Compsoft Technologies
Bangalore,Karnataka

AN INTERNSHIP REPORT
ON

" Quora for Engineering Students "

BACHELOR OF ENGINEERING

In

INFORMATION SCIENCE AND ENGINEERING

Submitted by: MULE MAHENDRA REDDY (1MJ18IS024)

Under the Supervision of: Sumukh S Jadhav

MVJ COLLEGE OF ENGINEERING
Since 1982
Engineering A Better Tomorrow

An Autonomous Institute

MVJ COLLEGE OF ENGINEERING

(Affiliated to Visvesvaraya Technological University and Approved by AICTE, New Delhi)Accredited By NAAC with 'A' Grade

ISO 9001-2015 and 14001-2015 certified Institute

BENGALURU, KARNATAKA- 560067

2021-2022

ABOUT THE COMPANY

CST is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At CST , we believe that service and quality is the key to success
We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

- Development - We develop responsive, functional and super-fast websites.
- We keep User Experience in mind while creating websites. A website should load quickly and shouldbe accessible even on a small view-port and slow internet connection.
- Mobile Application - We offer a wide range of professional Android, iOS& Hybrid app development services for our global clients, from a start up to a large enterprise.
- Design - We offer professional Graphic design, Brochure design & Logo design. We are expertsin crafting visual content to convey the right message to the customers.
- Consultancy - We are here to provide you with expert advice on your design anddevelopment requirement.
- Videos - We create a polished professional video that impresses your audience.

Table of Contents

OVERVIEW OF THE PROJECT

Project Name: **Quora for Engineering Students**

Team Members: MULE MAHENDRA REDDY, VINAYA H NAIKAR.

QES is a question and answer website for Engineering students. We tend to create a flagship site similar to the Stack Overflow Site of the Stack Exchange Network, created in 2008 by Jeff Atwood and Joel Spolsky. QES features questions and answers on a wide range of topics in Engineering. Our notion of a better-connected engineering community to tackle all kinds of academic and non Academical questions and problems would make the engineering community better. Establishing a community-based platform like this would help bypass the location barrier for Engineering students.

Short Term GOALS

1. Solve queries related to/limited to the engineering community
2. Make a better Community that can start discussions/mini topic debates
3. Upvote/Downvote answers

SPECIFICATIONS to be present in your initial build-Build 0.0.1

1. A working front end site with appropriate CSS
2. A working backend database where user data is to be stored
3. Registering upvote/downvote to the question/answer and the same being dynamically updated on the front end
4. Use of JS to add conditional features to the site
5. Login/Signup feature

Quora for Engineering Students

This is similar to Quora Website. Where user can do registeration and able to ask question. Even answer to those whoever has asked in the website. This is basically used to share the individual perspective on the particular question. If the any of the answer is good enough then they can upvote. I answer is not satisfactory then they can downvote it. Even then can comment on the basis of the answer user like to comment. People can search via search bar whatever they wanted to search it. User can even search on the basis of their interest through tags.

There are Homepage, Detail page, Tags page, Ask page, Login page, Register page, Answer page. Inside the Homepage we have set of question asked. How many comments are done. Who added the question with user name and date and time when the question has been added.

Inside the detail page we have the question with relevant answer. Even the upvote and downvote for those answer. Even the comment for those answer. Inside the tag page we the set tags.

Inside Ask page we have title field where we put down the question to be asked, detail field contains the message about the question and the tags field contain the tags of the question which tag is belong to.

Inside the Login page we have the username field, password field and submit button.

Inside the Register page we have the username field, password field, password confirmation and the register button.

TOOLS USED

Software Requirements

- Visual Studio Code 2019.
- Google Chrome or Microsoft Edge of latest version.
- Front End: HTML, CSS, JS
- Backend : Python, Django, SQLLite.
- Linux 7.1 or Windows XP/7/8/10 OS or Mac OS

Hardware Requirements

- Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).
- Monitor with a refresh rate of at least 40Hz for a smooth GUI experience (optional).

IMPLEMENTATION

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home Page</title>
    <!-- Bootstrap 4 -->
    <link rel="stylesheet" href="{% static 'bootstrap.min.css' %}" />
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="#">MY QUORA</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="#">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/tags">Tags</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/ask-question">Ask</a>
                    </li>
                    {% if user.is_authenticated %}
                    <li class="nav-item">
                        <a class="nav-link" href="/accounts/profile">Profile</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'logout' %}">Logout</a>
                    </li>
                    {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="/accounts/login">Login</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/accounts/register">Register</a>
                    </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>
    <!-- Search Section -->
```

```
    <div class="container py-5">
        <div class="row">
            <div class="col-md-8 offset-2">
                <form>
                    <div class="input-group">
                        <input type="text" name="q" class="form-control form-control-lg"
placeholder="Search" />
                        <div class="input-group-append">
                        <button class="btn btn-dark btn-lg" type="submit" id="button-
addon2">Search</button>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Question List -->



