TA Helper

Design Document
11/5/2018
Version 0.2

BIG BUG BUSTERS
(Nolen Young, Zach Snoen, Ben Mathews)

TABLE OF CONTENTS

I. Introduction

The purpose of this application is to allow for the course management for class scheduling. This application will allow a user to login to the system and see their information. After login depending if the person that logged in was a professor or a student. The appearance of the system will be different and the actions that each user can do will be different. This project's purpose is to make scheduling easy for TA's to register and choose classes they want to register to TA and for professor's to easily be able to find TA's for the class. This design document outlines the systems that we will create in order to make this web application work.

In section II of this document we will go over the overall architecture of the project. We will discuss the major components of the software and how they all interact and work together. In section III we will discuss more specific design details. We provide more in depth detailing of each subsystem, and provide details on how all major components of the architecture interact. We will also discuss communications protocols between each major component. We will also go over the database tables we will be using in our applications, as well as a mockup of the final user interface design.

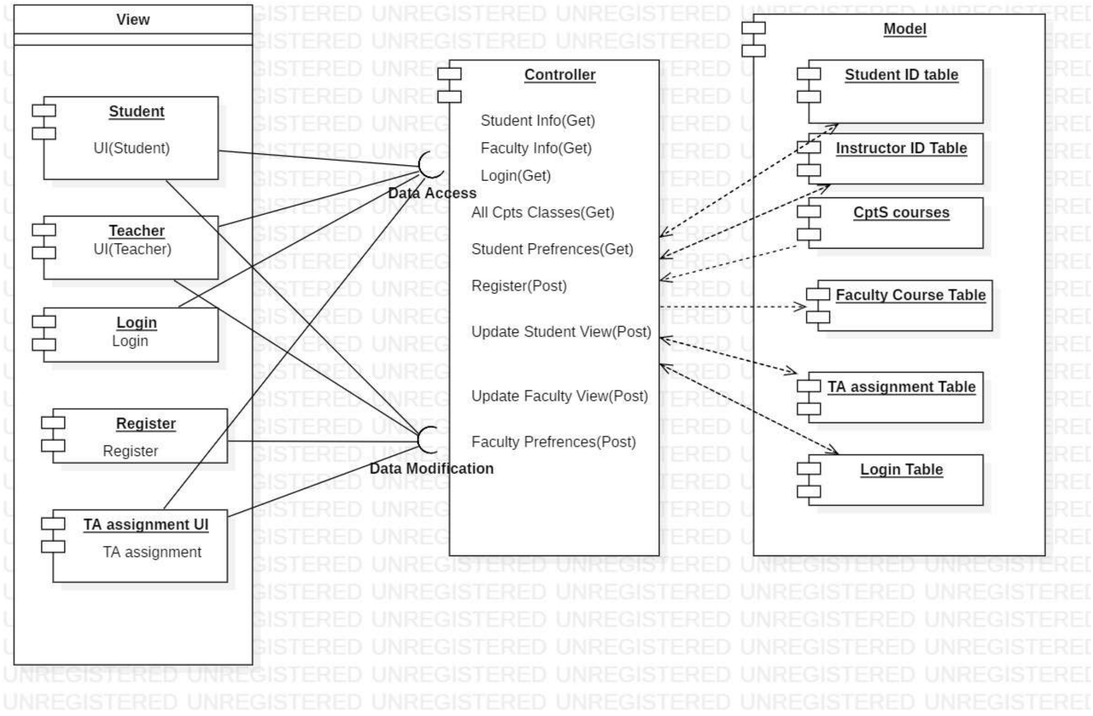
Document Revision History

Rev 0.1 10/22/2018 Initial Version
Rev 0.2 10/29/18 Second Version

II. Architecture Design

II.1 Overview

The pattern we used was a MVC architecture. The reason for this choice is that it allows for decreased coupling from the user interface and the data. The way it works is that the user will interact with the system either causing an update request for the controller to the model, or for the controller to retrieve data for the View. If it's modifying or its getting data, either way the user is not able to directly access the database's. The reason for this is that it allows for decreased coupling in the system.



View:

Register(UI): This component is comprised of inputs from the user for their email, and password. Once the registration is valid, a new Faculty or student account is made. The email must be unique in the system.

