Requirements Analysis for Course Management System

CSE 308: Software Engineering

Tom Biegner, William Peckham, John Paul Pennisi, Chris Scarola

Table of Contents

1. Overview Page 2
2. Super Admin Page 2
3. Department Admin Page 2
4. Student Users Page 3
5. Course Data Page 4
6. Major Data Page 5
7. User Files Page 6
8. Other Requirements Page 7
9. Use Case Diagram Page 8
10. Use Case Descriptions Page 9
11. Languages and Technologies Page 17

==========================================================================

1. OVERVIEW

The goal of this project is to create a system where students and administrators can upload course information and from that determine a possible schedule that can be used to fulfill major graduation requirements.

The system will consist of two different types of users. Students will be able to submit their current schedules and past schedules into the system. Administrators will be able to manage the information for courses and graduation requirements for their respective departments.

==========================================================================

2. SUPER ADMIN

Super admins will be able to first log into the system and manage all course information stored on the database. Possible functions will include:

upload course information file

edit/manage course offerings

delete course offerings

Upload Course Information File. This function allows super admins to upload a file containing all courses available. The file will also contain semesters offered, meeting times, and prerequisites for the courses.

Edit/Manage Course Offerings. This function allows super admins to edit and manage the course information that are offered.

Delete Course Offerings. This function allows super admins to delete courses listed as being offered.

==========================================================================

3. DEPARTMENT ADMIN

Department admins will be able to first log into the system and manage course data for their specific department. Possible functions will include:

upload a major for his department

add graduation requirements for their department

edit graduation requirements for their department

delete requirements for their department

view all course information for their department

Upload a Major for Their Department. This function allows department admins to add a major in their department with a guideline of requirements for graduation.

Add/Edit/Delete Graduation Requirements for Their Department. These functions allow department admins to add, edit or remove from the list of courses required for graduation as well as editing other requirements for graduation such as G.P.A or credits.

View All Course Information for Their Department. This function allows administrators to view all courses and graduation requirements for their department.

The department admin will need to log in using a departmental password in order to gain access to these functions. The system will also need to store the following data to allow administrator access.

department name

password

==========================================================================

4. STUDENT USERS

The system will allow student users to generate a sample schedule by uploading their current transcript. Student users will be able to do the following through the system:

upload completed courses file

edit information

add completed courses

check fulfilled requirements

check remaining requirements

generate sample upcoming schedules

download the edited document

Upload Completed Courses File. This function allows student users to upload a file that contains the data representing courses that they have completed and the grade for each course. The system can then read that file and then generate a list fulfilled requirements.

Edit Information. This function allows student users to edit information about their course history and grades record for the courses.

Add Completed Courses. This function allows student users to add recently completed courses to their course history as well as their received grade for the course. The system will then maintain that information as part of the uploaded file.

Check Fulfilled Requirements. This function allows student users to check the graduation requirements they have fulfilled according to their current course record.

Check Remaining Requirements. This function allows students users to check the graduation requirements they have yet to fulfill according to their current course record.

Generate Sample Upcoming Schedule. This function allows student users to view as sample schedule of classes that they can take in semester order so that they can graduation. It also allows users to see the minimum number of semesters required for graduation.

Download the Edited Document. This function allows student users to download the edited document containing updated information about course history and grades.

==========================================================================

5. COURSE DATA

The following information will need to be maintained by the system for each course.

course department

course number

majors fulfilled

course offerings

prerequisites

meeting time

minimum grade

major category

requirement start year

requirement end year

credits

Course Department. This data represents the department to which the course corresponds to.

Note: This is not necessarily correspond with the major.

Course Number. This data is the department identifier for the course.

Note: If the course number is greater than or equal to 300 the course is considered to be an upper division course.

Majors Fulfilled. This data represents the possible majors that this course can fulfill credit for.

Note: Some courses can fulfill multiple majors.

Course Offerings: This data represents the semesters for which this course is offered.

Note: This will be a two bit number representing the semesters for which it is available. The first bit will represent whether the course is offered in the spring while the 2nd bit will represent if the course is offered in the fall. I.e. a course that is both offered in the spring and fall will be represented by a 3 (11)2.

Prerequisites. This data represents courses that need to completed before this course can be taken. This can be multiple courses and is important for generating scheduling.

Meeting Time. This data represents the meeting time for the course in the future. This can help generate schedules.

Note: Because of the semester by semester inconsistency in meeting times this field may need to be removed.

Minimum Grade. This data represents the minimum grade the student must ascertain so that the course will count for major credit.

