Graduation Requirement Assessment Data System – System Requirements

Version <1.1>

Group # 15:

Dan Belling	3823441	*<belli058@umn.edu>*
David Cavalari	4586042	*<magax001@umn.edu>*
Christine Hallstrom	4210229	*<hall1404@umn.edu>*
Guobao Sun	4867911	*<sunxx987@umn.edu>*

Additional Information:

Instructor	Mats Heimdahl
Course	Software Engineering I – Csci 5801
Date	October 22nd 2014

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Dan Belling, David Cavalari, Christine Hallstrom, Guobao Sun	This is the initial completed version of the SRS for GRADS.	10/06/2014
1.1	Dan Belling, David Cavalari, Christine Hallstrom, Guobao Sun	Revised based on TA's comments.	10/22/2014

1 Introduction

The Graduation Requirement Assessment Data System will serve as an interactive database software system which grants exclusive permissions to students and program coordinators to compare and contrast student achievements (as logged on their student records) with specifically determined graduation requirements as dictated by their graduate path (Ph.D, M.S. Plan A, B or C) of choice.

1.1 Document Purpose

The purpose of this document is to present the functional framework of the Graduation Requirement Assessment Data System (GRADS). To address this, we intend to outline a specific user base, operating environment, and provide explicit characteristics of GRADS.

1.2 Product Scope

GRADS will serve as a simplified search and compile system for student records. Students and coordinators will be allowed to search the database for student records, while comparing student records, with graduation requirements. Upon retrieving a students record, he or she may view a summary of their respective achievements and other characteristics including their technical GPA, identification number, committee members and other pertinent milestone dates. Students will also be able to generate 'What-If' reports to track their progress towards other graduate paths within the Computer Science department.

Coordinators may find this system to be a convenient way to mass-produce records of all of their managed students for further processing such as, printing transcripts, amending records after important thesis or project dates, or tracking progress of all students concurrently enrolled in a graduate program.

1.3 Definitions, Acronyms and Abbreviations

Back-end: A functional framework which will interact with a GUI at a later date to be used by student and coordinator users of GRADS.

Breadth-Requirement: A mandated coursework pool for graduate students, spanning the topics of

Theory and Algorithms
Architecture, Systems and Software
Applications

Master's students must complete three courses; one from each subject area; while maintaining a 3.25 GPA to fulfill this requirement.
Ph.D students must complete five courses within the first three years of their program, with at least one course from each subject area; while maintaining a 3.45 GPA in these courses to fulfill this requirement.

Colloquium Requirement: One credit seminar requirement mandated by both M.S. and Ph.D students. Course candidates are outlined at: http://www.cs.umn.edu/research/colloquia.php

Grade Point Average (GPA): A calculated numerical floating point average of completed coursework in the Computer Science graduate department.

Graduate Program Coordinator (GPC): An external user, employed by the University of Minnesota, and exclusively assigned to a graduate student to ensure timely success of graduate requirements.

Graduate Requirement Assessment Database System (GRADS): An interactive database software system which grants exclusive permissions to students and program coordinators to compare and contrast student achievements (as logged on their student records) with specifically determined graduation requirements as dictated by their graduate path (Ph.D, M.S. Plan A, B or C) of choice.

New user: A student, recently admitted (< 6 mos. In any graduate program) with no active student record log of course performance. Transitions to student status after completion of one semester of coursework.

Student: A user, currently enrolled in a graduate (Ph.D or M.S.) program in Computer Science at the University of Minnesota, who has an active student record of locally completed (UMN coursework 5000 level and above) coursework explicitly offered by the CSci department. (For new admissions see 'new user')

Ph.D: A student user actively pursuing a Doctorate of Philosophy degree in Computer Science at the University of Minnesota.

Plan A (MS): A student user actively pursuing a Masters of Science degree in Computer Science at the University of Minnesota by completing a committee approved thesis to fulfil the 31-credit graduation requirement in addition to the Colloquium and Breadth-Requirements. Must pass further committee review as dictated by a final oral examination.

Plan B (MS): A student user actively pursuing a Masters of Science degree in Computer Science at the University of Minnesota by completing a research project in addition to the Colloquium and Breadth-Requirements to fulfil the 31-credit graduation requirement. Projects must be submitted for review at least one semester prior to the intended graduation date, and must be further assessed by a committee during an hour-long presentation.

Plan C (MS): A student user actively pursuing a Masters of Science degree in Computer Science at the University of Minnesota by completing coursework to fulfill the 31-credit requirement, while additionally meeting the Colloquium and Breadth-Requirements.

Record: An electronic log of academic performance maintained by a dynamic hash table of Student/Record key-value pairs. Retrieved by GRADS for further processing.

1.4 References and Acknowledgments

1. http://capstone.cs.ucsb.edu/cs189a/support/SRS-template.doc
2. Sommerville, Ian, and Perdita Stevens. *Introduction to Software Engineering*. Frenchs Forest, N.S.W.: Pearson, 2012.
3. https://ay14.moodle.umn.edu/pluginfile.php/679142/mod_resource/content/5/hw1-Requirements-Deliverable.pdf
4. http://www.cs.umn.edu/research/colloquia.php
5. http://www.cs.umn.edu/academics/graduate/Breadth-Courses-rev-8-13.pdf
6. http://www.cs.umn.edu/grad_handbook/grad_handbook.pdf

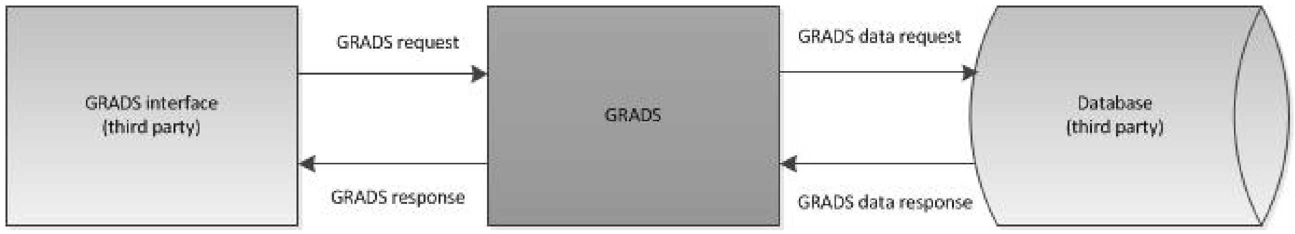
1.5 Intended Audience and Document Overview

This requirements document is intended for future user experience developers of GRADS in the hope that they will gain a more intimate understanding of the back-end functionality with which their interface wrapper will interact with. In this document we will provide an overview of the both the system and its user base. Use cases will be provided to better illustrate the functionality of GRADS and how users may interact with the record database.

