



**Dharmsinh Desai University, Nadiad**

**Faculty of Technology**

**Department of Computer Engineering**

**B. Tech. CE Semester-IV**

**Subject: Software Project**

**Project title: BlogApp**

**By:**

1)Sagar Vala, Roll no: CE146, Id: 19CEUOG081

2)Prince Vanani, Roll no: CE147, Id: 19CEUES108

3) Meet Vaishnani, Roll no: CE145, Id: 19CEUEG165

**Guided by: Prof. Jigar Pandya & Prof. Brijesh Bhatt**

## **Contents: -**

<b>1. Abstract.....</b>	<b>3</b>
<b>2.Introduction.....</b>	<b>3</b>
<b>3.Software Requirement Specification.....</b>	<b>4</b>
<b>4.Design.....</b>	<b>7</b>
Use-case-diagram.....	7
Class-diagram.....	8
Data-Dictionary.....	9
<b>5.Implementation Detail.....</b>	<b>10</b>
Modules.....	10
Function Prototype.....	11
<b>6.Screen-Shots.....</b>	<b>13</b>
<b>7.Conclusion.....</b>	<b>17</b>
<b>8.Limitation.....</b>	<b>17</b>
<b>9.Future extension.....</b>	<b>18</b>
<b>10.Bibliography.....</b>	<b>18</b>

## **1.) Abstract**

**-blog app is simple interface for maintenance of user**

**blog post.it be used for anything that involves communicating or**

**publishing information on the World Wide Web. Common uses include**

**teaching and educational and corporate use. Your blog can be a personal diary,**

**a project collaboration tool, a guide, or any means of communicating and**

**publishing information on the web**

## **2.) Introduction**

- In blog app user can easily read other people blog who are connected to this app. If user wants to write blog first, he/she need to register. Registration is done by the system admin for security reasons.After registration,the user is assigned username,password with which he/she can use to log into the system and enjoy services provided by the system such as Post blog or see other users blog.if user delete other user post then user cannot delete, update other user post.**

### **3.) Software Requirement Specification**

#### **Functional Requirements: -**

##### **1.User**

###### **R.1.1 sign up**

Description: if student is new for this system then first need sign-up for log-in.

input: student fill all information.

output: successfully signup message.

###### **R.1.2 login**

Description: Login can be done if student's information is stored in the system.

input: enter login detail username and password

output: Successfully login

status: failure if invalid detail entered.

###### **R.1.3 Forgot password**

Description: if any student forgot/change password.

input: enter new password.

output: successfully change password.

###### **R.1.4 Update User profile**

Description: if any student wants to update his/her profile.

input: enter new profile details.

output: change has been successfully stored

## 2.Post

### R.2.1 Add Post

Description: if user wants add new post

input: fill POST details

output: Successfully added post.

### R.2.2 Delete Post

Description: if user wants delete his/her post

input: click on delete post

output: Successfully deleted post

status: failure if user wants to delete other users's post.

### R.2.3 Update Post

Description: if any user wants to update his/her post

input: enter updated post details.

output: successfully updated.

status: failure if user wants to update other users's post.

### R.2.4 Display/See Post

Description: if any student wants to see all user's post.

input: display post

output: display all user's post

## 3.Admin

Description: in this module system admin provide different accessibility for Post and user. Admin also can add, update, delete any user if needed. Admin also can add, update, delete any user's POST if needed. Admin also can update, delete any user's profile if needed.

### R.3.1 Add user or user's post or user's profile

Description: if admin wants to add user, user's post and user's profile directly so admin can add.

Input: enter detail of record

Output: successfully added message

status: failed if record already exist.

### R.3.2 Update user or user's post or user's profile

Description: if admin wants to update user, user's post and user's profile directly so admin can update.

Input: enter update detail of record

Output: successfully updated message

status: failed if record not exist.

### R.3.3 Delete user or user's post or user's profile

Description: if admin wants to delete user, user's post and user's profile directly so admin can delete.

