Research Connect

Design Document

11/16//2021

Version 2

TeamRanks

Selina Nguyen

Sejal Welankar

Nick Kildall

Dan Rouhana

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I.	**Introduction**	2
II.	**Architectural and Component-level Design**	2
II.1.	System Structure	2
II.2.	Subsystem Design	3
II.2.1.	*Model*	3-6
II.2.2.	*Controller*	6-8
II.2.3.	*View and the User Interface Design*	8-12
III.	**Progress Report**	13
IV.	**Testing Plan**	13
V.	**References**	13
VI.	**Appendix: Grading Rubric**	13-14

I. Introduction

The purpose of this design document is to summarize the architecture and functional components of the web app we are developing. It details the connections between different models (including the many-to-many relationships) present, the routes that control program flow, and templates that render the web pages.

Our goal for this project is to create a framework for students and faculty to connect over research opportunities.

Section II describes the architectural and component-level design. The program system is divided into model, view, and controller subsystems. Model separates major functions and logic, view stores the HTML components required for rendering pages, and controller defines the routing between the pages.

Section III summarizes the progress completed by iteration one. All models were created, with rudimentary HTML pages for viewing and confirming features.

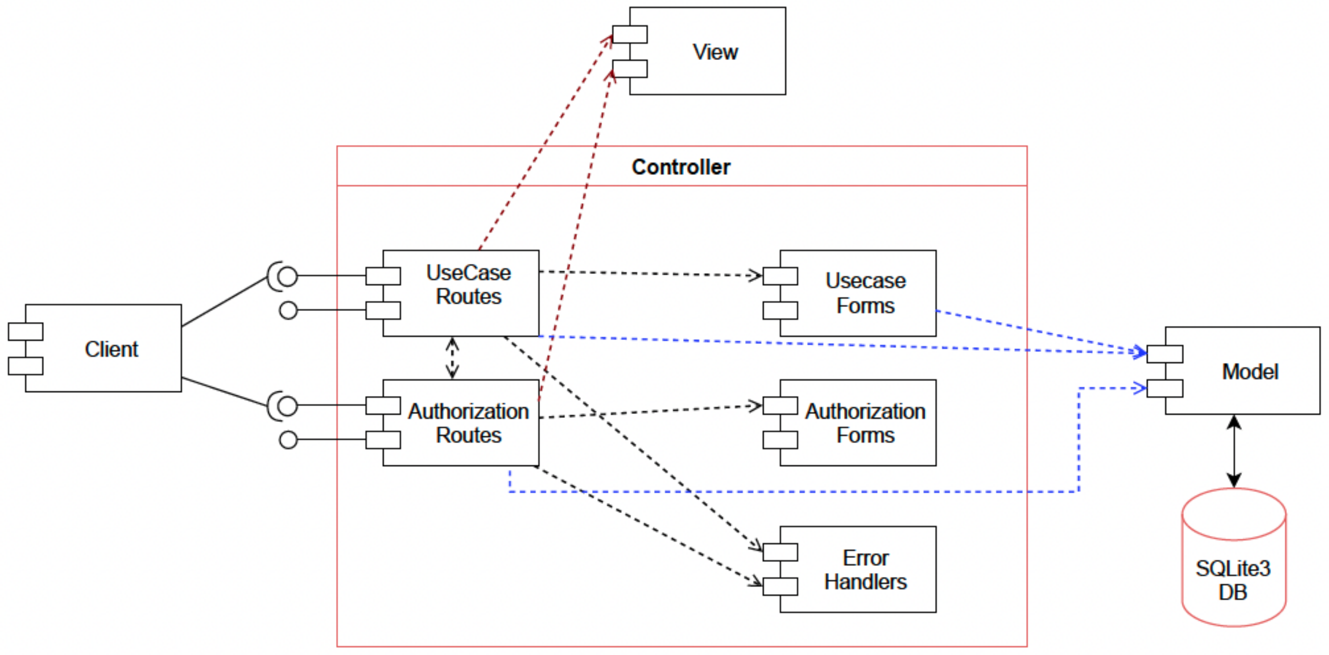
Document Revision History

Rev 2.0 2021-11-16 Iteration 2

II. Architectural and Component-level Design

II.1. System Structure

UML component diagram:



- Briefly explain the role of each subsystem in your architectural design and explain the dependencies between them.
    - Model
        - Controls data management and storage
        - provides data to the view via the controller
    - View
        - separates the HTML and the CSS from the model and controller
        - renders the appearance of data from the model in the user interface
        - Interacts with the model via the controller to update the html pages.
    - Controller
        - The controller takes user input to perform operations on the model
        - When the model changes the controller should update the view.

Rationale for the proposed decomposition in terms of cohesion and coupling.

- Generally we want our architecture to have low coupling to create systems that are independent from one another. By striving for low coupling, we make it simpler to make modifications to one subsystem without having a high impact on the other subsystems. To achieve low coupling we should ensure that calling classes need to know nothing about the internals of a called class. In our proposed decomposition we used the model, view, controller architecture to create low coupling.
- We want our architecture to have high cohesion to keep related items together and to keep unrelated things separated. This will make our system easier to understand and change. To achieve high cohesion we should make sure classes in a subsystem perform similar tasks and are related to one another. Additionally, we should ensure that most interactions occur within subsystems rather than across subsystems. In our proposed decomposition we used the model, view, controller architecture to keep similar subsystems clustered in the same directory. Additionally, we created subsystems in each directory that contain related items.

II.2. Subsystem Design

II.1.1. Model

The model is a way to separate the major functions and logic of our application. The model primarily controls data management and storage.

