

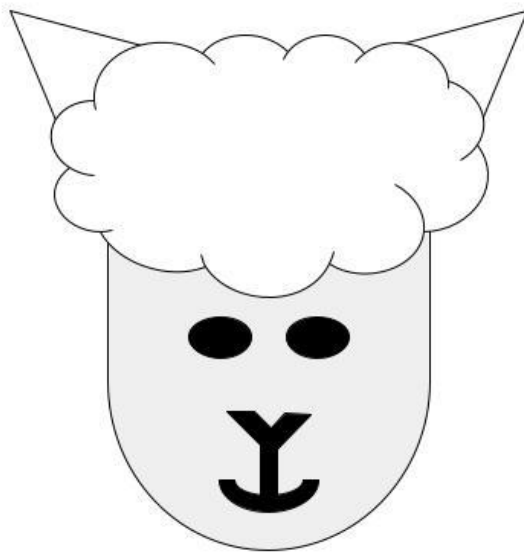
Research Position Search App

Design Document

11/16/2021

Iteration 2

TEAM LAMA



Albert Lipaev, Mitchell Kolb, Louis Kha, Alex Castillo

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

| | | |
|-------------|---|---------------|
| I. | INTRODUCTION | 3 |
| II. | ARCHITECTURAL AND COMPONENT-LEVEL DESIGN | 4 - 15 |
| II.1. | SYSTEM STRUCTURE | 4 - 5 |
| II.2. | SUBSYSTEM DESIGN | 5 - 15 |
| II.2.1. | <i>Model</i> | 5 - 6 |
| II.2.2. | <i>Controller</i> | 7 - 9 |
| II.2.3. | <i>View and the User Interface Design</i> | 9 - 14 |
| III. | PROGRESS REPORT | 14 |
| IV. | TESTING PLAN | 15 |
| V. | REFERENCES | 15 |

I. Introduction

The purpose of providing this design document is to show and explain the overall architecture of the Research Position Search App we are developing..

This is the second iteration of the design document, we have made some changes to the overall architecture of the website including. One of the changes that has been made is to the student and faculty account information database. In iteration 1 we had them in two different databases that held the account information. We were informed that it would be easier in the long run if we combined them and just added an “is current user faculty” check to make sure that the correct information is displayed. Another change to the website’s architecture is that now we have index pages for students and faculty that are relevant to each. Before we just had the site route to the base index page for both users. This allows us to now only make certain features show for the student like the apply button for research positions or the post new research position for the faculty users. Since iteration 1 we have made many quality of life changes that don't change the overall architecture of the website but affect how the site functions like simplifying the menu bar and adding the display profile feature and the ability to apply to research positions.

The Research Position Search App is a web application that allows college professors and undergraduate students to connect and communicate to each other on research projects. This application has two sides, a student section and a faculty section. The student section will allow for students to create an account that contains their profile information and lets them apply to faculty posted research positions. The faculty section will allow those users to create postings that students can apply to, and to get into contact with students who have applied to existing postings to potentially interview for the position. Our project goal is to create a web application that executes the idea that is described above.

Section I includes the introduction of the design document, a description of the changes that have been made since the last iteration, and a description of the project.

Section II includes the architectural and component-level design which describes the MVC pattern that our application is using.

Section III includes a progress report on how the application is coming along during the work that has been completed in iteration 2 and the problems our group has overcome during this time.

