CS-6400 Fall 2018

Team 12 Interim Project Report

PROJECT NAME: CoursePlanner

TEAM MEMBERS

• Choudhary, Palak; MS-ECE; choudhary.palak@gatech.edu

• Collins, Chris; MS-CS; ccollins63@gatech.edu

TEAM MEMBERS' BACKGROUND

Palak

Took the following database related courses during my Bachelor of Engineering degree at Jabalpur Engineering College, Madhya Pradesh, India:

- Database Management System – Fourth semester
- Data Warehousing and Mining – Seventh semester

Worked on an inventory database during my internship with the Indian Railways in summer of 2015 and 2016.

Chris

I have taken many database basics courses, many of which involved SQL. I took a Python for Informatics class online at UCSD Extension, which involved web scraping and basic database/SQL techniques. I also took an online SQL/Relational Algebra course with Stanford, which mainly covered querying and SQL basics. Additionally, I have some work experience with SQL, but only doing basic queries.

Problem Statement

We're aiming to develop a system for Georgia Tech CS graduate students that optimizes their course load. The scope will be limited to CS courses. Students often have a hard time finding a course load that checks all the boxes: convenient, intellectually stimulating, and appropriately challenging. Additionally, some students have special requirements they need their coursework to fulfill, such as minimum credits or graduation-related requirements. This is a difficult juggling act, and considering that a lot of the relevant data—such as course descriptions, historical course GPA's, and specialization requirements—is scattered across multiple sites, there is an obvious need for a tool that aggregates and distills this data in one place. Thus, by inputting their course load into our tool, students can get feedback about whether their course load is practical, and if there are other courses that would better suit their needs.

Approach to solve/ Functionality of the System

Our system will be a web app that lets users enter course numbers for their classes, which we then will match against our database. In the database, using data gathered from GT's Course Critique, we will have recent GPA info for each of the courses, with exceptions for courses that may have changed professors, are no longer offered, or are too new to have GPA data. We will also consider whether the students can reasonably walk from one class to another in the time between classes, among other things. Once the data on the course load is fetched, we will then compute the cumulative GPA based courses entered. Notably, we also prompt the student to input their desired cumulative GPA; if they achieve their desired cumulative GPA by

taking these courses, then their schedule is fine (assuming they have met all their specialization requirements). Otherwise, we try and find the best schedule for them that increases their GPA, while still prioritizing classes most relevant to their interests.

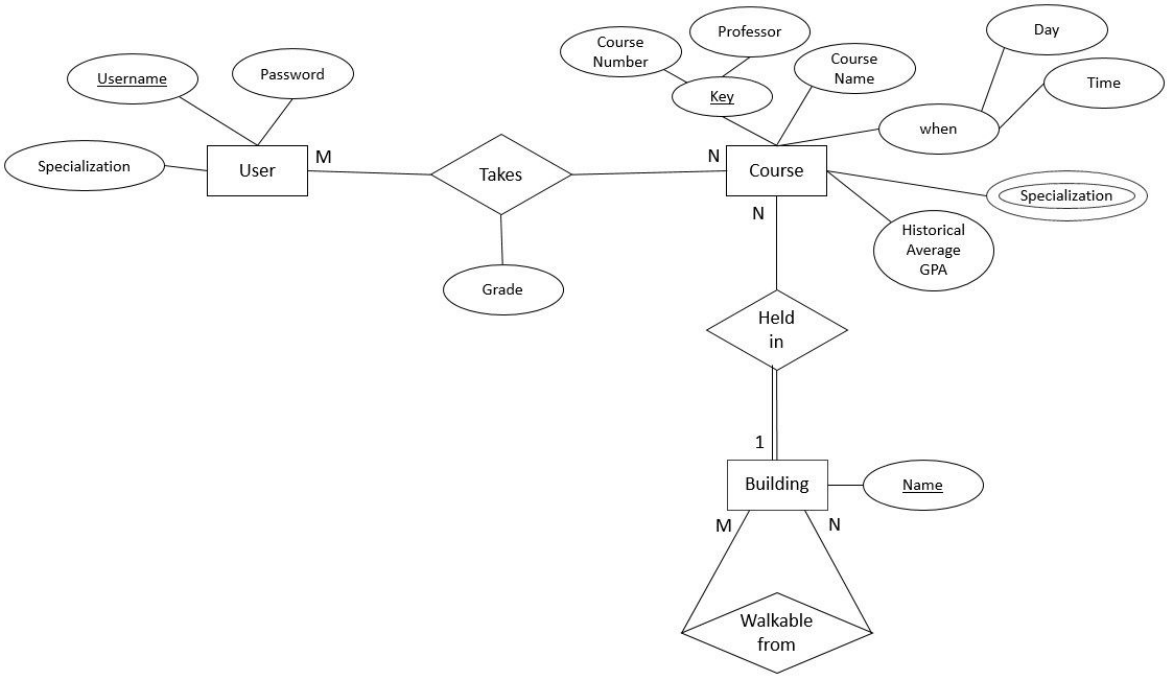
Data

Our database will have data scraped from three sources, and we plan to scrape data from each of these sources manually. The first source we intend to use is BuzzPort; specifically, we intend to gather the course number, professor, building, date, and time of classes being taught in Spring 2019.

