Be My TA

Design Document

11/12/2019

Version 1.2

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

Rev 1.1 2019-11-12 Updated during iteration 2

Rev 1.2 2019-11-12 Updated during iteration 3

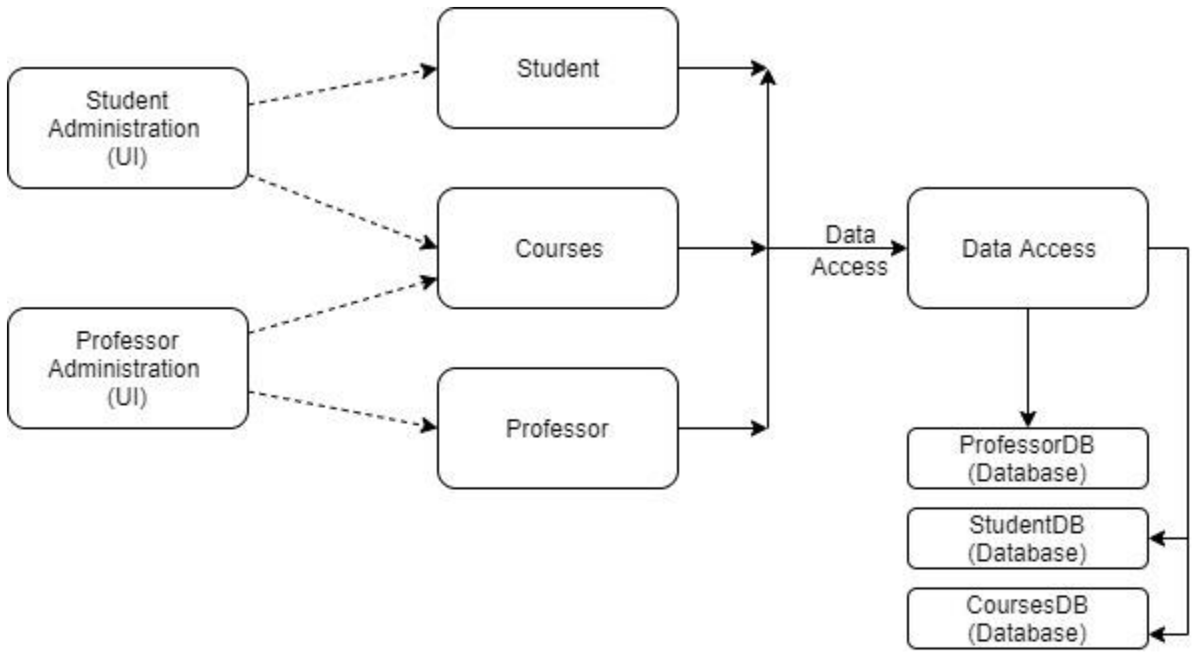
II. Architecture Design

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:



- 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:

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”
}
```

Create Account Professor: make POST request with given information

```
{
        “Name”: “First Last”,
        “Email”: “first.last@wsu.edu”,
        “Password”: “StrongPassword”,
}
```

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

```
{
        “Major”: “Computer Science”,
        “GPA”: 4.0,
        “Graduation_Date”: “Month Year”
        “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”

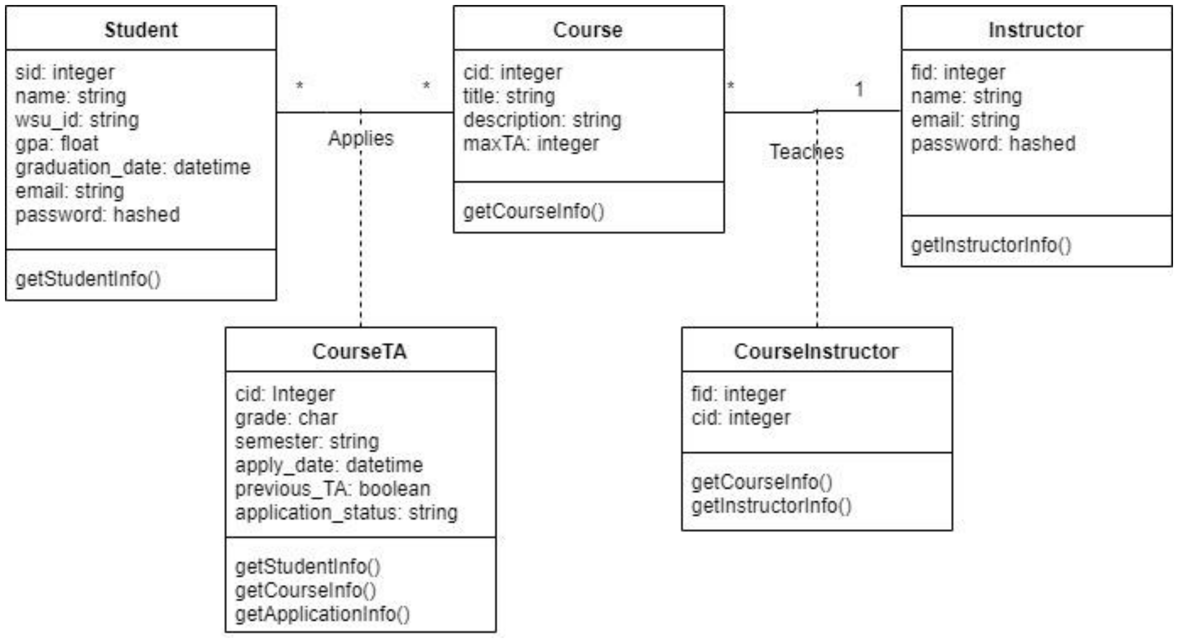
“Semester to TA”: “Spring 2020”