    <div class="container">
        {% for quest in quests %}
        <div class="card mb-4 shadow">
            <div class="card-body">
                <h4 class="card-title"><a href="/detail/{{quest.id}}">{{quest.title}}</a></h4>
                <p class="card-text">{{quest.detail}}</p>
                <p>
                    <a href="#" class="mr-2">{{quest.user.username}}</a>
                    <a href="#" class="mr-2">{{quest.answer_set.count}} Answers</a>
                    <a href="#" class="mr-2">{{quest.total_comments}} Comments</a>
                </p>
            </div>
            <div class="card-footer">
                <small>{{quest.add_time}}</small>
            </div>
        </div>
        {% endfor %}
    </div>
    {% if quests.has_other_pages %}
    <!-- Pagination -->
    <nav class="my-3 container">
        <ul class="pagination">
            {% if quests.has_previous %}
          <li class="page-item"><a class="page-link"
href="?page={{quests.previous_page_number}}">Previous</a></li>
          {% endif %}
          {% for i in quests.paginator.page_range %}
          {% if quests.number == i %}
          <li class="page-item active"><a class="page-link" href="#">{{i}}</a></li>
          {% else %}
          <li class="page-item"><a class="page-link" href="?page={{i}}">{{i}}</a></li>
          {% endif %}
          {% endfor %}
```

```
            {% if quests.has_next %}
            <li class="page-item"><a class="page-link"
href="?page={{quests.next_page_number}}">Next</a></li>
            {% endif %}
        </ul>
      </nav>
    {% endif %}
</body>
</html>
from django.shortcuts import render,HttpResponse
from django.http import JsonResponse
from .models import Question,Answer,Comment,UpVote,DownVote
from django.core.paginator import Paginator
from django.contrib import messages
from .forms import AnswerForm,QuestionForm,ProfileForm
from django.contrib.auth.forms import UserCreationForm
from django.db.models import Count
# Home Page
def home(request):
    if 'q' in request.GET:
        q=request.GET['q']
        quests=Question.objects.annotate(total_comments=Count('answer__comment')).filter(title__i
contains=q).order_by('-id')
    else:
        quests=Question.objects.annotate(total_comments=Count('answer__comment')).all().order_by(
'-id')
    paginator=Paginator(quests,10)
    page_num=request.GET.get('page',1)
    quests=paginator.page(page_num)
    return render(request,'home.html',{'quests':quests})

# Detail
def detail(request,id):
    quest=Question.objects.get(pk=id)
    tags=quest.tags.split(',')
    answers=Answer.objects.filter(question=quest)
    answerform=AnswerForm
    if request.method=='POST':
        answerData=AnswerForm(request.POST)
        if answerData.is_valid():
            answer=answerData.save(commit=False)
            answer.question=quest
            answer.user=request.user
            answer.save()
            messages.success(request,'Answer has been submitted.')
    return render(request,'detail.html',{
        'quest':quest,
        'tags':tags,
        'answers':answers,
        'answerform':answerform
    })

# Save Comment
```

```
def save_comment(request):
    if request.method=='POST':
        comment=request.POST['comment']
        answerid=request.POST['answerid']
        answer=Answer.objects.get(pk=answerid)
        user=request.user
        Comment.objects.create(
            answer=answer,
            comment=comment,
            user=user
        )
        return JsonResponse({'bool':True})

# Save Upvote
def save_upvote(request):
    if request.method=='POST':
        answerid=request.POST['answerid']
        answer=Answer.objects.get(pk=answerid)
        user=request.user
        check=UpVote.objects.filter(answer=answer,user=user).count()
        if check > 0:
            return JsonResponse({'bool':False})
        else:
            UpVote.objects.create(
                answer=answer,
                user=user
            )
            return JsonResponse({'bool':True})

# Save Downvote
def save_downvote(request):
    if request.method=='POST':
        answerid=request.POST['answerid']
        answer=Answer.objects.get(pk=answerid)
        user=request.user
        check=DownVote.objects.filter(answer=answer,user=user).count()
        if check > 0:
            return JsonResponse({'bool':False})
        else:
            DownVote.objects.create(
                answer=answer,
                user=user
            )
            return JsonResponse({'bool':True})

# User Register
def register(request):
    form=UserCreationForm
    if request.method=='POST':
        regForm=UserCreationForm(request.POST)
        if regForm.is_valid():
            regForm.save()
            messages.success(request,'User has been registered!!')
```

```
    return render(request,'registration/register.html',{'form':form})

# Ask Form
def ask_form(request):
    form=QuestionForm
    if request.method=='POST':
        questForm=QuestionForm(request.POST)
        if questForm.is_valid():
            questForm=questForm.save(commit=False)
            questForm.user=request.user
            questForm.save()
            messages.success(request,'Question has been added.')
    return render(request,'ask-question.html',{'form':form})


# Questions according to tag
def tag(request,tag):
    quests=Question.objects.annotate(total_comments=Count('answer__comment')).filter(tags__iconta
ins=tag).order_by('-id')
    paginator=Paginator(quests,10)
    page_num=request.GET.get('page',1)
    quests=paginator.page(page_num)
    return render(request,'tag.html',{'quests':quests,'tag':tag})

