Research Connect

Design Document

10/27//2021

Version I

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

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.

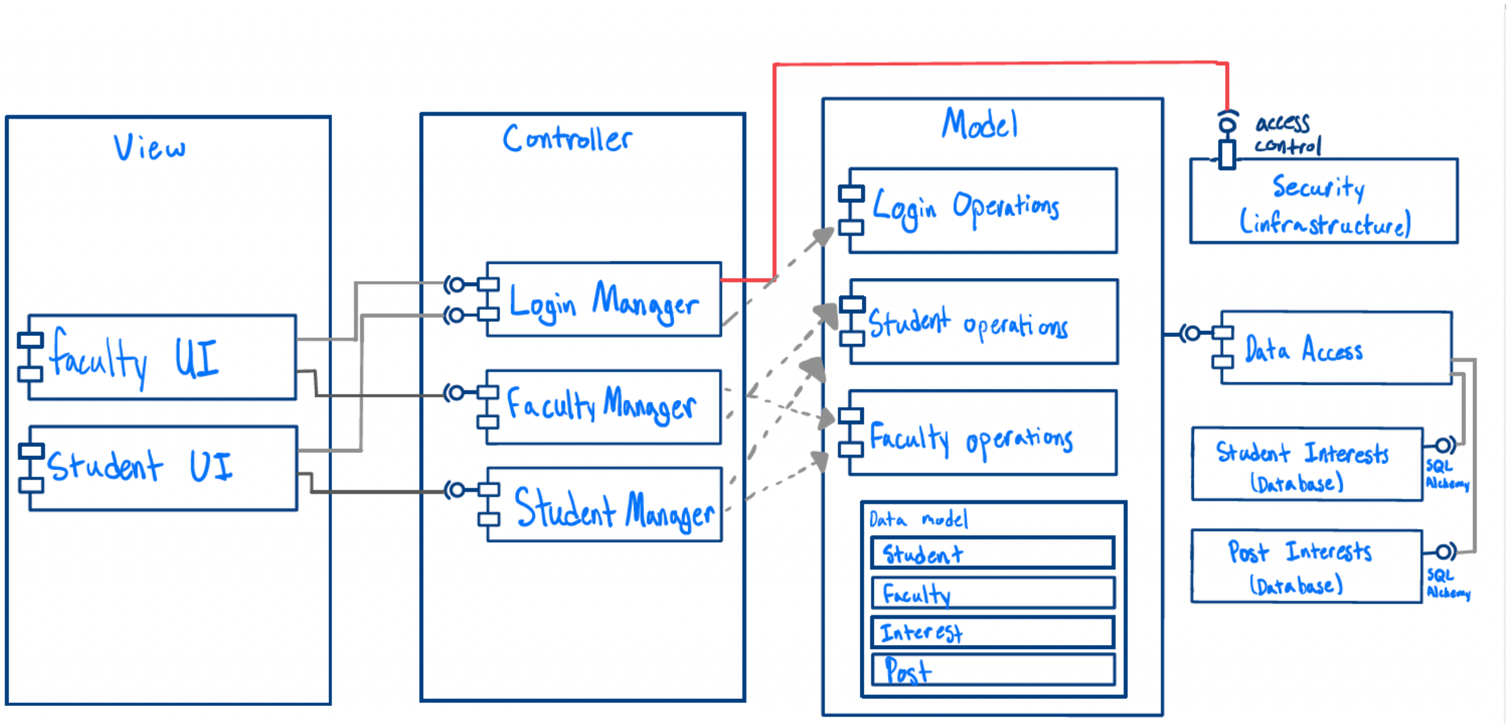
Document Revision History

Rev 1.0 2021-10-27 Iteration 1

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

- user
    - Description: Stores information all users share
        - id = an integer value that relates to the id of a 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
        - 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 id and username of the User
- student
    - Description: Inherits user model and stores additional information relevant to student users.
    - Attributes:
        - 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
        - prior_exp = string value that relates to the prior experience of a student

- faculty
    - Description: Inherits user model
- 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
        - function __repre__ = this function will display the id and username of the Interest
- userInterests
    - Description: Represents each user's interests

(in iteration -2) Revise the database model. Provide a UML diagram of your database model showing the associations and relationships among tables. Your UML diagram should also show the methods of your models.