“Served as TA”: “No”

“STATUS”: “Approved”

}

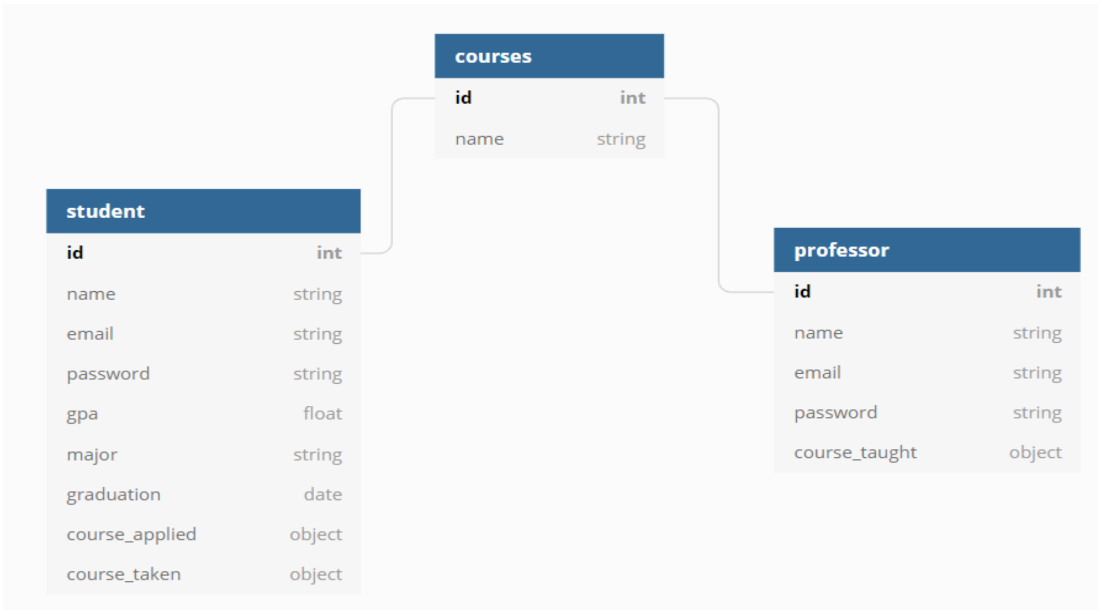
UML Class Diagram (updated in iteration 2):