- Post
    - Description:Used to store the relevant information for a research opportunity.
    - Attributes:
        - id = an integer value that relates to the id of a post
        - title = a string value that relates to the title of the post
        - enddate = a string values that relates to the end date of the post
        - startDate = a string value that relates to the start date of the post
        - description = a string value that relates to the description of the post
        - faculty_id= a string value that relates to the faculty id that created that post
        - commitment = an integer value that relates to the time commitment of the post
        - qualifications = a string value that relates to the qualifications of the post

        - students_applied = a relationship to the Apply model. Contains a list of Apply objects for the specific posts. (We will likely delete this variable if it isn't utilized in iteration 3 — we currently query the database to get the same information)
        - interests = a relationship to the interest model that relates to the interests of the post
        - function get_interests = returns the interest of the post
        - function __repre__ = this function will display the id and title of the post
        - function get_students_applied = returns the students that applied to that posting (this function will also be deleted if 'students_applied' is deleted)

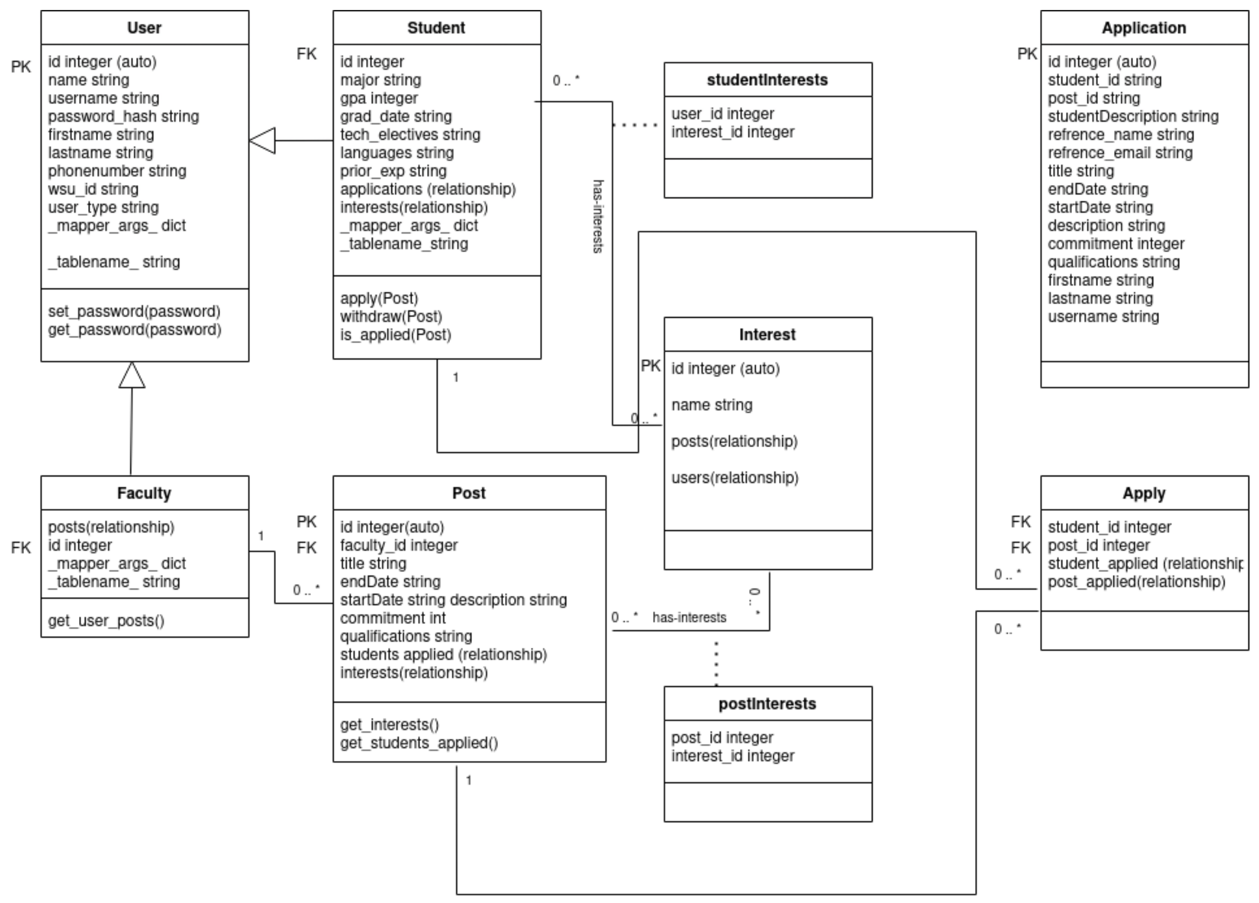
- user
    - Description: Stores information all users share
        - id = an integer value that relates to the id of a user
        - name = a string value that relates to the name of the user
        - username = a string value that relates to the username of a user
        - email = a string value that relates to the email of a user
        - password_hash = a string value that relates to the password of a user
        - firstname = a string value that relates to the first name of a user
        - lastname = a string value that relates to the last name of a user
        - phone_num =a string value that relates to the phone number of a user
        - wsu_id = a string value that relates to the wsu_id of a user
        - user_type = relates to the type of the user (faculty or student)
        - function set_password = this function will set the User's password to the string that is passed into the function
        - function get_password = this function will check if the password entered is the same password data as the User
        - function __repre__ = this function will display the user's name
- student
    - Description: Inherits user model and stores additional information relevant to student users.
    - Attributes:
        - id = a foreign key relationship to the user id
        - major = a string value that relates to the major of a student
        - gpa = a string value that relates to the gpa of a student
        - grad_date = a string value that relates to the graduation date of a student
        - tech_electives = string value that relates to the technical electives of a student
        - languages =   string value that relates to the prior language experience of a student
        - application = a relationship to the Apply model — contains a list of Apply objects for all posts the student user has applied to. (We will likely delete this variable if it isn't utilized in iteration 3 — we currently query the database to get the same information)
        - prior_exp =   string value that relates to the prior experience of a student
        - interests - a relationship to the Interest model that keeps track of the student's interests
        - function withdraw = withdraws (aka deletes) the current student user's application from the specified post

        - function apply = applies the current student user's application to the specified post
        - function is_applied = returns true if the current user is already applied to the specified post; if not, returns false.

