**Advisement Module Test Case: 1019AMT1.0**

**Test Case Specification Identifier**

* Unique ID: <ddmm>AMT<version>
* Version: 1.0
* Date: 10/19/2016
* Level: For use as unit test case and integration test case.
* Version Author: Geoffrey Pitman

**Test Items**

* Generate Warnings
  + Given a set of courses a student has taken, the respective course grades, and the student’s major, provide a warning if the student is at risk of a graduation delay.
    - The warnings will be sent to the Alerts log, and used by the Faculty UI module to visually represent them for the adviser.
    - To further specify what a “warning” is: a course that must be taken in the upcoming semester to prevent a graduation delay
* Generate Adviser-form Data
  + Provide data that corresponds to the forms that advisers are frequently required to fill out.
    - The data will be sent to the Forms log, and used by the Faculty UI module to visually represent it for the adviser.

**Input Specifications**

* Modules
  + Faculty UI
    - Student ID
* Human Actions
  + Adviser has entered student ID and clicked the ‘Update Record’ button
* Files
  + CSIT Course Data
    - csit.dat
  + Checksheets
    - See System Data
  + Student Record
    - alerts.log
    - courses.log
    - forms.log
    - student.dat
* Relationships
  + The Advisement Module is executed after the Advisor has indicated a desire to retrieve a record from the Faculty UI module, and completes before the Scheduling module is executed.
* Data
  + Checksheet
    - see System Data
  + Student Record
    - Major
    - Courses taken (semester, grade received)
    - Student ID

The input needed to start the test, passed from the Faculty UI module, is the Student ID, which is subsequently used to retrieve the student’s records from the log file directory structure. The records will be compared to the student’s major requirements to find out the courses that still need to be taken for their chosen major. Those courses will be analyzed for prerequisites to determine how many more semesters it will take to fulfill the full course sequence (sequence meaning the course and all of its prerequisite courses [of which could have prerequisites of its own] and requirements [mainly a minimum number of major requirements]). General education requirements have no prerequisites so they are incorporated into the results as one of the last steps of the algorithm. Finally, all course sequences are analyzed, including the sequences that have no prerequisites, meaning they are 1 course sequences (i.e. gen eds) and/or the sequence prerequisites have been satisfied. The longest sequence chains are compared, and courses in the same position are compared. For example, more courses require 135 than 125. Even though 125 has no prerequisites, it is in a lower priority position than 135 because it has less dependencies. When the remaining courses have the same priority, they are prioritized in order of major, university competencies, and finally gen eds. In summation, the algorithm puts courses in a priority list and then takes the difference between the position the course is in the list and the remaining semesters the student has, where total semesters equals the ceiling of the total number of credits for the degree divided by 15\*\* credits (credits taken per semester).

\*\*This plan assumes the student is taking 5 classes, or 15 credits, per semester, however the system itself has no sense of time. This is noted because it is theoretically possible to graduate in less than 4 years by taking more than 5 classes per semester, and/or taking classes over winter and summer sessions. Alternatively, if a student wants to graduate in 8 years the system will not begin to accumulate warnings.

Ex. Analysis on first semester freshman starting with CSC135:position 8

-Backtracking from 355 we find that 8 more semesters are needed to complete the degree from the start of 135. 8 semesters left for a first year freshman, minus 8 semesters needed to complete degree = 0 semesters.

-Any result that equals 0 must be taken in the upcoming semester or it will cause a graduation delay. Anything higher than 0 is prioritized accordingly. Anything less than 0 means a graduation delay WILL occur (unless corrective action is taken in winter/summer semester, or 6 classes are taken during fall/spring, etc).

**Output Specifications**

* Data
  + GPA
  + Warnings
  + Anticipated graduation date
  + Form field information
* Modules
  + Scheduling
    - Student ID
    - Prioritized course list
* Relationship
  + The completion of the Advisement module is the trigger for the Scheduling module.
* Files
  + alerts.log: list of courses that MUST be taken in upcoming semester to prevent graduation delay
  + forms.log: form field information
  + student.dat: updated GPA and anticipated graduation date.

The student\_ID is also passed to the Scheduling Module.

Output is sent to the student record logs, and subsequently collected first by the Scheduling Module, and finally the UI module. Output includes Course Warnings. This is a list of the courses that MUST be taken in the upcoming semester to prevent the graduation date from being pushed back at least a semester. Some courses are offered only in the fall/spring. If the student misses a fall/spring-only course it costs them 2 semesters. The student’s updated GPA and graduate date are output to the student data files as well. Output passed to the Scheduling module includes the student’s remaining courses (by priority) and the advisee student ID.

**Environmental Needs**

* System permission to execute server
* System permission to perform file IO
* Software
  + Node.js
  + npm
  + Browser
    - Chrome or Firefox

**Special Procedural Requirements**

There are countless ways a student could complete a degree, and extremely difficult to define a range of values, because there is nothing forcing a student to take courses in any specific way. As such, it may be impossible to provide every possible input. The algorithm used in the Advisement module does, of course, use a finite approach when determining course sequences and prioritizations. Please refer to the CSIT Course Dependencies Graph document. This graph can be compared with the student’s checksheet to visually find the correct output for a given set of input data. Please refer to input specifications.

**Intercase Dependencies**

* No prerequisite testing required in unit testing phase. In integration testing phase, the Faculty UI module must acquire the updated student records before this test case can be executed.
* After entering into the integration testing phase, this test case is a prerequisite of the Scheduling module test. Not required for unit testing.