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

(*in iteration-2*) Revise your route specifications, add the missing routes to your list, and update the routes you modified. Make sure to provide sufficient detail for each route. In iteration-2, you will be deducted points if you don't include all major routes needed for implementing the required use-cases or if you haven't described them in detail.

Table 1- Use an appropriate title for the table.

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	/edit_profile	routes to edit profile page
get, post	/user_registration	routes to user registration page
get, post	/student_home	routes to student home page
get, post	/faculty_home	routes to faculty home page
get, post	/apply	routes to application page
get, post	/create_application	routes to faculty create application page
get, post	/inbox	routes to messages page
get, post	/application_review/<post_id>	routes to application review
get, post	/withdraw_application	Changes database by removing a students application, routes to student page
get, post	/your_applications	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	Changes database by removing a faculties post, routes to faculty home page
get, post	/send_message	Routes to send message page

II.1.3. View and User Interface Design

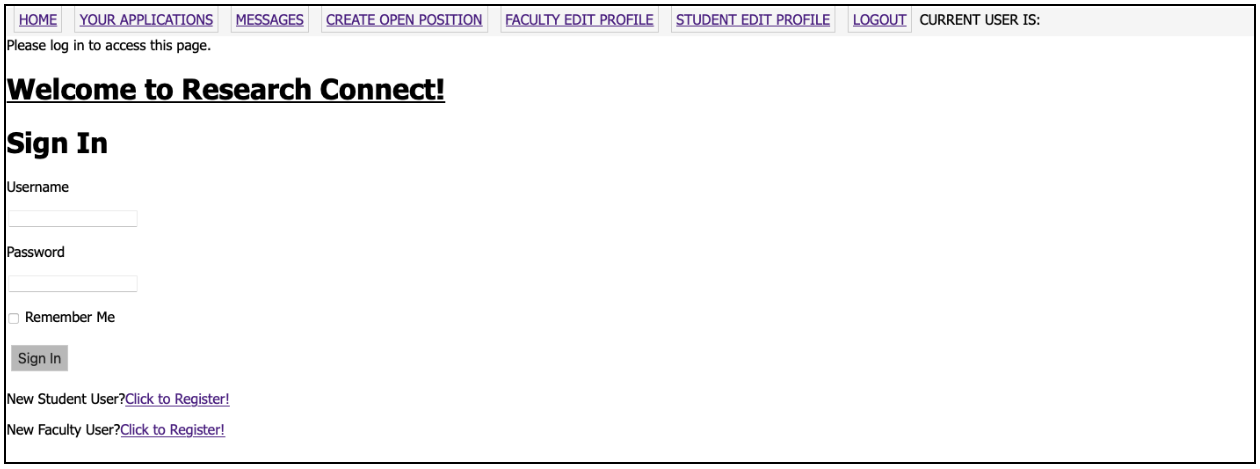
Role of the view:

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, 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

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:



HOME | YOUR APPLICATIONS | MESSAGES | CREATE OPEN POSITION | FACULTY EDIT PROFILE | STUDENT EDIT PROFILE | LOGOUT | CURRENT USER IS:

Please log in to access this page.

Welcome to Research Connect!

