Be My TA

Design Document

10/25/2019

Version 1.0

BE My TA

Drew McLaurin, Diwash Biswa, Dayton Dekam

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Introduction

This project is designed to solve the current problem of recruiting TAs by creating an online website that allows students to apply to be a TA by entering their course(s) of interest, contact information, and any requirements that help the professor to determine whether that student will be a good fit or not. This website also gives the professors access to look at all the students who have applied to be their course TA and see all their application info to determine if they are a qualified candidate. Then the professor can choose which students they want to be their new TA on the website. This application will eliminate the need for extra faculty to review and assign a TA's for each professor. And it will save time for the professor by being able to look at each application on the same website.

Section II includes the overview of our architectural design for our project.

Section III includes more details of our design including the backend attributes for our project.

Document Revision History

Rev 1.0 2019-08-25 Initial Version

II. Architecture Design

II.1. Overview

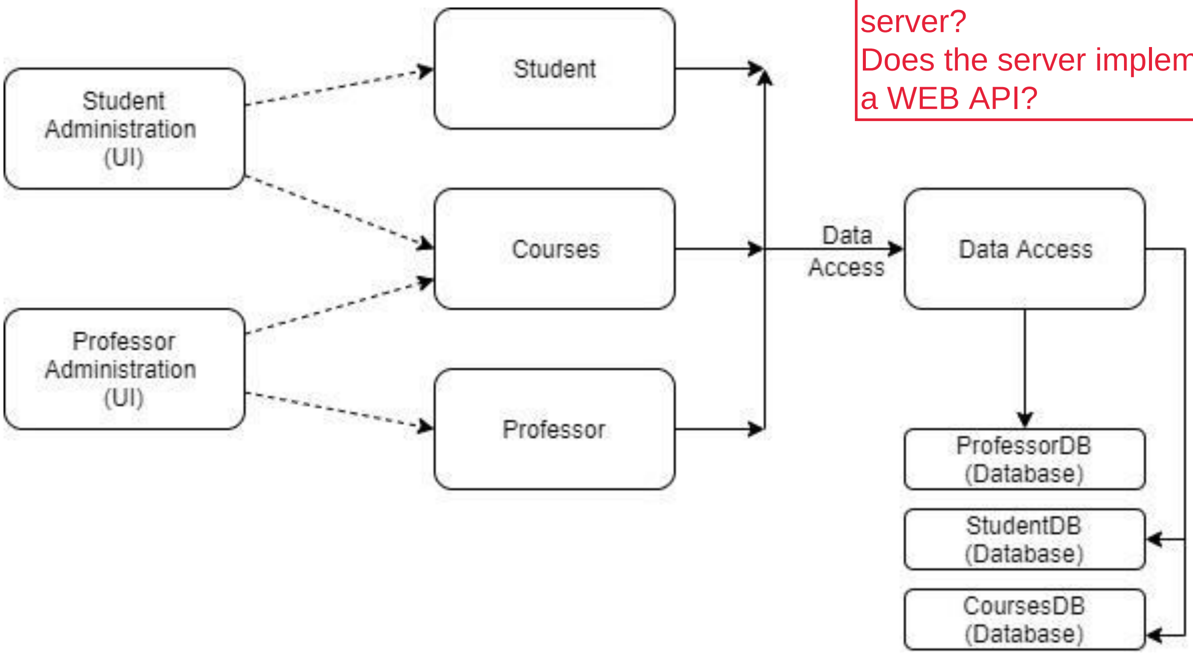
For our project we are doing the client-server architectural pattern. We started with the frontend work that includes the user application or client. We will begin the backend work in the next iteration where we will create three databases which we can access and manipulate the information logged from the user by using GET and POST requests.

In our architecture design we have few interdependencies between components. So, to reduce coupling, we have created separate database for professor and students so when the professor database is modified, student database is unaffected. And to maximize cohesion, both the student and professor can use the similar or related classes to retrieve the course information from the database.

Component Diagram UML:

not a UML diagram

Which component is the server here?
What is the interface of the server?
Does the server implement a WEB API?