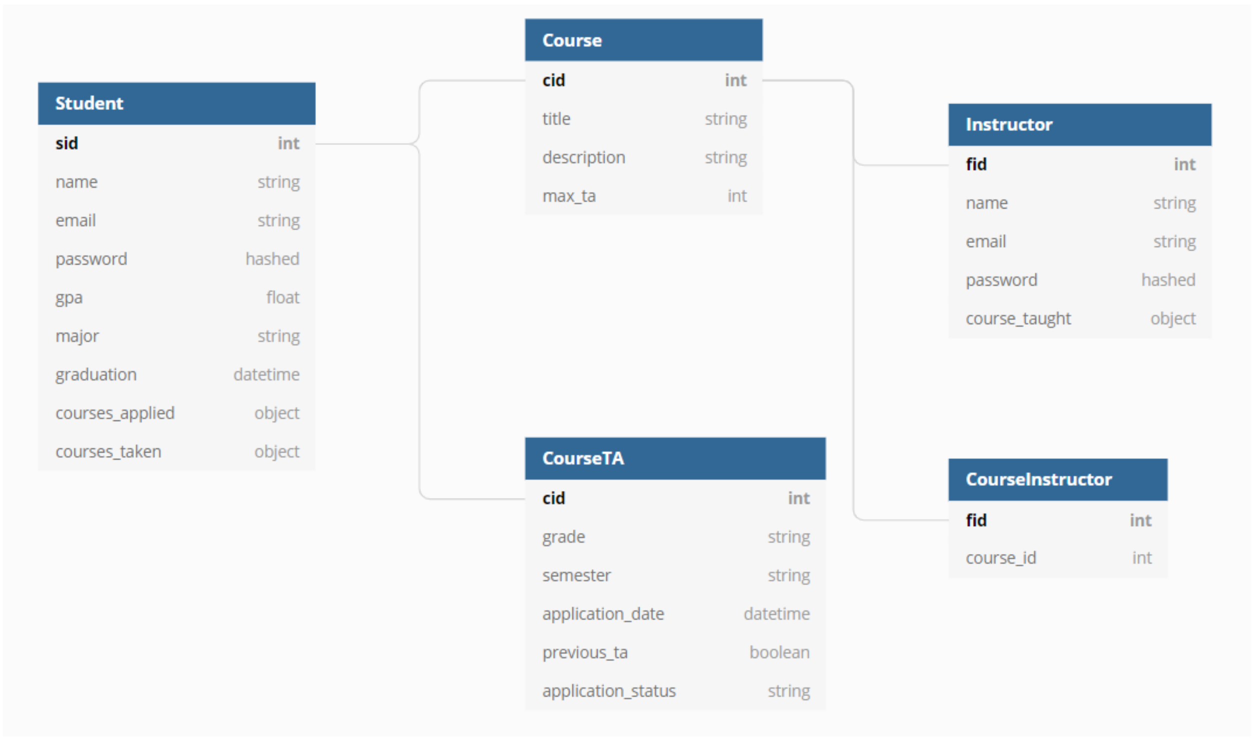
III.2. Data design

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

Database schema:



Database Schema (updated):



Student Database

Name	**Type**
std_id	int
name	string
password	hashed
GPA	float
major	string
Graduation date	datetime
Course IDs Applied	[object]
Courses taken	Course object

