Application- Beta Design

Executive Summary: Karen, a degree plan specialist at ACU, was in need of an application that would ease in the process of taking students from a scheduled class and moving them into another timeslot. After she had moved a class, Karen also needed a way to view the possible conflicts students may have with this change. This application is intended to assist in the rescheduling of classes by finding the best possible timeslots for a scheduled class that has the least amount of serious conflicts.

Requirements:

Date Given	Summary
9/24/2014	User should be able to change the time (and possibly) location of a course
9/24/2014	List students who have conflicts after a class has been changed
9/24/2014	Rank the severity of each conflict by the student's classification and if the class is a required prerequisite
9/24/2014	User should be able to login to application. It needs to be secure as the information that is being used is confidential.
9/24/2014	Data that is used to display schedule must be taken from banner.
9/24/2014	Be able to search a class by CRN. Each class must have a unique CRN.

Work Breakdown Structure

Jeff Killeen: Project Lead. Has more knowledge and experience in the development of web applications and working with Ruby and HTML. Project Lead will provide direction and assistance where it is needed.

Jon David Ice: Backend developer. The business logic and database access of our application will be handled in this area with the help of Ruby and MongoDB.

Jonathan Nix: Frontend developer. The views of the application will be handled in this area. The frontend developer will be in charge of making the webpages responsive with HTML, CSS, and Javascript with some help from other libraries such has Twitter Bootstrap.

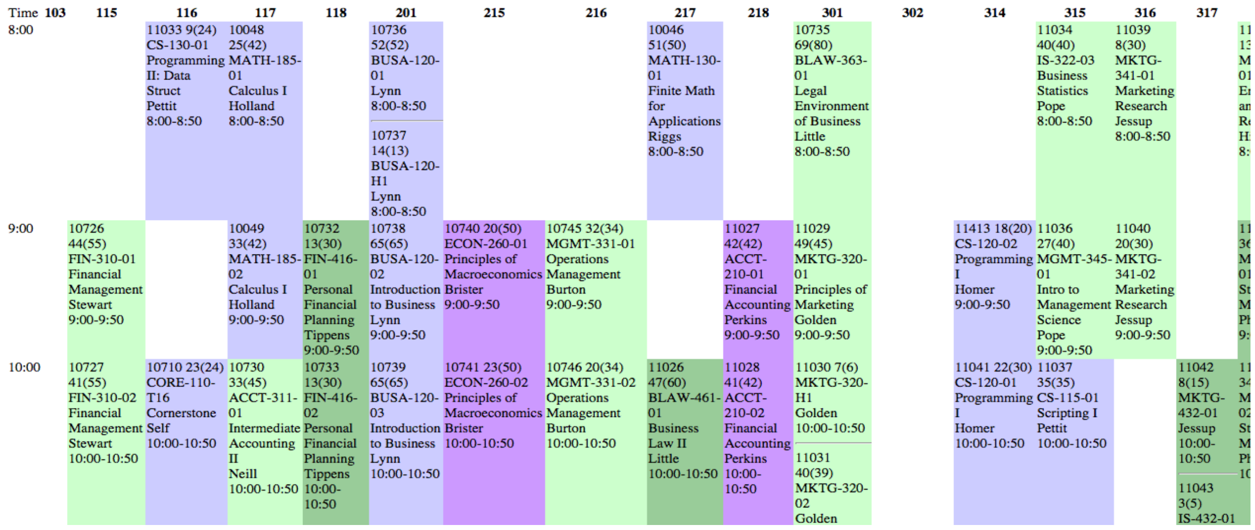
Project scope

Detailed design by activity

Functionality of Application

1) It will start off by prompting the user for a user name and password. (see security)

