URC - Undergraduate Research Coordinator

Team name: URCC - Undergraduate Research Coordinator Coordinators

Team members: Calvin Tu, Khoi Tran, Ping Cheng Chung

Abstract

We have extended the URC project to make it usable for real users. User registration now takes in additional data that can be displayed in a profile. Profiles exist and have chats to allow users to find and message each other directly. A CRUD has been implemented for the various tags across the app, such as skills/courses/interests. Previously, these many-to-many relations could only be created through seeding. For developer convenience, database seeding has been reworked to make it more declarative. Previously seed data IDs were hardcoded, causing issues with creating new opportunities or applications. For admins, an analytics dashboard has been implemented to track statistics across the URC which they can use to inform future decisions about the URC.

Site URLs

Calvin: http://ec2-75-101-242-168.compute-1.amazonaws.com/

Khoi: http://ec2-52-200-31-207.compute-1.amazonaws.com/

Ping: http://ec2-34-204-51-81.compute-1.amazonaws.com/

Introduction

We have extended the URC project to make it usable for real users. In the past two weeks, we have implemented the profile management system, a real time chat with another user, and central dashboard.

Profiles have been implemented to serve as a homepage for users, providing them links to other areas of the web application they are likely to use the most based on their role. Profiles and opportunity now support many-to-many tags such as courses/skills/interests/search tags which are displayed correspondingly. Profile data is collected during registration and can be updated later through the manage account interface. Applications are now to specific opportunities instead of giving each student a global application; the UI is updated accordingly such that professors see their specific applicants and students see their applications to their chosen opportunities.

A real time web chat has been implemented to serve as a conversation between 2 users, providing a way to interact/communicate with other users. I made use of the SignalR library to allow students to use this chat feature as an option to ask professors or representatives any questions which they need for clarification. The chat box is only visible on profiles page of the users, and only 2 people can see it - viewer and that users. The messages are also encrypted to make sure that they are secured while chatting between 2 users.

A central dashboard has been implemented where an admin can monitor various analytics such as web traffic, skill distributions, etc. Collecting users visiting records, and analyzing current research opportunities e.g. required skill distribution, students' interest tendency, most searched keyword, frequency distribution of time spent on different pages, counting and recording link clicks, account activity. Display these information on the dashboard through a delicate chart.

Feature Table

Feature	Stack	Primary Programmer	Time Spent (hrs)	File	Lines of Code
Separate seed data from DB seeder	Backend/DB	Calvin	5	Data/SeedData.cs Data/DbInitializer.cs	400
Application per Opportunity	Backend/UI	Calvin	3	OpportunityController StudentsController +associated views/models	120
Profiles	Full stack	Calvin	12	ProfileController ApplicationUser Pages/Account/Manage/*	390
CRUD for opportunity/student tags	Full stack	Calvin	5	Pages/Account/Manage/* OpportunitesController Views/Opportunities/Edit	400
Integrate branches/ Integrate Chat with Profile	Full stack	Calvin	3	ProfileController chat.js	180
Chat UI	Front-end	Kevin	2	Views/Profile/Index.cshtml css/chat.css	150
Chat Controller	Backend	Kevin	1	Controller/ProfileController.cs	30
Logic of chat API	Backend	Kevin	12	Hubs/ChatHub	200
Server handle chat API	Backend	Kevin	12	js/chat.js	200
Store message on	Backend	Kevin	0.5	Data/URC_Context	25

database					
Creating new table and models for chat	Backend	Kevin	0.5	Models/ChatMessage.cs Models/ChatRoom.cs	40
Dashboard	Front	Ping	12	Dashboard/Index.cshtml Dashboard.js DashboardController	900
SEO-meta	Front	Ping	1.5	Opportunity/Index.cshtml	20
SEO-robot.txt	Front-end	Ping	1.5	HomeController	10
Session-count-visitor	End	Ping	2	HomeConrtoller OpportunityConrtoller	40
Multi languages	Front-end	Ping	6	DashboardController.cs Dashboard.resx dashboard.zh.resx	100
Opportunity-click counter	end	Ping	2	OpportunityController.cs OpportunityCounter.cs	100

Individual Contribution Summary

Team Member	**Time Spent**	**Lines of Code**
Calvin	28	1490
Khoi	28	910
Ping	25	1170

Individual Contribution Summary

Follow the above table with a paragraph for each team member where you briefly summarize

their main contributions.
Calvin: I reworked database seeding so that new opportunities and applications could be created by URC users (previously only seed data could exist) and so that it is easier for developers to add seed data. I implemented user profiles that can store data gathered during registration and be updated later through a UI. I updated the database to support many-to-many tags (for profiles and opportunities) with the accompanying UI to update them. Lastly, I made applications apply to individual opportunities with corresponding UI changes so that students can see all their applications and so professors can only see applications that belong to their opportunities.

Khoi: Creating a small chat API for the web application, including front-end and back-end. Creating tables and storing the messages in the database as well as creating a socket to connect between 2 clients. To chat with a professor, a student can go to the page of the professor and there is a small box at the right bottom corner of the page where he/she can put the messages and send to the professor. Whenever going back to the profile page of the professor, the message history will be loaded and see what they had been talking about.

Ping: I created a dashboard which can generate decent charts to analyze students and opportunities.It can see the required skills of opportunity distribution, also can see how many students or professor members to help the admin know what the weakness is. Add opportunity counter,and meta tag on opportunities page. Using “Seesion” to avoid recounting the visitor who just visit our website in an hour. Add Robots.txt route to help search engines identify which pages need to be crawled. Furthermore, I add a multi-language resource which can let users switch to their preferred language.

Performance Level

We believe that we performed between good and superior. We have implemented the basic features we have laid out in our design document, in addition to some ideas that came to mind during implementation (e.g. database seeding, translatable dashboard) but were unable to add "above and beyond" features due to running out of time.

Calvin:

Entity Framework, at least our version, does not directly support many-to-many relations so I had to map these manually. Furthermore, database seeding had huge problems with the IDENTITY_INSERT flag rejecting our seeded data because we had specific primary keys. Previously we simply used [DatabaseGeneratedOption.None], but this means new instances of any model with this annotation could not be stored in the database. In the end I had to use raw sql to temporarily allow identity_insert during seeding but allow the database to generate primary keys thereafter.

Khoi:

I spent most of my time reading some documentation about the SignalR library and how to use it. Furthermore, I ran into some problems with enabling auto scroll with message history as well as enabling the enter button of message input instead of clicking on the “Send” button. On the other hand, I was having problems with storing the message in the database.

Ping:

Most of the stuff on the dashboard is easy, just need to spend time adding ajax method and controller response. The hardest part is when I try to change the culture of the current thread. I spend a lot of time figuring out that when it calls an action, it will run on a different thread. I tried to change the culture during the action instead of before executing action, and it did not affect my view page. I also spent time on studying what search engine optimization(SEO) is.

Summary

We have profiles with working tags that allow URC users to find each other and quickly skim profiles. Applications are now specific, so students can apply to only opportunities that they want. Profiles have working chats to allow users to message each other for such things as clarification or curiosity questions about the research opportunity. The database was reworked to allow new user data to be created (previously only seed data could exist), A dashboard tracks user statistics throughout the entire web app to give admins extra insight.