- faculty
    - Description: Inherits user model
    - attributes:
        - id = a foreign key relationship to the user id
        - function get_user_posts= this function will return all the posts that that specific faculty has created
- Interest
    - Description: Represents a single user interest and creates a relationship to User
        - id = an integer value that relates to the id of a Interest
        - name = a string value representing the name of the interest
        - posts = describes a many-to-many relationship between interests and users
        - users = describes the relationship between the user model and the interest model
        - function __repre__ = this function will display the id and username of the Interest
- Apply
    - Description: Creates a many-to-many relationship between Students and Posts. This allows several students to apply to the same posts and allows students to apply to several posts.
    - attributes:
        - student_id = an integer value that is a foreign key to the student's id
        - post_id = an integer value that is a foreign key to the post's id
        - student_applied = a relationship to the student model
        - post_applied = a relationship to the post model
- Application
    - description: stores all the information for one application
    - attributes:
        - id = an integer value that relates to the ID of the application
        - student_id = an integer value that relates to the student id that applied to the position
        - post_id = an integer value that corresponds to the post id of the application
        - student_description = a string value that relates to the student's reasons of why the applied to the position
        - reference_name = a string value that relates to the reference name that the student filled in
        - reference_email = a string value that relates to the email of their reference
        - title = a string value that relates to the title of the position
        - endDate= a string value that relates to the endDate of the position
        - startDate = a string value that relates to the start date of the position
        - description = a string value that relates to the description of the position
        - commitment = an integer value that relates to the time commitment of the position

- qualifications = a string value that relates to the qualifications of the position
- firstname = a string value that relates to the first name of the student applying
- lastname = a string value that relates to the last name of the student applying
- function __repre__ = this function will display the id and title of the application

UML Database Model:



II.1.2. Controller

Briefly explain the role of the controller. If your controller is decomposed into smaller subsystems (similar to the Smile App design we discussed in class), list each of those subsystems as subsections. For each subsystem:

The controller takes user input to perform operations on the model. When the model changes the controller should update the view.

- Explain the role of the subsystem (component) and its responsibilities.
  - auth_forms
    - controls login information: Faculty & Student registration forms
    - Interacts with auth_routes
    - auth_routes implements form

- auth_routes
    - controls login routes: Faculty & Student registration routes
    - interacts with auth_forms. Commits session to database. Renders View
- errors
    - controls errors
    - interacts with the routing process
- forms
    - creates wtforms for non-login operations: edit form & some interests relations
    - routes implements form
- routes
    - controls non-login routes: edit profile, apply
    - interacts with forms. Commits session to database. Renders View

Table 1 - Research Connect Routes

Methods	URL Path	Description
get, post	/	routes to login page
get, post	/login	routes to login page
get	/logout	routes to logout page
get, post	/f_edit_profile	routes to edit profile page for faculty
get, post	/f_registration	routes to faculty registration page
get, post	/student_registration	routes to student registration page
get, post	/s_edit_profile	routes to edit profile page for student
get, post	/s_index	routes to student home page
get, post	/f_index	routes to faculty home page
get, post	/apply	routes to application page
get, post	/create_post	routes to faculty create application page
get, post	/application_review/<post_id>	routes to application review
get, post	/withdraw	Changes database by removing a students application, routes to student page
get, post	/s_your_app	Routes to the posts the student has applied to
get, post	/all_posts	Routes faculty to page displaying all research opportunity post
get, post	/delete/<post_id>	Changes database by removing a faculties post, routes to faculty home page
get	/applicants/<post_id>	Routes to all the applications for a posting
get, post	/apply/<postid>	routes to the application page for a position
get, post	/allposts	routes to all open positions for a faculty to see

II.1.3. View and User Interface Design

Role of the view:

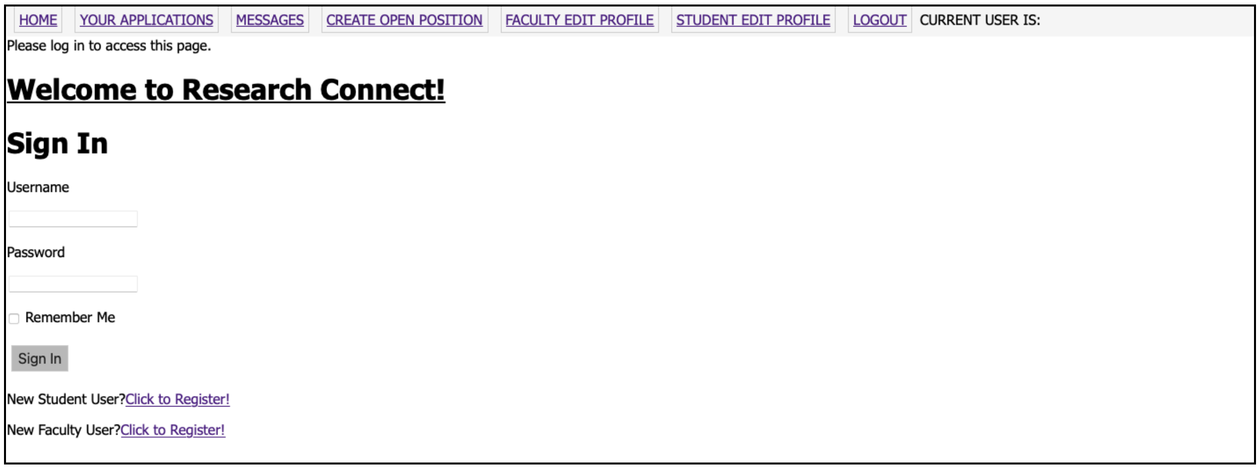
The architecture separates the HTML and the CSS from the model and controller. As a low level subsystem, the view renders the appearance of data from the model in the user interface. Interacts with the model to update the html pages.

