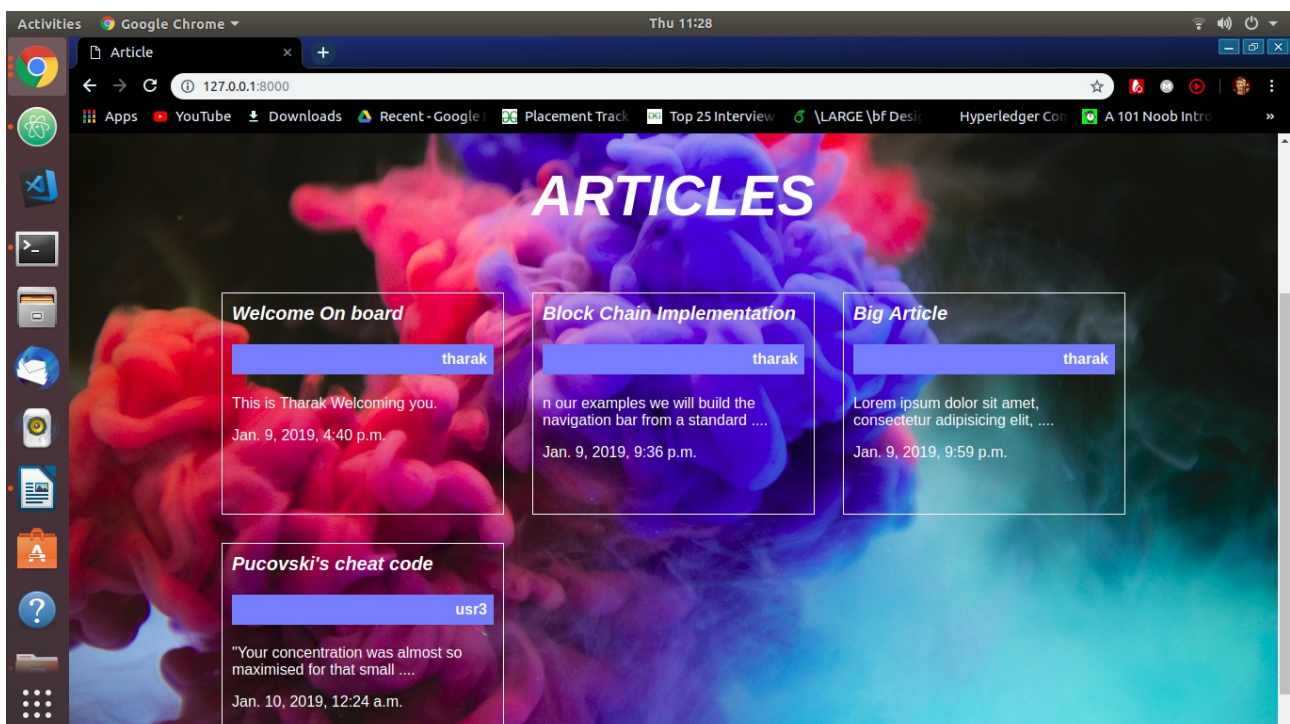


BLOG CREATION

I have used Python to configure my web application with the help of django framework. I have followed a modular approach, dividing the entire application into three modules,

