

Text: 


Submit

Figure 20:Add Comment for Anonymous

After submitting the comment, it won't be shown below the post, in the comment section (Figure 21).

On a much larger scale, solar-thermal power plants employ various techniques to concentrate the sun's energy as a heat source. The heat is then used to boil water to drive a steam turbine that generates electricity in much the same fashion as coal and nuclear power plants, supplying electricity for thousands of people.

Published June 20, 2019
Modified June 20, 2019



Kanza

Comments

Add comment

really helpful

Anonymous-tester

June 20, 2019, 8:37 p.m.

Figure 21; Comment section after anonymous's comment

Later on, The author who posted the article or an Admin should log in and approve the comments submitted by anonymous users, and an anonymous first name will be used by default plus the name of the author chosen before (Figure 22).

On a much larger scale, solar-thermal power plants employ various techniques to concentrate the sun's energy as a heat source. The heat is then used to boil water to drive a steam turbine that generates electricity in much the same fashion as coal and nuclear power plants, supplying electricity for thousands of people.

Published June 20, 2019
Modified June 20, 2019



Kanza

Update

Delete

Comments

Add comment

its working

Kanza

June 20, 2019, 8:36 p.m.

Remove

Approve

really helpful

Anonymous-tester

June 20, 2019, 8:37 p.m.

Such great information

Anonymous-user

June 30, 2019, 9:54 a.m.

Remove

Approve

Figure 22: Comment Section for an Admin or the Author of the post

Finally, the author or the admin can approve the comment or remove it if it's inappropriate or not related to the subject.

2.4.3. Tags:

in the right corner we can search specific articles by choosing one of the listed tags as shown in Figure 23.

Tags:

- #solarenergy
- #windenergy
- #django
- #highvoltage
- #Sports
- #Cinema
- #E-mobility
- #Satellite
- #Wave energy

Pending accounts:

Roles requests:

Role **Reader** - author (Author)

Figure 23: Tags

2.4.4. Sort:

On the Left-hand side of the page you will notice the product sort button where of users, 'sort by oldest date' will revert the effect of 'sort by newest'.

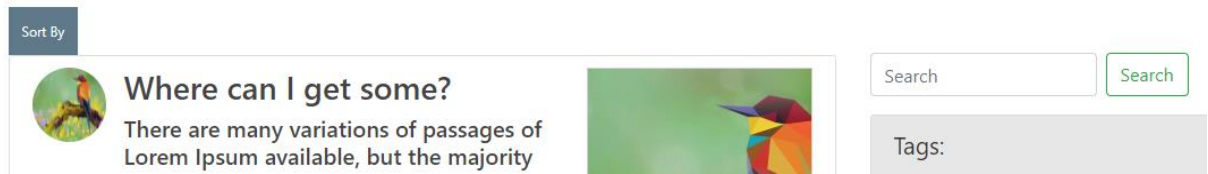


Figure 24:View of sort

2.4.5. Pagination:

After scrolling to the bottom of the page, you can see the latest articles by our Authors, you can browse to the older Articles by pressing on 'First >' or 'Last <' buttons at the end of the page and shows which page number user is currently viewing also user can switch to next article page or previous article page by clicking 'Previous>' or '<Next'.



Figure 25:View of pagination

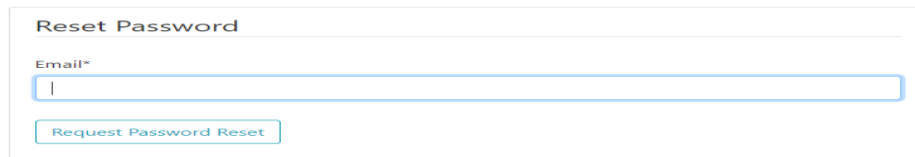
Chapitre 3:

Deviation from Plan

We have been able to achieve all requirements mentioned and set for us, therefore we added some other useful features to the blog to make it easier for the users.

3.1. Forgot password

This feature is made available for all the account types. At any time if you forgot your password then you can click on 'Forgot password' on the login page. By doing so, you will be redirected to one page where you can enter your account associated E-mail ID and submit it. You will receive an E-mail with password reset link. When you open that link, you will be redirected to the change password page. We suggest you make a new user account on our website with an actual working E-mail ID to test this feature.