2 Overall Description

2.1 System Perspective

GRADS is the processing back-end that will support the functionality described in Section 2.2. GRADS will replace the current manual process where a GPC must take a student's transcript and a paper-based degree program form and verify that all requirements have been met, as well as adding additional functionality that will allow students to look up information on their degree progress. The system will utilize an interface created by a third party that will facilitate the user interaction aspect of the system functionality. GRADS will utilize database tables created and maintained by a third party as well.



2.2 System Functionality

GRADS will support the following functionality for all users:

- Verify user exists and determine user type (student or GPC)

GRADS will support the following functionality for student users:

- Allow student user to view a summary of his or her degree progress
- Allow student user to view his or her GPA
    - This view will include an in-program GPA, a breadth requirement GPA, and an overall GPA, as well as the GPA levels that are required for each type of GPA
- Allow student to view the impact of the addition of a course on his or her progress in the degree that he or she is pursuing
- Allow student to amend their record

GRADS will support the following functionality for GPC users:

- Allow GPC user to view a summary of any student's degree progress
- Allow GPC user to view any student's GPA
    - This view will include an in-program GPA, a breadth requirement GPA, and an overall GPA, as well as the GPA levels that are required for each type of GPA
- Provide a list of all students and their respective ID numbers enrolled in a program selected by a GPC user
- Allow GPC user to view a transcript for any student in the program
- Allow GPC user to amend student records

2.3 Users and Characteristics

There are two different types of users that will use GRADS:

1. **Students**: these users are students currently enrolled in a graduate (Ph.D or M.S.) program in Computer Science at the University of Minnesota

   1.1. Student users will have the lowest level of access of any users of the system, as they will only be able to access their own records

2. **Graduate Program Coordinators (GPCs)**: these are external users that are employed by the University of Minnesota and exclusively assigned to a graduate student to ensure timely success of graduate requirements
   2.1. GPCs will have a higher level of privilege than student users, as they will have access to all students' records

2.4 Design and Implementation Constraints

- The functionality of GRADS must adhere to the graduate program requirements that can be found at http://www.cs.umn.edu/academics/graduate/index.php

2.5 Assumptions and Dependencies

This document has been created with the following assumptions in mind:

- GRADS shall utilize a user interface developed by a third party. It is assumed that this interface will send user requests to GRADS, and will interpret and present to the user the outputs produced by GRADS.
- GRADS shall utilize a database created and maintained by a third party. It is assumed that this database will contain student records and user data required by GRADS, and will be kept updated by the third party that owns the database.
- It is assumed that GRADS shall at all times operate as if a single user is using the application. Management of multiple sessions will not be considered in the design of this system.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Interfaces shall be a browser window. Application shall also be able to open popup windows. Input shall be done through the keyboard and mouse.

3.1.2 Hardware Interfaces

The database shall be stored on a central server and accessible remotely. The application shall run on a PC.

3.1.3 Software Interfaces

The application shall run in common web browsers including but not limited to Internet Explorer, Firefox, Chrome, and Safari, and it shall be compatible with Windows, OSX, and Linux. The database shall be accessed via SQL.

3.1.4 Communications Interfaces

Application shall run over an https connection and all transactions between the user and the server shall be encrypted.

3.2 Requirements

Requirement #: 1 **Use Case:** All

Requirement Description: A user's status and access privileges shall be determined based on the login credentials provided. After validating the username, GRADS shall retrieve the user's status (i.e. student or GPC) from the accounts database. If GRADS is unable to determine the user's type, it shall by default set the user type to Student.

Inputs: Login name

Outputs: None

Persistent changes: If username is valid, GRADS shall set current user accordingly, and shall correctly set the user status to either Student or GPC.

Test Cases: 1

Requirement #: 2 **Use Case:** 6

Requirement Description: GRADS shall generate a list of all students for a given semester containing student ID, name, address, phone number, class standing, and email address. GRADS shall only generate this list if the current user type is GPC. If the current user type is Student, GRADS shall return an error.

Inputs: Semester

Outputs: Database entry for each student enrolled during current semester

Persistent changes: None

Test Cases: 2

Requirement #: 3 **Use Case:** 7

Requirement Description: GRADS shall retrieve student transcript data. The data shall inclued the student's name, student ID, term started, department name, degree program, advisors' names, committee members' names, courses taken, milestones completed, and notes. If the current user is a student and the given student ID does not match the logged-in user, GRADS shall return transcript data for the logged-in user. If a null or otherwise invalid student ID is given, GRADS shall return an error. If a particular item of transcript data cannot be found, GRADS shall simply return a null value in its place.

Inputs: Student ID

Outputs: Student's advisor name, committee members' names, degree program name, and date that degree program was started.

Persistent changes: None

Test Cases: 3

Requirement #: 4 **Use Case:** 5

Requirement Description: The Check a Course's Impact function shall load all of the data required to generate a Student Progress Report (see Requirement #12). Then it shall add one selected additional course to the list of courses. Finally, it shall generate a Student Progress Report in accordance with Requirement #12, with the exception that the progress report shall now reflect the new course. The new course shall not be written to the database permanently, but shall only be fed to the Progress Report function for the purpose of creating the status report. GRADS shall return an error if any of the following occurs:

*An unrecognized course number is given
*A null value is given as either parameter
*The user is a GPC
*The user is a student and the given student ID does not correspond to the current user

Inputs: Selected Course Number, Student ID

Outputs: Same as Requirement #12, except given course shall be added to the retrieved course history for the student before the Progress Report is generated

Persistent changes: None

Test Cases: 4

Requirement #: 5 **Use Case:** 8

Requirement Description: GRADS shall allow a GPC to modify a student's record. The fields which can be modified are name, degree pursuing, department, committee members, courses, grades, and milestones. If the current user is not a GPC, the action shall be disallowed and GRADS shall simply return an error.

Inputs: Student ID, Field to modify, New value

Outputs: Error / No Error

Persistent changes: If modification of student record is permitted, the change shall be permanently written to the database.

Test Cases: 5

Requirement #: 6 **Use Case:** 3, 4

Requirement Description: GRADS take as input a student number and shall calculate that respective student's GPA, including in-program GPA, breadth requirement GPA, and overall GPA, as well as the GPA levels that are required for each type of GPA following the procedures as stated in Requirements 6a-6f.

- If the input student number is not valid, Calculate GPA will return an error message.
- If the input student number is valid, go on to calculate all GPA types as discussed in Requirements 6a-6f.

