COBA/SITC Course Scheduler Design

By Team Zaboo (Team 3), Oct 6, 2014
Raj Patel,
Jessica Lin,
Landon Gray

1 Main Introduction

Problem Statement:
No automated way of comparing student schedules to see the best time to move a class. The client specified that after students are enrolled in a class it is difficult to move class times. These class times conflict with other classes

Mission:
Provide Karen with a web based application that checks student schedules to see the best time to hold a rescheduled course.

Design Goals:

The purpose of this design document is to:

1. Help identify possible faults in our understanding about the overall system
2. Help assist developers to identify areas where we can design for change.
3. Provide the client with an overall outline how how the web app will be laid out
4. Help identify and point out areas where security methods need to be implemented
5. Express how this web app will meet the needs and goals of our client

Estimated Time of Completion:

A project of this magnitude should take approx. 90 - 120 hours to complete.

2 Architecture

Input/Output:
Users will be able to enter in a CRN and potential time change and results will consist of the total number of conflicts and which classes those conflicts are in. More detailed results will show which student conflicts are important by classification and the frequency a course is offered. This will be represented in calendar form, day of the week by time. Each entry will be highlighted a color of severity based off of the aforementioned measure of importance. There will also be a similar list of room availability in case a class needs to relocate as well.

Platform/Language:
For this project we will be using a system that runs on the web.We have chosen to use php as our main language for our web application(web app). As a result of this design decision we will be running out web app on a server running a LAMP stack. We choose an online platform because we would like our program to be system independent.

Data:
One of the largest portions of this project is the database. In this database will be the list of classes by CRN along with each classes' student roster, time of day, day of the week, and room number. There will also be each student's schedule and classification. This information is highly sensitive so security is a must and will be enforced through an application level authentication. The database will be interfacing with banner to get information from along with the front end of our project to print out this information.

We will be using a standard relational database called mysql. This design choice is due to the fact that we are using a LAMP stack. Also mysql is open source, free and widely available. It is used in many companies and open source projects and because of that it has been widely tested and is somewhat secure. In the future if developers decide to go with a different database system it will have minimal effect on our design. The only modules that will be affected are our api and and the database config file. Our view will not be affected because they only call the api.

There will be four entities with several attributes, the first table will detail the the 5000 students at ACU. We will also need a courses table. The third is the student schedule table, typically ranging from 3-6 courses. We need this entity in order to implement a many to many relationship between the students and courses table. The last table will hold classroom availability; a feature that isn't required, but if time allows it would like to be implemented.

Understanding entirely how the data from banner will be integrated isn't plausible as we have no real access to it. However, Dr. Reeves in a previous course gave us a sample table of what a courses schedule could look like, as shown below. As a result, our testing will be done with sample files, allowing a later adaptation for banner to be the source of information by authority figures.

Sample Schema

```
CREATE TABLE `students` (
  `id` smallint(6) NOT NULL DEFAULT '0',//some additional unique id besides banner
  `banner` smallint(6) NOT NULL DEFAULT '0',
  `major` varchar(10) DEFAULT NULL,
  `classification` varchar(50) DEFAULT NULL,
  `schedule` varchar(50) DEFAULT NULL,
  `expected_graduation` smallint(4) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
)
CREATE TABLE `schedule` ( // helps implement a many to many between courses and student table
  `crn` smallint(6) NOT NULL DEFAULT '0',
  `banner` smallint(6) NOT NULL DEFAULT '0',
  PRIMARY KEY (`crn`, `banner`)
)
CREATE TABLE `courses` (
  `crn` smallint(6) NOT NULL DEFAULT '0',
  `subject` varchar(10) DEFAULT NULL,
  `course_num` smallint(6) DEFAULT NULL,
  `section` smallint(6) DEFAULT NULL,
  `room` smallint(6) DEFAULT NULL,
  `days` char(8) DEFAULT NULL,
  `time_begin` smallint(6) DEFAULT NULL,
  `time_end` smallint(6) DEFAULT NULL,
  `enroll_max` smallint(6) DEFAULT NULL,
  `enroll_now` smallint(6) DEFAULT NULL,
  `title` varchar(50) DEFAULT NULL,
  `instructor` varchar(50) DEFAULT NULL,
  `cost` decimal(8,2) DEFAULT NULL,
  `paid` decimal(8,2) DEFAULT NULL,
  PRIMARY KEY (`crn`)
)
```

```
CREATE TABLE `availabilities` (
  `room_number` smallint(6) NOT NULL DEFAULT '0',
  `subject` varchar(10) DEFAULT NULL,
  `course_num` smallint(6) DEFAULT NULL,
  `section` smallint(6) DEFAULT NULL,
  `room` smallint(6) DEFAULT NULL,
  `days` char(8) DEFAULT NULL,
  `time_begin` smallint(6) DEFAULT NULL,
  `time_end` smallint(6) DEFAULT NULL,
  `enroll_max` smallint(6) DEFAULT NULL,
  `enroll_now` smallint(6) DEFAULT NULL,
  `title` varchar(50) DEFAULT NULL,
  `instructor` varchar(50) DEFAULT NULL,
  `cost` decimal(8,2) DEFAULT NULL,
  `paid` decimal(8,2) DEFAULT NULL,
  PRIMARY KEY (`room_number`)
)
```

