

# Term Project

## Design Document

10/23/2019

Version <2.0>



### SpaghettiCoders

Jimmy Zheng

Hien Duong

Peng-Yu Chu

Eric Furukawa

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

## TABLE OF CONTENTS

I.	Introduction..	2
II.	Architecture Design..	2
II.1.	Overview...	2
III.	Design Details.	4
III.1.	Subsystem Design..	4
III.2.	Data design..	7
III.3.	User Interface Design..	9
IV.	Progress Report.	16
V.	Testing Plan.	16
VI.	References.	17

# I. Introduction

This design document has been provided to outline and detail the appropriate subsystems introduced as part of the greater TA webapp in order to improve program timeliness, quality and scope. Subsystem details includes the associated databases, interaction with other subsystems and user interface accommodations. Additionally, the architectural design of choice for the project will be introduced and explained in the document. Progress and changes are also recorded to improve organization. This is the first iteration draft of the design document and thus no changes need to be discussed here. Section II will describe the architecture design, this section will include a UML component diagram. Section III will discuss design details. Such as subsystem designs, data design and user interface design. Section IV will discuss the progress of the project. Section V will include all references.

## Document Revision History

Rev 1.0 2019-10-23 Initial Version

Rev 2.0 2019-11-11 Iteration 2 Update

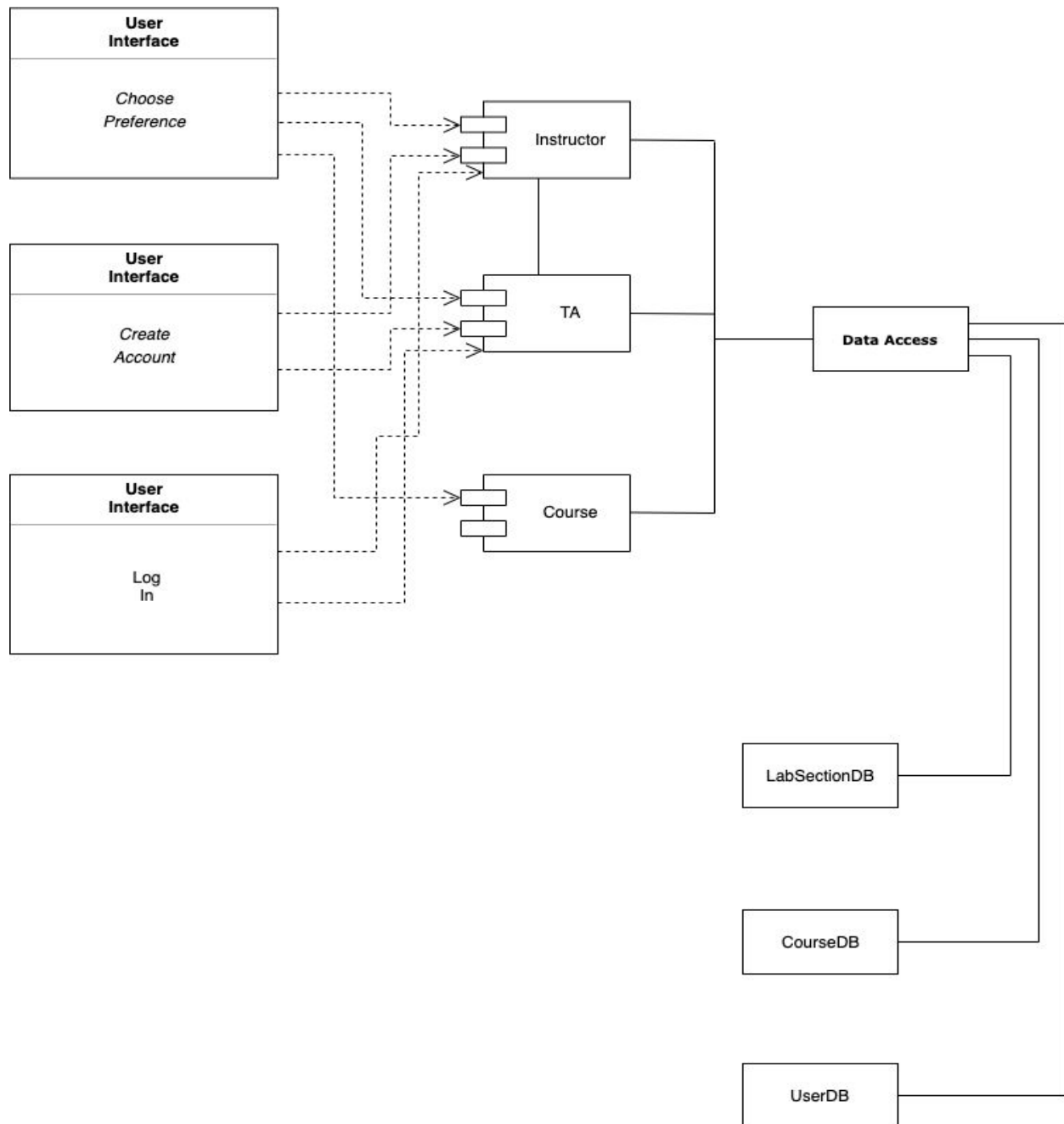
# II. Architecture Design

## II.1. Overview

The decided upon architectural pattern is the Client-Server Architectural Pattern. The TA webapp has a defined and easily identifiable front-end and back-end side to its implementation, correlating to the client and server side described in the C-S pattern respectively. Additionally, there is no need for any direct peer-to-peer communication which is generally a less reliable architectural pattern due to weak peer nodes. The TA webapp will need to be able to receive information from the user, perform data calculation and/or correlation and then return the appropriate information back to the user. This process fits the C-S pattern's functionality. Our server will be a web API, which will decrease the coupling between the server and the client as the client will only use get or post requests when interacting with the web API. The web API will in turn only give/receive info through these requests, keeping the implementation of the web API server and the client completely separate, with no components of either relying on each other.

The coupling of the client front-end and server back-end implementations must be kept low. This will be reflected in the fact that the only interaction between the two large subsystems will be the receiving of data through the HTML forms. Cohesion will however be kept through the similarities in the teacher and the TA forms as many functionalities, especially in login information, will overlap. This is why the subsystems will focus on division between tasks instead of between user types. The functionality of all

login functions, for example will be cohesive within the login component. It will be important to keep in check the coupling between the user types as functionality is similar but not the same.



