CPSC 471 Final Report

Timothy Mealey, Ben Roberts, Cory Jensen, Scott Saunders

April 8, 2017

Abstract

The existing tools for planning semester schedules are decentralized, making them difficult to use together. The goal with this project was to simplify the semester planning process by creating a single tool that encompassed the functionality of the UofC calendar, myUofC course search, degree navigator, and the UofC schedule builder. While we managed to implement most of the desired functionality, we did not implement the degree navigator.

Introduction

Problem

The existing tools for planning semester schedules are decentralized, making them difficult to use together. When building schedules with these tools, it is common to have more than four tabs open in one's internet browser, which is an unnecessarily complicated state for such a simple task. The goal with this project was to simplify the semester planning process by creating a single tool that encompassed the functionality of the UofC calendar, myUofC course search, degree navigator, and the UofC schedule builder.

Implementation

A course search feature was built which displays course names, descriptions, prerequisites, and whether they are offered in any upcoming or current semesters. It is easily navigable using a searchable sidebar tree organized by faculty, department, and course number - selecting any of which loads that node in the main page. Selecting a node also updates the browser URL to a shareable link, without reloading the page, so that students can easily share their course planning with each other.

Similar to the courses page, the schedule page contains a sidebar tree from which one can choose courses. Differing from the courses page, this sidebar is organized by semester, using a dropdown menu, before dissecting into the different course branches. When one selects a course from the tree, its lectures, tutorials, and labs are added to a sidebar on the right. These sections can then be added to the calendar, located in the centre of the page, from this right side bar. After adding the desired sections to the calendar, the user can save their progress as a schedule.

A login system was started, from which users can be added to the database using an email and password. If a user exists in the database, they can log in using their unique credentials.

Design

Users

“Discuss the different users of your system. Your discussion in this section should be considerably more detailed than what you described for the presentation - this section should describe a complete transaction collection and, consequently, provide a complete picture of the functionality offered by your system.”

Browsing Courses

Anyone interested in the courses offered by the University of Calgary, would navigate to our website to evaluate potential classes. They would then search for preferred courses by topic, department, course numbers, or course names. Alternatively one could browse the convenient side-bar for a complete listing. As they select courses, the course info, availability and other information appears in the center of the screen.

