**Team 11’s Default Project**

**Requirements Specifications**

**Team 11**

Eric Alvarez, Cole Hensley, Callum MacDonald

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

**TABLE OF CONTENTS**

**I.** **Introduction.. 2**

**II. Requirements Specification.. 3**

II.1. Customer, Users, and Stakeholders. **3**

II.2. Use Cases. **3-1**

II.3. Non-Functional Requirements. **13**

**III. User Interface Requirements.. 15**

**IV. References.. 15**

# 

# 

# 

# 

# 

# 

# 

# **I.** **Introduction**

For our term project we will be building the default project with Flask/Python and be utilizing SQL for our database. This application will facilitate the TA application process between WSU professors/faculty and students wishing to apply for a position. The goal of this application is to streamline this process of selecting TA’s for courses, as it can be quite tedious for professors to manually go through all applicants and consider students by all their preferences. The rest of this document will go into specifics of what users(students/faculty) are able to do, and functions of the application.

It is broken up into four main sections:

**Section I** - Introduction.

**Section II** - This section will describe the features and functions of the application in greater detail. It will cover all the users and all the functionalities available to each user with use case diagrams. It will also go over non-functional requirements which is how the application will fulfill the functional requirements, which includes but is not limited to: how/where the data of the application is stored, what programming principles we will follow as a team, and how we will communicate/manage issues.

**Section III** - Section three includes our specifications for the applications user interface. It will include a detailed description of the layout of our application as well as some diagrams for a visual aid/reference. This section will also distinguish all the different interfaces that will be implemented for each type of user including what it should look like for an unregistered user. This section will also include how the user interface may interact with the database and possible input fields that it will record.  
Section IV - In this section we will include any references to outside sources that we used to aid our design.

**Document Revision History**

Rev 1.0 2020-10-13 Added introduction

# **II.** **Requirements Specification**

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

## **II.1.** **Customer, Users, and Stakeholders**

Possible customers of this application would be universities looking to improve their TA hiring process, or even people looking to create a generic job searching site as this application will be very similar to a job search site. In this application there will be three main types of users: registered faculty user, registered student user, and unregistered user.

## **II.2.** **Use Cases**

The actors that will be involved with this application include the TA, the professor, and an unregistered user. The unregistered user will eventually become a TA or professor, but everyone will start out as being unregistered. That makes it an important case because a person may also try to join even if they’re not a TA or professor. The TA would be allowed to login, edit their information, and apply to certain TA positions. They also must meet the positions given requirements. The TA will also be allowed to withdraw an application from a position. A professor will be able to see certain applications from the TAs, edit their own information, make positions and the qualifications for those positions, and choose TAs for the positions. The last case is an unregistered user. They will be able to select if they are a TA or professor, and create an account based on what they select. They will be verified by their email, which must be a wsu.edu email.

**Use case # 1**

|  |  |
| --- | --- |
| Name | Registration |
| Users | All users |
| Rationale | Upon opening the application, an unregistered user needs some sort of functionality to input their information into the database and specify which type of user they are in order to take them to the right place and provide access to the right content for their type of user. |
| Triggers | An unregistered user opens the program, looking to register with the program. |
| Preconditions | A form is loaded and requires input in each field to create an account. |
| Actions | 1. The user indicates whether their account is a TA’s or a Faculty member’s 2. The user inputs their preferred personal information for their username (must be unique), and their password (hashed) 3. The user presses the submit button 4. The software takes said information and stores it in the database as a new user of specified type 5. The user is routed to the “enter contact information” page. |
| Postconditions | A new user now exists within the database and said user has access to their respective page |
| Acceptance Tests | Make sure that a user cannot be created with any of the same unique information the newly-registered user just provided. |
| Iteration | 1 |