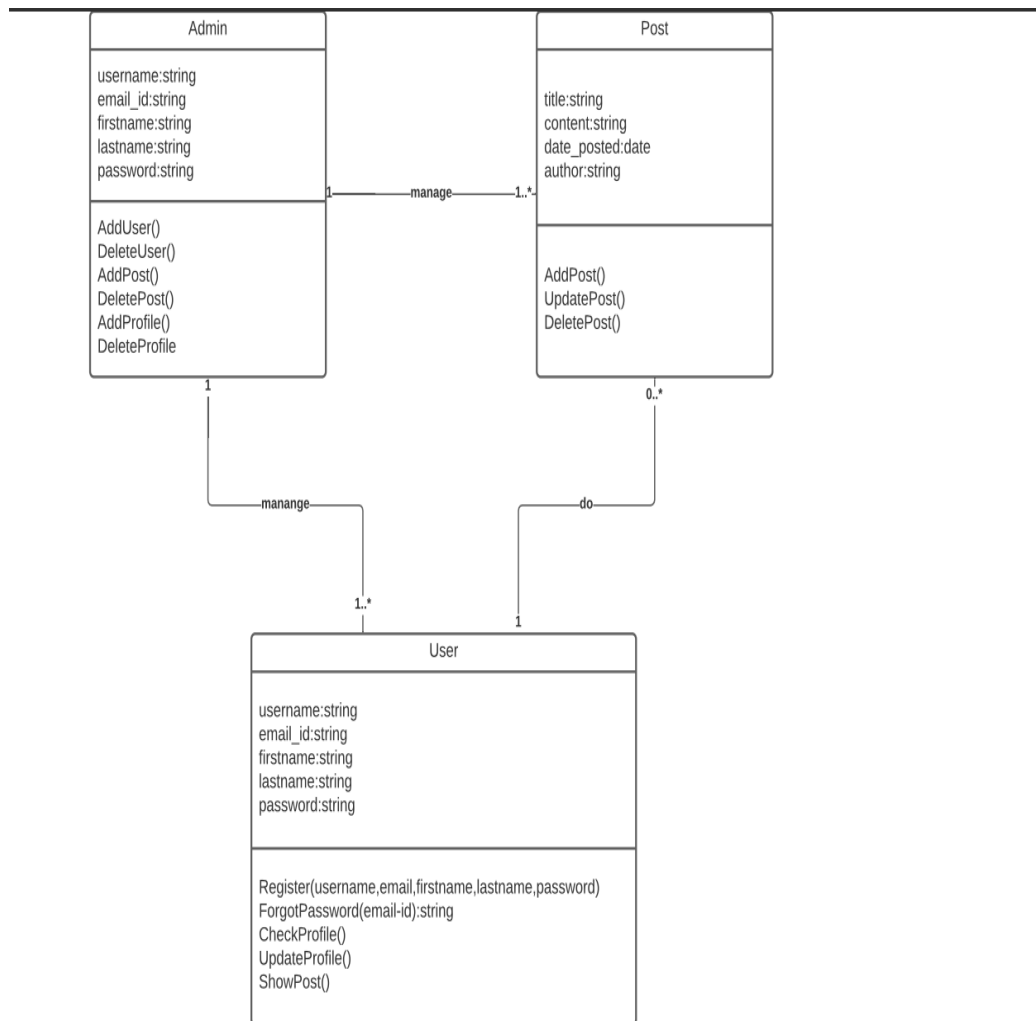
Input: enter detail of record

Output: successfully deleted message

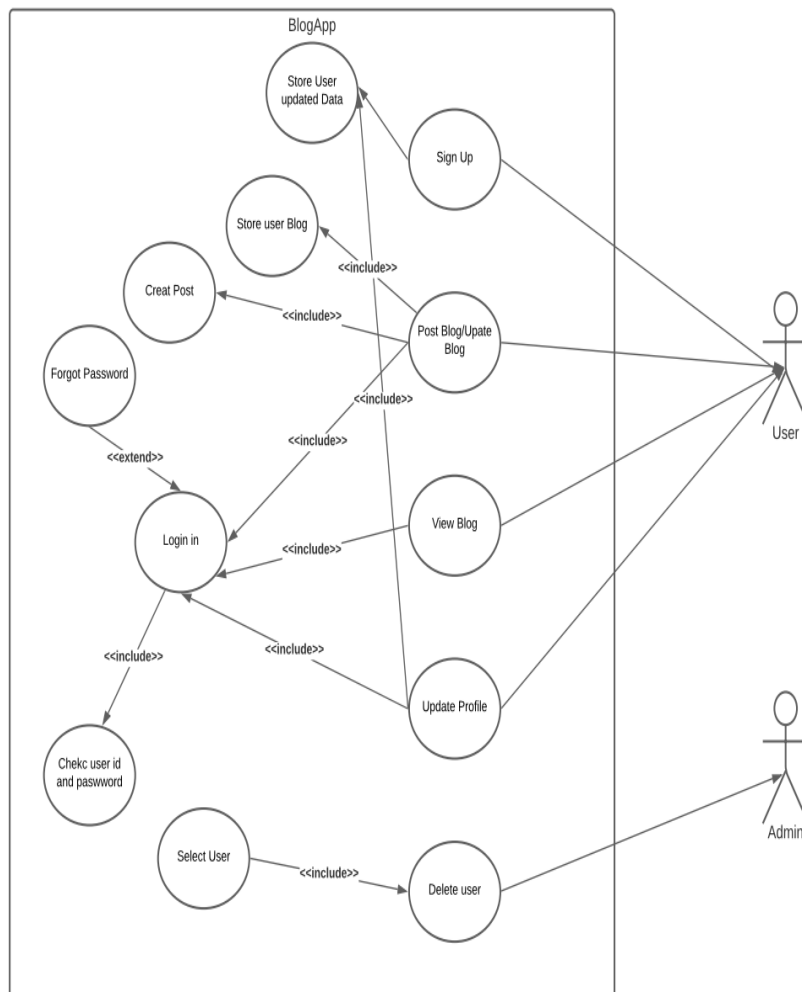
status: failed if record not found.

## 4.) Design

### Class diagram:



## Use-case diagram:





## Models:

### User's Profile:

```
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg', upload_to='profile_pics')

    def __str__(self):
        return f'{self.user.username} Profile'

    def save(self, *args, **kwargs):
        super(Profile, self).save(*args, **kwargs)

        img = Image.open(self.image.path)

        if img.height > 300 or img.width > 300:
            output_size = (300, 300)
            img.thumbnail(output_size)
            img.save(self.image.path)
```

### User's Post:

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

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    date_posted = models.DateTimeField(default=timezone.now)
    author = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.title

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

# Create your models here.
```

## **5.) Implementation**

### **Module:**

**-In This Software we mainly use two modules.**

**1) User Module**

**2) Post Module**

**Each module consists of several methods to implement the required functionality. Implementation is done using Django. Database used in these module Sqlite3.**

**1)Register User Module**

**Basic information of user is taken by system and stored in database.**

**2) Login User Module**

**Users are able to login themselves. System logs user in, then and only then user can use other functionalities of system.**

**3) Post Module**

**Users are able to post something .if users is already signup and login . After Post something user can delete or update their own post.**

## Function Prototype:

### 1. Register, Profile for user view

```
# Create your views here.
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'Your account successfully created ! Now you can able to log in!')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=request.user)
        p_form = ProfileUpdateForm(request.POST,
                                   request.FILES,
                                   instance=request.user.profile)
        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, f'Your account has been updated!')
            return redirect('profile')
    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form
    }

    return render(request, 'users/profile.html', context)
```

## 2. Post Module view

```
class PostDetailView(DetailView):
    model = Post

class PostCreateView(LoginRequiredMixin, CreateView):
    model = Post
    fields = ['title', 'content']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

class PostUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    model = Post
    fields = ['title', 'content']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