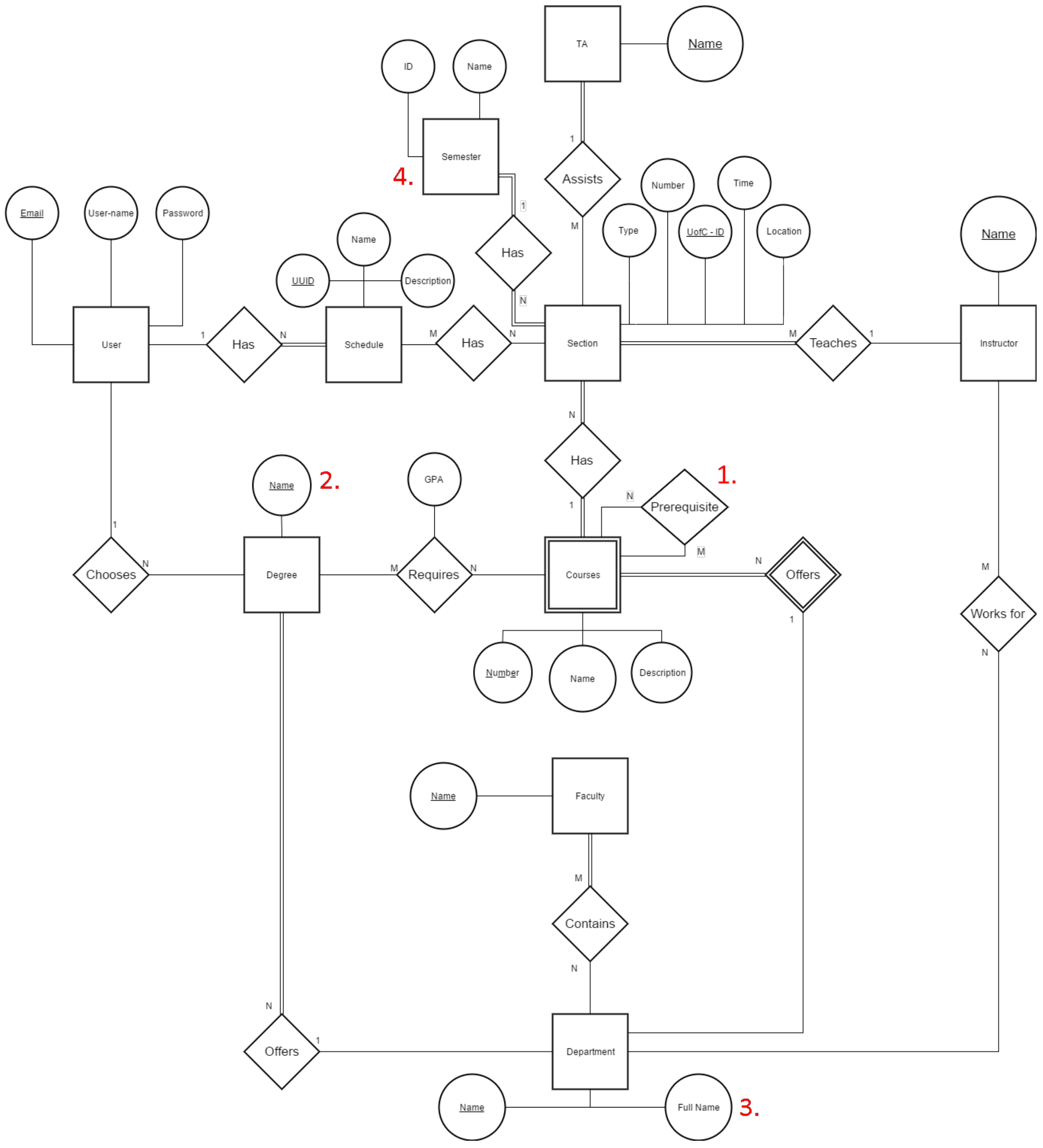
Building Schedule

Any Students or Perspective students getting organized for the upcoming semesters, would navigate to our website to construct a class schedule. Powered by the same technology of the dauwtrappen course list, one can find courses and add it to the schedule’s selected courses. On this list the user is provided a listing of class times for each section of the class, which they can then select the times they want. If a mistake was made, simply clicking the thing again... Once the user has the schedule to their liking, there are course numbers provided such that the user will be able to use these numbers to enroll directly in those classes via the my.ucalgary.ca Enrollment: add

Login System

Any new students, prospective students, or returning students, may use the login/account system to curate their own schedules, and checkup on their own interested courses. They would accomplish this by creating an account, (which adds them to the database), and verifying their email. From then on, they are free to login, and see any personalized data sets.

Entity Relationship Diagram



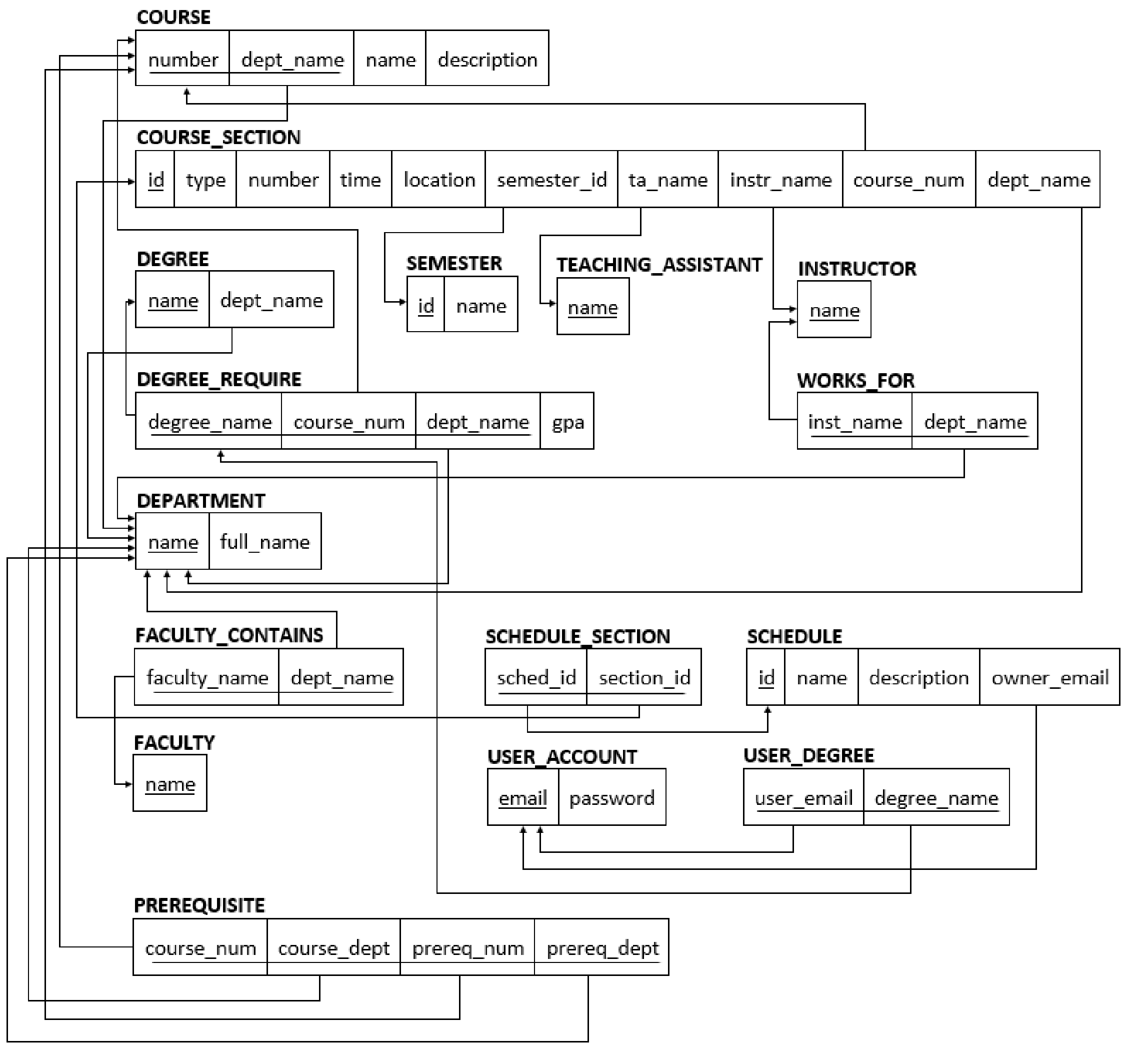
The red numbers on the diagram mark the changes from the initially presented diagram, and correspond to the list below.

