ANA ISSUE TRACKER

A Project Report

Submitted to Dr. Mrinal Kanti Das

for

Data Engineering Lab

in

Data Science

by

Abhichal Rajput

142002001

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

Other Group Members :-

Navjyoti Meena (142002013)

Akshatha MS (142002006)

Acknowledgement

We would like to express our gratitude and appreciation to all those who gave us the support to complete this course project and this report. A special thanks to our mentor, **Dr. Mrinal Kanti Das**, whose help, suggestions and encouragement helped us to execute this course project. We would like to acknowledge with much appreciation the crucial role of TAs of this course **Rimmon Bhosale** and **Shikha Mallick** who helped us to work on this project and let us develop our knowledge about the new things for implementation purpose. I also thank all the team members who gave us their suggestions for improvement in the work during the implementation of the project.

Abhichal Rajput

Introduction

An Issue tracker is a system, which resolves the problems of users. An issue tracker as the name suggest tracks the issues, it manages and maintains lists of issues. An issue tracking system is a software application that allows an enterprise to record and follow the progress of every problem or “issue”, which a computer system identifies until the problem is resolved. With an ITS, an issue which can be anything from a simple user question to a detailed technical report of an error or bug, cab be tracked by priority status, owner or some other customized criteria.

Objective

We are working on a project, which is a website called “ANA ISSUE TRACKER”, it is an issue tracker which tracks the problems of people like new students who wants to take admission in the new college, here we have collected data for IIT PALAKKAD for just a sample base. So on our website, one can search his queries related to anything like stream, sports, academics, faculty etc. The idea behind choosing this topic is as a student we have to face many challenges regarding admission in colleges like IITs, NITs, and Govt. Colleges etc. Therefore, everyone wants a quick reply to his/her queries without any confusion so that he/she can right decision to choose the best choice to him/her.

0.1 Required Tools

- HTML (Hyper Text Markup Language)
- CSS (Cascading Style Sheet)
- MariaDB
- Django

0.2 Project Description

We have used Django because we have to work on python and this is the best platform to work on python, it also can easily collaborated with the Maria dB and mongo dB, for making front end part that is login/signup page we used Html and CSS, CSS is mainly styling purpose software. Mari DB is used for data storage part, in Maria DB,

we can easily put our data in tables, and easily it can be accessed easily. In our project we have divided out data set into categories, the categories we used are-

- Hostel
- Academics

We have created questions according to these two categories, but in this project, we can add and reduce any numbers of categories and questions. The website can respond to one word search. For doing the project our team started with the front-end part, we started to collect things like those that we went through the tutorials and some website. As the categories are only related to our college, so I searched the questions part from the college website. I prepared questions according to the queries, which are most frequently asked. The task was challenging when we have to merge all those things. However, by Django this became easy for us.

0.3 Methodology

For implementation we divided our work in two part, first one is front end and the other one is back end part. I have done work on only front-end part that is why I am describing only front-end part. There are so many techniques of DBMS and data structure is required to make an issue tracker. Some specific techniques are-

- Data Collection
- Data Preprocessing
- Data Cleaning
- Data Storage
- Information Retrieval
- Searching
- Data Analysis

App Flow

For the front-end part, we used html and CSS, html is now a day very easy and easy to handle programming language and CSS is the styling oriented language, it makes the visualization interesting. In the Home Page, we have given three main icons.

The Home page icons are-

- Home
- Login
- Signup

A user and a super user can sign up by the login portal, after enrolling it can login with the password, which is already set. If anyone chooses any random password, he/she cannot login.

We have created two categories User and Super users (experts). The rolls are different for every category, a user can only ask questions, but a super user can insert, delete or update a query. If a query is not in the given data set, it is stored and can be replied after sometime by a super user. There is a table bar in the right corner, which can show the number of existing users, experts, total questions and unanswered questions, which is useful for both user and expert.

After logging in a user can search for the questions already present in the database through All Questions page. If required query is not found then the user can submit a query which will later be solved by the expert and when expert answers the query it get's added in the Database

Contribution

I worked on the back-end part, i.e. creating forms, databases, integration and also did some modifications in the front end part so that it can get supported by the django framework which we have used in the project creation.

