12-780 Advanced Python and Web Prototyping for Infrastructure Systems

Project Final Report:

Employee Training Management System

Name: Jingxiao Liu **Andrew ID**: jingxial
Date: Dec 2016 **Email**: jingxial@andrew.cmu.edu

1. Introduction

1.1. Problem 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 is also developed this time.

1.2. Objectives for Project

- Store data that contains the information about employees, instructors, courses, and trainings.
- Log-in by different types of users, which are employees or instructors.
- View identical profile about logged user.
- Employee can search their interested courses after logged-in.
- Instructor can add new course after logged-in.

1.3. Organization of the rest of the report

Section 2 includes the database used in Django, log-in page and its views function, search bar, "add-course" function, profile page and URL function. In section 3, some pages will be shown to assess this system and instruct readers how to use it. At the end, in section 4, I will discuss about the future works of this project.

2. Detailed features developed

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)

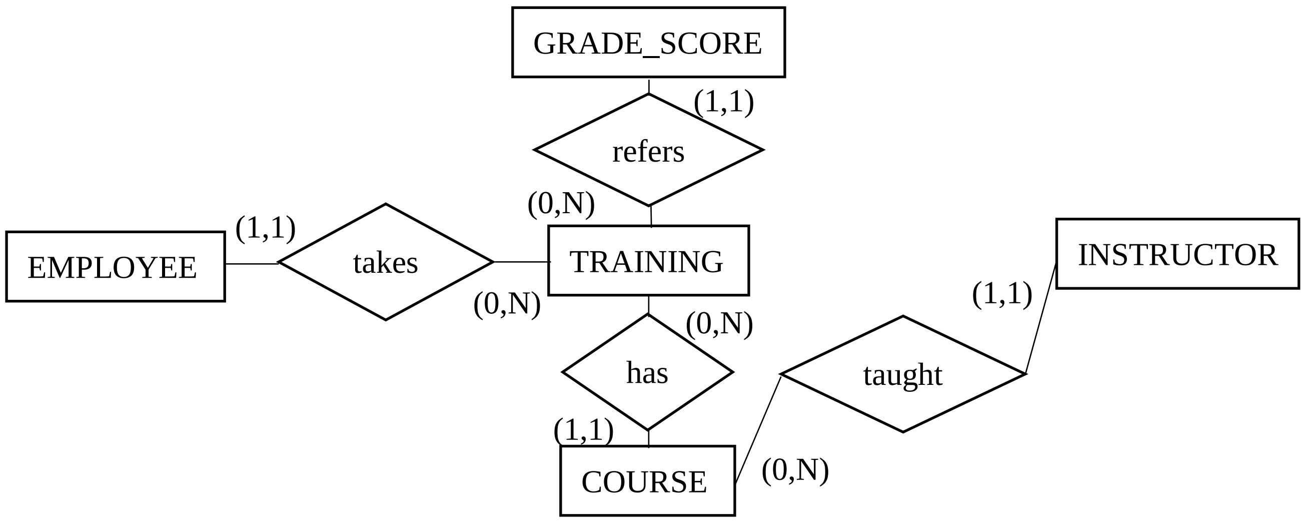
TRAINING (TID, Emp_ID@, Crs_ID@, Grade@)

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')
```

2.3 Profile

By using Django, the attributes inside database can be gotten to the private 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 in Appendix) I firstly split the data as attributes, then save them into class ‘Employee’. All that information will be shown in user’s profile page.

2.4 Search Courses

Search courses that you want to register in next semester is a very useful function for employees. In order to achieve this search bar, I used the Ajax to send search text from the employee page and filtered *Course* objects in the views using “contains” syntax. If the input text matches the title of courses in my database, it will be shown as a URL list under the search bar. By clicking on those lists, logged employee can view the details, such as level, instructor, and description, of that course. Following is the JavaScript.