# Profile
def profile(request):
    quests=Question.objects.filter(user=request.user).order_by('-id')
    answers=Answer.objects.filter(user=request.user).order_by('-id')
    comments=Comment.objects.filter(user=request.user).order_by('-id')
    upvotes=UpVote.objects.filter(user=request.user).order_by('-id')
    downvotes=DownVote.objects.filter(user=request.user).order_by('-id')
    if request.method=='POST':
        profileForm=ProfileForm(request.POST,instance=request.user)
        if profileForm.is_valid():
            profileForm.save()
            messages.success(request,'Profile has been updated.')
    form=ProfileForm(instance=request.user)
    return render(request,'registration/profile.html',{
        'form':form,
        'quests':quests,
        'answers':answers,
        'comments':comments,
        'upvotes':upvotes,
        'downvotes':downvotes,
    })

# Tags Page
def tags(request):
    quests=Question.objects.all()
    tags=[]
    for quest in quests:
        qtags=[tag.strip() for tag in quest.tags.split(',')]
        for tag in qtags:
```

```
            if tag not in tags:
                tags.append(tag)
    # Fetch Questions
    tag_with_count=[]
    for tag in tags:
        tag_data={
            'name':tag,
            'count':Question.objects.filter(tags__icontains=tag).count()
        }
        tag_with_count.append(tag_data)
    return render(request,'tags.html',{'tags':tag_with_count})
```

```
from django.db import models
from django.contrib.auth.models import User,AbstractUser


# Custom User Model
class CustomUser(AbstractUser):
    bio=models.TextField()
    location=models.CharField(max_length=200)


# Question Model
class Question(models.Model):
    user=models.ForeignKey(CustomUser,on_delete=models.CASCADE)
    title=models.CharField(max_length=300)
    detail=models.TextField()
    tags=models.TextField(default='')
    add_time=models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

# Answer Model
class Answer(models.Model):
    question=models.ForeignKey(Question,on_delete=models.CASCADE)
    user=models.ForeignKey(CustomUser,on_delete=models.CASCADE)
    detail=models.TextField()
    add_time=models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.detail

# Comment
class Comment(models.Model):
    answer=models.ForeignKey(Answer,on_delete=models.CASCADE)
    user=models.ForeignKey(CustomUser,on_delete=models.CASCADE,related_name='comment_user')
    comment=models.TextField(default='')
    add_time=models.DateTimeField(auto_now_add=True)
```

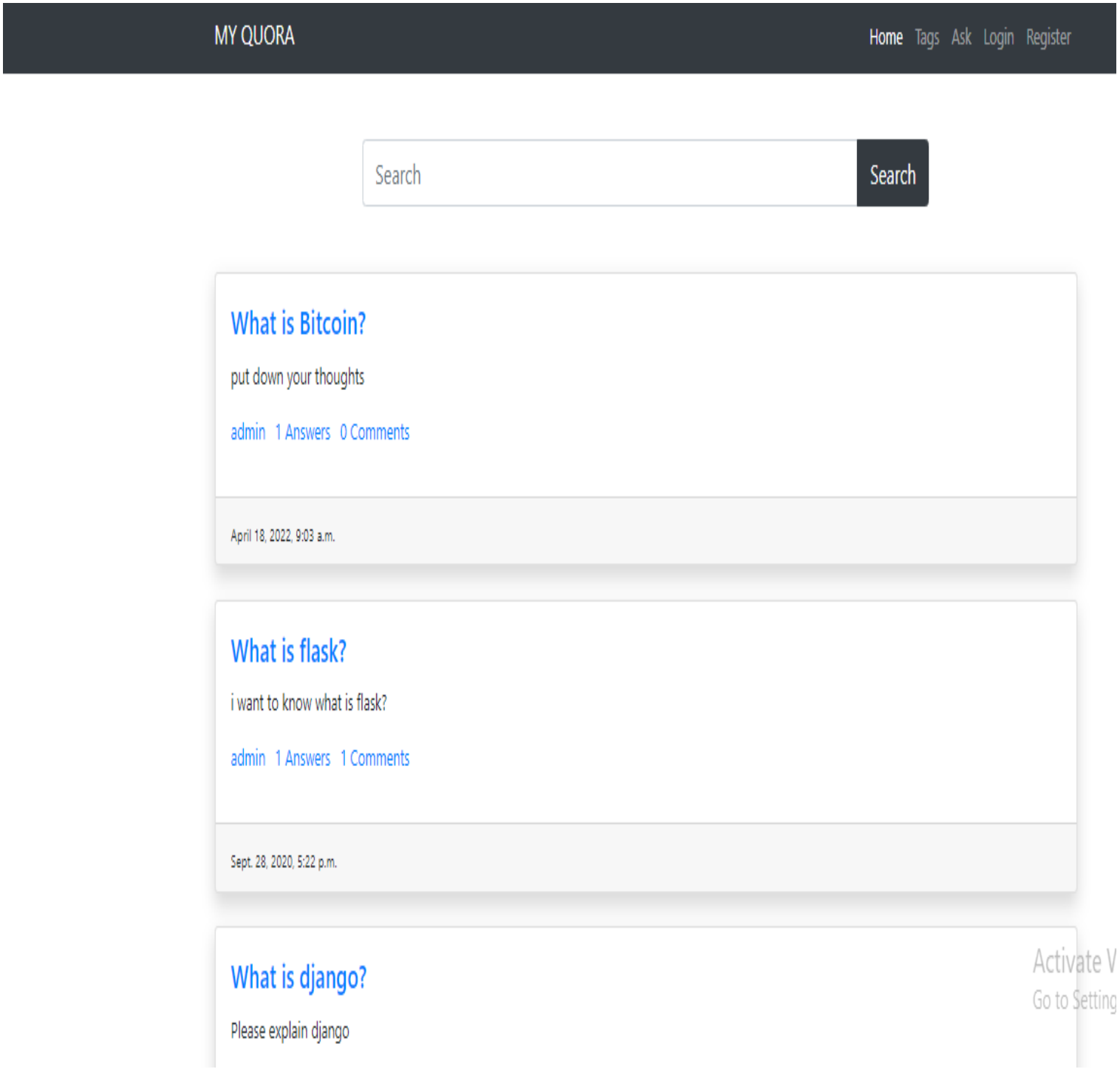
```
    def __str__(self):
        return self.comment