Inputs: Student Number

Outputs: In-program GPA, Breadth requirement GPA, Overall GPA, Required in-program GPA, Required breadth requirement GPA, Required overall GPA

Persistent changes: None

Test Cases: 6

Requirement #: 6a **Use Case:** 3,4

Requirement Description: To calculate in-program GPA, GRADS shall:

1. Retrieve grade information for all **CSCI** classes the selected student has taken.
   1.1. If the student has not taken any classes, return 0 as the GPA.
2. For each class, convert the letter grade to the number equivalent as listed in Table A.1.
3. Multiply the value calculated in step 2 by the number of credits the respective class is worth.
4. Repeat steps 2-3 until values for all classes have been calculated.
5. Add up the total number of grade points calculated in steps 2-4 and divide by the total number of credits taken. Return this value.

Inputs: Student Number

Outputs: In-Program GPA

Persistent changes: None

Test Cases: 6

Requirement #: 6b **Use Case:** 3,4

Requirement Description: To calculate breadth requirement GPA, GRADS shall:

1. Retrieve grade information for all CSCI classes the selected student has taken that fulfill the breadth requirement (see table A.2 for these courses).
   1.1. If the student has not taken any classes, return 0 as the GPA.
2. For each class, convert the letter grade to the number equivalent as listed in Table A.1.
3. Multiply the value calculated in step 2 by the number of credits the respective class is worth.

4. Repeat steps 2-3 until values for all classes have been calculated.
5. Add up the total number of grade points calculated in steps 2-4 and divide by the total number of credits taken. Return this value.

Inputs: Student Number

Outputs: Breadth Requirement GPA

Persistent changes: None

Test Cases: 6

Requirement #: 6c **Use Case:** 3,4

Requirement Description: To calculate overall GPA, GRADS shall:

1. Retrieve grade information for **all** classes the selected student has taken.
   1.1. If the student has not taken any classes, return 0 as the GPA.
2. For each class, convert the letter grade to the number equivalent as listed in Table A.1.
3. Multiply the value calculated in step 2 by the number of credits the respective class is worth.
4. Repeat steps 2-3 until values for all classes have been calculated.
5. Add up the total number of grade points calculated in steps 2-4 and divide by the total number of credits taken. Return this value.

Inputs: Student Number

Outputs: Overall GPA

Persistent changes: None

Test Cases: 6

Requirement #: 6d **Use Case:** 3,4

Requirement Description: GRADS will first determine the student type for the input student, and will return the appropriate required in-program GPA level as follows:

- If no student type is found or the student is not found in the database, return an error message.
- If the student is a PhD student, return 3.45
- If the student is an MS student, return 3.25

Inputs: Student Number

Outputs: Required In-Program GPA Level

Persistent changes: None

Test Cases: 6

Requirement #: 6e **Use Case:** 3,4

Requirement Description: GRADS will first determine the student type for the input student, and will return the appropriate required breadth requirement GPA level as follows:

- If no student type is found or the student is not found in the database, return an error message.
- If the student is a PhD student, return 3.45
- If the student is an MS student, return 3.25

Inputs: Student Number

Outputs: Required Breadth Requirement GPA Level

Persistent changes: None

Test Cases: 6

Requirement #: 6f **Use Case:** 3,4

Requirement Description: GRADS will first determine the student type for the input student, and will return the appropriate required overall GPA level as follows:

- If no student type is found or the student is not found in the database, return an error message.
- If the student is a PhD student, return 3.45
- If the student is an MS student, return 3.25

Inputs: Student Number

Outputs: Required Overall Requirement GPA Level

Persistent changes: None

Test Cases: 6

Requirement #: 7a **Use Case:** 1,2,7

Requirement Description: GRADS shall determine a PhD student's degree progress by showing whether they finished every milestone in PhD's degree requirement. Specifically speaking about courses,

- The student should have at least 31 course credits, of which 16 must be Computer Science program courses including 5 breadth courses and one credit of Colloquium (CSci 8970).
- In the 31 credits, there must be at least 6 credits from a supporting field. Whether a course can count into or not is decided by GPC.
- Courses used to purse a doctoral degree should have an overall GPA of at least 3.45 (Requirement 6c).
- Breadth courses should have a GPA of at least 3.45 (Requirement 6b).
- All courses should be 5000 level or above.
- All courses for the doctoral degree should be above C-.
- All Computer Science courses must be taken in A-F.
- The total credits of S-N courses should be less than ⅓ of total credits used for the doctoral degree.

Specifically speaking about researches,

- The Written Report should be finished.
- The Oral Prelim should be finished.
- Thesis proposal should be finished.
- Final Oral should be finished.

Inputs: Student ID

Outputs: Student's PhD degree progress, which states which has achieved and which hasn't.

Persistent changes: None

Test Cases: 7A, 7B

Requirement #: 7b **Use Case:** 1,2,7

Requirement Description: GRADS shall determine a Plan A MS student's degree progress by showing whether they finished every milestone in Plan A's degree requirement. Specifically speaking about courses,

- The student should have at least 22 course credits and 10 thesis credits.
- For all credits, at least 16 of them must be Computer Science Program courses including 3 breadth courses and one credit of the CS Colloquium (CSci 8970).
- All credits must be 5000 level or above.
- At least 3 of the total credits must be a regular 8000 level CS course.
- Courses used to purse a Plan A MS should have an overall GPA of at least 3.25 (Requirement 6c).
- Breadth courses should have a GPA of at least 3.25 (Requirement 6b).
- All courses should be above C-.
- All Computer Science courses must be taken in A-F.
- The total credits of S-N courses should be less than ⅓ of total credits used for the doctoral degree.

Specifically speaking about researches,

- The Thesis should be approved.

Inputs: Student ID

Outputs: Student's Plan A MS degree progress, which states which has achieved and which hasn't.

Persistent changes: None

Test Cases: 7C, 7D

Requirement #: 7c **Use Case:** 1,2,7

Requirement Description: GRADS shall determine a Plan B MS student's degree progress by showing whether they finished every milestone in Plan B's degree requirement. Specifically speaking about courses,

- The student should have at least 31 course credits, of which 16 must be Computer Science program courses including 5 breadth courses and one credit of Colloquium (CSci 8970).
- All credits must be 5000 level or above.
- At least 3 of the total credits must be a regular 8000 level CS course.
- Course CSci 8760 Plan B project is required, in addition to the required 3 credits of an 8000 level course.
- Courses used to purse a Plan B MS should have an overall GPA of at least 3.25 (Requirement 6c).
- Breadth courses should have a GPA of at least 3.25 (Requirement 6b).
- All courses should be above C-.
- All Computer Science courses must be taken in A-F.
- The total credits of S-N courses should be less than ⅓ of total credits used for the doctoral degree.