Login(UI): This component takes in input from the user on their account information. It then checks to make sure that the login information is valid in the database

Student(UI): This is the UI once a student has logged into their account. The user here can edit information and contact info. The specifics will be discussed below. This component makes a request to display the information associated with their account and can edit and change their information.

Faculty(UI): This UI is for the faculty once they have logged in. The faculty can update their contact information, as well as add or delete courses.

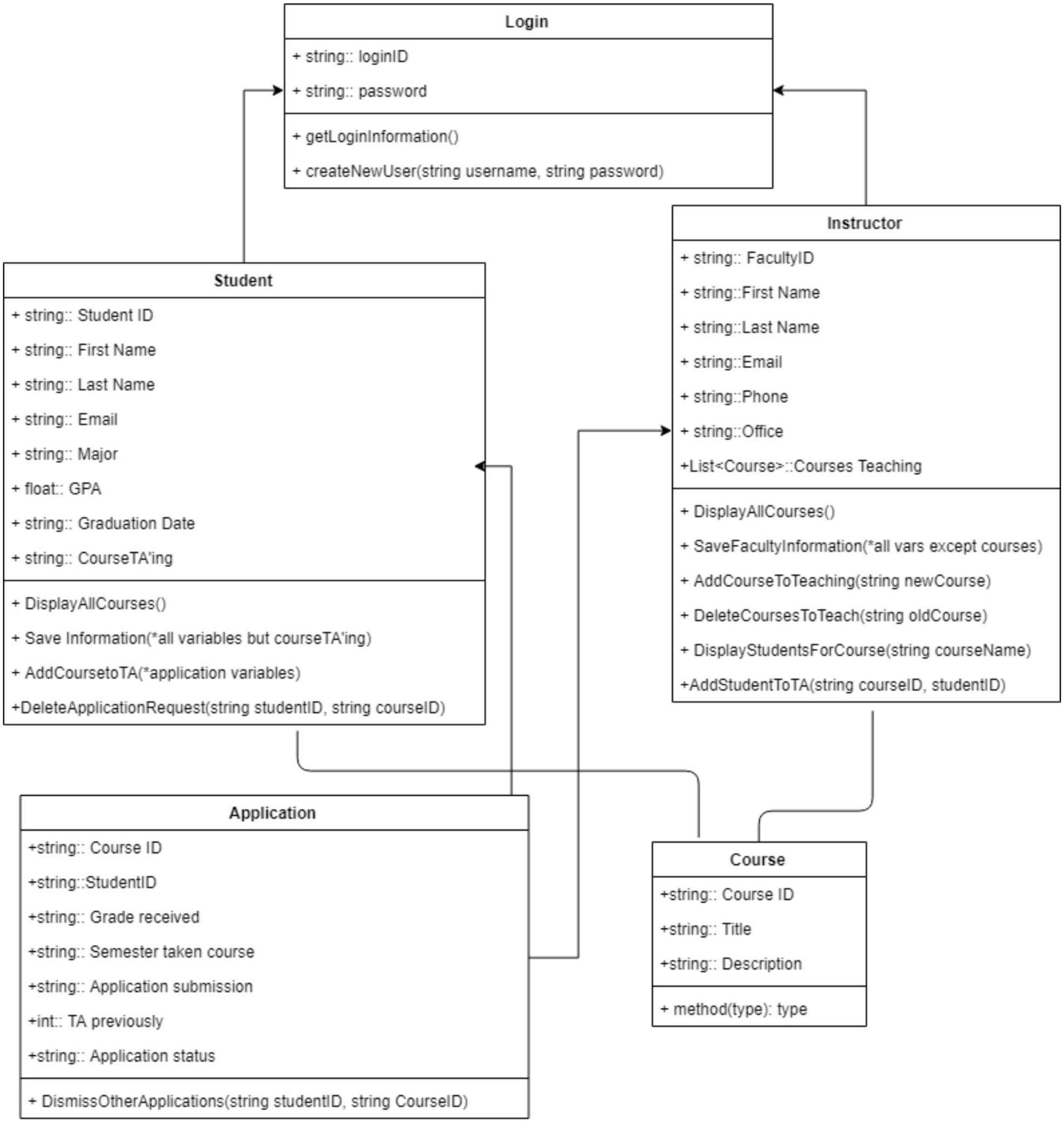
TA assignment(UI): This UI is for adding TA's to courses the faculty is teaching.