**Use case # 2**

|  |  |
| --- | --- |
| Name | Login |
| Users | Student, Faculty |
| Rationale | The login function allows users access content relative to what type of user they are, it entails access to the use cases listed below assuming they are already a registered user. |
| Triggers | The user enters their login information to the form and clicks “Login” |
| Preconditions | The username and password fields must be filled in |
| Actions | 1. The user enters their login information and clicks “Login” 2. The software checks the validity of the login information provided by the user by comparing it to the list of registered users 3. If the login information is invalid then they will have access to their respective content 4. If the login information is invalid then they will be prompted that the username/password was incorrect |
| Postconditions | The user is either allowed access to their respective content or denied access |
| Acceptance Tests | Make sure the user is a registered user |
| Iteration | Iteration 1 |

**Use case # 3**

|  |  |
| --- | --- |
| Name | Entering contact information |
| Users | Student, Faculty |
| Rationale | The user must be able to enter in their information associated with the account. This includes contact information such as email, name, phone number, and more. |
| Triggers | Once an account is registered, they will be prompted to enter in their contact information. |
| Preconditions | All contact fields must be filled in |
| Actions | 1. Once the new account is logged in, the user will be prompted to fill in their information in separate fields. 2. The user will enter in their first and last name, WSU ID, email, and phone number. 3. Once all fields are filled in, the user will click a “done” button to tell the program they are finished entering their information. 4. If the user is a Student, they are routed to the “enter additional information” page. |
| Postconditions | Once they are finished, the information will then be uploaded to the database for the associated account. |
| Acceptance Tests | Make sure all information entered is correctly added to the proper account. |
| Iteration | Iteration 1 |

**Use case # 4**

|  |  |
| --- | --- |
| Name | Enter additional information |
| Users | Student users |
| Rationale | The Student must be able to provide their additional information that gives the Faculty a deeper understanding of who they are, if they’re qualified for the position at hand, and if their schedule works with the position as well. |
| Triggers | Once the user’s contact information is filled out, the user will automatically be prompted for additional information. |
| Preconditions | A form is loaded and requires input. |
| Actions | 1. Once the contact information is registered, the user will be prompted to fill out the “additional information” form 2. The user fills out fields specifying their major, cumulative GPA, and expected graduation date. 3. If the user has any previous courses they TA’d for, they can optionally list those in an optional input field. 4. The user clicks a submit button, and said information is stored in the database in accordance with their account 5. The user is routed to the login page |
| Postconditions | The user’s additional information is now in the database under said account |
| Acceptance Tests | Attempt to submit the additional information form without required fields, and with required fields, to make sure proper errors display and proper information is stored. |
| Iteration | Iteration 1 |

**Use case # 5**

|  |  |
| --- | --- |
| Name | Enter courses and course information |
| Users | Faculty |
| Rationale | Faculty users can enter the courses they teach into the application so students will be able to apply for a TA position to the course |
| Triggers | The faculty user selects “add course” |
| Preconditions | User needs to be signed into a faculty account |
| Actions | 1. The user indicates that they want to add a course 2. The software asks for course information 3. The user then enters course information 4. The software check for form validity if incorrect form submission repeat from action 3 5. The software add the course information into the database and is concurrently added the the list of classes that students can apply to be a TA for |
| Postconditions | The course is added to the databases |
| Acceptance Tests | Make sure the course is successfully added to the database as well as viewable courses to apply for on the student pages |
| Iteration | 2 |

**Use case # 6**

|  |  |
| --- | --- |
| Name | List of applications |
| Users | Faculty |
| Rationale | Faculty must be able to see the applications for the TA position of their course, and select which ones to be a TA |
| Triggers | The instructor will select the course they would like to view the applications, then select applications |
| Preconditions | The faculty member must have a course with the information filled in. |
| Actions | 1. The instructor will click on the course they want to view. 2. Once on the course page, they will be able to select applications. 3. All applications for the course will be shown 4. The faculty member can then click on an application 5. They will be able to hit accept, or deny for the chosen application. |
| Postconditions | A TA slot for the given course will be filled if the application was accepted. |
| Acceptance Tests | Check that the student is now in a TA slot for the course |
| Iteration | 2 |