The first level of decomposition on the webapp splits the program into a choose preference functionality, login functionality, and a create account functionality. The choose preference component is responsible for allowing users to enter their preferred TA selection or class selection and have them associated to their account and displayed to the other respective party. The login component is responsible for allowing users to provide their username and password to associate their session with their section of the databases. The create account component is responsible for allowing users to

provide their basic info and account credentials for use in accessing their specific database section. Each of the top layer components will be used for the instructor, student and class pieces of the program. The instructor component encompasses all of the functions available to instructors. The TA component does the same for all student functions. The course component will be used in parallel to keep track of preferences per account, but only under the choose preference top component. All of these components will interact with three databases (Labsection, Course, and User). These databases could be considered external components, but are more so a part of the internal program. No other external interaction features have been introduced.

## III. Design Details

### III.1. Subsystem Design

The HTML front end is composed of multiple HTML files and is primarily responsible for presenting the webapp with a visually appealing but utility based user interface. It also provides forms for the user to send information to the backend, which is done with help from the Javascript component.

The Javascript portion will follow the suggested MVC pattern to model the cycle of receiving information and displaying it. This will enable the client and server interaction components discussed below.

Messages in the login and create account components will contain entered information to which the back end will check against the database or create a new database for each respectively. The course preference selection will take a form of desired classes on the TA side and send it to teacher sessions which will in turn take the teacher's selections and send it to the TA's. In this case "sending" really means storing the information in the database from one side and get requesting it on the otherside when the other logs in. For our web API, the login route gets the username and password using the local host url `http://localhost:5000/api/login?user_name`. The new user route posts a new user with all account data fields: username, password, name, last name, title, wsuid and phone at url `http://localhost:5000/api/newuser`. The route for creating a new course post with entries: name, professor, prof\_id, lab\_section and prof\_email has the url `http://localhost:5000/api/new_courses`. The get request route for courses receives the applicants of the course and its name with url `http://localhost:5000/api/courses?course_name`. Finally, the get route for getting a professor gets the list of courses the professor teaches at the URL `http://localhost:5000/api/course?professor_name`.