Controller: The controller is in charge of handling the different requests the user makes, as well as retrieving or modifying the data in the Model. The controller takes the request from the frontend and uses javascript to make requests and have the backend handle those requests.

Model: The model is where the data is stored on the server. It keeps track of all the user accounts, the information for each user, all available cpts courses, the logins of the users, and the courses each faculty are teaching, and the TA's available in each course. Right now the login's are not encrypted.

III. Design Details

III.1 Class Diagram


Login
+ string:: loginID
+ string:: password
+ getLoginInformation()
+ createNewUser(string username, string password)
Instructor
+ string:: FacultyID
+ string::First Name
+ string::Last Name
+ string::Email
+ string::Phone
+ string::Office
+List<Course>::Courses Teaching
+ DisplayAllCourses()
+ SaveFacultyInformation(*all vars except courses)
+ AddCourseToTeaching(string newCourse)
+ DeleteCoursesToTeach(string oldCourse)
+ DisplayStudentsForCourse(string courseName)
+AddStudentToTA(string courseID, studentID)
Student
+ string:: Student ID
+ string:: First Name
+ string:: Last Name
+ string:: Email
+ string:: Major
+ float:: GPA
+ string:: Graduation Date
+ string:: CourseTA'ing
+ DisplayAllCourses()
+ Save Information(*all variables but courseTA'ing)
+ AddCoursetoTA(*application variables)
+DeleteApplicationRequest(string studentID, string courseID)
Application
+string:: Course ID
+string::StudentID
+string:: Grade received
+string:: Semester taken course
+string:: Application submission
+int:: TA previously
+string:: Application status
+ DismissOtherApplications(string studentID, string CourseID)
Course
+string:: Course ID
+string:: Title
+string:: Description
+ method(type): type

The classes are laid out so that that when a user first enters the program, it can login if previously created, or can create a new student or instructor. When first created the student and faculty fields are completely empty except for the ID which is the same from the login ID. The password is not stored or changeable once past the login page. When a student adds a course, AddCoursetoTA() method, all the fields from application will be filled in, and the application status will be empty. These applications will be stored and then can be accessed by the instructor when they need to be. Both of these need to be able to access the course class because when the student or faculty are selecting their course, they should be able to find all of the classes. All of the Courses are going to be created beforehand. This is totally suitable for our application but in a larger scale project their would be a way to modify those entries. Instructors have all the different abilities to be able to add courses, delete courses, and add students to their classes. The application class also contains a method that once its field, “Application Status” gets set to “assigned” it will filter the rest of the applications by that student and set those applications to “dismissed”.

III.2 Subsystem Design

III.2.1 View Component

Register(UI): The user can enter in a valid email that must be unique. The user also selects either faculty, or student depending on their status. Once a user hit’s submit, that user’s email and password make a “POST” request to make a Login Database, and an empty user information database. The user information database will be either a faculty or a student depending on what the user selects.

Login(UI): The user enters their login information and makes a “GET” request to check if the login is valid. If the login is valid then the user or faculty information will be shown.

Student(UI): The user has all the information in the “Student Information Table” displayed on this page. To do this, once logged in the system makes a “GET” request to the controller, which accesses the information from the respective student’s information table. The information is then displayed with each piece of data having a box for editing. If the user hits the submit button on the page, then the system will make a “POST” request to the controller which makes then updates the information from the UI to the database. Also on this page is the ability to add and remove course preferences. The user can enter in the data in the TA assignment database table outlined in section III.2. Everything should be filled in to have a valid course preference update. The database is filled with the student’s information except for the faculty ID which will be filled addressed later in this section, and the “application status” which will be set to “pending”.

Faculty(UI):The faculty information gets displayed with a “GET” request once the login has been made. The information will be displayed and the information will be able to edit. If the faculty hits a submit button the information will be sent to the server and a “POST” request will be made. The Instructor Information table will be modified. The instructor can also fetch all of the cpts courses with a “GET” request to the CPTS database. The instructor can then select a course to add TA’s bring up the TA assignment UI. If the instructor selects a course a “GET” request is made to bring up the all of the students that have prefrenced that course.