We also intent to collect data from CourseCritique (https://critique.gatech.edu/); from here, we only want the course number, professor, and average course GPA. We may have to manually scrape this data as well, but we also might use a csv of data as our source.

Lastly, we intend to gather data about specializations (https://www.cc.gatech.edu/academics/degree-programs/masters/computer-science/specializations) and what classes are required for each respective specialization. This, too, will be scraped manually.

Design of Database



Tables

Courses

CRN	Name	Day	Time	Prof.	Building	Spec.	GPA

Building_Walkability

Building A	Building B	Walkability

User

Username	Password	Specialization

Courses Taken

Username	CRN	Prof	Grade

Interface

Before a user can use our system to generate a course schedule, they must first create an account. Defining their login credentials is a pretty standard process: they must input their Georgia Tech e-mail (guaranteed to be unique) and a password, which is stored as a hash. They also must list other data, however, to be able to use the system. First, they have to select their specialization from a drop-down menu; this is because we want to recommend them new courses which are in their specialization if we can, and not something irrelevant to their studies. Second, we have them enter any courses they've taken in the past. For each of these courses, the student must enter the course number and the grade they received, then select the professor from a drop-down menu. The drop-down menu is necessary, since some professors have names which may be entered incorrectly by a student; the drop-down prevents this. The point of this is that we need all this information to know what courses we can validly recommend to the student in the future. For example, if a student got a grade below a C in a class, then they can retake it. Additionally, if the student took an 8803 class, the course number is not enough alone to know what class it was; we must know the professor's name as well, since 8803 is a generic course number given to many "special topic" classes.

Once we know a student's course history, we prompt them to enter their classes in the same way they do when they create their account. We allow a maximum of seven courses to be

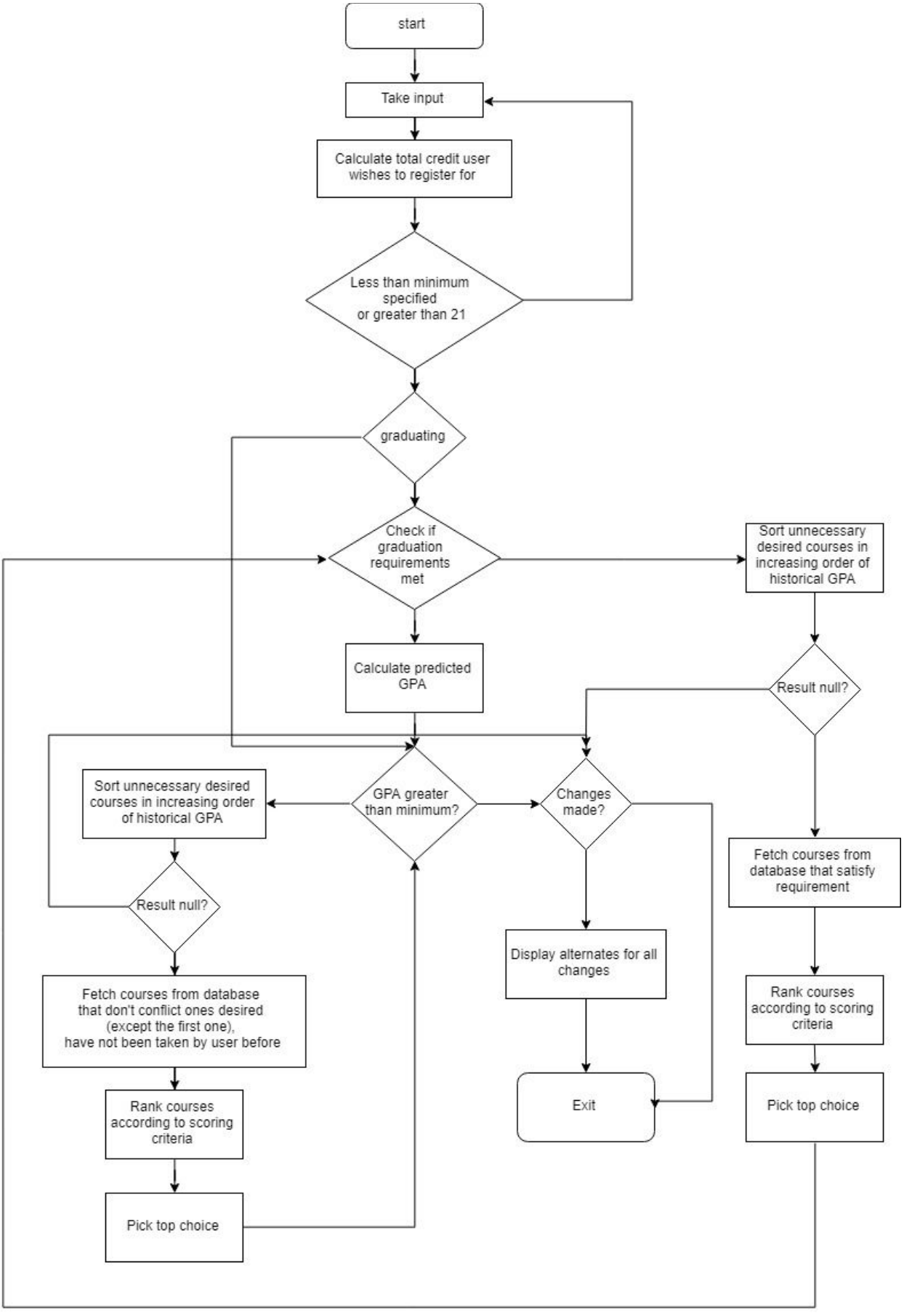
entered, since most courses should be worth three credit hours. Thus, as the school declares that they can take at most 21 credit hours, then they can take at most seven classes. We also prompt them to enter their desired cumulative GPA, to check whether we even need to recommend them classes, or if their schedule is already sufficient in that regard. They also can enter the minimum number of credits they need to take, if they have a visa, scholarship, or something similar that requires them to take a certain amount of units. Finally, they can enter whether they are graduating, in which case specialization requirements, minimum GPA requirements (3.0) and other such things take even greater precedence than usual.