Sign In

Username

Password

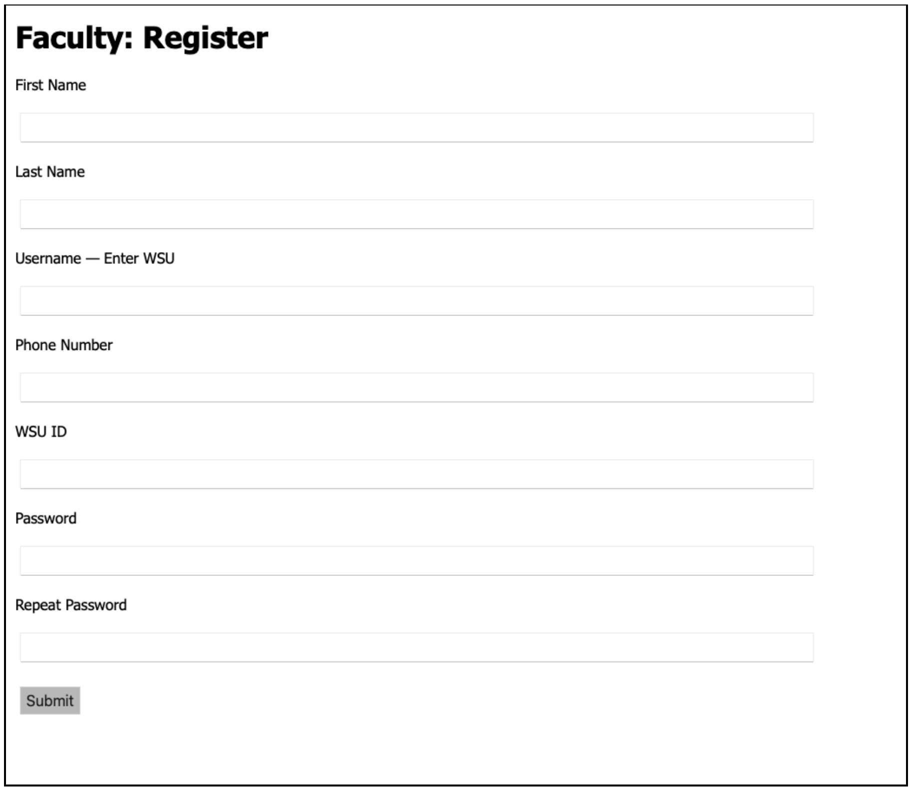
Remember Me

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

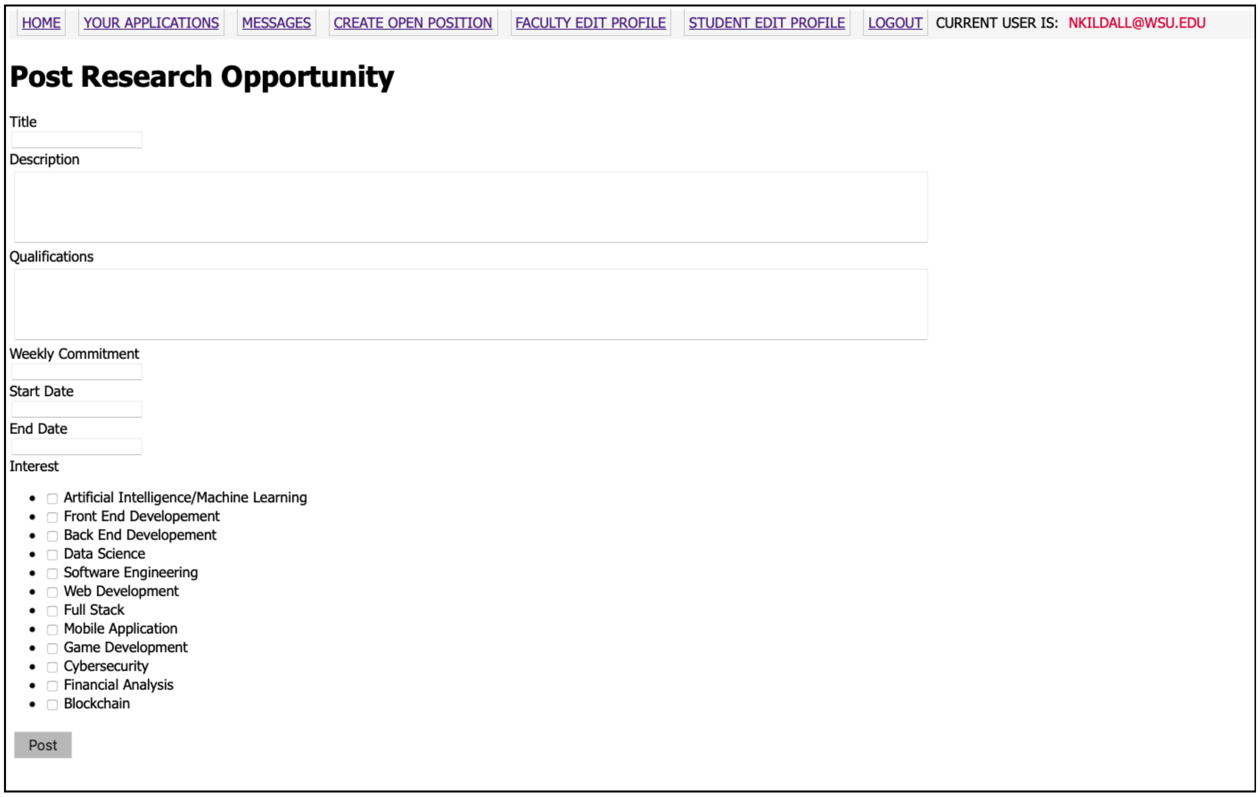
WSU ID

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

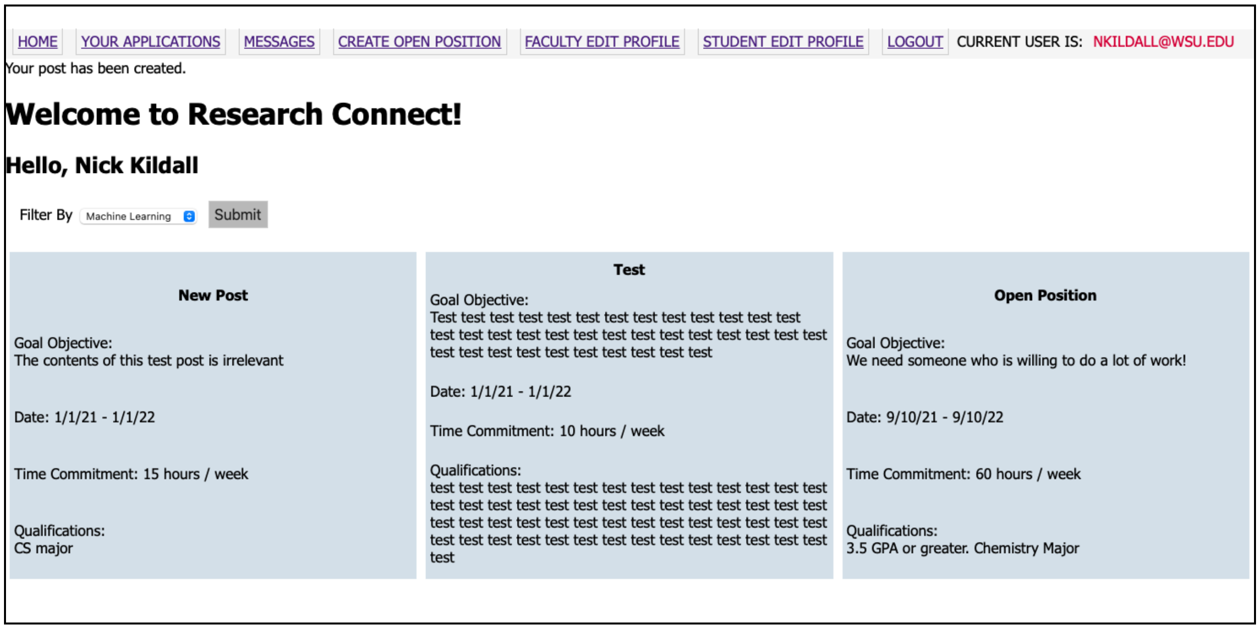
Interest

- ☐ 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)

Additionally we display a general home page showing all posts created by faculty members — the application takes the user's screen size to fit posts to their view:


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

Your post has been created.

Welcome to Research Connect!