```
$(function(){

    $('#search').keyup(function(){
        $.ajax({
            type: "POST",
            url: "/employee/search/",
            data: {
                'search_text' : $('#search').val(),
                'csrfmiddlewaretoken': $("input[name=csrfmiddlewaretoken]").val()
            },
            success: searchSuccess,
            dataType: 'html'
        });
    });
});

function searchSuccess(data, textStatus, jqXHR)
{
    $('#search-results').html(data);
}
```

To apply it to webpage, the Html codes are shown as follows:

```
{% if Courses.count > 0 %}

{% for Course in Courses %}
    <li><a href="/employee/get/{{ Course.id }}/">{{ Course.crs_Title }}</a></li>
{% endfor %}

{% else %}

<li>None to show!</li>

{% endif %}
```

2.5 Add New Course

For instructors, they may need to add new courses to this system's database. The instructor user will type in the title, level, and description of new courses. Also, the instructor's information of this course will be assigned to this course automatically. The "addCourse" function in the "views.py" is shown as follows, and the JavaScript codes are shown in Appendix.

2.6 URL

Finally, to accomplish login function, I create the URL patterns for connecting URLs of 'log-in', 'log-out', 'logged-in' pages and so on to Django framework (shown below). Also, static files, such as CSS style files and JavaScript files are connected to those pages. It should be noted that the "courseDetail" view function, which direct to a new page showing the details of each course, is a dynamic URL. They are different with different course ID defined in my database.

```
urlpatterns = [
    url(r'^login/$', views.login, name='login'),
    url(r'^auth/$', views.auth_view, name='auth_view'),
    url(r'^logout/$', views.logout, name='logout'),
    url(r'^employee/$', views.employee, name='employee'),
    url(r'^instructor/$', views.instructor, name='instructor'),
    url(r'^invalid/$', views.invalid_login, name='invalid_login'),
    url(r'^loadEmployee/$', views.employeeRequest, name='loadEmployee'),
    url(r'^loadInstructor/$', views.instructorRequest, name='loadInstructor'),
    url(r'^employee/search/$', views.search_Course, name='crouseSearch'),
    url(r'^employee/get/(?P<Course_id>\d+)/$',views.courseDetail, name='course'),
    url(r'^addCourse/', views.addCourse, name='addCourse'),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

3. Application Assessment

Figure 1 shows the ‘log-in’ page for this system.

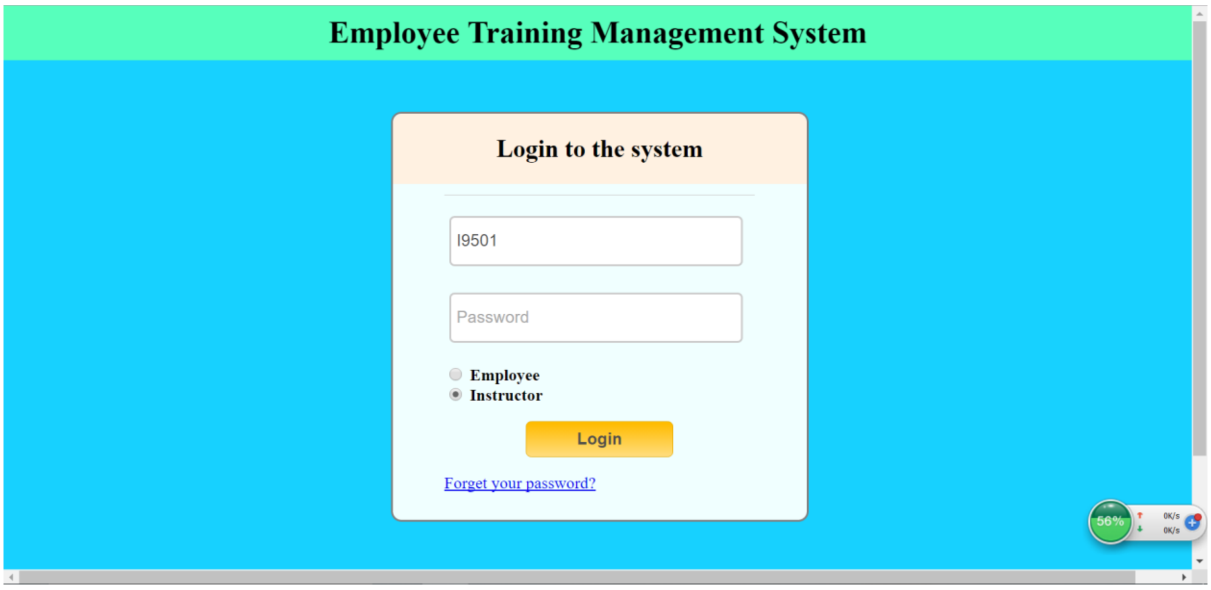
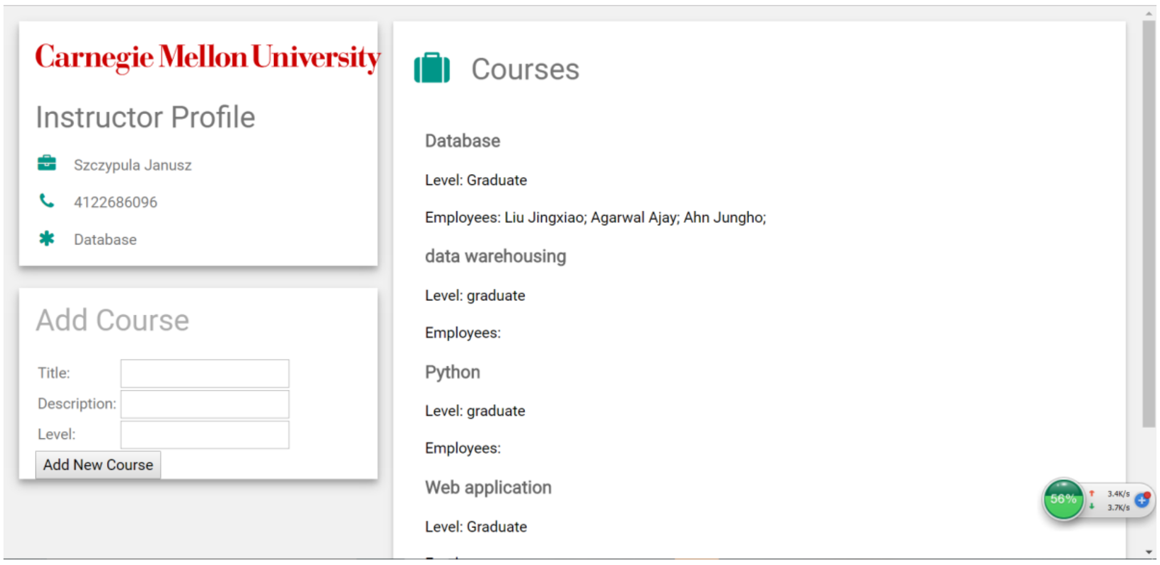