Professor Database

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

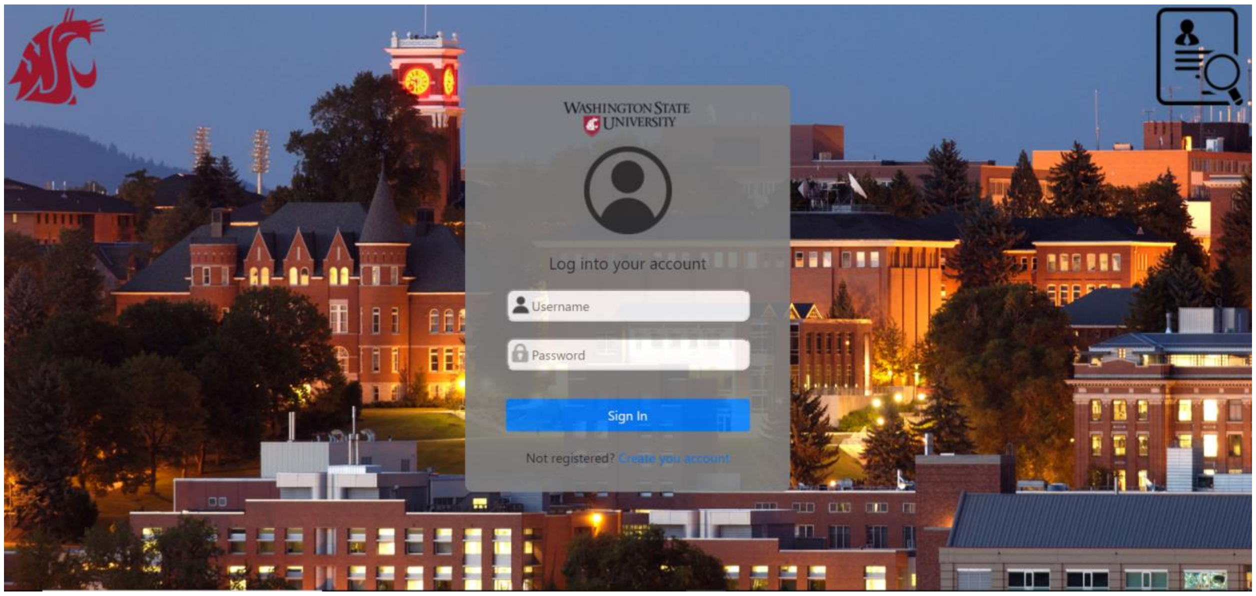
Courses Database

Name	**Type**

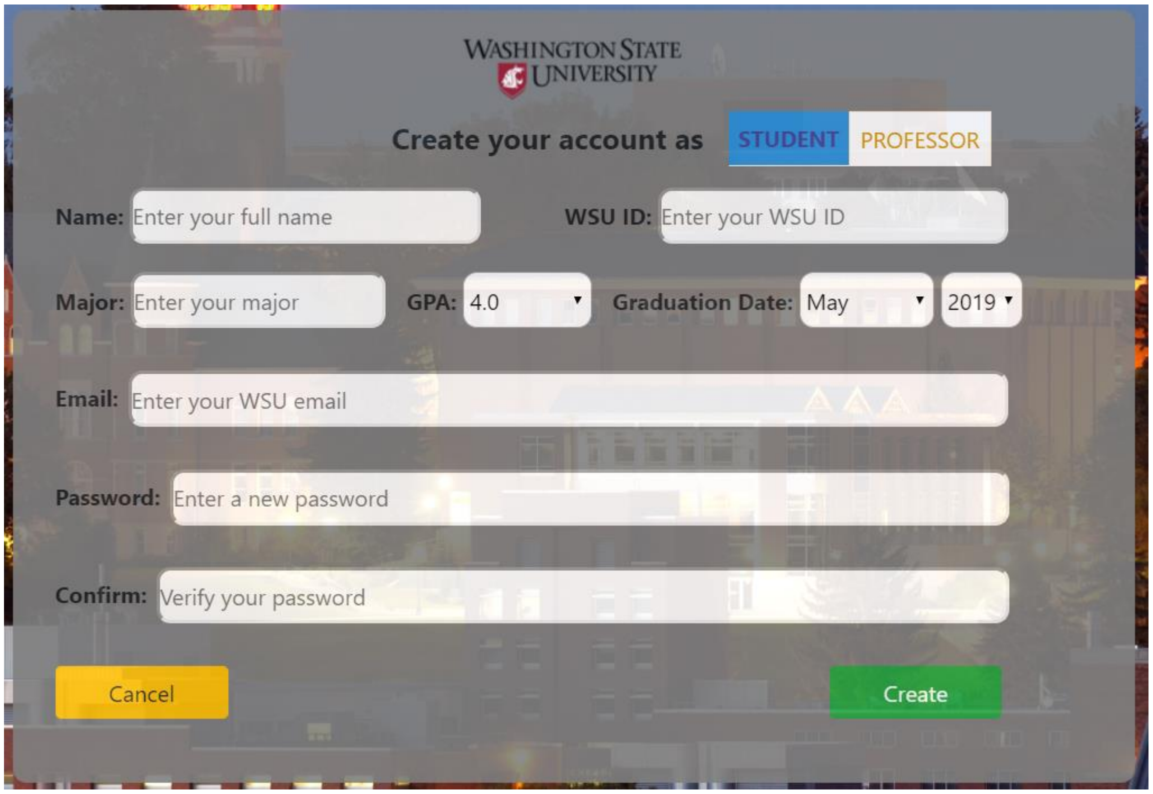
course_id	int
course title	string
course description	