# UpVotes
class UpVote(models.Model):
    answer=models.ForeignKey(Answer,on_delete=models.CASCADE)
    user=models.ForeignKey(CustomUser,on_delete=models.CASCADE,related_name='upvote_user')

# DownVotes
class DownVote(models.Model):
    answer=models.ForeignKey(Answer,on_delete=models.CASCADE)
    user=models.ForeignKey(CustomUser,on_delete=models.CASCADE,related_name='downvote_user')
```

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Detail Page{% endblock %}</title>
    <!-- Animate 4 -->
    <link rel="stylesheet" href="{% static 'animate.min.css' %}" />
    <!-- Bootstrap 4 -->
    <link rel="stylesheet" href="{% static 'bootstrap.min.css' %}" />
    <!-- Font Awesome 4 -->
    <link rel="stylesheet" href="{% static 'fa4/css/font-awesome.min.css' %}" />
    <!-- Jquery -->
    <script src="{% static 'jquery-3.5.1.min.js' %}"></script>
    <!-- Bootstrap 4 Js -->
    <script src="{% static 'bootstrap.min.js' %}"></script>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="/">MY QUORA</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <form class="ml-auto" action="/">
                    <div class="input-group">
                        <input type="text" name="q" class="form-control" placeholder="Search" />
                        <div class="input-group-append">
                        <button class="btn btn-success" type="submit" id="button-
addon2">Search</button>
                        </div>
                    </div>
```

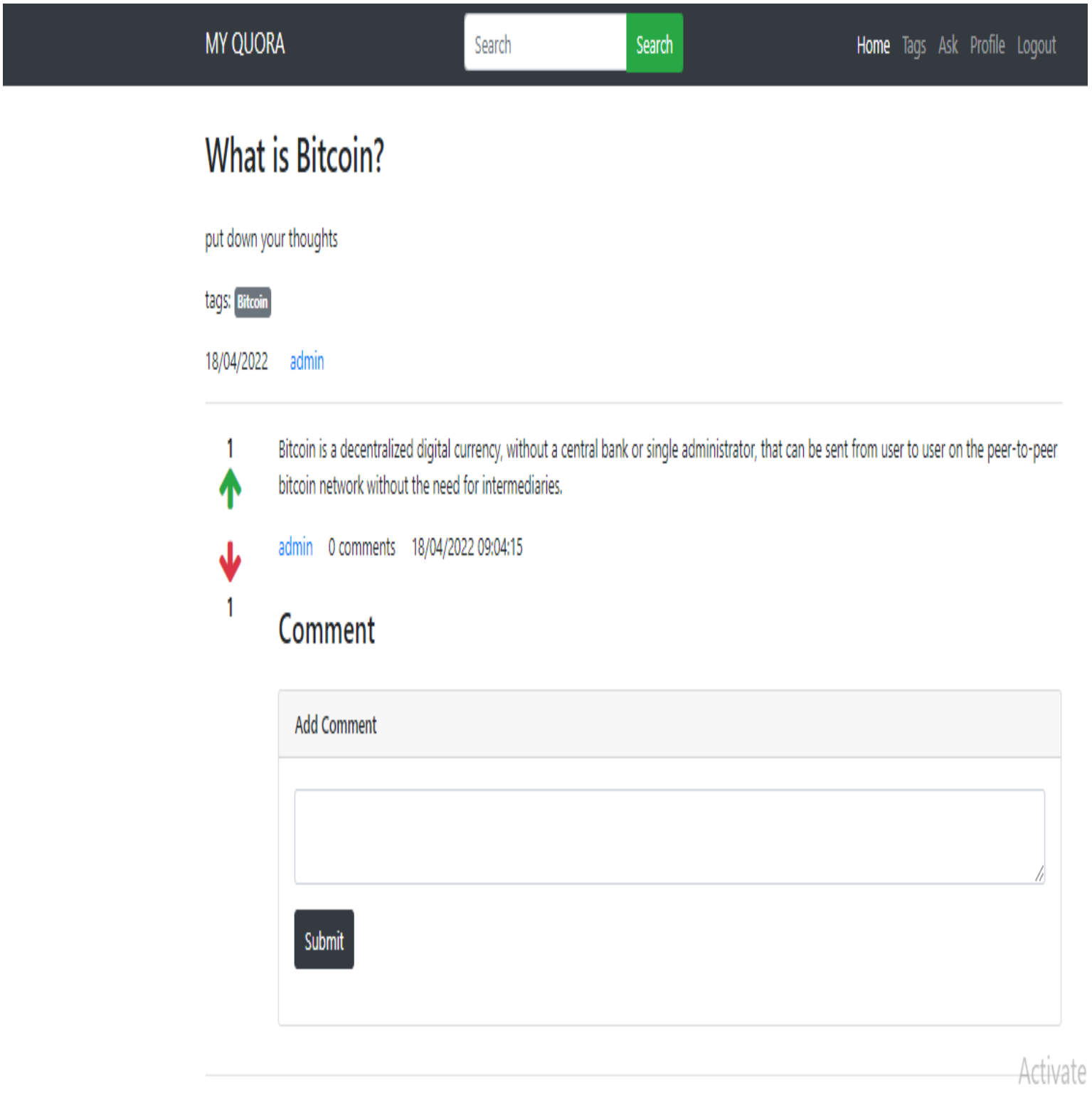
```
                </form>
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="/">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/tags">Tags</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'ask-question' %}">Ask</a>
                    </li>
                    {% if user.is_authenticated %}
                    <li class="nav-item">
                        <a class="nav-link" href="/accounts/profile">Profile</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'logout' %}">Logout</a>
                    </li>
                    {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="/accounts/login">Login</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/accounts/register">Register</a>
                    </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>
    {% block content %}
    {% endblock %}
</body>
</html>
```