Major Category. This data represents the category in the major that this course fulfills.

Note: A course can only satisfy one category in a major. I.e. A course cannot count towards a major elective and another requirements.

Start Year. This data represents the matriculation year where the course will first be applied as a major requirement.

Last Year. This data represents the last matriculation year where the course will be applied to a major requirement.

Credits. This data represents the number or credits towards the major the student will acquire.

==========================================================================

6. MAJOR DATA

This system will need to maintain the following data for each major registered in the system:

major name

department

categories

required courses

minimum credits required

Major Name. This data represents the 3 letter representation of the major.

Department. This data represents the department to which the major belongs.

Note. This allows majors to be controlled by their respective administrators.

Categories. This data represents a certain aspect of the major that requires a number of courses to fulfill.

Required Courses. This data represents all the courses that can fulfill requirements for the major.

Minimum Credits Required. This data represents the minimum number of credits required for graduation for the major.

==========================================================================

7. USER FILES

The following types of files will be uploaded by users for data entry:

Course listing file

Major requirements file

Student info

Course listing file-This file will be the file used by the super admin to upload course offering information. This file will need to maintain the following information about each course:

Course name

Meeting time (for scheduling)

Semesters offered

Prerequisites (as a non-comma delimited string)

This file be comma and line delimited.

Major requirements file- The major requirements file will be used by department administrators to upload major requirements. It will be able to be edited by them after upload, and can be re-downloaded and saved. The following data will need to be maintained by the file.

Major name

Minimum GPA

Required Courses

Minimum Grade

Student info file- This file will be used by the student to upload personal data and course data into the system. The information that will need to be contained in this file is:

Name

Major

Class name

Grade

The student can edit this file after it is uploaded, and the student will be able to re-download this file. It should be in comma separated format.

==========================================================================

8. OTHER REQUIREMENTS

Import CSV. The system will be able to import a transcript containing information about the courses a student has taken or is taking. The file contains lines with comma-separated-value format shown below:

semester,course,grade

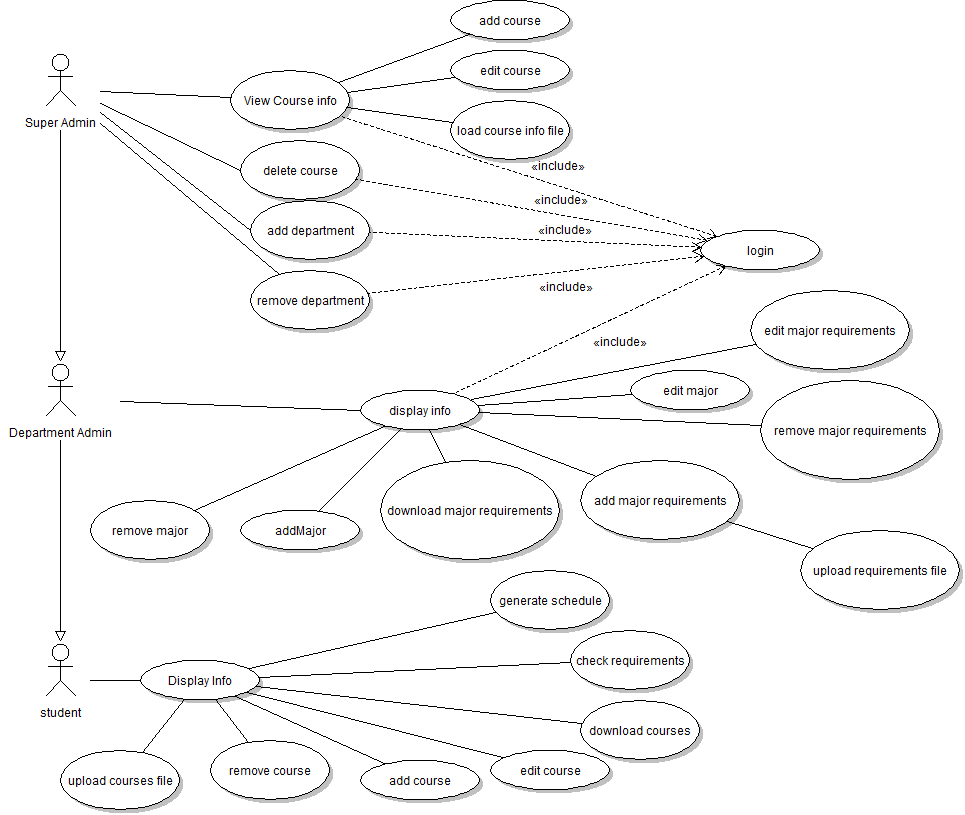
I.e.