Document Revision History

Rev 1.1 <2021-10-27> <Iteration 1, basic skeletal framework>

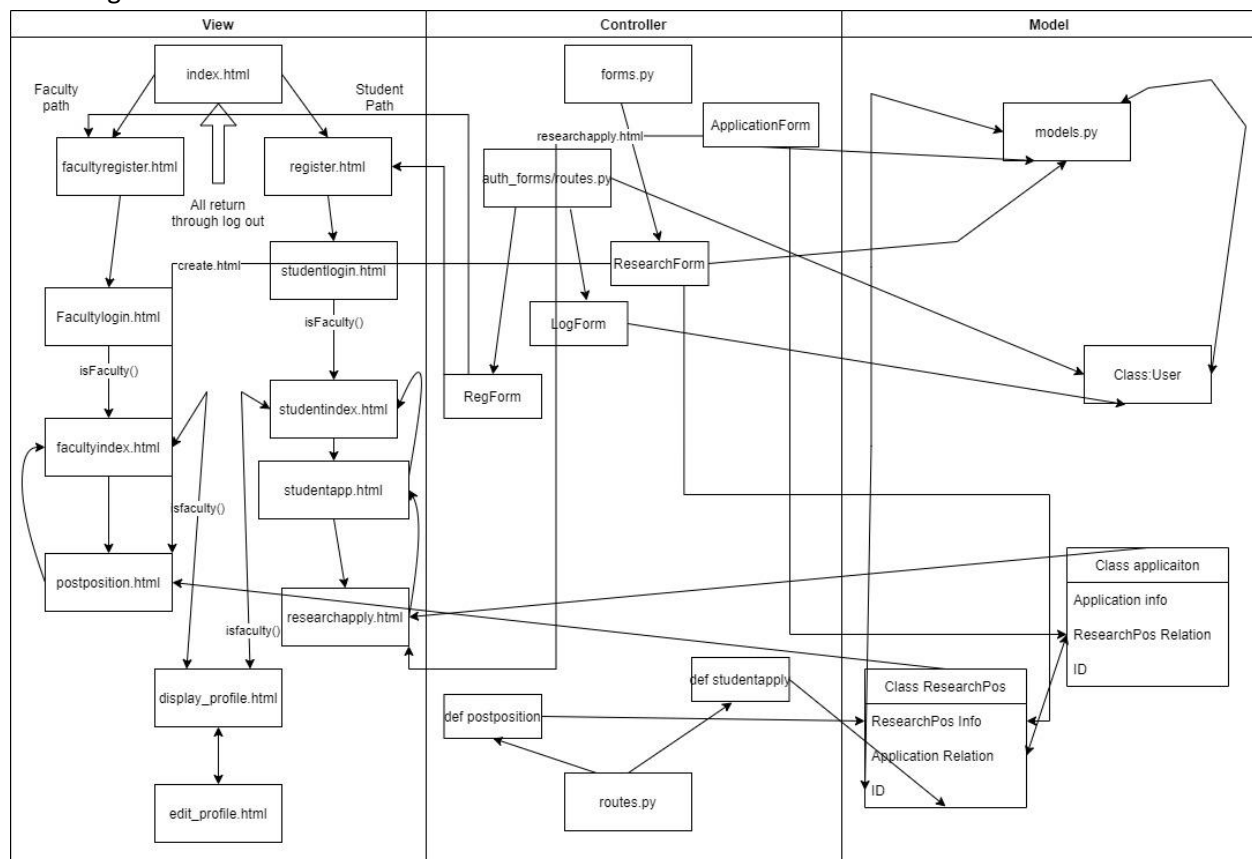
Rev 2.0 <2021-11-16> <Iteration 2, expanding the framework>

I. Architectural and Component-level Design

I.1. System Structure

While maintaining the Model-View-Controller format since before as our main architectural pattern, we have managed to improve upon our system's design, while allowing for even greater work to be performed between team members asynchronously as we now had a skeletal framework of the application to begin with. This cut down immensely on planning as well as worries over merge conflict issues as much of the work now revolved around different components and files, such as different html files or routes used while revolving around two or three key database models.

UML Diagram:



While several of the dependencies remain the same from the iteration before, there are key differences that are displayed in the UML diagram of drastic changes that were made to the framework of the application. Specifically, the View section of the has undergone a complete overhaul, now split between a faculty path and a student path which both organizes the work necessary to be done on each end through the lens of the user, while also marking clearly the dependencies used between each path. The Controller section remains largely the same with a new Form used that connects to the student path in which applications can be filled for research positions. The Models also introduce a new class known as the "application" Class which now also shares a many to one relation with the "ResearchPos" class. This

allows for the application class to keep the researchPos class' ID which will be used to link them in the faculty path to allow faculty to see what students applied to which positions in the future.

