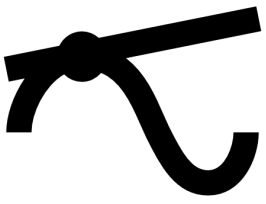
Research Position App

Design Document

10/25/2021

Version 1



Team TAN

Nolan Johnson,

Joseph Alonso,

Aidan Nunn,

Tanner Crane

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Introduction

The purpose of this document is to lay out a foundation for how we will code our project. We are figuring out what kinds of models, forms, and routes we will need before we start any coding.

The project aims to create a platform for teachers to post research opportunities for their students in a way that students can easily find and apply for. It also allows the teachers to easily notify the students if their application has been accepted for an interview.

Section II includes descriptions of the various design choices, as well as screenshots of the UI on the different pages of the webapp.

Section III includes a summary of the progress made on the webapp throughout the iterations.

Document Revision History

Rev 2.0 11/16/2021 Iteration 2

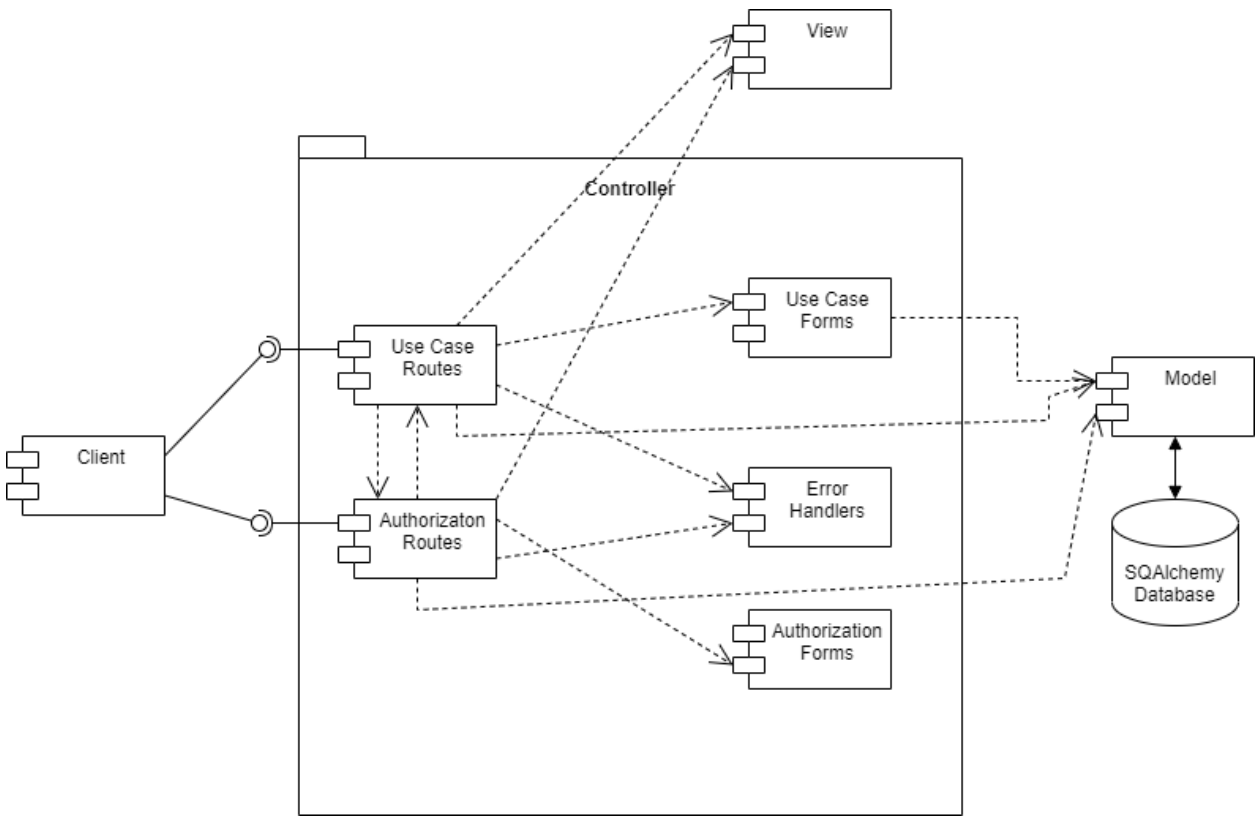
II. Architectural and Component-level Design

II.1. System Structure

This software will use the Model-View-Controller design pattern. We intend on using MVC design in order to decouple the business logic from the user interface, and because MVC is a great fit for the tools provided by the Flask framework. By using a Controller to decouple the View and the Model, we can update either of those two components without having to significantly update the other.

