TEXT EDITOR USING PYTHON

Muhammad Ahmad Ali 183232

Hamza Danish 183164

Syed Abdul Hanan 183204

Muhammad Adnan Ameen 183166

Muhammad Saad Naeem 183176

Text Editor

A text editor is a type of laptop application that edits simple text. This package is known as "notepad" software program, following the name of Microsoft Notepad. Text editors are provided with operating systems and software development packages, and can be used to change files such as changing the font, font style, saving a document etc.

Why Use A Text Editor?

- Faster to Edit
- Simple to use
- Can be used for simple search and replace.

Objectives

In this project, we will develop a simple text editor or notepad in python. We will also implement basic functionalities like changing the font, font style, saving a document, etc.

Process Flow

We have used a CKEditor library. It is an open source and known to be a rich text editor. We have implemented it on python. We have made a function by which we allow the user to download. We will write the text in the check box at first. We have given tools at the top which we got from the CKEditor library.

At first, we write the text and press the ‘submit’ button. Then a file is made and saved. When we click on ‘download’ button, the file is downloaded.

We have used a CSRF token which allows different users to see their own work and not the same thing. For example, if I have done a work and another person has done his/her work and we both download our works. Then we both will see our own work and not the same work. There will be no duplication.

Django's login form is returned using the POST method, in which the browser bundles up the form data, encodes it for transmission, sends it to the server, and then receives back its response. GET, by contrast, bundles the submitted data into a string, and uses this to compose a URL.

We have made a model which acquires rich text.

```
5  class Post(models.Model):
6      body = RichTextField(blank=True, null=
   True)
```

There is a url.py file which ensures two functions.

1) Shows us the text editor.
2) Download function

```
5  urlpatterns = [
6      path('', views.home, name='index'),
7      path('download', views.download, name=
   'd'),
8  ]
```

This class is made for the interface of writing text. It allows us with two functions.

1) CKEditor widget allows us to fancy text such as bold.
2) CharField allows us to write text inside the box. CKEditorwidget is inside the CharField. The label is ‘Text Editor’.

```
11 class PostForm(forms.ModelForm):
12     body = forms.CharField(widget=
   CKEditorWidget(), label="Text Editor")
13     class Meta:
14         model = Post
15         fields = ('body',)
16
```

Meta class has model and fields. ‘Model’ allows us to post and ‘fields’ is the body meaning the text will be taken in the body.

The function ‘home’ is made used to have further functions about the opened page. This will check if the method is post if the method is post it will proceed. Now it will check the text in the form and will save it inside the form variable. If valid, it will show ‘Form is valid’. Then it will clear the text and put it inside of task variable. Beautifulsoup will convert html to plain text. Asss will convert it into gfg. Now gfg text will be put out in Asss and will be saved in a file named as NEWTEXTFILE.

```
17  def home(request):
18      if request.method == 'POST':
19          form = PostForm(data=request.POST)
20          if form.is_valid():
21              task = form.cleaned_data['body']
22              gfg = BeautifulSoup(task)
23              Asss = gfg.get_text()
24              file1 = open("NEWTEXTFILE.txt",
    "a")
25              file1.write(Asss)
26              file1.close()
27      else:
28          form = PostForm()
29      return render(request, 'text.html', {
    'form': PostForm()})
```

If the form is not a post form, then it will make it in a post form format and start the above procedure.

Download function is used in the below code for the text file.

```
def download(request):
    BASE_DIR = os.path.dirname(os.path.
dirname(os.path.abspath(__file__)))
    filename = 'NEWTEXTFILE.txt'
    filepath = BASE_DIR + '/' + filename
    path = open(filepath, 'r')
    mime_type, _ = mimetypes.guess_type(
filepath)
    response = HttpResponse(path,
content_type=mime_type)
    response['Content-Disposition'] =
"attachment; filename=%s" % filename
    return response
```

In this code we have made a form. The action in this form will be on the URL given. Meaning post method will take action on this given URL which is our home page. CSRF token will open different page for different users. form.media will show the tools given above bar. form.as_p will show us the text which we type and edit. Then clicking on the submit button will submit. Href hyperlink is for the download function.

```
        </nav>
    <form action="http://127.0.0.1:8000
/texteditor/" method="POST">
    <div class="container-fluid">

        <div class="row">
            <div>
                <div class="texteditor">
                    {% csrf_token %}
                        {{form.media}}
                        {{form.as_p}}
                </div>
                <input type="submit">
                <a href="http://127.0.0
.1:8000/texteditor/download"><strong><i>
Download</i></strong></a>
```

Conclusion

Our editors have the advantage that users do not need to have any knowledge of programming or markup languages to create a web document. The representation of data is displayed in visual form, as in Microsoft Word, and can be edited. The formatting instructions are located in the background of the program.