Figure 1: Log-in Page.

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

Figure 2 and 5 are one employee and one instructor’s profile page (Template of this page is from W3school), information is gotten from databased established by Django.


Carnegie Mellon University

Instructor Profile

Szczypula Janusz

4122686096

Database

Add Course

Title:
Description:
Level:

Courses

Database

Level: Graduate

Employees: Liu Jingxiao; Agarwal Ajay; Ahn Jungho;

data warehousing

Level: graduate

Employees:

Python

Level: graduate

Employees:

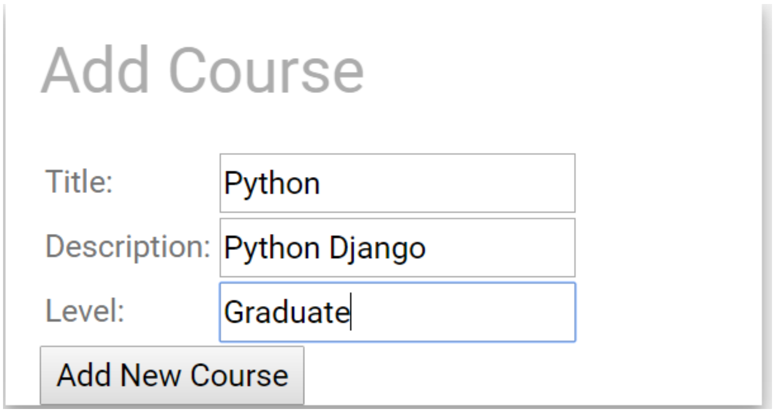
Web application

Level: Graduate

Figure 2: One example of Instructor Profile page.

There are also three functions in instructor's private page: Instructor Profile, Courses, Add Course. If logged instructor type in information about new course and click the button to add it into database, they can view it in the "Courses" part after new logging.