The Model will hold classes for organizing information in the SQLAlchemy database. The View will manage displaying the app to the user and getting user inputs. The Controller will connect the Model and View by providing routes and forms that will allow information to pass between the Model and View.

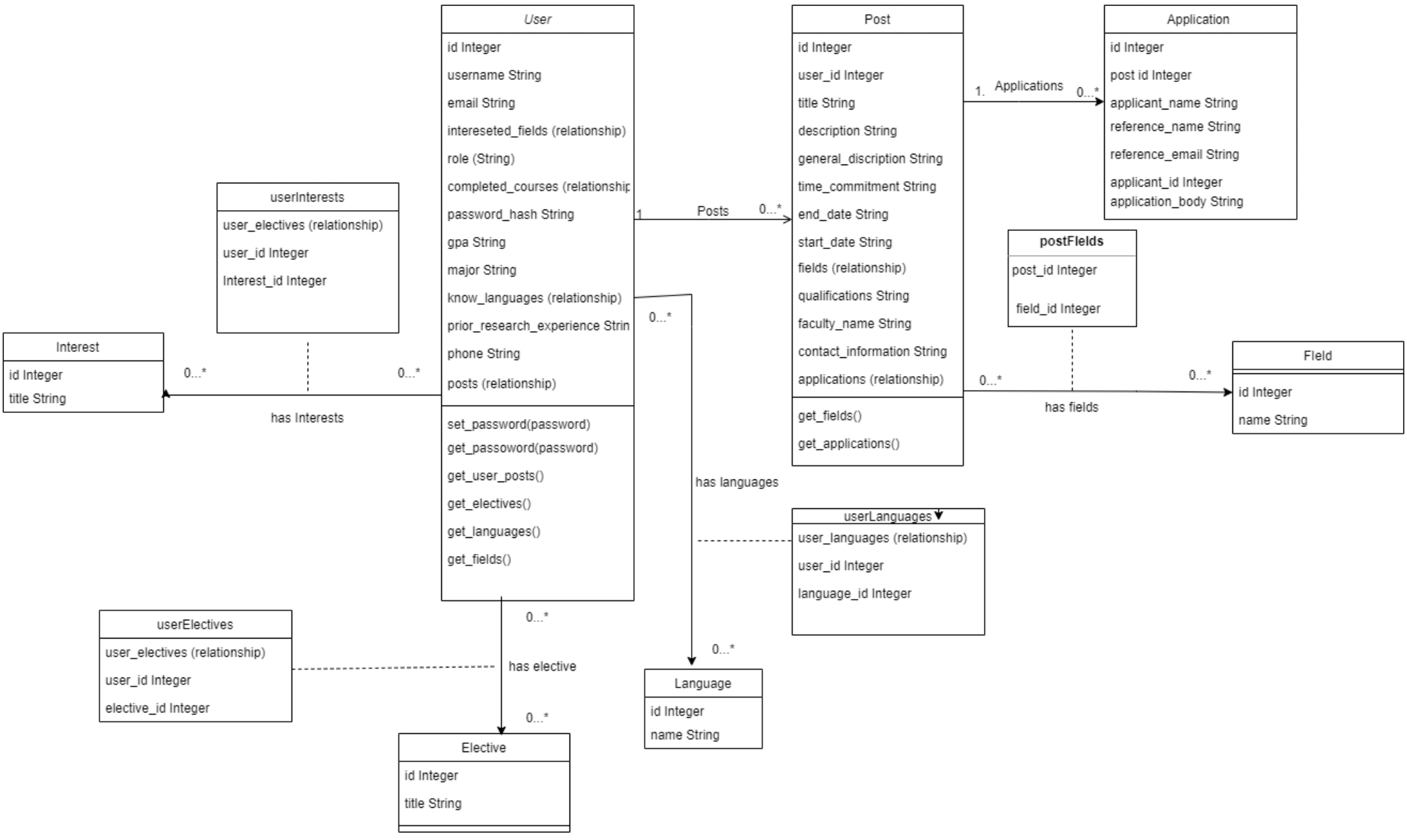
Figure 1 - UML Component Diagram



II.2. Subsystem Design

II.2.1. Model

Figure 2 - UML Class Diagram



The model will hold database objects for accounts and post / application data.