- The professor component represents all the information that needs to be stored in the backend for the professor account. His name, email, password, courses he teaches, etc. will all be stored in a database. The student component is very similar to the professor component. All the student information will be stored in a database as well. The courses component represents all the preferred courses each student will have. These courses will be stored in the backend as well. None of our components interact with outside components.

III. Design Details

III.1. Subsystem Design

This section provides more detail about each subsystem in your architecture. For each subsystem, include a sub-section and explain:

We will be using Model View Controller (MVC) to organize the client JavaScript code.

Model: contains the methods for accessing user and course information from the backend

View: getting the information from backend to display in the user interface

Controller: methods for getting inputs from the user interface and connecting the actions to the backend server

Communication between client and server: backend route:

URLs for the routes ?

Login: POST request with username(wsu email) and password

```
{
    “Username”: “first.last@wsu.edu”,
    “Password”: “StrongPassword”
}
```

Create Account Student: POST request with user information

```
{
    “Name”: “First Last”,
    ID: 12354,
    “Email”: “first.last@wsu.edu”,
    “Password”: “StrongPassword”,
    “Verify”: “StrongPassword”,
    “Major”: “Computer Science”,
    “GPA”: 4.0,

    “Graduation_Date”: “Month Year”

}
```

Create Account Professor: make POST request with given information

```
{

        “Name”: “First Last”,

        “Email”: “first.last@wsu.edu”,

        “Password”: “StrongPassword”,

        “Verify”: “StrongPassword”,

}
```

Apply for TA (Student application): Make POST request with given information

```
{

        “Courses 1”: “Course ID”:

        “Grade in Course”: “A”

        “Semester Taken”: “Spring 2018”

        “Semester to TA”: “Spring 2020”

        “Served as TA”: “No”

        “STATUS”: “Pending”

}
```

View applications for Professor: Make GET request

```
{

        “Name”: “First Last”

        “Courses 1”: “Course ID”:
```

```
        “Grade in Course”: “A”

        “Semester Taken”: “Spring 2018”

        “Semester to TA”: “Spring 2020”

        “Served as TA”: “No”

        “STATUS”: “Pending”

}
```

Professor action (approving or denying applications): Make POST request

```
{

        “Name”: “First Last”

        “Courses 1”: “Course ID”:

        “Grade in Course”: “A”

        “Semester Taken”: “Spring 2018”

        “Semester to TA”: “Spring 2020”

        “Served as TA”: “No”

        “STATUS”: “Approved”

}
```

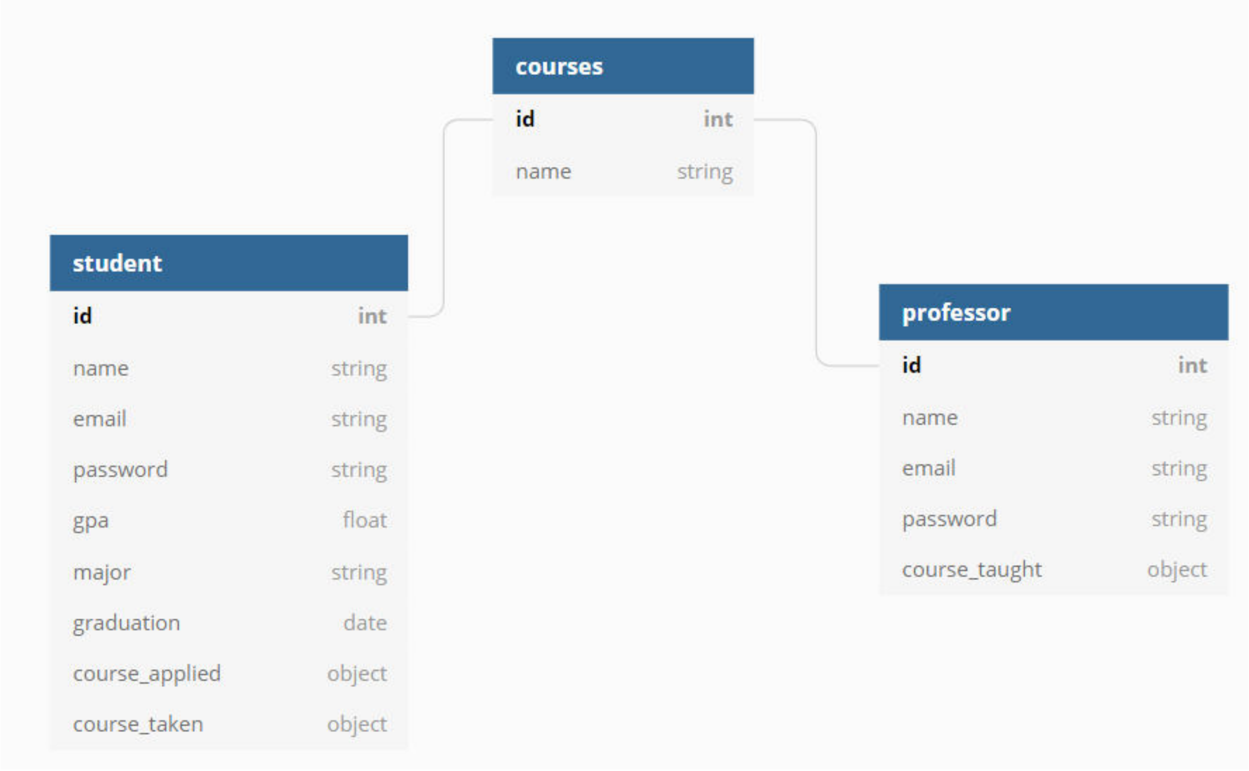
- **(in iteration -2)** Provide your class level design. You should include a UML class diagram visualizing your class level design. In addition, explain each class in detail, specify and explain their methods.