Our current user interface displays several forms, including: FacultyEditForm, StudentEditForm, sortForm, PostForm, FacultyRegForm, StudentRegForm, ApplicationForm, and Login.

Use-cases satisfied in iteration one: Login, Faculty: Registration Page, Student: Registration Page, Create Research opening, Faculty: Edit Profile, and Student: Edit Profile

Use-Cases satisfied in iteration two: display open positions, display your posted positions, apply to research opening, review application, withdrawal pending application, logout

It would be redundant to show all of the forms (since they look very similar to the Faculty registration form), so here are some examples:



Please log in to access this page.

Welcome to Research Connect!

Sign In

Username

Password

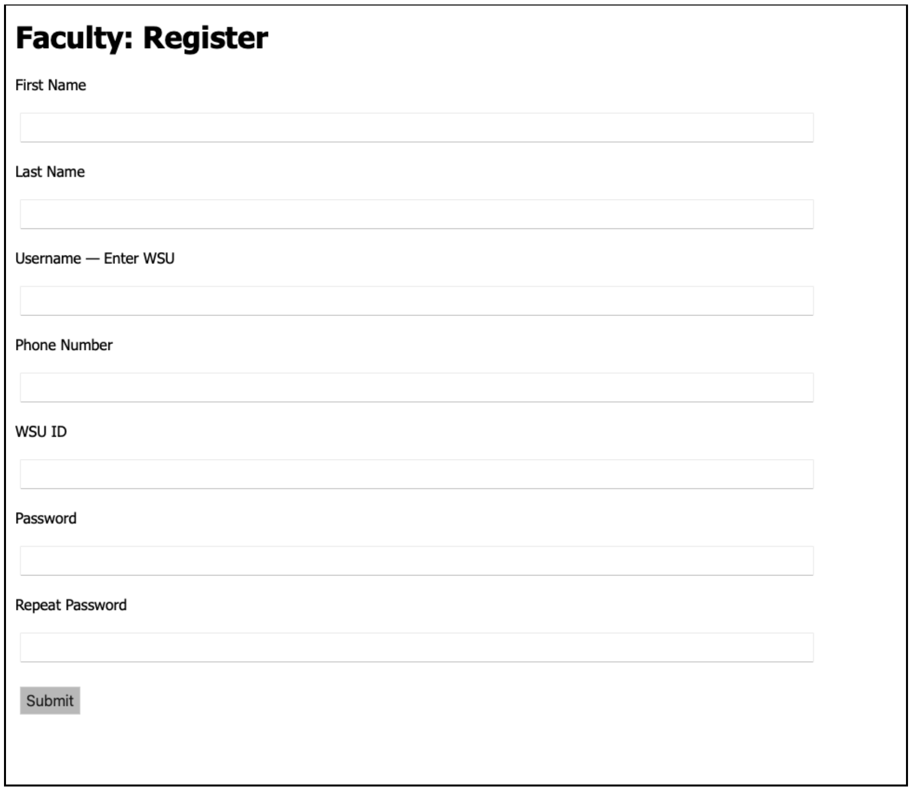
Remember Me

Sign In

New Student User?Click to Register!

New Faculty User?Click to Register!

Home Page. Implements: login



Faculty: Register

First Name

Last Name

Username — Enter WSU

Phone Number

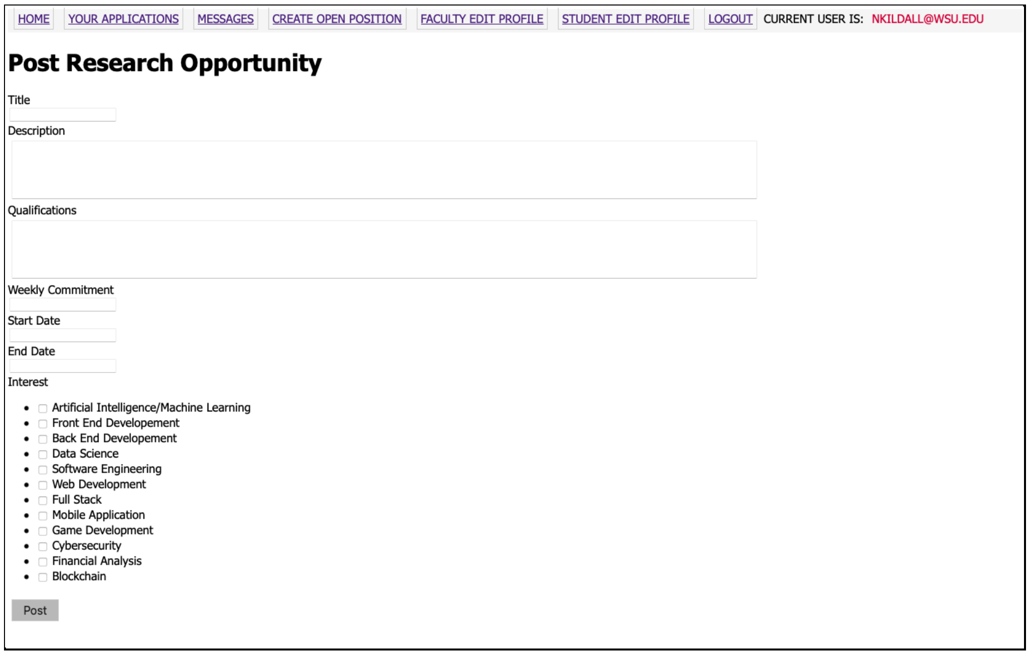
WSU ID

Password

Repeat Password

Submit

Faculty Registration. Similar forms include: FacultyEditForm, StudentEditForm, FacultyRegForm, StudentRegForm