Specifically speaking about researches,

- The Thesis should be approved.
- The Oral Final should be finished.

Inputs: Student ID

Outputs: Student's Plan B MS degree progress, which states which has achieved and which hasn't.

Persistent changes: None

Test Cases: 7E, 7F

Requirement #: 7d **Use Case:** 1,2,7

Requirement Description: GRADS shall determine a Plan C MS student's degree progress by showing whether they finished every milestone in Plan C's degree requirement. Specifically speaking about courses,

- The student should have at least 31 course credits, of which 16 must be Computer Science program courses including 5 breadth courses and one credit of Colloquium (CSci 8970).
- All credits must be 5000 level or above.
- Two regular 8000 level CS courses should be finished.
- Course CSci 8760 Plan B project is required, in addition to the required 3 credits of an 8000 level course.
- Courses used to purse a Plan B MS should have an overall GPA of at least 3.25 (Requirement 6c).
- Breadth courses should have a GPA of at least 3.25 (Requirement 6b).
- All courses should be above C-.
- All Computer Science courses must be taken in A-F.
- The total credits of S-N courses should be less than ⅓ of total credits used for the doctoral degree.

Inputs: Student ID

Outputs: Student's Plan C MS degree progress, which states which has achieved and which hasn't.

Persistent changes: None

Test Cases: 7G, 7H

3.3 Test Cases

Test Case 1: Test that GRADS correctly identifies user from username

1A:

Description: Verify that GRADS correctly logs in student and GPC users
Inputs: Username
Expected Result: GRADS sets current user to match username, and correctly sets user status
Dependencies: None
Setup: None
Test Procedure: Feed a student username to login function. Verify that GRADS sets current user to correspond to login credentials. Verify that current user type is set to student. Repeat same procedure for a GPC login.

1B:

Description: Verify that GRADS gracefully handles erroneous input
Inputs: Username
Expected Result: GRADS shall return an error and reject the login attempt
Dependencies: None
Setup: None
Test Procedure: Attempt to access login function with a username which does not exist in the database. Verify that an error is returned.

1C:

Description: Verify that GRADS gracefully handles missing user type
Inputs: Username
Expected Result: GRADS shall log user on as a student
Dependencies: None

Setup: Use the GPC account from Test Case 1B, but set the account type to null
Test Procedure: Attempt login using the given account and verify that the user type is set to Student.

Test Case 2: Generating list of all students
2A:
Description: Test that GRADS correctly returns list of students enrolled during current semester
Inputs: Current semester
Expected Result: Any randomly-selected student enrolled during current semester will appear in the GRADS list of enrolled students
Dependencies: Must have access to database of student entries
Setup: None
Test Procedure: Select twenty random students from current semester. Use GRADS to create the list of students for the current semester. Verify that each selected student is in the list.
2B:
Description: Test that GRADS does not include incorrect entries in student list
Inputs: Semester
Expected Result: Selected students do not appear on the student list
Dependencies: Must have entries for students in database
Setup: None
Test Procedure: Select a random semester and generate the student list. Select 100 random students who are/were not enrolled during the given semester. Verify that these students are not on the list.
2C:
Description: Ensure that student users cannot access full list of students
Inputs: None
Expected Result: GRADS returns an error
Dependencies: None
Setup: Must be logged in as a student
Test Procedure: Call function which returns list of all students. GRADS should return an error.

Test Case 3: Show Transcript
3A:
Description: Test that GRADS correctly gathers and returns transcript data
Inputs: Student ID number
Expected Result: GRADS shall return transcript data as enumerated in requirement #3
Dependencies: None
Setup: None
Test Procedure: Request transcript with given student ID and verify that correct information is returned
3B:
Description: Test that GRADS correctly handles bad input
Inputs: Student ID number
Expected Result: GRADS shall return an error
Dependencies: None
Setup: None
Test Procedure: Request transcript with nonexistent student ID and verify that an error is returned
3C:
Description: Test that GRADS will only return a student's own transcript
Inputs: Student ID number
Expected Result: GRADS shall return transcript for logged-in user
Dependencies: None
Setup: Must be logged in as a student
. **Test Procedure:** Request transcript using student ID which differs from that of current user. Verify that returned data corresponds to logged-in user

Test Case 4: Check a Course's Impact

4A:

Description: Test that GRADS correctly checks a course's impact
Inputs: Student ID number, Course Number
Expected Result: Both generated reports shall match
Dependencies: Must have extra student record in database with one course removed
Setup: Duplicate a student record in its entirety, but remove one in-progress course
Test Procedure: Run Student Progress Report on full record. Then run Check A Course's impact on the record with the course removed, and request to check the impact of the removed course. Make sure that the two reports are identical.

4B:

Description: Test that GRADS correctly returns an error for a null parameter value
Inputs: Student ID and/or Course Number and/or null value
Expected Result: GRADS shall return an error
Dependencies: None
Setup: None
Test Procedure: Attempt to check a course's impact using a null value for the course with a valid student ID. also try it with a valid course number and a null student ID. Verify that GRADS returns an error in both cases.

4C:

Description: Test that GRADS correctly generates an error for an unrecognized course
Inputs: Student ID number, Course Number
Expected Result: GRADS shall return an error
Dependencies: None
Setup: None
Test Procedure: Attempt to check a course's impact using a nonexistent course number. Verify that GRADS returns an error code.

4D:

Description: Test that GRADS declines this feature for a GPC
Inputs: Student ID number, Course Number
Expected Result: GRADS shall return an error
Dependencies: None
Setup: User must be logged in as a GPC
Test Procedure: Attempt to check a course's impact using any student ID and error function. GRADS should return an error.

Test Case 5: Modify Student Record

5A:

Description: Test that GRADS allows a GPC to modify a student record
Inputs: Student ID number, Field name, New Value
Expected Result: GRADS shall update the field to the new value
Dependencies: None
Setup: User must be logged in as a GPC
Test Procedure: Choose a student record and any field in that record. Feed GRADS this student's ID number, the name of the field, and a new value which does not match the current value. Then perform a query to ensure that GRADS has correctly updated the database.

5B:

Description: Test that GRADS will not allow a student to use this function
Inputs: Same as case 5A
Expected Result: GRADS shall return an error
Dependencies: None

Setup: User must be logged in as a Student
Test Procedure: Attempt to access this function as in Test Case 5A. GRADS should return an error.