If you have considered alternative designs, please describe briefly your reasons for choosing the final design.

III.2. Data design

Describe the database tables created as part of the application. Provide the schemas (attributes) of the tables.

Database schema:



courses	
id	int
name	string

student	
id	int
name	string
email	string
password	string
gpa	float
major	string
graduation	date
course_applied	object
course_taken	object

professor	
id	int
name	string
email	string
password	string
course_taught	object

Student Database

Name	Type
name	string
password	string
GPA	float
major	string
Graduation date	int
Course IDs Applied	[json object]
Courses taken	[string]

Professor Database

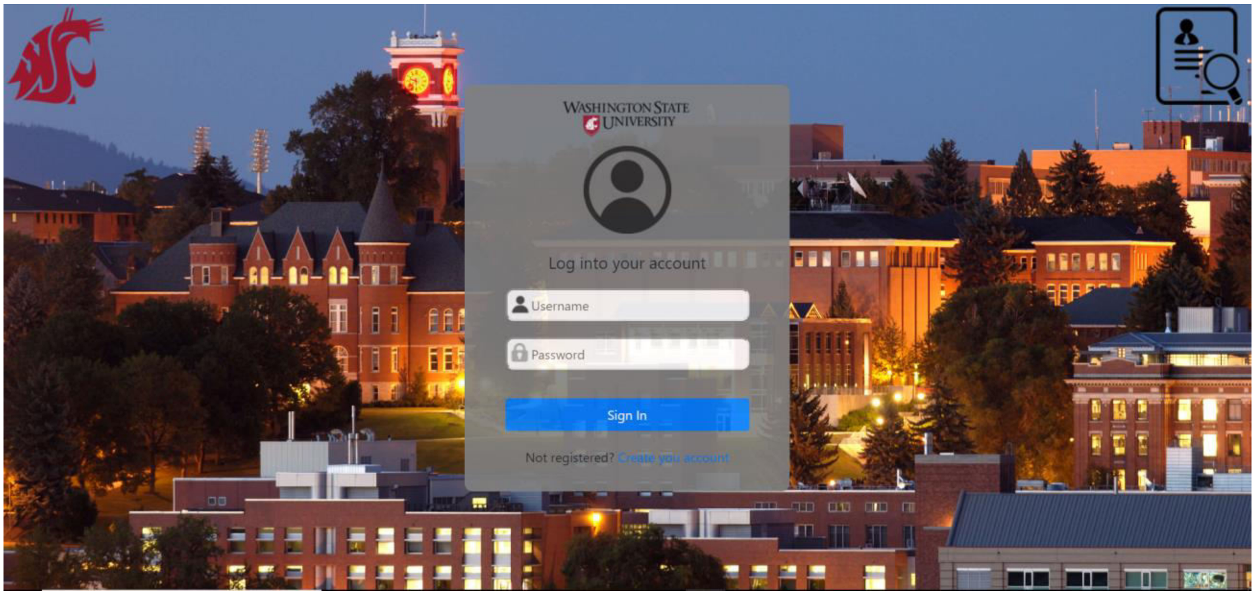
Name	**Type**
name	string
email	string
password	string
Courses they teach	[string]

