CptS 322 - Fall 2019
Term Project

I. Overview

For your term project in CptS322, you will build a web application for assigning teaching assistants to courses and labs, based on their course preferences. (see Section -2 for the project description). Alternatively, you may propose your own project idea (contingent on the approval of instructor).

- You will be working in **teams of 3 to 4 student**s. This will allow you to experience how to work as a team, and will also give you an idea of the issues that arise in software development groups. Each team will be able to choose the development languages, environments and tools that they want to use. You are welcome to use the same framework you used for your warm-up project.
- One member in your team should serve as the team **liaison** – not necessarily the leader, but will be responsible for the regular communications of your team with the course staff.
- Each team will work on refining the requirements, constructing the specification, designing the architecture and the individual modules, and of course the implementation and testing, both manual and automated.
- Your team will **present** the final project to the class at the end of the semester and will share the lessons they learned from the project.
- You will also **demonstrate** your final project to the instructor and the course TAs.
- Your individual performance will be **peer-reviewed** by your team mates.
- The project is structured in several stages, as described in Project Assignments (Section-IV).

II. Project Description

Every semester, the School of EECS at WSU recruits undergraduate teaching assistants for the introductory level courses and lab sections. This process has been carried out mostly manually. First, students who are willing to serve as TAs filled out a survey where they specify their course preferences. The faculty who is in charge of the TA assignments looked at the survey data and exchanges emails with the teaching faculty to gather their preferences of the TAs. Then he manually matches TAs to courses and labs based on student and faculty preferences. This is a very tedious process and is very time consuming for both instructors and administrative faculty. You will build a web application that:

- allows students to enter their contact information and course preferences for teaching assistantships,
- allows instructors to choose their teaching assistants among the students who showed interest in their courses.

Your application should have a “student page” and an “instructor page”.
On the student page, a student user can:

1. Create a student account and set the account password (user name should be the WSU email)
2. Login with username and password
3. Enter contact information (name, last name, WSU ID, email, phone)
4. Enter additional information (major, cumulative GPA, expected graduation date, whether s/he served as a TA before)
5. Enter course preferences for the TAship. A student can specify more than one course for her/his preferences. For each course the student should enter:
    a. the course number (such as CptS 121)
    b. the grade s/he earned when s/he took the course (e.g., the grade earned for CptS121)
    c. the year and semester s/he took the course
    d. the year and semester s/he applies for TAship
    e. whether s/he served as a TA for this course before (yes/no)

    Student users can add/remove courses to/from their list of preferred courses.
6. Save the entered information.

On the instructor page, a faculty user can:

1. Create an instructor account and set the account password (user name should be the WSU email)
2. Login with username and password
3. Enter contact information (name, last name, WSU ID, email, phone)
4. Enter the courses s/he teaches currently. Of course an instructor can teach more than one course and /or lab section during a semester. (If a particular course has more than one lab section, each of those section should be entered as a separate course. For example: CptS121-Lab1, CptS121-Lab2…etc.)
5. See the list of the students who applied for TAship. The list should show whether the students have already assigned to a course.
6. The instructor can select a student (who is not assigned to a course yet) and assign it to one of the courses/labs s/he teaches. S/he may also remove the current TA assignments for his courses/labs.

III. Data Models:

In the warmup project your databases had only two tables (models) to store the professor data and reviews. In the "Teaching Assistant" project, you will need several tables to store the students' information, instructors' information, the courses instructors teach, student assignments to courses, encrypted login information, etc.
Below are possible (basic) schemas you may use. You may need to add more tables in your schema.

1. Student Information Table (Students):
2. Instructor Information Table (Instructors):
3. Courses that instructors teach (InstructorCourse):
4. Student assignments to courses (StudentTA)
5. Encrypted login information (Logindata)

IV. Project Timeline:

i. Requirements document: 6%

1. Once we form the teams you will have to start right away getting the team working together and collecting the requirements. In this stage you only need to elaborate the details of the requirements for the complete project.
2. You must start using GitLab to manage your plan.
3. Since there are only couple weeks after you form the team until the full Requirements and Specification Doc is due, it is important for the team to start meeting right away, because it takes time to figure out what the team will be building.

ii. **Iterations:** The main part of the project will be done in several two-week iterations. In each iteration you will have to work on several tasks:

- **Iteration 1**: (8%)
  - First draft of your Design Document (5%)
    - Write a short progress report at the end of your design document where you briefly summarize the project progress.
  - A completion of a running version of your code with a partial set of features. All code should be uploaded to GitLab repo before the iteration deadline. (3%)

- **Iteration 2:** (8%)
  - Update your Design Document to reflect your current accomplishments and your test plans for iteration 2 (3%)
    - Write a short progress report at the end of your design document where you briefly summarize the project progress and tests performed.
  - A completion of a running version of your code with additional set of features. All code should be uploaded to GitLab repo before the iteration deadline. (5%)

- **Iteration 3:** (15%)
  - A final version of your code. All code should be uploaded to GitLab repo before the iteration deadline. (4%)
  - At the end of the semester each team will demo their project to the instructor and the course TA. (8%)
  - At the end of the semester each team will give a 3 min presentation to the class. (1%)
  - Testing support and tests for the implemented features. This iteration should include unit, integration, and functional tests. You should also include some automated tests. All test code should be uploaded to GitLab repo before the iteration deadline. (2%)

- **Project management:** (3%)
  - Your “project management” score will be based on:

- How effectively you used GitLab? (We will use GitLab issues to maintain the project backlog and milestones for sprint planning)
- Has the team members communicated effectively? Do all members have frequent commits throughout the semester?