Each class will have private information fields for encapsulation and security so necessary get functions are required for all private fields.

#### **III.1.1. Account**

The account class will be made for holding the form information which both student and professor accounts share. Each account will also have to be able to be created in the database and to be able to login via the login script. The create method will take in the information entered into the form and instantiate a new entry in the account database for it. The login method will take in a username and password from a form and search the database for a matching entry and return a confirmation or denial of a match.

#### **III.1.2. Student Account**

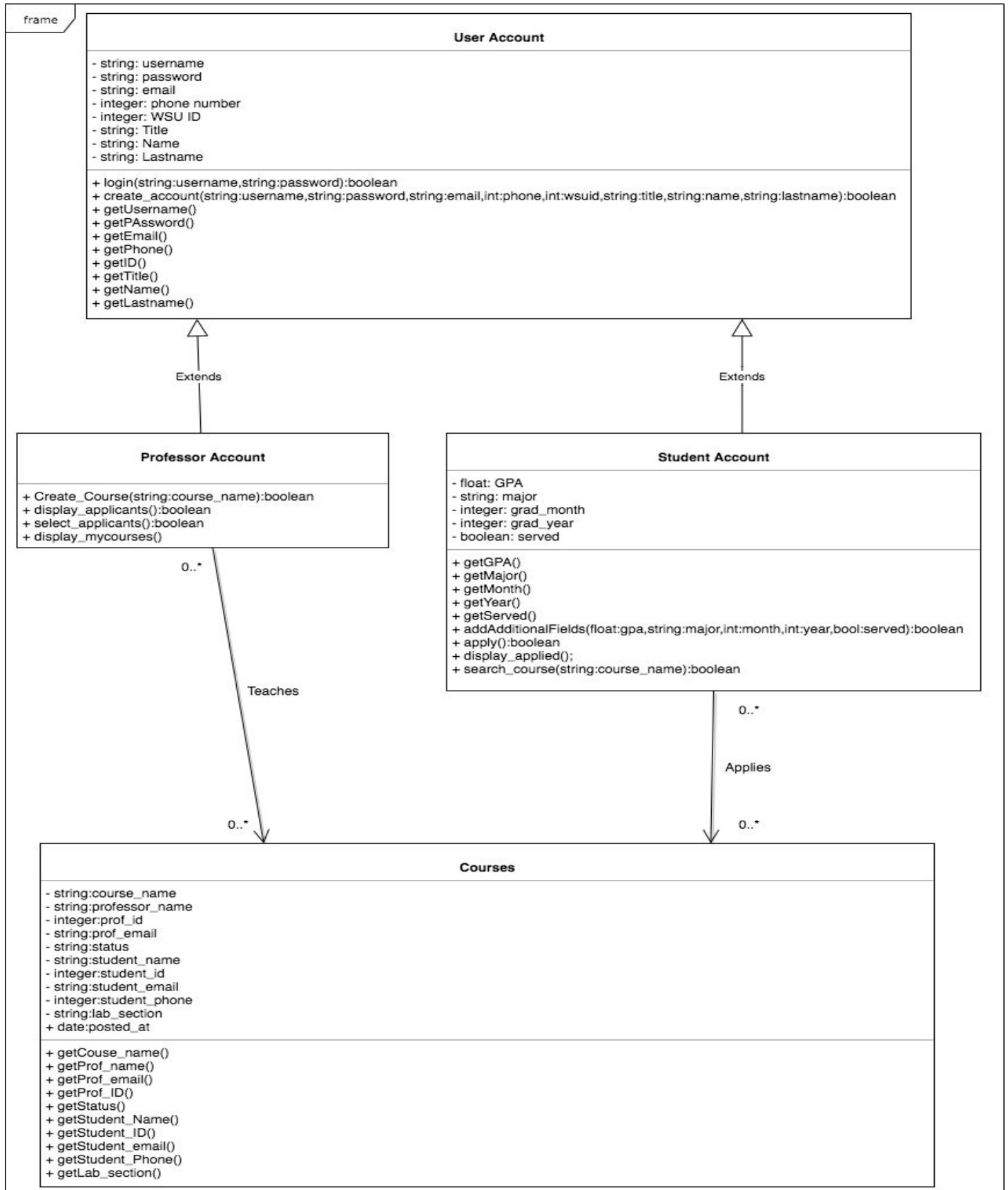
The student account is a specific type of account which has extra student specific fields such as gpa, month and year. They also have student specific functions to accommodate the requirement of being able to select preferred courses such as the select courses function and the view courses function. The select course method will add preferred courses to the student's preferred courses in the database while the display method will show this database in the form of a list to the student.