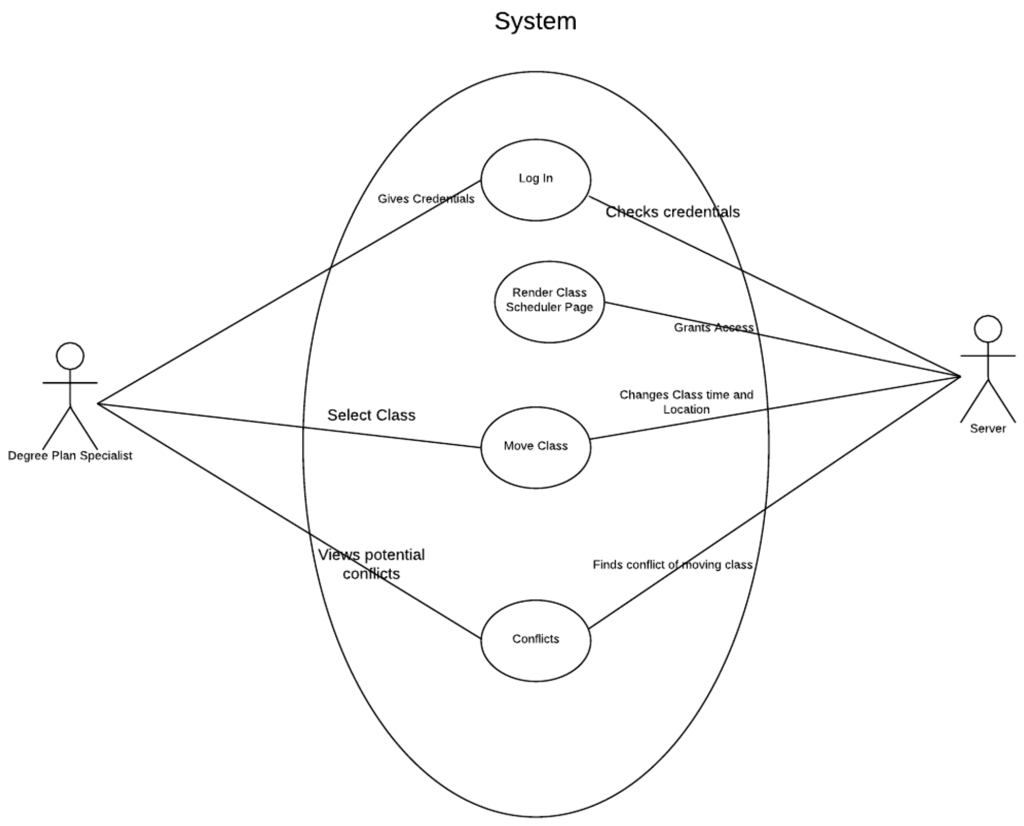
2) Once logged in, the user will be prompted with a drop down box to select a building. This will generate the class schedule calendar for that building. The picture below is an example of what the user will see.



Time	103	115	116	117	118	201	215	216	217	218	301	302	314	315	316	317	
8:00			11033 9(24) CS-130-01 Programming II: Data Struct Pettit 8:00-8:50	10048 25(42) MATH-185- 01 Calculus I Holland 8:00-8:50		10736 52(52) BUSA-120- 01 Lynn 8:00-8:50 10737 14(13) BUSA-120- H1 Lynn 8:00-8:50			10046 51(50) MATH-130- 01 Finite Math for Applications Riggs 8:00-8:50		10735 69(80) BLAW-363- 01 Legal Environment of Business Little 8:00-8:50			11034 40(40) IS-322-03 Business Statistics Pope 8:00-8:50	11039 8(30) MKTG- 341-01 Marketing Research Jessup 8:00-8:50		11 13 M 01 Er an Re H: 8:
9:00		10726 44(55) FIN-310-01 Financial Management Stewart 9:00-9:50		10049 33(42) MATH-185- 02 Calculus I Holland 9:00-9:50	10732 13(30) FIN-416- 01 Personal Financial Planning Tippens 9:00-9:50	10738 65(65) BUSA-120- 02 Introduction to Business Lynn 9:00-9:50	10740 20(50) ECON-260-01 Principles of Macroeconomics Brister 9:00-9:50	10745 32(34) MGMT-331-01 Operations Management Burton 9:00-9:50		11027 42(42) ACCT- 210-01 Financial Accounting Perkins 9:00-9:50	11029 49(45) MKTG-320- 01 Principles of Marketing Golden 9:00-9:50		11413 18(20) CS-120-02 Programming I Homer 9:00-9:50	11036 27(40) MGMT-345- 01 Intro to Management Science Pope 9:00-9:50	11040 20(30) MKTG- 341-02 Marketing Research Jessup 9:00-9:50		11 36 M 01 St M Ph 9:
10:00		10727 41(55) FIN-310-02 Financial Management Stewart 10:00-10:50	10710 23(24) CORE-110- T16 Cornerstone Self 10:00-10:50	10730 33(45) ACCT-311- 01 Intermediate Accounting II Neill 10:00-10:50	10733 13(30) FIN-416- 02 Personal Financial Planning Tippens 10:00- 10:50	10739 65(65) BUSA-120- 03 Introduction to Business Lynn 10:00-10:50	10741 23(50) ECON-260-02 Principles of Macroeconomics Brister 10:00-10:50	10746 20(34) MGMT-331-02 Operations Management Burton 10:00-10:50	11026 47(60) BLAW-461- 01 Business Law II Little 10:00-10:50	11028 41(42) ACCT- 210-02 Financial Accounting Perkins 10:00- 10:50	11030 7(6) MKTG-320- H1 Golden 10:00-10:50 11031 40(39) MKTG-320- 02 Golden		11041 22(30) CS-120-01 Programming I Homer 10:00-10:50	11037 35(35) CS-115-01 Scripting I Pettit 10:00-10:50		11042 8(15) MKTG- 432-01 Jessup 10:00- 10:50 11043 3(5) IS-432-01	11 34 M 02 St M Ph 10