TA assignment(UI): The instructor can select different students that they want to be assigned to that CptS course. Once students are selected, then a “POST” request is made to change that data to have the Faculty’s respective ID number and to change the application status to “approved”. This request also changes the other course preference application status entries that had the selected student's ID to be set to "dismissed".

III.2.2 Controller: The controller is implemented through Javascript code. The UI makes requests either "POST" and "GET" requests to the database that it needs. This allows for the View and Model to act independent of each other.

III.2.3 Model: The different databases and when they are accessed and modified are discussed in detail above. The Model contains the different databases that are needed to store the data in a way that makes it clean, efficient and secure to access.

III.3 Data Design

Student Information Table
First Name
Last Name
WSU ID
Email
Phone Number
Major
Cum GPA
Expected Graduation Date
Student ID

This database table will be created when a student registers their email and password. Their entered email and password will be stored in their respective login table, however their information table will also be created, if that email is unique. The 'student ID' attribute will be created with a value of 1 for student.

Instructor Information Table
Faculty ID
Name
Last Name

Email
Phone
Office

This database table will be created when a faculty registers their email and password and registers as a faculty. The ID will be created with a value of 2, for faculty.

CPTS Courses
Course Name
Title
Description

This database will be initialized and created with the application. This database should never change.

Courses Instructors Teach
Faculty ID
Course Name

This database will be initialized to be empty. Once a faculty user makes an addition to their class list that they teach then an addition will be added to the table with their ID and the course name.

TA assignments
Student ID
Faculty ID
Course Name
Course Grade
Year and Semester taken Course
Application Date
TA before?

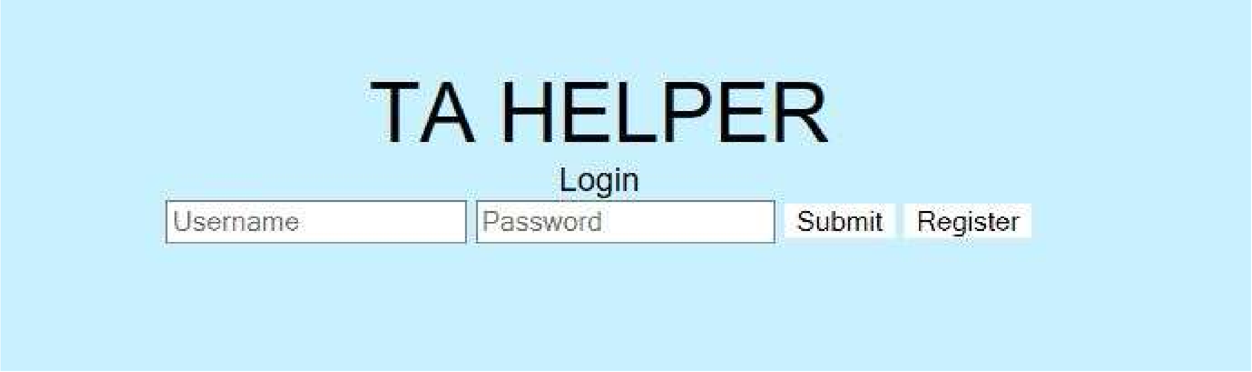
Ta before for this course?
Application Status

The database that will control TA assignments. Students will update in their information UI their courses that they want to add. All that information will get added into this table with an empty faculty id and an application status of ‘pending’. Once a faculty selects a student assignments every other entry in the table with that students ID will have an updated status of ‘dismissed’.

