Project Proposal

Title: Priority Algorithm Processed Course Offering Application for Trinity (PAP-COAT / PapCoat)
By Edwin Aldrich, Logan Drescher, Bettina King-Smith

Project Statement

The purpose of our project is an application that serves as a replacement for Trinity's current course registration system that matches students with their preferred course based on a list ranking courses by preference instead of the current first-past-the-post system.

Background

When investigating the way other universities do course selection, we found that our course registration system shares a lot in common with other NESCAC universities and state schools. Aside from a slightly more visually appealing user interface, other NESCAC schools, like Tufts University and Connecticut College, appear to have course registration systems that are fundamentally the same as Trinity. One key thing to note that all of these institutions share is the same shopping cart and queue system to handle registration, and also all have a separate database to manage available courses. It has been incredibly difficult to find relevant information on this topic, but to the best of our knowledge, the queue system seems to be the most popular model for university course registration.

The only exception in the system that we found was at Bowdoin College, which has a round based system. Bowdowin has stated that they do not believe timing of participation is a valid measure in giving a student their desired course. Registration is held in multiple rounds, with each round lasting a few days. Students during the first round apply to any class they want regardless of seat size. Then, based on faculty preferences and prerequisites, students are placed into class sections. If a student is unable to take a desired class after the first 2 rounds due to a greater number of requests than seats available, they are granted increased priority for that class in the following semester. After 4 rounds, classes are set. While we feel that this multi-round system might be a bit complex to understand and time intensive to implement, we do admire Bowdoin's efforts towards creating a system where student and professor preferences are considered the first priority. We feel like a ranked algorithm system might offer a simpler and less time intensive solution, and serve as an improvement to Trinity's current registration system.

Goals

The aim of our project is to create a web application that could serve as a replacement for Trinity's current course registration system. Features of our minimum viable product will include:

1. **Database to store all course information.** Likely, the MVP will start with a pared down selection of courses (for instance, a couple of courses each from a few different departments) so that our team can focus on building an effective application as opposed to a holistic survey of all courses available at Trinity. After we are confident in our database schema, user interface prototype, and optimization algorithm, we will go back and add all courses in the course catalog into our system.
2. **UI where users can view, select, and rank courses in order of preference.** We want a system where, for instance, a student could peruse available courses, add their favorite ones to a list, and then rank these courses in order of preference. Currently, we are thinking about allowing each course to have 2 potential "back up" courses, but our final decision will hinge on our matching algorithm.
3. **Database to store student's ranking of desired courses.** We will need a database to store student information and their top choice classes.
4. **An optimization algorithm to match students with courses.** This will require significant research. Currently, we think game theory research may be applicable to our application, such that a student's favorite course would be given a score of *n*, their next favorite course a score of *n-1*, and so on. The goal of our algorithm will be to maximize the total score. We will also need to consider algorithm efficiency when programming it.

Time allowing, we would also like to extend our application to potentially have some of the following features, including:

1. **OAuth verification.** Since Trinity already uses OAuth authentication, we would research whether or not we could tap into that system's API. This would also allow us to define different types of users - for instance, differentiating between students, faculty, staff, and administrators - and granting each kind of user different levels of access.
2. **Course catalog editor for faculty, staff, and other privileged individuals.** This would allow Trinity employees to update the course catalog to add, remove, or edit course offerings or descriptions as needed. This would allow Trinity to better maintain the system we created after we graduate.
3. **Modifications to algorithm to consider student graduation year, student major, and faculty/student desired time slot.** Assuming we do end up using a game theory style score maximization algorithm, we may want to expand our algorithm to make score determination more complex. For instance, we may want seniors to have priority to classes that they need in order to graduate, or allow computer science majors to have priority access to computer science classes.

Significance

The reason why we are attempting to create a replacement for Trinity's course registration system is because we believe that the current system does not serve the best interests of students and professors in matching students with their preferred courses.

For example, consider two hypothetical sophomores, Jane and Joe. Jane's top choice class is an art history course, and her backup plan is to take an English class. On the other hand, Joe's top choice class is an English class, and his backup class is the art history class. Because registering for the art history class and the English class are both highly competitive, Jane and Joe have both classes in their shopping cart while registering. Suppose Jane gets the English class but is waitlisted from the art history class, while Joe gets the art history class but is waitlisted from the English class. Since the system does not take a ranking of student preferences into account and it is unlikely that Jane and Joe will be able to communicate with each other and swap places, both Jane and Joe will be stuck in their second-choice classes. This is a disservice to Jane and Joe, who would be happier taking the classes most interesting to them, and a disservice to their professors, who may be teaching a class where the students are only somewhat interested in the subject matter.