#### **III.1.3. Professor Account**

The professor account is a specific type of account which shares the same fields as the general account class. It however has professor specific functions to accommodate the added functionality of selecting/creating taught courses and selecting preferred TAs, reflected in the extra functions. The select/create courses method allows the professors to add courses to the database of courses they teach. The display courses method will display all the courses they teach. The select preferred TA method will allow professors to select which TA they want for each course while being displayed by the display applicants method.

#### **III.1.4. Courses**

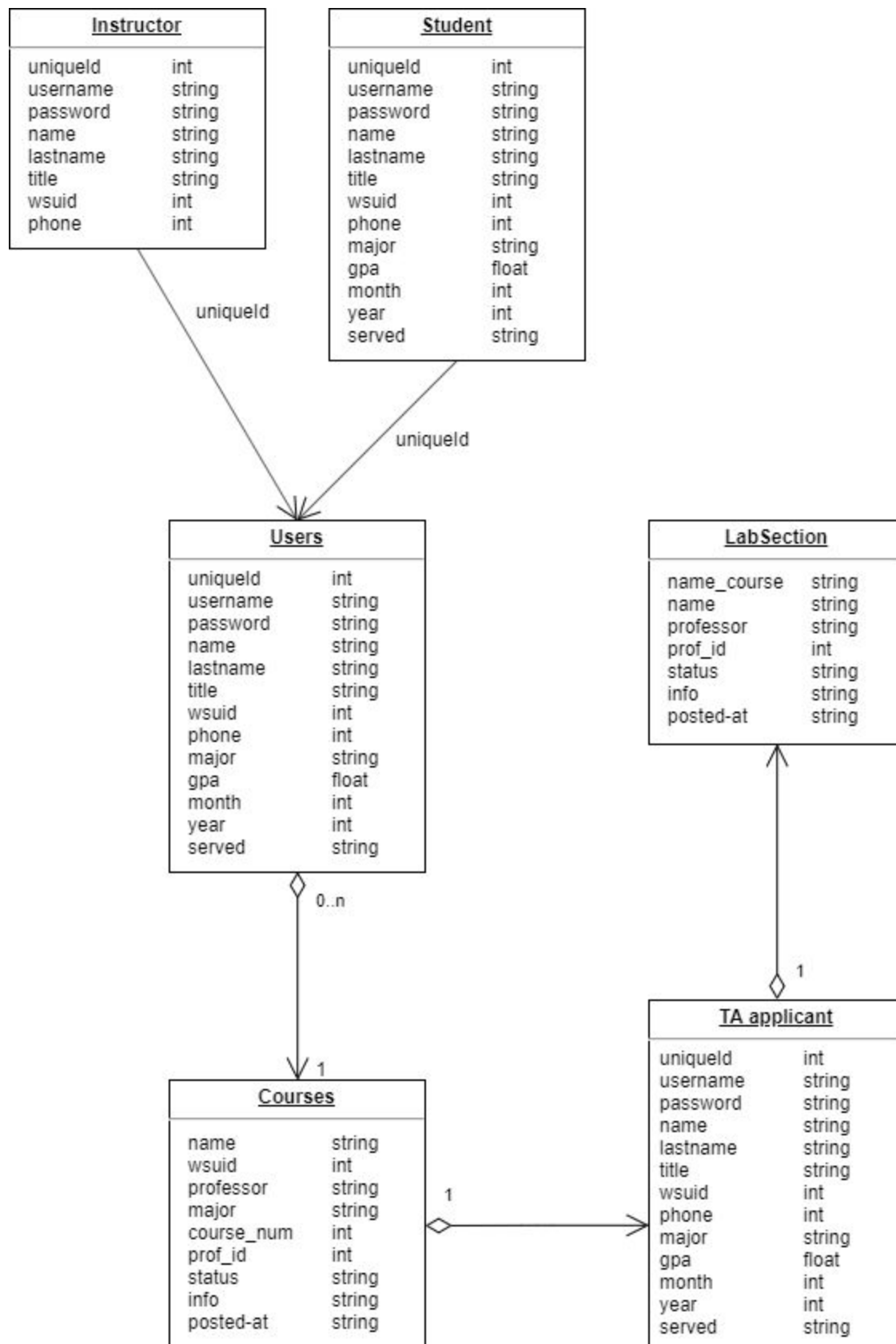
The course class holds the information on a given course but has no methods apart from the getters. Courses have "has a" relationship with both account types, as both accounts contain courses they either teach or applied to. Courses are also the data type to be displayed to both users.



