TA Portal

Requirements Specifications

Team 10

Logan Gorence

Marcus Sagisi

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Introduction

The goal of this project is to provide an easy-to-use application that will allow EECS students and faculty to easily find TA positions for classes. Prior to the beginning of the semester, a professor will enlist the courses that they are currently teaching on their profile. A student interested in being a TA will create a profile, and fill out a survey that will gather information from them and what classes they are interested in. The motivation of creating this application is to reduce the amount of labor required to manually collect survey results, validate them, and provide lists to professors to choose a TA.

In section II, we declare the requirements in terms of features, functionality, and specifications that we are required to implement in the product. In section III, we showcase sketches and mockups for the primary parts of the user interface.

Document Revision History

Rev 1.0 October 12th 2020 Initial version

II. Requirements Specification

In this section you will describe the features, functions, and other specifications that are requirements for your product.

II.1. Customer, Users, and Stakeholders

The customer is the school, since they would be the ones paying for the software for everyone to use. The stakeholders would be the instructors because they want a software that makes the TA finding process easier. The users are the students and teachers because the software was made for them to use.

II.2. Use Cases

Name	Create Account/Register
Users	Students
Rationale	In order to view and apply for open TA positions, a student needs to have an account with a WSU email. This would verify if the student is enrolled at WSU.
Triggers	“Register student” option
Preconditions	email has not been taken
Actions	1. The user indicates the software they want to register 2. The software responds by opening a registration page 3. The user enters credentials (e.g wsu email, password) 4. The user clicks register when done 5. The software responds by saving the credentials into the database and redirecting user to sign-in page
Postconditions	The user’s account is registered into the database
Acceptance Tests	Make sure the user can sign-in after registering
Iteration	Iteration 1

Name	Login
Users	Students and Teacher

Rationale	A student and teacher needs to be signed in order to access the options of their position. This is also to make sure if the user is part of WSU so we do not get bot accounts.
Triggers	If not already signed in, the page will redirect to sign-in page
Preconditions	WSU email is registered WSU email and password is right
Actions	1. The software responds by opening sign-in page if the user is not already signed-in 2. The user enters their email and password 3. The software responds by checking if the email and password is correct 4. If login is not correct, the software will flash incorrect message in sign-in page 5. If login is correct, the software will redirect the user to the homepage and set the current user as the one that signed in.
Postconditions	The current user is the one that signed in
Acceptance Tests	The user's account is in the home page if the login credentials are correct
Iteration	Iteration 1

Name	Additional Information
Users	Students
Rationale	The user should enter additional information such as major, GPA, etc. in-order for the instructor viewing their application to tell if the user is best fit for the position
Triggers	"Additional information" option
Preconditions	The user has an account and is logged in
Actions	1. The user tells the software that they want to add additional information 2. The software responds by opening the additional information page, which prompts the user for GPA, major, etc. 3. The user inputs the information they want to be displayed into the corresponding input boxes. 4. The software saves the given information into the database and flash a success message. 5. The software will stay in the "additional information" page in case the user still wants to edit their additional information
Postconditions	The additional information is saved in the database
Acceptance Tests	Ensure the user's additional information is saved in the database
Iteration	Iteration 2

Name	Enter past TAships
Users	Students
Rationale	The user should enter past experience as a TA (if any), in order for the instructor viewing their application to tell if the students is best fit for the position
Triggers	"Enter past TA experience" option

Preconditions	The user has an account and is logged in
Actions	1. The user tells the software they want to enter their past TAships 2. The software responds by bringing up a form for the user to enter their experience 3. The user inputs the corresponding information about their experience such as which course, semester, etc. The user will be able to add more TAships if he/she had multiple 4. The software will store the given information into the database and flash a success message 5. The software will stay in the "past experience" page in case the user still wants to enter more TA experience
Postconditions	The "TA experience" is saved in the database
Acceptance Tests	Ensure the user's TA experience is saved in the database
Iteration	Iteration 3

Name	View open TA positions
Users	Students
Rationale	The user should be able to view the open TA positions in order for the user to apply to the position they want. The page will also show recommended openings to make the choosing process easier for the user.
Triggers	"View Open Positions" option
Preconditions	The user has an account and is logged in
Actions	1. The user tells the software to view open positions 2. The software responds by bringing up a page that displays the TA openings. 3. The software will also show recommended TA positions for the user based on the user's grade history on a course they have taken. 4. The user will be able to pick one of the positions. 5. The software will bring up a information on the position that the user chose to view
Postconditions	The user and software are on the "open positions" page
Acceptance Tests	The open positions page has open positions to view
Iteration	Iteration 1