1. Blog - Main Application
2. Article - Articles and thier modelling
3. Users - Authentication

The website looks as follows,



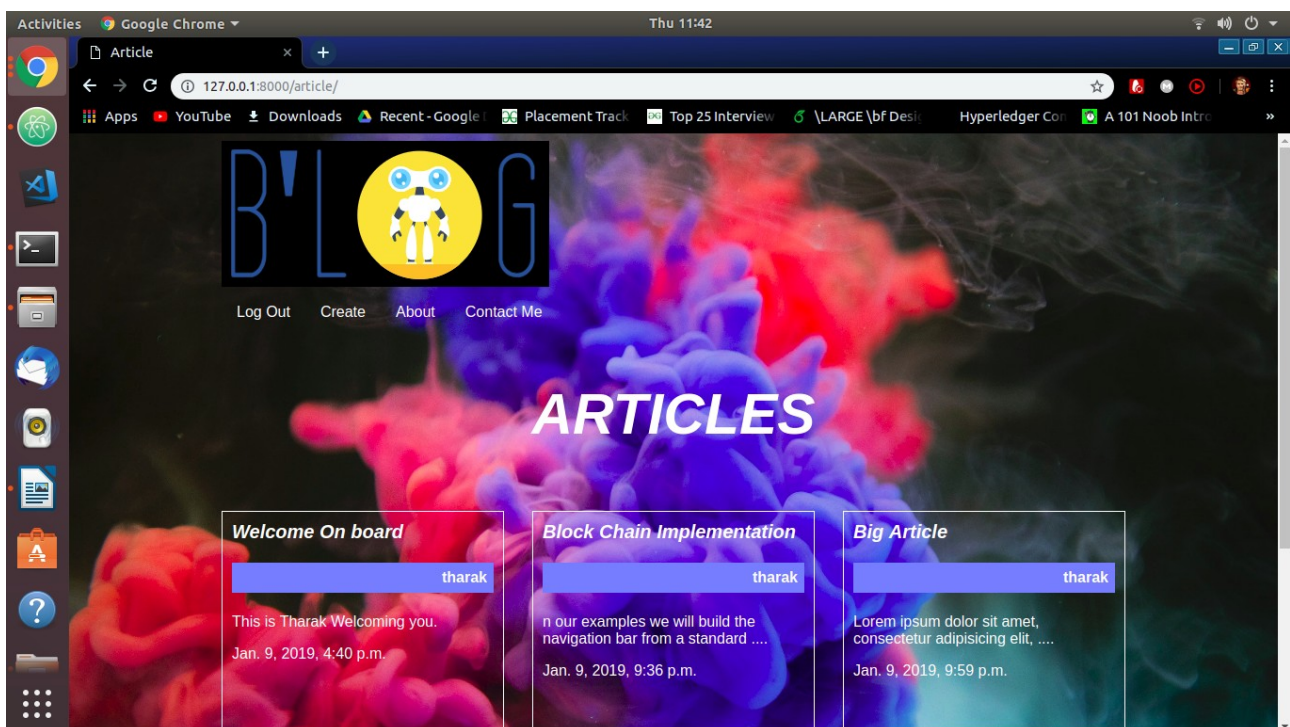
On clicking on the logo the page is redirected to the article section. I have taken this articleDetails.html page (which show cases all the articles in the database) as my home page.

Under the logo there is a dynamic nagigation bar, which updates its contents based on whether any user is logged in or not.

When any user is not logged in, the page appears as follows



When any user is logged in it changes its state to,



A Provision is made for the users to know about the developers information in the contact section.

Now coming to the details of the article app, a model form of an article is created as follows,

<http://root/article> will be the home path of this article app, root = 127.0.0.1:8000

When an app is created, it generally comes with views.py, models.py, urls.py files created. If not we have to manually add them since they control the responses we make on the front end.

The urls which we enter should be declared in the urls.py file and corresponding to each url there will be a function (this will be declared in the views.py file).

Eg: path('new/', views.create, name = 'create'), This means that we are declaring <http://root/article/new> url in the urls.py file and when this url comes in the browser it makes a GET request to the function create in views.py file.

In the views.py file, rendering, redirecting and saving the information internally to the database are carried out by the functions render and redirect present in django.shortcuts.

Article form is modelled as such,

```
class Article(models.Model):
    title = models.CharField(max_length = 100)
    slug = models.SlugField()
    body = models.TextField()
    date = models.DateTimeField(auto_now_add=True)
    thumbnail = models.ImageField(blank=True)
    author = models.ForeignKey(User, on_delete=models.DO_NOTHING, default=None)
```