1. A prerequisite relation from Course to Course was added.
2. Removed the “Course Requirements” attribute from Degree because it was redundant. It’s function is handled by the Requires relation from Degree to Course.
3. Added the “Full Name” attribute to Department because departments have a short code-name as well as a full English name. For example, “CPSC” and “Computer Science.”
4. A Semester entity type was added, and the Has relation from Semester to Section.

Implementation

Relational Schema Diagram

Below you will find the relational schema diagram, which was created using the algorithm described in class that converts ERDs to RSDs. As you can see, there is a tuple for each entity type and relationship from the ER diagram, with arrows pointing from foreign keys to the primary keys they reference.



During the conversion from the ERD to the RSD, redundancies were found and required information was missing. There were also some interesting constraints put on us by the DBMS that was used. For example, the “User” entity was renamed to “user_account” because “user” was a reserved keyword.

Database Management System

PostgreSQL is the DBMS used because it is a full-featured SQL implementation with good support and documentation. Here are the SQL queries that were written:

Get a list of all the semesters and the faculties they contain:

```
SELECT s.*, f.name AS fac_name
FROM faculty AS f,
  faculty_contains AS fc,
  course_section AS c,
    semester AS s
WHERE c.semester_id=s.id AND
    c.dept_name=fc.dept_name AND
    f.name=fc.faculty_name
GROUP BY s.id, f.name
```

Gets a list of all the semesters:

```
SELECT s.* FROM semester AS s
```

Gets a list of all the course sections from a faculty in a semester:

```
SELECT DISTINCT s.*, c.*,
    fc.faculty_name AS fac_name,
    d.full_name AS dept_full_name
FROM course AS c,
    faculty_contains AS fc,
    course_section AS s,
    department AS d
WHERE s.semester_id=<SEMESTER_ID> AND
    s.dept_name=fc.dept_name AND
    c.number=s.course_num AND
    c.dept_name=s.dept_name AND
    fc.faculty_name=<FACULTY_NAME> AND
    d.name=s.dept_name
```

Gets a list of all the course sections from a faculty:

```
SELECT DISTINCT s.*, c.*,
    fc.faculty_name AS fac_name,
    d.full_name AS dept_full_name
FROM course AS c,
    faculty_contains AS fc,
    course_section AS s,
    department AS d
WHERE s.dept_name=fc.dept_name AND
    c.number=s.course_num AND
    c.dept_name=s.dept_name AND
    fc.faculty_name=<FACULTY_NAME> AND
    d.name=s.dept_name
```

Checks if a user exists in the database with a given email and password:

```
SELECT email, password
FROM user_account
WHERE email=<EMAIL> AND password=<PASSWORD>
```

Checks if a user exists in the database with a given email:

```
SELECT email
FROM user_account
WHERE email=<EMAIL>
```

Adds a user account to the database with an email and password:

```sql
INSERT INTO user_account
VALUES (<EMAIL>, <PASSWORD>)
```

There are many more INSERT queries in the various scrapers and test-data scripts that populate the database with course information. These have been left out to shorten this document.

User Interface

“Present a brief description of your interface design, including several screenshots.”