SNAPSHOTS

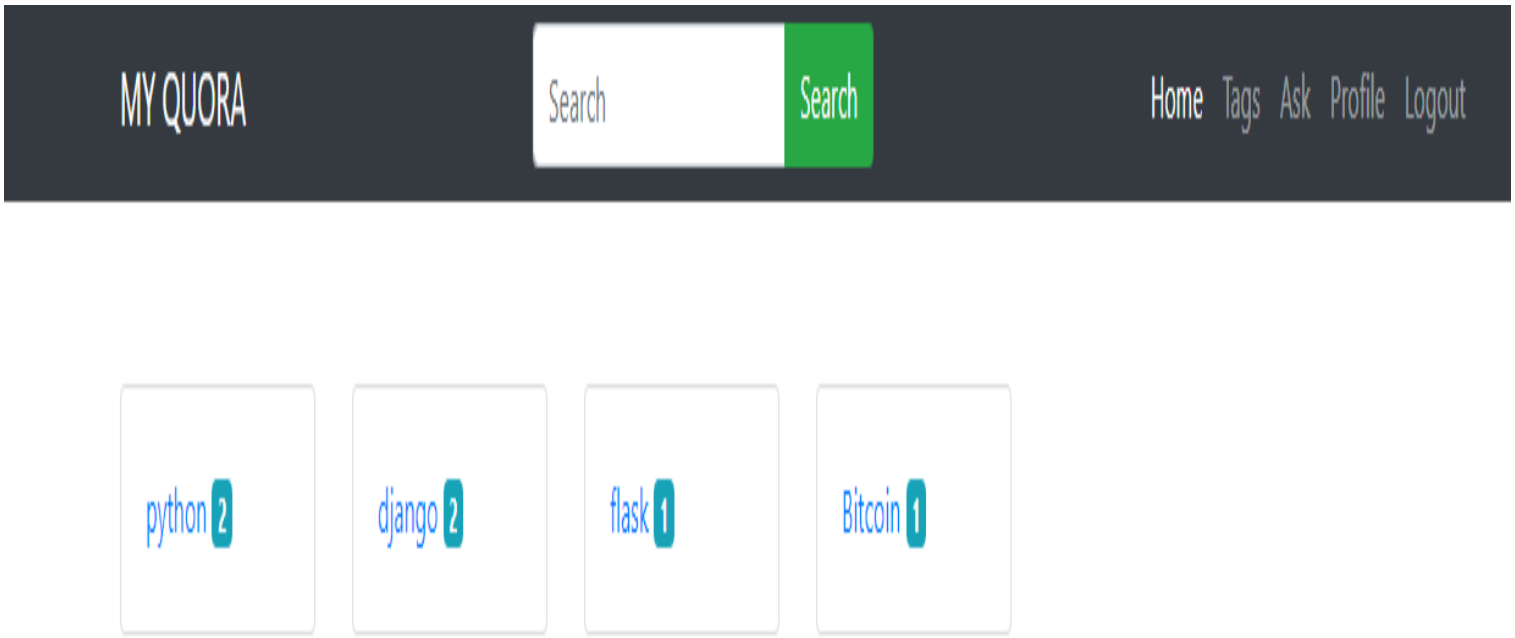
HOMPAGE



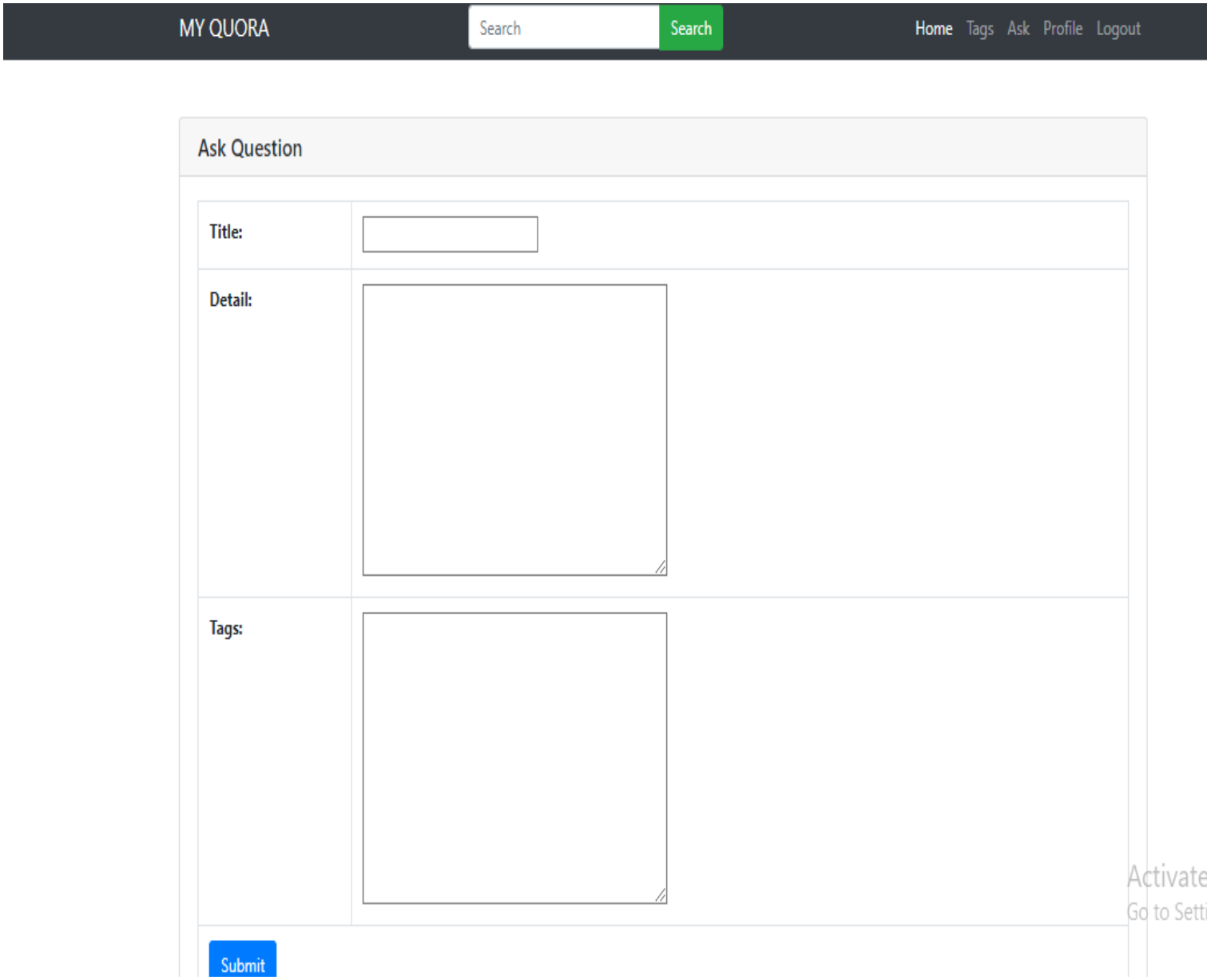
DETAILS PAGE