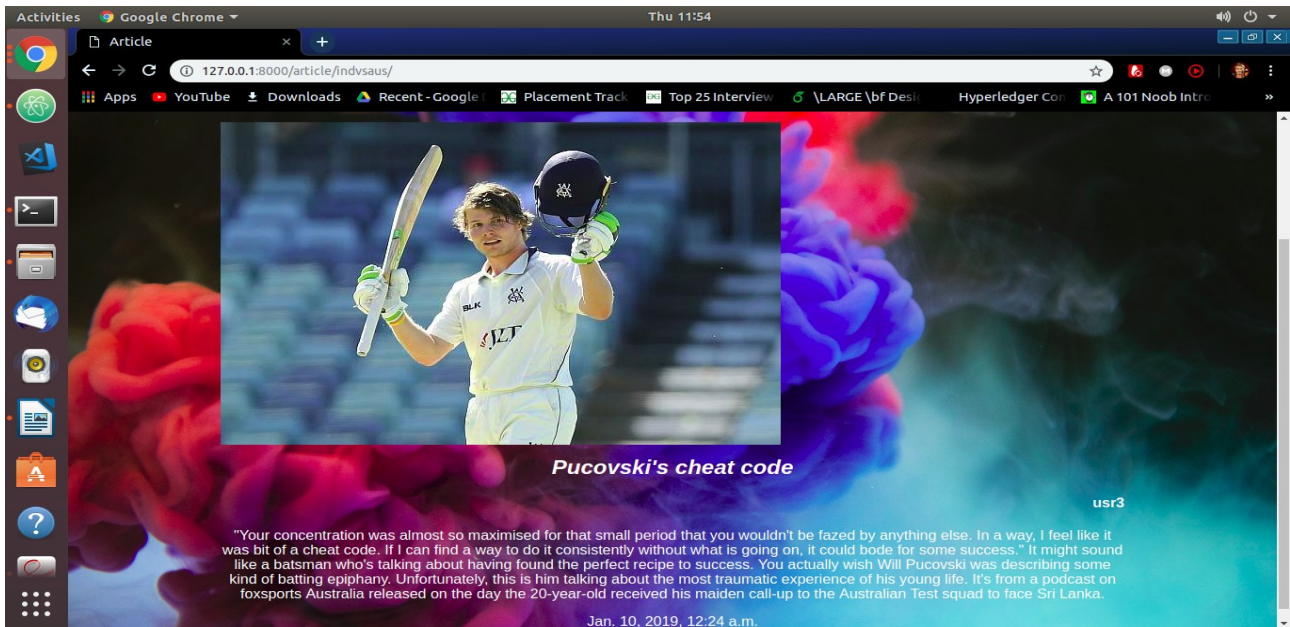
So every article will be having title, slug, body, date of creation, thumbnail and author sections, in which date and author will be automatically updated since only the logged in users are permitted to create an article.

For every article only a glimpse of it is shown in the articleList section, in order to know about the complete information about it one should click on that. On clicking it, one will be directed to the articleDetails.html where the complete details of the article are available.

Function short is used to give the glimpse of the article in articleList page.

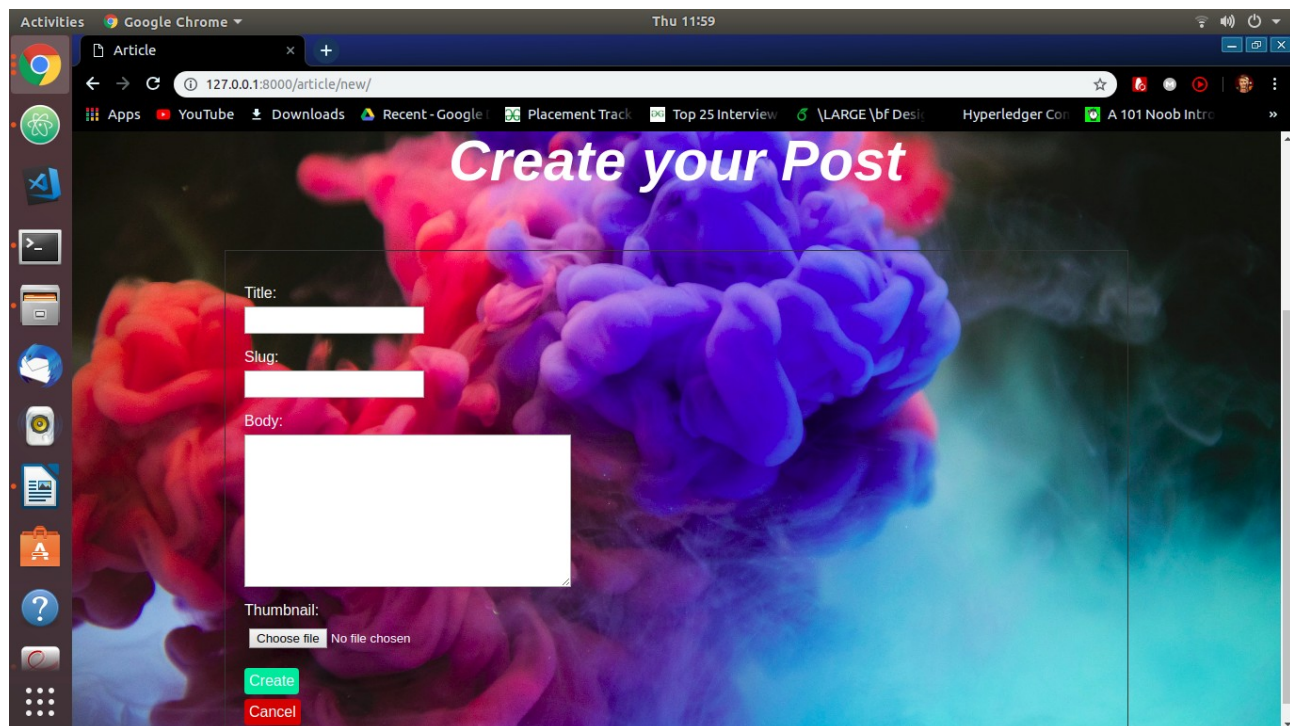
```
def short():
    if len(body) <= 50:
        return body
    else:
        a = body.split(' ')
        st = ""
        l = 0
        while l < 50:
            l = l + len(a[0])
            st = st + a[0] + ' '
            a.remove(a[0])
        else:
            return st + '....'
```