HOME | YOUR APPLICATIONS | MESSAGES | CREATE OPEN POSITION | FACULTY EDIT PROFILE | STUDENT EDIT PROFILE | LOGOUT | CURRENT USER IS: NKILDALL@WSU.EDU

Post Research Opportunity

Title

Description

Qualifications

Weekly Commitment

Start Date

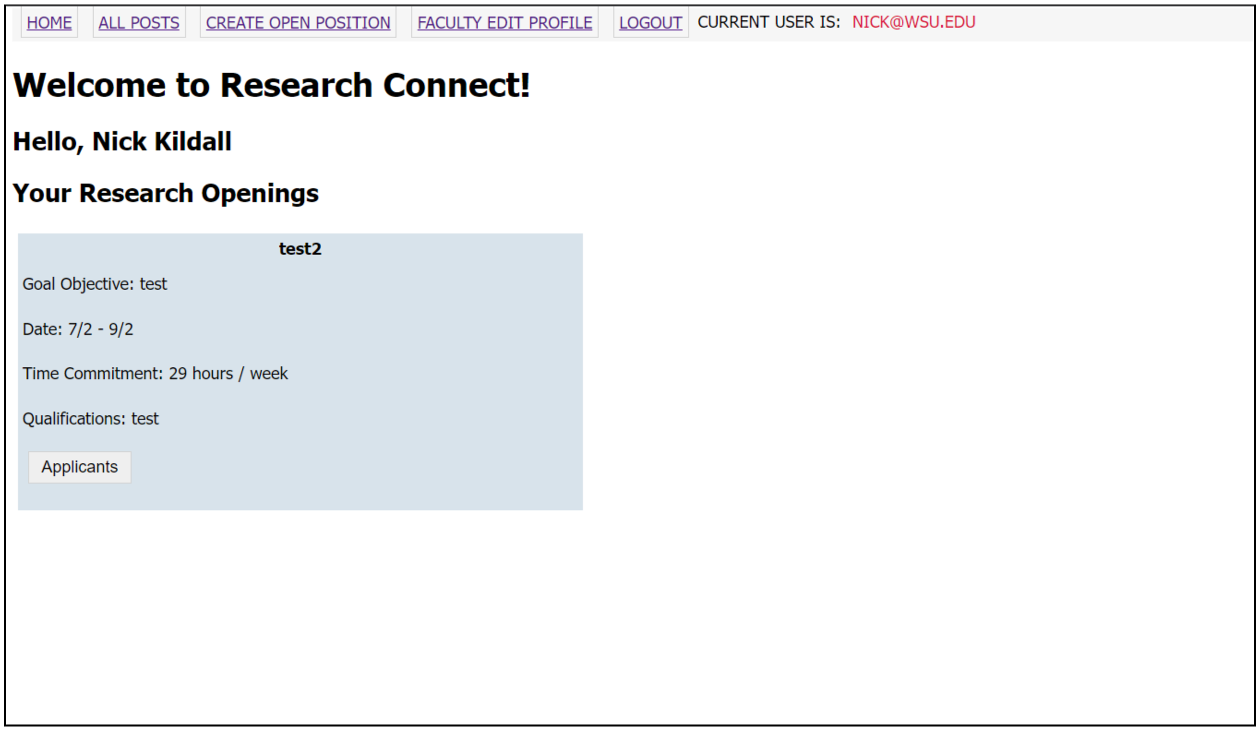
End Date

Interest

- ☐ Artificial Intelligence/Machine Learning
- ☐ Front End Developement
- ☐ Back End Developement
- ☐ Data Science
- ☐ Software Engineering
- ☐ Web Development
- ☐ Full Stack
- ☐ Mobile Application
- ☐ Game Development
- ☐ Cybersecurity
- ☐ Financial Analysis
- ☐ Blockchain

Post

Post Research Opportunity. Implements: PostForm (Note: StudentEditForm & StudentRegForm include the same interest checkbox list as shown above)



CURRENT USER IS: NICK@WSU.EDU

Welcome to Research Connect!