Architectural Overview

The key part of our system is a score for each class. Each class has a score based on three criteria: walkability, time of class, and relevance to specialization. Each of these three criteria is binary. A class is either walkable or it isn't, and a class is either in a specialization or it isn't. Time of class is a binary value dependent on whether the chosen course occurs at the same time as the course we're trying to replace, meaning we can directly swap the two. These three criteria are weighted as follows: a weight of 1 for specialization relevance, 0.5 for class time, and 0.125 for walkability. GPA is unweighted and takes the normal 4.0 scale value. Thus, our scoring formula is as follows:

Score = GPA + spec relevance + .5 * time + .125 * walk

The flow diagram for our process is shown on the next page. One key thing to note is that we do not include non-graded courses, such as seminars, special problems, thesis, etc.


start
Take input
Calculate total credit user wishes to register for
Less than minimum specified or greater than 21
graduating
Check if graduation requirements met
Sort unnecessary desired courses in increasing order of historical GPA
Calculate predicted GPA
Result null?
Sort unnecessary desired courses in increasing order of historical GPA
GPA greater than minimum?
Changes made?
Fetch courses from database that satisfy requirement
Result null?
Display alternates for all changes
Fetch courses from database that don't conflict ones desired (except the first one), have not been taken by user before
Rank courses according to scoring criteria
Exit
Rank courses according to scoring criteria
Pick top choice
Pick top choice

Technologies

We plan to use bcrypt for generating password hashes, as its not practical to store passwords as strings. Our app would be made on Node.JS because of the excellent integration it has with MySQL, which we plan to use for our database.

Demo Plan

The demo will primarily consist of us walking through how a single example user would interface with the system. We would create an account for this user by entering their specialization, login info, and putting some classes in their course history. Then, we would try out different values for the input parameters to demonstrate all the functionalities of our system. We would, for example, change their desired GPA to show what such a change would do to our system. We could also change whether the student were graduating or what courses they were taking to show how our system handles such demands for different students' needs.

Beyond just demonstrating our web app in action, however, we will also unveil the process that went into building the app. We will show the ER diagram and database definition of our system, since this is the best way to present how our various data sources are stored. We will also justify our system design by explaining why we made our various design decisions, and what assumptions we made to make our problem tractable. Finally, we will spend a little bit of time on *why* we had to make such assumptions; there were many design challenges we faced, and we will elaborate upon how we worked around them and why they were so difficult to deal with.