0.4 Codes

0.4.1 Login, Register, Add Question and Ask Question Forms

```
from django import forms
from django.contrib.auth.forms import UserCreationForm, UserChangeForm
from django.forms import Form,ChoiceField
from .models import CustomUser,questions,askquestions


class SignUpForm(UserCreationForm):
    full_name = forms.CharField(max_length=100, help_text='Required. 100 charaters or fewer.', )

    class Meta:
        model = CustomUser
        fields = UserCreationForm.Meta.fields + ('full_name', 'age',)

class question(forms.ModelForm):
    question=forms.CharField(widget=forms.Textarea,required=True)
    class Meta:
        model=questions
        fields=[
            "category",
            "question",
            "answer",
        ]

class askquestion(forms.ModelForm):
    question=forms.CharField(widget=forms.Textarea,required=True)
    class Meta:
        model=askquestions
        fields=[
            "category",
            "question",
        ]
```

Figure 1: Login, Register, Add Question and Ask Question Forms

0.4.2 Django Views

```python
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import user_passes_test
from .forms import SignUpForm,question,askquestion
from .models import questions,askquestions
from django.db.models import Count
from django.views.generic import ListView
from .filters import OrderFilter
from .models import CustomUser,questions
from .models import askquestions as ask


def home(request):
    sup_count=2
    user_count=CustomUser.objects.count()-sup_count
    que_count=questions.objects.count()
    unans_count=ask.objects.count()
    return render(request, 'home.html',{'user_count':user_count,'que_count':que_count,
                  'sup_count':sup_count,'unans_count':unans_count})



def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            user = form.save()
            user.save()
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=user.username, password=raw_password)
            login(request, user)
            return redirect('home')
    else:
        form = SignUpForm()
    return render(request, 'signup.html', { 'form' : form })

```

```python
def allquestions(request):
    context={}
    que=questions.objects.all()

    myFilter=OrderFilter(request.GET, queryset=que)
    que = myFilter.qs
    if que:
        context={'dataset':que,'myFilter':myFilter}
        return render(request, 'allquestions.html',context)
    else:
        return render(request,'noresults.html')


def addquestion(request):
    if request.user.is_superuser:
        context={}

        form=question(request.POST or None)
        if form.is_valid():
            form.save()
        context['form']=form
        return render(request,'addquestion.html',context)
    else:
        return render(request,'noaccess.html')

def askquestions(request):
    context={}

    form=askquestion(request.POST or None)
    if form.is_valid():
        form.save()
    context['form']=form
    return render(request,'askquestions.html',context)
```

Figure 2: Django Views

0.4.3 Django Models/ Database Tables

```python
from django.db import models
from django.contrib.auth.models import AbstractUser
from django import forms


# Create your models here.
class CustomUser(AbstractUser):
    full_name = models.CharField(max_length=100, blank=False)
    age = models.PositiveIntegerField(null=True, blank=True, default=0)


class questions(models.Model):
    options=[
            ("hostels","Hostels"),
            ("academics","Academics"),
            ]
    category=models.CharField(choices=options,max_length=20,default=0)
    question = models.CharField(max_length=200)
    answer=models.TextField()

    def __str__(self):
        return self.question

class askquestions(models.Model):
    options=[
            ("hostels","Hostels"),
            ("academics","Academics"),
            ]
    category=models.CharField(choices=options,max_length=20,default=0)
    question = models.CharField(max_length=200)
    def __str__(self):
        return self.question
```

Figure 3: Django Models/ Database Tables

0.4.4 Integration

```python
from django.contrib import admin
from django.urls import path,include
from django.contrib.auth import views as auth_views
from authen import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name='home'),
    path('login/', auth_views.LoginView.as_view(template_name='login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('signup/', views.signup, name='signup'),
    path('addquestion/',views.addquestion, name='addquestion'),
    path('allquestions/',views.allquestions, name='allquestions'),
    path('askquestions/',views.askquestions, name='askquestions'),
]

```

Figure 4: Integration

0.4.5 Filter/ Search Form

```python
import django_filters
from django_filters import CharFilter
from .models import *

class OrderFilter(django_filters.FilterSet):
    question=CharFilter(field_name='question', lookup_expr='icontains')
    class Meta:
        model = questions
        fields = '__all__'
        exclude=['answer']
```

Figure 5: Filter/Search Form

Future Scope

As today we can see that all the things are online like online shopping, net banking, online transactions, thus need of an issue tracker is big in the future. The issue tracker is like a bug fixer in any system. Frequently firms find that projects getting out of hand because of bugs are being developed from different sources to different team members and no one can really be confident about the bugs which are present, and which one have been fixed. Proper maintenance and details of bugs has to take great care. Issues happen at any stage of growth in any kind of product.

Conclusion

For this project, we have implemented all the things we have learnt during the course. By doing the project we have understood the things how we can practically implement them. Such important techniques are Data Collection, Information retrieval, data cleaning, data storage, use of Django administrator etc. For making a webpage how we use these things and how nicely they can work together. An issue tracker is really a challenging task, which can be implemented using so many tools of programming. Thus the project really worth in the practical use.