‘postFields’ Table: The ‘postFields’ table supports the relationship between research position posts and field tags.

‘userLanguages’ Table: The ‘userLanguages’ table supports the relationship between user accounts and their selection of known programming languages.

‘userElectives’ Table: The ‘userElectives’ table supports the relationship between user accounts and their selection of previously taken elective courses.

‘userInterests’ Table: The ‘userInterests’ table supports the relationship between user accounts and their selection of research fields they would like to be involved with.

User Account: The User model stores user information.

Attributes:

- id: ID of the account.
- username: Unique user identifier.
- email: User’s email address.
- WSU_id: user’s WSU ID number.
- password_hash: Hash of the user’s password.
- role: A tag that denotes what account type.

Student Specific:

- major: The user’s major.
- gpa: The user’s GPA score.
- completed_courses: The user’s list of completed courses.
- interested_fields: The user’s list of fields they are interested in.
- known_languages: The list of the user’s known programming languages.
- prior_research_experience: The user’s prior research experience.
- applications: Relationship between student and their applications

Faculty Specific:

- phone: The user’s phone number.
- posts: A relationship linking posts to a faculty account.

User Languages: The Language model stores a programming language known by a user.

Attributes:

- id: The ID of the language.
- name: The name of the language.

User Elective: The Elective model stores an elective course taken by a user.

Attributes:

- id: The ID of the elective.
- title: The title of the elective.

User Interest: The Interest model stores a research field that a user is interested in.

Attributes:

- id: The ID of the interest.
- title: The title of the interest.

Field: The Field model stores a research field that a research position is related to.

Attributes:

- id: The ID of the field.
- name: The name of the field.

Research Position Post: The Research Position Post model stores information regarding a research position.

Attributes:

- id: id of the post.
- user_id: id of the user who created the post.
- title: title of the post.
- general_description: a general description of the research position.
- time_commitment: time commitment for the research post.
- start_date: the start date for the research position.
- end_date: the end date for the research position.
- fields: a table of one or more research fields.
- qualifications: the qualifications a student must satisfy to apply.

Research Position Application: The Research Position Application model stores information regarding a research position application.

Attributes:

- id: id of the application
- student_id: relation containing id of the student who made the application
- name: name of the student who applied
- reference: email of the student who applied
- interest: text block explaining why the student wants to apply for the position
- status: status of the application (i.e. pending, cleared for interview, rejected, hired)
- post_id: relation connecting the application to a post

II.2.2. Controller

The controller is divided amongst several subsystems: authentication forms, authentication routes, forms, routes, and errors.

Authentication Forms: Handles forms which deal with authentication of a user, i.e. creating an account or logging in to an existing account.

- Register form: Creates an account for either a Student or Faculty member.
- Login form: Log in to an existing account.

Authentication Routes: Authentication routes direct the user to parts of the app which deal with logging in and out, and registering for an account.

- Register: Directs users to the registration page where they can create their account.
- Login: Directs user to login page where they can enter their username and password to login
- Logout: Directs the user to the logout page where they can log out of the system.

Forms: General forms

- Position sort form: Sorts the positions displayed according to certain criteria.
- Research post form: Form where the faculty member creates a research position to post to the webapp.
- Application form: Form for submitting an application to a research position.
- Edit profile: Allows the user to change information on their profile.

Routes: General routes to pages of the webapp

- Index page: Displays research positions and has buttons that lead to the registration, the login page, the logout page, and the sort form.
- Post research position: A page where faculty enters information about their research position to post.
- Position application: Page where a user can fill out an application for a research position
- View application: Page where faculty can view applications to a research position and accept / decline them
- Profile page: Displays student and faculty profile.
- Edit profile page: Allows students to change information they entered when creating an account.

Errors: Two error handlers for the webapp

- 404 error: Page not found.
- 500 error: Database conflict.