III.3. User Interface Design

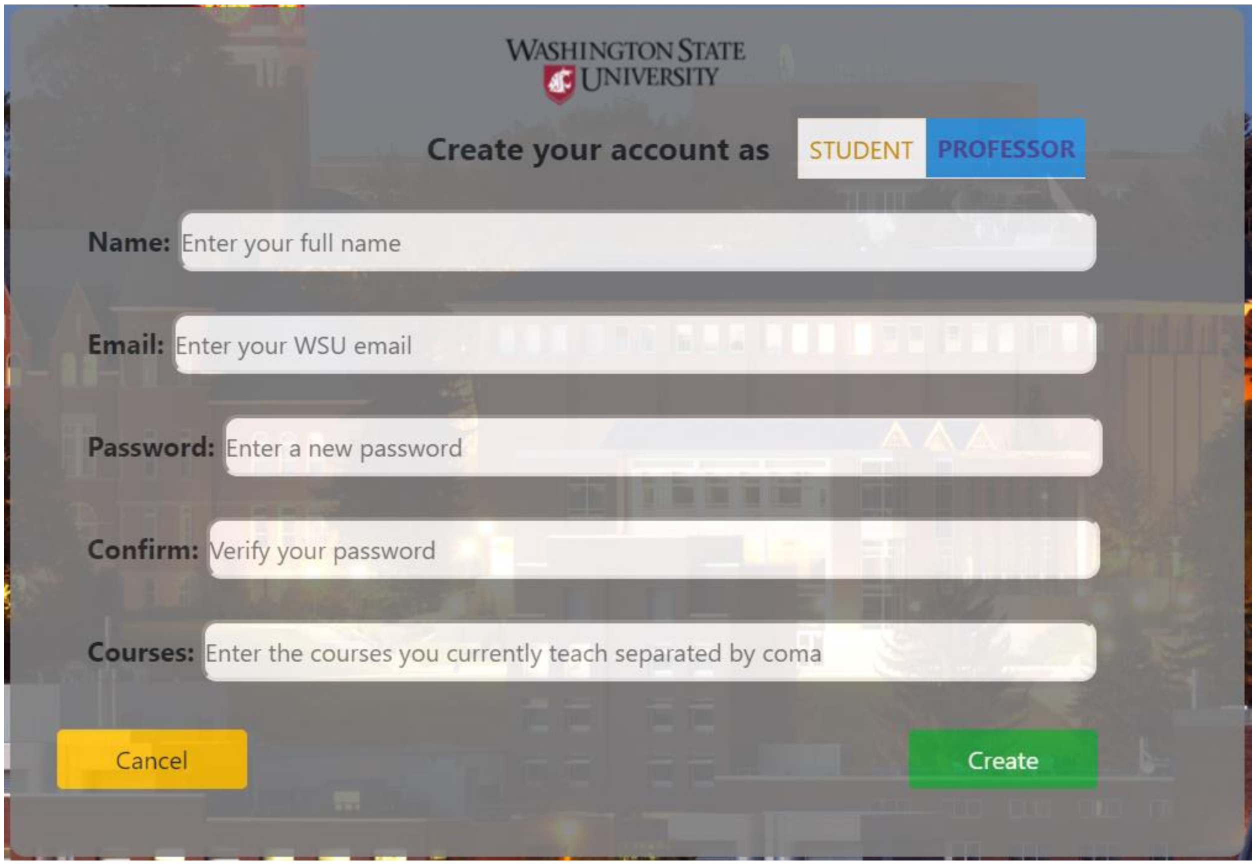
Use Case #3: Login


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

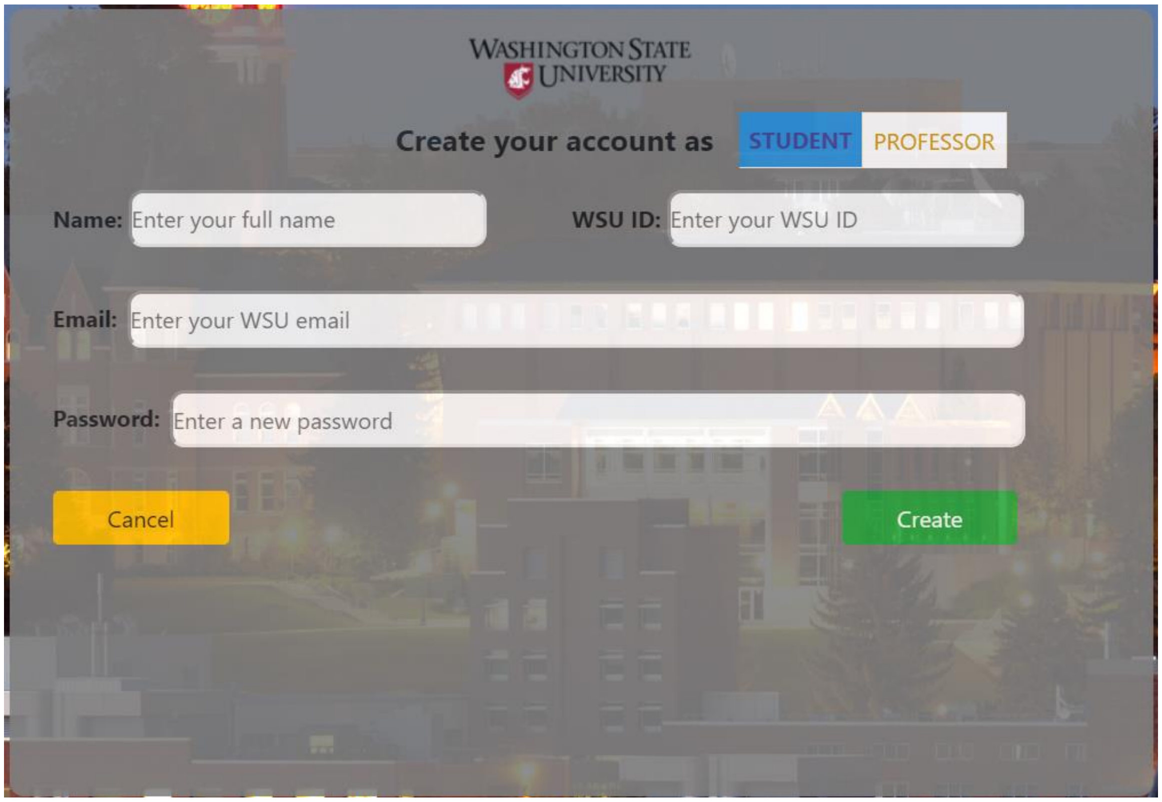
Use Case #1: Create Account Student

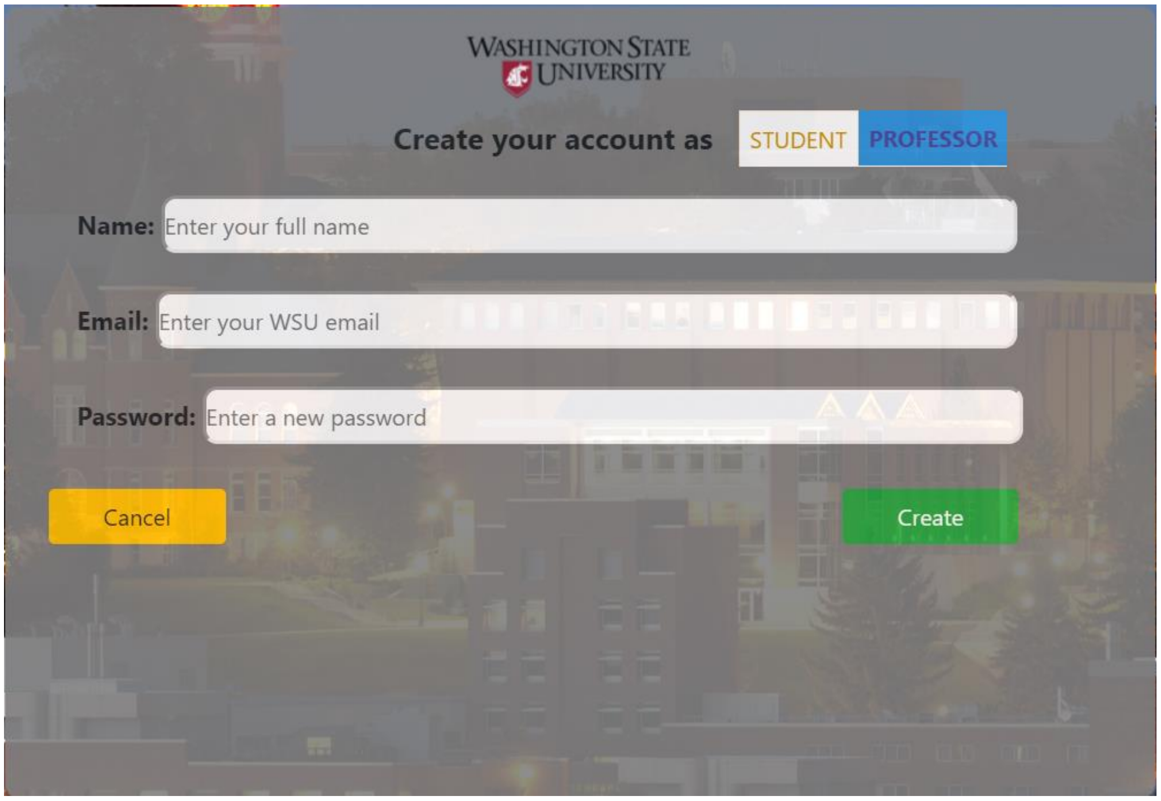

WASHINGTON STATE UNIVERSITY
Create your account as
STUDENT
PROFESSOR
Name: Enter your full name
WSU ID: Enter your WSU ID
Major: Enter your major
GPA: 4.0
Graduation Date: May
2019
Email: Enter your WSU email