Methods and Procedures

Our general design approach will hinge on three focuses:

1) Abstraction
2) Robust design
3) Cleanliness

In order to properly develop a safe and effective application, our first task will be to conduct research. We plan to communicate with Trinity's Registrar Office to understand the current system and features that administrators wish they had to increase the chance that our system will be adopted by Trinity as a replacement. This will likely require us asking the Registrar's Office for access to restricted tools and privileged information that students normally do not have access to in order to gather requirements. Should Trinity not share relevant information with us, the development schedule will not largely change, but the feasibility of Trinity implementing our application will likely suffer.

After completing our research, we will finalize our selection of environment, libraries, frameworks, and other development tools, and begin development. To prioritize efficiency while retaining big picture understanding and project integrity, the initial development cycle has been delegated into three sections, with each section being led by one of the members of the team. That said, development tasks may swap hands in order to encourage familiarity with all parts of the web-app, as well as to facilitate skill development in other areas than just the domain of any memeber's lead section. Testing will occur continuously throughout development. This additionally becomes helpful if we later expand the scope of the project.

Should time exist to expand upon the product, we will attempt to add additional features. Towards that effort, we have added this small section describing the change in methods at that time. After achieving our minimum viable product, the section structure will dissolve, and instead each member will either start developing a new feature, or preparing the existing release to better integrate with the new feature.

Should Trinity wholeheartedly invest in our web app, we may need to work with Trinity to help transition from one system to the next. Time has been allotted after development in such a case, where further development would occur. Should the project miss its deadline and time to transition is not available, we will leave the project in a state that will be easy to implement and maintain.

Due to the sensitive nature of academic data, we as a team have delayed our decision on what frameworks, libraries, etc. to use. However, we have begun brainstorming:

Section leads:
Front End Engineer: Logan Drescher
Back End Engineer: Bettina King Smith
Database Engineer: Edwin Aldrich

Front End Options:
- JavaScript + ReactJS
- JavaScript + Angular
- Kotlin + KTor Framework

Back End Options:
- Java + SpringBoot
- Go
- Ruby

Database Options:
- MySQL
- Oracle 12c
- MariaDB (Encryption at Network, Server, and application stages)

Special Considerations

None.

Timeline


Gantt chart CPSC Project
Logan Drescher, Edwin Aldrich, and Bettina King-Smith | September 23, 2020
Start Date: 09/27/20
End Date: 03/14/20
Fortnight 1
Fortnight 2
Fortnight 3
Fortnight 4
Fortnight 5
Fortnight 6
Fortnight 7
Fortnight 8
Fortnight 9
Fortnight 10
Fortnight 11
Fortnight 12
General
Create Grand Scheme
Find what Trin Needs
Clearly Define Interactions
Finalize & Learn Project Tools
Database
Determine Best Option
Design a Secure System
Design and Develop Tables
Design and Develop Query List
Build Input-to-Table Converter
Optimize Database
Back-End
Determine Best Option
Design Secure System
Build Framework
Build Inter-system Interface
Develop & Build The Algorithm
Develop and Connect O-Auth
Finish Interfaceing
Front-End
Determine Best Option
Design a Secure System
Design Site Map and Profiles
Create Login
Develop Student Webpages
Develop Registrar Webpages
Develop Teacher Webpages
Make Sure It's Pretty
Legend:
Logan
Edwin
Bettina

References

Bowdoin College. “Registration Instructions, Fall 2020.” Accessed September 25, 2020. https://www.bowdoin.edu/registrar/students/course-registration/registration-instructions.html

Connecticut College. “Course Registration.” Accessed September 24, 2020. https://www.conncoll.edu/academics/registrar/course-registration/.

Tufts University. “Academic Calendars.” Accessed September 24, 2020. https://students.tufts.edu/registrar/what-we-assist/course-registration-and-scheduling/academic-calendars.

Tufts University. “Course Registration and Scheduling.” Accessed September 24, 2020. https://students.tufts.edu/registrar/what-we-assist/course-registration-and-scheduling.