CptS 322 Term Project

Design Document

10/27/2021

Version 1

Error404

Denise Tanumihardja

Tay Jing Ren

Reagan Kelley

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Introduction

This design document provides an overview of our approach and design for this project. This overview is a step-by-step process of what we have done and how we did it. We are building an application for undergraduate students who are interested in research to get more involved. With this, our goal is simply to reduce redundancy and complexity of applying for a position; this makes the application simple and accessible anywhere without needing to go in person in the case of the applicant, and makes it easier for the faculty to decide based on an organized list of applicants, who to hire.

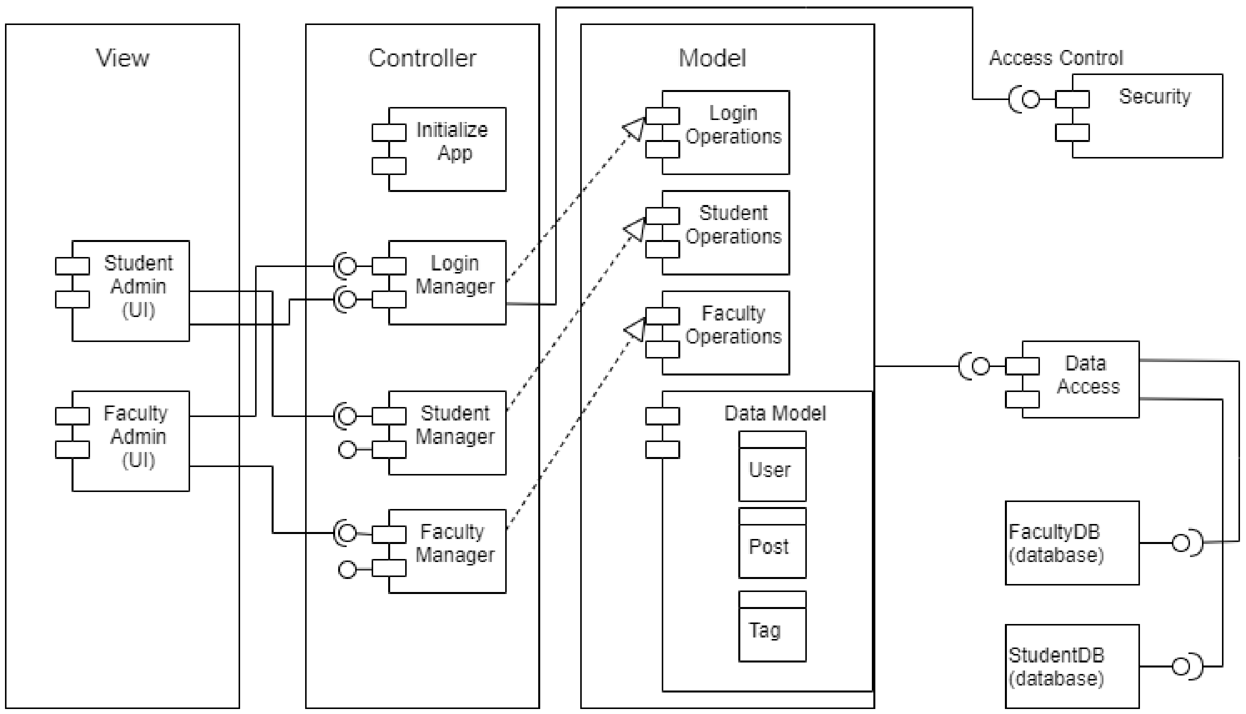
Section II includes the System Structure, which describes the structure of the system design, in other words, the system and subsystems and how they interact with each other. The Subsystem Design, which includes details of the software design pattern (MVC model), was the most salient agenda for Iteration-1.. Moreover, section III includes the current progress report for the project, which describes what functionalities and systems we have implemented so far.

Document Revision History

Rev 1.0 – 10/27/2021 – Initial version.

II. Architectural and Component-level Design

II.1. System Structure



1. View:
   - Student Admin – Displays the user interface for student users. It depends on the login manager to determine if the user account is a student one, and student manager to display all the necessary information for students.
   - Faculty Admin – Displays the user interface for faculty users. It depends on the login manager to determine if the user account is a faculty one, and faculty manager to display all the necessary information for faculty.