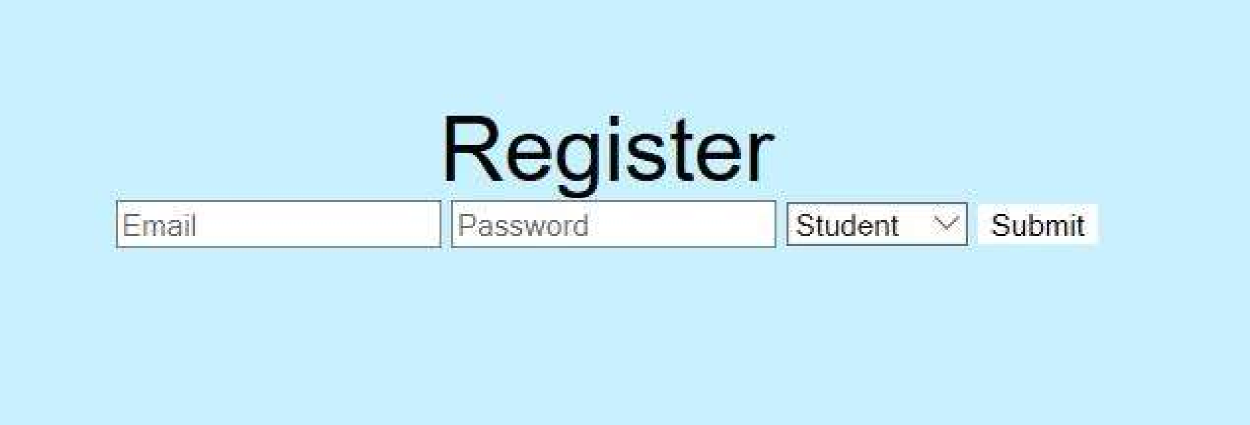
Login Table(not encrypted right now)8
Username(email)
Password

The database that keeps track of the emails and passwords of all users. At this time the passwords are not not encrypted. This database is accessed by the controller when a user attempts to login. This database is also used when a “POST” request is made when a new user registers.

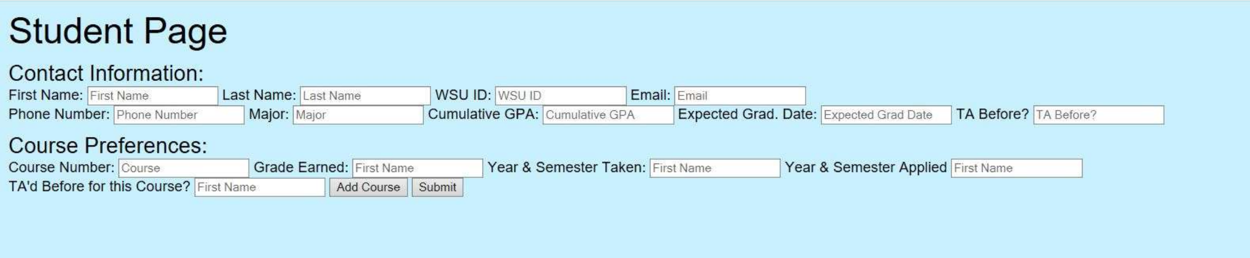
III.4 User Interface Design



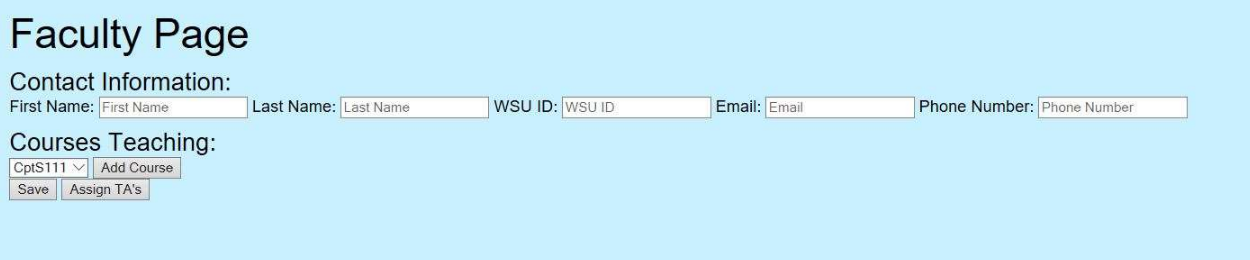
Example Login Frontend Design.(Above)



Example Registration Frontend Design(Above)



Example of student information frontend design (above)



Example of faculty information frontend design (above)

IV. Testing Plan

To test the project we will be implementing a mix of manual and automatic testing. We will test the server API routes using automated testing, we will implement test cases that should fail and succeed to make sure that the server is responding correctly to any request that could be sent to it. We will test functionality manually. We will go through the project and make sure that all the use cases are operating as expected, and make sure that the uses do not crash the system when used in ways that are unintended. We will also manually test our user interface to make sure that it works intuitively with users. We will also ensure that the user interface meets our collective understanding of quality graphic design. So far we've tested our User interface and made sure that they are working correctly for displaying. We've also successfully tested manually that login and register is working completely. We’ve tested with different cases however need to still write unit tests.