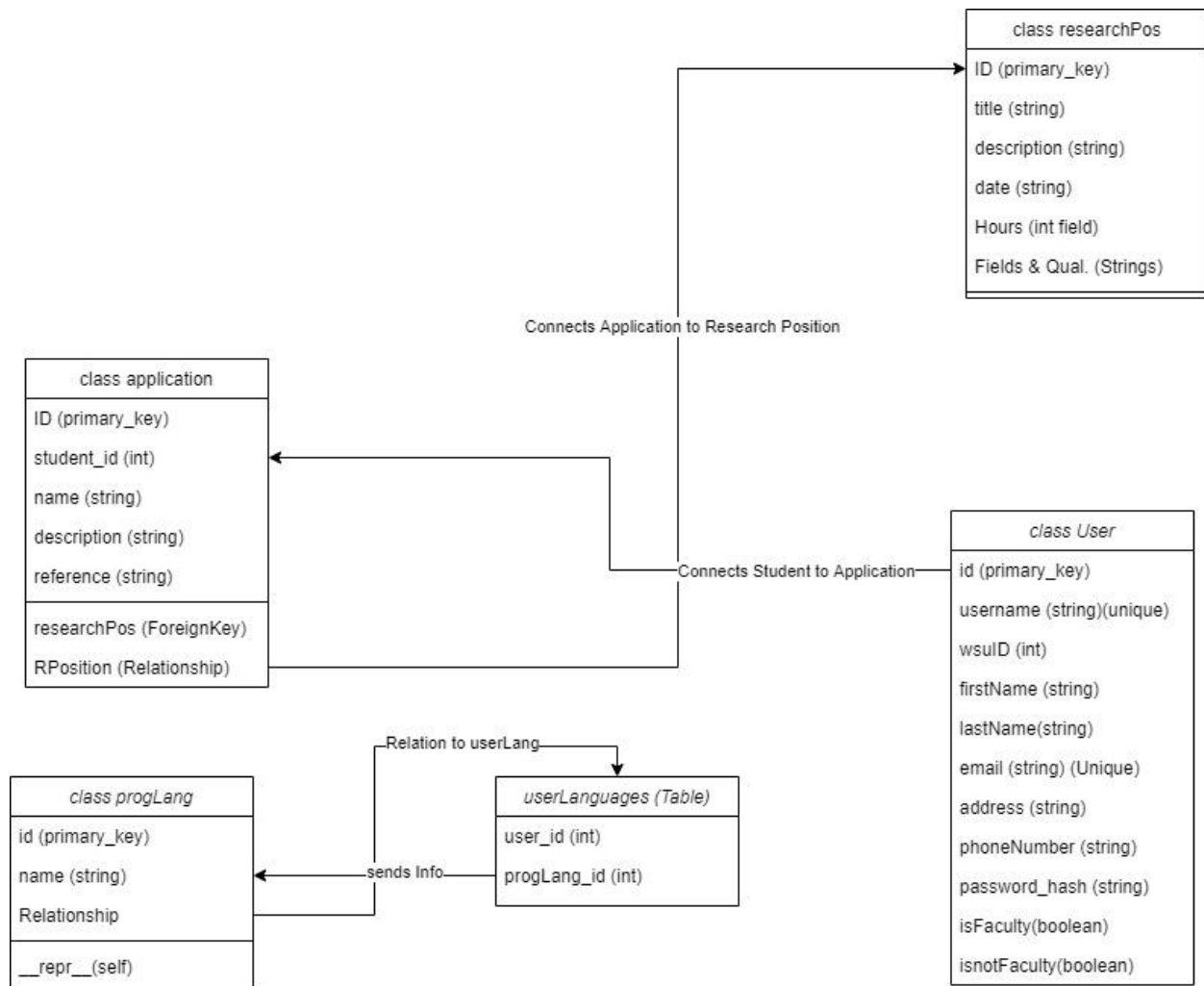
I.2. Subsystem Design

I.2.1. Model

The Model acts as the core of the program. We keep all of our data models and algorithms in this portion of the program. The model doesn't depend on the controller or the view.

When progressing from iteration1 we initially believed that splitting the User database model between student and faculty would have been beneficial to our progress we reversed our decision and are creating it as an abstract model of inheritance to split between faculty and student paths in our application. While most the model's functionality has remained the same from the last iteration, we have new introduction of classes including the application and progLang class, which respectively allow students to apply to research positions, and keep track of the ID of the research positions they have applied to and create a table relationship with the possible programming languages students can choose when filling out their profile.