**Use case # 7**

|  |  |
| --- | --- |
| Name | View the open TA positions |
| Users | Student |
| Rationale | The student must be able to see which TA positions are currently available to apply for |
| Triggers | The student page is loaded |
| Preconditions | A student is logged in and viewing the student page |
| Actions | 1. The user logs in to their account and is routed to the student access page 2. The software provides a primary table called “Recommended TA positions”, which matches the current student’s qualifications with TA positions asking for these qualifications 3. The software provides a secondary table with all the information regarding all current available TA positions, and each TA position is displayed with its respective course title, semester, instructor name and contact information, and qualifications required |
| Postconditions | The student page is loaded |
| Acceptance Tests | Test if recommended TA position table is filled with TA positions that accurately match classes with the qualifications of the current student user  Test if all the available TA positions are provided in the secondary table, even if they’ve been closed down by a faculty account |
| Iteration | Iteration 2 |

**Use case # 8**

|  |  |
| --- | --- |
| Name | Checking TA slots |
| Users | Faculty |
| Rationale | The faculty member needs to see the list of accepted TAs, and see if there are any empty, or duplicate, slots. |
| Triggers | The course has been selected by the faculty member |
| Preconditions | A course and its information must already be entered into the database by the faculty member |
| Actions | 1. The faculty member will select the course they want to view 2. The program will display current TA slots 3. The faculty member can see if there are more TAs than slots, or see if any TA is assigned to more than one slot |
| Postconditions | The slots are updated with the current information |
| Acceptance Tests | Check to make sure that the number of TAs accepted is less than or equal to the number of slots, and check no TA is in more than 1 slot. |
| Iteration | 3 |

**Use case # 9**

|  |  |
| --- | --- |
| Name | Apply for TA Positions |
| Users | Student |
| Rationale | The student must be able to apply for positions that they want to apply for based upon their qualifications |
| Triggers | A button is pressed routing the user to an application form |
| Preconditions | An application form is loaded and requires input |
| Actions | 1. The student presses an “apply” button on the student homepage  2. This button causes the software to route the student to a separate ‘/application’ URL that loads an application form  3. The software loads the application form and requires input for the course major, catalog number, and the semester/year desired for TAship   1. Once required boxes are filled, the user clicks an “apply” button which processes their application form and stores the information in the database under applications for the respective course applied for 2. The “apply” button routes the student user back to the student page, where they can apply to as many more TA positions as they like. |
| Postconditions | The application form with the student information is stored in the database under the respective course |
| Acceptance Tests | Test if the student can apply successfully to a course and have that application be sent to the correct course within the database  Test to make sure the student cannot submit multiple applications to the same course |
| Iteration | Iteration 2 |

**Use case # 10**

|  |  |
| --- | --- |
| Name | Withdraw a pending application |
| Users | Student |
| Rationale | The student must be able to withdraw a pending application for a variety of personal reasons from a course that they had previously applied to |
| Triggers | The student clicks the “withdraw” button |
| Preconditions | The student has one or more pending applications for TAship |
| Actions | 1. The student clicks a “withdraw” button that routes them to a page that displays all current pending applications within a table    1. If no current applications exist, the table will appear empty, and one button at the bottom of the page will show, stating “back to homepage” which routes the student to the student page from which they came 2. The software provides this table showing each pending application, and each application has an optional checkbox next to it 3. The user can check however many checkboxes they like, and then click the “withdraw selected applications” button underneath shown applications. 4. The software will take the applications with a checked checkbox, and remove said applications from the database entirely    1. If no applications are checked, the button will not submit properly, and will display an error message asking the student to check at least one application for removal 5. Upon deletion, the user will be routed back to the student page |
| Postconditions | The student page will be routed back to, and the desired removed pending applications will be removed from the database |
| Acceptance Tests | Check the database to make sure the applications are removed  Check to see if the withdraw page displays properly  Check if checkboxes work properly  Check if button to submit and remove checked applications displays an error if zero boxes are checked |
| Iteration | Iteration 3 |

## **II.3.** **Non-Functional Requirements**

1. There will be separate logins for a TA and faculty member: The user will have to select if they are a teacher or student. Then enter their login once chosen.