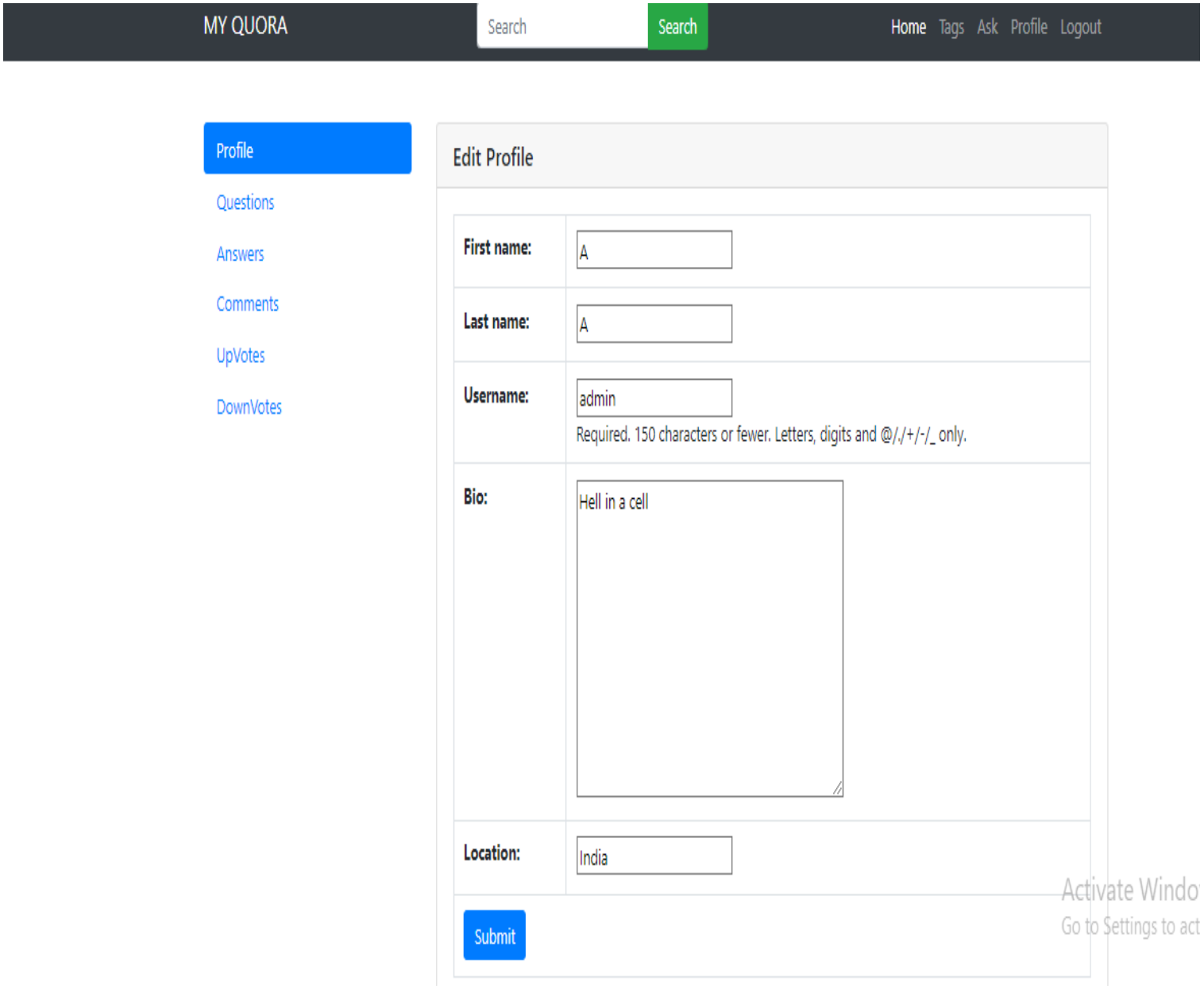
TAGS PAGE


MY QUORA
Search
Search
Home
Tags
Ask
Profile
Logout
python 2
django 2
flask 1
Bitcoin 1