3 Code

Introduction:

Languages we will be using in our project include standard web languages, markup, stylesheets. HTML, Javascript and CSS for front-end. We will also be using Php for back-end.

Frameworks:

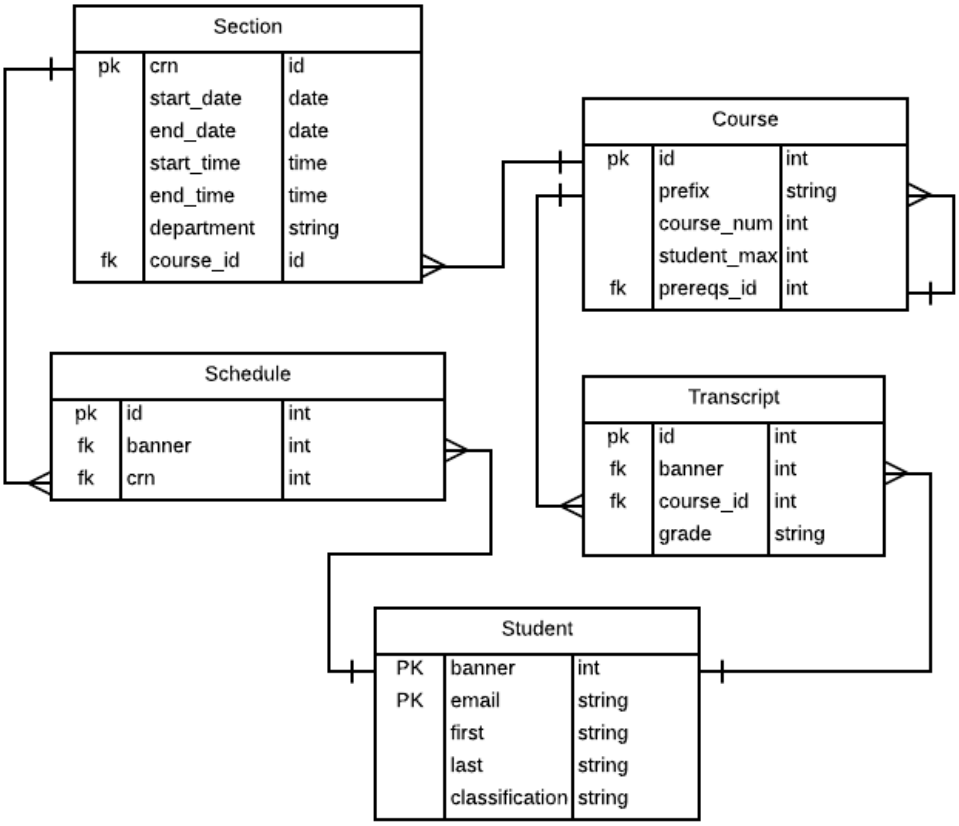
Bootstrap:

Bootstrap http://getbootstrap.com/ is a web framework used to make front-end development easy and fast. This will help us create clean user interfaces.