2. The course applications will be sorted by time: As a default method of showing the TA applications, the list will be automatically sorted from oldest to newest to show the students who submitted early above those who applied later.

3. Students can only apply to a course with a TA slot available: If a course has already filled its TA positions, then a student will not be allowed to apply to that course.

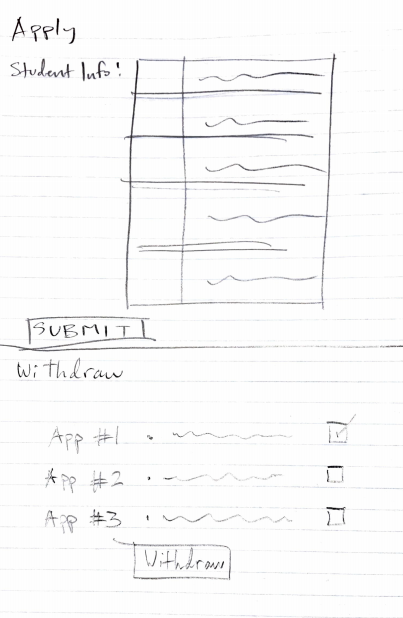
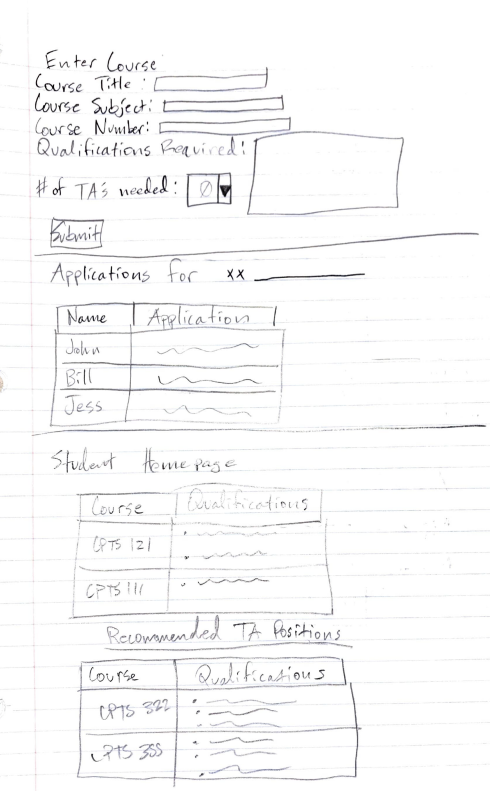
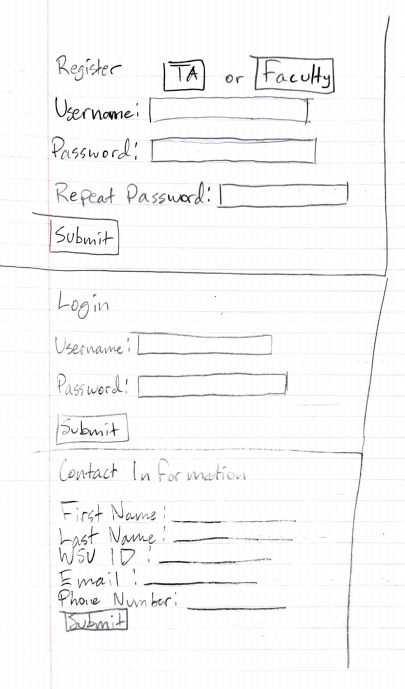
# 

# 

# 

# 

# **III.** **User Interface**



# 

# **IV.** **References**

Eriksson, Ulf. “Functional vs Non-Functional Requirements - Understand the Difference.” Difference.” *ReQtest*, 5 Apr. 2012, reqtest.com/requirements-blog/functional-vs-non-functional-requirements/.