Courses Database

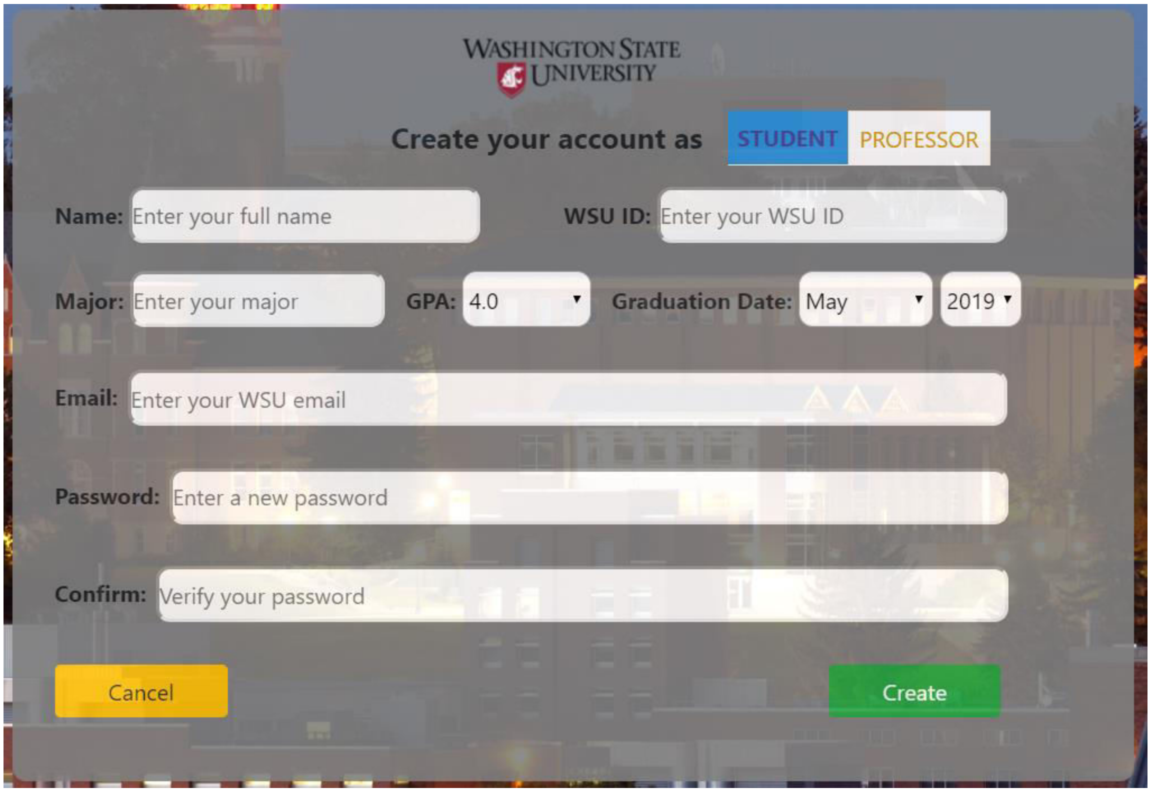
Name	**Type**
Course name	[string]

III.3. User Interface Design

Use Case #3: Login


WASHINGTON STATE UNIVERSITY
Log into your account
Username
Password
Sign In
Not registered? Create you account

