Be My TA

Requirements Specifications

TEAM BETA

Diwash Biswa, Drew McLaurin, Dayton Dekam

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Introduction

Provide a brief description of your project and summarize the objectives.

At WSU the School of Electrical Engineering and Computer Science has a system that students have to go through to become an undergraduate TA for lower-level classes that they have already taken. Currently, the system is manually done. The students of interest take a survey which is used for the faculty to know what courses they are interested in being a TA for. Then there are faculty members who have the job of emailing the course professor of interest to determine who that professor would like as a TA. Then they are able to assign the TA with the right professor and this process is also done manually. The problem with the current system of assigning TA's like this is that it takes a lot of time and creates extra work for the faculty members in charge of assignments and the professors who need TA's.

Our project is designed to solve this problem 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 student 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 Requirements Specification for this product. In this section, we describe who our customers, users and the stakeholders are. There's also a use case of each feature and other small requirements for our product.

Section III includes the User Interface Requirements for this product. In this section, we describe the main user interface of our product. We have included some sketches of our interface.

Document Revision History

Rev 1.0 2019-10-06	Started on the requirements document
Rev 1.1 2019-10-07	Completed the requirements document

II. Requirements Specification

The main features of our product include the ability for students to apply for the TA position for the course they choose and the ability for the professors to assign TA for courses. Our web application will have a separate page for the Instructor and the students who are applying for a teaching assistantship. See below for more detailed features and functions of our product (from project requirements).

On the student page, a student user can:

1. Create a student account and set the account password (username 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)
6. Student users can add/remove courses to/from their list of preferred courses.
7. Save the entered information.

On the instructor page, a faculty user can:

1. Create an instructor account and set the account password (username 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 sections 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.

II.1. Customer, Users, and Stakeholders

The stakeholder in this app is professor Sakire Arslan Ay. The customers and users of this app are the students that are prospective TA, current TAs, and the instructors that decide to use it to gain and manage TAs in their course.

II.2. Use Cases

This section will include the specification for your project in the form of use cases. The section should start with a short description of the actors involved (e.g., regular user, administrator, etc.) and then follow with a list of the use cases.

For each use case you should have at least: Name, Actors, Triggers (what initiates the use case), Preconditions (in what system state is this use case applicable), Actions (what actions will the code take to implement the use case), Postconditions (what is the system state after the use case is done), Acceptance tests (list one or more acceptance tests with concrete values for the parameters, and concrete assertions that you will make to verify the postconditions). Each use case should also have a field called "Iteration" where you specify in which iteration you plan to implement this feature.

Use case # 1

Name	Create account
Users	Students
Rationale	The user will enter a username(their WSU email) and create a password. Then, they will need to enter contact information (name, last name, email, phone).
Triggers	The user will select “Register as Student”, then click “Register” after all the data is entered
Preconditions	The user needs to enter all the account information

Actions	1. The user will select “Register as Student”, then enter all account data 2. After selecting “Register” the software will store the data of the user account in the backend 3. The software will also make the data visible and be able to edit the data
Postconditions	The account data is stored in the backend and visible to the user and the user can edit the data
Acceptance Tests	All the required user information is stored in the backend and can be visible on the user’s page
Iteration	Iterations #1

Use case # 2

Name	Create Account
Users	Professors
Rationale	The user will enter a username and password. Then enter contact information (name, last name, WSU ID, email, phone). They will also enter the courses s/he teaches currently.
Triggers	The user will select “Register as Professor”, then click “Register” after all the data is entered
Preconditions	The user needs to enter the account information
Actions	4. The user will select “Register as Professor” to then enter all account data 5. After selecting “Register” the software will store the data of the user account in the backend

	6. The software will also make the data visible and be able to edit the data
Postconditions	The account data is stored in the backend and visible to the user and the user can edit the data
Acceptance Tests	Make sure all information is stored correctly, make sure the professors can see the info, and make sure the professor can edit the data
Iteration	Iteration #1