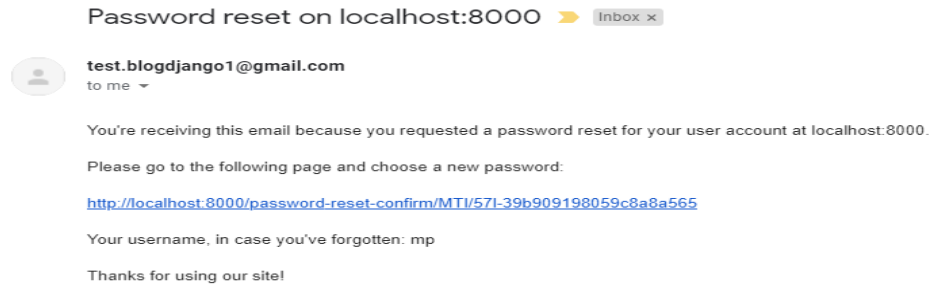


Reset Password

Email*

Request Password Reset

Figure 26:Reset request page



Password reset on localhost:8000 > Inbox x

test.blogdjangol@gmail.com
to me v

You're receiving this email because you requested a password reset for your user account at localhost:8000.

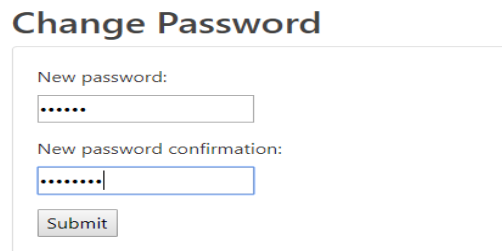
Please go to the following page and choose a new password:

<http://localhost:8000/password-reset-confirm/MTI/57I-39b909198059c8a8a565>

Your username, in case you've forgotten: mp

Thanks for using our site!

Figure 27:Reset request email sample



Change Password

New password:

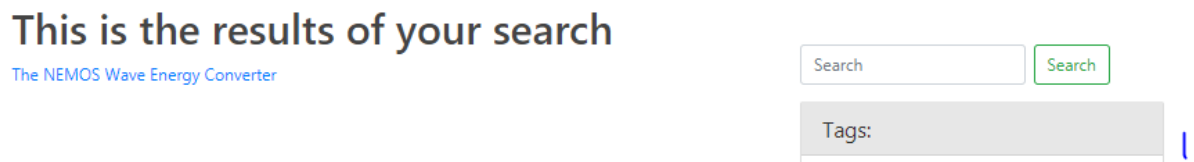
New password confirmation:

Submit

Figure 28:Change password

3.2. Search Button:

We added a search button to make it easy for users to look for posts by any word that exist in the title. In Figure 29 we get the results after searching for the word “wave”.



This is the results of your search

[The NEMOS Wave Energy Converter](#)

Search Search

Tags:

Figure 29:Results for search

3.3. Questions and Answers ‘Forum’:

This feature was for any question that can be asked about getting some ideas or even just users trying to help eachother to understand better about a certain subject.

How Wave Energy Converter works ? 3

Kanza: it works so well :D published June 24, 2019, 12:35 p.m.

author01: I don't know published June 25, 2019, 1:08 p.m.

Kanza: Test of an answer on question page published June 25, 2019, 1:30 p.m.

New Answer ..

Figure 30: Forum View

3.4. Checking the Approved or Changing role in database:

We used the following Figure 31 to handle changing roles and approving users in a simpler way, when we were working on that. After it helped us to check what type of a user, a profile has.

Change profile

User: ✎ +

Image: Currently: default.jpg
 Change: No file chosen

Type:

New type:

☒ Change role

☒ Approved user

Figure 31: Change Profile

Chapitre 4:

Code Documentation

For the creation of our blog, we use Django 2.2.2 which is a web development framework that contains set of libraries and tools for creating websites and web application. For coding

purpose, we used python as a programming language and Atom as the text editor. In django, there are major pieces that need to be dealt with: Settings, models, Forms, Views, Templates and Urls.