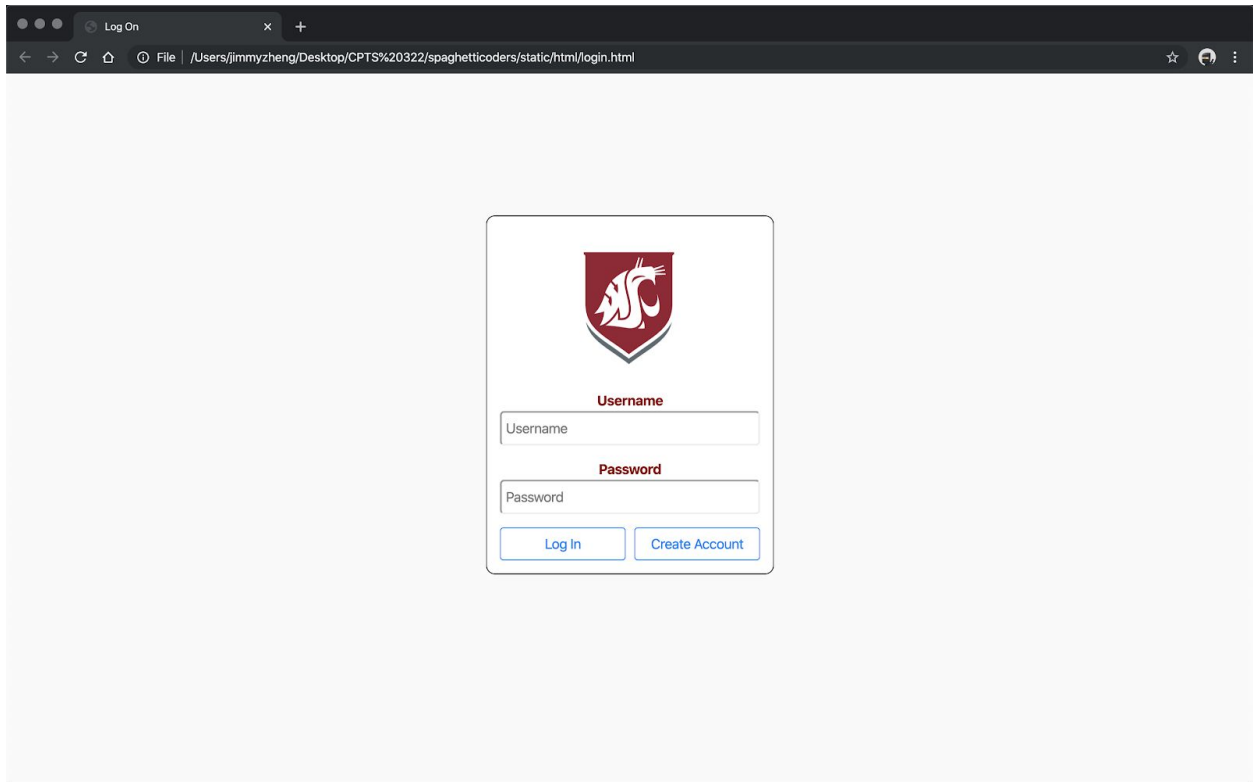
## **III.2. Data design**

The database being used is a SQLAlchemy database. The user database will contain a list of all users, professor and student, Courses database will be a list of all courses that the professor wants a TA for. Students will be able to get the course and submit their request. Lab database will contain all the labs sections TA can apply for. Professor can add lab sections to database.