Hello, Nick Kildall

Filter By Machine Learning Submit

New Post

Goal Objective:
The contents of this test post is irrelevant

Date: 1/1/21 - 1/1/22

Time Commitment: 15 hours / week

Qualifications:
CS major

Test

Goal Objective:
Test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test

Date: 1/1/21 - 1/1/22

Time Commitment: 10 hours / week

Qualifications:
test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test

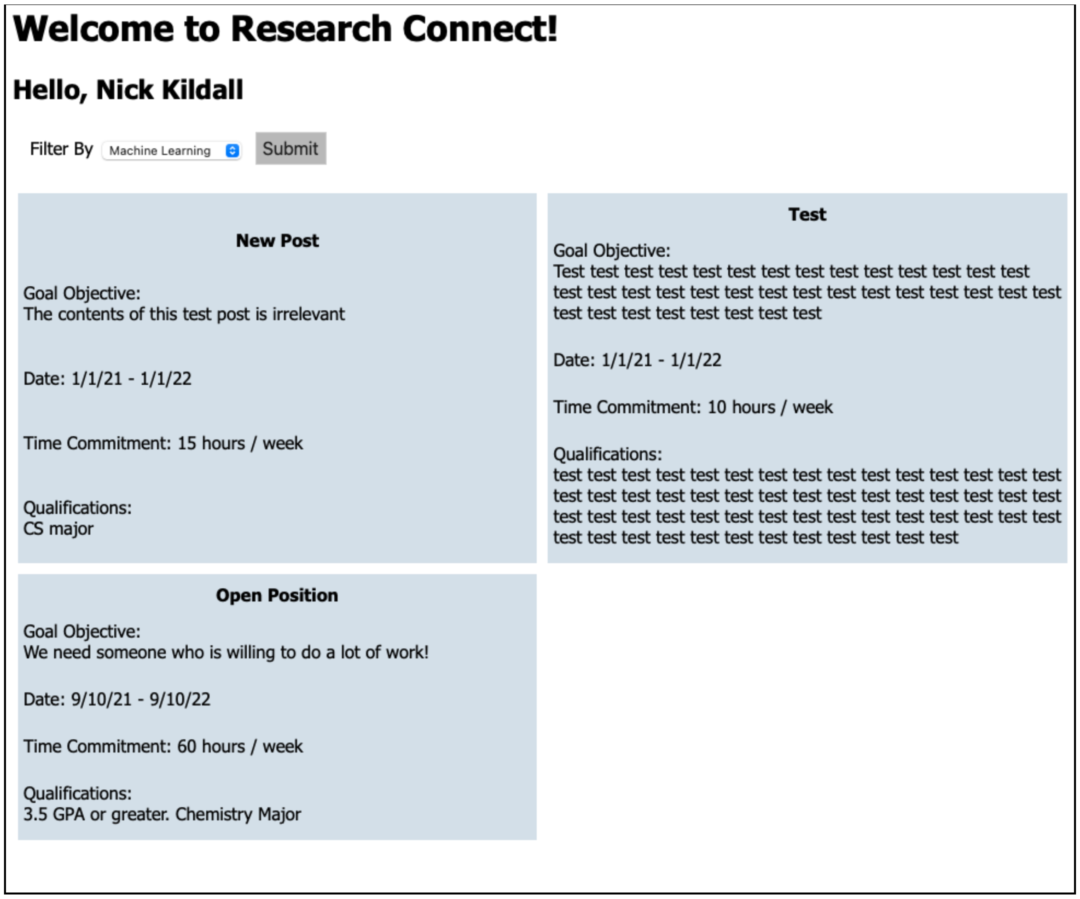
Open Position

Goal Objective:
We need someone who is willing to do a lot of work!

Date: 9/10/21 - 9/10/22

Time Commitment: 60 hours / week

Qualifications:
3.5 GPA or greater. Chemistry Major


Welcome to Research Connect!

Hello, Nick Kildall

Filter By Machine Learning Submit

New Post

Goal Objective:
The contents of this test post is irrelevant

Date: 1/1/21 - 1/1/22

Time Commitment: 15 hours / week

Qualifications:
CS major

Test

Goal Objective:
Test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test

Date: 1/1/21 - 1/1/22

Time Commitment: 10 hours / week

Qualifications:
test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test

Open Position

Goal Objective:
We need someone who is willing to do a lot of work!

Date: 9/10/21 - 9/10/22

Time Commitment: 60 hours / week

Qualifications:
3.5 GPA or greater. Chemistry Major

Satisfies the following use-cases: Display open research positions & sortForm

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.

IV. Testing Plan

(***in iteration 2***)
In this section , provide a brief description of how you plan to test the system. Thought should be given to how mostly automatic testing can be carried out, so as to maximize the limited number of human hours you will have for testing your system. Consider the following kinds of testing:

- **Unit Testing:** Explain for what modules you plan to write unit tests, and what framework you plan to use. (Each team should write automated tests (at least) for testing the API routes)
- **Functional Testing:** How will you test your system to verify that the use cases are implemented correctly? (Manual tests are OK)
- **UI Testing**: How do you plan to test the user interface? (Manual tests are OK)

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?