12-780 Advanced Python and Web Prototyping for Infrastructure Systems

Project Progress Report:

Employee Training Management System

Name: Jingxiao Liu **Andrew ID**: jingxial

Date: Nov 2016 **Email**: jingxial@andrew.cmu.edu

1. Brief Description

Employee training is important for companies and employees. Engaged in skilled categories of workers must undergo training before induction. Thus, employees have been encouraged to take classes at local colleges, and companies would like to use a system to administrate information about employees and trainings. For example, employees can search and register courses they are interested in, and review their grades and scores of courses they have completed by using that system on webpage. And instructors can assign courses, review students' profiles and do grade on that site.

I used Django as the framework for this project. A database driven website, which is dynamic, are established. In addition, log-in function was developed, because this system should provide different functions for employees and instructors. A search bar will be established for searching courses and instructors later.

2. Main Progress

2.1 Database (Models in Django)

As what I have mentioned in proposal, I created a simple database, whose model focuses on employee training. There are five tables, Employee, Training, Course, Instructor and Grade, inside my database.

Following is the schema of my database:

EMPLOYEE (Emp_ID, Emp_Lname, Emp_Fname, Email, DOB, Hire_Date, Street, City, State, Zip_Code)

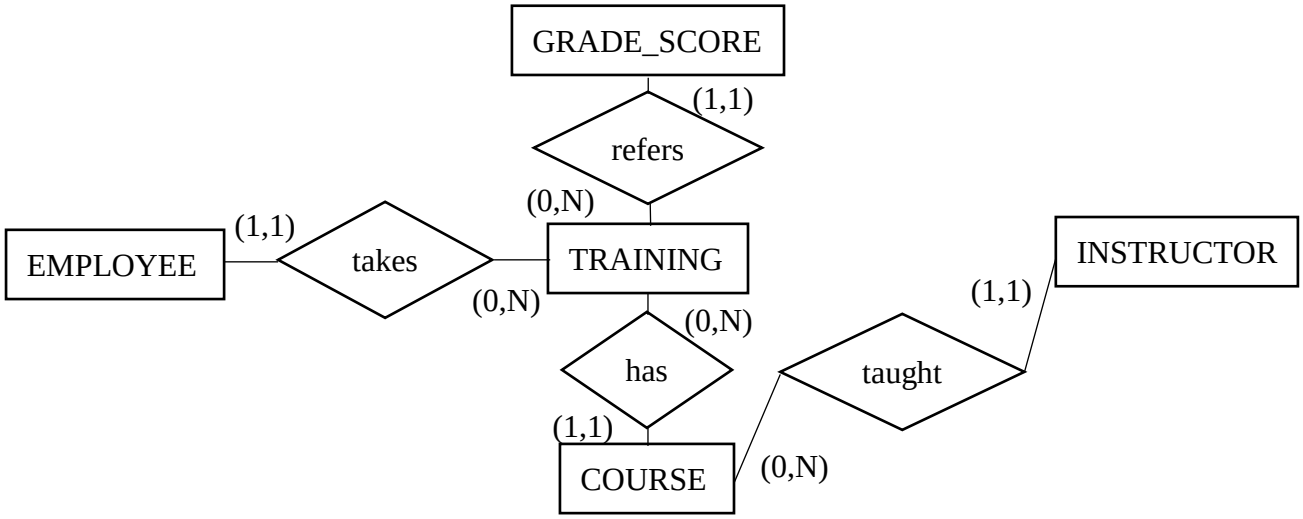
TRAINING (TID, Emp_ID@, Crs_ID@, Grade@)

COURSE (Crs_ID, Crs_Title, Crs_Type, Instr_ID@)

INSTRUCTOR (Instr_ID, Instr_Lname, Instr_Fname, Instr_Phone, Specialty)

GRADE_SCORE (Grade, Score)

The ER diagram of my database is shown in Figure 1.