UML Diagram:



1.2.2. Controller

Subsystems:

Authentication subsystems:

1. Auth_Forms:

- RegistrationForm interacts with the register/facultyregister method from auth_routes providing an instance of the data input class for registration.
- LoginForm interacts with the login/faculty & logout method from auth_routes. Provides an instance of data for login.
- Interdependencies: Auth_Routes

2. Auth_Routes:

- login/facultylogin method checks whether the input data matches what is stored in the database. If the data is not found, it will print an error on the screen. If the user instance is not matching by the 'isfaculty' type, it is denied login. Otherwise redirects to the main page (studentindex.html or facultyindex.html, depending on the user type). Logout method logs out the current user and goes back to the login screen (student login).

- register/facultyregister allows for different types of users to register by storing their information on the database.
- logout allows the current user to leave the current logged in session
- Interdependencies: Auth_Forms, Errors

Errors subsystem:

3. Errors:

- Used to display either error 404 (data not found on the server), or to display error 500 in case the server experienced an unexpected internal error.
- Interdependencies: Auth_Routes, Routes

Post handling subsystems:

4. Forms:

- ResearchForm provides a model for the data for the research post made by faculty
- EmptyForm provides a workaround for submitting an empty form without crashing the application
- ApplicationForm provides a model for data for student submitted application to the research position
- EditForm provides a model for data which is stored during the registration and which is able to be changed when it needs to be edited
- Interdependencies: Routes

5. Routes:

- Index route redirects user to the starting page, either it is a login page or studentindex or facultyindex depending on the user status
- Createpost redirects faculty members to the post creation page.
- Postposition allows faculty members to create and store their positions in the database and be displayed both on studentindex and facultyindex
- Studentindex route redirects to the studentindex page for logged in students
- Facultyindex route redirects logged in faculty members to the facultyindex
- Studentapply2 route allows students to view their applications for the research positions they have already applied to
- Studentapply route redirects students to the studentapply page where students can enter their information for the research position
- Displayprofile route redirects users to the display profile page which displays their info stored on the database
- Displayselected route allows for faculty to see students who applied to their research positions
- Editprofile route redirects users to the edit profile page where users can change their information they have provided previously or during a registration
- Researchapply route allows students to submit their application to the research position
- Interdependencies: Forms

Models:

- User model provides a model for all possible data stored about the user

- Researchpos model provides a model for data stored about the research position created by the faculty member
- Application model provides a model for data stored about the application which student can submit for the research position
- Proglang model stores possible programming languages student can choose to edit in the display information page
- Interdependencies: Forms, Routes, Auth_Forms, Auth_Routes

Page table with methods

| Methods | URL Path | Description |
|------------------|---------------------------------------|---|
| Login | http://127.0.0.1:5000/login | Student login is the default starting page set for the user who has not yet logged in or has not registered yet. Has a button which leads to the registration page or faculty login page. |
| Faculty Login | http://127.0.0.1:5000/FacultyLogin | When the user clicks the faculty login button, it redirects to the faculty login page. Allows login for faculty. Prevents students from logging in as a faculty member. Has a button which leads to the registration page for new faculty members and a button to switch back to student login. |
| Register | http://127.0.0.1:5000/register | After clicking a register button on a student login page, it redirects to the register page for students. There students can enter their information and register in case they have not registered yet. Register button after successful registration redirects students to the studentindex, which is the main page for students. |
| Faculty Register | http://127.0.0.1:5000/Facultyregister | After clicking a register button on a faculty login page, it redirects faculty members to the registration page, where new faculty members can enter their information and register. After successful registration, new faculty members are redirected to the facultyindex page, which is the main page for faculty. |
| studentindex | http://127.0.0.1:5000/studentindex | Main page for the students. Opens after the student is successfully logged in or registered. Also redirects students who are already logged in to this page. There students can view their profile & logout. They can also see the positions which are made by the faculty. If the student wishes to apply for the position, the 'apply' button appears under the post only visible for the student. Students cannot create new research positions. |