On clicking Pucovski cheat code,



Sqlite database is used by django to store all this information.

Only logged in users can create a new article,



After entering the data and upon clicking the submit button the users will be redirected to the articleList.html page showcasing the updated list of the articles present in the database.

There are 2 types of requests,

1. GET : In order to get a HTML template.
2. POST : After entering data in the form we send this data back to server this request.

So for a function which is triggered for both POST and GET requests we have to mention the conditions,

```
@login_required(login_url='users:login')
def create(request):
    if request.method == 'POST':
        form = forms.CreateArticle(request.POST, request.FILES)
        if form.is_valid():
            temp_form = form.save(commit=False)
            temp_form.author = request.user
            temp_form.save()
            return redirect('article:arlist')
    else:
        form = forms.CreateArticle()
    return render(request, 'article/createArticle.html', {'form':form})
```

We make sure that logging in is required to create a new article by adding a decorator on top of create function.

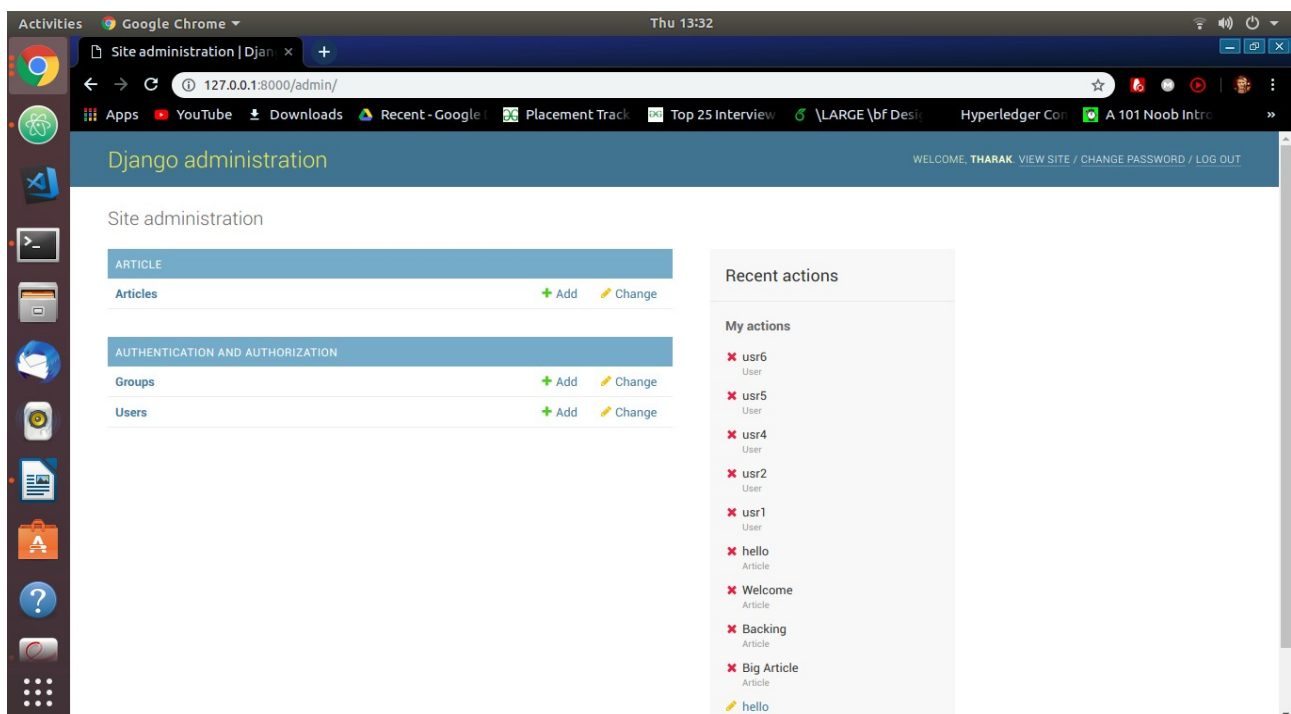
Now coming to the users app,

User Authentication function is handled by this app. Django has inbuilt AuthenticationForm and UserCreationForm in django.contrib.auth.forms for user authentication purpose and for new user creation.

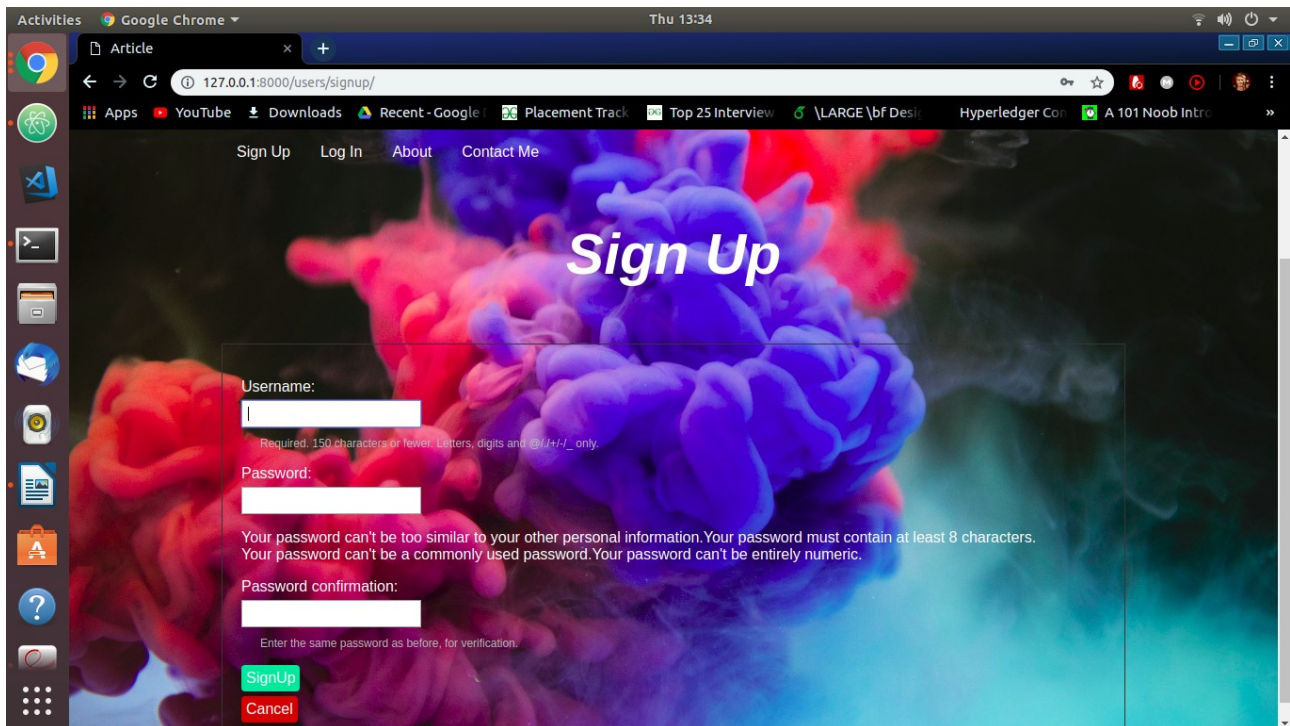
Apart from the normal users, there is another user type called superuser, which can only be created before running the server by entering the command,

```
python3 manage.py createsuperuser --user
```

After creating the superuser, open <http://root/admin> to login with the superuser credentials, While logging under the super user we will be having access to the complete data and we can add, delete and modify the articles and users.