In order to create database in my web application, I created five models in Django framework (codes are shown as follows), and migrated them to the SQLite3 engine. Furthermore, for convenient using in Django administration system, I registered the 'EmployeeTraining' and 'instructor' table into admin.py file. Thus, I can give them permission to log in this system.

```
class EmployeeTraining(models.Model):
    lname = models.TextField(max_length=30)
    fname = models.TextField(max_length=30)
    email = models.TextField(max_length=50)
    DOB = models.DateTimeField('Date of birth')
    hireDate = models.DateTimeField('Hire date')
    street = models.TextField(max_length=30)
    city = models.TextField(max_length=30)
    state = models.TextField(max_length=2)
    zip_code = models.IntegerField()
    empPassword = models.TextField(max_length=30)

class instructor(models.Model):
    instrLname = models.TextField(max_length=30)
    instrFname = models.TextField(max_length=30)
    instrPhone = models.IntegerField()
    specialty = models.TextField(max_length=30)
    instrPassword = models.TextField(max_length=30)

class course(models.Model):
    crs_Title = models.TextField(max_length=30)
    crs_Type = models.TextField(max_length=30)
    instructor = models.ForeignKey(instructor)

class gradeScore(models.Model):
    score = models.IntegerField()

class training(models.Model):
    EmployeeTraining = models.ForeignKey(EmployeeTraining)
    appr_Date = models.DateTimeField('Approve Date')
    course= models.ForeignKey(course)
```

```
    section = models.TextField(max_length=2)
    gradeScore = models.ForeignKey(gradeScore)
```

2.2 Log-in

Users need to enter their sign-in name and password to log in their account. There are two types of accounts, one is the employee account. And the other one is the instructor account. Different types of account will direct to different webpages. For employees, they can inquire the information of courses, review their profiles and grades. And for instructors, they can do grade, and search information of students in their class.

Furthermore, several views are created in the 'view.py' file to achieve login, logout, logged-in functions. For example, the 'auth_view' function is used to indicate whether the submitted username and password are equal to that in the administration database. If so, the webpage will be directed to the 'logged-in' page which will show employee's or instructor's profile; if nor, it will go to 'invalid login' page to ask user to login again. Those views' codes are shown as follows:

```
def login(request):
    c = {}
    c.update(request)
    return render_to_response('login.html', c)

def auth_view(request):
    username = request.POST.get('username', '')
    password = request.POST.get('password', '')
    user = auth.authenticate(username=username, password=password)

    if user is not None:
        auth.login(request, user)
        return HttpResponseRedirect('/loggedin/')
    else:
        return HttpResponseRedirect('/invalid/')

def loggedin(request):
    return render_to_response('loggedin.html',
                              {'full_name':request.user.username})

def invalid_login(request):
    return render_to_response('invalid_login.html')

def logout(request):
    auth.logout(request)
    return render_to_response('logout.html')
```

Finally, to accomplish login function, I create the URL patterns for connecting URLs of 'log-in', 'log-out', and 'logged-in' pages to Django framework (shown below). Also, static files, such as CSS style files and javascript files are connected to those pages.

```
urlpatterns = [
    url(r'^login/$', views.login, name='login'),
    url(r'^auth/$', views.auth_view, name='auth_view'),
    url(r'^logout/$', views.logout, name='logout'),
    url(r'^loggedin/$', views.loggedin, name='loggedin'),
    url(r'^invalid/$', views.invalid_login, name='invalid_login'),
```

```
    # url(r'^new_course/$', views.newCourse),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

2.3 Profile

By using Django, the attributes inside database can be gotten to our webpage. Firstly, I create a function in 'view.py' to request data from database and save those data to a string. The codes for this function are shown below:

```
def employeeRequest(request):
    employees = EmployeeTraining.objects.all()
    result = ""
    for employee in employees:
        result = result + employee.lname + " "
        result = result + employee.fname + ","
        result = result + employee.email + ","
        result = result + str(employee.DOB) + ","
        result = result + str(employee.hireDate) + ","
        result = result + employee.street + " "
        result = result + employee.city + " "
        result = result + employee.state + " "
        result = result + str(employee.zip_code) + ";"
    return HttpResponse(result)
```

Then, I need to add another URL in my 'urlpatterns', which uses 'view.employeeRequest' function. Furthermore, one javascript file is created to define classes and initialize data (codes are shown as follows). I firstly split the data as attributes, then save them into class 'Employee'. All those information will be shown in user's profile page.