Name	Display Position Information
Users	Students
Rationale	The user should be able to view information on an open position in order for the user to see if they are best fit for the position and meet the requirements.
Triggers	Choosing a position from "open position" page
Preconditions	The user has an account and is logged in and is on the "open position" page
Actions	1. The user tells the software they want to see information on an open position

	2. The software will bring up a page that shows information on the open position, such as course title, semester, instructor information, and qualifications
Postconditions	The user is on the position's information page
Acceptance Tests	The position has a complete information page
Iteration	Iteration 1

Name	Apply for position
Users	Students
Rationale	The user should be able to apply to a position they deem qualified for them to TA for.
Triggers	Choosing "apply" option on the position's information page
Preconditions	The user has an account and is logged in and is on the position's information page
Actions	1. The user tells the software they want to apply for the open position 2. The software brings up a page with a form application for the position. 3. The user inputs corresponding information to form, such as grade for the class they are applying for, year and semester they took the course, and year and semester they are applying for. 4. The software saves the application to the database and sends the application to the instructor of that open position
Postconditions	The application is saved into the database and is sent to the instructor. The application should be pending for the student
Acceptance Tests	Check if the application has been sent to the instructor and is a pending application on the students end.
Iteration	Iteration 1

Name	Withdraw Application
Users	Students
Rationale	The user should be able to withdraw from a pending application if they want to change their mind.
Triggers	Choosing the "withdraw" option
Preconditions	The user should have a pending application(s)
Actions	1. The user tells the software they want to withdraw a pending application. 2. The software brings them to a page that shows all pending applications 3. The user tells the software to withdraw a specific application 4. The software then prompts the user if they really want to withdraw the application. 5. The user chooses either yes or no. 6. If no, the software will remain on the withdraw page with no changes 7. If yes, the software will withdraw the application from the instructor and delete it from the database. 8. After the application is withdrawn, the software will refresh the withdraw page and the specified application is no longer shown

Postconditions	The application is removed from the database
Acceptance Tests	Check if the application has removed from the database of the user and instructor
Iteration	Iteration 3

Name	Create teacher account
Users	Teacher
Rationale	To be able to create courses, and review students which have applied to be a TA for a course, a teacher needs to be able to register their own account.
Triggers	“Register Teacher” option
Preconditions	Email has not already been used by another account (student or teacher)
Actions	1. The teacher indicates the software they want to register 2. The software responds by opening a registration page 3. The teacher enters credentials (e.g WSU email, password) 4. The user clicks register when done 5. The software responds by saving the credentials into the database and redirecting user to sign-in page
Postconditions	The teacher’s account is registered into the database
Acceptance Tests	Make sure the teacher can sign-in after registering
Iteration	Iteration 1

Name	Edit account contact information
Users	Students and teachers
Rationale	Students and teachers should be able to edit their account’s contact information.
Triggers	“Edit account” option
Preconditions	User has an account and is signed in
Actions	1. The user indicates the software they want to edit account contact information 2. The software responds by opening contact information form with previous values filled into respective fields 3. The user edits one or more fields of contact information 4. The user submits the form 5. The software responds by editing the contact information and redirecting user to their profile
Postconditions	The user’s account is updated with the new information
Acceptance Tests	Ensure that the account information is persisted to the database
Iteration	Iteration 1

Name	Add or edit current courses
Users	Teachers

Rationale	Teachers should be able to add a list of courses currently being taught for student users to show interest in TAship for a course.
Triggers	“Edit courses” option
Preconditions	Teacher has an account and is signed in
Actions	1. The teacher indicates to the software that they want to add or edit their course list 2. The software responds by opening a form that will show them their current courses, if any and the form will allow teacher to add or remove courses 3. The teacher will add, edit, or remove courses 4. The teacher will submit the changes 5. The software responds by adding / editing / removing the courses in the database that the teacher has specified in the form
Postconditions	The courses are added to the teacher’s course profile
Acceptance Tests	Ensure that the course changes are persisted to the database
Iteration	Iteration 2

Name	Edit course metadata
Users	Teachers
Rationale	Teachers should be able to edit metadata for a course that they are teaching. These are pieces of data like number of TAs required, minimum GPA, minimum grade earned, prior TA experience.
Triggers	“Edit course metadata” option on a course
Preconditions	Teacher has an account, is signed in, and has created at least one course
Actions	1. The teacher indicates to the software that they want to edit the metadata for a course 2. The software responds by opening a form that will allow them to edit a course, and will populate any pre-existing data from the database 3. The teacher will edit the data that they need to 4. The teacher will submit the changes 5. The software responds by updating the course metadata in the database that the teacher has specified in the form
Postconditions	The courses metadata has been updated
Acceptance Tests	Ensure that the course metadata updates are persisted to the database
Iteration	Iteration 2