Table 1- Route Details

Route	Methods	URL Path	Description
Register Student	Get, Post	'/register_student'	Renders the template for the student registration form.
Register Faculty	Get, Post	'/register_faculty'	Renders the template for the faculty registration form.
Login	Get, Post	'/login'	Log the user into their account.
Logout	Get	'/logout	Log the user out of their account .
Index	Get, Post	'/index' or '/'	Webapp home page, also where research positions are posted.
Post Research Position	Get, Post	'/post_position'	Page to fill out information on research position and post it to the main page.
Position Application	Get, Post	'/post_application<post_id>'	Apply for a specific research position.
View Applications	Get, Post	'/view_applications<current_user_id>'	Faculty can view and accept / decline applications to their posted positions
Profile Page	Get	'/display_profile'	Displays profile information
Edit Profile	Get, Post	'/edit_profile'	User can edit the information of their profile
Delete Position	Delete, Post	'/delete<post_id>'	Allow Faculty to delete their own posts
Delete(withdraw) Application	Get, Post	'/remove_application/<application_id>	Allow student to withdraw their own application
Update Application Status	Get, Post	/update_status<application_id>	Updates the status of a student's application to "Pending", "cleared for interview", etc.

II.2.3. View and User Interface Design

The view will display the pages of the webapp and provide options for the user to interact with the app. User interfaces will be built using a combination of HTML, CSS, and Jinja2.

The view will display the position posts in a 3X3 grid, with the buttons leading to the registration, login, logout, and profile along the top of the page. The layout of the posts will change depending on what device the user is using. The background color of the main page will be white, with the posts being light blue rectangles with text inside. The background of the profile page will be baby blue, with the information blocks being white. The registration, login, and logout pages will be a light green, with the text boxes being white. The overall font will be Tahoma.

The libraries we will be using are:

- flask_sqlalchemy
- flask_wtf
- wtforms
- wtf_sqlalchemy
- flask_login
- werkzeug.security

Screenshots of the UI:

Figure 3 - Index Page

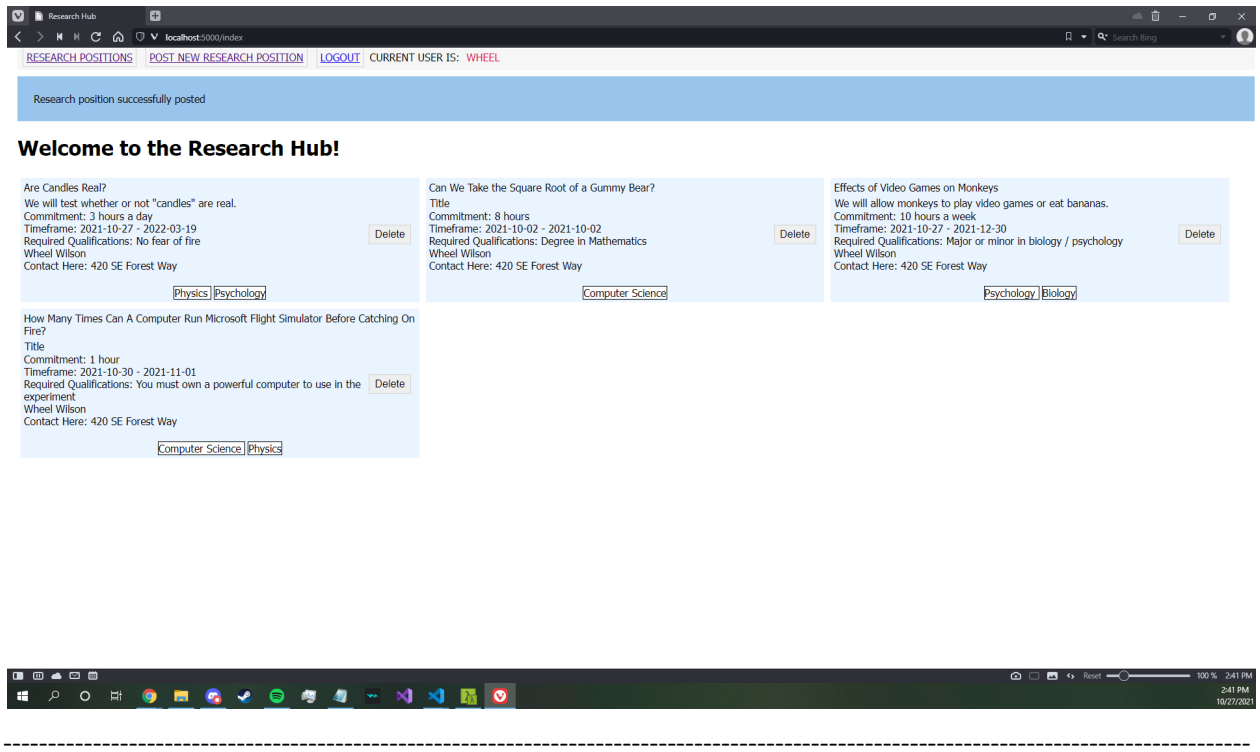


Figure 4 - Student Registration Page

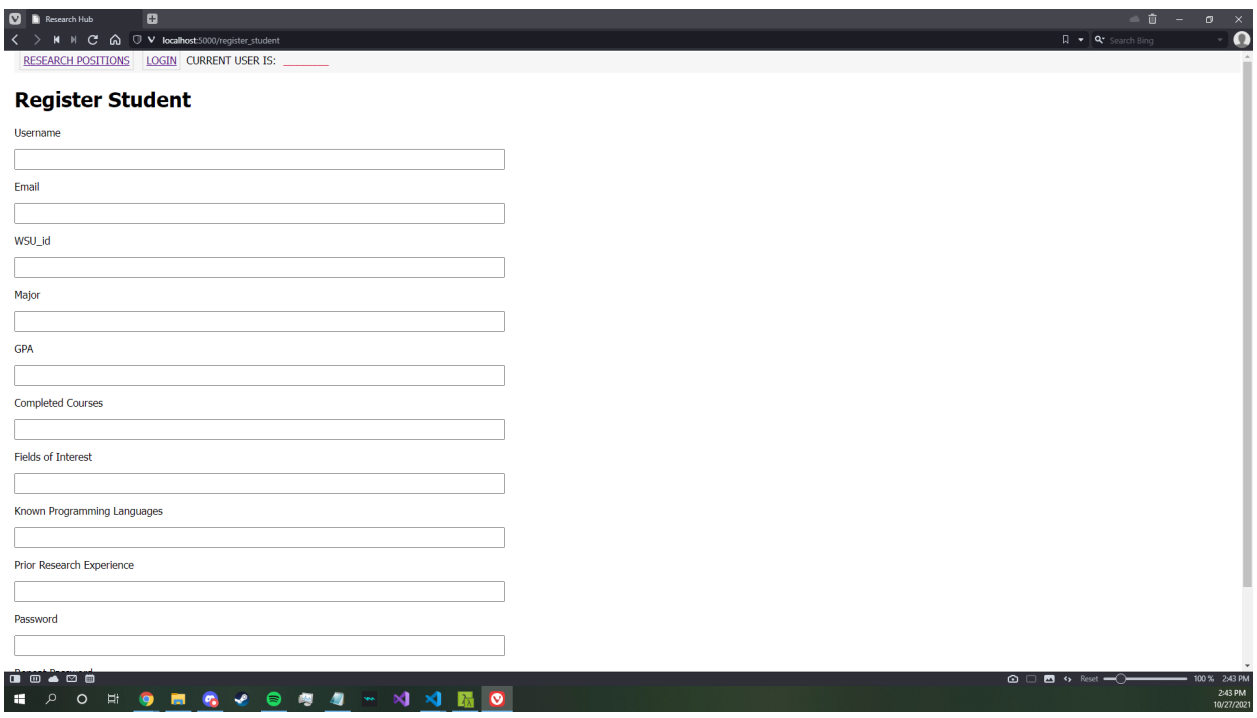


Figure 5 - Faculty Registration Page