5C:

Description: Test that GRADS gracefully handles bad input
Inputs: Student ID number, Field Name, New Value
Expected Result: GRADS shall return an error
Dependencies: None
Setup: User must be logged in as a GPC
Test Procedure: Attempt to access this function using a nonexistent student ID, a null student ID, a nonexistent Field Name, a null Field Name, and a null New Value. Verify that GRADS returns an error in each case.

Test Case 6: GPA

6A:

Description: Test that GRADS correctly calculates GPA information for an MS student with one or more courses on their record
Inputs: Student ID number
Expected Result: GRADS shall return the following data for a student user enumerated in Requirement #6:

- In-program GPA
- Breadth requirement GPA
- Overall GPA
- Required in-program GPA - expect 3.25
- Required breadth requirement GPA - expect 3.25
- Required overall GPA - expect 3.25

Dependencies: Must have student data in database.
Setup: Choose one test MS student user with valid course data to use as input.
Test Procedure: Feed chosen student ID to the Calculate GPA function. Verify that each of the GPA types has been correctly calculated and returned in the response.

6B:

Description: Test that GRADS correctly returns an error message when the Calculate GPA function is called with an invalid student number
Inputs: Invalid Student ID number
Expected Result: An error message stating the input student ID number is invalid
Dependencies: Must **not** have student data in database for test student ID number.
Setup: Choose one test student ID number that is not valid and verify that it does not exist in the database.
Test Procedure: Feed chosen student ID to the Calculate GPA function. Verify that an error message has been returned in the response.

6C:

Description: Test that GRADS correctly returns GPA values for the three types of GPA, and error messages for the GPA levels when Calculate GPA is called for a student whose student data does not indicate a degree type
Inputs: Valid Student ID number of a student whose student data does not indicate a degree type
Expected Result: GRADS shall return the following data for a student user enumerated in Requirement #6:

- In-program GPA
- Breadth requirement GPA
- Overall GPA
- Required in-program GPA - expect error messsage
- Required breadth requirement GPA - expect error message
- Required overall GPA - expect error message

Dependencies: Must have student data in database for test student ID number.
Setup: Choose one test student ID number that is valid and verify that the selected student's data does not include a degree type
Test Procedure: Feed chosen student ID to the Calculate GPA function. Verify that the values returned correctly reflect the expected result.

Test Case 7: Show Degree Progress

7A:

Description: Test that GRADS correctly determines an unfinished Ph.D degree plan.
Inputs: Student ID number
Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7a, with respect to this student's degree plan. The results should be that this student hasn't finished all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record B.1
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

7B:

Description: Test that GRADS correctly determines a finished Ph.D degree plan.
Inputs: Student ID number
Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7a, with respect to this student's degree plan. The results should be that this student succeeded in finishing all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record B.2
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

7C:

Description: Test that GRADS correctly determines an unfinished M.S. path A degree plan.
Inputs: Student ID number
Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7b, with respect to this student's degree plan. The results should be that this student hasn't finishied all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record A.2
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

7D:

Description: Test that GRADS correctly determines a finished M.S. path A degree plan.
Inputs: Student ID number
Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7b, with respect to this student's degree plan. The results should be that this student succeeded in finishing all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record A.2
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

7E:

Description: Test that GRADS correctly determines an unfinished M.S. path B degree plan.
Inputs: Student ID number
Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7c, with respect to this student's degree plan. The results should be that this student hasn't finishied all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record A.3
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

7F:

Description: Test that GRADS correctly determines a finished M.S.path B degree plan.
Inputs: Student ID number

Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7c, with respect to this student's degree plan. The results should be that this student succeeded in finishing all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record A.3
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

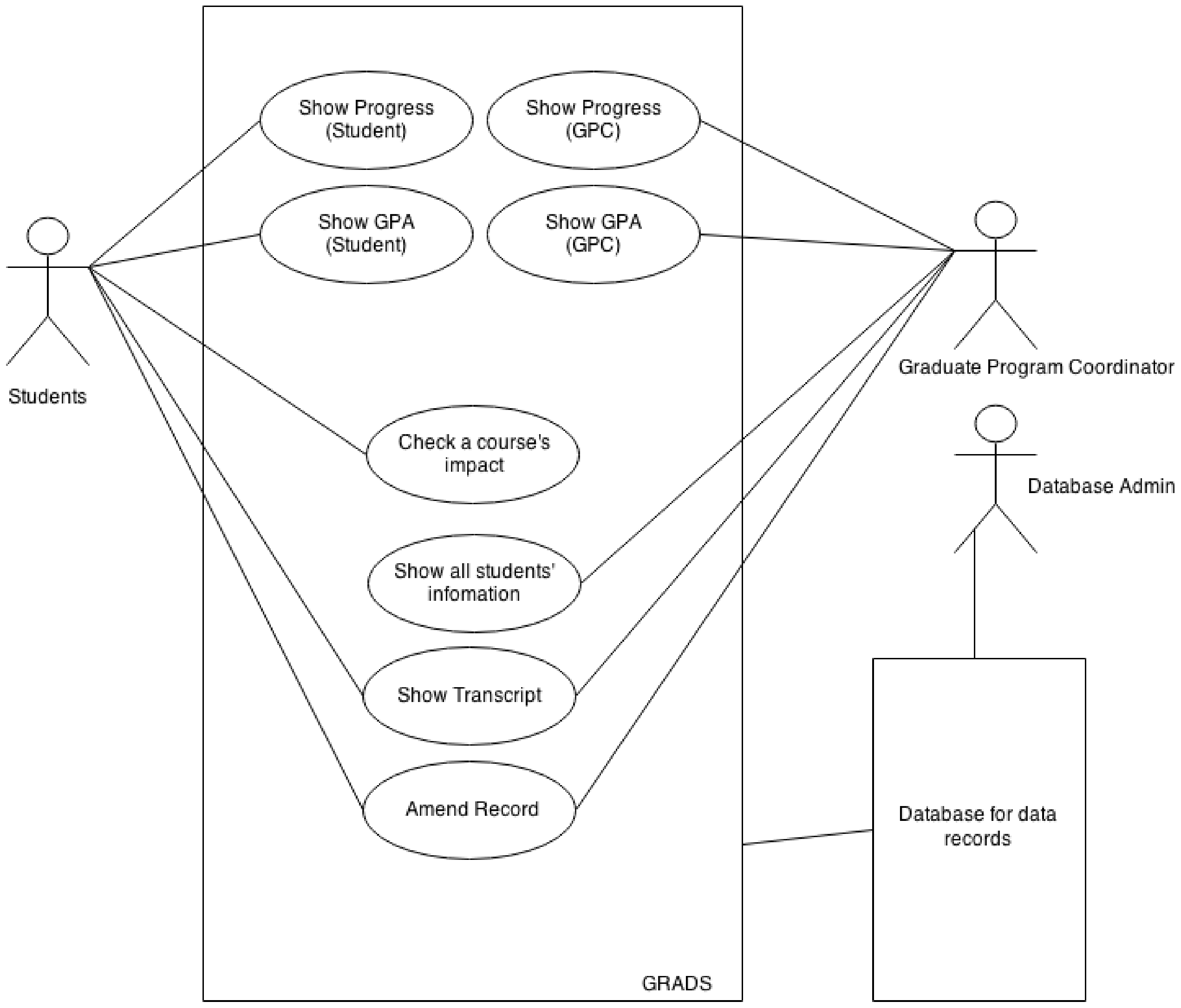
7G:

Description: Test that GRADS correctly determines an unfinished M.S. path C degree plan.
Inputs: Student ID number
Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7d, with respect to this student's degree plan. The results should be that this student hasn't finished all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record A.4
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

7H:

Description: Test that GRADS correctly determines a finished path C degree plan.
Inputs: Student ID number
Expected Result: GRADS shall examine this student's record, and determine whether he satisfies every requirement specified in Requirement #7d, with respect to this student's degree plan. The results should be that this student succeeded in finishing all degree requirements.
Dependencies: Must have up-to-date student data in database
Setup: Appendix sample Record A.4
Test Procedure: Request degree progress with given student ID and verify that correct information is returned

3.4 Use Case View



Use Case #: 1
Name: Show Progress (Student)

Summary:

The user (student) requests to track his degree progress.

Basic Course of Events:

1. The user requests to show his degree progress.
2. The system sends out this student's progress towards his degree, indicating completed milestones and uncompleted milestones.

Alternative Paths:

None

Exception Paths:

None

Assumptions:

None

Precondition:

The user has finished the use case *Log In*.

Postcondition:

None

Use Case #: 2
Name: Show Progress (GPC)

Summary:
The user (GPC) requests to track a certain student's degree progress.
Basic Course of Events:
1. The system makes sure that the current user is of role GPC.
2. The system asks for student ID.
3. The user sends the ID of the student who he wants to track.
4. The system sends out this student's progress towards his degree, indicating completed milestones and uncompleted milestones.
Alternative Paths:
None
Exception Paths:
In step 3, if the user sends an ID that does not exist, go back to step 2 and prompt again.
Assumptions:
None
Precondition:
The user has finished the use case *Log In*.
Postcondition:
None

Use Case #: 3
Name: Show GPA (Student)

Summary:
The user (student) requests to show his GPA.
Basic Course of Events:
1. The user requests to show his GPA.
2. The system sends out this student's GPA, including in-program GPA, breadth requirement GPA, and overall GPA as well as the GPA levels that are required for each.
Alternative Paths:
None
Exception Paths:
None
Assumptions:
None
Precondition:
The user has finished the use case *Log In*.
Postcondition:
None

Use Case #: 4
Name: Show GPA (GPC)

Summary:
The user (GPC) requests to show a certain student's GPA.
Basic Course of Events:
1. The system makes sure that the current user is of role GPC.
2. The system asks for student ID.
3. The user sends the ID of the student who he wants to track.
4. The system sends out this student's GPA, including in-program GPA, breadth requirement GPA, and overall GPA as well as the GPA levels that are required for each.
Alternative Paths:
None
Exception Paths:
In step 3, if the user sends an ID that does not exist, go back to step 2 and prompt again.
Assumptions:
None
Precondition:
The user has finished the use case *Log In*.
Postcondition:
None

Use Case #: 5
Name: Check A Course's Impact

Summary:

The user (student) requests to show the impact if a certain course is counted into his current degree pursuing progress.

Basic Course of Events:

1. The system asks for the course that the student wants to add.
2. The user types the course he wants to add.
3. The system shows the new degree progress, with the designated course counted into.

Alternative Paths:

None

Exception Paths:

If at step 3, the student inputs a course that does not exist, report error and go back to step 2.

Assumptions:

None

Precondition:

The user has finished the use case *Log In*.

Postcondition:

None

Use Case #: 6
Name: Show All Students' Information

Summary:

The user (GPC) requests to show all students in the program and corresponding ID numbers.

Basic Course of Events:

1. The system makes sure that the current user is of role GPC.
2. The system returns information about all students' names and IDs in the program.

Alternative Paths:

None

Exception Paths:

If step 1 fails (i.e., the user is not a GPC), report error and exit.

Assumptions:

None

Precondition:

The user has finished the use case *Log In*.

Postcondition:

None

Use Case #: 7
Name: Show Transcript

Summary:

The user (GPC) requests to show a given student's transcript.

Basic Course of Events:

1. The system makes sure that the current user is of role GPC.
2. The system asks for student ID.
3. The user sends the ID of the student who he wants to track.
4. The system returns this student's complete transcript.

Alternative Paths:

If step 1 fails (i.e., the user is a student), the system will directly go to step 4 and show this student's own complete transcript.

Exception Paths:

If at step 3, if the user typed an ID that does not exist, go back to step 2 and prompt again.

Assumptions:

None

Precondition:

The user has finished the use case *Log In*.

Postcondition:

None

Use Case #: 8
Name: Amend Record

Summary:

The user (GPC) requests to amend record for a given student.

Basic Course of Events:

1. The system makes sure that the current user is of role GPC.

2. The system asks for student ID.
3. The user sends the ID of the student who he wants to modify on.
4. The system returns the GPC to modify one of the items for the given student. Specifically speaking, GPC may modify the following records of a given student:

Name, Degree Pursuing, Department, Committee Members, Courses, Grades, Milestones in graduate careers, etc.

Alternative Paths:

If step 1 fails (i.e., the user is a student), the student can only modify his own records. Specifically speaking, he may modify the following records of his own:

Name, Degree Pursuing, Department, Committee Members, etc.

Exception Paths:

If at step 3, if the user sends an ID that does not exist, go back to step 2 and prompt again.

Assumptions:

None

Precondition:

The user has finished the use case *Log In*.

Postcondition:

The corresponding record of this student is modified.