### III.3. User Interface Design



The screenshot displays a web browser window with a single tab titled "Log On". The address bar shows the file path: `/Users/jimmyzheng/Desktop/CPTS%20322/spaghetticoders/static/html/login.html`. The main content area features a login form with a red shield logo containing a white stylized 'C' and a unicorn head. Below the logo, the form includes a "Username" label, a text input field, a "Password" label, another text input field, and two buttons: "Log In" and "Create Account".

The user interface shown above is the login screen. It consists of two text boxes and two buttons. The first text box is for the user's username and the second is for the user's password. The button on the left labeled login is for the user to login to their account and the button on the right is for users who would like to create an account. The use-case #3, which is "login account" for all users utilizes this user interface.

**Contact Information**

**WSU Email**  
Enter WSU Email

**Password**  
Enter Password

**Title**  
Professor

**First Name**  
Enter First Name

**Last Name**  
Enter Last Name

**WSU ID**  
Enter WSU ID

**Phone Number**  
Enter Phone Number

Submit Cancel

This user interface above is for the user to input their information such as their WSU email and their password. This screen consists of six text box, one drop down menu, and two buttons. We have text boxes for the users WSU email, password, first name, last name, WSU ID, and their phone number. One drop down menu for their title which has student and teacher. The button on the left is for the user to submit their information. The button on the right is to cancel, which will bring the user back to the login screen. The use-cases #1 “Create Instructor Account” and #2 “Create Student Account” utilize this user interface.

The screenshot shows a web browser window with the title 'Create Account - Student'. The address bar displays the file path: `/Users/jimmyzheng/Desktop/CPTS%20322/spaghetlicoders/static/html/additional.html`. The main content area features a form titled 'Additional Information' in a dark red font. The form is enclosed in a light gray border and contains the following elements:

- Major:** A text input field with the placeholder text 'Enter Major'.
- GPA:** A dropdown menu currently showing '4.0'.
- Graduation Date:** Two dropdown menus; the first shows 'January' and the second shows '2019'.
- Served Before:** A dropdown menu currently showing 'Yes'.
- Buttons:** Two buttons at the bottom, 'Submit' and 'Cancel', both with blue borders and text.

This user interface is the additional information page. This page will only show up if the user is a student. There is only one text box on this screen and it is used to mark the users major. This page consist of four drop down menu, the first one we will see is the GPA drop down menu which will allow the user to submit their GPA. The next set of drop downs we will see are the ones for the users graduation date. And lastly we have a drop down for if the user has served as a TA before. The button on the left is to submit additional information on this page. The button on the left will bring the user back to the login page. The use-cases #2 “Create Student Account” and #6 “Enter Additional Information” utilize this user interface.

benson chu  
11345567  
ben40404@hotmail.com

MY COURSES

APPLICANTS

ADD COURSES

### COURSES TEACHING

Cpts224  
N/A

Select

This user interface is for the professor. This page displays all the courses they teach. The page contains three tabs and a list of all courses taught by the professor. My Courses Tab is the default tab that will be displayed. The next tab is the applicants tab where the professor can find all the applicants that applied for his class. Lastly the add courses tab is for the professor to add classes he will be teaching. The use-cases #8 "Enter Course and Lab Section Instructor Teaches" utilize this user interface.

benson chu  
11345567  
ben40404@hotmail.com

MY COURSES

APPLICANTS

ADD COURSES

## APPLICANTS

Hien Duong  
12345678  
hien.duong@wsu.edu

Assign

This user interface is for the professor. This page displays all the applicants for a specific course they have selected. The page contains three tabs and a list of all applicants for a specific class. My Courses Tab is the default tab that will be displayed. The next tab is the applicants tab where the professor can find all the applicants that applied for his class. Lastly the add courses tab is for the professor to add classes he will be teaching. The use-cases #9 "Display list of applicants per class" utilize this user interface.

Benson Chu  
13214654  
benson.8212@wsu.edu

MY COURSES

APPLICANTS

ADD COURSES

### ADD COURSES

Course Teaching:

-select-

Lab Section:

-select-

Create Course

This user interface is for the professor. This page allows the professor to add classes he needs TAs for. The page contains three tabs and a list of all applicants for a specific class. My Courses Tab is the default tab that will be displayed. The next tab is the applicants tab where the professor can find all the applicants that applied for his class. Lastly the add courses tab is for the professor to add classes he will be teaching.