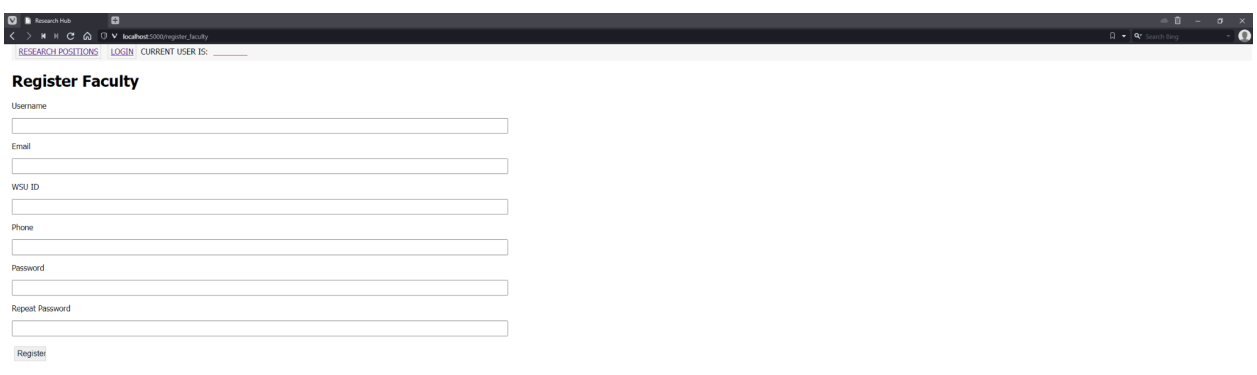


Figure 6 - Sign In Page

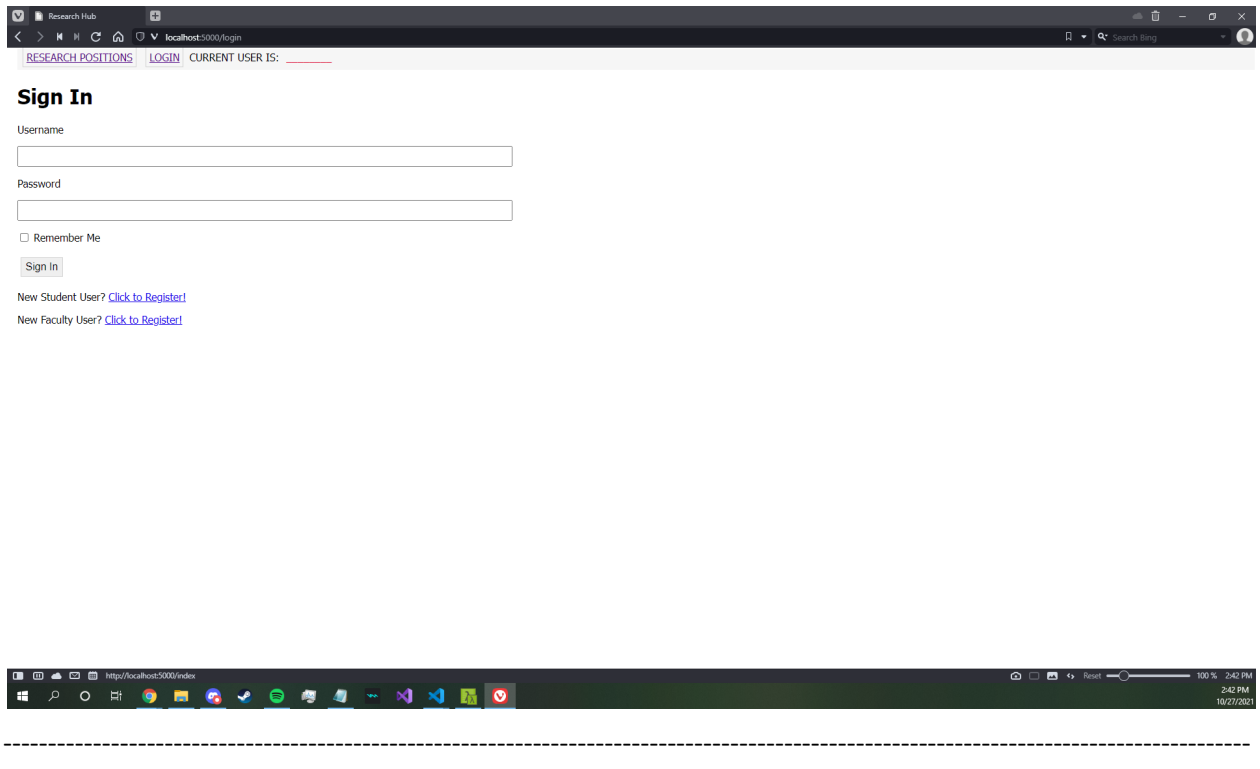
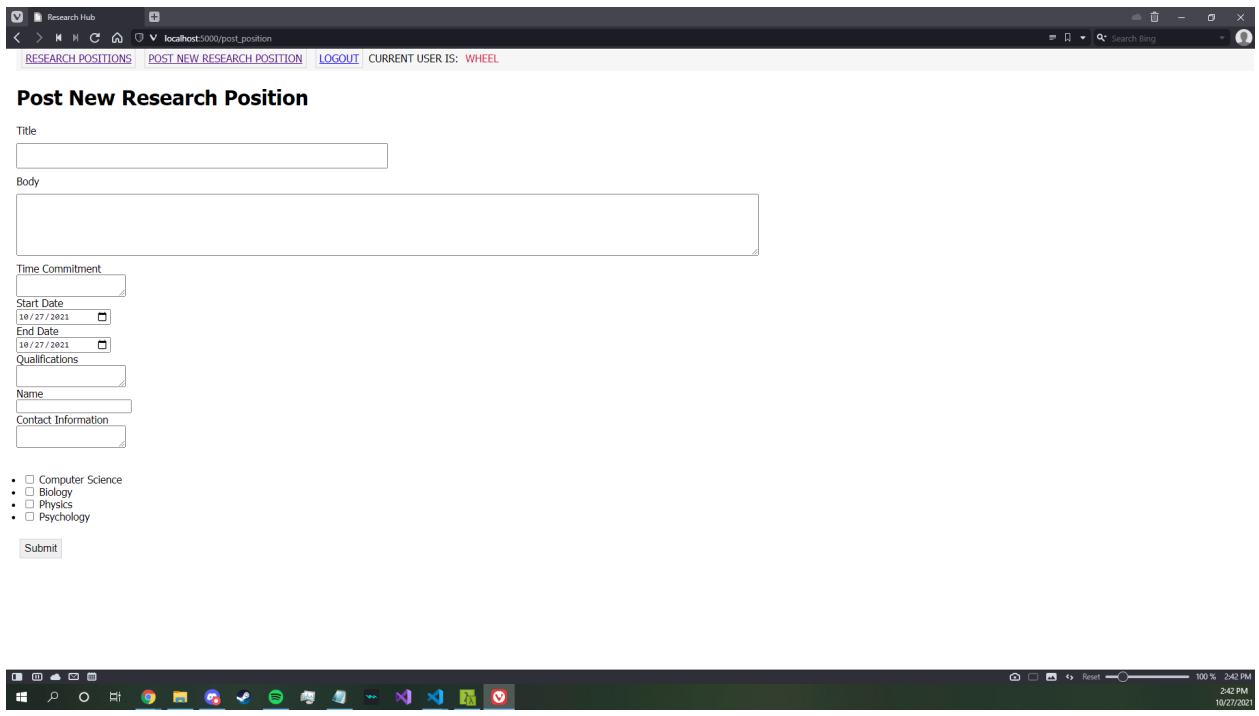