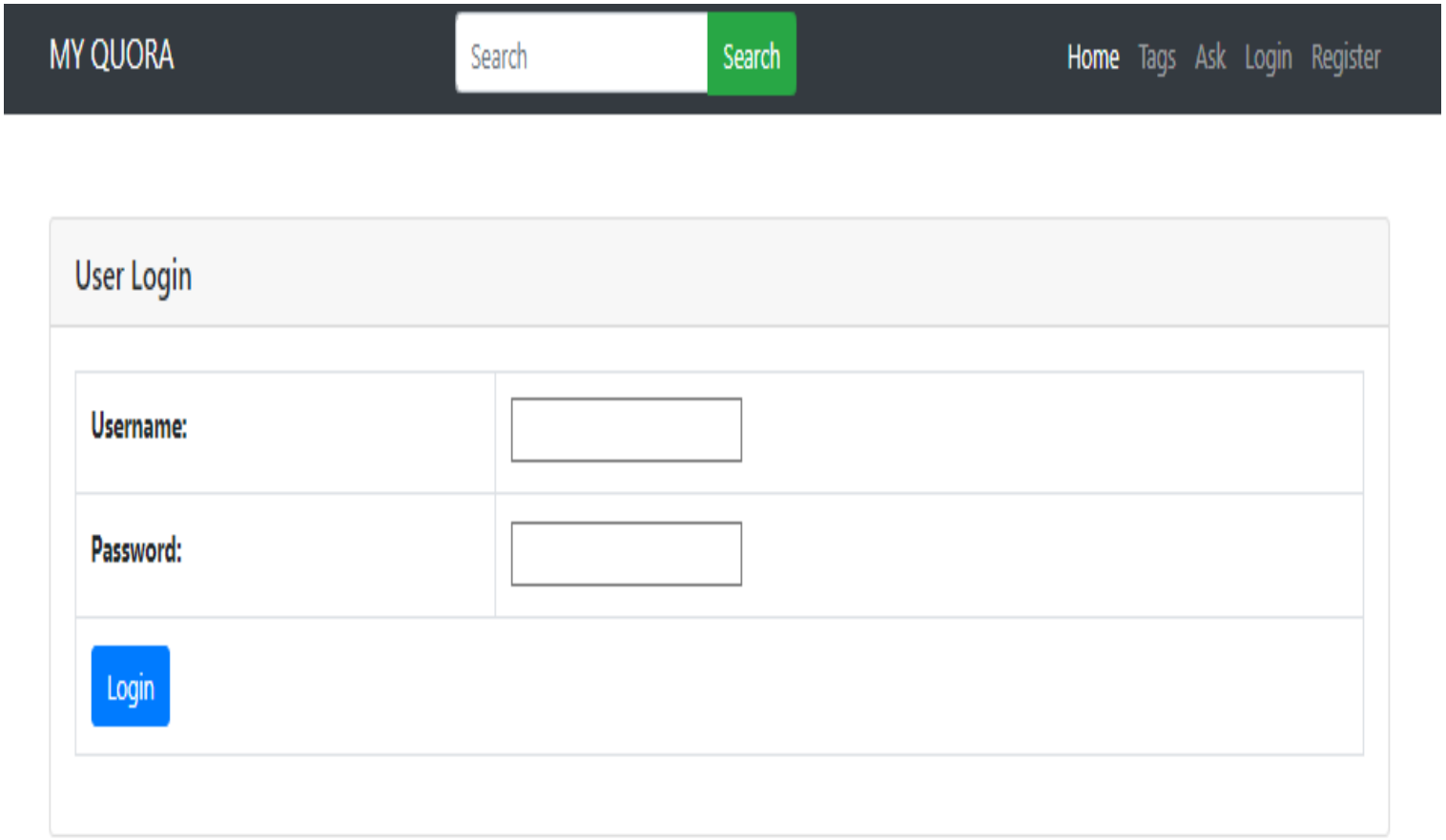
ASK PAGE


MY QUORA
Search
Search
Home
Tags
Ask
Profile
Logout
Ask Question
Title:
Detail:
Tags:
Submit
Activate
Go to Setti

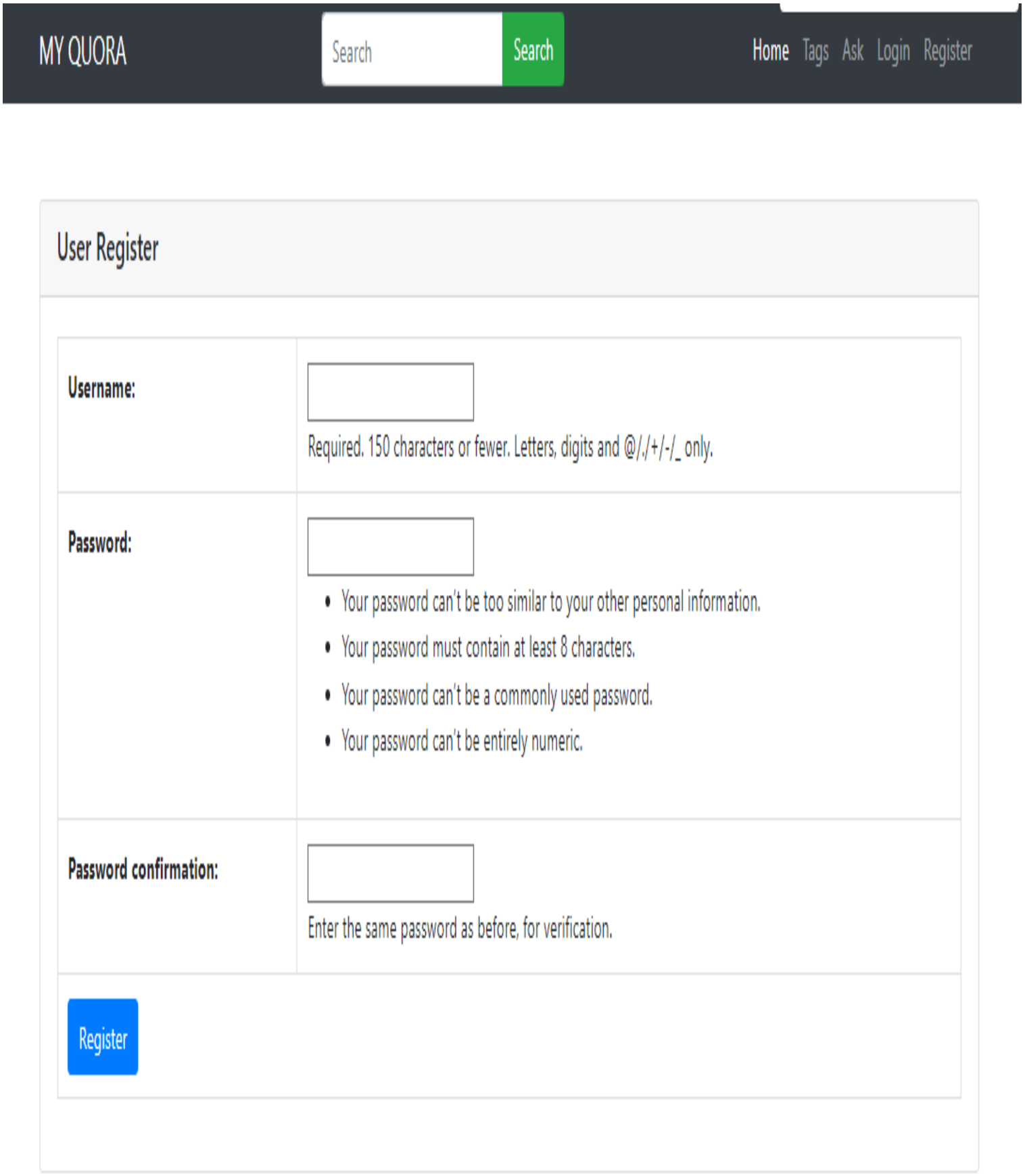
PROFILE PAGE



LOGIN PAGE


MY QUORA
Search
Search
Home
Tags
Ask
Login
Register
User Login
Username:
Password:
Login

REGISTER PAGE



BIBLIOGRAPHY

- https://www.w3schools.com
- https://www.geeksforgeeks org
- https://freefrontend.com
- https://css-tricks.com/
- https://www.takeiteasyengineers.com
- https://dev.to/mychi_darko/php-tips-and-tricks-4kpn