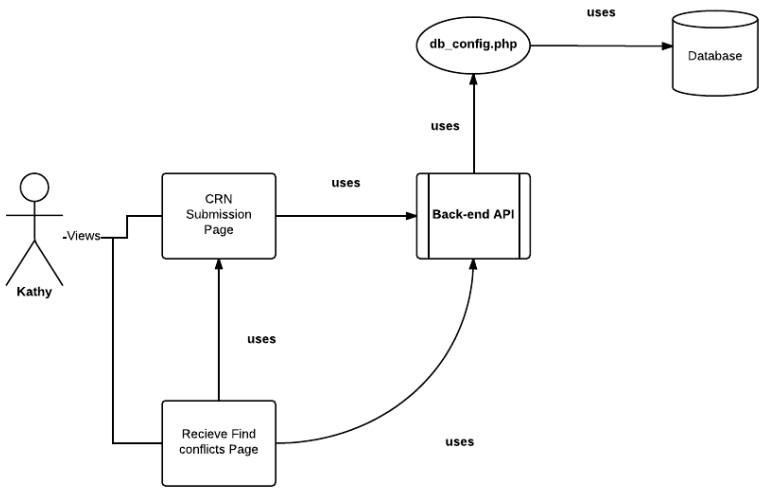
Google Calendar API:

We will use the Google calendar api Calendar API https://developers.google.com/google-apps/calendar/setup in our application in order to show the results of our clients queries into the system in a visually appealing way

ERD Diagram:


Section
pk crn id
start_date date
end_date date
start_time time
end_time time
department string
fk course_id id
Course
pk id int
prefix string
course_num int
student_max int
fk prereqs_id int
Schedule
pk id int
fk banner int
fk crn int
Transcript
pk id int
fk banner int
fk course_id int
grade string
Student
PK banner int
PK email string
first string
last string
classification string

UML Diagram:



Module:
Every file in our uml diagram except the database is consider a module.

View:
A view is a specific type of module. The purpose of a view is to be seen by a client so that they may interact with it. We have two views in our diagram

- **Submit CRN View:**
  User Kathy submits CRN and desired time for course specified. Returns a Receive/ Find conflict view
- **Receive/ Find Conflicts View**
  The module displays the course conflicts that have occurred based on what user submits via CRN View.

API (Application Programming Interface):
An API is a specific type of module thats only purpose is to providing other modules with function calls that allows these modules to access the database without knowledge of how the database is setup.
We currently have one API that could later be integrated into banner or even replaced by banner if needed.

- **Back-end/API:**
  We will create an API with availible function calls that return required data from the database. These will essentially be a series of sql queries on the database.

4 Operation:

User Types:

- Administrator: Data and User Management
- Advisor: searches conflicts for a schedule in a certain semester. (not the ones in the past)
- Students: just like advisors, but less output due to limited permissions.

Scenarios:

- Inputs student data(banner, classification, first name, last name, and a list of sections)
- Search in database for conflicts by:

1) time
2) course grade in transcript table of database
3) prereqs
4) max number of students in section

- Output conflicts for next semester.
- In Spring, output conflicts in summer.

5 Miscelanea:

This project would ideally be distributed to students, not just administrators and advisors so that students can check conflicts with their own schedule. We can easily abstract the portion that finds conflicts from the entire banner database and every other student schedule so that only that bit of code would be utilized by these unauthorized figures. Therefore, the only changes that would be made would be authorization. This would accept student username and password and see they are not faculty, and automatically limiting what's outputted to be just information that pertains to them,

Debugging is always a handful, but hopefully, by following the incrementality principle of design we can test with portions of the project to avoid searching through every bit we designed for some small crack.

5 Bibliography

Ghezzi, Carlo, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Englewood Cliffs, NJ: Prentice Hall, 1991. Print.

"Writing a Good Software Design Document." *BitFormation Consulting* -. N.p., n.d. Web. 06 Oct. 2014.