Use case # 3

Name	Login
Users	Students, Professors
Rationale	When the users have created an account already and are returning to the application, they will be able to use their username(their WSU email) and password to log in. The system will check to see if the username or password is correct or not. If they are correct, then this login will take the users to the main user's page of the application depending on who's using it, student or professor
Triggers	The users enter their username and password and clicks "Login" button
Preconditions	The user needs to enter their correct username and password
Actions	1. The user enters username and password 2. The user clicks on "Login" button 3. The software will detect whether the username or password is correct or not 4. If all information are correct, the software responds by taking the user to the user's page of the application 5. The software displays correct user's page with user information

Postconditions	The user is taken to the user's page of the application
Acceptance Tests	Make sure the correct user's page is shown based on the given username and password.
Iteration	Iteration #1

Use case # 4

Name	Apply for TA position
Users	Students
Rationale	The students will enter major, cumulative GPA, expected graduation date, whether s/he served as a TA before, etc. This information will be stored in the backend and will be visible to the professor(s) of course interest.
Triggers	The students will select "TA Application" option to enter the application, then select "Apply" after entering all the information
Preconditions	The user needs to enter the application data
Actions	1. The user will select "TA Application" option to enter the application 2. After selecting "Apply" after entering all the information, the software will store all that data in the backend 3. The software will also make this information visible for professor users
Postconditions	The application information is stored in the backend and is visible to the professors of interest

Acceptance Tests	Make sure all information is stored correctly, make sure the professors can see the info, and make sure the student can make changes to account info
Iteration	Iteration #2, #3

Use case # 5

Name	Assign TA
Users	Professors
Rationale	The professors can see the list of students who have applied to become TA for selected courses. Professors can review each application and be able to accept and assign students to courses. In the list, there will be an option to assign students to certain courses.
Triggers	The user selects user from the list and chooses course and clicks on “Assign this TA”
Preconditions	There must be a list of students who have applied to become TA
Actions	1. User reviews the list of students who have applied to become a TA 2. User selects user from the list and clicks “Assign this TA” for certain course 3. The software responds by giving an alert message saying that the user has successfully assigned TA for that course 4. The software updates the backend of both the professor and students - the student is notified that he/she is accepted to a TA position
Postconditions	The alert message is displayed saying that the user has successfully assigned a TA for a particular course.

Acceptance Tests	Make sure that backend is updated successfully, and the student is notified that he/she is accepted to be a TA for the course h/she applied
Iteration	Iteration #2, #3

II.3. Non-Functional Requirements

Below are some of the non-functional requirements:

Software quality requirements:

- Response time: The system should respond at a reasonable speed. There shouldn't be much delay in response.
- Scalability: The system should be able to support as many users as there are students and professors.
- Robustness: The system should be able to withstand rigorous testing without problems or bugs.
- Availability: The system should be available 24/7 for use
- Security: The system shall not disclose any personal user information
- Recovery from failure: The developers should have the backup codes in case the application is down and not serving its purpose
- Allowances for maintainability and enhancement: The developers should be able to send software and/or bug fix updates.
- Allowances for reusability: The system will have features that can be reused in other applications.

Platform requirements:

- Technology: The system should be available for multiple technologies including PCs, laptops, mobile devices, etc.
- Environment: The system should be compatible with major browsers such as Chrome, Firefox, Edge, etc.

Process requirements:

- Our deliverable documents shall conform to the SCRUM method

III. User Interface Requirements

Homepage: The instructors and students will both be able to view the homepage and select one of two buttons: “I am a Student”, or “I am a Professor”.

Student Login: The student will login or click “Create Student Account”

Create Student Account: The student will enter their username, password, account information, and course preferences. There will be a “Submit” button located at the bottom of the entries.

Student Page: Students may view their courses and edit them.