Fall 2009, CSE 308, A

Concurrent Session Management. The system will be available to multiple users at the same time.

Note: administrative functions might be disabled while system is in use.

System security. Administrative functions will be password protected. Each department will have its own password.



**Use Case Diagram**

Use Cases for Course Management System

==========================================================================

Upload Course Information File

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: file info loaded

Step

1 Prompts for upload file

2 User enters desired file

3 System reads and parses file into memory

4 System shows message "File successfully loaded

Alternate Flow: invalid File

Pre-Condition: file invalid format

3 Show error "File not in valid format"

4 Go to step 1

==========================================================================

Add Courses

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: course added

Step

1 System shows all course offerings

2 User enters course information

3 User submits selection

4 System adds course

5 System shows message “course successfully added”

6 Go to step 1

==========================================================================

Edit Courses

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: course edited

Step

1 System shows all course offerings

2 User selects course to edit

3 User changes appropriate course information

4 User submits selection

5 System edits the course information

6 System shows message “course succesfully edited”

7 Go to step 1

==========================================================================

Delete Courses

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: some or all courses deleted

Step

1 System shows all course offerings

2 User selects courses to delete

3 User submits selection

4 System deletes selected courses from memory

5 System shows message "selected offerings successfully deleted"

6 go to step 1

==========================================================================

Add Departments

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: department added

Step

1 System displays list of departments

2 User clicks add department button and enters name of department to add

3 User submits selection

4 System adds department to memory

5 System shows message “department successfully added”

6 Go to step 1

==========================================================================

Remove Departments

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: department removed from database

Step

1 System displays list of departments

2 User selects department to delete from list

3 User clicks the remove department button

4 System removes selected department from memory

5 System shows message “department successfully deleted”

6 Go to step 1

==========================================================================

Display Info

Primary actor: Department Admin

Secondary Actor: None

Pre-condition: user wants to view the current information

Post-condition: system displays information in a formatted way

Step

1 System displays information that is contained in the students information

==========================================================================

Add Major

Primary Actor: Department Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: Major Added

Step

1 System displays list of majors

2 User clicks add major button and enters name of major to add

3 User submits selection

4 System adds major to memory

5 System shows message “major successfully added”

6 Go to step 1

==========================================================================

Edit Major

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: Major name is edited

Step

1 System displays list of majors

2 User selects major from list and clicks edit major button

3 User enters new major name

4 System changes the name of the major

5 System shows message “major successfully edited”

6 Go to step 1

==========================================================================

Remove Major

Primary Actor: Super Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: department removed from database

Step

1 System displays list of majors

2 User selects major to delete from list

3 User clicks the remove major button

4 System removes selected major from memory

5 System shows message “major successfully deleted”

6 Go to step 1

==========================================================================

Upload Major Requirements

Primary Actor: Department Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: Major requirements updated