4.1. Settings:

This project is created using the command “python manage.py startproject tricalidee” and the blog applications are made using the command “python manage.py startapp blog”, where blog can be replaced with anything you wish to name your application. The file settings.py has code lines for the apps blog and users in INSTALLED-APPS.

```
INSTALLED_APPS = [  
    'blog.apps.BlogConfig', # Blog app  
    'users.apps.UsersConfig', # users app  
    'crispy_forms',  
    'rolepermissions',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'clear_cache',  
]
```

Figure 32:Settings.py / Installed-Apps

4.2. Models:

4.2.1. Models.py:

The objects that have been used for creating the blog are under the app ‘blog’ in the file models.py. A model in Django is an object which is used to save data in database. We have stored information of users, posts created by users and comments written by readers and uploaded images while creating blogs in database using models.py file. To access the file models.py, we need to go to tricalidee/blog/models.py.If ever changes were made in models.py a migration needs to be done in order to save the changes and run the new code, we use “python manage.py makemigrations” at first then “python manage.py migrate”

```

from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
from django.urls import reverse
from slugify import slugify

class Tag(models.Model):
    title = models.CharField(max_length=20)

    def __str__(self):
        return self.title

class Post(models.Model):
    Author=models.ForeignKey(User,on_delete=models.CASCADE)
    Title=models.CharField(max_length=120)
    Content=models.TextField()
    tags=models.ManyToManyField(Tag)
    time_of_publication=models.DateTimeField(default=timezone.now)
    time_of_modification=models.DateTimeField(auto_now=True,auto_now_add=False)
    thumb = models.ImageField(default='default.jpg',blank=True)

    def __str__(self):
        return self.Title

    def get_absolute_url(self):
        return reverse('post-detail', kwargs={'pk': self.pk})

    def snippet(self):
        return self.body[:50]

```

Figure 33:models

For each class, we have defined various types of fields and also have returned some value. E.g.: Author=models.ForeignKey(User,on_delete=models.CASCADE).

```

def __str__(self):
    return self.Title # returns title of the post

```

Figure 34:Code for return the title of the post

Here, the ‘__str__’ function is used to return the title of the post.

4.2.2. Class Diagram:

For the database, we followed the diagram below (Figure 35), to work on the interactions between the classes used for our blog “Tricalidee”. For example , A user can post multiple articles, but one article can be associated to one User only. The same for questions, it can be asked by one user and it can have several responses, and each response associated to Only One User.

In this diagram we didn’t mention the four types of users, this diagram will be fully applicable for An Admin and an Author.