Password: Enter a new password
Confirm: Verify your password
Cancel
Create

Use Case #2: Create Account Professor

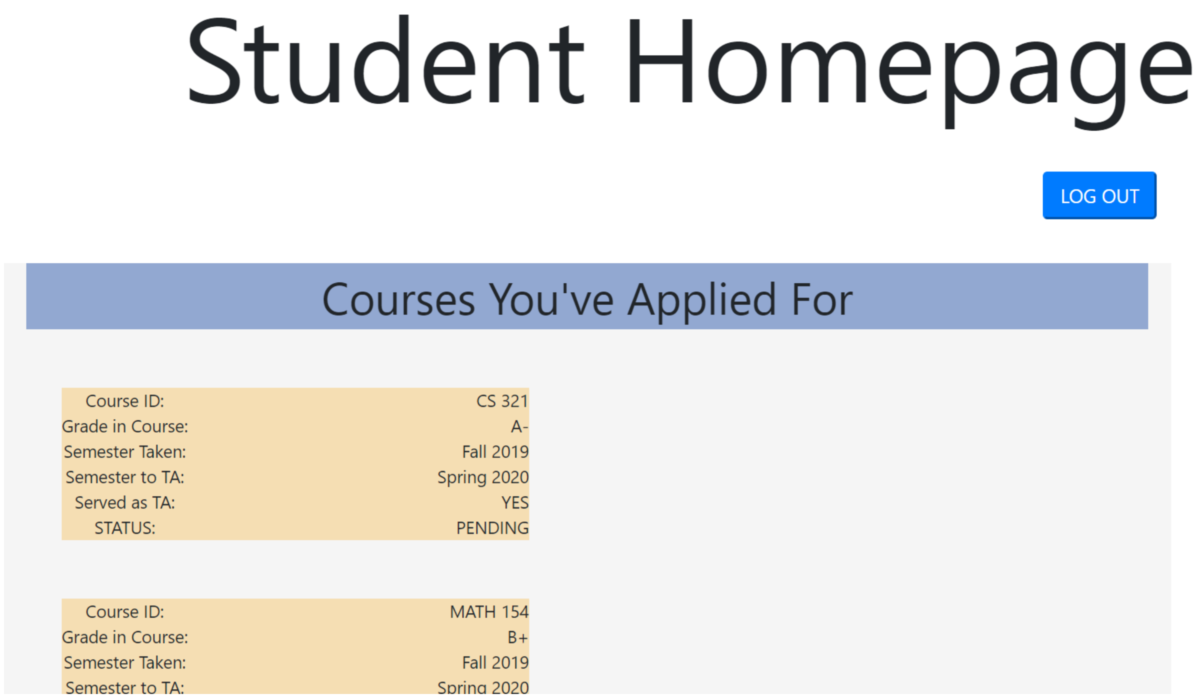

WASHINGTON STATE UNIVERSITY
Create your account as
STUDENT
PROFESSOR
Name: Enter your full name
Email: Enter your WSU email
Password: Enter a new password
Confirm: Verify your password
Courses: Enter the courses you currently teach separated by coma
Cancel
Create

Create Account Page (Updated in iteration 2):