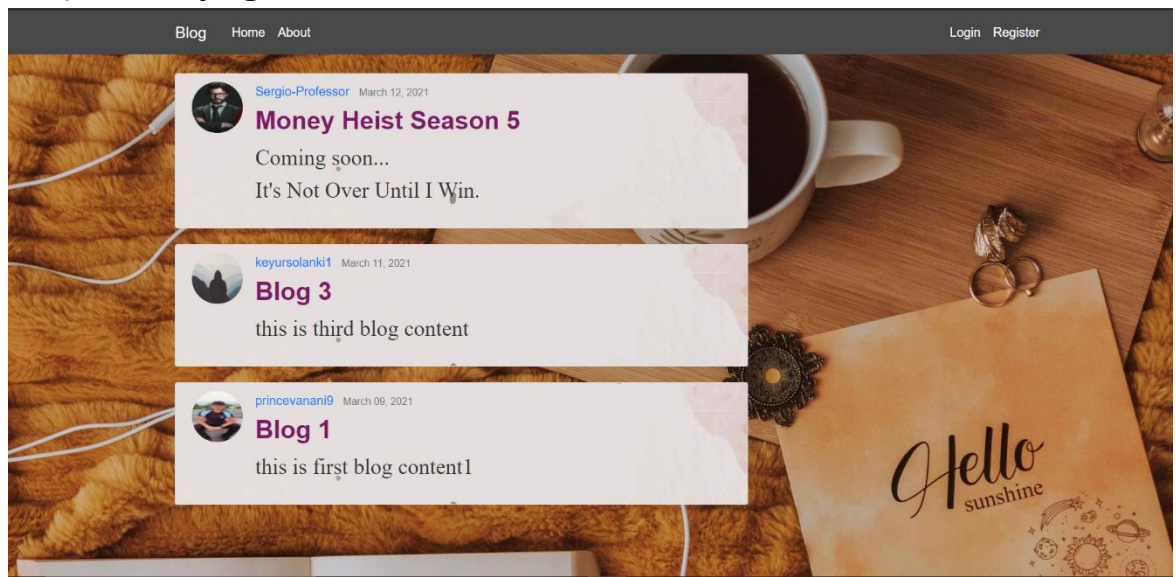
    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False

class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Post
    success_url = '/'