Tommy Chu  
11348098  
ben40404@hotmail.com

APPLIED COURSES

COURSES

SEARCH COURSES

### COURSES APPLIED

CptS 121  
Andy O'fallon  
Pending

X

CptS 121  
Andy O'fallon  
Pending

X

CptS 121  
Andy O'fallon  
Pending

X

CptS 121  
Andy O'fallon  
Pending

X

This user interface is for the student. This page displays all the applied courses the student has applied for. The page contains three tabs and a list of all applicants for the applied classes. The applied courses tab is the default tab and is the one currently being shown. The courses tab is all the classes professor has posted they need a TA for. And lastly the search courses tab is for the student to find a specific course they want to TA for. The use-case #4 "Enter course Preference makes use of this interface.

Tommy Chu  
11348098  
ben40404@hotmail.com

APPLIED COURSES COURSES SEARCH COURSES

## COURSES

CptS 121 Andy O'fallon <a href="#">Apply</a>	CptS 122 Andy O'fallon <a href="#">Apply</a>	CptS 122 Andy O'fallon <a href="#">Apply</a>	CptS 122 Andy O'fallon <a href="#">Apply</a>
CptS 122 Andy O'fallon <a href="#">Apply</a>	CptS 122 Andy O'fallon <a href="#">Apply</a>	CptS 122 Andy O'fallon <a href="#">Apply</a>	CptS 223 Andy O'fallon <a href="#">Apply</a>

This user interface is for the student. This page displays all the courses the student can apply for. The page contains three tabs and a list of all applicants for the applied classes. The applied courses tab is the default tab. The courses tab is all the classes professor has posted they need a TA for and is the one currently being shown. And lastly the search courses tab is for the student to find a specific course they want to TA for. The use-case #4 "Enter course Preference makes use of this interface.

Tommy Chu  
11348098  
ben40404@hotmail.com

APPLIED COURSES COURSES SEARCH COURSES

## Search Courses

Course Name:

Lab Section:

[Search Course](#)

This user interface is for the student. This page is used for when the student has a specific class they want to apply for. The page contains three tabs and a list of all applicants for the applied classes. The applied courses tab is the default tab. The courses tab is all the classes professor has posted they need a TA for. And lastly the search courses tab is for the student to find a



specific course they want to TA for and is the one currently being shown. The use-case #4 “Enter course Preference makes use of this interface.

## IV. Progress Report

The first major iteration accomplishment was creating the multiple HTML files for the create account, login, student and teacher pages (including appropriate information forms). CSS styling were added to make the user interface neater and more visually appealing. SQLAlchemy/Flask backend was created for local account information storage. The javascript for logging into an account stored on the database was added to work with the login HTML page. Javascript for creating an account was also added to accompany the create HTML page, which takes form entered information and adds an entry to the database.

In iteration two, we added to the html and javascript for the student and instructor changing their UI and adding more functionality. The Professor page now lists the courses which the instructor has selected as the courses they teach. These courses can be clicked on to show all student applicants for the specific course. Courses can also be added to the list of existing courses by instructors.

## V. Testing Plan

Unit Tests: We have implemented automated unit testing by having a case which instead of taking information from the html form, uses default values to create a new login for the database. We have another test to use in tandem to search for and check if any of the user information bundles have matching username and password entries of the default, hard coded values.

Our functionality testing will primarily be manual. Our preliminary testing will be done by ourselves to ensure functionality is preserved before usability. Of course, usability is also highly important, so having testers not involved in the coding will give us the best insight on this. Use cases will be addressed separately by questioning them to outside users. “Were users able to accomplish so and so” will be used to identify missing or needed functionalities.

User interface will be tested in a similar fashion to our functionality testing. Us as the coders will know the functionality of the program that casual users may not know about, so having users selected from the customer base will be the most beneficial. User interface should be both easy to navigate and visually pleasing. Questions on both of these aspects will be asked on the program as both individual pages and as a cohesive whole. It will be easier to keep track of which pages need UI improvement by asking about pages separately, but an over UI analysis is needed to find out if program navigation and cohesive theming are up to par.

## VI. References

Google, [google.com](https://www.google.com)

Blackboard, [learn.wsu.edu](https://learn.wsu.edu)

MyWSU, [my.wsu.edu](https://my.wsu.edu)

RateMyProfessor, Sakire Arslan, (Class Project)