3) The user will be asked which class they would like to reschedule. They will enter the CRN, Time, Day, and Room of the class and click “submit”.
4) Two representations of data will be generated.
   a) The calendar’s empty spaces will be populated with the number of students in the class that moving the class to that room and time slot affects.
   b) There will be a list of all the possible room and time slots to move the class ordered by increasing number of affected students.
5) Clicking on a field in the calendar or list will take the user to a page that gives a list of the students in the class, their classification, and which class the schedule change conflicts with.
6) Clicking on a specific student will display their current schedule for the user to figure out whether the conflict can be dealt with or not.

Use Case Diagram:



Architecture

In our application, we are going to base our implementation around the MVC design pattern. Using this design pattern will allow us to separate the logic so that there is no confusion on what each part of the application does and it will also allow the different layers of our application to communicate with one another.

Our application is initially set up using the Sinatra web framework. This means that the controller or business logic of our web application will be handled by Ruby. Component-based software engineering will be made easier for us thanks to Ruby's assortment of gems that have been made to speed up the web development process e.g. BCrypt gem, which can be used to hash and salt a password; or Pony, which will make it easy to email specific users of our site.

For the views aspect of our MVC design, we will be using embedded ruby files, or ERB for short. ERB will allow us to execute Ruby code on our previously existing HTML pages so that they will become much more dynamic.

Finally for our model, we will be making use of the Mongoid gem in Ruby, which allows us to wrap classes created in Ruby into Documents that will be used in MongoDB. The Mongoid gem lets us write code that appears to be written in Ruby that is easily translatable into MongoDB. MongoDB was chosen for the project because it is designed to be fast and scalable so that users will get minimum wait time for their queries.

Database

As mentioned above, we will be using MongoDB for the database for our web application. One helpful thing that MongoDB does is that it allows our entities to hold embedded documents. This means that we can easily see the information stored between one entity and another.

For our database, we will have multiple entities for our application. All of the entities have components that coincide with one another, expect for the User entity, which is primarily used for interaction with the site. The ERD listed below helps describes the relationships between our entities.

With the Mongoid gem, we are able to create classes in Ruby that can be translated into documents in Mongoid. The following classes will be created:

User: The User class is the only class that does not have a relation with any of the other entities. The User class will be used to keep track of users logged into the system.