For example, if instructor add a new course named as "Python", with some description and "Graduate" level. (shown in Figure 3) The database will have a new row as what we have added. (shown in Figure 4)

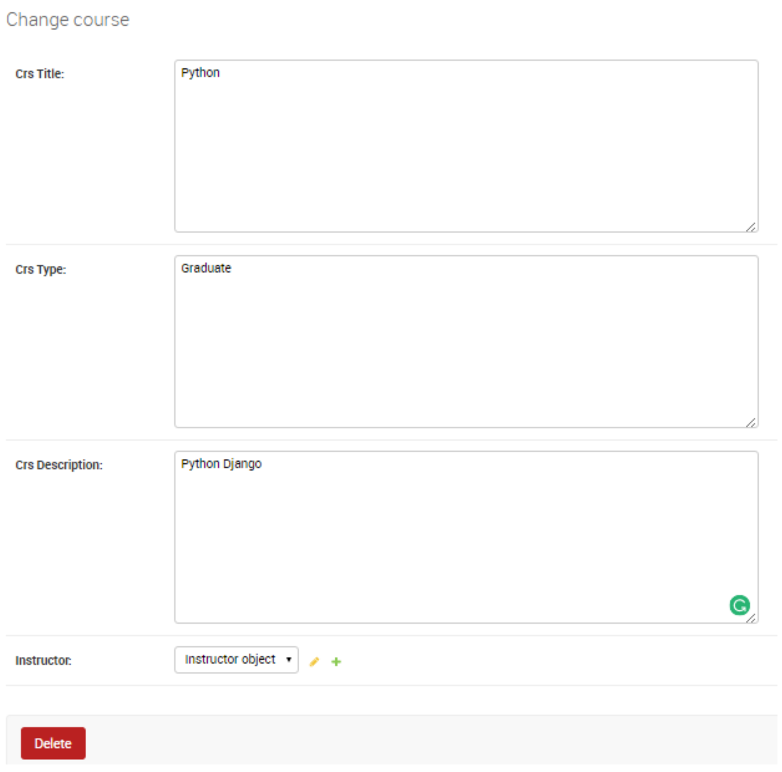

Add Course

Title: Python

Description: Python Django

Level: Graduate

Figure 3: Add new course example.



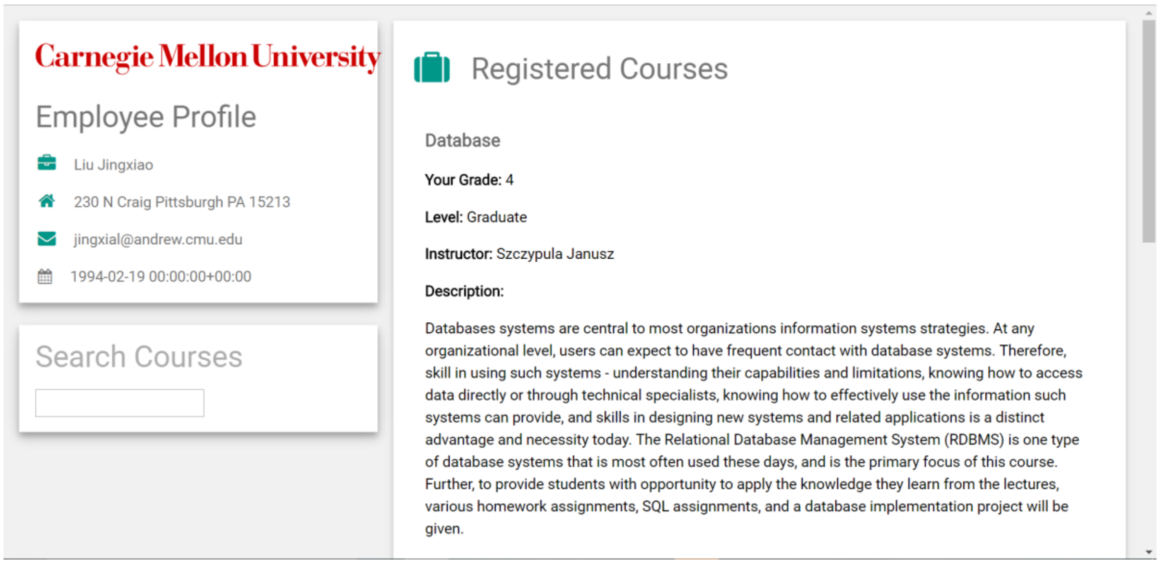
Change course

Crs Title:	Python
Crs Type:	Graduate
Crs Description:	Python Django
Instructor:	Instructor object

Delete

Figure 4: New added course in Django database.

There are three functions in the employee's page: Employee Profile, Search Courses, and Registered Courses. "Employee Profile" part can show users their private information got from the database. The "Registered Courses" part shows information about the registered courses, such as grade, level, instructor, description. And the "Search Courses" part can help employee to search courses by title in the database, Figure 6 shows the result by searching "database".