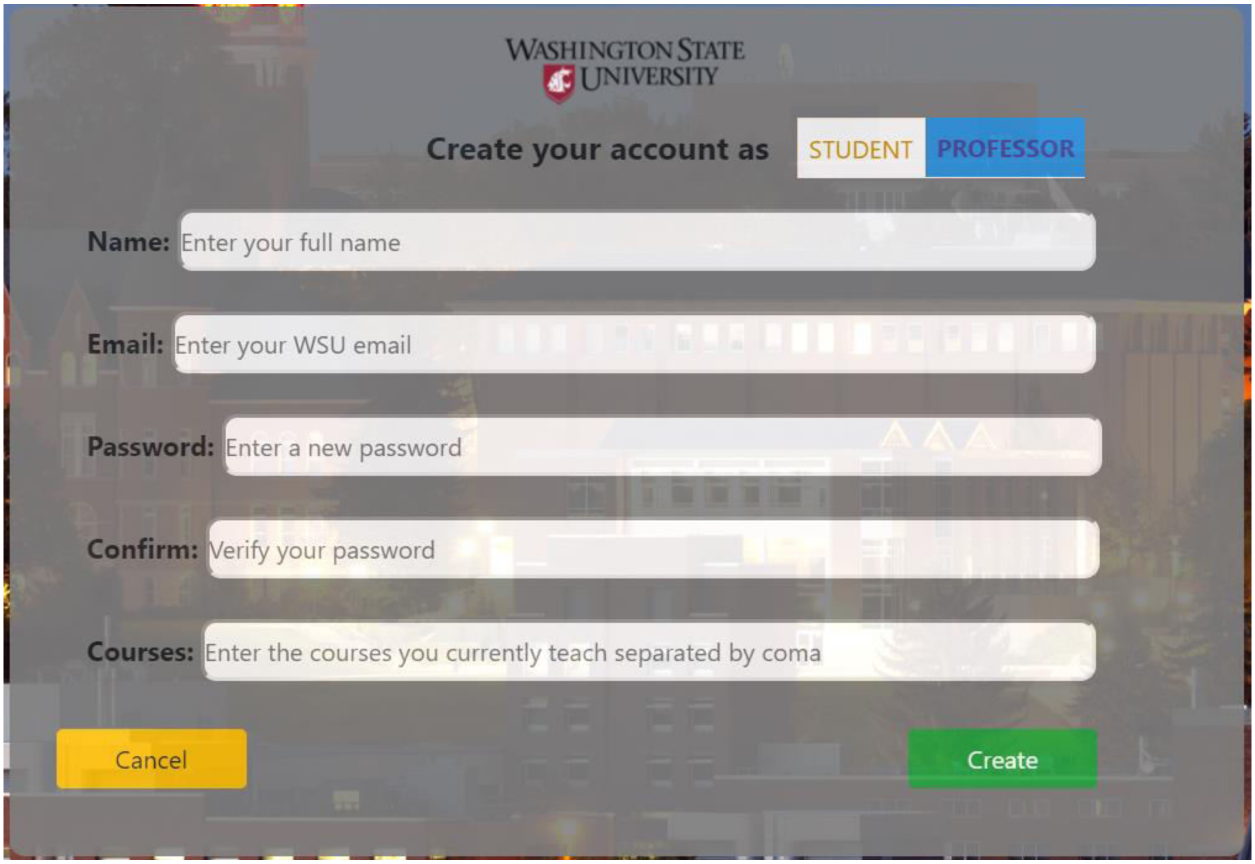
Use Case #1: Create Account Student



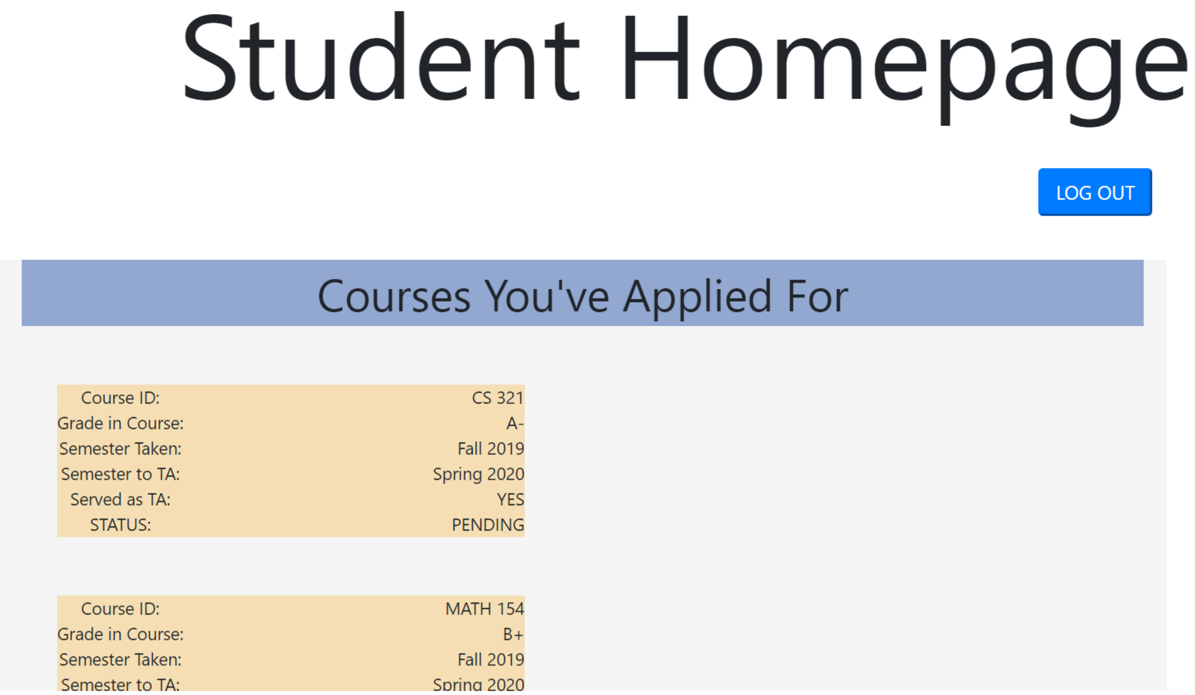
GPA: 4.0

Graduation Date: May 2019

Use Case #2: Create Account Professor


WASHINGTON STATE UNIVERSITY
Create your account as
STUDENT
PROFESSOR
Name: Enter your full name
Email: Enter your WSU email
Password: Enter a new password