Appendix A : Data Dictionary

Letter Grade	Number Equivalent
A	4.0
A-	3.667
B+	3.333
B	3.0
B-	2.667
C+	2.333
C	2.0
C-	1.667
D+	1.333
D	1.0
F	0.0

Table A.1 : GPA Translation Values

Theory and Algorithms 5302: Analysis of Numerical Algorithms 5304: Computational Aspects of Matrix Theory 5403: Computational Complexity 5421: Advanced Algorithms & Data Structures 5481: Computational Techniques for Genomics 5525: Machine Learning
Architecture, Systems, and Software 5103: Operating Systems 5104: System Modeling and Performance Evaluation

5105: Introduction to Distributed Systems 5106: Programming Languages 5161: Introduction to Compilers 5204: Advanced Computer Architecture 5211: Data Communications and Computer Networks 5221: Foundations of Advanced Networking 5231: Wireless and Sensor Networks 5451: Introduction to Parallel Computing: Architectures, Algorithms, and Programming 5708: Architecture and Implementation of Database Management Systems 5801: Software Engineering I 5802: Software Engineering II
Applications 5115: User Interface Design, Implementation and Evaluation 5125: Collaborative and Social Computing 5271: Introduction to Computer Security 5461: Functional Genomics, Systems Biology, and Bioinformatics 5471: Modern Cryptography 5511: Artificial Intelligence I 5512: Artificial Intelligence II 5521: Introduction to Machine Learning 5523: Introduction to Data Mining 5551: Introduction to Intelligent Robotic Systems 5561: Computer Vision 5607: Fundamentals of Computer Graphics I 5608: Fundamentals of Computer Graphics II 5609: Visualization 5611: Motion and Planning in Games 5619: Virtual Reality and 3D User Interaction 5707: Principles of Database Systems

Table A.2: Breadth Areas and Their Associated Courses

Requirement	Test Case	Use Case
1	1	All

2	2	6
3	3	7
4	4	5
5	5	8
6	6	3,4
6a	6	3,4
6b	6	3,4
6c	6	3,4
6d	6	3,4
6e	6	3,4
6f	6	3,4
7a	7	1,2,7
7b	7	1,2,7
7c	7	1,2,7
7d	7	1,2,7

Table A.3 : Requirements Traceability Matrix

Record A.2: Sample completed path A M.S. plan

```
[

        Courses Taken: [


                Course: [ Name:Analysis of Numerical Algorithms
                          ID: 5302
                          Credits:3
                          Semester: Fall
                          Year:2012

                          Grade Received: A-
                         ]

                Course: [ Name: Operating Systems
                          ID: 5103
                          Credits: 3
                          Semester: Fall
                          Year:2012

                          Grade Received: A-
                         ]

                Course: [ Name: Modern Cryptography
                          ID: 5471
                          Credits:3
                          Semester: Fall
                          Year:2012

                          Grade Received: A-
                         ]

                Course: [ Name: Matrix Theory
                          ID: 5304
                          Credits:3
                          Semester: Fall
                          Year:2012

                          Grade Received: A-
                         ]

                Course: [ Name: Compilers
                          ID: 5161
                          Credits:3
                          Semester: Spring
                          Year:2013

                          Grade Received: A-
                         ]

                Course:  [ Name:Parallel Computing
                          ID: 5451
                          Credits:3
                          Semester: Spring
                          Year:2013

                          Grade Received: A-
                         ]

                Course: [ Name:Data Mining
                          ID: 5523
                          Credits: 3
                          Semester: Spring
                          Year: 2013

                          Grade Received: A-
                         ]

                Course: [ Name: Machine Learning
                          ID: 5521
                          Credits:3
                          Semester: Spring
                          Year: 2013

                          Grade Received: A-
                         ]

                Course: [ Name: Software Engineering I
                          ID: 5801
                          Credits:3
                          Semester: Fall
                          Year: 2013

                          Grade Received: A-
                         ]

                Course: [ Name: Storage Capacity of Repairable Networks
                          ID: Coll
                          Credits:1
                          Semester: Fall
                          Year: 2013

                          Grade Received: A-
                         ]

        Milestone Completed: [
                         Thesis: Linux is the one true OS
                         Oral Review: Pass
                         Complete Date: Fall 2013
        ]


]
```

Record A.3 Sample completed path B M.S. plan

```
[

        Courses Taken: [


                Course: [ Name: Computational Aspects of Matrix Theory
                          ID: 5304
                          Credits:3
                          Semester: Fall
                          Year: 2012

                          Grade Received: A-
                         ]

                Course: [ Name:  System Modeling and Performance Evaluation
                          ID: 5104
                          Credits:3
                          Semester: Fall
                          Year: 2012

                          Grade Received: A-
                         ]

                Course: [ Name: Introduction to Computer Security
                          ID: 5271
                          Credits:3
                          Semester: Fall
                          Year:2012

                          Grade Received: A-
                         ]

                Course: [ Name: Graphics I
                          ID: 5106
                          Credits:3
                          Semester: Fall
                          Year: 2012

                          Grade Received: A-
                         ]

                Course: [ Name: Software Engineering I
                          ID: 5801
                          Credits:3
                          Semester: Spring
                          Year:2013

                          Grade Received: A-
                         ]

                Course:  [ Name:Computer Languages
                          ID: 5106
                          Credits:3
                          Semester: Spring
```

```
                              Year: 2013

                              Grade Received: A-
                             ]

                    Course: [ Name: Operating Systems
                              ID: 5103
                              Credits:3
                              Semester: Spring
                              Year: 2013

                              Grade Received: A-
                             ]

                    Course: [ Name: Data Mining
                              ID: 5523
                              Credits:3
                              Semester: Spring
                              Year: 2013

                              Grade Received: A-
                             ]

                    Course: [ Name:Analysis of Numerical Algorithms
                              ID: 5302
                              Credits:3
                              Semester: Fall
                              Year: 2013

                              Grade Received: A-
                             ]

                    Course: [ Name: Storage Capacity of Repairable Networks
                              ID: Coll
                              Credits:1
                              Semester: Fall
                              Year: 2013

                              Grade Received: A-
                             ]

          ]

          Milestone Completed: [
                             Project: Cloud Security Research
                             Review: Pass
                             Complete Date: Fall 2013
          ]

]
```