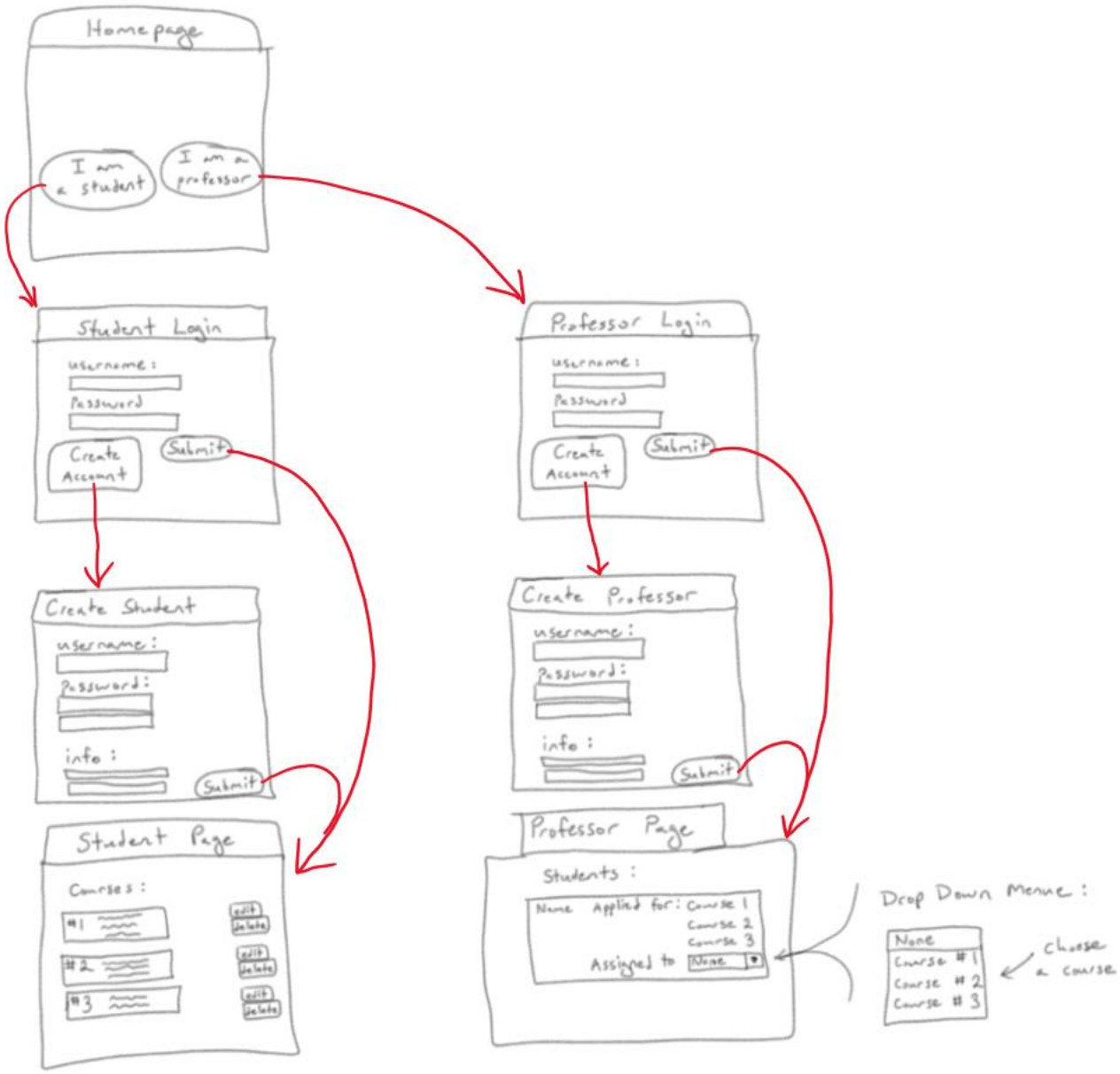
Professor Login: The instructor will login or click “Create Professor Account”

Create Professor Account: They will enter their contact information, courses they teach, and click the “Submit” button.

Professor Page: They will be able to view the students that applied, who are assigned to courses, and add them to courses or remove them from courses.

See diagram below for visual

User Interface



IV. References

CptS 322 – Fall 2019 Term Project, Shakire Arslan Ay. Link

Appendix – A (from Rubric):

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 a 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)