Hello, Nick Kildall

Your Research Openings

test2

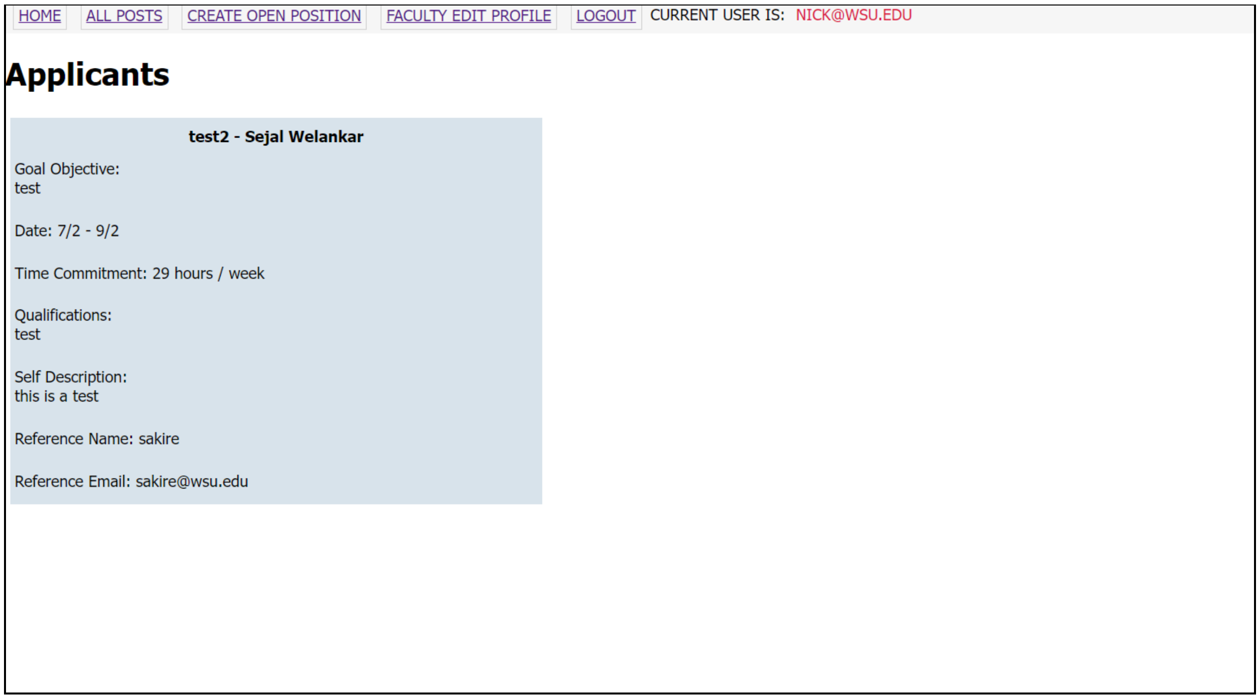
Goal Objective: test

Date: 7/2 - 9/2

Time Commitment: 29 hours / week

Qualifications: test

Faculty Home page: Implements the faculty home page use case



CURRENT USER IS: NICK@WSU.EDU

Applicants

test2 - Sejal Welankar

Goal Objective:
test

Date: 7/2 - 9/2

Time Commitment: 29 hours / week

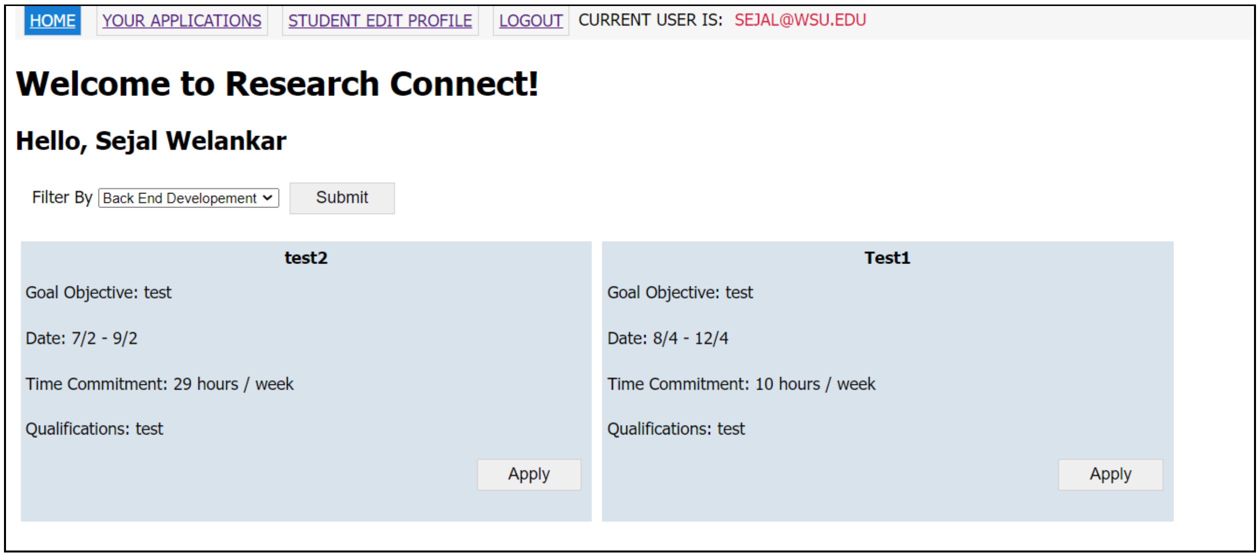
Qualifications:
test

Self Description:
this is a test

Reference Name: sakire

Reference Email: sakire@wsu.edu

Applicants: Implements Review Application use case



CURRENT USER IS: SEJAL@WSU.EDU

Welcome to Research Connect!