SignUp Form:



The screenshot shows a web browser window with the URL `127.0.0.1:8000/users/signup/`. The page has a dark background with a colorful, abstract smoke-like pattern in shades of red, orange, and blue. At the top, there is a navigation bar with links: [Sign Up](#), [Log In](#), [About](#), and [Contact Me](#). The main heading

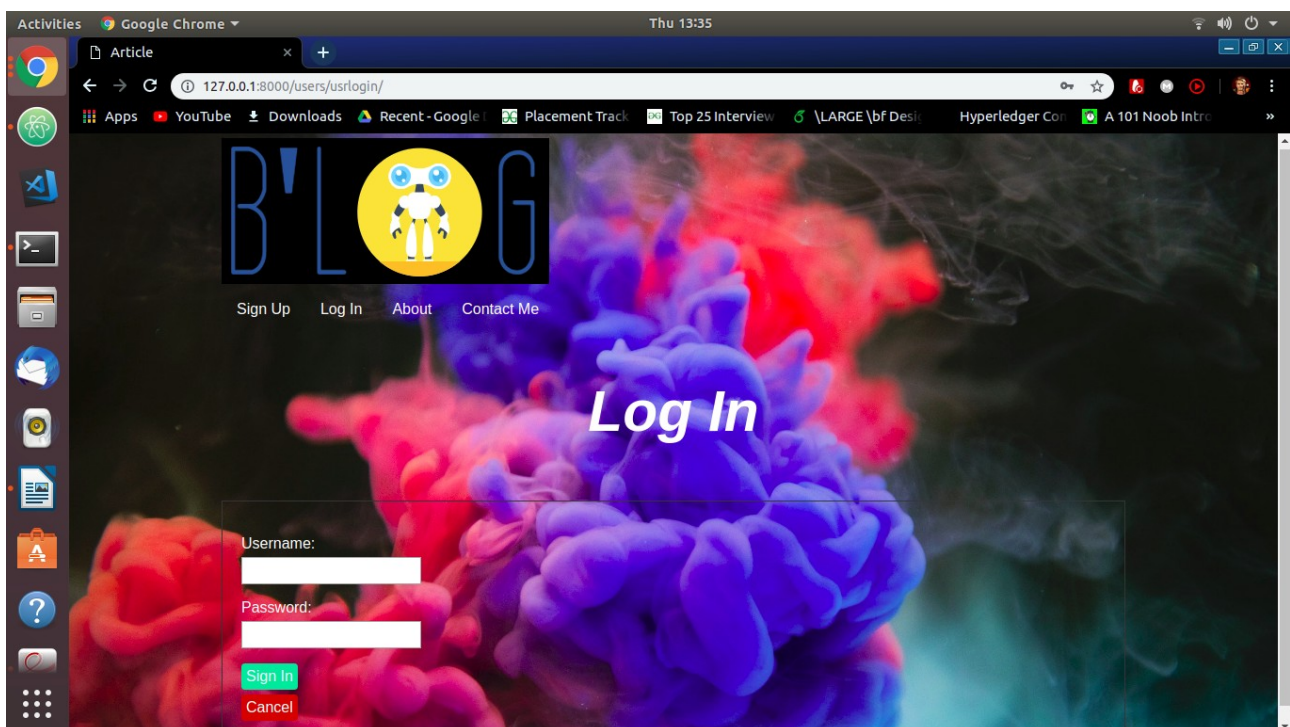
Sign Up

 is centered. Below it, there is a form with the following fields and labels:

- Username:** A text input field. Below it, a small note says: "Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only."
- Password:** A text input field.
- Password confirmation:** A text input field. Below it, a small note says: "Enter the same password as before, for verification."

At the bottom of the form, there are two buttons: a green **SignUp** button and a red **Cancel** button.

Log In Form:



The screenshot shows a web browser window with the URL `127.0.0.1:8000/users/login/`. The page has a dark background with a colorful, abstract smoke-like pattern in shades of red, orange, and blue. At the top, there is a navigation bar with links: [Sign Up](#), [Log In](#), [About](#), and [Contact Me](#). The main heading

Log In

 is centered. Below it, there is a form with the following fields and labels:

- Username:** A text input field.
- Password:** A text input field.

At the bottom of the form, there are two buttons: a green **Sign In** button and a red **Cancel** button.

Sqlite database is used to store the users data.

All the html templates are derived from base.html template.

Also added the facility of directly interacting with the designer or the admin of the site through already established social networking sites as facebook, twitter and LinkedIn.