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False
```

## 6.) Screenshot

### 6.1) home page



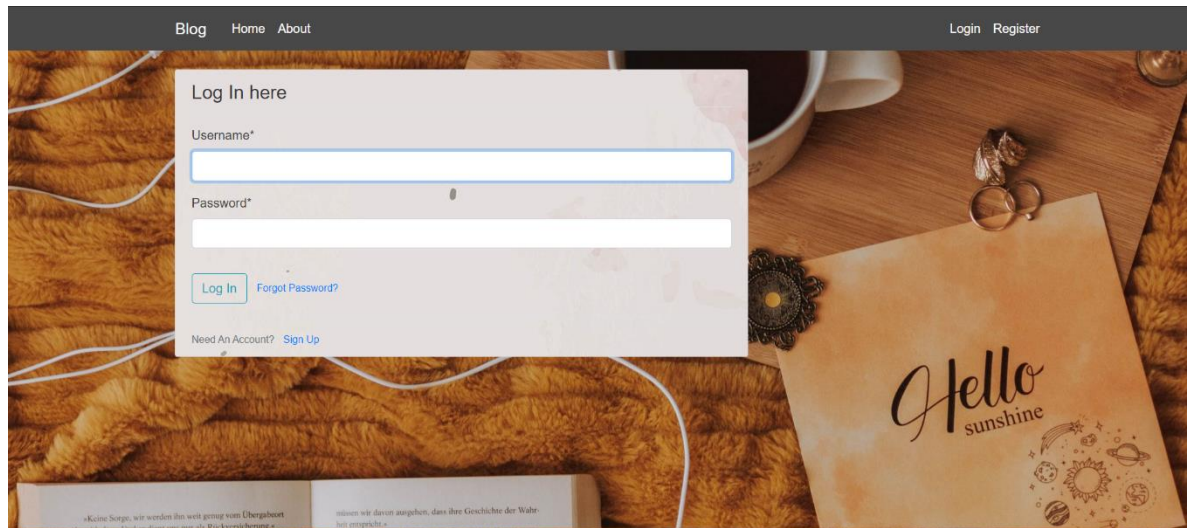
### 6.2) Register page

The screenshot shows the register page of the blog. The background is the same warm-toned image as the home page. The registration form is titled "For New Users" and includes the following fields and instructions:

- Username\***: A text input field. Below it, a note states: "Required: 150 characters or fewer. Letters, digits and @/./\_/- only".
- Email\***: A text input field.
- Password\***: A text input field. Below it, a list of password requirements is provided:
  - 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. Below it, a note states: "Enter the same password as before, for verification."

A "Sign Up" button is located at the bottom of the form.

## 6.3) Login page

A screenshot of a login page with a light gray background and a white login form. The form is titled "Log In here" and contains fields for "Username\*" and "Password\*". Below the password field is a "Log In" button and a "Forgot Password?" link. At the bottom of the form is a "Need An Account? Sign Up" link. The