Hello, Sejal Welankar

Filter By Back End Developement Submit

test2

Goal Objective: test

Date: 7/2 - 9/2

Time Commitment: 29 hours / week

Qualifications: test

Apply

Test1

Goal Objective: test

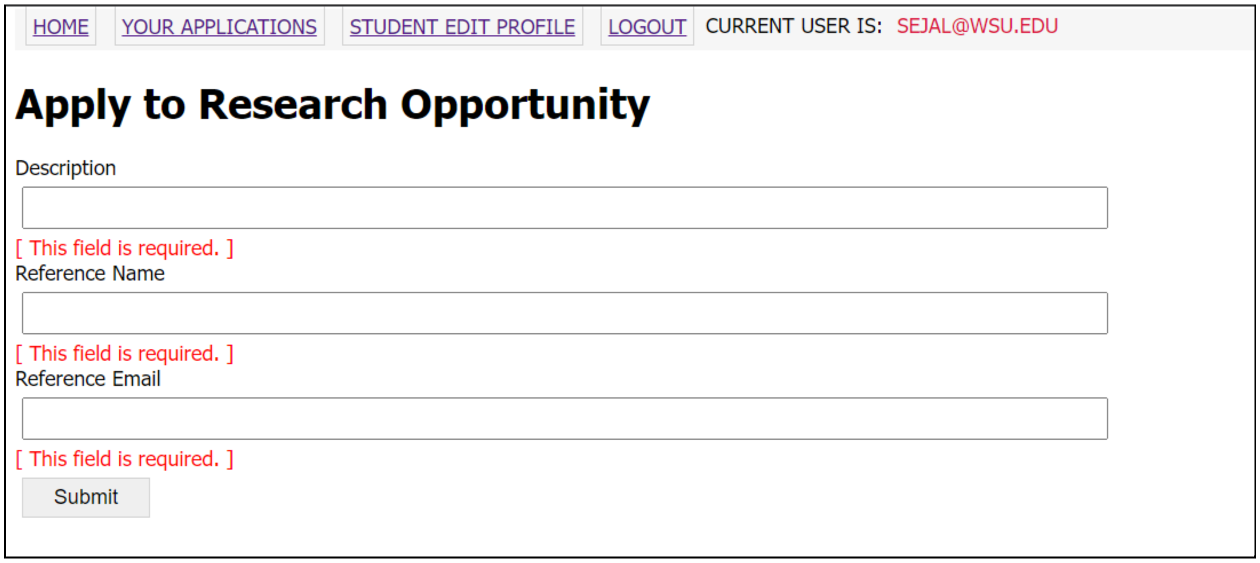
Date: 8/4 - 12/4

Time Commitment: 10 hours / week

Qualifications: test

Apply

Student Home page: Implements the student home page and display open positions use case



CURRENT USER IS: SEJAL@WSU.EDU

Apply to Research Opportunity

Description

[This field is required.]

Reference Name

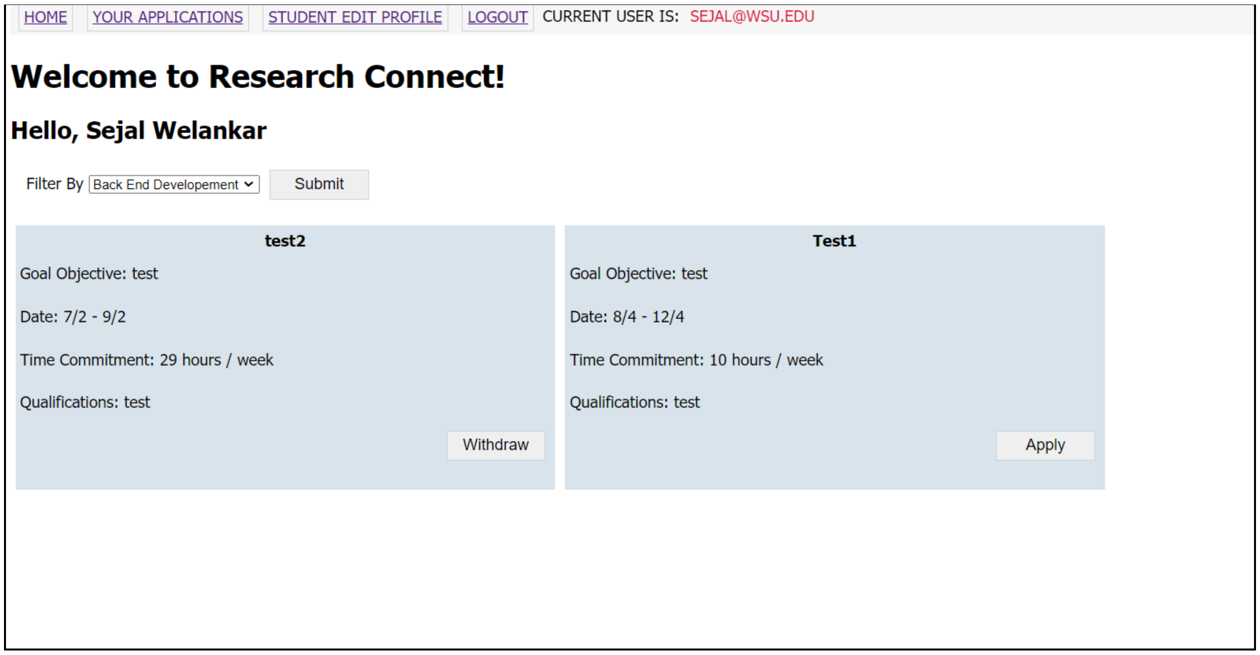
[This field is required.]

Reference Email

[This field is required.]

Submit

Application: Implements the application for a research position use case



CURRENT USER IS: SEJAL@WSU.EDU

Welcome to Research Connect!