Carnegie Mellon University

Employee Profile

Liu Jingxiao

230 N Craig Pittsburgh PA 15213

jingxial@andrew.cmu.edu

1994-02-19 00:00:00+00:00

Search Courses

Registered Courses

Database

Your Grade: 4

Level: Graduate

Instructor: Szczypula Janusz

Description:

Databases systems are central to most organizations information systems strategies. At any organizational level, users can expect to have frequent contact with database systems. Therefore, skill in using such systems - understanding their capabilities and limitations, knowing how to access data directly or through technical specialists, knowing how to effectively use the information such systems can provide, and skills in designing new systems and related applications is a distinct advantage and necessity today. The Relational Database Management System (RDBMS) is one type of database systems that is most often used these days, and is the primary focus of this course. Further, to provide students with opportunity to apply the knowledge they learn from the lectures, various homework assignments, SQL assignments, and a database implementation project will be given.

Figure 5: One example of Employee Profile page.

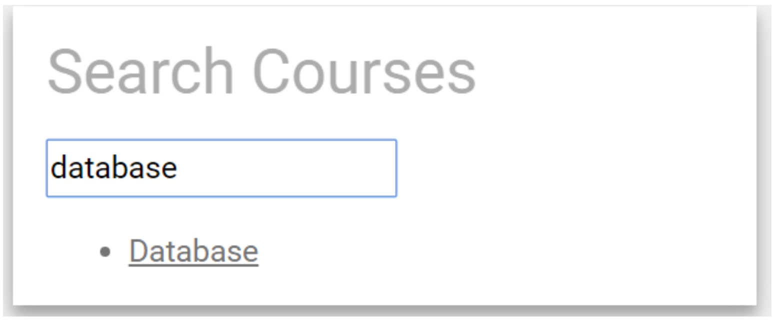


Figure 6: An example about search bar.

4. Future Work

There are several works in the future. The first one is the grading function, which will help instructor to do grading on registered students. It should refer to the courses and employees, and can be shown in employees' private page. Another work in the future is to add the registration function, this function is located in employee's private page and help them to register new course containing in my database. At the end, one new type of users, system manager, should be added in the future. They can manage the database of employee, instructor and course, even not enter the administration system of Django.

Appendix:

employee.js:

```
function Employee(na, em, dob, hd, add, ps)
{
    this.name = na;
    this.email = em;
    this.DOB = dob;
    this.hiredate = hd;
    this.address = add;
    this.password = ps;
}
var employees = [];
var userID = document.title;
function initEmployees()
{
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function()
    {
        if(this.readyState == 4 && this.status == 200)
        {
            var data = this.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],attributeStrings[5],attributeStrings[6])
                employees.push(newEmployee);
            }
            index = findIndex(employees, userID);
            document.getElementById("name").innerHTML = employees[index].name;
            document.getElementById("address").innerHTML =
employees[index].address;
            document.getElementById("email").innerHTML = employees[index].email;
            document.getElementById("DOB").innerHTML = employees[index].DOB;
        }
    };

    xhttp.open("GET", "/loadEmployee/",true);
    xhttp.send();
}

function findIndex(employeeClass, userID)
{
    for(i=0; i<employeeClass.length; i++)
    {
        if(employeeClass[i].password == userID)
        {
            return i;
        }
    }
}
```

instructor.js:

```
function Instructor(na, ph, sp, ps)
{
    this.name = na;
    this.phone = ph;
    this.speciality = sp;
    this.password = ps
}
var instructors = [];
var userID = document.title;
function initInstructor()
{
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function()
    {
        if(this.readyState == 4 && this.status == 200)
        {
            var data = this.responseText;
            var instructorStrings = data.split(";");

            for(var i=0; i<instructorStrings.length; i++)
            {
                var instructorString = instructorStrings[i];
                var attributeStrings = instructorString.split(",");
                var newinstructor = new
Instructor(attributeStrings[0],attributeStrings[1],attributeStrings[2],attributeStr
ings[3]);
                instructors.push(newinstructor);
            }

            index = findIndex(instructors, userID);
            document.getElementById("name").innerHTML = instructors[index].name;
            document.getElementById("phone").innerHTML = instructors[index].phone;
            document.getElementById("speciality").innerHTML =
instructors[index].speciality;
        }
    };
    xhttp.open("GET", "/loadInstructor/",true);
    xhttp.send();
}

function findIndex(Class, userID)
{
    for (i = 0; i < Class.length; i++) {
        if (Class[i].password == userID) {
            return i;
        }
    }
}

function addCourse()
{
    var phone = document.getElementById("phone").innerHTML;
    newxhttp = new XMLHttpRequest();
    var title = document.getElementById("title").value;
    var description = document.getElementById("description").value;
    var level = document.getElementById("level").value;
    if(title=='')
    {
        alert("Not valid.");
    }
    else
    {
        var parameter =
"?title="+title+"&description="+description+"&level="+level+"&phone="+phone;
        newxhttp.open("GET", "/addCourse/"+parameter, true);
        newxhttp.send();
        alert("Successful.");
    }
}
```