2. Controller:
   - Initialize App – Initializes the application.
   - Login manager – Manages the login function for users either Student or Faculty. It depends on the Security, which determines if a user has logged in or not. Additionally, it depends on Login Operations, to lock access for non-users to see posts.
   - Student Manager – Manages the information that the student user can see in the View, and accordingly changes the view when user inputs a command such as clicking on a link. It depends on the Student Operations to correctly collect information relating to the students and that user account from the database.
   - Faculty Manager – Manages the information that the faculty user can see in the View, and accordingly changes the view when the user inputs a command such as clicking on a link. It depends on the Faculty Operations to correctly collect information relating to the faculty and that user account from the database.
3. Model:
   - Login Operations – Determines if the login credentials the user inputted match to any in either the FacultyDB (database) or the StudentDB (database). Hence, it depends on Data Access to compare with both databases.
   - Student Operations – Determines which information is to be viewed for student users. It depends on Data Access to select what information is available in the StudentDB (database).
   - Faculty Operations – Determines which information is to be viewed for faculty users. It depends on Data Access to select what information is available in the FacultyDB (database).
   - Data Model – A template that takes user data input to be added into the database, either Student or Faculty.
4. Others:
   - Security – Locks access for non-users to see posts.
   - Data Access – It differentiates what type of data is given to Model, both information only Students and/or specific student users can see, and information only Faculty and/or specific faculty users can see.
   - FacultyDB – Stores data regarding faculty user information and data only faculty users can see.
   - StudentDB – Stores data regarding student user information and data only student users can see.

II.2. Subsystem Design

II.2.1. Model

The model manages the backend processing, specifically creating databases for the user, the post, and the tags used. In other words, becomes a communicator between the View and Controller model. Additionally, checks if the user logging in is a returning user.

1. User – Creates a database for user information such as their id (int model), username (string model), email (string model), password (string model), and user type which is either Student or Faculty (int model). Additionally, if user is faculty, it also includes the user's posts (relationship to the Post model).
2. Post – Creates a database for user's posts (if they are faculty). It includes the post id (int model), the user id of the poster (foreign key to the id of the User model), the post title (string model), a description of the post (string model), when it was posted (date time model) and tags, if any (relationship to the Tag model).
3. Tag – Creates a database for the tags used in a post. Includes the tag name (string) and its id (int).

II.2.2. Controller

The model takes in input from the user and accordingly interacts with the Model model. For this project, the Controller has been divided into smaller subsystems:

- Routes – It redirects the user to a specific page when they click on a link. It only redirects the user to the index page.
- Authentication routes – It redirects the user to a specific page when they click on a link, specifically for user authentication such as logging in, logging out, and user registration. As such, it redirects them to the login page if user has not logged in yet; it redirects them to the register page if user does not have an account to login to; and it redirects them to the index page and logs the user out when the user logs out.
- Authentication forms – It collects the authentication information inputted by the user and through the Authentication Routes subsystem, either that information is used to verify the user's account (in the case of login), or a database of the user is created with that information (in the case of registration), or redirects the user to another page (in the case of logout).
- Errors – User is redirected to a page from another, that displays an error message when an error has occurred.
- Forms – Forms is one of the important source files that is coupled with routes. However, for this Iteration, forms was not necessary to create the basic application structure and thus was not implemented.

Note: Some of your subsystems will interact with the Web clients (browsers). Make sure to include a detailed description of the Web API interface (i.e. the set of routes) your application will implement. For each route specify its "methods", "URL path", and "a description of the operation it implements".

You can use the following table template to list your route specifications.

(*in iteration-1*) Brainstorm with your team members and identify all routes you need to implement for the completed application and explain each route briefly. If you included most of the major routes but you missed only a few, it maybe still acceptable.

Table 1- Use an appropriate title for the table.