Hello, Sejal Welankar

Filter By Back End Developement Submit

test2

Goal Objective: test

Date: 7/2 - 9/2

Time Commitment: 29 hours / week

Qualifications: test

Withdraw

Test1

Goal Objective: test

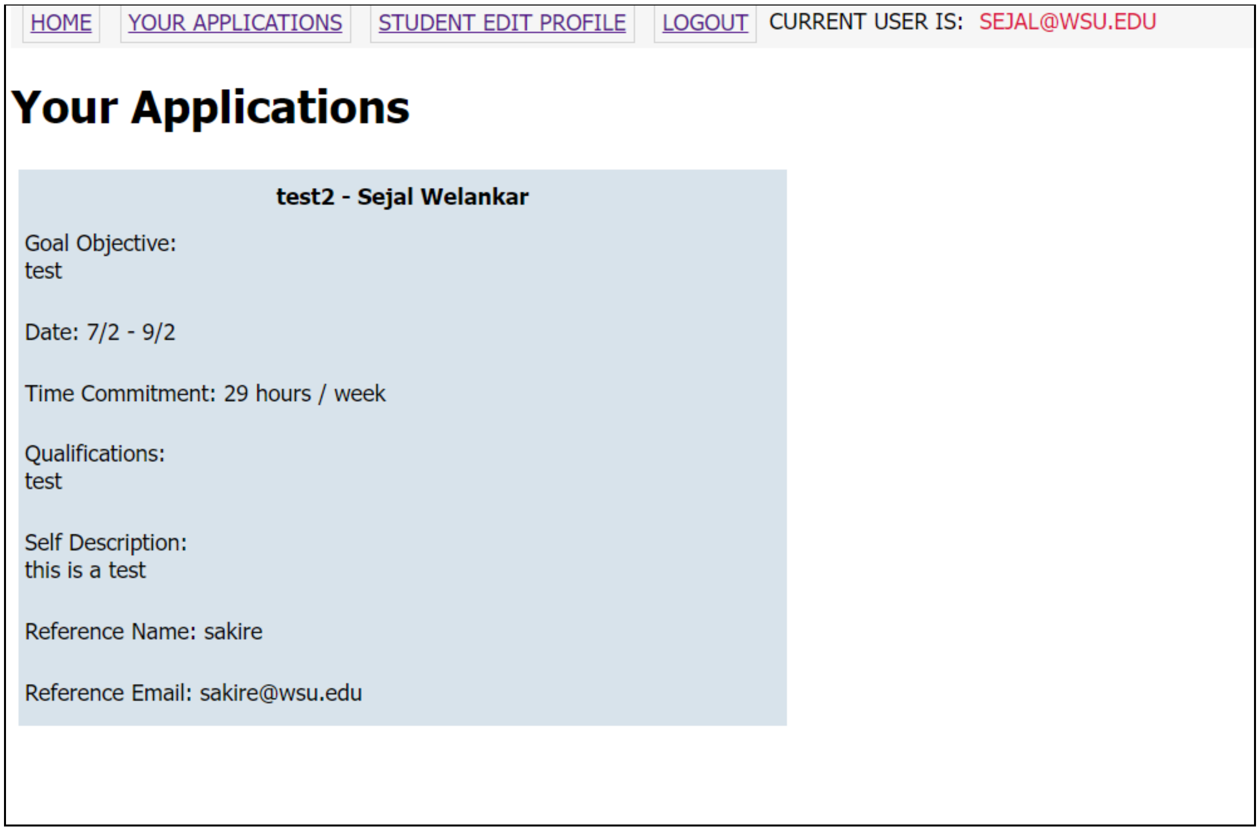
Date: 8/4 - 12/4

Time Commitment: 10 hours / week

Qualifications: test

Apply

Withdraw: Implements the withdrawal use case



CURRENT USER IS: SEJAL@WSU.EDU

Your Applications

test2 - Sejal Welankar

Goal Objective:
test

Date: 7/2 - 9/2

Time Commitment: 29 hours / week

Qualifications:
test

Self Description:
this is a test

Reference Name: sakire

Reference Email: sakire@wsu.edu

Your Applications: Implements the your application use case

III. Progress Report

In iteration 1 we laid the foundation for the web app. We implemented the basic database models needed, i.e Users, Interests and Posts. We also created flask forms and html pages for implementing registration for student and faculty, login, creation of research openings, editing profiles for both students and faculty.

In iteration 2, we focused on the the different users, Students and Faculty by splitting up the functionalities. For student users, we implemented the application form for when students want to apply to open research positions as well as the withdrawal functionality. For faculty, we implemented reviewing applicants for their own research positions.

IV. Testing Plan

In each iteration, we are implementing new and more functional features for the web app. In order to confirm the cohesive functionality of each feature, we will implement a set of tests to evaluate them.

First, we will conduct unit tests. As these will need to be performed repeatedly and iteratively, they are a great opportunity to automate testing for all aspects of our modules. For example, we can create unit tests that verify login credentials, validate logged in user types, and confirm models are linked correctly in the database (via many-to-many relationships). One framework we can use is xUNIT. This breaks up unit testing into the system under test, the dependant component, and the test fixture/method/case. These will be explicitly defined in their own test classes, which can be run at any point in time to confirm proper unit functionality.

Second, we can conduct functional tests. Many of these will have to be manual, but they allow us to experimentally validate the functionalities of our web app. Many of the functions are connected and dependent on the other's success (example: deleting a post relies on the correct creation of that post). We will also have to verify that access to certain functionalities are properly restricted or allowed (example: students should not be able to delete a faculty post). Critical features are also ideal to be tested with functional tests (example: only correct credentials can be used for login).

Lastly, we will need to test the user interface. This sort of testing lends itself almost entirely to manual testing. The finer details, such as page appearance, are entirely subjective and cannot be left to "pass/fail" by an automated test. However, some specifics must be explicitly verified. For example, "expected weekly requirement" for a posting should not accept an email or string as input. Manual testing is slower and (sometimes) more tedious than automated testing, but is the only way to ensure the proper functionality of the UI.

V. References

VI. Appendix: Grading Rubric

(Please remove this page in your final submission)

These is the grading rubric that we will use to evaluate your document.

Iteration-1

Max Points	**Design**
	Are all parts of the document in agreement with the product requirements?
10	Is the architecture of the system described well, with the major components and their interfaces? Is the rationale for the proposed decomposition in terms of cohesion and coupling explained well?
15	Is the document making good use of semi-formal notation (i.e., UML diagrams)? Does the document provide a clear and complete UML component diagram illustrating the architecture of the system?
15	Is the model (i.e., "database model") explained well with sufficient detail?
10	Is the controller explained in sufficient detail?
20	Are all major interfaces (i.e., the routes) listed? Are the routes explained in sufficient detail?
10	Is the view and the user interfaces explained well? Did the team provide the screenshots of the interfaces they built so far.
5	Is there sufficient detail in the design to start Iteration 2?
5	Progress report
	Clarity
5	Is the solution at a fairly consistent and appropriate level of detail? Is the solution clear enough to be turned over to an independent group for implementation and still be understood?
5	Is the document carefully written, without typos and grammatical errors?