V. Project Deliverables:

1. Requirements Specifications:

There is no standard for requirements or specifications documents and in fact many organizations blend aspects of requirements, and specification in a single document. We will use a simple template for requirements and specification and document requirements as UML use cases. Inevitably in preparing this document you will describe of how the system should interact with the outside world.

Please see the "Requirements Specifications" document template posted on Blackboard.

2. Design Document:

This will be a living document. For the first iteration you will fill in the document with the design details as you can see them before the first iteration. In subsequent iterations you will expand this document.

A template for the "Design Document" will be available on Blackboard.

3. Progress Report:

For iterations 1 and 2 you will write a short progress report covering the topics specified below. This report will be included at the end of your design document.

- **Main Difficulties:** What were the main difficulties so far? You should consider both technical and organization issues.
- **Features:** What features were completed? Were there any features you did not implement as planned? Are you pushing some features to later iterations, and if so, why?
- **Tests:** What tests did you prepare for this iteration, and what are they covering? What features are you not testing yet? Did you use any test frameworks?

4. Your code and Tests:

In each iteration you need to commit a running version of your code to the iteration branch (i.e., iteration1, iteration2, or iteration3) in your repo. All code should be uploaded to GitLab before the iteration deadline. There should be a link to your GitLab iteration branch in your progress report. In iteration3, you should also commit your test code.

5. Presentation:

At the end of the semester, each team will make a presentation to the whole class in one of the lecture slots. The goal is to allow the whole class to learn from the experience of all groups.

Each presentation should be delivered by the entire team. The presentation will be graded based on the course staff evaluation of how well prepared and useful the presentation is, along with the peer feedback submissions.

The presentation should be about **3 minutes** long and should have 2 parts:

- An **overview of your project**, perhaps with a very short demo of the current state. The chance of a demo breaking increases with the number of people in the audience. We recommend recording the demo as a backup.
- Some of the **lessons you have learned**. Please do not be afraid to say what went wrong; being frank and drawing educational conclusions from your experience is what we are looking for.

VI. Submission Instructions:

All project documents including "Requirements Specifications" and "Design Document" (both versions) should be committed to your team's GitLab repo.

All code and test case implementations should also be committed to your GitLab repo.

Your GitLab repo should have the following branch structure :

- master:
  - Final version of your code (after iteration 3);
  - (under "tests" directory) all your test code;
  - (under documents directory) Requirements Specifications document; final version of your Design Document.
- iIterationX:
  - All the code you produced in iteration X;
  - Iteration1 and Iteration 2 only - (under documents directory) Version X of your Design Document;
  - Iteration3 only - (under "tests" directory) your test code for iteration 3;

Appendix – A:

Below are the expected features by iteration:

Iteration 1:

Frontend:

Have the mock interfaces for:

- (Instructor) create account
- (Student) create account
- (Instructor/Student) login
- (Instructor) view profile
- (Student) view profile
- (Instructor) add course

Frontend/JS:

- Get the user input for creating an instructor account and initiate the POST request.
- Get the user input for creating an student account and initiate the POST request.

Backend:

- Route to create a new instructor profile (POST)
- Route to create a new student profile (POST)

Iteration 2:

Frontend:

Revise the user interfaces from the previous iterations.

Have the mock interfaces for:

- (Instructor) edit profile
- (Student) edit profile
- (Instructor) list all courses
- (Instructor) select the courses instructor teaches

Frontend/JS:

- Get the user login information, encrypt the user credentials and initiate the POST request for login.
- Initiate a GET request to get an instructor's profile and display the profile data
- Initiate a GET request to get a student's profile and display the profile data
- Allow instructor/student edit his/her profile and initiate a POST request to update his/her profile data
- Allow instructor to view the list of all courses; initiate a GET request to retrieve the list of about all CptS courses.
- Allow instructor choose courses and initiate a POST request to add (a) new course(s) to the instructor profile

Backend:

- Route to verify login (GET/POST)
- Route to retrieve instructor profile data (GET)
- Route to retrieve student profile data (GET)
- Route to retrieve list of all CptS courses (GET)
- Route to add new courses for the instructor (POST).

Iteration 3:

Frontend:

Revise the user interfaces from the previous iterations.

Create the interfaces for:

- (Student) list available TA positions and apply for an open TA position
- (Instructor) display all TA applications
- (Instructor) approve TA applications
- (Student) view status of (own) TA applications
- (Student) cancel (own) TA application

Frontend/JS:

- (student) Initiate a GET request to retrieve the courses that are not assigned to TAs yet.
- (student) Initiate a POST request to submit a TA application.
- (instructor) Initiate a POST request to approve the TAship application of a student for a course.
- (instructor) Initiate a GET request to get the list of the TAs assigned to an instructor's classes.
- (student) Initiate a GET request to get the status of a student's TA applications.
- (student) Initiate a DELETE request to cancel/remove a TA application

Backend:

- (instructor) Route to retrieve the courses that are not assigned TAs yet (GET)
- (student) Route to submit a TA application (POST) You may simply add the ID of the student and course name the student is assigned to in the CourseTA table. Initially the application status for the TA assignment should be "under review".
- (instructor) Route to update the status code of a TAs application as confirmed (POST) (i.e., update the status of an application in the courseTA table as "approved".) Assume that a student may be the TA for a single class only. So, the other applications of the student should be automatically assigned as "dismissed" when a student is assigned to a course.
- (instructor) Route to get the list of the TAs assigned to an instructor's classes. (GET)
- (student) Route to get the status of a student's TA applications. (GET)
- (student) Route to remove an application from CourseTA (DELETE)