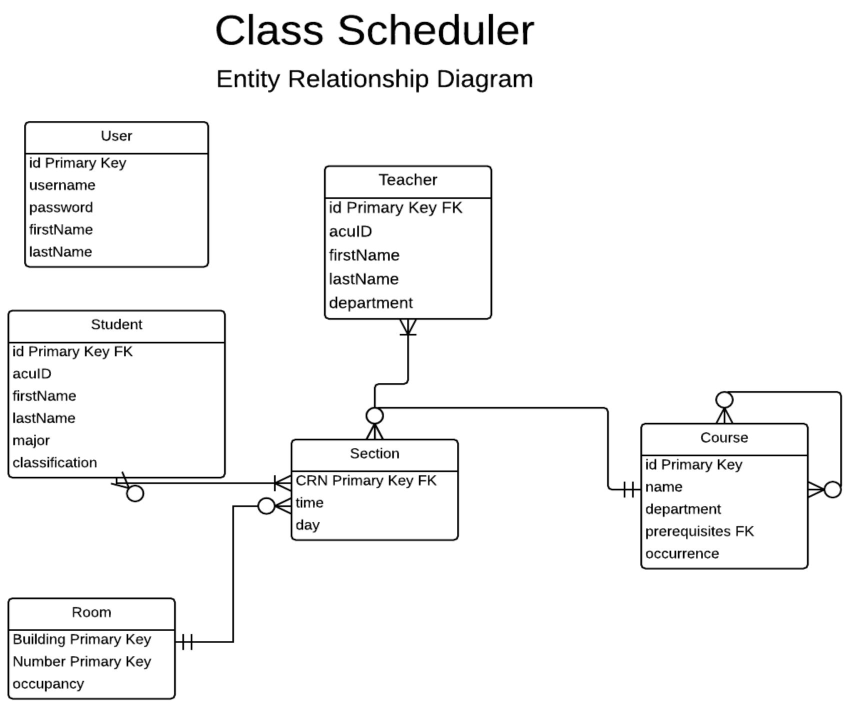
Teacher: A Teacher will have zero to multiple Sections. An important distinction to make is that a Teacher does not teach a Course but a Section. This will allow us to distinguish the different times a Teacher might be teaching a class.

Course: A Course will have a name, a department, and prerequisites, which will be a many-to-many relationship to itself. A course can have many sections or no sections at all, if it is not being taught that semester.

Section: A Section will have a unique CRN, with a certain date and time of the week. A section will belong to a Room.

Student: A Student will be enrolled in multiple sections. The student will need to take certain Sections that will correspond with a Course needed for graduation. The student's classification will determine the severity of changing a course within our application.

Room: A Room will have a building attribute and a room attribute to distinguish between other rooms. A Room can be home to many Sections and a Section must belong to some room. The occupancy attribute of the Room will determine the amount of Students allowed to enroll into a certain section.



Security

We want to make sure that only people who are authorized are able to use the application because they will be viewing confidential information. To ensure this, we will prompt with a form for their username and password. Once they submit the form, we will check their username to see if it already exists in our database. Then, if their password exists, we will hash their password with a salt and compare the salted password with the one stored in our database. Once both components match up, we will allow access to the user for our application.

Glossary

ACU ID- Both teachers and students have an ACU ID. This describes the unique ID that is used to identify a teacher or student.

Building- Each section has a building. This describes which building the section takes place in.

Course- A course is implemented by any number of specific sections

Classification- Each student has a classification. This describes the student's current standing at the university (e.g. Sophomore, Junior, Senior).

CRN- Each section has a CRN. This describes the unique number used to identify a section.

Day- Each section has a day. This describes which days the section is on.

Department- Each teacher has a department. This describes the department that the teacher belongs to.

First Name- Both teachers and students have a first name.

ID- Both teachers and students have ID's. This describes the unique number used to identify a teacher or student.

Last Name- Both teachers and students have a last name.

Major- Each student has a major. This describes what type of degree plan the student is currently on.

Name- Each course has a name. This describes what the content of the course is about to the user.

Occupancy- Each section has an occupancy attribute. This describes how many students are in a section and how many students can be in that section.

Occurrence- Each course has an occurrence attribute. This describes how often a course is offered by the school.

Password- A user has a password. Their password is used for security purposes and allows the user to access the application.

Prerequisites- Each course has a prerequisite. This describes which courses need to be taken in order for a student to take the course in question. This could be zero or more.

Room- Each section will be taught in a room

Room Number- Each section has a room number. This describes which room the section takes place in.

Section- A section is an instance of a course.

Student- A student is enrolled in any number of specific sections.

Teacher- A teacher teaches any number of specific sections.

User- A user is able to access the application via their username and password. They use the application to assist in rescheduling a specific section.

Username- A user has a username. The username is used for security purposes and, coupled with the password, allows the user to access the application.