```
function Employee(na, fn, em, dob, hd, add)
{
    this.name = na
    this.email = em;
    this.DOB = dob;
    this.hiredate = hd;
    this.address = add;
}

function initEmployees()
{
    var employees = [];

    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function()
    {
        if(xhttp.readyState == 4 && xhttp.status == 200)
        {
            var data = xhttp.responseText;
            var employeeStrings = data.split(";");

            for(var i=0; i<employeeStrings.length; i++)
            {
                var employeeString = employeeStrings[i];
                var attributeStrings = employeeString.split(",");
                var newEmployee = new
Employee(attributeStrings[0],attributeStrings[1],attributeStrings[2],attributeStrin
gs[3],attributeStrings[4])
                employees.push(newEmployee);
            }
        }
    }
```

3. Usage

Figure 1 shows the ‘log-in’ page for this system (HTML codes are shown in Appendix.).

Figure 1: Log-in Page.

By typing in correct username and password recorded in the administration system, user can be directed to their profile pages. If not, a massage will be shown to ask user to insert again.

Figure 2 and 3 are one employee and one instructor’s profile page (Template of this page is from W3), information is gotten from databased established by Django.

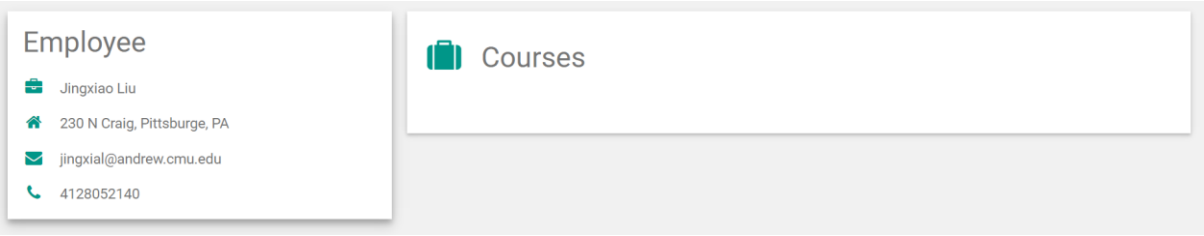


Figure 2: One example of Employee Profile page.

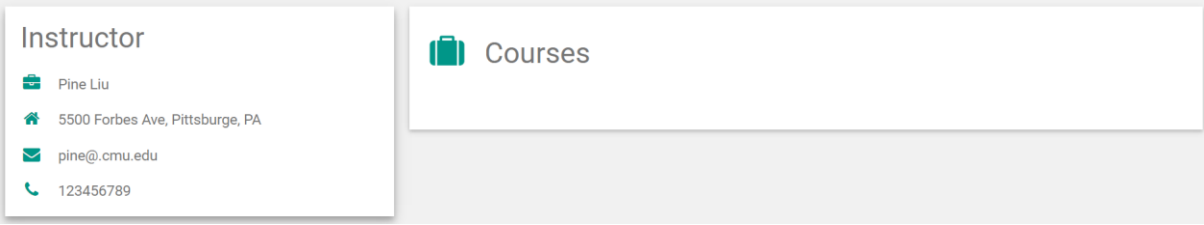


Figure 3: One example of Instructor Profile page.

4. Future Work

In the future, there are two main features will be developed:

Search bar:

This web application can help employees to search their interested courses from the database. Also, Django can help me to achieve this function, it has a framework like Google's search engine.

Grading:

For instructors, one main feature for them should be the grading function. In the near future, instructors can do gradings on employees who registered their course.

Appendix:

Log-in page codes:

```
{% block content %}

    <!DOCTYPE html>
    <html>
    <head>
      <title>Employee training management system</title>
      <link rel="stylesheet" href="../static/EmployeeTraining/homepagestyle.css">
    </head>
    <body>
      <section>
        <h1>Employee Training Management System</h1>
      </section>
      <section class="container">
        <div class="login">
          <h2>Login to the system</h2>
          {%  if form.errors %}
          <p>! Sorry your username and password didn't match, please try again.</p>
          {% endif %}
          <form method="post" action="/auth/">{% csrf_token %}
            <input type="text" id="username" name="username"
placeholder="Username">
            <input type="password" id="password" name="password"
placeholder="Password">
            <p class="remember_me">
              <label>
                <input type="checkbox" name="remember_me" id="remember_me">
                Remember me
              </label>
            </p>
            <p class="submit"><input type="submit" id="signin" value="Login"></p>
          </form>
          <p><a href="login.html">Forget your password?</a></p>
        </div>
      </section>

      <section class="about">
        <p class="about-author">
          &copy; 2016&ndash;2017 <a href="mailto:jingxial@andrew.cmu.edu"
target="_blank">Jingxiao Liu</a> -
          <a href="http://www.cmu.edu" target="_blank">CMU</a><br>
      </section>
    </body>
    </html>

{%  endblock %}
```