Name	View students that have applied
Users	Teachers
Rationale	Teachers should be able to view students that have applied for TAship of a teacher’s course.
Triggers	“View applied students” option on a course

Preconditions	Teacher has an account, is signed in, has created one course Student(s) has an account, has updated their profile, and has applied to the teacher's course
Actions	1. The teacher indicates to the software that they want to view the students that have applied to one of their courses, with some kind of selection for the course 2. The software responds by opening a list of students that have applied to the selected course
Postconditions	None
Acceptance Tests	Ensure that the students list contains the correct students
Iteration	Iteration 3

Name	View qualifications of student
Users	Teachers
Rationale	Teachers should be able to view the qualifications of an individual student that has applied for TAship of a teacher's course.
Triggers	"View student" option on the students applied page
Preconditions	Teacher has an account, is signed in, has created one course Student(s) has an account, has updated their profile, and has applied to the teacher's course
Actions	1. The teacher indicates to the software that they want to view a student that has applied to one of their courses 2. The software responds by opening the student's profile that the teacher has selected.
Postconditions	None
Acceptance Tests	Ensure that the student profile contains the correct qualification information
Iteration	Iteration 3

Name	Assign student to course
Users	Teachers
Rationale	Teachers should be able to assign a student to their course that has not already been taken
Triggers	"Assign student" option on the students applied page
Preconditions	Teacher has an account, is signed in, has created one course Student(s) has an account, has updated their profile, and has applied to the teacher's course
Actions	1. The teacher indicates to the software that they want to assign a student as a TA that has applied to one of their courses 2. The software will assign the student to the course **only if** there are still TA positions to fill, otherwise it will show an error. Also, if a student already has been assigned as a TA to another course, show an error to pick another student.

Postconditions	A student is assigned to the course
Acceptance Tests	Ensure that the student is assigned to the course in the database Ensure that if there are no TA positions left for a course, that an error will show if another student is assigned
Iteration	Iteration 3

II.3. Non-Functional Requirements

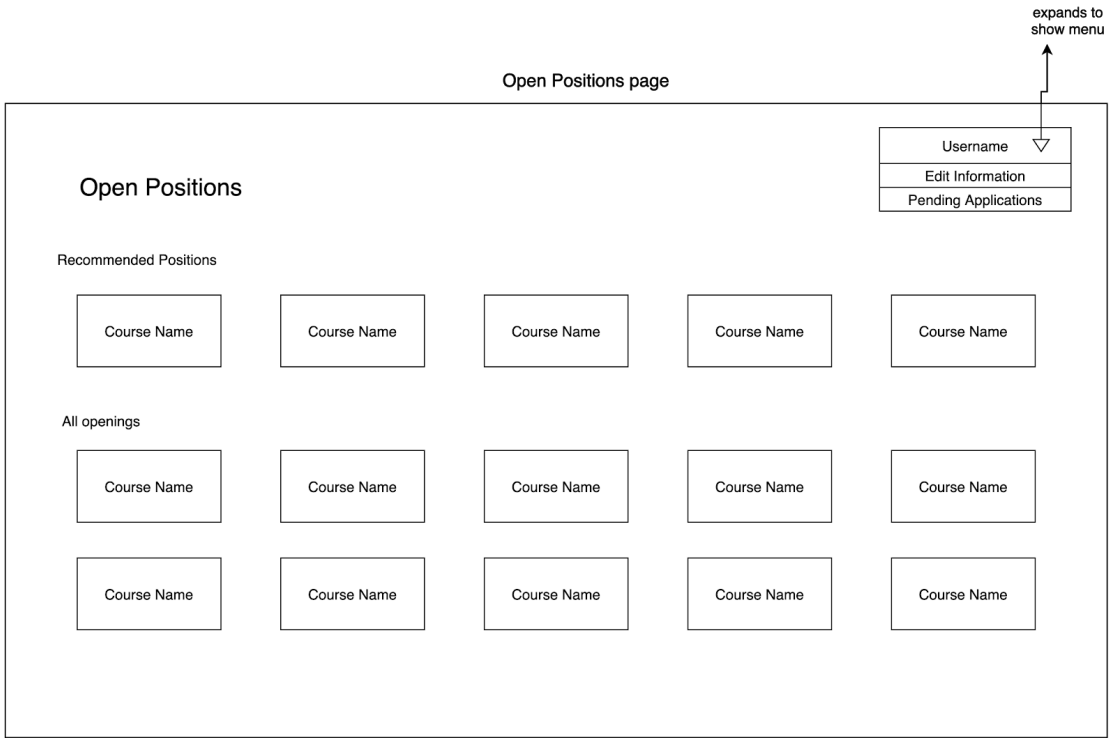
1. Browser Compatibility: The website should be compatible with all major browsers such as Google Chrome, Firefox, Safari, etc.