Figure 7 - Post New Research Position Page



III. Progress Report

Iteration 1 Progress:

In Iteration 1 we implemented a structured outline of the final product of our research project app. We added the user, post, and field models as well as a relationship table for the many to many relationship between fields and posts. We added the login page and implemented staff and student login, as well as staff and student registration. After logging in the user can access the index page that shows all active research positions, and as faculty you can post a new position as well as delete one of your posted positions.

Iteration 2 Progress:

In Iteration 2 we added the ability for students to apply for research positions and view their pending applications. Students are also able to withdraw pending applications. Faculty can view pending applications for positions they have posted. Both students and faculty can also view their profiles and edit fields in their profiles.

We also updated the database model so that users' completed electives, interested research positions, and known programming languages are represented as database relationships between the user class and classes for each of the other previously specified fields.

IV. Testing Plan

Unit testing:

We will write tests for each of the routes in ‘routes.py’ and ‘auth_routes.py’. We will use the unittest framework for writing automated tests. Each team member will write tests as they work on use cases so that we develop a backlog of tests for future regression testing.

Functional testing:

For our functional tests, we will look at our use cases that we decided to implement and then manually fill out the required information for each use case. We will be sure that each use case performs its task correctly and without fail, and we will use a variety of different input data to make sure of this. For example, to test the functionality of posting research positions, we will manually fill out a research position and attempt to post it.

UI Testing:

We will also be manually testing the UI by running the project and fixing any errors we come across when trying to use it. We will be manually entering data and clicking buttons to be sure that every functionality of the user interface works as intended.