COMSATS University Islamabad, Vehari Campus

Department of Computer Science

CSC336-Web Technologies

Assignment # 3

Distribution Date: October 25, 2021
Submission Date: **November 01, 2021**

Note:

- Assignment is individual, so no cheating will be acceptable. Do your own efforts.
- Assignment must be submitted in hand written format as well as CU Online. (Hard+Soft)
- Appropriate penalty is charged in case of late submissions i.e. losing 20% marks per day

---

Course Content WebApp

Problem Statement: **Marks: 10**

Design a webapp for Course content creation using Django framework.

Background:

As we know by far that all pages in a web project are structured using a markup language called HTML. Our browsers these days only feel comfortable talking with a latest version of the markup language say HTML5. However, it is not possible today to code everything on a website using merely HTML. Even to build a daily structure of a running website, it is not feasible to implement routine posts on HTML language. Moreover, most of the contents writers these days are not technical enough to understand markup language at its best.

In this assignment the goal is to build a course content creator webapp which will be used by some university staff, usually non-technical person, with zero knowledge of HTML5. Imagine a course coordinator for some English Literature course who have no clue how to code a hello world page in HTML. But these days, the apps are smart enough to provide platform for such nontechnical persons who are able to create web content very easily. In practice, it would start to get tedious if every page on Course Content webapp had to be written in HTML. Instead, it can be helpful to store outline/topics using a lighter-weight human-friendly markup language.

For this assignment, we 'll store course outline topics into the web structure using a markup language called Markdown. Read through GitHub's Markdown guide

(https://docs.github.com/en/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax) to get an understanding for how Markdown's syntax works. Pay attention in particular to what Markdown syntax looks like for headings, bold text, links, and lists. By having one Markdown file represent each course outline topic, we can make our course-content entries more human-friendly to write and edit. When a user views the course outline topic, though, we'll need to convert that Markdown (.md) file into HTML before displaying it to the user.

Distribution Code:

The skeleton code for this assignment may be downloaded from:
CSC336 Web Technologies Assignment3
(https://drive.google.com/file/d/1xmawFirgvx9_GE3zhPEtmaR1dNG_0GsQ/view?usp=sharing)
download and unzip it.

Understanding:

This source code is a Django project called course which contains a single app called outline.
First, open up `outline/urls.py`, where the URL configuration for this app is defined. Notice that I've started you with a single default route that is associated with the `views.index` function.
Now, look at `outline/util.py`. You won't need to change anything in this file, but notice that there are three functions that may prove useful for interacting with course outline topics. `list_topics` returns a list of the names of all course outline topics currently saved. `save_topic` will save a new outline topic, given its title and some Markdown content. `get_topic` will retrieve an outline topic by its title, returning its Markdown contents if the topic exists or `None` if the topic does not exist. Any of the views you write may use these functions to interact with course outline topics.
Each course outline topic will be saved as a Markdown file inside of the `topics/` directory. If you check there now, you'll see that I've pre-created a few sample topics. You're welcome to add more!
Now, let's look at `outline/views.py`. There's just one view here now, the `index view`. This view returns a template `outline/index.html`, providing the template with a list of all of the topics in the outline (obtained by calling `util.list_topics`, which we saw defined in `util.py`).

You can find the template by looking at `outline/templates/outline/index.html`. This template inherits from a base `layout.html` file and specifies what the page's title should be, and what should be in the body of the page: in this case, an unordered list of all of the topics in the outline. `layout.html`, meanwhile, defines the broader structure of the page: each page has a sidebar with a search field (that for now does nothing), a link to go home, and links (that don't yet work) to create a new page or visit a random page.

Assignment Requirements:

Complete the implementation of your Course outline. You must fulfil the following requirements:

- **Topic Page:** Visiting localhost:8000/course/TITLE, where TITLE is the title of a outline topic, should render a page that displays the contents of that outline topic.
    - The view should get the content of the outline topic by calling the appropriate `util` function.
    - If a topic is requested that does not exist, the user should be presented with an error page indicating that their requested page was not found.
    - If the topic does exist, the user should be presented with a page that displays the content of the topic. The title of the page should include the name of the topic.
- **Index Page:** Update `index.html` such that, instead of merely listing the names of all pages in the outline, user can click on any topic name to be taken directly to that topic page.
- **Search:** Allow the user to type a query into the search box in the sidebar to search for a outline topic.
    - If the query matches the name of an outline topic, the user should be redirected to that topic's page.
    - If the query does not match the name of an outline topic, the user should instead be taken to a search results page that displays a list of all outline topics that have the query as a substring. For example, if the search query were `ytho`, then `Python` should appear in the search results.
    - Clicking on any of the topic names on the search results page should take the user to that topic's page.
- **New Page:** Clicking "Create New Page" in the sidebar should take the user to a page where they can create a new outline topic.

    - Users should be able to enter a title for the page and, in a textarea (https://www.w3schools.com/tags/tag_textarea.asp), should be able to enter the Markdown content for the page.
    - Users should be able to click a button to save their new page.
    - When the page is saved, if an outline topic already exists with the provided title, the user should be presented with an error message.
    - Otherwise, the outline topic should be saved to disk, and the user should be taken to the new topic's page.
- **Edit Page:** On each topic page, the user should be able to click a link to be taken to a page where the user can edit that topic's Markdown content in a `textarea`.
    - The `textarea` should be pre-populated with the existing Markdown content of the page. (i.e., the existing content should be the initial value of the `textarea`).
    - The user should be able to click a button to save the changes made to the topic.
    - Once the topic is saved, the user should be redirected back to that topic's page.
- **Random Page:** Clicking "Random Page" in the sidebar should take user to a random outline topic.

Hints:

- **Markdown to HTML Conversion:** On each topic's page, any Markdown content in the topic file should be converted to HTML before being displayed to the user. You may use the python-markdown2 (https://github.com/trentm/python-markdown2) package to perform this conversion, installable via **`pip3 install markdown2`**.
    - Challenge for pro-developer only (adopt on your own risk and skills): If you're feeling more comfortable, try implementing the Markdown to HTML conversion without using any external libraries, supporting headings, boldface text, unordered lists, links, and paragraphs. You may find using regular expressions in Python (https://docs.python.org/3/howto/regex.html) helpful.
- By default, when substituting a value in a Django template, Django HTML-escapes the value to avoid outputting unintended HTML. If you want to allow for an HTML string to be outputted, you can do so with the safe (https://docs.djangoproject.com/en/3.0/ref/templates/builtins/#safe) filter (as by adding |safe after the variable name you're substituting).

- Following python code is helpful for search results page in case the query does not exactly match the stored topics:

```
list_topics = ['CSS','Django','Git','HTML', 'Python']
query_word = 'ytho'

result = list(filter(lambda x: query_word in x, list_topics))
```

The result is a list containing all possible topics.

How to Submit:

- **Soft files submission**

  Put the project folder i.e. **course** folder**,** inside a folder with your registration#+Full Name e.g. "FA17-BSE-003+AHMAR TARIQ". In simple words, your complete project source code must be inside the parent folder **(with your registration and name)** e.g. FA17-BSE-003+AHMAR TARIQ folder. Now make a zip of your parent folder e.g. FA17-BSE-003+AHMAR TARIQ.zip and upload on CU Online.

- **Hard pages submission**

  You are required to handwrite all the (.html) templates inside the templates of the above Django project along with the `views.py` and `urls.py` files defined inside the outline app of the Django project, course.

Best of Luck!☺