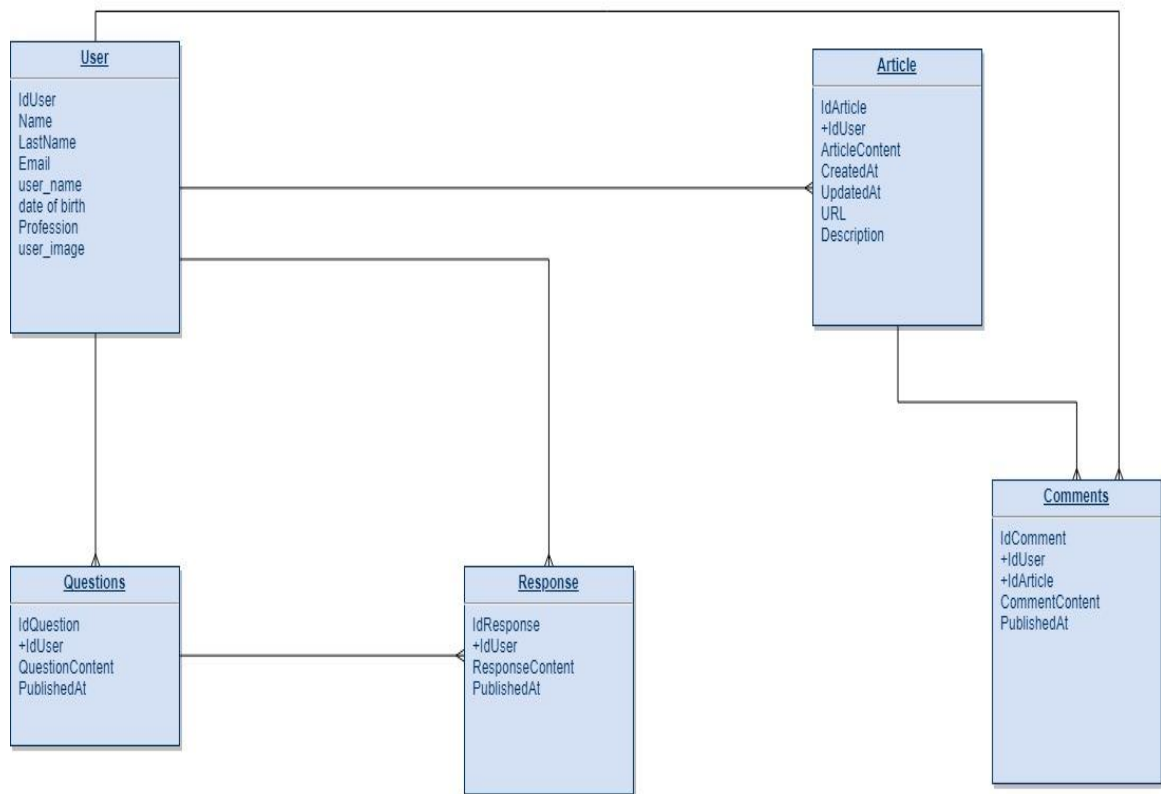


Figure 35: Class Diagram

4.3. Forms:

We have created forms.py under our app accounts to define Django's form class. We created 'Forum' and 'Add comment' functionality using form class. If we open tricalidee/blog/forms.py it will look like:

```

from django import forms
from .models import Comment, Answer

class CommentForm(forms.ModelForm):

    class Meta:
        model = Comment
        fields = ('text',)

        widgets = {
            #'author':forms.TextInput(attrs={'class':'textinputclass'}),
            'text':forms.Textarea(attrs={'class':'editable medium-editor-textarea'}),
        }

class AnonymousCommentForm(forms.ModelForm):

    class Meta:
        model = Comment
        fields = ('author','text')

        widgets = {
            'author':forms.TextInput(attrs={'class':'textinputclass'}),
            'text':forms.Textarea(attrs={'class':'editable medium-editor-textarea'}),
        }

class AnswerForm(forms.Form):
    answer_content = forms.CharField(label='',max_length=200,widget=forms.TextInput(attrs={'class': 'form-control'}))
    def clean(self):
        cleaned_data = super(AnswerForm, self).clean()
        return cleaned_data

```

Figure 36:Form

Fields which we have defined under fields class are available on webpage to user to enter data. Entered data will be saved in database after submitting corresponding form by End-user.

4.4. Views:

We have created views.py file under our application. It will request information from the created model and pass it to template. We have imported models, created forms and other necessary imports in views.py file. We created various functions using function (def) that takes request and return a function render which will render (put together) our mentioned template as follow:

```

def post_detail(request, id, slug):
    post = get_object_or_404(Post, id=id, slug=slug)
    is_liked = False
    if post.likes.filter(id=request.user.id).exists():
        is_liked = True
        context = {
            'post' : post,
            'is_liked' : is_liked,
            'total_likes' : post.total_likes(),
            'tags' : Tag.objects.all(),
        }
    return render(request, 'blog/post_detail.html', context)

```

Figure 37:Views.py

We also added ‘.order_by(‘-date’)’ under views.py to display the posts based on the date of publish of the post:

```
def get_query_set(self):
    user = get_object_or_404(User, username=self.kwargs.get('username'))
    return Post.objects.filter(author=user).order_by('-date_posted')
```

Figure 38: Codes of order by date

It will get data from model class and display it under template ‘blog/post_detail.html’ by writing code as follow:

```
<div class="bodyarticle">
    
    <p></p>
    <p class="article-content">{{ post.Content }}</p>
</div>
<div class="article-author article article-metadata">
    
    <a class="mr-2" href="#">{{ post.Author }}</a>

</div>
<div class="dates">
    <div>
        <small class="text-muted">Published {{ post.time_of_publication|date:"F d, Y" }}</small><br>
        <small class="text-muted">Modified {{ post.time_of_modification|date:"F d, Y" }}</small>
    </div>
</div>
</div>
<hr>
{% if post.Author == user %}
<div>
    <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{% url 'post-update' post.id %}">Update</a>
    <a class="btn btn-danger btn-sm mt-1 mb-1" href="{% url 'post-delete' post.id %}">Delete</a>
</div>
<hr>
{% endif %}
```

Figure 39 : Html codes of blog post

On the website, about page looks like follows with the photo of our team members and few words about our motivation:

About Us

Every student goes to google for new project ideas, be it the final step of graduating out of a college or just to improve understanding of the subject in their leisure time. Well, we think that we might have solved their problem.