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

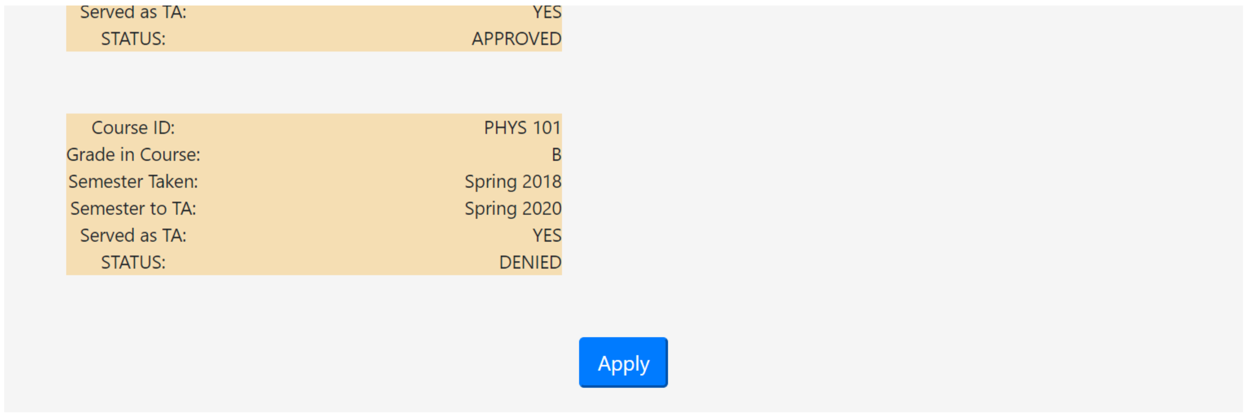


Student Homepage

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

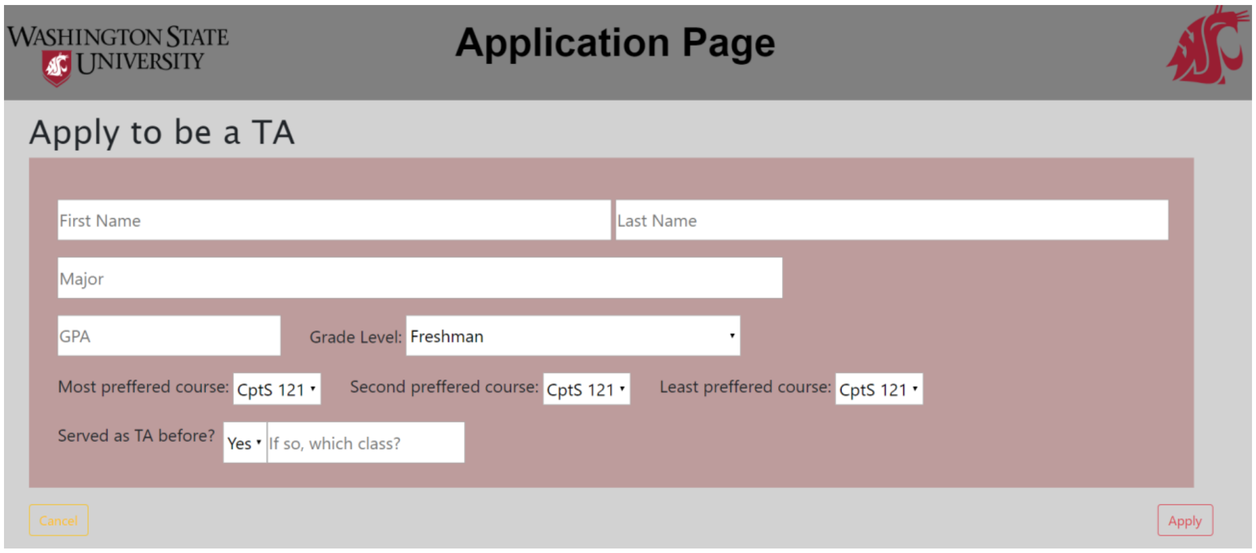
Course ID:	MATH 154
Grade in Course:	B+
Semester Taken:	Fall 2019
Semester to TA:	Spring 2020