Record A.4: Sample completed path C M.S. plan

```
[

                    Course: [ Name: Machine Learning
                              ID: 5525
                              Credits:3
                              Semester: Fall
                              Year: 2012

                              Grade Received: A-
                             ]

                    Course: [ Name:Programming Languages
                              ID: 5106
                              Credits:3
                              Semester: Fall
                              Year:2012

                              Grade Received: A-
                             ]

                    Course: [ Name: Artificial Intelligence I
                              ID: 5511
                              Credits:3
                              Semester: Fall
                              Year: 2012

                              Grade Received:A-
                             ]

                    Course: [ Name: Graphics I
                              ID: 5106
                              Credits:3
                              Semester: Fall
                              Year: 2012

                              Grade Received: A-
                             ]

                    Course: [ Name:Artificial Intelligence II
                              ID: 5512
                              Credits:3
                              Semester: Spring
                              Year:2013

                              Grade Received: A-
                             ]

                    Course:  [ Name:Software Engineering I
                              ID: 5801
                              Credits:3
                              Semester: Spring
                              Year: 2013
```

```
                Grade Received: A-
               ]

      Course: [ Name: Graphics II
                ID: 5608
                Credits:3
                Semester: Spring
                Year: 2013

                Grade Received: A-
               ]

      Course: [ Name:Analysis of Numerical Algorithms
                ID: 5302
                Credits:3
                Semester: Spring
                Year: 2013

                Grade Received: A-
               ]

      Course: [ Name:Software Engineering II
                ID: 5802
                Credits:3
                Semester: Fall
                Year: 2013

                Grade Received:A-
               ]

      Course: [ Name: Data Mining
                ID: 5523
                Credits:3
                Semester: Fall
                Year: 2013

                Grade Received: A-
               ]

      Course: [  Name: Storage Capacity of Repairable Networks
                ID: Coll
                Credits:1
                Semester: Fall
                Year: 2013

                Grade Received: A-
               ]

]
```

Record B.1: Sample completed Ph.D. Plan

[

"Courses Taken": [

{"Course": {"Name":"Artificial Intelligence", "ID":"CSci5511" "numCredits":"3"}, "Term": {"Semester":"Fall" "Year":"2013" }, "Grade":"A-" },

{"Course": {"Name":"Special Advanced Topics in Computer Science", "ID":"CSci8980" "numCredits":"3"}, "Term": {"Semester":"Fall" "Year":"2013" }, "Grade":"B" },

{"Course": {"Name":"Introduction to Research in Computer Science I", "ID":"CSci8001" "numCredits":"1"}, "Term": {"Semester":"Fall" "Year":"2013" }, "Grade":"A" },

{"Course": {"Name":"Practie University Teaching for Non-Native Speakers", "ID":"Grad5105" "numCredits":"2"}, "Term": {"Semester":"Fall" "Year":"2013" }, "Grade":"S" },

{"Course": {"Name":"Introduction to Research in Computer Science II", "ID":"CSci8002" "numCredits":"2"}, "Term": {"Semester":"Spring" "Year":"2014" }, "Grade":"A" },

{"Course": {"Name":"Advanced Computer Networks", "ID":"CSci8211" "numCredits":"3", "Term": {"Semester":"Spring" "Year":"2014" }, "Grade":"B" },

{"Course": {"Name":"Foundations of Advanced Networking", "ID":"CSci5221" "numCredits":"3"}, "Term": {"Semester":"Spring" "Year":"2014" }, "Grade":"A" },

{"Course": {"Name":"Advanced Algorithms and Data Structures", "ID":"CSci5421" "numCredits":"3"}, "Term": {"Semester":"Spring" "Year":"2014" }, "Grade":"A" },

{"Course": {"Name":"Computer Science Colloquium", "ID":"CSci9870" "numCredits":"1"}, "Term": {"Semester":"Spring" "Year":"2014" }, "Grade":"S" },

{"Course": {"Name":"Introduction to Machine Learning", "ID":"CSci5521" "numCredits":"3"}, "Term": {"Semester":"Fall" "Year":"2014" }, "Grade":"A" },

{"Course": {"Name":"Software Engineering I", "ID":"CSci5801" "numCredits":"3"}, "Term": {"Semester":"Fall" "Year":"2014" }, "Grade":"A" },

{"Course": {"Name":"Graduate Research Writing for Non-Native Speakers", "ID":"WRIT5051" "numCredits":"3"}, "Term": {"Semester":"Fall" "Year":"2014" }, "Grade":"S" },

{"Course": {"Name":"Graduate Research Presentation for Non-Native Speakers", "ID":"WRIT5052" "numCredits":"3"}, "Term": {"Semester":"Spring" "Year":"2015" }, "Grade":"S" },

]

```
        "Milestone Set": [
                {"Milestone":"Courses Finished", "Term":{"Semester":"Spring", "Year":"2015"}},

                {"Milestone":"Oral Prelim Finished", "Term":{"Semester":"Fall", "Year":"2015"}},

                {"Milestone":"Thesis Proposal Finished", "Term":{"Semester":"Fall", "Year":"2016"}},

                {"Milestone":"Final Oral Finished", "Term":{"Semester":"Fall", "Year":"2018"}},
        ]


]
```

Record B.2: Sample incompleted Ph.D. Plan

```
[


        "Courses Taken": [

                {"Course": {"Name":"Artificial Intelligence", "ID":"CSci5511" "numCredits":"3"}, "Term":
{"Semester":"Fall" "Year":"2013" }, "Grade":"A-" },

                {"Course": {"Name":"Special Advanced Topics in Computer Science", "ID":"CSci8980"
"numCredits":"3"}, "Term": {"Semester":"Fall" "Year":"2013" }, "Grade":"B" },


                {"Course": {"Name":"Practie University Teaching for Non-Native Speakers", "ID":"Grad5105"
"numCredits":"2"}, "Term": {"Semester":"Fall" "Year":"2013" }, "Grade":"S" },



                {"Course": {"Name":"Introduction to Research in Computer Science II", "ID":"CSci8002"
"numCredits":"2"}, "Term": {"Semester":"Spring" "Year":"2014" }, "Grade":"A" },

                {"Course": {"Name":"Advanced Computer Networks", "ID":"CSci8211" "numCredits":"3", "Term":
{"Semester":"Spring" "Year":"2014" }, "Grade":"B" },

                {"Course": {"Name":"Computer Science Colloquium", "ID":"CSci9870" "numCredits":"1"}, "Term":
{"Semester":"Spring" "Year":"2014" }, "Grade":"S" },



                {"Course": {"Name":"Introduction to Machine Learning", "ID":"CSci5521" "numCredits":"3"}, "Term":
{"Semester":"Fall" "Year":"2014" }, "Grade":"A" },

                {"Course": {"Name":"Software Engineering I", "ID":"CSci5801" "numCredits":"3"}, "Term":
{"Semester":"Fall" "Year":"2014" }, "Grade":"A" },

                {"Course": {"Name":"Graduate Research Writing for Non-Native Speakers", "ID":"WRIT5051"
"numCredits":"3"}, "Term": {"Semester":"Fall" "Year":"2014" }, "Grade":"S" },
```

{"Course": {"Name":"Graduate Research Presentation for Non-Native Speakers", "ID":"WRIT5052" "numCredits":"3"}, "Term": {"Semester":"Spring" "Year":"2015" }, "Grade":"S" },

]

"Milestone Set": [

{"Milestone":"Oral Prelim Finished", "Term":{"Semester":"Fall", "Year":"2015"}},

]

]