Tricalidee is a site which allows such students to browse through the projects submitted and also avoid the pitfalls their predecessor fell into while doing the respective project. The blog also enables users to post new ideas so many future generations can benefit from them.

Figure 40: About us Page

OUR TEAM



Figure 41: Our Team Page

4.5. Templates

Django template tags allow us to transfer Python-like things into HTML, so you can build dynamic websites faster. All the HTML files required for our projects are placed under the folder ‘templates’. We have used following HTML files for developing our Blog:

- **Base.html:** This has the basic layout of the HTML pages, e.g.: Navbar which is same for all the HTML pages, search bar, etc.
- **Home.html:** The base html file is extended, and some additional things are added. The home page contains the list of all posts sorted by Newest to Oldest. The sorting criteria can be changed by selecting appropriate sorting criteria. Also, the home page contains a list of all the ‘Tags’ which when clicked displays the list of posts sorted according to the tags. About, Admin and Contact Us pages can be navigated from the homepage (Figure 8).
- **Post_detail.html:** Once a post is clicked, this html page displays the article along with the comment section. The comment must be approved either by the author or the admin in order for any user to see the comment.

To print a variable in Django templates, we used double curly brackets with the variable's name inside. It will display data fetched from database of that variable. Other conditional lines like for and if statements are used to add conditions to display blogs on webpage. <header>, <style>, <footer>, <script>, <body> such HTML tags are used in HTML files to define layout and behaviour of webpage.

```

</div>
{% if user.is_authenticated and user.profile.approved user %}
<form method='POST' enctype='multipart/form-data' action="{% url 'add_comment_to_post' post.id %}">
<div class="modal-body">
<div class="content-section">
{% csrf_token %}
{{ form.as_p }}
</div>
</div>
<div class="modal-footer">
<button type="submit" class="btn btn-primary">Submit</button>
</div>
</form>
{% else %}
<form method='POST' enctype='multipart/form-data' action="{% url 'add_anonymous_comment_to_post' post.id %}">
<div class="modal-body">
<div class="content-section">
{% csrf_token %}
{{ anonym_form.as_p }}
</div>
</div>
<div class="modal-footer">
<button type="submit" class="btn btn-primary">Submit</button>
</div>
</form>
{% endif %}
</div>
</div>

```

The CSRF token is used to provide easy protection Cross Site Request Forgery. To defend from such attacks, GET requests and other ‘safe’ methods are used instead of ‘unsafe’ methods such as POST, PUT, and DELETE.

Style and Bootstrap links are specified in the base.html file itself in the head section:

```

<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3"
<link rel="stylesheet" type="text/css" href="{% static 'blog/main.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'blog/StyleAboutus.css' %}">
{% if title %}
<title>Tricalidee - {{ title }}</title>
{% else %}
<title>Tricalidee</title>
{% endif %}

```

3.1.1.1 3.6 URLS

The url patterns created for our website are defined in the urls.py file along with the necessary imports (like importing views, settings, etc.) mentioned at the top of the url patterns. If we open blog/urls.py, it looks like:

```

urlpatterns = [
    path('', views.home, name='blog-home'),
    path('newest', views.newest, name='newest'),
    path('oldest', views.oldest, name='oldest'),
    path('new_answer', views.new_answer, name='new_answer'),
    path('get_posts_by_tag/<str:title>', views.get_posts_by_tag, name='get_posts_by_tag'),
    path('user/<str:username>', UserPostListView.as_view(), name='user-posts'),
    path('profile/', user_views.profile, name='profile'),
    path('post/<int:pk>', views.displayPost, name='post-detail'),
    path('myPosts/<int:id_user>', views.list_myPosts, name='my_posts'),
    path('post/new/', PostCreateView.as_view(), name='post-create'),
    path('question/', views.question, name='questions'),
    path('post/<int:pk>/update/', PostUpdateView.as_view(), name='post-update'),
    path('post/<int:pk>/delete/', PostDeleteView.as_view(), name='post-delete'),
    path('register/', user_views.register, name='register'),
    path('about/', views.about, name='blog-about'),
    path('post/<int:pk>/comment/', views.add_comment_to_post, name='add_comment_to_post'),
    path('post/<int:pk>/anonymous_comment/', views.add_anonymous_comment_to_post, name='add_anonymous_comment_to_post'),
    path('post/<int:pk>/approve/', views.comment_approve, name='comment_approve'),
    path('post/<int:pk>/remove/', views.comment_remove, name='comment_remove'),
    path('profileUser/<int:pk>', views.profile_remove, name='profile_remove'),
    path('approveprofile/<int:pk>', views.profile_approve, name='profile_approve'),
    path('approverole/<int:pk>', views.role_approve, name='role_approve'),
    path('removerole/<int:pk>', views.role_remove, name='role_remove'),
    path('search/', views.search, name='search'),
    path('contact/', views.contact, name='Contact')
]