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


WASHINGTON STATE UNIVERSITY

Application Page

Apply to be a TA

First Name | Last Name

Major

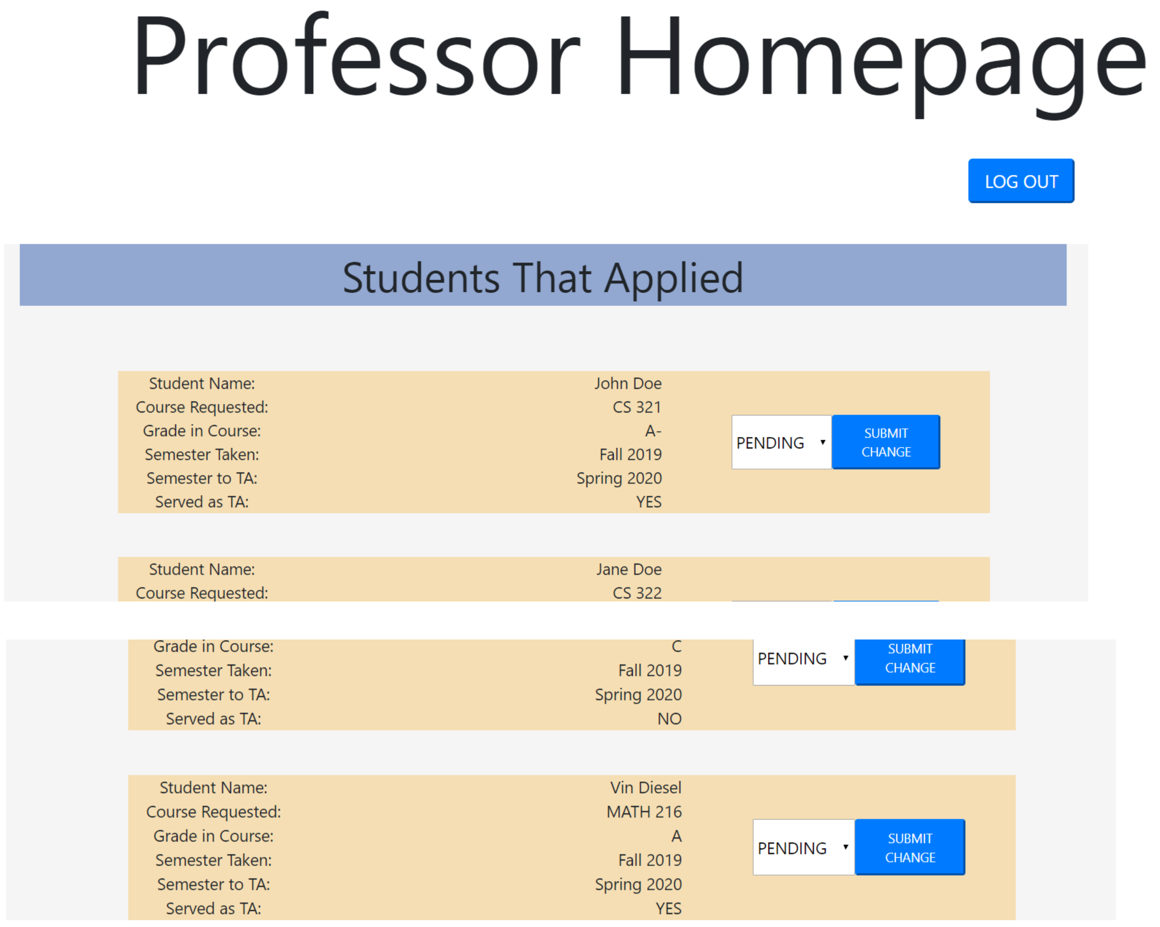
GPA | Grade Level: Freshman

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

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

Cancel | Apply

Use Case #5: Professor Homepage


Professor Homepage

LOG OUT

Students That Applied

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

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

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

IV. Progress Report

Iteration 1:

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.

Iteration 2:

Our progress for iteration 2 is mostly backend work with some JavaScript. We created databases for the student, instructor, and the courses. We then implemented the GET and POST requests for each database to store and retrieve info for creating accounts for both student and instructor classes. We worked a little bit on the implementing the JavaScript to connect the frontend and backend together, however we still need some time to get it completely working. We have all the template for the frontend ready. As soon as we could integrate frontend and backend using JavaScript properly, we'll be working on testing as described below and also improving the frontend.

Iteration 3:

For this iteration, we improved our existing frontend and added some required frontend. For example, we have the posting TA position page. Majority of our work in this iteration was integrating the frontend and backend using JavaScript. As of now, we have the frontend and backend integrated for login, create account for both student and professor, and displaying student/professor information in the student and professor homepage. We were also able to deploy our backend to Heroku server, so we all had access to the updated database and didn't have to rely on local server.

Some of the challenges we faced were trying get information from one page to another page. We also struggled a lot with integration. For example, we weren't able to make post request to the database when creating a new student/professor account. However, after so many frustrating hours, we were finally able to get this work. Same thing goes with getting the info from the server and displaying on our frontend – student/professor homepage. Because of our time constraint, we weren't able to test our backend. However, we did use Postman to black box testing to make sure we were getting the right result from the server.

V. Testing Plan

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:**

  We plan to use Unit testing for our API routes to ensure that the POST requests return a matching JSON object, and the GET requests return the matching JSON objects that were posted to the database. This will be Blackbox testing since we will not be accessing the inner workings of the backend.

- **Functional Testing:**

  We will test that the username and password are properly authenticated where it returns that it was rejected if the username was not found in the database or if the username is found and its password has doesn't match the one submitted.

- **UI Testing**:

  We will test the time that it takes for each page to load, and that the proper page is loaded from each button that is selected. This will be done by the test module subscribing to each button's click event, so when that button is clicked, the test module will authenticate that the proper page was next loaded.

VI. References

No references used other than the project rubric and PowerPoint for this course.