Confirm: Verify your password
Courses: Enter the courses you currently teach separated by coma
Cancel
Create

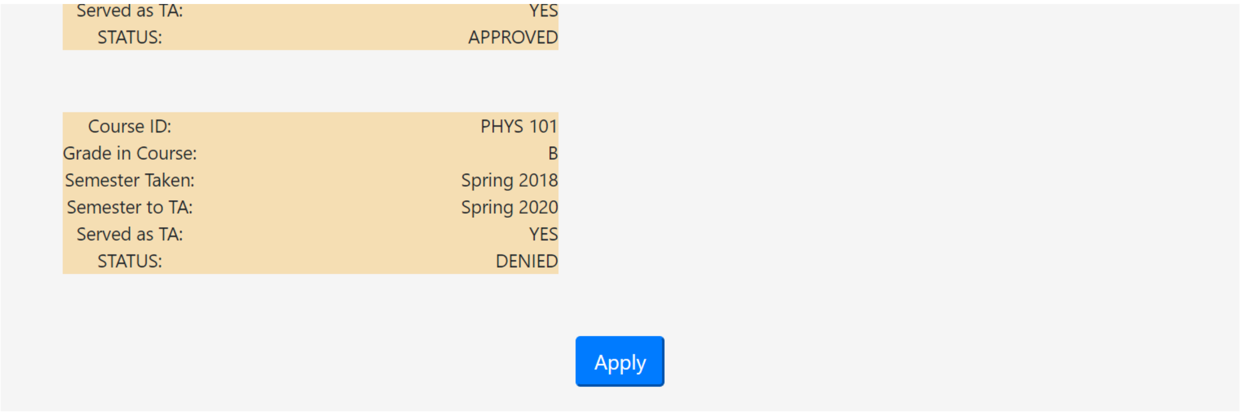
Use Case #4: Student Homepage/Apply for TA position



Courses You've Applied For

Course ID:	CS 321
Grade in Course:	A-
Semester Taken:	Fall 2019
Semester to TA:	Spring 2020
Served as TA:	YES
STATUS:	PENDING

