COMP 8347: Internet Applications and Distributed Systems

SUMMER LAB #10

PART 1: Authenticate users

1. Create views for user *login* and *logout*.

a. Edit your *views.py* file as follows:

```
# Import necessary classes and models

from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required, user_passes_test

# Create your views here.
def user_login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user:
            if user.is_active:
                login(request, user)
                return HttpResponseRedirect(reverse('myapp:index'))
            else:
                return HttpResponse('Your account is disabled.')
        else:
            return HttpResponse('Invalid login details.')
    else:
        return render(request, 'myapp/login.html')

@login_required
def user_logout(request):
    logout(request)
    return HttpResponseRedirect(reverse(('myapp:index')))
```

b. Create *login.html* in the appropriate dir. It should ask for a username and password and pass the information to the **user_login** view function given above.

c. Update *myapp/urls.py* to handle **login** and **logout**.

2. ***myaccount* view**. Define a view function **`myaccount(request)`** in your *views.py* file. The user must be logged in to access this function. For a logged in user:

   a. If the user is a *Student*, return the following:

   - The first and last name of the student.
   - All the topics that the student is interested in..
   - All the courses the student is registered in.

   b. If the user is not a student, display a message: 'You are not a registered student!'.

   c. Update *myapp/urls.py* with a suitable pattern for **`myaccount()`** view.

d. Create *myaccount.html* and pass the above information, full name, courses ordered, and topics interested in, as a context to be displayed in the template.

PART 2: Work with cookies

1.. Set cookie in *about* view.

a. In *about* view, check for a **cookie** 'about_visits' that indicates the number of times the about page has been visited so far.

b. If the cookie exists retrieve the value and increment by 1.

c. Store the updated information in a cookie ('about_visits') and set it to expire after 5 minutes.

d. Pass this information (i.e. number of visits) to the template. Update *about.html* template to show how many times the page has been visited.

2. Use session framework.

a. Update user_login view to

- generate the date and time of the current login.
- Store this value as a session parameter (*last_login*).
- Set the session expiry to 1 hour.

b. When the index page is visited, check if *last_login* exists in session. If so, display this value. Otherwise, display the message "Your las login was more than one hour ago".

Is the value changing each time you visit the index page? Check what happens when

i) you are logged in as an authenticated user and

ii) you are **not** logged in.