background of the page features a warm, textured orange-brown surface, a white mug, a wooden cutting board, and a card that says "Hello sunshine" with a sun and planets illustration. The top navigation bar is dark gray with links for "Blog", "Home", "About", "Login", and "Register".

Blog Home About Login Register

Log In here

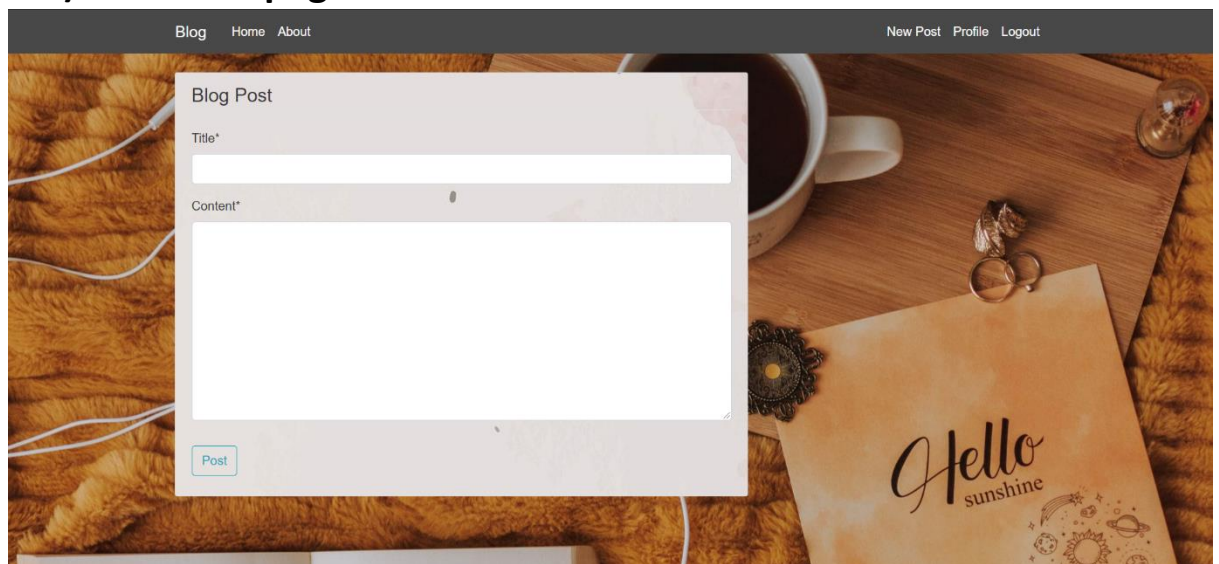
Username\*

Password\*

Log In Forgot Password?

Need An Account? Sign Up

## 6.4) New Post page

A screenshot of a "New Post" page with a light gray background and a white form. The form is titled "Blog Post" and contains fields for "Title\*" and "Content\*". Below the content field is a "Post" button. The background of the page is the same as the login page, featuring a warm, textured orange-brown surface, a white mug, a wooden cutting board, and a card that says "Hello sunshine" with a sun and planets illustration. The top navigation bar is dark gray with links for "Blog", "Home", "About", "New Post", "Profile", and "Logout".

Blog Home About New Post Profile Logout

Blog Post

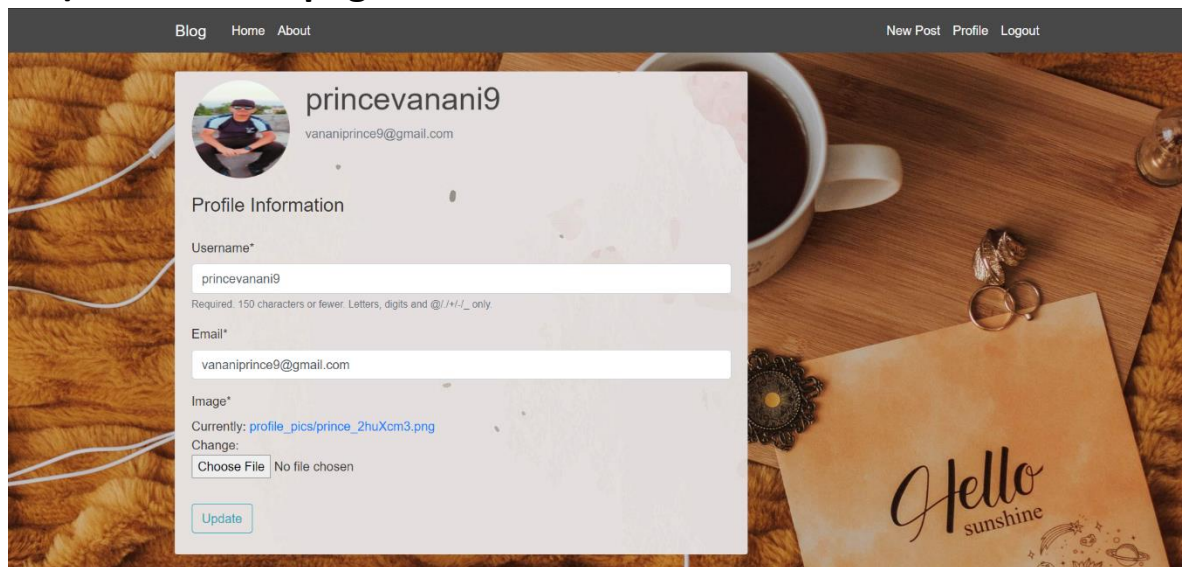
Title\*

Content\*

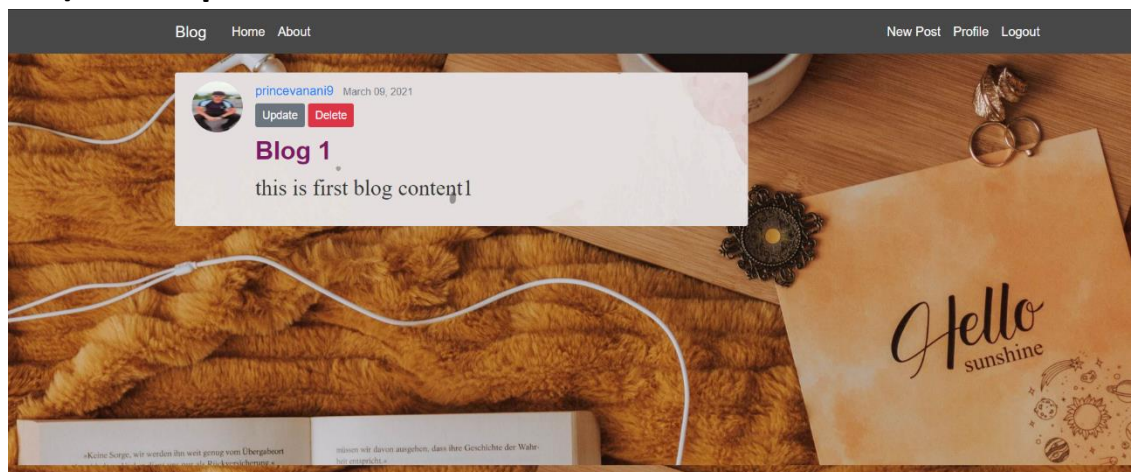
Post