Step

1 System Prompts for file

2 User selects file

3 System parses file

4 System updates course info

5 System shows message "Major requirements have been updated"

6 go to step 1

Alternate Flow: error in reading file

Precondition: error deleting

5 Show error "File not in valid format"

6 go to step 1

==========================================================================

Delete Major requirements

Primary Actor: Department Admin

Secondary Actor: None

Pre-Condition: Logged on

Post-Condition: some or all major requirements deleted

Step

1 System shows all major requirements for majors in department

2 User selects requirements to delete

3 User submits selection

4 System deletes selected major requirements from memory

5 System shows message "selected requirements successfully deleted"

6 go to step 1

Alternate Flow: error in deletion

Precondition: error deleting

5 Show error "Error deleting requirements"

6 go to step 1

==========================================================================

Add Major Requirements

Primary Actor: Department Admin

Secondary Actor: none

Pre-Condition: Logged on

Post-Condition: Requirements for major are added in database

Step

1 System prompts for new requirement info

2 User enters new info

3 System inserts requirement into requirements database

4 System shows message "selected requirements successfully deleted successfully deleted"

5 go to step 1

Alternate Flow: error in insertion

Precondition: error inserting into database

4 Show error "Error adding requirements"

5 go to step 1

==========================================================================

Download Major Requirements

Primary Actor: Department Admin

Secondary Actor: none

Pre-Condition: Logged on

Post-Condition: requirements for major downloaded into file

Step

1 System displays list of majors

2 User selects major to download requirements for

3 System prompts for file

4 User selects file to save to

5 System writes to the selected file

6 System shows message "major requirements successfully written to file"

7 go to step 1

Alternate Flow: error in writing file

Precondition: error writing file

6 Show error "Error writing to file"

7 go to step 1

==========================================================================

Edit Major Requirements

Primary Actor: Department Admin

Secondary Actor: none

Pre-Condition: Logged on

Post-Condition: requirements for major are edited in database

Step

1 System shows all major requirements for majors in department

2 User selects requirement to edit

3 System prompts for new requirement info

4 User enters and submits info

5 System inserts requirement into requirements database

6 System shows message "selected requirements successfully deleted"

7 go to step 1

Alternate Flow: error in update

Precondition: error updating database

6 Show error "Error adding requirements"

7 go to step 1

==========================================================================

Check Requirements

Primary Actor: Student

Secondary Actor: None

Pre-Condition: Major is available

Post-Condition: Requirements met will be displayed to screen

Step

1 System checks students’ courses against requirements

2 System generated list of fulfilled and unfulfilled requirements

3 System displays requirements screen

==========================================================================

Suggested Schedule

Primary Actor: Student

Secondary Actor: None

Pre-Condition: Major is available

Post-Condition: Suggested scheduled for graduation completion is displayed on screen

Step

1 System gets fulfilled requirements

2 System gets unfulfilled requirements

3 System generates schedule from requirement data and class history

4 System displays generated schedule on screen

Alternate Flow: error in generating schedule

Precondition: error generating schedule

1 Show error message "Schedule could not be generated"

==========================================================================

Select Major

Primary actor: Student

Secondary Actor: None

Pre-Condition: User wants to select a major

Post-Condition: Selected major is in the current student information

Step

1 System prompts user to select a major

2 User selects a major

3 System creates empty student information with that major

==========================================================================

Upload File

Primary actor: Student

Secondary Actor: None

Pre-condition: User wants to upload a file

Post-condition: Current student information contains a all information in the file

Step

1 System prompts user for a file to load

2 User browses local system for the file and selects it

3 System determines what the type of the file

4 System fills student information with the information in the file

Alternative Flow: File is not valid

Pre-condition: File cannot be parsed correctly

4 System shows the user an error message

==========================================================================

Display Info

Primary actor: Student

Secondary Actor: None

Pre-condition: user wants to view the current information

Post-condition: system displays information in a formatted way

Step

1 System displays information that is contained in the students information

==========================================================================

Exit

Primary actor: Student

Secondary Actor: None

Pre-condition: User wants to exit

Post-condition: System discards the student information

Step

1 System discards the information from the student information

==========================================================================

Download File

Primary actor: student

Pre-condition: user wants download the student information

Post-condition: system sends a file containing the student information to the user

Step

1 System sends generated file to user

==========================================================================

Edit Info

Primary actor: Student

Secondary Actor: None

Pre-condition: User wants edit a class from the info display

Post-condition: System changes the student information

Step

1 User chooses an element from the display to edit

2 System shows the user the editing form

3 User edits the information available

4 System changes information in student information

==========================================================================

Add Class

Primary actor: Student

Secondary Actor: None

Pre-condition: User chose to add a class

Post-condition: Class is added to the student information

Step

1 System displays the add class form

2 User fills the form out

3 System adds class information to the student information

==========================================================================

Remove Class

Primary actor: Student

Secondary Actor: None

Pre-condition: User wants to remove a class

Post-condition: System removes class from the student information

Step

1 System displays all classes in the student information

2 User selects one class to delete

3 System removes selected class from the student information

**Languages, Technologies, and Tools**

To create the UML diagrams we used Violet and Paint. In the implementation we will be using Java as the language. For development we will be using Eclipse and or Net Beans IDE. We will also be using the JPA, JDBC, Sockets API and Swing.

John Paul is familiar with everything except for Sockets API, JPA, and JDBC.

Tom has experience in Sockets, JDBC and Swing with some minor experience with JPA.

William has experience with everything except JPA and JDBC.

Chris is familiar with everything except for Sockets and JPA.

To learn certain technologies that we were not familiar with we used certain tutorials such as:

* <http://www.javaworld.com/jw-12-1996/jw-12-sockets.html> - Socket Programming
* <http://download.oracle.com/javaee/5/tutorial/doc/bnbpz.html> - JPA
* [download.oracle.com/javase/**tutorial**/**jdbc**/index.html](download.oracle.com/javase/tutorial/jdbc/index.html) - JDBC

To go along with the tutorials we had each team member who understood a technology help with whatever aspects of that technology other team members did not fully understand. We tried to work as a team to help each team members have a good grasp on all technologies even though they all might not focus on certain ones.