2. Throughput requirements: able to execute a given number (500 people) of transactions at a given time.

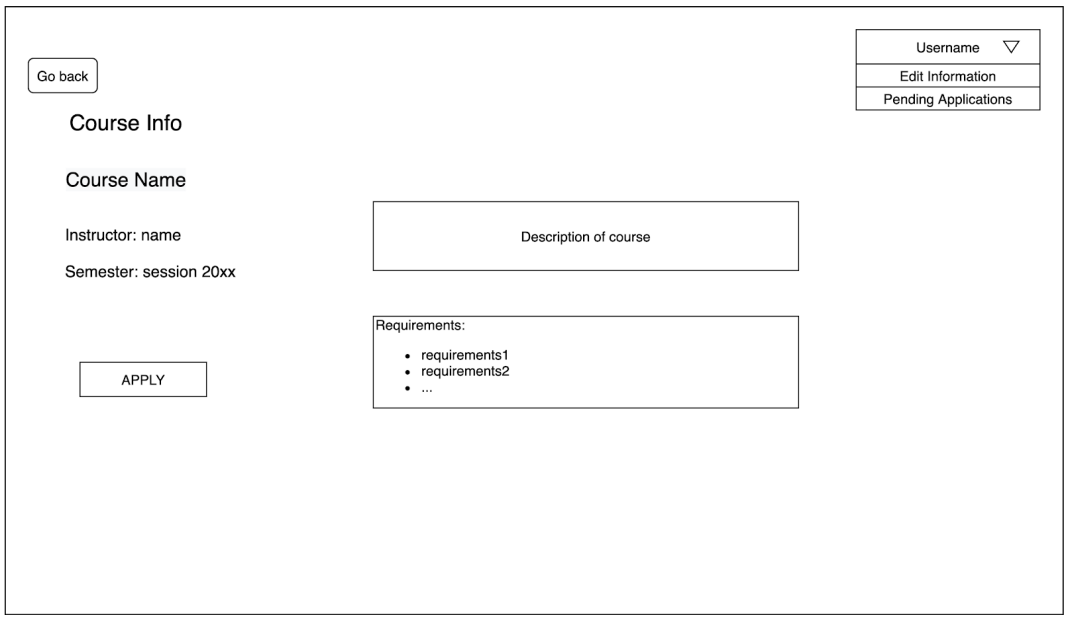
3. Response Times: An accepting time (<1 seconds)for the system to respond to requests on normal conditions

4. Security: The user's information shall not be distributed outside it's database

III. User Interface



Course info page



View student applicants page

This page contains a table of students that have applied to the course selected. For this mockup, we have added eleven uniquely named students. For each student, you may view their qualifications, and additionally you may assign or remove the student as a TA for the course. If the student has been selected for TAship of another course already, the "Assign Student" button will be disabled. Finally, it also includes the core pieces of the page, like the logged in user, a logout button, and a back button.

Students who have applied for Cpt_S 322

There are 1 positions filled for this course, 4 more are still needed

Logan Gorence	View Student	Assign Student
Marcus Sagisi	View Student	Assign Student
Student C	View Student	Remove Student
Student D	View Student	Assign Student
Student E	View Student	Assign Student
Student F	View Student	Assign Student
Student G	View Student	Assign Student
Student H	View Student	Assign Student
Student I	View Student	Assign Student
Student J	View Student	Assign Student
Student K	View Student	Assign Student

View courses page

This page contains a table of courses that have been created by the logged in professor. For this mockup, we have added three courses, 322, 355, and 451. For each class, you may view the applicants for that course, and additionally you may edit the course required qualifications or metadata. From this page, you can also navigate to the “New Course” page. Finally, it also includes the core pieces of the page, like the logged in user, a logout button, and a back button.

IV. References

Guru99. *What is Non-Functional Requirement? Types and Examples*. https://www.guru99.com/non-functional-requirement-type-example.html