login.html

```
{% 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="type">
              <label>
                <input type="radio" name="type" value="employee" checked>
                Employee<br>
                <input type="radio" name="type" value="instructor">
                Instructor
              </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 %}
```

employee.html

```
<!DOCTYPE html>
<html>
<title>{{ userID }}</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="http://www.w3schools.com/lib/w3.css">
<link rel='stylesheet' href='https://fonts.googleapis.com/css?family=Roboto'>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.6.3/css/font-awesome.min.css">
<style>
html,body,h1,h2,h3,h4,h5,h6 {font-family: "Roboto", sans-serif}
</style>
<script type="text/javascript"
src="../static/EmployeeTraining/employee.js"></script>
<script type="text/javascript" src="../static/EmployeeTraining/jquery-
```

```
1.11.3.min.js"></script>
<script type="text/javascript" src="../static/EmployeeTraining/ajax.js"></script>
<body class="w3-light-grey" onload="initEmployees()">

<!-- Page Container -->
<div class="w3-content w3-margin-top" style="max-width:1400px;">

  <!-- The Grid -->
  <div class="w3-row-padding">

    <!-- Left Column -->
    <div class="w3-third">

      <div class="w3-white w3-text-grey w3-card-4">
        <div class="w3-container">
          <h1><img id="wordmarkPrint" src="//www.cmu.edu/cmu-design-
2015/images/cmu-wordmark.png" width="350px"></h1>
          <h2>Employee Profile</h2>
          <p><i class="fa fa-briefcase fa-fw w3-margin-right w3-large w3-text-
teal"></i><label id="name"></label>
          <p><i class="fa fa-home fa-fw w3-margin-right w3-large w3-text-
teal"></i><label id="address"></label>
          <p><i class="fa fa-envelope fa-fw w3-margin-right w3-large w3-text-
teal"></i><label id="email"></label>
          <p><i class="fa fa-calendar fa-fw w3-margin-right"></i><label
id="DOB"></label>
        </div>
      </div><br>

      <div class="w3-white w3-text-grey w3-card-4">
        <div class="w3-container">
          <h2 class="w3-opacity">
              Search Courses
          </h2>{% csrf_token %}
            <input type="text" id="search" name="search"/>
            <ul id="search-results">
            </ul>
        </div>
      </div><br>

    <!-- End Left Column -->
    </div>

    <!-- Right Column -->
    <div class="w3-twothird">

      <div class="w3-container w3-card-2 w3-white w3-margin-bottom">
        <h2 class="w3-text-grey w3-padding-16"><i class="fa fa-suitcase fa-fw w3-
margin-right w3-xxlarge w3-text-teal"></i>Registered Courses</h2>
        <div class="w3-container">
          {% for Training in Trainings %}
              {% if Training.EmployeeTraining.empPassword == userID %}
                  <h5 class="w3-
opacity"><b>{{ Training.Course.crs_Title }}</b></h5>
                  <p><b>Your Grade: </b>{{ Training.GradeScore.score }}</p>
                  <p><b>Level: </b>{{ Training.Course.crs_Type }}</p>
                  <p><b>Instructor: </b>{{ Training.Course.Instructor.instrLname }}
{{ Training.Course.Instructor.instrFname }}</p>
                  <p><b>Description:
</b></p><p>{{ Training.Course.crs_Description }}</p>
              {% endif %}
          {% endfor %}
        </div>
    <!-- End Right Column -->
    </div>

  <!-- End Grid -->
  </div>
```

```
</div>
</div>
</body>
</html>
```