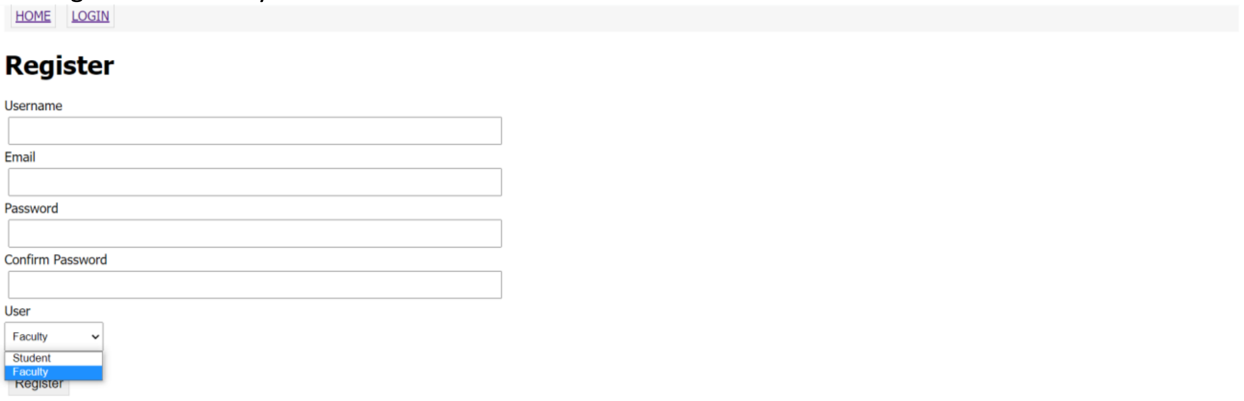
Methods	URL Path	Description
Get, Post	index.html	Main page for redirects
Get, Post	register.html	New users can sign up
Get, Post	login.html	Registered users can sign in
Get	routes.index	Main route origin

II.2.3. View and User Interface Design

The View manages the interface where the information is displayed, via templates, that communicate between the Model and the user via updating the view the user sees depending on what they inputted. In other words, the user interfaces. Moreover, it also manages the styling of the webpages, and if an error occurs, the according error page is displayed.

The current iteration of the project has the basic user interfaces for the login page, register page, a basic index page, and the navigation bar, along with basic stylings of each. With this, we have completed the following use-cases:

- Login
- Register for Students
- Register for Faculty



HOME LOGIN

Register

Username

Email

Password

Confirm Password

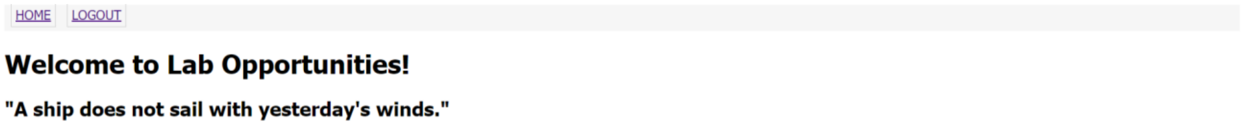
User

Faculty

Student

Faculty

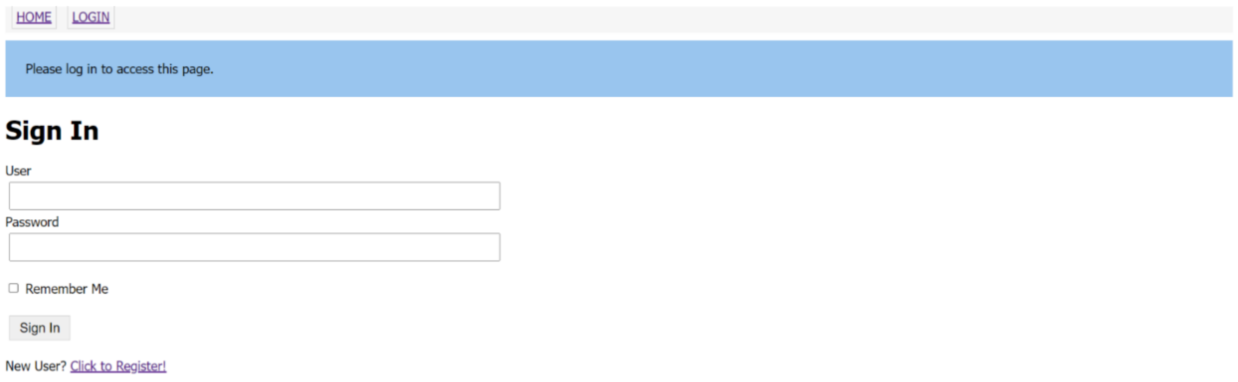
Register



HOME LOGOUT

Welcome to Lab Opportunities!

"A ship does not sail with yesterday's winds."



Please log in to access this page.

Sign In

User

Password

Remember Me

New User? Click to Register!

III. Progress Report

We have implemented the skeleton code, including the basic class infrastructure and imports. Currently the project has the general design implementations and infrastructure, in other words we have completed the code which would be used as a base for further implementation of more complex and/or specific functionalities.

IV. References

Sakire Arslan Ay – Smile App Project and Student App Project.