```

Figure 42:Urls

4.6. Tests :

We used Unit test as a tool for testing our code, it checks if all specific parts of our function's behavior are correct, which will make integrating them together with other parts much easier.

Test case is a collection of unit tests which together proves that a function works as intended, inside a full range of situations in which that function may find itself and that it's expected to handle. Test case should consider all possible kinds of input a function could receive from users, and therefore should include tests to represent each of these situations.

```

test_views.py  x  test_urls.py  test_forms.py
1  from django.test import SimpleTestCase
2  from blog.forms import (
3      CommentForm,
4      AnonymousCommentForm,
5      AnswerForm)
6
7
8
9  class TestForms(SimpleTestCase):
10
11      def test_CommentForm_valid_data(self):
12          form=CommentForm(data={
13              'text':'Good Article'
14          })
15
16          self.assertTrue(form.is_valid())
17
18      def test_CommentForm_no_data(self):
19          form=CommentForm(data={})
20
21          self.assertFalse(form.is_valid())
22          self.assertEqual(len(form.errors),3)

```

Figure 43:Test_Forms

Chapitre 5:

Usage Documentation

We got help from many sources, one of them were the professors lectures, which introduced us to new softwares and web framework Django, the most source used was : <https://docs.djangoproject.com/en/2.2/>(Figure 44),and <https://books.agiliq.com/projects/django-orm-cookbook/en/latest/index.html>, also Stack overflow for any errors we came across with.

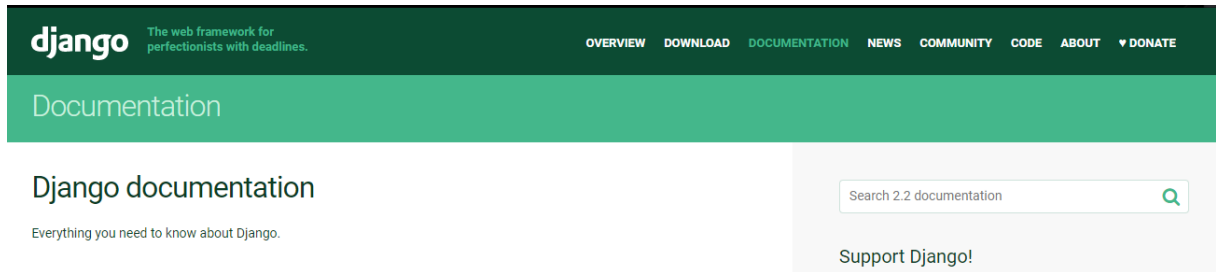


Figure 44:Django Documentation

We also went for different tutorials on YouTube to get familiar with the coding using Python, html, and Css. Also Wikipedia of course, for any definition and further knowledge on the history of the softwares.

Conclusion

Creating this Blog was such an enriching experience, we got to test our abilities in coding, and to have the right set of algorithms, it was an opportunity to learn other programming languages, most importantly python which is a widely used general-purpose, dynamic, extensible, high-level programming language. We learned how to use Django as a web Framework, which was bit challenging for us at first, but we got used to it once we understood how its working and realized how friendly the application is.

Tricalidee is a blog for sharing ideas about electrical matters, and it goes beyond that, people can get the help they want from sharing their articles.

For improvements of the blog, we can suggest to add a browsing history that a user can go back to, to get back to any read article, also add a View Later button to save all the articles that a user find interesting when it wasn't possible to check them. Follow an author can be attached to subscribe to any user of someone's liking, also displaying the users that are more active in the blog can be joined, as well as checking a user's profile.

Finally, we would like to thank our Professor Fischer Andreas, for all the help we got from him when it was needed and the detailed explaining and the suggestion of the tools that can be used to succeed to make the creation of a blog handable and fun.