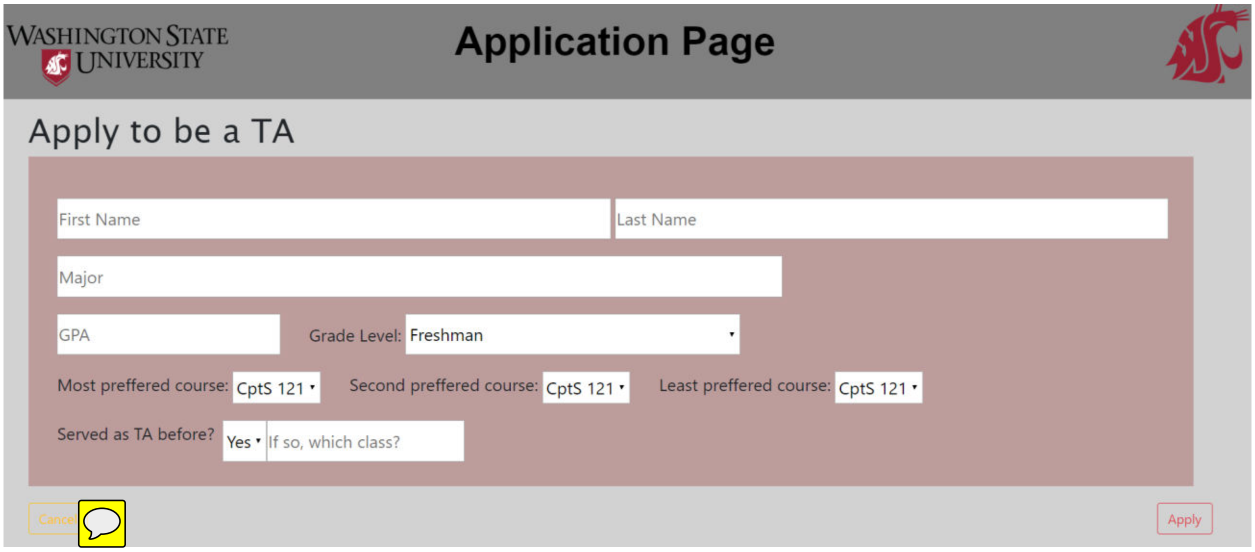
Course ID:	MATH 154
Grade in Course:	B+
Semester Taken:	Fall 2019
Semester to TA:	Spring 2020



Served as TA:	YES
STATUS:	APPROVED

Course ID:	PHYS 101
Grade in Course:	B
Semester Taken:	Spring 2018
Semester to TA:	Spring 2020
Served as TA:	YES
STATUS:	DENIED