| | | |
|-------------------|---|---|
| | | Students can click the main page button to be redirected back to their home page. |
| facultyindex | http://127.0.0.1:5000/facultyindex | Main page for the faculty members. Opens after the faculty member is successfully logged in or registered. Also redirects faculty members who have already been logged in to this page. Faculty members can view their profiles, logout and create new positions for students to apply to. Only faculty members can create new positions. Faculty members cannot apply for the created research positions. Faculty members can click on the main page button to be redirected to their home page. |
| post position | http://127.0.0.1:5000/postposition | Appears only for faculty members. Redirects faculty members to the postposition page. Faculty can create their research position posts and post them to be visible for other faculty members and students. Faculty cannot apply for the research positions. |
| main page | http://127.0.0.1:5000/studentindex | Appears for both faculty and students. When pressed, it redirects students to the studentindex page. |
| main page | http://127.0.0.1:5000/facultyindex | Appears for both faculty and students. When pressed, it redirects faculty to the facultyindex page. |
| logout | http://127.0.0.1:5000/logout | Button present for both faculty members and students. Allows users to log out of their accounts. Redirects back to the student login page. |
| display profile | http://127.0.0.1:5000/display_profile | Button present for both faculty members and user students. Redirects to the display profile page which shows information about the user. Also has a button for editing the profile. |
| Edit your profile | http://127.0.0.1:5000/edit_profile | Present for both students and faculty members. Allows to edit the profile information. Has a button to confirm the change of the profile information. Redirects back to either the studentindex or faculty index. |
| apply | http://127.0.0.1:5000/studentapply2/2? | Present only for the students. Appears under the research positions posts created by the faculty on the studentindex. When the button is pressed, the student is redirected to the researchapply page. |
| researchapply | http://127.0.0.1:5000/researchApply/2? | Present only for students. Allows students to enter their information when applying to the research position. Has a post button which allows students to submit their application with their |

| | | |
|--|--|---|
| | | information for the position. Redirects back to the studentindex. |
|--|--|---|

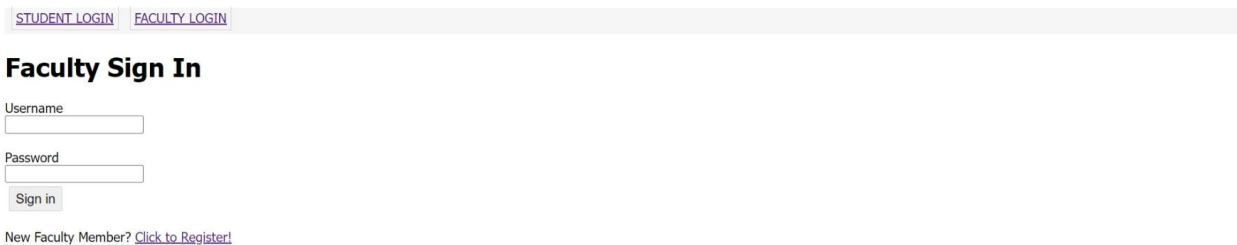
I.2.3. View and User Interface Design

View stores the html pages for the application. It creates the front end for the application, which is presented as a user interface (UI) and is a visual representation of the working code.

Showing the changes and additions from the last iteration from the beginning:

Login is now split between faculty and students

USE CASE: 1



STUDENT LOGIN FACULTY LOGIN

Faculty Sign In

Username

Password

New Faculty Member? [Click to Register!](#)

Students are now able to view and apply to applications

USE CASE:3,6

Student Main View

Welcome to Faculty SearchApp Portal!

Number of Available Positions:

| Test title |
|---|
| test postion 20 Hours a Week test fields Posted at: November 16, 2021 3:25 PM The post is created by: LOUIS <input type="button" value="Apply"/> |

Student's view of the application and their ability to apply
USE CASE: 7

Student Applying View

You are Currently Looking at A Research Position!