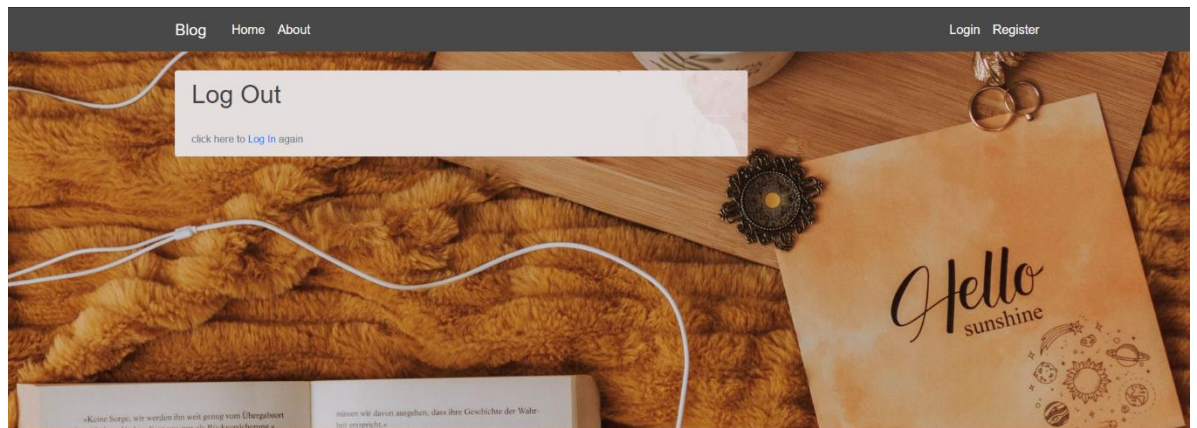
## 6.5) User Profile page



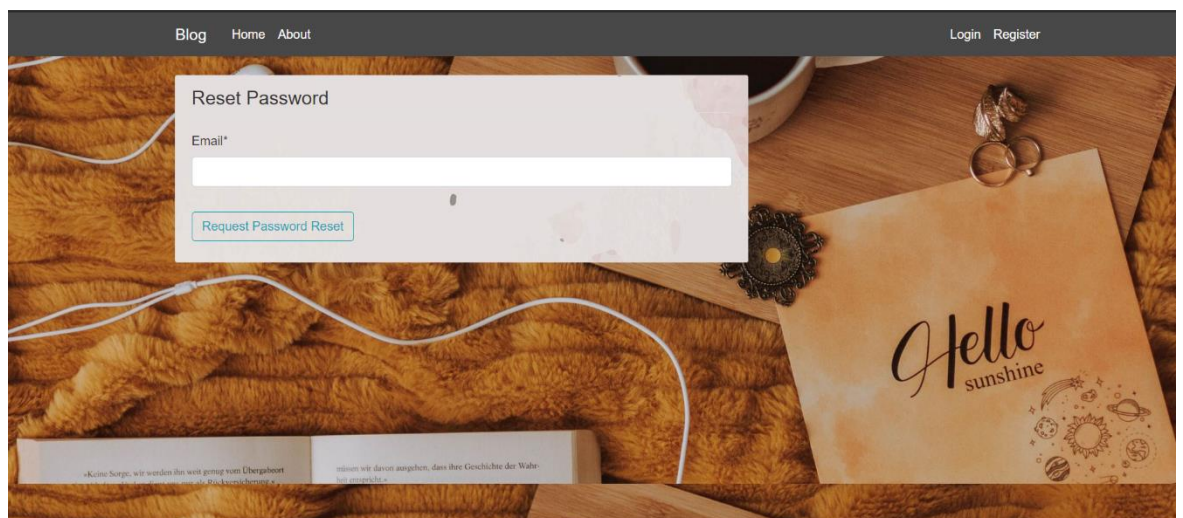
## 6.6) Post Update/Delete



## 6.7) Logout page



## 6.8) Reset password page





## **7.) Conclusion**

- This is a paperless work
- it is an informational website; it reduces the manpower required.
- It provides accurate information always. Malpractice can be reduced.  
All gathered and extra information can be saved and can be accessed at any time.
- So, it is better to have a Web-Based Information Posting system. We read it  
via internet

## **8.) Limitation**

- This system cannot work offline. it's must require internet.
- There is risk of cyber-attack from internal or external actors, which  
may manipulate the vote.
- if user wants to search particular post, he/she go through once and read it there  
is no Search option.

## **9.) Future extension**

-We improve our project and overcome from this limitation, add some security an also add more modules for search post. Particular user post in one page.

## **10.) Bibliography:**

### **Websites:**

<https://stackoverflow.com/>

<https://www.w3schools.com/>

<https://docs.djangoproject.com/en/3.1/>