Use Case #4: TA Application



Application Page

Apply to be a TA

First Name | Last Name

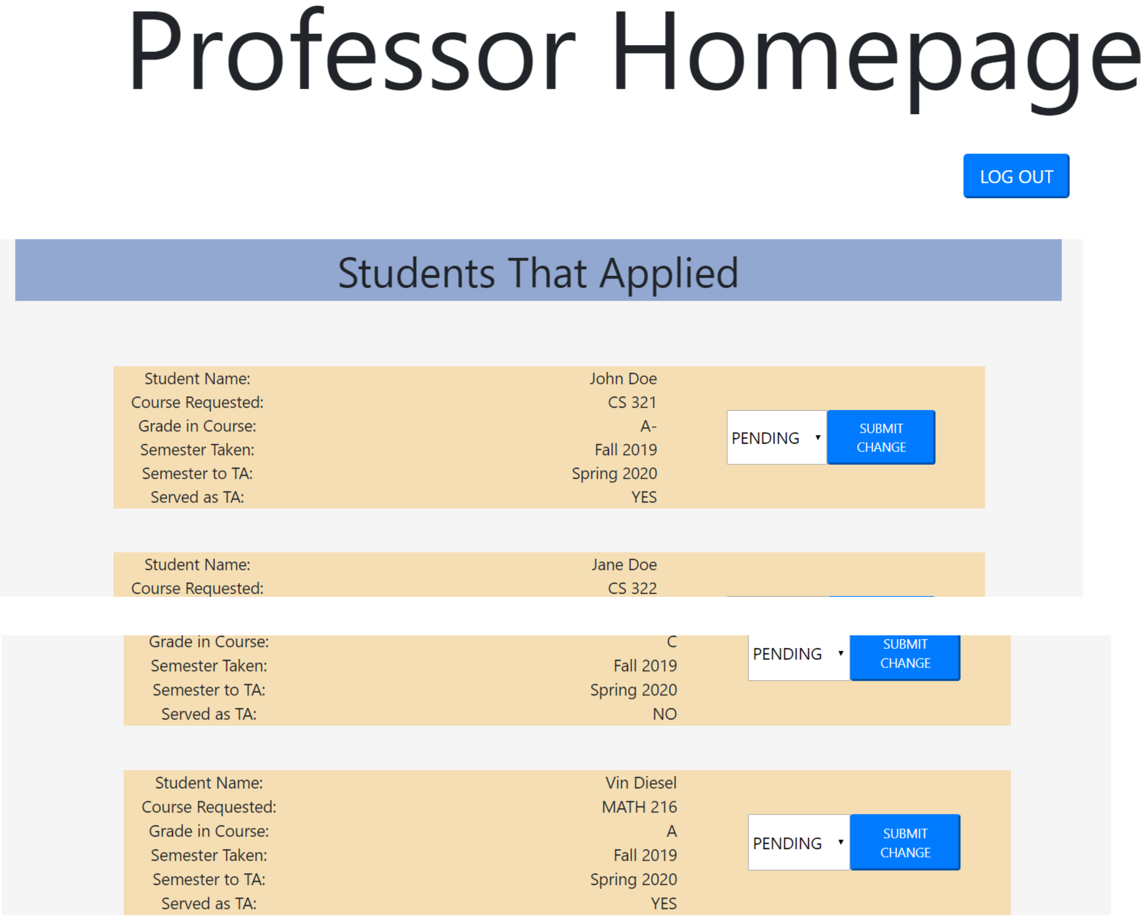
Major

GPA | Grade Level: Freshman

Most preffered course: CptS 121 | Second preffered course: CptS 121 | Least preffered course: CptS 121

Served as TA before? Yes | If so, which class?

Use Case #5: Professor Homepage



Professor Homepage

LOG OUT

Students That Applied

Student Name: John Doe
Course Requested: CS 321
Grade in Course: A-
Semester Taken: Fall 2019
Semester to TA: Spring 2020
Served as TA: YES

PENDING ▾ SUBMIT CHANGE

Student Name: Jane Doe
Course Requested: CS 322
Grade in Course: C
Semester Taken: Fall 2019
Semester to TA: Spring 2020
Served as TA: NO

PENDING ▾ SUBMIT CHANGE

Student Name: Vin Diesel
Course Requested: MATH 216
Grade in Course: A
Semester Taken: Fall 2019
Semester to TA: Spring 2020
Served as TA: YES

PENDING ▾ SUBMIT CHANGE

IV. Progress Report

For Iteration 1, we mostly focused on getting our user interface done. So, we tried our best to complete the fronted using HTML/CSS. I know our UI could be improved a lot and we will be working on it for 2nd and 3rd iterations.

We realized that our approach of just completing the main user interface for iteration 1 was a terrible idea and it didn't comply with our software development method. We wrote in our requirement document that we would use Agile Scrum to implement this project, but we ended up using like a waterfall method for the first iteration. We know we have made this mistake and won't repeat this for iteration 2. We will be using Scrum for iterations 2 and 3. We will also be focusing on completing few features by implementing both frontend and backend and make sure that it fully works as expected.

V. Testing Plan

(in iteration 2)

In this section goes a brief description of how you plan to test the system. Thought should be given to how mostly automatic testing can be carried out, so as to maximize the limited number of human hours you will have for testing your system. Consider the following kinds of testing:

- **Unit Testing:** Explain for what modules you plan to write unit tests, and what framework you plan to use. (Each team should write automated tests (at least) for testing the API routes)

- **Functional Testing:** How will you test your system to verify that the use cases are implemented correctly? (Manual tests are OK)

- **UI Testing**: How do you plan to test the user interface? (Manual tests are OK)

VI. References

No references used other than the project rubric.