Information Below:

Title: **Test title**
Research Description: **test postion**
Required Hours: **20**
Research Fields: **test fields**
Posted at: **November 16, 2021 3:25 PM**
The Position is created by: **LOUIS**
Testing Student ID View: **Louis**

Student form to apply to research position
USE CASE: 8

Design Document 10

Apply To Research Position Page

Student Name

Brief Description

Faculty References

Post

Student and Faculty capability to view profile, changes information displayed depending on user isFaculty() boolean

USE CASE: 3

Display Student Profile

Username: louis - 1

WSU ID: 11444138

Name: Louis Kha

Phone #: 1234567891

Email: l@wsu.edu

Address: asdasdasdasdasd

Technical Courses: WIP

Cumulative GPA: WIP

TC GPA: WIP

Research Fields: WIP

Programming Languages:

Experience Desc: WIP

Faculty: False

[Edit your profile](#)

Ability to edit and add profile information

USE CASE: 5

Edit Profile

Username
louis

Enter your WSU ID
11444138

First Name
Louis

Last Name
Khia

email
l@wsu.edu

Address
asdasdasdasdasd

Phone Number
1234567891

Password

Password Repeated

Programming Languages

- ☐ C ,
- ☐ C# ,
- ☐ C++ ,
- ☐ Java ,
- ☐ JavaScript ,
- ☐ R ,
- ☐ Swift ,
- ☐ Python ,
- ☐ PHP ,
- ☐ Swift ,
- ☐ Dart ,
- ☐ Kotlin ,
- ☐ MATLAB ,
- ☐ Perl ,
- ☐ Ruby ,
- ☐ Rust ,
- ☐ Scala ,

Submit

Lastly, it should be noted that base.html functionality has been changed only to display the appropriate routes to the appropriate users. No longer can students “post positions” for example.

```
{% extends "base.html" %}
```

II. Progress Report

Our group has made a good amount of progress since iteration 1. We have added some major features that we set up last time and have been smoothing out the rough edges that we find along the way. Some of the things that we have added to our project include the ability to now display the profile of the logged in user. Along with that we have made separate index pages for students and faculty so that they can show the correct information. We have also fixed many errors and made some quality of life changes like simplifying the menu bar at the top of the website. In the time frame between iteration 1 and 2 we have met many times in person and online to help each other out and fix the problems that we all run into. We had less trouble this time around merging all of our code together. I think this is because we had a good framework made to work off of and we could now just make our additions to the code without massively affecting each other. For the future we plan to continue the system that we have going now and we plan to have a finished and fully functional project by the time iteration 3 comes around.

III. Testing Plan

When planning to test our system, we will test our Models, API routes, and each of their respective functions. For automatic testing we will import pytest and unittest. When testing the models we will import unittest and create a ModelTest class that will contain member functions that specifically target important parts of the User Model.

With the Models, will test to make sure the password hashing is correct by creating a user and set their password. We will then use assertTrue() with the actual password and assertFalse() with the incorrect password in order to test if the Boolean of the expression is true with assertTrue() and false with assertFalse(). We decided this is a better way to test instead of using assertEquals() because it expresses a more helpful error message. When testing the research positions post with faculty, we will create a faculty user and a position posted by them with all of it's relevant information e.g. Title, Body, Language, Time commitment etc. and test if it is equivalent to the input we provided. By creating a similar function we can test if the correct information is being stored in the database for each user as well.

We will be using pytest to automatically test our route functions. We will test to make sure all of the get/post routes work correctly by creating their own test functions that check if they get the correct response by testing the get functionality of the routes, using assert to check for the correct response data. We also plan to test if the user can post, apply, delete, etc. by using assert to check if we received a successful or failed assertion.

For Functional Testing we will manually test certain things that users won't be able to do, for example a student shouldn't be able to view a faculty specific page. We can manually test this by typing the faculty specific route endpoint in the address bar as a student and check if we can visit the page or not. For UI testing we will manually test it by referring to our Use case diagrams and check each case that wasn't included in the automatic testing.

IV. References

"SQLAlchemy 1.4 Documentation." *Basic Relationship Patterns - SQLAlchemy 1.4 Documentation*, https://docs.sqlalchemy.org/en/14/orm/basic_relationships.html.

Quanit, M. (2021). Software Engineering Principle — Coupling & Cohesion. Retrieved 28 October 2021, from <https://dev.to/mquanit/software-engineering-principle-coupling-cohesion-1mma>